GUIDA ALLA REALIZZAZIONE DI SITI RESPONSIVI CON UIKIT.CSS

SITI MOBILE E RESPONSIVI

In questa guida cercheremo di affrontare in modo semplice i principali aspetti relativi alla creazione di un sito per dispositivi mobile ponendo particolare attenzione a tutto ciò che riguarda il **responsive design** e a come poter operare in Passweb per realizzare e gestire un sito responsivo.

Considerata l'enorme diffusione dei dispositivi mobile e la tendenza sempre maggiore, come evidenziato dalle più recenti statistiche, a navigare in internet utilizzando questo tipo di dispositivi, la realizzazione di un sito mobile friendly non è più un'opzione o una scelta, come poteva essere in passato, ma è diventata una vera e propria necessità, anche e soprattutto in relazione alla linee guida dettate da Google stesso in materia di SEO e di indicizzazione.

Sotto questo punto di vista Passweb offre due diversi approcci alla realizzazione di un sito ottimizzato per dispositivi mobile lasciando quindi all'utente la possibilità di scegliere quale strada seguire per soddisfare questo tipo di esigenza:

- **Sito Mobile**: realizzare un sito mobile significa creare un sito parallelo esplicitamente pensato per i dispositivi mobile tendo conto di tutte le limitazioni intrinseche di questi dispositivi anche in relazione alla connessione che spesso utilizzano

  In Passweb questo può essere fatto utilizzando le **Varianti Mobile** e **la libreria di Componenti sviluppati ad hoc** per questa particolare tipologia di dispositivi.

- **Sito Responsivo**: realizzare un sito responsivo significa creare un unico sito con un unico layout in grado di adattarsi automaticamente alle diverse risoluzioni del dispositivo su cui viene visualizzato rendendolo quindi comodo da navigare indipendentemente dal fatto di utilizzare un pc, un tablet e/o uno smartphone.

  Un sito responsivo sfrutta le cosiddette **Media Query**, introdotte dalle specifiche CSS 3, grazie alle quali poter "addomesticare" il sito in modo che si adatti automaticamente al dispositivo in uso, cambiando le dimensioni e le caratteristiche degli elementi di ogni pagina fino a farli, se necessario, completamente sparire.

  In Passweb questo può essere realizzato sfruttando le funzionalità avanzate dei Layout che permettono l'importazione di apposite librerie CSS e/o la scrittura di codice CSS, e conseguentemente di Media Query personalizzate

Quale dei due approcci sia il migliore è una delle domanda che ha animato il mondo web con discussioni confronti e riflessioni senza però trovare una risposta definitiva.

Entrambi i modi di operare presentano infatti dei vantaggi e degli svantaggi piuttosto evidenti sia in termini generali che, nello specifico, per quel che riguarda la loro implementazione all'interno di Passweb.

SITO MOBILE – PRO

- **Semplicità di realizzazione**: considerando che in Passweb è disponibile una libreria di componenti sviluppati ad hoc per le Varianti Mobile, la realizzazione di siti mobile non richiede particolari conoscenze tecniche essendo la metodologia di lavoro esattamente la stessa di quella adottata per la realizzazione del sito in versione desktop
- **Esperienza utente:** i Componenti Mobile disponibili in Passweb hanno un' usabilità pensata appositamente per questo tipo di dispositivi e consentono quindi di ottenere, senza particolari accorgimenti, un' usabilità del sito e di un esperienza utente piuttosto elevata
- **Sito distinto dalla versione desktop ma che risponde allo stesso url:** il fatto di creare un sito parallelo, permette di inserire all'interno delle varie pagine elementi diversi rispetto a quelli presenti nella corrispondente versione desktop limitando quindi i contenuti a quelli strettamente necessari ottimizzando così le prestazioni del sito.

  Inoltre la Variante Mobile risponde sempre allo stesso url cui risponde la Variante Standard del sito. La visualizzazione della versione mobile piuttosto che della versione desktop è gestita, in Passweb, interamente lato server sulla base del dispositivo che richiede la pagina web e della risoluzione (in larghezza) di questo stesso dispositivo, evitando così problemi di indicizzazione dovuti alla suddivisione della reputazione di una stessa pagina web su due url differenti.

SITO MOBILE – CONTRO

- **Manutenzione:** avere due diversi siti, uno per la visualizzazione su desktop, ed uno per la visualizzazione su smartphone, significa raddoppiare i tempi di realizzazione del sito oltre, ovviamente agli sforzi di gestione dei contenuti, di web marketing, di SEO ecc…

SITO RESPONSIVO – PRO

- **Un unico sito web**: contenuti semplici da amministrare perché non duplicati su due siti distinti

- **Web Marketing e SEO più semplici**: con un solo sito web gli sforzi si concentrano anziché dividersi (per poi moltiplicarsi)
- **Stesso url**: come per i siti mobile realizzati con Passweb, a maggior ragione per i siti responsivi non è necessario alcun redirect per dirottare gli utenti mobile sul sito a loro dedicato

SITO RESPONSIVO – CONTRO

- **Un unico sito web**: il fatto di avere un unico sito è sicuramente un punto a favore del responsive design ma, volendo, può essere considerato anche un punto a sfavore. In queste condizioni infatti l'utente non ha la possibilità di scegliere se visualizzare il sito mobile oppure quello desktop per accedere, eventualmente, a tutte le funzionalità da esso fornite
- **Complessità tecnica**: il responsive design fa uso delle media query e necessita quindi, per poter essere correttamente realizzato e mantenuto, di solide conoscenze in materia di CSS (e spesso anche a livello javascript)

Come detto, all'interno di questa guida prenderemo in considerazione l'approccio responsivo rimandando alla manualistica on line di Passweb per maggiori informazioni relativamente alla realizzazione di un sito mobile ad hoc mediante l'utilizzo delle Varianti Mobile e della relativa componentistica Passweb (http://www.edupass.it/manuali/manualistica-passweb/siti-ecommerce/manuale-prodotto?a=manuale-passweb-ecommerce/live-editing/varianti-sito)

ATENZIONE! per utenti che hanno una buona conoscenza di Passweb ma non dispongono di conoscenze avanzate a livello CSS e javascript è consigliabile realizzare un sito in versione mobile utilizzando la relativa componentistica Passweb

Di seguito passeremo ad analizzare alcuni concetti di base che fungono da prerequisito indispensabile per poter realizzare correttamente un sito responsivo.

Verrà poi analizzato, in termini generali, un particolare framework css (uikit.css) che facilità notevolmente la creazione e la gestione di questa particolare tipologia di siti.

Infine vedremo come poter applicare ed utilizzare questo stesso framework per realizzare un sito Passweb responsivo.

RESPONSIVE DESIGN – CONCETTI DI BASE

Il **responsive web design** è una tecnica di progettazione dei siti web che consente di adattare i contenuti delle pagine a qualsiasi risoluzione o orientamento dello schermo su cui si sta navigando. Riprendendo una definizione data da Kayla Knight in un articolo di Smashing Magazine potremmo dire che:

"Con Responsive Design si intende indicare quell'approccio per il quale ***la progettazione e lo sviluppo*** *di un sito dovrebbero adattarsi al comportamento e all'ambiente dell'utente in base a fattori come le dimensioni dello schermo, la piattaforma e l'orientamento del device.* ***La pratica consiste in un mix di griglie, layout e immagini flessibili, più un uso accorto delle media queries CSS****.*

Quando l'utente passa dal suo PC desktop ad un iPad, il sito dovrebbe automaticamente adattarsi alla nuova risoluzione, modificare le dimensioni delle immagini e le eventuali interazioni con la pagina. In altre parole, un sito dovrebbe implementare tutte quelle tecnologie utili per un adattamento automatico alle preferenze dell'utente."

Da quanto si è detto finora si potrebbe ricavare l'idea che il Responsive Design abbia a che fare unicamente con i CSS e con l'adattamento del layout di una pagina web alle risoluzioni dei vari dispositivi.

Effettivamente il meccanismo che sta alla base di tutto l'approccio responsivo sono proprio le Media Query, introdotte da CSS 3, inoltre se si tratta di costruire layout che si adattano, è sempre con le proprietà CSS che avremo a che fare.

Più in generale però per comprendere appieno questo argomento in modo tale da riuscire a realizzare, anche in ambiente Passweb, un sito responsivo è bene fissare fin da subito alcuni concetti di fondamentale importanza che verranno poi ripresi nel corso di questa guida.

MOBILE FIRST

Leggendo in rete qualche articolo o tutorial dedicato al responsive design prima o poi ci imbatteremmo sicuramente nell'espressione **"mobile first".** Cerchiamo quindi di capire di che cosa si tratta essendo questo il primo concetto in assoluto da prendere in considerazione nella progettazione di un sito responsivo.

Il Mobile First è un approccio. E' il workflow progettuale che parte non più dal desktop per poi rimpicciolirsi nel mondo mobile, ma fa completamente l'inverso.

L'approccio classico alla progettazione e alla realizzazione di un sito web mobile molto spesso è subordinato alla progettazione e alla realizzazione del sito stesso in versione desktop.

Generalmente si parte dalla progettazione del sito in ambiente desktop ragionando in profondità sull'aspetto grafico, cercando di occupare tutti gli spazi a disposizione (inserendo in questo senso anche elementi non strettamente necessari) e pensando a quelle che potrebbero essere tutte le funzionalità di cui dotare il sito in queste condizioni.

Dopo aver ragionato per mesi sulla versione desktop del sito (e soprattutto dopo averla implementata) si passa alla progettazione mobile ed è a questo punto che, generalmente, inizia il dramma tentando in ogni modo di incastrare tutta la grafica, i contenuti e le funzionalità pensate per un dispositivo desktop in uno schermo molto più piccolo e su dispositivi che nulla hanno a che vedere con lo schermo di un pc.

Il risultato è un sito web non pensato per gli utenti mobile, che spesso ricalca la stessa struttura e lo stesso approccio del sito per desktop e questo causa spesso inefficienza e una scarsa soddisfazione da parte degli utenti.

Il mobile first è un approccio che ribalta completamente il modo di pensare e progettare un sito web.

Prima viene pensato e progettato il sito su mobile e solo successivamente si lavora alla versione desktop avendo però già come chiaro punto di partenza il sito in versione mobile.

Concepire un sito con l'approccio mobile-first permette di:

- pensare e creare contenuti diretti e semplici da leggere
- creare una grafica minimale, sobria e funzionale
- pensare di più all'interfaccia utente rendendo divertente la navigazione
- creare menù semplici possibilmente senza sotto-menù (o se necessario, senza andare oltre un secondo livello).

Questa semplificazione, che abbraccia la filosofia "less is more", ci da la possibilità di lavorare meglio, con efficienza e senza le mille distrazioni causate da dibattiti sul pixel in più o in meno o sulle più disparate posizioni degli elementi. Anche se può sembrare limitante lavorare prima su schermi più piccoli, la realtà dei fatti dimostra che concentrarsi di più sui reali bisogni informativi degli utenti permette di creare un sito più leggero, efficiente e divertente da usare. E qui iniziamo ad abbandonare il concetto di **GUI (Graphical User Interface)** per abbracciare la **NUI (Natural User Interface)**.

Come si fa a rendere naturale l'interfaccia utente? Ponendoci delle domande su alcuni fattori quali:

- il mio menù è facilmente raggiungibile usando una mano sola su uno smartphone di circa 4 o 5 pollici?
- l'utente che ha le cosiddette "manone" è in grado di premere i pulsanti e i link?
- devo scorrere all'infinito lo schermo per arrivare al fondo pagina perché ci sono chilometri di testo descrittivo?
- mi sto perdendo in mezzo a decine di voci di menù?

ATTENZIONE! La progettazione di un sito web e/o del suo layout grafico è un processo indipendente da Passweb e/o da qualsiasi altro strumento utilizzato per realizzare il sito. Ogni sito andrebbe prima pensato e progettato su carta e solo una volta chiaro il progetto si dovrebbe passare poi alla sua implementazione.

INDIVIDUARE I CONTENUTI DA MOSTRARE

Uno degli errori più frequenti che porta a scarsi risultati è quello di voler mantenere intatto l'intero contenuto sui vari display: nulla di più sbagliato! Elementi come slider, immagini di background, banner, footer complessi, sidebar, widgets, menù secondari di navigazione ecc… su uno smartphone, oltre a non avere molta utilità, potrebbero anche essere deleteri e peggiorare la user experience.

Se la superficie del display si riduce, inevitabilmente deve ridursi anche la mole di oggetti nella pagina!

E' un po' come fare la valigia e trasferire degli oggetti dall'armadio ad un contenitore molto più piccolo. Possiamo comprimere quanto vogliamo ma bisogna comunque mettersi l'anima in pace, qualcosa deve restare fuori, **ed è bene che sia ciò che è superfluo garantendo uno spazio adeguato all'essenziale**.

IL META TAG VIEWPORT

Un altro prerequisito fondamentale per realizzare un sito responsivo è quello di utilizzare correttamente il **viewport** per indicare in maniera specifica al browser quale debba essere la larghezza della superficie su cui disegnare la pagina web.

Tecnicamente questo è un meta tag da inserire nella sezione < head > della pagina e che dovrà avere una struttura analoga a quella qui di seguito riportata.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Il meta tag viewport fornisce al browser istruzioni utili per adattare le dimensioni e le proporzioni della pagina alla larghezza dello schermo del dispositivo.

In assenza di tale elemento, i browser dei dispositivi mobili applicano l'impostazione predefinita, ovvero eseguono il rendering della pagina alla larghezza utilizzata per gli schermi desktop. Sulla base di ciò, per migliorare l'aspetto dei contenuti, i browser aumentano automaticamente la dimensione dei caratteri e/o ridimensionano i contenuti in base allo schermo, assicurandosi di visualizzare l'intera pagina senza costringere l'utente a dover scrollare orizzontalmente

Per gli utenti, questo significa che le dimensioni dei caratteri potrebbero avere un aspetto disomogeneo e che potrebbe essere necessario toccare due volte lo schermo o avvicinare le dita eseguendo lo zoom per visualizzare e interagire con i contenuti. Per Google, invece, la pagina potrebbe **non essere riconosciuta come ottimizzata per i dispositivi mobili**, poiché richiede questo tipo di interazione.

Al contrario utilizzando il meta tag viewport come sopra indicato, stiamo dicendo in maniera specifica al browser del dispositivo di assegnare al viewport esattamente la larghezza del device perché saremo poi noi ad occuparci di ottimizzare i contenuti e non avremo quindi bisogno di alcun adattamento automatico.

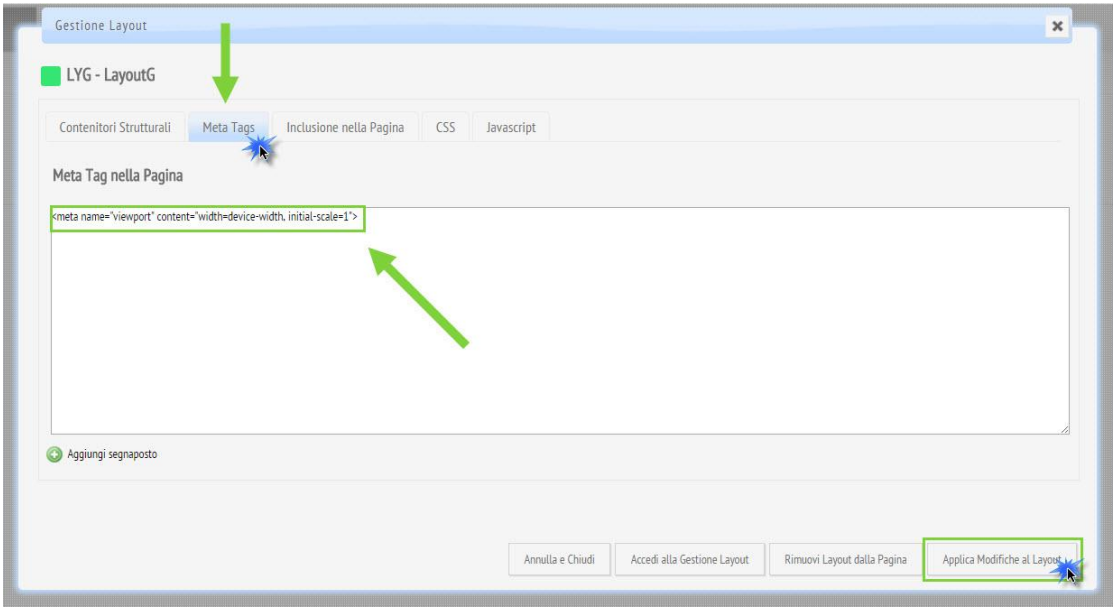
Per la pagina raffigurata a sinistra non è stato specificato alcun meta tag viewport. Di conseguenza il browser del dispositivo mobile presuppone che lo schermo abbia l'ampiezza di un computer desktop e adatta in modo conforme le proporzioni della pagina, ostacolando la lettura dei contenuti. A destra è rappresentata la stessa pagina con un viewport specificato, che corrisponde all'ampiezza dello schermo del dispositivo. Il browser del dispositivo mobile, pertanto, non ridimensiona la pagina e non impedisce la lettura dei contenuti.

A questo punto potrebbe però sorgere anche un dubbio: come facciamo a sapere, in pixel, quanto è largo il viewport dei vari dispositivi per ottimizzare i contenuti?

In realtà, come capiremo meglio in seguito, questa informazioni potrebbe non servirci affatto. In ogni caso se proprio non possiamo farne a meno, a questo indirizzo è possibile trovare una lista delle dimensioni del viewport su tutti i principali dispositivi.

IL META TAG VIEWPORT IN PASSWEB

In Passweb è possibile gestire il viewport **sia a livello di singola pagina che di intero sito** utilizzando per questo la sezione Meta Tags presente rispettivamente **nel layout di pagina e in quello di Variante**



All'interno di questa sezione è infatti possibile gestire tutti i meta tags non presenti nativamente nel codice delle pagine web generate da Passweb.

Per aggiungere un nuovo Meta Tag al layout è sufficiente inserirlo all'interno dell'apposita area di testo (**Meta Tag nella Pagina**) e cliccare poi sul pulsante **"Applica Modifiche al Layout"**.

ATTENZIONE! E' necessario inserire, all'interno di quest'area, l'intero markup del meta tag (come indicato nella figura sopra riportata) e non solo il valore dei suoi attributi.

Per maggiori informazioni in merito alla gestione dei layout Passweb si veda anche la sezione "*Live Editing – Layout*" del manuale di prodotto

LE MEDIA QUERY

Il concetto principale del web design responsivo è quello di **mantenere inalterato il codice html della pagina e agire sui fogli di stile per assegnare regole diverse in base al dispositivo da cui si sta navigando**.

In questo modo, ad esempio, è possibile nascondere sugli smartphone, alcuni elementi della pagina web che si è deciso invece di visualizzare su schermi più larghi quali possono essere quelli di un tablet o di un pc.

Per ottenere questo risultato dobbiamo ricorrere ad una funzionalità del CSS 3 chiamata **media query**. In sostanza andremo a porre delle condizioni in base alle quali un blocco di regole CSS potrà essere o non essere applicato.

Come recita la definizione che appare nelle specifiche CSS 3 *"una media query consiste nella dichiarazione di un tipo di media e di zero o più espressioni che verifichino le condizioni di validità o non validità delle caratteristiche di un certo media"*. Un semplice esempio chiarisce ogni dubbio in merito:

```
<link rel="stylesheet" media="only screen and (color)" href="colore.css" />
```

Con l'istruzione sopra evidenziata stiamo indicando di collegare il foglio di stile, colore.css, alla pagina web solo nel caso in cui questa pagina venga visualizzata sullo schermo di un pc e solo nel caso in cui questo sia uno schermo a colori. Cerchiamo di capire meglio come si arriva a questa conclusione.

Come si accennava nella definizione, una media query prevede innanzitutto l'uso di un tipo di media il che, in altre parole, equivale a specificare esattamente per quale dispositivo la regola CSS interessata dalla media query dovrà essere applicata. In questo senso possono essere presi in considerazione i seguenti media:

- **all**: il CSS si applica a tutti i dispositivi di visualizzazione;
- **screen**: schermo di computer;
- **print**: pagina stampata;
- **projection**: presentazioni e proiezioni;
- **speech**: dispositivi a sintesi vocale;
- **braille**: supporti basati sull'uso del braille;
- **embossed**: stampanti braille;
- **handheld**: dispositivi mobili con schermo piccolo e in genere dotati di browser con limitate capacità grafiche;
- **tty**: dispositivi a carattere fisso come i terminali;
- **tv**: visualizzazione su schermi televisivi.

ATTENZIONE! Nel caso in cui in una media query non venga indicato alcun media si intenderà che le specifiche regole CSS e/o lo specifico foglio di stile interessato dalla media query stessa sia rivolto a tutti i tipi di dispositivi (il che, in altri termini, equivale ad utilizzare come media della query il valore all)

Nell'esempio precedentemente considerato abbiamo usato come media il valore **screen** indicando, di fatto, che il foglio di stile colore.css debba essere applicato solo ed esclusivamente nel momento in cui il sito venga consultato da una specifica categoria di dispositivi: gli schermi dei computer.

Tornando ora sulla definizione, si ha anche che una media query è un'espressione logica che può essere vera o falsa a seconda del fatto che soddisfi o meno le condizioni indicate nella query. Ora, per costruire query più o meno complesse è possibile ricorre ad operatori logici come and, not, or …

Nell' esempio, media="screen **and** (color)", abbiamo inserito la parola chiave **and**. Il suo significato non è diverso da quello comune presente in tutti i linguaggi di programmazione ed indica quindi di eseguire un'unione logica tra le due espressioni.

La prima espressione, screen, indica, come detto, il fatto di applicare la media query solo nel caso in cui il sito venga visto sullo schermo di un pc; la seconda espressione, color, rappresenta invece una delle tante caratteristiche possibili dello schermo di un pc, nello specifico designa uno schermo a colori.

ATTENZIONE! da notare come, a livello sintattico, la seconda parte della media query, quella con le caratteristiche del tipo di media, debba essere racchiusa tra parentesi tonde

A questo punto il significato della nostra query è chiaro. Essa sarà vera (ovvero sarà applicato il foglio di stile colore.css) se la pagina viene visualizzata su uno schermo di computer a colori.

Una volta compreso il meccanismo si possono costruire query molto specifiche e complesse, per esempio concatenando tramite and più espressioni:

```
media="screen and (color) and (device-aspect-ratio: 16/9)"
```

Nell'esempio sopra indicato la query è vera se la pagina viene visualizzata su uno schermo a colori con aspect ratio di 16/9.

ATTENZIONE! Più media query possono essere raggruppate in una lista separandole con una virgola:

```
media="screen and (color), projection and (color)"
```

COME E DOVE DICHIARARE LE MEDIA QUERY

Le media query possono essere dichiarate in tre modi diversi:

1. Si definisce la query nel contesto dell'attributo **media** all'interno delle elemento **< link >**

   Esempio:

```
<link rel="stylesheet" media="only screen and (color)" href="colore.css" />
```

   In questo modo è possibile condizionare l'applicazione o meno di un intero foglio di stile al verificarsi delle condizioni indicate nella media query

Questo tipo di istruzione viene generalmente inserita nella sezione < head > della pagina

2. All'interno di un foglio di stile si può impostare una query mediante la direttiva **@media**

   Esempio:

```
@media screen and (color) {
  /* qui vanno le regole CSS */
}
```

   In questo modo è possibile condizionare l'applicazione di singole regole CSS al verificarsi o meno delle condizioni indicate nella media query.

   La direttiva **@media** può essere utilizzata solo ed esclusivamente all'interno di un foglio di stile come una qualsiasi altra regola CSS

3. Si può importare un CSS specifico attraverso una media query usando la direttiva **@import** all'interno di un altro foglio di stile

   Esempio:

```
@import url(colore.css) screen and (color);
```

   In questo modo è possibile condizionare l'importazione di tutte le regole presenti all'interno di un determinato foglio di stile in un altro foglio di stile al verificarsi o meno delle condizioni indicate nella media query.

   La direttiva **@import** può essere utilizzata solo ed esclusivamente all'interno di un foglio di stile

CARATTERISTICHE DEI MEDIA

Sulla base di quanto detto fino ad ora in relazione alle media query, appare abbastanza evidente come per poter sfruttare al meglio tutto il loro potenziale diventa di fondamentale importanza conoscere le caratteristiche dei media su cui si può intervenire per indirizzare le regole CSS.

In questo senso si potrebbe decidere di applicare determinate regole CSS in base, ad esempio, alle seguenti proprietà:

- **Larghezza dell'area di visualizzazione del documento – width (min-width, max-width)**

  Come precedentemente indicato **l'area di visualizzazione del documento è rappresentata, su di un normale browser web, dal viewport e non ha assolutamente nulla a che fare con le dimensioni dello schermo del dispositivo**.

  Per poter applicare delle regole CSS in base a questo elemento è possibile utilizzare la proprietà **width** esprimendone i valori in una qualsiasi unità di misura (px, em, %)

  Esempio

```
@media screen and (width: 400px) {
  /* regole CSS */
}
```

  Nell'esempio sopra riportato le regole CSS verranno applicate solo ed esclusivamente nel momento in cui l'area di visualizzazione del documento (e quindi il viewport impostato) sia uguale esattamente a 400px

  ATTENZIONE! Nella realizzazione di siti responsivi si tende ad indirizzare le varie regole CSS non tanto in base all'esatta larghezza del viewport quanto più genericamente in base a quella che possa essere la larghezza minima e/o massima dell'area di visualizzazione del documento.

  In questo senso andranno quindi utilizzate le proprietà **min-width** e **max-width**

  Esempio

```
@media screen and (min-width: 400px) and (max-width: 1024px) {
  /* regole CSS */
}
```

  Nell'esempio sopra riportato le varie regole CSS verranno applicate quando l'area di visualizzazione del documento ha una larghezza compresa tra i 400 e i 1200 pixel

- **Altezza dell'area di visualizzazione del documento – height (min-height**, **max-height)**

  Per poter applicare delle regole CSS in base a questo elemento è possibile utilizzare la proprietà **height** esprimendone i valori in una qualsiasi unità di misura (px, em, %).

  Come nel caso della larghezza anche questa volta è inoltre possibile ragionare in termini di altezza minima e massima utilizzando le proprietà **min-height** e **max-height**

- **Larghezza dell'intera area di rendering del dispositivo - device-width (min-device-width, max-device-width)**

  A differenza dell'area di visualizzazione del documento (il viewport) con il termine area di rendering del dispositivo si intende fare riferimento esattamente **alla larghezza dello schermo del dispositivo di visualizzazione**

  Per poter applicare delle regole CSS in base a questo elemento è quindi necessario utilizzare la proprietà **device-width**

  Esempio

```
@media screen and (device-width: 400px) {
  /* regole CSS */
}
```

  Nell'esempio sopra riportato le regole CSS verranno applicate solo ed esclusivamente nel momento in cui lo schermo del dispositivo di visualizzazione sia largo esattamente 400 pixel

  Anche in questo caso risulta molto più utile indirizzare le vari regole CSS non tanto in base all'esatta larghezza del dispositivo quanto più esattamente sulla base di quella che possa essere la sua larghezza minima o massima.

  In questo senso andranno quindi utilizzate le due proprietà **min-device-width** e **max-device-width**

- **Orientamento del dispositivo - orientation**

  Nella realizzazione di un sito responsivo può diventare molto importante adattare i contenuti della pagina anche in base a come l'utente ruota il dispositivo.

  In questo senso la proprietà **orientation** consente di indirizzare l'applicazione di determinate regole CSS in base al fatto che il dispositivo di visualizzazione sia posto in modalità **landscape** (orizzontale – la larghezza è maggiore dell'altezza) oppure **portrait** (verticale – l'altezza è maggiore della larghezza).

  Esempio

```
@media all and (min-device-width: 481px) and (max-device-width: 1024px) and (orientation:portrait){
  /* regole CSS */
}
```

  Nell'esempio sopra indicato le regole CSS verranno applicate solo ed esclusivamente sui dispositivi il cui schermo ha una larghezza compresa tra i 481 e i 1024 pixel e solo ed esclusivamente nel momento in cui questi dispostivi siano stati posti in modalità portrait (ossia in verticale)

- **Rapporto tra larghezza e altezza dell'area di visualizzazione del documento – aspect-ratio**

  La proprietà **aspect-ratio** identifica il rapporto tra la larghezza e l'altezza dell'area di visualizzazione del documento (che ricordiamo ancora una volta essere identificata dal viewport).

  I valori di questa proprietà di esprimono attraverso due numeri interi separati dal simbolo /

  Esempio

```
@media screen and (device-aspect-ratio: 16/9){
  /* regole CSS */
}
```

  Nell'esempio sopra indicato le regole CSS verranno applicate solo ed esclusivamente su quei dispositivi in cui il rapporto tra la larghezza e l'altezza del viewport è di 16/9

- **Rapporto tra larghezza e altezza dell'area di rendering del dispositivo – devie-aspect-ratio**

La proprietà **device-aspect-ratio** identifica il rapporto tra la larghezza e l'altezza dell'area di rendering del dispositivo (che ricordiamo ancora una volta essere identificata dalla larghezza dello schermo).

Anche in questo caso i valori di questa proprietà di esprimono attraverso due numeri interi separati dal simbolo /

- **Risoluzione del dispositivo di output - resolution**

  La proprietà **resolution** definisce la risoluzione (ovvero la densità di pixel) del dispositivo di output. I valori della risoluzione possono essere espressi in dpi (punti per pollice) oppure in dpcm (punti per centimetro).

  La proprietà può assumere anche i prefissi min e max per individuare rispettivamente la risoluzione minima e massima del dispositivo.

  Esempio

```
@media screen and (min-resolution: 300dpi){
  /* regole CSS */
}
```

Nell'esempio sopra indicato le regole CSS verranno applicate solo ed esclusivamente su quei dispositivi con una risoluzione di output maggiore di 300 dpi

DEFINIRE I BREAKPOINT

Come abbiamo visto nel capitolo precedente di questa guida grazie alle media query è possibile indicare al browser di applicare o meno determinate regole CSS al verificarsi di determinate condizioni.

I punti in cui si verificano queste condizioni sono esattamente ciò che viene definito **breakpoint**.

In altri termini dunque un breakpoint è un punto, su di una linea ideale che parte da 0, in cui si verifica una qualche modifica (tramite CSS) al layout di una pagina web.

Consideriamo un esempio per meglio comprendere questa definizione.

Supponiamo di dover progettare una pagina web il cui colore di sfondo dovrà variare in base alla larghezza dell'area di visualizzazione del documento (viewport) secondo le seguenti specifiche:

- Colore di sfondo predefinito ROSSO
- Viewport raggiunge i 480 px – Colore di sfondo GIALLO
- Viewport raggiunge i 768 px – Colore di sfondo VERDE
- Viewport raggiunge i 1024 px – Colore di sfondo BLU
- Viewport super i 1200 px – Colore di sfondo GRIGIO

Per tradurre in codice queste specifiche dovremo, ovviamente, ricorre a delle media query utilizzando, per indirizzare l'applicazione delle regole CSS, la proprietà **width,** e impostando i nostri breakpoint rispettivamente a **480 px, 768 px, 1024 px e 1200 px** come qui di seguito indicato:

```
body {
 background: red;
}

@media screen and (min-width: 480px) {
 body {
  background: yellow;
 }
}

@media screen and (min-width: 768px) {
 body {
  background: green;
 }
}

@media screen and (min-width: 1024px) {
 body {
  background: blue;
 }
}

@media screen and (min-width: 1200px) {
 body {
  background: gray;
 }
}
```

Una volta compreso il meccanismo che sta alla base delle media query viene naturale porsi una domanda:

Nella realizzazione di un sito responsivo, quanti breakpoint è opportuno/necessario gestire?

In linea generale la risposta a questa domanda dipende da vari fattori quali le caratteristiche del sito, il suo layout, le dimensioni degli schermi dei dispositivi su cui il sito dovrà essere visualizzato, le tecniche di implementazione adottate ecc … il che in sostanza equivale a dire "quanti ne servono affinchè la pagina si adatti al meglio alle caratteristiche dei dispositivi su cui dovrà essere visualizzata".

Fatta questa considerazione, va anche detto però che **la pratica ad oggi prevalente è quella che prevede di fissare i breakpoint in funzione alle dimensioni, in particolare alla larghezza, del viewport** per poi inserire qualche media query più specifica solo nel caso in cui ci si dovesse accorgere di qualche problema su determinati modelli di dispositivi.

In questo senso è anche abbastanza evidente, considerando l'ampia gamma di dispositivi disponibili sul mercato, come non possa essere possibile scrivere delle media query specifiche per ogni dispositivo, considerando esattamente quelle che sono le dimensioni del suo viewport. La tendenza è quindi quella di ragionare per tipologia di dispositivo prendendo in considerazione, per il layout del proprio sito, tre diverse versioni:

- un layout per gli **smartphone**
- un layout per i **tablet**
- un layout per i **pc desktop**

In considerazione di ciò, sia che si decida di adottare l'approccio mobile first, sia che si decida di lavorare in maniera "tradizionale", avremo sempre bisogno di almeno due media query generali.

Supponendo infatti di adottare l'approccio mobile first, il layout per smartphone non avrà bisogno di specifiche media query (essendo il default); saranno invece necessarie due media query: una per indirizzare la visualizzazione del sito sui tablet ed una per indirizzare la visualizzazione del sito sugli schermi pc.

Allo stesso modo, nel caso in cui si voglia utilizzare l'approccio tradizionale considerando come standard il layout per desktop, avremo comunque bisogno di almeno due media query necessarie questa volta per indirizzare la visualizzazione del sito su tablet e smartphone. Alcuni esempio di queste media query possono essere trovate al seguente indirizzo:

https://css-tricks.com/snippets/css/media-queries-for-standard-devices/

ATTENZIONE! I valori utilizzati in queste media query sono più o meno standard e nella maggior parte dei casi hanno come riferimento i device Apple. Questo non vuol dire però realizzare dei siti ottimizzati solo per iPhone o iPad a discapito di altri device; i valori indicati infatti servono più che altro come punto di riferimento per una specifica tipologia di dispositivo

Ognuna delle media query indicate potrebbe poi essere sdoppiata in base all'orientamento del dispositivo (portrait o landscape)

ATTENZIONE! L'utilizzo di determinati framework, come ad esempio **uikit.css** che andremo ad esaminare meglio nel seguito di questa guida (ma anche bootsrap, foundation ecc…), ci può aiutare notevolmente sotto questo aspetto in quanto ognuno di essi implementa già una versione standard (e comunque personalizzabile secondo le proprie specifiche esigenze) di tutte le media query necessarie per visualizzare correttamente i componenti del framework stesso su viewport "piccoli, medi e grandi"

LE MEDIA QUERY IN PASSWEB

Come evidenziato nei precedenti capitoli di questa guida ("*Come e dove dichiarare le media query*"), in generale le media query possono essere dichiarate in tre modi differenti.

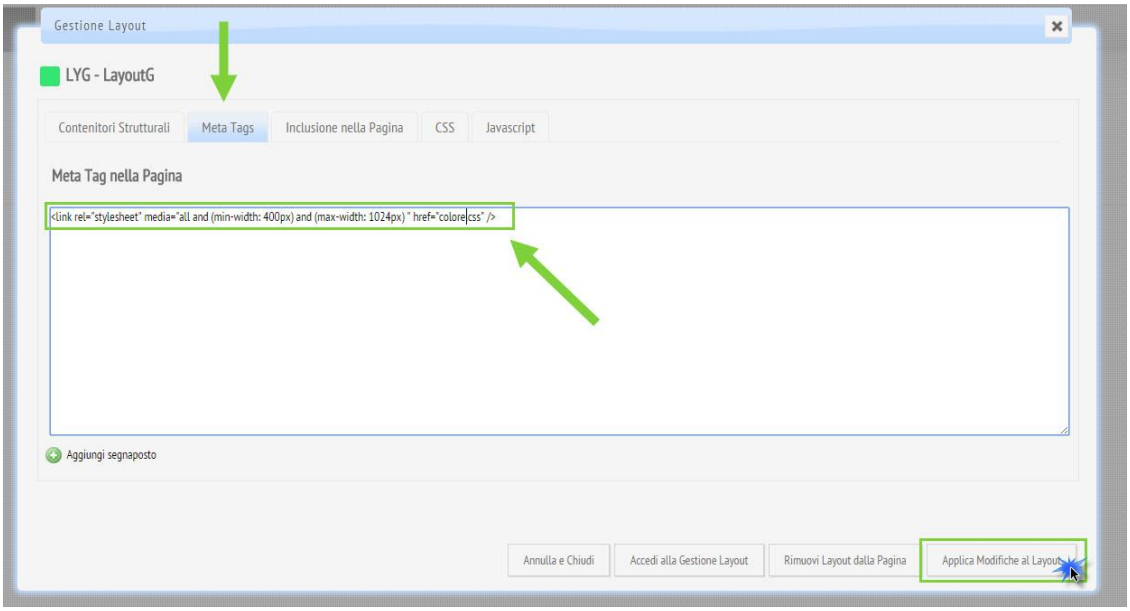
A seconda di come si intenda gestire questa dichiarazione, in Passweb sarà necessario agire in due modi diversi, che prevedono entrambi, l'utilizzo dei layout (**oltre al fatto, chiaramente, di dover scrivere tutto il codice CSS necessario**)

Nello specifico:

- nel caso in cui si dovesse decidere di creare uno o più fogli di stile (file .css) esterni in cui racchiudere tutte le regole CSS che dovranno essere applicate al verificarsi di specifiche condizioni, sarà poi necessario applicare la media query nel contesto dell'attributo **media** all'interno delle elemento **< link >** che identifica il file CSS da utilizzare.

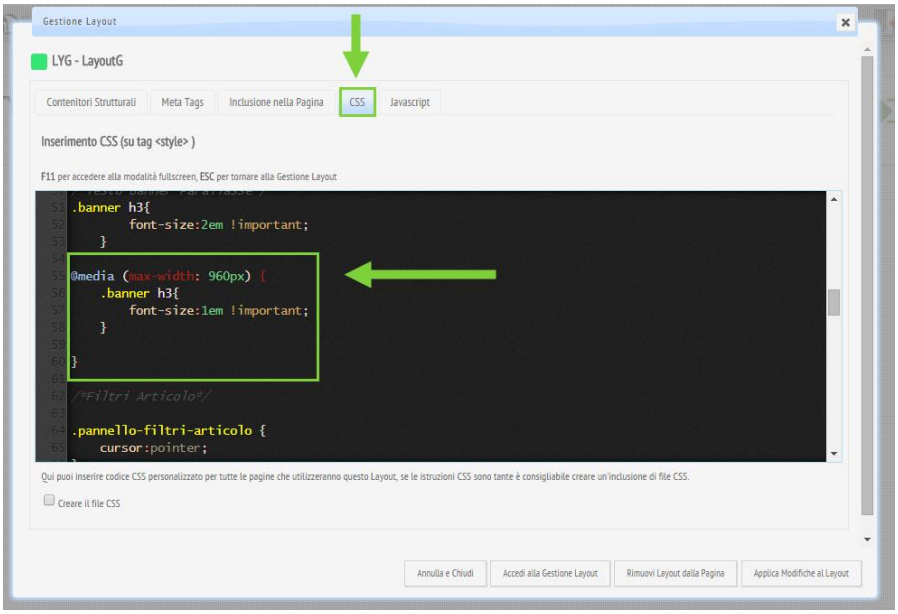
  ATTENZIONE! In queste condizioni non è sufficiente importare il file .css esterno tramite la sezione "Inclusione nella Pagina" perché questo tipo di inclusione non consente di gestire l'attributo media del relativo elemento **< link >**

  Per risolvere il problema sarà quindi necessario utilizzare la sezione "**Meta Tags**" del layout, sezione questa che ci consente di gestire manualmente l'elemento < link > potendo inserire per esso tutti gli attributi necessari, compreso ovviamente l'attributo media.



Nell' esempio riportato in figura il file colore.css viene caricato nelle pagine associate al layout in esame solo ed esclusivamente nel momento in cui il sito viene visitato da dispositivi in cui l'area di visualizzazione del documento (viewport) è compresa tra i 400 e i 1024 pixel

- nel caso in cui si desideri impostare una query mediante le direttive **@media** o **@import,** il relativo codice necessario dovrà essere inserito direttamente all'interno della sezione CSS del layout di pagina / sito



Per maggiori informazioni in merito alla gestione dei layout Passweb si veda anche la sezione "*Live Editing – Layout*" del manuale di prodotto

LAYOUT FLUIDI E MISURE RELATIVE

Da quanto visto nei precedenti capitoli di questa guida dovrebbe ormai essere abbastanza chiaro che quando si parla di design per dispositivi mobili, il concetto di pixel così come lo abbiamo sempre utilizzato assume sfaccettature diverse e non rappresenta più quel caposaldo che abbiamo sempre utilizzato nel mondo desktop.

Tra diverse dimensioni del viewport, pixel density, specifiche di ogni dispositivo, mille risoluzioni e orientamenti diversi, pensare, al giorno d'oggi, di realizzare un sito responsivo utilizzando, per i vari elementi della pagina, dimensioni fisse in pixel diventa assolutamente improponibile.

Per fortuna la soluzione a questo problema esiste, ed è anche abbastanza semplice: **non usare le dimensioni in pixel ma ragionare in termini percentuali.** Questo ci consente di adattare il layout a qualunque risoluzione, facendo in modo che gli elementi della pagina si adattino di conseguenza.

ATTENZIONE! L'utilizzo delle sole dimensioni percentuali non permette in alcun modo di realizzare un sito responsivo.

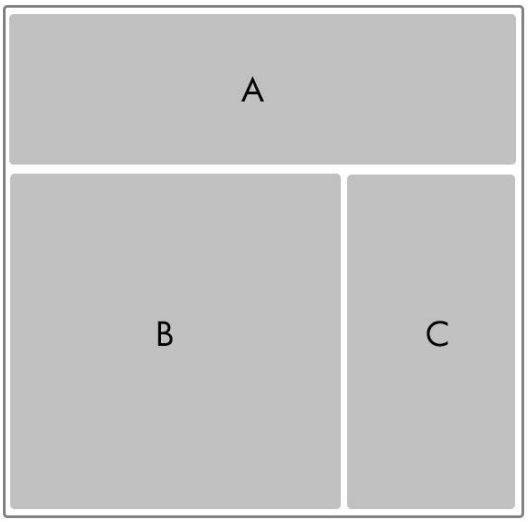
Il valore di queste dimensioni andrà comunque variato, infatti, a seconda del fatto che la pagina web venga visualizzata su schermi di grandi, medie o piccole dimensioni e per far questo è necessario ricorre alle media query.

Le dimensioni percentuali, assieme all'utilizzo delle media query, rappresentano, in buona parte, tutto quello che serve per partire nella realizzazione di un semplice sito responsivo. Se, ad esempio, dovessimo creare una pagina web strutturata, su desktop, in tre distinte colonne di pari dimensioni, ognuna di esse dovrebbe avere una larghezza del 33.333%; quando poi si passa agli schermi degli smartphone, con l'opportuna media query, si potrà impostare questa larghezza al 100% in modo che le tre colonne vadano ad occupare, in queste condizioni, l'intera larghezza della pagina disponendosi esattamente una sotto l'altra.

Indipendentemente dal fatto di utilizzare l'approccio "mobile first" o quello tradizionale, utilizzando quindi le media query per riadattare il layout del sito rispettivamente su schermi di grandi o di piccole dimensioni, la cosa di fondamentale importanza da avere sempre ben chiara in mente è come dovranno essere disposti i contenuti sugli schermi di piccole dimensioni e quale dovrà essere quindi il layout da utilizzare per ottenere questa disposizione.

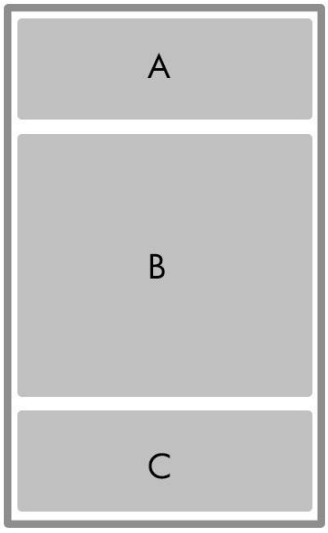
In questo senso la parola d'ordine è **semplificare,** ottimizzando al meglio il poco spazio a disposizione. Tradotto in codice per il layout ciò significa che sui dispositivi mobili, la modalità ottimale è quella della **linearizzazione degli elementi** che compongono la pagina. Nella stragrande maggioranza dei casi dunque su schermi di piccole dimensioni, il layout del sito dovrà avere un'unica colonna all'interno della quale i vari blocchi di contenuto dovranno disporsi uno sotto l'altro.

Avendo ben chiaro questo concetto, potremmo anche implementare il nostro sito utilizzando l'approccio tradizionale e creando quindi, per prima cosa, la versione desktop che potrebbe avere un layout del tipo di quello rappresentato in figura.

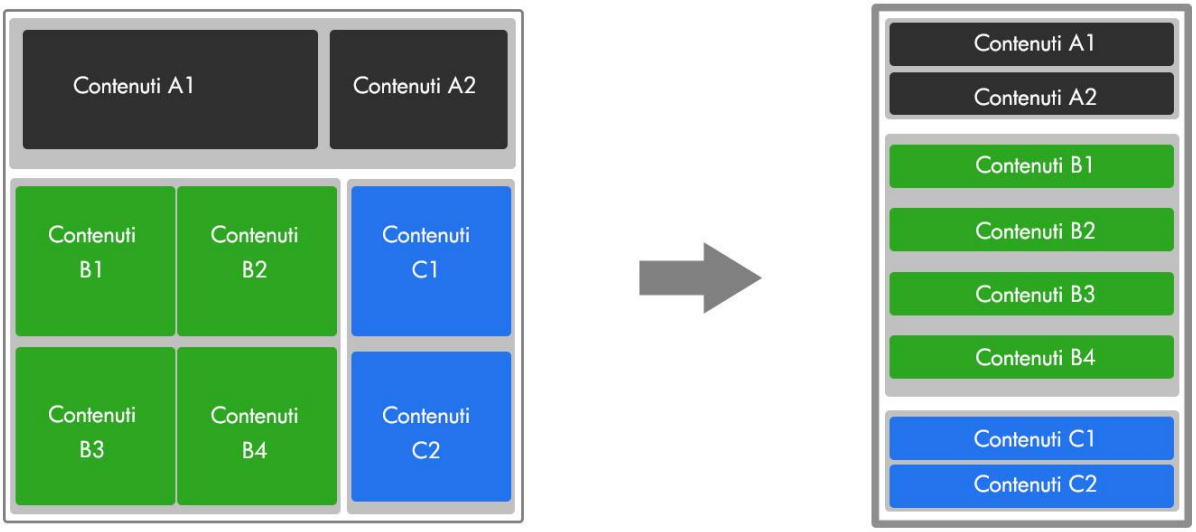


Nel fare questo però dovremo sempre avere ben presente l'obbiettivo finale; i tre elementi strutturali A (testata alta), B (colonna centrale) e C (colonna destra) dovranno quindi avere dimensioni in percentuale (es: A:100%, B:70%, C:30%), in maniera tale da potersi adattare automaticamente alle diverse risoluzioni degli schermi desktop e conseguentemente alle diverse possibili larghezze della pagina web.

Tramite una semplice media query dovremo poi variare, per schermi di piccole dimensioni, la larghezza dei contenitori strutturali B e C impostandola, in entrambi i casi al 100%, in modo tale che anche questi due contenitori occupino, in larghezza l'intera pagina disponendosi uno sotto l'altro come mostrato nella figura di seguito riportata.



Lo stesso tipo di ragionamento dovrà essere utilizzato anche per i contenuti inseriti all'interno dei blocchi strutturali A, B e C. Tali contenuti dovranno quindi essere a loro volta racchiusi all'interno di appositi box per i quali dovrà essere definita una larghezza in percentuale tale da affiancarli o linearizzarli (disponendoli uno sotto l'altro) a seconda del fatto che il sito venga visualizzato su schermi di grandi o piccole dimensioni.



Schema di Layout su più colonne (desktop)

Schema di Layout linearizzato (dispositivi mobile)

Per poter creare un layout di questo tipo, sia a livello di contenitori strutturali che di contenuti presenti all'interno di questi stessi contenitori, risulta particolarmente utile (anche e soprattutto nell'ottica di ricorrere poi ad un framework responsivo) strutturare la pagina web utilizzando un sistema a griglia analogo a quello descritto nei successivi capitoli di questa guida.

IL SISTEMA A GRIGLIE

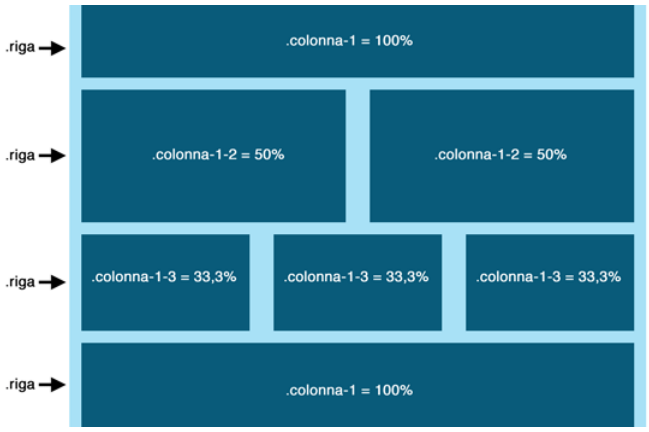
Il **Grid System** o **Sistema a Griglia** altro non è se non un sistema di suddivisione degli spazi di una pagina web particolarmente utile per cerare design proporzionati, efficaci e semplici da gestire.

In parole povere si tratta di suddividere la pagina web in un certo numero di righe e ogni riga in un certo numero di colonne immaginarie di dimensione fissa e separate da un margine (detto **gutter**) prestabilito.

In questo modo quando dovremo decidere la dimensione (in larghezza) di un certo elemento potremo ragionare non più in termini assoluti (pixel, punti e altro) ma in termini appunto di "colonne".

Un blocco di contenuti inserito all'interno di una riga non avrà più una larghezza di 100px o del 10% della larghezza del contenitore padre, ma occuperà ad esempio 1 delle N colonne in cui avevamo pensato di suddividere la riga in esame. L'unità di misura diventa quindi "la

colonna" e l'effettiva larghezza di un blocco di contenuti dipenderà da quante colonne occupa e da quante sono le N colonne in cui avevamo pensato di suddividere la riga



Nell'esempio illustrato in figura si è pensato di suddividere:

- la prima e la quarta riga in una sola colonna.

  In queste condizioni se dovessimo decidere di posizionare all'interno di queste righe un blocco di contenuti e di fargli occupare una colonna, la sua effettiva larghezza sarà ovviamente pari al 100% e andrà quindi ad occupare interamente la riga in esame

- la seconda riga in due colonne.

  In queste condizioni se dovessimo decidere di posizionare all'interno di questa riga un blocco di contenuti e di fargli occupare una sola colonna, la sua effettiva larghezza sarà pari alla metà (50%) della riga (il contenuto verrà posizionato quindi all'interno del primo dei due blocchi presenti in figura). Se poi volessimo, anche in questo caso, fare in modo che il nostro blocco di contenuti occupi, in larghezza, l'intera riga dovremo allo specificare che la sua occupazione non è più di una ma bensì di due colonne

- la terza riga in tre colonne.

  In questo caso per fare in modo che il nostro contenuto occupi, in larghezza, l'intera riga, dovremo assegnargli una dimensione di 3 colonne. Assegnandogli invece una dimensione di 2 colonne, lo spazio ad esso riservato sarà pari a quello individuato dai primi due blocchi presenti in figura.

Strutturare un layout a griglia significa quindi dare priorità ad elementi come l'ordine, la razionalità e la proporzione tra i vari elementi. Fondamentale, in questo senso, risulta la fase di progettazione che precede la scrittura del codice e l'effettiva implementazione del sito. Ad un livello basilare e molto semplificato, sarà sufficiente approntare un bozzetto del layout che stabilisca misure e proporzioni di quelli che sono i due componenti fondamentali di una griglia: **le colonne e lo spazio interno**, ovvero quello che separa una colonna dall'altra (il gutter).

Tipicamente, in un layout a griglia le colonne potranno assumere una larghezza variabile (pur rispondendo a precise regole di proporzionalità), mentre lo spazio interno sarà fisso.

Sempre semplificando, sono tre le operazioni di base da compiere nella fase di progettazione:

- definire la larghezza complessiva del layout
- impostare lo spazio che separa le colonne
- stabilire il numero di colonne che ci servono

Ora, se nel definire questi elementi si dovessero utilizzare delle dimensioni in pixel, quello che otterremmo sarebbe pur sempre un sistema a griglia in cui però la griglia stessa avrà delle dimensioni fisse.

Nell'ottica di realizzare un sito responsivo dovremmo invece poter disporre di una griglia fluida in grado di adattarsi alle diverse dimensioni dell'area di visualizzazione del documento (viewport). Per far questo dovremo quindi, ancora una volta, da una parte utilizzare dimensioni percentuali e dall'altra ricorre all'utilizzo delle media query in maniera tale da poter variare il numero di colonne occupate da un certo blocco di contenuti in relazione alle effettive dimensioni della pagina web.

In altri termini dunque mentre su schermi di grandi dimensioni due distinti blocchi di contenuti potrebbero, ad esempio, occupare ciascuno 2 delle 4 colonne in cui avevamo pensato di suddividere una riga, disponendosi quindi uno a fianco dell'altro, su schermi di piccole dimensioni questi stessi blocchi di contenuti dovranno occupare ciascuno 4 colonne in maniera tale da assumere una larghezza pari al 100%, linearizzare il layout e disporsi uno sotto l'altro.

Dal punto di vista tecnico, sarà necessario, per prima cosa, strutturare il markup HTML della pagina in maniera tale da implementare il layout a griglia pensato in fase di progettazione.

Facendo sempre riferimento alla griglia rappresentata nella figura precedente potremmo quindi utilizzare un markup di questo tipo:

```
<div class="riga">
  <div class="colonna-1">
```

```
    [...]
  </div>
</div>

<div class="riga">
  <div class="colonna-1-2">
    [...]
  </div>
  <div class="colonna-1-2">
    [...]
  </div>
</div>

<div class="riga">
  <div class="colonna-1-3">
    [...]
  </div>
  <div class="colonna-1-3">
    [...]
  </div>
  <div class="colonna-1-3">
    [...]
  </div>
</div>

<div class="riga">
  <div class="colonna-1">
    [...]
  </div>
</div>
```

I blocchi orizzontali sono racchiusi in un **div** con classe **.riga**. Ciascuna colonna all'interno di una riga è impostata con una classe che fa riferimento al suo rapporto in larghezza rispetto alla larghezza complessiva;

- .colonna-1-3 significa "colonna pari a 1/3 della larghezza complessiva della riga" (33.33%);
- .colonna-1-2 significa "colonna pari alla metà (1/2) della larghezza complessiva" (50%);
- .colonna-1 è la colonna che occupa l'intera larghezza (100%), la nostra unità di base.

Definito il markup HTML dovremo poi passare alla definizione del foglio di stile in maniera tale da rendere la griglia completamente funzionante. Nel definire le regole CSS dovremo fare in modo, ad esempio, che le varie righe occupino, in larghezza, l'intera area di visualizzazione del documento. Potremmo quindi impostare la classe .riga in questo modo

```
.riga {
   width: 100%;
   margin: 0 auto;
}
```

Considerando poi che la prima e la quarta riga della griglia sono state pensate per contenere una sola colonna allora anche il contenitore all'interno del quale andranno inseriti i contenuti della prima riga dovrà avere una larghezza del 100%.

```
.colonna-1{
  width: 100%;
}
```

La seconda riga della griglia dovrà invece contenere due colonne per cui i contenitori all'interno dei quali inserire i contenuti da posizionare in questa riga dovranno avere questa volta una larghezza del 50%

```
.colonna-1-2 {
  width: 50%;
}
```

In realtà impostando una width del 50% come sopra evidenziato, le due colonne saranno perfettamente attaccate e lo spazio tra di esse (il gutter) sarà nullo.

Se volessimo aggiungere dello spazio, tramite margini o (più correttamente) padding, dovremo di contro ridurre la larghezza dei due blocchi di contenuti per consentire di allinearli comunque sulla stessa riga. Considerando però che la larghezza dei due blocchi di contenuti deve

essere in percentuale mentre lo spazio di separazione tra di essi dovrebbe essere fissato in pixel, riuscire a gestire il tutto mantenendo i due blocchi allineati potrebbe risultare difficile e anche un solo pixel in più potrebbe distruggere il layout.

Per risolvere questo problema e impostare un gutter fisso tra le due colonne, mantenendo comunque la loro larghezza in percentuale (come del resto pensato in fase di progettazione) dovremo inserire un altro paio di regole CSS

```
* {
  box-sizing: border-box;
}

[class*='colonna-'] {
 padding: 0 20px;
}
```

Cerchiamo di spiegarne meglio il significato.

Nella prima delle due regole sopra evidenziate il **selettore universale *** consente di applicare la relativa regola a tutti gli elementi della pagina.

La proprietà **box-sizing** impostata sul valore **border-box** consente invece di gestire un box model più logico e naturale. In questo modo infatti le dimensioni assegnate ad un elemento **saranno comprensive anche dei padding e del bordo**.

Al contrario se per la proprietà box-sizing si utilizzasse il valore di default (content-box) come da specifiche del CSS 2.1, la dimensione indicata per un elemento andrebbe considerata escludendo padding e bordi.

In altri termini dunque se per un dato elemento utilizzassimo le seguenti proprietà

```
box-sizing: content-box;
width: 200px;
padding: 20px;
```

dovremmo poi assumere che la larghezza reale dell'elemento sia di 20px+200px+20px=240px.

Utilizzando invece

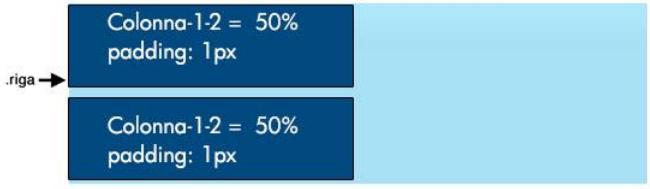
```
box-sizing: border-box;
width: 200px;
padding: 20px;
```

la larghezza complessiva dell'elemento sarà sempre di 200px e questo includerà anche i 40px complessivi dei bordi.

Ora è ovvio che a livello visuale, quello che si può ottenere usando border-box non è diverso da quello che si otterrebbe usando, con valori e misure diversi, content-box.

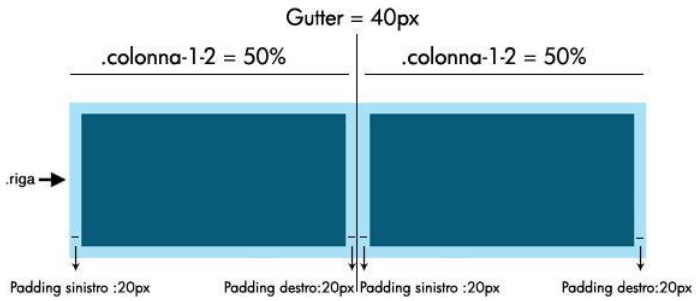
Ci sono però casi in cui i vantaggi sono evidenti in termini di approccio alla realizzazione di layout soprattutto nel gestire layout fluidi e misure percentuali.

Nel caso in questione infatti utilizzando content-box e fissando la larghezza dei due blocchi di contenuto presenti sulla seconda riga al 50%, basterebbe poi aggiungere anche un solo pixel di padding necessario per impostare lo spazio di separazione tra le due colonne, per fare in modo che la larghezza complessiva degli elementi presenti sulla riga superi il 100% e che quindi il secondo blocco di contenuti vada a disporsi sotto il primo.



Utilizzando invece border-box, il valore del 50% impostato per la larghezza dei due blocchi di contenuto, sarà quello complessivo compresi anche eventuali padding che a questo punto potranno quindi essere utilizzati per gestire lo spazio di separazione tra le due colonne.

Con la seconda delle due regole precedentemente analizzate viene fatta proprio questa operazione: con il **selettore di attributo [class*='colonna-']** selezioniamo tutti gli elementi che hanno una classe che nel nome comprende la stringa "colonna-", per essi andiamo poi ad impostare un padding destro e sinistro di 20px creando quindi uno spazio di separazione tra le due colonne pari a 40px



La terza riga della griglia, è stata infine pensata per contenere tre distinte colonne opportunamente spaziate. Come già fatto per le righe della seconda colonna dovremo quindi considerare le seguenti regole CSS

```
.colonna-1-3 {
  width: 33.33%;
}
```

Le due regole CSS precedentemente analizzate e relative al box-sizing e al padding garantiscono anche in questo caso una separazione tra i tre blocchi di contenuti di 40px.

Le regole CSS appena considerate sono sicuramente di fondamentale importanza per comprendere i principi di funzionamento della griglia ma, ovviamente, non sono le uniche da considerare. Per rendere la griglia completamente funzionante vanno infatti impostate anche altre regole altrettanto importanti: la larghezza complessiva della griglia andrebbe comunque limitata sia come larghezza massima (max-width) sia come larghezza minima (min-width), va gestito correttamente il posizionamento (float, margin, clear ecc…) dei vari elementi ecc…

Inoltre per rendere la griglia responsiva a tutti gli effetti, come sappiamo, non è sufficiente utilizzare delle dimensioni in percentuale ma vanno comunque scritte apposite media query che ci consentano, quando necessario, di linearizzare correttamente il layout.

In questo senso potremmo quindi pensare di assegnare ai contenitori dei blocchi di contenuti presenti nella seconda e nella terza riga un ulteriore classe .colonna-small-1

```
<div class="riga">
  <div class="colonna-1">
    [...]
  </div>
</div>

<div class="riga">
  <div class="colonna-1-2 colonna-small-1">
    [...]
  </div>
  <div class="colonna-1-2 colonna-small-1">
    [...]
  </div>
</div>

<div class="riga">
  <div class="colonna-1-3 colonna-small-1">
    [...]
  </div>
  <div class="colonna-1-3 colonna-small-1">
    [...]
  </div>
  <div class="colonna-1-3 colonna-small-1">
    [...]
  </div>
</div>

<div class="riga">
  <div class="colonna-1">
    [...]
  </div>
</div>
```

Questa nuova classe verrà utilizzata per sovrascrivere la regola con cui avevamo precedentemente fissato la larghezza di questi blocchi di contenuti e dovrà quindi essere applicata attraverso un’apposita media query ad esempio solo nel caso in cui l’area di visualizzazione del documento (viewport) sia inferiore ai 400px

```
@media only screen and (max-width:400px) {

 .colonna-small-1{
     width: 100%;
```

```
    }
}
```

In queste condizioni dunque su schermi di piccole dimensioni riusciremo a linearizzare correttamente il nostro layout.

Detto ora che in Passweb una griglia fluida e responsiva del tipo di quella analizzata fino a questo momento può essere realizzata da zero utilizzando tanti componenti Contenitore opportunamente annidati, fissando le varie regole CSS e le media query necessarie mediante lo Style Editor di Passweb e le apposite sezione dei layout di Pagina e/o di sito, l'obbiettivo di questa guida non è comunque quello di spiegare nei dettagli questo procedimento.

Fortunatamente, infatti, sebbene possibile, non è comunque strettamente necessario costruirsi da zero una griglia responsiva, ma possono tranquillamente essere utilizzati dei framework responsivi che si rifanno esattamente agli stessi concetti di base fino ad ora analizzati mettendo però già a nostra disposizione tutte le classi CSS e le media query necessarie per costruire una griglia completamente e perfettamente funzionante.

Grazie a questi framework sarà quindi possibile implementare in Passweb una griglia responsiva utilizzando componenti Contenitori annidati e assegnando loro, in fasi di configurazione, le corrette classi previste dallo specifico framework che si è deciso di utilizzare.

Nei successivi capitoli di questa guida vedremo un po' più nel dettaglio come utilizzare il framework uikit.css per implementare, tra le altre cose, nel nostro sito Passweb una griglia responsiva.

TIPOGRAFIA

Nei precedenti capitoli abbiamo messo in evidenza come uno dei capisaldi nella realizzazione di un sito responsivo, sia quello di non utilizzare mai, se non strettamente necessario, delle dimensioni fisse in pixel cercando invece di ragionare sempre in termini percentuali.

Questo è un concetto generale che non vale solo per il dimensionamento in larghezza dei vari elementi presenti all'interno di una pagina, ma che deve essere preso in considerazione, ad esempio, anche per la tipografia del sito.

Anche l' unità di misura utilizzata per definire la dimensione dei font deve quindi essere un'unità di misura percentuale.

Prima di procedere oltre è opportuno fare un minimo di chiarezza sulle diverse unità di misura che possono essere adottate.

I PIXEL

I pixel sono stati per molto tempo l'unità di misura prediletta dai Web Designer proprio a seguito della loro precisione. Una volta impostato un valore a scelta per il font, le sue dimensioni sarebbero state esattamente le stesse qualsiasi dispositivo o browser l'utente avesse scelto di utilizzare. Questo può senza dubbio essere visto come un grande vantaggio, grazie al controllo sul risultato che le dimensioni fisse garantiscono allo sviluppatore.

Una volta impostate le dimensioni per un elemento "genitore", esse saranno applicate automaticamente anche all'elemento "figlio" per la cosiddetta regola dell'ereditarietà. Impostando ad esempio un valore al font-size di body, tutto ciò che è in esso contenuto o da esso dipendente, assumerà quel valore. Ciò comporta che nel momento in cui ad ogni singolo elemento della pagina dovessero essere assegnate delle dimensioni del font differenti, queste dovranno essere impostate manualmente sullo specifico elemento.

Come è facile immaginare, nel momento in cui il sito dovesse poi essere sottoposto a manutenzione o restyling, il Web Designer dovrebbe modificare uno ad uno tutti i valori impostati e questo non è sicuramente un inconveniente da poco; un'altra complicazione collegata alla scelta delle dimensioni fisse, riguarda il fatto che esse comportano seri problemi di accessibilità, in particolar modo in browser come Internet Explorer piuttosto datati. Le versioni di IE precedenti la 9 infatti non consentono il ridimensionamento dei caratteri per cui alcuni utenti potrebbero non riuscire ad utilizzare, quando necessario, le funzioni di zoom del browser e potrebbero non riuscire a leggere correttamente i contenuti del sito.

Infine guardando, chiaramente, al responsive design e alle tecnologie che si stanno imponendo sul mercato la scelta dei pixel non appare affatto la più saggia; i nuovi device sono dotati di schermi di dimensioni diverse e differenti densità di pixel e la scelta di una dimensione unica che possa andar bene per tutti i dispositivi, non è sicuramente semplice.

GLI EM

Un em è l'equivalente della dimensione (in pixel) definita nella regola CSS font-size.

Se ad esempio impostiamo per un < div > un testo con font-size 16 pixel, 1 em sarà uguale a 16 pixel, 2 em corrisponderanno a 32 pixel e così via.

Gli em sono ridimensionabili in tutti i browser, rispondendo quindi correttamente alle relative funzioni di zoom, e non è necessario impostare il valore per ogni singolo elemento in quanto anche in questo caso la regola può essere applicata "a cascata" dall'elemento "genitore" all'elemento "figlio".

Ovviamente, anche per questa scelta ci sono dei pro e dei contro da tenere in considerazione.

Da una parte bisogna infatti considerare che l'utilizzo di questa unità di misura, oltre a garantire una corretta risposta alle funzioni di zoom del browser, facilita notevolmente la manutenzione del sito. E' infatti sufficiente apportare delle modifiche alla dimensione fissa in pixel cui fanno riferimento tutti i vari elementi impostati in em, per far si che questi si aggiornino poi di conseguenza.

Supponiamo ad esempio di aver impostato le seguenti regole CSS

```
body {
font-size: 16px;
}

p {
font-size: 1em;
}

h1 {
font-size: 1.5em;
}
```

In queste condizioni il testo presente all'interno dei paragrafi (tag p) avrà dimensione pari a 16 x 1 = 16 px, mentre la dimensione dei titoli h1 sarà di 16 x 1.5 = 24 px.

Supponendo ora di voler aumentare la dimensione del testo portando i paragrafi a 17 px e gli h1 a 25.5 px non sarà necessario variare singolarmente le due regole relative ai tag p e h1 ma sarà sufficiente portare a 17 pixel il font-size del body per fare in modo che le due dimensioni in em si aggiornino di conseguenza.

Va anche sottolineato però come il fatto che i cambiamenti impostati al font-size di riferimento vengano applicati in automatico a tutti i contenuti impostati in em, può rappresentare anche una difficoltà, in conseguenza del fatto che nella realizzazione di una pagina web occorrerà sempre trovare una buona proporzione tra i testi con dimensioni differenti.

Inoltre bisogna anche considerare che il font-size in em di un certo testo fa sempre riferimento al font-size dell'elemento "genitore" per cui nel caso di elementi annidati potrebbe diventare complicato sapere esattamente a quanti pixel corrisponde una dimensione impostata in em. Cerchiamo di comprendere meglio questa affermazione con un semplice esempio.

Consideriamo il seguente markup HTML

```
<div class="contenitore_1">
  <div class="paragrafo_1 ">
    Testo Paragrafo 1
  </div>
  <div class="contenitore_2">
    <div class="paragrafo_2 ">
       Testo Paragrafo 2
    </div>
  </div>
</div>
```

e le seguenti regole CSS

```
.contenitore_1 {
  font-size: 15px;
}

.paragrafo_1 {
  font-size: 1 em;
}

.contenitore_2 {
  font-size: 1.5 em;
}

.paragrafo_2 {
  font-size: 1 em;
}
```

Nell'esempio sopra indicato pur avendo considerato sia per il paragrafo_1 che per il paragrafo_2, una dimensione di 1 em, questa corrisponderà nel primo caso (paragrafo_1) a 15 px mentre nel secondo caso (paragrafo_2) a 22.5 px.

Il paragrafo_2 infatti ha come elemento "genitore" il contenitore_2 per il quale è stato considerato un font-size di 1.5 em, e che ha a sua volta come elemento "genitore" il contenitore_1.

Per poter determinare a quanti pixel corrisponde il font-size del paragrafo_2 occorre quindi per prima cosa determinare i pixel del font-size del contenitore_2. Avremo quindi

font-size contentiore_2 → 1.5 x 15 = 22.5 px

fon-size paragrafo_2 → 1 x 22.5 = 22.5px

Considerando che il markup di una pagina HTML è generalmente costituito da svariati elementi anche piuttosto annidati tra loro è semplice comprendere come ci si possa facilmente perdere nel cercare di capire a quanti pixel corrisponda esattamente una dimensione in em.

PERCENTUALE

Al pari degli em, esaminati nel capitolo precedente, anche la percentuale rappresenta, ovviamente, un'unità di misura relativa e quindi perfettamente scalabile. Anche in questo caso, inoltre, le varie regole possono essere applicate "a cascata" dall'elemento "genitore" all'elemento "figlio".

In sostanza dunque, non c'è una gran differenza tra gli em e le dimensioni percentuali, anche se, allo stato attuale gli em sembrano essere l'unità di misura preferita sul web.

Indipendentemente dal fatto di utilizzare gli em o le percentuali, **occorre però tenere sempre in considerazione che entrambe queste unità di misura hanno bisogno di un valore di riferimento sulla base del quale verranno poi calcolate le corrispondenti dimensioni in pixel**.

Questo valore di riferimento non deve necessariamente essere un valore fissato in pixel ma può essere esso stesso un valore relativo (in em o in %). Ovviamente però, risalendo nella cascata degli elementi del DOM, bisognerà pur sempre giungere ad un valore del font-size fissato in pixel e sulla base del quale poter poi calcolare anche tutte le altre dimensioni.

Nel caso in cui dunque anche sugli elementi body o html dovesse essere impostato un font-size in unità di misura relativa (em o %) il valore di riferimento in pixel sarà quello indicato nei fogli di stile adottati a default dallo specifico browser.

In questo senso occorre anche fare una considerazione piuttosto importante legata, principalmente, all'utilizzo di Internet Explorer.

Nel caso in cui il font-size di riferimento fosse esso stesso relativo ed impostato in em, dipendentemente dalla versione utilizzata, Explorer potrebbe non essere in grado di calcolare correttamente le dimensioni del testo aumentando o diminuendo, senza seguire una regola specifica, il modo in cui i caratteri verranno effettivamente visualizzati.

Questo problema può essere risolto utilizzando appunto le dimensioni in percentuale ed impostando quindi nel body il font-size di riferimento al 100%

```
body {
  font-size: 100%;
}
```

In questo modo potremo definire il font-size degli altri elementi della pagina web in em avendo comunque la certezza che i contenuti testuali presenti all'interno della pagina, vengano correttamente ridimensionati senza alcun tipo di esagerazione.

I REM

I rem (**root em**) offrono una valida alternativa agli em. Essi si comportano alla stessa maniera degli em, ad eccezione di una differenza di fondamentale importanza: **i rem sono relativi sempre e soltanto all'elemento html (elemento root) piuttosto che ad ogni loro elemento padre**.

Questa differenza sostanziale agevola, non di poco, il corretto ridimensionamento del font, permettendo di gestire in maniera molto più semplice tutti i problemi che possono presentarsi nella costruzione a cascata del codice CSS con elementi annidati.

In queste condizioni infatti non sarà più necessario calcolare il font-size in pixel di un elemento genitore per determinare l'equivalente in pixel del font-size espresso in rem per i suoi elementi figli. Per effettuare questo calcolo, al contrario, sarà necessario prendere come valore di riferimento sempre e soltanto quello impostato per l'elemento html

ATTENZIONE! Nel caso in cui si dovessero utilizzare i rem come unità di misura diventa necessario impostare il valore di riferimento sull'elemento html

I rem sono supportati nella maggior parte dei browser moderni, incluso Opera, sin dalla versione 11, oltre ad Internet Explorer a partire dalla versione 9.

FONT-SIZE – CONCLUSIONI

Ci sono due princìpi fondamentali per creare una effettiva tipografia responsive.

- Il primo è **l'implementazione di un font ridimensionabile**. Ciò significa che non deve solo adattarsi in base alle dimensioni dello schermo, ma che sia ridimensionabile da parte dell'utente.
- Il secondo è **l'ottimizzazione della lunghezza delle righe di testo, per garantire un'ottima leggibilità**.

Per quanto riguarda il primo punto, la maggior parte dei webmaster utilizza i pixel o gli em per le dimensioni del testo. **Usare gli em o i rem è la scelta migliore, in quanto questi permettono agli utenti di ridimensionare il font nel browser.** Gli em sono però "relativi", dipendono cioè, dall'elemento padre e questo vuol dire che è un po' più complicato rispetto all'uso dei pixel, perché si dovrebbero fare dei calcoli per avere una dimensione di font omogenea, quando nello stesso sito sono presenti diversi elementi che necessitano diverse dimensioni del font.

I rem offrono una valida alternativa agli em. Si comportano alla stessa maniera ad eccezione di una differenza fondamentale: i rem sono relativi all'elemento html piuttosto che ad ogni loro elemento padre. Questa differenza sostanziale agevola non di poco il corretto ridimensionamento del font.

Nel momento in cui dovessero essere utilizzati i rem quale unità di misura per i font, è importante applicare una dichiarazione CSS all'elemento html, e non al body del documento.

La regola dovrebbe essere la seguente**:**

```
html { font-size:100%; }
```

In questo modo le unità di misura rem verranno applicate alla dimensione di default del dispositivo.

Oltre a ciò è necessario anche specificare la dimensione del font per ciascuna dimensione del device e, in questo senso, ci vengono in aiuto ancora una volta, le media query. Sarebbe opportuno provare diverse dimensioni di font su dispositivi reali, al fine di ottenere una perfetta leggibilità (considerando, tra le altre cose, che il tutto dipende anche dal tipo di font scelto).

Potremmo utilizzare ad esempio delle media queries come quelle qui di seguito riportate:

```
@media (max-width: 640px)
{
body {font-size:1.3rem;}
}

@media (min-width: 641px)
{
body {font-size:1.2rem;}
}

@media (min-width:960px)
{
body {font-size:1.4rem;}
}

@media (min-width:1100px)
{
body {font-size:1.6rem;}
}
```

Per quel che riguarda invece il secondo punto è bene sottolineare come le linee guida, dettate in questo senso da quelle che sono le principali tendenze sul web, prevedono di utilizzare tra i 50 ed i 75 caratteri per riga.

In ogni caso tutto quando detto fin' ora relativamente alla tipografia, non rappresenta una regola da dover seguire ad ogni costo. Bisogna infatti considerare sempre le specifiche esigenze del caso. Tanto per intenderci non è assolutamente detto che scalare proporzionalmente le dimensioni di tutti i testi del sito sia sempre la soluzione migliore.

Possono tranquillamente esserci casi in cui il font-size di determinati elementi (es. i titoli di pagina) non debba essere scalato come avviene invece per il testo del body; **nulla vieta dunque, di trattare i singoli elementi in maniera diversa utilizzando, laddove necessario delle dimensioni del testo in pixel**

I FONT FAMILY

Rimanendo sempre in ambito di tipografia, un altro elemento importante da prendere in considerazione oltre alle dimensioni del testo è quello relativo alla scelta del tipo di font da utilizzare.

I font disponibili su pc infatti sono generalmente diversi rispetto a quelli disponibili sugli smartphone e/o sui tablet, alcuni dei quali hanno giusto un font per famiglia generica (sans-serif, serif, monospace …).

Utilizzando, ad esempio, una regola come quella di seguito riportata

```
h1 {
font-family:Arial, Helvetica, sans-serif;
}
```

molto probabilmente su alcuni dispositivi Android i titoli h1 non verranno visualizzati ne in Arial, ne in Helvetica ma in Droid Sans.

I CSS3 e i vari servizi come Google Web Font (implementato nativamente in Passweb) possono permetterci di superare questi limiti, integrando nel sito font non comunemente presenti sul dispositivo.

Questa soluzione necessita però di scaricare dei contenuti aggiuntivi (i file dei font) dai vari fornitori di servizio, per cui mentre potrebbe rivelarsi effettivamente molto comoda nel caso in cui il sito venga visualizzato su schermi di pc (dove generalmente non si hanno grossi problemi a livello di connessione e una decina di kb in più o in meno non fa alcuna differenza), in mobilità, dove invece i limiti a livello di connessione potrebbero essere determinanti, potrebbe non essere la soluzione ottimale nel senso che potremmo ridurre le prestazioni del sito e aumentare i tempi di caricamento della pagina.

Anche in questo caso ci vengono in aiuto le media query che, nel momento in cui dovessimo porre l'attenzione sull'ottimizzazione delle prestazioni, potranno consentirci di utilizzare font differenti per differenti dispositivi.

Potremmo utilizzare, ad esempio, due media query di questo tipo:

```
@media only screen (max-width: 1024px) {
  body  {
    font-family:Arial, Helvetica, sans-serif;
  }
}


@media only screen (min-width: 1025px) {
  @import url(http://fonts.googleapis.com/css?family=Nothing+You+Could+Do);

  body {
    font-family: 'Nothing You Could Do';
  }
}
```

Nell'esempio indicato:

- nel caso in cui l'area di visualizzazione del documento (viewport) sia inferiore ai 1024px (assumiamo che in tal modo si possano identificare i dispositivi mobile) verrà utilizzato uno dei font standard presenti sul dispositivo (Arial, Helvetica o il font della famiglia generica sans-serif). In queste condizioni non verrà quindi scaricata nessuna risorsa esterna.
- nel caso in cui l'area di visualizzazione del documento (viewport) sia superiore ai 1024 px (assumiamo che in tal modo si possano indentificare la visualizzazione del sito su schermi per pc) verrà scaricato da Google, ed utilizzato all'interno del sito, il web font 'Nothing You Could Do'

IMMAGINI RESPONSIVE

Come più volte accennato nei precedenti capitoli di questa guida la chiave per ottenere un layout realmente flessibile e adattabile consiste nell'utilizzo di unità di misura relative in maniera tale che i vari elementi che compongono la pagina possano adattarsi in modo naturale ad una vasta gamma di device e risoluzioni di schermo.

Mentre però per i "contenitori di contenuti" (semplici tag div) possiamo facilmente giocare con le unità di misura per ottenere la fluidità di cui abbiamo bisogno, all'interno della pagina possiamo anche trovare oggetti, come le immagini, che per loro stessa natura hanno dimensioni fisse e questo, ovviamente, in un contesto responsive non va bene.

Una prima soluzione a questo problema esiste ed è anche piuttosto semplice.

Per avere immagini fluide e adattive rispetto al contesto in cui sono inserite è infatti necessario:

- **gestire in percentuale la dimensione, in larghezza del box, che le contiene**
- **impostare la larghezza (width) e l'altezza (height) dell'immagine sul valore auto**
- **impostare la larghezza massima dell'immagine sul valore 100%**

Queste operazioni possono essere effettuate, in Passweb, sia a livello di singole immagini sia a livello globale per tutte le immagini presenti all'interno del sito.

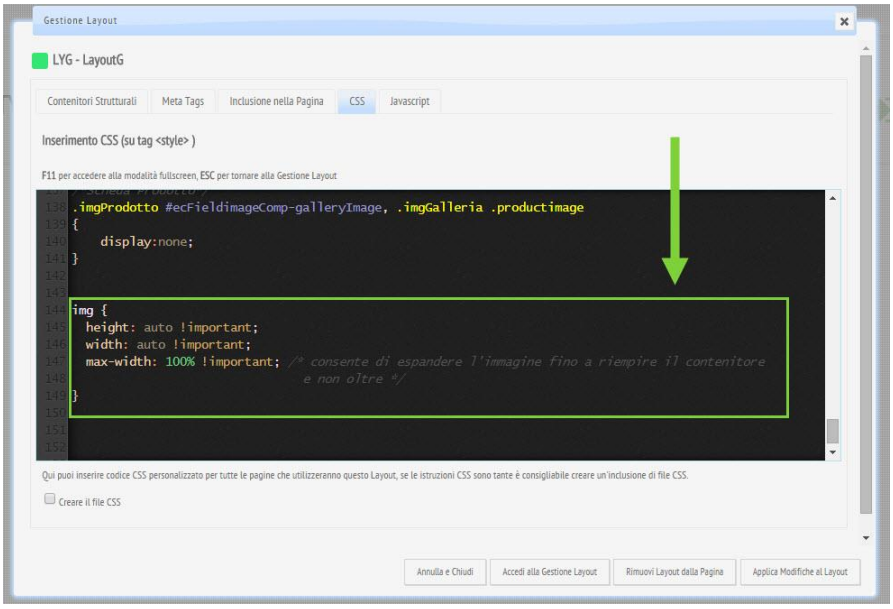
Nel primo caso sarà necessario agire, tanto per il contenitore dell'immagine quanto per l'immagine stessa, sulle rispettive proprietà presenti direttamente nello Style Editor di Passweb.

Nel secondo caso sarà invece necessario scrivere apposite regole CSS da inserire poi nella relativa sezione del layout di pagina e/o di sito.

Considerando ora che, dipendentemente dallo specifico layout che andremo poi a realizzare, il contenitore dell'immagine potrà avere dimensioni differenti, in relazione anche alla posizione della pagina in cui viene inserito, la strada più semplice da seguire è sicuramente quella che prevede, da una parte, di impostare la dimensione dello specifico contenitore dell'immagine (individuato in Passweb, dal "Componente Immagine") direttamente dallo Style Editor

dall'altra parte di assegnare invece i valori di altezza, larghezza e larghezza massima dell'immagine attraverso una specifica regola CSS da inserire nella relativa sezione del layout.

```
img {
  height: auto !important;
  width: auto !important;
  max-width: 100% !important; /* consente di espandere l'immagine fino a riempire il contenitore e non oltre */
}
```



ATTENZIONE! La stringa "!important" presente nella regola sopra evidenziata consente di assegnare una priorità maggiore al valore della relativa proprietà CSS rispetto a quello che potrebbe essergli assegnato a default (rispetto a quella che potrebbe essere, ad esempio, l'effettiva larghezza dell'immagine)

Operando in questo modo avremo quindi la certezza che restringendo/allargando la pagina si restringerà/allargherà proporzionalmente anche il contenitore dell'immagine e con esso, ovviamente, l'immagine stessa.

ATTENZIONE! l'unica accortezza da avere è quella di usare sempre immagini sufficientemente grandi e adeguate a tutte le dimensioni che il layout può raggiungere.

Se l'immagine utilizzata dovesse infatti essere troppo piccola rispetto alle dimensioni che potrà assumere all'interno del layout ad un certo punto risulterà sicuramente sgranata.

La tecnica appena analizzata ha dalla sua il fatto che è piuttosto semplice da implementare, oltre alla piena compatibilità cross browser ma, volendo essere precisi, risolve solo una parte dei problemi posti dalle immagini in un contesto responsive.

Rimane infatti aperta una seconda questione, anche piuttosto importante: se responsive design significa adattamento automatico all'ambiente d'uso del sito, allora sarebbe necessario operare in maniera tale da servire immagini ad hoc a seconda dei dispositivi e delle loro caratteristiche evitando quindi di far scaricare a chi usa uno smartphone un'immagine ad altissima risoluzione e dall'elevato peso (un'immagine di 500kb e 1300x700px è inutile se lo schermo del dispositivo non supera i 480px)

Usando la tecnica del max-width: 100% potremo adeguare l'immagine al layout, ma rimane aperto il problema dello 'spreco' di banda e quindi del 'peso' dell'immagine.

Le tecniche elaborate per risolvere questo problema sono diverse:

- l'istruzione object-fit del CSS3
- il tag picture dell'HTML5
- il tag HTML5 appoggiato da jQuery
- specifici plugin javascript
- …

Ognuna di queste soluzioni ha dei pro e dei contro ma, purtroppo, allo stato attuale nessuna di esse rappresenta ancora uno standard e, per questa ragione, non verranno approfondite nel corso di questa guida.

IMMAGINI DI SFONDO

Se per le immagini inserite direttamente nella pagina (tag img) non esiste ancora una soluzione standard che ci consenta di servire immagini diverse in relazione al dispositivo su cui la pagina web verrà effettivamente visualizzata, il discorso cambia invece per le immagini di sfondo.

In questo caso infatti essendo l'immagine di sfondo gestibile con la proprietà CSS "background-image", il problema può essere facilmente superato utilizzando le solite media-query.

Scendiamo nei dettagli tecnici della soluzione e supponiamo di dover gestire un immagine di sfondo (sfondo1920.jpg) su di un blocco di contenuti che in un tradizionale monitor pc può arrivare a raggiungere una larghezza massima di 1920px.

```
<div id="contenitore">
  [...]
</div>
```

Detto che, come al solito, il nostro contenitore di contenuti dovrà avere una larghezza in percentuale e che l'immagine che andremo ad inserire come sfondo dovrà avere una larghezza non inferiore ai 1920px (per essere sicuri di non ottenere situazioni in cui l'immagine stessa possa risultare sgranata), va anche considerato che in dispositivi di piccole dimensioni per cui l'area di visualizzazione del documento non supera in larghezza i 600px, il fatto di utilizzare un'immagine grande più del doppio non è sicuramente la soluzione migliore.

Anche se perfettamente scalata questa rimarrebbe pur sempre un'immagine di 1920px con il relativo peso ed i relativi tempi di download e di caricamento della pagina.

La soluzione ottimale in queste condizioni sarebbe quindi quella di fornire per schermi con un viewport non superiore ai 600px un'immagine diversa o, al limite, anche la stessa immagine, ma opportunamente ridimensionata a monte, in maniera tale che non superi effettivamente i 600 px del dispositivo (sfondo600.jpg).

Per ottenere questo risultato è sufficiente utilizzare il seguente codice CSS

```
#contenitore{
    background-image: url("sfondo600.jpg");
}

@media only screen and (min-width: 600px)
{
    # contenitore
    {
        background-image: url("sfondo1920.jpg");
    }
}
```

La prima delle due regole CSS imposta, in perfetto accordo con l'approccio mobile first, come immagine di sfondo di default per il nostro contenitore l'immagine piccola (sfondo600.jpg).

La media-query viene invece applicata solo per i dispositivi in cui l'area di visualizzazione del documento è superiore ai 600px e in queste condizioni sovrascrive la precedente regola permettendoci di fornire l'immagine grande (sfondo1920.jpg).

Così facendo riusciamo quindi a fornire ad ogni dispositivo un immagine adeguata ottimizzando anche il consumo di banda e le prestazioni di pagina

Se poi dovessimo decidere, addirittura, di non voler utilizzare proprio l'immagine di sfondo nei dispositivi di piccole dimensioni, dando quindi priorità alle prestazioni della pagina, dovremo modificare la nostro media query come segue

```
#contenitore{
    background: none;
}

@media only screen and (min-width: 600px)
{
    # contenitore
    {
        background-image: url("sfondo.jpg");
    }
}
```

In realtà potremmo anche essere tentati dal fatto di utilizzare una dichiarazione come display:none per nascondere l'intero blocco di contenuti su schermi di piccole dimensioni.

Pur raggiungendo il nostro scopo questa non sarebbe però la soluzione ottimale in quanto il nostro problema non è tanto quello di nascondere l'intero blocco di contenuti e con esso anche l'immagine di sfondo, ma solo quest'ultima.

Inoltre è sempre bene considerare anche **che il display:none impostato per un contenuto, pur consentendoci di nascondere quel contenuto, non evita che questo venga comunque scaricato dal browser lasciando quindi aperto il problema relativo all'ottimizzazione delle prestazioni**.

SISTEMI DI NAVIGAZIONE

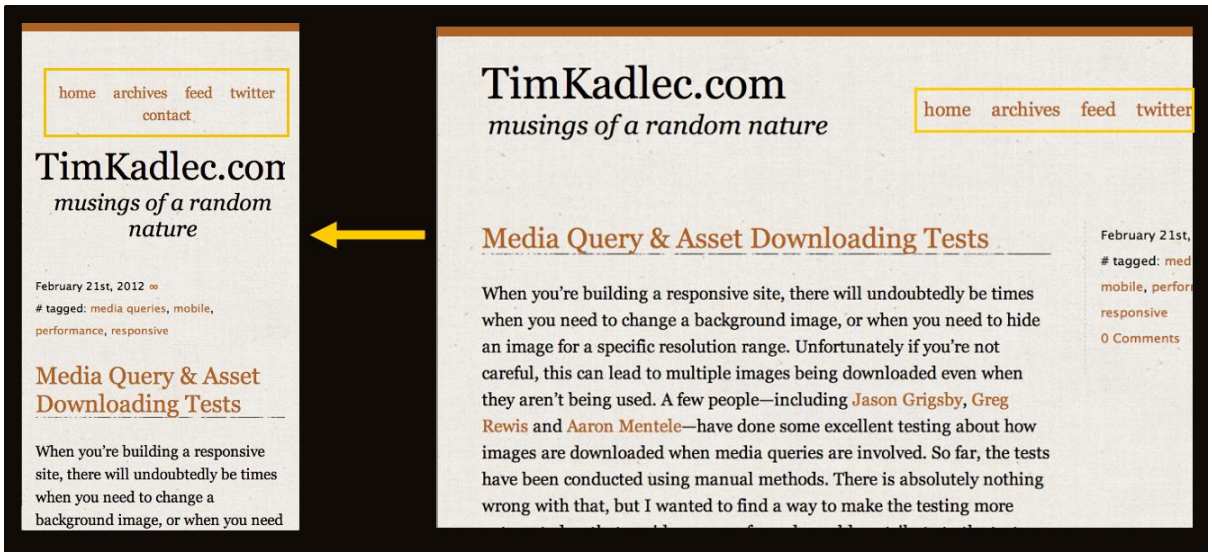
Nel contesto del layout di un sito responsivo, un altro elemento piuttosto critico da considerare è, indubbiamente, quello che riguarda il sistema di navigazione.

In questo senso esistono diversi pattern da poter adottare per gestire questo tipo di problematica. Nel seguito esamineremo rapidamente i 4 più utilizzati avendo comunque ben chiaro che l'obiettivo primario è sempre quello di ottimizzare lo spazio disponibile dando sempre rilievo ai contenuti principali del sito.

DO NOTHING APPROACH

Questa è sicuramente la soluzione più semplice da implementare in quanto, di fatto, non prevede particolari accorgimenti oltre a quelli di cui abbiamo parlato fino a questo momento e che sono necessari, in generale, per realizzare un sito responsivo.

Il menu di navigazione va collocato, generalmente, in testata alla pagina web, deve avere una larghezza in percentuale e un'altezza che si adatti automaticamente al contenuto



In queste condizioni diminuendo l'area di visualizzazione del documento quello che potrà succedere è che le voci di menu di primo livello, inizialmente disposte su di un'unica riga shiftino una sotto l'altra disponendosi dunque su due o più righe (come evidenziato in figura).

PRO

- Semplice da realizzare – Può essere implementato utilizzando i normali componenti presenti in Passweb
- Non richiede l'utilizzo di plugin esterni

CONTRO

- Altezza complessiva variabile – Considerando che nei dispositivi di piccole dimensioni si tende a linearizzare il layout sviluppando il sito in verticale, l'altezza degli elementi diventa un aspetto fondamentale in relazione all'immediata visualizzazione o meno di determinati contenuti. Il fatto che un menu possa disporsi su due o più righe, anziché su una soltanto, potrebbe far scorrere verso il basso e non rendere quindi immediatamente visibili, altri contenuti importanti della pagina
- Difficilmente scalabile – Aggiungere voci al menu può modificare sensibilmente l'altezza complessiva del componente
- Voci difficilmente cliccabili – Cercando di avvicinare le voci per far rientrare il menu all'intero di una singola riga potrebbe rendere difficile selezionare con un dito la pagina di destinazione.

COME REALIZZARLO IN PASSWEB

E' sufficiente utilizzare la normale componentistica di Passweb (Componente Menu, Componente Paragrafo con link ecc…) prestando particolare attenzione al fatto di utilizzare delle dimensioni in percentuale

MENU NEL FOOTER

Questo pattern prevede di inserire il menu di navigazione nel piede della pagina web lasciando in testata una semplice ancora che punta al piede stesso consentendo quindi all'utente, quando necessario, di far scorrere automaticamente la pagina fino a visualizzare il menu di navigazione.

PRO

- Semplice da realizzare – Può essere implementato utilizzando i normali componenti presenti in Passweb
- Non richiede l'utilizzo di plugin esterni
- Facilmente scalabile

CONTRO

- Lo scorrimento automatico della pagina verso il basso può disorientare l'utente
- Il tipo di interazione utente non è tra le più eleganti

COME REALIZZARLO IN PASSWEB

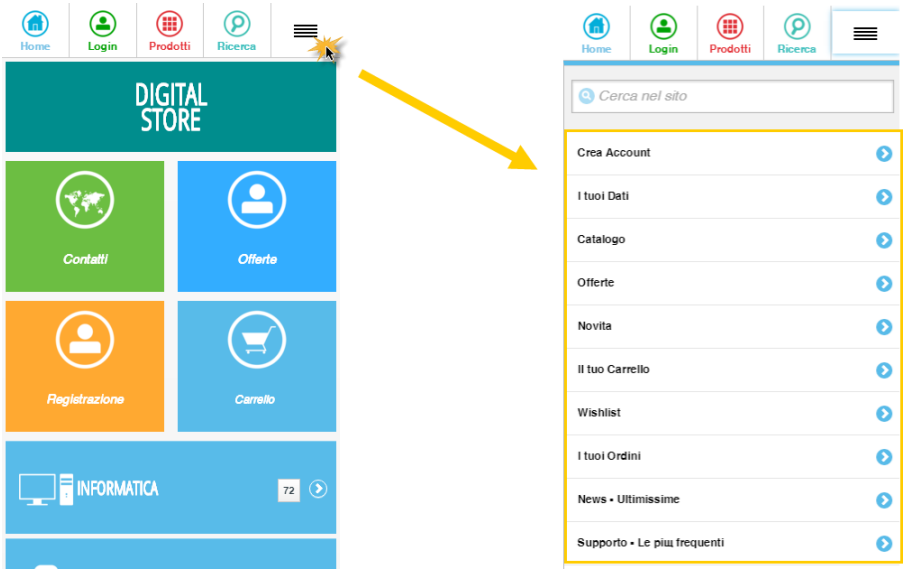
La soluzione ottimale è forse quella che prevede di utilizzare un normale Componente Paragrafo o HTML per realizzare l'ancora da porre in testata alla pagina e un normale Componente Menu, sviluppato in verticale, da posizionare nel piede.

Entrambi questi elementi possono essere nascosti, attraverso apposite media query in cui utilizzare il display:none, e visualizzati solo nel caso in cui la pagina venga visualizzata su schermi di piccole dimensioni.

A differenza infatti di quanto avviene per le immagini, il fatto di nascondere semplici elementi testuali, come possono essere un'ancora o un menu, non causa sostanziali peggioramenti nelle prestazioni del sito.

TOGGLE MENU

Questo approccio è simile a quello appena considerato ma anziché far scorrere automaticamente la pagina verso il basso per visualizzare il menu, quest'ultimo dovrà aprirsi in corrispondenza del pulsante di apertura sovrapponendosi agli altri contenuti del sito o, al massimo, facendoli scorrere verso il basso.



PRO

- Semplice da realizzare – Può essere implementato utilizzando i normali componenti presenti in Passweb
- Facilmente scalabile

CONTRO

- Il menu può coprire interamente il resto dei contenuti della pagina
- Utilizzo di javascript per gestire le animazioni di apertura/chiusura del menu

COME REALIZZARLO IN PASSWEB

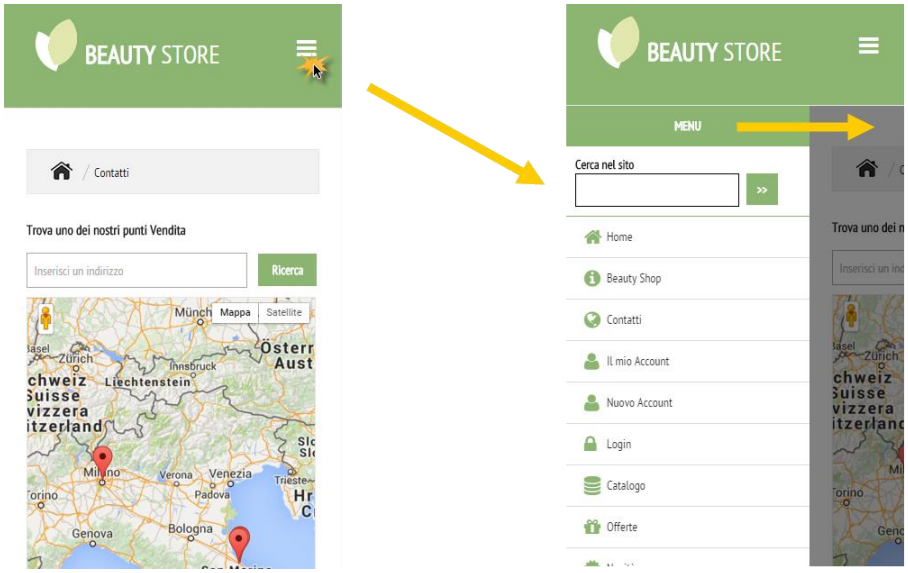
In Passweb questa soluzione può essere implementata in diversi modi. Il più semplice è quello che prevede di utilizzare un componente "Accordion" all'interno del quale inserire un Componente Menu, gestendo poi un piccolo script in maniera tale da visualizzare, inizialmente, l'unico pannello dell'accordion chiuso.

In alternativa è possibile utilizzare anche un componente HTML assieme ad uno dei tantissimi plugin disponibili in rete per visualizzare / nascondere elementi al click su di un pulsante.

OFF CANVAS MENU

Si tratta della soluzione ad oggi più utilizzata nei siti responsivi e, forse, la più elegante. In questo scenario il menu viene inizialmente nascosto, come nel pattern precedente, ma, questa volta, collocandolo in un'area esterna alla finestra del browser.

Quando il menu viene visualizzato, cliccando per questo su di un apposito pulsante, il corpo principale del documento scorre interamente verso destra (sinistra) per lasciare spazio proprio al menu



PRO

- Interazione utente ormai standard sui dispositivi mobile
- Facilmente scalabile una volta implementato

CONTRO

- Non realizzabile nativamente in Passweb – Richiede necessariamente l'uso di javascript e/o di un apposito plugin

COME REALIZZARLO IN PASSWEB

Come detto Passweb non dispone di un componente che permetta di realizzare nativamente questo tipo di interazione, per cui per realizzare un menu di questo tipo è necessario ricorrere ad uno dei tanti plugin disponibili in rete (es. http://mmenu.frebsite.nl/) o, nel caso in cui si decida di utilizzare un framework CSS responsivo, al relativo componente messo eventualmente a disposizione dal framework stesso.

FRAMEWORK E LIBRERIE DI SUPPORTO

Se, da una parte, tutti i concetti esaminati fino a questo momento (media query, break-point, griglie fluide ecc…) possono tranquillamente essere implementati "manualmente" senza dover per forza di cose ricorrere a strumenti esterni, dall'altra parte è comunque vero che un concetto fondamentale, non solo del responsive design, ma di tutta l'informatica è quello di **riutilizzare il più possibile quanto di buono è già stato fatto.**

In questo senso, in ambito responsivo, non mancano di certo le opportunità. Innanzitutto però è importante distinguere due casi e cioè se il sito è nuovo o va comunque rifatto da zero, o se la necessità è invece quella di adattare un'qualcosa di già esistente rendendolo responsive.

Nel primo caso non c'è che l'imbarazzo della scelta, esistono orami tantissimi framework responsivi completi, ben fatti ed efficaci, che implementano già tutte regole viste e molto di più.

Volendo citarne alcuni tra i più famosi ed utilizzati potremmo indicare sicuramente **Boostrap**, ma anche **Foundation**, **Boilerplate** o **Uikit**, tutti framework questi che, oltre a dare un valido supporto per la realizzazione di siti responsivi, ci permettono anche di utilizzare un'ampia libreria di componenti ed utility grazie alle quali poter ottenere ottimi risultati con poco sforzo.

Se invece dobbiamo rendere responsivo un sito web già esistente, l'introduzione di un framework potrebbe essere una soluzione troppo invasiva e il discorso diventerebbe quindi più complicato. In queste condizioni la soluzione migliore sarebbe indubbiamente quella di sporcarsi le mani andando a modificare ed ottimizzare il codice CSS esistente anche se, come inizialmente evidenziato, riuscire a rendere responsivo un sito pensato, progettato e implementato non con questa finalità diventa un'impresa veramente ardua.

L'obbiettivo di questa guida non è comunque quello di spiegare come poter adattare un sito già esistente rendendolo responsivo, quanto più esattamente quello di fornire gli elementi base per riuscire a poi realizzare, **partendo da zero**, un sito Passweb perfettamente responsivo potendosi avvalere, in questo processo, di un valido supporto come quello rappresentato dal framework **Uikit.css** che ben si presta ad essere integrato all'interno di Passweb.

UIKIT.CSS

La parola 'framework' associata al web design indica, in genere, **un insieme di file CSS da associare a strutture standardizzate di codice HTML** per velocizzare e supportare lo sviluppo del sito, la creazione e l'implementazione dei vari elementi dell'interfaccia, specie nelle fasi iniziali del progetto.

Il fine ultimo di un framework, in fin dei conti, è proprio quello di costituire la base dello sviluppo a partire dalla quale si potranno poi ottenere prodotti finiti (siti web nel nostro caso) completamente diversi in fatto di scelte stilistiche ed estetiche, senza dover ogni volta "reinventare" la ruota ma, al contrario, potendosi avvalere di un'ampia base di componenti perfettamente riutilizzabili.

Inizialmente i framework CSS sono nati con una finalità ben precisa, vale a dire la strutturazione di una griglia e, conseguentemente, del solo layout del sito; con l'andare del tempo, e soprattutto con l'avvento del responsive design, sono state sviluppate soluzioni sempre più complesse ed evolute fino alla realizzazione di framework che sarebbe piuttosto riduttivo considerare come destinati solamente alla creazione e gestione di semplici griglie.

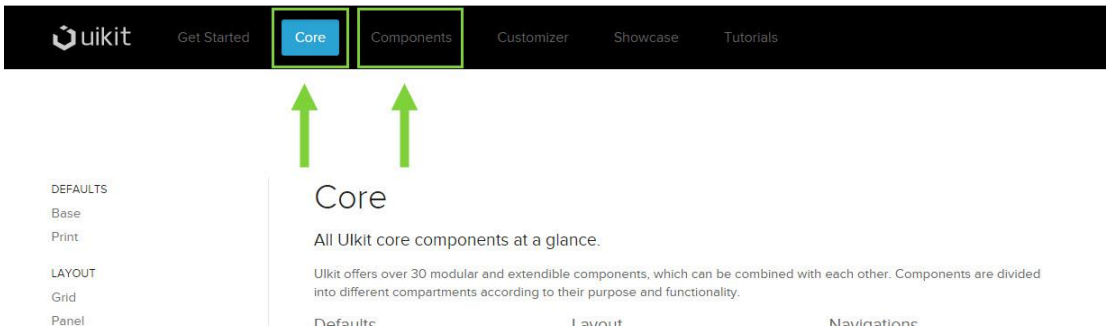
Davanti a prodotti di questo tipo si preferisce parlare infatti di "front end framework" o "UI framework" e si va ben oltre la semplice griglia perché consentono di configurare rapidamente l'intera interfaccia di un sito o di un'applicazione, dalla tipografia alla navigazione fino alle più svariate interazioni javascript.

Allo stato attuale, parlando di framework per la creazione dell'interfaccia utente, si intende un sistema completo che abbia almeno le seguenti caratteristiche:

- deve fornire le basi per la costruzione di solidi layout a griglia
- deve offrire un CSS strutturato e modulare per la formattazione dei principali elementi dell'interfaccia attraverso l'utilizzo di classi standardizzate nel codice HTML
- deve risparmiare il più possibile allo sviluppatore la fatica di risolvere problemi di incompatibilità tra i browser
- dovrebbe fornire una serie di plugin Javascript per widget e componenti di interfaccia come slideshow, tooltip, box modali ecc…
- **deve essere perfettamente responsivo**

Uikit, come indicato nella pagina del progetto (http://getuikit.com/index.html), è "un front-end framework leggero e modulare che consente di sviluppare rapidamente interfacce web", che soddisfa appieno tutte le caratteristiche appena elencate e che si propone, grazie alla sua elevata modularità e alla sua facilità di utilizzo, come ottima alternativa ai più conosciuti (e per certi versi anche più pesanti) Bootstrap o Foundation.

Il set di componenti e di funzionalità messe a disposizione da questo framework può essere suddiviso in due distinti ambiti: CORE e COMPONENTS



UIKIT – CORE

E' il cuore di tutto il progetto e, come tale, comprende tutto il set dei componenti di base del framework.

Mette a disposizione dell'utente più di 30 diversi componenti perfettamente combinabili l'uno con l'altro e suddivisi in 6 diverse categorie a seconda di quello che è il loro scopo e di quelle che sono le loro specifiche funzionalità.

- **Defaults:** i componenti presenti all'interno di questa sezione consentono di normalizzare i vari elementi HTML in maniera tale da poter gestire al meglio eventuali problemi legati alla compatibilità cross browser. Sono presenti anche alcuni stili di base per i principali tag HTML

- **Layout:** all'interno di questa sezione possiamo trovare tutti i componenti necessari per definire e creare il layout del sito, partendo da una griglia completamente responsiva passando attraverso un semplice sistema di pannelli per finire con tutta una serie di utilità mediante le quali poter centrare e allineare i vari elementi della pagina web, nasconderli o visualizzarli in base alle dimensioni del viewport ecc…

- **Navigations:** all'interno di questa sezione possiamo trovare tutti gli elementi utilizzati da uikit per gestire i diversi sistemi di navigazione, dalle navigation bar, alle side bar, alle briciole di pane ecc…

- **Elements:** all'interno di questa sezione vengono definiti gli stili per i componenti HTML di base come liste, form, tabelle ecc…

- **Common:** all'interno di questa sezione troviamo gli elementi più comunemente utilizzati nelle pagine di un sito web vale a dire pulsanti, icone, overlays, animazioni, alert ecc…

- **Javascript:** i componenti presenti all'interno di questa sezione si basano sull'utilizzo di javascript e necessitano quindi dell'utilizzo delle apposite librerie del framework. E' in questa sezione che possiamo trovare, ad esempio, tutto ciò che serve per poter attivare e gestire un menu Off-canvas (per maggiori informazioni relativamente a questa particolare tipologia di menu si veda anche il precedente capitolo "*Sistemi di Navigazione*" di questa guida)

ATTENZIONE! L'intero set di componenti base è racchiuso all'interno di un unico file uikit.css disponibile sia in versione "normale" che minificata.

UIKIT – COMPONENTS

Uikit mette a disposizione dell'utente anche tutta una serie di componenti avanzati non inclusi nella parte core del framework.

Si tratta di componenti come Slider, Slideshow, Date Picker, Parallasse ecc… non strettamente necessari alla realizzazione di un sito web e che possono quindi essere o meno utilizzati a seconda dello specifico progetto che si intende realizzare.

ATTENZIONE! A differenza della parte core, racchiusa in un unico file, ciascuno dei componenti presenti all'interno di questa sezione ha una sua specifica libreria CSS e/o javascript che dovranno quindi essere caricate all'interno del sito in aggiunta ai file uikit.css e uikit.js (parte core del framework)

Anche se piuttosto ovvia, una cosa importante da mettere in evidenza è che per lo specifico progetto che stiamo realizzando potrebbe anche non essere necessario utilizzare tutti i componenti e gli elementi messi a disposizione dal framework.

In questo senso l'elevata modularità di uikit torna particolarmente utile in quanto ci permette di non dover necessariamente appesantire la pagina web con elementi CSS o Javascript non strettamente necessari.

Nel caso in cui dovessimo decidere di non utilizzare all'interno del nostro progetto alcuni dei componenti aggiuntivi messi a disposizione dal framework, sarà sufficiente, ovviamente, non caricare all'interno della pagina le librerie CSS e/o Javascript relative a quei componenti che effettivamente non vengono utilizzati.

BREAKPOINTS

Una delle caratteristiche principali di un front-end framework è, come precedentemente evidenziato, quella di essere perfettamente responsivo.

Uikit soddisfa appieno questo requisito mettendo a disposizione dell'utente tutta una serie di classi responsive (e relative media query) che consentono di adattare perfettamente la visualizzazione dei contenuti del sito a differenti dimensioni del viewport.

I breakpoint utilizzati per gestire le diverse visualizzazioni sono impostati su 5 diverse dimensioni di base identificate come: **Mini, Small, Medium, Large e Xlarge**.

A ciascuna di queste dimensioni corrisponde una ben precisa larghezza in pixel del viewport (e corrispondentemente una certa categoria di dispositivi) come indicato nella sottostante tabella:

SIZE	BREAKPOINTS – DIMENSIONE VIEWPORT	DEVICE
Mini	Da **0px** a **479px**	Smartphone in modalità portrait
Small	Da **480px** a **767px**	Smartphone in modalità landscape
Medium	Da **768px** a **959px**	Tablet in modalità portrait
Large	Da **960px** a **1199px**	Tablet in modalità landscape, Desktop standard
Xlarge	Maggiore o uguale a **1200px**	Desktop di grandi dimensioni

ATTENZIONE! I valori dei breakpoint presenti in tabella sono quelli presenti a default nel framework.

In ogni caso è sempre possibile modificare tali valori secondo le specifiche esigenze del caso intervenendo direttamente all'interno del file uikit.css oppure, in maniera molto più semplice, impostando i valori desiderati, ancora prima di effettuare il download del framework, mediante l'apposito **Customizer** (per maggiori informazioni in merito alla personalizzazione del framework si veda anche la sezione *"Temi Personalizzati"* di questa guida)

Concludiamo questa panoramica generale del framework con altre due considerazioni piuttosto importanti:

- Uikit utilizza **box-sizing** impostata sul valore **border-box** in maniera tale che le larghezze dei propri componenti rimangano consistenti indipendentemente dal padding che può essere loro applicato. E' quindi possibile aggiungere padding o bordi ai componenti del framework senza che questo vada a compromettere il layout della pagina

  Per maggiori informazioni relativamente all'importanza del box-sizing si veda anche il precedente capitolo *"Il sistema a griglie"* di questo manuale
- Considerando l'ottimo funzionamento del framework in relazione alla definizione e creazione del layout della pagina, è bene non apportare modifiche ai suoi componenti principali. Nel caso in cui sia necessario personalizzare alcuni elementi si consiglia quindi di inserire questi stessi elementi all'interno di un componente di base, come può essere una griglia o un pannello e stilizzare poi questi elementi piuttosto che la griglia o il pannello stesso.

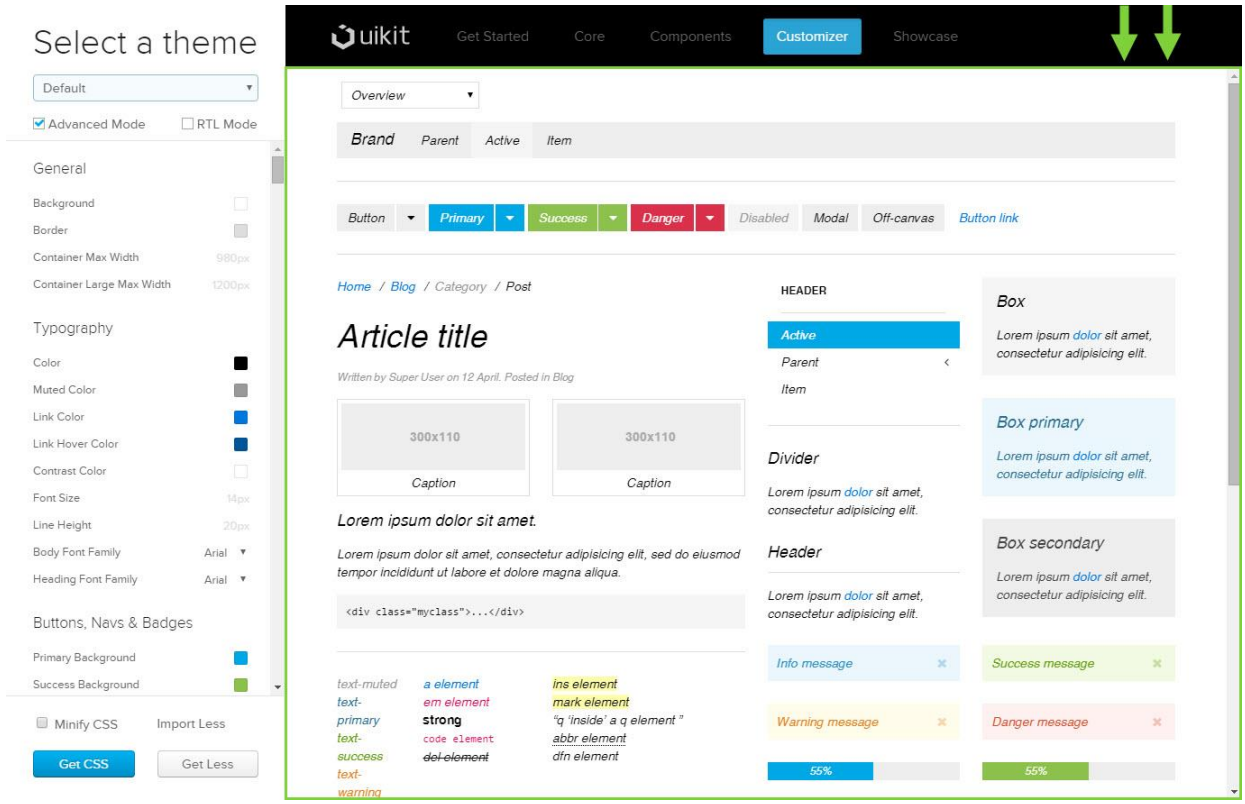
Nella realizzazione di un sito Passweb responsivo utilizzeremo proprio questo approccio: ci preoccuperemo quindi di strutturare il layout della pagina utilizzando i componenti di base del framework e all'interno di questi componenti andremo poi ad inserire la "normale" componentistica Passweb che potrà poi essere stilizzata mediante lo Style Editor così come avviene in generale per tutti i siti Passweb.

TEMI PERSONALIZZATI

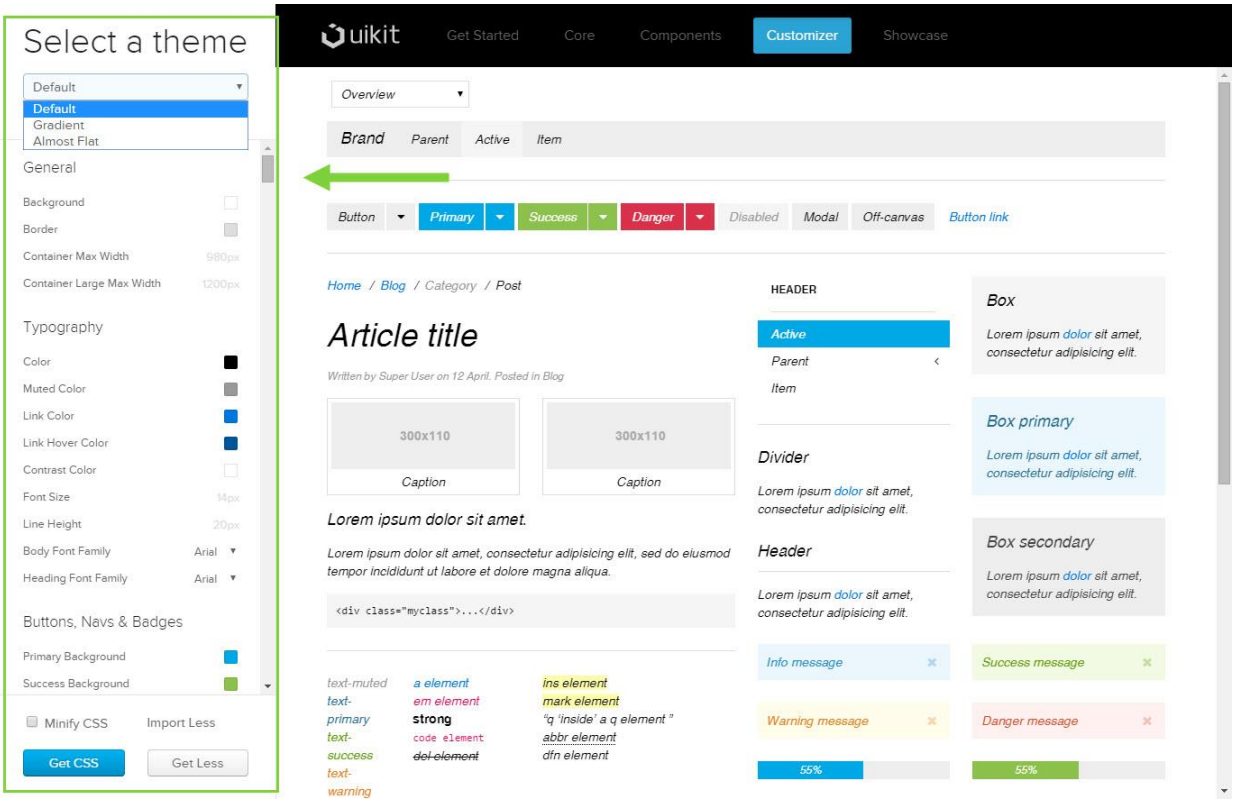
Un altro aspetto interessante di uikit è rappresentato dalla possibilità di creare facilmente temi grafici personalizzati da poter utilizzare in alternativa a quelli proposti a default dal framework stesso.

Se, ad esempio, dovessimo decidere di utilizzare una tipografia differente da quella proposta a default da uikit o se volessimo stilizzare i pulsanti del sito a nostro piacimento potremmo utilizzare il "customizzatore" presente direttamente sul sito di riferimento del progetto http://getuikit.com/docs/customizer.html

Nella parte destra della pagina vengono visualizzati tutti gli elementi di base del framework con la relativa formattazione grafica.



Nella parte sinistra è invece possibile variare il tema di default oltre che intervenire, mediante una semplice interfaccia grafica (simile allo style editor di Passweb) sui valori di tutte le proprietà CSS (compresi anche i breakpoint) utilizzate dal framework.



Una volta apportate le modifiche desiderate e creato, di fatto, il proprio tema grafico sarà sufficiente cliccare sul pulsante "**Get CSS**" per generare e downlodare il file uikit.css personalizzato secondo le proprie specifiche esigenze e pronto quindi per essere utilizzato all'interno del sito al posto di quello classico.

Ovviamente questo tipo di modifiche potrebbero essere effettuate anche andando ad editare manualmente il file uikit.css, anzi in questo modo avremmo sicuramente una maggiore libertà di azione potendo ad esempio:

- eliminare quelle parti di codice non utilizzate all'interno del sito alleggerendo ulteriormente il peso complessivo della pagina

- aggiungere regole e/o proprietà CSS non presenti in maniera nativa all'interno del framework ma utili per lo specifico progetto che stiamo realizzando
- …

Intervenire manualmente sul contenuto delle librerie CSS e/o javascript richiede però una certa conoscenza tecnica e potrebbe compromettere il corretto funzionamento di tutto il sito soprattutto se queste modifiche dovessero essere effettuate sulla parte core del framework (e quindi sul file uikit.css)

ATTENZIONE! Modificare il file uikit.css potrebbe compromettere il corretto funzionamento di tutto il framework.

Nel caso in cui non si dovesse essere assolutamente certi delle operazioni effettuate è consigliabile quindi non editare mai manualmente il contenuto delle librerie messe a disposizione dal framework e di effettuare invece le modifiche desiderate ricorrendo alle specifiche interfacce grafiche.

In questo senso è anche bene sottolineare come **il fatto di utilizzare uikit all'interno del proprio sito Passweb, non compromette in alcun modo la possibilità di andare poi a stilizzare i vari elementi della pagina secondo quelle che sono le normali modalità di lavoro tipiche di Passweb.**

In altri termini se anche l'utilizzo del framework dovesse assegnare a tutti i link del sito il colore rosso oltre che un font di 12px, nulla ci impedirà comunque di poter poi intervenire mediante lo style editor di Passweb per formattare diversamente uno specifico link assegnandogli, ad esempio, un colore verde e un font di 16px.

In generale infatti le proprietà CSS impostate mediante Style Editor hanno sempre una priorità maggiore (essendo più interne) rispetto ai valori impostati per queste stesse proprietà in una qualsiasi libreria caricata nel rispettivo layout di pagina e/o di sito.

DOWNLOAD DEL FRAMEWORK

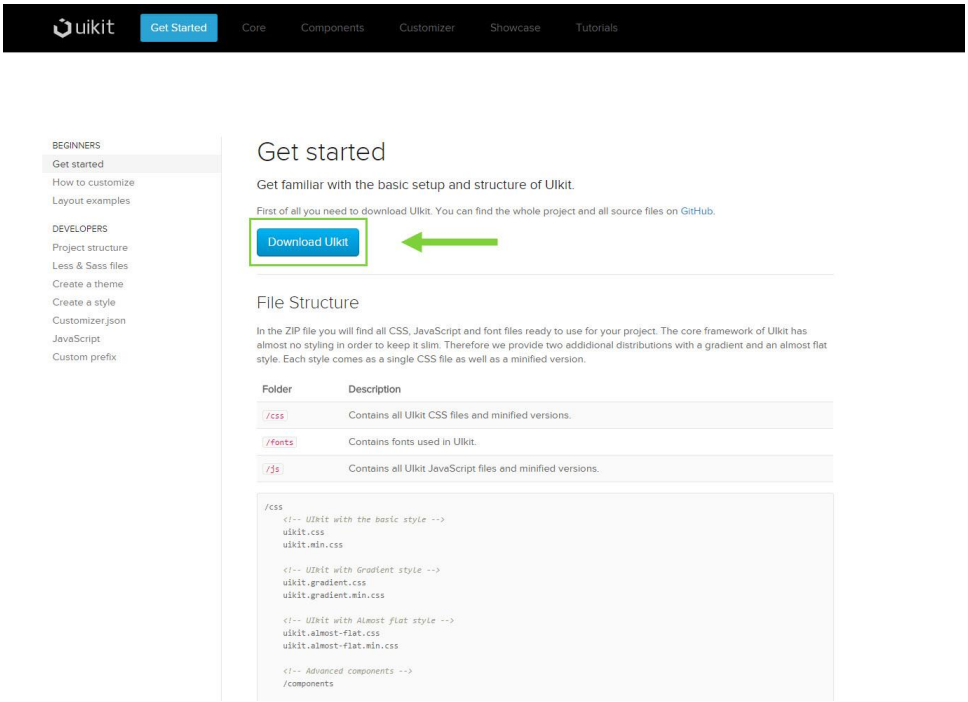
Il primo passo da fare per poter utilizzare uikit all'interno del proprio sito Passweb è, ovviamente, quello di scaricare il framework.

Abbiamo tre opzioni per il download:

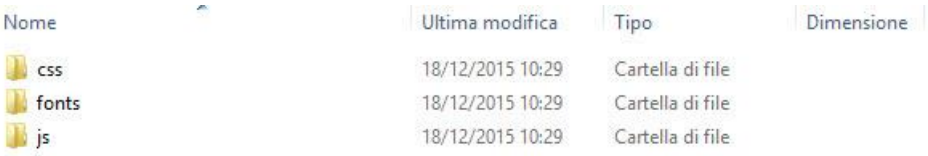
1. Il download della versione completa del framework
2. Il download di una versione CSS personalizzata
3. Il download di una versione basata su Less e Sass

Detto che la terza opzione non verrà presa in considerazione all'interno di questa guida in conseguenza del fatto che in Passweb si lavora con CSS già compilati e pronti per l'uso, andiamo invece ad analizzare le prime due opzioni.

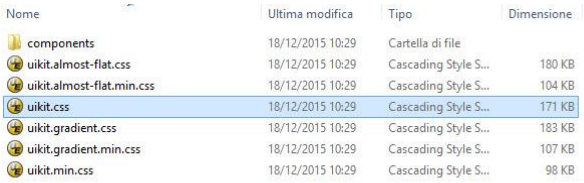
Nel primo caso, per effettuare il download della versione completa del framework è sufficiente portarsi sul sito web del progetto nella sezione "**Get Started**" (http://getuikit.com/docs/documentation_get-started.html) e cliccare sul pulsante **Download Uikit**



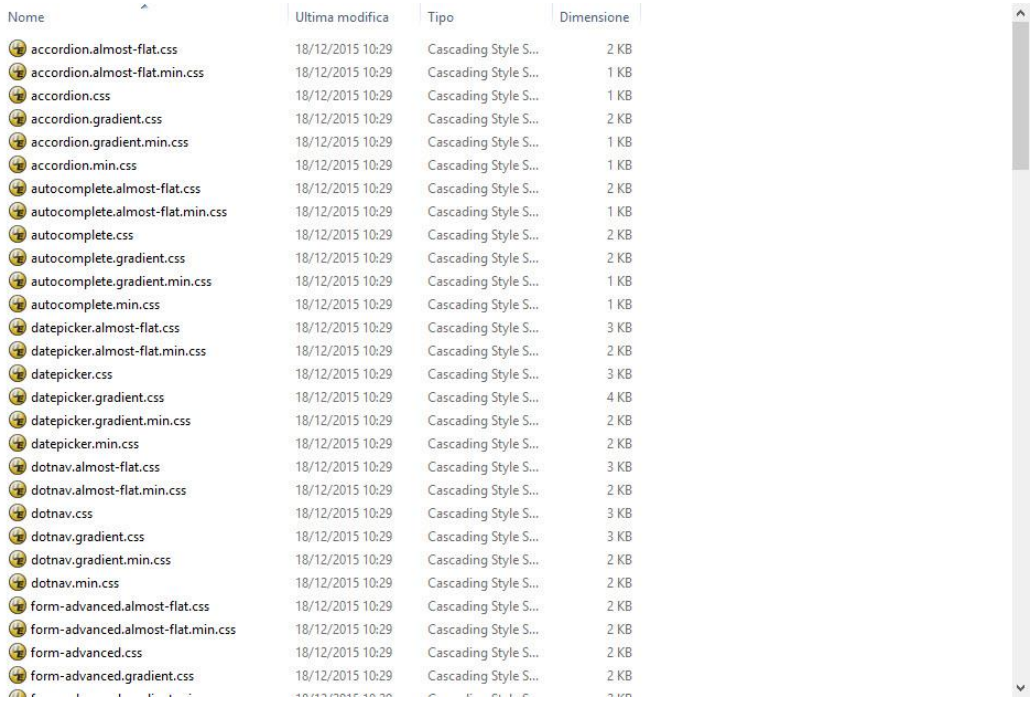
Una volta scompattato il file .zip oggetto del download ci ritroveremo con questa struttura di cartelle:



1. **css:** contiene tutti i file CSS del framework, dal core (ukit.css) alle librerie necessarie per il corretto funzionamento di ogni singolo componente avanzato. Ogni libreria è presente nella sua versione estesa (ukit.css) e minificata (uikit.min.css) oltre che nei due temi grafici disponibili a default "**almost-flat**" (uikit.almost-flat.css) e "**gradient**" (uikit.gradient.css).

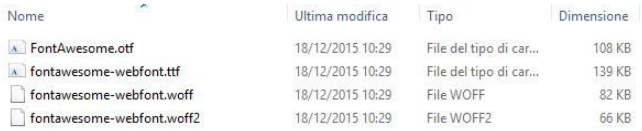


ATTENZIONE! Le librerie CSS necessarie per il corretto funzionamento dei componenti avanzati si trovano all'interno della cartella "**components**"



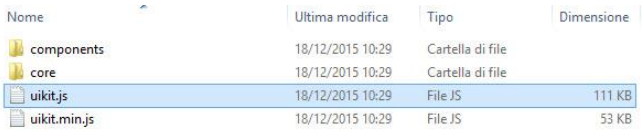
Nome	Ultima modifica	Tipo	Dimensione
accordion.almost-flat.css	18/12/2015 10:29	Cascading Style S...	2 KB
accordion.almost-flat.min.css	18/12/2015 10:29	Cascading Style S...	1 KB
accordion.css	18/12/2015 10:29	Cascading Style S...	1 KB
accordion.gradient.css	18/12/2015 10:29	Cascading Style S...	2 KB
accordion.gradient.min.css	18/12/2015 10:29	Cascading Style S...	1 KB
accordion.min.css	18/12/2015 10:29	Cascading Style S...	1 KB
autocomplete.almost-flat.css	18/12/2015 10:29	Cascading Style S...	2 KB
autocomplete.almost-flat.min.css	18/12/2015 10:29	Cascading Style S...	1 KB
autocomplete.css	18/12/2015 10:29	Cascading Style S...	2 KB
autocomplete.gradient.css	18/12/2015 10:29	Cascading Style S...	2 KB
autocomplete.gradient.min.css	18/12/2015 10:29	Cascading Style S...	1 KB
autocomplete.min.css	18/12/2015 10:29	Cascading Style S...	1 KB
datepicker.almost-flat.css	18/12/2015 10:29	Cascading Style S...	3 KB
datepicker.almost-flat.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
datepicker.css	18/12/2015 10:29	Cascading Style S...	3 KB
datepicker.gradient.css	18/12/2015 10:29	Cascading Style S...	4 KB
datepicker.gradient.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
datepicker.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
dotnav.almost-flat.css	18/12/2015 10:29	Cascading Style S...	3 KB
dotnav.almost-flat.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
dotnav.css	18/12/2015 10:29	Cascading Style S...	3 KB
dotnav.gradient.css	18/12/2015 10:29	Cascading Style S...	3 KB
dotnav.gradient.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
dotnav.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
form-advanced.almost-flat.css	18/12/2015 10:29	Cascading Style S...	2 KB
form-advanced.almost-flat.min.css	18/12/2015 10:29	Cascading Style S...	2 KB
form-advanced.css	18/12/2015 10:29	Cascading Style S...	2 KB
form-advanced.gradient.css	18/12/2015 10:29	Cascading Style S...	2 KB

2. **fonts:** contiene i file di Font Awesome utilizzati da uikit per gestire le icone standard



Nome	Ultima modifica	Tipo	Dimensione
FontAwesome.otf	18/12/2015 10:29	File del tipo di car...	108 KB
fontawesome-webfont.ttf	18/12/2015 10:29	File del tipo di car...	139 KB
fontawesome-webfont.woff	18/12/2015 10:29	File WOFF	82 KB
fontawesome-webfont.woff2	18/12/2015 10:29	File WOFF2	66 KB

3. **js:** contiene tutti i file javascript necessari per il corretto funzionamento del framework dal core (ukit.js) alle librerie necessarie per il corretto funzionamento di ogni singolo componente. Ogni libreria è presente nella sua versione estesa (ukit.js) e minificata (uikit.min.js)



Nome	Ultima modifica	Tipo	Dimensione
components	18/12/2015 10:29	Cartella di file	
core	18/12/2015 10:29	Cartella di file	
uikit.js	18/12/2015 10:29	File JS	111 KB
uikit.min.js	18/12/2015 10:29	File JS	53 KB

ATTENZIONE! Le librerie js necessarie per il corretto funzionamento dei componenti di base si trovano all'interno della cartella **core**. All'interno della cartella **components** troviamo invece le librerie necessarie per il corretto funzionamento dei componenti avanzati.

Per quel che riguarda invece la seconda opzione, nel caso in cui si dovesse decidere di personalizzare alcune proprietà di base del framework (colore dei pulsanti, dimensione del font, valori dei breakpoint ecc…) è possibile agire mediante l'apposito Customizer e scaricare poi la versione core del framework (uikit.css) opportunamente personalizzata.

ATTENZIONE! La personalizzazione riguarda solo i contenuti del file uikit.css. Per il corretto funzionamento dei componenti utilizzati all'interno del sito potrebbe quindi essere necessario caricare comunque anche alcune delle librerie presenti nella versione completa del framework.

STRUTTURA DEI FILE CSS

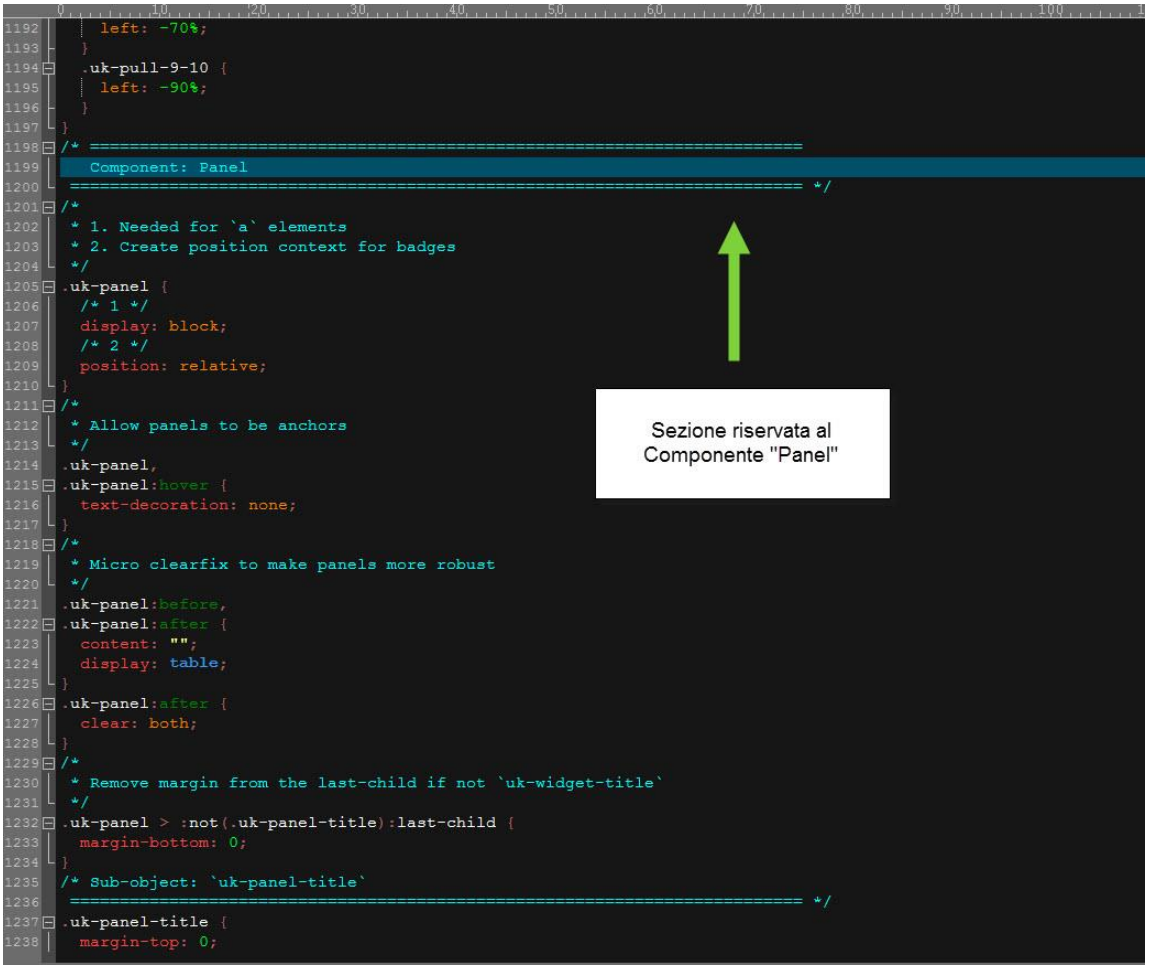
Prima di passare ad esaminare più nel dettaglio come poter effettivamente utilizzare all'interno del nostro progetto le varie librerie messe a disposizione dal framework, è bene dare uno sguardo anche alla struttura dei file CSS in maniera tale da poter intervenire con cognizione di causa qualora dovesse essere necessario apportare loro delle modifiche, o qualora si volesse, molto più semplicemente, individuare all'interno dei vari file tutte le proprietà relative ad uno specifico componente.

In questo senso possiamo evidenziare innanzitutto che, per evitare possibili conflitti con altre elementi o componenti normalmente utilizzati all'interno del sito (come possono essere i componenti Passweb con i relativi stili), tutte le classi gestite da uikit hanno come prefisso la stringa "**uk-**"

Esempio → uk-grid è la classe di base utilizzata per stilizzare il componente grid

All'interno di un file CSS ogni componente ha una sua specifica sezione (facilmente individuabile grazie ad apposite righe di commenti) dove vengono definite le classi mediante le quali poter gestire:

1. le proprietà generali del componente



2. le proprietà dei suoi sotto componenti, i cosiddetti sub-object. Sono elementi interni al componente stesso necessari per poterne garantire il corretto funzionamento (es. il pulsante di chiusura di un alert-box)

```
  clear: both;
}
/*
 * Remove margin from the last-child if not `uk-widget-title`
 */
.uk-panel > :not(.uk-panel-title):last-child {
  margin-bottom: 0;
}
/* Sub-object: `uk-panel-title`
 ========================================================================== */
.uk-panel-title {
  margin-top: 0;
  margin-bottom: 15px;
  font-size: 18px;
  line-height: 24px;
  font-weight: normal;
  text-transform: none;
  color: #444444;
}
/* Sub-object: `uk-panel-badge`
 ========================================================================== */
.uk-panel-badge {
  position: absolute;
  top: 0;
  right: 0;
  z-index: 1;
}
```

Formattazione del Sotto-Componente "Titolo del Pannello"

3. le proprietà dei suoi Modificatori (Modifier) ossia apposite classi che consentono di alterare e definire nel dettaglio lo stile dello specifico componente e dei suoi sotto componenti

```
  color: #444444;
}
.uk-panel-box-secondary-hover:hover {
  color: #444444;
}
.uk-panel-box-secondary .uk-panel-title {
  color: #444444;
}
/* Modifier: `uk-panel-hover`
 ========================================================================== */
.uk-panel-hover {
  padding: 15px;
  color: #444444;
}
.uk-panel-hover:hover {
  background: #f5f5f5;
  color: #444444;
}
.uk-panel-hover .uk-panel-badge {
  top: 10px;
  right: 10px;
}
.uk-panel-hover > .uk-panel-teaser {
  margin-top: -15px;
  margin-left: -15px;
  margin-right: -15px;
}
/* Modifier: `uk-panel-header`
 ========================================================================== */
.uk-panel-header .uk-panel-title {
  padding-bottom: 10px;
  border-bottom: 1px solid #dddddd;
  color: #444444;
}
```

Modificatore Pannello
Formattazione del pannello al passaggio del mouse

COME UTILIZZARLO

Arrivati a questo punto della guida abbiamo tutti i concetti e gli elementi necessari per poter effettivamente applicare al nostro progetto un framework come uikit e per iniziare quindi a realizzare il nostro primo sito responsivo.

Concettualmente la procedura da eseguire è estremamente semplice. Si tratta infatti di:

1. **Caricare tra le risorse del sito tutte le librerie e i file necessari per il corretto funzionamento del framework (css, js e file dei font)**

   E' consigliabile, anche solo per una pura questione di ordine, mantenere la stessa struttura di cartelle presente all'interno del file .zip ottenuto dal download del framework.

   ATTENZIONE! Il fatto di caricare tutte le librerie del framework tra le risorse, non significa che debbano necessariamente essere tutte utilizzate all'interno del sito.

2. **Inserire nella sezione < head > di ogni singola pagina i collegamenti alle librerie css e js relative ai soli componenti del framework effettivamente utilizzati in quella stessa pagina.**

   In questo senso è bene ricordare anche che i componenti javascript di uikit utilizzano jQuery per cui sarà necessario caricare all'interno del proprio sito anche questa libreria.

   In generale dovremo quindi verificare che nella sezione < head > di ogni pagina siano presenti, almeno, i collegamenti alle seguenti librerie (ovviamente il percorso dei vari file deve essere adattato alle specifiche del progetto che si sta realizzando)

```
<link rel="stylesheet" href="uikit.min.css" />

<script src="jquery.js"></script>
<script src="uikit.min.js"></script>
```

   In questo modo avremo la certezza di poter utilizzare tutti gli elementi di base del framework.

   Nel caso in cui dovessimo poi utilizzare all'interno di una certa pagina anche alcuni dei componenti avanzati, allora oltre alle librerie sopra indicate dovranno essere inseriti anche i collegamenti alle specifiche librerie del componente, facendo ovviamente attenzione, per quel che riguarda le librerie javascript, all'ordine di inclusione.

   Supponendo, ad esempio, di voler utilizzare il componente avanzato **Slider**, dovremo allora inserire anche i collegamenti alle librerie **slider.css** e **slider.js** (meglio se nella loro versione minificata).

   Nella sezione < head > della pagina dovranno quindi essere presenti i seguenti collegamenti:

```
<link rel="stylesheet" href="uikit.min.css" />
<link rel="stylesheet" href="components/slider.min.css" />

<script src="jquery.js"></script>
<script src="uikit.min.js"></script>
<script src="components/slider.min.js"></script>
```

   ATTENZIONE! Le librerie js relative ai componenti avanzati devono essere incluse dopo la libreria di base uikit.js

3. **Impostare il viewport del sito**

   Uikit è concepito per funzionare secondo l'approccio mobile first ed è perfettamente responsivo per cui in ogni progetto che lo utilizza è necessario includere il meta tag viewport con questi parametri

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

4. **Definire il markup dei componenti come previsto dal framework**
5. **Assegnare ad ogni componente la relativa classe CSS e/o i relativi attributi secondo quanto previsto dalle specifiche di funzionamento del componente stesso**

Ora se per i primi 3 punti si tratta, più che altro, di operazioni preliminari, gli ultimi due identificano invece quello che è il vero e proprio utilizzo del framework.

Ogni componente deve avere un suo ben preciso markup HTML dal quale non si può prescindere. La formattazione grafica e le funzionalità, a livello di interazione utente, del componente sono determinate invece dalle specifiche classi CSS e/o dagli specifici attributi HTML che gli vengono assegnati.

In definitiva dunque, posto di aver eseguito correttamente tutte le operazioni preliminari, per poter utilizzare un qualsiasi componente del framework all'interno del nostro sito dovremo, molto semplicemente, riprodurre il suo markup HTML e assegnargli le classi CSS e/o gli attributi HTML corretti.

Per i dettegli relativamente al markup HTML di uno specifico componente, alle classi CSS o agli attributi HMTL necessari per poter attivare determinate configurazione del componente oltre che per esempi di funzionamento, si consiglia di fare sempre riferimento alla documentazione ufficiale presente sul sito del progetto http://getuikit.com/index.html, documentazione questa alla quale faremo effettivamente riferimento anche nei successivi capitoli di questa guida.

PASSWEB – OPERAZIONI PRELIMINARI

Per poter realizzare un sito responsivo sfruttando al meglio, e contemporaneamente, tutte le funzionalità e le possibilità offerte da uikit e dal Live Editing di Passweb occorre effettuare innanzitutto un paio di semplici operazioni preliminari.

Nello specifico sarà necessario:

- Creare e configurare correttamente un Layout di Variante
- Apportare alcune semplici modifiche alle regole presenti all'interno del file uikit.css

Vediamo nel dettaglio come e perché effettuare queste operazioni preliminari

LAYOUT DI VARIANTE

La prima cosa da fare è indubbiamente quella di creare, per il nostro sito Passweb, un **Layout di Variante da poter poi utilizzare su tutte le pagine del sito** (per maggiori informazioni sui Layout di Pagina e/o di Variante si veda il relativo capitolo del manuale di prodotto)

Tale layout è necessario, essenzialmente, per due ragioni:

- **Impostare il viewport del sito**
- **Inserire nella sezione < head > di ogni singola pagina i collegamenti alle librerie uikit.css e uikit.js necessarie per il corretto funzionamento del framework**

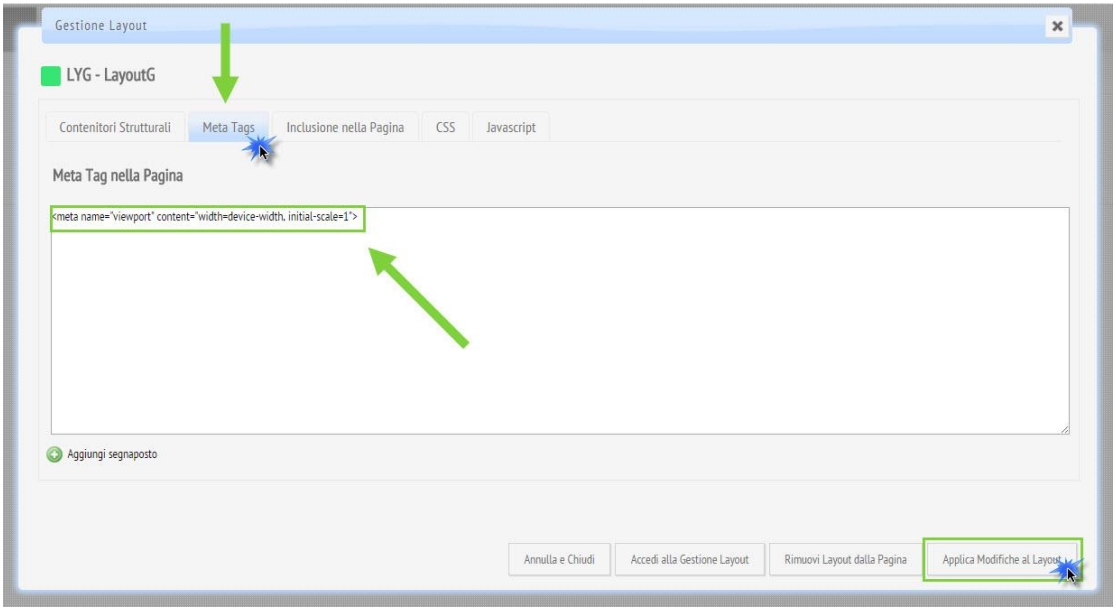
Per quel che riguarda l'impostazione del viewport il discorso è estremamente semplice.

Sulla base di quanto visto nei precedenti capitoli sappiamo infatti che una condizione imprescindibile per poter realizzare un sito responsivo è quella di impostare su tutte le pagine il meta tag viewport come di seguito indicato

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

In questo modo riusciamo ad indicare al browser del dispositivo di assegnare al viewport esattamente la larghezza del device in quanto saremo poi noi ad occuparci di ottimizzare i contenuti e non avremo quindi bisogno di alcun adattamento automatico

Considerando quindi che il Layout di Variante viene applicato automaticamente a tutte le pagine per impostare il viewport del sito è sufficiente copiare ed incollare la stringa sopra indicata all'interno della sezione "**Meta Tags**" del Layout di Variante come indicato in figura



Per quel che riguarda invece i collegamenti alle librerie necessarie per il corretto funzionamento del framework è necessario fare alcune considerazioni aggiuntive.

Per effettuare questa operazione possiamo infatti agire in due modi diversi:

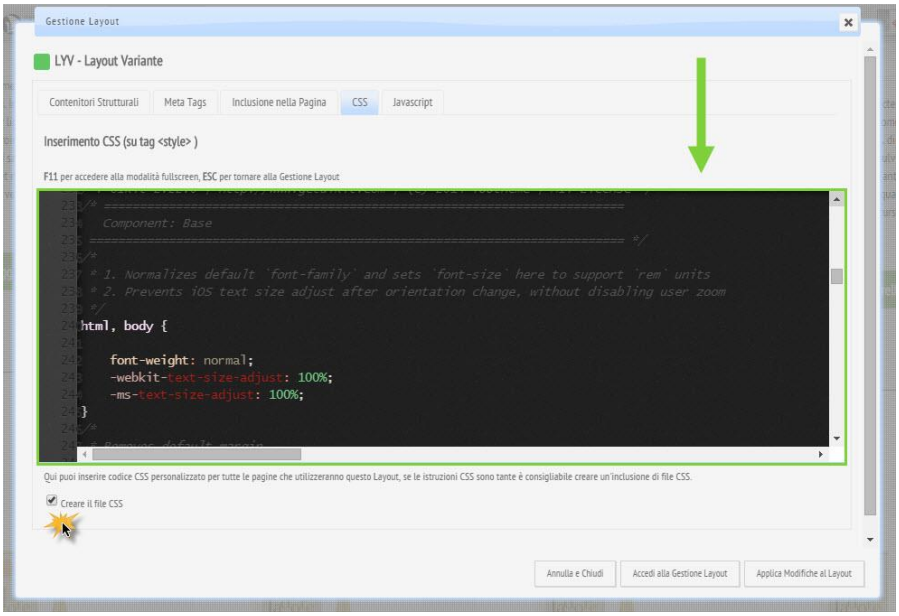
- Includere le librerie uikit.css e uikit.js lavorando per questo all'interno della sezione **"Inclusioni nella Pagina"** del layout di Variante
- Copiare ed incollare il contenuto, rispettivamente, dei file **uikit.css** e **uikit.js** nelle relative sezioni CSS e Javascript del Layout di Variante

La soluzione ottimale è una combinazione delle due possibilità sopra indicate.

Nel momento in cui dovessimo infatti decidere di includere entrambe le librerie mediante la sezione "Inclusioni nella Pagina" dovremmo poi considerare che queste verranno effettivamente applicate solo lato front end e non potremo quindi sfruttarne le funzionalità all'interno del live editing.

Se questo non rappresenta un problema per quel che riguarda le librerie javascript in quanto, come noto, l'esecuzione di codice javascript aggiuntivo è comunque bloccata lato Wizard, lo è invece per quel che riguarda le librerie CSS. Il fatto di poter applicare le regole definite nel file uikit.css solo lato front end ci impedisce infatti di avere all'interno del Live Editing di Passweb l'esatta percezione di quello che sarà poi l'effettivo risultato all'interno del sito rendendo quindi piuttosto complicato il fatto di poter lavorare con gli strumenti tradizionali di Passweb.

Per risolvere questo problema è sufficiente evitare di includere il file uikit.css nella sezione "Inclusioni nella Pagina" e, piuttosto, copiare ed incollare il suo contenuto direttamente nella sezione CSS del Layout di Variante.



In questo modo le regole presenti all'interno del file verranno applicate anche lato Wizard permettendoci quindi di ottenere anche nel Live Editing lo stesso risultato che si avrà poi sul front end del sito.

Riassumendo è quindi necessario:

- Creare un Layout di Variante
- Impostare il viewport del sito nella sezione "Meta Tags" del suddetto Layout
- Includere il collegamento alla libreria uikit.js nella sezione "Inclusioni nella Pagina" del suddetto Layout
- Copiare il contenuto del file uikit.css ed incollarlo all'interno della sezione "CSS" del suddetto Layout (si consiglia anche di flaggare il parametro "Creare il file CSS")

ATTENZIONE! Ovviamente nel caso in cui si dovesse utilizzare in una specifica pagina del sito un componente aggiuntivo di uikit per il quale le sole librerie uikit.css e uikit.js non sono più sufficienti a garantirne il corretto funzionamento, si dovrà procedere sempre secondo la stessa logica ossia caricando la specifica libreria javascript nella sezione "Inclusioni nella Pagina" del Layout di pagina e copiando ed incollando il contenuto della specifica libreria css nella sezione CSS dello stesso Layout.

PERSONALIZZAZIONE DEL FILE UIKIT.CSS

Di base uikit non crea alcun tipo di conflitto con quello che è il normale funzionamento di Passweb sia lato front end che lato back end permettendoci quindi di poterlo integrare all'interno del sito senza troppi problemi.

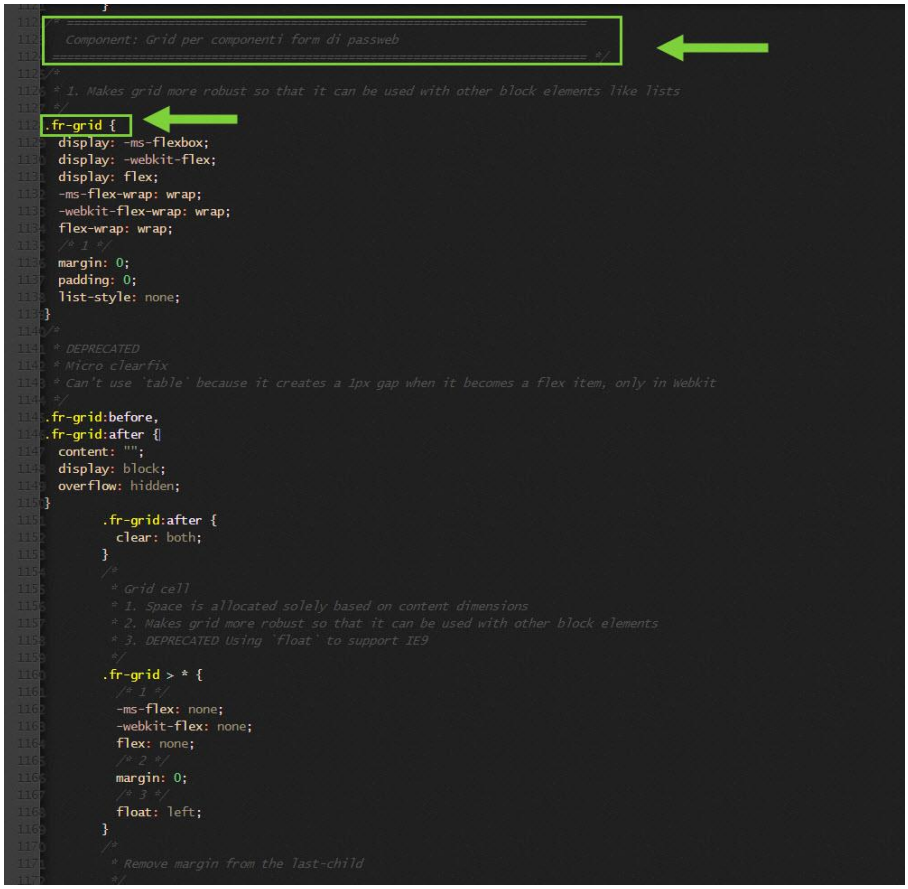
In realtà per poter lavorare al meglio è necessario apportare al file uikit.css alcune semplici modifiche relative, essenzialmente, al componente Griglia.

La prima di queste modifiche è dovuta ad alcuni problemi che potrebbero verificarsi tendando di applicare il Componente Griglia, nella sua versione responsiva, ai Componenti Passweb di tipo form, come ad esempio i Componenti "Registrazione Utente" o "Profilo Utente".

Nei file strutturali di Passweb infatti sono presenti alcune regole, utilizzate per impostare la larghezza dei componenti interni al form, che hanno una priorità maggiore rispetto alle regole definite nel file uikit.css e che dovrebbero essere utilizzate per impostare la larghezza di questi stessi componenti nel momento in cui si dovesse decidere di applicare la Griglia Responsiva di uikit ai form di Passweb.

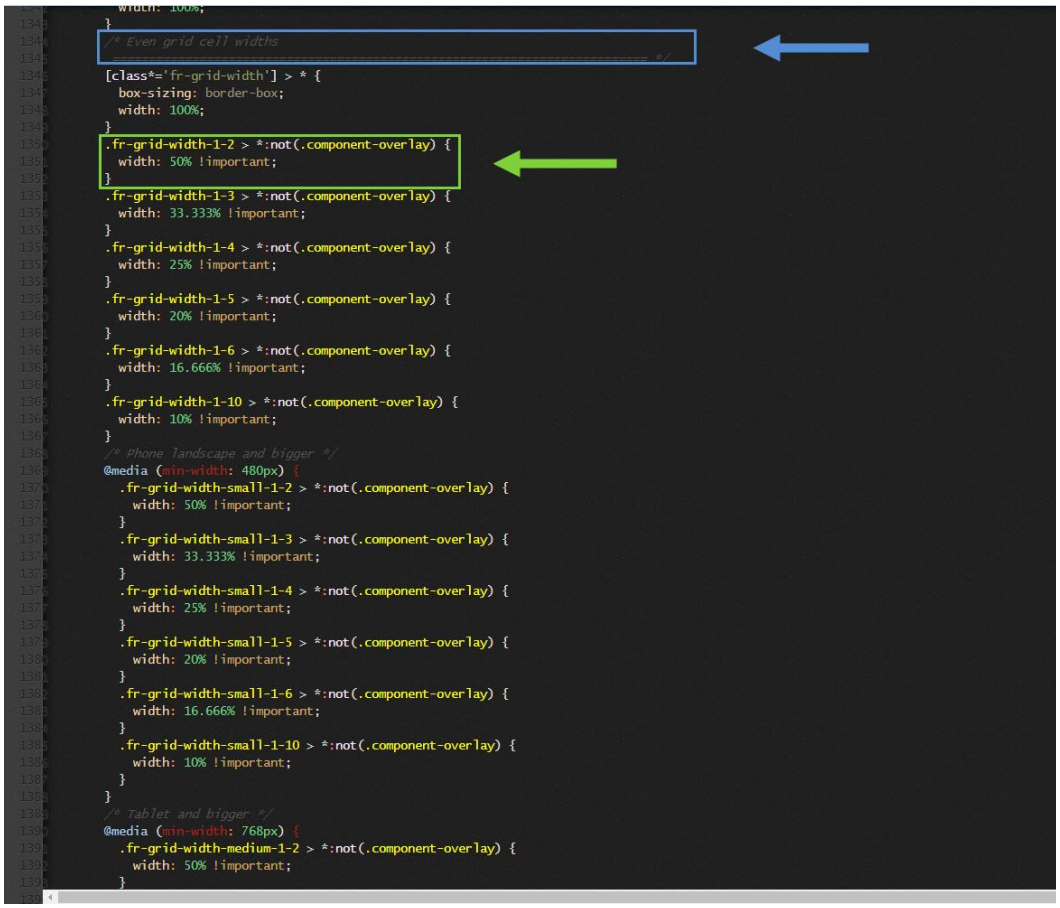
Il modo più semplice per risolvere questo problema e per poter quindi implementare anche all'interno dei Componenti Passweb "Registrazione Utente" e "Profilo Utente" la Griglia di uikit è il seguente:

- Creare all'interno del file uikit.css una nuova sezione denominata, ad esempio, "**Component: Grid per componenti form di Passweb"**
- Copiare all'interno di questa nuova sezione tutte le regole presenti all'interno della sezione "**Component: Grid**" del file uikit.css stesso e **sostituire per esse il suffisso uk con un suffisso personalizzato, ad esempio fr**



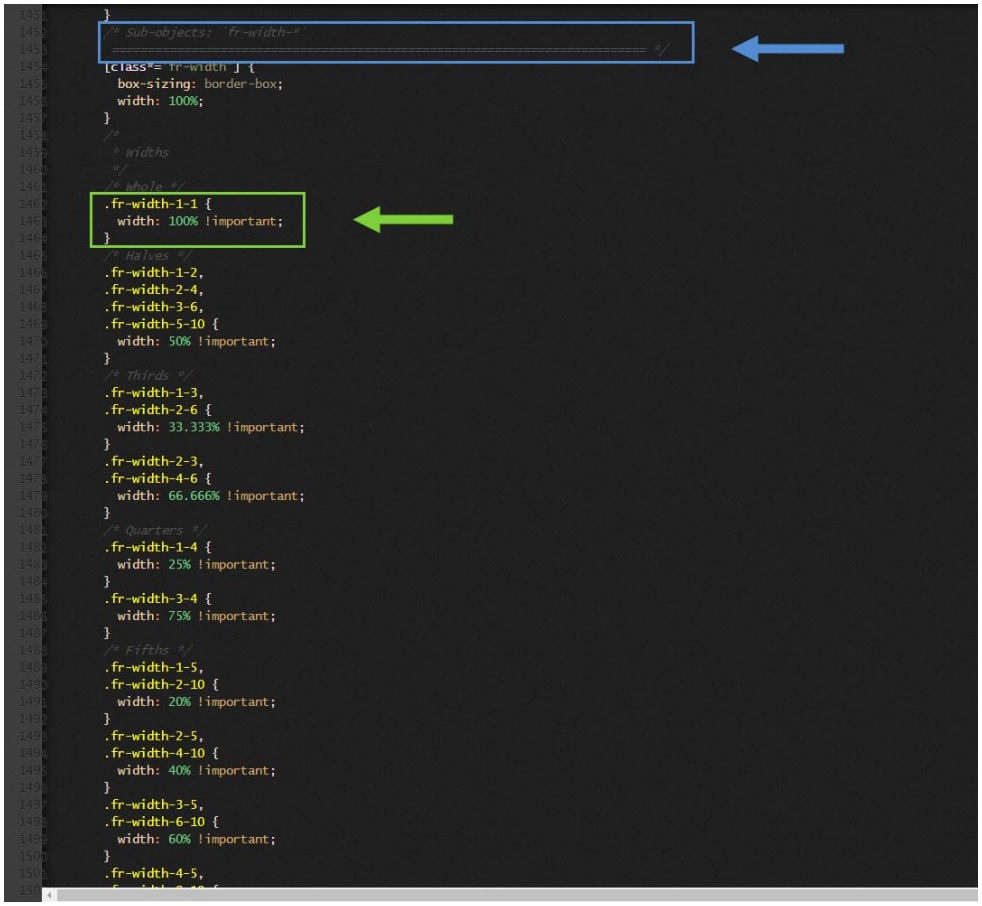
```
/* ==========================================================================
   Component: Grid per componenti form di passweb
   ========================================================================== */
/*
 * 1. Makes grid more robust so that it can be used with other block elements like lists
 */
.fr-grid {
  display: -ms-flexbox;
  display: -webkit-flex;
  display: flex;
  -ms-flex-wrap: wrap;
  -webkit-flex-wrap: wrap;
  flex-wrap: wrap;
  /* 1 */
  margin: 0;
  padding: 0;
  list-style: none;
}
/*
 * DEPRECATED
 * Micro clearfix
 * Can't use `table` because it creates a 1px gap when it becomes a flex item, only in Webkit
 */
.fr-grid:before,
.fr-grid:after {
  content: "";
  display: block;
  overflow: hidden;
}
        .fr-grid:after {
          clear: both;
        }
        /*
         * Grid cell
         * 1. Space is allocated solely based on content dimensions
         * 2. Makes grid more robust so that it can be used with other block elements
         * 3. DEPRECATED Using `float` to support IE9
         */
        .fr-grid > * {
          /* 1 */
          -ms-flex: none;
          -webkit-flex: none;
          flex: none;
          /* 2 */
          margin: 0;
          /* 3 */
          float: left;
        }
        /*
         * Remove margin from the last-child
         */
```

- assegnare l'attributo **!important** a tutte le regole utilizzate, **all'interno della nuova sezione**, per impostare la proprietà width dei diversi elementi della griglia. Nello specifico tale attributo andrà aggiunto alle regole presenti:
    - Nella sotto sezione "**Even grid cell widths**"



```
  }
  /* Even grid cell widths
     ========================================================================== */
  [class*='fr-grid-width'] > * {
    box-sizing: border-box;
    width: 100%;
  }
  .fr-grid-width-1-2 > *:not(.component-overlay) {
    width: 50% !important;
  }
  .fr-grid-width-1-3 > *:not(.component-overlay) {
    width: 33.333% !important;
  }
  .fr-grid-width-1-4 > *:not(.component-overlay) {
    width: 25% !important;
  }
  .fr-grid-width-1-5 > *:not(.component-overlay) {
    width: 20% !important;
  }
  .fr-grid-width-1-6 > *:not(.component-overlay) {
    width: 16.666% !important;
  }
  .fr-grid-width-1-10 > *:not(.component-overlay) {
    width: 10% !important;
  }
  /* Phone landscape and bigger */
  @media (min-width: 480px) {
    .fr-grid-width-small-1-2 > *:not(.component-overlay) {
      width: 50% !important;
    }
    .fr-grid-width-small-1-3 > *:not(.component-overlay) {
      width: 33.333% !important;
    }
    .fr-grid-width-small-1-4 > *:not(.component-overlay) {
      width: 25% !important;
    }
    .fr-grid-width-small-1-5 > *:not(.component-overlay) {
      width: 20% !important;
    }
    .fr-grid-width-small-1-6 > *:not(.component-overlay) {
      width: 16.666% !important;
    }
    .fr-grid-width-small-1-10 > *:not(.component-overlay) {
      width: 10% !important;
    }
  }
  /* Tablet and bigger */
  @media (min-width: 768px) {
    .fr-grid-width-medium-1-2 > *:not(.component-overlay) {
      width: 50% !important;
    }
```

- Nella sotto sezione " **Sub-objects: fr-width-***"



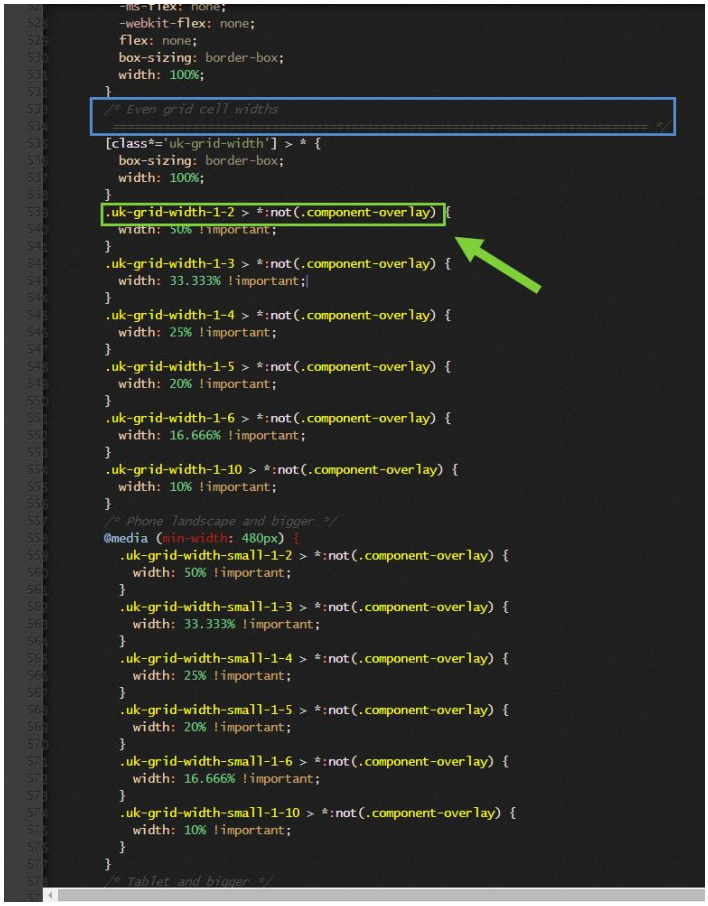
```
}
/* Sub-objects: `fr-width-*`
   ========================================================================== */
[class*=" fr-width-"] {
  box-sizing: border-box;
  width: 100%;
}
/*
 * Widths
 */
/* Whole */
.fr-width-1-1 {
  width: 100% !important;
}
/* Halves */
.fr-width-1-2,
.fr-width-2-4,
.fr-width-3-6,
.fr-width-5-10 {
  width: 50% !important;
}
/* Thirds */
.fr-width-1-3,
.fr-width-2-6 {
  width: 33.333% !important;
}
.fr-width-2-3,
.fr-width-4-6 {
  width: 66.666% !important;
}
/* Quarters */
.fr-width-1-4 {
  width: 25% !important;
}
.fr-width-3-4 {
  width: 75% !important;
}
/* Fifths */
.fr-width-1-5,
.fr-width-2-10 {
  width: 20% !important;
}
.fr-width-2-5,
.fr-width-4-10 {
  width: 40% !important;
}
.fr-width-3-5,
.fr-width-6-10 {
  width: 60% !important;
}
.fr-width-4-5,
```

In questo modo potremo poi implementare la Griglia di uikit all'interno dei Componenti Passweb di tipo form ("Registrazione Utente", "Profilo Utente" ecc…) operando esattamente allo stesso modo in cui si opererà per implementare questo stesso componente in una qualsiasi altra pagina del nostro sito Passweb.

L'unica accortezza sarà ovviamente quella di utilizzare per questi Componenti non le classi con suffisso uk ma bensì quelle con suffisso fr

La seconda modifica ci consente invece, di poter lavorare al meglio nel Live Editing di Passweb e consiste nell'evitare che alcune delle regole relative al componente Griglia di uikit vengano applicate anche all'overlay (riquadro blu) utilizzato nel Live Editing per evidenziare, al passaggio del mouse, i componenti da gestire.

Nello specifico sarà quindi necessario utilizzare il **selettore css :not** per fare in modo che le regole presenti all'interno della sotto sezione "**Even grid cell widths**" della sezione "**Component: Grid**" non vengano applicate agli elementi di classe **.component-overlay** (che sono effettivamente quelli utilizzati all'interno del Wizard per gestire i riquadri blu di selezione di cui parlavamo prima).



ATTENZIONE! La stessa cosa andrà fatta, ovviamente, anche per le analoghe regole con suffisso fr presenti all'interno della nuova sezione precedentemente creata.

L'ultima modifica, infine, è più che altro un'integrazione del file uikit.css con le regole necessarie per poter assegnare a componenti Passweb quali il Catalogo Ecommerce Offerte / Novità Popolarità Prodotto ecc… un comportamento responsivo secondo la logica propria di uikit stesso.

Per maggiori informazioni relativamente a questo argomento si veda nel dettaglio il capitolo "*Griglia e Catalogo Ecommerce*" di questa guida

In ogni caso è possibile scaricare il file uikit.css utilizzato nel modello di riferimento, e contenente già tutte le modifiche appena esaminate, all'interno della sezione "*Risorse*" di questa guida (uikitEc29.css).

UIKIT – COMPONENTI

Ovviamente all'interno di questa guida non verranno presi in considerazione tutti i componenti del framework perché non tutti sono effettivamente indispensabili per realizzare un sito responsivo.

D'altra parte verrà invece spiegato come poter rendere responsivo, ad esempio, il Catalogo Ecommerce di Passweb adottando per questo la stessa logica del framework uikit.

L'obbiettivo principale, in definitiva, è sempre quello di capire come dover operare per combinare uikit e Passweb al fine di ottenere un sito ecommerce responsivo facilmente gestibile.

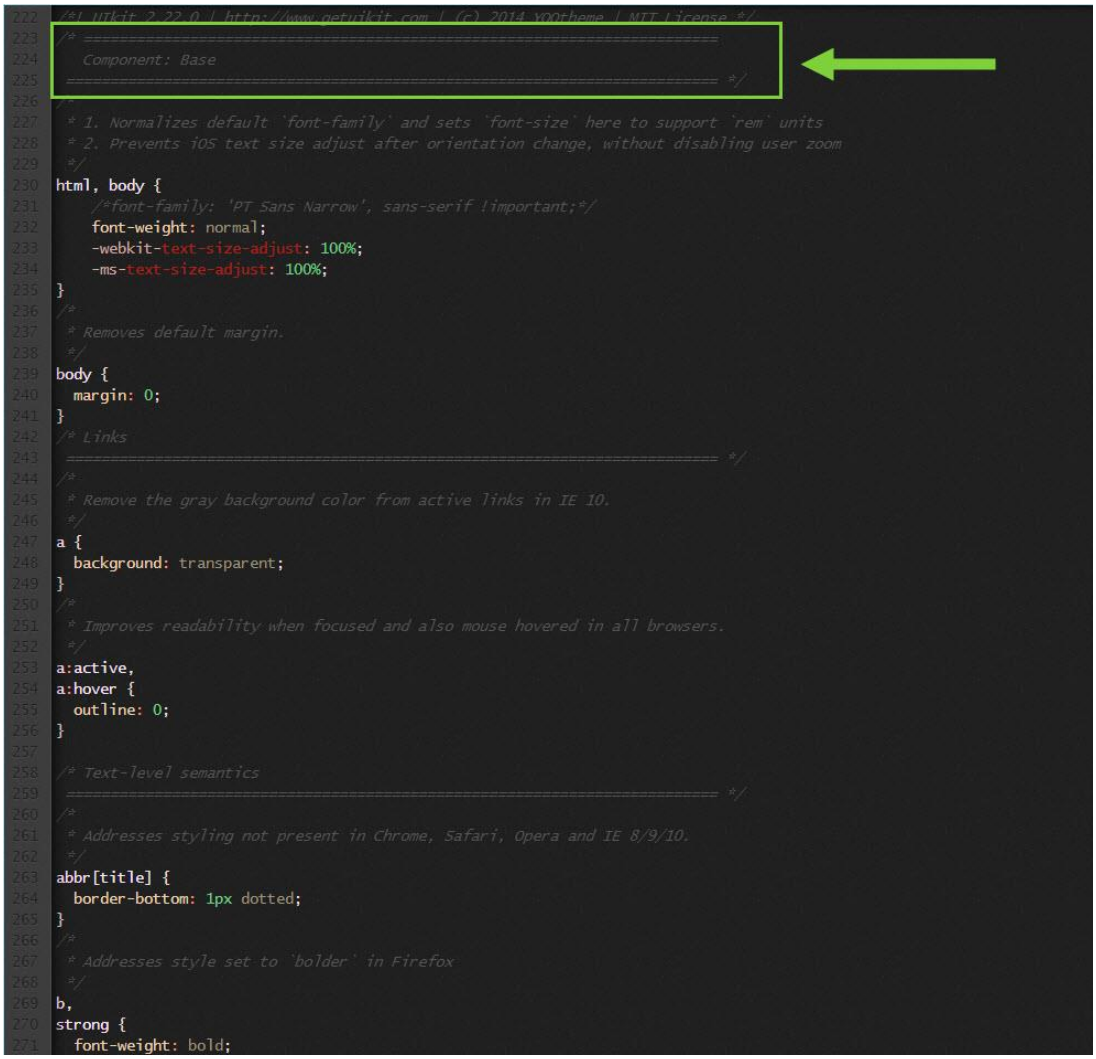
In ogni caso una volta compreso come poter gestire una griglia, un pannello o uno slider, poi il modo di operare sarà sempre lo stesso e il fatto di utilizzare un componente piuttosto che un altro dipenderà essenzialmente dallo specifico progetto da realizzare.

Detto che il sito di riferimento del nostro progetto è raggiungibile all'indirizzo www.ecommerce29.passweb.it passeremo ora ad analizzare, per prima cosa, i principali componenti di uikit utilizzati all'interno di questo modello in termini generali, vedremo poi come poterli implementare in Passweb e, quando possibile, metteremo in evidenza dove e come questi componenti sono stati effettivamente utilizzati nel modello di riferimento.

ELEMENTI DI BASE – TIPOGRAFIA

Il primo argomento fondamentale da analizzare è indubbiamente quello che riguarda la **tipografia** del sito. Avere consapevolezza dei default fissati nel framework aiuta enormemente nel momento in cui si desideri apportare poi delle modifiche in maniera tale da personalizzare con stili propri l'aspetto del testo.

ATTENZIONE! Le regole che determinano l'aspetto dei diversi elementi testuali (paragrafi, link, titoli ecc…) sono inserite tutte nella parte iniziale del file uikit.css e, nello specifico, all'interno della sezione "Component: Base"



```
/*! UIkit 2.22.0 | http://www.getuikit.com | (c) 2014 YOOtheme | MIT License */
/* ========================================================================
   Component: Base
 ========================================================================== */
/*
 * 1. Normalizes default `font-family` and sets `font-size` here to support `rem` units
 * 2. Prevents iOS text size adjust after orientation change, without disabling user zoom
 */
html, body {
    /*font-family: 'PT Sans Narrow', sans-serif !important;*/
    font-weight: normal;
    -webkit-text-size-adjust: 100%;
    -ms-text-size-adjust: 100%;
}
/*
 * Removes default margin.
 */
body {
  margin: 0;
}
/* Links
 ========================================================================== */
/*
 * Remove the gray background color from active links in IE 10.
 */
a {
  background: transparent;
}
/*
 * Improves readability when focused and also mouse hovered in all browsers.
 */
a:active,
a:hover {
  outline: 0;
}

/* Text-level semantics
 ========================================================================== */
/*
 * Addresses styling not present in Chrome, Safari, Opera and IE 8/9/10.
 */
abbr[title] {
  border-bottom: 1px dotted;
}
/*
 * Addresses style set to `bolder` in Firefox
 */
b,
strong {
  font-weight: bold;
```

IMPOSTAZIONI BASE PER IL TESTO

All'inizio della sezione "Component: Base" del file uikit.css, troviamo la regola con cui si setta, tra le altre cose, la dimensione di base del testo

```
html {
  font: normal 14px / 20px "Helvetica Neue", Helvetica, Arial, sans-serif;
  -webkit-text-size-adjust: 100%;
  -ms-text-size-adjust: 100%;
  background: #ffffff;
  color: #444444;
}
```

Analizzando tale regola possiamo osservare che:

- La dimensione di base del font è di 14px. Questo è anche il valore che verrà utilizzato come riferimento per determinare le effettive dimensioni di tutti quegli elementi che utilizzano come unità di misura **em**, **rem**, o **%** (per maggiori informazioni in merito a queste unità di misura si veda anche la sezione "*Responsive Design: Concetti di Base – Tipografia*" di questa guida).

  Di base, dunque, 1em equivale esattamente a 14px

- L'altezza riga (line-height) utilizzata a default è di 20px
- Alla pagina viene applicato uno sfondo di colore bianco (#ffffff)
- Il colore primario utilizzato per il testo è nero con codice esadecimale #444444
- Di default uikit utilizza una font-family sans-serif definita a livello del selettore html.

  Tutti gli elementi testuali ereditano questa impostazione ad eccezione dei blocchi di codice racchiusi dal tag < code > per i quali viene utilizzato invece una "**font-family: Consolas,monospace,serif**" come successivamente specificato nella rispettiva regola presente sempre all'interno della sezione "Component: Base"

Analizzando le altre regole presenti all'interno di questa sezione possiamo osservare inoltre che.

- I paragrafi (< p >) e gli altri elementi di blocco ereditano i valori dal selettore html. Anche per essi è quindi utilizzato un font di base a 14px, un'altezza di riga di 20 px ecc…
- Volendo enfatizzare del testo lo si può racchiudere all'interno di specifici tag come **< em >**, per il quale viene applicato a default un colore con codice #dd0055, **< strong >** o **< b >** per i quali viene applicato a default il grassetto, **< mark >** che consente di evidenziare il testo in giallo (#ffffaa) …
- Volendo togliere enfasi al testo è possibile racchiuderlo all'interno del tag **< small >** per il quale viene applicato un font-size pari al 80% (corrispondente quindi a 14 x 0.8 = 11.2 px)
- Per quando riguarda i link il colore primario è azzurro (#0077dd). Sul hover, i link assumono invece una variante di blu leggermente più scura (#005599) senza sottolineatura
- Per gli elenchi puntati viene utilizzato un padding sinistro di 30 px

Le proprietà relative ai titoli (tag da < h1 > a < h6 >) sono racchiuse, infine, all'interno della sotto-sezione Headings.

```
/* Headings
 ========================================================================== */
h1,
h2,
h3,
h4,
h5,
h6 {
  margin: 0 0 15px 0;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-weight: normal;
  color: #444444;
  text-transform: none;
}
/*
 * Margins
 */
* + h1,
* + h2,
* + h3,
* + h4,
* + h5,
* + h6 {
  margin-top: 25px;
}
/*
 * Sizes
 */
h1,
.uk-h1 {
  font-size: 36px;
  line-height: 42px;
}
h2,
.uk-h2 {
  font-size: 24px;
  line-height: 30px;
}
h3,
.uk-h3 {
  font-size: 18px;
  line-height: 24px;
}
h4,
```

Per ognuno di essi viene definita una dimensione del font diversa (dai 36 px del tag h1 ai12 px del tag h6) e una diversa altezza di riga. Ad ogni titolo è inoltre applicato a default un margine superiore di 25 px.

PERSONALIZZARE LA TIPOGRAFIA

Come ormai chiaro, le regole presenti all'interno della sezione "Component: Base" del file uikit.css definiscono un insieme di stili applicati a default alla tipografia del sito.

Questi stili possono comunque essere modificati e personalizzati in un qualsiasi momento arrivando così a definire una tipografia completamente diversa da quella applicata a default dal framework.

In questo senso è possibile agire in modi diversi:

- **Utilizzando il Customizer di uikit (http://getuikit.com/docs/customizer.html)**

  Come precedentemente evidenziato questo strumento mette a disposizione dell'utente un'interfaccia grafica mediante la quale poter personalizzare elementi quali il colore del testo, la dimensione del font, il colore dei link ecc… e di creare quindi una versione del file uikit.css personalizzata secondo quelle che sono le proprie specifiche esigenze.

  Per maggiori informazioni in merito si veda anche la sezione "Uikit.css – Temi personalizzati" di questa guida

- **Intervenendo manualmente sul contenuto del file uikit.css**

  Nel caso in cui non sia possibile intervenire tramite Customizer sul valore di una certa proprietà, come può essere ad esempio la dimensione del font o il valore dei margini per i titoli h1 – h2 … h6, è sufficiente modificare i relativi valori direttamente all'interno del file uikit.css.

  Si tratterà infatti di individuare all'interno della sezione "Component: Base" la specifica regola e variare da qui i valori impostati a default dal framework

- **Utilizzando lo style editor di Passweb**

  Considerando che le proprietà CSS impostate mediante Style Editor hanno, generalmente, una priorità maggiore (essendo più interne) rispetto ai valori impostati per queste stesse proprietà in una qualsiasi libreria caricata nel rispettivo layout di pagina e/o di sito, è possibile utilizzare lo Style Editor di Passweb per effettuare l'override di queste stesse regole andando così a sovrascrivere i valori applicati a default dal framework.

Volendo potremmo anche decidere di eliminare completamente dal file uikit.css tutta la sezione "Component: Base" e gestire così, interamente, la tipografia del sito con il solo Style Editor di Passweb così come avviene per un qualsiasi altro sito che non utilizza alcun framework.

MODELLO ECOMMERCE 29

Nel modello di riferimento è stato scelto di personalizzare la tipografia del sito utilizzando principalmente lo Style Editor di Passweb.

In conseguenza di ciò sono state eliminate dal file uikit.css tutte le regole relative alla gestione della Tipografia

ATTENZIONE! E' possibile scaricare il file uikit.css utilizzato all'interno del modello di riferimento nella sezione "*Risorse*" di questa guida

ELEMENTI DI BASE – IMMAGINE

In uikit tutte le immagini sono responsive a default.

Andando infatti ad analizzare la regola applicata nella sezione "Component: Base" al tag img

```
img {
  max-width: 100%;
  height: auto;
  box-sizing: border-box;
  border: 0;
  vertical-align: middle;
}
```

ritroviamo esattamente quanto detto nel precedente capitolo **"Responsive Design: Concetti di base – Immagini Responsive"** di questa guida.

In sostanza le due proprietà "**max-width: 100%**" e "**hieght: auto**" garantiscono che riducendo le dimensioni del browser, e ovviamente, dei contenitori all'interno dei quali sono inserite le immagini, si riducano automaticamente e proporzionalmente anche esse.

Per annullare il comportamento responsivo di un' immagine e fare in modo che mantenga le sue dimensioni originali, è sufficiente aggiungere al **tag img** un'apposita classe: **.uk-img-preserve** che come indicato nella relativa regola

```
.uk-img-preserve,.uk-img-preserve img {
  max-width: none;
}
```

non fa altro che annullare il valore impostato a default per la larghezza massima.

A questo punto è bene fare una precisazione. In Passweb le immagini possono essere gestite in modi diversi:

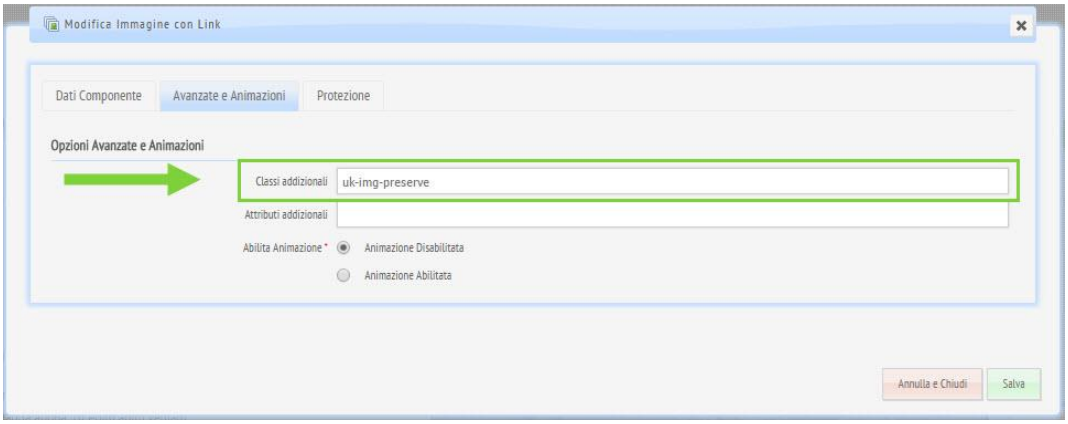
- Mediante il componente "Immagine con Link"
- All'interno del componente Paragrafo
- Mediante il componente HML
- Mediante il componente "Immagine" interno a componenti quali il "Catalogo Ecommerce" la "Scheda Prodotto" ecc…
- …

Ora, mentre in alcuni casi (componente “HTML” o sezione codice del componente “Paragrafo”) la struttura del componente Passweb permette di accedere direttamente al tag img e di applicargli quindi tutte le classi desiderate, compresa quella richiesta da uikit per annullare il comportamento responsivo, in altri casi il markup dello specifico componente Passweb non ci consente di accedere direttamente a questo tag.

Prendiamo ad esempio il classico componente “Immagine con link che produce un markup di questo tipo

```
<div id="imagelink_3165" class="imagelinkComp component columned ">
  <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
</div>
```

Applicando la classe richiesta da uikit per annullare il comportamento responsivo dell’immagine a questo componente, e utilizzando per questo l’apposito campo presente nella sua maschera di configurazione



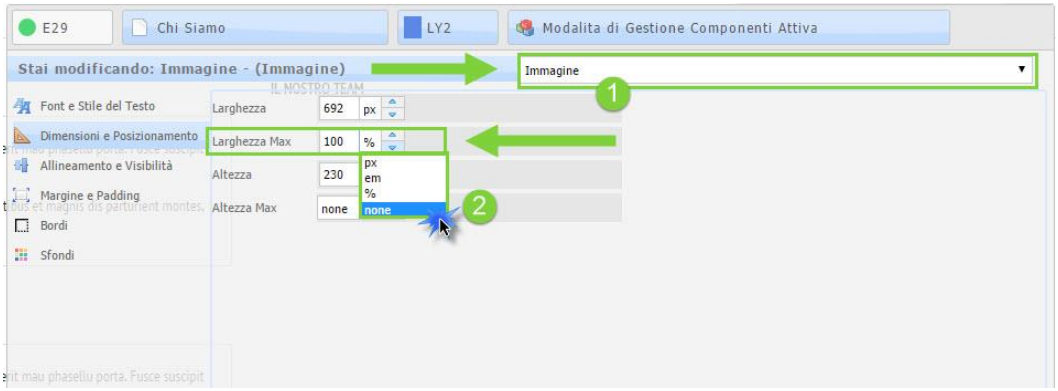
in realtà questa classe viene applicata non al tag img ma bensì al suo contenitore più esterno

```
<div id="imagelink_3165" class="imagelinkComp component columned uk-img-preserve">
  <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
</div>
```

Fortunatamente riusciamo comunque, anche in questo caso, a raggiungere il nostro scopo in quanto la regola definita nel file uikit.css sulla classe in esame è strutturata in maniera tale da annullare l’impostazione di default sulla larghezza massima non solo se la classe viene applicata direttamente al tag img ma anche nel caso in cui questa classe venga applicata ad un contenitore esterno all’immagine (.uk-img-preserve img).

In ogni caso potremmo tranquillamente annullare il comportamento responsivo dell’immagine anche utilizzando direttamente lo style editor di Passweb.

La regola sulla classe .uk-img-preserve infatti, non fa altro che impostare per l’immagine la max-width sul valore none, cosa questa che potremmo benissimo fare aprendo lo style editor del componente, selezionando l’elemento **“Immagine” (1)** e impostando per esso la proprietà **Larghezza Max** sul valore **none (2)**



ATTENZIONE! Nel caso in cui si dovesse utilizzare lo Style Editor di Passweb per fissare l’altezza, la larghezza o la larghezza massima di un’immagine su specifici valori in pixel, questi sovrascriveranno (avendo una maggior priorità) i rispettivi valori applicati a default per queste stesse proprietà dal framework annullando, di fatto, il comportamento responsivo dell’immagine.

Si ricorda infine, nel caso in cui si decida di gestire immagini responsive, di **utilizzare sempre immagini sufficientemente grandi e adeguate a tutte le dimensioni che il layout può raggiungere.**

LA GRIGLIA

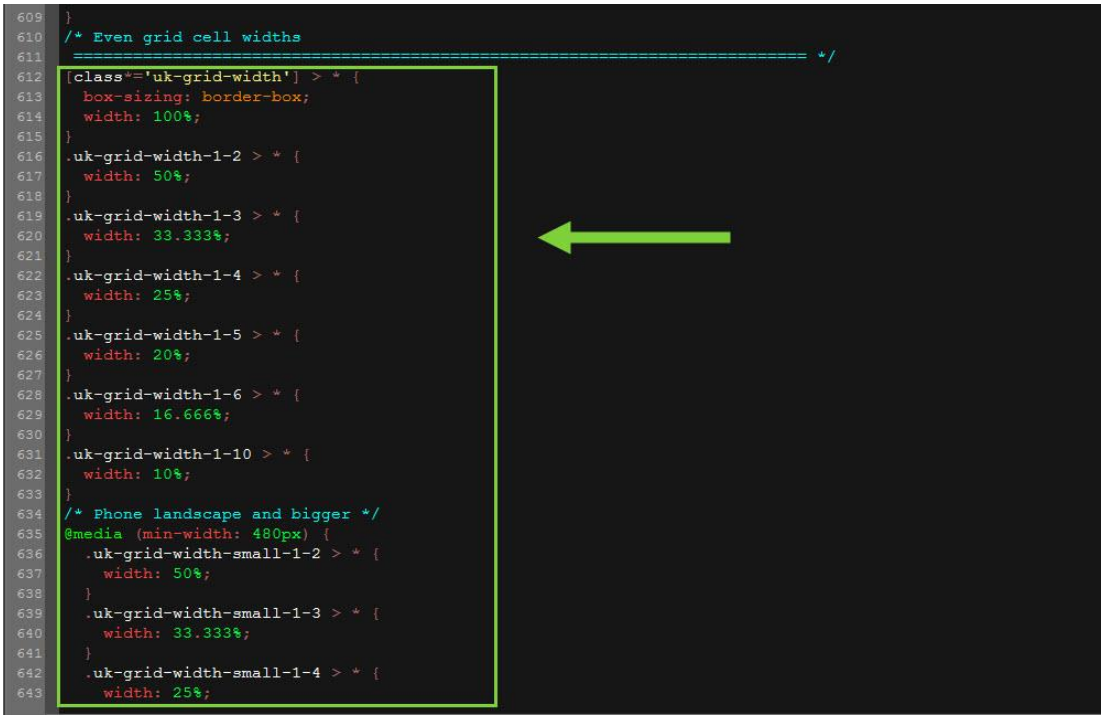
Il sistema a Griglie adottato da uikit mette a disposizione dell'utente uno strumento estremamente sofisticato, flessibile ed efficace che rappresenta, di fatto, il cuore di tutto il framework.

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/grid.html

La Griglia di uikit è stata progettata seguendo l'approccio mobile first, è fluida e consente di suddividere lo spazio a sua disposizione ospitando da un minimo di 1 ad un massimo di 10 distinte colonne (o blocchi) di contenuti.

ATTENZIONE! Più esattamente ogni griglia può essere suddivisa in 1,2,3,4,5,6 o 10 distinte colonne (non è quindi possibile pensare di inserire all'interno di una griglia su di una stessa riga 7, 8 o 9 distinti blocchi di contenuto)

La larghezza di ogni colonna è definita in percentuale, come è possibile osservare analizzando la sezione del file uikit.css riservata a questo specifico componente.



```
}
/* Even grid cell widths
 ========================================================================== */
[class*='uk-grid-width'] > * {
  box-sizing: border-box;
  width: 100%;
}
.uk-grid-width-1-2 > * {
  width: 50%;
}
.uk-grid-width-1-3 > * {
  width: 33.333%;
}
.uk-grid-width-1-4 > * {
  width: 25%;
}
.uk-grid-width-1-5 > * {
  width: 20%;
}
.uk-grid-width-1-6 > * {
  width: 16.666%;
}
.uk-grid-width-1-10 > * {
  width: 10%;
}
/* Phone landscape and bigger */
@media (min-width: 480px) {
  .uk-grid-width-small-1-2 > * {
    width: 50%;
  }
  .uk-grid-width-small-1-3 > * {
    width: 33.333%;
  }
  .uk-grid-width-small-1-4 > * {
    width: 25%;
```

ATTENZIONE! Le regole CSS relative al componente Griglia sono localizzate tutte nel file uikit.css, all'interno della sezione "**Component: Grid**"

```
/* ========================================================================
   Component: Grid
 ======================================================================== */
/*
 * 1. Makes grid more robust so that it can be used with other block elements like lists
 */
.uk-grid {
  display: -ms-flexbox;
  display: -webkit-flex;
  display: flex;
  -ms-flex-wrap: wrap;
  -webkit-flex-wrap: wrap;
  flex-wrap: wrap;
  /* 1 */
  margin: 0;
  padding: 0;
  list-style: none;
}
/*
 * DEPRECATED
 * Micro clearfix
 * Can't use `table` because it creates a 1px gap when it becomes a flex item, only in Webkit
 */
.uk-grid:before,
.uk-grid:after {
  content: "";
  display: block;
  overflow: hidden;
}
.uk-grid:after {
  clear: both;
```

Nei successivi capitoli verranno esaminate le caratteristiche principali di questo componente e verrà spiegato come fare per poterle implementare. Per una trattazione completa di tutte le diverse possibili opzioni di configurazione si consiglia comunque di fare sempre riferimento alla documentazione ufficiale.

CONFIGURAZIONE – 1

Come per tutti i componenti di uikit, anche in questo caso per poter utilizzare una Griglia all'interno del nostro sito le operazioni da fare, di base, sono due:

- replicare il markup HTML previsto da questo componente
- assegnare ai diversi elementi del suddetto markup le classi e/o gli attributi corretti secondo quanto indicato nella documentazione ufficiale del framework

A livello di markup la Griglia è costituita **da un elemento contenitore (es. un < div >) più esterno, che individua il contenitore della Griglia, all'interno del quale andranno poi inseriti tanti elementi contenitore (es. tanti < div >) quanti sono i blocchi di contenuti e quindi le colonne che si desidera inserire nella griglia stessa**

Per quel che riguarda invece le classi da assegnare ai diversi elementi di questo markup:

- **al contenitore più esterno dovrà essere assegnata la classe .uk-grid,**
- **ai contenitori interni alla griglia dovrà essere assegnata una classe .uk-width-*** dove il carattere * dovrà essere sostituito da un sistema di due numeri il primo dei quali indica quante colonne dovranno essere effettivamente occupate da quello specifico componente rispetto al totale in cui si è pensato di suddividere la griglia, mentre il secondo indica proprio questo totale.

ATTENZIONE! La classe .uk-width-* è quella che determina l'effettiva larghezza che andrà ad assumere all'interno della griglia un blocco di contenuti

Supponendo dunque di assegnare ad un blocco di contenuti interno alla griglia la classe **.uk-width-1-3,** ciò starà a significare che si è pensato di suddividere la griglia in 3 distinte colonne e il blocco di contenuti in oggetto andrà ad occupare una sola di queste colonne.

Seguendo questa logica sulla stessa riga potranno quindi essere inseriti:

- altri due distinti blocchi di contenuti a ciascuno dei quali dovrà essere assegnata la classe **.uk-width-1-3** in maniera tale che ciascuno di essi occupi una delle restanti due colonne
- un solo altro blocco di contenuti al quale assegnare la classe **.uk-width-2-3** in maniera tale che occupi tutte e due le restanti colonne

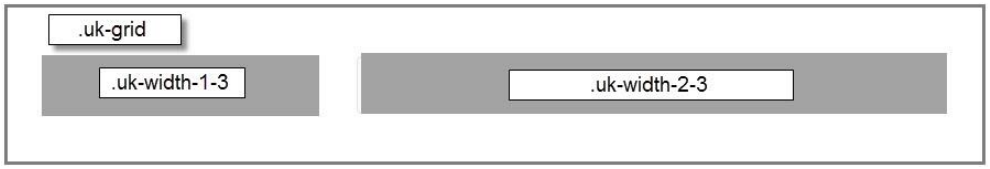
Nel primo caso (tre distinte colonne di contenuti ciascuna delle quali occuperà 1/3 dello spazio disponibile) dovremo quindi avere, complessivamente, un markup di questo tipo

```
<div class="uk-grid">
   <div class="uk-width-1-3">
   .../*Contenuti della prima colonna*/
   </div>
```

```
   <div class="uk-width-1-3">
   .../*Contenuti della seconda colonna*/
   </div>
   <div class="uk-width-1-3">
   .../*Contenuti della terza colonna*/
   </div>
</div>
```

Nel secondo caso (due distinte colonne di contenuti, la prima che occupa 1/3 dello spazio disponibile e la secondo i restanti 2/3) dovremo invece avere, complessivamente, un markup di questo tipo

```
<div class="uk-grid">
   <div class="uk-width-1-3">
   .../*Contenuti della prima colonna*/
   </div>
   <div class="uk-width-2-3">
   .../*Contenuti della seconda colonna*/
   </div>
</div>
```



Considerando che, come precedentemente evidenziato, la Griglia è in grado di supportare 1,2,3,4,5,6 o 10 colonne, la tabella seguente mostra l'insieme di classi che, in generale, possono essere applicate ai blocchi di contenuti interni ad essa.

CLASSE	DESCRIZIONE
.uk-width-1-1	La griglia è suddivisa in una sola colonna. Può ospitare un solo blocco di contenuti che occuperà il 100% della sua larghezza
.uk-width-1-2	La griglia è suddivisa in 2 colonne. Può ospitare due distinti blocchi di contenuti ciascuno dei quali occuperà il 50% della sua larghezza
Da **.uk-width-1-3** a **.uk-width-2-3**	La griglia è suddivisa in 3 colonne. Può ospitare: -) tre distinti blocchi di contenuti ciascuno dei quali occuperà 1/3 della sua larghezza -) due distinti blocchi di contenuti uno che occupi 1/3 della larghezza e l'altro i restanti 2/3
Da **.uk-width-1-4** a **.uk-width-3-4**	La griglia è suddivisa in 4 colonne. Può ospitare: -) quattro distinti blocchi di contenuti ciascuno dei quali occuperà 1/4 della sua larghezza -) due distinti blocchi di contenuti uno che occupi 1/4 della larghezza e l'altro i restanti 3/4 -) due distinti blocchi di contenuti uno che occupi 2/4 della larghezza e l'altro i restanti 2/4
Da **.uk-width-1-5** a **.uk-width-4-5**	La griglia è suddivisa in 5 colonne. Può ospitare: -) cinque distinti blocchi di contenuti ciascuno dei quali occuperà 1/5 della sua larghezza -) due distinti blocchi di contenuti uno che occupi 1/5 della larghezza e l'altro i restanti 4/5 -) due distinti blocchi di contenuti uno che occupi 2/5 della larghezza e l'altro i restanti 3/5 -) tre distinti blocchi di contenuti due che occupino 2/5 della larghezza e l'altro i restanti 1/5
Da **.uk-width-1-6** a **.uk-width-5-6**	La griglia è suddivisa in 6 colonne. Può ospitare: -) sei distinti blocchi di contenuti ciascuno dei quali occuperà 1/6 della sua larghezza -) due distinti blocchi di contenuti uno che occupi 1/6 della larghezza e l'altro i restanti 5/6 -) due distinti blocchi di contenuti uno che occupi 2/6 della larghezza e l'altro i restanti 4/6 -) tre distinti blocchi di contenuti ciascuno dei quali occuperà 2/6 della sua larghezza -) ….

Da **.uk-width-1-10** a **.uk-width-9-10**	La griglia è suddivisa in 10 colonne. Può ospitare: -) dieci distinti blocchi di contenuti ciascuno dei quali occuperà 1/10 della sua larghezza -) due distinti blocchi di contenuti uno che occupi 1/10 della larghezza e l'altro i restanti 9/10 -) due distinti blocchi di contenuti uno che occupi 2/10 della larghezza e l'altro i restanti 8/10 -) ….

ATTENZIONE! Come è semplice comprendere il sistema appena descritto, ed utilizzato per dimensionare in larghezza i blocchi di contenuti interni alla griglia, è ridondante nel senso che, ad esempio, le classi .uk-width-1-2, .uk-width-2-4, .uk-width-3-6 e .uk-width-5-10 portano tutte a fare in modo che il blocco di contenuti al quale vengono applicate occupi, in larghezza, il 50% dello spazio complessivo a disposizione della griglia.

Un'ultima considerazione estremamente importante da fare, in relazione soprattutto al fatto di voler poi implementare questa componente all'interno del proprio sito Passweb, è quella che riguarda il valore della proprietà **float** assegnato agli elementi "strutturali" della Griglia.

In questo senso è bene chiarire che:

- **uikit non assegna gli elementi strutturali della griglia (ne al contenitore esterno ne tanto meno ai singoli blocchi di contenuto in esso inseriti) uno specifico valore per la proprietà float, il che equivale esattamente, come noto, ad impostare per questi stessi elementi la proprietà float sul valore none**

LA GRIGLIA IN PASSWEB

Per poter implementare all'interno del proprio sito Passweb la griglia di uikit nella sua configurazione base, è necessario un passaggio preliminare estremamente importante che si basa sull'ultimo concetto espresso nel capitolo precedente.

Uikit non assegna agli elementi strutturali della griglia (ne al contenitore esterno ne tanto meno ai singoli blocchi di contenuto) uno specifico valore per la proprietà float, il che equivale esattamente ad impostare per questi stessi elementi la proprietà float sul valore none

Ora, il sistema di posizionamento dei Componenti adottato da Passweb non prevede la possibilità di inserire un Componente nella pagina impostando automaticamente la sua proprietà float sul valore none.

Nello specifico infatti per posizionamenti di tipo:

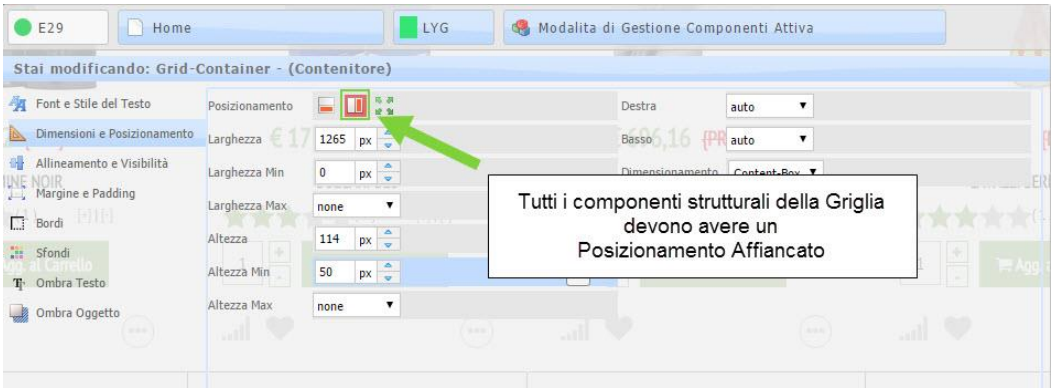
- **Incolonnato:** Passweb considera "**float: left;**" e "**width:100% !important;**"
- **Affiancato:** Passweb considera "**float: left;**" e "**width: auto**"
- **Posizionato:** Passweb considera "**float: left;**" e "**position: absolute**"

Considerando quindi che per implementare all'interno del nostro sito la Griglia di Uikit dovremo necessariamente utilizzare dei Componenti nativi di Passweb, il problema diventa ora capire quale dei tre posizionamenti adottare e soprattutto come fare per poter assegnare ai Componenti che andremo ad utilizzare un "float: none".

Per quel che riguarda il tipo di posizionamento, possiamo sicuramente escludere il "**Posizionato**" in quanto in queste condizioni il componente verrebbe gestito in assoluto (position: absolute).

Anche il posizionamento "**Incolonnato**" potrebbe dare dei problemi a causa delle impostazioni sulla larghezza. Con questo posizionamento infatti il Componente assumerebbe sempre, in larghezza, il 100% del suo contenitore. Inoltre l'attributo "!important" potrebbe anche pregiudicare l'applicazione della larghezza definita nelle media query di uikit, compromettendo quindi il comportamento responsivo della griglia.

La risposta alla prima domanda è quindi utilizzare il posizionamento "Affiancato".

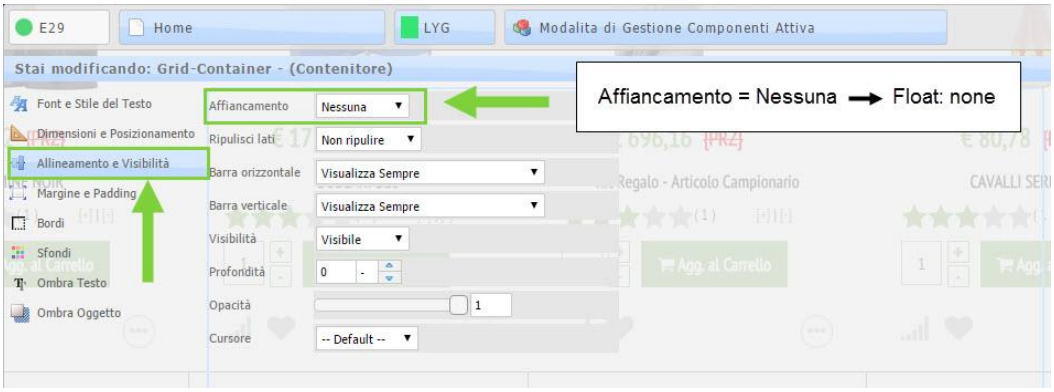


Resta ora da capire come poter impostare per Componenti Passweb con posizionamento "Affiancato" la proprietà float sul valore none, considerando che, a default, viene utilizzato un "float: left".

In questo senso possiamo agire in due modi diversi.

a. **Direttamente dallo style editor di Passweb.**

   In questo caso è sufficiente impostare la proprietà "**Affiancamento**" presente nella sezione "**Allineamento e Visibilità**" dello Style Editor, sul valore "**Nessuna**"



b. **Definendo una specifica classe da inserire nel layout di sito e da assegnare poi a tutti i componenti strutturali della Griglia**.

   All'interno di questa classe dovrà ovviamente essere inserita la seguente regola

```
.uikit {
   float:none !important;
}
```

   Dove l'aggiunta dell'attributo !important garantisce che il valore assegnato tramite questa regola alla proprietà float vada effettivamente a sovrascrivere quello assegnato, a default, da Passweb a questa stessa proprietà.

Le prima soluzione è, forse, più semplice perché consente di agire mediante l'interfaccia grafica offerta dallo style editor.

La seconda è invece un po' più funzionale perché una volta definita la classe e inserita nel layout di sito si tratterà poi di ricordarsi di aggiungerla ai Componenti Passweb utilizzati per costruire la griglia così come dovranno essere aggiunte anche le altre classi previste dal framework.

Nel seguito di questa guida, adotteremo la seconda soluzione. Nulla vieta comunque di seguire la prima strada ottenendo esattamente gli stessi risultati.

Riassumendo possiamo dunque dire che: **tutti i Componenti Passweb utilizzati per costruire la struttura della Griglia (quindi il contenitore esterno e i singoli blocchi di contenuti che individuano le colonne) devono necessariamente avere Posizionamento Affiancato e "float: none"**

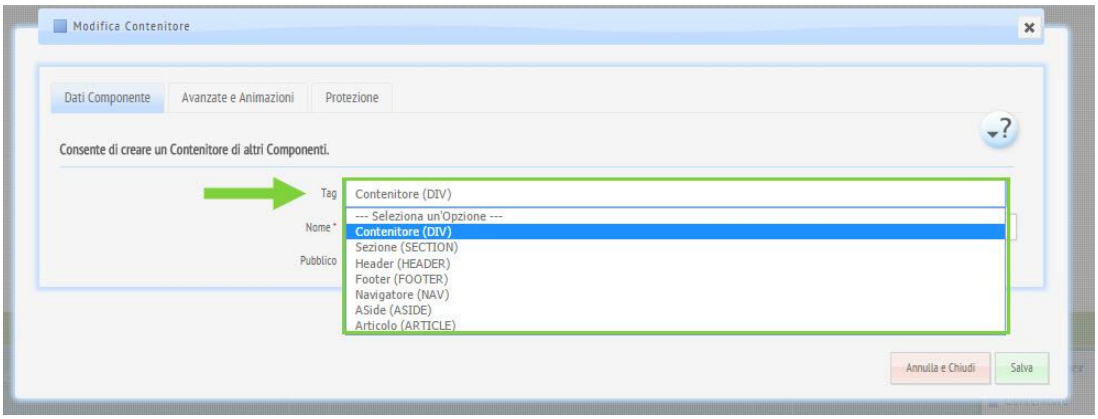
Fatta questa importante premessa vediamo ora quali Componenti utilizzare per creare la struttura della griglia e qui il discorso è estremamente semplice.

Supponiamo ad esempio di voler creare una semplice griglia di una sola riga con due colonne la prima delle quali dovrà occupare 1/3 dello spazio disponibile e la seconda i restanti 2/3.

Conosciamo già quello che deve essere il markup richiesto da uikit per ottenere questo risultato:

```
<div class="uk-grid">
   <div class="uk-width-1-3">
   .../*Contenuti della prima colonna*/
   </div>
   <div class="uk-width-2-3">
   .../*Contenuti della seconda colonna*/
   </div>
</div>
```

Sappiamo inoltre che il **Componente Contenitore** di Passweb può essere configurato per creare, a livello di markup, proprio delle < div >, anzi questa è proprio la sua configurazione base.

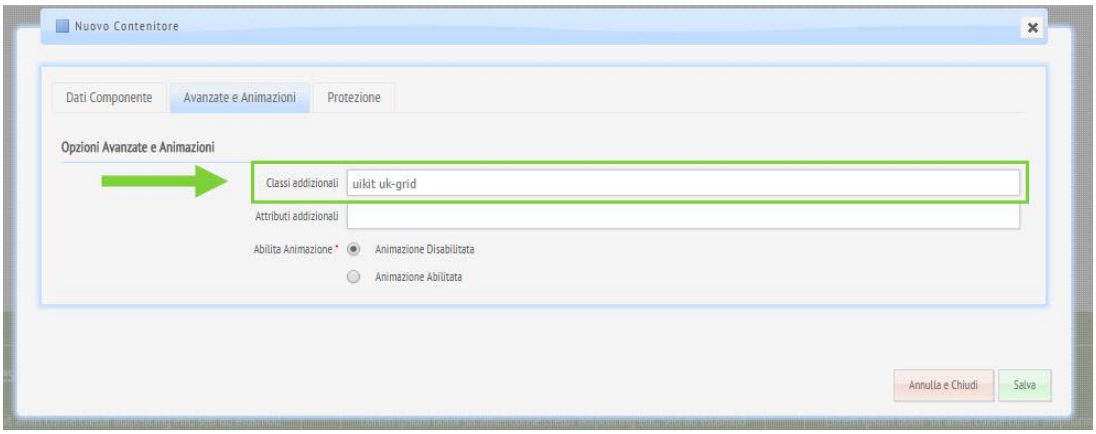


Mettendo insieme le due cose si arriva immediatamente a comprendere che **per implementare in Passweb la Griglia di uikit è sufficiente utilizzare un Componente Contenitore per il contenitore esterno della griglia all'interno del quale andare poi ad inserire altri Componenti Contenitore per gestire i singoli blocchi di contenuto e quindi le singole colonne. A ciascuno di questi Componenti Contenitore andrà poi assegnata la classe e/o l'attributo corretto utilizzando per questo i rispettivi campi presenti in configurazione del Componente**.

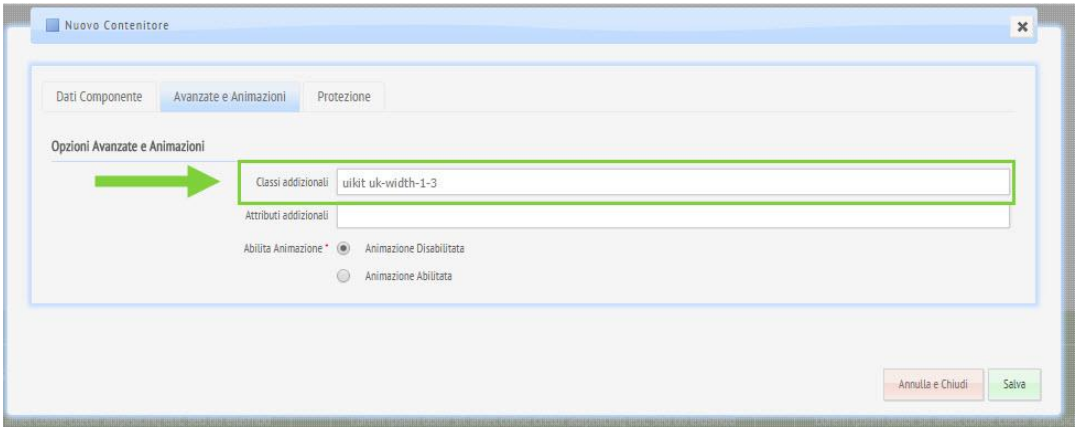
Nel caso specifico dovremo quindi

- Utilizzare un primo Componente Contenitore con posizionamento Affiancato, per gestire il contenitore esterno della griglia.

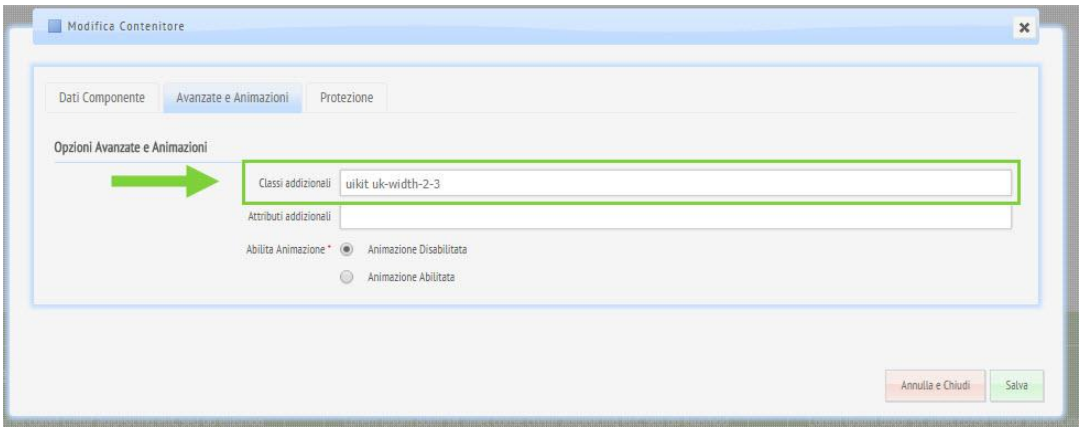
  Ad esso andremo quindi ad assegnare le due classi aggiuntive "**uikit**", appositamente creata per gestire il "float: none", e "**uk-grid**" come indicato nelle specifiche del framework



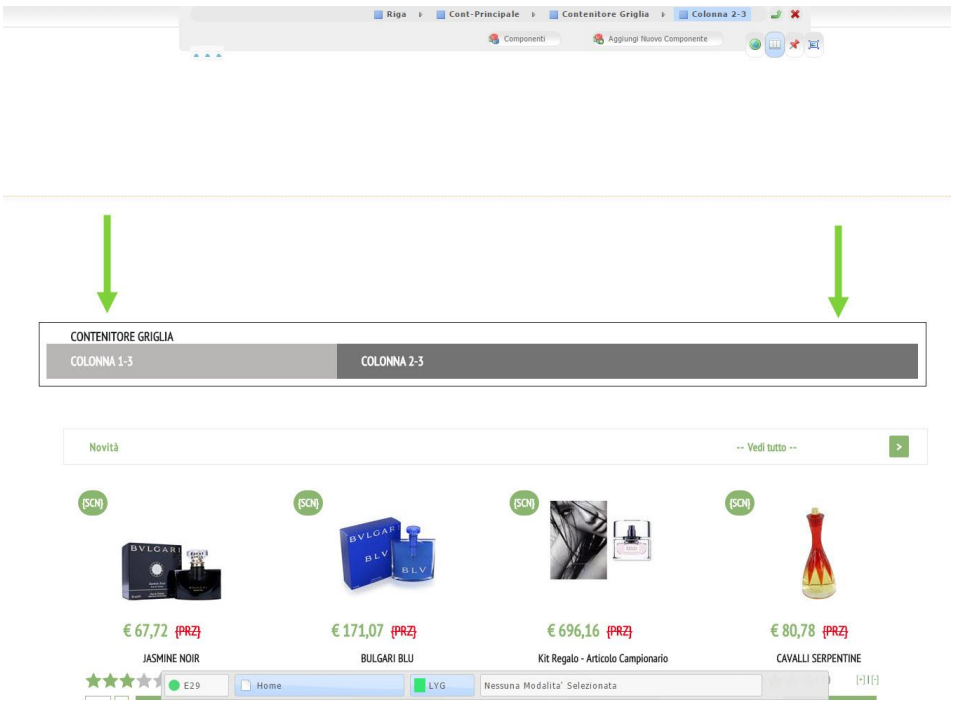
- Inserire all'interno del contenitore della griglia un primo Componente Contenitore sempre con posizionamento Affiancato al quale assegneremo la solita classe "**uikit**" e la classe "**uk-width-1-3**" necessaria, come previsto dalle specifiche del framework, per dire al componente di occupare 1/3 di tutto lo spazio a disposizione della griglia



- Inserire all'interno del contenitore della griglia un secondo Componente Contenitore sempre con posizionamento Affiancato al quale assegneremo la solita classe "**uikit**" e, questa volta, la classe "**uk-width-2-3**" necessaria, come previsto dalle specifiche del framework, per dire al componente di occupare i restanti 2/3 dello spazio a disposizione della griglia



Quello che otterremo, come già evidenziato, sarà una struttura del tipo di quella rappresentata in figura.



ATTENZIONE! I contenuti della Griglia andranno ovviamente inseriti all'interno dei Componenti Contenitore "Colonna 1-3" e "Colonna 2-3"

MODELLO ECOMMERCE 29

La griglia di uikit per come è stata esaminata fino a questo momento è, indubbiamente, un componente fluido in quanto i suoi elementi strutturali hanno larghezze in percentuale.

Non rappresenta ancora però un componente responsivo.

Per poter ottenere una griglia responsiva, come noto, non basta infatti utilizzare delle colonne con dimensione percentuale ma occorre anche che il valore di queste percentuali cambi in corrispondenza di determinati breakpoint modificando di conseguenza anche il numero di colonne per riga.

Per questa ragione nel modello di riferimento non è mai stata utilizzata una griglia di questo tipo.

Prima di analizzare come poter rendere responsiva la griglia di uikit dobbiamo prendere in considerazione un altro aspetto fondamentale della griglia ossia la gestione dello spazio tra le varie colonne.

GRID GUTTER

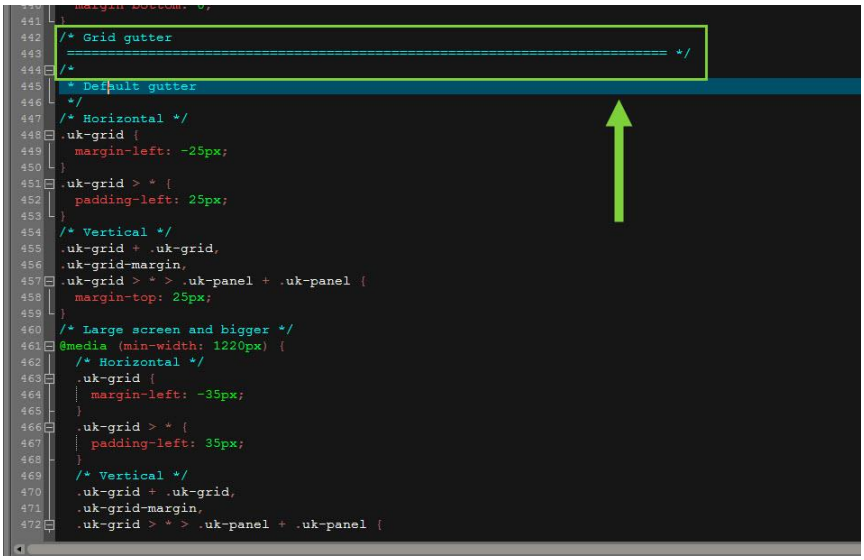
Come indicato nei capitoli iniziali di questa guida, il **Gutter** altro non è se non lo spazio di separazione tra i diversi elementi (righe e colonne) di una griglia.

Il componente Grid di uikit crea automaticamente uno spazio predefinito, in orizzontale tra le diverse colonne, e in verticale tra due griglie successive.

Lo spazio orizzontale tra le colonne è gestito mediante un padding sinistro impostato sulle colonne stesse.

Lo spazio in verticale tra due griglie successive è gestito invece con un margine alto sulla specifica griglia.

Le regole CSS per la gestione del Gutter sono localizzate tutte nel file uikit.css, all'interno della sezione "**Component: Grid**", più esattamente nella sotto sezione "**Grid Gutter**



```
/* Grid gutter
 ========================================================================== */
/*
 * Default gutter
 */
/* Horizontal */
.uk-grid {
  margin-left: -25px;
}
.uk-grid > * {
  padding-left: 25px;
}
/* Vertical */
.uk-grid + .uk-grid,
.uk-grid-margin,
.uk-grid > * > .uk-panel + .uk-panel {
  margin-top: 25px;
}
/* Large screen and bigger */
@media (min-width: 1220px) {
  /* Horizontal */
  .uk-grid {
    margin-left: -35px;
  }
  .uk-grid > * {
    padding-left: 35px;
  }
  /* Vertical */
  .uk-grid + .uk-grid,
  .uk-grid-margin,
  .uk-grid > * > .uk-panel + .uk-panel {
```

Analizzando queste regole possiamo osservare innanzitutto che:

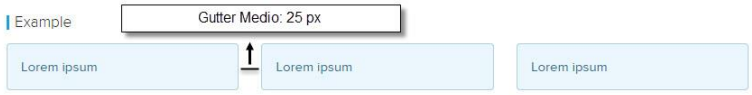
- Per viewport superiori a 1220px lo spazio tra le colonne è di 35px
- Per viewport inferiori a 1219px lo spazio tra le colonne si riduce invece a 25px

Questi sono i valori di default modificabili, come al solito, agendo direttamente all'interno del file uikit.css, dal Customizer o direttamente dallo style editor di Passweb.

Volendo è comunque possibile variare le dimensioni di questo gutter in maniera molto più semplice e uniforme sfruttando tre classi predefinite messe a disposizione da uikit che, se applicate ai giusti elementi, nello specifico **al Contenitore della griglia**, consentono di impostare automaticamente lo spazio tra le colonne su un valore medio, piccolo o al limite di azzerarlo completamente.

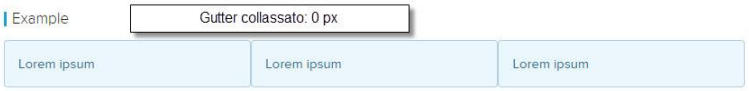
Vediamole più nel dettaglio:

- **.uk-grid-medium:** applicando questa classe **al contenitore della griglia** lo spazio tra le colonne, indipendentemente dalle dimensioni del viewport, verrà impostato su di un valore medio, dove per medio si intende, a default, un valore pari a **25px**.



- **.uk-grid-small:** applicando questa classe **al contenitore della griglia** lo spazio tra le colonne, indipendentemente dalle dimensioni del viewport, verrà impostato su di un valore piccolo, dove per piccolo si intende, a default, un valore pari a **10px**

- **.uk-grid-collapse :** applicando questa classe **al contenitore della griglia** lo spazio tra le colonne, indipendentemente dalle dimensioni del viewport, verrà completamente azzerato



ATTENZIONE! I valori di default corrispondenti al gutter medio e piccolo possono essere personalizzati agendo direttamente all'interno del file uikit.css

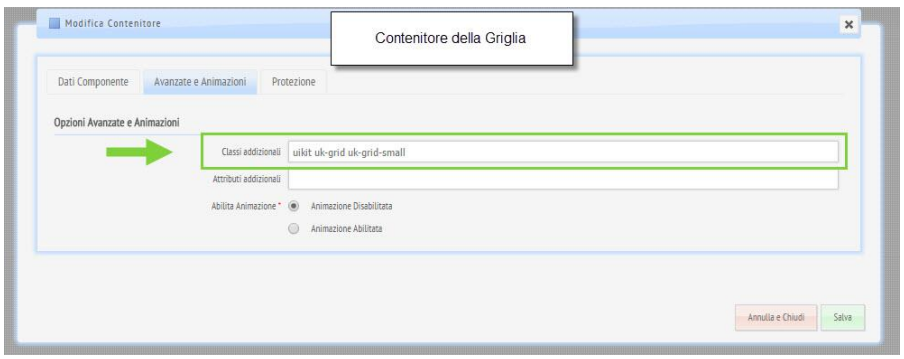
Supponendo dunque di voler realizzare una griglia con tre distinte colonne di contenuti ciascuna delle quali occupi 1/3 dello spazio disponibile separate l'una dall'altra da un gutter piccolo, dovremo utilizzare un markup di questo tipo

```
<div class="uk-grid uk-grid-small">
   <div class="uk-width-1-3">
   .../*Contenuti della prima colonna*/
   </div>
   <div class="uk-width-1-3">
   .../*Contenuti della seconda colonna*/
   </div>
   <div class="uk-width-1-3">
   .../*Contenuti della terza colonna*/
   </div>
</div>
```

GUTTER IN PASSWEB

In Passweb, per variare lo spazio di separazione tra le colonne di una Griglia impostandolo sul valore medio, piccolo o, al limite, azzerandolo completamente, **è sufficiente assegnare una delle tre classi indicate nel capitolo precedente, al Componente utilizzato per creare il contenitore della griglia.**

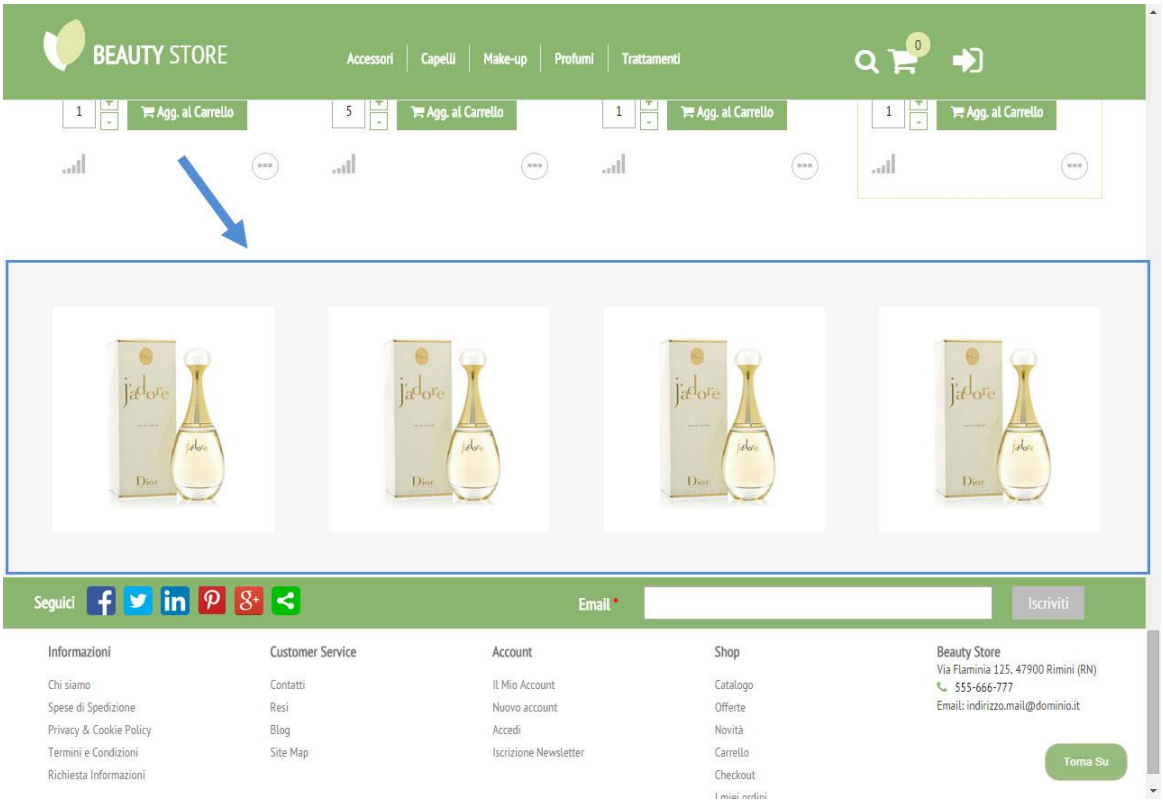
Supponendo, ad esempio, di voler impostare il gutter di una griglia sul valore "piccolo", in fase di configurazione del Componente Passweb utilizzato per definire il contenitore della griglia, oltre alle classi .uikit e .uk-grid dovremo aggiungere anche la classe **.uk-grid-small**



MODELLO ECOMMERCE 29

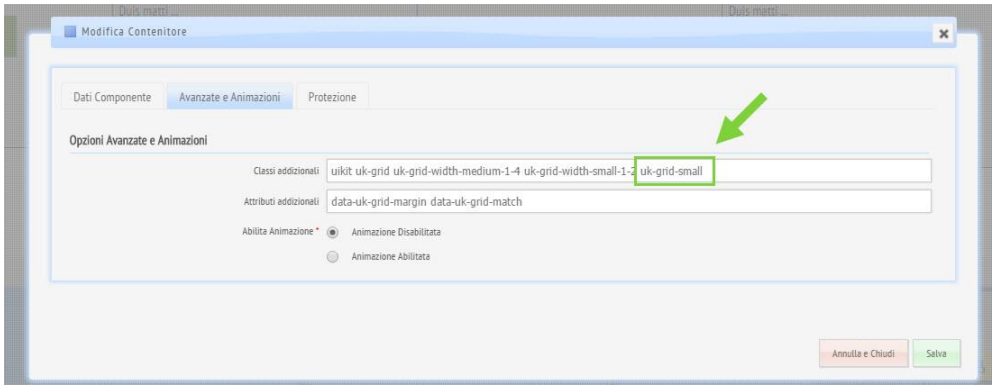
E' possibile trovare un esempio di gestione del gutter orizzontale nella **Home Page** del modello di riferimento.

La griglia in questione è quella utilizzata per gestire i banner degli articoli nella parte bassa della pagina immediatamente al di spora del piede.



Osservando la configurazione del Componente "Griglia-Banner-Articoli" utilizzato per gestire il contenitore della griglia, è possibile notare come tra le classi aggiuntive ad esso assegnate (e necessarie per implementare la griglia nel sul complesso) sia presente anche la classe

uk-grid-small



che come noto consente di ridurre a 10px lo spazio tra le colonne.

E' quindi possibile verificare come inserendo al posto della suddetta classe, la classe

- **uk-grid-medium** → lo spazio tra le colonne aumenti passando da 10 a 25 px riducendo di conseguenza la dimensione delle immagini inserite all'interno delle varie colonne
- **uk-grid-collapse** → lo spazio tra le colonne diminuisca passando da 10 a 0 px aumentando di conseguenza la dimensione delle immagini inserite all'interno delle varie colonne

Per maggiori informazioni sulle altre classi applicate a questo componente si vedano anche i successivi capitoli della guida

GRIGLIA RESPONSIVA – 1

La griglia di uikit per come è stata strutturata ed utilizzata fino a questo momento si dimostra essere un componente perfettamente fluido.

I suoi elementi strutturali utilizzano infatti delle dimensioni in percentuale tali per cui diminuendo o aumentando la dimensione del viewport anche questi diminuiranno o aumenteranno proporzionalmente la loro larghezza.

Come sappiamo però il solo utilizzo di larghezze percentuali non è sufficiente per realizzare layout responsivi. Per ottenere questo risultato è necessario che, in corrispondenza di diverse dimensioni del viewport, si modifichi anche il numero di colonne presenti su di una stessa riga.

Passando ad esempio dallo schermo di un pc ad un tablet il numero di colonne su di una stessa riga dovrebbe passare, ad esempio, da 4 a 2 fino ad arrivare sugli smartphone a linearizzare completamente il layout disponendo quindi una sola colonna per riga.

In questo senso uikit mette a disposizione dell'utente un certo numero di classi che soddisfano appieno queste esigenze (griglia fluida e responsiva).

Tali classi lavorano sostanzialmente come le classi .uk-width-* precedentemente esaminate (si veda il capitolo "*Utilizzo Base*" della Griglia), hanno nel loro nome i termini small, medium e large e, soprattutto, **essendo legate ad apposite media query verranno applicate solo ed esclusivamente in corrispondenza di certi breakpoint.**

Vediamole più nel dettaglio:

CLASSE	DESCRIZIONE
.uk-width-*	Le proprietà di questa classe verranno applicate per una qualsiasi larghezza del viewport. Le colonne della grigli si posizioneranno una a fianco all'altra
.uk-width-small-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 480px** Per larghezze del viewport inferiori ai 480px la griglia verrà linearizzata e le colonne si disporranno una sopra l'altra
.uk-width-medium-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 768px** Per larghezze del viewport inferiori ai 768px la griglia verrà linearizzata e le colonne si disporranno una sopra l'altra
.uk-width-large-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 960px** Per larghezze del viewport inferiori ai 960px la griglia verrà linearizzata e le colonne si disporranno una sopra l'altra

ATTENZIONE! il carattere * dovrà essere sostituito dal solito sistema di due numeri il primo dei quali indica quante colonne dovranno essere effettivamente occupate da quello specifico componente rispetto al totale in cui si è pensato di suddividere la griglia, mentre il secondo indica proprio questo totale.

ATTENZIONE! le dimensioni in pixel corrispondenti alle sigle small, medium e large (in sostanza i diversi breakpoint) possono essere personalizzate agendo da Customizer oppure operando direttamente all'interno del file uikit.css

Ovviamente è possibile applicare ad un blocco di contenuti più di una classe contemporaneamente, anzi solitamente questa è la prassi in quanto è proprio in questo modo che è possibile specificare quante colonne dovranno esserci per riga in corrispondenza di diverse dimensioni del viewport.

Un esempio chiarisce meglio questi concetti. Prima però di passare ad analizzare questo esempio c'è un'altra considerazione importante da fare.

Nel momento in cui le colonne di una griglia dovessero disporsi su più righe diverse potrebbe nascere il problema della separazione verticale tra i vari blocchi di contenuti su righe diverse.

Il gutter precedentemente analizzato infatti gestisce, come evidenziato, lo spazio di separazione orizzontale tra le colonne e verticale tra griglie successive. Non viene quindi preso in considerazione lo spazio verticale i diversi blocchi di contenuti disposti, all'interno di una stessa griglia, su righe diverse.

Per risolvere questo problema e creare quindi un margine tra blocchi di contenuti disposti, all'interno di una stessa griglia, su righe diverse è necessario aggiungere al Contenitore della Griglia **l'attributo data-uk-grid-margin**

ATTENZIONE! in assenza dell'**attributo data-uk-grid-margin** impostato sul Contenitore della Griglia eventuali blocchi di contenuti disposti su righe successive saranno attaccati uno all'altro (nessun gutter verticale). Assegnando invece questo attributo, lo spazio di separazione verticale tra questi elementi seguirà esattamente le stesse regole del gutter orizzontale.

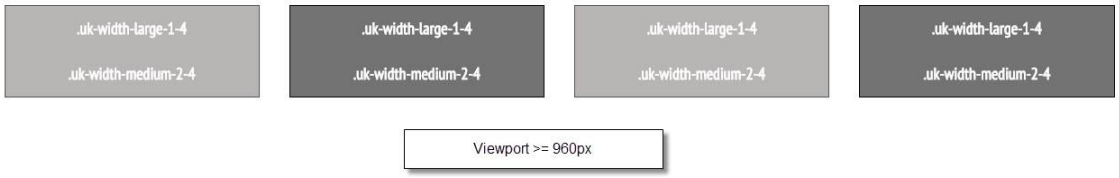
Prendiamo ora in considerazione l'esempio di cui parlavamo prima e supponiamo di dover realizzare una griglia che si comporti in questo modo:

- per viewport **large** (maggiori o uguali a 960px) 4 blocchi di contenuto di ugual larghezza disposti tutti su di una stessa riga
- per viewport **medium** (maggiori o uguali a 768px) 4 blocchi di contenuto di ugual larghezza disposti su due righe (2 blocchi per ogni riga)
- in tutti gli altri casi (viewport inferiori a 768px) la griglia dovrà linearizzarsi. I 4 blocchi di contenuto dovranno quindi disporsi uno sotto l'altro in 4 distinte righe e coprire ciascuno l'intera larghezza della griglia.
- Nel momento in cui la griglia si strutturi su più righe, il gutter verticale tra di esse dovrà comportarsi esattamente come il gutter orizzontale

Considerando quanto detto fino ad ora, per ottenere questo risultato dovremo utilizzare un markup di questo tipo:

```
<div class="uk-grid" data-uk-grid-margin>
   <div class="uk-width-large-1-4 uk-width-medium-2-4">
   .../*Contenuti della prima colonna*/
   </div>
   <div class="uk-width-large-1-4 uk-width-medium-2-4">
   .../*Contenuti della seconda colonna*/
   </div>
   <div class="uk-width-large-1-4 uk-width-medium-2-4">
   .../*Contenuti della terza colonna*/
   </div>
   <div class="uk-width-large-1-4 uk-width-medium-2-4">
   .../*Contenuti della quarta colonna*/
   </div>
</div>
```

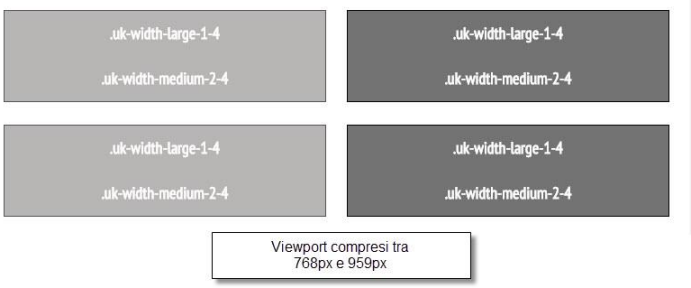
In questo modo, relativamente ai blocchi di contenuto interni alla griglia, per dimensioni del viewport **maggiori o uguali a 960px** verranno applicate le proprietà relative alla classe .**uk-width-large-1-4** e otterremo quindi 4 colonne su di una stessa riga



Ora se avessimo applicato solo la classe .uk-width-large-1-4, come indicato nella precedente tabella, per dimensioni del viewport inferiori a 960px la griglia si sarebbe dovuta linearizzare e i 4 blocchi di contenuti si sarebbero dovuti disporre uno sotto l'altro.

Considerando però che oltre a questa abbiamo utilizzato anche la classe **.uk-width-medium-2-4** allora per **dimensioni del viewport comprese tra 768px e 959px** verranno applicate le proprietà relative a questa classe per cui in questo range i 4 blocchi di contenuto si disporranno su due distinte righe e ognuno di essi avrà una larghezza pari a metà della riga.

Considerando inoltre che al Contenitore della griglia è stato applicato l'attributo **data-uk-grid-margin** il gutter verticale tra i diversi blocchi di contenuto sarà uguale al gutter orizzontale

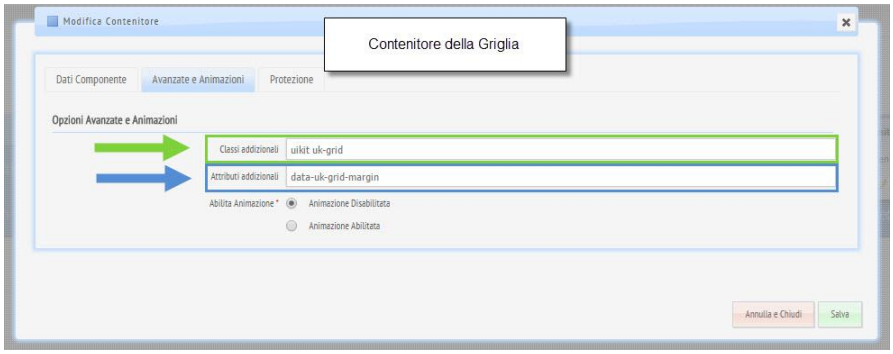


Resta ora da capire cosa succede per dimensioni del viewport inferiori a 768px. In questo senso considerando che per queste dimensioni del viewport non è stato specificato nulla (non è stata utilizzata ne la classe .uk-width-small-* ne tanto meno la .uk-width-*), secondo quanto indicato nella precedente tabella per dimensioni inferiori a 768px la griglia si linearizzerà e i blocchi di contenuto si disporranno quindi su 4 righe distinte

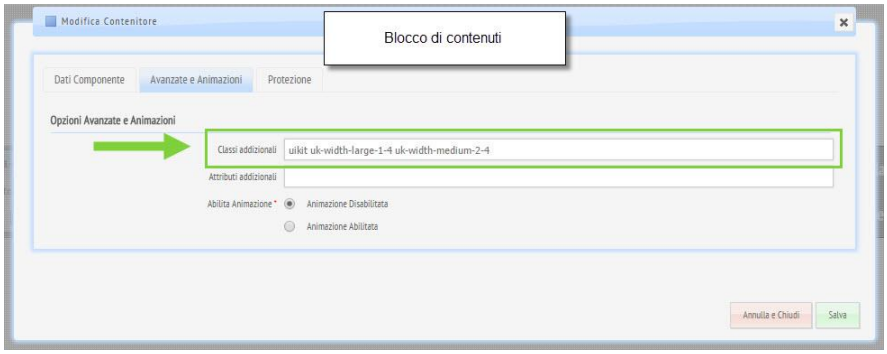
GRIGLIA RESPONSIVA IN PASSWEB – 1

Per creare in Passweb una griglia responsiva che si comporti esattamente come quella indicata nell'esempio del precedente capitolo sarà necessario utilizzare:

- Un Componente Contenitore per gestire il contenitore esterno della griglia. Per esso dovrà essere impostato un posizionamento Affiancato e dovranno essergli assegnate le solite classi .**uikit** e .**uk-grid.** Inoltre per gestire il gutter verticale sarà necessario assegnarli anche l'attributo **data-uk-grid-margin**



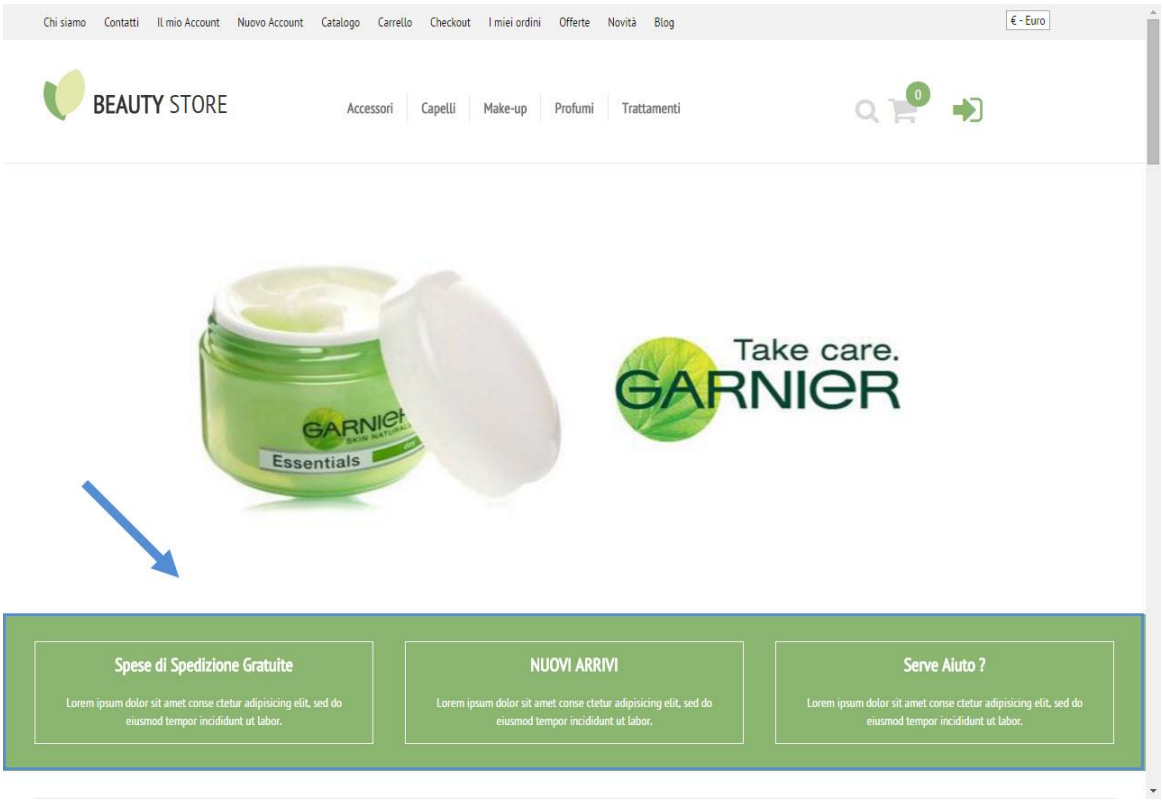
- Quattro Componenti Contenitore, inseriti all'interno del Contenitore della griglia e utilizzati per gestire i 4 distinti blocchi di contenuto. Per ciascuno di essi dovrà essere utilizzato un posizionamento Affiancato e dovranno essergli assegnate le classi aggiuntive .**uikit, .uk-width-large-1-4** e **.uk-width-medium-2-4,**



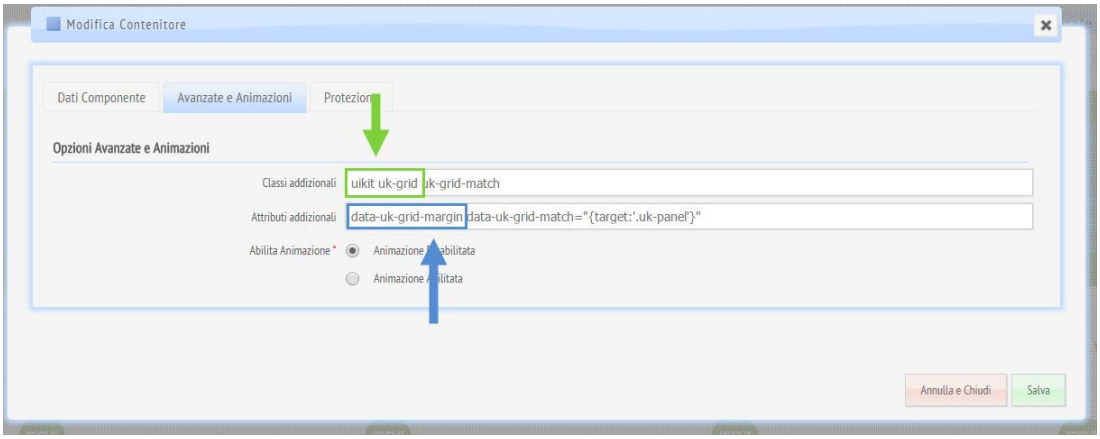
MODELLO ECOMMERCE 29

E' possibile trovare un esempio della Griglia responsiva analizzata nei precedenti capitoli nella **Home Page** del modello di riferimento.

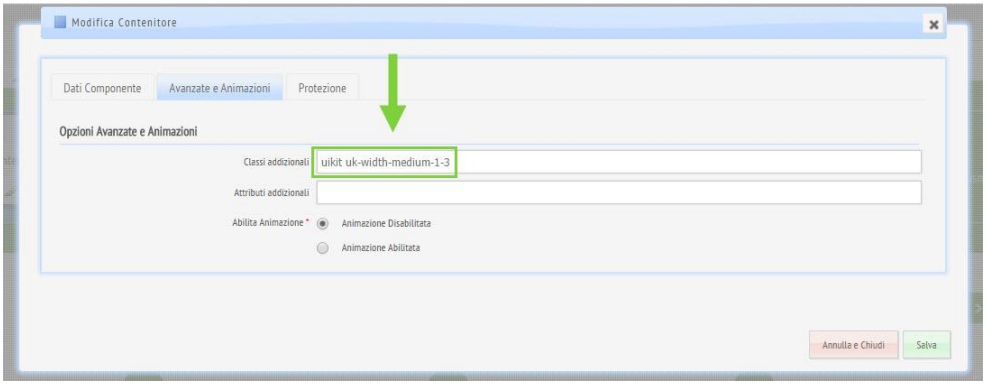
La griglia in questione è quella utilizzata per gestire i banner testuali immediatamente al di sotto dello slider presente in testata



Osservando la configurazione del Componente "Griglia-Banner-Testata" utilizzato per gestire il contenitore della griglia, è possibile notare la presenza delle classi **uikit** e **uk-grid** oltre che dell'attributo **data-uk-grid-margin** necessario per gestire il gutter verticale nel momento in cui i contenuti delle 3 colonne andranno a disporsi uno sopra l'altro



All'interno di questo componente sono stati inseriti te distinti Componenti Contenitore configurati come in figura



La sola classe **uk-width-medium-1-3** assegnata a ciascuno di questi contenitori fa si che:

- Per viewport maggiori o uguali a 768px questi occupino esattamente 1/3 della larghezza della griglia disponendosi dunque su di una sola riga.
- Per viewport inferiori a 768px la griglia si linearizzerà, i contenitori occuperà il 100% dello spazio a loro disposizione disponendosi esattamente uno sopra l'altro.

Sostituendo alla classe **uk-width-medium-1-3** impostata sui contenitori interni alla griglia, ad esempio, la classe **uk-width-small-1-3** è possibile verificare come il breakpoint si sposti sul valore 480px e come quindi la linearizzazione avvenga solo per viewport inferiori a tale valore

GRIGLIE ANNIDATE

Una volta compreso come fare a gestire una griglia, sia in termini generali che all'interno di Passweb, gestire poi un sistema di **griglie annidate** è estremamente semplice.

Si tratta infatti di creare una prima griglia con un certo numero di colonne, seguendo esattamente quanto visto nei precedenti capitoli di questa guida, e di inserire poi all'interno di una (o più) di queste colonne non dei semplici contenuti ma, innanzitutto, la struttura di un'altra griglia.

Utilizzando ad esempio un markup di questo tipo

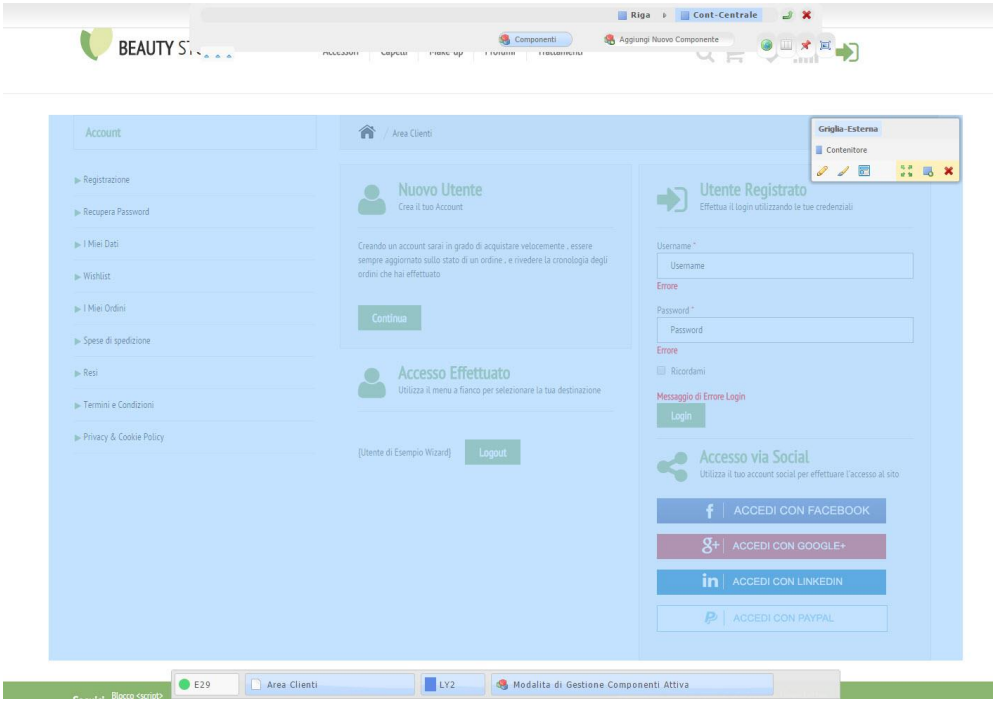
```
<div class="uk-grid"> /*Griglia esterna*/
    <div class="uk-width-1-2">
     ... /*Contenuti della prima colonna*/
    </div>
    <div class="uk-width-1-2">
        <div class="uk-grid"> /*Griglia interna alla seconda colonna*/
            <div class="uk-width-1-2">
            ... /*Contenuti della prima colonna della seconda griglia*/
            </div>
            <div class="uk-width-1-2">
            ... /*Contenuti della seconda colonna della seconda griglia*/
            </div>
        </div>
    </div>
</div>
```

otterremo una griglia esterna di due colonne dove nella prima colonna inseriamo un semplice blocco di contenuti mentre all'interno della seconda colonna utilizziamo una seconda griglia mediante che ci consente di dividerla in ulteriori due colonne.

MODELLO ECOMMERCE 29

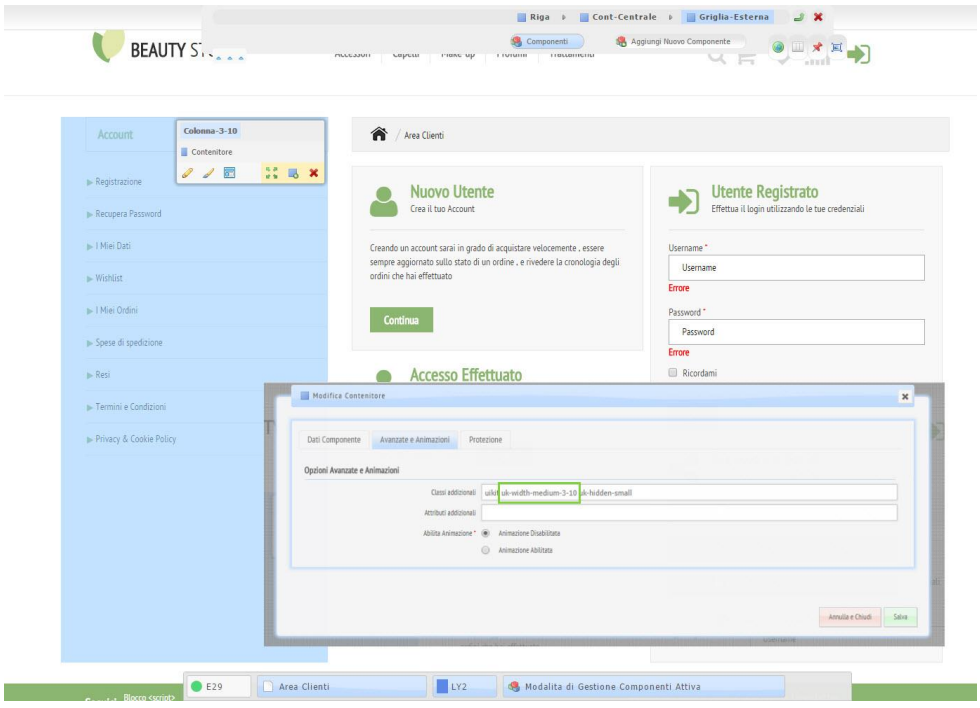
E' possibile trovare un esempio di Griglie annidate nella pagina **Area Clienti** del modello di riferimento

All'interno di questa pagina è stata implementata una prima griglia (componente "**Griglia-Esterna**")

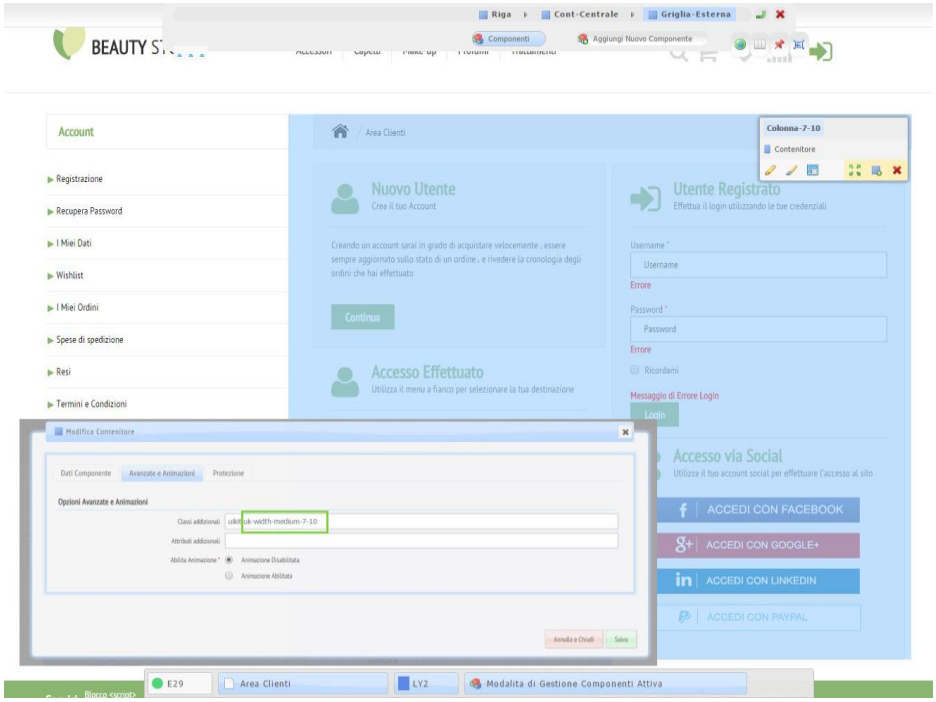


all'interno della quale sono state inserite due colonne:

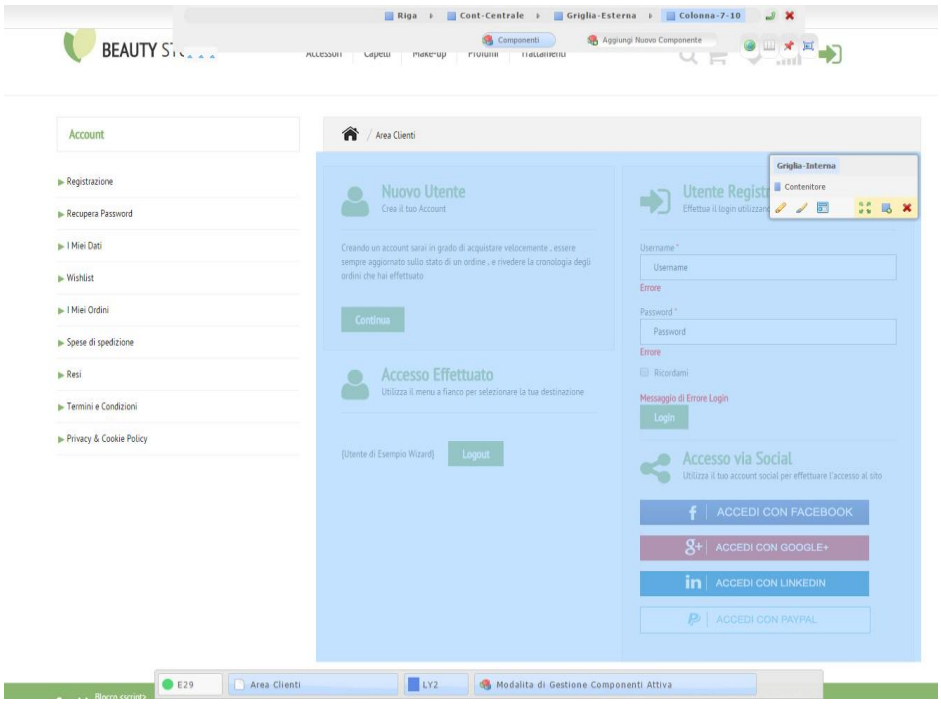
- La colonna di sinistra che occupa i 3/10 della griglia esterna (classe **uk-width-medium-3-10**)



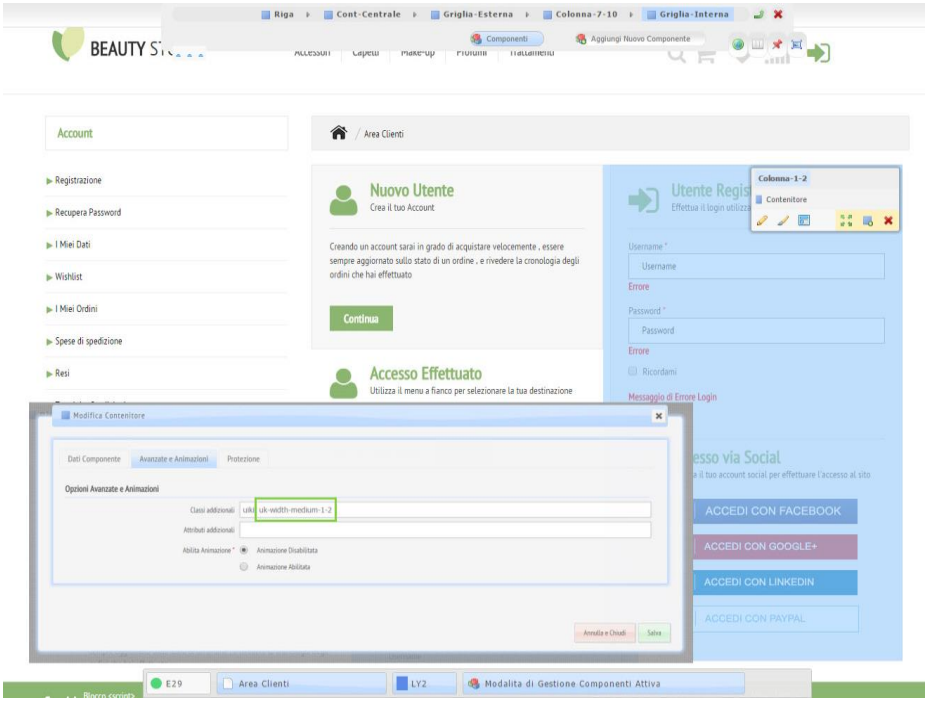
- La colonna di destra che occupa i restanti 7/10 dello spazio disponibile (classe **uk-width-medium-3-10**)



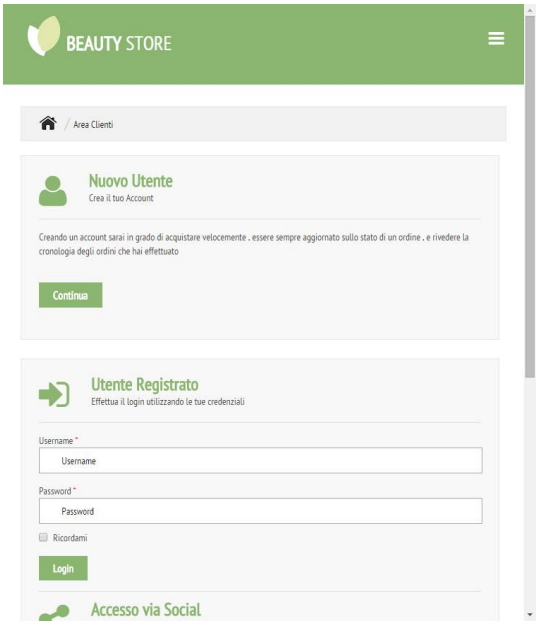
All'interno della colonna destra è stata poi inserita una seconda griglia (componente "**Griglia Interna**")



all'interno della quale sono state inserite due colonne ciascuna delle quali occupa metà dello spazio a disposizione della Griglia Interna (classe **uk-width-medium-1-2**)

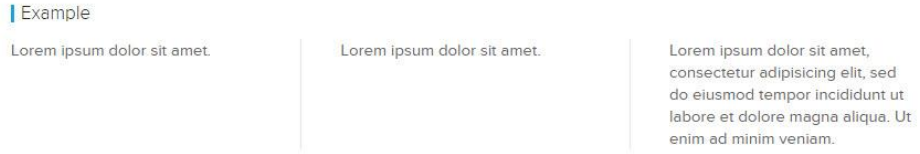


Considerato l'utilizzo delle sole classi responsive **uk-width-medium-*** tutto il layout si linearizza in corrispondenza di viewport inferiori a 768px



ALTEZZA DELLE COLONNE

La griglia di uikit utilizza il modulo CSS3 Flexible Box Layout (più noto, semplicemente, come Flexbox). Grazie ad esso **le colonne disposte su di una stessa riga assumono tutte, automaticamente, la stessa altezza** (che sarà quindi stabilita da quella con il maggior numero di contenuti)



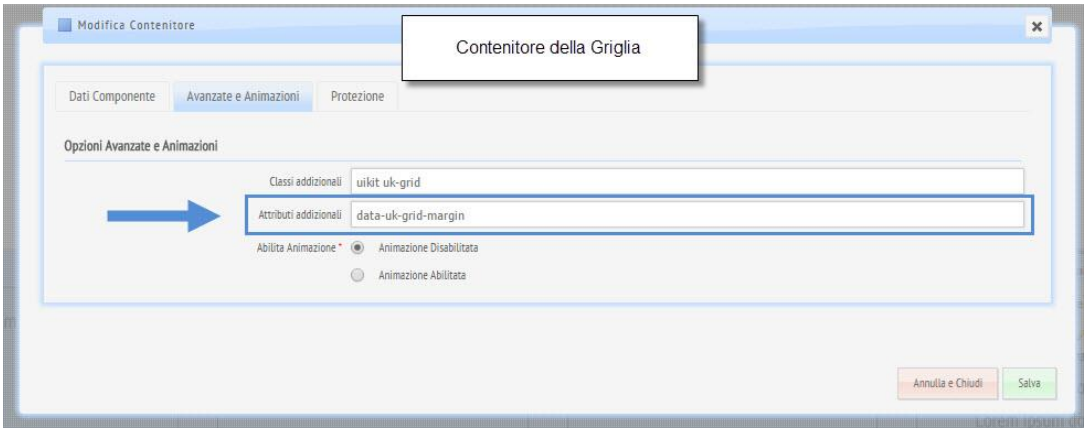
Il problema in questo senso è rappresentato dal fatto che non tutte le versioni dei browser attualmente in commercio supportano correttamente le specifiche del CSS3 (e tra queste anche il modulo Flexbox).

Per garantire quindi che le colonne di una stessa riga della griglia mantengano tutte la stessa altezza anche nei browser più datati è sufficiente aggiungere **al contenitore esterno della griglia l'attributo data-uk-grid-match**

Il markup del contenitore esterno della griglia dovrà quindi essere il seguente

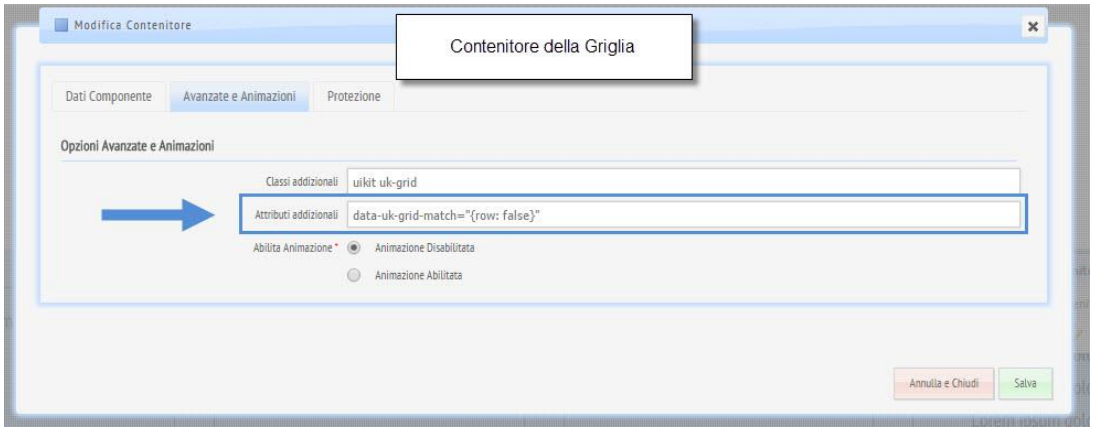
```
<div class="uk-grid" data-uk-grid-match>...</div>
```

In Passweb questo attributo andrà inserito nell'apposito campo della maschera di configurazione del Componente utilizzato per definire il contenitore della griglia



Nel momento in cui le colonne dovessero disporsi su più righe quanto detto fino ad ora potrebbe non essere più valido nel senso che all'interno di ogni singola riga le colonne manterranno tutte la stessa altezza ma tra righe diverse potremmo anche avere colonne con altezza diversa.

Se l'esigenza dovesse essere quella di ottenere colonne con la stessa altezza anche su righe diverse sarà sufficiente assegnare al contenitore della griglia l'attributo **data-uk-grid-match="{row: false}"**

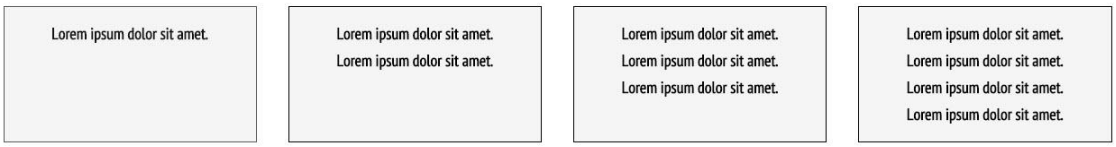


Un'ultima osservazione interessante da fare, relativamente a questo argomento, è quella che riguarda il caso in cui **vengano inseriti all'interno delle colonne dei componenti Pannello** (per maggiori informazioni sul componente Pannello di uikit si vedano anche i successivi capitoli di questa guida).

In queste condizioni potremo gestire l'altezza di questi Pannelli in due modi diversi:

- Ogni Pannello mantiene una sua specifica altezza dettata esclusivamente da quelli che sono i contenuti inseriti all'interno del pannello stesso

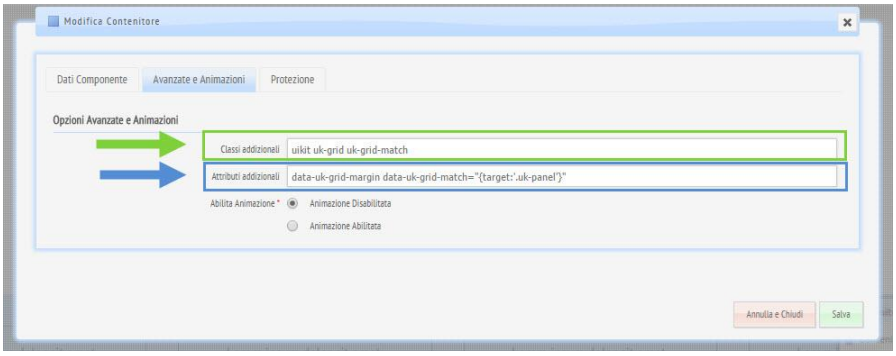
- I Pannelli inseriti all'interno delle diverse colonne hanno tutti la stessa altezza esattamente come avviene già per le colonne della griglia



Nel primo caso (Pannelli con altezza diverse) non è necessario fare nulla essendo questo il comportamento di default

Se invece l'esigenza dovesse essere quella di mantenere la stessa altezza anche per i Pannelli inseriti all'interno delle diverse colonne, sarà necessario assegnare al contenitore della griglia la classe **.uk-grid-match**

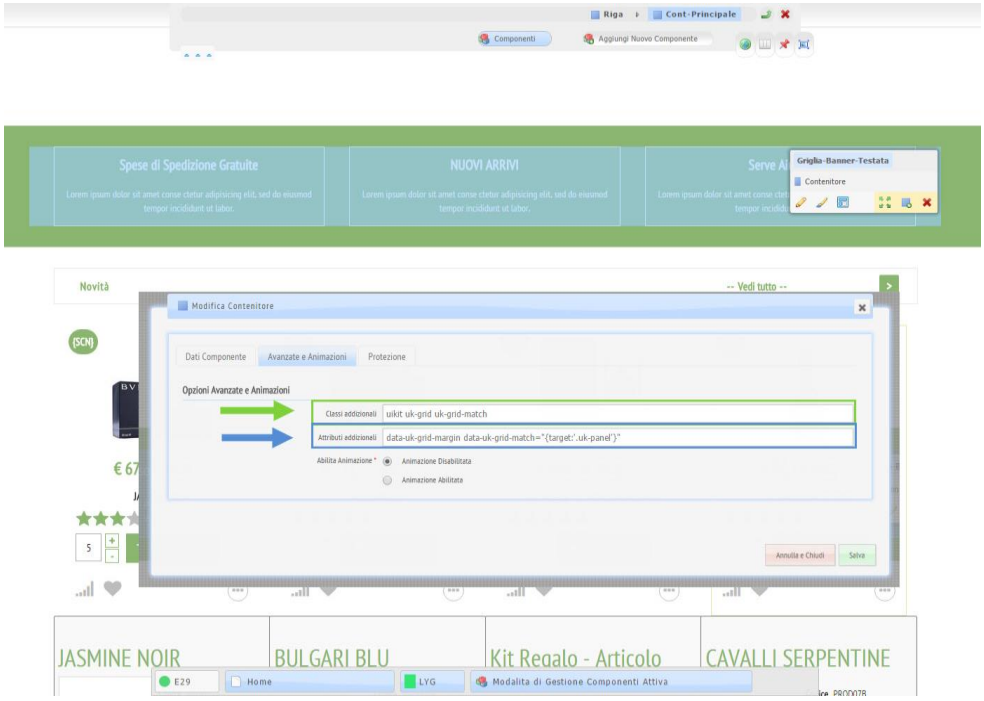
Infine, per garantire questo stesso funzionamento anche su browser più datati oltre alla suddetta classe sarà necessario assegnare al contenitore della griglia anche il seguente attributo **data-uk-grid-match="{target:'.uk-panel'}"**



MODELLO ECOMMERCE 29

E' possibile trovare un esempio di Griglia responsiva configurata in maniera tale che i Pannelli inseriti all'interno delle colonne mantengano tutti la stessa altezza nella **Home Page** del modello di riferimento.

La griglia in questione è quella utilizzata per gestire i banner testuali immediatamente al di sotto dello slider presente in testata



E' possibile verificare come aggiungendo un'ulteriore riga di testo ad uno dei tre pannelli, aumentando quindi la sua altezza, venga automaticamente aumentata anche l'altezza degli altri due pannelli

CONFIGURAZIONE – 2

Considerando quanto visto fino a questo momento dovrebbe ormai essere chiaro che per poter creare e configurare la Griglia di uikit è necessario, oltre ad utilizzare il corretto markup HTML, assegnare specifiche classi tanto al Contenitore della Griglia quanto ai singoli blocchi di contenuto, quindi alle singole colonne, presenti al suo interno.

In particolare la classe .uk-width-*, assegnata ad ogni singolo blocco di contenuti, ci permette di stabilire esattamente la larghezza che quello specifico blocco dovrà assumere all'interno della griglia.

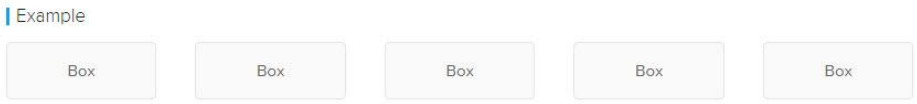
Operare in questo modo è sicuramente utile nel momento in cui dovessimo gestire all'interno della nostra griglia blocchi di contenuti di larghezza diversa.

Nel caso in cui dovessimo invece gestire una griglia con tante colonne tutte della stessa larghezza, anziché andare ad applicare la stessa classe ad ogni singola colonna si potrebbe agire in modo molto più rapido **assegnando direttamente al contenitore della griglia** una delle classi **.uk-grid-width-*,** dove il carattere * dovrà, ovviamente, essere sostituito dalla solita coppia di numeri.

Vediamo quindi il significato di queste classi (considerando sempre che la griglia di uikit è in grado di supportare 1,2,3,4,5,6 o 10 colonne)

CLASSE	DESCRIZIONE
.uk-grid-width-1-2	Consente di dividere automaticamente la griglia in 2 colonne. Ogni blocco di contenuti inserito all'interno della griglia andrà quindi ad occupare, automaticamente, metà della larghezza complessiva della griglia
.uk-grid-width-1-3	Consente di dividere automaticamente la griglia in 3 colonne. Ogni blocco di contenuti inserito all'interno della griglia andrà quindi ad occupare, automaticamente, 1/3 della larghezza complessiva della griglia
.uk-grid-width-1-4	Consente di dividere automaticamente la griglia in 4 colonne. Ogni blocco di contenuti inserito all'interno della griglia andrà quindi ad occupare, automaticamente, 1/4 della larghezza complessiva della griglia
.uk-grid-width-1-5	Consente di dividere automaticamente la griglia in 5 colonne. Ogni blocco di contenuti inserito all'interno della griglia andrà quindi ad occupare, automaticamente, 1/5 della larghezza complessiva della griglia
.uk-grid-width-1-6	Consente di dividere automaticamente la griglia in 6 colonne. Ogni blocco di contenuti inserito all'interno della griglia andrà quindi ad occupare, automaticamente, 1/6 della larghezza complessiva della griglia
.uk-grid-width-1-10	Consente di dividere automaticamente la griglia in 10 colonne. Ogni blocco di contenuti inserito all'interno della griglia andrà quindi ad occupare, automaticamente, 1/10 della larghezza complessiva della griglia

Supponendo ora di voler realizzare una griglia con 5 colonne tutte della stessa larghezza



potremmo quindi utilizzare un markup di questo tipo

```
<div class="uk-grid .uk-grid-width-1-5">
   <div>
   .../*Contenuti della prima colonna*/
   </div>
   <div>
   .../*Contenuti della seconda colonna*/
   </div>
   <div>
   .../*Contenuti della terza colonna*/
   </div>
   <div>
   .../*Contenuti della quarta colonna*/
   </div>
   <div>
   .../*Contenuti della quinta colonna*/
   </div>
</div>
```

Come si può osservare dunque non è più necessario assegnare una specifica classe ad ogni colonna in quanto la classe **.uk-grid-width-1-5** applicata direttamente al contenitore della griglia farà si che ogni blocco di contenuti presente al suo interno assuma automaticamente una larghezza pari ad 1/5 della larghezza complessiva della griglia stessa.

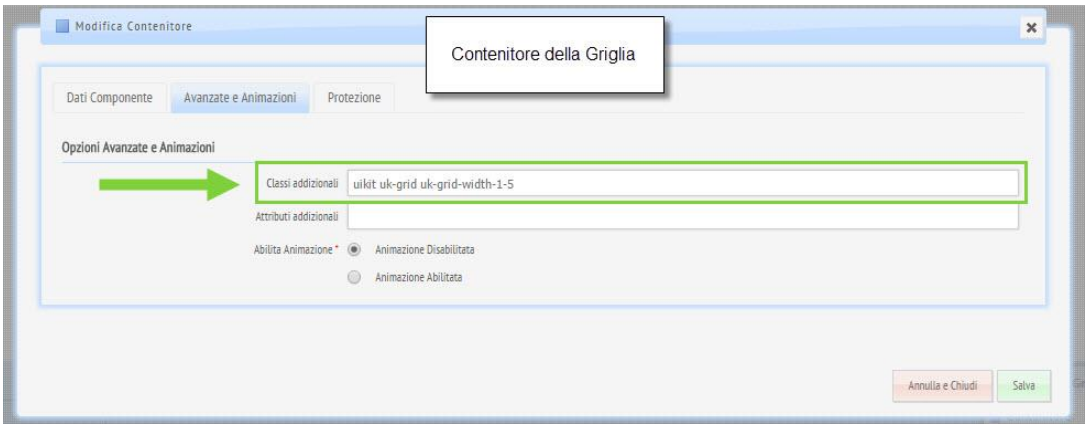
LA GRIGLIA IN PASSWEB

Sulla base di quanto detto nel precedente capitolo per implementare, in Passweb, una griglia con 5 colonne tutte della stessa larghezza sarà necessario:

- Utilizzare un primo Componente Contenitore **con posizionamento Affiancato**, per gestire il contenitore esterno della griglia.

  Ad esso andranno poi assegnate le seguenti classi aggiuntive:

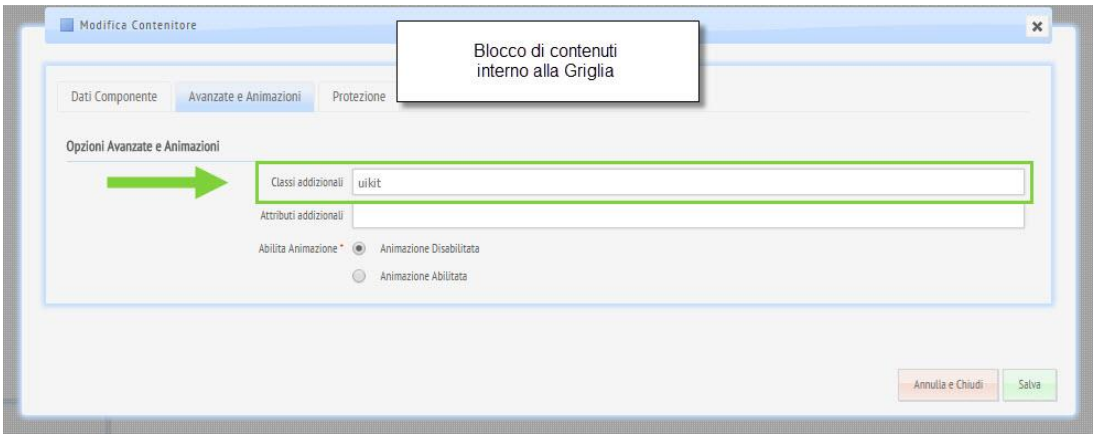
  - **uikit -** appositamente creata per gestire il "float: none"
  - **uk-grid** – consente di identificare il Contenitore della griglia
  - **uk-grid-width-1-5 –** consente di suddividere automaticamente la griglia in 5 colonne



- Inserire all'interno del contenitore della griglia 5 distinti Componenti Contenitore con posizionamento Affiancato.

  A ciascuno di questi Componenti andrà assegnata **la sola classe uikit** necessaria per gestire il "float: none".

  La larghezza di questi Componenti Contenitore sarà determinata dalla specifica classe assegnata nel punto precedente al Contenitore della Griglia



MODELLO ECOMMERCE 29

L'applicazione, al contenitore della griglia, delle classi **uk-grid-width-*** al pari delle classi **uk-width-*** applicate alle singole colonne non è sufficiente a rendere la griglia responsiva.

Per questa ragione nel modello di riferimento non sono state implementate griglie di questo tipo.

GRIGLIA RESPONSIVA – 2

Le classi .uk-grid-width-* applicate al contenitore della griglia consentono, come visto, di suddividere automaticamente la griglia stessa in un certo numero di colonne tutte esattamente della stessa larghezza.

Va detto però che tali classi, al pari di quelle del tipo .uk-width-*, non conferiscono alla griglia un comportamento responsivo

Uikit risolve anche questo problema mettendo a disposizione dell'utente un' insieme di classi da applicare sempre al contenitore della griglia, che consentono sempre di suddividere automaticamente la griglia stessa in un certo numero di colonne tutte della stessa larghezza e che, inoltre, le conferiscono anche un comportamento responsivo permettendo all'utente di indicare quante colonne dovranno essere effettivamente disposte su di una stessa riga in corrispondenza di viewport piccoli, medi, grandi ecc…

Queste nuove classi sono, in sostanza, il corrispettivo di quelle esaminate nel capitolo "*Griglia Responsiva – 1*" solo che essendo applicate non al singolo blocco di contenuti ma direttamente al contenitore della griglia, conterranno nel loro nome anche la parola grid.

Vediamole più nel dettaglio:

CLASSE – Applicata al Contenitore della Grigla	**DESCRIZIONE**
.uk-grid-width-*	Le proprietà di questa classe verranno applicate per una qualsiasi larghezza del viewport. Le colonne della griglia avranno tutte la stessa larghezza e si posizioneranno una a fianco all'altra
.uk-grid-width-small-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 480px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 480px si disporranno una sopra l'altra
.uk-grid-width-medium-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 768px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 768px si disporranno una sopra l'altra
.uk-grid-width-large-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 960px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 960px si disporranno una sopra l'altra
.uk-grid-width-xlarge-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 1220px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 1220px si disporranno una sopra l'altra

ATTENZIONE! il carattere * dovrà essere sostituito dal solito sistema di due numeri. In questo caso però il primo numero dovrà essere sempre 1 mentre il secondo variando tra 1, 2, 3, 4, 5, 6 e 10 indicherà esattamente il numero di colonne in cui dividere la griglia

ATTENZIONE! le dimensioni in pixel corrispondenti alle sigle small, medium, large e xlarge (in sostanza i diversi breakpoint) possono essere personalizzate agendo da Customizer oppure operando direttamente all'interno del file uikit.css

Anche in questo caso ovviamente è possibile applicare ad un blocco di contenuti più di una classe alla volta in maniera tale da specificare quante colonne dovranno esserci per riga in corrispondenza di diverse dimensioni del viewport.

Consideriamo, in questo senso, lo stesso esempio esaminato nel capitolo "*Griglia Responsiva – 1*" e supponiamo dunque di dover realizzare una griglia che si comporti in questo modo:

- per viewport **large** (maggiori o uguali a 960px) 4 blocchi di contenuto di ugual larghezza disposti tutti su di una stessa riga
- per viewport **medium** (maggiori o uguali a 768px) 4 blocchi di contenuto di ugual larghezza disposti su due righe (2 blocchi per ogni riga)
- in tutti gli altri casi (viewport inferiori a 768px) la griglia dovrà linearizzarsi. I 4 blocchi di contenuto dovranno quindi disporsi uno sotto l'altro in 4 distinte righe e coprire ciascuno l'intera larghezza della griglia.
- Nel momento in cui la griglia si strutturi su più righe, il gutter verticale tra di esse dovrà comportarsi esattamente come il gutter orizzontale

Per ottenere questo risultato potremo utilizzare ora un markup di questo tipo:

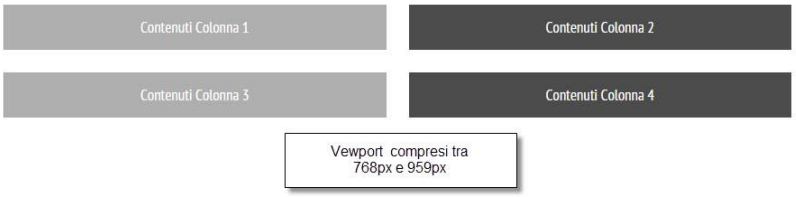
```
<div class="uk-grid uk-grid-width-large-1-4 uk-grid-width-medium-1-2" data-uk-grid-margin>
   <div>
   .../*Contenuti della prima colonna*/
   </div>
   <div>
   .../*Contenuti della seconda colonna*/
   </div>
   <div>
   .../*Contenuti della terza colonna*/
   </div>
   <div>
   .../*Contenuti della quarta colonna*/
   </div>
</div>
```

In questo modo per dimensioni del viewport **maggiori o uguali a 960px** verrà applicata la classe **uk-grid-width-large-1-4** definita sul contenitore della griglia; ogni colonna assumerà automaticamente una larghezza pari ad 1/4 della griglia stessa e otterremo quindi 4 blocchi di contenuto di ugual larghezza posti tutti sulla stessa riga

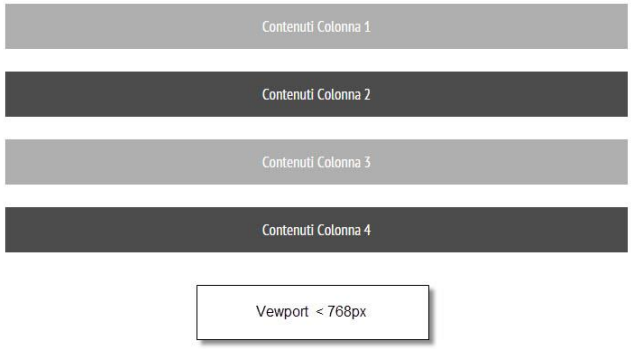
Se avessimo applicato solo la classe .uk-grid-width-large-1-4, come indicato nella precedente tabella, per dimensioni del viewport inferiori a 960px la griglia si sarebbe dovuta linearizzare e i 4 blocchi di contenuti si sarebbero dovuti disporre uno sotto l'altro.

Considerando però che oltre a questa abbiamo utilizzato, sul contenitore della griglia, anche la classe **.uk-grid-width-medium-1-2** allora per **dimensioni del viewport comprese tra 768px e 959px** verranno applicate le proprietà relative a questa classe per cui in questo range le 4 colonne della griglia assumeranno ciascuna una larghezza pari a metà della griglia stessa disponendosi su due righe distinte.

Considerando inoltre che al Contenitore della griglia è stato applicato anche l'attributo **data-uk-grid-margin** il gutter verticale tra i diversi blocchi di contenuto sarà uguale al gutter orizzontale



Infine, non essendo stato specificato nulla per dimensioni del viewport inferiori a 768px (non è stata utilizzata ne la classe .uk-gridi-width-small-* ne tanto meno la .uk-grid-width-*), in queste condizioni la griglia si linearizzerà e i blocchi di contenuto si disporranno uno sotto l'altro su 4 righe distinte



GRIGLIA RESPONSIVA IN PASSWEB – 2

Per creare in Passweb una griglia responsiva che si comporti esattamente come quella indicata nell'esempio del precedente capitolo sarà necessario:

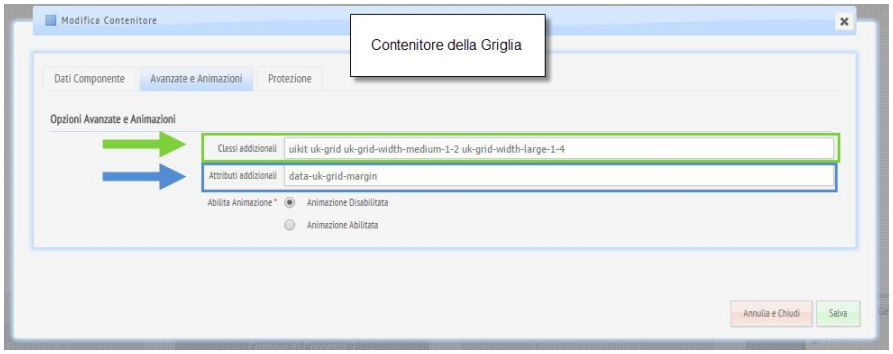
- Utilizzare un primo Componente Contenitore **con posizionamento Affiancato**, per gestire il contenitore esterno della griglia.

  Ad esso andranno poi assegnate le seguenti classi aggiuntive:

  - **uikit -** appositamente creata per gestire il "float: none"
  - **uk-grid** – consente di identificare il Contenitore della griglia
  - **uk-grid-width-large-1-4 –** consente di suddividere automaticamente la griglia, per viewport maggiori o uguali a 960px, in 4 colonne
  - **uk-grid-width-medium-1-2 –** consente di suddividere automaticamente la griglia, per viewport maggiori o uguali a 768px, in 2 colonne

  e il seguente Attributo aggiuntivo

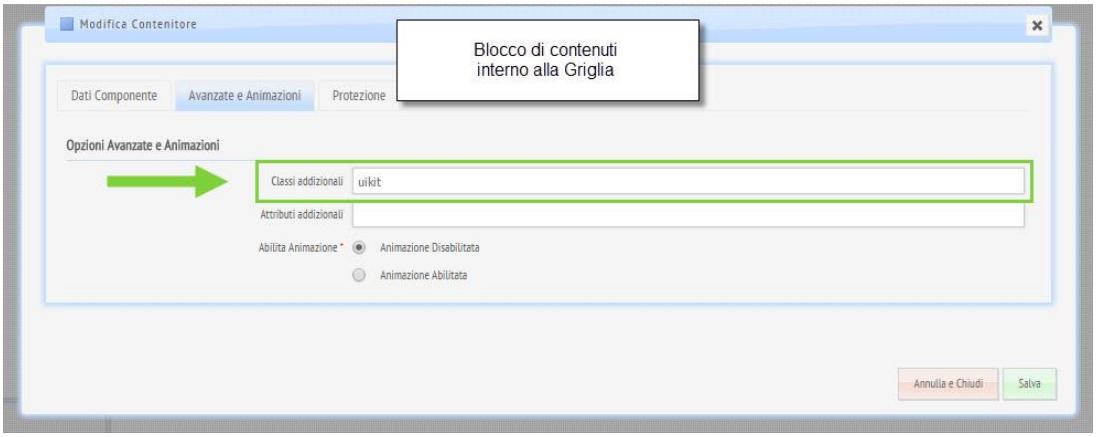
  - **data-uk-grid-margin –** consente di gestire il gutter verticale tra i diversi blocchi di contenuto



- Quattro Componenti Contenitore, inseriti all'interno del Contenitore della griglia e utilizzati per gestire i 4 distinti blocchi di contenuto.

  A ciascuno di questi Componenti andrà assegnata **la sola classe uikit** necessaria per gestire il "float: none".

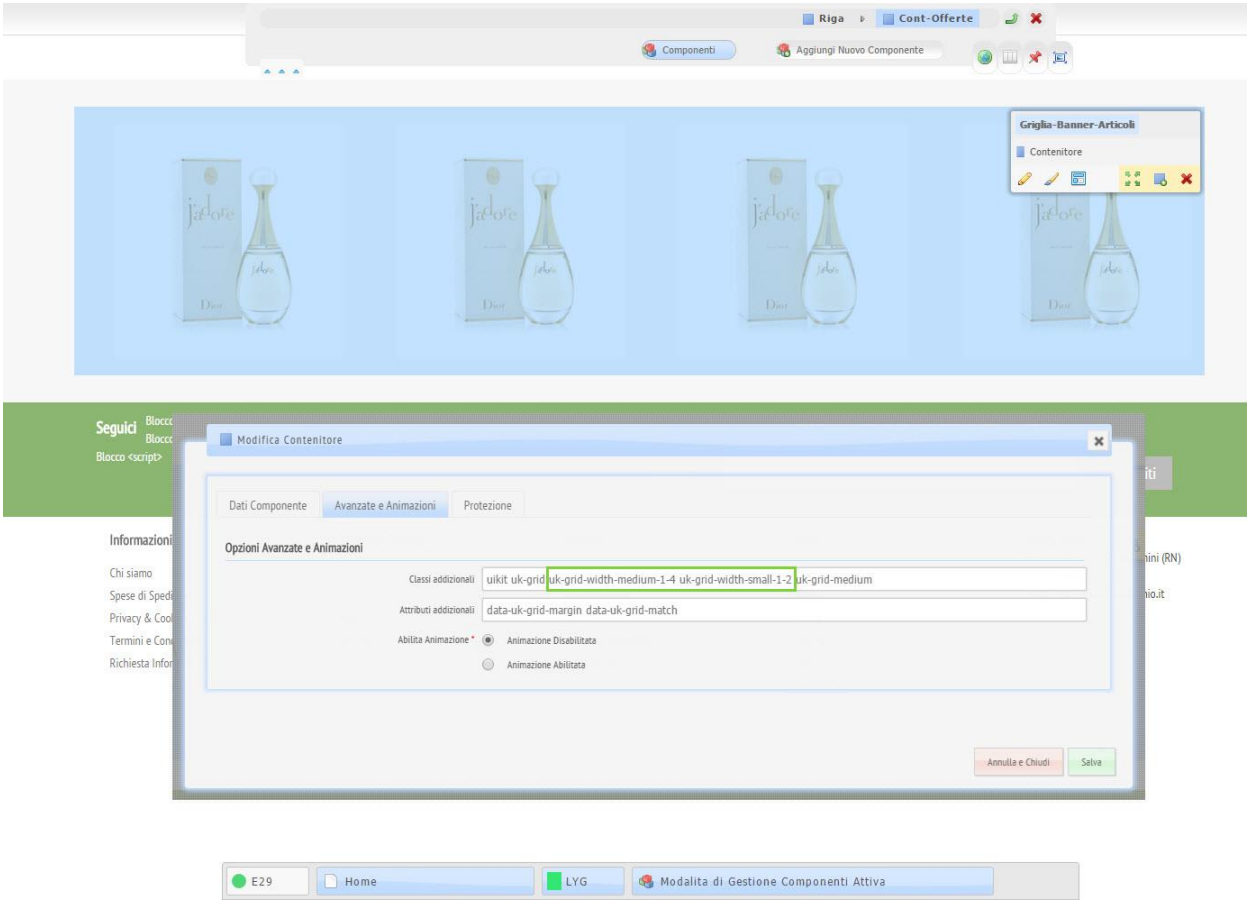
  La larghezza di questi Componenti Contenitore sarà determinata dalla specifica classe assegnata nel punto precedente al Contenitore della Griglia



MODELLO ECOMMERCE 29

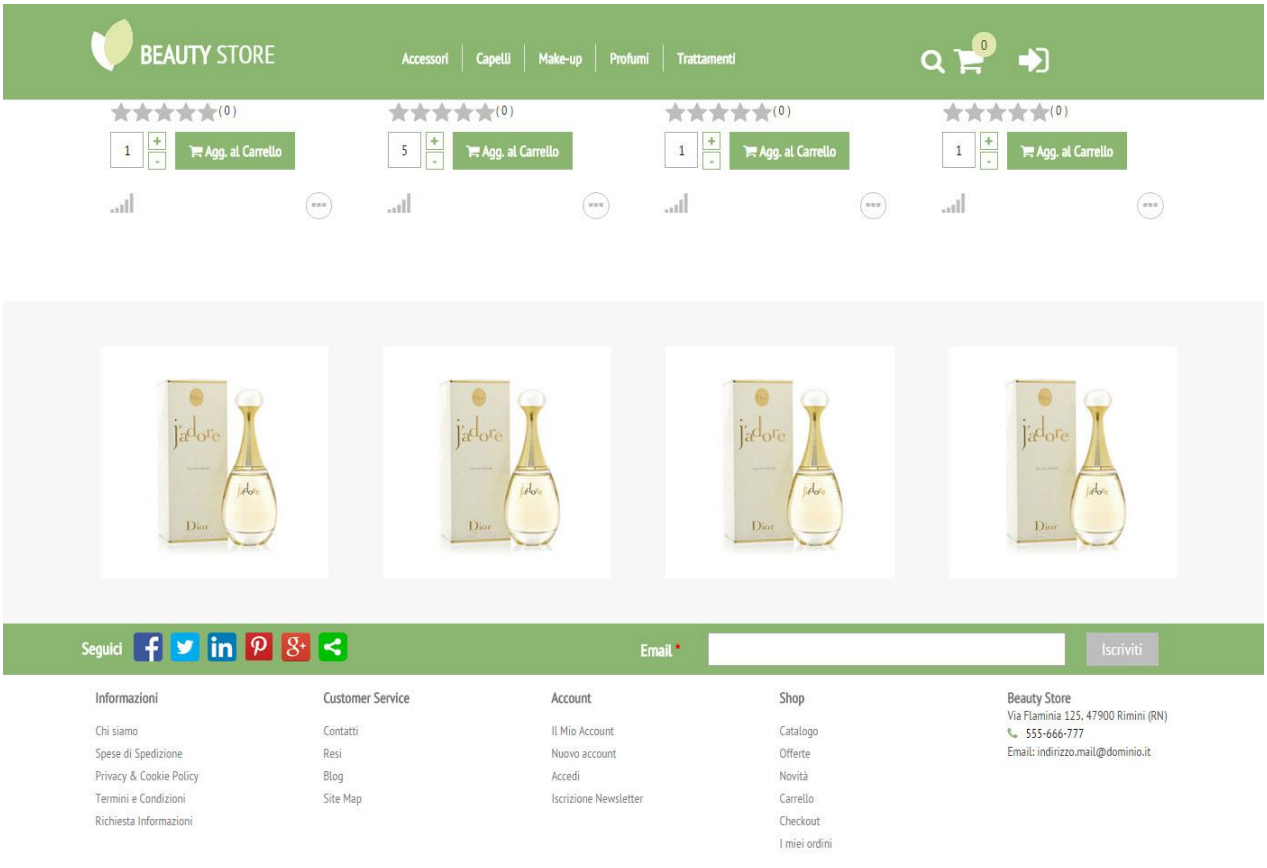
E' possibile trovare un esempio di griglia responsiva in cui il numero di colonne è determinato dalla specifica classe assegnata direttamente al contenitore della griglia nella **Home Page** del modello di riferimento.

La griglia in questione è quella utilizzata per gestire i banner degli articoli nella parte bassa della pagina immediatamente al di spora del piede.



Le due classi **uk-grid-width-medium-1-4** e **uk-grid-width-small-1-2** applicate al contenitore della griglia fanno si che:

- per viewport maggiori o uguali a 768px (**uk-grid-width-medium-1-4**) i 4 blocchi di contenuto interni alla griglia occupino tutti 1/4 dello spazio complessivo disponendosi quindi su di una sola riga



- per viewport compresi tra i 480px e i 767px (**uk-grid-width-small-1-2**) i 4 blocchi di contenuto interni alla griglia occupino tutti metà dello spazio complessivo disponendosi quindi su due righe

- per viewport inferiori ai 768px la griglia si linearizza e i 4 blocchi di contenuto occuperanno il 100% dello spazio complessivo disponendosi uno sotto l'altro

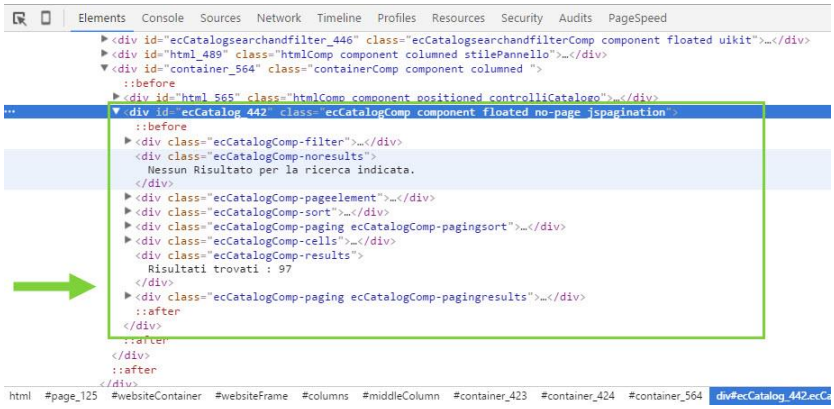
GRIGLIA E CATALOGO ECOMMERCE

Nei capitoli precedenti di questa guida abbiamo visto come poter implementare in Passweb una Griglia responsiva mediante un sistema di Componenti Contenitore annidati ai quali assegnare le classi richieste dal framework uikit.

Questo modo di operare non è però sufficiente per poterci consentire di realizzare, in Passweb, un sito Ecommerce perfettamente responsivo.

Tra i componenti Ecommerce di Passweb possiamo infatti trovarne alcuni che a livello strutturale si presterebbero molto bene per poter essere utilizzati come una Griglia responsiva.

Analizzando ad esempio il markup HTML del Componente "**Catalogo Ecommerce**" possiamo notare una struttura analoga a quella cui sappiamo di dover far riferimento per poter realizzare, secondo i dettami di uikit, una griglia responsiva



Le celle dei singoli articoli, in particolare, sono tutte racchiuse all'interno di un contenitore più esterno, nello specifico quello che, nello style editor di Passweb, è identificato come "**Catalogo – Riquadro celle**" (in figura la div con classe **ecCatalogComp-cells).**

Ogni cella all'interno di questo contenitore è poi identificata da una singola div (in figura quelle con classi **ecCatalogComp-cell**) all'interno della quale sono inseriti i componenti interni al Catalogo Ecommerce (Immagine Articolo, Prezzo, Descrizione ecc…) che ne definiscono i contenuti veri e propri.

Sulla base di questa semplice analisi viene quindi automatico **pensare al "Riquadro Celle" come al Contenitore della Griglia e alle singole Celle come ai blocchi di contenuto, ossia alle colonne, interne alla griglia stessa**.

In teoria sarebbe quindi sufficiente assegnare al Riquadro Celle, ad esempio, le classi .uk-grid, .uk-grid-width-large-1-4 e .uk-grid-width-medium-1-2 per far si che:

- per viewport **large** (maggiori o uguali a 960px) le celle del catalogo assumano tutte la stessa larghezza disponendosi 4 per riga
- per viewport **medium** (maggiori o uguali a 768px) le celle del catalogo assumano tutte la stessa larghezza disponendosi 2 per riga
- per viewport inferiori a 768px il Riquadro Celle si linearizzarsi e le singole celle si dispongano in un'unica colonna una sotto l'altra.

Il problema nel tentare di fare una cosa di questo tipo è rappresentato dal fatto che il markup del Componente "Catalogo Ecommerce" è generato in automatico dal programma e l'utente non ha la possibilità di definire delle classi aggiuntive per l'elemento "Riquadro Celle" ne tanto meno per le singole Celle.

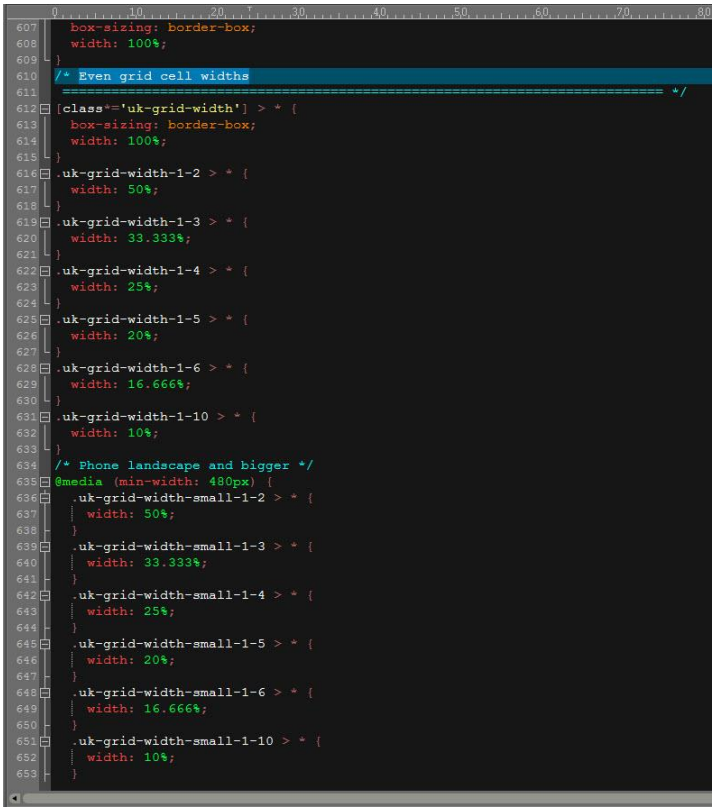
Eventuali classi aggiuntive possono essere aggiunte, mediante la maschera di configurazione del componente solo all'elemento più esterno, ossia quello che racchiude, oltre al Riquadro celle anche tutti gli altri elementi del componente come la paginazione, i campi di ordinamento ecc…

In considerazione di ciò **se vogliamo rendere responsivo il Catalogo Ecommerce di Passweb dobbiamo per forza di cose intervenire manualmente, integrando la libreria uikit.css e scrivendo le media query necessarie per assegnare a questo componente un comportamento responsivo.**

Tutto questo può però essere fatto in maniera piuttosto semplice replicando esattamente la stessa logica di gestione adottata da uikit.

Di base si tratta infatti di andare a prendere le media query utilizzate da uikit per gestire una griglia responsiva con colonne tutte della stessa larghezza assegnando apposite classi al solo contenitore esterno della griglia, e riadattarle al nostro caso specifico.

Le media query in questione sono esattamente quelle presenti nella sotto-sezione "**Even grid cell widths**" della sezione "**Component: Grid"** presente all'interno del file uikit.css



```
  box-sizing: border-box;
  width: 100%;
}
/* Even grid cell widths
 ================================================================== */
[class*='uk-grid-width'] > * {
  box-sizing: border-box;
  width: 100%;
}
.uk-grid-width-1-2 > * {
  width: 50%;
}
.uk-grid-width-1-3 > * {
  width: 33.333%;
}
.uk-grid-width-1-4 > * {
  width: 25%;
}
.uk-grid-width-1-5 > * {
  width: 20%;
}
.uk-grid-width-1-6 > * {
  width: 16.666%;
}
.uk-grid-width-1-10 > * {
  width: 10%;
}
/* Phone landscape and bigger */
@media (min-width: 480px) {
  .uk-grid-width-small-1-2 > * {
    width: 50%;
  }
  .uk-grid-width-small-1-3 > * {
    width: 33.333%;
  }
  .uk-grid-width-small-1-4 > * {
    width: 25%;
  }
  .uk-grid-width-small-1-5 > * {
    width: 20%;
  }
  .uk-grid-width-small-1-6 > * {
    width: 16.666%;
  }
  .uk-grid-width-small-1-10 > * {
    width: 10%;
  }
```

In questo senso, ad esempio, la regola di uikit

```
.uk-grid-width-1-2 > * {
  width: 50%;
}
```

riadatta alle nostre esigenze potrebbe diventare

```
.pw-grid-width-1-2 div[class*="cell_"]{
  width: 50%;
}
```

dove il selettore il selettore di attributo **[class*='cell_']** ci consente di applicare la regola a tutte le celle del Catalogo.

In definitiva per poter assegnare al Catalogo Ecommerce un Comportamento responsivo è necessario:

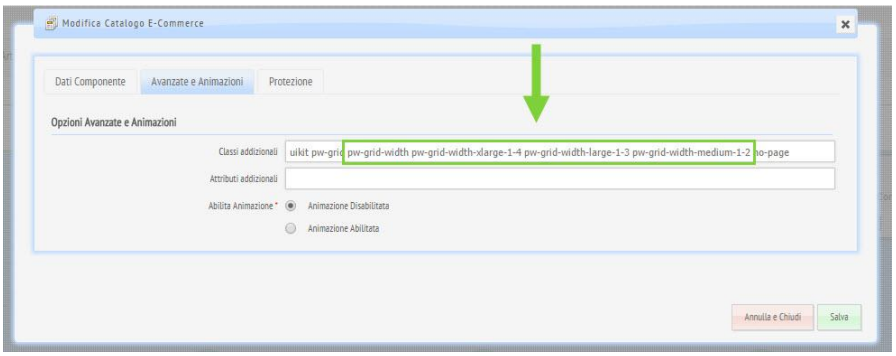
- Integrare il file uikit.css con il contenuto del file gridCatalogo.css scaricabile nella sezione Risorse di questa guida
- Assegnare al Componente Catalogo Ecommerce le seguenti classi:
  - **uikit:** necessaria per gestire il “float: none”
  - **pw-grid:** necessaria per poter applicare al componente le regole definite nel file gridCatalogo.css
  - una o più classi tra quelle presenti nella tabella di seguito riportata

CLASSE	DESCRIZIONE
.pw-grid-width-*	Le proprietà di questa classe verranno applicate per una qualsiasi larghezza del viewport. Le colonne della griglia avranno tutte la stessa larghezza e si posizioneranno una a fianco all’altra
.pw-grid-width-small-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 480px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 480px si disporranno una sopra l’altra
.pw-grid-width-medium-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 768px**

	Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 768px si disporranno una sopra l'altra
.pw-grid-width-large-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 960px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 960px si disporranno una sopra l'altra
.pw-grid-width-xlarge-*	Le proprietà di questa classe verranno applicate per **larghezze del viewport maggiori o uguali a 1220px** Le colonne manterranno tutte la stessa larghezza e per viewport inferiori ai 1220px si disporranno una sopra l'altra

ATTENZIONE! il carattere * dovrà essere sostituito dal solito sistema di due numeri. In questo caso però il primo numero dovrà essere sempre 1 mentre il secondo variando tra 1, 2, 3, 4, 5, 6 e 10 indicherà esattamente il numero di celle per riga

Supponendo quindi di assegnare al Catalogo Ecommerce le classi aggiuntive **pw-grid-width, pw-grid-width-xlarge-1-4, pw-grid-width-large-1-3, pw-grid-width-medium-1-2** (come avviene nel modello di riferimento)



otterremo un Catalogo in cui:

- per viewport **maggiori o uguali a 1220px** le celle articolo assumeranno tutte la stessa larghezza disponendosi 4 per riga
- per viewport **compresi tra 960px e1219px** le celle articolo assumeranno tutte la stessa larghezza disponendosi 3 per riga
- per viewport **compresi tra 768px e959px** le celle articolo assumeranno tutte la stessa larghezza disponendosi 2 per riga
- per viewport **inferiori a 768px** il Catalogo si linearizzerà e gli elementi in esso contenuti, tra cui le singole celle, si disporranno in un'unica colonna uno sotto l'altro

Per poter gestire al meglio un Catalogo Ecommerce responsivo è consigliabile inoltre:

- Azzerare eventuali margini e/o padding laterali (destro e sinistro) assegnati a default da Passweb al Riquadro Celle e alla singole Celle de Catalogo
- Azzerare i bordi assegnati a default da Passweb alla singole Celle de Catalogo
- Racchiudere tutti i Componenti interni alla Cella (Immagine Articolo, Prezzo, Titolo ecc…) in un unico Componente Contenitore e assegnare ad esso eventuali margini e/o padding per spaziare sia verticalmente che orizzontalmente le celle del catalogo.

Infine, un'ultima osservazione molto importante da fare, è quella secondo cui il markup di Componenti Ecommerce quali, Offerte / Novità, Popolarità Prodotto, Abbinati Ecommerce, Risultati Ricerca ecc… è del tutto analogo a quello utilizzato per il Catalogo Ecommerce

In conseguenza di ciò è possibile assegnare a questi componenti un comportamento responsivo operando esattamente allo stesso modo di quanto appena visto per il Catalogo Ecommerce.

GRIGLIA E FORM DI REGISTRAZIONE / PROFILO UTENTE

Per poter implementare il Componente Griglia all'interno dei componenti Passweb "Registrazione Utente" e "Profilo Utente" ottenendo così dei form perfettamente responsivi, sarebbe sufficiente, di base, gestire tutto il sistema di Componenti Contenitore annidati ai quali assegnare le classi richieste dal framework come componenti interni al Componente Registrazione e/o Profilo Utente.

In sostanza dovrebbero quindi essere inseriti all'interno dei due Componenti Passweb tutti i Contenitori necessari per creare l'infrastruttura della Griglia e all'interno di questi contenitori andrebbero poi inseriti i vari campi del form.

Nel momento in cui dovessimo però configurare l'infrastruttura della Griglia utilizzando le classi con suffisso uk-* (quelle standard di uikit) potremo incontrare dei conflitti con alcune delle proprietà CSS presenti nei file strutturali di Passweb e non riusciremo quindi a implementare la nostra griglia responsiva.

Per risolvere questo problema, come indicato nel precedente capitolo "*Passweb – Operazioni Preliminari*" di questa guida, è possibile apportare una semplice integrazione al file uikit.css.

Facendo quindi riferimento a quanto indicato nel capitolo "*Passweb – Operazioni Preliminari*" potremo implementare la Griglia di uikit all'interno dei Componenti Registrazione e Profilo utente operando esattamente allo stesso modo di come si opera per implementare la Griglia in una qualsiasi altra pagina del sito con l'unica accortezza di utilizzare non le classi con suffissi uk- ma bensì quelle con suffissi fr-

ATTENZIONE! Affinchè l'utilizzo delle classi con suffisso fr- possa funzionare correttamente è necessario verificare di aver apportato al file uikit.css l'integrazione descritta nel capitolo "*Passweb – Operazioni Preliminari"*

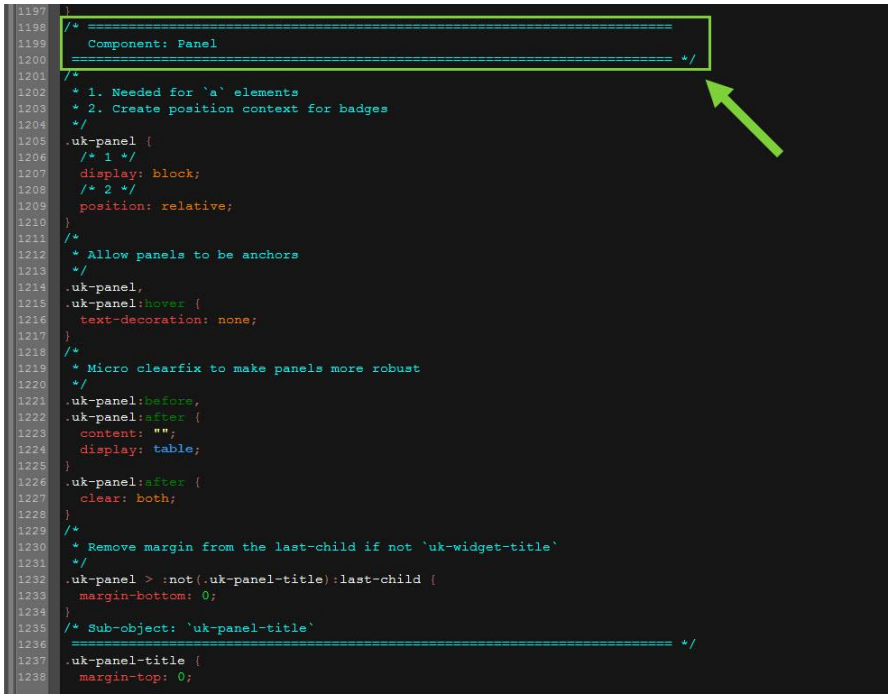
E' possibile visualizzare un esempio di quanto detto nelle pagine "Registrazione Utente" e "Profilo Utente" del modello di riferimento

I PANNELLI

Il componente Pannello di uikit consente di evidenziare determinate porzioni di contenuto all'interno di una pagina web. Tipicamente questi componenti sono utilizzati per racchiudere i contenuti da inserire nelle colonne di una Griglia.

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/panel.html

Le regole CSS relative a questo componente sono localizzate tutte nel file uikit.css all'interno della sezione "**Component: Panel**"

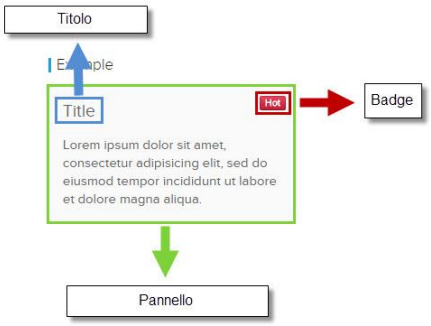


```
/* ========================================================================
   Component: Panel
 ========================================================================== */
/*
 * 1. Needed for `a` elements
 * 2. Create position context for badges
 */
.uk-panel {
  /* 1 */
  display: block;
  /* 2 */
  position: relative;
}
/*
 * Allow panels to be anchors
 */
.uk-panel,
.uk-panel:hover {
  text-decoration: none;
}
/*
 * Micro clearfix to make panels more robust
 */
.uk-panel:before,
.uk-panel:after {
  content: "";
  display: table;
}
.uk-panel:after {
  clear: both;
}
/*
 * Remove margin from the last-child if not `uk-widget-title`
 */
.uk-panel > :not(.uk-panel-title):last-child {
  margin-bottom: 0;
}
/* Sub-object: `uk-panel-title`
 ========================================================================== */
.uk-panel-title {
  margin-top: 0;
```

CONFIGURAZIONE

Ogni componente di tipo Pannello è costituito, essenzialmente, da 3 diversi elementi:

- il Pannello in se con tutti i suoi contenuti
- un Titolo
- un Badge



ATTENZIONE! Non tutti questi elementi devono necessariamente essere presenti all'interno di un Pannello

A livello di markup, nella sua versione completa, un Pannello sarà quindi costituito **da un elemento contenitore, tipicamente una < div >, più esterna che individua il Pannello vero e proprio** e all'interno del quale andranno quindi inseriti i contenuti del Pannello stesso. Tra questi contenuti, **oltre a del normale testo**, potremmo avere **un Titolo, rappresentato tipicamente un tag di tipo < h1 >, <h2 > … , e un badge ossia un'altra < div >** stilizzata in un certo modo a seconda della specifica classe CSS che le viene assegnata.

Per quel che riguarda invece le classi da assegnare ai diversi elementi di questo markup avremo che:

- al contenitore più esterno, quindi al Pannello vero e proprio, deve essere assegnata la classe **.uk-panel**
- al Titolo del Pannello deve essere assegnata la classe **.uk-panel-title**
- al Badge del Pannello devono essere assegnate le classi **.uk-badge** e **.uk-panel-badge**

In definitiva, dunque, il markup complessivo di un componente Pannello dovrà essere esattamente del tipo di quello qui di seguito evidenziato

```
<div class="uk-panel">
    <div class="uk-badge uk-panel-badge">
       Badge del Pannello
    </div>
    <h3 class="uk-panel-title">
       Titolo del Pannello
    </h3>
  Contenuti del Pannello
</div>
```

ATTENZIONE! Come già visto per gli elementi strutturali della Griglia, anche in relazione al componente Pannello, uikit non assegna uno specifico valore alla sua proprietà float

Anche in questo caso dunque, per poter implementare questo componente all'interno del nostro sito Passweb, dovremo impostare manualmente la proprietà **float** del contenitore esterno che individua il Pannello sul valore **none** (e come già per la griglia anche in questo caso ricorreremo alla classe uikit da noi appositamente realizzata).

STILI DEL PANNELLE E MODIFICATORI

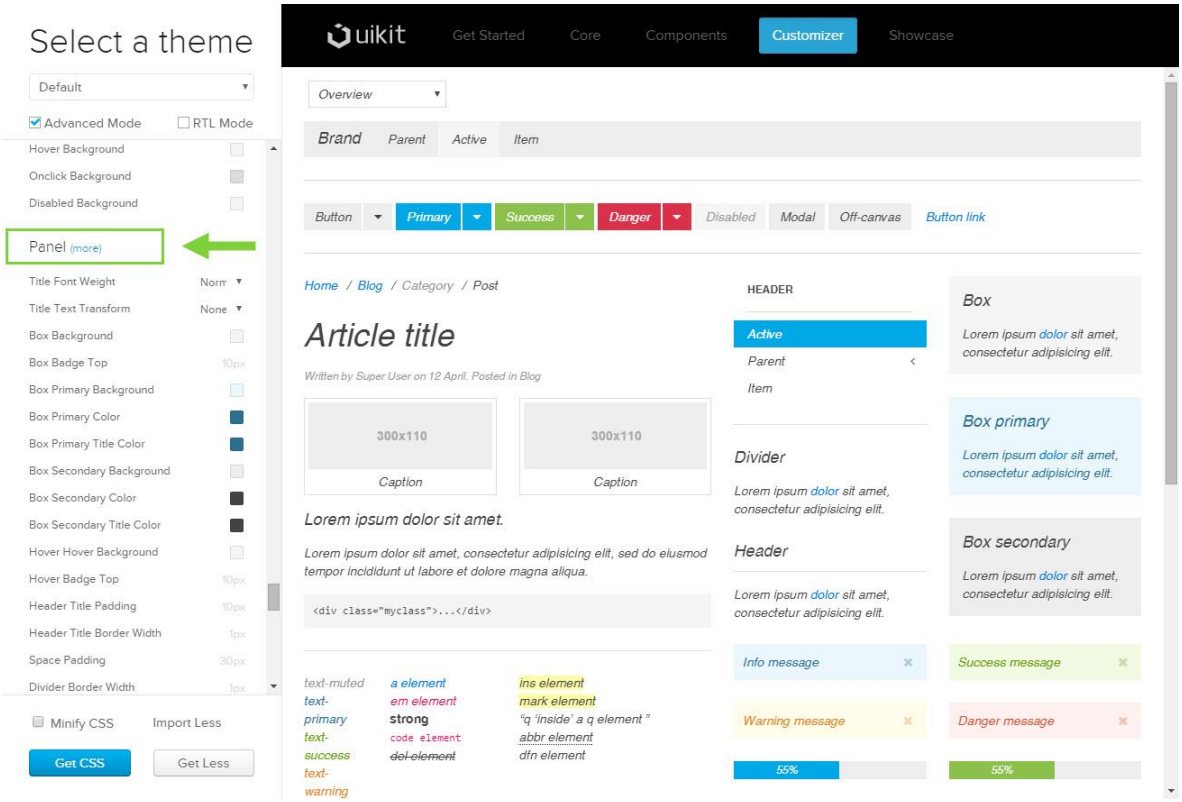
Prima di passare ad analizzare come poter implementare un componente Pannello all'interno del proprio sito Passweb, è necessario fare alcune semplici considerazioni relativamente allo stile adottato da questo componente.

In questo senso va detto infatti che a default il componente Pannello non ha associato alcun tipo di stile.

Gli stili del Pannello sono infatti definiti da appositi modificatori, ossia specifiche classi CSS aggiuntive localizzate, all'interno del file uikit.css, in apposite sotto sezioni, di tipo "Modifier …", della sezione "Component: Panel"

```
}
/* Modifier: `uk-panel-box`
 ========================================================================== */
.uk-panel-box {
  padding: 15px;
  background: #f5f5f5;
  color: #444444;
}
.uk-panel-box-hover:hover {
  color: #444444;
}
.uk-panel-box .uk-panel-title {
  color: #444444;
}
.uk-panel-box .uk-panel-badge {
  top: 10px;
  right: 10px;
}
.uk-panel-box > .uk-panel-teaser {
  margin-top: -15px;
  margin-left: -15px;
  margin-right: -15px;
}
/*
 * Nav in panel
 */
.uk-panel-box > .uk-nav-side {
  margin: 0 -15px;
}
/*
 * Sub-modifier: `uk-panel-box-primary`
 */
.uk-panel-box-primary {
  background-color: #e6f7fd;
  color: #2d7091;
}
.uk-panel-box-primary-hover:hover {
  color: #2d7091;
}
.uk-panel-box-primary .uk-panel-title {
  color: #2d7091;
}
/*
 * Sub-modifier: `uk-panel-box-secondary`
 */
```

Assegnando quindi una di queste classi modificatori al contenitore del Pannello non faremo altro che assegnargli uno specifico stile, stile questo che potrà ovviamente essere modificato, a livello generale, agendo sulle regole CSS presenti all'interno della relativa sezione del file uikit.css sopra evidenziata, oppure a priori mediante il Customizer di uikit



Nel momento in cui dovessimo invece implementare questo componente all'interno del nostro sito Passweb potremmo sempre agire mediante lo style editor per variare tutte le proprietà grafiche che ci interessano per lo specifico Pannello.

Analizziamo ora alcuni di questi modificatori, nella loro versione di default rimandando, per maggiori dettagli ed esempi live alla documentazione ufficiale.

PANEL BOX

Assegnando al contenitore del Pannello la classe **.uk-panel-box**

```
<div class="uk-panel uk-panel-box">
    <h3 class="uk-panel-title">
       Titolo del Pannello
    </h3>
  Contenuti del Pannello
</div>
```

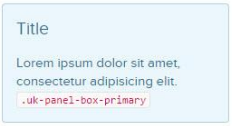
il Pannello stesso assumerà un aspetto, nella sua configurazione di default, del tipo di quello evidenziato in figura (sfondo grigio, padding di 15px)

PANEL BOX PRIMARY

Assegnando al contenitore del Pannello, oltre alla precedente classe **.uk-panel-box** anche la classe **.uk-panel-box-primary**,

```
<div class="uk-panel uk-panel-box uk-panel-box-primary">
    <h3 class="uk-panel-title">
       Titolo del Pannello
    </h3>
  Contenuti del Pannello
</div>
```

il Pannello stesso assumerà un aspetto, nella sua configurazione di default, del tipo di quello evidenziato in figura (sfondo azzurro, padding di 15px)



ATTENZIONE! utilizzando la sola classe **.**uk-panel-box-primary non verrà applicato il padding di 15px

PANEL BOX SECONDARY

Assegnando al contenitore del Pannello, oltre alla precedente classe **.uk-panel-box** anche la classe **.uk-panel-box-secondary**,

```
<div class="uk-panel uk-panel-box uk-panel-box-secondary">
    <h3 class="uk-panel-title">
       Titolo del Pannello
    </h3>
  Contenuti del Pannello
</div>
```

il Pannello stesso assumerà un aspetto, nella sua configurazione di default, del tipo di quello evidenziato in figura (sfondo bianco, padding di 15px)

ATTENZIONE! utilizzando la sola classe **.**uk-panel-box-secondary non verrà applicato il padding di 15px

EFFETO HOVER

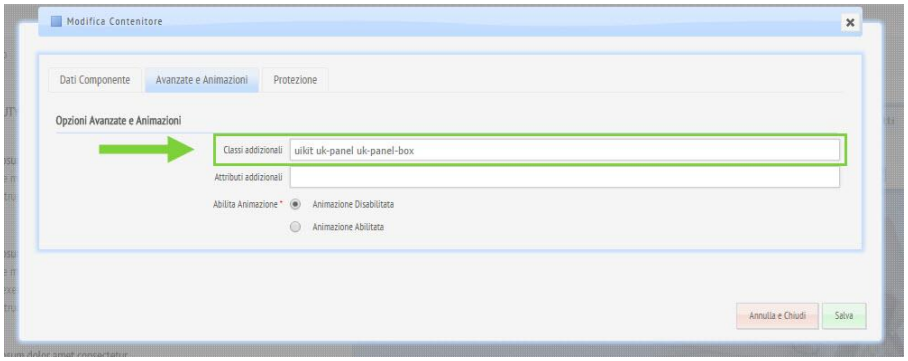
Assegnando al contenitore del Pannello la classe **.uk-panel-hover** è possibile modificarne lo stile e quindi evidenziarne i contenuti al passaggio del mouse

```
<div class="uk-panel uk-panel-hover ">
    <h3 class="uk-panel-title">
       Titolo del Pannello
    </h3>
  Contenuti del Pannello
</div>
```

IL PANNELLO IN PASSWEB

Per implementare in Passweb un componente Pannello è necessario utilizzare

- Un Componente Contenitore per gestire il l'elemento Pannello. Per esso dovrà essere impostato un **posizionamento Affiancato** e dovranno essergli assegnate le classi:
  - .**uikit** necessaria per gestire il "float: none"
  - .**uk-panel** necessaria per gestire il componente Pannello
  - eventuali classi modificatore, es. **.uk-panel-box** necessarie per assegnare al Pannello un determinato stile

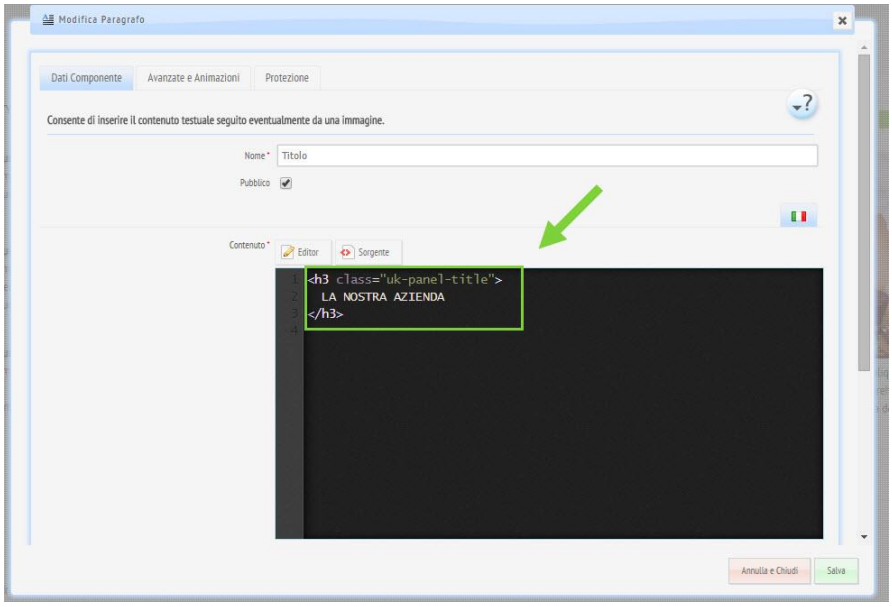


- All'intero del Componente Contenitore indicato al punto precedente potrà essere inserito un qualsiasi altro Componente Passweb necessario per gestire quelli che dovranno essere i contenuti del Pannello.

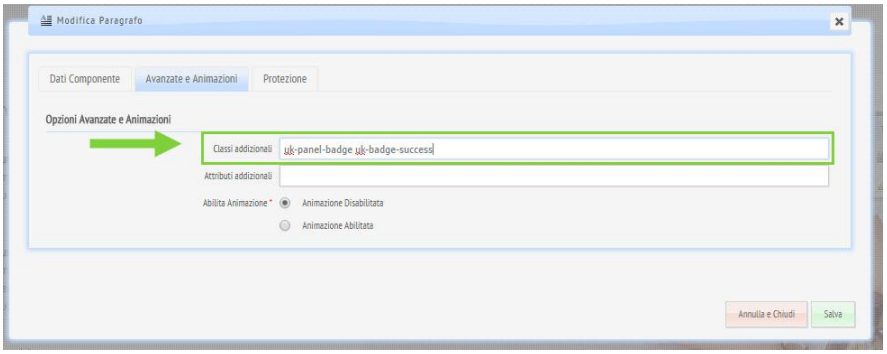
  Nel caso più semplice all'interno del contenitore del Pannello potrà essere inserito:

  - un semplice Componente Paragrafo, utilizzato per gestire il Titolo del Pannello, e il suo contenuto sia testuale sia a livello di immagini

    In queste condizioni potremmo utilizzare la sezione "**Sorgente**" del componente Paragrafo per assegnare al tag h utilizzato per il titolo del Pannello, la classe **.uk-title-panel** come richiesto dal framework



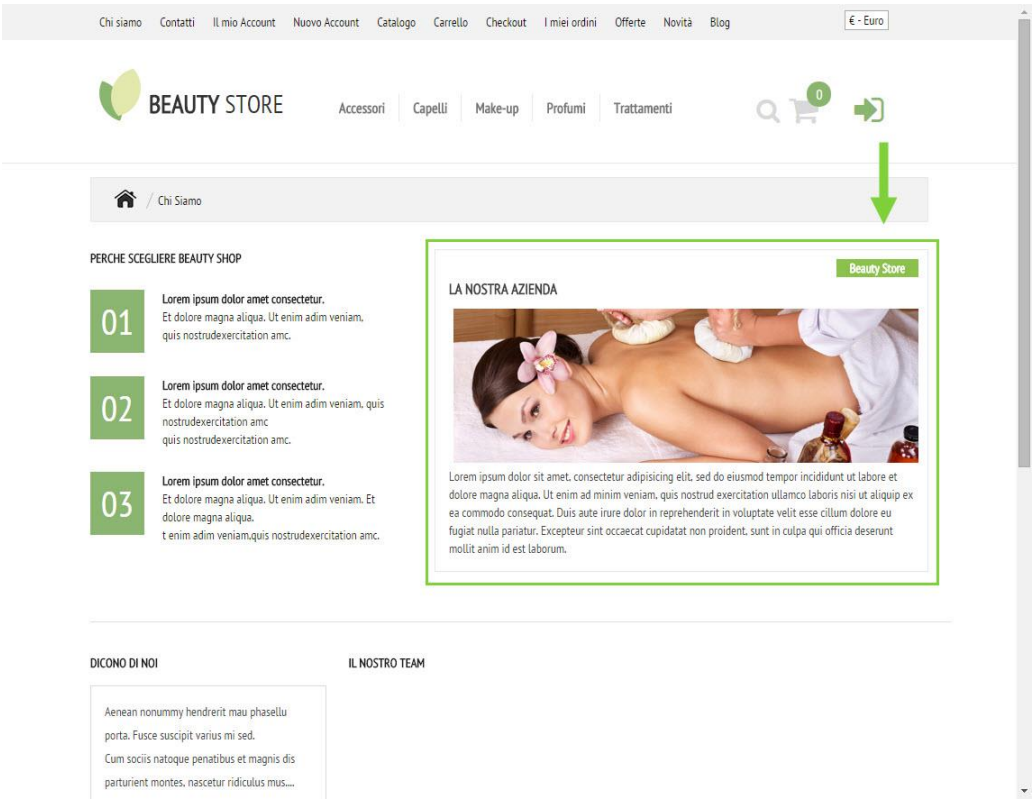
  - un ulteriore Componente Contenitore (o al limite un semplice Componente Paragrafo) utilizzato per gestire il Badge del Pannello al quale andranno quindi assegnate, come richiesto dal framework le cassi **.uk-badge** e **.uk-panel-badge** (più eventuali classi modificatori del Badge necessarie per assegnargli uno specifico stile)



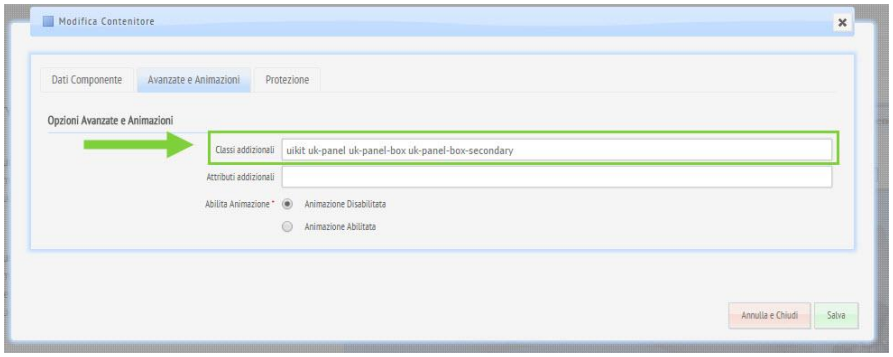
ATTENZIONE! Il Badge è un ulteriore componente di Uikit gestito sostanzialmente come il Pannello, nel senso che il suo stile di base è definito dalla specifica classe modificatore che gli viene assegnata. Per maggiori informazioni relativamente al componente Badge si rimanda alla documentazione ufficiale (http://getuikit.com/docs/badge.html)

MODELLO ECOMMERCE 29

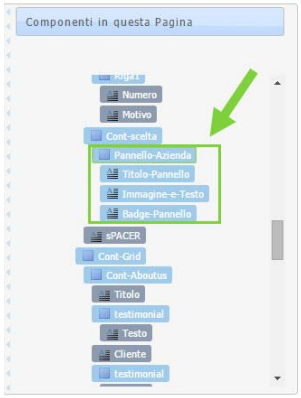
E' possibile trovare un esempio del componente Pannello nella pagina "**Chi Siamo**" del modello di riferimento (componente "Pannello-Azienda")



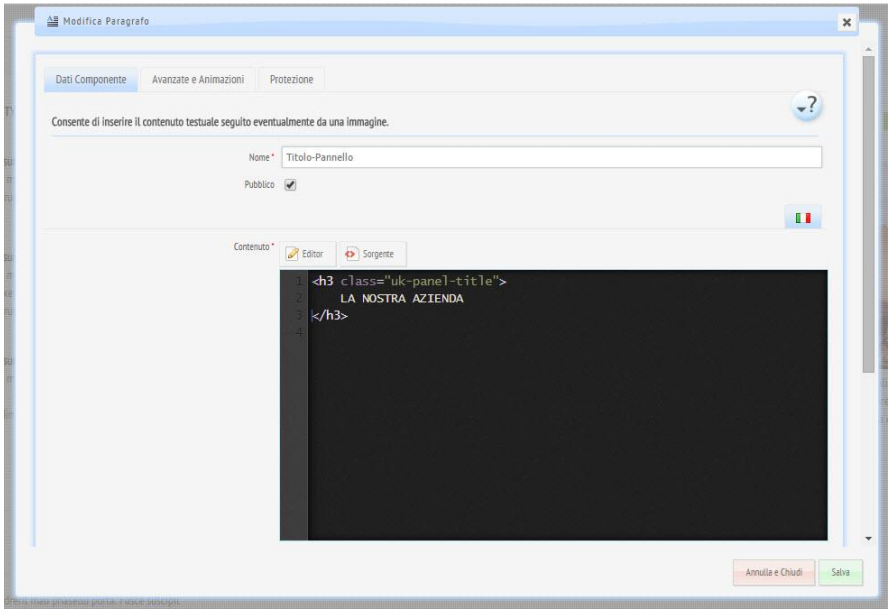
Per implementare il Pannello evidenziato in figura è stato utilizzato un Componente Contenitore con posizionamento Affiancato e con classi aggiuntive **uikit**, **uk-panel**, **uk-panel-box, uk-panel-box-secondary** (le ultime due necessarie per assegnargli uno stile di base costituito da uno sfondo bianco ed un padding di 15px).



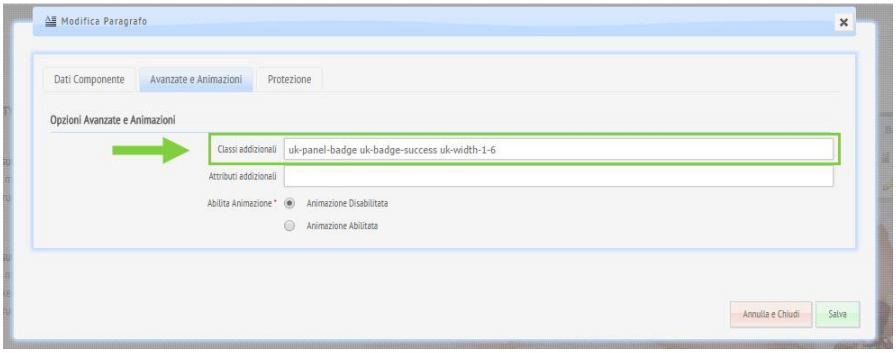
All'interno di questo contenitore sono stati inseriti tre distinti componenti Paragrafo:



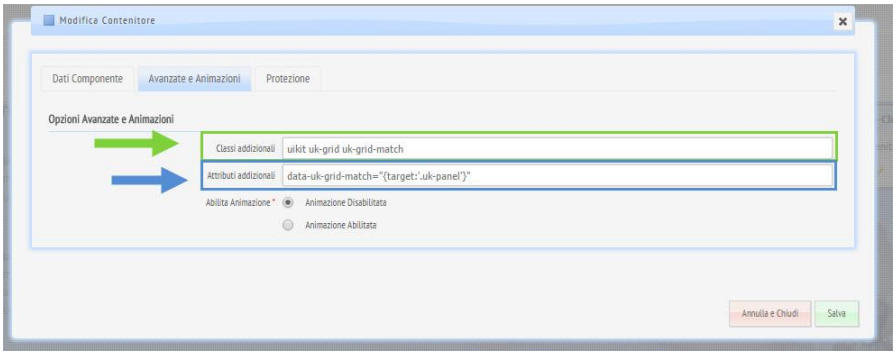
- il primo (**Titolo-Pannello**) per gestire il Titolo del Pannello



- il secondo (**Immagine-e-Testo**) per gestire i contenuti, testo e immagine, del Pannello
- il terzo (**Badge-Pannello**) per gestire il Badge del Pannello.



Il Pannello in questione è inoltre inserito all'interno di una griglia di due colonne (**Griglia-Chi-Siamo**) impostata in maniera tale che i Pannelli utilizzati per racchiudere i contenuti di queste colonne mantengano sempre la stessa altezza.



Per maggiori informazioni in merito si veda anche il capitolo "*Altezza delle colonne*" del "*Componente Griglia*" di questa guida

UTILITY

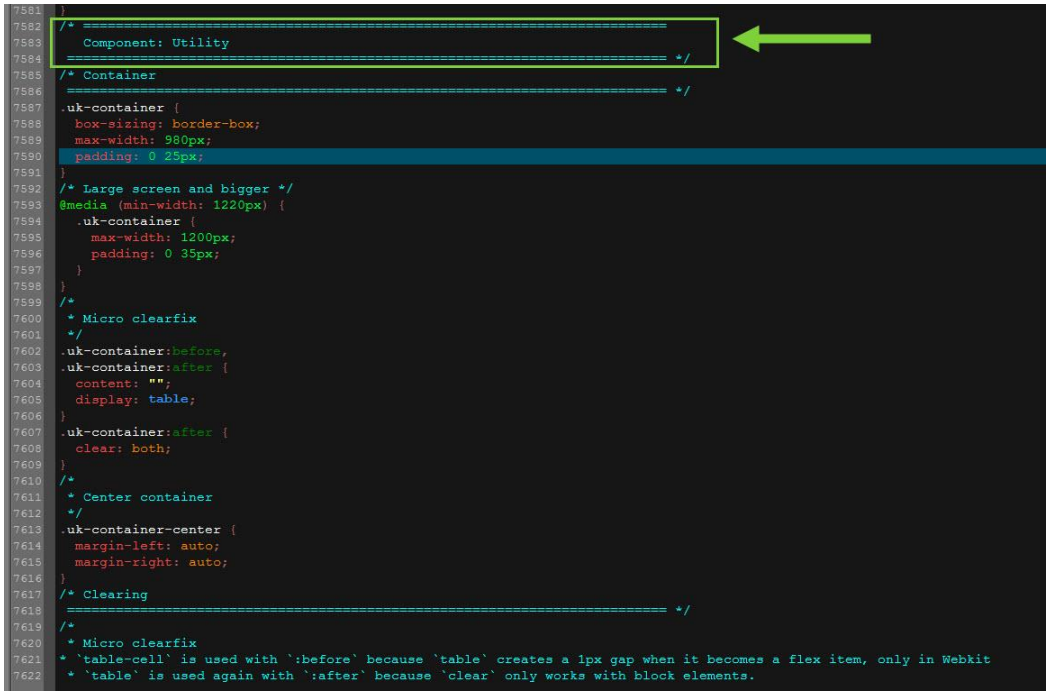
Uikit mette a disposizione dell'utente tutta una serie di classi estremamente utili per stilizzare in un certo modo i vari elementi presenti all'interno di una pagina web.

Mediante queste classi è possibile ad esempio:

- centrare elementi verticalmente e orizzontalmente
- posizionare elementi della pagina in posizioni predefinite
- gestire margini e padding uguali su diversi elementi
- visualizzare o meno determinati elementi in corrispondenza di determinate dimensioni del viewport
- …

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/utility.html

Le regole CSS relative a queste classi sono localizzate tutte nel file uikit.css all'interno della sezione "**Component: Utility**"



ATTENZIONE! La maggior parte delle classi della sezione Utility possono essere utilizzate in combinazione anche con altri componenti del Framework

I risultati che si ottengono applicando le classi della sezione Utility agli elementi di un sito Passweb possono, in genere, essere ottenuti anche agendo direttamente dallo style editor di Passweb stesso.

Ora però, mentre in alcuni casi (es. allineamento di immagini e testo) può effettivamente essere più semplice agire tramite le interfacce di gestione di Passweb (style editor o configurazione del componente), in altri casi risulta invece più rapido assegnare ad un elemento una certa classe piuttosto che andare a settare singolarmente, mediante lo style editor, diverse proprietà CSS correndo magari il rischio di saltarne una e non ottenere così il risultato desiderato.

Per poter assegnare una o più di queste classi ad un Componente Passweb sarà necessario agire come al solito mediante l'apposito campo "Classi Aggiuntive" presente nella maschera di configurazione del relativo componente

ATTENZIONE! gli elementi cui vengono applicate le classi della sezione Utility devono essere, tipicamente, in float: none per cui, lato Passweb, sarà necessario associare loro oltre alla specifica classe del framework anche la solita classe .uikit

Detto ciò vediamo analizziamo ora alcune di queste classi .

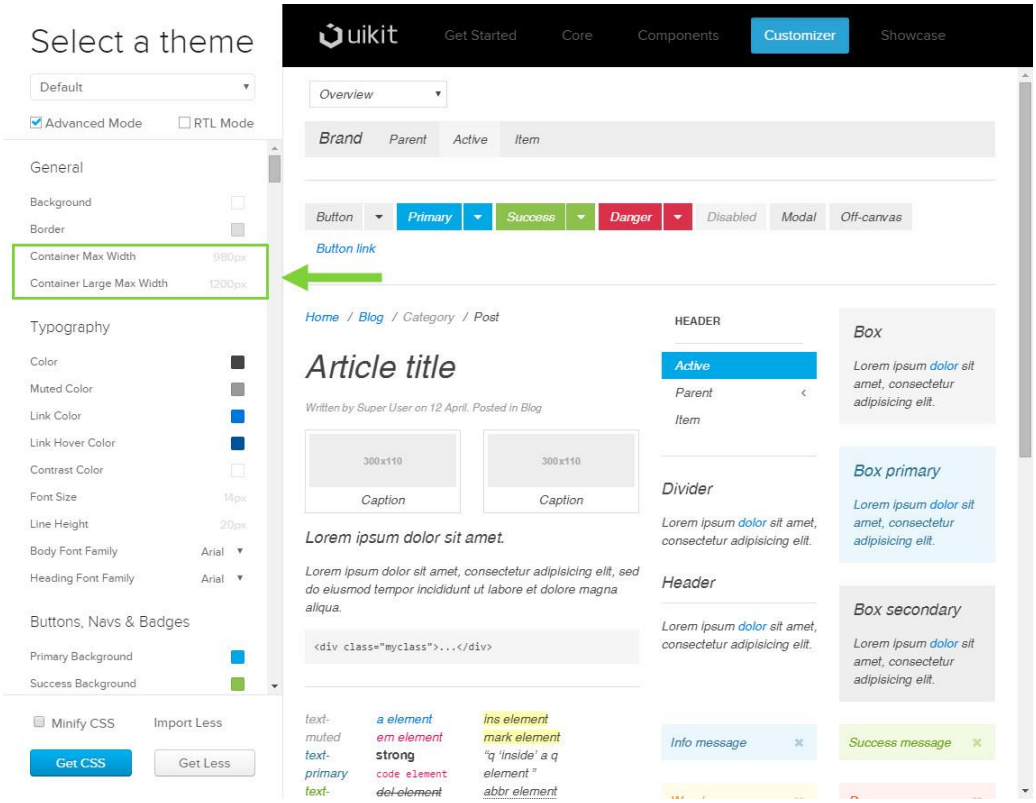
CONTAINER

La classe **.uk-container** assegnata ad un elemento di blocco, come può essere ad esempio una < div >, consente di impostare per l'elemento stesso una ben precisa larghezza massima.

Nello specifico:

- per viewport minori di 1220px la classe **.uk-container** assegna all'elemento cui viene applicata una larghezza massima di 980px
- per viewport maggiori o uguali a 1220px la classe **.uk-container** assegna all'elemento cui viene applicata una larghezza massima di 1200px

Quelli appena indicati sono i valori di default utilizzati dal framework, valori questi che possono comunque essere personalizzati come al solito agendo dallo style editor di Passweb, direttamente all'interno del file uikit.css oppure a priori mediante il Customizer



Generalmente questa classe viene assegnata ai cosiddetti “contenitori strutturali” ossia quelli che dovranno poi accogliere al loro interno tutti i contenuti della pagina web.

Proprio per questa ragione un’altra esigenza fondamentale potrebbe essere non solo quella di assegnare a questi elementi una larghezza massima ma anche e soprattutto fare in modo che i loro contenuti siano perfettamente centrati.

Per soddisfare entrambe queste esigenze, ossia per racchiudere determinati contenuti all’interno di un container, quindi di un elemento con una certa larghezza massima, e allo stesso tempo, posizionare questo container esattamente al centro del suo elemento padre (generalmente al centro della pagina web) sarà necessario utilizzare oltre alla classe .uk-container anche la classe **.uk-container-center**

.uk-container-center: consente di centrare orizzontalmente (rispetto al suo elemento padre) lo specifico elemento di blocco cui viene applicata

Per comprendere meglio quanto detto consideriamo un semplice esempio. Supponiamo di voler racchiudere tutti i contenuti che andranno in Testata al nostro sito, all’interno di un contenitore che abbia una larghezza massima preimpostata e che si posizioni esattamente al centro della pagina web.

Per soddisfare questa esigenza sarà sufficiente utilizzare un markup di questo tipo:

```
<div id="Testata" class="uk-container uk-container-center">
  ... /*Contenuti da inserire in testata*/
</div>
```

MODELLO ECOMMERCE 29

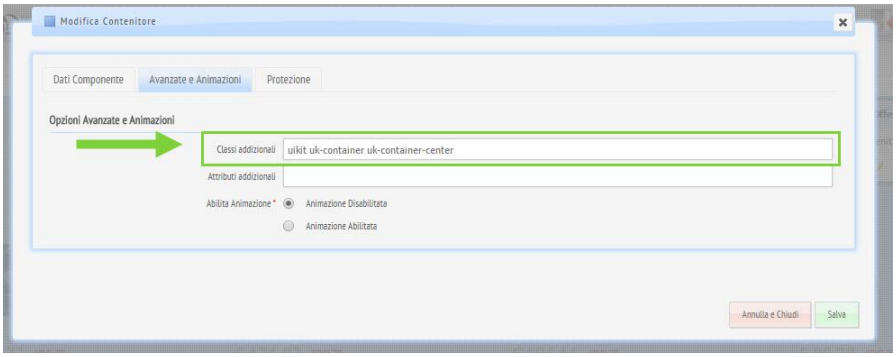
E’ possibile trovare un esempio di applicazione delle classi .uk-container .uk-container-center nella **Home Page** del modello di riferimento.

Nello specifico, all’interno di questa pagina è stato utilizzato, in Colonna Centrale, un primo Componente Contenitore (**Riga-Offerte**) con posizionamento **Incolonnato** che, per come è stato strutturato il layout del sito, andrà occupa in larghezza il 100% della pagina

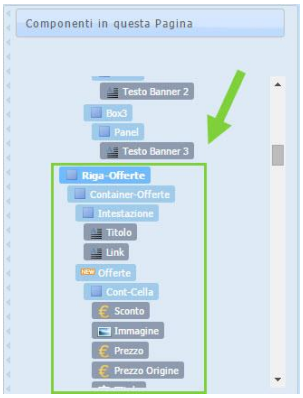
La necessità è poi quella di assegnare a tutti i contenuti di questo contenitore una larghezza massima predefinita e assicurarsi anche che siano tutti posizionati al centro di esso e quindi al centro della pagina web.

Per ottenere questo risultato è stato quindi inserito al suo interno un ulteriore Componente Contenitore (**Container-Offerte**) con posizionamento **Affiancato**, al quale sono state assegnate le classi aggiuntive:

- **uikit:** necessaria per gestire il float: none
- **uk-container:** necessaria per assegnare al Componente Container-Offerte una larghezza massima
- **uk-container-center:** necessaria per posizionare il Componente Container-Offerte (e tutti i suoi contenuti) esattamente al centro della pagina



I contenuti veri e propri (es. Componente Offerte Novità) sono stati quindi inseriti all'interno del "Container Offerte"



CLEARING AND FLOATING

Come ormai chiaro uikit, di base non assegna alcun valore alla proprietà float dei suoi Componenti.

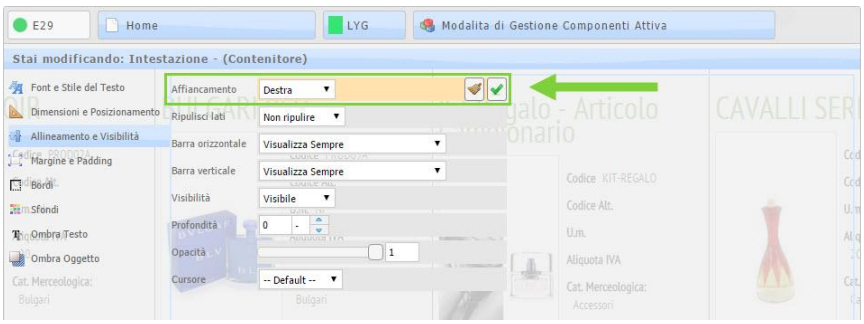
Anche nella costruzione e gestione di un layout interamente basato su uikit, può però essere necessario, in determinate circostanze, tornare ad utilizzare elementi flottanti allineati a sinistra o a destra a seconda dei casi.

Per gestire questa problematica il framework mette a disposizione dell'utente le seguenti classi:

- **.uk-float-left:** consente di impostare la proprietà **float** dell'elemento cui viene assegnata sul valore **left**
- **.uk-float-right:** consente di impostare la proprietà **float** dell'elemento cui viene assegnata sul valore **right**

Lato Passweb lo stesso risultato si può ottenere molto più semplicemente utilizzando per lo specifico componente un posizionamento **Affiancato** che, come noto, assegna a default alla proprietà float il valore left.

Nel caso in cui si voglia invece utilizzare un "float: right" sarà sufficiente impostare la proprietà "**Affiancamento**" presente nella sezione "**Allineamento e Visibilità**" dello style editor, sul valore "**Destra"**



VERTICAL ALIGNMENT

Un' esigenza piuttosto sentita all'interno di un sito web può essere quella di riuscire ad **allineare verticalmente** determinati oggetti posizionandoli, nello specifico, **nella parte centrale o bassa del loro elemento contenitore**.

ATTENZIONE! allineare verticalmente un'oggetto posizionandolo nella parte alta del suo elemento contenitore rappresenta il comportamento di base per cui per ottenere questo risultato non è eseguire nessuna operazione aggiuntiva.

Per soddisfare questa esigenza è necessario verificare, innanzitutto che l'elemento da allineare verticalmente sia racchiuso all'interno di un ben preciso elemento contenitore rispetto al quale verrà poi effettuato l'allineamento.

Fatta questa verifica sarà poi necessario utilizzare le seguenti classi:

- **.uk-vertical-align:** deve essere assegnata all'elemento contenitore rispetto al quale dovrà essere realizzato l'allineamento.

  ATTENZIONE! l'elemento contenitore rispetto cui realizzare l'allineamento **deve avere necessariamente un'altezza fissata**

- **.uk-vertical-align-middle:** deve essere assegnata all'elemento da allineare e consente di allinearlo centralmente
- **.uk-vertical-align-bottom:** deve essere assegnata all'elemento da allineare e consente di allinearlo alla parte bassa del suo elemento contenitore

ATTENZIONE! l'elemento da allineare deve avere una larghezza o una larghezza massima inferiore a quella del suo elemento contenitore

Supponendo ora di voler centrare, verticalmente, il testo racchiuso all'interno di un certo box, dovremo utilizzare un markup di questo tipo.

```
<div class="uk-vertical-align" style="height:400px">
    <div class="uk-vertical-align-middle">
    ... /*Contenuti centrati verticalmente*/
    </div>
</div>
```

In Passweb tutto questo può essere realizzato utilizzando due Componenti Contenitore annidati, il primo da utilizzare come elemento paddre dei contenuti da centrare verticalmente e il secondo come contenitore dei contenuti veri e propri.

Al primo andrà quindi assegnata la classe .**uk-vertical-align** oltre alla solita classe uikit, e andrà impostata per esso una specifica altezza.

Al secondo andrà invece assegnata la classe .**uk-vertical-align-middle** e al suo interno andranno poi inseriti i contenuti veri e propri.

VISIBILITY UTILITIES

Tra le classi messe a disposizione da uikit all'interno della sezione Utility le più interessanti sono indubbiamente, soprattutto nell'ottica di un loro utilizzo all'interno di un sito Passweb, quelle **che consentono di gestire la visibilità o meno di un determinato elemento in relazione al verificarsi di specifiche circostanze** (come ad esempio il passaggio del mouse o il fatto di utilizzare un dispositivo touch)

Vediamo quindi quali sono queste classi e in quali circostanze ci permettono di nascondere o visualizzare l'elemento a cui vengono applicate:

- **.uk-hidden:** consente di nascondere l'elemento cui viene applicata indipendentemente dal tipo di dispositivo utilizzato per visualizzare la pagina web.

  Vengono utilizzate le proprietà "**visibility: hidden**" e "**display: none**" per cui l'elemento è nascosto e non viene neppure considerato il suo ingombro all'interno della pagina web

- **.uk-hidden-touch:** consente di nascondere l'elemento cui viene applicata nel momento in cui la pagina web dovesse essere visitata da un dispositivo touch

  Vengono utilizzate le proprietà "**visibility: hidden**" e "**display: none**" per cui l'elemento è nascosto e non viene neppure considerato il suo ingombro all'interno della pagina web

- **.uk-hidden-notouch** consente di nascondere l'elemento cui viene applicata nel momento in cui la pagina web dovesse essere visitata da un dispositivo NON touch

  Vengono utilizzate le proprietà "**visibility: hidden**" e "**display: none**" per cui l'elemento è nascosto e non viene neppure considerato il suo ingombro all'interno della pagina web

- **.uk-invisible:** consente di nascondere l'elemento cui viene applicata indipendentemente dal tipo di dispositivo utilizzato per visualizzare la pagina web.

  A differenza della precedente classe .uk-hidden viene applicata la sola proprietà "**visibility: hidden**" per cui in queste condizioni l'elemento cui viene applicata questa classe è si nascosto ma mantiene sempre il suo ingombro all'interno della pagina web.

- **.uk-visible-hover:** deve essere utilizzata in combinazione con una delle precedenti classi che consentono di nascondere gli elementi cui vengono applicate; deve essere applicata all'elemento padre di quello nascosto e consente di visualizzare quest'ultimo nel momento in cui l'utente dovesse passa con il mouse sopra l'elemento cui viene applicata.

  Per visualizzare l'elemento nascosto è utilizzata la proprietà "**display: block**"

- **.uk-visible-hover-inline:** deve essere utilizzata in combinazione con una delle precedenti classi che consentono di nascondere gli elementi cui vengono applicate; deve essere applicata all'elemento padre di quello nascosto e consente di visualizzare quest'ultimo nel momento in cui l'utente dovesse passare con il mouse sopra l'elemento cui viene applicata.

  Per visualizzare l'elemento nascosto è utilizzata la proprietà "**display: inline-block**"

Ora mentre le prime 4 classi sono estremamente semplici da comprendere, anche in quello che può essere il loro utilizzo all'interno di Passweb, le ultime due meritano invece un piccolo approfondimento.

Innanzi tutto è bene sottolineare ancora una volta come queste due classi essendo utilizzate per visualizzare elementi inizialmente nascosti dovranno necessariamente essere utilizzate in combinazione con una di quelle classi che consentono invece di nascondere un elemento. Allo stesso modo essendo l'elemento che devono visualizzare inizialmente nascosto non potranno essere applicate direttamente ad esso ma dovranno invece essere applicate al suo elemento padre.

In sostanza dunque, il markup cui fare riferimento per poter utilizzare correttamente una di queste due classi dovrà essere del tipo di quello qui di seguito indicato:

```
<div class="uk-visible-hover">
    <div class="uk-hidden">
      ... /*Contenuto inizialmente nascosto*/
    </div>
</div>
```

In Passweb un markup di questo tipo si può ottenere, come al solito, utilizzando due Componenti Contenitore annidati ai quali assegnare le rispettive classi aggiuntive mediante l'apposito campo presente nella loro maschera di configurazione.

Infine per comprendere appieno la differenza tra queste due classi e cosa comporti effettivamente, per visualizzare un dato elemento, il fatto di impostare la proprietà display sul valore "block" piuttosto che sul valore "inline-block" è bene richiamare velocemente un concetto di base del CSS: gli **elementi blocco** e gli **elementi inline**

Cercando di semplificare al massimo questo concetto potremmo considerare una pagina web visualizzata all'interno di un browser come un insieme di tanti box rettangolari, non importa che si tratti di paragrafi, immagini, div o tabelle, si tratta pur sempre di box rettangolari.

Osservando attentamente l'immagine sopra riportata è possibile notare però come non tutti queste box rettangolari sono uguali. Alcuni contengono altri box al loro interno, altri sono invece contenuti all'interno dei primi.

Alcuni se si trovano (come accade in figura) in mezzo al testo lasciano che esso scorra loro intorno senza interromperne il flusso e senza andare a capo.

Questa considerazione piuttosto semplice, ci offre la rappresentazione della fondamentale distinzione tra **elementi blocco** (quelli in racchiusi in figura da un bordo nero) e **elementi inline** (quelli racchiusi in figura da un bordo rosso).

- **Elementi blocco:** sono box che possono contenere altri elementi, sia di tipo blocco che di tipo inline. Quando un elemento di blocco è inserito nel documento viene automaticamente creata una nuova riga nel flusso del documento stesso
- **Elementi inline:** non possono contenere elementi blocco, ma solo altri elementi inline (oltre che ovviamene il loro stesso contenuto, essenzialmente del testo). Quando sono inseriti in un documento non danno origine ad una nuova riga

Ogni elemento di una pagina web ha una sua visualizzazione predefinita, nel senso che, a default, è un elemento blocco (es. titoli, paragrafi, div …) oppure un elemento inline (es. span …). Attraverso la proprietà display possiamo agire su questi valori predefiniti trasformando, ad esempio, un elemento inline in uno di tipo blocco e viceversa.

I principali valori utilizzati per la proprietà display sono:

- **block:** l'elemento viene presentato come un blocco (a prescindere dalle sue caratteristiche di default);
- **inline:** l'elemento viene presentato come un in linea (a prescindere dalle sue caratteristiche di default);
- **inline-block:** l'elemento viene presentato come un blocco in linea, cioè un particolare tipo di blocco che pur avendo dimensioni, margini, padding e bordi come i normali elementi di tipo block, si dispone orizzontalmente e non produce alcun ritorno a capo;
- **none**: l'elemento non viene visualizzato nella pagina e non occupa alcun ingombro all'interno del documento.

Chiariti questi concetti possiamo ora tornare alle nostre due classi della sezione Utility avendo già ben chiaro cosa succederà nel momento in cui si dovesse utilizzare la classe **.uk-visible-hover** piuttosto che quella **.uk-visible-hover-inline.**

Nel primo caso, essendo applicata la proprietà **display: block** l'elemento visualizzato verrà considerato come un elemento blocco per cui verrà creata una nuova riga nel flusso del documento e eventuali contenuti successivi all'elemento visualizzato scorreranno verso il basso.

Nel secondo caso, essendo applicata la proprietà **display: inline-block** per cui l'elemento visualizzato comparirà sulla stessa riga senza produrre nessun ritorno a capo e, di fatto, senza interrompere il flusso del documento.

VISIBILITA' RESPONSIVA

Oltre alla possibilità di visualizzare o nascondere un elemento a seconda del fatto di passarci sopra con il mouse e/o di utilizzare un dispositivo touch, nella realizzazione e gestione di un sito responsivo la cosa veramente importante è, ovviamente, la possibilità di visualizzare o meno determinati contenuti a seconda delle dimensioni del viewport e quindi a seconda della particolare tipologia di dispositivo utilizzata.

All'interno della tabella di seguito riportata sono indicate le classi utilizzate da uikit per gestire questo tipo di visibilità e il risultato che si otterrà, sulle diverse tipologie di dispositivi, applicando queste stesse classi ad un qualsiasi elemento presente all'interno della pagina web.

CLASSE	SMALL (smartphone) Viewport minori di 768px	MEDIUM (teblet) Viewport compresi tra 768px 959px	LARGE (desktop) Viewport maggiori o uguali a 960px
.uk-visible-small	Visibile	Nascosto	Nascosto
.uk-visible-medium	Nascosto	Visibile	Nascosto
.uk-visible-large	Nascosto	Nascosto	Visibile
.uk-hidden-small	Nascosto	Visibile	Visibile
.uk-hidden-medium	Visibile	Nascosto	Visibile
.uk-hidden-large	Visibile	Visibile	Nascosto

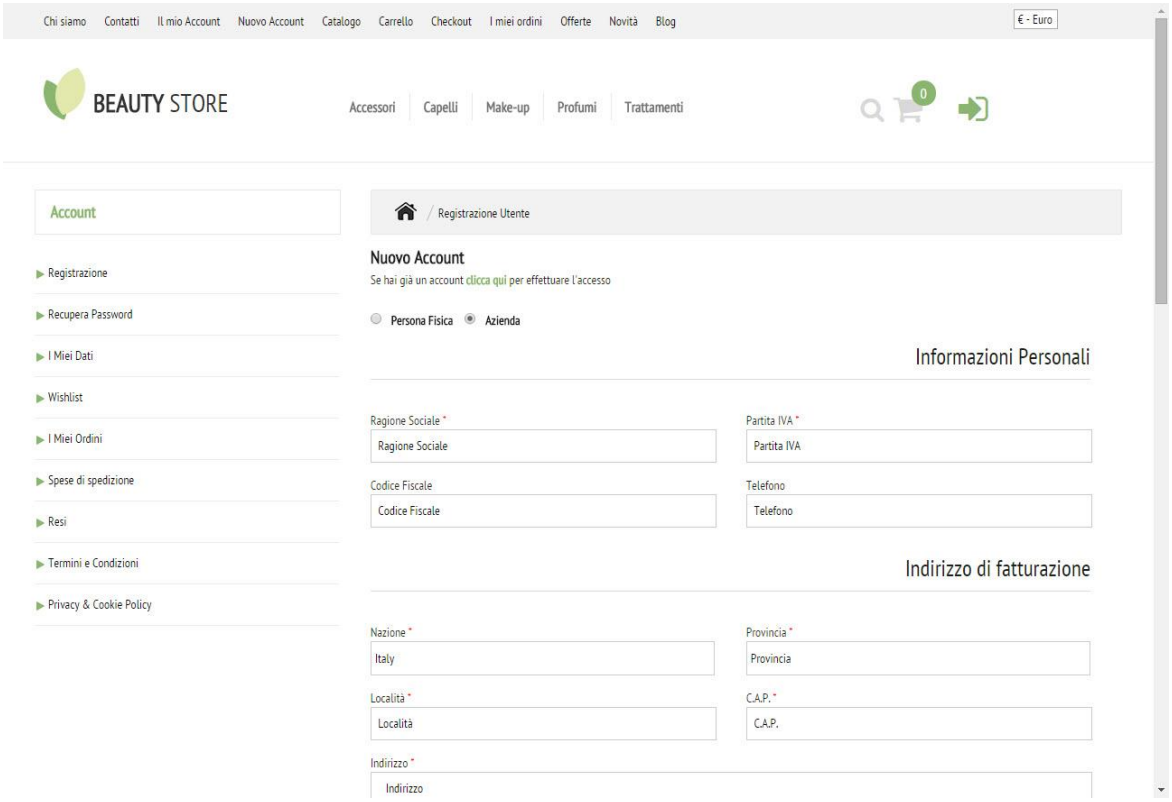
Supponendo dunque di voler visualizzare un dato elemento soltanto su schermi di grandi dimensioni ma non sui tablet o sugli smartphone dovremo assegnarli la classe **.uk-visible-large.** Se invece lo stesso elemento dovrà essere nascosto solo sugli smartphone dovremo allora utilizzare la classe **.uk-hidden-small**

ATTENZIONE! Per nascondere un dato elemento viene applicata la proprietà "**display: none**" il che significa che l'elemento verrà nascosto e non verrà considerato neppure il suo ingombro all'interno della pagina.

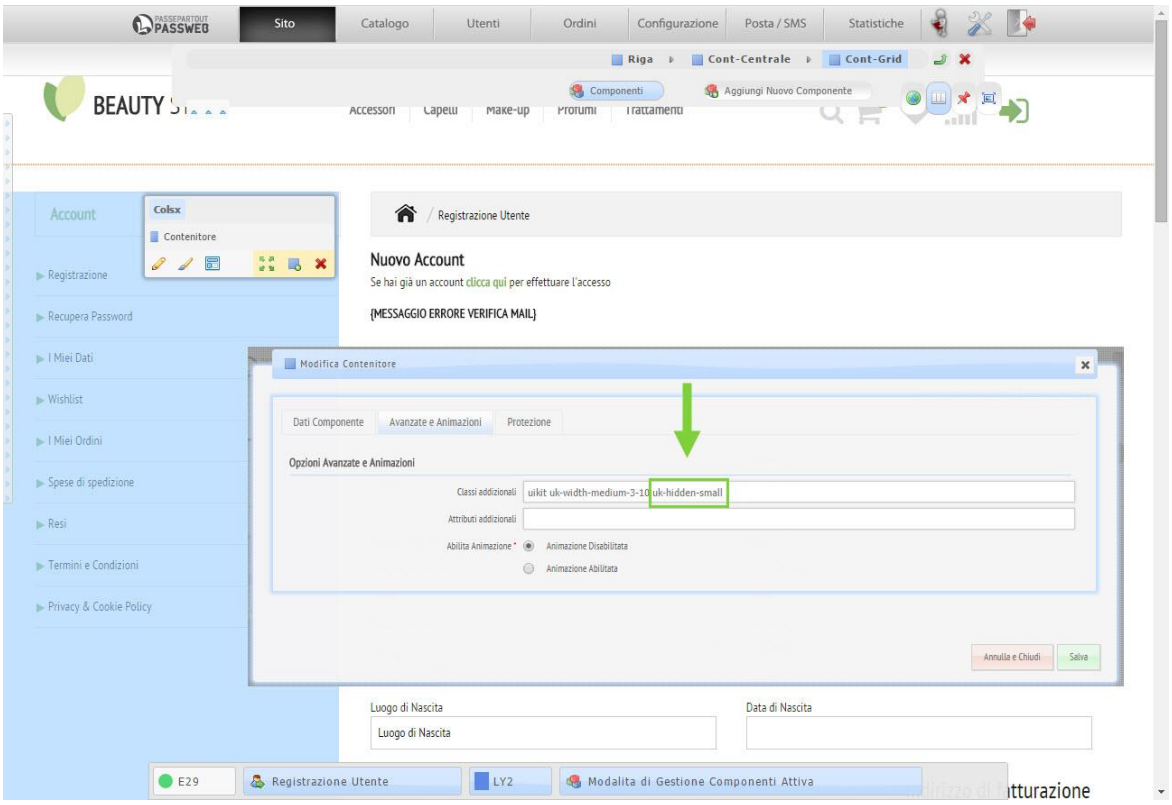
Le dimensioni del viewport corrispondenti agli schermi **Piccoli** (smartphone) **Medi** (Tablet) e **Large** (Desktop) possono essere personalizzate agendo, come al solito, direttamente all'interno del file uikit.css oppure a priori mediante l'apposito Customizer

MODELLO ECOMMERCE 29

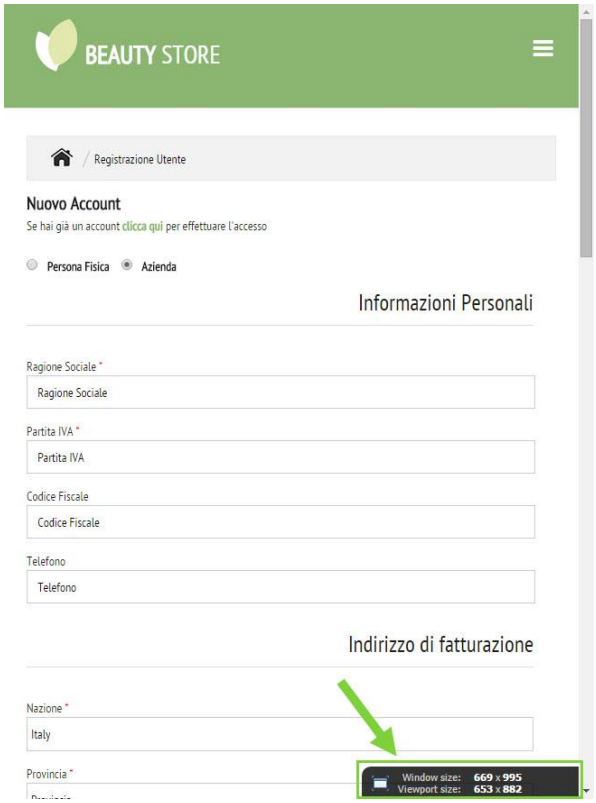
E' possibile trovare un esempio di applicazione delle classi di gestione della visibilità responsiva nella pagina "**Registrazione Utente**" del modello di riferimento.



All'interno di questa pagina è stata utilizzata una Griglia di due colonne; al Componente Contenitore utilizzato per individuare la colonna sinistra (**Colsx**) è stata assegnata la classe **.uk-hidden-small**



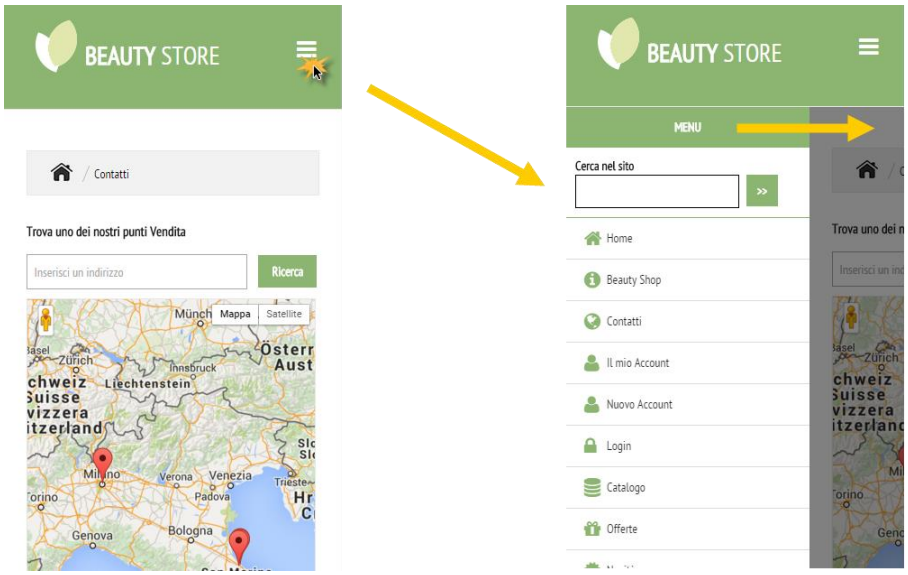
In queste condizioni dunque è possibile verificare che nel momento in cui si dovesse restringere la finestra del browser fino a portarla ad assumere una larghezza inferiore ai 768px la colonna sinistra non verrà più visualizzata



NAVIGAZIONE OFF-CANVAS

Come evidenziato nel capitolo "*Sistemi di navigazione*" di questa guida il cosiddetto "Menu off canvas" rappresenta, ad oggi, il pattern più utilizzato per gestire i menu di navigazione all'interno di un sito responsivo nel momento in cui il sito stesso dovesse essere visualizzato su di un dispositivo di piccole dimensioni (smartphone).

In questo schema quando il menu viene visualizzato, cliccando per questo su di un apposito pulsante, il corpo principale del documento scorre interamente verso destra (o sinistra) per lasciare spazio proprio al menu



Uikit consente di realizzare un sistema di navigazione del tipo di quello evidenziato in figura mediante il componente "**Off-canvas**" della sezione javascript

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/offcanvas.html

Le regole CSS relative a questo componente sono localizzate tutte nel file uikit.css all'interno della sezione "**Component: Off-canvas**"

```
  -webkit-animation: uk-rotate 2s infinite linear;
  animation: uk-rotate 2s infinite linear;
}
/* ========================================================================
   Component: Off-canvas
 ========================================================================== */
/*
 * This is the offcanvas overlay and bar container
 * 1. Hide by default
 * 2. Set fixed position
 * 3. Deactivate browser touch actions in IE11
 * 4. Mask the background page
 */
.uk-offcanvas {
  /* 1 */
  display: none;
  /* 2 */
  position: fixed;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 1000;
  /* 3 */
  touch-action: none;
  /* 4 */
  background: rgba(0, 0, 0, 0.1);
}
.uk-offcanvas.uk-active {
  display: block;
}
/* Sub-object `uk-offcanvas-page`
 ========================================================================== */
/*
 * Prepares the whole HTML page to slide-out
 * 1. Fix the main page and disallow scrolling
 * 2. Side-out transition
 */
.uk-offcanvas-page {
  /* 1 */
  position: fixed;
  /* 2 */
  -webkit-transition: margin-left 0.3s ease-in-out;
  transition: margin-left 0.3s ease-in-out;
}
/* Sub-object `uk-offcanvas-bar`
 ========================================================================== */
```

Le istruzioni javascript necessarie per il corretto funzionamento del componente sono localizzate invece nel file **uikit.js**

```
            $ele.css(tmp); // reset element
        }

        return height;
    }

})(UIkit);

(function(UI) {

    "use strict";

    var scrollpos = {x: window.scrollX, y: window.scrollY},
        $win      = UI.$win,
        $doc      = UI.$doc,
        $html     = UI.$html,
        Offcanvas = {};

        show: function(element) {

            element = UI.$(element);

            if (!element.length) return;

            var $body     = UI.$('body'),
                bar       = element.find(".uk-offcanvas-bar:first"),
                rtl       = (UI.langdirection == "right"),
                flip      = bar.hasClass("uk-offcanvas-bar-flip") ? -1:1,
                dir       = flip * (rtl ? -1 : 1),

                scrollbarwidth =  window.innerWidth - $body.width();

            scrollpos = {x: window.pageXOffset, y: window.pageYOffset};

            element.addClass("uk-active");

            $body.css({"width": window.innerWidth - scrollbarwidth, "height": window.innerHeight}).addClass("uk-offcanvas-page");
            $body.css((rtl ? "margin-right" : "margin-left"), (rtl ? -1 : 1) * (bar.outerWidth() * dir)).width(); // .width() - force redraw

            $html.css('margin-top', scrollpos.y * -1);

            bar.addClass("uk-offcanvas-bar-show");

            this._initElement(element);

            bar.trigger('show.uk.offcanvas', [element, bar]);
```

ATTENZIONE! Il componente Off-canvas è un componente javascript per cui, affinchè possa funzionare in maniera corretta è necessario verificare di aver inserito nella sezione < head > della pagina non solo il collegamento alla libreria uikit.css ma anche quello alla libreria javascript uikit.js

In realtà, nel caso in cui questo dovesse essere l'unico componente javascript del framework che si intende implementare all'interno del sito, si potrebbe pensare di utilizzare, non l'intera libreria uikit.js, ma solamente la libreria relativa allo specifico componente ossia il file **offcanvas.js** (o meglio ancora la sua versione minificata).

In questo modo è vero si che otterremo un vantaggio (seppur piccolo) in termini di prestazioni della pagina (la libreria offcanvas.js ha sicuramente un peso minore rispetto al file uikit.js) occorre però fare molta attenzione nell'adottare questa soluzione, perché nel momento in cui dovessimo poi utilizzare un qualsiasi altro componente javascript la presenza, all'interno del nostro sito, della sola libreria offcanvas.js, non ne garantirà ovviamente il corretto funzionamento.

ATTENZIONE! Nel caso in cui non si dovesse essere assolutamente sicuri di quale libreria javascript caricare si consiglia di utilizzare sempre l'intera libreria uikit.js (magari nella sua versione minificata)

CONFIGURAZIONE

Il menu Off-Canvas è costituito, essenzialmente, da tre diversi componenti, nello specifico:

- Un elemento utilizzato per attivare la visualizzazione del menu di navigazione
- Una Sidebar (barra laterale) inizialmente posizionata al di fuori della pagina e la cui visualizzazione è attivata dal precedente elemento
- Una Off-canvas bar interna alla precedente Sidebar utilizzata per gestire il menu di navigazione

Per quel che riguarda l'elemento necessario ad attivare la visualizzazione del menu è sufficiente utilizzare un semplice link, quindi un **tag a , in cui la destinazione del collegamento (valore della proprietà href) dovrà essere esattamente l'id dell'elemento utilizzato per gestire la Sidebar.**

Inoltre per poter attivare tutte le funzioni javascript necessarie per visualizzare la Sidebar è necessario anche assegnare al link l'attributo aggiuntivo **data-uk-offcanvas**

ATTENZIONE! In assenza dell'attributo **data-uk-offcanvas** non verranno attivate le relative funzioni javascript e non sarà quindi possibile visualizzare il menu di navigazione

Per quel che riguarda la **Sidebar** è necessario utilizzare invece un semplice tag **div** al quale assegnare la classe **.uk-offcanvas**

Infine, **all'interno della Sidebar** è necessario inserire un ulteriore tag **div** necessario per gestire l'Off-canvas bar e al quale assegnare la classe **.uk-offcanvas-bar**

ATTENZIONE! Il menu di navigazione vero e proprio andrà inserito all'interno della div con classe .**uk-offcanvas-bar**

In definitiva dunque, posto di aver caricato correttamente sia le librerie css che quelle js, per implementare il sistema di navigazione off-canvas potremo fare riferimento ad un markup di questo tipo

```
<!-- Link utilizzato per attivare la visualizzazione della Sidebar -->
<a href="#sidebar-id" data-uk-offcanvas>
  Menu
</a>

<!-- Sidebar -->
<div id="sidebar-id " class="uk-offcanvas">
   <!-- Off-canvas Bar -->
   <div class="uk-offcanvas-bar">
     /*Inserire qui il menu di navigazione*/
   </div>
</div>
```

In queste condizioni cliccando sul link "Menu" il corpo principale del documento scorrerà verso destra lasciando il posto sulla sinistra al Sidebar ed al menu di navigazione in essa contenuto.

Volendo è anche possibile invertire questo comportamento facendo scorrere il corpo principale della pagina verso sinistra e facendo comparire la Sidebar, con il relativo menu di navigazione, sulla destra.

Per ottenere questo risultato è sufficiente assegnare al contenitore utilizzato per gestire la Off-canvas bar, oltre alla classe .uk-offcanvas-bar anche la classe .**uk-offcanvas-bar-flip**

In tutto ciò rimane però ancora aperto un piccolo problema.

Nelle condizioni considerate il link utilizzato per attivare la visualizzazione del menu è, ovviamente, sempre visibile per cui questo particolare sistema di navigazione sarebbe attivo indipendentemente dal fatto che il sito venga visualizzato sullo schermo di un pc, su di un tablet oppure su uno smartphone.

Posto che questo potrebbe benissimo essere il risultato che si vuole ottenere, va comunque detto che, generalmente, per la visualizzazione del sito su schermi di grandi dimensioni si preferisce utilizzare un menu di navigazione "classico" mentre la navigazione Off-canvas viene attivata soltanto nel momento in cui il sito dovesse essere visualizzato su schermi di piccole dimensioni.

Il problema è comunque facilmente risolvibile.

La Sidebar contenente il menu di navigazione nasce infatti già al di fuori della pagina web per cui l'unica cosa da fare è da una parte, gestire la visualizzazione dell'elemento utilizzato per la richiamare la Sidebar soltanto su schermi di piccole dimensioni, dall'altra parte preoccuparsi di nascondere, sempre su schermi di piccole dimensioni, il sistema di navigazione classico.

In entrambi i casi è sufficiente ricorre alle classi utilizzate da uikit per gestire la visibilità responsiva (per maggiori informazioni in merito si veda anche il capitolo " *Utility – Visibilità responsiva*" di questa guida)

In definitiva dunque il markup completo, al quale fare riferimento per poter gestire un sistema di navigazione Off-canvas, potrebbe essere del tipo di quello qui di seguito indicato

```
<!-- Link utilizzato per attivare la visualizzazione della Sidebar -->
<a href="#sidebar-id" data-uk-offcanvas class="uk-visible-small ">
  Menu
</a>

<!-- Sidebar -->
<div id="sidebar-id " class="uk-offcanvas">
   <!-- Off-canvas Bar -->
   <div class="uk-offcanvas-bar">
     /*Inserire qui il menu di navigazione*/
   </div>
</div>
```

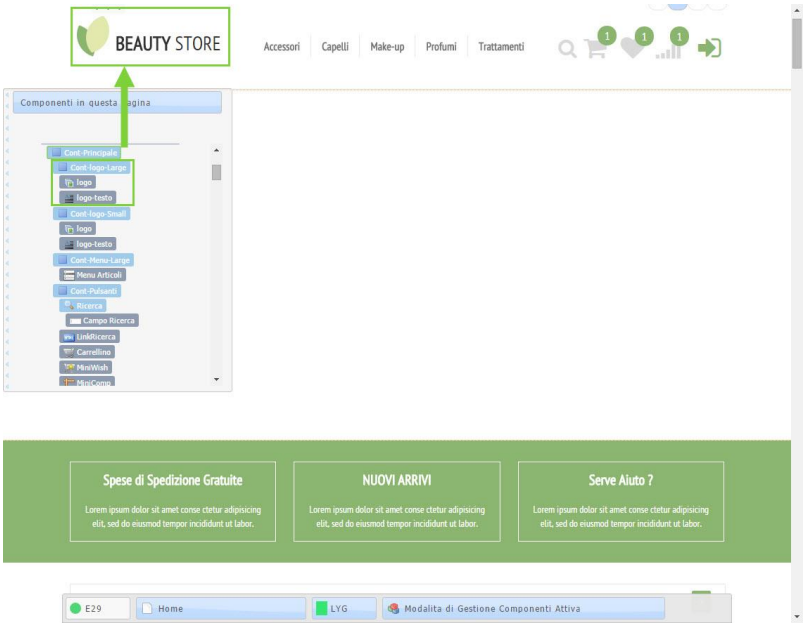
dove la classe .**uk-visible-small** consente di visualizzare il link per attivare la visualizzazione della sidebar solo nel caso in cui il viewport sia minore di 768px

PASSWEB NAVIGAZIONE OFF CANVAS – MODELLO ECOMMERCE 29

Per comprendere come poter implementare all'interno di un sito Passweb un sistema di navigazione Off-canvas analogo a quello descritto nel capitolo precedente, possiamo utilizzare direttamente l'**Home Page** del nostro modello di riferimento, fissando la nostra attenzione sugli elementi presenti in Testata.

In Testata a questa pagina sono stati infatti inseriti diversi componenti. Nello specifico possiamo notare:

- Un Componente Contenitore, **Cont-logo-Large**, all'interno del quale sono stati inseriti gli elementi necessari per gestire il logo

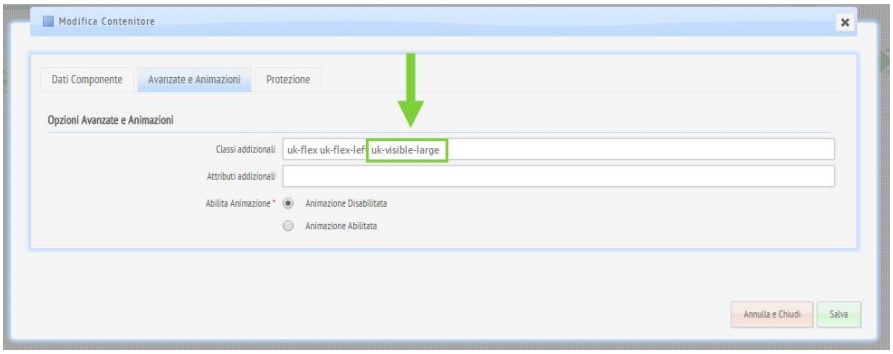


Ora, il progetto di base del sito prevede che nel momento in cui dovessero diminuire le dimensioni del viewport, arrivando a raggiungere valori inferiori ai 960px, il colore di sfondo della Testata dovrà cambiare passando da bianco a verde.

Nelle stesse condizioni (viewport inferiori a 960px) sarà quindi necessario modificare gli elementi del logo in modo tale che questo possa essere ben visibile anche su sfondo verde.

Per risolvere questo problema è stato scelto di utilizzare le classi di uikit relative alla visibilità responsiva.

E' stata quindi assegnata al Contenitore **Cont-logo-Large** la classe **.uk-visible-large**

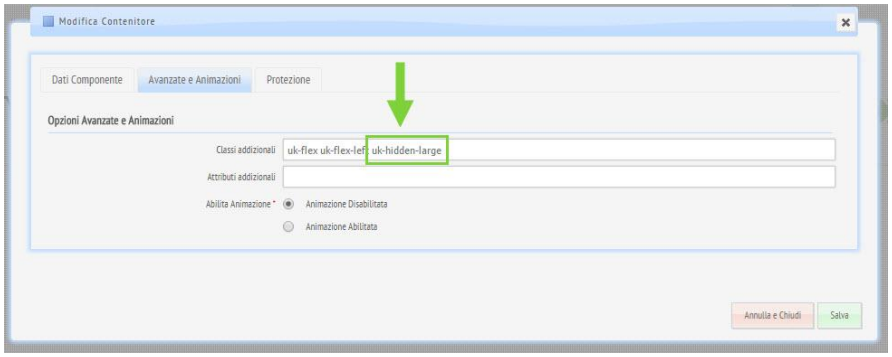


in modo tale che per viewport inferiori a 960px il Contenitore in esame e con esso tutti i suoi contenuti venga effettivamente nascosto.

A questo punto si rende però necessario inserire un nuovo Componente Contenitore analogo a quello appena esaminato, che contenga gli elementi necessari per visualizzare correttamente il logo su sfondo verde e che, soprattutto sia visualizzato solo per viewport inferiori a 960px

- Un Componente Contenitore, **Cont-logo-Small**, all'interno del quale sono stati inseriti gli elementi necessari per gestire il logo su sfondo verde

Sulla base di quanto detto al punto precedente al Contenitore **Cont-logo-Small** è stata poi assegnata la classe **.uk-hidden-large**



in maniera tale da poterlo effettivamente visualizzare, assieme a tutti i suoi contenuti, solo per dimensioni del viewport inferiori a 960px (quando cioè lo sfondo della Testa diventerà verde).

A questo punto è bene fare una considerazione che potrebbe anche sembrare ovvia ma che risulta di fondamentale importanza per poter lavorare al meglio con gli strumenti messi a disposizione da Passweb.

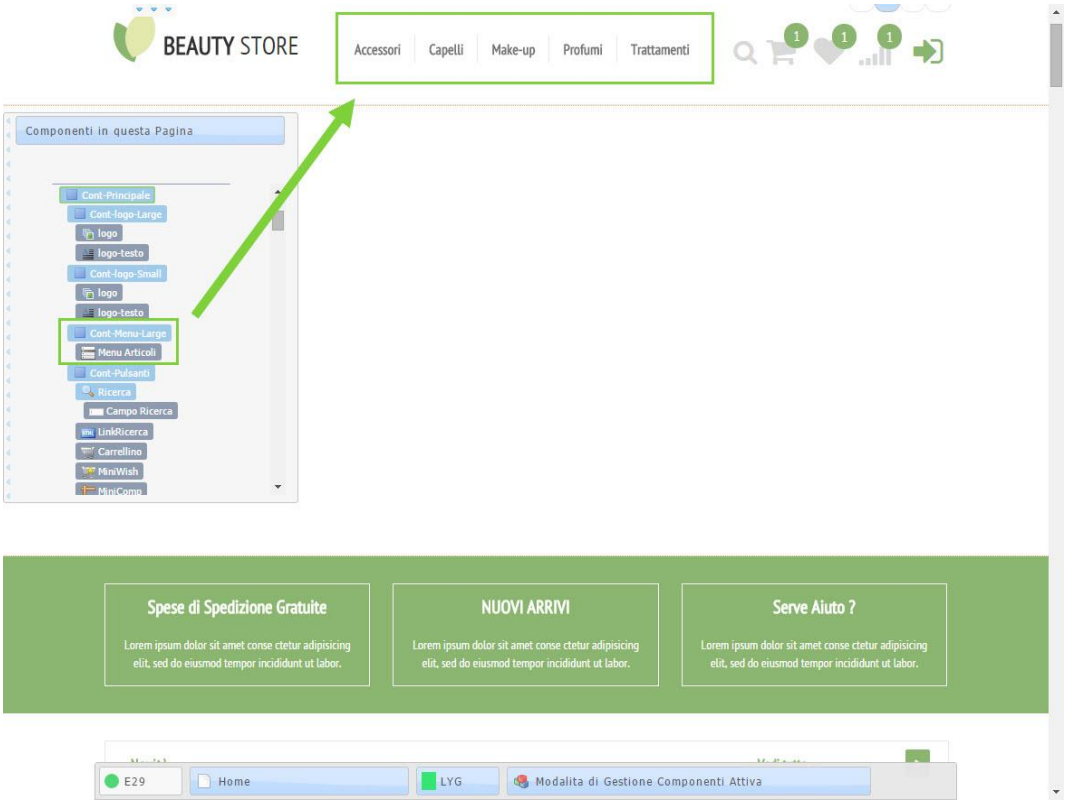
Nel momento in cui andiamo ad assegnare al Contenitore **Cont-logo-Small** la classe **.uk-hidden-large** questo ovviamente "scomparirà" dal Wizard. Tale classe consente infatti di visualizzare il Componente cui viene associata solo per viewport inferiori a 960px mentre lo schermo del pc sul quale si lavora per costruire il sito ha tipicamente, un viewport superiore.

Il risultato di tutto ciò è che dopo aver assegnato la classe .uk-hidden-large al nostro Contenitore questo non sarà più visibile nel Live Editing di Passweb e quindi non potremo più selezionarlo direttamente passandoci sopra con il mouse.

Il Componente continuerà comunque ad essere selezionabile e gestibile dall'albero dei componenti di Passweb ma, se da una parte questo è perfettamente sufficiente, ad esempio, per poter accedere al suo Style Editor dall'altra parte non è invece sufficiente nel momento in cui dovessimo inserire al suo interno altri componenti.

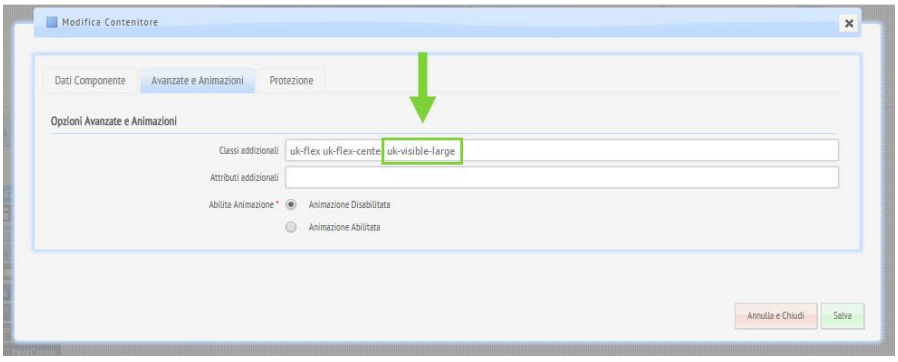
In generale dunque nel momento in cui si dovesse decidere di assegnare a Componenti di Passweb una classe che ne abilità la visibilità solo per viewport di piccole dimensioni, prima di fare ciò è necessario gestire e posizionare correttamente tutti quelli che dovranno essere eventuali componenti interni ad esso.

- Un Componente Contenitore, **Cont-Menu-Large**, all'interno del quale è stato inserito un menu di navigazione "classico" .

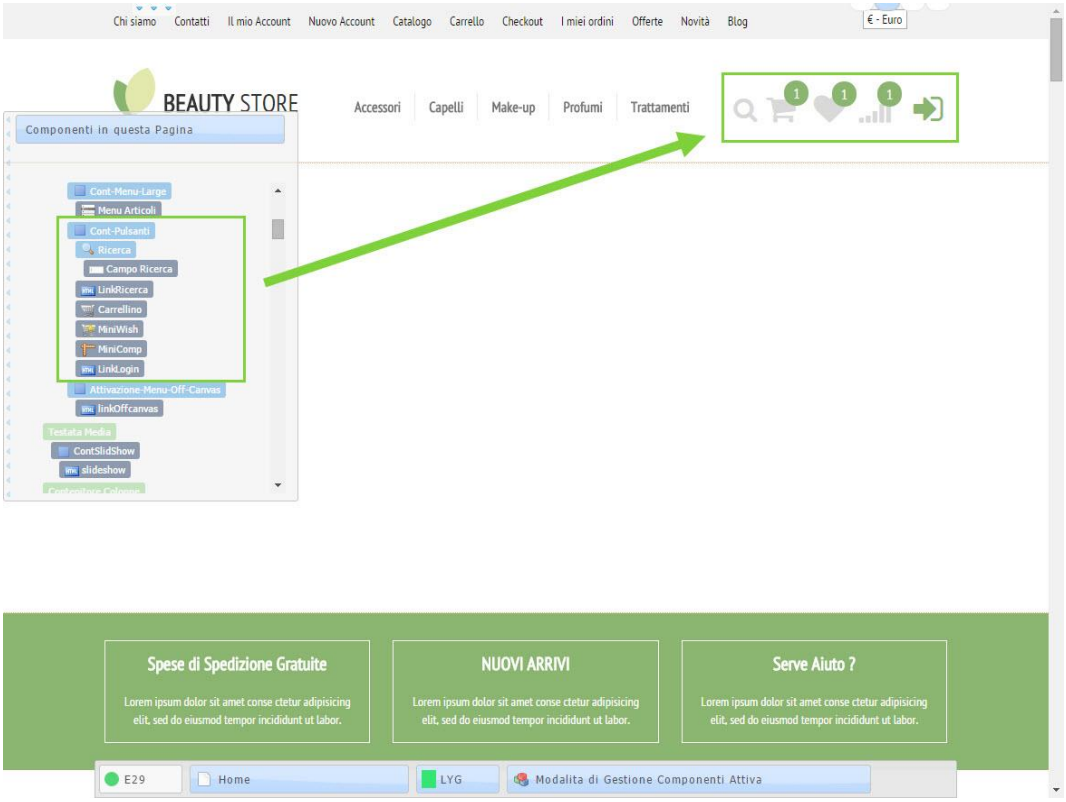


Tale menu dovrà essere visualizzato solo su schermi di grandi dimensioni mentre per schermi di piccole dimensioni dovrà essere sostituito dal sistema di navigazione Off-Canvas.

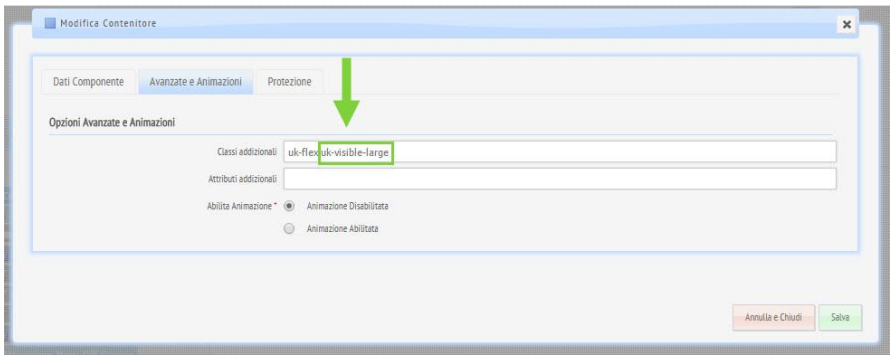
Per questa ragione anche al Contenitore **Cont-Menu-Large** è stata assegnata la classe **.uk-visible-large**



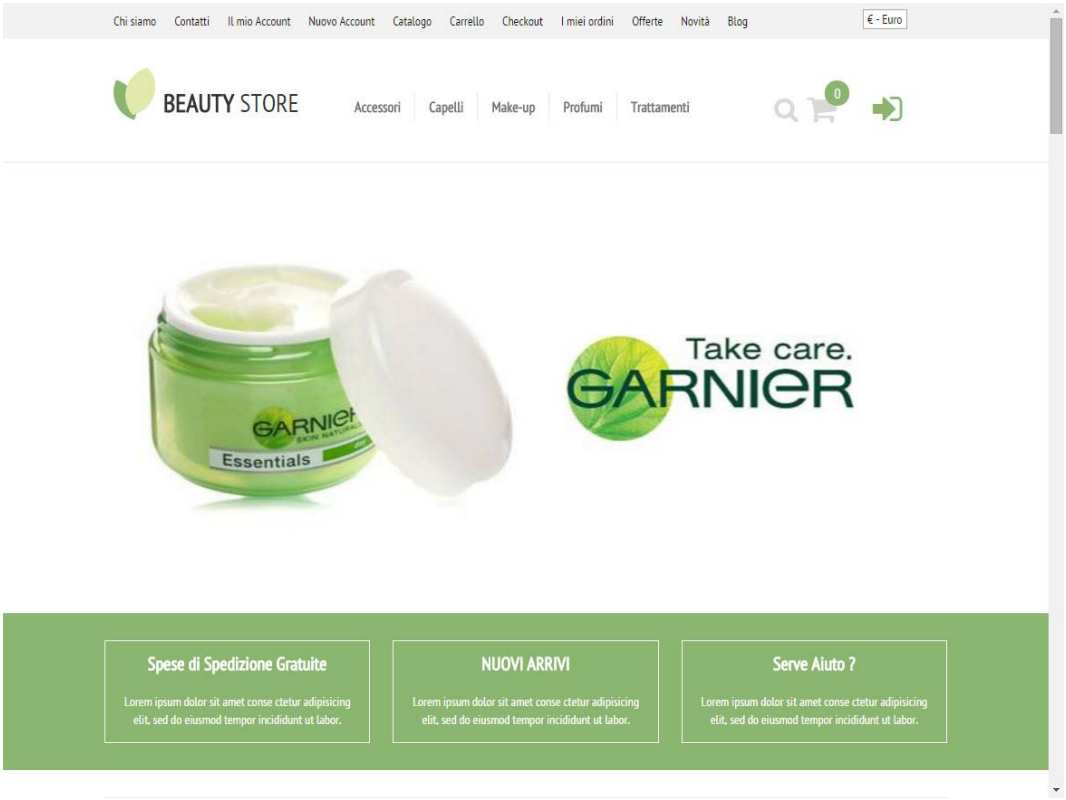
- Un Componente Contenitore, **Cont-Pulsanti**, all'interno del quale sono stati inseriti alcuni elementi di interazione, un Pannello di Ricerca, il Carrellino, la Mini Wishlist ecc…



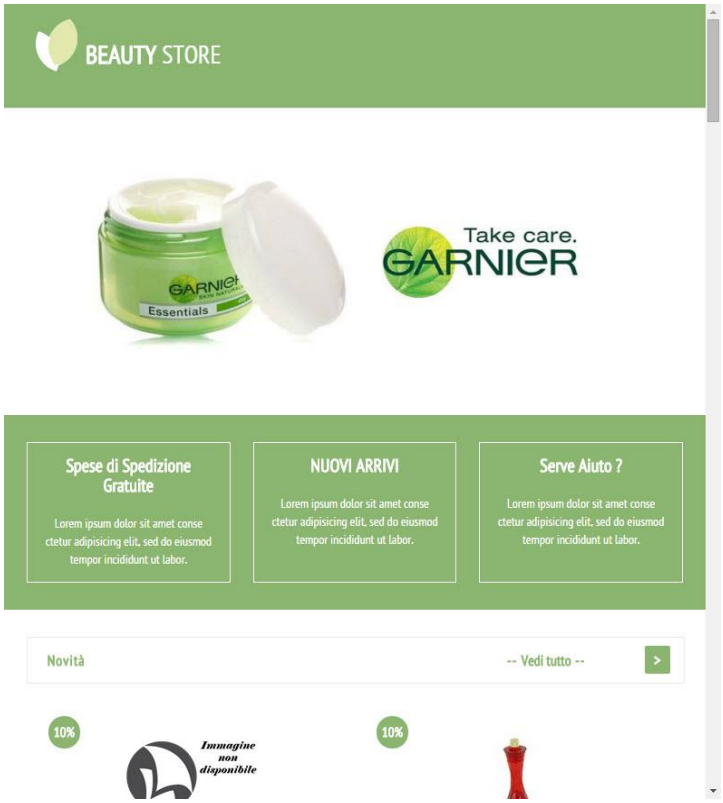
Anche questi elementi dovranno essere visualizzati solo su schermi di grandi dimensioni e quindi anche al Contenitore **Cont-Pulsanti** è stata assegnata la classe **.uk-visible-large**



Sulla base di quanto visto fino a questo momento la Testata del nostro sito, su schermi di grandi dimensioni apparirà come in figura



Per viewport inferiori a 960px avremo invece una situazione di questo tipo



Ci resta quindi da implementare il sistema di navigazione Off-Canvas facendo in modo di attivarlo solo per viewport inferiori a 960px

In questo senso sarà quindi sufficiente seguire passo passo la procedura di seguito indicata:

- Inserire nel piede della pagina un Componente Contenitore (**Sidebar**) da utilizzare per gestire la Sidebar al quale dovrà quindi essere assegnata, come noto, la classe .uk-offcanvas**.**

  Considerando però che la Sidebar verrà posizionata fuori dalla pagina, nel momento in cui andremo ad assegnare al nostro Componente Contenitore questa classe poi non avremo più la possibilità di selezionarlo dal Live Editing (se non utilizzando la

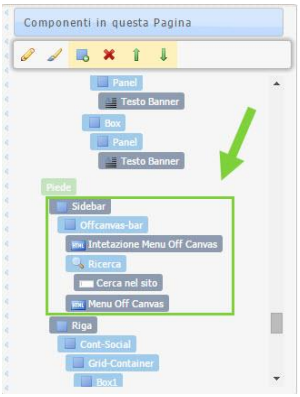
barra laterale dei componenti) e non potremo quindi inserire al suo interno i componenti necessari per completare il sistema di navigazione Off-Canvas.

Prima di assegnare quindi la classe .uk-offcanvas a questo Contenitore dovremo gestire i componenti interni ad esso

ATTENZIONE! La Sidebar è stata inserita per semplicità di gestione nel piede della pagina ma potrebbe anche essere collocata in una qualsiasi altra posizione.

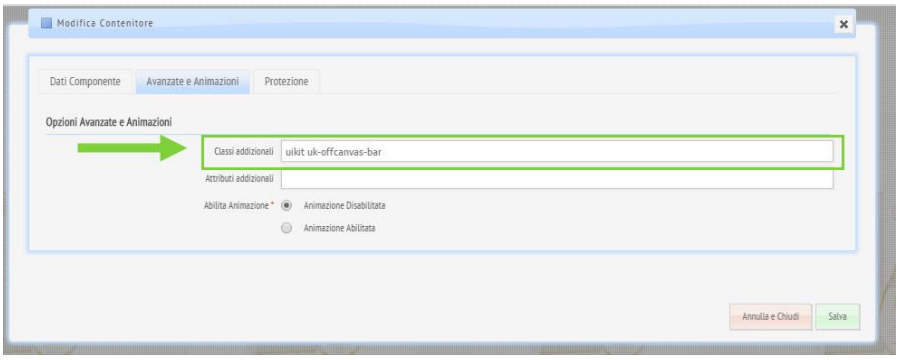
- Inserire all'interno del Contenitore precedente un nuovo Componente Contenitore (**Offcanvas-bar**) da utilizzare per gestire la Off-canvas bar
- Inserire all'interno della Off-canvas bar il menu di navigazione (realizzato nel nostro caso mediante il componente HTML "**Menu Off Canvas**") più eventuali ulteriori componenti che dovranno essere visualizzati assieme ad esso nella Sidebar laterale.

  Nel nostro caso, ad esempio, oltre al menu di Navigazione, è stato inserito anche un Pannello di Ricerca.

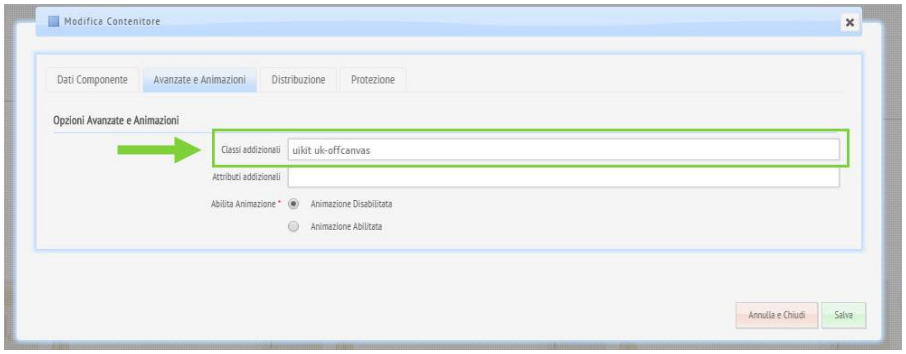


In queste condizioni gli elementi del sistema di navigazione sono ancora tutti perfettamente visibili nel Live Editing di Passweb e possono quindi essere stilizzati come "normali" componenti Passweb

- Assegnare al Componente Contenitore utilizzato per gestire la Off-canvas bar le seguenti classi aggiuntive:
  - **.uikit:** necessaria per gestire il float: none
  - **.uk-offcanvas-bar:** necessaria per individuare la Offcanvas bar



- Assegnare al Componente Contenitore utilizzato per gestire la Sidebar le seguenti classi aggiuntive:
  - **.uikit:** necessaria per gestire il float: none
  - **.uk-offcanvas:** necessaria per individuare la Sidebar

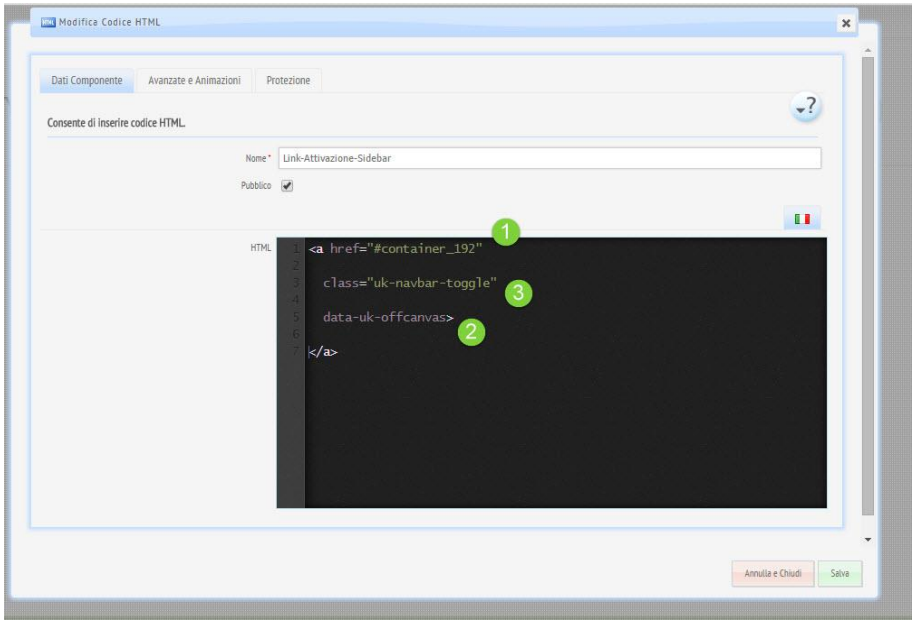


ATTENZIONE! A questo punto la Sidebar (così come i componenti interni ad essa) non sarà più visibile nel Live Editing del sito

Per completare il sistema di navigazione Off-Canvas l'ultimo passo da fare è ora quello di inserire il pulsante mediante il quale poter attivare la visualizzazione della Sidebar laterale.

Volendo gestire tale pulsante mediante un semplice link sarà quindi necessario:

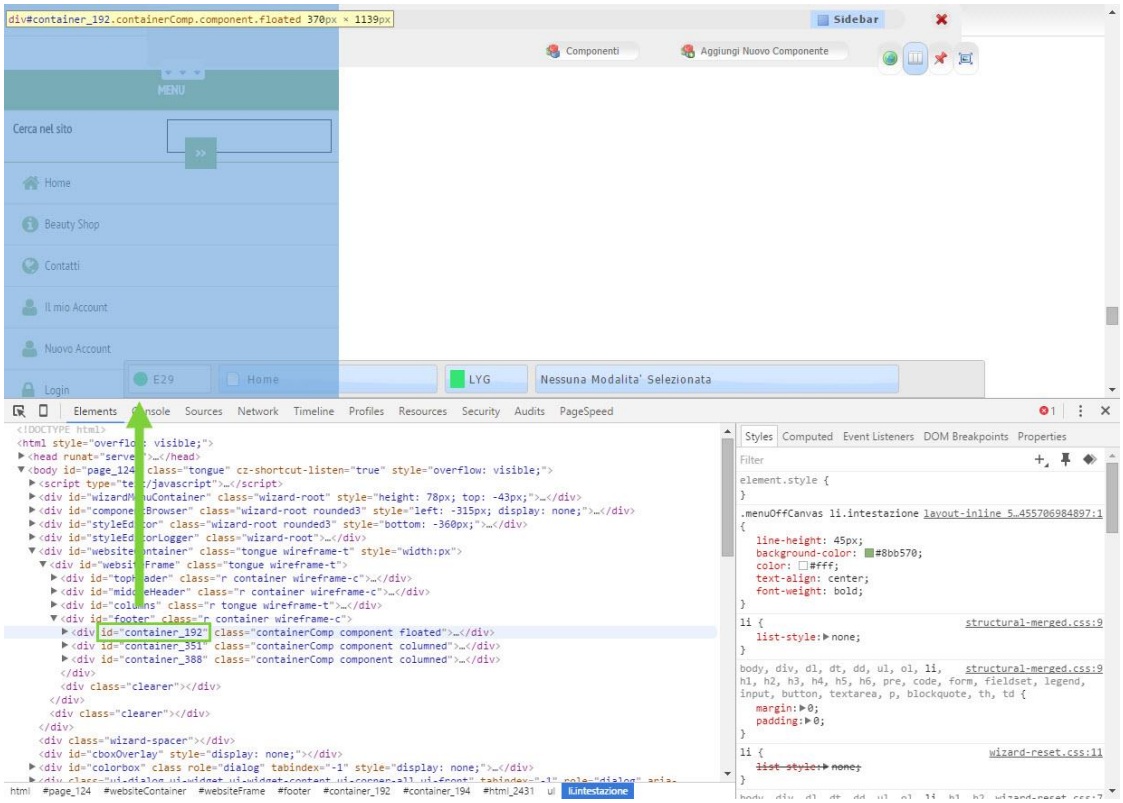
- Inserire in Testata un Componente HTML (**Link-Attivazione-Sidebar**) ed utilizzarlo per creare un tag < a > come evidenziato in figura



In particolare è necessario verificare di:

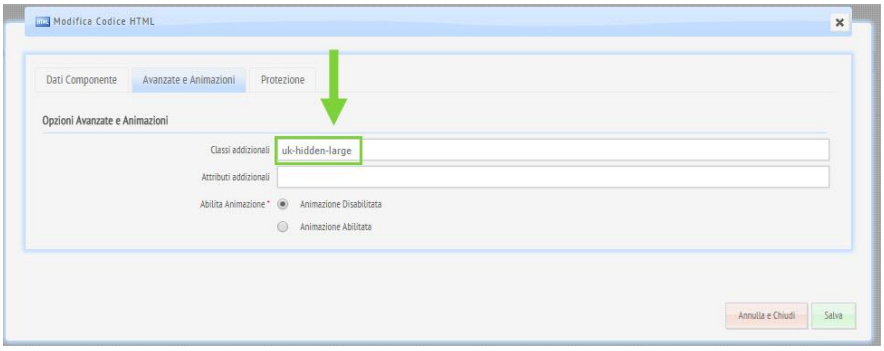
- Utilizzare come **destinazione del collegamento l'id del Componente Contenitore utilizzato per gestire la Sidebar** (nell'esempio in figura **#conteiner_192**) (1)

  ATTENZIONE! Per reperire corretto identificativo della Sidebar è necessario utilizzare gli strumenti per sviluppatore del browser



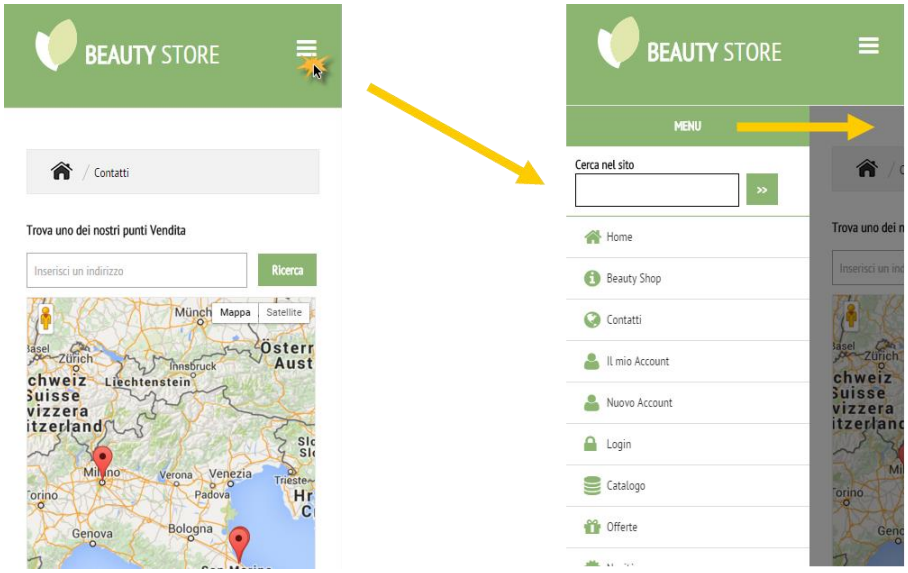
- Assegnare al tag < a > l'attributo **data-uk-offcanvas** (2)
- La classe **.uk-navbar-toggle** assegnata al al tag < a > (3) consente infine di utilizzare come contenuto del link l'icona tipicamente utilizzata nei siti mobile per identificare un menu di navigazione (☰)

- Assegnare al componente HTML di cui al punto precedente, la classe aggiuntiva **.uk-hidden-large**



In questo modo il componente in oggetto verrà visualizzato solo per viewport inferiori a 960px e di conseguenza la Sidebar laterale potrà essere attivata solo ed esclusivamente nelle stesse condizioni.

Alla fine di tutto il processo, visualizzando il sito su viewport inferiori a 960px otterremo un risultato analogo a quello evidenziato in figura

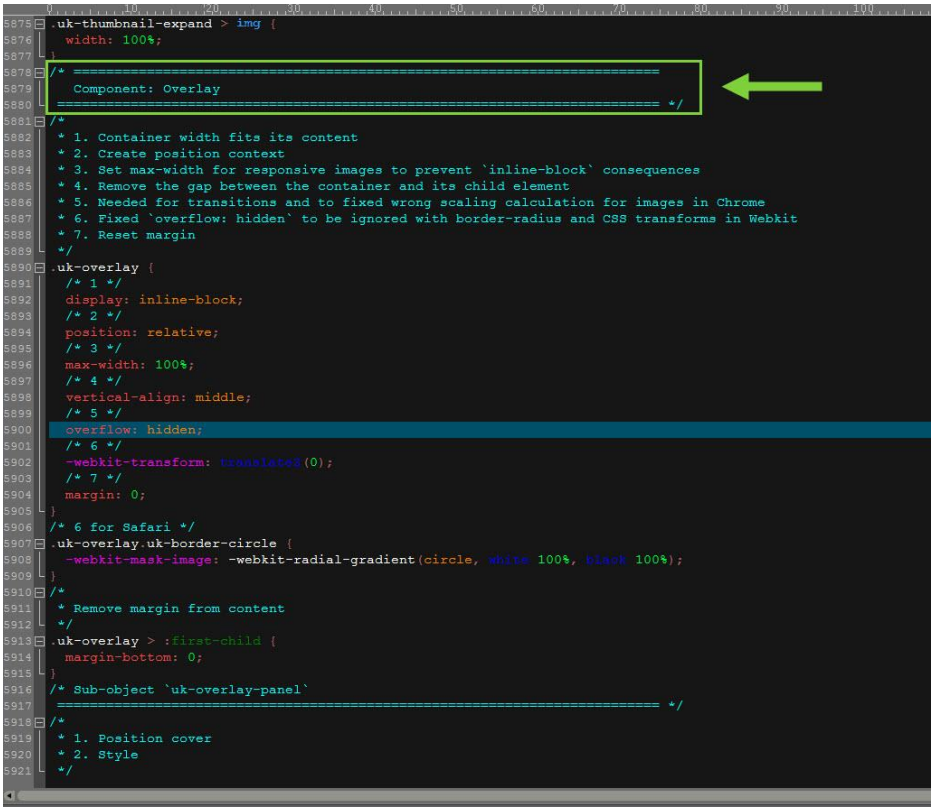


OVERLAY

Il componente Overlay consente di creare e gestire facilmente immagini con effetto di overlay e stili differenti

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/overlay.html

Le regole CSS relative a questo componente sono localizzate tutte nel file uikit.css all'interno della sezione "**Component: Overlay**"



```
.uk-thumbnail-expand > img {
    width: 100%;
}
/* ========================================================================
   Component: Overlay
 ========================================================================== */
/*
 * 1. Container width fits its content
 * 2. Create position context
 * 3. Set max-width for responsive images to prevent `inline-block` consequences
 * 4. Remove the gap between the container and its child element
 * 5. Needed for transitions and to fixed wrong scaling calculation for images in Chrome
 * 6. Fixed `overflow: hidden` to be ignored with border-radius and CSS transforms in Webkit
 * 7. Reset margin
 */
.uk-overlay {
    /* 1 */
    display: inline-block;
    /* 2 */
    position: relative;
    /* 3 */
    max-width: 100%;
    /* 4 */
    vertical-align: middle;
    /* 5 */
    overflow: hidden;
    /* 6 */
    -webkit-transform: translate3d(0);
    /* 7 */
    margin: 0;
}
/* 6 for Safari */
.uk-overlay.uk-border-circle {
    -webkit-mask-image: -webkit-radial-gradient(circle, white 100%, black 100%);
}
/*
 * Remove margin from content
 */
.uk-overlay > :first-child {
    margin-bottom: 0;
}
/* Sub-object `uk-overlay-panel`
 ========================================================================== */
/*
 * 1. Position cover
 * 2. Style
 */
```

Nei successivi capitoli esamineremo solo alcune delle possibili opzioni di configurazione di questo componente e vedremo anche come poterle implementare in Passweb.

In ogni caso per una trattazione completa di tutte le diverse possibili opzioni di configurazione e, soprattutto, per poter visualizzare degli esempi di utilizzo si consiglia di fare riferimento alla relativa pagina della documentazione ufficiale http://getuikit.com/docs/overlay.html

CONFIGURAZIONE

Di base l'implementazione di questo componente è estremamente semplice.

Si tratta infatti di:

- Utilizzare un contenitore esterno per l'immagine cui deve essere applicato l'effetto di overlay e assegnarli la classe **.uk-overlay**
- Inserire all'interno di questo contenitore oltre all'immagine, anche un altro contenitore necessario per gestire il contenuto dell'overlay assegnando a quest'ultimo la classe **.uk-overlay-panel**
- Inserire il contenuto dell'overlay nel contenitore con classe .uk-overlay-panel

In definitiva dunque per visualizzare del semplice testo in overlay ad un'immagine

potremmo considerare un markup di questo tipo

```
<div class="uk-overlay">
   <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
   <div class="uk-overlay-panel">
        Testo in overlay all'imamgine
   </div>
</div>
```

ATTENZIONE nella configurazione di default di uikit il testo utilizzato per l'overlay sulle immagini è di colore bianco, per cui se viene utilizzata un'immagine anch'essa a sfondo bianco il testo in overlay potrebbe non essere visibile.

OVERLAY SULL'HOVER

Nella sua configurazione di base gli elementi di overlay sull'immagine sono sempre visibili.

Un'esigenza piuttosto comune è invece quella **di visualizzare il contenuto dell'overlay solo nel momento in cui si dovesse passare con il mouse sull'immagine**.

Per ottenere questo risultato è sufficiente assegnare al contenitore esterno dell'immagine oltre alla classe .uk-overlay anche la classe anche la classe **.uk-overlay-hover**

In queste condizioni dunque il markup cui fare riferimento dovrà essere di questo tipo

```
<div class="uk-overlay uk-overlay-hover">
   <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
   <div class="uk-overlay-panel">
        Testo in overlay all'imamgine
   </div>
</div>
```

Per poter consultare un esempio in tal senso si consiglia di fare riferimento alla relativa pagina della documentazione ufficiale
http://getuikit.com/docs/overlay.html#overlay-on-hover-active

OVERLAY CON BACKGROUND

Volendo è possibile gestire l'overlay su di un immagine assegnandogli anche un background in modo tale da farlo risaltare meglio

Per poter ottenere questo risultato è sufficiente assegnare **al contenitore utilizzato per gestire i contenuti dell'overlay** la classe **.uk-overlay-background**

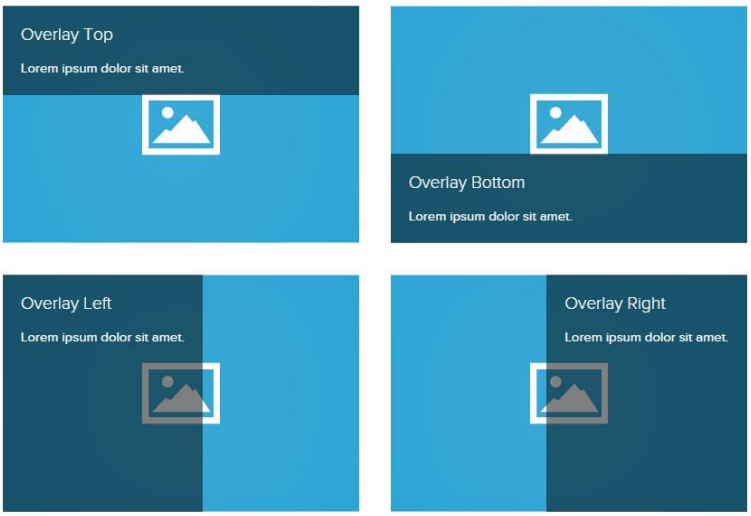
In queste condizioni dunque il markup cui fare riferimento potrebbe essere di questo tipo

```
<div class="uk-overlay">
   <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
   <div class="uk-overlay-panel uk-overlay-background">
        Testo in overlay all'imamgine
   </div>
</div>
```

POSIZIONE DELL'OVERLAY

Nella sua configurazione di base l'overlay va ad occupare tutto lo spazio riservato al contenitore esterno dell'immagine.

Se l'esigenza dovesse invece essere quella di ottenere un overlay che non copra interamente l'immagine cui viene applicato e che possa essere anche collocato in posizioni diverse, come mostrato in figura



sarà necessario allora assegnare **al contenitore utilizzato per gestire i contenuti dell'overlay** una delle seguenti classi:

- **.uk-overlay-top**: consente di allineare l'overlay alla parte alta dell'immagine
- **.uk-overlay-bottom:** consente di allineare l'overlay alla parte bassa dell'immagine
- **.uk-overlay-left:** consente di allineare l'overlay sulla sinistra dell'immagine
- **.uk-overlay-right:** consente di allineare l'overlay sulla destra dell'immagine

In queste condizioni dunque il markup cui fare riferimento potrebbe essere di questo tipo

```
<div class="uk-overlay">
   <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
   <div class="uk-overlay-panel uk-overlay-background uk-overlay-bottom">
        Testo in overlay all'imamgine
   </div>
</div>
```

OVERLAY CON TRANSIZIONI

Un'ultima esigenza, anch'essa piuttosto comune, potrebbe essere quella di visualizzare l'overlay al passaggio del mouse sull'immagine utilizzando anche specifici effetti di transizione come ad esempio il fade, lo slide ecc…

Per poter ottenere questo risultato è sufficiente assegnare **al contenitore utilizzato per gestire i contenuti dell'overlay** una delle seguenti classi:

- **.uk-overlay-slide-top:** consente di animare la visualizzazione del pannello di overlay con effetto slide che compare dall'alto
- **.uk-overlay-slide-bottom:** consente di animare la visualizzazione del pannello di overlay con effetto slide che compare dal basso
- **.uk-overlay-slide-left:** consente di animare la visualizzazione del pannello di overlay con effetto slide che compare da sinistra
- **.uk-overlay-slide-right:** consente di animare la visualizzazione del pannello di overlay con effetto slide che compare da destra
- **.uk-overlay-fade:** consente di animare la visualizzazione del pannello di overlay con effetto fade
- **.uk-overlay-scale:** consente di animare la visualizzazione del pannello di overlay con effetto scale
- **.uk-overlay-spin:** consente di animare la visualizzazione del pannello di overlay con di rotazione sulla destra
- **.uk-overlay-grayscale:** consente di animare la visualizzazione del pannello di overlay con una desaturazione del colore

ATTENZIONE! Volendo queste classi possono essere applicate anche all'immagine ottenendo come risultato quello di animare, con il relativo effetto, anche la visualizzazione dell'immagine stessa

Nel caso in cui volessimo dunque animare la visualizzazione del pannello di layout il markup cui fare riferimento potrebbe essere di questo tipo

```
<div class="uk-overlay uk-overlay-hover">
   <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4">
   <div class="uk-overlay-panel uk-overlay-background uk-overlay-fade">
        Testo in overlay all'imamgine
   </div>
</div>
```

Nel caso in cui volessimo invece animare la visualizzazione sia del pannello del overlay che dell'immagine potremmo fare riferimento ad un markup di questo tipo:

```
<div class="uk-overlay uk-overlay-hover">
   <img src="/Resources/Graphics/blog-img4.jpg" alt="blog-img4" title="img4" class=" uk-overlay-slide-right">
   <div class="uk-overlay-panel uk-overlay-background uk-overlay-fade">
        Testo in overlay all'imamgine
   </div>
</div>
```

ATTENZIONE! Le classi sopra indicate funzionano, ovviamente, solo in combinazione con la classe .**uk-overlay-hover**

Per poter consultare alcuni esempi di overlay con effetti di transizione si consiglia di fare riferimento alla relativa pagina della documentazione ufficiale http://getuikit.com/docs/overlay.htm#overlay-transitions

OVERLAY IN PASSWEB

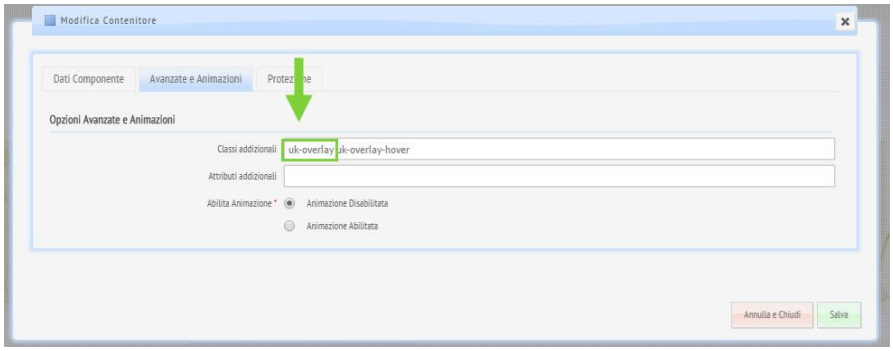
Per poter implementare all'interno del nostro sito Passweb il componente Overlay di uikit, in una qualsiasi delle configurazioni esaminate nei precedenti capitoli di questa guida, è possibile operare in modi diversi:

- Utilizzando un **Componente HTML** oppure la parte "**Sorgente**" di un **Componente Paragrafo**

  In entrambi i casi avremo accesso diretto al "codice" del componente per cui sarà sufficiente copiare ed incollare direttamente il markup HTML necessario per ottenere la configurazione desiderata.

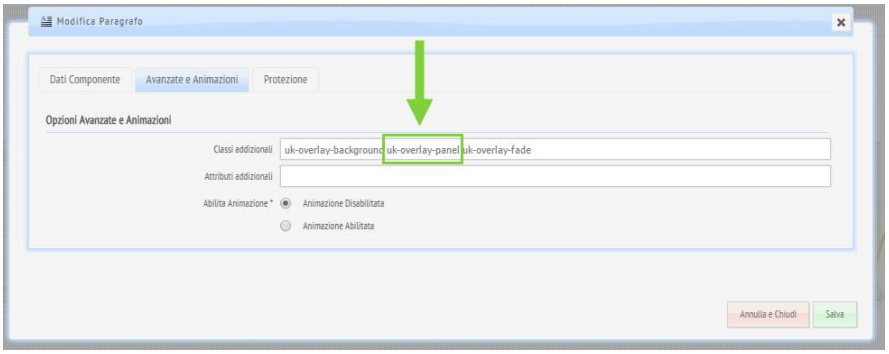
```
<div class="uk-overlay uk-overlay-hover">
    <img alt="" class="uk-overlay-scale" src="/Resources/Graphics/img/img-banner1.jpg" />
    <div class="uk-overlay-panel uk-overlay-background uk-overlay-fade">
        <h3 class="uk-panel-title uk-margin-small">
            Trendhair</h3>
        <p>
            BEAUTY</p>
    </div>
</div>
```

- Utilizzando un sistema di Componenti annidati costituito in particolare da:
    - Un primo **Componente Contenitore**, con posizionamento Affiancato, utilizzato come contenitore esterno dell'immagine, al quale assegnare, sicuramente, almeno la classe .**uk-overlay**



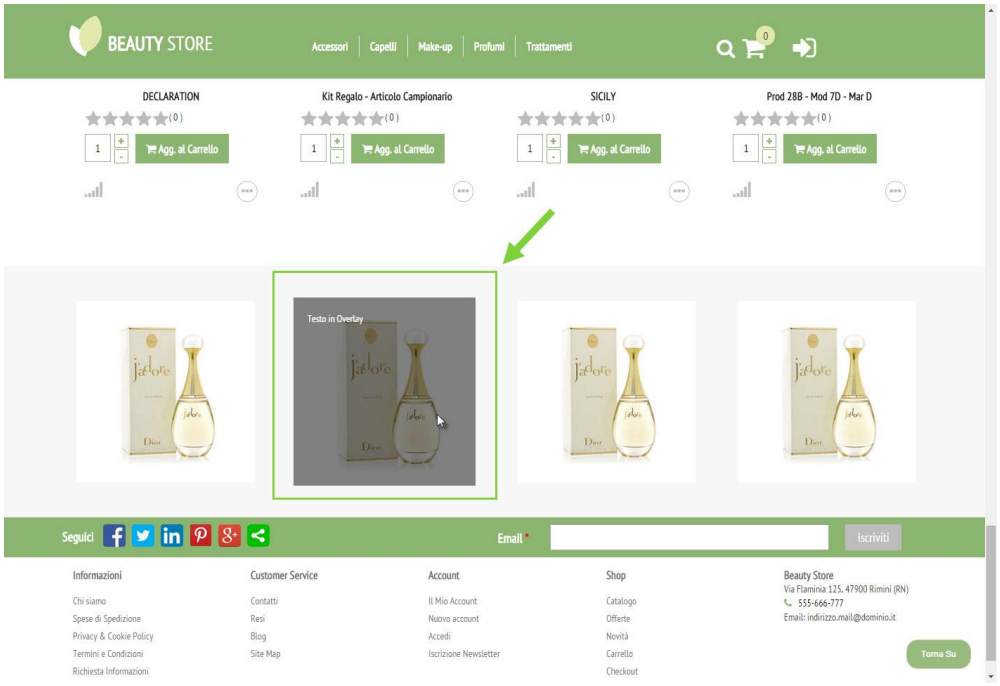
    - All'interno del precedente Contenitore, andrà poi inserito un **Componente Immagine** necessario per gestire l'immagine su cui visualizzare l'overlay e, ad esempio, un **Componente Paragrafo** utilizzato per gestire i contenuti del pannello di Overlay.

Al componente Paragrafo sarà necessario assegnare, sicuramente, almeno la classe .**uk-overlay-panel**



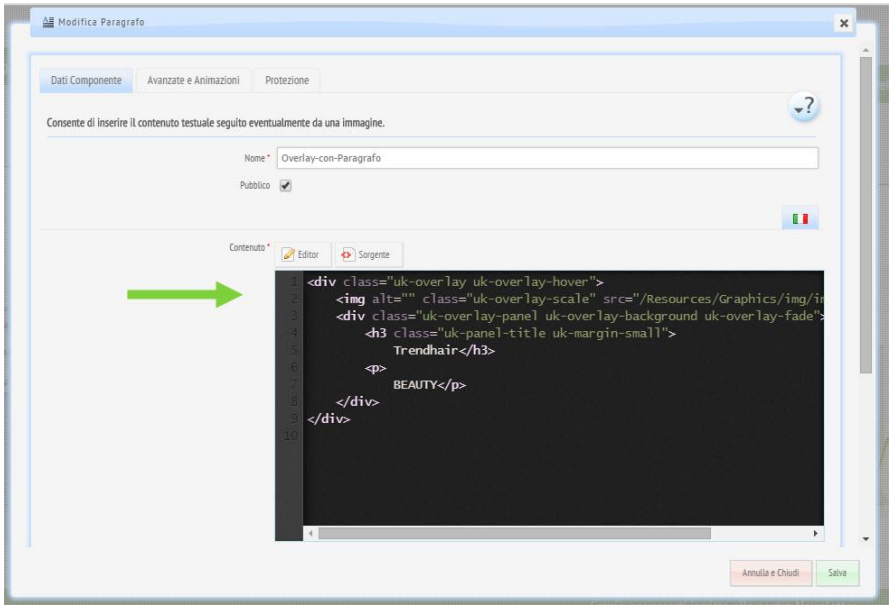
MODELLO ECOMMERCE 29

E' possibile trovare un esempio di applicazione del componente Overlay nella pagina "**Home**" del modello di riferimento.



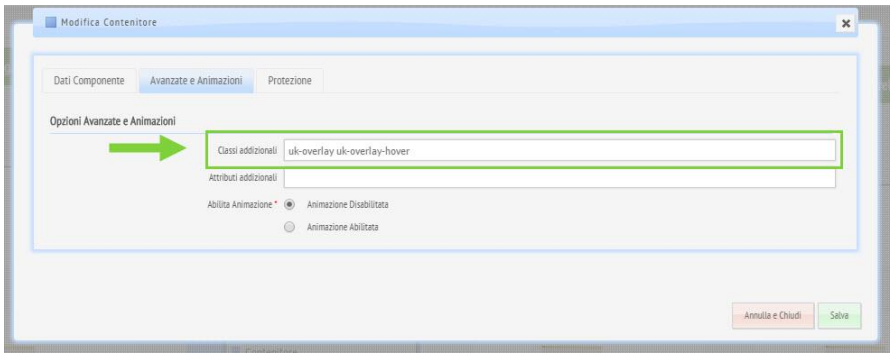
Nella parte bassa della pagina immediatamente al di spora del piede è stata utilizzata una Griglia responsiva di 4 colonne dove all'interno della prima colonna è stato inserito un Pannello con un Componente Paragrafo (**Overlay-con-Paragrafo**).

In configurazione del Componente Paragrafo, nella parte Sorgente è stato inserito il markup HTML necessario per ottenere un Overlay con background in cui al passaggio del mouse sull'immagine, l'immagine stessa si anima con effetto scale e il pannello di overlay compare con effetto fade



All'interno del pannello presente nella seconda colonna è stato invece inserito un primo Componente Contenitore (**Contenitore-Immagine**) utilizzato come contenitore esterno dell'immagine, al quale sono state quindi assegnate le classi:

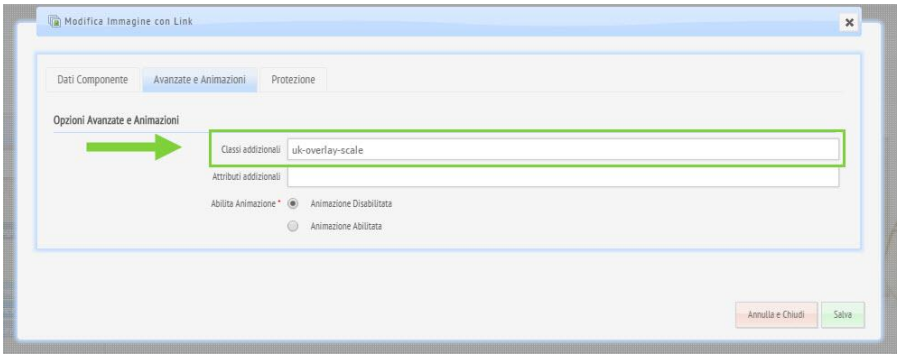
- **.uk-overlay:** necessaria per individuare il contenitore esterno dell'immagine
- **.uk-overlay-hover**: necessaria per visualizzare il pannello di overlay all'hover sull'immagine



All'interno di questo Contenitore è stato inserito un Componente Immagine (**Immagine-Overlay**), necessario per gestire l'immagine sulla quale visualizzare il pannello di Overlay, e un Componente Paragrafo (**Pannello-Overlay)** utilizzato per gestire il pannello di overlay

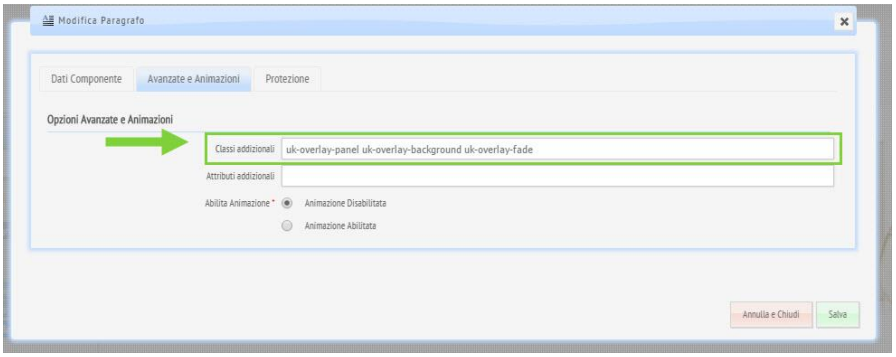
Considerando che si voleva ottenere anche in questo caso un Overlay con background in cui al passaggio del mouse sull'immagine, l'immagine stessa si anima con effetto scale e il pannello di overlay compare con effetto fade, al componente Immagine è stata assegnata la classe aggiuntiva

- **.uk-overlay-scale:** necessaria per animare l'immagine all'hover del mouse con effetto scale



mentre al Componente Paragrafo sono state assegnate le classi aggiuntive

- **.uk-overlay-panel:** necessaria per individuare il pannello di overlay
- **.uk-overlay-background:** necessaria per attivare un overlay con background
- **.uk-overlay-fade:** necessaria per animare la visualizzazione del pannello di overlay all'hover sull'immagine con effetto fade



MODAL

Il componente Modal consente di creare e gestire facilmente finestre modali con stili differenti.

E' un componente Javascript e necessita quindi, per poter essere utilizzato correttamente, anche della libreria uikit.js

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/modal.html

Le regole CSS relative a questo componente sono localizzate tutte nel file uikit.css all'interno della sezione "**Component: Modal**"

```
/* ========================================================================
   Component: Modal
 ======================================================================== */

/*
 * This is the modal overlay and modal dialog container
 * 1. Hide by default
 * 2. Set fixed position
 * 3. Allow scrolling for the modal dialog
 * 4. Mask the background page
 * 5. Fade-in transition
 * 6. Deactivate browser history navigation in IE11
 * 7. force hardware acceleration to prevent browser rendering hiccups
 */

.uk-modal {
  /* 1 */
  display: none;
  /* 2 */
  position: fixed;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 1010;
  /* 3 */
  overflow-y: auto;
  -webkit-overflow-scrolling: touch;
  /* 4 */
  background: rgba(0, 0, 0, 0.6);
  /* 5 */
  opacity: 0;
  -webkit-transition: opacity 0.15s linear;
  transition: opacity 0.15s linear;
  /* 6 */
  touch-action: cross-slide-y pinch-zoom double-tap-zoom;
  /* 7 */
  -webkit-transform: translateZ(0);
  transform: translateZ(0);
}
```

Le istruzioni javascript necessarie per il corretto funzionamento del componente sono localizzate invece nel file **uikit.js**

```
            $ele.css(tmp); // reset element
        }

        return height;
    }

})(UIkit);

(function(UI) {

    "use strict";

    var scrollpos = {x: window.scrollX, y: window.scrollY},
        $win      = UI.$win,
        $doc      = UI.$doc,
        $html     = UI.$html,
        Offcanvas = {

        show: function(element) {

            element = UI.$(element);

            if (!element.length) return;

            var $body     = UI.$('body'),
                bar       = element.find(".uk-offcanvas-bar:first"),
                rtl       = (UI.langdirection == "right"),
                flip      = bar.hasClass("uk-offcanvas-bar-flip") ? -1:1,
                dir       = flip * (rtl ? -1 : 1),

                scrollbarwidth =  window.innerWidth - $body.width();

            scrollpos = {x: window.pageXOffset, y: window.pageYOffset};

            element.addClass("uk-active");

            $body.css({"width": window.innerWidth - scrollbarwidth, "height": window.innerHeight}).addClass("uk-offcanvas-page");
            $body.css((rtl ? "margin-right" : "margin-left"), (rtl ? -1 : 1) * (bar.outerWidth() * dir)).width(); // .width() - force redraw

            $html.css('margin-top', scrollpos.y * -1);

            bar.addClass("uk-offcanvas-bar-show");

            this._initElement(element);

            bar.trigger('show.uk.offcanvas', [element, bar]);
```

ATTENZIONE! Il componente Modal è un componente javascript per cui, affinchè possa funzionare in maniera corretta è necessario verificare di aver inserito nella sezione < head > della pagina non solo il collegamento alla libreria uikit.css ma anche quello alla libreria javascript uikit.js

In realtà, nel caso in cui questo dovesse essere l'unico componente javascript del framework che si intende implementare all'interno del sito, si potrebbe pensare di utilizzare, non l'intera libreria uikit.js, ma solamente la libreria relativa allo specifico componente ossia il file **modali.js** (o meglio ancora la sua versione minificata).

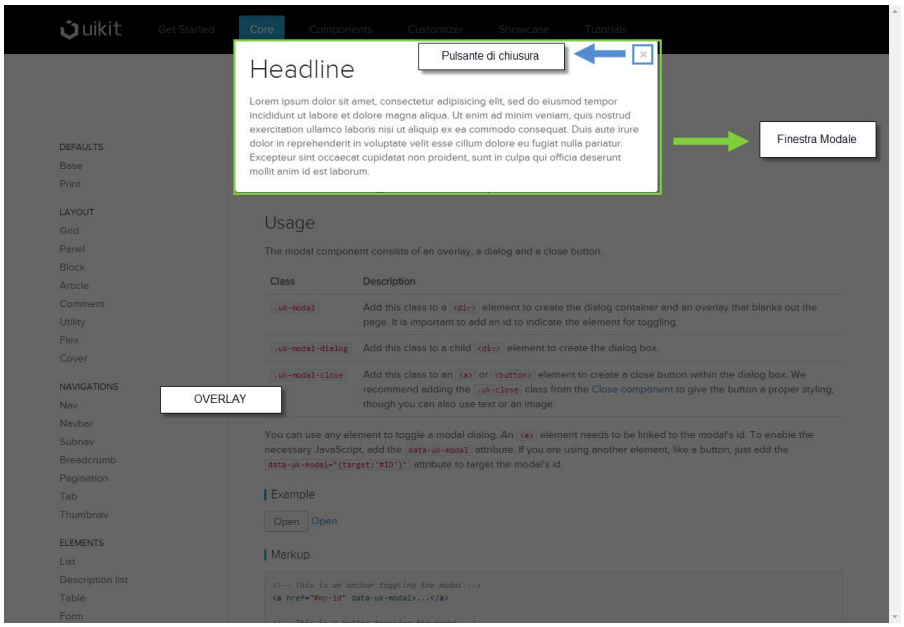
Nei successivi capitoli esamineremo solo alcune delle possibili opzioni di configurazione di questo componente e vedremo anche come poterle implementare in Passweb.

In ogni caso per una trattazione completa di tutte le diverse possibili opzioni di configurazione e, soprattutto, per poter visualizzare degli esempi di utilizzo si consiglia di fare riferimento alla relativa pagina della documentazione ufficiale http://getuikit.com/docs/modal.html

CONFIGURAZIONE

Il Componente Modal è costituito, essenzialmente, da tre diversi elementi, nello specifico:

- Un' overlay mediante il quale "nascondere" i contenuti della pagina
- Una finestra modale all'interno della quale dovranno essere inseriti i contenuti da mostrare
- Un pulsante di chiusura



Questi tre componenti dovranno rispettare un markup ben preciso e a ciascuno di essi dovrà essere assegnata una specifica classe.

In particolare sarà necessario considerare:

- Un primo contenitore esterno (tag div) da utilizzare come contenitore generale della finestra di dialogo e dell'overlay necessario per "nascondere" i contenuti della pagina.

  A questa prima div dovrà essere assegnata la classe **.uk-modal**

  ATTENZIONE! E' di fondamentale importanza assegnare a questo elemento anche uno specifico id, che verrà poi utilizzato per attivare la visualizzazione della finestra modale

- Un secondo contenitore (tag div) interno al precedente necessario per creare la finestra di dialogo.

  A questa seconda div dovrà essere assegnata la classe **.uk-modal-dialog**

  ATTENZIONE! I contenuti della finestra modale dovranno essere inseriti all'interno di questo contenitore

- Un elemento, da inserire all'interno del contenitore di classe .uk-modal-dialog, necessario per creare il pulsante di chiusura della finestra modale.

  In questo senso può essere utilizzato un semplice link (tag a) oppure un apposito pulsante (tag button).

  In entrambi i casi sarà necessario assegnare a questo elemento la classe **.uk-modal-close.**

  Si consiglia inoltre di assegnare a questo elemento anche la classe **.uk-close** in maniera tale da poter stilizzare il pulsante di chiusura come previsto dalle specifiche del Componente Close (per maggiori informazioni relativamente a questo componente si consiglia di consultare la documentazione ufficiale del framework a questo indirizzo http://getuikit.com/docs/close.html)

Il markup della nostra finestra modale dovrà quindi essere del tipo di quello qui di seguito indicato

```
<div id="my-id" class="uk-modal">
    <div class="uk-modal-dialog">
        <a class="uk-modal-close uk-close"></a>
        Contenuti della finestra modale
    </div>
</div>
```

In tutto ciò manca ancora un elemento fondamentale, vale a dire quello che ci consente di attivare la visualizzazione della finestra modale.

Anche in questo caso può essere utilizzato un semplice link (tag a) oppure un apposito pulsante (tag button).

Nel momento in cui si dovesse decidere di utilizzare un link sarà poi necessario:

- Utilizzare come **destinazione del collegamento (valore della proprietà href) l'id del componente di classe .uk-modal**
- Assegnare al tag a l'attributo addizionale **data-uk-modal**

Nel momento in cui si dovesse invece decidere di utilizzare un pulsante sarà invece necessario:

- Assegnare al tag button l'attributo addizionale **data-uk-modal="{target:**'#id'**}"** dove ovviamente al posto di #id andrà inserito l'identificativo del componente di classe **.uk-modal**

In definitiva dunque il markup completo al quale fare riferimento per poter implementare il componente Modal potrebbe essere il seguente

```
<!-- Pulsante per attivare la visualizzazione della finestra modale -->
<a href="#my-id" data-uk-modal>
  Apri finestra
</a>


<!-- Finestra modale -->
<div id="my-id" class="uk-modal">
    <div class="uk-modal-dialog">
        <a class="uk-modal-close uk-close"></a>
        Contenuti della finestra modale
    </div>
</div>
```

ATTENZIONE! Nella configurazione considerata la finestra modale si chiuderà automaticamente anche cliccando in un qualsiasi punto dell'overlay esterno alla finestra stessa.

FINESTRA MODALE SENZA CHIUSURA AUTOMATICA

Nel capitolo precedente abbiamo visto come nella sua configurazione base la finestra modale di uikit possa essere chiusa non solo cliccando sul relativo pulsante di chiusura ma anche cliccando in una qualsiasi altra posizione esterna alla finestra stessa.

Se l'esigenza dovesse invece essere quella di fare in modo che **la finestra modale possa chiudersi solo ed esclusivamente cliccando sul relativo pulsante di chiusura** sarà necessario apportare una semplice variazione al markup dell'elemento utilizzato per attivare la visualizzazione della finestra stessa. In particolare:

- Nel caso in cui si utilizzi come elemento di apertura della finestra modale un link, oltre ad utilizzare come destinazione del collegamento l'id del componente di classe .uk-modal, sarà necessario assegnare al tag a anche l'attributo addizionale **data-uk-modal="{bgclose:false}"**
- Nel caso in cui si utilizzi invece come elemento di apertura un pulsante, sarà necessario assegnare la tag button l'attributo addizionale **data-uk-modal="{target:'#id',bgclose:false}",** dove ovviamente al posto di #id andrà inserito l'identificativo del componente di classe **.uk-modal**

In entrambi i casi il valore **bgclose:false** garantirà che la finestra modale possa chiudersi solo cliccando sul relativo pulsante di chiusura (che in queste condizioni dovrà quindi essere necessariamente inserito all'interno della finestra modale)

Il markup di riferimento per ottenere questo risultato potrebbe quindi essere il seguente:

```
<!-- Pulsante per attivare la visualizzazione della finestra modale -->
<a href="#my-id" data-uk-modal="{bgclose:false}">
  Apri finestra
</a>


<!-- Finestra modale -->
<div id="my-id" class="uk-modal">
    <div class="uk-modal-dialog">
        <a class="uk-modal-close uk-close"></a>
        Contenuti della finestra modale
    </div>
</div>
```

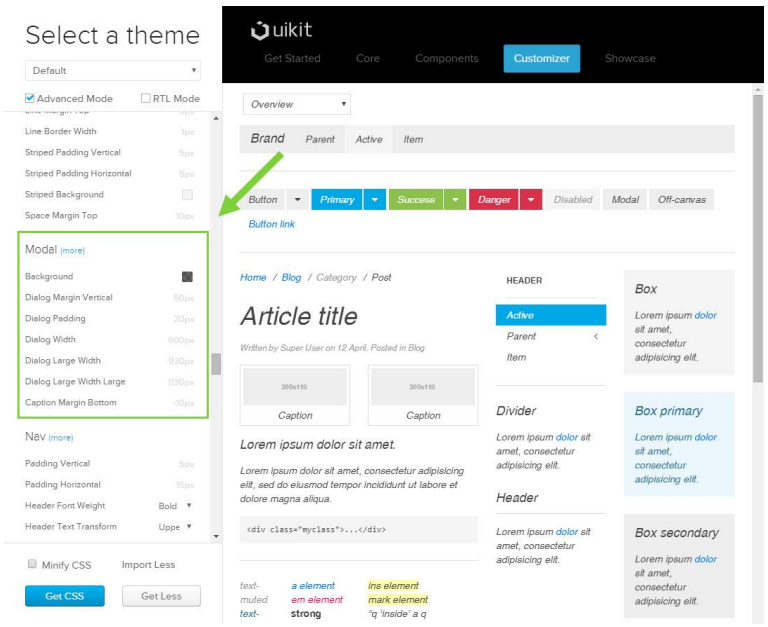
FINESTRA MODALE LARGE

Nella sua configurazione di base e con le impostazioni di default di uikit ogni finestra modale assume una larghezza di 600px e una larghezza massima del 100%.

Ovviamente è possibile variare questa impostazione sia agendo direttamente all'interno del file uikit.css

```
/* Sub-object: `uk-modal-dialog`
 ========================================================================== */
/*
 * 1. Create position context for caption, spinner and close button
 * 2. Set box sizing
 * 3. Set style
 * 4. Slide-in transition
 */
.uk-modal-dialog {
    /* 1 */
    position: relative;
    /* 2 */
    box-sizing: border-box;
    margin: 50px auto;
    padding: 20px;
    width: 600px;
    max-width: 100%;
    max-width: calc(100% - 20px);
    /* 3 */
    background: #ffffff;
    /* 4 */
    opacity: 0;
    -webkit-transform: translateY(-100px);
    transform: translateY(-100px);
    -webkit-transition: opacity 0.3s linear, -webkit-transform 0.3s ease-out;
    transition: opacity 0.3s linear, transform 0.3s ease-out;
}
/* Phone landscape and smaller */
@media (max-width: 767px) {
```

sia mediante l'apposito Customizer



Volendo è inoltre possibile variare in maniera "automatica" questa larghezza impostandola sempre su valori predefiniti ma comunque superiori ai precedenti 600px.

Per fare questo è sufficiente assegnare al contenitore utilizzato per identificare la finestra modale oltre alla classe .uk-modal-dialog anche la classe **.uk-modal-dialog-large**

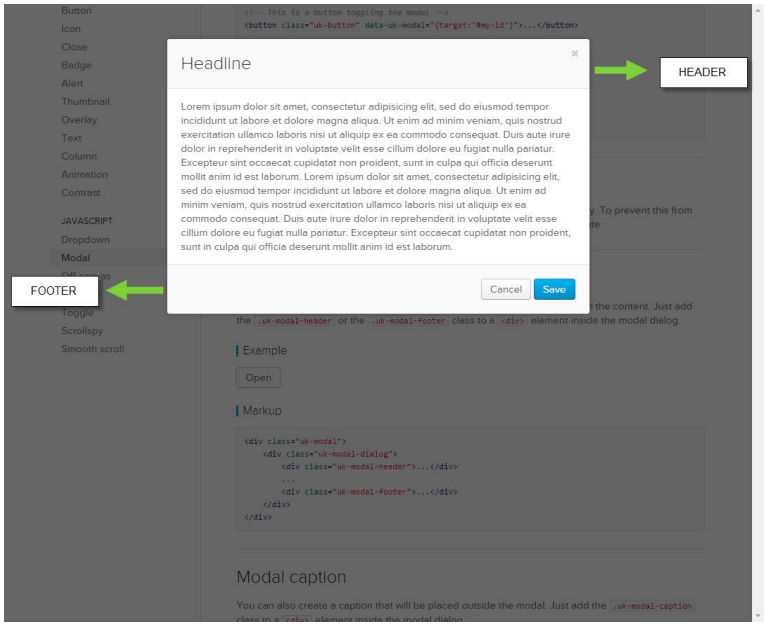
.uk-modal-dialog-large: assegnata al contenitore della finestra modale fa si che, in condizioni di default, la sua larghezza sia di 930px per viewport compresi tra 768 e 1220 pixel, e di 1130px per viewport superiori a 1220px.

Il markup al quale fare riferimento sarà quindi il seguente

```
<!-- Pulsante per attivare la visualizzazione della finestra modale -->
<a href="#my-id" data-uk-modal">
  Apri finestra
</a>


<!-- Finestra modale -->
<div id="my-id" class="uk-modal uk-modal-dialog-large">
    <div class="uk-modal-dialog">
        <a class="uk-modal-close uk-close"></a>
        Contenuti della finestra modale
    </div>
</div>
```

FINESTRA MODALE CON HEADER E FOOTER

Volendo è possibile creare facilmente anche una finestra modale dotata di header e footer analoga a quella evidenziata in figura:



Per ottenere questo risultato è sufficiente fare riferimento alla configurazione di base del componente inserendo all'interno del contenitore di classe .uk-modal-dialog:

- prima dell'effettivo contenuto della finestra modale un contenitore (tag div) con classe **.uk-modal-header**
- dopo l'effettivo contenuto della finestra modale un contenitore (tag div) con classe **.uk-modal-footer**

Il markup al quale fare riferimento sarà quindi il seguente

```
<!-- Pulsante per attivare la visualizzazione della finestra modale -->
<a href="#my-id" data-uk-modal">
  Apri finestra
</a>


<!-- Finestra modale -->
<div id="my-id" class="uk-modal">
    <div class="uk-modal-dialog">
      <a class="uk-modal-close uk-close"></a>
      <div class="uk-modal-header">Contenuti della Testata</div>
      Contenuti della finestra modale
      <div class="uk-modal-footer">Contenuti della Piede</div>
    </div>
</div>
```

COMPONENTE MODAL IN PASSWEB

Detto che in Passweb è possibile gestire in maniera nativa delle finestre modali (**e questa è ovviamente la soluzione migliore da adottare**) se proprio si volesse utilizzare il componente Modal di uikit, la sua implementazione sarà comunque estremamente semplice.

Considerando infatti quello che è il markup al quale fare riferimento l'implementazione di questo componente si riduce alla gestione di una serie di Componenti Contenitore annidati ai quali assegnare le classi corrette, e di un semplice link da utilizzare per attivare la visualizzazione della finestra modale.

In tutto questo c'è però una considerazione di fondamentale importanza da fare ossia:

nel momento in cui andremo ad assegnare ad un Componente Contenitore la classe .uk-modal, considerate le proprietà CSS che verranno applicate al Componente stesso, questo "scomparirà" e, all'interno del Live Editing, non potrà più essere raggiunto se non utilizzando la barra laterale dei componenti

Come già evidenziato per il componente Offcanvas, questo è perfettamente sufficiente per stilizzare il componente stesso con lo Style Editor di Passweb (anche se non avremo la possibilità di verificare i risultati ottenuti se non lato front end), ma ci impedisce di poter inserire al suo interno altri componenti.

In conseguenza di ciò nel caso in cui si dovesse decidere di utilizzare all'interno del proprio sito Passweb il componente Modal di uikit, sarà necessario prima di tutto inserire e stilizzare correttamente tutti i Componenti necessari ad ottenere il risultato desiderato e solo alla fine di questo processo andare poi ad assegnare ai vari contenitori le classi corrette per attivare effettivamente il componente Modal.

TOGGLE

Il Componente Toggle altro non è se non un semplice interruttore mediante il quale poter gestire la comparsa di determinati elementi presenti all'interno della pagina web.

Essendo un componente javascript **affinchè possa funzionare in maniera corretta è necessario verificare di aver inserito nella sezione head della pagina non solo il collegamento alla libreria uikit.css ma anche quello alla libreria javascript uikit.js**

A differenza di quanto visto fino a questo momento al componente Toggle non sono associate specifiche regole all'interno del file uikit.css,

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/toggle.html

CONFIGURAZIONE

Il Componente Modal è costituito, essenzialmente, da due diversi elementi, nello specifico:

- **Un interruttore** mediante il quale poter attivare o disattivare la visualizzazione di un determinato elemento, presente all'interno della stessa pagina web
- **L'elemento da visualizzare / nascondere**

Per quel che riguarda l'interruttore è possibile utilizzare indifferentemente un semplice link (tag a) oppure un apposito pulsante (tag button), la cosa importante è quella di associare a questo elemento l' attributo aggiuntivo **data-uk-toggle="{target: #ID}"** dove, ovviamente, al **posto di #id, andrà inserito l'identificativo del componente da visualizzare / nascondere.**

Per quel che riguarda invece l'elemento da visualizzare / nascondere questo può essere un qualsiasi elemento presente nella stessa pagina dell'interruttore, **la cosa importante è che possa essere individuato mediante uno specifico id o, al limite, mediante una specifica classe.**

ATTENZIONE! Per individuare l'identificativo dell'elemento da visualizzare / nascondere è possibile utilizzare gli strumenti per sviluppatore del browser

Il markup completo al quale fare riferimento per poter implementare il componente Toggle potrebbe quindi essere il seguente

```
<!-- Pulsante per attivare visualizzare/nascondere l'elemento con id=my-id -->

<a data-uk-toggle="{target:'#my-id'}"> Clicca qui</a>


<!-- Elemento da visualizzare/nascondere -->

<div id="my-id">
   Contenuti da visualizzare/nascondere
</div>
```

Nel caso in cui l'esigenza dovesse essere quella di visualizzare/nascondere più elementi contemporaneamente, sarà sufficiente assegnare a tutti questi elementi la stessa classe, che dovrà poi essere utilizzata anche come target dell'attributo data-uk-toggle.

In queste circostanze dunque il markup cui fare riferimento dovrà essere il seguente

```
<!-- Pulsante per attivare visualizzare/nascondere gli elemento con classe  uguale a .my-class -->

<a data-uk-toggle="{target:'.my-class'}"> Clicca qui</a>


<!-- Elementi da visualizzare/nascondere -->

<div class="my-class">
   Contenuti da visualizzare/nascondere
</div>

<div class="my-class">
   Contenuti da visualizzare/nascondere
</div>
```

Dal punto di vista tecnico la visualizzazione / scomparsa dell'elemento o degli elementi interessati viene gestita da uikit aggiungendo o togliendo, a seconda dei casi, la classe **.uk-hidden.**

Più esattamente, ad ogni click sull'interruttore verrà aggiunta la classe .uk-hidden agli elementi target privi di tale classe (tali elementi verranno quindi nascosti) e verrà invece eliminata da quegli elementi ai quali era già stata assegnata (tali elementi torneranno quindi visibili)

Sfruttando lo stesso meccanismo è anche possibile assegnare o rimuovere dagli elementi target, al click sull'interruttore, una classe personalizzata.

Per far questo è necessario assegnare all'elemento utilizzato come interruttore il seguente attributo:

data-uk-toggle="{target:'#my-id', cls:'my-class'}"

dove inserire al posto di #id l'identificativo dell'elemento target e al posto di my-class la classe da assegnare / rimuovere al click sull'interruttore.

Questo potrebbe essere utile non tanto per visualizzare / nascondere determinati elementi quanto più esattamente per modificare la loro formattazione assegnandogli quindi stili diversi (es. un diverso colore di sfondo).

TOGGLE IN PASSWEB

Ogni Componente Passweb ha un suo identificativo e una sua specifica classe assegnate in automatico dal programma. Inoltre è possibile assegnare ad essi anche una qualsiasi altra classe personalizzata.

In conseguenza di ciò è possibile implementare all'interno del proprio sito Passweb il componente Toggle di uikit per nascondere / visualizzare un qualsiasi elemento presente all'interno della pagina web.

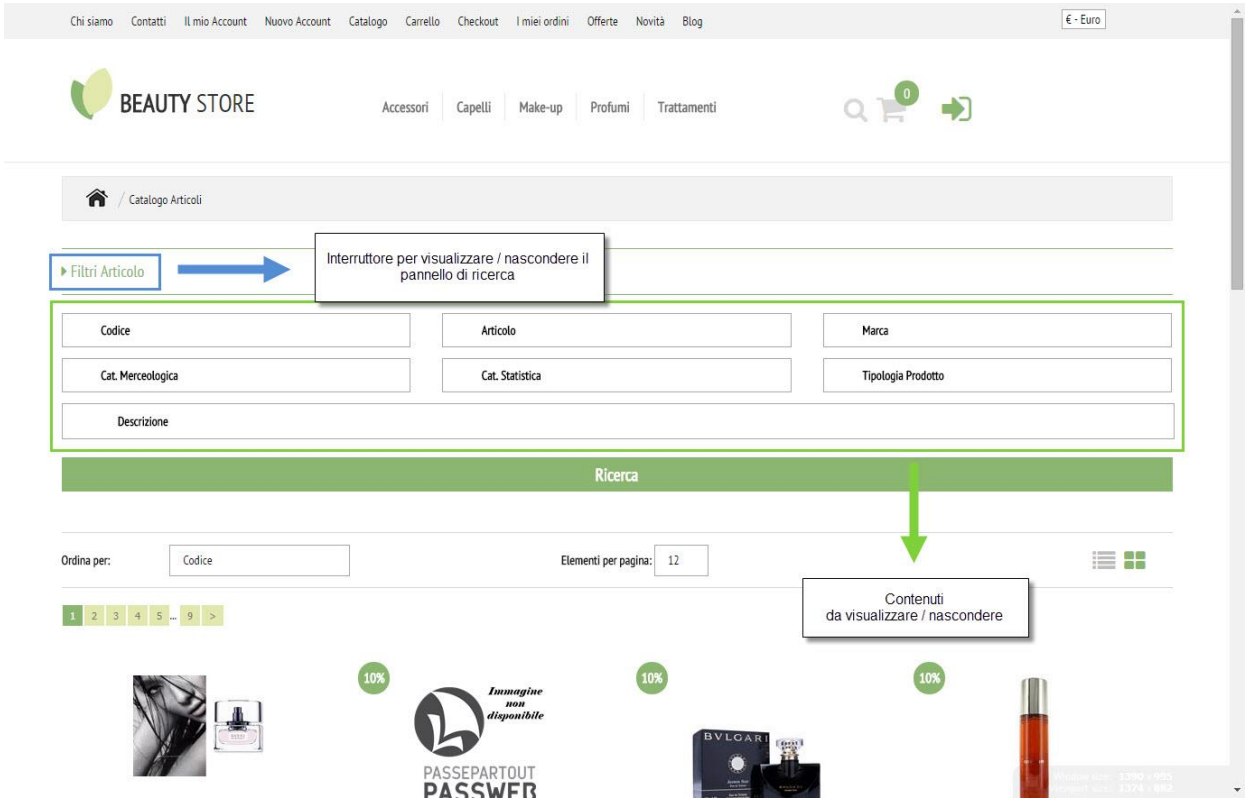
Le uniche cosa da fare in questo senso saranno:

- Individuare l'id o la classe dell'elemento da visualizzare / nascondere
- Implementare, mediante un semplice Componente Paragrafo o HTML l'interruttore per attivare la visualizzazione / scomparsa dell'elemento target secondo le specifiche indicate nel precedente capitolo.

MODELLO ECOMMERCE 29

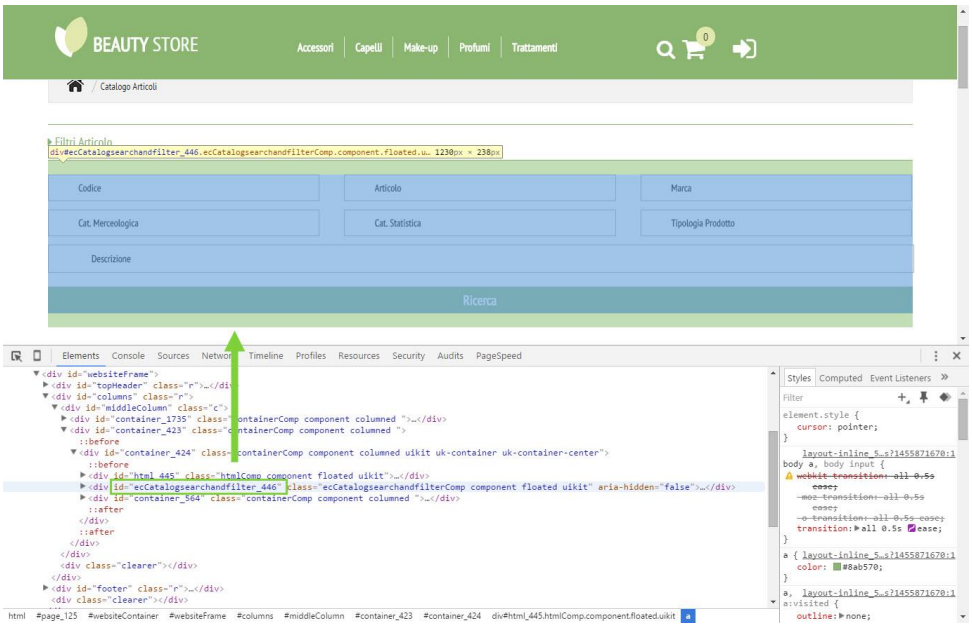
E' possibile trovare un esempio di applicazione del componente Toggle nella pagina "**Catalogo Articoli**" del modello di riferimento.

In questo senso l'esigenza è quella di visualizzare / nascondere il pannello di ricerca utilizzato per applicare determinati filtri agli articoli presenti in catalogo, cliccando su di un apposito interruttore.



Per soddisfare questa esigenza è necessario:

- Individuare l'id del componente "Filtro/Ricerca Catalogo" utilizzato per gestire il pannello di ricerca

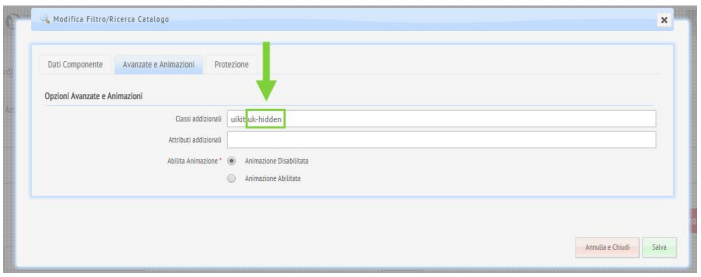


- Inserire un Componente HTML (**Interruttore-Filtri**) da utilizzare per creare il seguente link

```
<a data-uk-toggle="{target:'#ecCatalogsearchandfilter_446'}">
  Filtri Articolo
</a>
```

L'attributo **data-uk-toggle="{target:'#ecCatalogsearchandfilter_446'}"** farà si che cliccando su questo link venga visualizzato/nascosto l'elemento con id=#ecCatalogsearchandfilter_446' che è esattamente il nostro pannello di ricerca.

Nel caso in cui il pannello di ricerca, inizialmente, dovesse essere chiuso, sarà necessario assegnare al Componente Filtro/Ricerca Catalogo anche la classe **.uk-hidden**



ATTENZIONE! Assegnando la classe **.uk-hidden** al componente Filtro/Ricerca Catalogo questo non sarà più visibile all'interno del Live Editing. Prima di assegnare questa classe è quindi necessario apportare al componente tutte le modifiche necessarie.

C'è un ultima osservazione molto importante da fare necessaria per poter garantire, in questo caso particolare, il corretto funzionamento del componente Toggle di uikit.

I componenti javascript di uikit vengono inizializzati soltanto una volta, nello specifico, dopo che la pagina è stata caricata.

Nel nostro caso particolare, quando viene applicato e/o rimosso un filtro di ricerca, la pagina viene parzialmente ricaricata per mostrare gli articoli presenti in catalogo che soddisfano il filtro di ricerca.

Questo reload parziale della pagina non consente di inizializzare nuovamente i componenti javascript di uikit per cui, senza un'apposita azione correttiva, quello che si otterrà sarà esattamente di non poter più aprire/chiudere il pannello di ricerca dopo aver applicato e/o rimosso un filtro.

Fortunatamente nelle ultime versioni di uikit è presente una funzione che può essere utilizzata, quando necessario, per effettuare una nuova inizializzazione dei componenti javascript evitando quindi problemi analoghi a quello appena descritto.

Per risolvere il nostro problema e poter continuare quindi ad aprire/chiudere il pannello di ricerca anche dopo aver applicato e/o eliminato un certo filtro sarà necessario inserire nella sezione javascript del Layout di pagina le seguenti istruzioni

```
function inizializzaUikit()
{
  $.UIkit.init()
  setTimeout(inizializzaUikit, 50);
}

$(document).ready(function() {
  inizializzaUikit();
});
```

dove la funzione **inizializzaUikit(),** richiamata al document.ready non fa altro che eseguire ogni 50ms l'inizializzazione dei componenti javascript di uikit (mediante l'istruzione $.UIkit.init();).

SMOOTH SCROLL

Il componente Smooth Scroll consente di gestire lo scroll automatico del browser aggiungendo un effetto di rallentamento per rendere più gradevole la transizione.

Essendo un componente javascript **affinchè possa funzionare in maniera corretta è necessario verificare di aver inserito nella sezione head della pagina non solo il collegamento alla libreria uikit.css ma anche quello alla libreria javascript uikit.js**

Come per il componente Toggle anche per lo Smooth Scroll non sono presenti specifiche regole all'interno del file uikit.css.

ATTENZIONE! La documentazione ufficiale può essere consultata al seguente indirizzo http://getuikit.com/docs/smooth-scroll.html

CONFIGURAZIONE

L'implementazione dello Smooth Scroll di uikit è estremamente semplice.

Si tratta infatti, da una parte, di individuare l'id dell'elemento in corrispondenza del quale far scorrere, automaticamente, la finestra del browser e, dall'altra parte, di creare un' apposito link utilizzando come destinazione del collegamento (valore della proprietà href) l'id in questione, assegnando inoltre al link stesso l'attributo **data-uk-smooth-scroll**

Supponendo dunque di aver utilizzato all'interno della pagina web un elemento con id=my-id, per creare un pulsante mediante il quale far scorrere automaticamente la finestra del browser in corrispondenza di tale elemento e attivare anche l'effetto di rallentamento, sarà sufficiente inserire all'interno della stessa pagina il seguente link

```
<a href="#my-id" class=" " data-uk-smooth-scroll>
  Vai all'elemento my-id
</a>
```

Se necessario è anche possibile impostare un offset mediante il quale poter specificare quanti pixel, prima o dopo il target indicato, dovrà terminare lo scroll automatico della finestra del browser.

Nello specifico assegnando, ad esempio, al link utilizzato per attivare lo scroll automatico l'attributo:

- **data-uk-smooth-scroll="{offset: 90}" :** lo scroll automatico del browser terminerà 90 pixel prima dell'elemento target
- **data-uk-smooth-scroll="{offset: -90}" :** lo scroll automatico del browser terminerà 90 pixel dopo l'elemento target

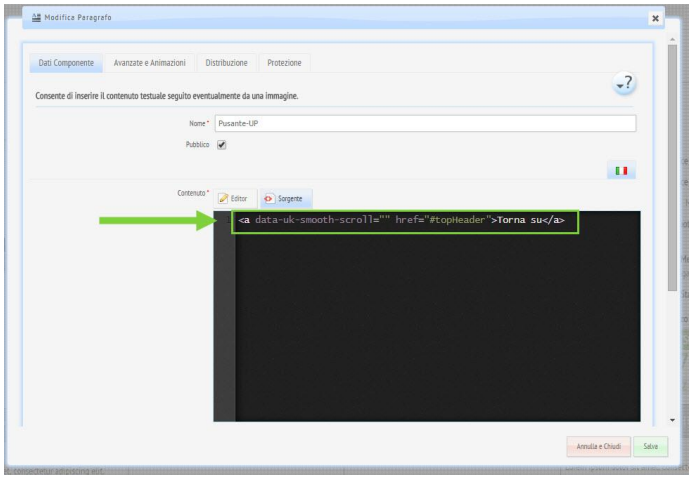
MODELLO ECOMMERCE 29

E' possibile trovare un esempio di applicazione del componente Toggle in una qualsiasi pagina del modello di riferimento.

In questo senso infatti l'esigenza da soddisfare è quella di avere un pulsante mediante il quale poter ripotare automaticamente la finestra del browser nella sua posizione iniziale.

Considerando quindi che il primo contenitore strutturale della pagina è la "Testata Alta" e, soprattutto, che tale elemento ha id=topHeader per soddisfare l'esigenza in questione è sufficiente inserire nella pagina un **componente Paragrafo (Pulsante-UP)** da utilizzare per creare il seguente link:

```
<a data-uk-smooth-scroll href="#topHeader">Torna su</a>
```



ATTENZIONE! Il componente è stato stilizzato normalmente con lo Style Editor di Passweb. E' stato inoltre applicato un piccolo javascript (indipendente da uikit) mediante il quale far visualizzare il pulsante solo quando l'utente effettua un scroll verso il basso superiore a 200px

SCROLLSPY

Il componente Scrollspy consente di attivare automaticamente determinati eventi allo scorrimento della pagina web.

Tipicamente questo componente viene utilizzato per visualizzare elementi, non appena questi entrano nell'area di visualizzazione della pagina web, utilizzando vari effetti di animazione.

Essendo un componente javascript **affinchè possa funzionare in maniera corretta è necessario verificare di aver inserito nella sezione head della pagina non solo il collegamento alla libreria uikit.css ma anche quello alla libreria javascript uikit.js**

Come per il componente Toggle o per lo Smooth Scroll anche in questo caso non sono presenti specifiche regole all'interno del file uikit.css.

ATTENZIONE! La documentazione ufficiale, assieme ad alcuni esempi di utilizzo, può essere consultata al seguente indirizzo http://getuikit.com/docs/scrollspy.html

CONFIGURAZIONE

L'implementazione di questo componente è, di per sé, estremamente semplice, si tratta infatti di assegnare all'elemento che deve essere visualizzato con una certa animazione, non appena comparirà all'interno della pagina, l'attributo aggiuntivo **data-uk-scrollspy** specificando anche l'animazione da utilizzare.

Il markup al quale fare riferimento sarà quindi del tipo di quello qui di seguito indicato

```
<div data-uk-scrollspy="{cls:'uk-animation-fade'}">...</div>
```

dove al posto di uk-animation-fade può essere indicata una qualsiasi altra animazione definita nel componente Animate di uikit stesso (per maggiori informazioni sulle possibili animazione e i relative valori da poter applicare si consiglia di fare riferimento alla documentazione presente al seguente indirizzo http://getuikit.com/docs/animation.html)

Nell'esempio sopra considerato la visualizzazione dell'elemento cui è stato applicato l'attributo aggiuntivo **data-uk-scrollspy,** sarà animata con un effetto fade solo la prima volta in cui l'elemento stesso entra nell'area di visualizzazione della pagina web.

Se l'esigenza dovesse invece essere quella di attivare questa animazione ogni volta in cui l'elemento entra nell'area di visualizzazione della pagina sarà sufficiente utilizzare l'opzione **repeat:true** facendo quindi riferimento ad un markup di questo tipo:

```
<div data-uk-scrollspy="{cls:'uk-animation-fade', repeat: true}">...</div>
```

Volendo è anche possibile ritardare l'animazione che porta alla visualizzazione dell'elemento interessato indicando uno specifico delay

```
<div data-uk-scrollspy="{cls:'uk-animation-fade', repeat: true, delay:900}">...</div>
```

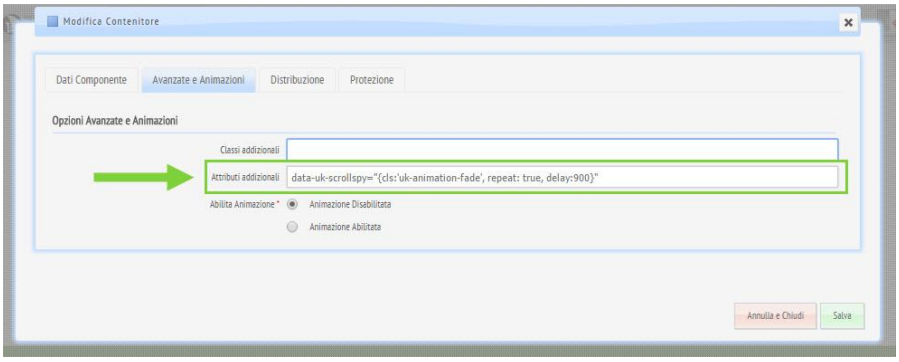
Nell'esempio sopra indicato, l'animazione che porta alla visualizzazione dell'elemento cui è stato assegnato l'attributo data-uk-scrollspy non partirà immediatamente nel momento stesso in cui l'elemento entra nell'area di visualizzazione della pagina web ma soltanto dopo 900 millisecondi.

Infine nel caso in cui si debba attivare questo tipo di animazione non su di un singolo elemento ma su di un gruppo di elementi, è necessario racchiudere questi stessi elementi all'interno di un unico contenitore, marcarli tutti con una stessa classe (es. my-class) e utilizzare, sul contenitore, l'attributo data-uk-scrollspy con l'opzione target:'.my-class'

```
<div data-uk-scrollspy="{cls:'uk-animation-fade', delay:300, target:'.my-class'}">

    <!-- Questo elemento verrà visualizzato senza ritardo -->
    <div class="my-class">...</div>

    <!-- Questo elemento verrà visualizzato con un ritardo di 300ms -->
    <div class="my-class">...</div>

    <!-- Questo elemento verrà visualizzato con un ritardo di 600ms -->
    <div class="my-class">...</div>

</div>
```

SCROLLSPY IN PASSWEB

Sulla base di quanto detto nel precedente capitolo, l'implementazione del componente Scrollspy in Passweb dovrebbe essere estremamente semplice, nel senso che sarebbe sufficiente applicare l'attributo data-uk-scrollspy al componente interessato utilizzando l'apposito campo presente nella maschera di configurazione del componente stesso



In realtà procedendo in questo modo diventerebbe poi difficile gestire il sito all'interno del Live Editing di Passweb.

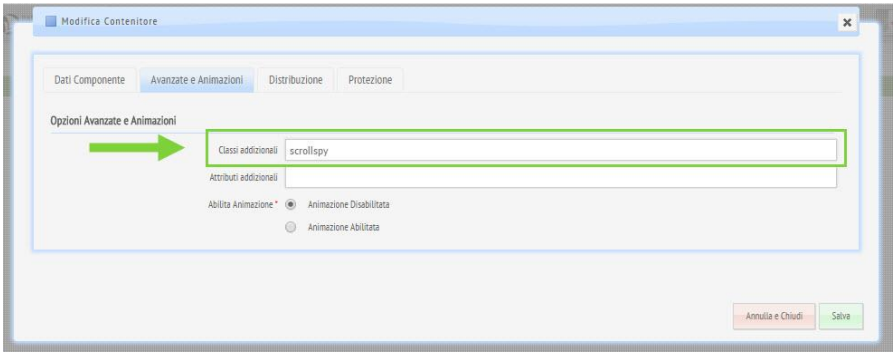
Il Componente cui viene applicato lo Scrollspy infatti, inizialmente non è visibile e le classi che attivano l'animazione che porta poi alla sua visualizzazione vengono applicate via javascript allo scroll della pagina.

Il problema in questo senso è che, come noto, il codice javascript di terze parti non viene eseguito lato Wizard per cui mentre sul front end del sito tutto funzionerebbe correttamente, all'interno del Live Editing **il Componente cui è stato applicato lo Scrollspy non sarebbe più visibile e non avremmo quindi modo di gestirlo "normalmente"**.

Per poter gestire al meglio il proprio sito Passweb utilizzando gli strumenti tradizionali offerti dal Live Editing e non rinunciare comunque all'utilizzo dello Scrollspy la soluzione potrebbe allora essere quella di assegnare agli elementi interessati l'attributo data-uk-scrollspy, con le diverse opzioni di configurazione, via javascript e quindi solo sul front end del sito.

Per poter far questo dovremo quindi:

- Assegnare ai Componenti cui dovrà essere applicato lo Scrollspy una classe personalizzata (es. .scrollspy) sfruttando l'apposito campo presente nella maschera di configurazione del Componente stesso.



- Inserire nel layout di pagina o meglio ancora in quello di Variante il seguente codice javascript:

```
$(document).ready(function() {
  $('.scrollspy').attr("data-uk-scrollspy","{cls:'uk-animation-fade'}");
});
```

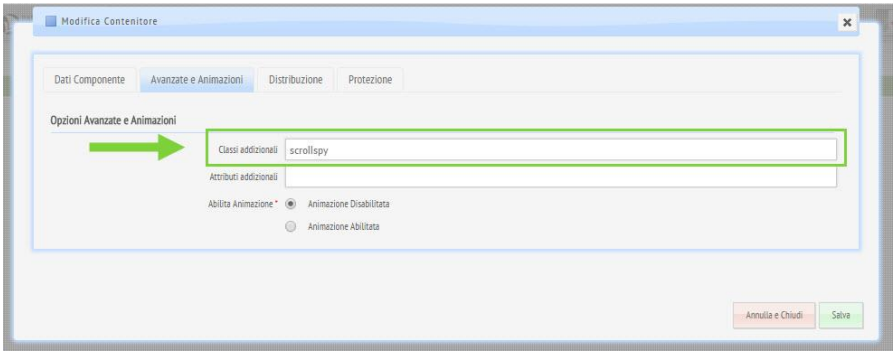
Queste semplici istruzioni non fanno altro che individuare, una volta caricata la pagina, tutti gli elementi di classe .scrollspy, e assegnargli l'attributo aggiuntivo **data-uk-scrollspy="{cls:'uk-animation-fade'}"** attivando, di fatto, lo scrollspy con le opzioni indicate.

MODELLO ECOMMERCE 29

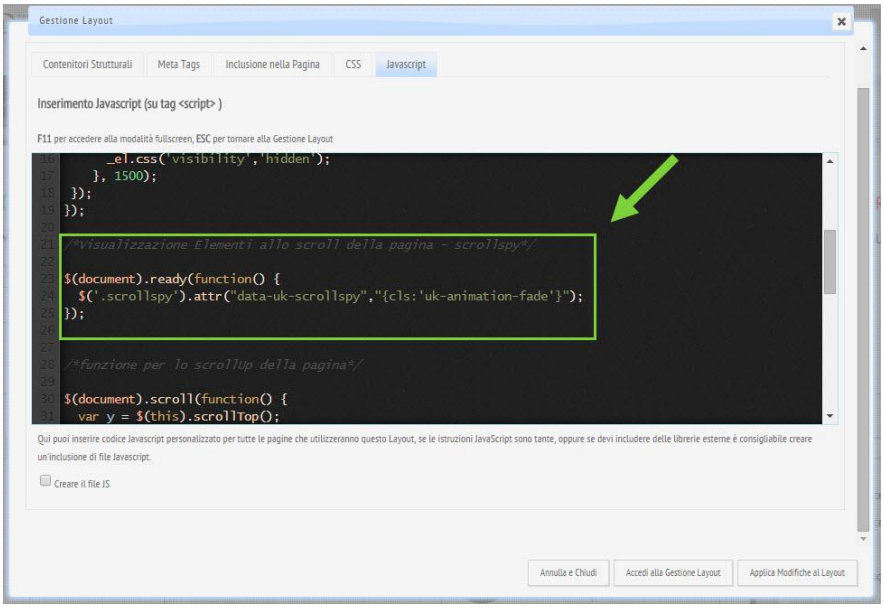
E' possibile trovare un esempio di applicazione dello Scrollspy nella **Home Page** del modello di riferimento.

Come indicato nel precedente capitolo l'implementazione delle Scrollspy è stata effettuata via javascript in maniera tale da mantenere inalterate le modalità di utilizzo del Live Editing.

Da una parte è stata quindi assegnata a tutti i Componenti Contenitore denominati "**Riga-Scrollspy**" la classe aggiuntiva scrollspy



Dall'altra parte sono state inserite nel Layout di Variante le istruzioni javascript indicate nel precedente capitolo



In questo modo mano a mano che si scorre la pagina verso il basso, non appena i Componenti “Riga-Scrollspy” entreranno nell’area di visualizzazione del documento, verrà attivata un’animazione di fade che porterà alla loro visualizzazione.

SLIDER

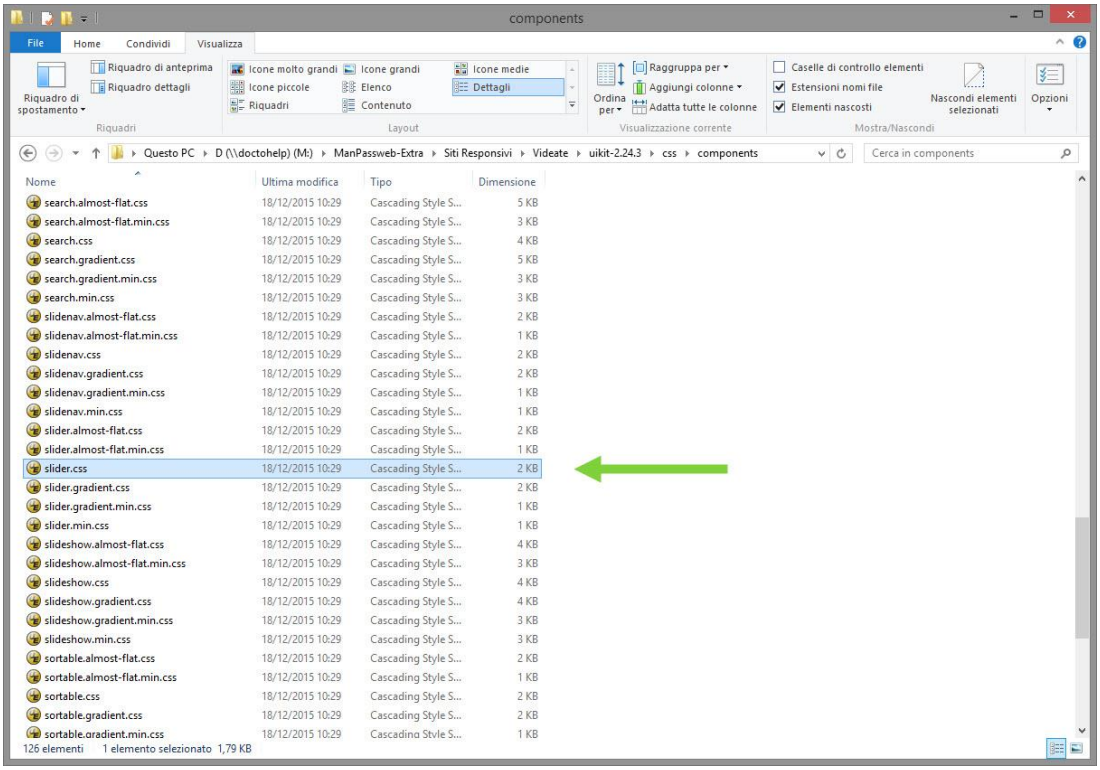
Il componente Slider di uikit consente di gestire un insieme di elementi facendoli scorrere in orizzontale uno dopo l’altro.

Lo scorrimento può essere attivato cliccando su appositi pulsanti di avanzamento, trascinando con il mouse, verso destra o verso sinistra, gli elementi presenti all’interno dello slider oppure utilizzando, laddove possibile (dispositivi touch), la classica gesture di swipe.

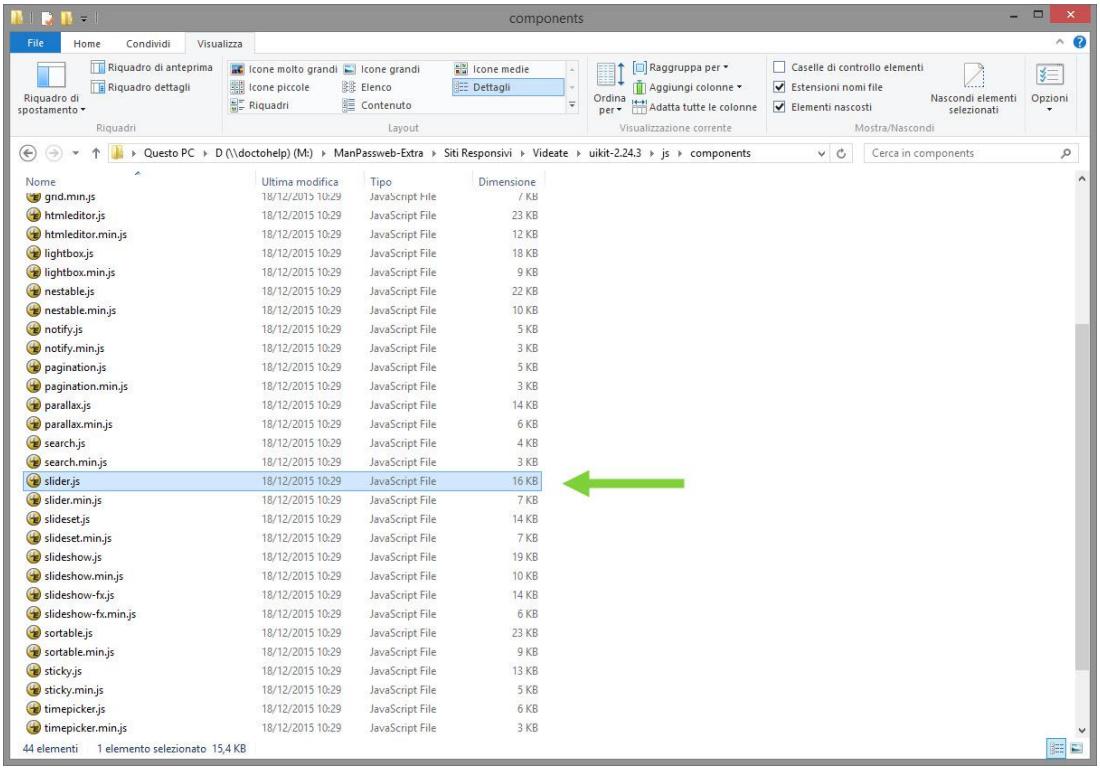
ATTENZIONE! La documentazione ufficiale, assieme ad alcuni esempi di utilizzo, può essere consultata al seguente indirizzo http://getuikit.com/docs/slider.html

Lo Slider è il primo componente “avanzato” che analizziamo all’interno di questa guida. Nel sito ufficiale del progetto lo troviamo all’interno della sezione Components e, **come tutti i componenti avanzati presenti all’interno di questa sezione, utilizza specifiche librerie CSS e javascript che dovranno quindi essere caricate all’interno del sito in aggiunta ai file “standard” uikit.css e uikit.js (parte core del framework)**

In particolare le regole CSS relative a questo componente sono localizzate nel file **slider.css** (o nella sua versione minificata) che possiamo trovare nella cartella “**css – components**” ottenuta una volta decompattato l’archivio .zip scaricato dal sito del progetto (per maggiori informazioni in merito al download del framework si veda anche il relativo capitolo di questa guida)



Le istruzioni javascript necessarie per il corretto funzionamento del componente sono localizzate invece nel file **slider.js** (o nella sua versione minificata) che possiamo trovare all'interno della cartella "**js – components**"



CONFIGURAZIONE

Come evidenziato nel precedente capitolo di questa guida per poter implementare correttamente il componente Slider di uikit è necessario caricare all'interno del sito, oltre alla parte core del framework (uikit.css e uikit.js) anche le librerie css e js utilizzate in maniera specifica da questo componente.

La prima cosa da fare sarà quindi verificare di aver inserito nella sezione < head > della pagina in cui sui vuol implementare questo componente i collegamenti alle librerie slider.css e slider.js (o meglio ancora alla loro versione minificata)

```
<link rel="stylesheet" href="uikit.css" />
<link rel="stylesheet" href="slider.css" />


<script src="uikit.js"></script>
<script src="slider.js"></script>
```

ATTENZIONE! La libreria slider.js deve necessariamente essere caricata dopo la libreria uikit.js

Fatto questo sarà poi necessario definire:

- Un contenitore esterno (tag div) dello slider al quale assegnare l'attributo **data-uk-slider**
- Lo slider (tag div) al quale assegnare la classe **.uk-slider-container**
- La lista dei contenuti che dovranno essere visualizzati all'interno dello slider.

  Tale lista dovrà essere gestita mediante un **tag < ul > (e relativi < li >)** al quale assegnare la classe **.uk-slider**
- Il numero di elementi che dovranno essere visualizzati contemporaneamente all'interno dello slider.

In particolare, relativamente all'ultimo punto, è possibile agire in due modi diversi:

- Impostando la larghezza di ogni singolo elemento mediante una delle classi **.uk-width-*** assegnata direttamente allo specifico elemento (tag < li >)
- Impostando la stessa larghezza per tutti gli elementi dello slider mediante una delle classi **.uk-grid-width-*** impostata direttamente sull'elemento di classe .uk-slider

Nel primo caso il markup al quale fare riferimento potrebbe quindi essere il seguente

```
<div data-uk-slider>
  <div class="uk-slider-container">
    <ul class="uk-slider">
      <li class="uk-width-1-3">...</li>
      <li class="uk-width-1-5">...</li>
      <li class="uk-width-2-5">...</li>
      ...
    </ul>
  </div>
</div>
```

Nel caso in cui si volesse invece assegnare la stessa larghezza a tutti gli elementi dello slider si dovrà fare riferimento ad un markup di questo tipo

```
<div data-uk-slider>
  <div class="uk-slider-container">
    <ul class="uk-slider uk-grid-width-1-4">
      <li>...</li>
      <li>...</li>
      <li>...</li>
      ...
    </ul>
  </div>
</div>
```

dove la classe .uk-grid-width-1-4 indica che l'area visibile dello slider dovrà essere suddivisa in 4 slot uguali e ogni contenuto verrà inserito all'interno di uno di questi slot (4 elementi visibili contemporaneamente)

ATTENZIONE! Per maggiori informazioni sull'utilizzo delle classi .uk-width-* e .uk-grid-width-* si veda anche il capitolo di questa guida relativo al componente Griglia

Fino a questo momento abbiamo esaminato quella che è la configurazione di base del componente Slider.

Quello che faremo ora sarà invece analizzare alcune delle sue possibili opzioni di configurazione.

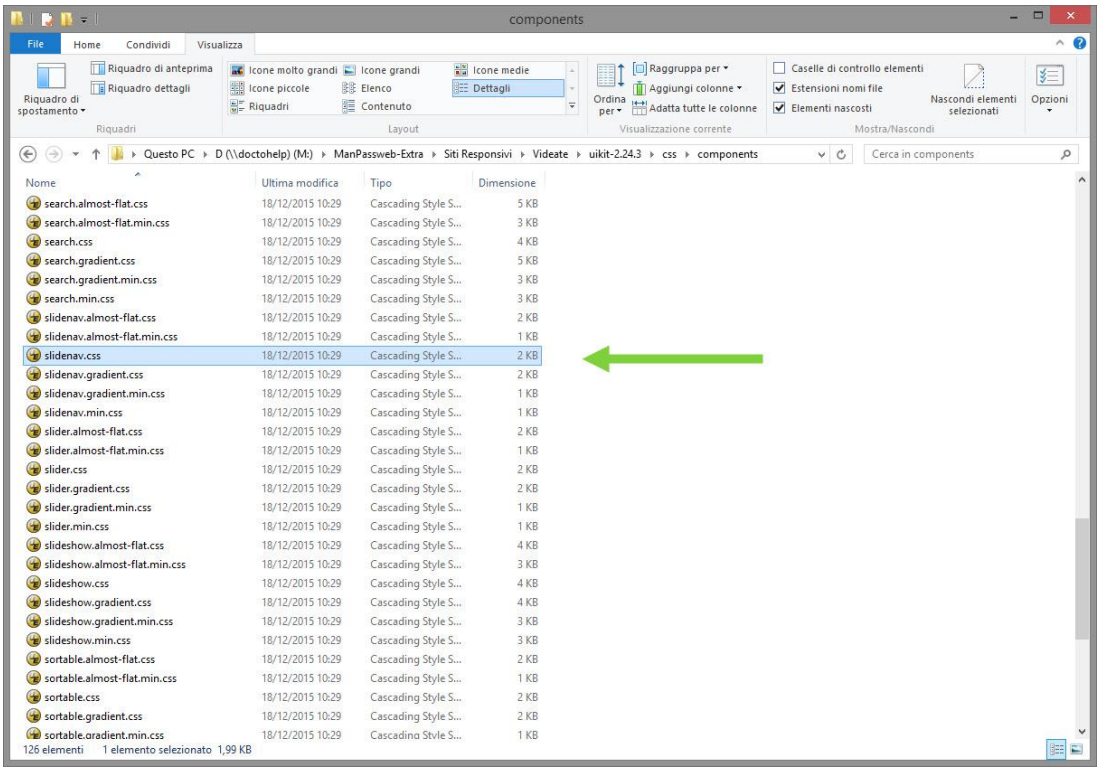
NAVIGAZIONE

Tra le diverse modalità di scorrimento degli elementi presenti all'interno di uno Slider una possibilità è rappresentata dal click su appositi pulsanti di avanzamento che potranno essere visualizzati solo nel momento in cui si dovesse passare con il mouse sul componente stesso.

Per poter implementare questa configurazione è necessario ricorrere al componente **Slidenav** che, al pari del componente Slider, è uno dei componenti avanzati di uikit.

Per prima cosa sarà quindi necessario accertarsi di aver caricato correttamente, oltre alle librerie core e a quelle utilizzate dal componente Slider, anche quelle necessarie per il corretto funzionamento del componente Slidenav.

Nello specifico per il corretto funzionamento di questo componente sarà necessario utilizzare anche la libreria **slidenav.css**



Tra le librerie incluse nella sezione < head > della pagina in cui sui vuol implementare lo slider dovremo quindi inserire anche quello alla libreria in oggetto (o meglio ancora alla sua versione minificata)

```
<link rel="stylesheet" href="uikit.css" />
<link rel="stylesheet" href="slider.css" />
<link rel="stylesheet" href="slidenav.css" />


<script src="uikit.js"></script>
<script src="slider.js"></script>
```

Fatto questo per poter poi attivare i pulsanti di navigazione dello slider sarà necessario:

- Assegnare al contenitore esterno dello slider (quello cui si è precedentemente associato l'attributo data-uk-slider) la classe **.uk-slidenav-position**
- Inserire all'interno del contenitore esterno dello slider due link **(tag < a >)** che serviranno per creare i pulsanti di precedente e successivo e ai quali dovrà essere assegnata:
    - la classe **.uk-slidenav**
    - la classe **.uk-slidenav-previous**, per il pulsante "precedente" e quella **.uk-slidenav-next**, per il pulsante "successivo"
    - l'attributo **data-uk-slider-item="previous"**, per il pulsante "precedente" e quello **data-uk-slider-item="next"**, per il pulsante "successivo"

In queste condizioni dunque il markup al quale fare riferimento dovrà essere il seguente

```
<div class="uk-slidenav-position" data-uk-slider>
  <div class="uk-slider-container">
    <ul class="uk-slider uk-grid-width-medium-1-4">
      <li>...</li>
       ...
    </ul>
  </div>

  <a href="" class="uk-slidenav uk-slidenav-previous" data-uk-slider-item="previous"></a>

  <a href="" class="uk-slidenav uk-slidenav-next" data-uk-slider-item="next"></a>

</div>
```

SLIDER RESPONSIVO

Nella configurazione di base del componente Slider, abbiamo visto come sia possibile utilizzare le classi **.uk-width-* e .uk-grid-width-*** del componente griglia, per dimensionare i vari elementi presenti all'interno dello slider.

Entrambe queste tipologie di classi non consentono però di variare il numero di elementi, visualizzati contemporaneamente all'interno dello slider, in relazione alle effettive dimensioni del viewport il che, in altri termini, equivale a dire che lo Slider non avrà un comportamento responsivo.

Se l'esigenza dovesse invece essere proprio quella di creare uno slider responsivo, in cui il numero di elementi visualizzati contemporaneamente possa variare in relazione alle effettive dimensioni del viewport si dovrà allora ricorrere alle classi di tipo **.uk-width-small-*, .uk-width-medium-*, .uk-width-large-*** ecc…, oppure a quelle del tipo **.uk-grid-width-small-*, .uk-grid-width-medium-*, .uk-grid-width-large-*** ecc… (applicate rispettivamente sui singoli elementi dello slider o sullo slider stesso) già viste in occasione del componente griglia e utilizzate, come noto, proprio per conferire alla griglia stessa un comportamento responsivo.

Facendo quindi riferimento ad un markup del tipo di quello qui di seguito indicato

```
<div class="uk-slidenav-position" data-uk-slider>
  <div class="uk-slider-container">
    <ul class="uk-slider uk-grid-width-medium-1-3 uk-grid-width-large-1-4">
      <li>...</li>
       ...
    </ul>
  </div>

  <a href="" class="uk-slidenav uk-slidenav-previous" data-uk-slider-item="previous"></a>

  <a href="" class="uk-slidenav uk-slidenav-next" data-uk-slider-item="next"></a>

</div>
```

otterremo uno slider con i pulsanti precedente e successivo che visualizzerà:

- per viewport large 4 elementi contemporaneamente
- per viewport medium 3 elementi contemporaneamente
- per viewport small un solo elemento

ATTENZIONE! Per maggiori informazioni sull'utilizzo delle classi .uk-width-small-*, .uk-width-medium-*, .uk-grid-width-small-*, .uk-grid-width-medium-* ecc… si veda anche il capitolo di questa guida relativo al componente Griglia

ELEMENTI CENTRATI

Nella configurazione di default gli elementi dello slider sono tutti allineati sulla sinistra del contenitore più esterno.

Se l'esigenza dovesse essere quella di centrare gli elementi facendo quindi in modo di inserire nell'area di visualizzazione metà dell'elemento successivo e metà dell'elemento precedente, come evidenziato in figura

sarà sufficiente impostare per l' attributo **data-uk-slider** del contenitore più esterno il valore **center:true**

In queste condizioni dunque il markup al quale fare riferimento dovrà essere il seguente:

```
<div data-uk-slider="{center:true}">
  <div class="uk-slider-container">
    <ul class="uk-slider uk-grid-width-medium-1-4">
      <li>...</li>
      ...
    </ul>
  </div>
</div>
```

SCROLLING INFINITO DISABILITATO

Nella configurazione di default il componente slider di uikit è impostato per ciclare all'infinito tra tutti i suoi elementi. Una volta arrivati all'ultimo elemento si ripartirà quindi dal primo.

Se l'esigenza dovesse essere quella di modificare questo comportamento in maniera tale che il loop termini una volta raggiunto l'ultimo elemento (potendo quindi solo tornare indietro) sarà sufficiente impostare per l' attributo **data-uk-slider** del contenitore più esterno il valore **infinite:false**

In queste condizioni dunque il markup al quale fare riferimento dovrà essere il seguente:

```
<div data-uk-slider="{infinite:false}">
  <div class="uk-slider-container">
    <ul class="uk-slider uk-grid-width-medium-1-4">
      <li>...</li>
      ...
    </ul>
  </div>
</div>
```

GUTTER TRA GLI ELEMENTI DELLO SLIDER

Nella configurazione di default gli elementi dello slider sono uno attaccato all'altro. Nel caso in cui l'esigenza dovesse essere quella di inserire dello spazio orizzontale tra uno e l'altro, come evidenziato in figura

sarà sufficiente assegnare allo slider, e quindi allo stesso elemento cui era già stata assegnata la classe .uk-slider, anche una delle classi **.uk-grid, .uk-grid-medium** o **.uk-grid-small** utilizzate, come noto, anche per impostare il gutter orizzontale tra le colonne di una griglia.

In queste condizioni dunque il markup al quale fare riferimento dovrà essere il seguente:

```
<div data-uk-slider">
  <div class="uk-slider-container">
    <ul class="uk-slider uk-grid uk-grid-width-medium-1-4">
      <li>...</li>
      ...
    </ul>
  </div>
</div>
```

SLIDER IN PASSWEB

Il modo più semplice per implementare il componente Slider di uikit all'interno del proprio sito Passweb è quello di utilizzare il componente HTML.

In queste condizioni infatti, posto di aver inserito tramite il layout di Pagina e/o di Variante i collegamenti a tutte le librerie css e js necessarie per il corretto funzionamento dello Slider, sarà poi sufficiente inserire all'interno del componente HTML uno dei markup esaminati nei precedenti capitoli e necessario per realizzare lo Slider secondo le specifiche esigenze del caso.

Per quel che riguarda il collegamento alle varie librerie CSS è possibile seguire due strade differenti:

- Caricare le librerie (slider.css e slidenav.css) dalla sezione "Inclusione nella pagina" del Layout, gestendole quindi come file esterni al Wizard.

  In queste condizioni le regole css necessarie per la corretta formattazione del componente verranno applicate solo lato front-end

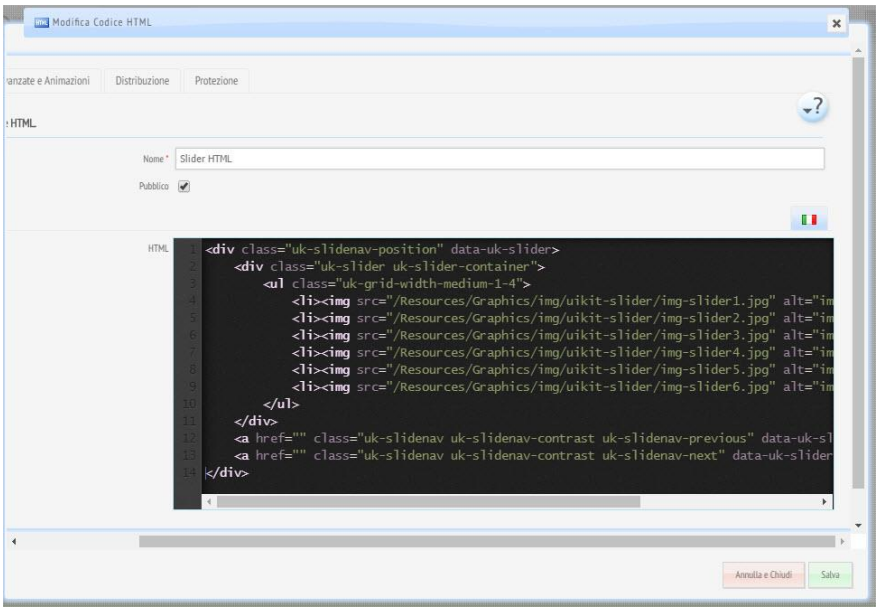
- Inserire il contenuto dei file slider.css e slidenav.css direttamente all'interno della sezione CSS del Layout (come già fatto per la parte core del framework ossia per la libreria uikit.css).

  In queste condizioni le regole CSS necessarie per il corretto funzionamento dello Slider verranno applicate anche all'interno del Wizard

Supponendo ora di voler implementare uno Slider, dotato dei controlli di navigazione e responsivo, in maniera tale che visualizzi, nello specifico:

- per viewport medium e large 4 elementi alla volta
- per viewport small 1 solo elemento alla volta

dovremo allora inserire all'interno del Componente HTML un markup del tipo di quello presente in figura



Il vantaggio nell'utilizzare il componente HTML è dato, chiaramente, dalla possibilità di replicare esattamente il markup richiesto da uikit per ottenere la configurazione dello slider desiderata senza doversi quindi preoccupare di quello che è il markup generato in automatico da Passweb.

Lo svantaggio è rappresentato invece dal fatto di dover gestire tutto, dai contenuti dello slider (nel caso in esame semplici immagini) alla loro formattazione grafica, lato codice.

Inoltre nel momento in cui si dovesse decidere di inserire il contenuto delle librerie slider.css e slidenav.css direttamente nella sezione CSS del Layout, anziché includerle come file esterni, sarà poi possibile visualizzare il risultato ottenuto solo ed esclusivamente sul front-end del sito.

Le classi di gestione dello Slider lo rendono infatti non visibile all'interno del Wizard per cui anche questa volta, come già esaminato per altri casi nel corso di questa guida, il componente HTML in cui è stato inserito il markup dello Slider potrà essere gestito solo ed esclusivamente dalla barra laterale dei componenti.

Oltre al componente HTML, esaminando il markup dei vari Componenti Passweb generato in automatico dall'applicativo possiamo trovare altri due casi in cui poter utilizzare lo Slider di uikit

- il componente comune **Galleria di Immagini** gestibile in una qualsiasi pagina del sito
- il componente Ecommerce **"Immagine Rappresentativa"** gestibile come componente interno alla **Scheda Prodotto**

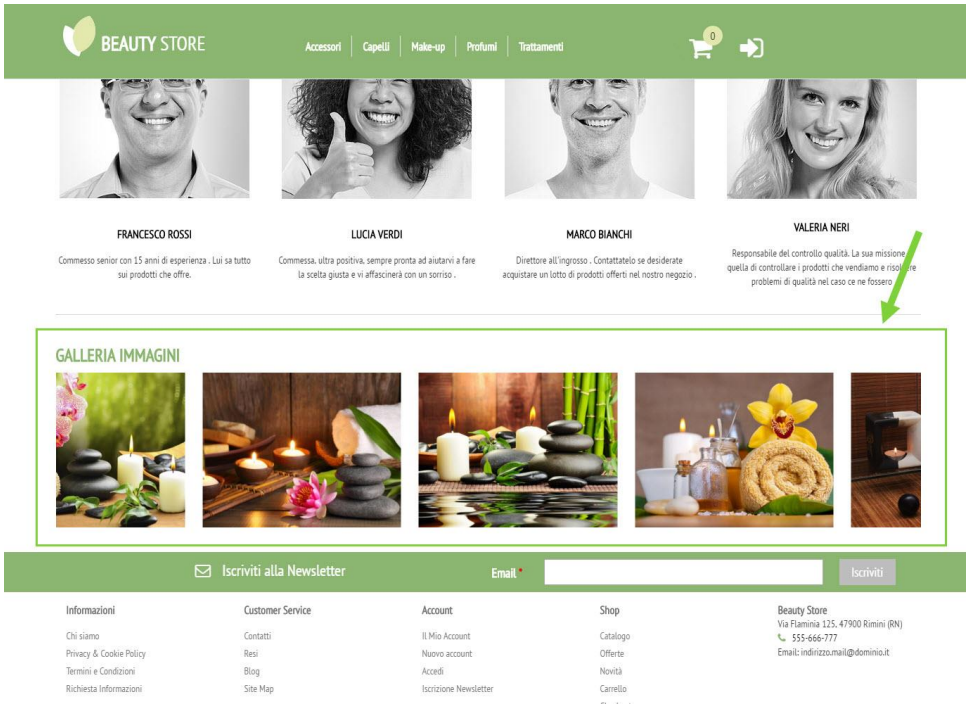
In entrambi i casi infatti abbiamo infatti a che fare con gallerie di immagini gestite esattamente con lo stesso markup richiesto da uikit per realizzare uno slider. In queste condizioni quindi si tratterà di assegnare ai diversi elementi del markup le classi e/o gli attributi corretti in relazione alla specifica configurazione dello slider che si desidera realizzare.

Nel successivo capitolo vedremo come poter applicare lo Slider di uikit sul componente Galleria Immagini di Passweb.

MODELLO ECOMMERCE 29

Nel modello di riferimento non sono presenti Slider gestiti con il Componente HTML.

Nella pagina "**Chi Siamo**" è possibile trovare invece un esempio di applicazione dello Slider di uikit sul Componente Passweb "**Galleria di Immagini**"



Osservando il markup generato in automatico da Passweb per questo componente

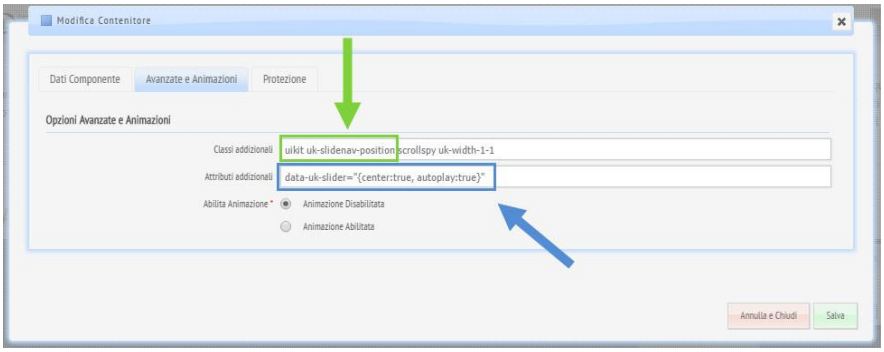
```
<div id="imagegallery_100" class="imagegalleryComp component floated ">
  <ul>
   <li>
              <a href="/Resources/Graphics/wtc7.png" id="galleryimage_1" class="imagegalleryComp-image" title="ImmagineA9"
rel="prettyPhoto[_]">
     <img src="/ThumbGallery/1/100_wtc7.png" alt="ImmagineA9" title="ImmagineA9">
    </a>
   </li>
   <li>
              <a href="/Resources/Graphics/wtc6.png" id="galleryimage_2" class="imagegalleryComp-image" title="ImmagineA8"
rel="prettyPhoto[_]">
     <img src="/ThumbGallery/1/100_wtc6.png" alt="ImmagineA8" title="ImmagineA8">
              </a>
   </li>
     ...
  </ul>
</div>
```

possiamo notare immediatamente come questo sia del tutto analogo a quello richiesta da uikit per implementare uno slider.

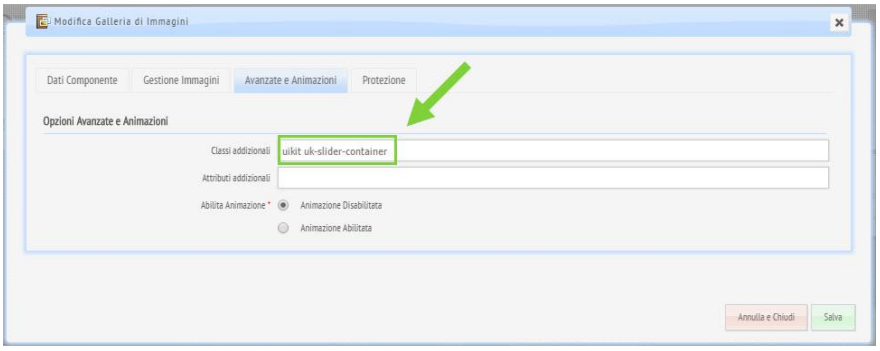
In realtà, in questo markup manca il contenitore esterno dello slider e, nel caso in cui si volessero gestire, i pulsanti di navigazione. Il contenitore esterno allo slider può essere gestito con un semplice Componente Contenitore, mentre la maniera migliore per aggiungere, in queste condizioni, i pulsanti di navigazione è quella di agire via javascript.

Vediamo quindi passo passo quella che è la procedura da seguire per poter ottenere un risultato analogo a quello presente nella pagina “Chi Siamo” del nostro modello di riferimento.

- Inserire un **Componente Contenitore** (Contenitore-Galleria) da utilizzare come contenitore esterno dello Slider al quale assegnare, mediante gli appositi campi di configurazione:
  - la classe **.uikit** necessaria per gestire il “float: none”
  - la classe **.uk-slidenav-position** necessaria per attivare sullo slider i pulsanti di navigazione
  - l’attributo **data-uk-slider="{center:true, autoplay:true}",** dove le opzioni center:true e autoplay:true consentono rispettivamente di centrare gli elementi dello slider e avviare lo scorrimento automatico (con pausa all’hover del mouse)

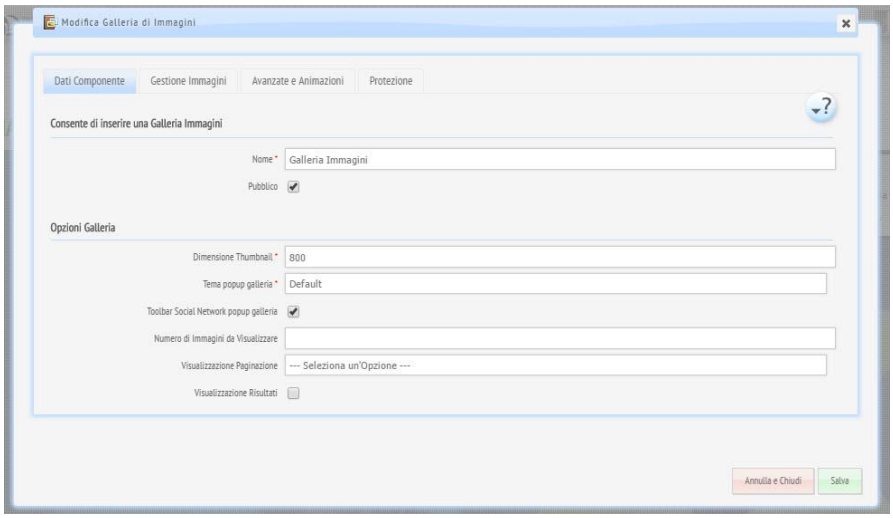


- Inserire all'interno del precedente contenitore un Componente **"Galleria di Immagini"** da utilizzare come Slider e al quale assegnare dunque:
    - la classe **.uikit** necessaria per gestire il "float: none"
    - la classe **.uk-slider-container** necessaria per identificarlo come Slider

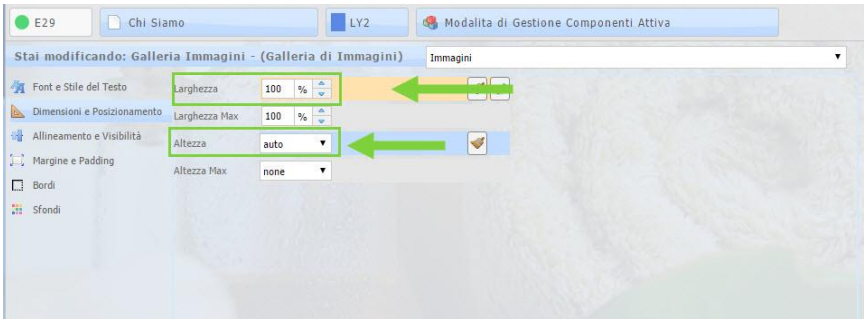


- Configurare il componente Galleria di Immagini inserendo, come normalmente avviene in Passweb

  In questo senso oltre ad aggiungere, ovviamente, le immagini che dovranno rappresentare i contenuti del componente, sarà ance necessario:
    - Non attivare la paginazione, in modo tale da fare sempre visualizzare tutti gli elementi della galleria
    - Impostare come "Dimensione Thumbnail" la dimensione massima delle immagini in maniera tale da evitare uno streching delle miniature nel momento in cui se ne dovesse vedere, all'interno dello slider una sola per volta



    - Impostare, tramite style editor di Passweb, la **Larghezza** dell'elemento "**Immagini**" sul valore **100%** e la sua **Altezza** sul valore "**auto**"



Arrivati a questo punto per terminare la configurazione dello Slider sarebbe sufficiente assegnare identificare la lista dei contenuti dello Slider assegnando per questo al tag < ul > la classe **.uk-slider** oltre alle classi **.uk-grid-medium** e **.uk-grid-width-medium-1-4** necessarie per fare in modo che per viewport medi (e superiori) lo slider visualizzi 4 elementi alla volta mentre per viewport di dimensioni small ed inferiori ne visualizzi solo una alla volta.

Infine andrebbero anche inseriti i link necessari per creare i due pulsanti di navigazione inserendoli nel contenitore esterno dello Slider e posizionandoli esattamente allo livello dello Slider stesso.

Ora considerando che, da una parte non abbiamo la possibilità di assegnare mediante interfaccia grafica delle classi aggiuntive al tag < ul > del componente "Galleria Immagini", e che, dall'altra parte, se inserissimo dentro al Contenitore esterno dello Slider un Componente HTML per gestire i due pulsanti di navigazione non otterremmo esattamente il risultato desiderato (in quanto i pulsanti verrebbero poi inseriti all'interno di un altro div non essendo più allo stesso livello dello Slider), entrambi i problemi si risolvono con due semplici istruzioni javascript, da inserire nell'apposita sezione del layout di pagina:

```
$(document).ready(function() {

  $('#imagegallery_3341 ul').addClass( "uk-slider uk-grid-medium uk-grid-width-medium-1-4" );

  $('#imagegallery_3341').append('<a href="" class="uk-slidenav uk-slidenav-contrast uk-slidenav-previous" data-uk-slider-
item="previous"></a><a href="" class="uk-slidenav uk-slidenav-contrast uk-slidenav-next" data-uk-slider-item="next"></a>');

});
```

La prima delle due istruzioni sopra indicate individua l'id del Componente Galleria Immagini di Passweb presente sulla pagina, prende in considerazione in suo tag < ul > ($('#imagegallery_3341 ul')) e gli assegna (addClass) le classi **.uk-slider .uk-grid-medium .uk-grid-width-medium-1-4** necessarie, come detto, per identificare la lista dei contenuti dello slider e per definire il suo comportamento responsivo

La seconda delle due istruzioni individua sempre l'id del Componente "Galleria Immagini" (che sappiamo essere il nostro Slider) e aggiunge subito dopo di esso (append), allo stesso livello, il markup HTML necessario per creare i due pulsanti di navigazione.

Operando in questo modo riusciremo quindi ad applicare lo Slider di uikit al Componente Galleria Immagini di Passweb ottenendo il duplice effetto di far scorrere le miniature della galleria (come previsto dallo Slider) e di visualizzarne l'ingrandimento cliccandoci sopra con il mouse (come previsto dal componente Galleria Immagini di Passweb)

SLIDESHOW

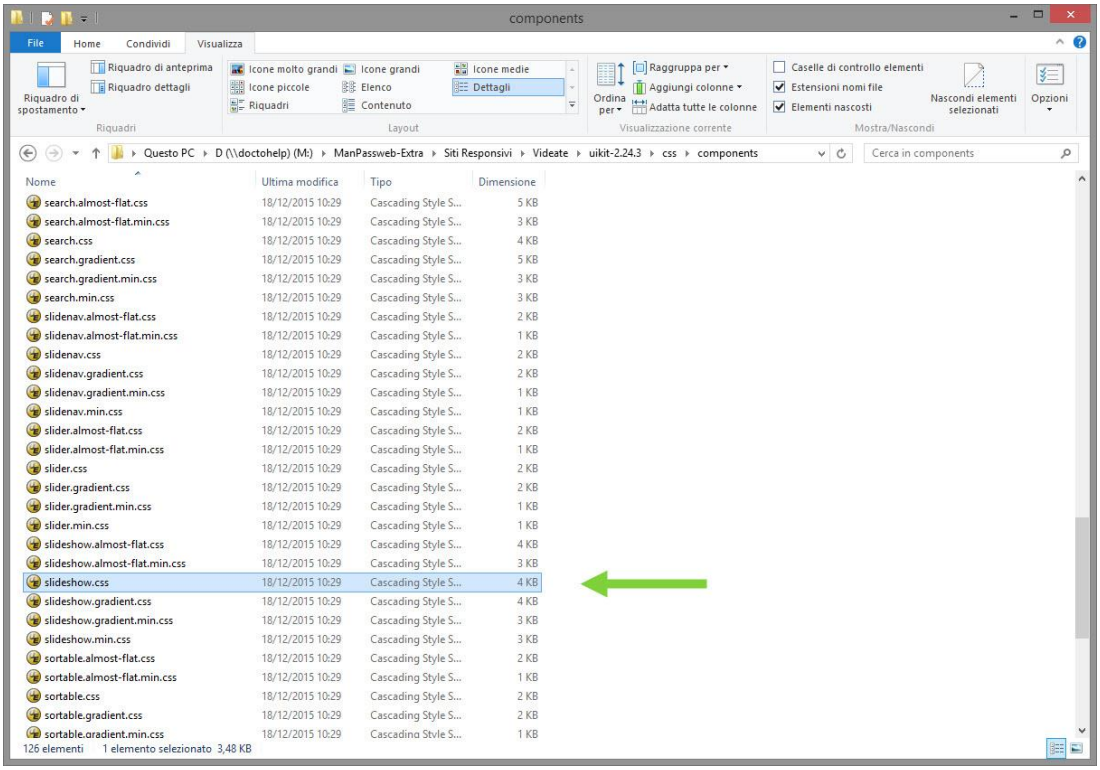
Il componente Slideshow di uikit consente di creare uno slideshow di immagini, testi, video ecc… con diversi possibili effetti di transizione tra un contenuto e l'altro.

Il passaggio tra i diversi contenuti può avvenire in maniera automatica oppure può essere controllato con appositi pulsanti di avanzamento.

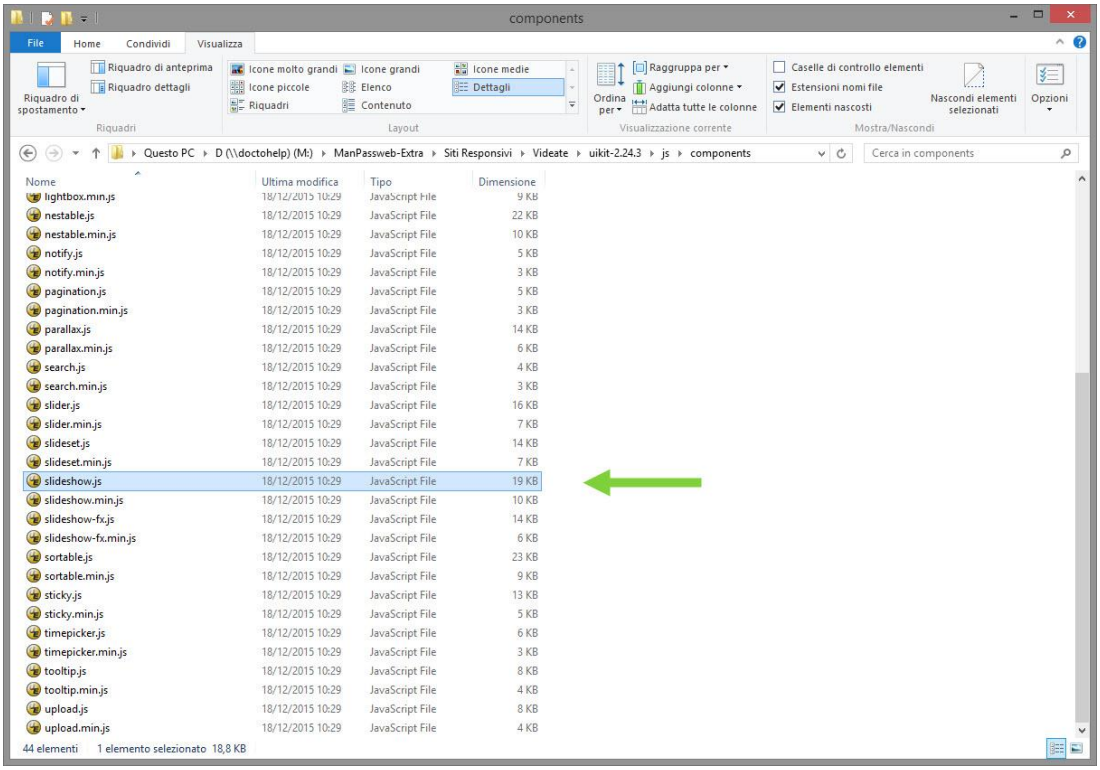
ATTENZIONE! La documentazione ufficiale, assieme ad alcuni esempi di utilizzo, può essere consultata al seguente indirizzo http://getuikit.com/docs/slideshow.html

Al pari dello Slider esaminato nei precedenti capitoli di questa guida, anche lo Slideshow è uno dei componenti avanzati di uikit e, come tale, necessita anche, oltre alla parte core del framework, di specifiche librerie CSS e JS.

In particolare le regole CSS relative a questo componente sono localizzate nel file **slideshow.css** (o nella sua versione minificata) che possiamo trovare nella cartella "**css – components**" ottenuta una volta decompattato l'archivio .zip scaricato dal sito del progetto (per maggiori informazioni in merito al download del framework si veda anche il relativo capitolo di questa guida)



Le istruzioni javascript necessarie per il corretto funzionamento del componente sono localizzate invece nel file **slideshow.js** (o nella sua versione minificata) che possiamo trovare all'interno della cartella "**js – components**"



CONFIGURAZIONE

Come evidenziato nel precedente capitolo di questa guida per poter implementare correttamente il componente Slideshow di uikit è necessario caricare all'interno del sito, oltre alla parte core del framework (uikit.css e uikit.js) anche le librerie css e js utilizzate in maniera specifica da questo componente.

La prima cosa da fare sarà quindi verificare di aver inserito nella sezione < head > della pagina in cui sui vuol implementare questo componente i collegamenti alle librerie slideshow.css e slideshow.js (o meglio ancora alla loro versione minificata)

```
<link rel="stylesheet" href="uikit.css" />
<link rel="stylesheet" href="slideshow.css" />


<script src="uikit.js"></script>
<script src="slideshow.js"></script>
```

ATTENZIONE! La libreria slideshow.js deve necessariamente essere caricata dopo la libreria uikit.js

Fatto ciò sarà poi necessario definire l'elenco dei contenuti dello Slideshow utilizzando per questo un apposito elenco puntato, **tag < ul >**, in cui ogni contenuto dovrà essere gestito come un punto dell' elenco, **e andrà quindi posizionato all'interno di un tag < li >.**

Inoltre sarà necessario assegnare al **tag < ul >** anche:

- La classe **.uk-slideshow** necessaria per identificare l'elenco dei contenuti dello slideshow
- L'attributo **data-uk-slideshow** necessario per poter attivare le funzioni javascript che garantiscono il corretto funzionamento del componente

In questa configurazione di base quindi il markup al quale fare riferimento sarà esattamente il seguente:

```
<ul class="uk-slideshow" data-uk-slideshow>
  <li><img src="…" width="…" height="…" alt="…"></li>
  <li><img src="…" width="…" height="…" alt="…"></li>
  <li><img src="…" width="…" height="…" alt="…"></li>
  ..
</ul>
```

In realtà il markup sopra evidenziato seppur corretto non ci consente di ottenere uno Slideshow completamente funzionante in quanto non contempla una modalità di scorrimento tra si suoi contenuti, per cui nelle condizioni sopra indicate verrà sempre e solo visualizzato il primo contenuto dello Slideshow.

NAVIGAZIONE

Per cercare di risolvere il problema evidenziato nel capitolo precedente e fare quindi in modo di poter scorrere i diversi contenuti dello Slideshow è possibile procedere in due modi differenti: abilitare l'autoplay oppure implementare appositi pulsanti di navigazione.

AUTOPLAY

Per poter abilitare l'autoplay facendo quindi scorrere in automatico i diversi contenuti dello Slideshow **è sufficiente utilizzare nell'attributo data-uk-slideshow l'opzione autoplay:true**

Il seguente markup

```
<ul class="uk-slideshow" data-uk-slideshow="{autoplay:true}">
  <li><img src="…" width="…" height="…" alt="…"></li>
  <li><img src="…" width="…" height="…" alt="…"></li>
  <li><img src="…" width="…" height="…" alt="…"></li>
  ...
</ul>
```

ci consente quindi di ottenere un semplice slideshow di immagini che scorrono tra di loro in maniera completamente automatica (utilizzando come effetto di transizione il fade)

PULSANTI DI NAVIGAZIONE

Oltre all'avanzamento automatico è possibile abilitare anche appositi pulsanti di navigazione tra un contenuto e l'altro.

In questo senso è possibile ricorrere ancora una volta al componente Slidenav esattamente allo stesso modo di quanto fatto per lo Slider esaminato nei precedenti capitoli di questa guida.

Per prima cosa sarà quindi necessario inserire tra le librerie incluse nella sezione < head > della pagina anche il collegamento alla libreria necessaria per il corretto funzionamento del componente Slidenav:

```
<link rel="stylesheet" href="uikit.css" />
<link rel="stylesheet" href="slideshow.css" />
<link rel="stylesheet" href="slidenav.css" />


<script src="uikit.js"></script>
<script src="slideshow.js"></script>
```

Fatto questo per poter poi attivare i pulsanti di navigazione sullo slideshow sarà necessario:

- Racchiudere l'elenco dei contenuti dello Slideshow (quindi il tag < ul >) all'interno di un contenitore (tag < div >) più esterno al quale assegnare:
    - L'attributo **data-uk-slideshow** precedentemente assegnato al tag ul
    - La classe **.uk-slidenav-position**
- Inserire all'interno del precedente contenitore due link **(tag < a >)** che serviranno per creare i pulsanti di precedente e successivo e ai quali dovrà essere assegnata:
    - la classe **.uk-slidenav**
    - la classe **.uk-slidenav-previous**, per il pulsante "precedente" e quella **.uk-slidenav-next**, per il pulsante "successivo"
    - l'attributo **data-uk-slideshow-item="previous"**, per il pulsante "precedente" e quello **data-uk-slideshow-item="next"**, per il pulsante "successivo"

Considerando quindi un markup di questo tipo:

```
<div class="uk-slidenav-position" data-uk-slideshow>

  <ul class="uk-slideshow">
    <li><img src="…" width="…" height="…" alt="…"></li>
    <li><img src="…" width="…" height="…" alt="…"></li>
    <li><img src="…" width="…" height="…" alt="…"></li>
     ...
  </ul>

  <a href="" class="uk-slidenav uk-slidenav-previous" data-uk-slideshow-item="previous"></a>
  <a href="" class="uk-slidenav uk-slidenav-next" data-uk-slideshow-item="next"></a>

</div>
```

nel momento in cui dovessimo passare con il mouse sullo Slideshow compariranno i due pulsanti grazie ai quali poter passare alla slide precedente o a quella successiva.

EFFETTI DI TRANSIZONE

Nella configurazione di default il passaggio tra un contenuto e l'altro dello Slideshow avviene utilizzando l'effetto fade.

Volendo è possibile utilizzare anche un effetto di transizione differente indicandolo come valore dell'opzione **animation** all'interno dell'attributo **data-uk-slideshow**

Considerando ad esempio un markup di questo tipo:

```
<div class="uk-slidenav-position" data-uk-slideshow="{autoplay:true, animation: 'scale'}">

  <ul class="uk-slideshow">
    <li><img src="…" width="…" height="…" alt="…"></li>
    <li><img src="…" width="…" height="…" alt="…"></li>
    <li><img src="…" width="…" height="…" alt="…"></li>
     ...
  </ul>

  <a href="" class="uk-slidenav uk-slidenav-previous" data-uk-slideshow-item="previous"></a>
  <a href="" class="uk-slidenav uk-slidenav-next" data-uk-slideshow-item="next"></a>

</div>
```

otterremo uno Slideshow con avanzamento automatico, dotato dei pulsanti di "precedente" e "successivo" e in cui il passaggio da un contenuto all'altro avverrà utilizzando **l'effetto scale**

ATTENZIONE: Per maggiori informazioni relativamente ai diversi effetti di transizione che è possibile attivare, oltre che per esempi live di tali effetti si consiglia di fare riferimento alla pagina della documentazione ufficiale (http://getuikit.com/docs/slideshow.html). Come indicato all'interno di questa pagina, per poter utilizzare determinati effetti di transizione è necessario inserire nella sezione head della pagina anche il collegamento alla libreria **slideshow-fx-js**

EFFETTO KEN BURNS

Oltre agli effetti di transizione tra un contenuto e l'altro è possibile anche animare il singolo contenuto dello Slideshow con l'effetto Ken Burns.

Per avere un'idea di questo effetto si consiglia di fare riferimento, ancora una volta, alla pagina della documentazione ufficiale nella sezione relativa appunto all'effetto Ken Burns.

Dal punto di vista tecnico per poter attivare questa animazione sui singoli contenuti dello Slideshow è sufficiente utilizzare all'interno dell' attributo **data-uk-slideshow** l'opzione **kenburns:true**

In queste condizioni quindi il markup al quale fare riferimento sarà esattamente del tipo di quello di seguito indicato

```
<div class="uk-slidenav-position" data-uk-slideshow="{kenburns:true}">

  <ul class="uk-slideshow">
    <li><img src="…" width="…" height="…" alt="…"></li>
    <li><img src="…" width="…" height="…" alt="…"></li>
    <li><img src="…" width="…" height="…" alt="…"></li>
     ...
  </ul>

  <a href="" class="uk-slidenav uk-slidenav-previous" data-uk-slideshow-item="previous"></a>
  <a href="" class="uk-slidenav uk-slidenav-next" data-uk-slideshow-item="next"></a>

</div>
```

SLIDESHOW IN PASSWEB

Considerata la finalità di questo componente, oltre che ovviamente il suo markup, il modo più semplice ed immeditato per poterlo implementare all'interno del proprio sito Passweb è quello di ricorrere al Componente HTML.

In queste condizioni infatti, posto di aver inserito tramite il layout di Pagina e/o di Variante i collegamenti a tutte le librerie css e js necessarie per il corretto funzionamento dello Slider, sarà poi sufficiente inserire all'interno del componente HTML uno dei markup esaminati nei precedenti capitoli e necessario per realizzare lo Slideshow secondo le specifiche esigenze del caso.

Per quel che riguarda il collegamento alle varie librerie CSS è possibile, come al solito, seguire due strade differenti:

- Caricare le librerie (slideshow.css e slidenav.css) dalla sezione "Inclusione nella pagina" del Layout, gestendole quindi come file esterni al Wizard.

  In queste condizioni le regole css necessarie per la corretta formattazione del componente verranno applicate solo lato front-end

- Inserire il contenuto dei file slideshow.css e slidenav.css direttamente all'interno della sezione CSS del Layout (come già fatto per la parte core del framework ossia per la libreria uikit.css).

  In queste condizioni le regole CSS necessarie per il corretto funzionamento dello Slideshow verranno applicate anche all'interno del Wizard

Supponendo ora di voler implementare uno Slideshow di immagini, privo di autoplay, dotato dei controlli di navigazione e che utilizzi come effetto di transizione tra i diversi contenuti l'effetto scale dovremo inserire all'interno del Componente HTML un markup del tipo di quello qui di seguito indicato

```
<div style="max-height: 500px;" class="uk-slidenav-position" data-uk-slideshow="{animation: 'scale'}">
  <ul class="uk-slideshow" style="height:500px;">
    <li style="max-height: 500px;"><img src="/Resources/Graphics/img/slidea.jpg" alt="" style="width: 100%; height: auto;">
    </li>
    <li style="max-height: 500px;"><img src="/Resources/Graphics/img/slideb.jpg" alt="" style="width: 100%; height: auto;">
    </li>
    <li style="max-height: 500px;"><img src="/Resources/Graphics/img/slidec.jpg" alt="" style="width: 100%; height: auto;">
    </li>
    ...
  </ul>
  <a href="#" class="uk-slidenav uk-slidenav-contrast uk-slidenav-previous" data-uk-slideshow-item="previous"></a>
  <a href="#" class="uk-slidenav uk-slidenav-contrast uk-slidenav-next" data-uk-slideshow-item="next"></a>
</div>
```

Ancora una volta ,il vantaggio nell'utilizzare il componente HTML è dato dalla possibilità di replicare esattamente il markup richiesto da uikit per ottenere la configurazione dello slideshow desiderata senza doversi quindi preoccupare di quello che è il markup generato in automatico da Passweb.

Lo svantaggio è rappresentato invece dal fatto di dover gestire tutto, dai contenuti (nel caso in esame semplici immagini) alla loro formattazione grafica, lato codice.

E' possibile trovare un esempio di Slideshow, implementato mediante l'utilizzo del Componente HTML, nella Home Page del modello di riferimento

PARALLAX

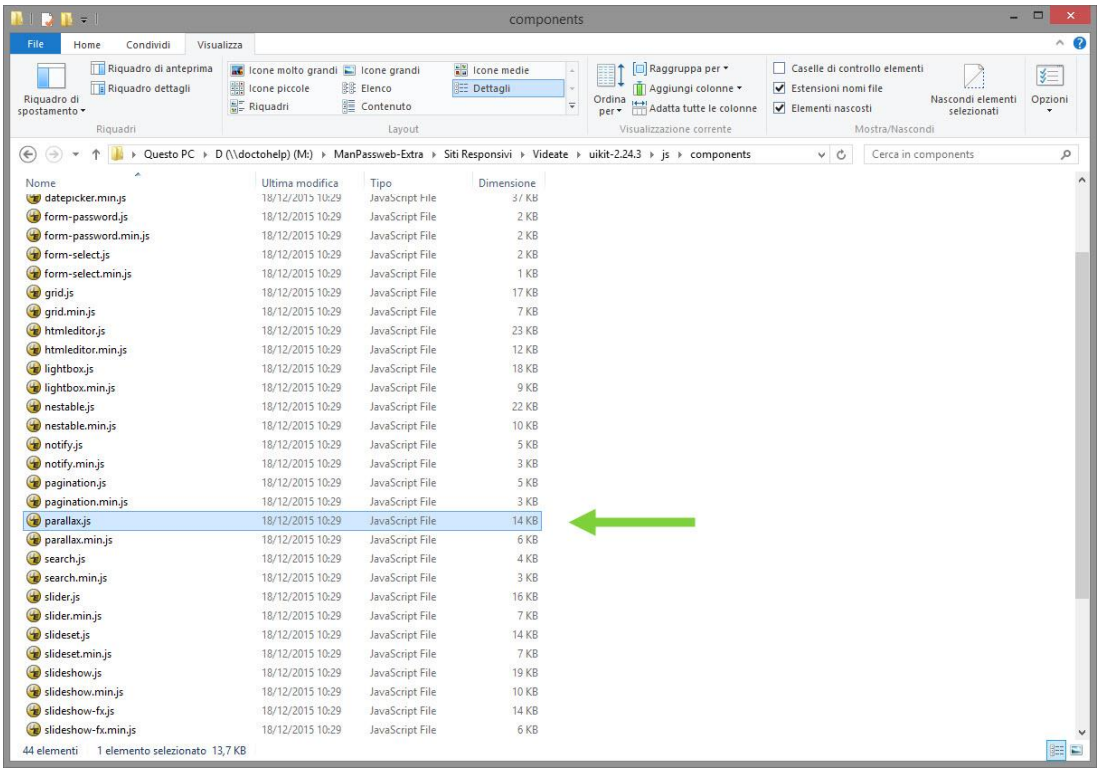
Il componente Parallax di uikit consente di animare determinate proprietà CSS in base allo scrolling del browser.

Tipicamente è utilizzato per far scorrere l'immagine di sfondo di un contenitore ad una velocità differente rispetto al resto della pagina creando il classico effetto di parallasse.

ATTENZIONE! La documentazione ufficiale, assieme ad alcuni esempi di utilizzo, può essere consultata al seguente indirizzo http://getuikit.com/docs/parallax.html

Anche il Parallax, come già lo Slider o lo Slideshow, è uno dei componenti avanzati di uikit e, come tale, necessita anche, oltre alla parte core del framework, di specifiche librerie CSS e JS.

Nello specifico per il componente in oggetto non sono necessarie specifiche librerie CSS (sotto questo punto di vista è quindi sufficiente la parte core del framework, uikit.css). Per quel che riguarda invece le istruzioni javascript necessarie per il corretto funzionamento del componente, queste sono localizzate nel file **parallax.js** (o nella sua versione minificata) che possiamo trovare all'interno della cartella "**js – components**"



CONFIGURAZIONE

Come evidenziato nel precedente capitolo per poter implementare correttamente il componente Parallax di uikit è necessario caricare all'interno del sito, oltre alla parte core del framework (uikit.css e uikit.js) anche la libreria js utilizzata in maniera specifica da questo componente.

La prima cosa da fare sarà quindi verificare di aver inserito nella sezione < head > della pagina in cui sui vuol implementare questo componente il collegamento alla libreria parallax.js (o meglio ancora alla sua versione minificata)

```
<link rel="stylesheet" href="uikit.css" />

<script src="uikit.js"></script>
<script src="parallax.js"></script>
```

ATTENZIONE! La libreria parallax.js deve necessariamente essere caricata dopo la libreria uikit.js

Fatto questo sarà poi sufficiente **assegnare al contenitore per il quale si vuol animare una certa proprietà CSS allo scrolling del browser, l'attributo data-uk-parallax indicando come opzione la specifica proprietà CSS da animare e, ovviamente, il valore che deve essere utilizzato per realizzare l'animazione stessa**.

Di seguito è indicato un elenco di alcune opzioni utilizzabili sull'attributo data-uk-parallax e la relativa proprietà CSS che consentono di animare.

OPZIONE	DESCRIZIONE
x	Consente di animare la posizione sull'asse x del contenitore cui è stato assegnato l'attributo data-uk-parallax.

	E' necessario indicare di quanti pixel dovrà traslare orizzontalmente il contenitore allo scrolling del browser.
xp	Consente di animare la posizione sull'asse x del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare il valore in percentuale della traslazione orizzontale allo scrolling del browser.
y	Consente di animare la posizione sull'asse y del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare di quanti pixel dovrà traslare verticalmente il contenitore allo scrolling del browser.
yp	Consente di animare la posizione sull'asse x del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare il valore in percentuale della traslazione verticale allo scrolling del browser.
bg	Consente di animare l'immagine di background del contenitore cui è stato assegnato l'attributo data-uk-parallax facendola scorre ad una velocità differente rispetto al resto della pagina. E' necessario indicare la differenza di velocità in pixel
bgp	Consente di animare l'immagine di background del contenitore cui è stato assegnato l'attributo data-uk-parallax facendola scorre ad una velocità differente rispetto al resto della pagina. E' necessario indicare la differenza di velocità in percentuale
rotate	Consente di animare la rotazione in senso orario del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare di quanti gradi dovrà ruotare il contenitore allo scrolling del browser.
scale	Consente di animare lo scaling del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare di quanto dovrà scalare il contenitore allo scrolling del browser
color	Consente di animare il colore del testo presente all'interno del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare il colore di partenza e quello di arrivo
background-color	Consente di animare il colore di sfondo del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare il colore di partenza e quello di arrivo
border-color	Consente di animare il colore dei bordi del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare il colore di partenza e quello di arrivo
opacity	Consente di animare l'opacità del contenitore cui è stato assegnato l'attributo data-uk-parallax. E' necessario indicare un valore compreso tra 0 e 1

Supponendo dunque di voler creare il classico effetto di parallasse sull'immagine di sfondo di un contenitore, il markup al quale fare riferimento sarà, molto semplicemente, il seguente:

```
<div id="contenitore-parallax" data-uk-parallax="{bg: '-800'}">...</div>
```

dove l'immagine di sfondo dovrà, ovviamente, essere impostata sul div "contenitore-parallax".

ATTENZIONE! Di base il componente Parallax consente di animare una qualsiasi proprietà CSS che abbia un solo valore. E' possibile, ad esempio, animare l'altezza (height) o la larghezza (width) del contenitore cui è applicato l'attributo data-uk-parallax, indicando la relativa proprietà come opzione dell'attributo e specificando anche il valore cui deve tendere la proprietà in oggetto allo scrolling del browser.

E' anche bene sottolineare poi come, a default:

- l'animazione della proprietà CSS va dal valore attuale della proprietà stessa verso il valore indicato nell'opzione dell'attributo data-uk-parallax
- l'animazione inizia nel momento in cui il contenitore cui è stato assegnato l'attributo data-uk-parallax compare all'interno della pagina e termina nel momento in cui lo stesso contenitore scompare

Entrambe queste configurazioni di default possono comunque essere modificate.

VALORI DI STAR E STOP

Come evidenziato nel precedente capitolo, a default, la proprietà oggetto del componente Parallax di uikit si anima dal suo valore attuale verso il valore indicato all'interno dell'attributo data-uk-parallax.

Volendo è possibile modificare questo comportamento indicando esattamente il valore iniziale (start) e quello finale (stop) dell'animazione.

Per fare questo è sufficiente passare all'opzione indicata nell'attributo data-uk-parallax una stringa costituita da due diversi valori separati da una virgola.

Il primo rappresenta il valore che dovrà assumere la proprietà in oggetto all'inizio dell'animazione, il secondo rappresenta invece il valore che dovrà assumere la stessa proprietà alla fine dell'animazione.

ATTENZIONE! Come indicato nella tabella riportata nel capitolo precedente, alcune delle proprietà animabili con il componente Parallax di uikit (color, background-color ecc…) necessitano obbligatoriamente di entrambi i valori di star e stop

Facendo quindi riferimento ad un markup di questo tipo

```
<div id="contenitore-parallax" {x: '-100,100', 'background-color': '#ffffff,#ff2200'}">...</div>
```

il "contenitore-parallax" partirà, non appena visibile all'interno della pagina, spostato, rispetto alla sua posizione attuale, di 100px verso sinistra. Allo scrolling del browser verso il basso il contenitore traslerà in orizzontale spostandosi verso destra ed arrivando ad occupare al termine dell'animazione una posizione spostata di 200px verso destra.

Durante questa animazione inoltre il suo colore di sfondo passerà dal bianco (#ffffff) al rosso (#ff2200)

ANIMAZIONI ANNIDATE

Grazie al componente Parallax è possibile creare, in maniera molto semplice, delle animazioni annidate.

Per fare questo è sufficiente inserire all'interno del primo contenitore cui è stato assegnato l'attributo data-uk-parallax, un secondo contenitore al quale assegnare, ancora una volta, l'attributo data-uk-parallax e le relative proprietà da animare.

Il markup al quale fare riferimento, sarà quindi di questo tipo:

```
<div data-uk-parallax="{bg: -200}">
  <div data-uk-parallax="{opacity: '0,1', scale: '0,1'}">...</div>
</div>
```

VELOCITA' DELL'ANIMAZIONE

E' possibile variare la velocità dell'animazione oggetto del componente Parallax utilizzando, all'interno del solito attributo data-uk-parallax l'opzione velocity

Facendo quindi riferimento ad un markup di questo tipo:

```
<div data-uk-parallax="{velocity: '0.5'}">...</div>
```

otterremo una velocità di animazione dimezzata rispetto al suo valore originale

DURATA DELL'ANIMAZIONE

Come indicato nei precedenti capitoli di questa guida nella configurazione di default l'animazione, oggetto del componente Parallax, inizia nel momento in cui il contenitore cui è stato assegnato l'attributo data-uk-parallax compare all'interno della pagina e termina nel momento in cui lo stesso contenitore scompare.

E' comunque possibile personalizzare la durata di questa animazione in due modi differenti:

- utilizzando l'opzione **target**
- utilizzando l'opzione **viewport**

L'opzione **target** consente di legare la durata dell'animazione alla visibilità, all'interno del viewport, di un componente diverso da quello cui è stato assegnato l'attributo data-uk-parallax.

Il valore dell'opzione target dovrà quindi essere esattamente l'id del componente a cui legare la durata dell'animazione.

Facendo quindi riferimento ad un markup di questo tipo

```
<div id="elemento-target">...</div>

<div id="contenitore-parallax" data-uk-parallax="{ bg: -200, target: '#elemento-target'}">...</div>
```

otterremo che l'animazione impostata sul background del "contenitore-parallax" inizierà nel momento in cui il div "elemento-target" diventerà visibile all'interno della pagina e terminerà nel momento in cui questo stesso elemento scomparirà dal viewport.

L'opzione **viewport** consente invece di personalizzare la durata dell'animazione in relazione alla visibilità all'interno del viewport dell'elemento cui viene applicato l'attributo data-uk-parallax.

In questo senso impostando questa opzione sul valore **1** oppure sul valore **false** otterremo esattamente il comportamento di default: l'animazione inizierà nel momento in cui il contenitore cui è stato assegnato l'attributo data-uk-parallax compare all'interno della pagina e terminerà nel momento in cui lo stesso contenitore scompare

Impostando invece questa opzione ad esempio, sul valore 0.5, l'animazione inizierà nel momento in cui il contenitore cui è stato assegnato l'attributo data-uk-parallax compare all'interno della pagina e terminerà quando lo stesso contenitore raggiungerà la metà del viewport.

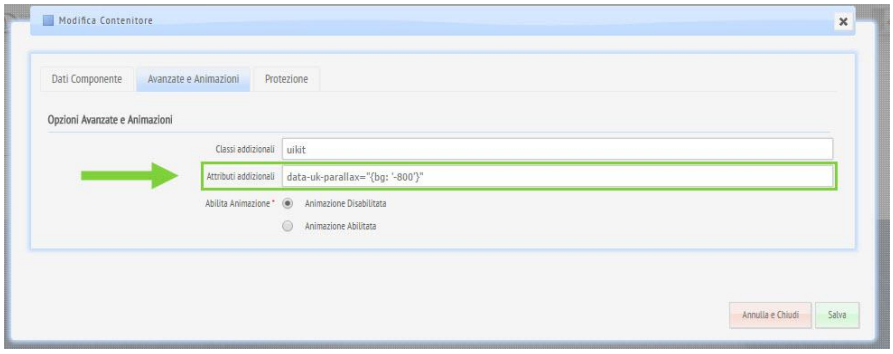
Il markup al quale fare riferimento per questa opzione sarà quindi il seguente:

```
<div id="contenitore-parallax" data-uk-parallax="{bg: -200, viewport: '0.5'}">...</div>
```

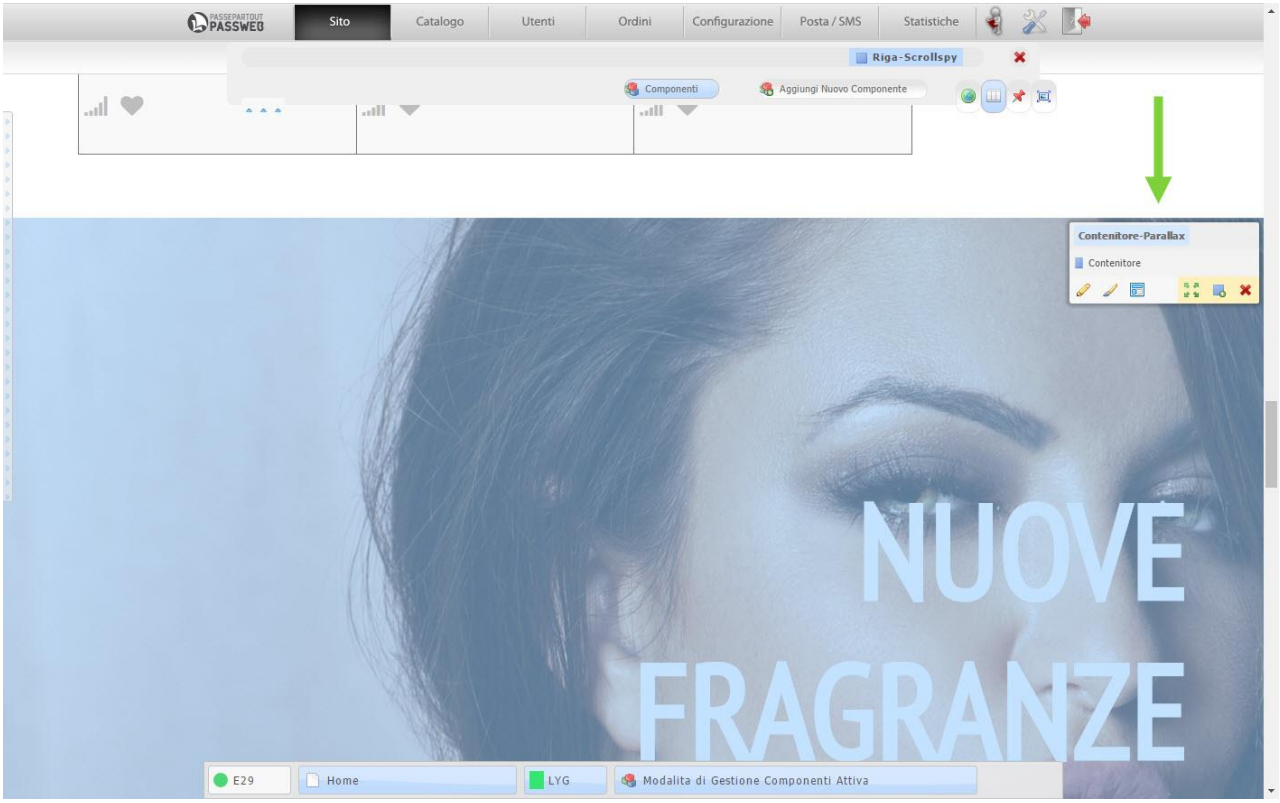
PARALLAX IN PASSWEB

Considerato quanto visto nei precedenti capitoli di questa guida, l'implementazione in Passweb del componente Parallax è estremamente semplice.

Posto infatti di aver inserito tramite il layout di Pagina e/o di Variante i collegamenti a tutte le librerie css e js necessarie per il corretto funzionamento del componente, il tutto si riduce poi ad assegnare l'attributo **data-uk-parallax** con le opzioni e i valori desiderati (in relazione al tipo di configurazione che si vuole ottenere) al Componente per il quale si desidera animare una certa proprietà CSS allo scrolling del browser e, come noto, questo può essere fatto mediante l'apposito campo presente tra i parametri di configurazione di ogni componente.



E' possibile trovare un esempio di applicazione del componente Parallax di uikit nella Home Page del nostro modello di riferimento (Componente Contenitore "Contenitore-Parallax")



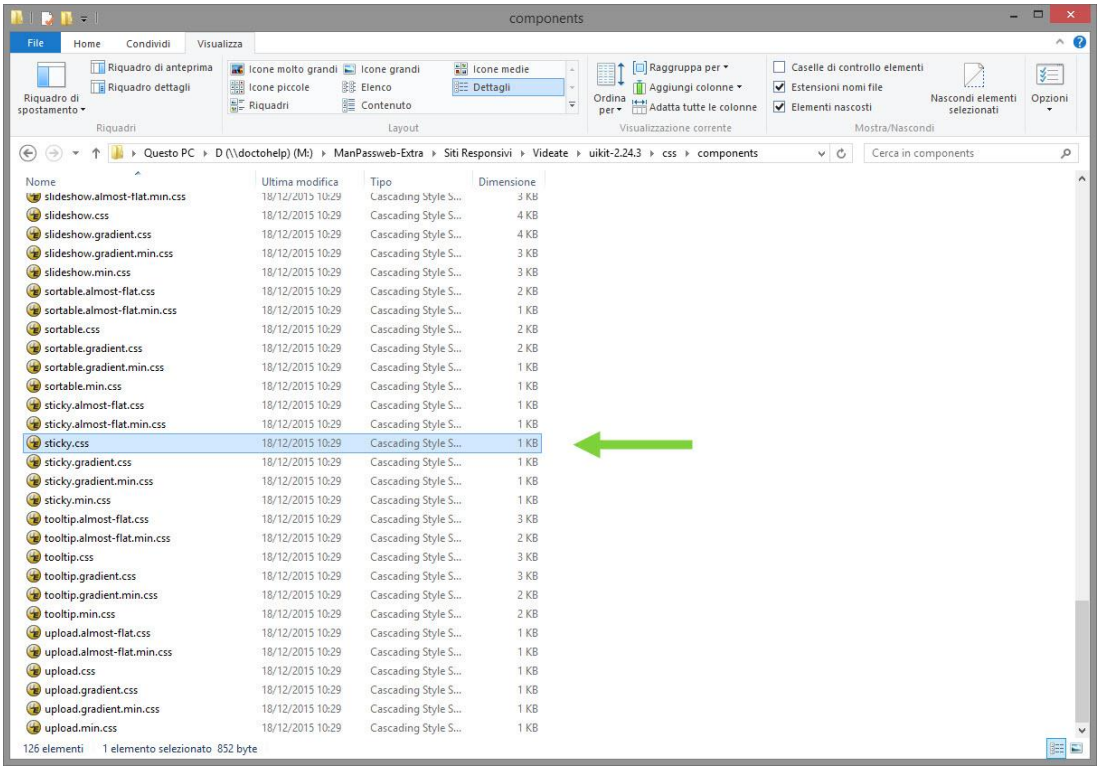
STICKY

Il componente Sticky di uikit consente di fissare l'elemento cui viene applicato al bordo superiore del viewport (ancorandolo quindi alla parte alta della finestra del browser).

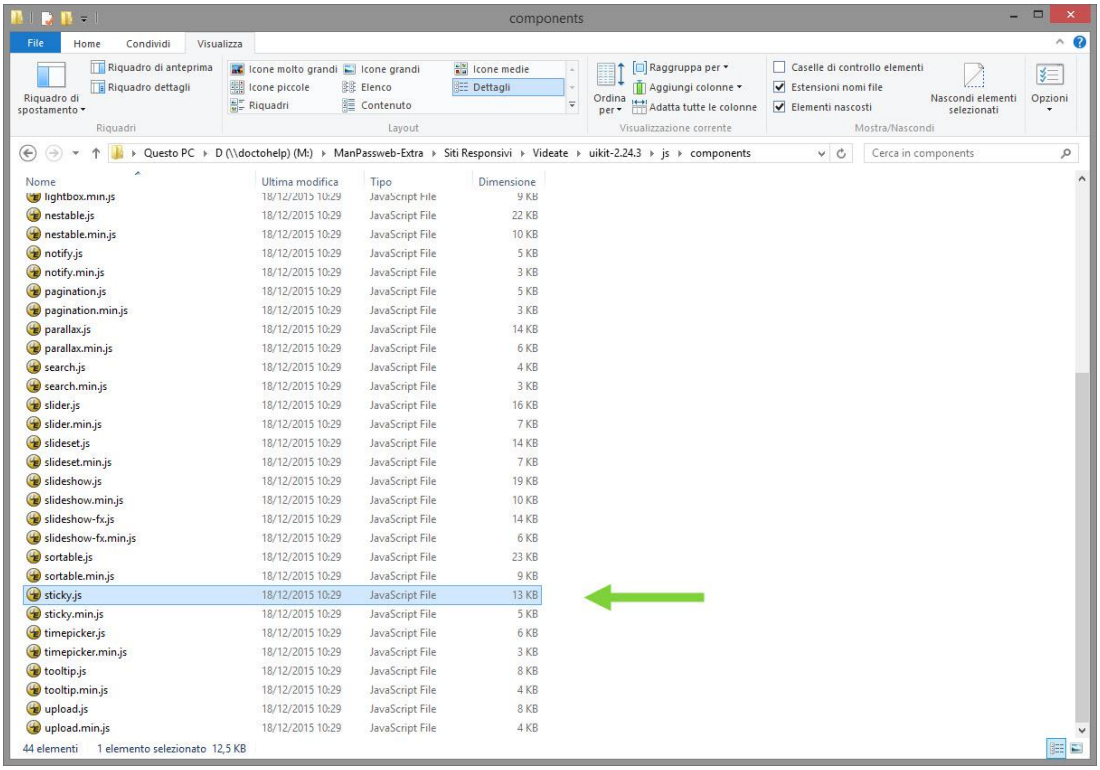
ATTENZIONE! La documentazione ufficiale, assieme ad alcuni esempi di utilizzo, può essere consultata al seguente indirizzo http://getuikit.com/docs/sticky.html

Anche il componente Sticky, al pari dello Slider, dello Slideshow e del Parallax, è uno dei componenti avanzati di uikit e, come tale, necessita, oltre alla parte core del framework, anche di specifiche librerie CSS e JS.

In particolare le regole CSS relative a questo componente sono localizzate nel file **sticky.css** (o nella sua versione minificata) che possiamo trovare nella cartella "**css – components**" ottenuta una volta decompattato l'archivio .zip scaricato dal sito del progetto (per maggiori informazioni in merito al download del framework si veda anche il relativo capitolo di questa guida)



Le istruzioni javascript necessarie per il corretto funzionamento del componente sono localizzate invece nel file **sticky.js** (o nella sua versione minificata) che possiamo trovare all'interno della cartella "**js – components**"



CONFIGURAZIONE

Come evidenziato nel precedente capitolo per poter implementare correttamente il componente Sticky di uikit è necessario caricare all'interno del sito, oltre alla parte core del framework (uikit.css e uikit.js) anche le librerie css e js utilizzate in maniera specifica da questo componente.

La prima cosa da fare sarà quindi verificare di aver inserito nella sezione < head > della pagina in cui sui vuol implementare questo componente il collegamento alle librerie sticky.css e sticky.js (o meglio ancora alle loro versioni minificate)

```
<link rel="stylesheet" href="uikit.css" />
<link rel="stylesheet" href="sticky.css" />

<script src="uikit.js"></script>
<script src="sticky.js"></script>
```

ATTENZIONE! La libreria sticky.js deve necessariamente essere caricata dopo la libreria uikit.js

Fatto questo, per fare in modo che un dato elemento rimanga ancorato alla parte alta della finestra del browser, sarà sufficiente assegnare a questo stesso elemento l'attributo **data-uk-sticky**

Nella configurazione di base quindi il markup al quale fare riferimento sarà, molto semplicemente, il seguente:

```
<div id="elemento-sticky" data-uk-sticky>...</div>
```

In questa condizioni, quando la div "elemento-sticky" raggiungerà la parte alta della pagina si ancorerà al bordo superiore senza più seguire il resto della pagina.

OFFSET

Nel caso in cui l'esigenza dovesse essere quella di ancorare un dato elemento non esattamente al bordo superiore del viewport ma più in basso rispetto ad esso, **è possibile utilizzare sull'attributo data-uk-sticky l'opzione top** indicando anche a quanti pixel dal bordo superiore del viewport dovrà essere ancorato l'elemento in questione.

Facendo quindi riferimento ad un markup di questo tipo

```
<div id="elemento-sticky" data-uk-sticky="{top:100}">...</div>
```

otterremo che la div "elemento-sticky" resterà ancorata ad una distanza di 100 pixel dal bordo superiore della finestra del browser

Un'altra opzione interessante del componente Sticky di uikit è quella che consente di ancorare un dato elemento al bordo superiore del viewport o ad una certa distanza rispetto ad esso, non immediatamente nel momento in cui l'elemento in questione raggiunge quella posizione, ma soltanto dopo aver effettuato un ulteriore scrolling di N pixel.

Per fare questo è sufficiente assegnare all'opzione top un valore negativo.

Facendo quindi riferimento al markup di seguito indicato

```
<div id="elemento-sticky" data-uk-sticky="{top:-200}">...</div>
```

otterremo che la div "elemento-sticky" resterà ancorata al bordo superiore del viewport. L'ancoraggio però non avverrà subito, ossia esattamente nel momento in cui questo elemento arriverà a toccare il bordo superiore del viewport, ma soltanto dopo aver effettuato un ulteriore scrolling di 200 pixel verso il basso.

La div "elemento-sticky" potrebbe, quindi, anche scomparire totalmente dall'area di visualizzazione del viewport per poi ricomparire, ancorata al bordo superiore, nel momento in cui la pagina scorrerà verso il basso di altri 200 pixel.

In queste condizioni, infine, potrebbe essere interessante gestire la "ricomparsa" della div "elemento-sticky" mediante un'animazione.

Per questo è possibile utilizzare una qualsiasi animazione definita per il componente "**Animation**" di uikit (http://getuikit.com/docs/animation.html) indicando il nome dell'animazione da utilizzare come valore per l'opzione "**animation**" dell'attributo data-uk-sticky

Il markup al quale fare riferimento in queste condizioni sarà dunque il seguente:

```
<div id="elemento-sticky" data-uk-sticky="{top:-200}, animation: 'uk-animation-slide-top'}">...</div>
```

COMPORTAMENTO RESPONSIVO

Il componente Sticky di uikit può essere disabilitato in corrispondenza di certe dimensioni del viewport in modo tale, ad esempio, che un dato elemento resti ancorato al bordo superiore del browser solo per schermi di grandi dimensioni, mentre su smartphone o tablet continui a comportarsi normalmente seguendo, come tutti gli altri elementi, lo scrolling della pagina.

Per fare questo e sufficiente utilizzare l'opzione "**media**" dell'attributo data-uk-sticky indicando esattamente le dimensioni del viewport al di sotto delle quali l'effetto sticky dovrà essere disabilitato.

Facendo quindi riferimento ad un markup di questo tipo:

```
<div id="elemento-sticky" data-uk-sticky="{media: 640}">...</div>
```

otterremo che la div "elemento-sticky" resterà effettivamente ancorata alla parte superiore del browser solo per viewport di dimensioni maggiori di 640 pixel

Oltre alla dimensione del viewport l'effetto sticky può essere condizionato anche ad una qualsiasi altra media-query che dovrà essere indicata per l'attributo data-uk-sticky, come valore dell'opzione media.

Facendo quindi riferimento ad un markup di questo tipo:

```
<div id="elemento-sticky" data-uk-sticky="{media: '(min-width: 640px) and (orientation: landscape)'}>...</div>
```

otterremo che la div "elemento-sticky" resterà effettivamente ancorata alla parte superiore del browser solo per viewport di dimensioni maggiori di 640 pixel e solo nel caso in cui questi siano orientati in modalità landscape.

CLASSI PERSONALIZZATA SULL'ELEMENTO STICKY

Un'ultima opzione particolarmente interessante da prendere in considerazione per il componente Sticky, è quella che consente di assegnare all'elemento ancorato al bordo superiore della pagina, nel momento in cui diventa "sticky", una classe CSS personalizzata.

E' quindi possibile sfruttare questa classe personalizzata per formattare l'elemento in questione quando diventa sticky in maniera diversa da quando non lo è.

Per fare questo è sufficiente utilizzare l'opzione **clsactive** dell'attributo data-uk-sticky, indicando il nome della classe che si desidera assegnare all'elemento nel momento in cui questo diventerà "sticky".

Facendo quindi riferimento ad un markup di questo tipo

```
<div id="elemento-sticky" data-uk-sticky="{clsactive: 'my-class'>...</div>
```

nel momento in cui la div "elemento-sticky" andrà ad ancorarsi al bordo superiore del browser gli verrà applicata la classe ".my-class" con le relative formattazioni grafiche.

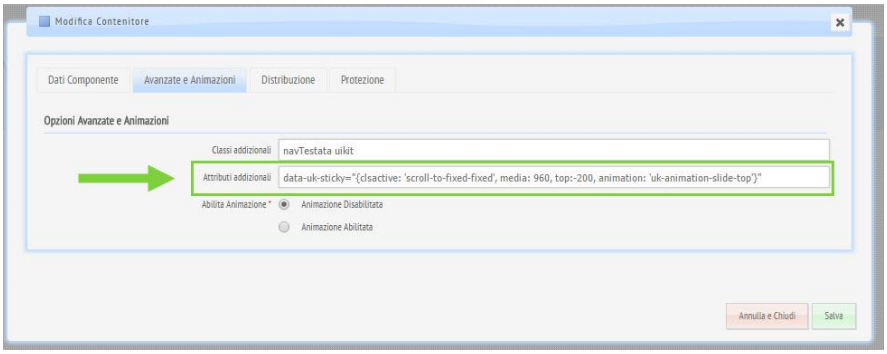
Operando allo stesso modo è possibile assegnare all'elemento sticky delle classi personalizzate anche nel momento in cui questo diventa "sticky" per la prima volta o quando torna ad essere non sticky. Nello specifico l'opzione:

- **clsinit:** consente di impostare una classe personalizzata sull'elemento sticky nel momento in cui questo diventerà sticky per la prima volta
- **clsinactive**: consente di impostare una classe personalizzata sull'elemento sticky che verrà assegnata all'elemento stesso nel momento in cui questo non dovesse essere ancorato alla parte superiore del browser

STICKY IN PAASSWEB

Come già per il componente Parallax, anche l'implementazione del componente Sticky all'interno del proprio sito Passweb è estremamente semplice.

Posto infatti di aver inserito tramite il layout di Pagina e/o di Variante i collegamenti a tutte le librerie css e js necessarie per il corretto funzionamento del componente, il tutto si riduce, anche in questo caso, ad assegnare l'attributo **data-uk-sticky** con le opzioni e i valori desiderati (in relazione al tipo di configurazione che si vuole ottenere) al Componente che si desidera ancorare al bordo superiore del viewport e, come noto, questo può essere fatto mediante l'apposito campo presente tra i parametri di configurazione di ogni componente.



E' possibile trovare un esempio di applicazione del componente Sticky di uikit in una qualsiasi pagina del nostro modello di riferimento.

In Testata Alta è stato infatti inserito un Componente Contenitore denominato **"Contenitore-Sticky"** al quale è stato assegnato l'attributo data-uk-sticky con le seguenti opzioni:

data-uk-sticky="{top:-200, animation: 'uk-animation-slide-top', media: 960, clsactive: 'scroll-to-fixed-fixed'}"

Il risultato ottenuto è che:

- il "Contenitore-Sticky" (e tutti i componenti interni ad esso) si ancorerà al bordo superiore del browser non quando questo stesso componente raggiunge tale posizione ma soltanto dopo un ulteriore scrolling della pagina di 200 pixel verso il basso
- la comparsa del "Contenitore-Sticky" ancorato al bordo superiore del browser è gestita con un animazione di slide dall'alto verso il basso
- nel momento in cui il "Contenitore-Sticky" riappare ancorato al bordo superiore del browser gli verrà assegnata la classe ".scroll-to-fixed-fixed" che è stata poi utilizzata nella sezione CSS del layout di Variante per personalizzare graficamente questo elemento (e quelli interni ad esso)
- l'effetto sticky verrà disabilitato per viewport inferiori ai 640 pixel

RISORSE

- **gridCatalogo.css** – contiene l'integrazione al file standard uikit.css necessaria per poter assegnare al componente Passweb "Catalogo E-commerce" (e simili) un comportamento responsivo operando secondo la logica standard di uikit.

  Consente di utilizzare le classi di tipo **pw-grid**

  Download

- **formUtente.css** – contiene l'integrazione al file standard uikit.css necessaria per poter implementare la Griglia di uikit all'intero del Componenti Passweb "Registrazione Utente" e "Profilo Utente" operando secondo la logica standard di uikit

  Consente di utilizzare le classi di tipo **fr-grid**

  Download

- **uikitEc29.css** – versione completa del file uikit.css utilizzata nel modello di riferimento (www.ecommerce29.passweb.it).

  Contiene le integrazioni al file uikit.css standard necessarie per poter assegnare al componente Passweb "Catalogo E-commerce" (e simili) un comportamento responsivo e per poter implementare la Griglia di uikit all'intero del Componenti Passweb "Registrazione Utente" e "Profilo Utente".

  Contiene inoltre la personalizzazione al file uikit.css standard e necessaria per evitare che alcune delle regole relative al componente Griglia di uikit vengano applicate anche all'overlay (riquadro blu) utilizzato nel Live Editing per evidenziare, al passaggio del mouse, i componenti da gestire.

  Consente di utilizzare sia le classi di tipo **pw-grid** che quelle di tipo **fr-grid**

UIKIT.CSS E PASSWEB

Una volta comprese le caratteristiche ed il funzionamento di base del framework possiamo ora passare ad esaminare più nel dettaglio come questo possa essere integrato ed utilizzato in Passweb.

Ovviamente non verranno presi in considerazione tutti i componenti del framework perché non tutti sono effettivamente indispensabili per realizzare un sito responsivo.

D'altra parte verrà invece spiegato come poter rendere responsivo, ad esempio, il Catalogo Ecommerce di Passweb adottando per questo la stessa logica del framework uikit.

L'obbiettivo principale, in definitiva, è sempre quello di capire come dover operare per combinare uikit e Passweb al fine di ottenere un sito ecommerce responsivo facilmente gestibile.

In ogni caso una volta compreso come poter gestire una griglia, un pannello o uno slider, poi il modo di operare sarà sempre lo stesso e il fatto di utilizzare un elemento piuttosto che un altro dipenderà esclusivamente dallo specifico progetto da realizzare.

Per tutti i dettegli sui vari componenti del framework e sulle relative possibilità di configurazione si rimanda quindi alla relativa documentazione presente sul sito del progetto http://getuikit.com/index.html, documentazione alla quale faremo comunque riferimento anche nei successivi capitoli di questa guida

Il sito di riferimento è raggiungibile all'indirizzo www.ecommerce29.passweb.it