SITO PORTFOLIO

di Carla Manavella

carlaisabelle.github.io/CM_Portfolio

INDICE

P. 02
La Consegna
I requisiti richiesti per la creazione del sito

P. 03
La Preparazione
Gli step preparatori

P. 07
Bootstrap e SASS
Perché li ho scelti e come li ho usati

P. 09
Navbar e Footer
Come ho costruito navbar e footer

P. 11
Gli Heading
Come ho gestito i titoli principali delle pagine

P. 13
La Homepage
I componenti principali della homepage

P. 18
"Chi Sono"
I componenti principali della pagina "Chi sono"

P. 20
"Curriculum"
I componenti principali della pagina "Curriculum"

P. 22
"Progetti"
I componenti della pagina "Progetti"

P. 24
"Contattami"
I componenti della pagina "Contattami"

LA CONSEGNA

Il sito portofolio è stato costruito (anche) come prova pratica per il corso-master “Blockchain Development” proposto dalla piattaforma “Start2Impact”. Per questo motivo, sebbene la parte creativa sia stata lasciata totalmente alla mia iniziativa, avevo delle precise linee guida a cui dovevo attenermi.

Utilizza esclusivamente HTML e CSS.

Per il tuo progetto ti chiediamo di creare almeno due pagine. La pagina con il CV deve essere in formato HTML.

Al fine di dimostrare di saperti destreggiare con quanto appreso durante il Corso, ti chiediamo di prevedere, all’interno del tuo sito, le seguenti pagine e/o tecnologie:

- Crea, oltre alle pagine che desideri, anche una pagina CV, rigorosamente in HTML.
- Crea, oltre alle pagine che desideri e alla pagina CV, una pagina “contattami” che contenga un form.
- Opzionalmente, se desideri, puoi utilizzare servizi come EmailJS per inviare il contenuto del form alla tua email.
- Un framework front end (Bootstrap, Materialize, ecc.).
- Una favicon.
- Un menu sticky, almeno per la modalità mobile.
- Crea almeno una pagina, tra quelle proposte o quelle previste dai requisiti, che implementi il layout tramite FlexBox o Grid System (o, perché no, tutti e due).
- Utilizza Sass o Less per i tuoi CSS.

Ricorda che il tuo sito dovrà essere 100% responsive.

Dopo avere caricato il lavoro su GitHub, **dovrai pubblicarlo online con GitHub Pages** (questo passaggio è obbligatorio). Rendere il tuo sito facilmente fruibile online sarà fondamentale per dimostrare di cosa sei capace a un potenziale datore di lavoro!

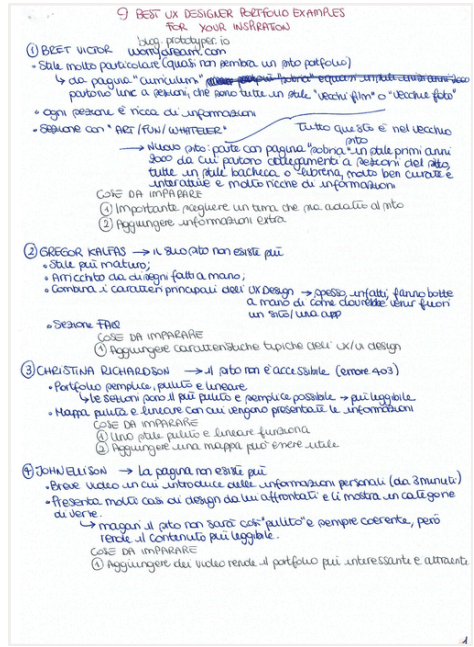
Siamo sicuri che ogni lavoro sarà valido, ma ricorda sempre che per eccellere fra tanti bisogna lasciare un tocco personale in più, un po’ come fanno i grandi artisti.

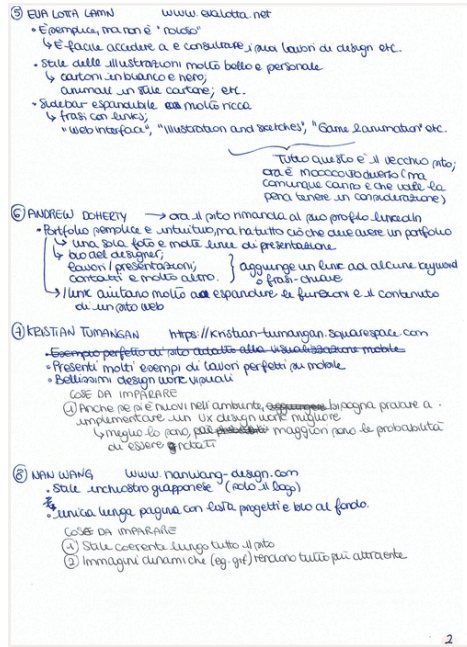
Una volta verificato che il tuo sito funzioni e che abbia qualche dettaglio che ci farà brillare gli occhi, procedi a inviare la presentazione. Testa che tutto funzioni su Github prima di inviare.

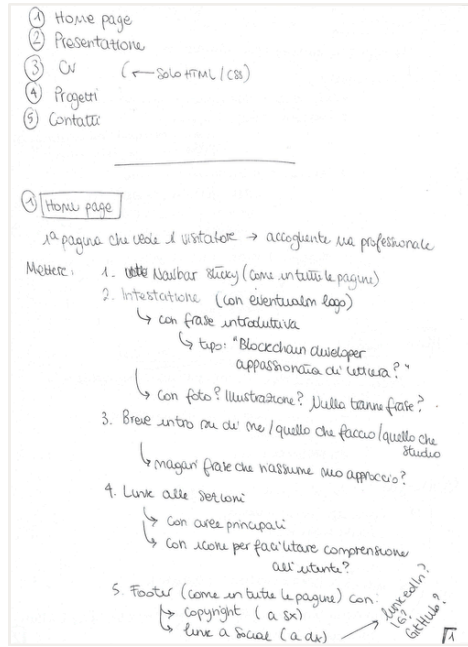
LA PREPARAZIONE

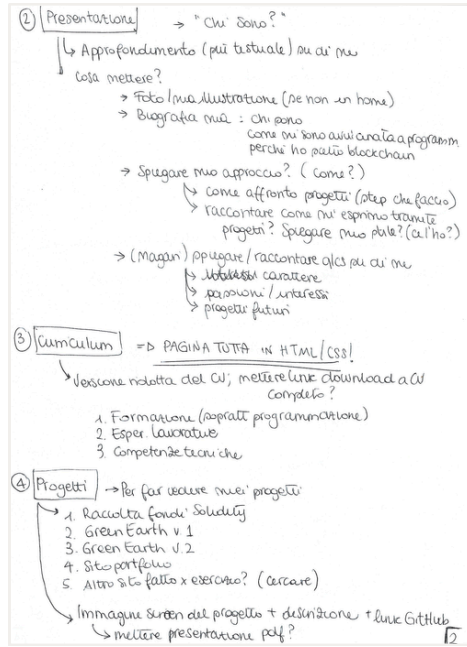
Per prepararmi alla creazione del sito, per prima cosa ho visionato altri siti portfolio (ho usato delle pagine web in cui venivano raccolti i “migliori siti portfolio”) e cercato online dei suggerimenti sull’impostazione da dargli, le pagine che ci sarebbero dovute essere etc. Come ulteriore approfondimento, ho chiesto a un’intelligenza artificiale (nel mio caso: ChatGPT) cosa, secondo lei, ci sarebbe dovuto essere in un sito portfolio che avrei voluto creare; all’AI non ho chiesto solamente quali erano le parti più importanti da inserire, ma le ho anche specificato il mio carattere e i miei interessi in modo che i suggerimenti fossero calibrati sulla mia personalità.

Ho preso appunti durante tutto il processo di ricerca e, quando mi sono ritenuta soddisfatta delle informazioni raccolte, ho creato una prima bozza della struttura che avrei voluto dare al sito.

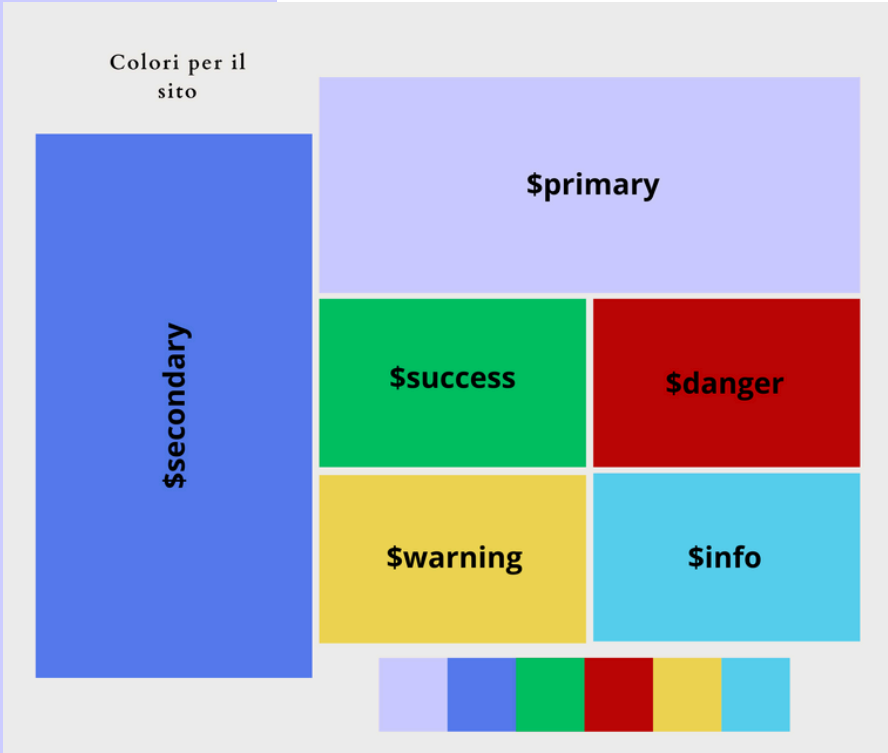








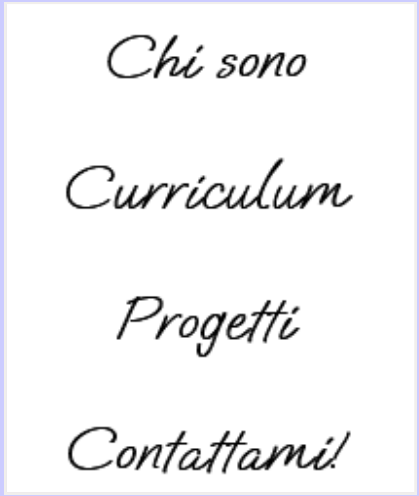
Torna alla home



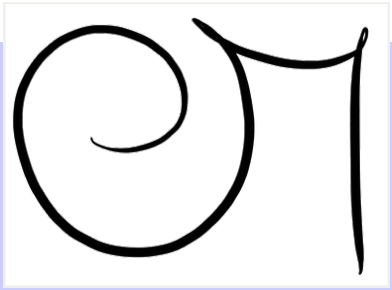
Avevo deciso la struttura, ma ancora non avevo deciso nulla circa la parte grafica. Mi sono, quindi, messa a cercare quali colori potessero adattarsi alla mia personalità.

Il colore principale che ho scelto è il **viola**: questo, infatti, viene considerato da molti sinonimo non solo di creatività, ma anche di magia e di mondi fantastici (cosa che si abbina molto bene alla mia passione per i videogiochi e la lettura); ho deciso di usarlo in una tonalità chiara (#ccccff) perché non fosse troppo preponderante.

Una volta stabilito il colore primario per il sito, ho cercato online delle idee per palette di colori che vi si abbinassero: avevo già deciso di usare Bootstrap come framework, e volevo che tutti i colori ($primary, $secondary, $success, $warning, $danger, $info) stessero bene tra loro. Una volta scelti, ho usato un color picker (https://www.w3schools.com/colors/colors_picker.asp) per selezionare in modo preciso la tonalità che desideravo.

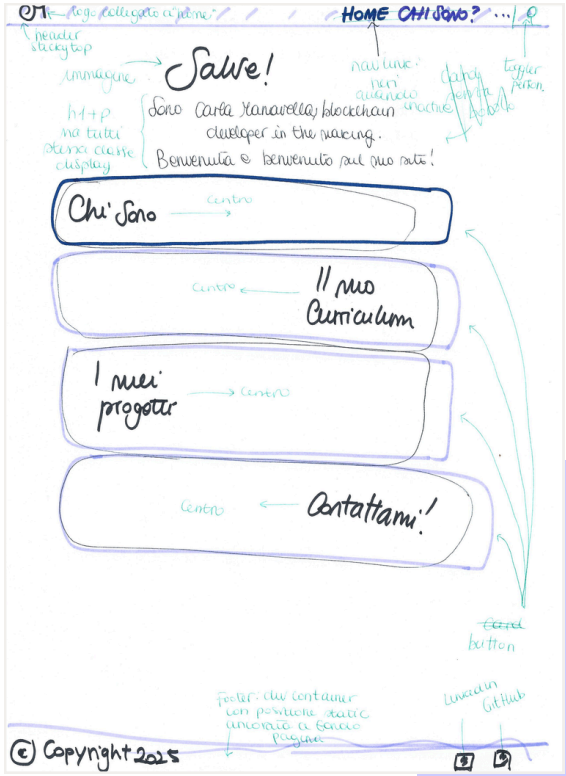


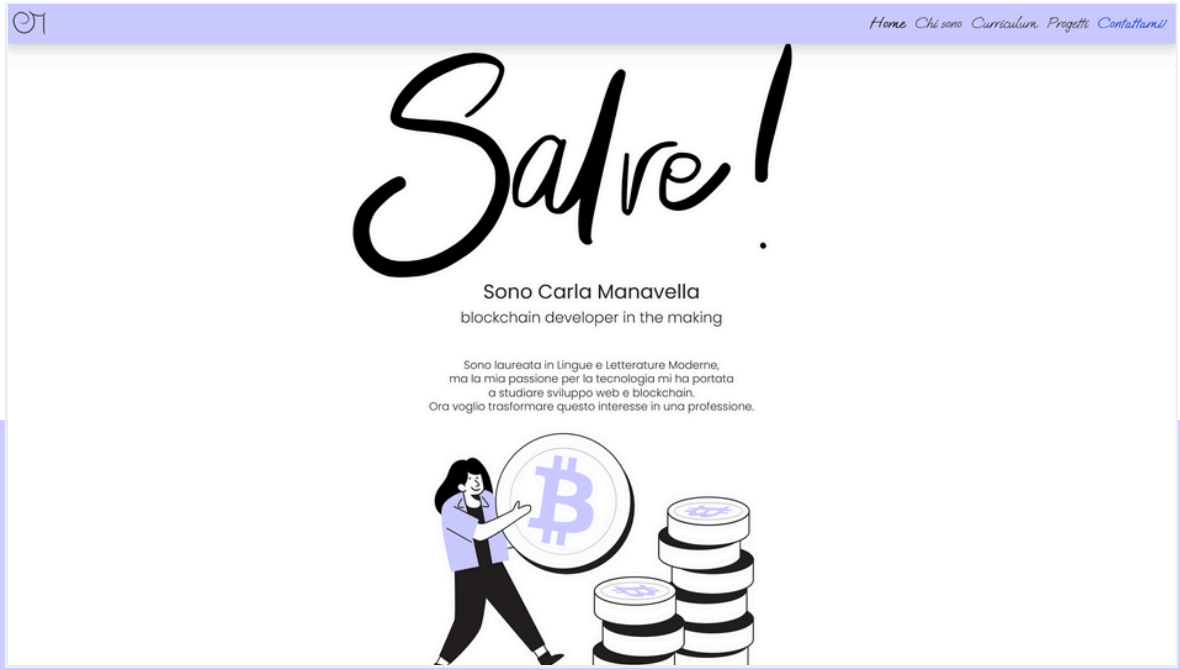
Per rendere il sito più personale, ho deciso di metterci “le mani”: ho “disegnato” il favicon e i titoli per le sezioni con un iPad e una Apple Pen, li ho modificati con Gimp (per rimuovere lo sfondo e tagliare i bordi bianchi) e li ho, poi, esportati in .png e convertiti in .svg (sia per rendere le immagini più leggere, sia per migliorarne la resa a schermo).



Per rendere il tutto più coerente, ho poi cercato su Google Fonts un web font che fosse simile alla mia scrittura. Una volta trovato (“Oooh Baby”), l’ho usato per elementi come i link nella navbar, i link nella home page, il copyright nel footer e i titoli dei progetti nella pagina dedicata.

Infine, con carta, penna e pennarelli ho fatto un mockup di ogni pagina del sito per avere una reference visiva di come avrei voluto che apparissero. In alcuni casi il risultato finale è stato leggermente diverso da come lo avevo progettato, ma mi è servito comunque per avere una guida da cui partire.





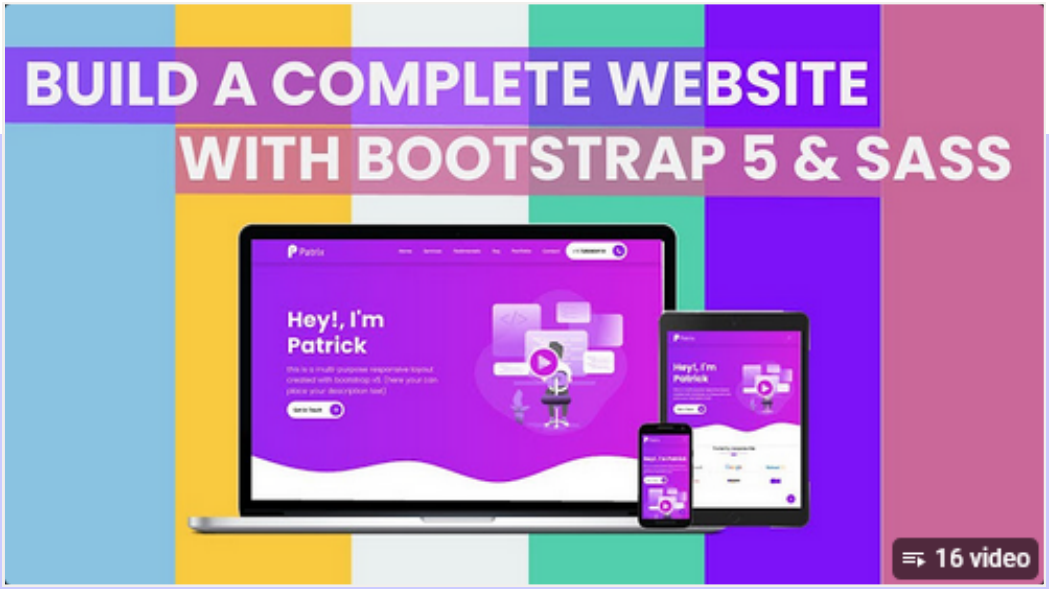
BOOTSTRAP E SASS

Come detto in precedenza, per la creazione del sito avevo delle linee guida da seguire; alcune di queste dicevano che avrei dovuto usare sia un framework frontend, sia un compilatore CSS a scelta tra Sass e Less.

Come framework frontend ho scelto **Bootstrap**, che ho avuto modo di scoprire durante il corso. L'ho scelto non solo per la sua semplicità e intuitività di utilizzo, ma anche perché trovo che questo framework permetta di fare praticamente qualunque cosa.

Come compilatore CSS, invece, ho scelto **Sass** sia per la sua versatilità e semplicità di utilizzo, sia perché è facilmente integrabile con Bootstrap.

Per integrare Sass in Bootstrap, e per modificare le variabili che avrei voluto personalizzare, ho seguito la prima parte del tutorial “Build a complete portfolio website with Bootstap 5 & Sass” del canale “Code With Patrick” di Patrick Muriungi (lo stesso tutorial è stato riproposto anche sul canale “freeCodeCamp.org”).



LA NAVBAR E IL FOOTER

La navbar è stata la prima sezione che ho inserito nel sito. Per crearla, mi sono affidata ad alcuni dei components che si trovano nella documentazione di Bootstrap:

- ho usato lo snippet di “Navbar > Image” per dargli la struttura di navbar orizzontale in cui, a sinistra, il brand era rappresentato dal suo logo (a cui ho aggiunto il link alla pagina *index.html*);
- ho inserito i link alle pagine del sito (con la classe *nav-link*) all’interno di un *div*;
- ho poi preso spunto dallo snippet nella sezione “Navs & Tabs > Right aligned” per allineare i link a sinistra (con *justify-content-end*);
- alla navbar ho anche aggiunto la classe Bootstrap *sticky-top* per renderla, appunto, sticky rispetto alla parte alta dello schermo.

La navbar dello snippet prevede anche un bottone (*button*) con una classe Bootstrap chiamata *navbar-toggler*: questa fa sì che la navbar si comprima quando lo schermo raggiunge un determinato breakpoint. Ho personalizzato il pulsante scegliendo (tra le icone proposte da Bootstrap al sito https://icons.getbootstrap.com/) quello che rappresenta un libro, inserendolo nel codice come web font.

Con Sass, poi, ho tolto all’icona il contorno e l’ombreggiatura (che Bootstap applica in automatico) perfezionando la classe *.navbar-toggle* all’interno del partial *_navbar.scss*. Sempre nello stesso partial, ho personalizzato il colore dei *nav-link* (con le prorietà *color*, *font-size*, *font-weight*) e il loro comportamento quando su questi link passa sopra il cursore o quando ci si clicca sopra.

Ho poi usato la classe *.active* per segnalare e personalizzare il link nella navbar quando ci si trova su quella pagina; ad ogni link *active* ho anche aggiunto *aria-current=“page”* per segnalarne lo stato attivo alle tecnologie assistive.

La creazione del footer è stata meno immediata. Per far comparire il footer sempre a fondo pagina, a prescindere dalla quantità del contenuto, ho fatto sì che *body* e *html* occupassero sempre almeno il 100% dell'altezza visibile (*min-height: 100vh*) a partire dai breakpoint *md*; ho poi creato un *main* (per segnalare che quello è il contenuto principale della pagina) a cui ho assegnato una classe *page*, che ho poi richiamato in un partial Sass per fare in modo che occupasse il 90% dell'area visibile tra i breakpoint *md* e *lg* (*min-height: 90vh*) e l'85% a partire dal breakpoint *xl* (*min-height: 85vh*). Tutte queste regole si trovano nel partial *_custom.scss*.

Per la parte grafica, invece, ho scelto di adottare il secondo degli esempi riportati da Bootstrap nella pagina https://getbootstrap.com/docs/5.3/examples/footers/:

- si tratta di un flexbox (*d-flex* nell'HTML) il cui contenuto è "wrappato" (*flex-wrap*) per fare in modo che non esca mai dalla linea e allineato al centro (*align-content-center*), con un margine superiore di 5, margine inferiore di 0 e classe *footer;*
- il "copyright" (*footer__copy*) e i link ai social (LinkedIn e GitHub) sono contenuti in due *div* diversi (per mantenere le due icone social raggruppate).

Lo stile del footer e dei suoi elementi è stato personalizzato nel partial *_footer.scss*:

- gli ho dato un *border-top* (del colore primario del sito) per creare solamente la linea superiore;
- a *footer__copy* ho dato la font-family "Oooh Baby" (la stessa della navbar, ad esempio), una dimensione del 130% maggiore rispetto al valore standard, un *font-weight: bold* e il colore *$black* di Bootstrap;
- alle icone ho assegnato un font responsivo (1.5rem), un *font-weight: bolder* per farle risaltare di più, il colore *$black* di Bootstrap (reso più chiaro con la proprietà *lighten* per renderli più simili al colore della scritta) e uno stile hoover con il colore primario ($primary) più scuro di 10;
- infine, per fare in modo che non occupase l'intera larghezza della pagina, ho inserito tutto quanto in un div.

GLI HEADING

Gli heading sono stati una delle parti più complicate da rendere correttamente. Come detto poco fa, per rendere il sito più personale ho deciso di creare i titoli delle pagine con la mia scrittura. Questo, però, mi ha posta di fronte ad alcuni problemi:

- le scritte non avevano tutte la stessa dimensione;
- le scritte non occupavano tutte lo stesso spazio nella pagina;
- la loro visualizzazione non era consistente tra le pagine del sito.

Ho risolto questi problemi con delle media queries specifiche per ogni heading, che ho inserito nel partial *_typography.scss*:

- nel codice HTML di ogni pagina, a ogni immagine ho dato una classe principale "*.title*";
- per ogni singola immagine, alla classe ho aggiunto un modifier col nome della pagina per poter modificare ogni heading a seconda delle necessità
  - nel partial, quindi, ci sarà una classe .title seguita dai modifier che richiamano l'immagine nelle loro pagine ("*&--about*", "*&--projects*", "*&--contacts*");
  - nel partial non è incluso l'heading per la pagina "cv" (questa, infatti, viene gestita dal relativo CSS).

Ho scelto le media queries durante la fase di ottimizzazione del sito, aiutandomi con il DevTool di Chrome.

Siccome volevo che il titolo principale delle pagine del sito fosse l'immagine con la mia scritta, senza però penalizzarne la struttura gerarchica, ho:

- inserito un *h1* in ogni pagina a cui ho assegnato la classe *page-heading*;
- richiamato la classe *page-heading* nel partial Sass *_typography.scss* e l'ho "reso invisibile" con la proprietà *visibility* (*visbility: hidden*);
- aggiunto *aria-label="heading"* alle immagini per facilitarne l'identificazione agli screen reader.

LA HOMEPAGE (INDEX)

Nella homepage (*index.html*) si possono distinguere 4 blocchi:

- il primo blocco è quello del **saluto all'utente**, che contiene un'immagine ("Salve!"), un sottotitolo con il mio nome e una catchphrase, una prima introduzione su di me (che si potrà poi approfondire nella pagina "Chi sono") e una grafica che mi rappresenta in cui si vede una giovane donna che impila Bitcoin (per ricollegarsi al concetto di blockchain development);

- nel secondo blocco si trova un **elenco delle mie competenze**, sia quelle già acquisite che quelle "in studio" (in linea con il concetto "in the making" della catchphrase);

- il terzo blocco è composto da un carosello che presenta un'**anteprima dei miei progetti migliori** (da aggiornare col tempo) dove
  - le prime tre immagini (i progetti) hanno un link che porta alle rispettive pagine su GitHub;
  - l'ultima immagine rimanda alla pagina "Progetti" del sito;

- il quarto blocco è semplicemente un **link alla pagina "Contattami!"** del sito in modo che, se lo si desidera, non si debba andarlo a cercare tra i link della navbar.

I blocchi sono separati tra loro da una piccola linea di separazione (classe *separation-line*) descritta nel partial Sass *_index.scss*.

Il primo blocco è formato da un *container* con classe "*hello*" che contiene:

- l'immagine con il "saluto" ;
- il mio nome e la catchphrase;
- una breve descrizione di chi sono;
- la rappresentazione di una giovane donna che impila Bitcoin.

→ L'immagine di saluto è inserita in un altro *div* (con classe *hello__imagebox*) in modo da renderla responsiva usando i breakpoint forniti da Bootstrap; l'immagine, poi, viene mostrata come blocco (*d-block*) e ha una sua classe child di *hello* (*hello__image*) che viene richiamata nel partial Sass *_index.scss* per fare in modo che occupi il 95% del suo div;

→ Il mio nome e cognome, che è anche il titolo principale della pagina (*h1*), è inserito in un successivo div; all'*h1* ho attribuito una classe child di *hello* (*hello__name*) per sistemarne lo stile nel partial Sass *_index.scss*;

→ La catchphrase è inserita nello stesso div del mio nome e cognome ma, in questo caso, si tratta di un paragrafo (*p*) a cui ho assegnato la classe *hello__catchphrase* per poterne sistemare lo stile nel partial Sass *_index.scss*;

→ Il div successivo è quello che contiene un breve riassunto della mia storia: ha un titolo *h2* nascosto (*style="display: none;"*), in modo che non sia visibile a schermo, seguito da 4 paragrafi (*p*) con classe *hello__myStory*, resi responsivi con alcune classi di Bootstrap e con uno stile attribuito loro nel partial Sass *_index.scss*;

→ Infine, l'ultimo div è quello chiamato *hello__picture*, che contiene la grafica della giovane donna che impila Bitcoin (resa responsive dalla classe Bootstrap "*img-fluid*"). L'immagine viene dalla selezione gratuita di "Manypixels" (https://www.manypixels.co/gallery), in cui ho potuto impostare il colore primario del sito come tonalità principale dell'illustrazione.

Il secondo blocco contiene un elenco delle mie competenze (presenti e future):

→ il contenuto del blocco è inserito in una *row* (reso un flexbox con *d-flex*) e centrato con le classi di Bootrstrap *mx-auto* e *justify-content-center*; alla *row* ho assegnato una classe *mainHomeContent*;

→ nella row è contenuta una *column* (resa responsive con i breakpoint di Bootrstrap) con classe child *mainHomeContent__skills*; qui c'è anche il titolo della sezione: un *h2* con classe *mainHomeContent__title-section* (che ho richiamato nel partial Sass *_index.scss* per assegnargli uno stile);

→ per fare in modo che la larghezza dell'elenco non superasse mai la dimensione del titolo della sezione, ho creato una nuova *row* (con classe *text-center* per centrare il testo) seguito da una *column* resa responsive con i breakpoint di Bootstrap; nella *column*, poi, ho inserito una serie di paragrafi *p* preceduti da una icona Bootstrap a forma di rombo (con cui volevo richiamare la forma del simbolo dell'Ether, per rimandare di nuovo alla blockchain).

Il terzo blocco, invece, contiene un carosello per presentare un'anteprima dei progetti a cui ho lavorato:

→ il contenuto del blocco è inserito in un *div container* il cui primo elemento è un *h2* con il titolo della sezione (a cui ho assegnato la classe *mainHomeContent__title-section* per renderlo uguale al precedente);

→ al titolo segue, poi, il carosello, che ho preso direttamente dalla documentazione di Bootstrap (https://getbootstrap.com/docs/5.3/components/carousel/) e ho adattato perché rispondesse meglio alla mia idea; nello specifico:

- nel partial *_index.scss* ho fatto in modo che la classe *.carousel-info* (quella, cioè, che contiene la didascalia dell'immagine) avesse un colore di sfondo (*background-color: rgba($primary, 0.85);*) che coprisse l'interezza dell'immagine; le ho, poi, dato la proprietà *visibility: hidden;* per fare in modo che questo colore, normalmente, non fosse visibile;

- ho poi fatto in modo che colore di sfondo e didascialia ricoprissero l'immagine quando ci si passa sopra con il cursore con la pseudoclasse *hover* (*&:hover {.carousel-info {opacity: 2; visibility: visibile;* etc.);

- nel partial *_custom.scss*, invece, ho richiamato le variabili dei controlli (le frecce) e degli indicatori del carosello per poterne cambiare i colori e l'opacità, migliorandone la visibilità; a questi ho assegnato il colore $purple di Bootstrap.

Infine, il quarto blocco è composto semplicemente dalla scritta “Contattami!”: un link alla medesima pagina del sito.

→ La scritta è contenuta in un *div container* (il cui contenuto viene centrato con la classe Bootstrap *text-center*) a cui ho attribuito la classe *linkToContact* per poterne gestire lo stile nel partial Sass *_index.scss*;

→ nel div è contenuto un *h2* (per metterlo sullo stesso piano dei precedenti titoli delle sezioni), centrato con la classe Bootstrap *mx-auto*;

→ all’interno dell’h2 è contenuto il link (*a*) che porta alla pagina “Contattami!” del sito (*contacts.html*); la scritta è preceduta da un’icona Bootstrap che rappresenta un aeroplano di carta.

Nel partial *_index.html*, poi, ho gestito come questa scritta appare a schermo:

→ ho dato all’intero container dei margini (automatici a destra e sinistra, di 10vh a quello superiore e inferiore per far sì che questi si adattino allo schermo di ogni dispositivo);

→ ho dato alla scritta del link (*a*) il font family “Oooh Baby” perché sia uguale a quello delle altre scritte del sito; questa scritta ha anche una dimensione di 5rem, un “peso” *bold*, il colore nero di Bootstrap (*$black*) e ho tolto la sottolineatura (*text-decoration: none*);

→ ho anche dato uno stile all’icona, assegnandole il colore primario del sito (*color: $primary*) e cambiandole la dimensione (*font-size: 4rem*) e il “peso” (*font-weight: bold*);

→ infine, alla scritta del link e all’icona ho assegnato una media query per fare in modo che queste diventino più piccole quando la scritta viene visualizzata su schermi minori del breakpoint “md” di Bootstrap.

LA PAGINA “CHI SONO”

La pagina “Chi sono” (*about.html*) volevo che venisse visualizzata in modo diverso a seconda del dispositivo da cui si sta guardando:

- immagine in alto e testo sotto (con giustificazione centrale) quando il dispositivo ha uno schermo “piccolo”;
- immagine a sinistra e testo a destra (con giustificazione a sinistra) quando il dispositivo è più grande.

Per farlo, prima di tutto ho creato un div con una classe parent *about*; al suo interno ho inserito una *row* il cui contenuto è mostrato come un flexbox (*d-flex*) con un allineamento dei contenuti al centro (*align-items-center*) e un margine inferiore di 5 (*mb-5*, una classe di Bootstrap). Qui ho messo il contenuto della pagina:

- il primo elemento che si incontra è un div con le classi di Bootstrap *col-12* e *col-lg-6* che ne determinano la dimensione; al suo interno ho inserito la prima versione dell’immagine della pagina (l’immagine, che è fluida (*img-fluid*) per renderla responsiva, è un child di about (*about__image-smallScreen*));
- l’elemento successivo è un altro div, anch’esso child di *about* (*about__text*) al cui interno si trovano un’altra immagine (fluida (*img-fluid*) e child di *about* (*about__image-bigScreen*)) e una serie di paragrafi (*p*),

Il testo e le immagini sono gestite nel partial *_about.scss*:

- il testo (*&__text*) viene trasformato con una media query che dice che questo elemento deve essere mostrato con un allineamento giustificato al centro (*text-align: center*) e con un padding sui lati di 2 rem (*padding: 0 2rem*) fino al breakpoint *lg* escluso (a partire da lg, l’allineamento diventa quello di default che, appunto, è quello con la giustificazione a sinistra);

- viene detto al sito di mostrare un’immagine piuttosto che l’altra usando delle media queries e la proprietà *display*:
  - *&__image-smallScreen* non viene mostrata (*display: none*) fino al breakpoint *md* (@*include media_breakpoint_down(lg)*), mentre per il solo breakpoint md (*@include media_breakpoint_only(md)*) l’immagine deve essere alta almeno il 40% dell’altezza visibile (*height: 40vh*);
  - *&__image-bigScreen* viene mostrata a partire dal breakpoint lg (*@include media_breakpoint_up(lg)*).

Anche l’immagine di questa pagina viene dalla selezione gratuita di “Manypixels” https://www.manypixels.co/gallery().

LA PAGINA "CURRICULUM"

Come da consegna, il sito doveva avere una pagina "Curriculum" che fosse costruita solamente con HTML: per questo, ad eccezione della navbar e del footer, in questa pagina non viene usato nulla che abbia a che fare con Bootstrap. Io, poi, ho deciso di non usare Sass per gestire lo stile della pagina.

Anche in questa pagina, l'heading è un'immagine la cui dimensione è gestita con delle media queries che ne alterano la grandezza a seconda di quella dello schermo su cui il sito viene visualizzata (queste si trovano nello stylesheet *cv.css*).

Il contenuto principale della pagina è una tabella (un grid): è inserito dentro un container (con classe *cv-grid*) e ogni elemento della tabella è composto da

- un div con classe *cv-entry* che, a sua volta, contiene due testi (*p*) con classi diverse (*cv-date* per la data e *cv-position* per la descrizione della posizione);
- un div con classe *cv-descript* e, al suo interno, un testo *p* per la descrizione dell'occupazione svolta.

Come detto, gli elementi di questa pagina sono inseriti all’interno di un div chiamato *cv-grid* che, all’interno del file *cv.css*, viene “trasformato” in una tabella (*display: grid*) composta da:

- due colonne (la prima grande il 25% del totale, la seconda grande il 60%) e delimitate da tre “linee” chiamate *first*, *half* e *last* (*grid-template-column: [first] 25% [half] 60% [last]*);
- un numero automatico di righe, che si aggiorna a seconda dei “*cv-entry*” che ci sono sull’HTML e la cui dimensione dipende dal contenuto che contengono (*grid-template-row: repeat(auto, minmax(auto))*);
- un gap del 3% della larghezza visibile (*column-gap: 3vw*) tra le colonne e del 4% dell’altezza visibile (*row-gap: 4vh*) tra le righe, una giustificazione al centro per il contenuto (*justify-content: center*) e un margine superiore del 5% dell’altezza visibile (*margin-top: 5vh*).

La distribuzione del contenuto nella tabella, invece, è affidato alle classi *-cventry* e *-cvdescript*:

- il contenuto di *-cventry* (la data di inizio e la descrizione della posizione) va nella colonna che inizia con la linea *first* (*grid-column: [first]*) e si allinea verticalmente al centro della cella (*align-self: center*) mentre, orizzontalmente, si sposta verso la fine (*justify-self: end* e *text-align: end*);
- il contenuto di *-cvdescript* (la descrizione del lavoro svolto) si inserisce nella colonna che inizia con la riga *half* (*grid-column: [half]*) e si allinea verticalmente al centro del cella (*align-self: center*), mentre il testo si giustifica verso l’inizio della stessa (*text-align: start*).

Questa struttura è pensata in ottica di manutenibilità: permette, infatti, di non dover alterare né la dimensione, né il numero o la struttura delle celle della tabella, ma di aggiornarla semplicemente inserendo il contenuto desiderato nell’HTML (facendo attenzione, ovviamente, a usare correttamente le classi *-cventry* e *-cvdescript*).

LA PAGINA "PROGETTI"

La pagina dei progetti (*projects.html*) è pensata come una vetrina-portfolio per presentare i progetti su cui ho lavorato. Al momento della creazione del sito i progetti sono solo quelli creati come progetti per il corso di Start2Impact, ma l'idea è quella di tenere aggiornata la pagina con quelli su cui lavorerò in futuro. La pagina si presenta come un elenco di progetti (composto da un'immagine, una breve descrizione e 1 o 2 icon-link) che appaiono in modo diverso a seconda dello schermo su cui vengono visualizzati:

- sugli schermi "piccoli" come una colonna, con l'immagine in alto e la descrizione sotto di essa;
- sugli schermi più grandi, invece, come un elenco alternato di immagine-descrizione e descrizione-immagine.

Per ottenere questo risultato, ho usato dei Flexbox:

- ogni progetto "dispari" è contenuto in un div con classe *container* e allineamento al centro (*text-center*);
- il primo elemento nel *container* è una "riga" (classe *row* di Bootstrap), mostrata come un flexbox in linea (*d-flex flex-row*) in cui gli elementi sono allineati verticalmente al centro (*align-items-center*) e il contenuto è giustificato orizzontalmetne al centro (*justify-content-center*);

- ogni progetto "pari" è contenuto in un div con classe *container* e allineamento al centro (*text-center*);
- il primo elemento nel container è una "riga" (classe *row* di Bootstrap), mostrata come un flexbox in linea (*d-flex flex-row*) in cui gli elementi sono allineati verticalmente al centro (*align-items-center*) e il contenuto è giustificato orizzontalmetne al centro (*justify-content-center*);
- quando lo schermo diventa più grande del breakpoint *sm*, il contenuto viene mostrato in ordine inverso rispetto a com'è nell'HTML (*flex-sm-row-reverse*), invertendone quindi l'ordine rispetto a quello dei progetti "pari".

Il resto, invece, è uguale in tutti i progetti:

- un div (con una dimensione diversa a seconda del breakpoint e giustificato al centro con *justify-content-center*) che contiene un’immagine responsiva (*img-fluid*) che rappresenta il progetto;

- un div (anch’esso con una dimensione diversa a seconda del breakpoint e giustificato al centro con *justify-content-center*) che, a sua volta, contiene
  - un *h5* per il titolo del progetto (*project__head*) e un *p* per la sua descrizione (*project__description*);
  - un altro *container* con una riga, mostrata come un flex con direzione orizzontale (*row d-flex flex-row*), in cui gli elementi sono distribuiti equamente nello spazio a disposizione (*justify-items-evenly*), che contiene due icon-link che portano alla pagina GitHub del progetto e, dove disponibile, a una presentazione pdf.

LA PAGINA “CONTATTAMI!”

L’ultima pagina del sito è quella dedicata al form per contattarmi. Ho voluto mantenerlo piuttosto semplice, ma volevo che comprendesse comunque alcune informazioni fondamentali (*required*) da sapere qualora venissi contattata:

- il nome completo di chi scrive (da inserire in un *form-control* di tipo testo);
- l’azienda (da inserire in un *form-control* di tipo testo);
- l’indirizzo email (da inserire in un *form-control* di tipo email);
- il messaggio (da inserire in una *textarea*).

Altro componente obbligatorio per la compilazione del form è il checkbox finale con l’autorizzazione al trattamento dei dati personali.

Oltre ai campi “*required*”, ho anche previsto dei campi per:

- la posizione ricercata (da inserire in un *form-control* di tipo testo);
- la data da cui dovrebbe iniziare la mia disponibilità (selezionabile da un campo di tipo calendario).
  ⟶ Questi non devono essere compilati obbligatoriamente per inviare il form (volendo, infatti, le informazioni si potrebbero scrivere in maniera estesa all’interno del messaggio).

Una volta compilato correttamente il form, questo può essere inviato cliccando sul pulsante “Invia” (*button type=”submit”*) che farà partire l’invio (tramite FormSubmit) di una email di notifica a me con il messaggio che mi è stato inviato; per confermare il corretto invio, invece, all’utente si aprirà una pagina di ringraziamento. Oltre a questo, ho anche previsto un pulsante “Ricomincia” per resettare il contenuto del form (*button type=”reset”*) .

La pagina prevede anche uno script che controlla che i campi obbligatori (*required*) del form siano compilati correttamente (in caso contrario, in corrispondenza dell’errore viene mostrato il messaggio che, sull’HTML, è contenuto nel div con classe *invalid-feedback*); la libreria a cui fa riverimento si trova al fondo della pagina (prima del tag di chiusura del *body*), mentre lo script è nel file *assets* > *checkRequired.js*.

LA PAGINA “CONTATTAMI!”

L’ultima pagina del sito è quella dedicata al form per contattarmi. Ho voluto mantenerlo piuttosto semplice, ma volevo che comprendesse comunque alcune informazioni fondamentali (*required*) da sapere qualora venissi contattata:

- il nome completo di chi scrive (da inserire in un *form-control* di tipo testo);
- l’azienda (da inserire in un *form-control* di tipo testo);
- l’indirizzo email (da inserire in un *form-control* di tipo email);
- il messaggio (da inserire in una *textarea*).

Altro componente obbligatorio per la compilazione del form è il checkbox finale con l’autorizzazione al trattamento dei dati personali.

Oltre ai campi “*required*”, ho anche previsto dei campi per:

- la posizione ricercata (da inserire in un *form-control* di tipo testo);
- la data da cui dovrebbe iniziare la mia disponibilità (selezionabile da un campo di tipo calendario).
  ⟶ Questi non devono essere compilati obbligatoriamente per inviare il form (volendo, infatti, le informazioni si potrebbero scrivere in maniera estesa all’interno del messaggio).

Una volta compilato correttamente il form, questo può essere inviato cliccando sul pulsante “Invia” (*button type=”submit”*) che farà partire l’invio (tramite EmailJS) sia di una email al mittente di corretta ricezione, sia di una email di notifica a me con il messaggio che mi è stato inviato. Oltre a questo, ho anche previsto un pulsante “Ricomincia” per resettare il contenuto del form (*button type=”reset”*) .

La pagina prevede anche due script:

- il primo (di Bootstrap) controlla che i campi obbligatori (*required*) del form siano compilati correttamente (in caso contrario, in corrispondenza dell’errore viene mostrato il messaggio che, sull’HTML, è contenuto nel div con classe *invalid-feedback*);
- il secondo (di EmailJS) triggera l’invio delle email quando il form è compilato correttamente.

Le librerie a cui fanno riferimento gli script sono inserite al fondo della pagina (prima del tag di chiusura del *body*), mentre gli script veri e propri si trovano nel file *assets* > *scripts.js*.

GRAZIE PER L'ATTENZIONE!