# Relazione per il Tirocinio A.A 2018/2019

*Data visualization e interazione con dati*

Tutor: Mirri Silvia ( [silvia.mirri@unibo.it](mailto:silvia.mirri@unibo.it) )  
Studente: Guri Sokol ( [sokol.guri@studio.unibo.it](mailto:sokol.guri@studio.unibo.it) )

*Università degli studi di Bologna  
Campus di Cesena  
Facoltà di Ingegneria e Scienze Informatiche*



Tabella dei contenuti

1. **Analisi dei requisiti**  
   1.1 Tema del tirocinio  
   1.2 Obbiettivi  
   1.3 Attività previste
2. **Progettazione Concettuale e Logica**2.1 Schemi concettuali e idea iniziale  
    2.1.1 Idea iniziale

2.1.2 Mock Up  
2.2 Fase di sviluppo

2.2.1 Implementazione *Home Page*

2.2.2 Implementazione *Galleria*

2.3 Tecnologie usate

1. **Mia esperienza**

# Capitolo 1 – Analisi dei requisiti

1.1 Tema del tirocinio

L’idea che finalmente potevo applicare le mie conoscenze in modo pratico mi ha entusiasmato tantissimo e mi sentivo pronto per questa nuova esperienza. Non avevo dubbi che il tutor per me doveva essere professoressa Mirri. Mi sono state proposte tantissimi temi però ce n’era uno che mi ispirava di più. Il tema consisteva nel creare un’applicazione gestionale per poter salvare le lauree Honoris Causa dell’Unibo. Dovevo creare il prototipo di un sito web nel quale verranno salvate queste lauree.

1.2 Obbiettivi

Il software sarà in grado di gestire e memorizzare in generale ogni laurea Honoris Causa e dare all’utente la possibilità di interagire con i dati. Ovviamente l’utente può filtrare le informazioni tramite diversi filtri: *laura/struttura/periodo della laurea.* Un’altra funzionalità è la possibilità di cercare i dati inserendo: *nome/cognome/struttura*.   
 L’obbiettivo principale era di progettare meccanismi di data visualization volti a consentire all’utente l’interazione e la navigazione di dati (aggregati e non) con dimensione temporale e geografica, strutturati in varie categorie e classificazioni.   
 La cura per il design è un elemento imprescindibile in qualsiasi progetto web, ma non si deve mai dimenticare il pubblico che usufruirà del sito. L’obbiettivo era quello di partire dalla creazione di un design mobile friendly e responsive, per rendere perfettamente fruibile il sito web sui dispositivi mobile e quindi a tutti i visitatori in mobilità, per poi passare alla versione desktop.   
 Acquisire conoscenze e competenze nell’ambito della data visualization, della Human Computer Interaction e nel contesto delle applicazioni Web e mobile era lo scopo principale di questo tirocinio.

1.3 Attività previste

Le attività previste erano studio e acquisizione di conoscenze relative agli elementi alla base di: data visualization, human computer interaction e tecnologie per lo sviluppo di applicazioni Web e applicazioni mobile. Inoltre, si prevedevano attività di design partecipativo e basate sul coinvolgimento di target user per la definizione dei meccanismi di interazione con i dati. Infine, il design e lo sviluppo di un sistema prototipal

# Capitolo 2 – Progettazione Concettuale e Logica

2.1 Schemi concettuali e progettazione iniziale

2.1.1 Idea Iniziale

Dai consigli del tutor avevo creato un’idea di come il mio sito web doveva essere organizzato e quale doveva essere la sua struttura. Praticamente avevo pensato dall’inizio di avere una pagina di ***welcome*** dove l’utente poteva vedere informazioni sulle laure Honoris Causa ed il tipico footer di una pagina. Le informazioni principali dovevano essere salvate nella pagina della ***galleria***. L’utente poteva visitare questa pagina cliccandola dal navbar.

***2.1.1 Mock Up***

Il wireframe è sostanzialmente la prima bozza del sito e serve a definire l’organizzazione dei contenuti e le funzionalità delle singole pagine web. Questo scheletro è composto da blocchi, tratti e didascalie, con lo scopo di descrivere e tenere traccia della disposizione dei vari elementi all’interno del layout, senza disegni, immagini o colori. Esso è utile sia al webdesigner che al cliente perché permette di definire a grandissime linee il layout e il risultato finale del sito web.   
 Io ho usato *Balsamiq* perché è un prodotto tutto italiano di spopolato successo a livello mondiale: un software per sviluppatori, designer, progettisti e creativi, sviluppato per creare wireframe, ovvero design di interfacce utente.

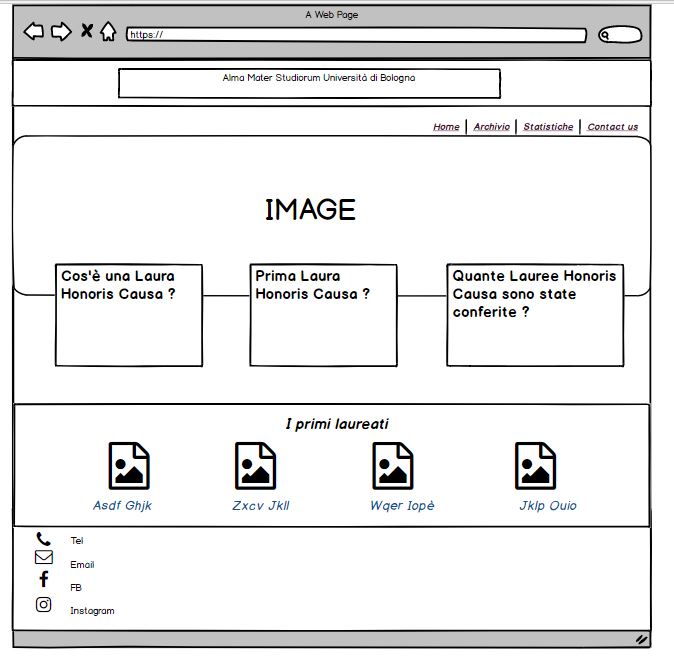


Figure 1. Mock-up "Home Page"

Idea iniziale del “*Home Page*”:   
 1- in alto, doveva essere il logo dell' Unibo, presente in tutte le pagine.   
 2- sotto il logo, un navbar con diverse sezioni *Home, Archivio, Statistiche*   
 3- nel centro una foto (presa dal sito del “Archivio Unibo”)   
 4- sotto questa foto, tre contenitori rettangolari con 3 domande con le rispettive   
 5- sotto questi tre contenitori, le foto dei primi 4 laureati   
 6- il footer conterrà numero tel., e-mail, indirizzo e tutti i social del Unibo

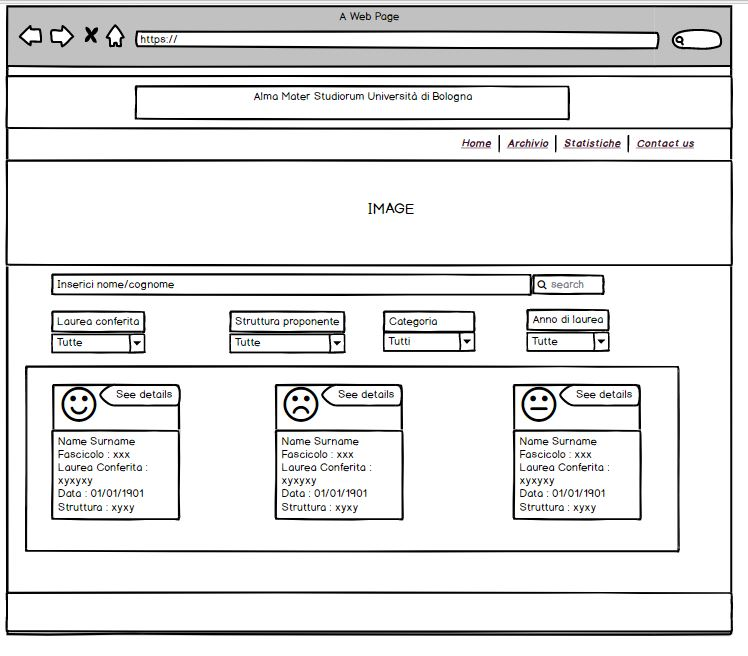


Figure 2. Mock-up "Galleria"

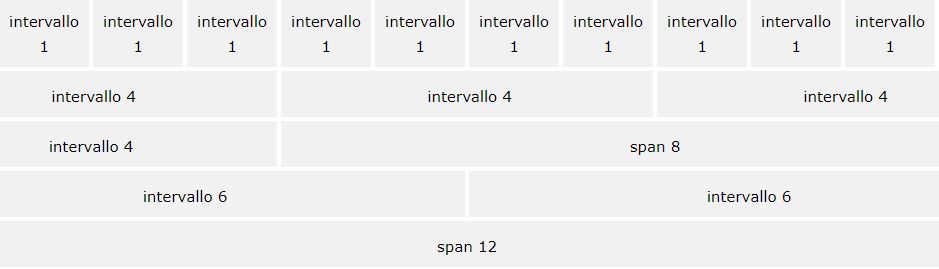
Idea iniziale della pagina “*Galleria*”:   
 1- in alto logo e sotto il navbar come nella *Home Page*   
 2- al centro una foto che rappresenta un archivio  
 3- sotto questa foto il contenitore degli studenti   
 4- sotto questa foto, tre contenitori rettangolari con 3 domande con le rispettive   
 5- doveva essere due sezioni:

* uno per fare la ricerca, tramite il *nome/cognome/laurea* studente
* uno per filtrare le informazioni tramite:

1. Laurea conferita (Agraria, Architettura, ... )
2. Struttura proponente
3. Anno di Laurea (diversi periodi)

6- elenco degli studenti laureati

***2.2 Fase di sviluppo***   
***2.2.1 Implementazione Home Page***

*Bootstrap Grid System*: Questo Sistema di griglia consente fino a 12 colonne nella pagina. Se non si desidera utilizzare tutte e 12 le colonne singolarmente, è possibile raggruppare le colonne per creare colonne più ampie.

Il sistema di griglia di Bootstrap è reattivo e le colonne si riorganizzano a seconda delle dimensioni dello schermo: su un grande schermo potrebbe sembrare migliore con il contenuto organizzato in tre colonne, ma su uno schermo piccolo sarebbe meglio se gli elementi di contenuto fossero impilati uno sopra l'altro.   
 Classi predefinite come .row e .col-sm-n sono disponibili per creare rapidamente layout di griglia. Così usando questa struttura il “Home Page” è strutturato in diversi rows.

Rows:

* Row0: In questa sezione della pagina si trova il logo dell’Università di Bologna e il nome scritto testualmente. Il font family del testo è uguale al font-family ufficiale usato dal sito di Unibo (*Cinzel-Regular*).



* Row1: In questo row si trova il navbar per poter navigare le pagine del sito. Nella parte destra del navbar sono : Home Page, Galleria e Statistiche. Nella parte sinistra c’è scritto Archivio come logo del sito.

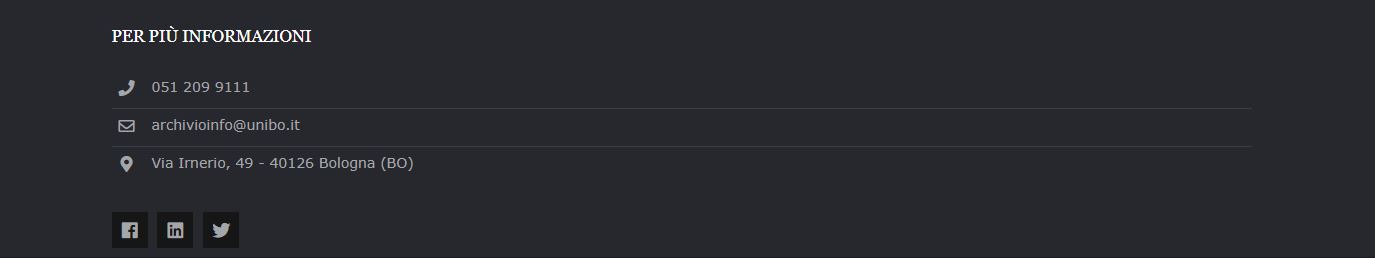


* Row2: I box delle tre domande con le rispettive risposte.



* Row3: Una lista con i primi 4 studenti laureati con una laurea Honoris Causa.

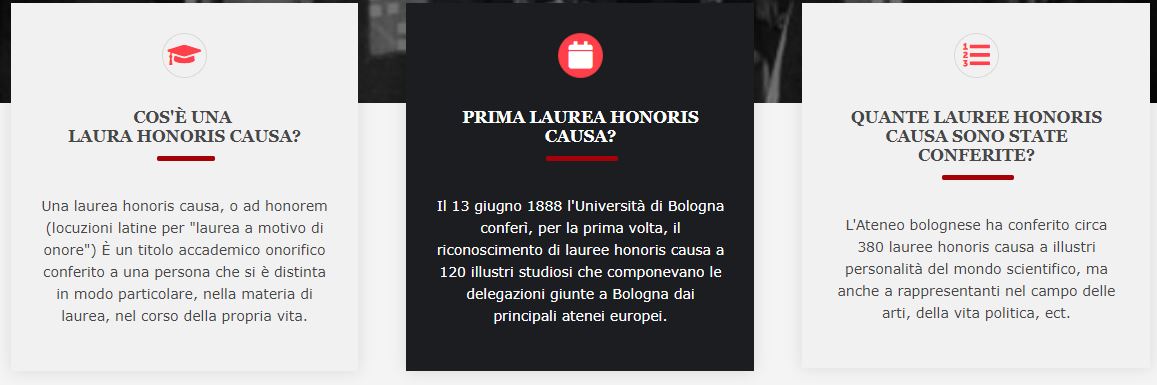


* Row4: Il tipico footer di una pagina web. In questo footer ci sono le informazioni riguardo l’indirizzo, numero di telefono e i social network dell’università.
* Row5: Un prototipo di “Copyright”.



 In questa pagina si trova anche una freccia. Lo scopo di questa freccia è di poter riportare al row0 della pagina. Questa è una funzionalità che semplifica la navigazione dell’utente in questa pagina.

Per migliorare l’esperienza dell’utente durante la navigazione ci sono anche degli effetti wow. Uno di questi è “fade box”. Quando l’utente passa con i mouse sopra uno di questi box, cambia il colore da bianco a nero.

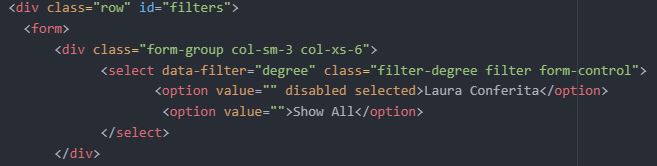


***2.2.2 Implementazione Galleria***

Dopo aver implementato Home Page, il core del nostro sito web era la pagina dell’archivio, ovvero la pagina della Galleria. Anche per la pagina della Galleria ho usato la stessa strategia usata nella prima pagina. L’organizzazione in rows era efficiente e molto compatto.   
 Row0 e row1 sono uguali alla prima pagina, dove si trova il logo e navbar. Nel row2 c’è una foto e in questa foto nella parte sinistra si vede il path della navigazione.   
 La parte più importante è la sezione del container, che contiene le sezioni di search, dei filtri e i le view-card di ogni studente.

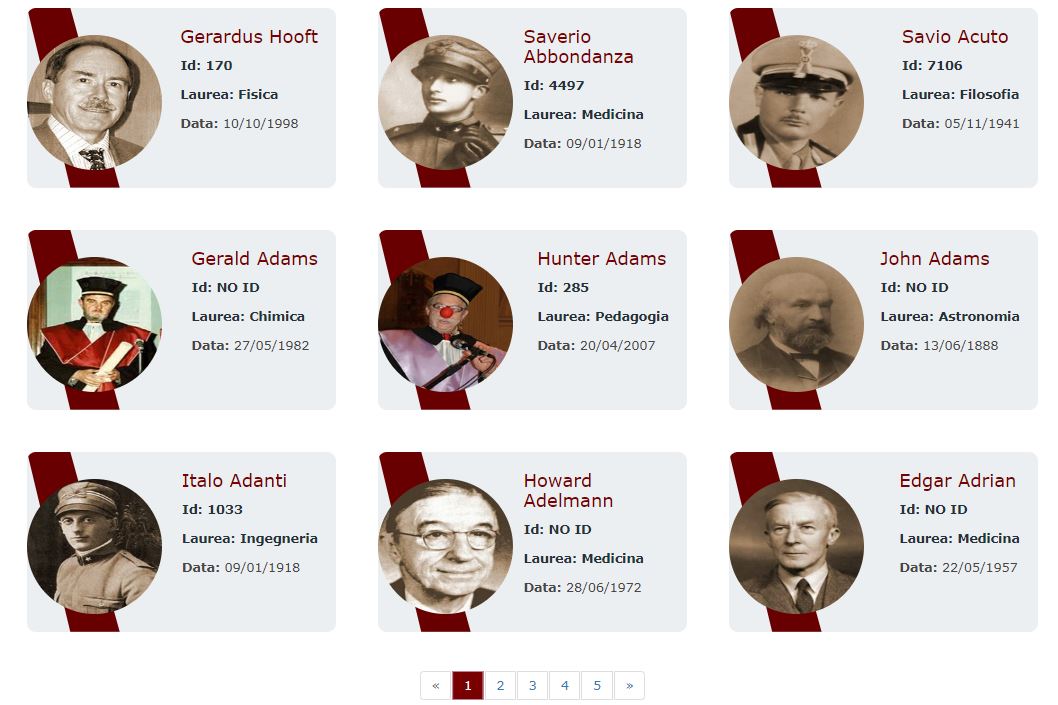
* Button-search: Un componente con tag <button> e definisce un pulsante cliccabile. All'interno di un elemento <button> puoi inserire contenuti, come testo o immagini. Questa è la differenza tra questo elemento e i pulsanti creati con l'elemento <input>. Il tag <input> specifica un campo di input in cui l'utente può inserire i dati. Gli elementi <input> vengono utilizzati all'interno di un elemento <form> per dichiarare controlli di input che consentono agli utenti di immettere dati.
* Filters: Componenti con tag <select> e crea un elenco di opzioni a dropdown da una finestrina. I tag <option> all'interno dell'elemento <select> definiscono le opzioni disponibili nell'elenco.





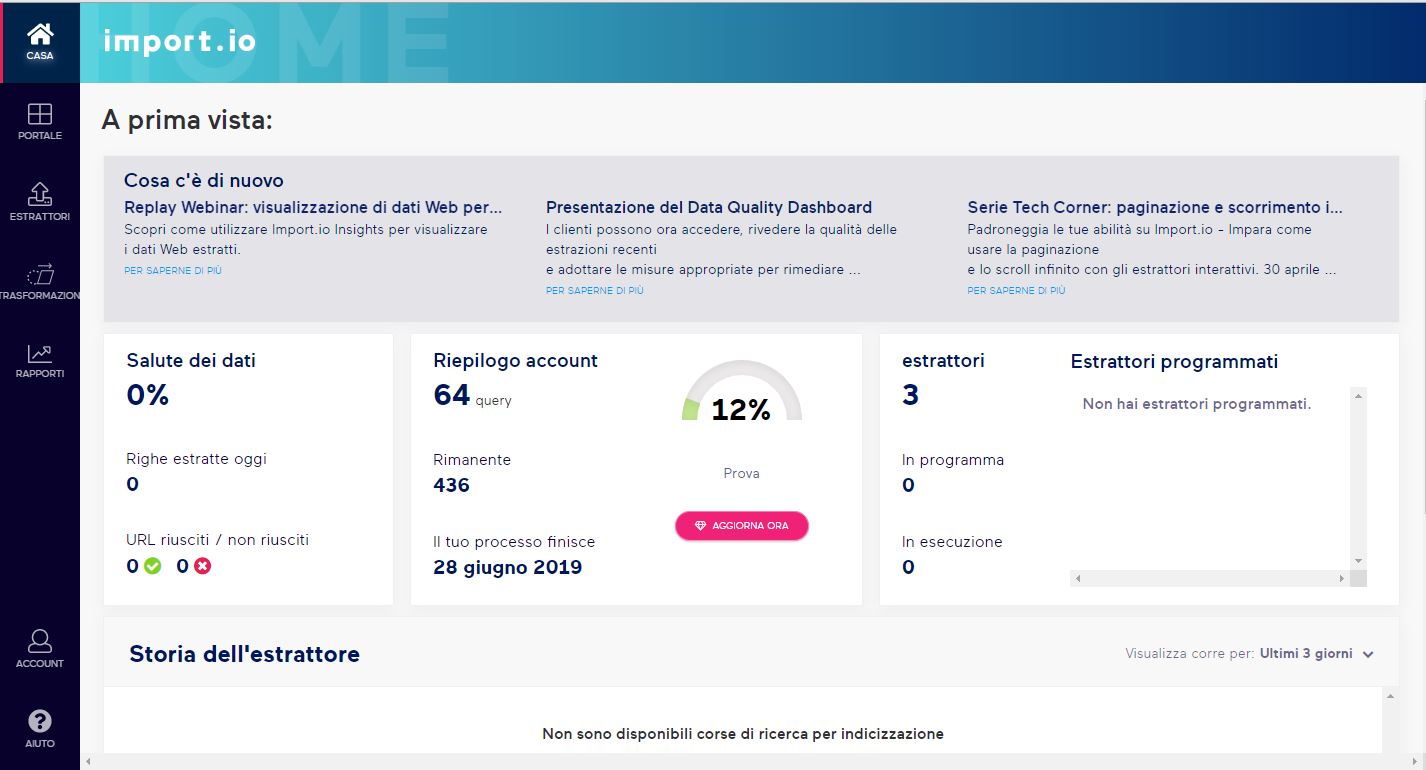
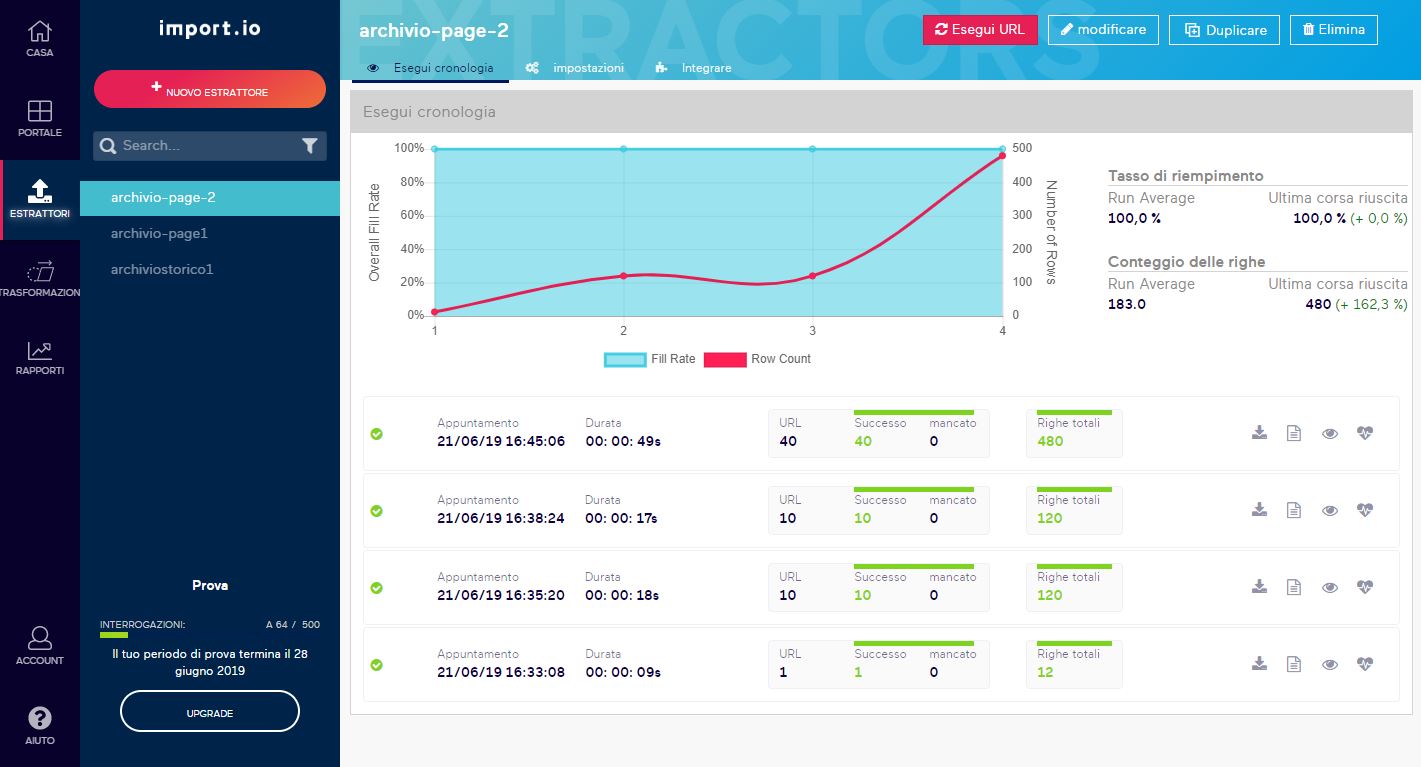
Con l’attributo “disable selected” non lascio l’utente di selezionare quell’opzione perché l’ho uso come nome del filtro.

* Students-container: In questa parte della pagina si mettono in evidenza gli view-card degli studenti. Usando attributi dentro il tag potevo esportare i dati dello studente e poi salvarli, tipo : “data-name, data-surname, data-id, …”. View-card di ogni studente si genera dinamicamente usando *javascript* che verrà spiegato dopo. In una pagina si trovano 9 view-card, però l’utente può vederne altri navigano tramite la sezione *pagination*.



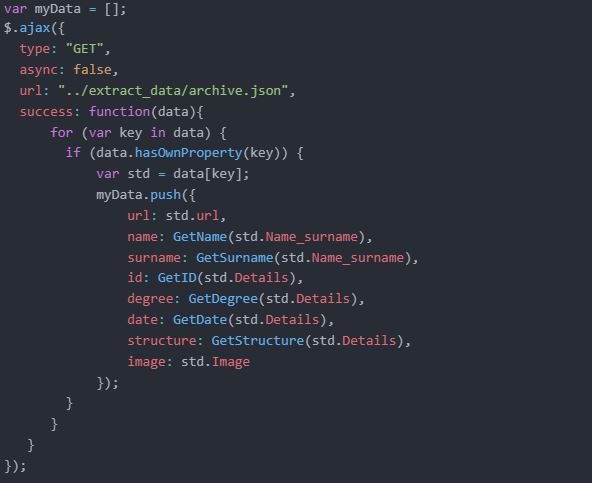
***2.2.3 Tecnologie usate***

* *Come prendere i dati degli studenti ?*

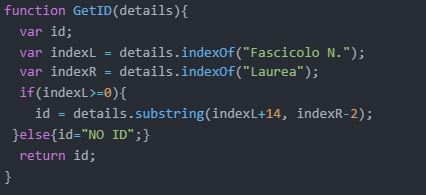
Usando il portale Import.io con una semplice interfaccia point and click, era possibile ricevere dati strutturati pronti per l'analisi o l'integrazione nell'applicazione o nei sistemi interni. Import.io offre i dati web come un servizio. Import.io offre un'estrazione di precisione, fondamentale per l'accuratezza delle informazioni sui consumatori che forniamo ai nostri clienti.  
   
 Per estrarre i dati bastava mettere l’URL del server del Unibo e l’interfaccia ti offriva la possibilità di creare le colonne con i dati che mi servivano. Nel mio sito vengono salvati la metà dei dati del server, perché la quantità era tropo elevata.

I dati vengono salvati in un file con estensione CSV ( File con valori separati da virgola ). Usando un software CSVJSON (<https://www.csvjson.com/csv2json>) potevo fare la conversione del file CSV to JSON che facilitava l’elaborazione e l’estrazione dei dati.

* *Estrarre i dati e creare card-view*

Dopo aver scaricato i dati dovevo elaborarli e creare in HTML le card-view degli studenti. Adesso entra in gioco javascript. Usando AJAX potevo accedere al file JSON e salvare i dati ed elaborarli nel mio file JS. 

Le funzioni *GetName(), GetSurname(), GetID, GetDegree(). GetDate(). GetStructure()* sono funzioni che mi servono per estrarre le informazioni da una stringa. La conversione da CSV a JSON mi aveva portato una sola stringa con tutti le informazioni inglobulate in essa.

*Esempio*: *GetID()*

Dal file JSON, l’ID era nascosta tra Fascicolo e Laurea. Substring serve per prendere una sottostringa da una stringa intera.



Con questa porzione di codice riuscivo a creare una view-card per ogni studente inserendo i dati rispettivi estratti tramite AJAX.

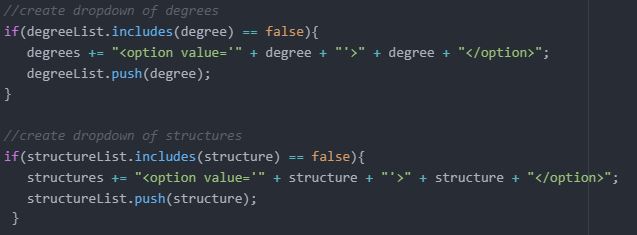


Con la funzione *html*(students) aggancio alla mia struttura html, le card-view create.

* *Cercare e filtrare i dati*

JavaScript offre una funzione *submit()* per eseguire una porzione di codice quando un buttone viene premuto. Quando il button search viene premuto, nome/cognome/laurea inserita vengono trasformata in *lowerCase* e questo input si salva in una query. Poi per ogni studente si va a cercare se contiene questa stringa inserita nel suo nome/cognome/laurea. La funzione *slice(*0,9\*num) fa apparire solo 9\*num view-card e serve per il pagination che verrà spiegato dopo.

Prima di filtrare i dati mi servivano le opzioni per eseguire i filtri. Così prima di implementare l’algoritmo della filtrazione ho aggiunto le opzioni di filtro. Dopo la creazione della stringa contente il codice html, l’ho agganciato al body del html come ho fatto per le view-card degli studenti.



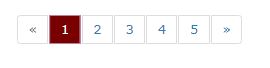


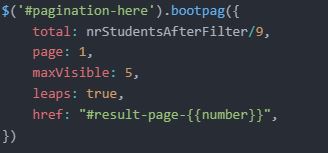
Dopo la creazione vengono aggiunte al body del html.

*Come filtrare i dati ?* Tramite la funzione on(“change”, function()) potevo eseguire una porzione di codice quando uno dei filtri cambiava valore, ovvero quando l’utente sceglieva un filtro. Quando l’utente sceglieva questo filtro io l’ho salvavo in un array e creavo la stringa *finalFilter* = [data-"+key+"='"+filtersObject[key]+"']. Visto che ad ogni view-card c’era un attributo data-degree, data-structure, …etc. io usavo questi attributi per poter filtrare i dati.

 Alla fine, potevo applicare la funzione filter passandoli come parametro il *finalFilter*.

La cosa si complicava con il filtro del periodo temporale. Io ho usato gli operatori AND e OR per poter creare combinazioni diversi di filtri. L’utente sceglieva un periodo per filtrare i dati e io creavo un finalFilter che era una combinazione del *filter* + *yearFilter*.

* *Pagination*

 Per poter migliorare l’UserExperience ho pensato di creare la paginazione delle view-card. Praticamente l’utente all’inizio poteva vedere solo un numero limitato di view-card, in questo caso 9. Tramite una lista di pagine l’utente poteva navigare tramite queste pagine per vedere più view-card.

Usando *bootpag* - plugin jQuery per impaginazione dinamica, potevo implementare la paginazione in modo dinamico.

Tramite la funzione *on(*“page”, function()) che JS offre potevo far vedere la quantità dei dati tramite il numero della pagina che l’utente premeva.

