

# Talking Campus

## Applicazioni e Servizi Web

Simone Luzi - 000725342 {simone.luzi@studio.unibo.it}  
Christian D'Errico - 0001001718 {christian.derrico@studio.unibo.it}

Novembre 2021

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Requisiti</b>	<b>4</b>
2.1	Focus Group . . . . .	5
<b>3</b>	<b>Design</b>	<b>7</b>
3.1	Target users . . . . .	7
3.2	Personas . . . . .	7
3.2.1	Alessandro Spinelli . . . . .	7
3.2.2	Giovanna Esposito . . . . .	8
3.2.3	Tito Bruno Alessandroni . . . . .	8
3.2.4	Antonio Martinelli . . . . .	8
3.3	Scenario d'uso . . . . .	9
3.3.1	Profilo Studente . . . . .	9
3.3.2	Profilo Professore . . . . .	10
3.3.3	Profilo Admin . . . . .	10
3.4	Mockup . . . . .	11
3.4.1	Pagina login . . . . .	11
3.4.2	Mappa . . . . .	11
3.4.3	Profilo studente e professore - Area personale . . . . .	12
3.4.4	Profilo studente - Locali registrati . . . . .	12
3.4.5	Profilo studente - Notifiche . . . . .	14
3.4.6	Profilo professore - Orario di ricevimento . . . . .	14
3.4.7	Profilo professore - Gestione dei corsi . . . . .	15
3.4.8	Profilo admin - Aggiunta locale . . . . .	15
3.4.9	Profilo admin - Modifica/Elimina locale . . . . .	16
3.5	Prodotto finale . . . . .	18
3.6	Schema . . . . .	24
<b>4</b>	<b>Tecnologie</b>	<b>26</b>
4.1	TypeScript . . . . .	26
4.2	Sass . . . . .	26
4.3	Frontend . . . . .	26
4.3.1	React . . . . .	26

4.3.2	React Router . . . . .	27
4.3.3	Antd . . . . .	27
4.3.4	Leaflet . . . . .	27
4.3.5	React-Leaflet . . . . .	27
4.3.6	Font Awesome . . . . .	27
4.3.7	Axios . . . . .	28
4.4	BackEnd . . . . .	28
4.4.1	Node.js . . . . .	28
4.4.2	Express . . . . .	28
4.4.3	JSON Web Token (JWT) . . . . .	28
4.4.4	Socket.io . . . . .	28
4.4.5	SwaggerHub . . . . .	28
4.4.6	Postman . . . . .	29
4.5	Database . . . . .	29
4.5.1	MongoDB . . . . .	29
4.5.2	Mongoose . . . . .	29
<b>5</b>	<b>Codice</b>	<b>30</b>
5.1	Gestione Beacon . . . . .	30
5.2	Json Web Token (JWT) . . . . .	31
<b>6</b>	<b>Test</b>	<b>34</b>
6.1	Valutazione euristica . . . . .	34
6.2	Cognitive Walkthrough . . . . .	35
6.2.1	Login e Registrazione . . . . .	36
6.2.2	Profilo: Studente . . . . .	36
6.2.3	Profilo: Professore . . . . .	37
6.2.4	Profilo: Admin . . . . .	38
6.3	Co-discovery Learning . . . . .	39
6.3.1	Profilo: Professore . . . . .	39
6.3.2	Profilo: Studente . . . . .	39
6.4	Usability Test . . . . .	41
<b>7</b>	<b>Deployment</b>	<b>43</b>
<b>8</b>	<b>Conclusioni</b>	<b>44</b>
8.1	Possibili miglioramenti futuri . . . . .	44
8.2	Commenti finali . . . . .	44

# **Capitolo 1**

## **Introduzione**

Questo progetto mira alla realizzazione di una Single Page Application che utilizzi i beacon (simulati) rilevati dagli smartphone e installati all'esterno delle aule e altri locali e servizi pubblici (servizi igienici, reception, mensa) all'interno di uno smart campus, per fornire ai frequentatori informazioni utili per la loro permanenza all'interno dell'edificio.

# Capitolo 2

## Requisiti

Dopo un'attenta fase di analisi preliminare abbiamo deciso di dotare il nostro sistema dei seguenti requisiti, adottando, per la selezione delle funzionalità, un approccio d'ispirazione MoSCoW (M - Must have; S - Should have; C - Could have; W - Won't have):

Must have:

- Registrazione e Login sulla base di diversi ruoli utente tramite RBAC (Role Base Access Control):
  - Si prevede la realizzazione di tre profili, professore (che fornisce al sistema il proprio orario di ricevimento, i corsi che insegna e gli orari e le aule delle lezioni), studente (che riceve delle notifiche riguardanti i locali ai quali è interessato) e amministratore (per effettuare operazioni CRUD sugli spazi).
- Visualizzazione tramite mappa e monitoraggio real time dei seguenti locali del campus:
  - Sale studio
  - Servizi igienici
  - Aule
  - Mense
  - Reception
  - Per ogni locale (ad eccezione delle reception) viene mostrato il numero di persone all'interno in rapporto al numero totale di posti disponibili.
  - Per alcuni locali vengono elencate informazioni aggiuntive:
    - Aule: orari delle lezioni (lezione attualmente in svolgimento e le lezioni previste).
    - Mense: informazioni e collegamenti a servizi di ristorazione

- Reception: orari di apertura e chiusura
- Notifiche Push agli studenti interessati quando si sta per riempire un'aula che stanno monitorando.

Could have:

- Visualizzazione dei diversi piani del campus, con la possibilità di spostarsi da uno ad un altro.
- Permettere di prenotare un posto in aula studio per un determinato orario.
- Permettere di prendere appuntamento col professore per ricevimento al quale arriverà una notifica.
- Area personale dello studente e del professore, dove è possibile controllare le proprie informazioni personali.

## 2.1 Focus Group

Nella fase di definizione dei requisiti del progetto abbiamo scelto di condurre anche un focus group con l'obiettivo di capire meglio cosa effettivamente un possibile utente del nostro dominio si aspetti dal sistema in sviluppo.

Si è formato, quindi, un gruppo abbastanza variegato di 6 persone, dal quale abbiamo provato ad ottenere più feedback possibili per scoprire svantaggi e vantaggi delle features di cui avevamo pianificato l'implementazione.

L'interazione guidata da noi ha coinvolto i partecipanti, che si sono mostrati interessati, e ciò ha portato a commenti e domande che hanno fatto emergere risultati inattesi. Infatti, le idee iniziali presentate come funzionalità di base sono state apprezzate dal gruppo, essendo features basilari ma fondamentali, e in seguito hanno fornito spunti per ulteriori funzionalità non previste.

È stata accolta con piacere la possibilità di ricevere le notifiche relative alla situazione di una data aula in cui si frequenteranno le lezioni, anche se sarebbe stata gradita un'integrazione con l'applicativo Virtuale. In questo modo si avrebbe un'iscrizione spontanea alle aule in cui si terranno le lezioni dei corsi a cui ci si è iscritti, ed anche un link ai contenuti didattici proposti da questo portale.

A questo proposito, il gruppo avrebbe apprezzato la possibilità di accedere al sistema usufruendo del single-sign-on a partire dalle credenziali in uso presso Virtuale, Studenti Online, Almaesami e gli altri applicativi dell'università (in questo modo si eviterebbe una registrazione dedicata a "Talking Campus").

Un altro aspetto discusso è stata la funzionalità di ricerca di un professore per nome, senza dover controllare nelle varie aule se il professore sta svolgendo una lezione in quel preciso momento.

Ipotizzando la presenza di un sistema di prenotazione di un posto in un certa aula, i partecipanti hanno suggerito di inserire un calendario in cui è possibile selezionare le lezioni a cui si parteciperà, in modo da effettuare una prenotazione

automaticamente. Sulla base della stessa intuizione, il gruppo ha valutato anche la possibilità di richiedere un colloquio con un professore dotando il sistema di una gestione autonoma di eventuali file e scorrimenti degli orari dei ricevimenti in base alle persone in coda. Uno degli studenti ha richiesto l'inserimento di una chat relativa ad ogni lezione effettuata, ma la proposta non ha ottenuto il consenso del resto del gruppo, che ha valutato la maggiore utilità di altre applicazioni di messaggistica, che sarebbero più immediate e di gran lunga più utilizzate.

Sono stati sollevati dei dubbi su aspetti di privacy, nello specifico per quanto riguarda il fatto che l'app, in tempo reale, possa conoscere la posizione dello studente che la utilizza e fornisca informazioni che possano risultare sensibili su determinati locali e spazi universitari. D'altra parte, per i corsi a frequenza obbligatoria i professori avrebbero meno difficoltà a sapere quali studenti siano presenti, evitando possibili falsificazioni delle registrazioni.

Un'ulteriore funzionalità di cui si è discusso è stata la presenza di dati statistici elaborati a partire dalle posizioni degli studenti in determinate ore per fornire informazioni all'utente riguardanti le ore di maggior affollamento nei locali, cosa che comporterebbe però anche la storizziazione di una grande quantità di dati. L'allestimento del focus group ha richiesto l'elezione di un moderatore e di un co-moderatore (compito che abbiamo assolto personalmente), la produzione di una lista di topic di discussione (che prevedeva features essenziali ed elementi del dominio da considerare), mentre non si è effettuata l'organizzazione degli spazi e la registrazione dell'incontro, vista la modalità con cui si è svolto (ovvero in videochiamata). In ogni caso è stata incentivata la discussione fra tutti i partecipanti e sono stati raccolti tutti i feedback necessari a garantire la fairness tipica dei focus group.

Generalmente abbiamo ricevuto pareri positivi sulle proposte avanzate dal team di sviluppo relativamente alle funzionalità già previste; alcune proposte hanno esplorato dinamiche che non avevamo considerato e ci hanno fornito ottimi spunti per eventuali futuri sviluppi dell'applicativo, mentre altre sono risultate irrealizzabili per motivi logistici (ad esempio l'integrazione con Virtuale). Pertanto, complessivamente, abbiamo ritenuto questo tipo di attività proficua, considerando anche che si è svolta nella fase di definizione dei requisiti del progetto, prima della produzione dei mockup, un momento sempre delicato in cui è fondamentale delineare bene funzionalità, aree tematiche e modalità di interazione.

# Capitolo 3

## Design

### 3.1 Target users

Analizzare il dominio applicativo ci ha portato alla conclusione di riconoscere in professori e studenti i principali utilizzatori della nostra applicazione. Pertanto, le interfacce e il funzionamento di Talking Campus sono stati progettati per essere utilizzati da una platea di utenti piuttosto ampia, che abbiamo deciso di estendere, poi, introducendo anche il profilo “admin”.

### 3.2 Personas

#### 3.2.1 Alessandro Spinelli

Alessandro Spinelli è uno studente appena diplomato al Liceo delle Scienze Umane Perticari di Senigallia con un voto di 70 centesimi; ha fatto fatica all'esame di quinta perché si è reso conto di non aver scelto la scuola adatta a sè e ha deciso di cambiare il suo percorso scolastico iscrivendosi all'università Ingegneria e Scienze Informatiche alla sede UNIBO di Cesena. Tra le sue passioni ci sono i videogiochi, cosa che lo aveva portato a coltivare la pessima abitudine di arrivare in ritardo alle lezioni a causa di prolungate sessioni notturne con i suoi amici. Mantenendo il vizio di svegliarsi tardi rischierebbe di non riuscire a trovare posto alle sue nuove lezioni universitarie, e di conseguenza a non seguire bene i corsi che, per di più, gli risulterebbero abbastanza ostici; infatti la sua conoscenza delle materie informatiche è abbastanza elementare vista l'assenza di materie simili nel suo precedente corso di studio. Utilizza costantemente il suo smartphone come tutti i suoi coetanei soprattutto per chiacchierare con i suoi amici e non ha nessuna difficoltà a navigare in internet o ad utilizzare funzioni più complesse.

### **3.2.2 Giovanna Esposito**

Giovanna Esposito è una ragazza di 35 anni che ha interrotto i suoi studi all'età di 20 anni a causa della nascita di sua figlia Nicole, che ha dovuto accudire mentre il suo compagno lavorava per mantere la famiglia. Giovanna è sempre stata una ragazza ambiziosa e, una volta cresciuta abbastanza sua figlia, ha deciso di riprendere gli studi per cercare di intraprendere la carriera che sognava e gli era stata preclusa a causa delle vicissitudini familiari. Ha pochissime competenze informatiche di base, sa accendere a malapena il computer e utilizza il telefono solamente per mandare foto della figlia alla madre e per chiacchierare con le colleghe. Ha scelto di iscriversi ad architettura perché molte delle sue ex compagne di superiori avevano questa passione e col tempo sono diventate molto ricche. Riuscirebbe ad utilizzare un'applicazione molto semplice date le sue scarse conoscenze e le risulterebbe molto comoda una mappa per orientarsi nel nuovo campus non avendo amiche coetanee che frequentano insieme a lei e con cui poter fare gruppo.

### **3.2.3 Tito Bruno Alessandroni**

Tito Bruno Alessandroni è un cinquantenne abbastanza introverso che di professione fa il professore di Algoritmi e Strutture Dati presso UNIBO; lavora sia nella sede di Bologna che nella sede di Cesena. Il suo percorso di vita lo ha portato ad affidarsi completamente ai processi di digitalizzazione, portandolo ad essere un po' scontroso e poco accomodante nelle relazioni personali. Vorrebbe mettere a disposizione il proprio orario di lezione e di ricevimento in un portale dove gli studenti, che reputa essere persone mature, si possano informare autonomamente; inoltre risponde raramente alle email e non ama negoziare appuntamenti straordinari, quindi si aspetta uno strumento di immediato utilizzo che fornisca tutte le informazioni del caso in maniera chiara e completamente accessibile, per evitare assenze alle lezioni e fraintendimenti. La sua professione lo ha reso molto pratico nell'utilizzo di tutti i tipi di tecnologia e costantemente informato sulle novità di questo settore.

### **3.2.4 Antonio Martinelli**

Antonio Martinelli è un professore di Analisi che si è appena trasferito da Milano perchè stanco della vita frenetica e felice di accettare una cattedra in una città più piccola. La cosa che più ha sofferto nella metropoli lombarda è stata la confusione generata dal traffico e dalle persone di corsa, visto che lui si orienta difficilmente e ha dei ritmi poco compatibili con quelli delle persone comuni; infatti tende ad organizzarsi all'ultimo, è disordinato, sempre impegnato e non programma mai le sue attività. Ha competenze digitali di base: riuscirebbe a gestire un profilo in cui inserisce i propri dati, magari con l'aiuto di qualche collega con cui confrontarsi; sarebbe utile per lui avere un supporto digitale che lo aiuti a rispettare i suoi doveri d docente.

### **3.3 Scenario d'uso**

Lo scenario d'uso tipico di un qualunque utente di Talking Campus richiede di utilizzare l'applicazione per ottenere informazioni da un determinato locale:

1. L'utente si registra nell'applicazione.
2. Dopo una corretta registrazione, effettua il login.
3. Una volta autenticato può visualizzare la mappa del campus e cercare il locale di suo interesse.
4. Questo potrebbe trovarsi su un piano differente della mappa, quindi è necessario spostarsi al livello corretto.
5. Individuato il punto, il click sul marker apre un pop-up con le informazioni che l'utente vuole ottenere.

Per le aule, è possibile ricevere informazioni sulle lezioni in svolgimento (per le quali è visualizzabile anche il nome del docente, con annesso numero di telefono, e-mail e orario di ricevimento) e sulle lezioni in programma:

1. l'utente clicca sul marker che è collegato all'aula di suo interesse: se nella determinata ora in cui si consulta il pop up informativo vi è una lezione nell'aula, questa viene mostrata. Un click sul pulsante info accanto al nome del docente ne mostra i dati.
2. l'utente clicca sul pulsante "lezioni in programma": viene visualizzata una tabella che mostra le lezioni in programma durante la settimana in quell'aula, con l'indicazione dell'orario di svolgimento e del professore di riferimento.

#### **3.3.1 Profilo Studente**

Effettuato il login come studente, è possibile effettuare le seguenti operazioni:

- Consultare le proprie informazioni personali (premendo sull'apposito pulsante "Informazioni personali").
- Visualizzare le notifiche ricevute ed eliminarle (pulsante "Notifiche")
- Visualizzare i locali monitorati e rimuoverli (pulsante "Locali registrati")

Per aggiungere un locale alla lista di quelli osservati (dai quali si vuole ricevere degli avvisi) lo studente deve:

1. cliccare sul marker del locale d'interesse
2. premere il pulsante "Osserva locale" che sarà presente nel popup apparso

### **3.3.2 Profilo Professore**

Un login effettuato da un utente con profilo professore permette le seguenti operazioni:

- Consultare le proprie informazioni personali (premendo sull'apposito pulsante “Informazioni personali”).
- Modificare il proprio orario di ricevimento, aggiungendo orari settimanali o eliminando i vecchi (pulsante “Modifica orario di ricevimento”)
- Controllare i corsi insegnati (anche eliminarli) e per ognuno di essi aggiungere lezioni settimanali o eliminare quelle già presenti (pulsante “Gestisci i tuoi corsi”)

### **3.3.3 Profilo Admin**

Infine, un utente con i privilegi di admin può interagire con il sistema effettuando le seguenti azioni:

- Aggiungere un locale:
  1. l'admin preme il pulsante “Aggiungi locale” abilitando la modalità “aggiungi”
  2. seleziona il punto sulla mappa in cui inserire il nuovo marker, la cui posizione può essere comunque modificata fino al completamento dell'operazione di aggiunta mediante drag dell'indicatore comparso sullo schermo
  3. inserisce i dati sul nuovo locale compilando il form
  4. conclude l'inserimento premendo sul pulsante “aggiungi” (le operazioni sono comunque sempre annullabili con la pressione del pulsante “annulla”)
- Modificare le info di un locale già esistente:
  1. l'admin preme il pulsante “Modifica locale” abilitando la modalità “modifica”
  2. seleziona il marker corrispondente al locale di cui si vogliono modificare le informazioni
  3. effettua le modifiche
  4. salva le operazioni premendo sul pulsante “modifica” (le operazioni sono comunque sempre annullabili con la pressione del pulsante “annulla”)
- Eliminare un locale:
  1. l'admin preme il pulsante “Elimina locale” abilitando la modalità “elimina”

2. clicca sul marker del locale che vuole cancellare
3. conferma l'eliminazione premendo “si”, o annulla tutto cliccando “no”

## 3.4 Mockup

In questa sezione mostriamo come inizialmente abbiamo ideato le schermate della nostra applicazione, mettendo in evidenza anche il processo creativo evolutivo che ci ha portato dai primi mockup alla costituzione del prodotto finale. Oltre a fornire una guida per lo sviluppo dell'applicazione, i mockup sono stati utilizzati per tutti i test presenti nel Capitolo 6.

### 3.4.1 Pagina login

Abbiamo predisposto un form iniziale a più tab: nella prima scheda abbiamo inserito una pagina di login, mentre le altre due schermate consentono la registrazione dei due profili, “Studente” e “Professore”. Il design è minimale e semplice, poichè il nostro scopo è quello di facilitare le operazioni all’utente senza appesantire e rendere troppo “pompose” le interfacce utente iniziali

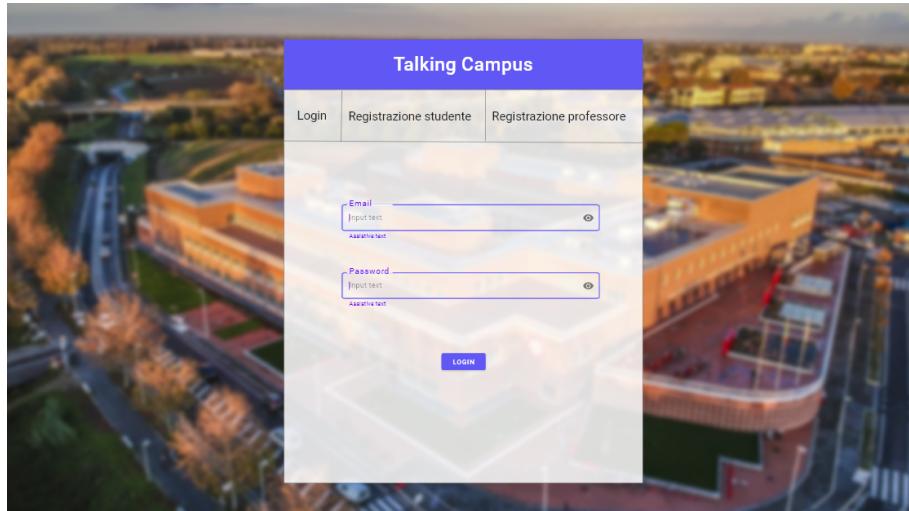


Figura 3.1: Mockup del form iniziale

### 3.4.2 Mappa

Il cuore di Talking Campus è la mappa dello smart campus, consultabile da ogni tipologia d’utente: si tratta di un pannello centrale realizzato con l’utilizzo del framework javascript Leaflet, che viene utilizzato per creare mappe web

interattive. Il mockup in questo caso è stato utilizzato come uno scheletro base, poi esteso in produzione: ispirandoci alle mappe dei grandi campus delle università americane, che mostrano i marker corrispondenti ai locali d'interesse utilizzando un sistema di icone che migliora la user experience dell'utente e rende più confortevole l'orientamento nella navigazione, abbiamo adottato una soluzione analoga anche per Talking Campus, ricorrendo a Font Awesome per la scelta delle piccole immagini da associare alle stanze. Le frecce che nel mockup permettono di spostarsi da un piano all'altro dell'edificio sono state poi sostituite da una combobox che permette di visualizzare il piano corrente e sapere su quale piano eventualmente spostarsi.



Figura 3.2: Mockup della mappa

### 3.4.3 Profilo studente e professore - Area personale

Studenti e professori hanno una sezione all'interno della quale poter consultare le proprie informazioni personali.

### 3.4.4 Profilo studente - Locali registrati

Uno studente può registrare dei locali a cui è interessato e per i quali vuole ricevere degli avvisi. Tali notifiche riguardano ad esempio la disponibilità o meno di un posto libero in un aula che era stata completamente occupata o che può essersi appena riempita. La lista dei locali registrati viene mostrata nella sezione “Aule registrate”, che nella versione finale dell'applicazione è stata rinominata in “Locali registrati”.

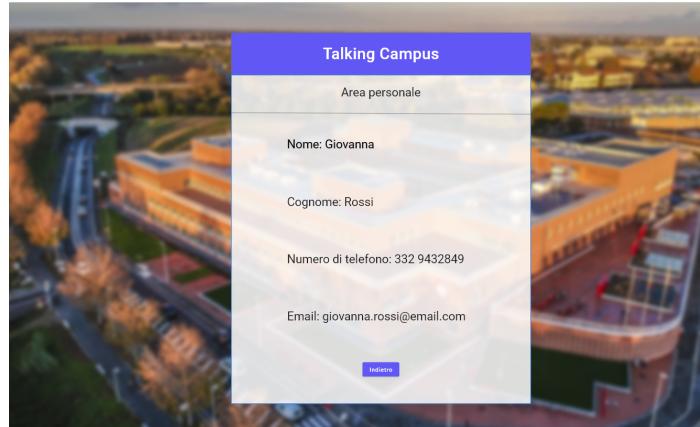


Figura 3.3: Mockup della pagina con le info personali

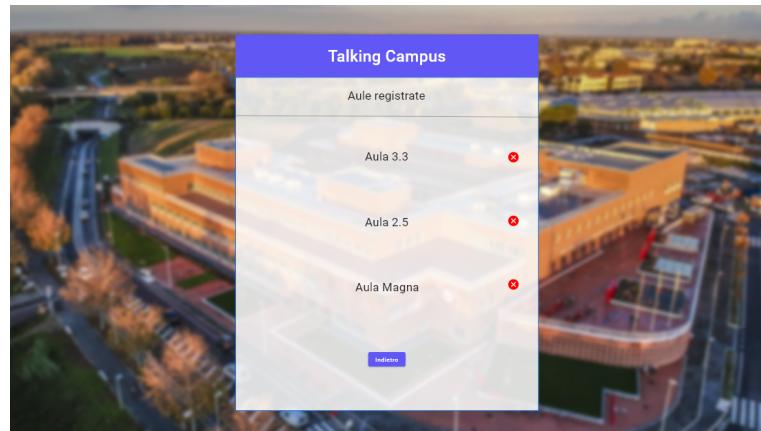


Figura 3.4: Mockup della pagina locali registrati

### 3.4.5 Profilo studente - Notifiche

Tali notifiche vengono mostrate nella sezione “Notifiche”.

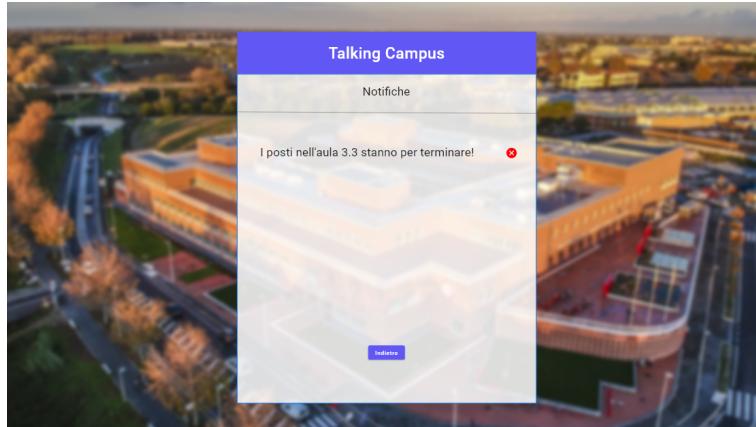


Figura 3.5: Mockup della pagina notifiche

### 3.4.6 Profilo professore - Orario di ricevimento

Ogni professore può specificare il proprio orario di ricevimento settimanale grazie ad una apposita sezione. Qui può eliminare i vecchi orari e aggiungerne di nuovi ogni volta lo desideri.

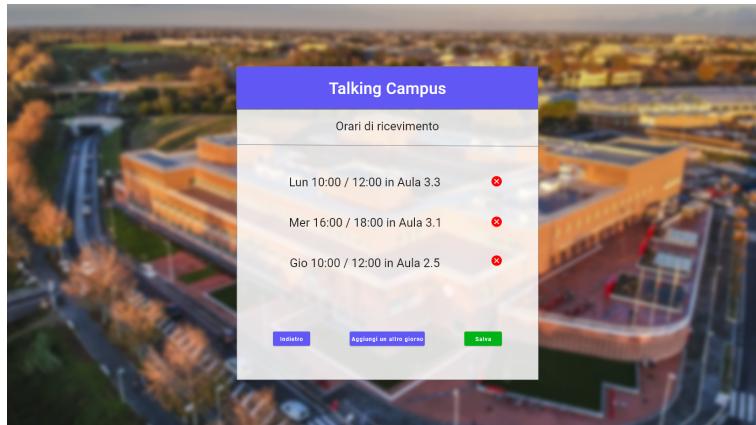


Figura 3.6: Mockup della pagina con gli orari di ricevimento

### 3.4.7 Profilo professore - Gestione dei corsi

Ogni professore può insegnare più corsi; in questa sezione può controllarli e per ognuno di essi può modificare gli orari delle lezioni e aggiungerne di nuove. Può anche inserire un nuovo corso che insegherà in futuro oppure eliminarne uno che non insegherà più con tutte le relative lezioni.

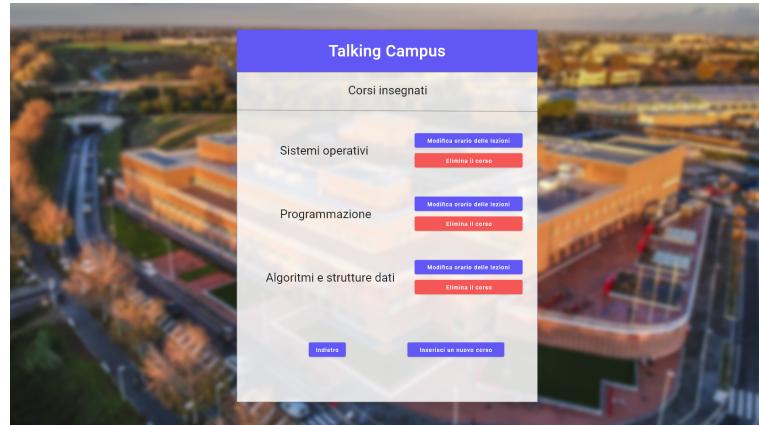


Figura 3.7: Mockup della pagina di gestione dei corsi

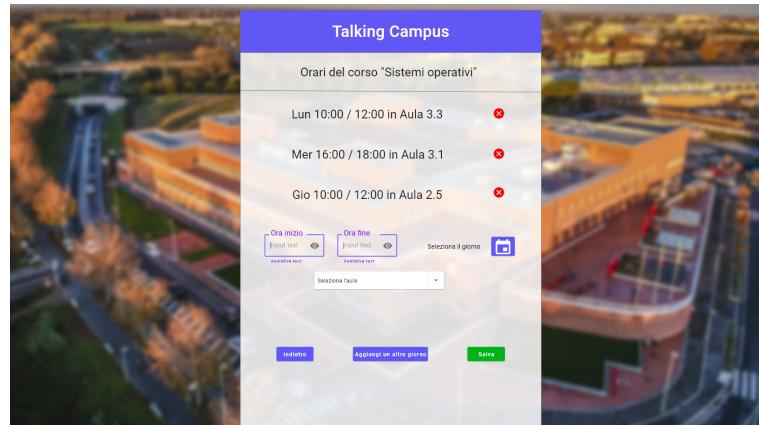


Figura 3.8: Mockup della pagina con le lezioni di un corso

### 3.4.8 Profilo admin - Aggiunta locale

L'admin può effettuare operazioni CRUD sui marker che vengono mostrati all'interno della mappa. Tali azioni vengono performate all'interno di pop up dedicati ad ogni possibile interazione. Ciò si traduce nella possibilità di attivare

tre modalità differenti con cui lavorare direttamente sulla mappa. L'attivazione della modalità “aggiungi”, ad esempio, permette all'admin di inserire nuovi marker alla pressione di un punto sulla mappa (marker che è comunque dragabile fino al completamento delle operazioni). Scelto il punto, viene aperto un popup che permette l'inserimento dei dati per il nuovo locale.

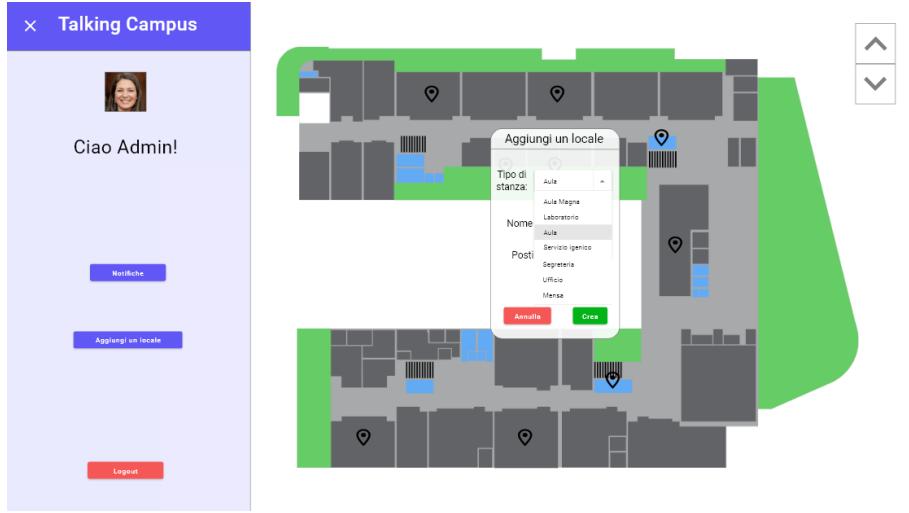


Figura 3.9: Mockup del popup “aggiungi”

### 3.4.9 Profilo admin - Modifica/Elimina locale

In modalità “modifica” l'admin può premere su un marker già esistente e aprire un popup analogo al precedente, ma che questa volta mostra un form precompilato con i dati del locale, in attesa di modifiche provenienti dall'utente. In un primo momento avevamo messo in conto di accorpate la funzionalità di modifica con l'eliminazione del marker (concepita come una modifica estrema), introducendo nel popup del mockup il pulsante “Elimina stanza”. Nella versione finale abbiamo però scelto di separare le due funzionalità, creando una modalità elimina e un popup apposito. Quella che è stata mantenuta però è la richiesta della conferma di un'eventuale eliminazione, allo scopo di rendere l'admin sempre consapevole dell'operazione che sta effettuando e fargli prendere la decisione corretta, coerentemente con le sue reali intenzioni.

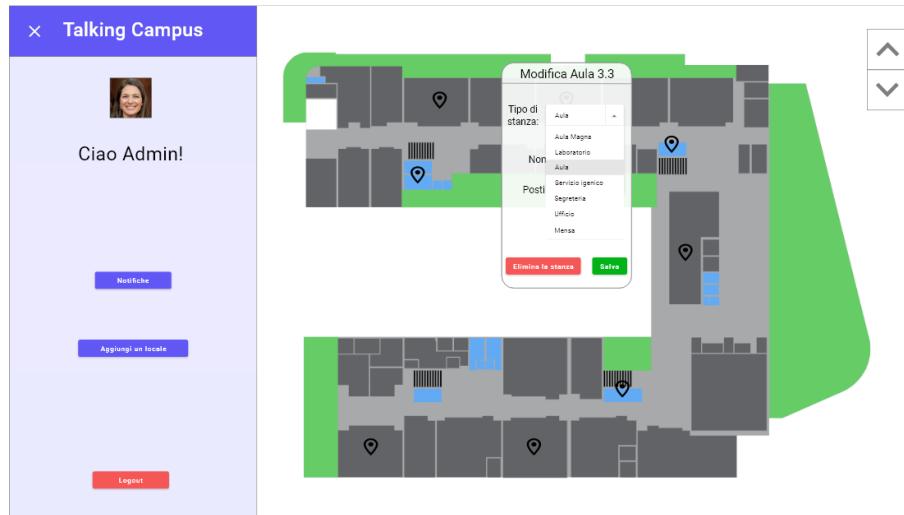


Figura 3.10: Mockup del popup “modifica”

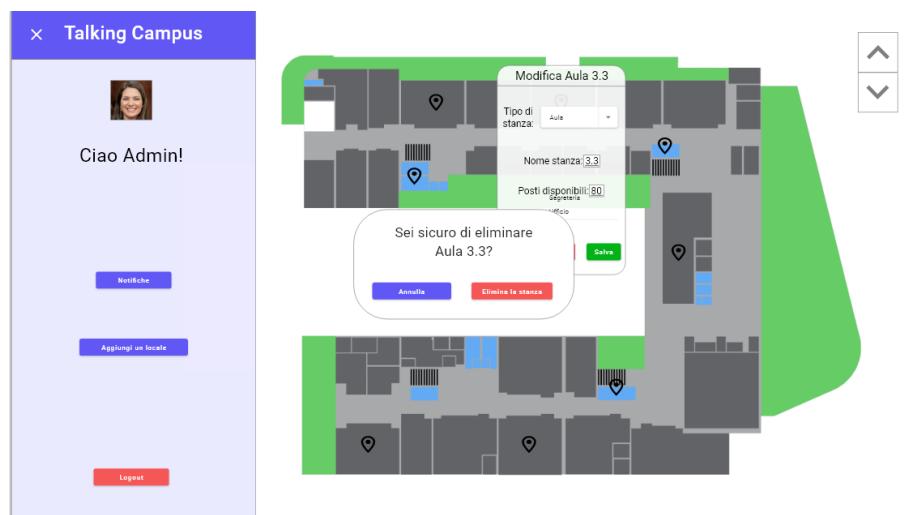


Figura 3.11: Mockup della dialog di "conferma eliminazione"

### 3.5 Prodotto finale

Ecco dunque come si presenta Talking Campus nella sua versione definitiva

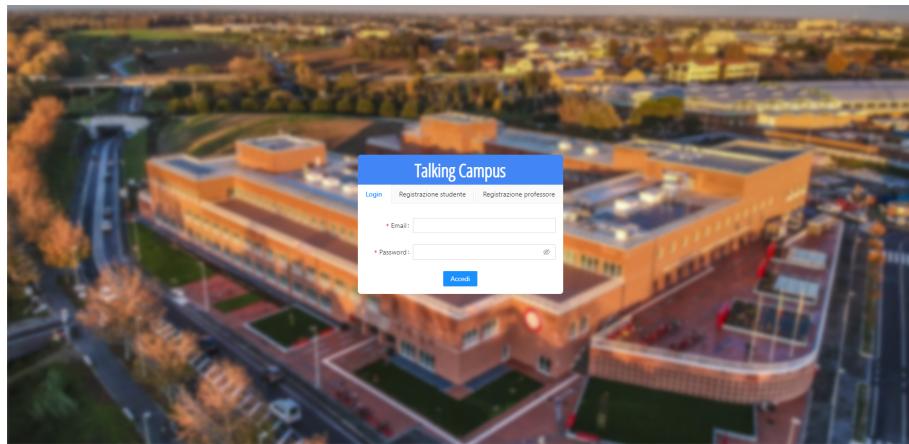


Figura 3.12: Pagina “Login”



Figura 3.13: Mappa principale

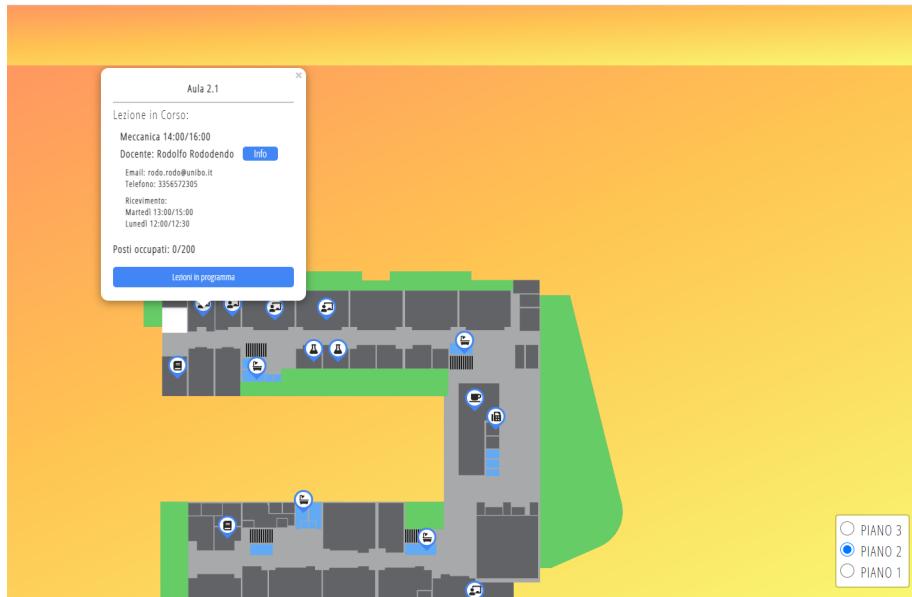


Figura 3.14: Pop up informativo

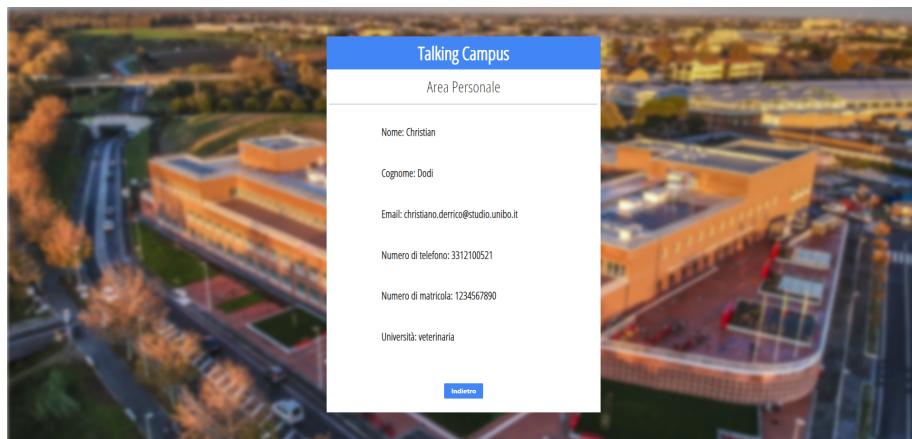


Figura 3.15: pagina “Informazioni personali”

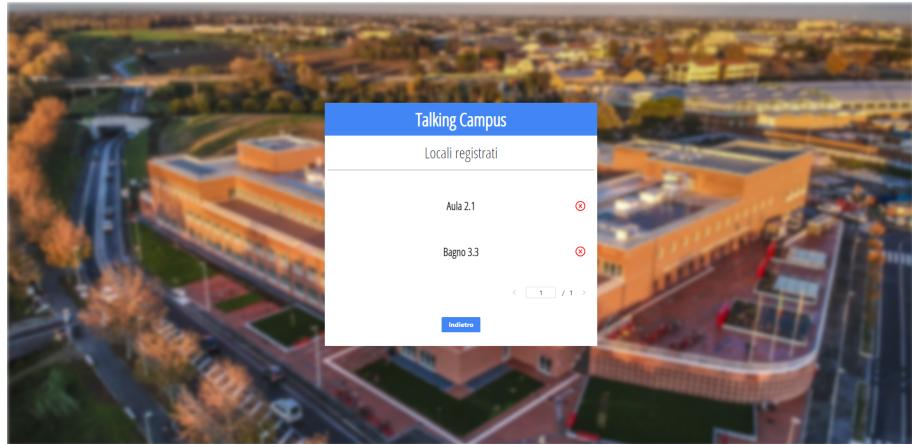


Figura 3.16: pagina “Locali osservati”

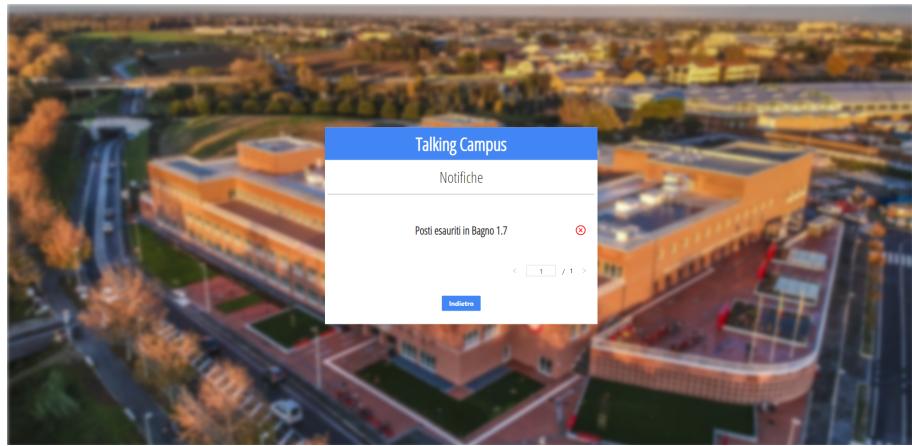


Figura 3.17: pagina “Notifiche”

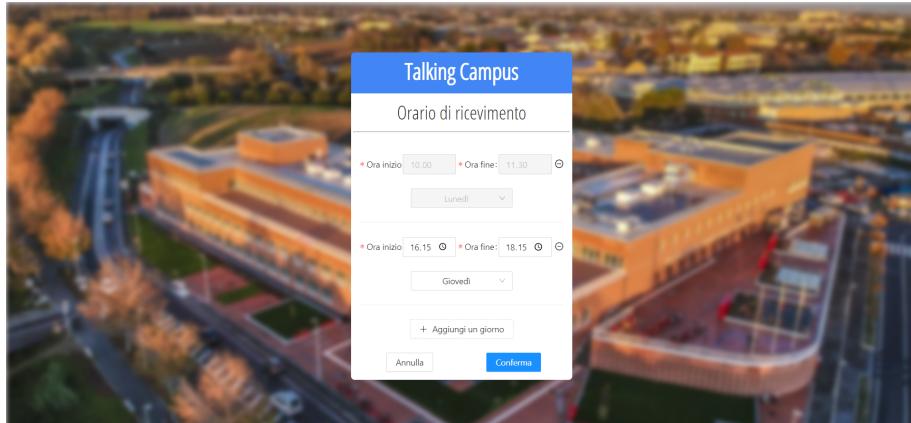


Figura 3.18: pagina “Orario di ricevimento”

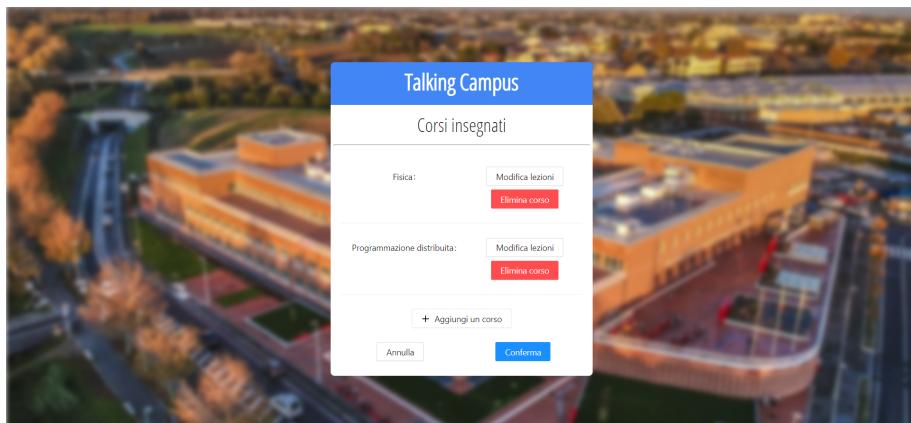


Figura 3.19: pagina “Gestisci i tuoi corsi”

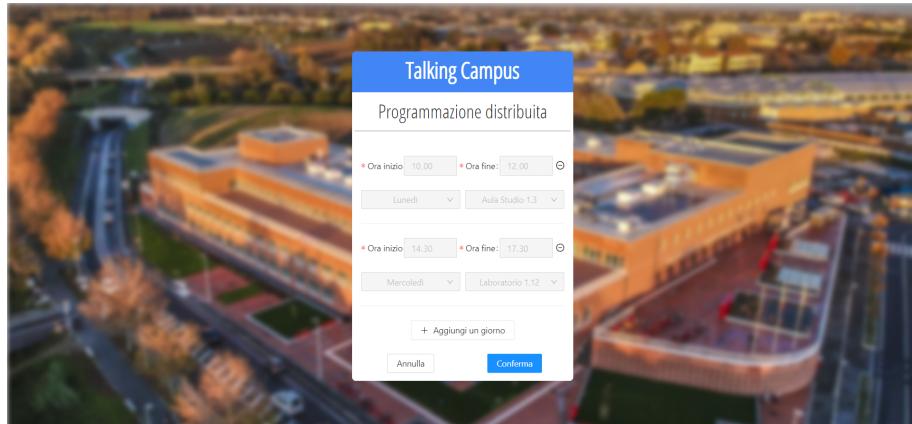


Figura 3.20: pagina “Modifica orari lezioni”

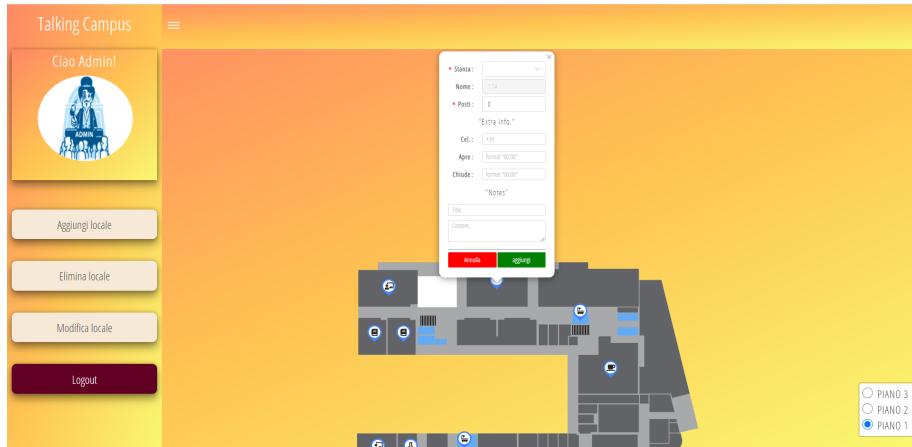


Figura 3.21: Mode “aggiungi”

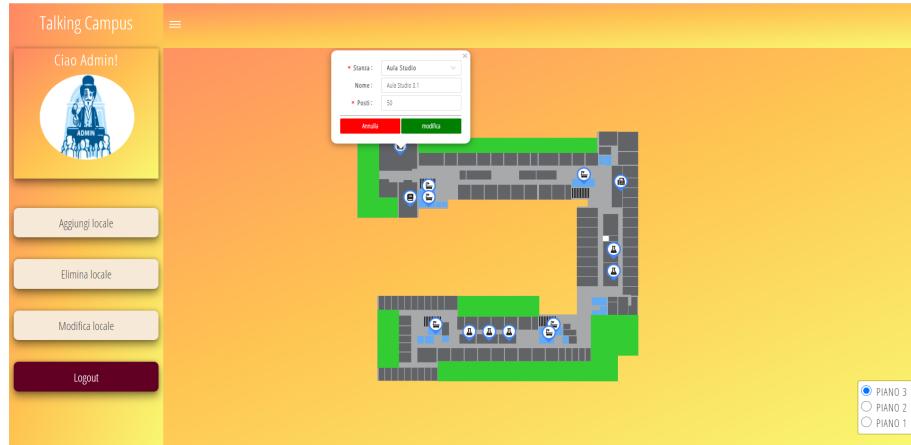


Figura 3.22: Mode “modifica”



Figura 3.23: Mode “elimina”

### 3.6 Schema

Il seguente grafico mostra le diverse entità di Talking Campus:

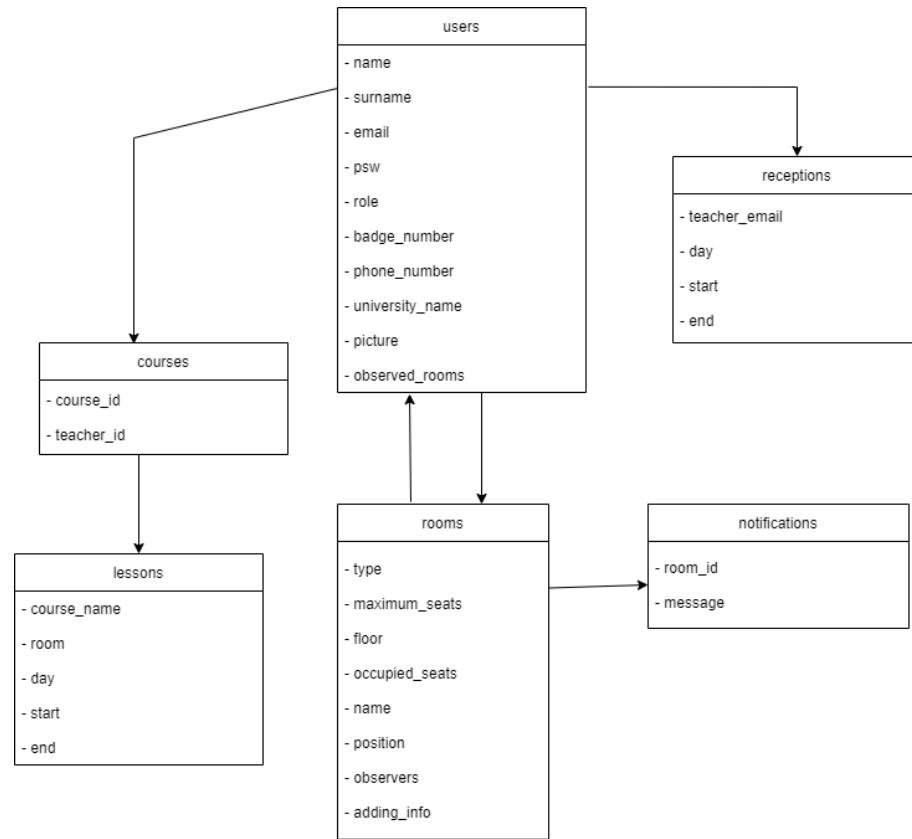


Figura 3.24: Le entità del sistema

Alle collezioni presenti nell'immagine sono state associate le seguenti operazioni:

- Users: funzioni per inserire utenti e ottenerne i dati.
- Rooms: funzioni per gestire e ottenere i dati dei locali.
- Receptions: funzioni per portare a termine le operazioni CRUD sugl'orari di ricevimento dei professori
- Notifications: funzioni per gestire le notifiche inviate dai locali
- Lessons: funzioni per aggiungere e ottenere le lezioni in programma.
- Courses: funzioni per aggiungere e ottenere i corsi tenuti dai professori.

L'immagine che segue rappresenta il percorso seguito dalle richieste che vengono inoltrate alla nostra applicazione, nonchè, in una sintesi perfetta, la struttura che abbiamo scelto per il Backend.

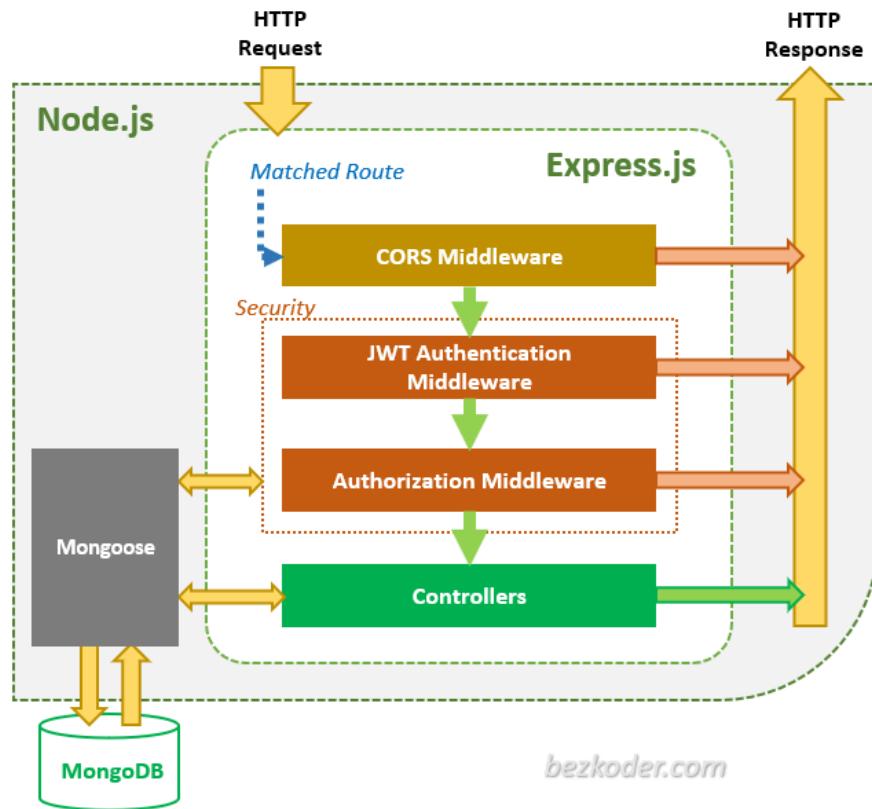


Figura 3.25: Struttura del BackEnd del sistema

# Capitolo 4

## Tecnologie

Abbiamo deciso di utilizzare lo stack MERN, cosa che ci ha permesso di costruire facilmente un'architettura a 3 livelli interamente utilizzando Javascript e JSON.

### 4.1 TypeScript

TypeScript[11] è un Super-set di JavaScript che basa le sue caratteristiche su ECMAScript 6; estende la sintassi del linguaggio nativo in modo che qualunque programma sia anche in grado di funzionare con TypeScript senza nessuna modifica. Una delle feature principali introdotte è un sistema di annotazione dei tipi, che consente di controllare i tipi durante la fase di compilazione.

### 4.2 Sass

Sass (Syntactically Awesome StyleSheets) è un'estensione del linguaggio CSS che permette di utilizzare variabili, di creare funzioni e di organizzare il foglio di stile in più file.

Il linguaggio Sass si basa sul concetto di preprocessore CSS, il quale serve a definire fogli di stile con una forma più semplice, completa e potente rispetto ai CSS e a generare file CSS ottimizzati, aggregando le strutture definite anche in modo complesso. È compatibile con tutte le versioni di CSS.

### 4.3 Frontend

#### 4.3.1 React

React [8] è una libreria Javascript open source, usata per costruire interfacce utente. Sviluppata da Facebook, è stata utilizzata per la realizzazione del news feed all'interno del social network e per costruire la versione web di Instagram.

Oltre a vantare una community in costante crescita, React è apprezzata e utilizzata da famose aziende. Uno dei punti di forza è sicuramente l'approccio dichiarativo e il fatto di semplificare la creazione di applicazioni single-page dinamiche (Single Page Applications o SPA). Utilizzando React un'applicazione web viene strutturata in tante parti diverse, i componenti o React Components. In maniera approssimativa, possiamo dire che i componenti sono i vari elementi che compongono l'interfaccia e che permettono di costituire un'applicazione riutilizzando parti comuni. Lo sviluppatore può decidere quali componenti creare e come strutturarli oppure usarne alcuni già presenti all'interno di apposite librerie.

#### 4.3.2 React Router

React Router è una libreria di routing leggera e completa per React, ovvero permette di spostarsi dinamicamente tra le varie pagine dell'applicazione senza dover ogni volta ricaricare la pagina.

#### 4.3.3 Antd

Ant Design [1] è una libreria React UI dotata di molti componenti facili da usare ed utili per creare interfacce utente eleganti. Creato dal conglomerato cinese Alibaba, Ant Design è utilizzato da diversi grandi nomi: Alibaba (ovviamente), Tencent, Baidu e altri. Mentre Material-UI rimane la libreria React UI più popolare con oltre 40k stelle su Github, Ant Design è attualmente al secondo posto e sta rapidamente colmando il divario.

#### 4.3.4 Leaflet

Leaflet[4] è la principale libreria JavaScript open source per mappe interattive ottimizzate per dispositivi mobili. Veramente leggera, ha tutte le funzionalità di mappatura di cui la maggior parte degli sviluppatori ha bisogno.

È stata progettata pensando alla semplicità, alle prestazioni e all'usabilità. Funziona in modo efficiente su tutte le principali piattaforme desktop e mobili, può essere estesa con molti plug-in, ha delle API facili da usare e ben documentate e un codice sorgente semplice e leggibile a cui è un piacere contribuire.

#### 4.3.5 React-Leaflet

React Leaflet fornisce collegamenti tra React e Leaflet. Non sostituisce Leaflet, ma la sfrutta per astrarre i layer come componenti React.

#### 4.3.6 Font Awesome

Font Awesome è un toolkit di icone basato su CSS e Less.

### 4.3.7 Axios

Axios è un client HTTP promise-based per node.js e browser. È isomorfo (ovvero lo stesso codice può essere eseguito sia nel browser che su Node.js). Lato server utilizza il modulo http nativo di Node.js, mentre sul client (browser) utilizza XMLHttpRequests.

## 4.4 BackEnd

### 4.4.1 Node.js

Node.js [6] è un runtime Javascript costruito sul motore JavaScript V8 di Chrome. La sua forza risiede nell'utilizzo di un modello I/O non bloccante e ad eventi, che lo rende un framework leggero ed efficiente che punta ad ottimizzare grazie al suo design il Throughput e la scalabilità nelle applicazioni web con molte operazioni di input/output, risultando ottimo anche per sistemi real-time.

### 4.4.2 Express

Express è un web application framework server side, minimale e flessibile, per lo sviluppo in Node.js. Facilita l'uso di quest'ultimo, non oscurandone le funzionalità, rende più facile implementare API REST e supporta diversi template engine.

### 4.4.3 JSON Web Token (JWT)

JWT[3] è uno standard aperto ( RFC 7519 ) che definisce un modo compatto e autonomo per trasmettere in modo sicuro informazioni tra le parti come oggetto JSON. Queste informazioni possono essere verificate e attendibili perché firmate digitalmente. Nella sua forma compatta, i token Web JSON sono costituiti da tre parti separate da punti ( . ), che racchiudono Intestazione, Payload e firma.

### 4.4.4 Socket.io

Socket.IO[9] è una libreria che consente la comunicazione in tempo reale, bidirezionale e basata su eventi tra il browser e il server. Essa consiste in un server Node.js e una libreria client Javascript per il browser (che può essere eseguita anche da Node.js). Le caratteristiche che fanno il successo di Socket.io sono l'affidabilità, la possibilità di riconnessione automatica, il buffering dei pacchetti, gli acknowledgments, la trasmissione a tutti o a un sottosinsieme di client e il multiplexing.

### 4.4.5 SwaggerHub

SwaggerHub [10] è una piattaforma online per la progettazione e lo sviluppo di API REST che abbiamo utilizzato per documentare alcune delle API che

abbiamo sviluppato e che utilizziamo nella nostra applicazione. In particolare, tra le altre, abbiamo definito come effettuare le richieste quando si contatta un beacon del campus.

La documentazione è disponibile al seguente link:

<https://app.swaggerhub.com/apis/luzzo98/TalkingCampus/1.0.0#/>

#### 4.4.6 Postman

Postman [7] è un software che permette di testare le API che si sviluppano in maniera molto semplice e immediata, in particolare effettuando non solo richieste GET (eseguibili anche da browser) ma anche POST ecc. Nel nostro caso abbiamo utilizzato Postman per effettuare le richieste relative ai beacon simulati.

### 4.5 Database

#### 4.5.1 MongoDB

MongoDB [5] (da “humongous”, enorme) è un DBMS non relazionale, orientato ai documenti. Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l’integrazione di dati di alcuni tipi di applicazioni più facile e veloce.

#### 4.5.2 Mongoose

Mongoose è una libreria Object Data Modeling (ODM) per MongoDB e Node.js. Gestisce le relazioni tra i dati, fornisce la convalida dello schema e viene utilizzata per effettuare il mapping fra oggetti nel codice e la rappresentazione di tali oggetti in MongoDB.

# Capitolo 5

## Codice

### 5.1 Gestione Beacon

Abbiamo simulato cosa accadrebbe qualora un dispositivo mobile rilevi la presenza di un beacon nelle vicinanze. In un contesto reale, alla ricezione di un contatto bluetooth fra sensori e beacon, il client TalkingCampus notificherebbe il server per comunicare la posizione dell'utente e innescare gli aggiornamenti sulle frequenze nelle aule. Quello che abbiamo predisposto di questo scenario è la possibile richiesta “POST” che il server riceverebbe: non è stata prodotta per questo una risorsa beacon, ma si sono concatenate diverse risposte per le entrate e uscite dalle aule, provenienti dalle risorse messe in gioco da questa funzionalità.

```
import {Application} from "express";

module.exports = function setBeaconRoutes(app: Application) {

    const studentController = require("../Controllers/UsersController.ts")
    const roomController = require("../Controllers/RoomsController.ts");
    const notificationController =
        require("../Controllers/NotificationsController.ts")

    app.route("/api/entry-beacon")
        .post(studentController.findStudent,
              roomController.checkMaximumSeats,
              roomController.incrementSeats,
              notificationController.sendNotification
        );

    app.route("/api/exit-beacon")
        .post(studentController.findStudent,
              roomController.checkMinimumSeats,
```

```

        roomController.decrementSeats,
        notificationController.sendNotification
    );
}

```

## 5.2 Json Web Token (JWT)

Per gestire l'accesso all'applicativo si è deciso di rilasciare, qualora un utente effettui correttamente il login, un JWT della validità di due ore; questo verrà salvato insieme ad altre informazioni come email e ruolo nel Local Storage, ovvero un archivio dati chiave/valore disponibile nel browser. Salvare il JWT è indispensabile, infatti sarà richiesto in alcune operazioni dove si è scelto di richiedere una ulteriore livello di sicurezza. In queste situazioni il token viene inviato nell'header e poi verrà controllato dal server; se la verifica ha avuto successo viene controllato anche che l'email di chi ha effettuato la richiesta sia effettivamente la stessa presente come valore all'interno del token. Se il token non è valido, se è scaduto o se l'email non coincide, verrà restituito un errore.

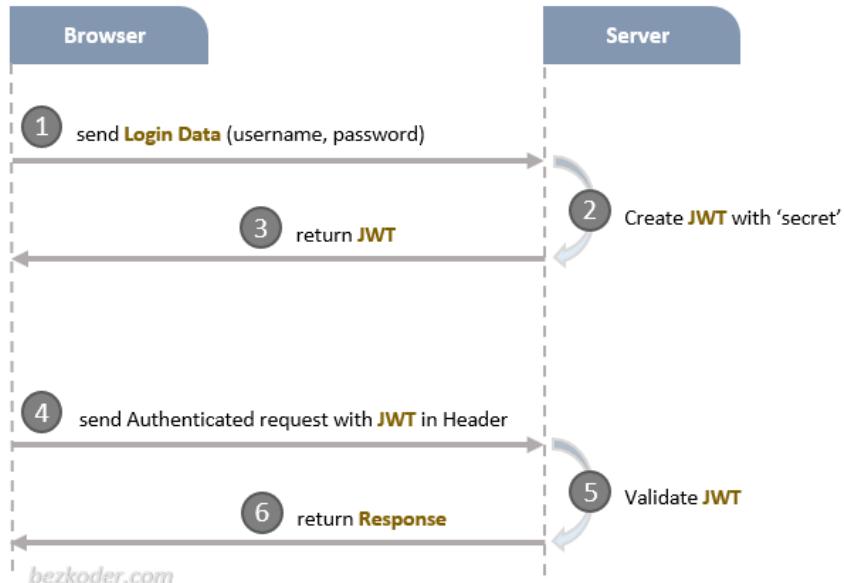


Figura 5.1: Flusso relativo all'utilizzo dei JWT

Si ricorda che l'intento dei JWT è quello di verificare che le informazioni siano generate da una fonte autenticata, infatti il server verifica che la firma del token sia stata generata con lo stesso algoritmo e chiave precedentemente

utilizzati. Utilizzare i JWT non fornisce la protezione dei dati in quanto non effettua una crittografia completa e non fornisce sicurezza per quanto riguarda gli attacchi di tipo “Man-in-the-middle” per i quali occorre comunque utilizzare https.

Creazione del token al login:

```
const {user} = require("../Model/User.ts");

exports.signin = function (req, res) {
    const jwt = require("jsonwebtoken");
    const PRIVATE_SECRET_KEY = "*****";

    user.findOne(req.body, function (err, user){
        if(err)
            res.send(err)

        const payload = user ? user.email : ""

        const token = jwt.sign({payload}, PRIVATE_SECRET_KEY, {
            algorithm: "HS512",
            expiresIn: "2h"
        });
        if (user) {
            res.send({
                email: user.email,
                name: user.name,
                role: user.role,
                picture: user.picture,
                accessToken: token
            })
        } else {
            res.status(404).send("Student not found")
        }
    })
};
```

Inserimento del token in una richiesta:

```
addCourse(course_id: string, teacher_id: string) {
    const user = JSON.parse(localStorage.getItem("user") as string);
    return axios
        .post(API_URL + "courses/addCourse", {
            course_id, teacher_id
        }, { headers: {
            "x-access-token": user.accessToken
        }
    })
}
```

```
)  
}
```

Controllo del token:

```
exports.checkToken = function(req, res, next) {  
    const jwt = require("jsonwebtoken");  
    const PRIVATE_SECRET_KEY = "*****";  
  
    const token = req.headers["x-access-token"];  
    if (token == null) {  
        res.sendStatus(403)  
    }  
    jwt.verify(token, PRIVATE_SECRET_KEY, (err, info) => {  
        if(err) {  
            res.sendStatus(403);  
        } else {  
            if (req.params.id == info.payload) {  
                next()  
            } else {  
                res.sendStatus(403);  
            }  
        }  
    })  
}
```

# **Capitolo 6**

## **Test**

Per valutare l'efficacia e la soddisfazione d'uso di Talking Campus abbiamo deciso di adottare diversi approcci di testing della Usability del sistema (coinvolgendo talora anche target users nelle nostre operazioni), allo scopo di sottoporre il nostro prodotto ad un percorso di monitoraggio e controllo standardizzato e in qualche modo certificante di un buon livello di qualità della nostra applicazione.

### **6.1 Valutazione euristica**

Abbiamo scelto di effettuare una valutazione euristica sull'interfaccia nella fase preliminare del design, prima di esaurire gli usability test che coinvolgono gli utenti reali. L'euristica è uno dei metodi più informali per effettuare una verifica dei requisiti di usabilità; ci siamo personalmente immedesimati nell'esperto che conduce la valutazione, che è stata portata avanti nel contesto di use cases (task), con lo scopo di ridurre il numero e la quantità degli errori. Non si sono rispettate tutte le euristiche di Nielsen, ma quante più possibili, poichè, trovandoci di fronte ad un progetto di dimensioni contenute, non tutte si adattavano al contesto (nello specifico abbiamo tralasciato “Flessibilità ed efficienza d'uso”, “Aiuto all'utente” e “Documentazione”).

1. **Visibilità dello stato del sistema:** Il sistema tiene sempre informato l'utente grazie a dei titoli, sottotitoli, testi che informano costantemente l'utente della porzione del sistema in cui si trova. Vengono, inoltre, forniti feedback tramite notifiche al completamento delle procedure.
2. **Corrispondenza tra sistema e mondo reale:** le parole utilizzate non appartengono ad un registro linguistico estraneo all'utente e vengono utilizzate icone riconoscibili per indicare i diversi locali.
3. **Controllo e libertà:** le procedure sono semplici, l'utente può muoversi liberamente nella mappa e nelle sezioni che lo riguardano direttamente del menù laterale.

4. Consistenza e standard: le schermate simili tra loro conferiscono all'utente la sensazione di essere sempre nello stesso ambiente.
5. Prevenzione dell'errore: si sono evitate ambiguità e mancanza di comprensione che possano portare ad errori; è sempre fornita la possibilità di tornare indietro, e per effettuare modifiche o eliminazioni viene richiesta la conferma.
6. Riconoscimento anziché ricordo: le schermate sono poche e layout semplici.
7. Design e estetica minimalista: è stata data più importanza al contenuto che all'estetica, utilizzando uno stile semplice che non distraga l'utente.

## 6.2 Cognitive Walkthrough

Durante il cognitive walkthrough abbiamo interpretato noi stessi gli esperti che effettuano la valutazione delle interfacce, nonostante vi possa essere un bias legato al fatto che siamo anche sviluppatori; per questo, abbiamo cercato di essere il più possibile imparziali, per simulare un'esperienza quanto più realistica, agendo come se l'applicazione, e quindi i relativi mockup, fossero a noi sconosciuti.

Ad ogni step ci siamo posti le seguenti domande:

- l'utente riuscirà a capire quali sono i sotto-task necessari a compiere un'azione?
- l'azione corretta è visibile nel momento in cui si è deciso di compierla?
- il feedback fornito dopo un'azione corretta sarà adeguato, in modo che l'utente capisca di aver raggiunto il proprio obiettivo?

Questi i task che abbiamo giudicato.

- Task generici
  - Registrazione
  - Login
- Profilo Studente
  - consulta locali su più piani
  - accede all'area personale, consulta notifiche e le gestisce
  - controlla situazione di un'aula, aggiungila fra le aule seguite e ottieni informazioni sul professore che sta svolgendo lezione
  - consulta le aule osservate ed eliminane una su cui non farà lezione
- Profilo Professore

- consulta l’orario delle lezioni di un corso insegnato e modificalo
- aggiungi un nuovo corso di insegnamento
- modifica l’orario di ricevimento
- Profilo Admin
  - aggiunge un locale
  - modifica un locale
  - elimina un locale

### 6.2.1 Login e Registrazione

Le schermate per il login e la registrazione risultano semplici da utilizzare, si capisce immediatamente quali dati sottomettere e includono operazioni già svolte da tutti gli utenti su altri siti senza particolari novità; non si evidenziano problemi di usabilità, cosa fondamentale visto che si tratta dei primi step da compiere per potere utilizzare il resto dell’applicazione.

### 6.2.2 Profilo: Studente

1. Consulta locali su più piani: le frecce sono visibili e suggeriscono la possibilità di navigare su più livelli all’interno dell’edificio. Manca un’indicazione del piano su cui ci si trova. I marker sulla mappa invitano l’utente al click per avere informazioni sui locali, che sono quindi in questo modo “visitabili”. Questi dettagli rendono il compito più semplice all’utente, che facilmente ottiene il risultato sperato e le informazioni che si aspetta.
2. Accede all’area personale, consulta notifiche e le gestisce: la schermata mostra immediatamente la sezione personale dell’utente e le label dei bottoni chiariscono il compito che svolgono. Il click sul pulsante “Area personale” apre la schermata con i dati che vengono forniti in maniera semplice e precisa, senza sconvolgere l’utente. La sezione “Notifiche”, a cui si accede analogamente, raggruppa gli avvisi ricevuti dall’utente in un elenco senza particolari dettagli fuorvianti; tali avvisi sono eliminabili in maniera altrettanto semplice con l’apposito pulsante. All’eliminazione, la notifica scompare semplicemente dall’elenco.
3. Controlla situazione di un’aula, aggiungila fra le aule seguite e ottieni informazioni sul professore che sta svolgendo lezione: i marker cliccabili non suggeriscono immediatamente il tipo di locale a cui fanno riferimento; si suggerisce l’utilizzo di un indicatore per facilitare il riconoscimento del tipo di stanza. Individuata un’aula, appare un pop-up con le informazioni che lo studente si aspetta di ricevere sulla lezione in svolgimento. Fra queste vi è il nome del professore che, se premuto, mostra i dati del docente (e-mail, telefono e orario di

ricevimento). Il passaggio non è immediato poichè non si fornisce nessun tipo di feedback che spinga l'utente a compiere tale azione. La registrazione dell'aula, invece, è molto intuitiva per via della presenza di un pulsante che fornisce un'indicazione chiara per il completamento dell'operazione, e un feedback esplicito dell'avvenuta registrazione.

4. Consulta le aule osservate ed eliminane una su cui non farà lezione: le operazioni che non coinvolgono direttamente la mappa passano dal menù a lato, in cui si trova il pulsante “Aule registrate”. Alla pressione si apre una finestra con le aule, da cui queste possono essere eliminate in maniera semplice, esattamente come per la sezione notifiche.

### 6.2.3 Profilo: Professore

1. Consulta l'orario delle lezioni di un corso insegnato e modificalo: dalla vista professore, il pulsante con il testo “Gestisci i tuoi corsi” potrebbe, in collaborazione con il pulsante “Area personale”, risultare ambiguo, in quanto si potrebbe pensare che le informazioni sulla didattica di un docente si trovino nella sua area personale, quindi si consiglia di rinominare area personale in “Informazioni personali”.
2. Il click su “Gestisci i tuoi corsi” apre una finestra in cui è chiara la divisione fra i vari corsi insegnati e le azioni che possono essere compiute: la modifica richiesta su un determinato corso si esegue facilmente attraverso l'apposito pulsante “modifica orario delle lezioni”. La schermata seguente fornisce un supporto visivo immediato ed efficace per la rimozione di un vecchio orario e laggiunta di uno nuovo (non è prevista una modifica diretta di un orario già presente). Anche dal mockup è evidente che in assenza di modifiche il pulsante “Salva” non sia cliccabile, mentre effettuandone almeno una è possibile eseguire il salvataggio, operazione di cui l'utente riceve un feedback, se terminata con esito positivo, tramite una notifica.
3. Aggiungi un nuovo corso di insegnamento: sempre dalla schermata precedente, è possibile, cliccando sul pulsante “aggiungi nuovo corso”, inserire il nome del nuovo corso e, eventualmente, aggiungere da subito gli orari delle lezioni, tramite un pulsante “definisci gli orari delle lezioni” che, vista la mancata implementazione nel mockup, si immagina innescchi le stesse operazioni del punto precedente. Il corso aggiunto viene visualizzato fra i corsi insegnati, in modo che l'utente capisca che le operazioni si sono concluse con successo.
4. Modifica l'orario di ricevimento: grazie al menù principale della vista è evidente come per effettuare questa operazione sia necessario cliccare sul pulsante “modifica orario di ricevimento”, che, premuto, consente di effettuare operazioni di modifica con le stesse logiche già viste per gli altri task. Il riutilizzo delle schermate agevola l'esperienza dell'utente,

che, una volta interagito con una di queste, è in grado di muoversi con sicurezza nelle altre.

#### 6.2.4 Profilo: Admin

Si suppone che l'admin sia un utente esperto del sistema, e quindi sappia già con certezza come muoversi all'interno dell'interfaccia, ma per completezza abbiamo deciso di analizzare le azioni che svolge per individuare i sottotask che le compongono, focalizzandoci meno su aspetti di usabilità.

1. Aggiunge un locale: anche senza una formazione particolare, è evidente qual'è il primo passo per creare un nuovo locale. Premendo il pulsante “aggiungi un locale” compare una finestra che permette l'inserimento dei dati relativi alla nuova stanza; alla pressione del pulsante “crea”, compare una finestra che invita l'utente a cliccare la posizione della mappa in cui vuole aggiungere il locale. Selezionato il punto, sulla mappa compare la nuova stanza.
2. Modificare un locale: la modifica del locale non parte analogamente da un pulsante, ma viene richiesto il click sul marker che si riferisce al locale che si vuole modificare. Successivamente, si presenta una finestra che permette di cambiare i dati relativi allo spazio selezionato (tipo di stanza, nome, posti disponibili). Il pulsante “salva” completa le operazioni, e viene fornito un feedback del fatto che le operazioni sono state completate con successo tramite una finestra di notifica.
3. Eliminare un locale: anche l'eliminazione comincia dalla pressione di un marker; infatti, nella stessa finestra in cui è possibile modificare le informazioni è presente un pulsante “elimina la stanza”. Prima di completare l'operazione è richiesta una conferma all'utente, che può scegliere se procedere effettivamente con l'eliminazione oppure annullare tutto. Come per la modifica, viene visualizzata una finestra di notifica che mostra che le operazioni sono avvenute con successo.

I principali problemi evidenziati dal Cognitive Walkthrough sono:

- Non è presente l'indicazione del piano su cui ci si trova.
- Non è stata prevista la distinzione dei marker in base alla tipologia dei locali.
- Non è suggerita in alcun modo la possibilità di cliccare il nome di un professore per avere ulteriori informazioni.
- Il pulsante “Area Personale” presenta un testo ambiguo.

Come risulta evidente, non si sono evidenziati grandi cambiamenti da apportare al sistema (dovuto anche al fatto che si tratta di un applicativo con

poche features) ma che comunque potrebbero impattare considerevolmente l'esperienza utente, visto che l'obiettivo di questa analisi era proprio quello di identificare eventuali problemi di usabilità.

Complessivamente, l'interfaccia e l'interazione sono risultate facili da apprendere, poiché i task che un nuovo utente potrebbe voler eseguire all'interno dell'applicativo sono stati il più possibile semplificati e guidati.

### 6.3 Co-discovery Learning

Abbiamo anche utilizzato l'approccio del co-discovery learning, una variante del “think aloud protocol” in cui gli utenti sono in coppia; è preferibile questa soluzione per agevolare i partecipanti ad effettuare considerazioni a proposito dell'interfaccia, in modo da far emergere i processi cognitivi dalla discussione instaurata tra i due partecipanti. Come consigliano le direttive di questo tipo di approccio la sessione di test è stata interamente video-registrata.

Ai partecipanti è stato fornito un mockup interattivo sviluppato con Adobe XD in cui potevano navigare abbastanza liberamente, al netto di alcune caselle di testo e pulsanti non funzionanti.

Sono stati scelti dei compiti che gli utenti dovranno svolgere per misurare l'usabilità dell'interfaccia, ovvero la lista di task che devono effettuare mentre sono in un ambiente controllato con un supervisore che osserva e raccoglie i feedback. I task sono divisi per scenari d'uso visto che ci sono diversi profili, quindi si dovranno immedesimare prima in un professore e poi in uno studente

#### 6.3.1 Profilo: Professore

- effettua l'iscrizione e inserisci il tuo orario di ricevimento e un corso che insegni
- naviga nell'interfaccia spostandoti di piano e apprendo e chiudendo il menù
- il prossimo anno avrai un nuovo corso da insegnare: inseriscilo
- il corso sistemi operativi invece non si terrà più: eliminalo
- stanno per arrivare le vacanze e non andrai in università per un po': elimina i tuoi orari di ricevimento

#### 6.3.2 Profilo: Studente

- effettua il login

- sei sicuro che le tue informazioni siano corrette? controlla la tua area personale
- avrai una lezione tra poco nell'aula 3.3: controlla lo stato attuale
- hai alcune domande da chiedere al prof X: controlla quando effettua il ricevimento per mandargli una mail
- andrai spesso nell'aula 3.3: inseriscila tra le aule seguite
- controlla le tue notifiche e, successivamente, eliminale
- il prossimo semestre non sarai mai nell'aula 3.3: eliminala dalle aule seguite

Una volta spiegata l'idea alla base del sito web “Talking Campus” e comunicato lo scenario per simulare una situazione realistica si è cercato di capire i problemi e i punti di forza dell'interfaccia.

La gestione dell'iscrizione è risultata molto intuitiva, e questo probabilmente è dovuto al fatto che entrambi i partecipanti hanno sicuramente effettuato molte iscrizioni su altri siti web; in particolare non hanno riscontrato nessuna difficoltà ad effettuare l'iscrizione di un professore (più complessa rispetto a quella dello studente). I partecipanti hanno espresso alcuni pareri interessanti riguardo il pulsante che permette la selezione dei giorni per quanto riguarda le lezioni e gli orari di ricevimento: anche se hanno riferito di aspettarsi una scelta tra i giorni della settimana (ad esclusione del sabato e della domenica) e non un vero e proprio calendario, hanno affermato che potrebbe essere una buona idea prevedere la possibilità di una doppia configurazione a seconda del semestre, nonché un sistema automatico che consideri i giorni di vacanza e le feste nazionali. Si sono mosse bene nell'interfaccia ed hanno tranquillamente compreso aspetti leggermente più complessi come la possibilità di inserire più corsi per ogni professore.

Appena arrivati alla schermata con la mappa hanno manifestato un certo senso di smarrimento non riuscendo a capire dove fosse l'ingresso e in quale piano si trovassero, hanno così affermato che servirebbe un testo che informi l'utente di questo aspetto; dopo aver compreso come muoversi tra i piani hanno compreso meglio come utilizzare la mappa anche se ancora insicure per alcuni aspetti come la scelta dei colori utilizzati nella mappa. I partecipanti, pur comprendendo la possibilità di accedere ad alcune risorse grazie al menù laterale, hanno riscontrato alcune problematicità con quest'ultimo, ad esempio non capendo bene come aprirlo e chiuderlo.

Per quanto riguarda la gestione e l'eliminazione di alcuni elementi si sono mossi con sicurezza, esprimendo però alcuni dubbi riguardo l'eventuale perdita del materiale, non sapendo dell'esistenza dell'applicativo “Virtuale”. I partecipanti, infatti, non si sentivano a loro agio ad eliminare definitivamente corsi e orari di ricevimento, tanto da proporre per questi ultimi la possibilità di sospenderli momentaneamente, ad esempio di un mese; un'ulteriore proposta è stata quella di inserire una nota visibile dagli studenti se un

professore che vogliono contattare non effettua ricevimenti in quel periodo. Nello scenario in cui si sono immedesimati come uno studente hanno concordato sulla mancanza di informazioni importanti nell'area personale, come la matricola, l'anno di corso, l'università frequentata e la media. I partecipanti hanno anche sottolineato come non fosse chiaro il significato di "registrarsi ad un'aula" senza una spiegazione adeguata ed hanno espresso anche la volontà di poter visualizzare una lista con tutti i professori e tutti i corsi dove poter controllare orari di ricevimento e delle lezioni. Si è anche riscontrato un problema riguardante le informazioni di un professore in un aula, in quanto non viene fornito nessun indizio che spinga l'utente a cliccare il suo nome per vederle, si sarebbero aspettati una sottolineature blu come avviene nei link. Sempre riguardo queste informazioni hanno discusso della possibilità di essere indirizzati ad una pagina Gmail se si clicca la email del professore. Come ultimo commento entrambi hanno riferito di apprezzare la funzionalità della registrazione all'aula in quanto semplice, rapida e molto utile.

Il tempo che hanno impiegato per eseguire i task spazia dai 20 secondi per le azioni più veloci ai 4 minuti per la registrazione del professore e tutti i task sono stati completati in modo corretto anche se a volte i partecipanti navigavano in altre zone prima di trovare la giusta schermata; gli errori hanno comunque una frequenza molto ridotta e si possono ignorare in quanto dopo aver compreso il funzionamento di una schermata non sono più stati commessi. Le valutazioni qualitative, invece, sono le seguenti: è stato decretato il successo generale del sistema che richiede uno sforzo contenuto per essere utilizzato, le varie azioni si portano a termine senza dover ricevere aiuto e, in generale, la qualità dell'interfaccia è stata considerata buona.

## 6.4 Usability Test

Abbiamo condotto un Usability Test: sono stati mostrati ai tester i mock-up dell'applicativo e ne è stato enunciato il funzionamento in tutti i suoi aspetti. Questo perché, nonostante il profilo del soggetto in questione si avvicini a quello dello studente universitario medio che decide di utilizzare Talking Campus, per motivi pratici e logistici questi è stato chiamato a svolgere i task per ogni tipologia di utilizzatore del sistema, adempiendo anche ai compiti propri del Professore e dell'Admin.

Questi, con precisione, i task che sono stati assegnati:

- Per ogni tipologia di utente:
  - Registrazione
  - Login
- in qualità di Studente:
  - consulta locali su più piani

- accedi all'area personale, consulta notifiche ed eliminale
- controlla l'orario della reception
- controlla l'orario di ricevimento di un prof x
- registrati ad un'aula ed eliminane una su cui non farai lezione
- in qualità di Professore:
  - consulta orario delle lezioni e modificalo
  - aggiungi un nuovo corso di insegnamento
  - modifica l'orario di ricevimento aggiungendone uno nuovo

I task di registrazione e login sono stati completati senza troppi indugi: il tester si è trovato a suo agio fra le schermate, ha capito immediatamente l'iter da seguire e ha completato senza problemi le operazioni.

Immedesimandosi in uno studente, questi ha poi svolto le operazioni che riguardano la visualizzazione delle info di ogni locale, attraverso i popup predisposti (task: “controlla l'orario della reception” e “consulta locali su più piani”), ma ha evidenziato la necessità di un meccanismo di segnalazione che utilizzi delle icone per permettere di individuare il tipo di locale e di un'ulteriore indicazione del piano per orientarsi agevolmente sui diversi livelli della mappa.

Registrazione ed eliminazione (task: “registrati ad un'aula ed eliminane una su cui non farai lezione”) di aule dalle quali un ipotetico utente potrebbe essere intenzionato o meno a ricevere aggiornamenti sono state operazioni che non hanno creato particolari grattacapi al performer, cosa che non è valsa per la scoperta delle informazioni legate ad un professore (orario di ricevimento) con una lezione in svolgimento all'interno di una determinata aula (task: “controlla l'orario di ricevimento di un prof x”): è stata infatti denunciata una difficoltà nel comprendere come poter ottenere tali informazioni, data l'assenza di un pulsante o di un qualsiasi meccanismo che suggerisse il passo da svolgere. Nei panni di un professore, il tester non ha avuto grossi problemi: la semplicità dei menù ha reso semplice comprendere il percorso per completare i task e questo lo ha messo a suo agio con l'applicativo, per il quale ne è stata apprezzata, per questa particolare sezione, la facilità di utilizzo.

# Capitolo 7

# Deployment

Abbiamo deciso di effettuare il deploy di Talking Campus sia utilizzando il gestore di container Docker[2] e la sua estensione Docker-compose, sia alcuni script da lanciare nel singolo host principale.

Quest'ultimo approccio riguarda l'ambiente Windows, in cui i task da eseguire per lanciare correttamente Talking Campus sono:

1. inizializzare il database Mongo, lanciando lo script contenuto nella folder “mongo\_scripts”, in una shell Mongo
2. lanciare, dopo aver aperto una prima shell e essersi posizionati all'interno della sotto-directory “frontend”:
  - (a) “npm install” per installare le dipendenze
  - (b) “npm run frontend-start” per eseguire il frontend dell'applicazione
3. lanciare, invece, in un'altra shell, dalla sotto-directory “backend”:
  - (a) “npm install” per installare le dipendenze
  - (b) “npm run backend-start” per eseguire il backend dell'applicazione

Per quanto riguarda invece l'utilizzo di Docker, per backend e frontend è stato creato il file di istruzioni per creare l'immagine e grazie a Docker-compose è stato possibile avviare tutti i container in maniera centralizzata all'interno di un unico stack applicativo.

Questi dunque i comandi da lanciare nella directory del progetto:

1. docker-compose down -v (per eliminare qualunque versione datata, rimuovendo anche eventuali volumi)
2. docker-compose up –build -d (per buildare il progetto e lanciare l'applicazione in modalità “detached”)

# Capitolo 8

# Conclusioni

## 8.1 Possibili miglioramenti futuri

Sicuramente il progetto ha spazio per migliorare: in primis, come già analizzato in sede di focus group, si potrebbe accedere al sistema usufruendo delle credenziali in uso presso Virtuale, Studenti Online, Almaesami e gli altri applicativi dell'università. Altra possibile miglioria potrebbe essere storicizzare i dati riguardo le presenze nei locali, in modo da poter produrre nuove statistiche sugl'orari di maggior frequenza degl'ambienti e così informare gli studenti su quali potrebbero essere i momenti migliori in cui prendere posto nelle aule o utilizzare i servizi igienici evitando attese. Si potrebbe poi dotare il sistema di servizi di prenotazione di vari aspetti legati a Talking Campus (come ad esempio ricevimenti con i professori e posti nelle aule studio) e predisporre di conseguenza notifiche e avvisi adeguati alle circostanze.

## 8.2 Commenti finali

Questo progetto è stata un'occasione importante per approfondire la conoscenza di un insieme di tecnologie che hanno un ampio impiego professionale nell'ambito della programmazione web odierna, in particolare gli aspetti principali dello sviluppo fullstack in Javascript. Durante i lavori abbiamo utilizzato Git come strumento di versionamento del codice e Docker per la distribuzione del sistema, cercando il più possibile di emulare quanto avviene nelle aziende che creano web application come Talking Campus. Pertanto, ci riteniamo soddisfatti di questa esperienza, poichè ha contribuito ad arricchire notevolmente il nostro bagaglio personale di conoscenze.

# Bibliografia

- [1] ANTD - <https://ant.design/index-cn>.
- [2] Docker - <https://www.docker.com/>.
- [3] JSON Web Token - <https://jwt.io/>.
- [4] Leaflet - <https://leafletjs.com/>.
- [5] MongoDB - <https://www.mongodb.com/it-it>.
- [6] Node - <https://nodejs.org/it/>.
- [7] Postman - <https://www.postman.com/>.
- [8] React - <https://it.reactjs.org/>.
- [9] SocketIO - <https://socket.io/>.
- [10] SwaggerHub - <https://swagger.io/tools/swaggerhub/>.
- [11] Typescript - <https://www.typescriptlang.org/>.