

Boardify

Applicazioni e Servizi Web

Tommaso Mandoloni - 0000982426
Alessandro Marcantoni - 0000980506
Simone Romagnoli - 0000995763

28 giugno 2021

1 Introduzione

Boardify è una piattaforma pensata per organizzare il lavoro di un team, in particolare per migliorarne la coordinazione e ordinare al meglio lo svolgimento dei compiti in corso d'opera.

L'idea si basa sulla modalità di lavoro agile, ormai utilizzata su larga scala nell'ambito dell'ingegneria del software: nello specifico, ci si è ispirati alla metodologia di sviluppo *Scrum* che definisce una strategia di sviluppo flessibile dove il team lavora come un'unica entità per raggiungere un obiettivo comune.

La piattaforma prende ispirazione anche dal celebre *GitHub* per quanto riguarda la gestione dei profili: ci sono poche informazioni personali per dare più peso ai "progetti" a cui si partecipa, in questo caso, le bacheche.

Una bacheca è costituita da uno schema in cui le righe sono rappresentate dalle persone e le colonne indicano dei compiti da svolgere: questa suddivisione permette di indicare in ogni casella il carico di lavoro progressivo nello svolgimento di ogni task; inoltre, permette una miglior coordinazione nella visione dell'andamento complessivo del lavoro e nell'individuazione dei task futuri.

2 Requisiti

L'applicazione web proposta ha come obiettivo quello di permettere ad un team di organizzarsi, attraverso una bacheca condivisa e appuntando man mano il proprio progresso all'interno di un progetto/lavoro. Si prevedono le seguenti pagine:

- **Autenticazione** - pagina che permette di registrarsi inserendo dati personali o di effettuare l'accesso al proprio spazio personale;
- **Home** - permette di visualizzare il proprio spazio contenente dati personali e una dashboard con le proprie bacheche; inoltre, vi è una sezione "notifiche": quando una persona viene aggiunta ad una bacheca deve accettare l'invito che verrà notificato nella propria home ed è possibile farlo in due modi, accettando la notifica o scannerizzando un QR code appositamente generato.
- **Bacheca** - permette di visualizzare uno specifico progetto in corso: una bacheca viene creata da una persona chiamata "admin" che può modificarla aggiungendo persone, togliendole e aggiungendo o togliendo task da svolgere; inoltre, opzionalmente, sarà possibile creare dei gruppi interni per suddividere meglio il lavoro;
- **Aggiunta Bacheca** - permette di creare una nuova bacheca definendone nome, partecipanti e task (che possono essere anche aggiunti a posteriori).

3 Design

Per quanto riguarda il design dell'architettura del sistema, l'obiettivo principale fin dalle prime fasi è stato quello di evitare il più possibile di accumulare debito tecnico. Prevedendo infatti di adottare un approccio di design partecipativo, un'architettura modulare e facilmente adattabile ci avrebbe permesso di modificare il sistema in modo agile rispetto ai cambiamenti proposti dagli utenti coinvolti. In particolare, ciò si è rivelato molto utile per confermare le funzionalità già presenti al momento della presentazione delle specifiche e anche per aggiungerne di nuove: un esempio è la possibilità di generare un codice QR con il quale poter accedere ad una bacheca (suggeritaci da un collega universitario). Volendo creare un'architettura adattabile a qualsiasi moderno framework lato client, la prima fase di progettazione è stata incentrata su quelle parti del sistema meno soggette a cambiamenti futuri. Nello specifico, inizialmente è stata decisa la struttura dei documenti all'interno del database (listati 1, 2, 3) per poi passare alla scelta delle API esposte dal lato server; queste ultime sono state raccolte all'interno di una specifica *Swagger* reperibile a swagger.com/boardify, come mostrato in figura 1.

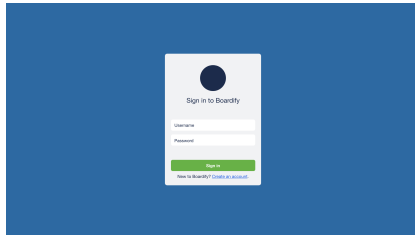
user Operations on users		^
GET	/users/{username} Checks if the user exists	✓ ↗
PUT	/users/{username} Updates the details of a user	✓ ↗
POST	/signup Registers a new user	✓ ↗
POST	/signin Login	✓ ↗
board Operations on boards		^
GET	/board/{owner}/{title} Returns the board identified by the owner and the title	✓ ↗
GET	/board/{owner}/{title}/qr Downloads a QRCode that allows users to join the specified board	✓ ↗
PUT	/board/{owner}/{title}/newTask Adds a new task to the board	✓ ↗
PUT	/board/{owner}/{title}/newTopic Adds a new topic to the board	✓ ↗
GET	/board/{owner}/{title}/addSessionUser Adds the current user to the specified board	✓ ↗
PUT	/board/{owner}/{title}/newUsers Adds the current user to the specified board	✓ ↗
PUT	/board/{owner}/{title}/assign Assigns a task of the specified board to a user	✓ ↗
PUT	/board/{owner}/{title}/taskState Updates the state of a task (to the next step)	✓ ↗
PUT	/board/{owner}/{title}/removeUser Removes a user from the board	✓ ↗
PUT	/board/{owner}/{title}/deleteTask Deletes a task from the board	✓ ↗
PUT	/board/{owner}/{title}/comment Adds or updated the comment to a task	✓ ↗
PUT	/board/{owner}/{title}/removeTask Removes a task from the user who is taking care of it	✓ ↗
GET	/board/{owner}/{title}/get/{task} Returns the specified task	✓ ↗
GET	/projects Returns all the boards for the current user	✓ ↗
POST	/project Creates a new board	✓ ↗
notification Operations on notifications		^
GET	/notification Gets all notifications for the logged user	✓ ↗
PUT	/notification Reads all the notifications for the specified board	✓ ↗
POST	/notification Adds a new notification	✓ ↗

Figura 1: API Swagger del sistema

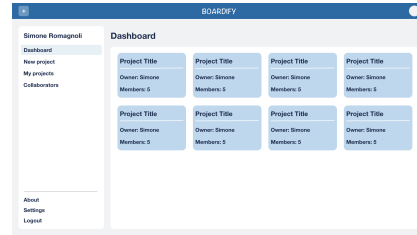
Per quanto riguarda le interfacce, è stato adottato un approccio progettuale basato su *mockup*. Questi sono stati realizzati con Adobe XD in quanto ritenuto il più professionale tra gli strumenti suggeriti. Inoltre, l'utilizzo di tale piattaforma ha permesso di realizzare in modo agevole una *storyboard* che è stata utile per simulare un primo modello di interazione con l'utente.

Inoltre, è stata condotta un'intervista a diversi utenti dalla quale è emerso che la piattaforma di utilizzo preferita sarebbe stata quella desktop: siccome tale pensiero era diffuso già nelle prime fasi di sviluppo, il design del sistema non

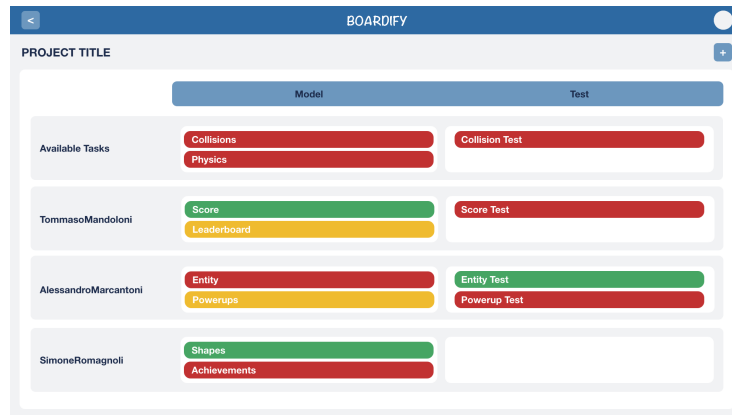
è stato realizzato con filosofia *mobile-first*. Tuttavia, le interfacce sono state rese responsive per garantire anche un utilizzo da sistemi mobili, prediligendo la modalità *landscape*.



(a) Mockup di signin



(b) Mockup della home



(c) Mockup della bacheca


Figura 2: Mockup del sistema

I mockup in figura 2 rappresentano quindi un'idea iniziale delle interfacce del sistema che sono state man mano raffinate grazie ai consigli forniti dagli utenti e considerando aspetti di user experience. Alcuni esempi di modifiche sostanziali sono:

- la sostituzione della **sidebar** con una **navbar**: ciò ha permesso di avere a disposizione più spazio orizzontale per la visualizzazione delle bacheche;
- è stata cambiata la palette di colori principali delle interfacce: ciò ha favorito l'adozione di colori meno invasivi garantendo una migliore usabilità ed accessibilità;
- sono stati aggiunti grafici che permettono di capire a colpo d'occhio l'andamento dei task all'interno di una o più bacheche;

- la pagina della bacheca è stata quasi completamente ridisegnata riducendo le aree colorate e rendendola più intuitiva;

Alla luce dei cambiamenti appena descritti, si riportano di seguito le interfacce definitive del sistema:




Sign in to Boardify

[Sign in](#)

Don't have an account yet? [Sign up](#)

Figura 3: Form del login



Create your account

[Sign up](#)

Already have an account? [Sign in](#)

Figura 4: Form della registrazione

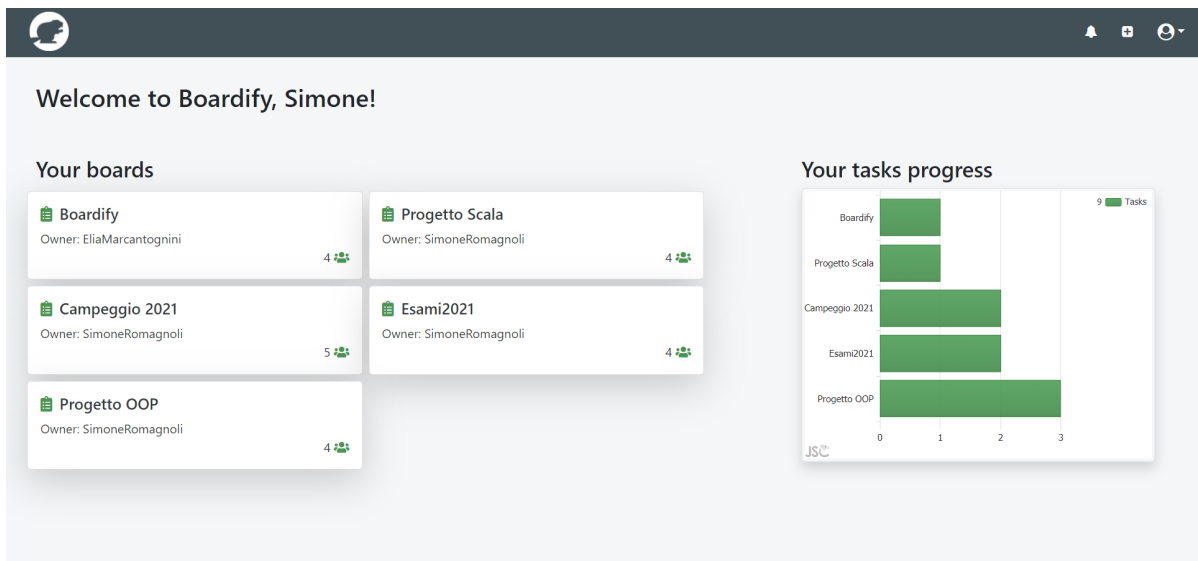


Figura 5: Interfaccia della home

The screenshot shows the "Create a new project" form. It features a central form box with the following fields and controls:

- Title**: A text input field.
- Description**: A text area with a small icon in the bottom right corner.
- Member**: A text input field with a green "+" button to its right.
- Topic**: A text input field with a green "+" button to its right.
- Create**: A large green button at the bottom of the form.

Figura 6: Interfaccia della creazione di una bacheca

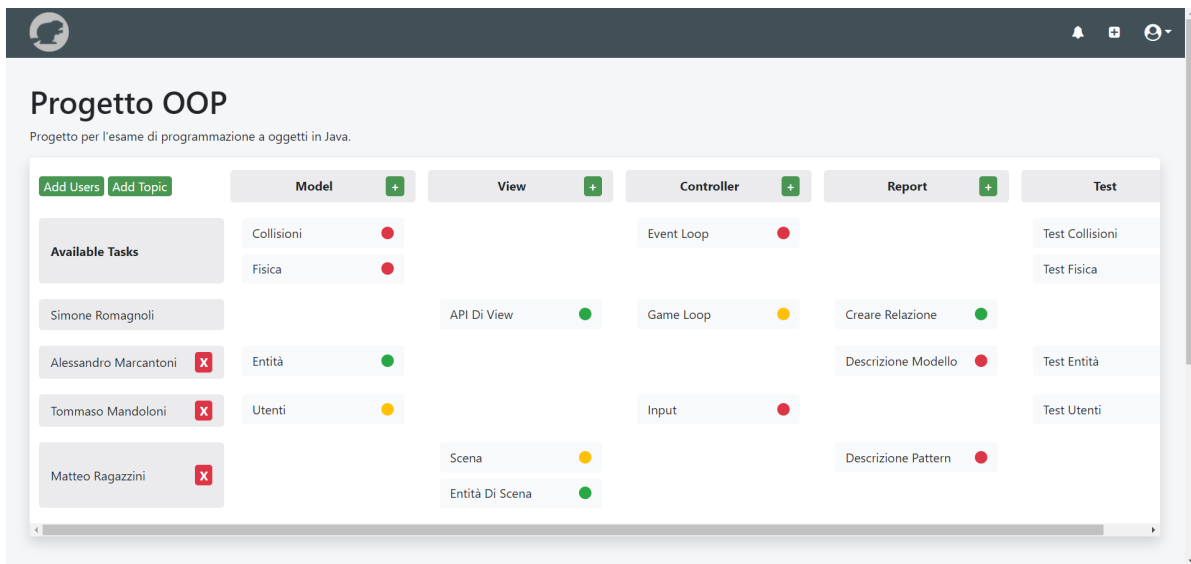


Figura 7: Interfaccia di visualizzazione di una bacheca

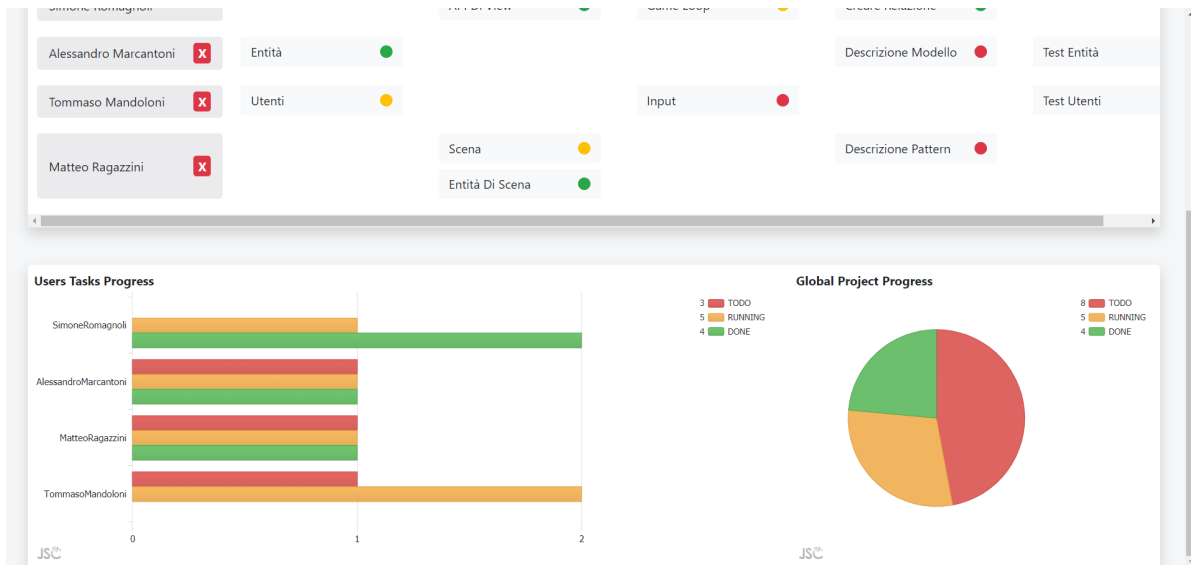


Figura 8: Grafici sottostanti alle bacheche

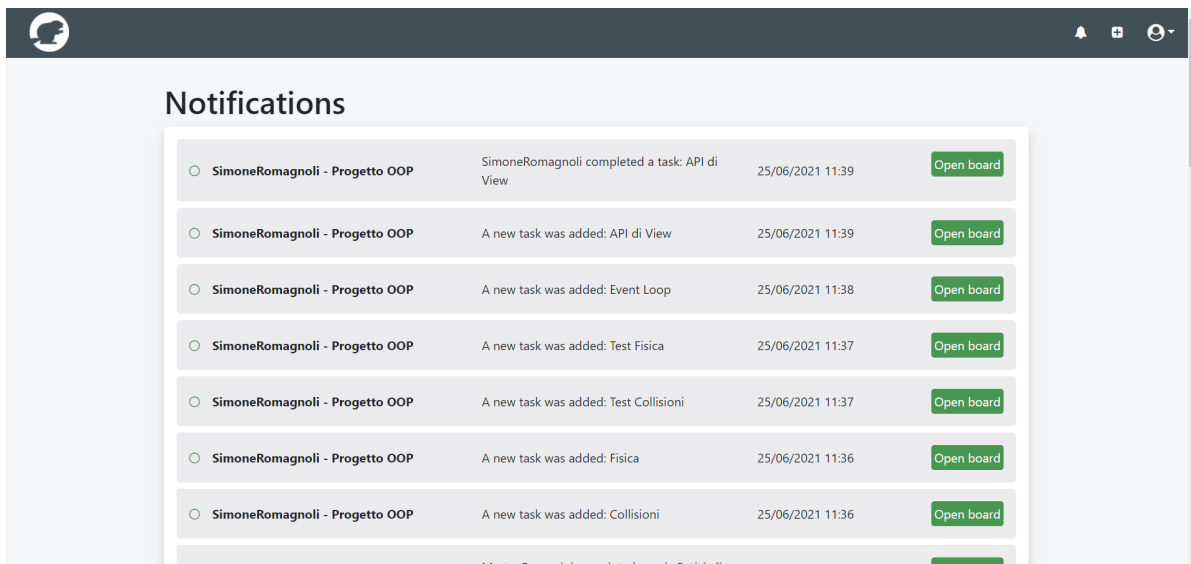


Figura 9: Interfaccia delle notifiche

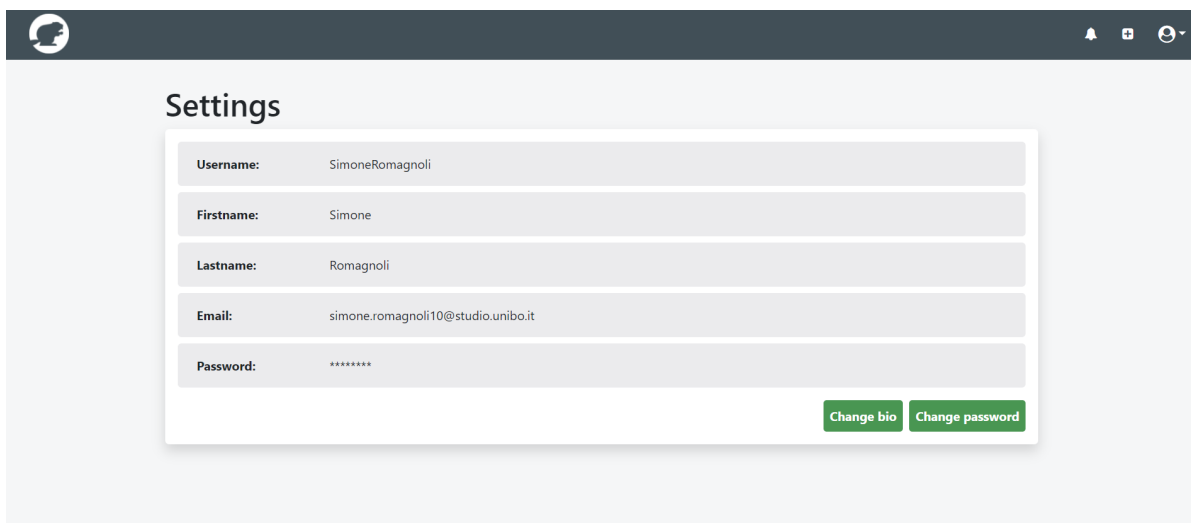


Figura 10: Interfaccia dei settings

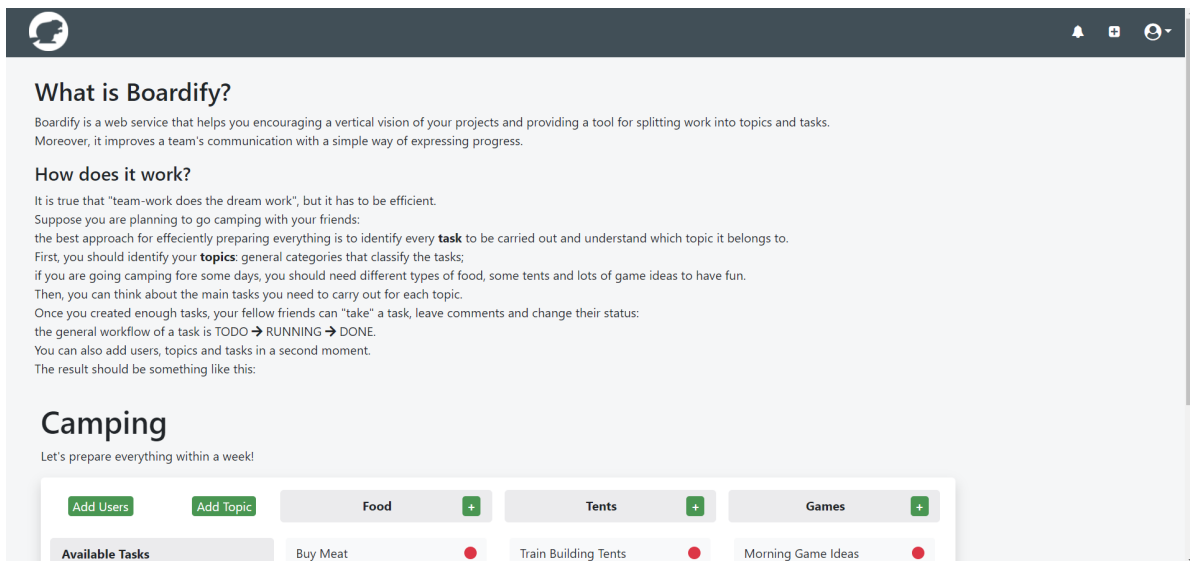


Figura 11: Interfaccia delle informazioni generali sull'applicazione

4 Tecnologie

Per la realizzazione del sistema sono state utilizzate diverse tecnologie; in particolare, si elencano di seguito quelle ritenute più importanti per quanto riguarda l'aspetto architetturale:

- Stack *MEVN*: come da specifica d'elaborato sono stati utilizzati *MongoDB* come piattaforma database, *Node.js* e, nello specifico, *Express* come framework lato server; a questi abbiamo integrato il framework lato client *Vue.js* per questioni di leggerezza, comodità e facilità d'uso.
- Bootstrap: che ha semplificato di molto lo sviluppo di una interfaccia grafica intuitiva e user-friendly.
- Socket.io: una libreria di *Node.js* per la gestione di notifiche push che ha permesso di evitare un'architettura a polling in favore di una ad eventi.
- Passport: una libreria di *Node.js* utilizzata per effettuare autenticazione all'interno del sistema.

Altre librerie che sono state utilizzate all'interno del sistema sono: *Sass*, *jQuery*, *JSCharting*, *Bcrypt* e *Qrcode*.

5 Codice

In questa sezione vengono riportati alcuni snippet di codice ritenuti rilevanti per meglio comprendere il design del sistema. Nello specifico, si elencano i modelli

delle collezioni dati di *MongoDB* e la strategia di autenticazione con libreria *Passport*.

Listing 1: Struttura del documento di una bacheca

```
const board_schema = mongoose.Schema({
  title: String,
  owner: String,
  description: String,
  topics: [String],
  members: [String],
  tasks: [{
    name: String,
    description: String,
    topic: String,
    user: String,
    state: String,
    comment: String
  }]
})
```

Listing 2: Struttura del documento di un utente

```
const user_schema = mongoose.Schema({
  username: String,
  firstname: String,
  lastname: String,
  email: String,
  password: String
})
```

Listing 3: Struttura del documento di una notifica

```
const notification_schema = mongoose.Schema({
  to: [{
    user: String,
    read: Boolean
  }],
  project: {
    title: String,
    owner: String
  },
  message: String,
  object: String,
  url: String,
  date: Date
});
```

Listing 4: Autenticazione con Passport e Bcrypt

```
passport.use(
  new LocalStrategy({
    usernameField: "username",
    passwordField: "password"
  }, (username, password, done) => {
    PassportUser.find({ username:username }, (err, doc) => {
      if (err) { return done(err); }
      else if (doc.length <= 0) {
        return done(null, false, { message: "Incorrect username"
        });
      } else if (!bcrypt.compareSync(password, doc[0].password)) {
        return done(null, false, { message: "Incorrect password"
        })
      } else { return done(null, doc[0]) }
    })
  })
)
```

6 Test

Per quanto riguarda il test del codice, tutte le API esposte dal lato server sono state testate mediante l'applicazione *Postman*: questo ha permesso di controllare che il comportamento del sistema fosse sempre coerente anche nelle prime fasi in cui non erano state sviluppate le corrispondenti interfacce grafiche. L'usabilità del sistema è stata valutata durante lo sviluppo dello stesso con due metodi:

- **Valutazione euristica:** si è tenuto conto della percezione e dell'esperienza degli utenti modificando elementi grafici per favorire l'intuitività delle interfacce. L'attenzione è stata posta principalmente sul minimalismo estetico delle componenti grafiche (non vi sono elementi superflui o puramente decorativi ma solo funzionali), sull'utilizzo di simboli ed icone le cui funzionalità siano subito chiare all'utente (come l'utilizzo di "+" per l'aggiunta e di "x" per la rimozione di elementi) e sulla consistenza in modo che l'utente acquisisca velocemente familiarità con l'ambiente grafico (la navbar costituisce un punto di riferimento all'interno ogni pagina e permette sempre di ritornare al proprio spazio personale).
- **Usability test:** in diverse fasi dello sviluppo sono state condotte delle verifiche che hanno coinvolto persone reali per avere più feedback possibili e capire come migliorare l'interazione uomo-macchina. Durante i test è stato richiesto agli utenti di portare a termine determinati task (creazione di un account, creazione di una bacheca, aggiunta di task alla bacheca, etc.) e, dopo averne osservato il comportamento, sono stati chiesti lo-

ro dei pareri riguardo all'esperienza e consigli su come poter migliorare l'interazione.

7 Deployment

L'applicazione è ospitata all'interno di un Raspberry Pi che funziona da server remoto, raggiungibile in qualsiasi momento. Questo ci ha permesso di testare il sistema di notifiche push, realizzate con architettura ad eventi, in modo più semplice. In particolare, il sistema è stato containerizzato con *Docker* per garantire maggiore interoperabilità e indipendenza dalla piattaforma di sviluppo (oltre a quella già offerta da *Node.js*).

8 Conclusioni

Le abilità sviluppate durante la realizzazione dell'elaborato riflettono gli obiettivi didattici del corso sostenuto: non solo si è stati capaci di apprendere dei nuovi framework e delle nuove tecnologie, ma si è anche stati in grado di progettare un sistema complesso sfruttando vecchie e nuove conoscenze.

In generale, l'applicazione realizzata rispecchia le aspettative progettuali iniziali ed è motivo di soddisfazione sia per quanto riguarda il risultato ottenuto sia per il processo di sviluppo, portato avanti in maniera agile.

Si ritiene inoltre che questa esperienza costituisca un'ottima base per progetti futuri in ambito di applicazioni web, facilmente spendibile anche e soprattutto in ambito lavorativo.

Per ulteriori informazioni il repository dell'elaborato è disponibile al link github.com/boardify.