WorkMatch

Una Piattaforma per il Collegamento di Aziende e Studenti Post-Diploma

Elaborato di Applicazioni e Servizi Web

Veronika Folin

0001026205 - veronika.folin@studio.unibo.it

Gennaio 2024

L'applicazione WorkMatch ha l'obiettivo di mettere in contatto aziende e neo-diplomati, in particolare, degli istituti tecnici o professionali che non hanno intenzione di proseguire gli studi e si stanno affacciando per la prima volta al mondo del lavoro. Il target di riferimento per questa nuova applicazione sono quelle aziende che hanno difficoltà nel reperire nuove figure professionali e vogliono investire sulla formazione dei giovani. D'altra parte, gli studenti avranno auspicabilmente più possibilità di trovare un'occupazione che possa corrispondere alle conoscenze acquisite durante il loro percorso formativo e che possa accrescere il loro curriculum di competenze.

Contents

List of Figures

1 Requisiti

Per determinare le specifiche dell'applicazione WorkMatch sono stati analizzati i requisiti incrementalmente e sotto vari punti di vista.

1.1 Requisiti di Business

L'idea di WorkMatch nasce dalla necessità di creare una piattaforma che permetta alle aziende di reperire nuove giovani figure, da inserire in un percorso, prima di formazione e poi lavorativo. Tali figure non devono necessariamente rappresentare persone con esperienze pregresse, in quanto verranno direttamente istruite dall'azienda che le assumerà.

Dall'altra parte, viene riconosciuto che nel mondo delle scuole superiori non esiste un vero e proprio sistema che faciliti l'inserimento nel mondo del lavoro dei neo-diplomati, come invece accade nel contesto universitario (e.g., AlmaLaurea).

1.2 Personas

Al fine di delineare nello specifico i requisiti, si è deciso di simulare delle Personas che siano rappresentative del target che si vuole raggiungere con il sistema che si andrà a sviluppare.

Elisa

- Nome: Elisa.
- Età: 19 anni.
- Impiego: Studentessa neo-diplomata.
- Contesto: Elisa si è appena diplomata all'Istituto Tecnico commerciale ed è alla ricerca del suo primo impiego. Il suo desiderio sarebbe quello di fare esperienza in uno studio commerciale, ma la ricerca attraverso gli uffici per l'impiego non ha ancora avuto esito positivo. Per il momento, si mantiene in contatto con il suo docente di Economia Aziendale e, sporadicamente, riceve una mail da quest'ultimo quando si presenta qualche opportunità, ma i destinatari come lei sono tanti ed Elisa non è ancora riuscita a trovare l'occupazione a cui ambisce. Sarebbe disposta anche a spostarsi fuori città e per questo vorrebbe avere una visione più ampia dell'offerte presenti sul territorio.

Lorenzo

- Nome: Lorenzo.
- Età: 38 anni.
- Impiego: Imprenditore.

- Contesto: L'azienda di Lorenzo è da poco entrata nel mercato internazionale e, per questo, sta subendo una forte crescita. Tuttavia, l'ufficio delle risorse umane ha difficoltà a reperire nuove figure qualificate, dunque decide di offrire in prima persona dei corsi di formazione col fine di selezionare nuovo personale. L'idea sarebbe quella di proporre questa iniziativa su una piattaforma online, dove è possibile raggiungere la maggior parte dei giovani che ancora non sono entrati nel mondo del lavoro.

1.3 Requisiti Funzionali

Effettuata l'analisi del contesto definito dalle Personas, si sono identificati i seguenti requisiti funzionali che l'applicativo dovrà soddisfare.

Ad ogni tipologia d'utente dovrà essere permesso di:

- Registrare un account fornendo un indirizzo mail valido e una password.
- Accedere all'account mediante mail e password precedentemente scelte.
- Visualizzare il proprio profilo personale.
- Modificare i dati inseriti nel profilo personale.
- Caricare un'immagine per modificare l'immagine di profilo pre-impostata.
- Visualizzare uno storico delle notifiche ricevute.
- Visualizzare ulteriori informazioni sul mittente della notifica ricevuta.
- Segnare come letta una notifica ricevuta.
- Effettuare log out.
- Eliminare il proprio account.

In aggiunta, vengono identificati i seguenti requisiti specifici.

Allo **studente** dovrà essere permesso di:

- In fase di registrazione, fornire alcuni dati anagrafici.
- Visualizzare le offerte di lavoro pubblicate dalle aziende e il dettaglio dei rispettivi profili.
- Inviare una candidatura rispetto a una determinata offerta di lavoro.
- Ricevere una notifica quando un'azienda esprime un interesse per il suo profilo presente sulla piattaforma.
- Opzionalmente:

  - Filtrare le offerte di lavoro (e.g., in base alla località).

Al **referente aziendale** dovrà essere permesso di:

- In fase di registrazione, fornire alcuni dati anagrafici e una breve descrizione dell'azienda.
- Inserire offerte di lavoro tramite un form pre-impostato.
- Visualizzare l'elenco delle offerte di lavoro inserite sulla piattaforma.
- Eliminare una determinata offerta di lavoro.
- Ricevere una notifica quando uno studente si candida per un'offerta di lavoro.
- Visualizzare l'elenco degli studenti e il dettaglio dei rispettivi profili.
- Inviare una notifica a uno specifico studente per segnalare un eventuale interesse.
- Opzionalmente:
  - Modificare una determinata offerta di lavoro.
  - Filtrare l'elenco degli studenti (e.g., in base allo specifico indirizzo di studio).

Inoltre, lo **stato** visualizzato dagli utenti deve essere **aggiornato e consistente**:

- Gli elenchi degli studenti visualizzati dal referente aziendale devono contenere solo i riferimenti a profili ancora attivi all'interno della piattaforma.
- Le offerte di lavoro relative a un account aziendale che si è cancellato dalla piattaforma non potranno più essere visualizzabili.
- Le notifiche inviate da un account non più presente sulla piattaforma dovranno essere rimosse dal database, in quanto ogni eventuale interesse precedentemente espresso non viene più considerato valido.

1.4 Requisiti non Funzionali

Di seguito sono descritti i requisiti non funzionali del sistema:

- **Usabilità**: l'interfaccia grafica dovrà essere semplice e intuitiva, così da permettere a un utente non esperto del dominio di comprendere quali sono le principali operazioni che si possono eseguire.
- **Reattività** dell'interfaccia utente rispetto alle azioni di quest'ultimo.
- **Sicurezza**: è necessario salvare in modo sicuro la password dell'utente mediante un meccanismo di crittografia.

2 Design

In fase di progettazione è stato realizzato il design del sistema e dell'interfaccia utente dell'applicazione precedentemente descritta.

2.1 Architettura del Sistema

Per la progettazione del sistema, si è deciso che WorkMatch verrà sviluppata come **single-page application (SPA)**: invece di effettuare il rendering di pagine separate per ogni azione dell'utente, una SPA carica una singola pagina all'inizio e successivamente aggiorna dinamicamente il contenuto a ogni interazione dell'utente.

Lo sviluppo seguirà il pattern architetturale **Model-View-ViewModel (MVVM)**:

- il *Model* rappresenta i dati dell'applicazione.
- la *View* si focalizza sulla presentazione dei dati provenienti dal ViewModel e sulla gestione degli eventi generati dall'interazione dell'utente con l'interfaccia.
- il *ViewModel* è il collegamento tra la View e il Model; facilita la separazione della logica di visualizzazione dalla logica di business garantendo reattività della gestione dello stato dell'applicazione (e.g., attraverso il binding dei dati).

2.2 Interfaccia Utente

Per la progettazione dell'interfaccia utente si è deciso di seguire un approccio **mobile-first**, favorendo lo sviluppo di un'interfaccia utente ottimizzata per i dispositivi mobili, come smartphone e tablet, anziché per i dispositivi desktop. Tale scelta è guidata dall'utilizzo pervasivo di dispositivi con diverse limitazioni, come la grandezza dello schermo e la navigazione touch e dal tipo di target dell'applicativo WorkMatch (i.e., studenti che, nella quotidianità, utilizzano prevalentemente il cellulare). Successivamente il design dell'interfaccia è stato adattato ai dispositivi di tipo desktop.

Al fine di definire l'aspetto dell'interfaccia e l'interazione dell'utente, sono stati realizzati i mockup riportati di seguito.

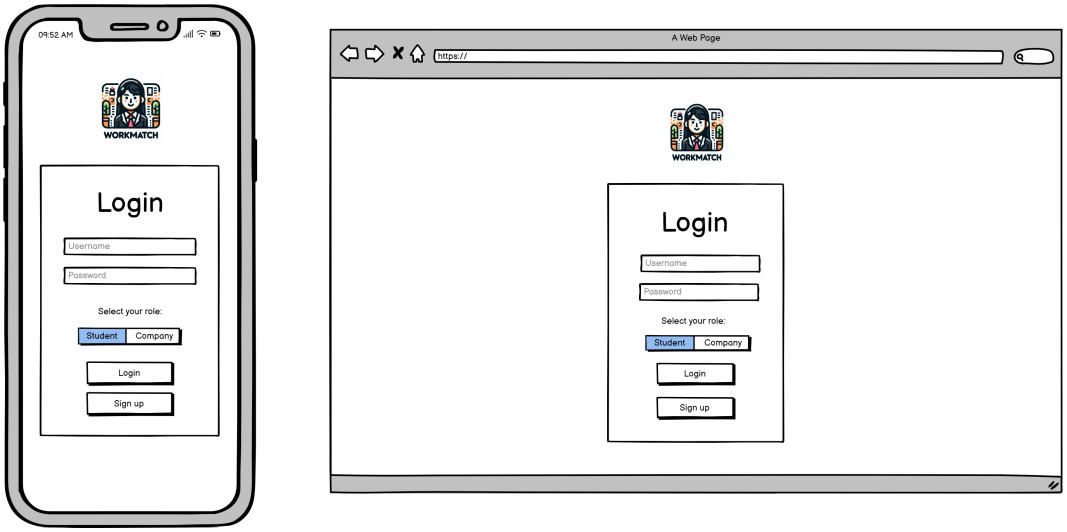

Figure 1: Pagina di login.

Figure 2: Pagina di registrazione.

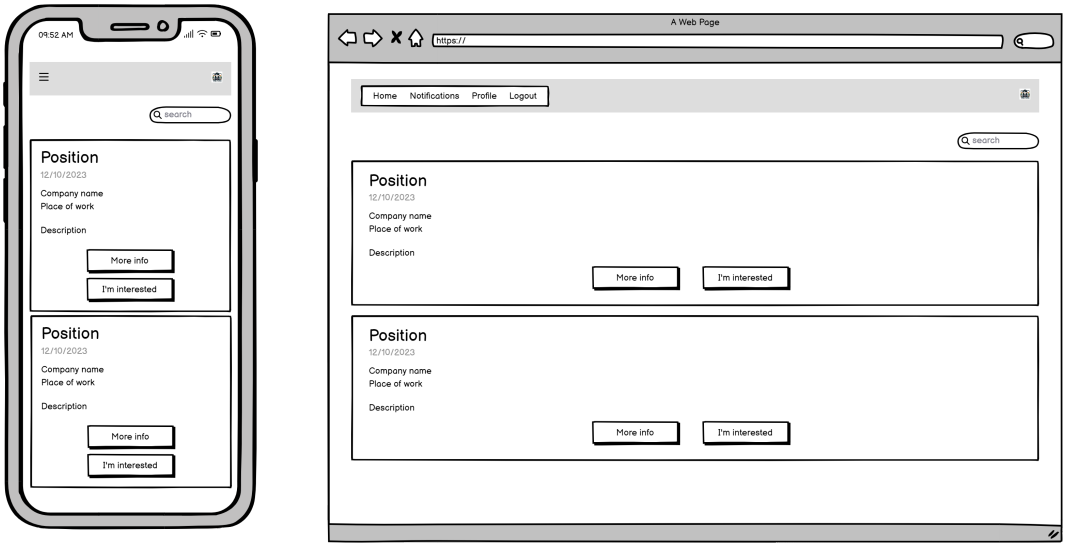

Figure 3: Homepage dello studente.

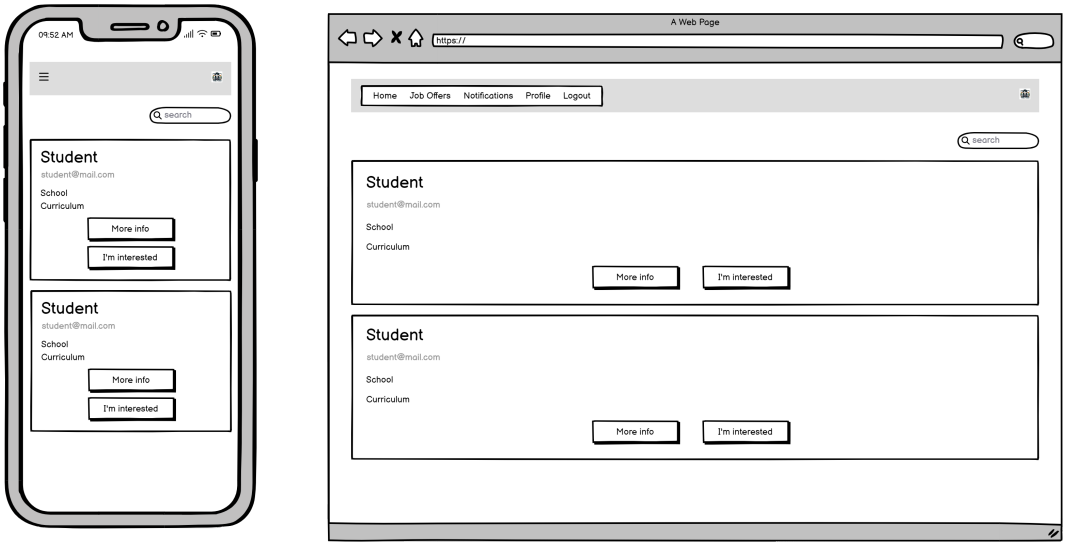

Figure 4: Homepage dell'azienda.

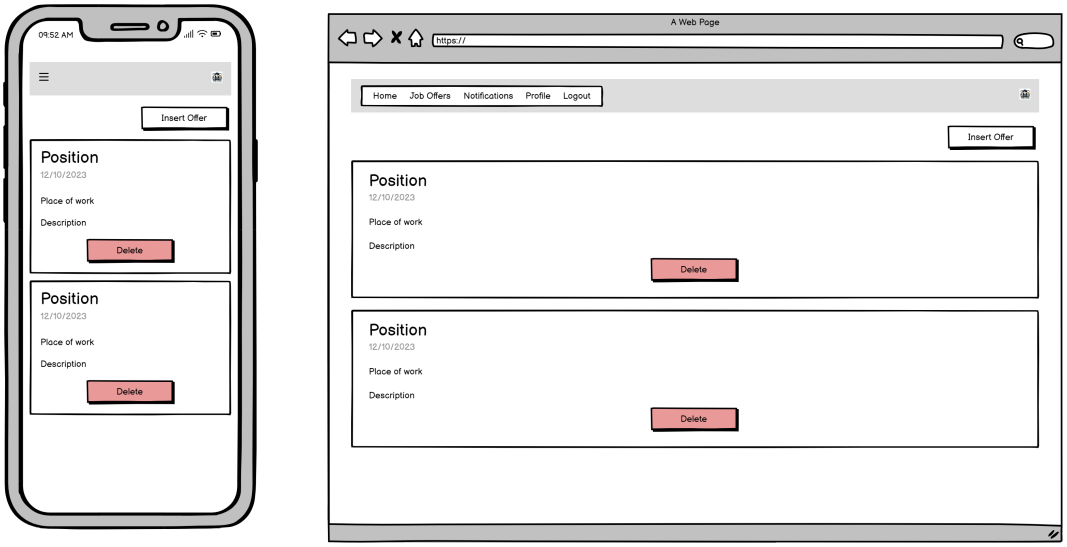

Figure 5: Visualizzazione delle offerte di lavoro, lato azienda.

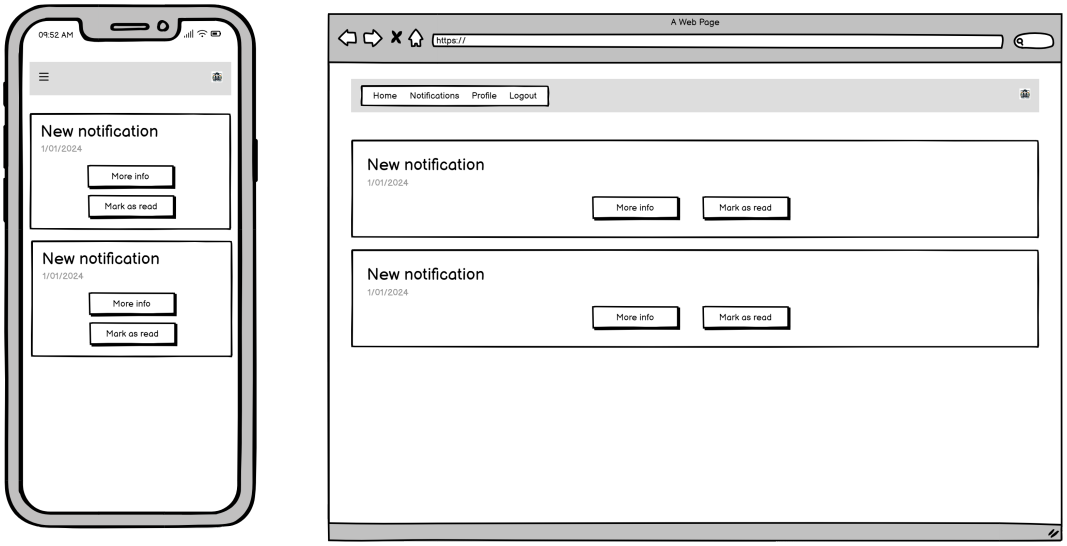

Figure 6: Visualizzazione delle notifiche.

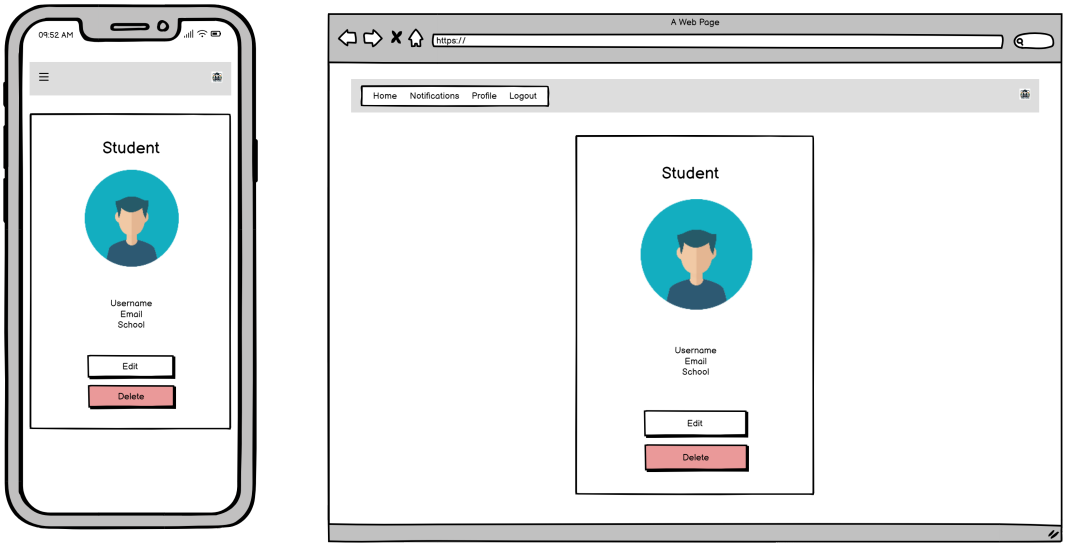

Figure 7: Visualizzazione del profilo utente.

3 Implementazione

Il progetto è stato sviluppato con l'utilizzo dell'IDE WebStorm e ha seguito la macro-struttura riportata in Figura 8.[1]

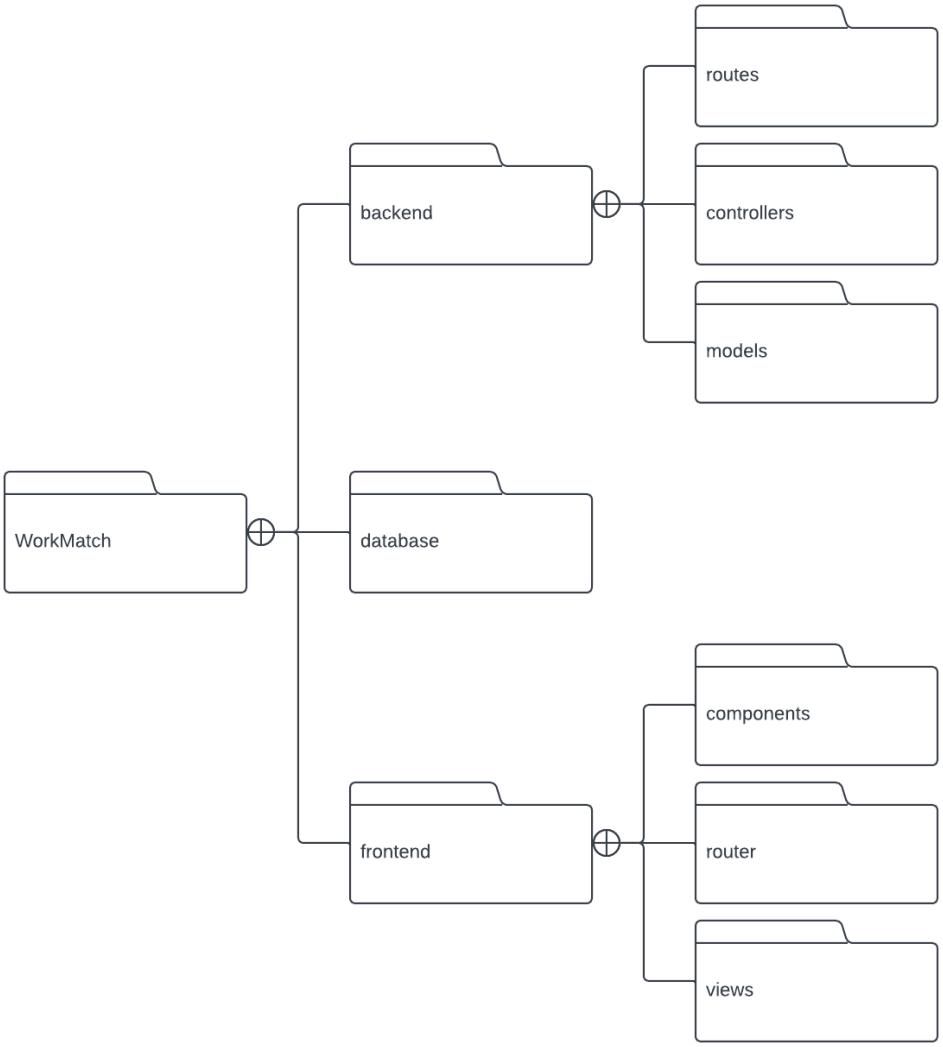

Figure 8: Macro struttura del progetto WorkMatch.

I principali linguaggi utilizzati per l'implementazione sono stati i seguenti:

- JavaScript
- HTML

[1]Il codice sorgente è disponibile in `https://github.com/veronikafolin/workmatch`.

- CSS

3.1 Stack di sviluppo

Al fine di rispettare la scelta architetturale di tipo MVVM, è stato adottato l'utilizzo dello stack di sviluppo **MEVN**, che copre tutti gli aspetti per lo sviluppo di un'applicazione web.

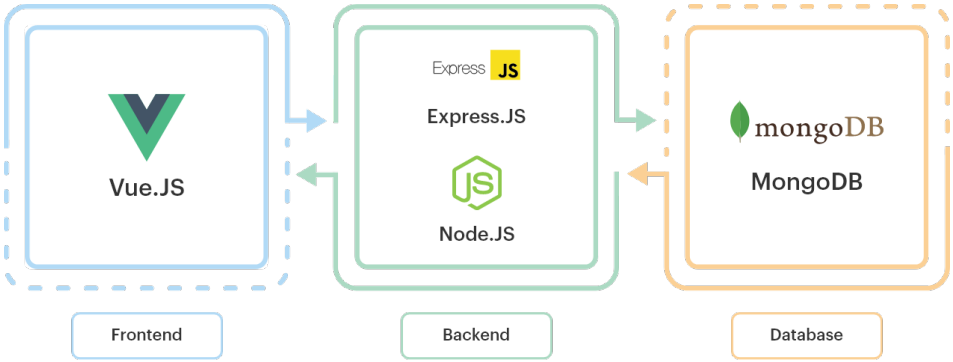

Figure 9: Architettura dello stack MEVN. (Fonte: arkasoftwares)

MEVN è uno dei full-stack JavaScript più popolari e consiste nelle seguenti tecnologie:

- **MongoDB** è un DBMS non relazionale orientato ai documenti. È stato sviluppato per gestire grandi quantità di dati non strutturati o semi-strutturati. Difatti, archivia i dati in documenti *JSON-like*, cioè in formato *BSON (Binary JSON)*, per cui i campi possono variare da documento a documento e la struttura dei dati può essere modificata nel tempo. Inoltre, supporta la replica dei dati, favorendo alta disponibilità e tolleranza ai guasti, e la distribuzione su più server, favorendo la scalabilità orizzontale. Di conseguenza, è possibile implementare l'indicizzazione dei dati per migliorare le prestazioni delle query.

- **Express** è un framework per applicazioni web basate su Node.js, ampiamente utilizzato per la creazione di API RESTful. Offre un sistema di routing che consente di definire facilmente le rotte URL e le azioni da eseguire quando vengono richiamate quest'ultime, facilitando la gestione delle richieste HTTP in entrata e la definizione delle risposte.

- **Vue** è un framework JavaScript per la creazione di interfacce utente e applicazioni web single-page. Promuove l'organizzazione dell'interfaccia utente in componenti riutilizzabili. Implementa il modello MVVM: a differenza del Controller che esegue la logica di business prima del rendering della View, il ViewModel definisce il comportamento dell'applicazione a runtime. Inoltre, offre reattività tra i dati e la View: quando i dati cambiano, la View si aggiorna automaticamente, e viceversa.

- **Node.js** è un runtime environment JavaScript open-source e multipiattaforma, utilizzato principalmente per creare applicazioni server-side, real-time ed event-driven. Node è single-threaded, asincrono e non bloccante in termini di I/O. Include NPM (Node Package Manager), ossia un gestore di pacchetti che offre accesso a migliaia di librerie e moduli JavaScript open-source, facilitando la gestione delle dipendenze e l'aggiunta di funzionalità aggiuntive.

3.2 Database

Come specificato precedentemente, è stato utilizzato il DBMS **MongoDB**. Al termine dell'implementazione, la struttura del database può essere rappresentata da Figura 10.

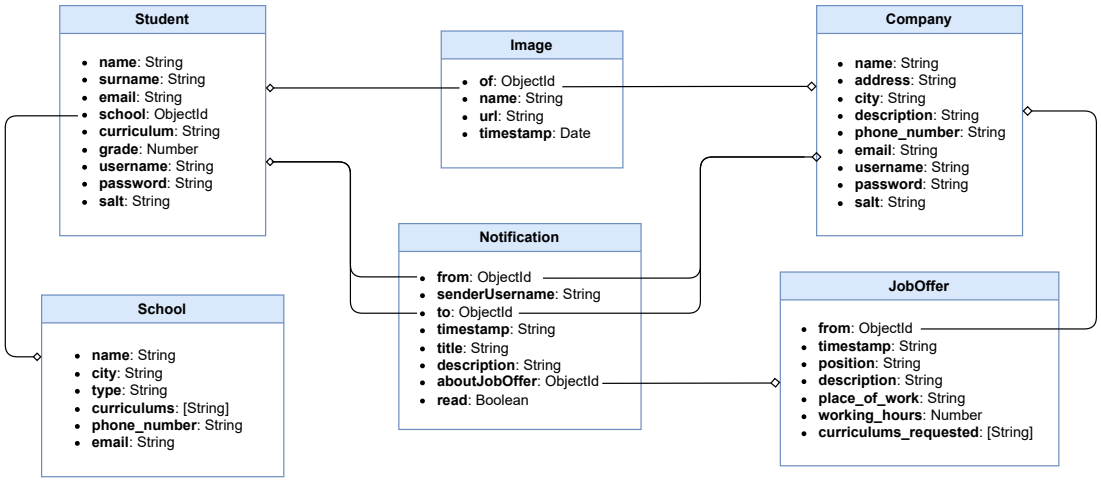

Figure 10: Struttura del database.

Per automatizzare la creazione delle collezioni all'interno del database, sono stati predisposti un file di configurazione `docker_compose.yaml` per l'avvio di un container **Docker** contenente un'immagine di MongoDB, ed una directory di documenti con cui popolare il database.

3.3 Backend

Le funzionalità del backend sono state implementate mediante l'utilizzo delle seguenti librerie messe a disposizione da NPM:

- **bcrypt**, per fare hashing delle password e proteggere il sistema da *rainbow-table attack* attraverso l'utilizzo di un valore definito *salt*.
- **body-parser** è un middleware che estrae il body di una `request` in entrata e la rende disponibile in un nuovo formato (e.g., JSON).

- **cors** è un middleware che abilita cross-origin resource sharing (CORS), ossia una politica di sicurezza implementata nei browser web che controlla e limita le richieste HTTP tra domini o origini differenti. Quando un'applicazione web tenta di effettuare una richiesta HTTP da un dominio diverso rispetto a quello in cui è ospitata, il browser applica la politica CORS per determinare se questa richiesta è consentita o meno. Questo aiuta a prevenire attacchi di sicurezza come l'accesso non autorizzato ai dati dell'utente o alle risorse del server. La politica CORS si basa su header HTTP che il server invia al browser per specificare quali domini sono autorizzati a fare richieste e quali tipi di richieste sono consentiti (e.g., GET, POST, etc.).

- **jsonwebtoken** implementa lo standard JSON Web Token per trasmettere in modo sicuro le informazioni tra le parti sotto forma di oggetti JSON. Queste informazioni possono essere verificate e ritenute attendibili in quanto sono firmate con chiavi pubbliche e private. Viene utilizzato per il confronto della password inserita dall'utente con quella presente nel database.

- **express**, vedi sezione 3.1.

- **mongodb**, ossia il driver ufficiale di MongoDB per Node.js. Fornisce un'API per interagire direttamente con il database MongoDB.

- **mongoose**, una libreria JavaScript orientata agli oggetti che crea una connessione tra MongoDB e Node.js per gestire le operazioni di accesso e manipolazione dei dati nel database.

- **multer** è una libreria middleware per Node.js che semplifica la gestione dei file caricati da parte dell'utente attraverso richieste HTTP. Viene utilizzata per effettuare l'upload di un'immagine profilo scelta dall'utente da locale.

3.4 Frontend

Le funzionalità del frontend sono state implementate mediante l'ausilio delle seguenti librerie:

- **axios**, libreria JavaScript che permette di effettuare richieste HTTP da Node.js, anche asincrone.

- **primeflex** è una libreria di layout che fornisce una serie di classi CSS predefinite per facilitare la creazione di layout flessibili e responsivi nelle applicazioni web.

- **primeicons** offre una vasta raccolta di icone che possono essere utilizzate per arricchire l'interfaccia utente.

- **primevue** fornisce un insieme di componenti per UI basate su Vue.

- **vue** è il pacchetto fondamentale che consente di utilizzare Vue.js.

- **vue-router** è lo strumento che permette la gestione delle rotte all'interno di una applicazione Vue.js single-page.

Uno dei focus principali dell'implementazione è stato quello di garantire uno stato della visualizzazione dei componenti aggiornato a ogni azione dell'utente. Molte richieste lato backend sono state evitate grazie alla gestione della **comunicazione tra i componenti**, mediante l'utilizzo delle props e degli eventi.

Le **props** consentono di passare dati da un componente padre a un componente figlio. Più in dettaglio, sono attributi che il componente figlio può utilizzare per renderizzare il suo contenuto o eseguire operazioni basate sui dati ricevuti. Le props sono passate come attributi HTML nei componenti Vue e sono reattive: ogni cambiamento nel componente padre viene riflesso automaticamente nei componenti figlio che le utilizzano.

Gli **eventi** sono utilizzati per permettere la comunicazione dal componente figlio al componente padre. Un componente figlio può emettere un evento e il componente padre può ascoltare questi eventi utilizzando l'attributo v-on o il metodo @ nelle direttive HTML, prendendo azioni in base a ciò che è stato emesso.

4 Test

L'applicazione è stata testata mediante:

- Test delle API con la piattaforma **Postman**.
- Browser differenti, per verificarne la **portabilità**:
  - Google Chrome.
  - Microsoft Edge.
  - Mozilla Firefox.
- Lo strumento per sviluppatori del browser Google Chrome, col fine di verificare l'**adattabilità** tra le varie tipologie di dispositivo.
- **Test di usabilità** con un utente reale, con l'obiettivo di individuare punti di miglioramento lato frontend e casistiche non gestite correttamente lato backend.

Per il futuro, sarà possibile migliorare l'usabilità e l'accessibilità dell'applicazione verificando il rispetto dell'**euristiche di Nielsen** e delle **Web Content Accessibility Guidelines (WCAG)**.

5 Guida all'Utilizzo

Per testare l'utilizzo dell'applicativo WorkMatch, è possibile seguire i passaggi descritti di seguito.

1. Scaricare il repository localmente:

   `git clone https://github.com/veronikafolin/workmatch.git`

2. Aprire il progetto mediante un IDE (e.g., WebStorm).

3. Installare le dipendenze del progetto, a partire dalle directory che contengono un file denominato `package.json` (i.e., root, frontend, backend):

   `npm install`

4. Creare un Docker container con un'istanza di MongoDB, popolata con i dati presenti nella directory `database/init-mongodb/data`. A partire dalla directory `database`, lanciare il seguente comando:

   `docker compose up`

5. Creare una nuova connessione su MongoDB Compass mediante l'indirizzo URL:

   `mongodb://localhost:27018`

6. Avviare il server:

   `node index.js`

7. Lanciare il seguente comando dalla directory `frontend` e seguire l'URL generato per visualizzare la pagina iniziale:

   `npm run dev`

8. Per interrompere l'esecuzione del container Docker, bisogna eseguire i comandi:

   `docker compose down; docker volume rm $(docker volume ls -q)`

9. In contemporanea all'interruzione del server è bene interrompere il rendering del componente frontend, eseguendo per entrambi la combinazione di tasti:

   `Ctrl + C`.

6 Conclusioni e Sviluppi Futuri

WorkMatch facilita gli studenti a entrare nel mondo del lavoro, attraverso l'offerta di percorsi di formazione o lavorativi specificamente delineati per individui senza esperienze pregresse. Inoltre, fornisce alle aziende un accesso diretto a un'amplia platea di nuove figure da istruire professionalmente, senza l'intermediazione dei centri per l'impiego o di referenti scolastici.

Gli aspetti fondamentali che danno valore a quest'applicazione sono:

- *Profilazione degli utenti*: la web app consente di creare, in modo semplificato, dei profili completi con le informazioni necessarie per descrivere efficacemente le proprie potenzialità.
- *Portabilità*: gli studenti possono cercare opportunità di lavoro, tirocini o programmi di formazione in un unico portale, direttamente dal loro dispositivo preferito, che sia uno smartphone o un pc.
- *Interazione soft*: gli utenti comunicano il proprio interesse mediante l'invio di notifiche, il cui contenuto è predefinito. Ciò permette di avere un'interazione non invasiva: per esempio, non possono verificarsi casistiche in cui uno studente viene "bombardato" da messaggi diretti da parte delle aziende che intendono diffondere le proprie attività. Difatti, ogni azienda non può segnalare il proprio interesse verso una persona presente sul portale più di una volta. D'altra parte, gli studenti non possono selezionare nuovamente un'offerta come interessante dopo la prima volta. Il destinatario della notifica avrà la possibilità di valutare il profilo del mittente e di contraccambiare o meno l'interesse. Inoltre, non sarà obbligato a inviare un feedback nell'immediato e potrà recuperare i messaggi ricevuti mediante lo storico delle notifiche.

Per il futuro, si sono identificate possibili specifiche con cui estendere le funzionalità di WorkMatch:

- Consentire allo studente di dettagliare approfonditamente le proprie esperienze pregresse e i propri interessi, mediante form per la compilazione di un curriculum pre-impostato o il caricamento di un file PDF già esistente.
- Permettere agli utenti di effettuare delle ricerche mirate, mediante l'implementazione di filtri avanzati.
- Consentire la registrazione alla piattaforma con account già esistenti (e.g., Google, Microsoft, Facebook, etc.);

In conclusione, lo sviluppo dell'applicazione ha migliorato le mie competenze in molteplici tecnologie, rendendomi più versatile nel campo dello sviluppo web e fornendomi una base solida per la realizzazione di progetti futuri.