Università degli Studi di Ferrara

UNIVERSITÀ DEGLI STUDI DI FERRARA

CORSO DI LAUREA IN INFORMATICA

Interfaccia web per la gestione dei permessi in una piattaforma E-learning per scuole superiori

Relatore:

Prof. Fabrizio RIGUZZI

Secondo relatore:

Dr. Ing. Arnaud NGUEMBANG FADJA

Laureando:

Solomon Olamide TAIWO

ANNO ACCADEMICO 2023/2024

Indice

Introduzione

La presente tesi di laurea trae origine dal lavoro svolto in collaborazione con System Afrik Information Technology [1]: questa realtà di origine ferrarese rappresenta un autentico faro di innovazione nel panorama delle tecnologie digitali, posizionandosi come un punto di riferimento per lo sviluppo di progetti tecnologicamente avanzati che, anche attraverso la collaborazione attiva con l'Università degli Studi di Ferrara, uniscono la realtà accademica a quella imprenditoriale.

Il contatto fra questi due mondi ha portato e sta portando alla realizzazione di soluzioni di alta qualità per clienti di tutto il mondo: una di queste è la piattaforma E-learning, attualmente in corso di sviluppo e su cui ho avuto il piacere di lavorare, pensata per la gestione di istituti scolastici desiderosi di garantire ai propri studenti, ai loro familiari ed al personale scolastico un servizio moderno, rapido ed affidabile.

L'attività principale svolta è l'implementazione di un modulo volto a gestire i permessi degli utenti utilizzatori del suddetto portale e le annesse logiche di funzionamento, insieme ai controlli che assicurano l'accesso (o il blocco) a specifiche funzionalità della web application.

Le conoscenze acquisite nel corso di laurea sono state fondamentali nell'approccio al codebase e nell'apprendimento rapido delle tecnologie che poi sono state adoperate nel progetto: queste, che meglio verranno descritte nell'apposito capitolo, includono il linguaggio di programmazione Ruby, il database PostgreSQL ed i framework Ruby on Rails, Quasar e Vue.js, una libreria JavaScript.

Questa tesi si occuperà inizialmente di descrivere i requisiti del progetto, per

poi illustrarne le suddette tecnologie utilizzate ed infine enunciare l'argomento principale, ossia l'implementazione della gestione dei permessi utente.

Capitolo 1

Requisiti del progetto

Questo capitolo definisce in dettaglio ciò che deve essere implementato nel codebase applicativo, ossia i requisiti del progetto: questi requisiti sono il fondamento su cui si basa lo sviluppo del sistema di gestione dei permessi degli utenti nell'ambito della piattaforma E-learning di SYSAIT e fanno da punto di riferimento nello svolgimento del lavoro.

Il principale obiettivo del progetto è, per l'appunto, l'implementazione di un'area dedicata alla gestione dei permessi degli utenti: quest'area deve permettere agli amministratori, ossia allo staff degli istituti scolastici che fanno affidamento alla piattaforma, di definire e gestire i suddetti permessi, che ogni singolo utente ha individualmente e nell'ambito del profilo in cui lo stesso è inserito.

Queste autorizzazioni, costituite da risorse e azioni potenziali su esse che meglio verranno definite nel capitolo relativo al backend, devono essere gestibili in modo rapido ed intuitivo - il tutto con un'interfaccia chiara e semplice.

Prima di iniziare a scrivere codice, è innanzitutto fondamentale avere chiaro come dovrebbe essere la pagina, sia lato desktop che mobile, e come dovrebbe funzionare, motivo per cui è essenziale avere un mockup a cui fare riferimento. Inoltre, nel rispetto di uno dei principi più importanti della programmazione, il DRY (Don't Repeat Yourself), è necessario studiare il codice già implementato al fine di poter riutilizzare componenti già definiti e, conseguentemente, semplificare la realizzazione di nuovi.

Di corredo, ma non meno importante, alla realizzazione di questa interfaccia è il controllo delle autorizzazioni degli utenti nelle pagine e nelle componenti del progetto che fanno uso delle suddette azioni, affinchè non sia possibile visualizzare o interagire con le pagine per le quali non si disponga dell'appropriata autorizzazione, fornendo all'utente un messaggio informativo inerente.

Riassumento, quindi, è possibile definire i requisiti del progetto nel seguente modo:

Requisiti funzionali

- **Creazione dei componenti che visualizzano le risorse e le azioni**: attraverso il riutilizzo e l'adattamento di codice già esistente, si andranno a creare i componenti necessari a visualizzare le risorse e le azioni che costituiscono i permessi del sito;
- **Realizzazione del layout della pagina principale**: mediante i componenti realizzati, occorrerà creare l'interfaccia e renderla responsiva, visivamente appagante e facile da utilizzare;
- **Gestione dei permessi**: gli amministratori devono poter assegnare agli utenti o ai loro rispettivi profili determinati privilegi e accessi alle funzionalità della piattaforma, oppure rimuoverli;
- **Accesso alle funzionalità**: gli utenti, a cui sono stati garantiti o revocati specifici permessi, devono poter agire di conseguenza all'interno del portale: devono, cioè, poter visionare ed eseguire le azioni delle quali hanno il rispettivo permesso oppure, in caso negativo, visualizzare un messaggio che avvisa del diniego.

Requisiti non funzionali

Oltre ai requisiti funzionali, il progetto deve soddisfare anche requisiti non funzionali che riguardano prestazioni, sicurezza, usabilità e manutenibilità del sistema. Questi requisiti, come specificato prima, sono cruciali per garantire un'esperienza utente ottimale e la robustezza del sistema nel lungo periodo.

Descrizione del caso d'uso

Il caso d'uso della gestione dei permessi copre una serie di operazioni che possono essere eseguite da due categorie di attori, i membri dello staff scolastico e gli amministratori della piattaforma, per amministrare i permessi degli utenti. Queste operazioni includono l'assegnazione di nuovi permessi, la modifica dei permessi esistenti, la revoca di permessi e la semplice visualizzazione dei permessi di un utente.

1. **Visualizzazione dei permessi di un utente (o profilo)**: il processo di visualizzazione dei permessi di un utente inizia con l'autenticazione dello stesso che, una volta inserite le proprie credenziali, può accedere al modulo di gestione dei permessi aprendo - attraverso il menu laterale - la homepage della struttura di riferimento da cui, tra le varie azioni, è possibile visionare l'elenco dei membri dello staff dello specifico istituto. A questo punto, l'amministratore può selezionare l'utente di cui si desidera visionare i permessi e accedere alla relativa pagina;

2. **Assegnazione di permessi ad un utente (o profilo)**: quest'azione si svolge in modo molto simile alla normale visualizzazione, con la sola differenza che l'utente, per assegnare nuovi permessi, deve selezionare il permesso desiderato e decidere di salvare, attraverso l'apposito pulsante di salvataggio, tale assegnazione. Dopo l'avvenuto salvataggio, la relativa icona verrà disabilitata e l'amministratore può decidere se fare ulteriori modifiche o meno;

3. **Revoca dei permessi di un utente (o profilo)**: la revoca dei permessi di un utente è essenziale per mantenere la sicurezza del sistema e segue un processo molto simile a quello dell'assegnazione, con l'unica differenza che il checkbox relativo all'azione selezionata sarà, al termine del processo, vuoto.

Diagramma del caso d'uso:

Per rappresentare graficamente questi casi d'uso, è possibile utilizzare un diagrama UML (`Unified Modeling Language`):

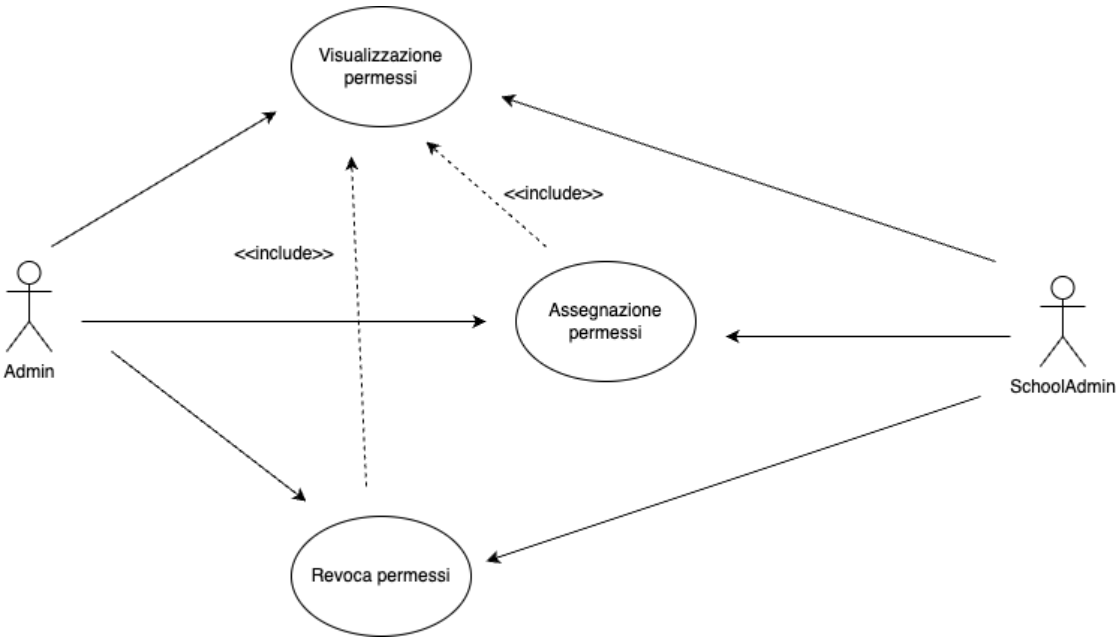

Figura 1.1: Diagramma di caso d'uso per la gestione dei permessi.

Capitolo 2

Tecnologie utilizzate

L'implementazione delle pagine e funzionalità descritte nel capitolo precedente richiede l'adozione di una serie di tecnologie avanzate, scelte per la loro affidabilità, scalabilità e capacità di facilitare lo sviluppo rapido di applicazioni web complesse. Questo capitolo descrive in dettaglio le principali tecnologie utilizzate nel progetto: Ruby e Ruby on Rails, PostgreSQL, Vue.js e Quasar.

Ruby e il suo framework Ruby on Rails sono stati utilizzati per lo sviluppo del backend dell'applicazione, offrendo un ambiente di sviluppo ricco di funzionalità che permette una rapida iterazione durante il processo di sviluppo.

PostgreSQL è stato scelto come sistema di gestione del database per la sua robustezza, per le sue prestazioni e per la capacità di gestire dati in modo efficiente.

Per il frontend, sono stati usati Vue.js e Quasar: Vue.js è una libreria JavaScript progressiva che consente la costruzione di interfacce utente interattive e dinamiche, mentre Quasar è un framework - basato su Vue.js - che facilita lo sviluppo di interfacce utente moderne e reattive, ottimizzate sia su desktop che su dispositivi mobili.

Nelle sezioni che seguono verranno approfondite queste tecnologie, illustrandone le caratteristiche principali.

2.1 Ruby e Ruby on Rails

Ruby

Ruby [2] è un linguaggio di programmazione dinamico e orientato agli oggetti, noto per la sua sintassi semplice ed elegante, che enfatizza la produttività del programmatore e la leggibilità del codice. Creato da Yuhikiro "Matz" Matsumoto, Ruby combina parti di Perl, Smalltalk, Eiffel, Ada e Lisp.

Caratteristiche principali:

- **Sintassi leggibile e concisa**: la sintassi di Ruby è progettata per essere naturale da leggere e facile da scrivere. Ad esempio, un semplice programma per stampare "Hello, World!" è scitto come segue:

```
1          puts 'Hello, World!'
```

- **Orientato agli oggetti**: tutto in Ruby è un oggetto, inclusi i tipi di dati primitivi. Ad esempio:

```
1          5.times do
2            puts "Hello, World!"
3          end
```

  Qui, "5" è un oggetto che ha un metodo "times". Il codice illustrato stampa dunque la stringa "Hello, World!" cinque volte.

- **Dinamicamente tipizzato**: Ruby utilzza un sistema di tipi dinamico e una gestione automatica della memoria. Non è necessario dichiarare i tipi delle variabili.

- **Garbage Collection**: Ruby automatizza la gestione della memoria tramite garbage collection, liberando la memoria non più utilizzata.

Ruby on Rails

Ruby on Rails [3], spesso semplicemente chiamato Rails, è un framework di sviluppo web open-source basato su Ruby. Creato da David Heinemeier Hansson, Rails è progettato per facilitare lo sviluppo di applicazioni web seguendo il paradigma Model-View-Controller (MVC).

Caratteristiche principali:

- **Convention over Configuration (CoC)**: Rails preferisce le convenzioni rispetto alla configurazione, riducendo la quantità di codice che i programmatori devono scrivere.
  Ad esempio, seguendo le convenzioni di Rails, la creazione di una risorsa "Article" associa automaticamente le rotte (che sono le definizioni che mappano le URL alle azioni corrispondenti nei controller), il controller e le viste necessarie.

```
1          rails generate scaffold Article title:string
              body:text
```

  Questo comando genera tutto il codice boilerplate per gestire articoli.

- **Don't Repeat Yourself (DRY)**: Rails incoraggia a riutilizzare il codice e a evitare le duplicazioni. I moduli e le librerie possono essere inclusi facilmente nei vari componenti dell'applicazione.

- **RESTful design:** Rails promuove l'uso di risorse RESTFUL - che sono un modello di progettazione per le applicazioni web in cui le risorse, rappresentate da URL, vengono gestite tramite un insieme standard di operazioni HTTP (GET, POST, PUT, DELETE) che corrispondono alle azioni CRUD (Create, Read, Update, Delete) - per sfruttare le applicazioni web, facilitando la creazione di `API (Application Programming Interface)` ben strutturate e relativamente semplici da gestire.

- **Generazione automatica di codice**: Rails fornisce generatori di codice per lo scaffolding, tecnica che genera automaticamente il codice di base per le

operazioni CRUD, controller, modelli e altro ancora, accelerando il processo di sviluppo.

2.2 PostgreSQL

PostgreSQL [4] è un sistema di gestione di database relazionali ad oggetti open-source, noto per la sua robustezza, performance e conformità agli standard SQL. È stato scelto per la sua capacità di gestire query complesse e volumi di dati elevati, nonchè per le sue caratteristiche avanzate.

Caratteristiche principali:

- **ACID Compliance**: PostgreSQL garantisce Atomicità, Consistenza, Isolamento e Durabilità delle transazioni. Questo significa che ogni transazione è completamente eseguita oppure completamente annullata.
- **Supporto avanzato per tipi di dati**: include tipi di dati avanzati come JSON e XML. Ad esempio:

```
1  CREATE TABLE documents (
2  id SERIAL PRIMARY KEY,
3  data JSONB
4  );
```

  Qui, la colonna 'data' può memorizzare documenti JSON.

- **Estendibilità**: PostreSQL permette l'uso di estensioni per aggiungere funzionalità. Un esempio può essere l'estensione PostGIS aggiunge il supporto per dati geografici:

```
1  CREATE EXTENSION postgis;
```

- **MVCC (Multi-Version Concurrency Control)**: permette la gestione concorrente delle transazioni senza lock pesanti, migliorando le performance

in ambienti ad alta concorrenza.

- **Backup e ripristino**: offre strumenti robusti per il backup e il ripristino dei dati. Ad esempio, "pg_dump" è utilizzato per creare backup:

```
pg_dump mydb > db_backup.sql
```

2.3 Vue.js e Quasar

Vue.js

Vue.js [5] è una libreria JavaScript progressiva per la costruzione di interfacce utente interattive, creato da Evan You e progettata per essere incrementale, permettendo ai programmatori di adottare il framework gradualmente.

Caratteristiche principali:

- **Componenti reattivi**: Vue.js utilizza componenti reattivi, ossia che rispondono automaticamente ai cambiamenti dei dati, aggiornando il DOM in modo efficiente e dinamico per riflettere lo stato attuale dell'applicazione, e modulari per costruire interfacce utente. Ad esempio:

```
<template>
  <div>
    <p>{{ message }}</p>
    <button @click="reverseMessage">Reverse
       Message</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
```

```
12              message: 'Hello, Vue!'
13            }
14          },
15          methods: {
16            reverseMessage() {
17              this.message = this.message.split('').
                    reverse().join('')
18            }
19          }
20        }
21        </script>
```

Questo codice definisce un componente che visualizza un messaggio e include un pulsante che, quando cliccato, inverte il testo del messaggio.

- **Data binding**: offre un binding bidirezionale dei dati per sincronizzare il modello e la vista.

- **Virtual DOM**: usa un DOM virtuale per migliorare le performance delle applicazioni. Il virtual DOM consente di fare aggiornamenti efficienti al DOM reale solo quando necessario.

- **Ecosistema ricco**: include strumenti come Vue Router per il routing e Vuex per la gestione dello stato. Ad esempio, Vue Router permette di definire le rotte dell'applicazione:

```
1         import Vue from 'vue'
2         import Router from 'vue-router'
3         import Home from '@/components/Home'
4
5         Vue.use(Router)
6
7         export default new Router({
8           routes: [
9             {
10              path: '/',
```

```
11                    name: 'Home',
12                    component: Home
13                  }
14                ]
15              })
```

- **Facilità di integrazione**: può essere integrato facilmente in progetti esistenti senza sovrascrivere il codice.

Quasar

Quasar [6] è un framework basato su Vue.js che facilita lo sviluppo di applicazioni web responsive, ossia capaci di adattarsi ed essere fruibili su un'alta gamma di dispositivi, e interattive. Consente di creare applicazioni multipiattaforma con un unico codice base.

Caratteristiche principali:

- **Componenti UI predefiniti**: include una vasta gamma di componenti UI pronti all'uso. Ad esempio:

```
<template>
  <q-btn label="Click me" @click="handleClick" />
</template>

<script>
export default {
  methods: {
    handleClick() {
      this.$q.notify({
        message: 'Bottone premuto!',
        color: 'green'
      })
    }
  }
}
</script>
```

Questo codice Vue.js utilizza il framework Quasar per creare un semplice pulsante che, quando cliccato, mostra una notifica. Nel template, viene definito un pulsante <q-btn> con l'etichetta "Click me" e un evento @click che chiama il metodo handleClick quando il pulsante viene premuto. Nella parte script, il componente esporta un oggetto che include un metodo handleClick. Questo metodo utilizza $q.notify per visualizzare una notifica con il messaggio "Bottone premuto!" di colore verde.

- **Sviluppo multipiattaforma**: supporta la creazione di applicazioni web, mobile (iOS e Android) e desktop (con Electron).
- **Prestazioni ottimizzate**: include strumenti per ottimizzare le prestazioni e ridurre il tempo di caricamento, come il lazy loading dei componenti.
- **CLI potente**: Quasar CLI (`Command Line Interface`, ossia 'Interfaccia a riga di comando', che consente agli utenti di interagire con il software tramite comandi testuali inseriti in una console o terminale) facilita la gestione e la configurazione del progetto, permettendo di generare rapidamente scaffolding per nuove applicazioni.

```
1 quasar create my-project
```

- **Supporto per PWA**: permette di sviluppare Progressive Web Apps, ossia applicazioni web che utilizzano le più recenti tecnologie web per offrire un'esperienza utente simile a quella delle applicazioni native, inclusa l'accessibilità offline, le notifiche push e l'installazione sul dispositivo con facilità.

Altre tecnologie

Sistema di versionamento (Git)

Il controllo versione del codice svolge un ruolo cruciale nello sviluppo software moderno, consentendo ai team di lavorare in modo collaborativo, tenere traccia delle modifiche e gestire il codice in modo efficiente.

Nel progetto è stato adottato Git [7] come sistema di versionamento principale, che fornisce un modo flessibile e affidabile per gestire il codice, consentendo di lavorare in branch separati, collaborare su funzionalità diverse contemporaneamente e integrare facilmente le modifiche tramite pull requests.

Di seguito i comandi più utilizzati:

Inizializzazione di un repository

```
1   git init
```

Aggiunta di un file al repository

```
1   git add <file>
```

Creazione di un commit

```
1   git commit -m "Messaggio del commit"
```

Caricamento del commit al repository remoto

```
1   git push
```

Aggiornamento della repository locale

```
1   git pull
```

Creazione di una nuova branch

```
1   git checkout -b <nome_branch>
```

Merging di una nuova branch

```
1  git merge <nome_branch>
```

Visualizzazione dello stato

```
1   git status
```

Visualizzazione dei log

```
1  git log
```

Hosting delle repositories (Bitbucket)

Per ospitare le repositories Git e facilitare la collaborazione tra i membri del team, è stato utilizzato Bitbucket, strumento che fornisce un'infrastruttura sicura e scalabile per archiviare il codice, con funzionalità avanzate come il controllo degli accessi basato su ruoli, il tracciamento delle issue e la gestione delle pull request.

Documentazione del progetto (Confluence)

Per gestire la documentazione del progetto è stata usata Confluence [8], piattaforma di gestione delle conoscenze e della collaborazione, che fornisce uno spazio

centralizzato per creare, organizzare e condividere documentazione importante, come specifiche dei requisiti, documenti tecnici, guide per gli sviluppatori e altro ancora.

Capitolo 3

Implementazione della gestione dei permessi utente

3.1 Gestione del frontend dei permessi

Nell'era digitale moderna, il web design ha assunto un ruolo cruciale nel determinare il successo di una piattaforma: un design efficace e semplice da usare non solo migliora l'esperienza dell'utente, ma aumenta anche l'efficienza operativa e la soddisfazione complessiva. Esso deve inoltre essere intuitivo, accessibile e reattivo, garantendo che gli utenti possano navigare e utilizzare le funzionalità della piattaforma senza incontrare ostacoli.

Una delle principali sfide nel design delle interfacce utente (UI) è trovare l'equilibrio tra funzionalità e usabilità, infatti esse dovrebbero offrire tutte le funzionalità necessarie senza risultare complesse o sovraccariche.

Questo principio è particolarmente rilevante nel contesto della piattaforma e-learning, dove sia le funzionalità che i dispositivi utilizzati variano enormemente e occorre trovare un equilibrio per soddisfare la più ampia platea possibile, senza sovraccaricare l'utente e l'interfaccia.

Dal punto di vista tecnico, un web design efficace segue le best practice del responsive design, garantendo che l'interfaccia sia ottimizzata per vari dispositivi, dai desktop ai dispositivi mobili.

3.1.1 Design adottato

L'interfaccia utente per la gestione dei permessi, inizialmente progettata sotto forma di mockup da usare come riferimento durante l'implementazione (mostrata nelle figure 3.1 e 3.2), è intuitiva e facile da usare, sia per dispositivi desktop che mobili.

Nella versione desktop, l'interfaccia utente è organizzata in modo chiaro e funzionale: in cima alla pagina è presente un'intestazione con il titolo "Permission Management", sotto la quale vengono visualizzate le informazioni dell'utente, tra cui il nome completo, il numero di contatto, l'indirizzo email e il profilo a cui appartiene.

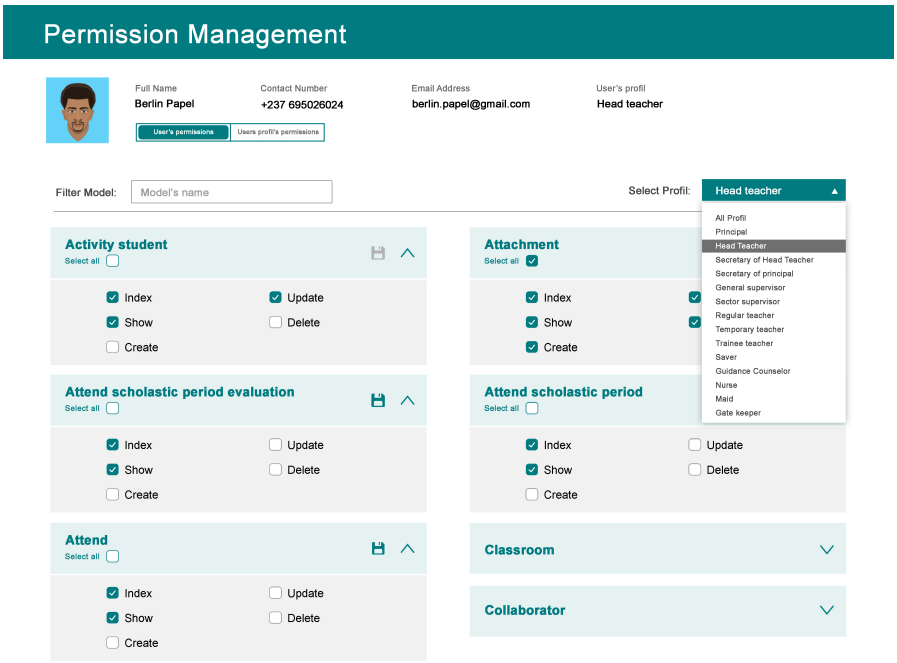

Figura 3.1: Mockup dell'interfaccia desktop per la gestione dei permessi.

Subito sotto l'intestazione troviamo una sezione di navigazione che include due pulsanti per la gestione dei permessi: `User's permissions` e `Users profile permissions`, che sono necessari alla selezione del tipo di permessi da gestire (quelli del singolo utente oppure quelli del ruolo scelto).

Sotto a questi pulsanti, vi è un campo di input che permette di filtrare le risorse per nome e un menu a tendina che consente di selezionare il profilo da modificare, con varie opzioni disponibili come `Principal, Secretary of Head Teacher, General Supervisor` e molte altre.

La gestione dei permessi è suddivisa in diverse risorse, come `Activity student, Attend scholastic period evaluation` e `Attachment`, ciascuna delle quali include una serie di azioni (`Index, Show, Create, Update, Delete`) che possono essere selezionate o deselezionate dall'utente. Ogni risorsa dispone di un'opzione `Select all` per facilitare la selezione di tutti i permessi contemporaneamente e un'icona di salvataggio per confermare le modifiche effettuate.

Nella versione mobile, l'interfaccia è stata adattata per garantire un'esperienza utente ottimale su schermi piccoli. Le informazioni dell'utente sono presentate in modo più compatto, omettendo il numero di contatto e l'indirizzo email visibili nella versione desktop. Il titolo `Permission Management` è posizionato in alto, accompagnato dall'avatar dell'utente.

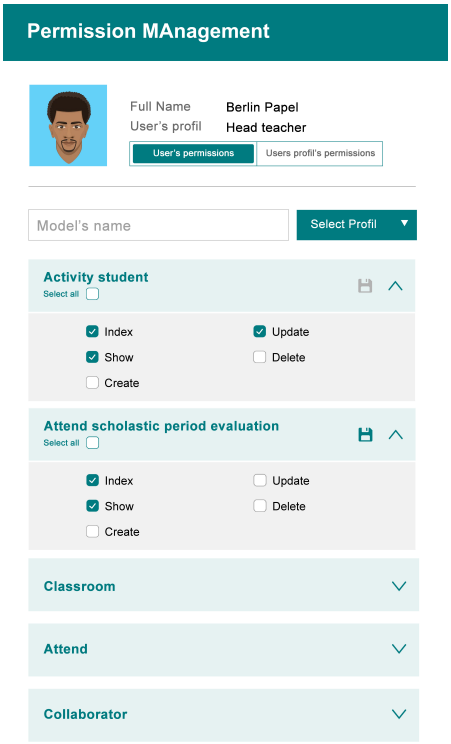

Figura 3.2: Mockup dell'interfaccia mobile per la gestione dei permessi.

La sezione di navigazione nella versione mobile mantiene gli stessi pulsanti `User's permissions` e `Users profile permissions`, un campo di input per filtrare i modelli per nome e un menu a tendina per la selezione del profilo dell'utente. La gestione dei permessi in questa versione è organizzata in gruppi (di risorse) che possono essere espansi o compressi, facilitando la navigazione. Ogni risorsa include un pulsante `Select all` e le stesse opzioni di permessi individuali (`Index`, `Show`, `Create`, `Update`, `Delete`) presenti nella versione desktop. Le modifiche ai permessi possono essere salvate tramite un'icona di salvataggio presente in ogni sezione.

Come specificato in precedenza, il design da adottare è stato inizialmente realizzato come mockup che, nel corso del tempo e durante l'implementazione del codice, è stato modificato e rivisto in base al sorgere di nuove esigenze.

Passando al design effettivo implementato per i desktop, è possibile notare come siano state apportate diverse modifiche rispetto ai mockup iniziali.

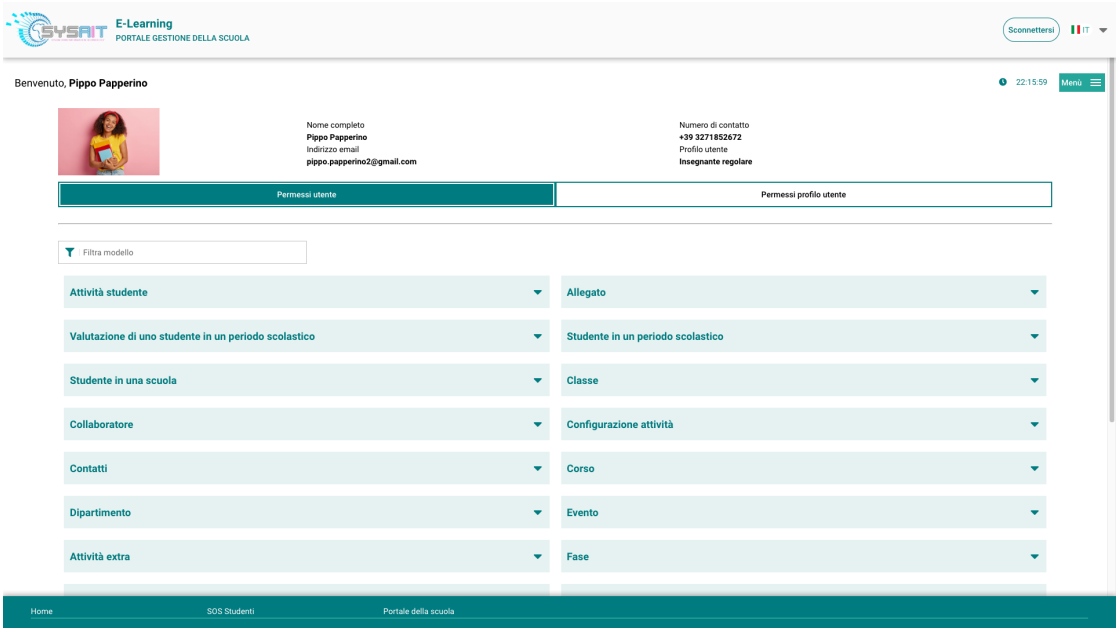

Figura 3.3: Design effettivo dell'interfaccia desktop per la gestione dei permessi.

Le informazioni dell'utente e il filtro delle risorse sono stati ricalcati dal mockup, mentre - per motivi di visibilità e facilità di implementazione - i pulsanti per la selezione del tipo di permessi da gestire sono stati resi più grandi.

Come detto in precedenza, le modifiche apportate rispetto ai mockup originali possono essere attribuite a diversi fattori, tra cui la necessità di migliorare l'usabilità, l'estetica, la navigabilità dell'interfaccia e, non da meno, la modalità di implementazione (alcune scelte di design sono dipese dalla facilità di scrittura del codice necessario).

Discorso simile per il design effettivo dell'interfaccia mobile, che - anche in questo caso - rispetto al mockup, vede i suddetti pulsanti per la selezione del più grandi e fruibili.

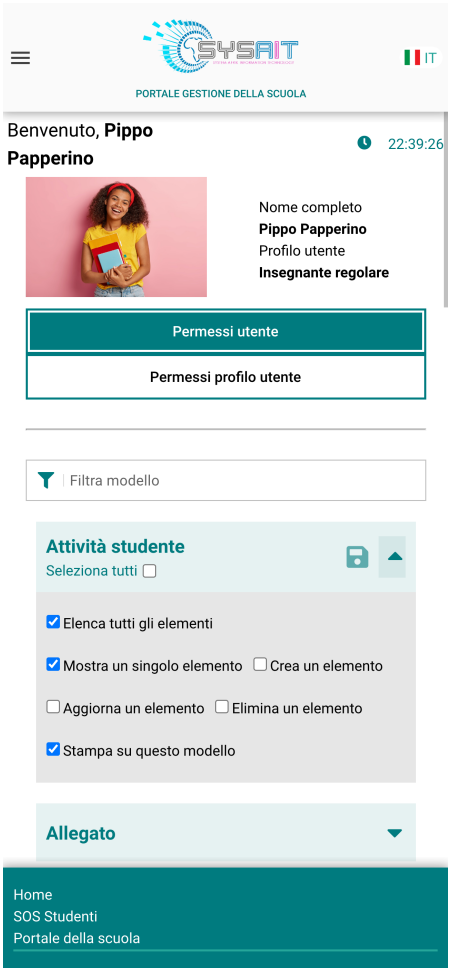

Figura 3.4: Design effettivo dell'interfaccia mobile per la gestione dei permessi.

3.1.2 Implementazione

Per creare l'interfaccia web per la gestione dei permessi utente sono stati utilizzati diversi componenti. Di seguito viene descritto il componente `ButtonComponent`, che è stato utilizzato come base per creare bottoni personalizzati nell'applicazione e, in questo caso, il pannello collassabile che poi viene utilizzato per visionare le risorse e le relative azioni.

ButtonComponent

Il componente `ButtonComponent` realizza un bottone altamente configurabile che utilizza il framework Quasar per Vue.js. Il suo template è mostrato nel Listing 3.1:

```
<template>
  <q-btn ... @click.prevent="handleClick">
    <slot></slot>
  </q-btn>
</template>

<script>
export default {
  name: "ButtonComponent",
  props: {
    fab: { type: Boolean, default: false },
    align: { type: String, default: "center" },
    show: { type: Boolean, default: true },
    disable: { type: Boolean, default: false },
    rounded: { type: Boolean, default: false },
    round: { type: Boolean, default: false },
    noCaps: { type: Boolean, default: false },
    padding: { type: String, default: "sm" },
    title: { type: String, default: null },
    btnClass: { type: String, default: "text-white bg-primary
       " },
    blank: { type: Boolean, default: false },
```

```
22      size: { type: String, default: null },
23      minScreen: { type: Number, default: 0 },
24      to: { type: String, default: null },
25      event: { type: String, default: null },
26      badge: { type: String, default: null },
27      badgeColor: { type: String, default: "red" },
28      outline: { type: Boolean, default: false },
29      label: { type: String, default: "" },
30      icon: { type: String },
31      iconRight: { type: String },
32      flat: { type: Boolean, default: true },
33      color: { type: String, default: "primary" },
34      textColor: { type: String, default: "white" },
35      icon_right: { type: String, default: "" },
36      propStyle: { type: Object, default: null },
37    },
38    computed: {
39      ...
40    },
41    methods: {
42      handleClick(evt) {
43        const events = {
44          evt,
45          event: this.event,
46        };
47        this.event
48          ? this.$emit("click", events)
49          : this.$router.push({ name: this.to });
50      },
51    },
52  };
53  </script>
```

Listing 3.1: Template di ButtonComponent

Il template del componente `ButtonComponent` utilizza l'elemento `q-btn` di Quasar per creare un bottone altamente configurabile attraverso le proprietà `props`,

che permettono di modificare aspetti come la visibilità, l'allineamento, il colore, l'icona e molto altro.

Il metodo `handleClick` gestisce l'evento di click sul bottone. Se viene specificato un evento `this.event`, esso viene emesso, altrimenti il bottone effettua un redirect alla pagina specificata dalla proprietà `to` del `prop`.

CollapsiblePanel

Il componente `CollapsiblePanel` utilizza il `ButtonComponent` per creare una sezione espandibile che può contenere altri elementi. Il Listing 3.2 mostra il codice

```
<template>
  <div class="col WidgetWrapper" :class="`${$options.name}`">

    ...

      <div class="col-auto text-right">
        <div class="right-buttons row no-wrap">
          <div v-if="showBody">
            <ButtonComponent @click="() => $emit('buttonClick')" textColor="primary" btnClass="grey"
              v-for="(button, index) in allButtons" v-bind="button" :key="`${index}buttons`">
            </ButtonComponent>
          </div>
          <q-btn-dropdown @click="showBody = !showBody" flat>
            </q-btn-dropdown>
        </div>
      </div>
    </div>

    ...

  </div>
</template>
```

```
22
23 <script>
24 import ButtonComponent from "src/components/utils/
      ButtonComponent.vue";
25
26 export default {
27   name: "CollapsiblePanel",
28   components: {
29     ButtonComponent,
30   },
31   props: {
32     title: { type: String, default: "Test Title" },
33     allButtons: { type: Array, default: () => [] },
34     items: { type: Array, default: () => [] },
35   },
36   data() {
37     return { showBody: false };
38   },
39 };
40 </script>
```

Listing 3.2: Implementazione di CollapsiblePanel

Il componente `CollapsiblePanel`, come detto sopra, è una sezione espandibile che può contenere altri contenuti, utilizzando il `ButtonComponent` per creare bottoni che possono essere cliccati per eseguire azioni specifiche.

PermissionCollapsiblePanel

Il componente `PermissionCollapsiblePanel` estende `CollapsiblePanel` per gestire i permessi degli utenti. Il codice è mostrato nel Listing 3.3:

```
1 <template>
2   <div :class="`${$options.name} row`" class="collapsible-
       panel-container">
3     <CollapsiblePanel :title="panel.title" :allButtons="
         allButtons" class="collapsible-panel" style="padding:
```

```
       10px"
4      @buttonClick="saveProcess">
5      <template v-slot:subtitle-slot>
6        <label style="display: flex">{{ $t('selectAll') }} <
           input type="checkbox" style="margin-left: 5px"
7           v-model="selectAll" /></label>
8      </template>
9      <template v-slot:content-body>
10       <div class="row elements">
11         <div v-for="(item, index) in panel.items" :key="
             index" class="checkbox">
12           <label>
13             <input class="single-element" type="checkbox" v
                 -model="item.checked" @change="updateChanges
                 " />
14             {{ $t(`${item.label}`) }}
15           </label>
16         </div>
17       </div>
18     </template>
19   </CollapsiblePanel>
20 </div>
21 </template>
22
23 <script>
24 import CollapsiblePanel from "src/components/permissions/
     CollapsiblePanel.vue";
25
26 export default {
27   name: "PermissionCollapsiblePanels",
28   components: { CollapsiblePanel },
29   props: {
30     panel: { type: Object, default: () => { } },
31   },
32   data() {
33     return {
```

```
34        allButtons: [{ icon: "fas fa-save", event: 'save',
              disabled: true }],
35        changesMade: false,
36      };
37    },
38    computed: {
39      selectAll: {
40        ...
41      }
42    },
43    methods: {
44      selectAllItems(value) {
45        ...
46      },
47      allItemsChecked() {
48        return this.panel.items.every(item => item.checked)
49      },
50      ...
51      saveProcess() {
52        let permissions = {};
53
54        for (let item of this.panel.items) {
55          permissions[item.label] = item.checked;
56        }
57
58        this.$emit('updatePermissions', { permissions, model:
              this.panel.model });
59
60        ...
61
62      }
63    }
64  };
65  </script>
```

Listing 3.3: Implementazione di PermissionCollapsiblePanel

Il componente `PermissionCollapsiblePanel` aggiunge la gestione dei permessi, includendo una lista di checkbox (le azioni) che rappresentano i permessi, con un'opzione per selezionare tutti i permessi tramite una checkbox globale, e il pulsante per salvare le modifiche fatte.

Il metodo `saveProcess` raccoglie lo stato dei permessi e lo invia tramite l'evento `updatePermissions`.

Permission Collapsible Panels

Infine, il componente `PermissionCollapsiblePanels` utilizza `PermissionCollapsiblePanel` per visualizzare più pannelli di permessi. Ecco il codice:

```
1 <template>
2   <div :class="`${$options.name} row`" class="collapsible-
       panel-container">
3     <PermissionCollapsiblePanel v-for="(panel, index) in
         panels" :key="`${index}panels`" :panel="panel"
4       @updatePermissions="(params) => $emit('
           updatePermissions', params)" class="col-md-6 col-12"
           >
5     </PermissionCollapsiblePanel>
6   </div>
7 </template>
8
9 ...
```

Listing 3.4: Implementazione di PermissionCollapsiblePanels

Ogni volta che vengono aggiornati i permessi in uno dei pannelli, l'evento `updatePermissions` viene emesso per informare il componente padre dei cambiamenti.

Permissions.vue

Il componente `Permissions.vue`, ossia la pagina che l'utente visualizza e di cui il design è stato mostrato nell'apposito capitolo, utilizza `PermissionCollapsiblePanels`

per fornire un'interfaccia completa per la gestione dei permessi utente. Ecco il codice:

```
<template>
  <div class="page-container" :class="`${$options.name}`">
    <q-page :class="`${$options.name}`">
      <div class="permissions" style="margin: 0 5%">
        <div class="row permissions-user-infos items-center"
           v-if="userId && isUserDataLoaded">

          ...

          <div class="permission-buttons-col col-12">
            <div class="row permission-type-buttons">
              <ButtonComponent class="user-permissions-button
                  col-12 col-md-6" v-bind="
                 userPermissionsProp"
                @click="handleUserPermissionsSelection" />
              <ButtonComponent class="user-profile-
                 permissions-button col-12 col-md-6"
                v-bind="userProfilePermissionsProp" @click="
                   handleUserProfilePermissionsSelection" />
            </div>
          </div>
        </div>

        ...

        <div class="row permissions-filters items-center">
          <div class="col-md-3 col-12">
            <InputComponent class="filter-model" :options="
               panels" v-bind="filterProps" @updated="
               handleModelSearch" />
          </div>
          <!-- Selection of profile, only visible if there is
              no userId -->
```

```
26            <div class="col-md-4 col-12 offset-md-5">
27              <div v-if="profileOptions.length > 0 && !userId">
28                <InputComponent class="profile-selection" :
                     options="profileOptions" v-bind="selectProps
                     "
29                  @updated="handleProfileSelection" />
30              </div>
31            </div>
32          </div>
33
34          <div class="row permissions-actions" :key="`${
                showPanels}showPanels`" v-if="showPanels">
35            <div v-if="noModelFound">
36              <b>{{ this.$t(`noModelFound`) }}</b>
37            </div>
38            <div v-else>
39              <PermissionCollapsiblePanels :panels="
                     filteredPanels" @updatePermissions="
                     updatePermissions" />
40            </div>
41          </div>
42        </div>
43      </q-page>
44    </div>
45  </template>
46
47  <script>
48  ...
49
50  export default {
51    name: "Permissions",
52    mixins: [checkingDependencyOfModels],
53    components: {
54      PermissionCollapsiblePanels,
55      InputComponent,
56      ButtonComponent,
```

```
57  },
58  data() {
59    return {
60      permissionsParams: {},
61      panels: [],
62      filteredPanels: [],
63      userId: this.$route.params.userId,
64      schoolId: this.$route.params.schoolId,
65      userData: [],
66      profileOptions: [],
67      selectedProfile: null,
68      noModelFound: false,
69      selectProps: {
70        isSelect: true,
71        isRequired: true,
72        mapOptions: true,
73        emitValue: true,
74        prependIcon: "fas fa-user",
75        name: "profile",
76        useInput: false,
77        clearable: true,
78        label: this.$t(`selectProfile`),
79        outlined: true
80      },
81      filterProps: {
82        prependIcon: "fas fa-filter",
83        name: "filter",
84        useInput: true,
85        clearable: true,
86        label: this.$t(`filterModel`),
87        maxlength: 60,
88        outlined: true
89      },
90      userPermissionsButtonIsSelected: true,
91      userPermissionsProp: {
92        label: this.$t(`userPermissions`),
```

```
93          noCaps: true,
94          event: 'handleUserPermissionSelection',
95          btnClass: 'bg-primary',
96          textColor: 'white',
97        },
98        userProfilePermissionsProp: {
99          ...
100       }
101     };
102   },
103
104   ...
105
106   },
107
108   mounted() {
109     this.fetchProfiles();
110     this.fetchModels();
111     this.fetchUserData();
112     this.permissionsButtonSelection();
113   },
114   methods: {
115     async updatePermissions(permissionsParams) {
116       this.permissionsParams = { ...permissionsParams };
117       try {
118         const savedPermissions = await this.$hs.methods.
              ajaxCall(`/users/edit_permissions`, "patch", {
              params: this.form });
119         // console.log("Edited permissions: ",
              savedPermissions);
120       } catch (error) {
121         console.error("Error updating permissions: ", error);
122       }
123     },
124     async fetchModels() {
125       if (this.isUserPermissions) {
```

```
126         // If there's an userId, get user's permissions and
                models here
127         try {
128           const permissionsResp = await this.$hs.methods.
                ajaxCall(`/users/permissions?user_id=${this.
                userId}`, "get");
129           // Transformation of the object into an array of
                panel objects
130           this.panels = Object.entries(permissionsResp.data.
                permissions).map(([title, permissions]) => ({
131             title: this.$t(`permissions.${title}`),
132             model: title,
133             items: Object.entries(permissions).map(([label,
                  checked]) => ({
134               label,
135               checked
136             }))
137           }));
138           this.filteredPanels = this.panels;
139         } catch (error) {
140           console.error("Error fetching user permissions: ",
                error);
141         }
142       }
143     },
144     async fetchProfiles() {
145       if (!this.userId) {
146         try {
147           const profilesResp = await this.$hs.methods.
                ajaxCall(`/profiles?minimal=true`, "get");
148           this.profileOptions = profilesResp.data;
149         } catch (error) {
150           console.error("Error fetching profiles: ", error);
151         }
152       }
153     },
```

```
154     async fetchUserData() {
155       if (this.userId) {
156         try {
157           const userDataResp = await this.$hs.methods.
                ajaxCall(`/users/user_home?user_id=${this.userId
                }`, "get");
158           this.userData = userDataResp.data;
159         } catch (error) {
160           console.error("Error fetching user data: ", error);
161         }
162       }
163     },
164     ...
165   }
166 }
167
168 ...
```

Listing 3.5: Implementazione di Permissions.vue

Questa complessa pagina, costituita da varie chiamate al backend che gestiscono i dati degli utenti (la loro ricezione e la loro visualizzazione), il caricamento dei loro permessi e la modifica degli stessi, fa ampio uso dei componenti sopra descritti e dei relativi `prop` per personalizzarli e renderli funzionali allo scopo del progetto.

3.2 Gestione backend dei permessi

3.2.1 Schema del DB

Il progetto, come già detto, utilizza `PostgreSQL` come sistema di gestione di database relazionali, mentre per visualizzare e amministrare il database in modo efficace viene utilizzato `pgAdmin 4`, programma dotato di un'interfaccia grafica potente e intuitiva che facilita la gestione delle strutture del database, l'esecuzione di query SQL e l'analisi dei dati.

A proposito della struttura del database, si rende ora necessario definire il diagramma ERD (`Entity-Relationship Diagram`), e lo schema relazionale, al fine di avere una panoramica complessiva della parte di DB relativa ai permessi utente.

Diagramma Entità-Relazione (`ERD`)

Per modellare le relazioni tra le entità coinvolte nella gestione dei permessi, è possibile utilizzare un diagramma entità-relazione (`ERD`, `Entity-Relationship Diagram`). Questo diagramma mostra come le entità "Serves", "User" e "School" siano correlate tra loro.

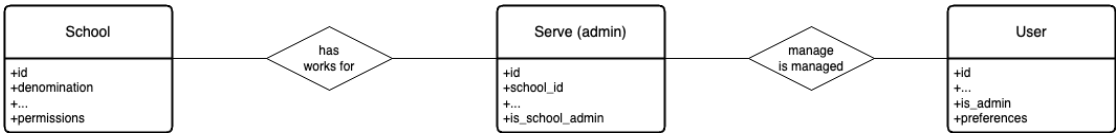

Figura 3.5: Diagramma ERD per la gestione dei permessi

Tabella users

La tabella users rappresenta gli utenti dell'applicazione. Di seguito vengono descritti i campi principali:

- **id**: Identificatore unico per ogni utente;
- **email**: Indirizzo email dell'utente;
- **first_name**: Nome dell'utente;

- **last_name**: Cognome dell'utente;
- **is_admin**: Booleano che indica se l'utente ha privilegi amministrativi;
- **userable_type**: Tipo di entità collegata all'utente (ad es. "Staff" o "Student");
- **userable_id**: ID dell'entità collegata all'utente;
- **identification_number**: Numero identificativo univoco dell'utente;
- **preferences**: Preferenze specifiche relative alla scuola preferita dell'utente in formato JSON;
- **avatar_url**: URL dell'immagine avatar dell'utente;

Tabella schools

La tabella `schools` rappresenta le scuole gestite dall'applicazione. Di seguito vengono descritti i campi principali:

- **id**: Identificatore unico per ogni scuola;
- **denomination**: Nome ufficiale della scuola;
- **contacts_info**: Informazioni di contatto della scuola in formato JSON (email, indirizzo, numeri di telefono);
- **social_media**: Informazioni sui social media della scuola in formato JSON (URL dei profili social);
- **identification_number**: Numero identificativo univoco della scuola;
- **theme_id**: ID del tema grafico associato alla scuola;
- **root_id**: ID della scuola radice nel caso di un'organizzazione gerarchica;
- **parent_id**: ID della scuola genitore nel caso di un'organizzazione gerarchica;
- **category_id**: ID della categoria della scuola;

- **permissions**: Permessi dei profili associati alla scuola in formato JSON;

Tabella serves

La tabella `serves` rappresenta il collegamento tra il personale (staff) e le scuole. Di seguito vengono descritti i campi principali:

- **id**: Identificatore unico per ogni record;
- **school_id**: ID della scuola a cui il personale è assegnato;
- **job_id**: ID del lavoro specifico del personale (può essere nullo);
- **profile_id**: ID del profilo del personale;
- **departement_id**: ID del dipartimento del personale (può essere nullo);
- **first_serving_date**: Data di inizio del servizio del personale nella scuola;
- **is_school_admin**: Boleano che indica se il personale è amministratore della scuola.

A conclusione di questa sezione, si riporta lo schema relazionale dei dati finora descritti:

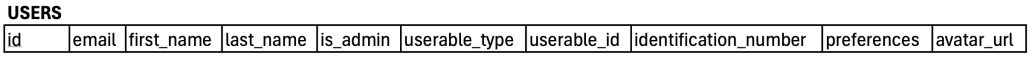

Figura 3.6: Schema relazionale di Users

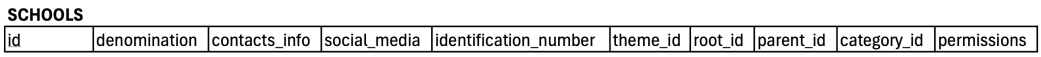

Figura 3.7: Schema relazionale di Schools

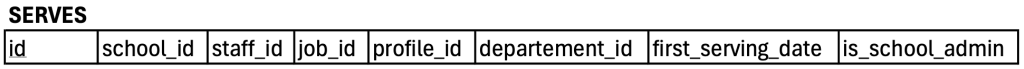

Figura 3.8: Schema relazionale di Serves

In tutti e tre i casi, `id` è la chiave primaria.

3.2.2 Inizializzazione del database

Il progetto di tirocinio, oggetto di questa tesi, è stato sviluppato principalmente utilizzando per il backend il linguaggio di programmazione Ruby e il framework Ruby on Rails.

Per poter testare ed utilizzare la piattaforma sul proprio dispositivo di sviluppo, è stato ovviamente necessario inizializzare il database e, più in generale, l'intero backend: per fare ciò, sono state sfruttate le pagine presenti su Confluence e appositamente realizzate per permettere un veloce ed efficace onboarding di nuovi membri del team di sviluppatori e tirocinanti.

Per avviare il progetto, è stato innanzitutto necessario installare Ruby e Ruby on Rails nell'ambiente di sviluppo che, nel caso dello scrivente, è MacOS su hardware utilizzante architettura ARM. Questo dettaglio viene specificato perchè ciò ha richiesto delle fasi aggiuntive di `troubleshooting` che hanno reso il processo di inizializzazione del progetto leggermente più tedioso di quando viene utilizzato un dispositivo dotato di processore con architettura x64 o x84.

Dopo aver risolto le problematiche causate dall'architettura ARM ed aver installato strumenti fondamentali come Visual Studio Code (già adoperato per vari insegnamenti del corso di studi) e qualche estensione per facilitare la scrittura del codice (come, ad esempio, `Auto Close Tag` per automatizzare la chiusura dei tag HTML/XML, `Beautify css/sass/less` per formattare automaticamente il codice CSS per renderlo più leggibile e coerente e `Bracket Pair Colorizel` per assegnare colori diversi alle coppie di parentesi corrispondenti), si è proceduto con la creazione di un account Bitbucket che, come descritto prima, è il sito su cui è stato caricato il progetto e con cui, grazie a Git, si sono sviluppate le funzionalità richieste ed è stato condiviso il codice.

A seguito della creazione dell'account, è stato necessario configurare `SSH (Secure Shell)`, protocollo sicuro di accesso remoto che consente di stabilire connessioni sicure tra computer su una rete non protetta come Internet. Alcuni dei vantaggi dell'utilizzo di SSH per interagire con Bitbucket, rispetto alle semplici credenziali

HTTPS, è quello di utilizzare coppie di chiavi crittografiche eliminano la necesità di inserire ripetutamente la password e, conseguentemente, offrire una maggiore sicureza ed automatizzare attività come il psuh e il pull del codice. Ciò è particolarmente utile in scenari di integrazione continua e deployment continuo (`CI/CD`).

Per configurare la connessione SSH sono stati eseguiti i seguenti comandi, che hanno prodotto la chiave pubblica `id_rsa.pub` che poi è stata caricata nelle impostazioni dell'account BitBucket:

```
1  ssh-keygen
2  ssh-add ~/.ssh/id_rsa
```

Listing 3.6: Generazione della chiave SSH

Dopo l'inizializzazione della chiave SSH, si è proceduto con l'installazione di Ruby nella sua versione 3.0.2 e di Ruby on Rails 6.1.4, per poi proseguie con PostgreSQL e pgAdmin (strumento importante avere una GUI - Graphic User Interface - da cui poter vedere rapidamente il database del progetto).

A tutto ciò è seguita la clonazione dalla repository remota di BitBucket della piattaforma di E-learning sul dispositivo locale e la creazione di un file di testo semplice `.env` che contiene variabili d'ambiente. Queste variabili sono coppie chiave-valore che memorizzano informazioni di configurazione come chiavi API, password del database, URL di servizi esterni e altre impostazioni specifihe dell'ambiente in cui l'applicazione viene eseguita.

Una volta terminata la realizzazione del file `.env`, sono state installate le dipendenze di progetto (chiamate `gemme` o `gems` in Ruby, pacchetti di codice che aggiungono funzionalità) e `bundler`, strumento che facilità la gestione delle suddette dipendenze sfruttando un file chiamato `Gemfile`, contente l'elenco di tutte le gemme di cui il progetto ha bisogno.

A questo punto si è arrivati alla fase effettiva di inizializzazione del database locale con il comando

```
1  rails db:create
```

Listing 3.7: Inizializzazione del DB locale

E la creazione delle tabelle con

```
1 rails db:migrate
2 rails db:migrate RAILS_ENV=test
```

Listing 3.8: Creazione tabelle

Ed infine

```
1 rails server -p 3002
```

Listing 3.9: Avviamento del progetto

Per eseguire l'applicativo in un web browser.

3.2.3 Implementazione

Prima di parlare dell'effettiva implementazione del modulo di gestione dei permessi utente, occorre descrivere alcune gemme fondamentali: `Devise`, `Pundit` e `Rolify`. Di seguito verranno illustrate le caratteristiche principali di ciascuna gemma e le modalità di utilizzo nel contesto dell'applicazione.

Devise

Devise [9] è una gemma per Ruby on Rails che facilita la gestione dell'autenticazione degli utenti. Fornisce un set completo di moduli per gestire tutte le funzionalità legate all'autenticazione, come registrazione, login, recupero password, conferma dell'email, ecc.

Caratteristiche principali:

1. **Autenticazione**: gestisce il login e la registrazione degli utenti;
2. **Recupero password**: permette agli utenti di recuperare la password dimenticata;
3. **Conferma account**: invia email di conferma per la verifica dell'account;
4. **Timeout**: gestisce la scadenza delle sessioni degli utenti inattivi.

Di seguito i comandi rilevanti per utilizzare la gemma:

```
1 gem 'devise'                    # Aggiunta gemma al 'Gemfile'
2 bundle install                  # Installazione della gemma
3 rails generate devise:install   # Configurazione di Devise
4 rails generate devise User      # Generazione modello User
5 rails db:migrate                # Esecuzione delle migrazioni
```

Listing 3.10: Utilizzo di Devise

Devise aggiunge automaticamente le rotte necessarie per l'autenticazione. Nel file config/routes.rb:

```
1    Rails.application.routes.draw do
2      devise_for :users
3      # Altre rotte
4    end
```

Listing 3.11: Configurazione delle rotte

Pundit

Pundit [10] è una gemma per la gestione delle autorizzazioni in un'applicazione Ruby on Rails. Utilizza una combinazione di plain Ruby e oggetti policy per autorizzare le azioni degli utenti.

Caratteristiche principali:

1. **Policy**: ogni modello ha una policy associata che definisce chi può fare cosa.
2. **Scopes**: controllano quali record un utente può vedere.
3. **Facile integrazione**: si integra facilmente con i controller di Rails.

I comandi per utilizzarla sono simili a quelli di Devise:

```
1  gem 'pundit'                    # Aggiunta gemma al 'Gemfile'
2  bundle install                  # Installazione della gemma
3  rails generate pundit:install # Configurazione di Devise
4  rails generate pundit:policy Post # Generazione policy
```

Listing 3.12: Utilizzo di Pundit

Nel file app/policies/post_policy.rb:

```
class PostPolicy < ApplicationPolicy
  def show?
    user.admin? || record.published?
  end

  def update?
    user.admin? || (record.user == user)
  end
end
```

Listing 3.13: Definizione delle autorizzazioni nella policy

Nei controller:

```
  class PostsController < ApplicationController
    before_action :set_post, only: [:show, :edit, :update,
       :destroy]
    before_action :authenticate_user!
    after_action :verify_authorized, except: :index

    def show
      authorize @post
    end

    def update
      authorize @post
      if @post.update(post_params)
        redirect_to @post, notice: 'Post was successfully
           updated.'
      else
        render :edit
      end
    end

    private
```

```
20
21      def set_post
22        @post = Post.find(params[:id])
23      end
24    end
```

Listing 3.14: Utilizzo delle policy nei controller

Rolify

Rolify [11] è una gemma per la gestione dei ruoli degli utenti che permette di assegnare ruoli a diversi utenti e di gestire le autorizzazioni in base a questi ruoli.

Caratteristiche principali:

1. **Assegnazione dei ruoli**: consente di assegnare e rimuovere ruoli agli utenti;
2. **Gerarchia dei ruoli**: supporta una gerarchia di ruoli complessa;
3. **Integrazione**: si integra facilmente con Devise e Pundit.

L'implementazione segue lo stesso pattern delle gemme precedenti, con l'unica differenza nel seguente comando di configurazione:

```
1  ...
2  rails generate rolify Role User
3  ...
```

Listing 3.15: Configurazione di Rolify

Mentre nel modello `User` avviene l'assegnazione dei ruoli:

```
1  class User < ApplicationRecord
2    rolify
3  end
```

Listing 3.16: Configurazione di Rolify

Esempio di assegnazione e verifica dei ruoli:

```
1  user = User.find(1)
2  user.add_role :admin
3  user.has_role? :admin # => true
```

Listing 3.17: Esempio assegnazione ruoli Rolify

Ora che si è descritto il funzionamento di Devise, Pundit e Rolify, è possibile passare all'implementazione del codice relativo al modulo per la gestione dei permessi utente all'interno del portale E-learning: più nello specifico, il focus sarà su alcuni modelli e controller che hanno permesso l'effettivo raggiungimento dell'obiettivo prefissato inizialmente.

Modello `user.rb`

Il modello `User` include varie funzioni per la gestione dei permessi degli utenti. Di seguito vengono illustrate le principali funzioni relative ai permessi.

```
1  def generate_permissions
2    permissions = {}
3
4    self.roles.each do |role|
5      resource_type = role["resource_type"]
6      resource_action = role["name"].to_sym
7
8      next unless resource_type # Skip roles without a
           resource_type
9
10     permissions[resource_type] ||= {}
11     if resource_action == :destroy
12       permissions[resource_type]["delete"] = true
13     else
14       permissions[resource_type][resource_action] = true
15     end
16   end
17
18   { permissions: permissions }
19 end
```

Listing 3.18: Funzione `generate_permissions`

Questa funzione genera i permessi per l'utente in base ai ruoli assegnati: per ogni ruolo, viene determinato il tipo di risorsa, l'azione permessa e e viene costruita una struttura di permessi.

```
def user_permissions_in_school(school_id)
  profile_code = self.user_profile(school_id)
  school = School.find(school_id)
  all_permissions = school[:permissions][profile_code]["
     permissions"]
  permissions = {}
  all_permissions.each do |permision|
    controller, action, right = permision.split(":")
    if self.has_role? "#{controller}:#{action}", school
      permissions[controller] ||= {}
      permissions[controller][action.to_sym] = true
    else
      permissions[controller] ||= {}
      permissions[controller][action.to_sym] = false
    end
  end

  { school_id: school_id, permissions: permissions }
end
```

Listing 3.19: Funzione `user_permissions_in_school`

Questa funzione restituisce i permessi dell'utente per una determinata scuola, che vengono verificati in base ai ruoli dell'utente e alle configurazioni specifiche della scuola.

```
def edit_permissions_in_school(school_id, model, permissions)
  school = School.find(school_id)
  permissions.keys.each do |key|
    permission_name = model + ":" + key
    if permissions[key].to_s.downcase == "true"
      self.add_role permission_name, school
    else
      self.remove_role permission_name, school
    end
  end
```

```
11    self.save!
12  end
```

Listing 3.20: Funzione `edit_permissions_in_school`

Questa funzione modifica i permessi dell'utente in una determinata scuola, aggiungendo o rimuovendo ruoli basati sui permessi forniti.

```
1  def create_permissions(school_id)
2    school = School.find(school_id)
3    sub_schools = school.subtree
4    current_local = I18n.locale
5    I18n.locale = I18n.default_locale
6    u_profile = user_profile(school_id)
7
8    return unless u_profile
9
10   permissions = school.permissions[u_profile]["permissions"]
11   if permissions.any?
12     roles.destroy_all # I remove all existing permissions to
         reinitialise it
13     permissions.each do |permission_name|
14       sub_schools.each do |subschool|
15         controller, action, right = permission_name.split(":"
             )
16         self.add_role "#{controller}:#{action}", subschool if
             right == "T"
17       end
18     end
19   end
20   I18n.locale = current_local
21 end
```

Listing 3.21: Funzione `create_permissions`

Questa funzione crea i permessi per l'utente in base alle configurazioni della scuola e dei suoi sottosistemi, assegnando i ruoli appropriati.

Controller `users_controller.rb`

Il controller `UsersController` include varie azioni per la gestione dei permessi degli utenti. Di seguito vengono illustrate le principali azioni relative ai permessi.

```
def add_role
  manage_role("add", @user)
end
```

Listing 3.22: Azione `add_role`

Questa azione aggiunge un ruolo a un utente.

```
def remove_role
  manage_role("remove", @user)
end
```

Listing 3.23: Azione `remove_role`

Questa azione rimuove un ruolo da un utente.

```
def create_permissions
  if @user
    @user.create_permissions(params[:school_id])
    render json: { message: "Permissions created successfully
       ." }, status: :ok
  else
    render json: { message: "User not found." }, status: :
       not_found
  end
end
```

Listing 3.24: Azione `create_permissions`

Questa azione crea i permessi per un utente specifico, chiamando il metodo `create_permissions` del modello `User`.

```
def edit_permissions
  if @user
    @user.edit_permissions_in_school(params[:school_id],
       params[:model], params[:permissions])
```

```
4     render json: { message: "Permissions updated successfully
          ." }, status: :ok
5   else
6     render json: { message: "User not found." }, status: :
          not_found
7   end
8 end
```

Listing 3.25: Azione `edit_permissions`

Questa azione modifica i permessi di un utente specifico, chiamando il metodo `edit_permissions_in_school` del modello `User`.

Modello `school.rb`

Il modello `School` rappresenta una scuola e gestisce le sue relazioni, convalide e funzioni principali. Di seguito vengono descritti i principali aspetti della gestione dei permessi, a partire dai metodi per gestire i permessi associati ai profili del personale all'interno della scuola. Questi metodi permettono di definire, modificare e recuperare i permessi assegnati a ciascun profilo.

```
1 class School < ApplicationRecord
2   # ... altre relazioni, convalide e metodi ...
3
4   # Modifica i permessi di un profilo in una scuola
5   def edit_profile_permissions_in_school(profile_id, model,
        new_permissions)
6     profile_code = find_profile_code_by_id(profile_id)
7     profile_permissions = permissions[profile_code]["
          permissions"]
8     formatted_new_permissions = get_new_permissions(model,
          new_permissions)
9     formatted_new_permissions.each do |permission_name|
10      if profile_permissions.include?(permission_name)
11        next
12      elsif profile_permissions.include?(neg_permission(
            permission_name))
```

```
13          profile_permissions[profile_permissions.index(
                neg_permission(permission_name))] =
                permission_name
14        else
15          profile_permissions << permission_name
16        end
17        permissions[profile_code]["permissions"] =
             profile_permissions
18        save!
19      end
20    end
21
22    # Metodo per recuperare il codice del profilo basato su un
         ID
23    def find_profile_code_by_id(profile_id)
24      permissions.each do |code, hash|
25        return code if hash["id"] == profile_id.to_i
26      end
27      nil # Restituisce nil se non viene trovato un hash
           corrispondente
28    end
29
30    # Metodo per recuperare i permessi associati a un dato
         profilo
31    def permissions_by_profile_id(profile_id)
32      profile_code = find_profile_code_by_id(profile_id)
33      permissions[profile_code]["permissions"]
34    end
35
36    # Restituisce i permessi del profilo
37    def profile_permissions(profile_id)
38      permissions = {}  # Inizializza un hash vuoto per
           memorizzare i permessi
39      current_local = I18n.locale
40      I18n.locale = I18n.default_locale
41      profile_code = find_profile_code_by_id(profile_id)
```

```
42      I18n.locale = current_local
43
44      self.permissions[profile_code]["permissions"].each do |
           permission|
45        # Divide il nome del permesso in controller e azione (
             es. "attachment:index" => ["attachment", "index"])
46        controller, action, right = permission.split(":")
47        # Inizializza il controller nell'hash dei permessi se
             non esiste
48        permissions[controller] ||= {}
49        # Imposta l'azione (index, show, create, update, delete
             ) a true nell'hash dei permessi
50        permissions[controller][action.to_sym] = true if right
             == "T"
51        permissions[controller][action.to_sym] = false if right
             == "F"
52      end
53
54      {
55        school_id: id,
56        permissions: permissions
57      }
58    end
59
60    private
61
62    # Restituisce il permesso negativo corrispondente a un
         permesso dato
63    def neg_permission(permission_name)
64      controller, action, right = permission_name.split(":")
65      temp = controller + ":" + action
66      neg_permission_name = right == "T"? temp + ":F" : temp +
           ":T"
67    end
68
69    # Format the new permissions for a specific model
```

```
70    def get_new_permissions(model, permissions)
71      new_permissions = []
72      permissions.keys.each do |key|
73        permission_name = model + ":" + key
74        if permissions[key].to_s.downcase == "true"
75          permission_name += ":T"
76        else
77          permission_name += ":F"
78        end
79        new_permissions << permission_name
80      end
81      new_permissions
82    end
83
84  end
```

Listing 3.26: Modello School: gestione dei permessi

Metodi:

- `edit_profile_permissions_in_school(profile_id, model, new_permissions)`: modifica i permessi di un profilo in una scuola.
- `find_profile_code_by_id(profile_id)`: recupera il codice del profilo basato su un ID;
- `permissions_by_profile_id(profile_id)`: recupera i permessi associati a un dato profilo;
- `profile_permissions(profile_id)`: restituisce i permessi del profilo;
- `neg_permission(permission_name)`: restituisce il permesso negativo corrispondente a un permesso dato;
- `get_new_permissions(model, permissions)`: formatta i nuovi permessi per poterli fornire al frontend nel modo appropriato.

Capitolo 4

Test delle funzionalità

Il processo di test è una fase cruciale dello sviluppo software, in quanto garantisce che le funzionaltà si comportino come previsto e che l'applicazione nel suo complesso sia affidabile e priva di errori. In questo capitolo, verranno descritte le strategie e gli strumenti utilizzati per testare le funzionalità del modulo di gestione dei permessi utente.

Test del frontend

Per il frontend sono stati utilizzati test unitari per verificare il corretto funzionamento dei componenti Vue.js. Il framework di testing Jest, in combinazione con Vue Test Utils, è stato impiegato per scrivere ed eseguire questi test.

`Framework di Testing Jest`

Jest è un framework di testing sviluppato da Facebook, ampiamente utilizzato per applicazioni JavaScript. È conosciuto per la sua semplicità e velocità, offrendo un'ampia gamma di funzionalità per la scrittura e l'esecuzione di test unitari. Alcune delle caratteristiche principali di Jest includono:

1. **Test runner**: Jest è un test runner che esegue i test in modo parallelo, aumentando la velocità complessiva del processo di testing;

2. **Mocking**: Jest consente di creare mock di moduli e funzioni, facilitando il test di unità isolate senza dipendenze esterne;

3. **Snapshot testing**: Questa funzionalità permette di catturare lo stato del componente in un dato momento e di confrontarlo con gli stati successivi per identificare cambiamenti imprevisti;

4. **Coverage**: Jest fornisce rapporti dettagliati sulla copertura del codice, aiutando a identificare le parti del codice che non sono state testate.

Vue Test Utils

`Vue Test Utils` è una libreria ufficiale per il testing dei componenti Vue.js. Fornisce una serie di utility per montare e interagire con i componenti, permettendo di simulare vari scenari di utilizzo. Le principali funzionalità di Vue Test Utils includono:

1. **Mounting**: Permette di montare i componenti Vue.js in modo che possano essere testati in isolamento.

2. **Wrapper API**: Fornisce un'API per interagire con i componenti montati, permettendo di accedere e manipolare il DOM del componente.

3. **Simulazione di eventi**: Consente di simulare eventi dell'utente come click, input, e submit, per verificare il comportamento del componente in risposta a tali eventi.

4. **Slot testing**: Supporta il testing degli slot dei componenti, assicurando che i contenuti passati tramite slot vengano renderizzati correttamente.

Copertura dei test unitari

I test unitari hanno coperto vari aspetti del comportamento dei componenti Vue.js, tra cui:

1. **Rendering dei componenti**: Verifica che i componenti Vue.js vengano renderizzati correttamente con i dati forniti. Questo include il controllo che il markup HTML generato corrisponda alle aspettative quando i componenti ricevono differenti proprietà (props) e dati;

2. **Interazione dell'utente**: Simulazione di eventi dell'utente (click, input, ecc.) e verifica che i componenti rispondano correttamente. Questo tipo di test assicura che i componenti reagiscano correttamente alle azioni dell'utente;

3. **Stato dei componenti**: Verifica che lo stato dei componenti Vue.js venga aggiornato correttamente in base alle interazioni dell'utente e ai dati ricevuti dal backend. Questo assicura che la logica interna del componente funzioni come previsto.

Test del backend

`Insomnia` è stato lo strumento principale utilizzato per verificare il corretto funzionamento delle chiamate eseguite dal frontend al backend. Insomnia è un'applicazione open-source progettata per semplificare lo sviluppo e il debug delle API. È particolarmente apprezzata per la sua interfaccia intuitiva e le sue potenti funzionalità, che facilitano il lavoro degli sviluppatori di backend in diversi modi.

Una delle funzionalità fondamentali di Insomnia è la capacità di inviare richieste HTTP al server backend: questo permette di testare facilmente le varie endpoint delle API. Gli sviluppatori possono configurare richieste di tipo GET, POST, PUT, DELETE, e altri metodi HTTP, specificando tutti i dettagli necessari come URL, headers, parametri di query, e body della richiesta. Insomnia visualizza le risposte del server in modo chiaro e strutturato, facilitando l'analisi dei dati restituiti. Questo è particolarmente utile per debug e troubleshooting, permettendo di identificare rapidamente eventuali problemi o errori nelle API.

Oltre all'invio di richieste manuali, Insomnia consente la creazione di test automatici per verificare il comportamento delle API nel tempo. Gli sviluppatori possono definire una serie di richieste e asserzioni per garantire che le API funzionino come previsto, anche dopo modifiche al codice. Questi test possono essere eseguiti regolarmente per assicurarsi che nuove modifiche non introducano regressioni o bug.

Insomnia, infine, supporta diversi formati di dati sia per le richieste che per le risposte, tra cui JSON e XML. Questo lo rende uno strumento molto versatile, in grado di gestire API che utilizzano vari formati di dati. La capacità di visualizzare le risposte in un formato leggibile e organizzato aiuta a comprendere meglio la struttura e il contenuto dei dati trasmessi, migliorando l'efficienza nel debug e

nello sviluppo.

Test manuali

Oltre ai test automatici, sono stati eseguiti anche test manuali per verificare aspetti come l'usabilità, l'accessibilità e la compatibilità cross-browser dell'applicazione. Questi test sono stati condotti dagli sviluppatori che, durante la scrittura del codice, hanno in contemporanea provato l'applicativo - anche tramite mezzi forniti dai browser come i `DevTools`, strumenti che permettono di analizzare, modificare e debuggare pagine web.

Capitolo 5

Conclusione

Il presente elaborato ha approfondito il lavoro svolto durante la collaborazione con SYSAIT, focalizzandosi sull'implementazione di un modulo cruciale per la gestione dei permessi utente all'interno della piattaforma E-learning. Questo modulo rappresenta un tassello fondamentale nel più ampio ecosistema della piattaforma, consentendo una gestione efficiente e sicura degli accessi alle diverse funzionalità.

I risultati conseguiti nel corso di questo progetto sono stati soddisfacenti, rispondendo in modo completo ai requisiti funzionali e non funzionali delineati all'inizio del percorso. L'interfaccia utente, realizzata con l'ausilio di Quasar e Vue.js, si distingue per la sua chiarezza e intuitività, facilitando l'interazione sia su desktop che su dispositivi mobili. Gli amministratori scolastici possono ora gestire i permessi degli utenti in modo semplice e immediato, senza dover affrontare complesse procedure o configurazioni.

La gestione dei permessi, inoltre, è stata progettata per essere granulare: gli amministratori hanno la possibilità di assegnare o revocare autorizzazioni specifiche non solo a singoli utenti, ma anche a interi profili, offrendo un livello di controllo molto elevato. Questa flessibilità si traduce in una maggiore capacità di personalizzazione, consentendo a ciascun istituto scolastico di adattare la piattaforma alle proprie esigenze specifiche.

L'integrazione del modulo di gestione dei permessi nel backend Ruby on Rails è stata eseguita con successo, sfruttando le potenzialità del database PostgreSQL per garantire la persistenza e l'integrità dei dati. Inoltre, sono stati implementati

rigorosi controlli delle autorizzazioni in ogni sezione della piattaforma, assicurando che gli utenti possano accedere solo alle funzionalità per cui dispongono dei permessi necessari.

Grazie all'utilizzo di tecnologie consolidate e affidabili come Ruby on Rails e PostgreSQL, il sistema è stato progettato per essere scalabile e manutenibile nel tempo. Questo significa che la piattaforma potrà crescere ed evolversi insieme alle esigenze degli istituti scolastici, senza richiedere interventi invasivi o costosi.

Le prospettive future per questo sistema sono promettenti, in futuro infatti si potrebbero esplorare l'implementazione di funzionalità avanzate come la gestione di ruoli più complessi, l'auditing delle attività degli utenti e la possibilità di personalizzare ulteriormente i permessi in base a criteri specifici.

L'esperienza maturata durante lo sviluppo di questo progetto è stata estremamente formativa, consentendo di mettere in pratica le conoscenze acquisite durante il percorso di studi e di confrontarsi con sfide tecniche stimolanti. Il supporto costante del team di SYSAIT è stato fondamentale per il successo di questo lavoro.

Bibliografia

[1] *System Afrik Information Technology*. 4 Mar. 2024. URL: http://sysait-prod.herokuapp.com/.

[2] *Ruby*. 25 Mar. 2024. URL: https://ruby-doc.org/3.2.2/index.html.

[3] *Ruby on Rails*. 30 Mar. 2024. URL: https://guides.rubyonrails.org/index.html.

[4] *PostgreSQL*. 2 Apr. 2024. URL: https://www.postgresql.org/about/.

[5] *Vue.js*. 7 Apr. 2024. URL: https://vuejs.org/.

[6] *Quasar*. 7 Apr. 2024. URL: https://quasar.dev/docs.

[7] *Git*. 10 Apr. 2024. URL: https://git-scm.com/.

[8] *Confluence*. 11 Apr. 2024. URL: https://www.atlassian.com/software/confluence.

[9] *Devise*. 17 Apr. 2024. URL: https://github.com/heartcombo/devise.

[10] *Pundit*. 18 Apr. 2024. URL: https://github.com/varvet/pundit.

[11] *Rolify*. 19 Apr. 2024. URL: https://github.com/RolifyCommunity/rolify.

Ringraziamenti

A mia madre, che in questi anni ha potuto e saputo sostenermi, permettendomi di perseguire i miei sogni e le mie ambizioni.

Al dottor Arnaud Nguembang Fadja, all'ingegnere Wilfried Ndomo Kenfack e al professor Fabrizio Riguzzi, che in questi ultimi mesi di tirocinio e stesura della presente tesi di laurea mi hanno guidato con professionalità e disponibilità.