Politecnico di Torino

Politecnico Di Torino

Ingegneria Informatica
A.a. 2020/2021
Sessione di laurea Dicembre 2021

SQL Interactive: applicazione di supporto per corsi di basi di dati

Relatori

Luigi De Russis

Laura Farinetti

Candidato

Edoardo Convertini

Sommario

Lo svolgimento di esercizi per il corso di basi di dati da parte di studenti in remoto ha da sempre presentato numerose criticità, in quanto veniva richiesto allo studente di ricreare sulla propria macchina un ambiente di sviluppo idoneo.
In questa tesi verrà quindi proposta un'alternativa, presentando un'applicazione web sviluppata con React e NodeJs che, interfacciandosi con un database PostgreSQL, permetta al docente la gestione di esercizi e laboratori, e lo svolgimento da parte di studenti senza la necessità di installare software aggiuntivi.
Il docente del corso di studio avrà la possibilità di creare o caricare nuovi esercizi interagendo direttamente con il database per la creazione e il popolamento delle tabelle necessarie, che verranno rese disponibili agli studenti abilitati senza la possibilità di essere modificate. Gli stessi esercizi potranno essere resi disponibili allo studente anche sotto forma di laboratori, dove sarà presente un canale di comunicazione con il docente in caso di necessità durante lo svolgimento.
L'applicazione web permetterà inoltre la raccolta automatica di statistiche utili a valutare l'efficacia del corso di studio e sull'utilizzo dell'applicazione, attraverso analisi sui principali errori commessi, la percentuale di esercizi risolti correttamente e i tentativi effettuati.

Ringraziamenti

Concludendo con questa tesi il mio percorso universitario, vorrei prima di tutto ringraziare il professore Luigi De Russis e la professoressa Laura Farinetti, che mi hanno permesso di lavorare su un progetto interessante e mi hanno sempre supportato con precisione, professionalità e gentilezza.

Grazie per il supporto ricevuto in questi anni da tutta la mia famiglia allargata, super allargata, ma vorrei rubare un paio di righe di questa tesi per ringraziare qualcuno in particolare:

Grazie a mia mamma, per la serenità e la spensieratezza che porti nella mia vita, e perché oltre ad essere tuo figlio, la cosa più bella è che ti assomiglio. Ti meriti di avere tutto il meglio dalla vita. Tranne la televisione in sala la sera se ci sono io.

Grazie a mio papà, per non aver mai dubitato in me e perché con la nostra gara a chi è più testardo mi ha insegnato a non mollare mai. Qualcuno potrà dire che lo siamo un po' troppo, ma sarà lui ad essere in errore.

Grazie a nonna Bettina, sento che a forza di candele accese e preghiere sei stata il principale motivo per cui sia riuscito a passare la maggior parte degli esami. E sopratutto perché aver imparato a spiegare ad altre persone i miei studi vale più di qualsiasi laurea.

Grazie alla mia Chiara, entrata nella mia vita a causa della mia passione per l'informatica, rimasta nonostante la mia passione per l'informatica, e principale motivo per cui questa tesi risulterà scritta in un italiano non da quinta elementare. Non vedo l'ora di scoprire cosa succederà ci riserverà il futuro da ora in poi.

Vorrei però dedicare questo piccolo traguardo a mia sorella Vittoria, allo stesso tempo la persona con il peggiore e il migliore carattere del mondo. Una sensibilità esagerata e una dolcezza fuori dal comune. A prescindere dall'età, dall'altezza e da quello che succederà nelle nostre vite, rimarrai sempre la mia sorellina. E non smetterò mai di essere geloso di chiunque ti si avvicini.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Presente come insegnamento obbligatorio nei corsi di Laurea Triennale di Ingegneria Informatica e Gestionale presso il Politecnico di Torino, il corso di basi di dati si pone come obiettivo principale quello di descrivere i sistemi per la gestione di database, considerando sia le metodologie di progettazione di basi di dati, sia lo sviluppo di applicazioni di interrogazione e gestione di basi di dati.

Negli ultimi anni, anche a causa di una improvvisa modalità di didattica esclusivamente remota, sono emerse numerose criticità circa lo svolgimento di esercizi e laboratori.

Trattandosi di un corso di studio prettamente pratico, in particolare durante la prima parte di corso dove viene trattato il linguaggio SQL, sia docenti che studenti si sono trovati in difficoltà sulla gestione del corso e sul suo insegnamento.

Lo studente infatti, che prima aveva a disposizione sessioni di laboratorio in presenza assistite da docenti e assistenti, si è trovato a dover replicare l'ambiente di sviluppo presente in laboratorio sul proprio computer, senza la possibilità, se non in rari casi, di ricevere un'adeguata valutazione e correzione.

Di conseguenza, il docente del corso troverà difficoltà a valutare la reale efficacia dell'insegnamento prima dello svolgimento di un esame di fine corso, se non con l'assegnazione di elaborati da svolgere singolarmente.

Queste condizioni hanno portato, sopratutto negli ultimi due anni, a un notevole aumento della difficoltà percepita dagli studenti del corso.

1.1 Obiettivo

La tesi si pone come obiettivo l'analisi, la progettazione e il relativo sviluppo di un'applicazione web di supporto per i corsi di studio di basi di dati del Politecnico di Torino.

L'applicazione avrà lo scopo di semplificare lo svolgimento di esercizi pratici da parte degli studenti, mettendo a disposizione un'interfaccia su cui eseguire query interfacciandosi direttamente con un database, e potrà essere utilizzata sia in laboratorio sia in remoto.

Utilizzando l'applicazione per lo svolgimento di esercizi verranno raccolte statistiche utili a valutare l'efficacia e migliorare l'apprendimento del corso di studio.

Per lo sviluppo dovrà essere utilizzata un'architettura web, in modo da non rendere necessaria l'installazione da parte dello studente di software aggiuntivi e diminuire il carico di lavoro.

1.2 Struttura della tesi

La tesi, sviluppata in più capitoli, sarà quindi cosi composta:

- Capitolo 2: fornirà concetti utili a contestualizzare il progetto, e introdurrà gli strumenti utilizzati
- Capitolo 3: verrà descritta l'architettura dell'applicazione sviluppata, dettagliandone le funzionalità proposte e fornendone per ognuna un caso d'uso reale
- Capitolo 4: tratterà l'effettiva implementazione dell'applicazione web, descrivendone le rilevanti scelte implementative
- Capitolo 5: verrà valutato il possibile effettivo utilizzo dell'applicazione, e presi in considerazione eventuali sviluppi futuri
- Capitolo 6: riassumerà gli obiettivi della tesi valutando i risultati ottenuti

Capitolo 2

Background e analisi generale

2.1 Introduzione ai corsi di basi di dati

Come introdotto nel capitolo precedente, il corso di basi di dati è presente come insegnamento obbligatorio nei corsi di Laurea Triennale di Ingegneria Informatica e Gestionale presso il Politecnico di Torino, ponendosi come obiettivo principale quello di descrivere i sistemi per la gestione di basi di dati. Durante il corso vengono prese in considerazione e analizzate sia le metodologie di progettazione, sia lo sviluppo di applicazioni di interrogazione e gestione di basi di dati.

Il corso prevede sia una componente di spiegazioni tecniche frontali più tradizionali che permette di acquisire le conoscenze e le competenze fondamentali, sia esercitazioni in laboratorio inerenti gli argomenti trattati nelle lezioni teoriche. In particolare i temi principali delle esercitazioni riguardano nella prima parte di corso il linguaggio SQL e nella seconda parte l'algebra relazionale e la progettazione concettuale e logica di una base di dati.

Con l'avvento della pandemia da Covid-19, e il conseguente trasferimento delle attività didattiche in remoto, lo svolgimento di tali esercitazioni, si è rivelato particolarmente complesso. In particolare, veniva richiesto allo studente di ricreare sulla propria macchina un ambiente idoneo allo sviluppo di esercizi, che comprendeva l'installazione di un database e un'interfaccia grafica per le interrogazioni. Questi meccanismi comportavano un notevole aumento del carico di lavoro da parte dello studente per il setup iniziale del laboratorio, che differiva in base al sistema operativo utilizzato dallo studente, e imponeva limiti critici sulle correzioni di esercizi.

2.2 Scenari d'uso

Diventa quindi necessario considerare un'alternativa che permetta di migliorare l'apprendimento di SQL e facilitarne l'insegnamento all'interno dei corsi di studio.

Partendo quindi dai problemi precedente definiti, è possibile delineare una serie di possibili scenari d'uso:

- Facilitare lo svolgimento di esercizi da parte degli studenti, senza la necessità di installare software aggiuntivi sul proprio computer
- Affiancare lo studente durante la preparazione necessaria per affrontare un esame: in particolare potrà risultare utile lo svolgimento di precedenti temi d'esame caricati sull'applicazione
- Autovalutazione delle competenze: lo studente potrà svolgere autonomamente più esercizi caricati dal docente, e verificare quanto effettivamente appreso durante le lezioni teoriche del corso di studio
- Partecipazione a laboratori didattici in remoto: attraverso l'utilizzo dell'applicazione lo studente potrà partecipare a una sessione di laboratorio senza la necessità di essere presente in sede, con la possibilità di comunicare direttamente con il docente. La presenza di questo canale di comunicazione permette infatti di ridurre un secondo problema dello svolgimento del laboratorio da remoto. Da un lato gli studenti avranno maggiori possibilità di chiedere chiarimenti e delucidazioni durante lo svolgimento del laboratorio; dall'altro lato i docenti potranno più agevolmente correggere gli esercizi e fornire un feedback del lavoro svolto.
- Facilitare l'analisi da parte del docente dell'effettiva efficacia degli insegnamenti teorici, attraverso la visualizzazione grafica di statistiche relativa allo svolgimento di esercizi da parte degli studenti
- Contribuire a formare una solida raccolta di esercizi di difficoltà crescente da mantenere nel corso degli anni
- Sviluppare un agente conversazionale che sia in grado di interagire con lo studente in caso di problemi durante lo svolgimento di un esercizio, interpretando l'errore ricevuto dal database e traducendolo in un errore di più alto livello.
- Mostrare allo studente una panoramica sul livello corrente di apprendimento, attraverso una dashboard riassuntiva degli errori più frequenti durante lo svolgimento di esercizi

2.3 Strumenti di sviluppo

Per lo sviluppo di SQL Interactive è stato scelto un ambiente interamente basato su Javascript, utilizzando Node.js per la parte server e React per la parte frontend.

La scelta è ricaduta su queste due librerie perché, oltre ad essere particolarmente adatte per progetti di sviluppo web, vengono affrontate durante il percorso di Laurea Magistrale in Ingegneria Informatica; questo permetterà, in futuro, a studenti e tesisti del Politecnico di Torino di modificare l'applicazione e migliorarla in base alle esigenze che emergeranno nei prossimi anni.

2.3.1 Javascript

Linguaggio di programmazione non compilato con una sintassi simile a quella di Java e C, Javascript [1] venne creato nel 1995 da Brendan Eich per interagire con elementi HTML e rendere quindi interattive pagine web.

Il primo browser a implementarne l'utilizzo fu Internet Explorer 3.0 l'anno successivo, sotto il nome di JScript, e successivamente venne poi implementato da tutti i browser web incorporando un interprete dedicato.

Nel corso degli anni sono state effettuate più standardizzazioni del linguaggio di parte di ECMAScript (o ES), arrivando all'undicesima versione rilasciata a giugno 2020.

In ognuna delle standardizzazioni finalizzate sono state aggiunte nuove funzionalità non presenti in origine, come la tipizzazione di dati la gestione di funzioni asincrone (aggiunte rispettivamente nella terza e nella sesta versione), e meglio documentate/corrette funzioni esistenti.

Javascript è alla base della maggior parte di tutti i framework di sviluppo frontend, utilizzato infatti da React, Vue.js, Svelte e Angular e nonostante il suo utilizzo fu originariamente pensato per un ambiente di sviluppo lato client, con l'avvento di Node.js iniziò ad essere utilizzato anche per lo sviluppo di architetture server.

E' possibile sviluppare con Javascript anche applicazione desktop e mobile attraverso l'utilizzo di specifici framework come Electron e React Native.

2.3.2 Node.js

Sviluppato nel 2009 da un team guidato da Ryan Dahl, Node.js [2] è un runtime Javascript costruito sull'interprete javascript di Google Chrome, chiamato anche V8.

All'epoca del suo sviluppo venne scelto questo particolare interprete in quanto ero distribuito in licenza open source e progettato per permettere performance elevate.

La principale caratteristica di Node.js, che lo distingue dalle altre architetture server, risiede nella gestione degli eventi. Al contrario di quanto avviene ad esempio in Java con la creazione di thread separati, o in generale nella programmazione concorrente, Node.js permette l'accesso alle risorse del sistema operativo in modalità event-driven. Ogni azione risulta quindi asincrona rispetto alle altre; viene, infatti, eseguita un'azione immediatamente in risposta a un input e non è necessario attendere il completamento del ciclo dell'azione precedente.

Le funzionalità base di Node.js sono inserite all'interno di moduli già compresi nell'installazione, ma basandosi su un linguaggio Javascript, si ha accesso completo a NPM (Node Package Manager), una repository di librerie scritte per essere utilizzate con Node.js che permette di ampliare i moduli utilizzabili.

2.3.3 Express

Le principale richieste di rete (GET, POST, DELETE..) non sono però direttamente implementate da Node.js, per questo durante lo sviluppo di SQL Interactive è stato utilizzato Express, framework web che consente di creare API di routing e di impostare middleware per rispondere alle richieste HTTP.

Express infatti, per semplificare lo sviluppo di servizi web, mette infatti a disposizione due componenti essenziali:

- Router [3]: fa riferimento alla definizione di endpoint dell'applicazione (URI) e alla loro modalità di risposta alle richieste del client. Express supporta tutti i principali metodi di routing (GET, POST, DELETE..); inoltre, possono essere definiti dei percorsi di rete statici o dinamici, attraverso l'uso di espressioni regolari.

- Middleware [4]: Le funzioni middleware sono funzioni con accesso all'oggetto richiesta (req), all'oggetto risposta (res) eseguite prima che il controller prenda in carico la richiesta. Queste funzioni posso essere utilizzate per intercettare chiamate web e verificarne ad esempio la presenza e validità di un token di autorizzazione, o per pre-processare dati inviati nella request.

Di seguito si potrà notare come la gestione di una chiamata di rete attraverso Express sia più chiara rispetto a uno sviluppo nativo in Node.js.

```
1  const http = require('http');
2
3  const hostname = '127.0.0.1';
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7    res.statusCode = 200;
8    res.setHeader('Content-Type', 'text/plain');
9    res.end('Hello World');
10 });
11
12 server.listen(port, hostname, () => {
13   console.log(`Server running at http://${hostname}:${port}/`);
14 });
```

Listing 2.1: Hello World in Node.js

```
1  const express = require('express')
2
3  const app = express()
4  const port = 3000
5
6  app.get('/', (req, res) => {
7    res.send('Hello World!')
8  })
9
10 app.listen(port, () => {
11   console.log(`Example app listening at http://localhost:${port}`)
12 })
```

Listing 2.2: Hello World con Express

2.3.4 React

Creata da Meta e rilasciata al pubblico nel 2013, React è una libreria Javascript che è stata in grado negli ultimi anni di diventare una delle principali librerie dedicate interamente allo sviluppo frontend.

La principale novità introdotta da React è l'utilizzo di un DOM virtuale [5], ovvero un'astrazione di quanto presente al momento in pagina, per migliorare le performance di renderizzazione dei componenti. Ogni volta che viene recepita una modifica da visualizzare sulla pagina web, viene prima aggiornato il DOM virtuale e confrontato con il precedente, in modo da poter identificare i cambiamenti effettivi da eseguire sul DOM vero e proprio, senza la necessità di aggiornare tutta la pagina. Il confronto tra i due DOM virtuali e estremamente veloce, e limita al minimo i cambiamenti, molto più lenti, da effettuare sul DOM reale.

Nell'ambito dello sviluppo, React permette di costruire interfacce utente complesse attraverso lo sviluppo di diversi componenti, definiti come funzioni Javascript.

```
1 function Greeting() {
2   return <h1>Ciao!</h1>;
3 }
```

Listing 2.3: Componente in React

La parte di codice restituita dalla funzione viene definita JSX [6], ed è un'estensione della sintassi Javascript che permette di emulare codice HTML per creare dei template dinamici, in quanto all'interno è possibile inserire qualsiasi espressione logica che verrà valutata e renderizzata nel codice.

L'utilizzo di JSX è consigliato ma non obbligatorio, in quanto ogni espressione JSX al momento della compilazione verrà comunque trasformato in codice Javascript interpretabile da React.

```
1 function Greeting() {
2   const name = "Edoardo";
3   return <h1>Hi, {name}!</h1>;
4 }
```

Listing 2.4: Template JSX

Uno dei punti di forza di React risiede nel poter suddividere componenti e logica complessa in componenti più semplici. Per permettere questo comportamento è però necessario che i componenti possano comunicare tra di loro attraverso il passaggio di dati. React incapsula gli attributi definiti dall'utente in un oggetto che prende il nome di "props" [7], utilizzabili dentro il componente.

In questo oggetto possono essere incapsulati sia semplici dati che funzioni, richiamabili dal componente che ne ha accesso.

```
1 function Greeting(props) {
2   return <h1>Ciao, {props.nome}</h1>;
3 }
4 ...
5 function App() {
6   return <Greeting name={"Edoardo"}/>;
7 }
```

Listing 2.5: Componenti e props

Durante lo sviluppo, potrebbe essere necessario che un dato debba essere salvato al livello di un componente. React, dalla versione 16.8, mette a disposizione gli Hooks [8], una particolare sintassi Javascript che permette di utilizzare gli stati (dati privati del componente) senza la necessità di scrivere una classe.

```
1  import React, { useState } from 'react';
2
3  function Esempio() {
4
5    const [contatore, setContatore] = useState(0);
6
7    return (
8      <div>
9        <p>Hai cliccato {contatore} volte</p>
10       <button onClick={() => setContatore(contatore + 1)}>
11         Cliccami
12       </button>
13     </div>
14   );
15 }
```

Listing 2.6: React Hooks

A partire da questa versione inoltre, è stata cambiata la gestione del ciclo di vita dei componenti: le funzioni componentDidMount e componentDidUpdate, presenti nella programmazione a classi di React, vengono sostituite da un Hook che prende il nome di useEffect [9]. Utilizzando questo Hook è possibile definire una o più funzioni da eseguire dopo che è stato effettuato un un render generico o di un singolo componente.

```
1 import { useEffect } from 'react';
2 function Greeting({ name }) {
3   const message = `Ciao, ${name}!`;
4   useEffect(() => {
5     document.title = `Benvenuto ${name}`;
6   }, [name]);
7   return <div>{message}</div>;
8 }
```

Listing 2.7: React useEffect

Utilizzando quando descritto in questo capitolo e aggiungendo concetti più avanzati come l'interazione con il DOM e i context, è possibile sviluppare qualsiasi interfaccia grafica che possa interfacciarsi con un server.

2.3.5 DBMS

Per lo sviluppo di SQL Interactive si è scelto di usare PostgreSQL [10] per la parte di database.

A differenza di Oracle, PostgreSQL è un progetto open-source, e può essere installato facilmente su diversi sistemi operativi.
PostgreSQL in particolare è un ORDBMS, acronimo di Object Relational DataBase Management System, cioè un software relazionale e ad oggetti per la gestione di basi di dati.

Come per altri database, le tabelle presenti vengono gestite da un linguaggio di alto livello chiamato SQL, che permetterà quindi di interrogare e gestire l'intera base di dati.

L'acceso al database installato è possibile mediante psql, interfaccia frontend basata su terminale, che permette l'interrogazione sulle tabelle dati e la visualizzazione del risultato, ma anche una gestione generale del database, con crezione di tabelle e schema, modifiche agli utenti e operazioni più complesse.

Capitolo 3

Progettazione

Su quanto esposto nel capitolo precedente e sui problemi raccolti da studenti dei corsi di base di dati del Politecnico di Torino si basa lo sviluppo di SQL Interactive, con la progettazione di un'architettura che supporti le seguenti funzionalità:

- Gestione di esercizi e laboratori
- Svolgimento esercizi
- Partecipazione a laboratori
- Gestione utenti docente

3.1 Architettura generale

Come introdotto nel capitolo precedente, l'applicazione sarà suddivisa in tre parti: lo sviluppo di un client utilizzabile da studenti e docenti, una parte server per la gestione delle richieste di rete e l'elaborazione di dati, e l'utilizzo di un database per la persistenza dei dati e lo svolgimento di esercizi.

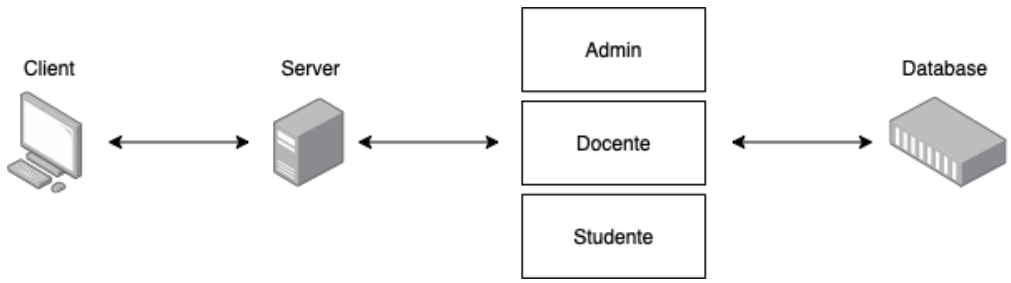

Figura 3.1: Struttura generale

La parte di sviluppo client dovrà presentare un'interfaccia grafica diversa in base all'utente che utilizza l'applicazione, in quanto ogni utente avrà permessi di accesso diversi.

La stessa divisione dovrà essere riportata durante la parte di sviluppo server, con la creazione di tre moduli distinti che si occuperanno di gestire le chiamate di rete e l'interfacciamento con il database per i diversi tipi di utente.

Quest'ultimo avrà bisogno di una particolare progettazione, in modo che possa essere utilizzato sia per il salvataggio di dati, sia per lo svolgimento di esercizi.

Nei paragrafi successivi verrà quindi analizzata nel dettaglio l'architettura dell'applicazione, con la definizione dei requisiti da rispettare, e un esempio di caso d'uso per ognuna delle funziona da implementare.

3.2 Requisiti

Il lavoro di sviluppo che verrà presentato in questa tesi integrerà alcuni degli scenari di utilizzo introdotto nel capitolo precedente, formando uno scheletro che, dopo future valutazioni effettuate su un reale corso di studio, potrà essere ampliato e aggiornato.

Partendo dall'analisi dei possibili scenari d'utilizzo, definiti nel capitolo precedente, sono stati definitivi i seguenti requisiti, funzionali e non:

3.2.1 Requisiti funzionali

- Il docente dovrà poter creare, importare e/o modificare esercizi didattici
- Lo studente potrà svolgere esercizi e partecipare a sessioni di laboratorio
- Durante un laboratorio dovrà essere presente un canale di comunicazione tra studente e docente
- Il docente potrà visualizzare statistiche relative allo svolgimento di esercizi didattici
- Un utente admin potrà gestire i docenti registrati all'applicazione

3.2.2 Requisiti non funzionali

- Dovrà essere presente un sistema di autenticazione a ruoli per differenziare l'accesso al database
- L'applicazione dovrà essere sviluppata come applicazione web, in modo da poter essere utilizzata su piattaforme diverse
- Le azioni eseguite dallo studente, come un tentativo di svolgimento di un esercizio, dovranno essere salvate su un database in modo da poter essere successivamente analizzate a aggregate
- L'interfaccia grafica durante lo svolgimento di un esercizio dovrà essere chiara e ben suddivisa, mostrando i risultati delle query eseguite dallo studente e gli eventuali errori

3.3 Casi d'uso

L'applicazione web dovrà presentare allo studente un ambiente di lavoro dedicato all'apprendimento del linguaggio SQL, proponendo esercizi di difficoltà crescente fruibili singolarmente o attraverso laboratori, interfacciandosi con un DBMS per l'esecuzione delle query realizzate dallo studente ed intercettando la risposta del DBMS per fornire un feedback adeguato.

Nella stessa applicazione il docente dovrà poter gestire i corsi di studio di cui è il titolare, modificando e caricando nuovi esercizi, creare sessioni di laboratorio e visualizzare statistiche relative all'utilizzo dell'applicazione da parte degli studente registrati ai corsi.

Ognuna di queste funzioni, definite nel paragrafo precedetene, verrà di seguito analizzata attraverso un caso d'uso significativo.

3.3.1 Admin

Sarà possibile gestire gli account destinati ai docenti attraverso un utente admin, che dopo aver effettuato il login all'applicazione potrà aggiungere un nuovo docente, modificarne la password, o disabilitarne l'accesso eliminando l'utente.

Sarà possibile anche cambiare la password dell'utente admin, senza la possibilità però di essere eliminato.

Tabella 3.1: Accesso all'applicazione

Caso d'uso	Accesso all'applicazione
Utente	Admin
Precondizioni	Nessuna
Azioni	L'utente apre la pagina web dell'applicazione L'utente inserisce username e password

Tabella 3.2: Aggiunta nuovo docente

Caso d'uso	Aggiunta nuovo docente
Utente	Admin
Precondizioni	Autenticazione effettuata
Azioni	L'admin inserisce l'username e la password del nuovo account

Tabella 3.3: Modifica/rimozione docente

Caso d'uso	Modifica/rimozione docente
Utente	Admin
Precondizioni	Autenticazione effettuata
Azioni	L'admin modifica la password di un docente esistente, o lo rimuove dal database

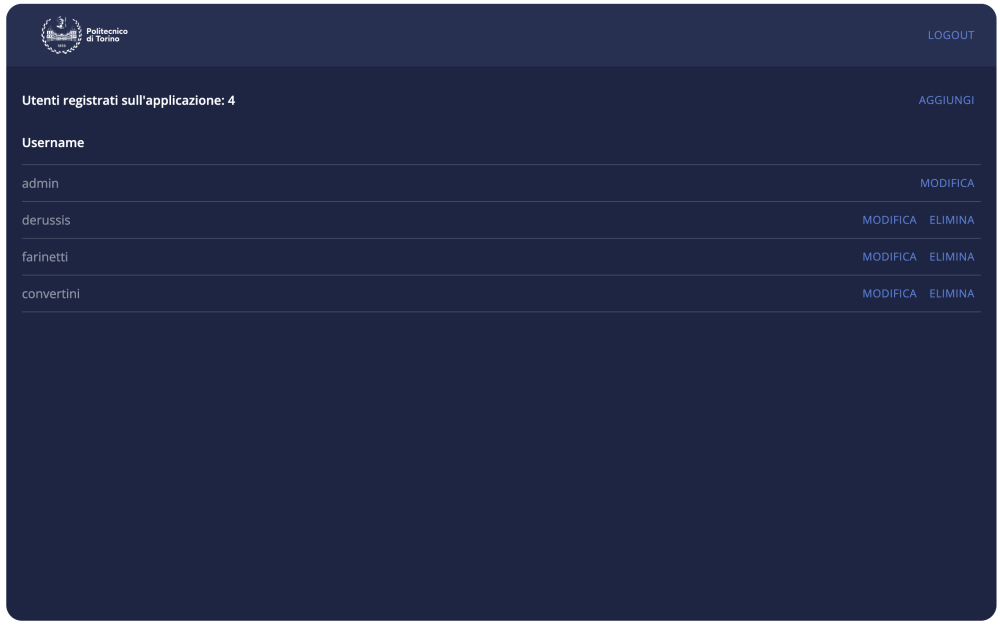

Figura 3.2: Gestione utenti - Admin

3.3.2 Docente

Dopo aver effettuato l'accesso alla piattaforma, il docente potrà creare, modificare e visualizzare corsi da lui gestiti.

Per ogni corso creato, il docente avrà a disposizione una dashboard personalizzata da cui poter analizzare le statistiche raccolte dallo svolgimento di esercizi da parte di studenti iscritti al corso, in modo da poter analizzare quanto è stato effettivamente appreso e se sono state riscontrate difficoltà.

Tabella 3.4: Accesso all'applicazione

Caso d'uso	Accesso all'applicazione
Utente	Docente
Precondizioni	Nessuna
Azioni	Il docente apre la pagina web dell'applicazione Il docente inserisce username e password

Tabella 3.5: Creazione corso di studio

Caso d'uso	Creazione corso di studio
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente inserisce il nome del nuovo corso di studio

Tabella 3.6: Visualizzazione dashboard

Caso d'uso	Visualizzazione dashboard
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente sceglie quale corso visualizzare tra quelli creati

Dalla dashboard del corso sarò possibile creare, modificare o importare nuovi esercizi. Ogni esercizio sarà composto principalmente da una traccia descrittiva del problema, una base dati di partenza e la soluzione dell'esercizio. Saranno disponibili altri parametri per la personalizzazione dell'esercizio, come l'ordine di visualizzazione e il numero massimo di tentativi.

Tabella 3.7: Creazione o modifica esercizio didattico

Caso d'uso	Creazione o modifica esercizio didattico
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente popola il database con le tabelle necessarie Il docente inserisce la soluzione e i parametri relativi allo svolgimento dell'esercizio

Tabella 3.8: Import raccolta di esercizi

Caso d'uso	Import raccolta di esercizi
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente carica un file contenente una raccolta di esercizi

Per ogni corso, verranno raccolte statistiche sullo svolgimento di esercizi caricati sulla piattaforma, come numero di tentativi effettuati, errori commessi e sul numero di accessi all'applicazione. Questi dati dovranno essere accessibili al docente sotto forma di grafici e/o tabelle, relative al corso in generale o al singolo esercizio.

Il docente avrà inoltre la possibilità di creare una sessione di laboratorio, consentendo agli studenti partecipanti di svolgere solo determinati esercizi, e potrà gestire eventuali richieste d'aiuto attraverso un canale di comunicazione integrato nell'area di lavoro.

Tabella 3.9: Creazione sessione di laboratorio

Caso d'uso	Creazione sessione di laboratorio
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente sceglie un nome per il nuovo laboratorio, gli esercizi da includere, e la durata massima

Dalla sezione studenti, accessibile dalla dashboard, sarà invece possibile aggiungere singolarmente un nuovo studente o importare un file csv con una lista di studenti automaticamente associati al corso di studio. Per ogni studente verrà generata una password casuale che permetterà l'accesso all'applicazione, con la possibilità di svolgere gli esercizi caricati e partecipare a sessioni di laboratorio.

Tabella 3.10: Aggiunta/modifica di un singolo studente

Caso d'uso	Aggiunta/modifica di un singolo studente
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente inserisce la matricola del nuovo studente

Tabella 3.11: Import di studenti multipli

Caso d'uso	Import di studenti multipli
Utente	Docente
Precondizioni	Autenticazione effettuata
Azioni	Il docente carica un file csv con la lista di utenti da esportare

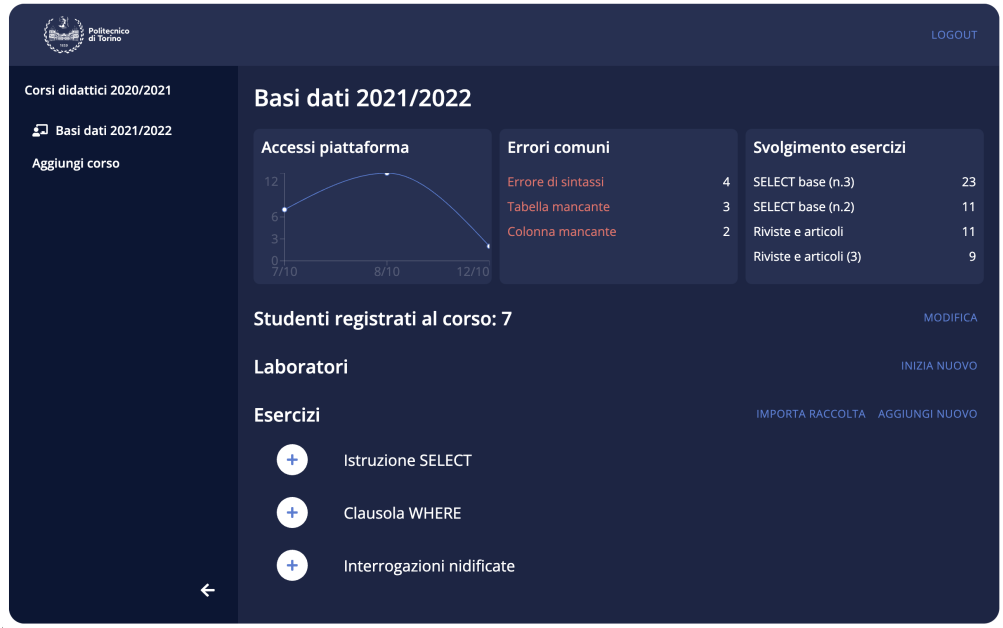

Figura 3.3: Dashboard docente

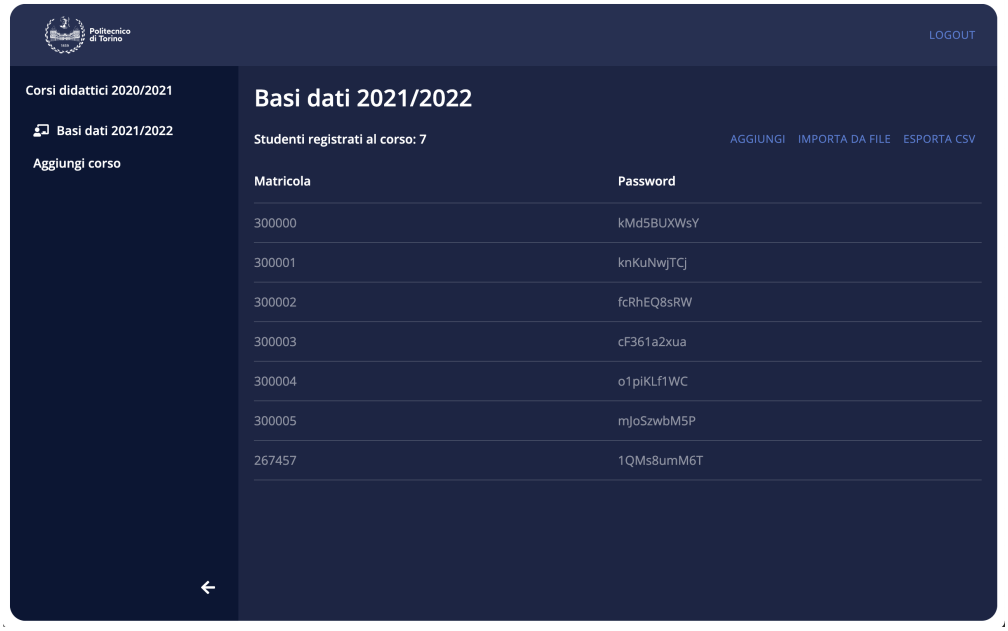

Figura 3.4: Gestione studenti

3.3.3 Studente

Dopo aver effettuato l'accesso alla piattaforma utilizzando la propria matricola e la password generata dal docente, lo studente potrà svolgere gli esercizi a disposizione abilitati. Gli esercizi saranno suddivisi per categoria, ordinati in base alle proprietà dell'esercizio, e se presente mostreranno chiaramente il limite massimo di tentativi a disposizione per la risoluzione. Nel caso in cui vengano raggiunti i tentativi massimi, lo studente non potrà più accedere all'esercizio.

Tabella 3.12: Accesso all'applicazione

Caso d'uso	Accesso all'applicazione
Utente	Studente
Precondizioni	Nessuna
Azioni	Lo studente apre la pagina web dell'applicazione Lo studente inserisce username e password

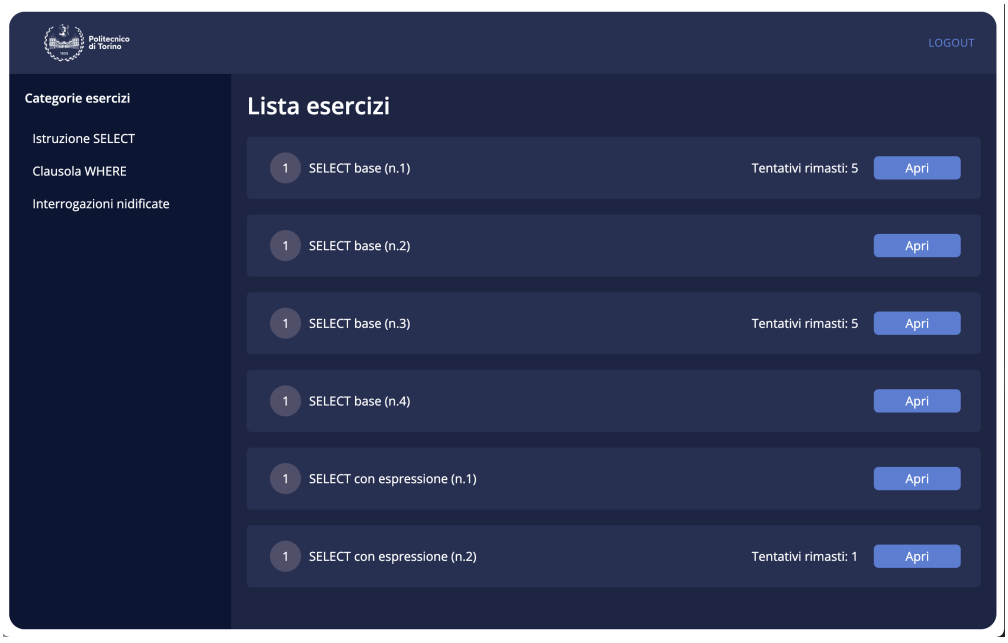

Figura 3.5: Homepage studenti

Durante lo svolgimento dell'esercizio, lo studente avrà a disposizione uno spazio da cui sarà possibile eseguire query direttamente sul DBMS e una descrizione delle tabelle e delle colonne da utilizzare.

Tabella 3.13: Svolgimento singolo esercizio

Caso d'uso	Svolgimento singolo esercizio
Utente	Studente
Precondizioni	Autenticazione effettuata Il docente ha reso disponibile un esercizio
Azioni	Lo studente visualizza gli esercizi di un particolare categoria Lo studente sceglie un esercizio da svolgere Lo studente esegue una query visualizzandone il risultato

Nel caso in cui sia abilitata dal docente, sarà possibile partecipare a una sessione di laboratorio real time, dove lo studente potrà svolgere particolari esercizi scelti precedentemente dal docente titolare. In questo caso in aggiunta a quanto descritto precedentemente, nell'area di lavoro lo studente avrà a disposizione una chat in cui è possibile comunicare con il docente in caso di necessità, e una tabella dati con il risultato atteso.

Tabella 3.14: Partecipazione a una sessione di laboratorio

Caso d'uso	Partecipazione a una sessione di laboratorioo
Utente	Studente
Precondizioni	Autenticazione effettuata Il docente ha attivato una sessione di laboratorio
Azioni	Lo studente visualizza i laboratori attivi Lo studente sceglie un esercizi da svolgere tra quelli messi a disposizione

Ogni tentativo di risoluzione dell'esercizio, sia corretto che errato, da parte dello studente verrà salvato come log nel database insieme ad un identificato univoco per identificare il tentativo, in modo da poter essere analizzati successivamente dal docente del corso. Nel caso particolare di una query errata, oltre all'identificato del tentativo verranno salvati sul database anche la query eseguita e l'errore restituito dal database.

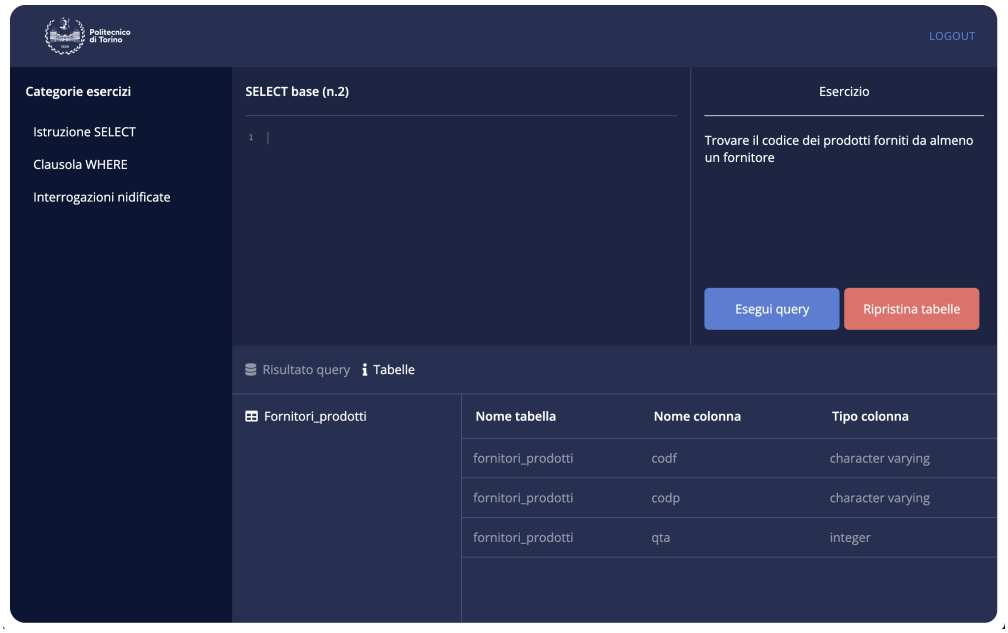

Figura 3.6: Workspace

3.4 Studio sulla struttura DBMS

Un'adeguata struttura del database risulta fondamentale per lo sviluppo dell'applicazione, in quanto ogni operazione eseguita sia dal docente che dallo studente prevede un'interazione con il database, in particolar modo durante lo svolgimento di un esercizio.

Il principale problema che verrà discusso in questo paragrafo sarà l'impostazione della base dati relativo a un esercizio, in quanto sulle stesse tabelle dovranno agire il docente (per il popolamento dei dati), e lo studente (per la risoluzione dell'esercizio), isolando allo stesso tempo i due ruoli in modo che lo studente non possa modificare la base dati di partenza dell'esercizio.

Per rispettare questo requisito è stata sviluppata la seguente struttura DBMS: In un singolo database, devono essere presenti due utenti sul database, uno con permessi di lettura/scrittura su schema public, e uno senza destinato agli studenti. Nello schema public verranno memorizzati dati generali all'applicazione (corsi, utenti, log, esercizi, laboratori), mentre per ogni docente e studente verrà creato una schema, destinato alle tabelle da utilizzare per lo svolgimento di esercizi.

Durante la creazione di un esercizio, il docente potrà agire sul proprio schema per popolare il database, e selezionare tra le proprietà dell'esercizio le tabelle necessarie alla risoluzione dello stesso.

Quando uno studente accederà alla pagina di un esercizio, verranno copiate dallo schema del docente le tabelle necessarie nello schema dello studente, in modo che la versione originale delle tabelle rimanga inaccessibile, e lo studente possa svolgere l'esercizio sempre con la possibilità di resettare la base dati allo stato originale.

Capitolo 4

Implementazione

In questo quarto capitolo verrà analizzata l'effettiva implementazione di quanto presentato precedentemente, soffermandosi principalmente sulle scelte implementative intraprese.

Per ogni funzionalità implementata, verranno presi in considerazione e discussi lo sviluppo frontend e i corrispettivi servizi backend, mostrando e commentando le parti più significative di codice sviluppato.

4.1 Integrazione client/server

Il collegamento tra la parte frontend dell'applicazione e la parte backend avviene in due modalità differenti in base all'ambiente corrente:

- Durante la fase di sviluppo, l'applicativo React sarà disponibile sulla porta 3000, mentre la parte Node.js sulla 3001, e attraverso la configurazione di un proxy le chiamate di rete inviate dal client verranno indirizzate all'indirizzo corretto:

```
1  // client/package.json
2
3  ...
4  "proxy": "http://localhost:3001",
5  ...
6
```

Listing 4.1: Proxy

E' stata scelta questa configurazione per sfruttare le funzionalità di hot-reload di React, in modo da poter visualizzare in tempo reale le modifiche al codice senza doverne compilare una versione.

- Per un ambiente di produzione invece, React e Node.js dovranno essere disponibili sullo stesso dominio. Per rispettare questa condizione è necessario che Node.js riesca a gestire entrambe le chiamate di rete e la visualizzazione dell'applicazione React. Con il seguente codice viene quindi abilitato l'accesso a Node.js ai file statici, e se una chiamata non è gestita da una delle route principali, verrà gestita dall'applicazione React.

  A seguire uno spaccato del file index.js configurato per un ambiente di produzione.

```
1    // server/index.js
2
3    const app = express();
4
5    const path = require('path');
6    const express = require('express');
7
8    ...
9
10   app.use(bodyParser.json());
11   app.use(fileUpload());
12
13   ...
14
15   // Abilita l'acceso a Node a file statici
16   app.use(express.static(`${__dirname}/build`));
17
18   // Gestisce le richieste di rete alle route principali
19   app.use("/api/v1/student", studentController);
20   app.use("/api/v1/teacher", teacherController);
21
22   ...
23
24   // Tutte le altre richieste risponderanno con l'
     applicazione React
25   app.get("*", (req, res) => {
26     res.sendFile(path.join(__dirname + "/build/index.html"))
27   });
28
29   app.use(errorHandler);
30
31   server.listen(port, () => {
32     console.log(`Server listening on the port::${port}`);
33   });
34
```

Listing 4.2: Index.js file per ambiente di produzione

Le chiamate di rete effettuate dal client sono invece trasparenti a questa duplice configurazione, e vengono effettuate tramite una libreria Javascript chiamata Axios [11].

Utilizzando questa libreria è possibile eseguire tutte le principali richieste di rete in maniera asincrona.

Tra le molte opzioni messe a disposizione per personalizzare la chiamata da effettuare, in SQL Interactive verranno utilizzati gli headers per inviare al server un token JWT per verificare la corretta autenticazione dell'utente.

Di seguito verrà mostrata la struttura di una risposta generata da Axios, e un esempio di chiamata Axios utilizzata in SQL Interactive.

```
{
  // Risposta ricevuta dal server
  data: {},

  // Codice di stato ricevuto dal server
  status: 200,
  statusText: 'OK',

  // HTTP headers con cui ha risposto il server
  headers: {},

  // Configurazione fornita a Axios per la richiesta di rete
  config: {},

  // Richiesta che ha generato la response
  request: {}
}
```

Listing 4.3: Axios response

```
// client/src/services/teacher-services.js
export const getCourseInfo = async (id) => {
  try {
    const response = await axios.get(API_URL + `courses/${id}/info`, {
      headers: authHeader(),
    });
    return response.data;
  } catch (error) {
    return handleResponse(error);
  }
};
```

Listing 4.4: Axios request

4.2 Struttura del progetto Node.js

Come introdotto precedentemente, durante lo sviluppo del progetto è stato utilizzato Express per la creazione di API di routing e per impostare middleware per rispondere alle richieste HTTP.

Nel caso di SQL Interactive, sono state creati due handler di route generali, rispettivamente una per lo studente e una per il docente, che contengono tutte le route dell'applicazione Node.js.

```
1  // server/index.js
2  ...
3  app.use("/api/v1/student", studentController);
4  app.use("", teacherController);
5
```

Listing 4.5: Router principali

Per la gestione dell'autenticazione di entrambi i ruoli, è stato utilizzato un Middleware a livello di route, in modo che per ogni chiamata possano essere definiti permessi e modalità di accesso differenti. La funzione authorize è composta da due diversi Middleware, il primo si occuperà di validare il token JWT autenticando la richiesta HTTP, il secondo controllerà che l'utente abbia i permessi per accedere alla route richiesta.

```
1  // server\helpers\authorize.js
2  import { secret } from '../db/config';
3  const jwt = require('express-jwt');
4
5  export const authorize = (roles = []) => {
6      ...
7      return [
8          jwt({ secret, algorithms: ['HS256'] }),
9          (req, res, next) => {
10             if (roles.length && !roles.includes(req.user.role)
   ) {
11                 return res.status(401).json({ message: '
   Unauthorized' });
12             }
13             next();
14         }
15     ];
16 }
17
```

Listing 4.6: Middleware autenticazione

Viene utilizzato un ulteriore Middleware per la gestione globale degli errori, in modo da centralizzarne il più possibile la logica e evitare duplicazione di codice nel progetto.

```
1     // server\helpers\error-handler.js
2     export const errorHandler = (err, req, res, next) => {
3         if (typeof (err) === 'string') {
4             return res.status(400).json({ message: err });
5         }
6         if (err.name === 'UnauthorizedError') {
7             return res.status(401).json({ message: 'Invalid Token'
      });
8         }
9         return res.status(500).json({ message: err.message });
10    }
11
```

Listing 4.7: Middleware gestione errori

4.3 Documentazione API server

Di seguito verranno elencati gli endpoint sviluppati per i due controller presentati nel paragrafo precedente.

Per quanto riguarda il docente saranno presenti:

- GET /auth/signin: gestisce le richieste di autenticazione
- GET /courses: restituisce tutti i corsi associati a un docente
- POST /courses: aggiunge un nuovo corso associato al docente
- GET /courses/:id/info: restituisce le informazioni generali di un corso
- GET /courses/:id/laboratories: restituisce i laboratori associati a un corso
- GET /courses/:id/students: restituisce gli studenti associati a un corso
- POST /courses/:id/students: aggiunge un nuovo studente o ne modifica uno esistente
- GET /courses/:id/exercises: restituisce gli esercizi associati a un corso
- POST /exercises: aggiunge un nuovo esercizio o ne modifica uno esistente
- GET /exercises/categories: restituisce le possibili categorie da associare a un esercizio
- POST /exercises/upload: gestisce l'inserimento di esercizi multipli
- GET /exercises/:id: restituisce i dati associati a un esercizio
- DELETE /exercises/:id: rimuove un esercizio
- POST /laboratories: aggiunge un nuovo laboratorio
- DELETE /laboratories/:id: rimuove un laboratorio
- GET /query/:schema/tables: restituisce tutte le tabelle presenti nello schema di un docente
- POST /query/:schema: esegue una query sullo schema selezionato

Un utente con i permessi da studente avrà invece a disposizione:

- GET /auth/signin: gestisce le richieste di autenticazione
- GET /courses/:id/exercises: restituisce tutti gli esercizi associati a un corso
- GET /courses/:id/laboratories: restituisce tutti i laboratori associati a un corso
- GET /exercises/:id: restituisce le informazioni generali di un esercizio
- GET /exercises/:id/data: popola lo schema dello studente con i dati necessari allo svolgimento, e restitusce una descrizione delle tabelle
- POST /exercises/:id/clean: elimina tutte le tabelle dallo schema di uno studente
- POST /query/:schema: esegue una query e restituisce i dati ottenuti
- POST /logs: inserisce un nuovo record in seguito a un'azione dello studente

Infine, un utente admin potrà gestire le utenze destinate ai docenti attraverso:

- GET /teachers: restituisce tutti i docenti registrati
- POST /teachers: aggiunge un nuovo docente o ne modifica uno esistente
- DELETE /teachers/:id: rimuove un docente esistente

4.4 Connessione al database PostgreSQL

Le funzionalità descritte nei capitoli precedenti necessitano di effettuare in tempi brevi numerose query sul database, in quanto ogni esercizio viene eseguito direttamente sulla base dati. Utilizzare un client diverso per ogni chiamata rallenterebbe significativamente la velocità di utilizzo dell'applicazione, in quanto la connessione al database impiegherebbe un tempo tra i 20 e i 30 millisecondi, durante i quali viene effettivamente stabilito un collegamento. Inoltre, PostgreSQL è in grado di gestire un numero limitato di client in base alla memoria a disposizione del server.

Per ovviare a questi problemi si è scelto di utilizzare il pooling [12], ovvero creare un insieme di client messi a disposizione per effettuare direttamente query sul database. In particolare, in questa applicazione sono state create due pool, una per i docenti e una per gli studenti, dato che i due ruoli hanno permessi diversi sul database.

Dopo aver creato una pool, il metodo connect() permette di acquisire il primo client disponibile. Nel caso in cui non ci fossero client accessibili, verrà automaticamente creata una coda FIFO finché uno non diventerà disponibile per l'utilizzo.

Dopo che il client è stato assegnato, potrà essere utilizzato per eseguire query sul database attraverso il metodo query().

E' importante sottolineare come dopo aver utilizzato un client sia necessario che venga rilasciato con il metodo release(), in modo da non saturare la pool con client assegnati ma inutilizzati, in attesa di essere automaticamente rilasciati dopo un tempo di timeout stabili alla creazione della pool.

```
1   // server/db/db.js
2   const { Pool } = require("pg");
3
4   export const studentPool = new Pool({
5     host: DB_HOST,
6     password: DB_STUDENT_PASSWORD,
7     port: DB_PORT,
8     user: DB_STUDENT_USER,
9     database: DB_DATABASE,
10    max: 20,
11    idleTimeoutMillis: 10000,
12    connectionTimeoutMillis: 10000,
13  });
14
```

Listing 4.8: PostgreSQL pool

4.5 Librerie esterne

Durante lo sviluppo di SQL Interactive sono state utilizzate alcuni librerie Javascript esterne, tra le più importanti è possibile elencare:

- jsonwebtoken: crea e utilizza token per gestire le sessioni degli utenti
- pg: gestisce ed espone una o più connessioni a un database
- socket.io: permette la connessione real-time tra server e client
- dayjs: libreria che permette di manipolare date e orari
- bcrypt: libreria per effettuare hash di password
- uuid: generazione di identificativi unici conformi allo standard UUID
- express: gestione routing richieste web
- framer-motion: libreria grafica per l'animazione di componenti
- react-ace: realizzazione dell'editor testuale per le query
- react-csv: creazione di un file csv da esportare
- fortawesome: repository di icone grafiche
- lodash: libreria di utlity Javascript
- recharts: generazione di grafici interattivi

La lista completa delle librerie utilizzate è consultabile nel file package.json, utilizzato da npm come repository per configurazioni generali legate al progetto, come script node eseguibili, nome e descrizione del progetto, repository git, licenza, e appunto la lista delle dipendenze necessarie.

4.6 Sviluppo delle funzionalità - Docente

4.6.1 Login

Il docente potrà effettuare il login all'applicazione attraverso username e password, create precedentemente da un utente admin.

Quando viene rilevato un tentativo di login da parte di un docente, dopo averne verificato l'effettiva presenza sul database e la correttezza della password inserita, il sistema controllerà se sul database è già presente uno schema dedicato al docente, che verrà altrimenti creato, e genererà un token JWT che verrà utilizzato per le successive richieste di rete.

4.6.2 Gestione corsi

Ogni docente avrà la possibilità di creare più corsi associati alla propria utenza, ognuno con studenti ed esercizi indipendenti gli uni dagli altri.

Al momento della creazione di un nuovo corso, l'unico dato richiesto al docente sarà il nome del corso, e il sistema si occuperà di creare il nuovo corso nella rispettiva tabella del database e associarlo all'utenza del docente.

4.6.3 Gestione lista studenti

Per ogni corso, il docente potrà gestire gli studenti abilitati ad accedere agli esercizi caricati, aggiungendo individualmente uno studente partendo dalla matricola, o importando un file CSV precedentemente esportato dal portale del Politecnico di Torino. Il parsing del file CSV viene effettuavo direttamente lato client con l'utilizzo della libreria "papaparse", in modo da non aumentare il carico di lavoro sul server.

In entrambi i casi, il sistema genererà per ogni studente una passsword che, insieme alla matricola, permetterà allo studente di accedere all'applicazione web. Sarà inoltre possibile esportare le utenze generate in un file CSV, in modo da essere distribuite agli studenti iscritti.

4.6.4 Gestione esercizi

Dalla pagina principale di ogni corso il docente potrà creare un nuovo esercizio o importarne una raccolta.

Per la creazione di un esercizio da zero, il docente dovrà inserire le seguenti proprietà:

- Titolo e testo dell'esercizio
- Soluzione dell'esercizio

- Difficoltà (livello di difficoltà)
- Categoria
- Ordine dell'esercizio nella categoria
- Numero massimo di tentativi permessi
- Tabelle necessarie alla risoluzione

In aggiunta a quanto elencato, il docente dovrà, nello spazio dedicato alle query, creare e popolare le tabelle con i dati necessari alla risoluzione dell'esercizio. Queste tabelle verranno create direttamente nello schema dedicato al docente, e quando necessario copiate nello schema dello studente.

```
1    // server/controllers/teacher.js
2    ...
3    const schema = req.params.schema;
4    const query = req.body.query;
5    ...
6    const response = await client.query(query);
7    ...
```

Listing 4.9: Esecuzione query su DBMS

4.6.5 Gestione laboratori

Dopo aver creato o importato gli esercizi, il docente del corso potrà creare una sessione di laboratorio interattiva. Alla creazione sarà necessario inserire il nome del laboratorio, selezionare gli esercizi da includere, e stabilirne la durata.

Accedendo al laboratorio, il docente avrà a disposizione un'area su cui eseguire query, una su cui visualizzarne il risultato, e una chat integrata dedicata alla comunicazione con gli studenti che partecipano al laboratorio.

Per sviluppare la chat è stata utilizzata socket.io, libreria NPM che permette la comunicazione real-time.

```
1    // server/controllers/teacher.js
2    ...
3    socket = io(ENDPOINT, { transports: ["websocket"], upgrade:
     false });
4    ...
```

Listing 4.10: Inizializzazione socket

La conversazione tra studente e docente è resa possibile grazie alla creazione di "room", canali di comunicazione dove il socket può entrare e uscire. La gestione di questi canali è rimandata al server, che riceve le richieste dei client e le indirizza nelle rispettive rooms.

In particolare per SQL Interactive, verranno create 3 tipologie di rooms diverse, una per il corso, una per il docente, e una per ogni studente che accede all'applicazione.

La room dedicata al corso verrà utilizzata per avvertire lo studente che un nuovo laboratorio è iniziato, mentre le due rimanenti saranno utilizzate per lo scambio dei messaggi.

Per inviare un nuovo messaggio, o più in generale un oggetto, è disponibile il metodo emit(), che permette di inviare un oggetto in una o più rooms:

```
1  // server/controllers/teacher.js
2  ...
3  io.to(course_id).emit("new_laboratory", "new lab created");
4  ...
```

Listing 4.11: Inizializzazione socket

In aggiunta alla classica modalità di comunicazione, quando uno studente richiederà un intervento da parte del docente, nella parte inferiore della schermata sarà visibile la soluzione dell'esercizio aperto dallo studente e la storia relativa al tentativo di risoluzione corrente, con gli errori commessi e le query eseguite sul database.

4.6.6 Visualizzazione statistiche

Ogni azione eseguita dallo studente durante l'utilizzo dell'applicazione verrà registrata attraverso l'inserimento di un record in tabelle dedicate del database. Questo permetterà un analisi da parte del docente sull'utilizzo dell'applicazione e sullo svolgimento degli esercizi.

Sono state create due tabelle destinate alla raccolta di questi dati:

- audit_log: utilizzata per la raccolta dei dati relativi ai login effettuati
- audit_exercises: utilizzata per la raccolta dei dati relativi allo svolgimento degli esercizi da parte degli studenti, come query effettuate e errori restituiti dal database

Entrambe le tabelle vengono utilizzate sia per la creazione dei tre principali indicatori visibili sulla dashboard del corso da parte del docente, sia per una più dettagliata analisi di ogni singolo esercizio, riportando, su una modale dedicata, i tentativi effettuati, le risoluzioni corrette e il tasso di abbandono.

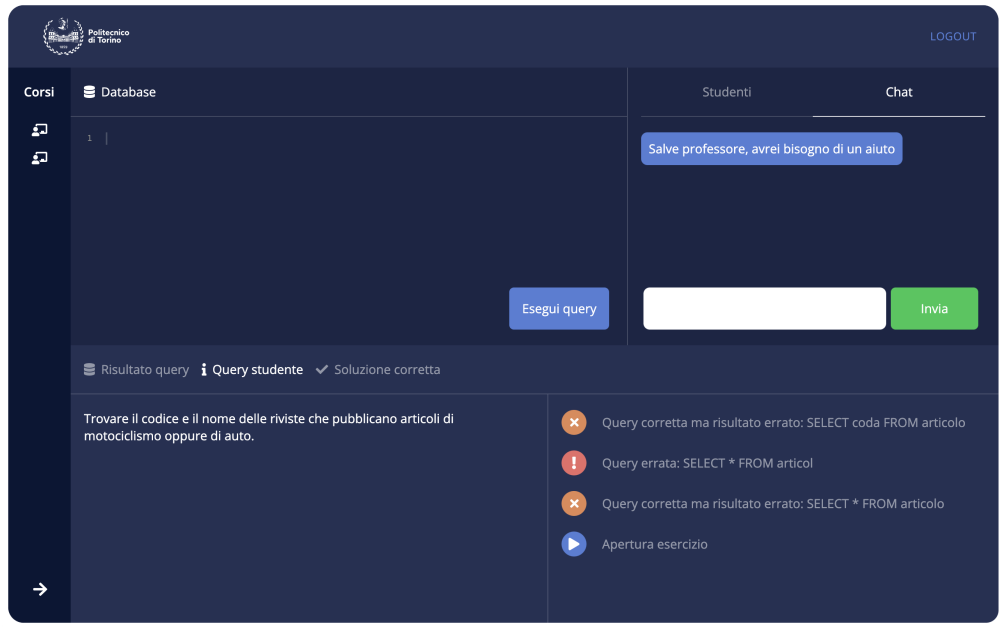

Figura 4.1: Laboratorio - Docente

4.7 Sviluppo delle funzionalità - Studente

4.7.1 Login

Dalla schermata iniziale di SQL Interactive lo studente, dopo aver ricevuto dal docente le credenziali necessarie, potrà effettuare il login all'area riservata al proprio corso di studio.

L'applicazione provvederà a verificare la presenza sul database di uno schema dedicato allo studente, e in caso negativo verrà automaticamente creato, garantendo permessi di scrittura e lettura all'utente studente del database. Grazie alla creazione di uno schema dedicato, lo studente potrà eseguire qualsiasi query sul database senza poter modificare i dati delle tabelle originali dell'esercizio (salvate nello schema del professore), e in ogni momento resettare il database allo stato iniziale.

4.7.2 Svolgimento esercizi

Dopo aver effettuato il login, lo studente avrà a disposizione la libreria di esercizi pubblicata dal docente del corso.

Gli esercizi saranno raggruppati in categorie, elencate nella colonna sinistra, e per ognuno saranno chiaramente visibili alcune informazioni generali:

- Titolo
- Testo dell'esercizio
- Tentativi rimasti

Nel caso in cui lo studente finisca i tentativi a disposizione, l'esercizio rimarrà visibile ma non potrà più essere svolto.

Accedendo alla pagina di un esercizio, l'area di lavoro sarà così composta: un editor testuale su cui sarà possibile inserire le query da eseguire, il testo dell'esercizio da risolvere e una descrizione delle tabelle e delle colonne da utilizzare.

Eseguendo una query, potranno verificarsi 3 diverse situazioni:

- Query sintatticamente errata: nel caso in cui lo studente esegua una query sintatticamente errata, la stessa verrà eseguita sul database, ne verrà intercettato l'errore e mostrato allo studente
- Query corretta ma esercizio non risolto: la query verrà eseguita sul database, e verranno restituiti al client i dati necessari per costruire graficamente una tabella con il risultato
- Query corretta ed esercizio risolto: verrà costruita la stessa tabella del caso precedente, e lo studente verrà informato che l'esercizio è stato risolto correttamente

Per verificare la corretta risoluzione dell'esercizio, dato che query diverse possono portare alla stessa soluzione, l'applicazione confronterà il risultato della query eseguita dallo studente e quello della soluzione inserita dal docente eseguite nei rispettivi schemi.

L'implementazione di questo confronto avverrà in due passaggi: i due oggetti restituiti dal database dopo l'esecuzione delle query (rispettivamente dello studente e del docente) verranno prima trasformate in due matrici, eliminando i riferimenti al nome delle colonne (supportando in questo modo l'utilizzo di alias nelle query), e successivamente confrontate attraverso il metodo isEqual(), messo a disposizione dalla libreria Lodash.

```
1   // client/src/components/exerciseWorkspace.js
2     const creteArrayBodyFromResponse = (response) => {
3       const array = [];
4       for (let index = 0; index < response.rows.length; index++)
    {
5         const element = response.rows[index];
6         const row = [];
7         for (let key of Object.keys(element).sort()) {
8           row.push(element[key]);
9         }
10        array.push(row);
11      }
12      return array;
13    };
14
15    const testQueryWithSolution = async (queryData) => {
16      const queryBody = creteArrayBodyFromResponse(queryData);
17      const resultBody = creteArrayBodyFromResponse(resultData);
18      if (_.isEqual(queryBody, resultBody)) {
19        ...
20        setExerciseResolved(true);
21      } else {
22        ...
23      }
24    };
```

Listing 4.12: Risoluzione esercizio

4.7.3 Svolgimento laboratori

Dalla schermata iniziale, nel momento in cui il docente del corso lo rende disponibile, lo studente potrà accedere a una sessione di laboratorio.

L'area di lavoro sarà simile a quella del normale svolgimento di un esercizio, con la differenza che sarà visualizzabile il risultato atteso, e un chat con il docente integrata per richieste di assistenza durante lo svolgimento del laboratorio.

4.7.4 Raccolta statistiche

Sia durante lo svolgimento di un esercizio che durante un laboratorio, i tentativi di risoluzione da parte degli studenti verranno utilizzati per raccogliere log e statistiche con l'obiettivo di migliorare l'apprendimento di SQL del corso di studio e l'utilizzo dell'applicazione.
In particolare, per quanto riguarda lo svolgimento degli esercizi, verranno registrati quattro diverse tipologie di dati:

- Apertura dell'esercizio
- Query sintatticamente errata
- Query corretta ma esercizio non risolto
- Query corretta e esercizio risolto

In aggiunta al tipo di log, nello stesso record verranno salvati ulteriori dati, riguardanti l'esercizio corrente:

- Query eseguita dallo studente
- Identificativo univoco del tentativo
- Identificativo dell'esercizio
- Username dello studente
- ID Laboratorio (opzionale)
- Codice errore ricevuto dal database (opzionale)

In una tabella dedicata verranno anche registrati tutti gli accessi all'applicazioni effettuati dallo studente.

4.8 Sviluppo delle funzionalità - Admin

Se a effettuare il login all'applicazione sarà un utente admin, saranno visualizzati tutti gli utenti docente presenti nella tabella users.

Per ogni utente sarà possibile modificare la password, che verrà crittografata con la libreria bcrypt prima di essere salvata sul database, o eliminare direttamente l'utente, che non potrà quindi più accedere all'applicazione.

Figura 4.2: Gestione utenti - Admin

Capitolo 5

Valutazioni

5.1 Valutazioni generali

Per ottenere valutazioni pratiche sull'utilizzo di SQL Interactive, sviluppata in questo progetto di laurea, sarà necessario attendere il suo effettivo utilizzo durante un corso di studio.

In questo paragrafo verranno quindi proposte delle valutazioni prettamente teoriche attraverso il confronto tra il risultato ottenuto e i possibili scenari d'utilizzo introdotti nel secondo capitolo e approfonditi successivamente. Per fare ciò ci si è interrogati un una serie di domande, legate allo svolgimento della attività di laboratorio e di esercizio durante il corso di basi di dati, che sono alla base dell'intero lavoro.

- *Lo svolgimento di un esercizio didattico da parte di uno studente risulta più agevole?*

  Utilizzando SQL Interactive lo studente non avrà più la necessità di replicare un ambiente idoneo allo sviluppo di un esercizio SQL sul proprio dispositivo, che comprendeva l'installazione di un database, un'interfaccia grafica e il popolamento delle tabelle necessarie.

  Selezionando un esercizio tra quelli disponibili lo studente potrà eseguire sul momento query direttamente su un database già correttamente configurato, diminuendo il carico di lavoro e la possibilità di errori.

- *Le difficoltà di partecipazione ai laboratori da remoto sono state risolte?*

  La modalità di partecipazione a un laboratorio didattico offerto attraverso SQL Interactive non presenterà differenze tra una fruizione in presenza o in remoto.

In entrambe le modalità lo studente potrà svolgere l'esercizio assegnato e contattare il docente attraverso un canale di comunicazione integrato direttamente nell'area di lavoro per richieste di aiuto.

- *Lo studente riscontrerà meno difficoltà durante la fase di preparazione dell'esame?*

  Uno dei principali problemi riscontrati dagli studenti durante la preparazione dell'esame è risultato essere la difficile reperibilità di esercizi SQL da svolgere autonomamente

  I lucidi esercitativi forniti dai docenti, o reperiti da internet, normalmente non comprendono tabelle di dati utilizzabili, rendendo difficile l'effettiva esecuzione di query e portando lo studente a studiare direttamente la soluzione dell'esercizio senza un tentativo di risoluzione.

  Con l'utilizzo dell'applicazione proposta in questa tesi il docente potrà preparare una libreria di esercizi e temi d'esame sempre disponibili per gli studenti, che potranno essere utilizzati durante il corso di studio come preparazione all'esame.

- *Il docente avrà una visione più chiara sull'andamento del corso di studio e sulla comprensione dei temi trattati*

  Lo svolgimento di esercizi didattici su SQL Interactive, oltre a facilitare l'apprendimento di SQL agli studenti del corso di studio, permetterà al docente di raccogliere dati utili a una valutazione dell'efficacia del corso di studio.

  Verranno automaticamente raccolti dall'applicazione tutti gli errori commessi dagli studente, e gli accessi all'applicazione.

  In particolare per ogni esercizio potrà essere valutata la difficoltà percepita dagli studenti attraverso percentuali di successo e abbandono; questo può rivelarsi una dato utile per tarare la difficoltà dell'esame finale e comprendere quanto appreso dagli studenti.

5.2 Struttura e risultati stress test

Dato l'utilizzo prettamente didattico previsto dall'applicazione, che verrà usata contemporaneamente da più studenti, sono stati effettuati una serie di stress test sulla parte server/database per valutare i tempi di elaborazione e risposta. Per effettuare questi test è stato utilizzato JMeter, strumento specializzato nel testing di servizi web e database remoti.

Nel caso studio analizzato è stato preso in considerazione un gruppo di quindici studenti, con periodo di ramp-up tra uno studente e il successivo di due secondi.

Per simulare un'effettiva architettura remota è stato effettuato un rilascio dell'applicazione su un server Heroku, in modo da tenere in considerazione anche un ritardo di connessione medio.

Il test effettuato prende in considerazione le prime quattro chiamate di rete effettuate dallo studente, dove è logico aspettarti un picco di chiamate concorrenti:

- Login
- Lista esercizi
- Dettagli esercizio
- Dati esercizio

E' stato ritenuto significante questo gruppo di chiamate di rete in quanto può verificarsi che più utenti accedano contemporaneamente all'applicazione e alla pagina di un esercizio.

In particolare, sia la prima chiamata di autenticazione, sia quella per recuperare i dati necessari alla risoluzione di un esercizio, includono più accessi al database e operazioni complesse, che comporteranno, di conseguenza, un tempo di risposta non immediato.

Al momento del login infatti, oltre ad andare a verificare l'effettiva presenza di un account per lo studente e il recupero della password, nel caso in cui lo studente non abbia mai fatto accesso all'applicazione verrà creato un schema dedicato. Questo schema verrà successivamente popolato con le tabelle necessarie nel momento in cui uno studente accederà alla pagina di un esercizio.

Il risultato ottenuto dall'esecuzione in sequenza di queste 4 chiamate, contemporaneamente da un gruppo di quindici studenti è il seguente:

Tabella 5.1: Risultati stress test multiplo

Label	Chiamate	Media	Max	Min
Login	15	515ms	855ms	377ms
Lista esercizi	15	278ms	581ms	127ms
Dettagli esercizio	15	362ms	509ms	326ms
Dati esercizio	15	767ms	1193ms	540ms

I risultati rispecchiano quanto introdotto precedentemente, la chiamata di autenticazione e i dati dell'esercizio impiegano un tempo leggermente superiore alle altre chiamate per essere completate, ma con tempi di risposta più che accettabili, considerando il breve tempo di ramp-up utilizzato.

Ripetendo lo stesso test ma su un singolo studente, si ottengono le seguenti tempistiche:

Tabella 5.2: Risultati stress test singolo

Label	Chiamate	Media	Max	Min
Login	1	377ms	377ms	377ms
Lista esercizi	1	214ms	214ms	214ms
Dettagli esercizio	1	249ms	249ms	249ms
Dati esercizio	1	616ms	616ms	616ms

Dal confronto delle due tabelle, si evince come la contemporaneità tenda ad aumentare il tempo di risposta massimo del server, ma la struttura del database, dettagliata nel capitolo 4 di questa tesi, permette di limare questa differenza in modo da rendere il tempo di attesa accettabile anche in caso di accessi multipli.

Capitolo 6

Conclusioni

La soluzione proposta in questa tesi, composta da un'analisi del problema, progettazione e sviluppo di un'applicazione web, nasce dalla necessità di risolvere alcune criticità presenti nel corso di basi di dati.

Veniva infatti richiesto allo studente, sia per esercitazioni personali, sia per la partecipazione a laboratori da remoto, di ricreare un ambiente di sviluppo idoneo, con l'installazione di un database, il popolamento delle tabelle necessarie alla risoluzione dell'esercizio, e un'interfaccia grafica per l'esecuzione delle query.

Per risolvere queste problematiche è stata sviluppata SQL Interactive, un'applicazione web, divisa in una parte client e una server, che interfacciandosi con un database permetterà allo studente di svolgere esercizi didattici e partecipare a sessioni di laboratorio senza la necessita di installare sul proprio computer software aggiuntivi.

In questo modo, lo studente avrà a disposizione una libreria di esercizi utili a facilitare l'apprendimento di SQL, e verrà uniformata l'esperienza di laboratorio remota rispetto a quella in presenza, aprendo un canale di comunicazione tra docente e studente direttamente nell'area di lavoro.

Contemporaneamente, al docente verrà data la possibilità di analizzare quanto appreso dagli studenti del corso, attraverso la visualizzazione di statistiche dettagliate per ogni esercizio reso disponibile.

6.1 Sviluppi futuri

L'applicazione sviluppata può essere considerata un buon punto di partenza per numerosi sviluppi futuri, con l'obiettivo di migliorare l'utilizzo dell'applicazione sia per il docente che per lo studente.

- Sviluppo di una dashboard dedicata allo studente: utilizzando i dati raccolti automaticamente dall'applicazione potrebbe essere utile mostrare allo studente gli errori più commessi, la percentuale di esercizi risolti e un'indicazione sul livello di preparazione.
- Creazione di un percorso didattico standard: nell'area di lavoro durante lo svolgimento di un esercizio didattico potrebbe essere integrata una parte teorica di supporto.
- Implementazione di un agente conversazionale: lo sviluppo di un agente conversazionale permetterebbe allo studente di ricevere consigli specifici sullo svolgimento di un esercizio anche quando non è disponibile un canale di comunicazione con un docente, e interpretare gli errori ricevuti dal database.
- Svolgimento di elaborati obbligatori: spesso durante i corsi di studio vengono assegnati allo studente elaborati da svolgere che saranno parte di una valutazione, SQL Interactive potrebbe essere configurata per permettere lo sviluppo di questi elaborati allo studente, e la possibilità di verificare quanto prodotto da ogni studente al docente del corso

Bibliografia

[1] *Javascript*. URL: https://en.wikipedia.org/wiki/JavaScript (cit. a p. 5).

[2] *Introduction to Node.js*. URL: https://nodejs.dev/learn (cit. a p. 6).

[3] *Express Routing*. URL: https://expressjs.com/en/guide/routing.html (cit. a p. 6).

[4] *Express Middleware*. URL: https://expressjs.com/en/guide/writing-middleware.html (cit. a p. 6).

[5] *React Virtual DOM*. URL: https://it.reactjs.org/docs/faq-internals.html (cit. a p. 8).

[6] *Introduzione a JSX*. URL: https://it.reactjs.org/docs/introducing-jsx.html (cit. a p. 8).

[7] *Componenti e Props*. URL: https://it.reactjs.org/docs/components-and-props.html (cit. a p. 8).

[8] *Introduzione agli Hooks*. URL: https://it.reactjs.org/docs/hooks-intro.html (cit. a p. 9).

[9] *React useEffect Hook*. URL: https://it.reactjs.org/docs/hooks-effect.html (cit. a p. 10).

[10] *About PostgreSQL*. URL: https://www.postgresql.org/about/ (cit. a p. 11).

[11] *Axios*. URL: https://github.com/axios/axios (cit. a p. 27).

[12] *PostgreSQL Pooling*. URL: https://node-postgres.com/features/pooling (cit. a p. 32).