Università degli studi di Messina

DIPARTIMENTO DI SCIENZE MATEMATICHE E INFORMATICHE, SCIENZE FISICHE E SCIENZE DELLA TERRA

Corso di Laurea triennale in Informatica L-31

Curriculum: Scienze Informatiche

SVILUPPO APPLICAZIONE PER LA GESTIONE DI CONVEGNI: APPROCCIO FULL STACK

Relatore:

Prof. Andrea Nucita

Candidato:

Emanuele Russo

Anno Accademico

2023/2024

Indice

Elenco delle figure

Listings

Capitolo 1

Introduzione

1.1 Contesto e importanza della gestione dei convegni

Nel mondo accademico e professionale contemporaneo, i convegni rappresentano una colonna portante per l'accrescimento delle proprie conoscenze e lo sviluppo delle carriere professionali, in quanto sono sempre più comuni i corsi di aggiornamento dei lavoratori. La gestione di questi eventi è diventata una necessità sempre più comune, vista anche la crescente complessità e frequenza degli incontri, sia in presenza che a distanza.

Grazie alle tecnologie attuali, si sono sviluppate diverse soluzioni per ottimizzare l'organizzazione e la partecipazione ai convegni. Come evidenziato nell'articolo "Event-Based Mobile Social Networks: Services, Technologies, and Applications" [1], "Le applicazioni basate su eventi online hanno fornito piattaforme adatte per gli utenti per creare, pubblicare e gestire attività sociali. Oltre a supportare le funzionalità di social networking online rappresentative (ad esempio, pubblicare e condividere commenti, video e foto), queste applicazioni basate su eventi promuovono anche le comunicazioni sociali offline faccia a faccia."

L'evoluzione di queste applicazioni ha portato a un ulteriore miglioramento dell'esperienza dei partecipanti. Infatti, come sottolineano gli stessi autori, "Le applicazioni event-based moderne possono trasformare gli eventi in ambienti più dinamici e coinvolgenti dove i partecipanti smettono di essere semplicemente spettatori. Diventano creatori di contenuti e interagiscono continuamente e forniscono feedback attraverso sondaggi e social media" [1].

L'importanza di queste piattaforme è diventata ancora più evidente in periodi di crisi, come durante la pandemia di COVID-19. Gli autori Chalkia e Papageorgiou [2], nell'articolo "The Management of Conferences and Business Events in Periods of Crisis. The New Digital Paradigm", hanno evidenziato come la pandemia abbia portato a una transizione del settore MICE (Meetings, Incentives, Conferences, and Exhibitions) verso un nuovo ambiente digitale. Le tecnologie digitali sono diventate fondamentali non solo per l'organizzazione di incontri ed eventi, ma anche per il loro intero svolgimento. Questo cambiamento ha permesso al settore di adattarsi rapidamente alle restrizioni imposte dalla pandemia, mantenendo viva l'interazione e lo scambio di conoscenze anche in un periodo di limitata mobilità fisica.

In questo contesto, un'applicazione dedicata alla gestione dei convegni si rivela uno strumento capace non solo di colmare il divario tra le esigenze degli organizzatori e quelle dei partecipanti, ma anche di arricchire significativamente l'esperienza dell'evento, facilitando sia le interazioni online che quelle in presenza e promuovendo un coinvolgimento attivo di tutti i partecipanti. Inoltre, come dimostrato durante la pandemia, queste applicazioni offrono la flessibilità necessaria per adattarsi rapidamente a situazioni di crisi, garantendo la continuità delle attività di networking e formazione anche in circostanze atipiche come quelle vissute durante la pandemia.

1.1.1 Integrazione con i sistemi esistenti

Un'applicazione per la gestione dei convegni può essere utilizzata in concomitanza con i siti web istituzionali degli atenei o delle organizzazioni professionali. Questa integrazione permette di centralizzare le informazioni e semplificare l'accesso per gli utenti.
Allo stesso tempo, l'applicazione è progettata per funzionare in modo autonomo, offrendo una piattaforma completa e indipendente per la gestione degli eventi. Questa flessibilità la rende adatta a molteplici contesti, dalle più piccole riunioni accademiche alle grandi conferenze internazionali.

1.1.2 Semplificazione e ottimizzazione dei processi

L'applicazione si pone come guida intuitiva per l'utente, semplificando i processi di partecipazione ed organizzazione dei convegni. L'obiettivo è quello di automatizzare attività come la registrazione, la gestione del programma e le comunicazioni, riducendo il carico di lavoro manuale e allo stesso tempo le possibilità di errore.

1.1.3 Gestione delle comunicazioni

Uno dei vantaggi dell'applicazione è, come detto, la sua capacità di centralizzare e ottimizzare le comunicazioni, eliminando alcune pratiche manuali spesso utilizzate in tali contesti, come l'invio di email di promemoria o l'inserimento degli impegni nel proprio calendario. L'applicazione offre un sistema di notifiche capace di avvertire l'utente circa la conferma di partecipazione ai convegni a cui desidera aderire.

Un'applicazione dedicata alla gestione dei convegni rappresenta un passo in avanti significativo nell'ottimizzazione di questi eventi cruciali per il mondo accademico e professionale. L'app si propone, quindi, di trasformare l'esperienza dei convegni, rendendola più efficiente, coinvolgente e produttiva per tutti i partecipanti e allo stesso tempo più facile da gestire per gli organizzatori.

1.2 Tecnologie principali utilizzate

Nello sviluppo di questa applicazione per la gestione dei convegni, sono state impiegate diverse tecnologie, ciascuna scelta per le sue caratteristiche specifiche e per la sua capacità di integrarsi efficacemente in un'architettura moderna e scalabile.

Per il lato frontend dell'applicazione, è stato scelto **Flutter** [3], un framework open-source sviluppato da Google. Flutter, basato sul linguaggio di programmazione Dart, permette la creazione di interfacce utente native e performanti sia per dispositivi mobili (iOS e Android) che per il web, garantendo un'esperienza utente fluida e coerente su tutte le piattaforme.

Il lato backend dell'applicazione è stato, invece, implementato utilizzando **Node.js**, un ambiente di runtime JavaScript lato server. Node.js è noto per la sua efficienza e scalabilità, caratteristiche fondamentali per gestire le richieste in tempo reale e il carico di lavoro variabile tipico delle applicazioni.

Per quanto riguarda la gestione dei dati, è stato scelto **MySQL**, un sistema di gestione di database relazionali ampiamente utilizzato ed affidabile. MySQL offre ottime capacità di memorizzazione e recupero dei dati, essenziali per gestire le informazioni relative ai convegni, ai partecipanti e alle interazioni all'interno dell'applicazione.

Infine, l'applicazione è stata strutturata seguendo i principi dell'architettura **API REST** (Representational State Transfer), che facilita la comunicazione efficiente tra il frontend e il backend. Tale approccio permette una maggiore flessibilità e manutenibilità del sistema.

Come ben spiegato nel sito di RedHat [4], "Un'API REST, nota anche come API RESTful, è un'interfaccia di programmazione delle applicazioni (API o API web) conforme ai vincoli dello stile architetturale REST, che consente l'interazione con servizi web RESTful. Il termine REST, coniato dall'informatico Roy Fielding, è l'acronimo di REpresentational State Transfer."

L'integrazione di queste tecnologie ha permesso di creare un'applicazione scalabile e user-friendly, capace di soddisfare le esigenze complesse della gestione dei convegni. Nei capitoli successivi, verrà spiegato in dettaglio come queste tecnologie sono state implementate e integrate per realizzare le funzionalità dell'applicazione.

Capitolo 2

Stato dell'arte

2.1 Panoramica delle tecnologie attuali

Tra le tecnologie attuali che mirano alla gestione degli eventi, è possibile osservare una varietà di soluzioni, le quali tentano di affrontare le esigenze legate all'organizzazione e alla partecipazione ai convegni in modo differente l'una dall'altra. Tuttavia, da un'approfondita ricerca condotta sul web, è possibile riscontrare una maggiore attenzione verso eventi fruibili da remoto attraverso mezzi messi ha disposizione dalle applicazioni stesse ponendo in secondo piano l'organizzazione di convegni pensati per la fruizione in loco.

Tra le applicazioni che si occupano della creazione e gestione degli eventi vi sono alcune criticità, tra cui la mancanza di specificità per alcune tipologie di convegni. La maggior parte delle app non è ottimizzata per le esigenze specifiche dei convegni accademici o professionali che vengono organizzate all'interno delle università o del luogo di lavoro.

2.1.1 Approccio "su misura" e le sue limitazioni

Un fenomeno emerso dalla ricerca è la presenza di associazioni e aziende che offrono lo sviluppo di applicazioni personalizzate per singoli convegni. Tale approccio, seppur capace di soddisfare le esigenze specifiche di un determinato evento, presenta diverse limitazioni:

- **Costi elevati**: La creazione di un'applicazione dedicata per ogni convegno comporta importanti investimenti finanziari.
- **Assenza di scalabilità**: Queste soluzioni sono pensate per un utilizzo limitato, risultando quindi prive di scalabilità.
- **Frammentazione dell'esperienza utente**: Partecipanti abituali a diversi convegni si trovano a dover imparare e utilizzare interfacce diverse per ogni evento.
- **Inefficienza nella gestione multi-evento**: Organizzazioni che gestiscono più convegni si trovano a dover amministrare molteplici applicazioni separate.

2.2 Analisi delle piattaforme leader nel settore

Per comprendere meglio lo stato attuale delle tecnologie del settore, bisogna esaminare alcune delle piattaforme principali che offrono funzionalità simili a quelle proposte dall'applicazione sviluppata.

La prima piattaforma da esaminare è **Fourwaves**[5], la quale si distingue per la gestione di eventi accademici, supportando formati virtuali, ibridi e in presenza. Le sue caratteristiche principali includono: la gestione avanzata di sessioni virtuali con strumenti interattivi e conversazioni video in tempo reale, la possibilità di accesso alla piattaforma tramite dispositivi mobili, possibilità di organizzare sessioni e di interazione tra utenti. Fourwaves rappresenta un esempio di come l'interattività e la flessibilità siano caratteristiche importanti nelle moderne piattaforme di gestione degli eventi.

La seconda è **Ex Ordo** [6], la quale si focalizza specificamente sulla gestione delle conferenze accademiche. Le sue caratteristiche distintive riguardano principalmente strumenti per la creazione del programma dell'evento e l'assegnazione delle sessioni. Inoltre è specializzata per tutti gli eventi che prevedono una quota di iscrizione, in quanto vengono automatizzati i processi di monitoraggio di iscrizioni e di guadagno totale.
L'approccio di Ex Ordo evidenzia l'importanza di funzionalità specializzate per la fase di preparazione e organizzazione dei contenuti.

Infine, la terza è **Cvent** [7], il quale si pone come una soluzione versatile per la gestione di eventi di vario tipo. Le sue caratteristiche principali sono simili a quelle viste in Fourwaves ed Ex Ordo, tra cui: il supporto per eventi in presenza, virtuali e ibridi, la gestione integrata dei pagamenti e registrazioni. Tuttavia, presenta anche delle novità come la creazione di opportunità di networking basate su intelligenza artificiale e l'introduzione di strumenti di marketing.
Cvent dimostra come la scalabilità e l'integrazione con altri sistemi aziendali siano caratteristiche importanti per le soluzioni di gestione eventi.

2.3 Analisi comparativa tra le applicazioni attuali e l'applicazione proposta

2.3.1 Punti di forza delle soluzioni esistenti

Dopo aver introdotto le applicazioni esistenti è possibile effettuare un'analisi comparativa con quella proposta in questa tesi. Ponendo l'attenzione principalmente sulle piattaforme precedentemente proposte: Fourwaves, Ex Ordo e Cvent.
Si può notare come queste presentino diversi punti di forza comuni, in particolare per quanto riguarda la facilità d'uso generale da parte degli utenti, l'integrazione social e di networking, la gestione delle partecipazioni e la compatibilità multi-piattaforma.

Nello specifico, tutte e tre le soluzioni si distinguono per le loro interfacce user-friendly. Fourwaves offre una web app personalizzabile che permette ai partecipanti di accedere facilmente alle informazioni del convegno. Ex Ordo è molto indicato per la gestione dei processi, grazie ad un'interfaccia intuitiva pensata per questo scopo specifico. Cvent, infine, presenta un'interfaccia versatile che si adatta efficacemente a diverse tipologie di eventi.

Per quanto riguarda l'integrazione social e di networking, Cvent si distingue per le sue funzionalità di condivisione e promozione degli eventi sui principali social media. Fourwaves offre, invece, interessanti funzionalità di networking, permettendo ai partecipanti di collegare i propri profili facilitando in tal modo le connessioni professionali.

Relativamente alla gestione delle registrazioni, Cvent offre un sistema con funzionalità per gestire pagamenti e biglietti. Allo stesso modo, Ex Ordo include un proprio sistema integrato per le registrazioni alle conferenze accademiche.

Inoltre, tutte e tre le piattaforme garantiscono un approccio multi-piattaforma. Fourwaves supporta eventi in formato virtuale, ibrido e in presenza, con un'applicazione accessibile da qualsiasi dispositivo. Cvent, oltre alla piattaforma web, offre delle app mobile dedicate, mentre Ex Ordo garantisce un sito web responsive ottimizzato, perciò, anche per l'accesso mobile.

Infine, ciascuna piattaforma presenta caratteristiche distintive nel proprio ambito specifico: Fourwaves mira ad una migliore fruizione online delle sessioni tramite "poster virtuali" con strumenti che sono capaci di simulare dei puntatori laser; Ex Ordo è particolarmente specializzaro nel processo di peer review, ovvero revisione tra pari, metodo per valutare la validità, la qualità e l'originalità di un lavoro scientifico destinato alla pubblicazione attraverso un giudizio formulato da uno o più valutatori con competenze simili a quelle del valutato; Cvent, invece, si distingue negli eventi aziendali con le sue integrazioni CRM, quest'ultimo definisce una categoria di software e applicazioni che aiutano le aziende a gestire, analizzare e ottimizzare le interazioni con clienti e futuri clienti, attraverso tutti i relativi dati.

Nonostante questi punti di forza significativi, le soluzioni esistenti presentano ancora alcune limitazioni che l'applicazione proposta in questa tesi si propone di superare.

2.3.2 Debolezze delle soluzioni correnti

Dopo aver analizzato i punti di forza, è possibile notare come, le stesse applicazioni, presentino anche diverse limitazioni significative.
Tra le debolezze riscontrate vi sono: una limitata interattività in tempo reale, una scarsa flessibilità per eventi pensati su più giornate e nella modifica degli eventi stessi e la mancanza di specificità verso i convegni accademici e professionali pensati principalmente per la fruizione in presenza.

Volendo analizzare ciascuna delle debolezze elencate, riguardo la limitata interattività in tempo reale, Fourwaves, offre funzionalità interattive in tempo reale diverse in base alle varie tipologie di eventi, questo, per l'appunto, non garantisce sempre interattività. Ex Ordo, essendo principalmente focalizzato sulla gestione pre-evento, non fornisce soluzione sufficienti per l'interazione tra relatori e partecipanti durante lo svolgimento effettivo del convegno.

Riguardo la scarsa flessibilità per eventi pensati su più giornate e della modifica generale degli eventi, Fourwaves può presentare problemi nel momento in cui vi sono più sessioni dello stesso evento in contemporanea. Cvent, invece, risulta essere limitato nel momento in cui si cerca di modificare un evento.

Infine, avendo preso in esame queste tre piattaforme, si evince come queste siano pensate per eventi molto vasti che per tale ragione prevedono delle conferenze in remoto. Questo aspetto può portare, talvolta, inefficienza nell'organizzazione di convegni principalmente accademici a sfondo universitario o professionali che invece hanno bisogno di un'organizzazione "raccolta" ed intuitiva che preveda informazioni sintetiche ma precise che sappiano indirizzare adeguatamente studenti o dipendenti ai luoghi di svolgimento delle sessioni dell'evento.

Queste limitazioni evidenziano la necessità di una soluzione più semplice e specializzata per la gestione dei convegni accademici e professionali. L'applicazione proposta in questa tesi mira a colmare questa necessità.

2.3.3 Differenziazione e innovazione dell'applicazione proposta

L'applicazione sviluppata in questa tesi si propone, quindi, di colmare le lacune identificate durante l'analisi appena effettuata sulle piattaforme prese in esame, differenziandosi in modi significativi.

Il concetto di base su cui si fonda lo sviluppo di questa applicazione è quello di fornire a studenti e relatori una piattaforma in grado di visualizzare e partecipare ai convegni proposti dai vari enti e, allo stesso tempo, organizzarne di nuovi con la possibilità di una partecipazione libera o su richiesta. Tutto ciò è reso disponibile attraverso una sola piattaforma capace di racchiudere più eventi contrapponendosi alla limitazione di alcuni servizi che invece propongono un'app per convegno. Questo garantisce, inoltre, grande scalabilità e duttilità della stessa applicazione che può essere utilizzata in vari campi, non solo quello accademico a sfondo prettamente universitario ma anche nel campo professionale.

Molto importante è anche la possibilità di modificare il programma degli eventi in qualsiasi momento, adattandosi ai possibili inconvenienti che possono avvenire durante l'organizzazione di tali eventi. I partecipanti agli eventi possono visualizzare le possibili modifiche in qualsiasi momento grazie alla pagina "programma" che permette di visualizzare tutte le sessioni giorno per giorno ed in base alla fascia oraria.

Da quest'ultima pagina dell'applicazione appena introdotta è possibile anche interagire in tempo reale con i relatori della sessione scrivendo delle domande nell'apposita sezione. In questo modo si cerca di incentivare la partecipazione attiva dei partecipanti per permettere un'esposizione più incalzante che tenga l'alta l'attenzione dei partecipanti.

Infine è importante menzionare l'approccio utilizzato per lo sviluppo dell'applicazione, tramite il quale è stato possibile progettare ed implementare le varie funzionalità dell'app. L'utilizzo di tecnologie come Flutter, Node.js e MySQL che interagiscono all'interno di un'architettura RESTful, garantisce prestazioni elevate e una manutenzione efficiente e semplice.

2.3.4 Resoconto dell'analisi

Dopo essere venuti a conoscenza delle piattaforme leader del settore, i loro vantaggi e svantaggi, e dopo aver compreso cosa può offrire l'applicazione oggetto di questa tesi, è possibile trattare gli aspetti più tecnici dell'implementazione di quest'ultima. Tramite la conoscenza di questi aspetti tecnici si comprende come, le funzionalità descritte in questo capitolo, sono state rese effettive e come queste si differenzino da quelle presenti nelle altre applicazioni, confermando in tal modo il ruolo potenzialmente rilevante che può avere quest'app nel settore.

Capitolo 3

Materiali e Metodi

3.1 Panoramica dell'architettura

L'applicazione per la gestione di convegni sviluppata in questa tesi si basa su un'architettura client-server scalabile. Questa architettura è stata progettata per garantire prestazioni ottimali e una gestione efficiente dei dati vista la gestione di molteplici convegni simultaneamente.
Nello specifico, l'architettura del sistema si compone di tre elementi fondamentali:

1. **Client**: Sviluppato utilizzando Flutter, il client fornisce un'interfaccia utente intuitiva. Questa parte dell'applicazione gestisce la presentazione dei dati e l'interazione dell'utente.

2. **Server**: Implementato con Node.js e il framework Express, il server funge da intermediario tra il client e il database. Espone un'API RESTful per la comunicazione con il client.

3. **Database**: L'utilizzo di MySQL garantisce la persistenza dei dati, garantendo l'integrità e la consistenza delle informazioni relative ai convegni, agli utenti e alle interazioni.

3.1.1 Flusso di dati e interazioni

Dopo aver approfondito l'architettura di base dell'applicazione, è importante analizzare anche il flusso di dati. Nel caso studiato in questa tesi viene utilizzato un modello bidirezionale.

Il flusso inizia nel momento in cui il client Flutter invia richieste HTTP al server Node.js attraverso l'API RESTful per esaudire tutte le operazioni che vengono effettuate dagli utenti nell'app, come la creazione di un convegno, la partecipazione ad un evento, l'invio di un commento eccetera. Successivamente, il server riceve ed elabora le richieste, interagisce con il database MySQL secondo le necessità richieste, ed invia risposte al client con i dati e le conferme di operazioni eseguite.

Per quanto riguarda invece le funzionalità come commenti e domande in diretta, viene forzato un refresh dello stato, in tal modo è possibile visualizzare tutte i commenti che vengono inseriti sotto i post dei convegni e le domande che vengono effettuate durante le sessioni.

A garantire la persistenza, la consistenza e l'integrità dei dati è il database MySQL le cui operazioni di lettura e scrittura vengono gestite dal server.

La Figura 3.1 illustra l'architettura complessiva del sistema, evidenziando i tre elementi fondamentali presentati precedentemente e di come questi interagiscono fra loro inviandosi i dati.

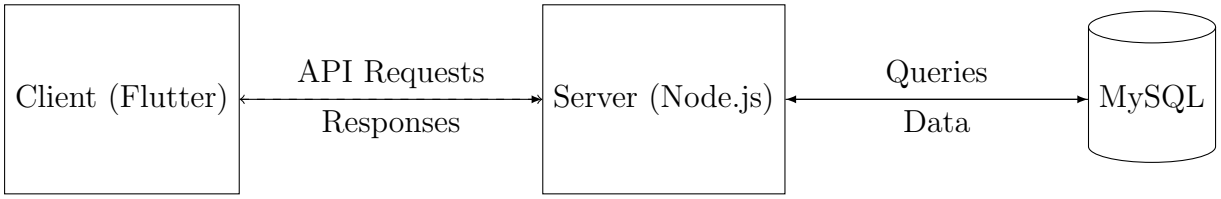

Figura 3.1: Architettura del sistema per la gestione di convegni

I vantaggi dell'utilizzo di questa architettura sono: **scalabilità**, in quanto, la separazione tra client e server, permette di scalare, in modo indipendente, i vari componenti in base al carico, **flessibilità**, poiché, l'utilizzo di un'API RESTful, facilita l'aggiunta di nuove funzionalità e l'integrazione con sistemi esterni ed, infine, la **manutenibilità**, che nasce dalla chiara separazione delle responsabilità tra i componenti, inoltre, semplifica la manutenzione e l'aggiornamento del sistema.

3.2 Tecnologie utilizzate

Oltre a definire l'architettura più adatta per la progettazione e l'implementazione di un'applicazione, è altrettanto importante scegliere accuratamente le tecnologie da utilizzare per lo sviluppo della stessa al fine di garantire prestazioni, scalabilità e la migliore esperienza utente possibile.
Procedendo ad analizzare le principali tecnologie utilizzate per il frontend, il backend e il database, verranno anche motivate le scelte effettuate e illustrati i vantaggi che offrono queste possono offrire.

3.2.1 Frontend

Per lo sviluppo del frontend dell'applicazione, come già anticipato nei precedenti capitoli, è stato scelto l'utilizzo di **Flutter** e il linguaggio di programmazione **Dart**.

Le motivazioni per le quali sono stati scelte queste due tecnologie per l'implementazione del frontend sono diverse. Prima di tutto Flutter permette di sviluppare applicazioni **cross-platform**, ovvero capaci di essere utilizzabili attraverso diverse piattaforme (iOS, Android, web) con un unico codice, riducendo significativamente i tempi e i costi di sviluppo. Inoltre, il fatto che le applicazioni siano cross-platform non incide sulle loro prestazioni in quanto queste ultime sono paragonabili a quelle delle app native. Con app native si intende l'insieme di app pensate principalmente per una piattaforma specifica.

Una funzionalità molto utile per gli sviluppatori che decidono di utilizzare Flutter è l'**hot Reload**. Questa funzionalità accelera notevolmente il processo di sviluppo, permettendo di vedere immediatamente l'effetto delle modifiche apportate al codice. L'hot reload è molto importante in quanto Flutter, essendo basato su l'utilizzo di una vasta gamma di **widget**, ovvero dei meccanismi che aiutano lo sviluppatore ad implementare molte funzionalità che hanno un risvolto nelle interfacce degli utenti, necessita di continui aggiornamenti per visualizzare le modifiche che vengono apportate tramite l'utilizzo proprio dei diversi widget.

In generale, i vantaggi nell'utilizzo di Flutter per lo sviluppo di applicazioni web e mobile riguardano diversi aspetti relativi, in primo luogo, all'esperienza di utilizzo dell'utente, infatti, come già detto, grazie allo sviluppo cross-platform, l'esperienza utente risulta coerente su tutte le piattaforme. In secondo luogo, il suo utilizzo, facilita il lavoro degli sviluppatori i quali possono approcciare Dart senza troppe difficoltà in quanto simile a linguaggi conosciuti come Java e JavaScript.

Dart permette di scrivere codice efficiente garantendo un'alta manutenibilità, il tutto facilitato dalla compilazione **ahead-of-time (AOT)** caratteristica di Flutter, che si basa sull'ottimizzazione degli avvii fornendo, in tal modo, prestazioni fluide per l'applicazione sviluppata. Infine, Flutter offre diverse soluzioni per la gestione dello stato dell'applicazione, facilitando lo sviluppo di applicazioni molto complesse.

3.2.2 Backend

Per quanto riguarda il lato backend dell'applicazione, è stato scelto l'utilizzato di **Node.js** supportato dal framework **Express**.

Le ragioni per le quali si è optato per questa scelta riguardano, per prima cosa, le prestazioni elevate che vengono garantite da Node.js per applicazioni ad alta concorrenza, come il caso specifico dell'applicazione trattata in questa tesi. Node.js supporta, tra l'altro, API ad alte prestazioni grazie alla gestione di I/O non bloccanti, ovvero basate su chiamate che ritornano non appena possibile, anche se l'I/O non è ancora terminato, migliorando le performance.

Node.js presenta, inoltre, un'ampia disponibilità di pacchetti scaricabili tramite il gestore di pacchetti ufficiale **npm**, abbreviazione di **Node Package Manager**. Tramite npm è anche possibile installare librerie, framework come Express e altri strumenti di sviluppo di progetti, inoltre, permette di accedere a progetti Node.js sicuri già esistenti. Tutti questi mezzi velocizzano la fase di sviluppo dell'applicazione, aiutando notevolmeente lo sviluppatore.

Infine, Node.js, si basa sul linguaggio di programmazione JavaScript che, come detto, risulta essere simile, a livello di sintassi, a Dart che viene utilizzato nel frontend con Flutter. In questo modo la combinazione di utilizzo tra Dart per il frontend e JavaScript per il backend risulta essere un'ottima scelta di sviluppo per evitare stravolgimenti nella scrittura del codice.

Per quanto riguarda invece il ruolo del framework Express, questo ha il compito di semplificare la creazione di **API RESTful** attraverso l'implementazione di un routing intuitivo, che consiste nel processo di selezione del percorso in una rete, e middleware flessibili, ovvero un software che consente la comunicazione e la gestione dei dati per le applicazioni distribuite.

Express, in generale, permette grande libertà nella strutturazione dell'applicazione, adattandosi alle esigenze specifiche del progetto.

3.2.3 Database

Nello sviluppo di un'applicazione che richiede la gestione di una grossa mole di dati è importante garantire la persistenza dei dati stessi, per tale ragione è stato scelto **MySQL** come sistema di gestione del **database relazionale**.

La scelta di MySQL è dovuta alla sua capacità di garantire l'integrità referenziale e la consistenza dei dati. Tale aspetto, come già detto, è molto importante per la gestione di molteplici informazioni come quelle dei convegni. MySQL, infatti, offre buone prestazioni anche con grandi volumi di dati, supportando la scalabilità orizzontale, la quale comporta l'aggiunta di risorse simili per la gestione del carico di lavoro.

MySQL si basa sul linguaggio SQL che permette query complesse ed operazioni come il join tra tabelle, utili per estrarre informazioni correlate in modo efficiente. MySQL, a tal proposito, supporta transazioni **ACID** (**Atomicity**, **Consistency**, **Isolation**, **Durability**)[8]. Analizzando le singole proprietà, con atomicità si intende che ogni transazione viene gestita come singola unità, impedendo che i dati vadano persi o vengano corrotti nel caso in cui, ad esempio, la sorgente di dati in streaming venga interrotta a metà del flusso. Con coerenza si intende l'intenzione di garantire che le transazioni apportino modifiche alle tabelle solo con modalità predefinite e prevedibili. Con isolamento si fa riferimento al caso in cui più utenti leggono e scrivono contemporaneamente sulla stessa tabella, l'isolamento delle loro transazioni assicura che transazioni contemporanee non interferiscano fra loro. Infine, grazie alla durabilità viene garantito che le modifiche ai dati apportate da transazioni eseguite con successo vengano salvate, anche in caso di guasto del sistema.

Struttura del database e principali tabelle

La struttura del database MySQL è stata progettata per gestire efficacemente tutti gli aspetti dei convegni, inclusi utenti, eventi, sessioni, e interazioni. Le principali tabelle includono:

- `users`: Memorizza informazioni sugli utenti, inclusi nome, cognome, email, password e organizzazione.
- `posts`: Contiene i dettagli di ogni convegno, inclusi titolo, descrizione, date, luogo e informazioni sull'autore.
- `event_sessions`: Rappresenta le singole sessioni all'interno di un convegno, con dettagli come titolo, descrizione, data e orari.
- `comments`: Registra i commenti degli utenti relativi ai post (convegni).
- `event_participations`: Gestisce le registrazioni degli utenti ai convegni, incluso lo stato della partecipazione.
- `questions`: Memorizza le domande poste dagli utenti durante le sessioni dei convegni.
- `notifications`: Gestisce le notifiche inviate agli utenti, relative alla partecipazione agli eventi.

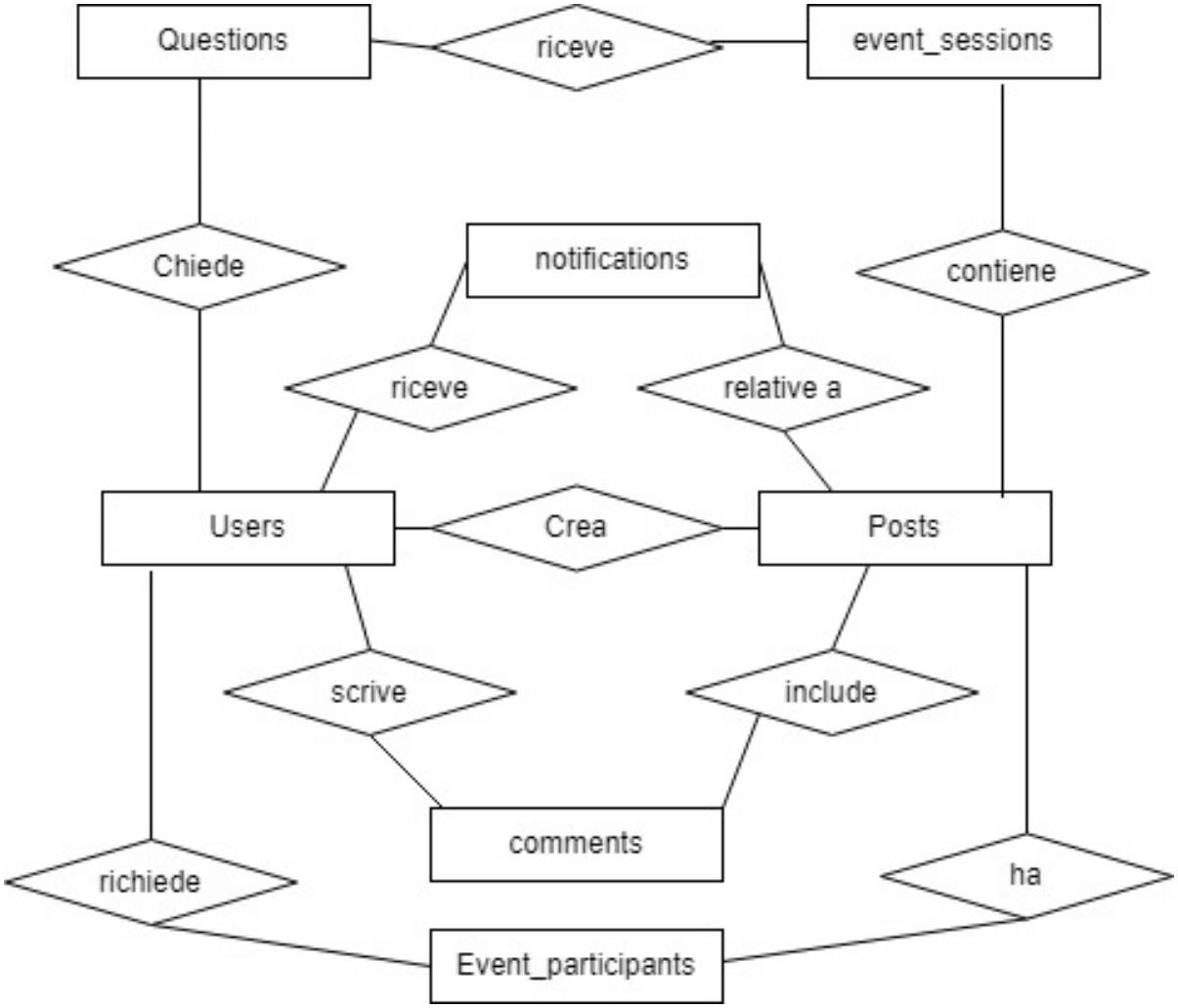

Figura 3.2: Schema ER del database

Come è possibile vedere dalla struttura raffigurata in figura 3.2, il database è stato progettato per permettere una gestione efficiente e scalabile dei dati relativi ai convegni, ponendo in primo piano la fattibilità di tutte le funzionalità e quindi le operazioni che possono essere effettuate nell'applicazione.

Nello specifico è stata posta l'attenzione sulla gestione completa degli utenti e delle loro interazioni, come la registrazione, il login e la modifica del proprio profilo che comprende cambio della password e cancellazione del profilo. Inoltre, è stato attenzionato tutto il lato organizzativo dei convegni e delle loro sessioni con annesso tracciamento delle partecipazioni a loro riferite, il sistema di notifiche che scaturisce dalla partecipazione e anche le possibili modifiche ai dettagli sia dei convegni che delle sessioni. Infine, è stato attenzionato anche il sistema di commenti e domande per garantire l'interazione durante gli eventi.

Un ultimo importante concetto relativo alla struttura del database riguarda l'utilizzo di **chiavi esterne** ed **indici** che garantiscono l'integrità referenziale tra le varie tabelle e ottimizza le prestazioni delle query. Tali aspetti sono cruciali per un'applicazione di questo genere data la necessità di gestire grandi volumi di dati e interazioni frequenti tra le varie entità.

3.3 Implementazione del frontend

A livello tecnico, l'implementazione del frontend dell'applicazione, si basa su una struttura modulare che segue le **best practice** di sviluppo per garantire manutenibilità e scalabilità. Con best practice si intende l'utilizzo di tutti quei metodi che bisognerebbe seguire per assicurarsi una corretta implementazione del progetto, mirando ad evitare errori banali, garantendo, quindi, una buona base di sviluppo.

Iniziando ad analizzare la struttura dell'applicazione è possibile identificare un file principale che ha il compito di organizzare le altre pagine. Il file in questione è il *main.dart*, il quale funge da punto di ingresso per l'applicazione.

Il main presenta due elementi su cui si basa l'intera app. Analizzandoli singolarmente, il primo elemento è la classe **MyApp**, che possiamo definire come una radice poiché ha il compito di inizializzare l'applicazione, inoltre, definisce il tema e la schermata iniziale dell'applicazione. MyApp estende a sua volta la classe **StatelessWidget** che rende l'app un **widget**. All'interno di MyApp viene poi invocato il costruttore della classe, nonché il secondo elemento su cui si basa l'app, ovvero **MyHomePage**. Quest'ultima si occupa di inizializzare il codice di tutte le schermate dell'app. Inoltre, anche MyHomePage estende una classe come MyApp ma in questo caso si tratta della classe **StatefulWidget** che è cruciale in quanto gli elementi contenuti in essa cambieranno in alle azioni compiute dall'utente.

Gli elementi contenuti in StatefulWidget, come detto, sono le schermate vere e proprie dell'applicazione. Nello specifico vi sono:

- **HomePage**: Tramite questa schermata è possibile visualizzare l'elenco dei convegni, visualizzarne i dettagli e parteciparvi. Inoltre vi è l'opzione di ricerca per facilitare la navigazione.
- **ProgramPage**: In questa schermata viene mostrato il programma dei convegni con le relative sessioni organizzate per giorni e fascia oraria.
- **CreatepostPage**: Questa schermata permette la creazione di nuovi convegni compilando una sorta di form con tutte le informazioni necessarie.
- **AccountPage**: In questa schermata è possibile gestire le informazioni dell'account utente, visualizzare i convegni pubblicati e modificarli.
- **MorePage**: Infine, questa schermata fornisce maggiori informazioni sull'app.

La navigazione tra queste schermate è possibile grazie all'implementazione di una barra di navigazione inferiore personalizzata, che offre un'esperienza utente rapida ed intuitiva.

3.3.1 Gestione dello stato

Per permettere una corretta navigazione tra le schermate appena presentate, è importante la gestione dello stato dell'applicazione. Per tale ragione, viene utilizzato il **pattern Provider**, in quanto è capace di offrire un approccio efficiente e scalabile per la condivisione dei dati tra i widget.

Nello specifico, Provider è utilizzato per l'iniezione delle dipendenze e la gestione dello stato globale. Ciò è possibile grazie all'implementazione della classe **ChangeNotifier** che viene utilizzato all'interno delle classi provider per notificare i widget dei cambiamenti di stato e quindi delle azioni eseguite dagli utenti.

Nel codice seguente, relativo al file main.dart, è possibile vedere i provider principali che sono stati implementati per gestire il cambiamento di stato di tutti gli elementi dell'app:

```
MultiProvider(
  providers: [
    Provider<NotificationService>(
      create: (_) => notificationService
    ),
    ChangeNotifierProvider(
      create: (context) => UserProvider(
        userService,
        notificationService)
    ),
    ChangeNotifierProvider(
      create: (context) => PostsProvider(
        postService,
        eventParticipationService,
        questionService)
    ),
    ChangeNotifierProvider(
      create: (context) => CommentsProvider(commentService)
    ),
    ChangeNotifierProvider(
      create: (context) => EventSessionProvider(
        ↪ eventSessionService)
    ),
  ],
  child: const MyApp(),
)
```

3.3.2 Implementazione delle funzionalità chiave

Un altro importante aspetto tecnico relativo al frontend riguarda l'implementazione delle funzionalità chiave dell'applicazione. Per comprendere meglio il loro sviluppo è importante trattarle singolarmente. In ordine vi sono: creazione e gestione dei convegni, sistema di partecipazione agli eventi ed il sistema che si occupa dei commenti e delle domande in tempo reale.

Creazione e gestione dei convegni

La prima funzionalità da approfondire a livello tecnico è la creazione e gestione dei convegni, la quale viene implementata principalmente nella schermata *CreatepostPage*. In questa pagina, accessibile solo dopo aver fatto l'accesso o essersi registrati all'interno dell'app, verrà visualizzato un form completo con tutti i campi relativi ai dettagli del convegno:

1. **Informazioni di base**: Per prima cosa vengono richieste delle informazioni di base come titolo, descrizione, luogo, date.

2. **Dettagli delle sessioni**: In base alle date che vengono inserite verranno richieste maggiori informazioni riguardo le singole sessioni per ogni giorno. Le informazioni sono titolo, descrizione, orario e luogo. Inoltre vi è la possibile di aggiungere più sessioni per lo stesso giorno.

3. **Caricamento immagini**: Se necessario è possibile aggiungere una immagine ad esempio del luogo o la locandina dell'evento.

4. **Configurazioni avanzate**: Vi sono, infine, delle opzioni aggiuntive per abilitare/disabilitare commenti e moderazione oppure aggiungere direttamente alla partecipazione altri relatori oltre al creatore dell'evento, oppure ancora caricare un file .csv che permetta di registrare all'applicazione ed ammettere all'evento un grande numero di utenti contemporaneamente.

```
Widget _buildFormFields() {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      TextFormField(
        controller: _titleController,
        decoration: const InputDecoration(labelText: 'Titolo/
            ↪ tipologia evento *'),
        // ...
      ),
      // ... (altri campi del form)
    ],
  );
}
```

Sistema di partecipazione agli eventi

La seconda funzionalità da approfondire riguarda il sistema di partecipazione agli eventi, il quale è implementato nella schermata *HomePage* dove vengono gestiti vari stati di partecipazione degli utenti. Infatti, quando un utente decide di partecipare ad un evento dovrà premere sul pulsante "Partecipa", a questo punto se l'evento non presenta moderazione apparirà immediatamente lo stato "accettata" in verde al posto del pulsante partecipa. Se invece è richiesta la moderazione per quell'evento, apparirà la scritta "Richiesta inviata" che indica l'invio di una notifica al creatore dell'evento che, successivamente, attraverso la schermata *AccountPage*, visualizzando le notifiche, deciderà se accettare o meno l'utente. In base alla decisione del creatore dell'evento apparirà all'utente che ha effettuato la richiesta lo stato "accettato" o "rifiutato".

```
Widget _buildParticipateButton(Post post) {
  // ... (logica per determinare lo stato di partecipazione)
  return FutureBuilder<String>(
    future: postsProvider.getParticipationStatus(post.id!,
       ↪ userProvider.user!.id!),
    builder: (context, snapshot) {
      // ... (rendering del pulsante appropriato in base allo
         ↪ stato)
    },
  );
}
```

Implementazione del sistema di commenti e domande in tempo reale

La terza funzionalità da approfondire riguarda il sistema di commenti e domande in tempo reale. Partendo con l'implementazione relativa alla scrittura dei commenti, questa è implementata nella schermata *HomePage* e *AccountPage* ed è disponibile solo nel momento in cui l'utente, durante la creazione dell'evento, abilita i commenti.

La possibilità di scrivere commenti, come la partecipazione, è data solo a coloro che accedono all'applicazione tramite il proprio account. L'utente visualizza la sezione dei commenti all'interno della card dell'evento nella parte inferiore. Quando l'utente pubblica un commento, questo sarà visualizzabile a tutti con il nome dell'autore. Inoltre, per una questione di ordine e di estetica, ogni commento viene separato tramite una linea sottile per facilitarne la comprensione dell'autore e del testo stesso.

I commenti, infine, vengono aggiornati ad ogni refresh della pagina, in quanto la query di ricerca viene effettuata ogni volta che si accede alla Home.

```
Widget _buildCommentsList(Post post, CommentsProvider
    ↪ commentsProvider) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      FutureBuilder<List<Comment>>(
        future: commentsProvider.getCommentsForPost(post.id!),
        builder: (context, snapshot) {
          // ... (rendering della lista dei commenti)
        },
      ),
      _CommentInput(
        commentsProvider: commentsProvider,
        postId: post.id!,
        onCommentAdded: () {
          setState(() {});  // Aggiorna l'UI quando viene
              ↪ aggiunto un nuovo commento
        },
      ),
    ],
  );
}
```

Per quanto riguarda invece il sistema di scrittura delle domande in tempo reale, questa è stata implementata nella schermata *ProgramPage*. Nella quale come anticipato, non solo vengono visualizzate tutte le sessioni organizzate in giorni e fasce orarie, ma all'interno del riquadro di ogni sessione, è possibile premere sull'icona in alto a destra raffigurante delle "nuvolette di messaggi". Dopo aver premuto sull'icona apparirà una piccola interfaccia dalla quale sarà possibile scrivere e allo stesso tempo visualizzare le domande relative unicamente a quella sessione.

Inoltre, l'icona sopracitata, apparirà all'utente solo nella fascia oraria specificata dalla sessione stessa, in modo tale che la sezione delle domande sia relegata al momento di fruizione della sessione.

Dopo aver scritto la domanda, l'interfaccia si chiuderà automaticamente ed, una volta riaperta, sarà possibile vedere la propria domanda "pubblicata" e allo stesso tempo vederne di nuove da parte degli altri utenti. Come nel caso dei commenti, per ogni domanda sarà possibile visualizzare l'autore.

La funzione che si occupa di questa funzionalità specifica relativa alle domande è *_showAskQuestionDialog*:

```
1 void _showAskQuestionDialog(BuildContext context, EventSession session)
    ↪ {
2   final questionController = TextEditingController();
3   final postProvider = Provider.of<PostsProvider>(context, listen:
      ↪ false);
4   final userId = Provider.of<UserProvider>(context, listen: false).user
      ↪ !.id!;
```

```
showDialog(
  context: context,
  builder: (context) => AlertDialog(
    title: const Text('Domande per la sessione'),
    content: Column(
      children: [
        Flexible(
          child: FutureBuilder<List<Question>>(
            future: postProvider.getQuestionsForSession(session.id!),
            builder: (context, snapshot) {
              if (snapshot.connectionState == ConnectionState.waiting
                  ↪ ) {
                return const CircularProgressIndicator();
              } else if (snapshot.hasError || !snapshot.hasData) {
                return Text(snapshot.hasError ? 'Errore' : 'Nessuna
                    ↪ domanda');
              } else {
                return ListView.builder(
                  itemCount: snapshot.data!.length,
                  itemBuilder: (context, index) {
                    final question = snapshot.data![index];
                    return ListTile(
                      title: Text(question.question),
                      subtitle: Text('${question.userName} - ${
                          ↪ question.timeStamp}'),
                    );
                  },
                );
              }
            },
          ),
        ),
```

```
1
2           TextField(
3             controller: questionController,
4             decoration: const InputDecoration(hintText: 'Scrivi la tua
                ↪ domanda...'),
5           ),
6         ],
7       ),
8       actions: [
9         TextButton(
10          onPressed: () => Navigator.of(context).pop(),
11          child: const Text('Chiudi'),
12        ),
13        TextButton(
14          onPressed: () async {
15            if (questionController.text.isNotEmpty) {
16              await postProvider.insertQuestion(session.id!, userId,
                  ↪ questionController.text);
17              Navigator.of(context).pop();
18              ScaffoldMessenger.of(context).showSnackBar(
19                const SnackBar(content: Text('Domanda inviata')),
20              );
21            }
22          },
23          child: const Text('Invia'),
24        ),
25      ],
26    ),
27  );
28 }
```

Analizzando l'implementazione riportata sopra, è possibile notare l'utilizzo di diversi elementi che sono tipici di Flutter e Dart.

Per prima cosa è possibile notare l'utilizzo di **FutureBuilder** che permette una visualizzazione dinamica delle domande, permettendo il caricamento e la conseguente visualizzazione delle domande esistenti per la sessione corrente. Unito a questo funzionamento di controllo di domande già esistenti, vi è l'implementazione di un feedback visivo per l'utente, aggiornandolo riguardo il caricamento delle domande e notificandogli possibili errori.

Oltre all'utilizzo di FutureBuilder vi è quello relativo alle **chiamate asincrone** al fine di inviare nuove domande al backend, garantendo che l'interfaccia utente rimanga disponibile e reattiva a nuove istruzioni. Anche per quanto riguarda l'invio delle domande, vengono forniti dei feedback immediati all'utente relativi all'invio di una domanda attraverso uno **SnackBar**. Questo non è altro che una sorta di pop-up che appare nella parte inferiore dello schermo del dispositivo che indica l'avvenuto invio della domanda.

Sebbene non implementi un vero e proprio sistema di aggiornamento in tempo reale (come **WebSocket**), l'implementazione è progettata per essere facilmente estendibile per supportare aggiornamenti in tempo reale. Attraverso le seguenti righe di codice di esempio, si può vedere come è stato simulato un sistema di aggiornamento in tempo reale attraverso un refresh automatico ogni 30 secondi:

Listing 3.1: Esempio di aggiornamento periodico

```
1 Timer.periodic(Duration(seconds: 30), (Timer t) =>
    _refreshQuestions());
```

3.4 Sviluppo del backend

Il backend dell'applicazione, come già anticipo nei capitoli precedenti, è stato sviluppato utilizzando Node.js con il framework Express.js. Grazie a quest'ultimo è stato possibile fornire un servizio **API RESTful** robusto e scalabile, ma soprattutto adatto sia a piattaforme mobile che web.

Analizzando l'API è possibile comprendere come, la sua struttura sia modulare, con l'utilizzo di **routes** separate per ciascuna entità principale dell'applicazione. Le entità corrispondono, sostanzialmente, alle tabelle del database.

Di seguito è riportata l'implementazione delle routes:

```
1 app.use('/api/users', require('./src/routes/userRoutes'));
2 app.use('/api/posts', require('./src/routes/postRoutes'));
3 app.use('/api/event-sessions', require('./src/routes/eventSessionRoutes
    ↪ '));
4 app.use('/api/comments', require('./src/routes/commentRoutes'));
5 app.use('/api/questions', require('./src/routes/questionRoutes'));
6 app.use('/api/event-participations', require('./src/routes/
    ↪ eventParticipationRoutes'));
7 app.use('/api/notifications', require('./src/routes/notificationRoutes
    ↪ '));
```

Grazie a questa organizzazione, basata su routes separate, si ha una chiara separazione delle responsabilità, garantendo maggiore facilità nell'individuazione di possibili errori, facilitando, in tal modo, la manutenzione e allo stesso tempo la potenziale espansione dell'API.

Gestione delle richieste e risposte

Oltre alle routes, nello sviluppo del backend, è importante la gestione delle richieste. Queste ultime vengono gestite utilizzando il middleware messo a disposizione dal framework Express, unito ai **controller** dedicati ad ogni entità. Il compito dei controller è quello di interagire con il database e formattarne le risposte in modo tale da inviare al frontend i dati richiesti e farli visualizzare correttamente all'utente. Inoltre, i controller, lavoro con le routes delle entità corrispondenti. Questo garantisce che le richieste sia indirizzate correttamente evitando errori.

Nell'esempio qui sotto è possibile visualizzare l'implementazione del file route dedicato alle domande:

```
1 const router = express.Router();
2 const questionController = require('../controllers/questionController')
    ↪ ;
3
4 router.post('/', questionController.insertQuestion);
5 router.get('/session/:sessionId', questionController.
    ↪ getQuestionsForSession);
```

Come è possibile vedere da questa implementazione, il file richiama il controller delle domande ed i metodi implementati al suo interno che agiscono in base alla richiesta http.

Nell'esempio qui sotto, è possibile visualizzare, come esempio, il metodo interno al controller delle domande che si occupa di gestire l'inserimento di una nuova domanda:

```
exports.insertQuestion = async (req, res) => {
      const { sessionId, userId, question } = req.body;
        {...}
      try {
        {...}
        await db.query(
          `INSERT INTO questions (session_id, user_id, user_name,
             ↪ user_surname, question, timestamp)
           VALUES (?, ?, ?, ?, ?, ?)`,
          [sessionId, userId, userName, userSurname, question, new Date
             ↪ ().toISOString()]
        );

        console.log('Question inserted successfully');
        res.status(201).json({ message: 'Question inserted successfully
           ↪ ' });
      } catch (error) {
        console.error('Error in insertQuestion:', error);
        res.status(500).json({ message: 'Error inserting question',
           ↪ error: error.message });
      }
    };
```

Come si può vedere, viene implementata un'eccezione **try/catch** in cui viene implementat la query di inserimento nel database della domanda ed, in caso di errori, l'eccezione viene catturata restituendo l'errore.

3.5 Integrazione e comunicazione client-server

Le richieste che vengono effettuate durante la comunicazione tra client e server, sono richieste **HTTP** che vengono utilizzate per le operazioni sul database. Le operazioni in questione sono definite **CRUD**, rispettivamente CREATE, READ, UPDATE, DELETE. Attraverso queste è possibile manipolare il database in base alle azioni effettuate dall'utente e alle funzionalità messe a disposizione dall'app.

Le operazioni CRUD corrispondono a delle richieste HTTP specifiche. In particolare tramite **GET** è possibile recuperare dati all'interno del database e mostrarle all'utente. Tramite **POST** è possibile inviare nuovi dati all'interno del database, in questo modo è possibile creare nuovi record all'interno delle varie entità. Tramite **PUT/PATCH** è invece possibile aggiornare dati già esistenti nel database andando a modificare i record delle tabelle. Infine, tramite **DELETE** è possibile rimuovere i dati dal database cancellando record delle tabelle.

Queste richieste HTTP vengono inviate lato client tramite l'utilizzo di **Provider**. Questi ultimi sono capaci di incapsulare la logica per effettuare chiamate API al backend, garantendo la comunicazione e interazione tra client e server.

3.6 Implementazione del database

Nelle sezioni precedenti è stata analizzata la struttura generale del database, presentando le entità create e le relazioni fra esse. Nella seguente sezione, invece, verranno trattati gli aspetti tecnici dell'implementazione del database e le ottimizzazioni per renderlo più efficiente possibile.

Per prima cosa è importante analizzare la connessione al database MySQL che viene gestita attraverso un **pool** di connessioni, implementato nel file **db.js**:

```
const mysql = require('mysql2/promise');

const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  port: process.env.DB_PORT || 3306,
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});
```

I vantaggi di questo approccio sono relativi alla gestione di connessioni multiple che vengono gestite tramite un limite massimo consentito. Tali connessioni possono essere riutilizzate in modo tale da ridurre l'overhead di creazione, ovvero un eccesso che potrebbe causare problemi generali al sistema. Infine, un ultimo vantaggio, riguarda la configurazione flessibile che viene garantita grazie alle variabili d'ambiente implementate.

Per quanto riguarda invece l'ottimizzazione del database, questa comprende diversi campi tra cui la sicurezza. Ad esempio, nella seguente query che si occupa del recupero delle domande per una sessione specifica, viene utilizzato un parametro *preparato* per prevenire le **SQL injection**, ovvero dei possibili attacchi informatici che mirano a rubare i dati.

```
1        const [rows] = await db.query('SELECT * FROM questions WHERE
            ↪ session_id = ?', [sessionId]);
```

Un'altra ottimizzazione riguarda l'utilizzo di query combinate che inseriscono dati da più fonti in un'unica operazione, come nell'esempio sotto relativo all'inserimento di una nuova domanda unendo i dati dell'utente. Queste query sono anche dette query parametrizzate, proprio perché fanno uso di paramatri.

```
1        await db.query(
2          `INSERT INTO questions (session_id, user_id, user_name,
               ↪ user_surname, question, timestamp)
3           VALUES (?, ?, ?, ?, ?, ?)`,
4          [sessionId, userId, userName, userSurname, question, new Date
               ↪ ().toISOString()]
5        );
```

Infine, l'ottimizzazione più consistente, riguarda l'implementazione generale del database e la sua struttura, che si compone di diversi fattori. Il primo fattore relativo è relativo all'ottimizzazione della struttura è l'utilizzo di indici sulle colonne più utilizzate nelle operazioni "WHERE" e "JOIN", un esempio è il caso di *session_id* nella tabella *questions*.

Un altro fattore è l'utilizzo di pool di connessioni, trattato anche prima, che riduce il tempo di latenza per l'apertura di nuove connessioni. A questo fattore, si aggiunge anche la verifica dell'avvenuta connessione col database, questa viene gestita attraverso la seguente funzione:

```
async function checkDatabaseConnection() {
  try {
    const connection = await db.getConnection();
    console.log('Successfully connected to the database.');
    connection.release();
  } catch (error) {
    console.error('Failed to connect to the database:', error);
    process.exit(1);
  }
}
```

Questa funzione verifica la connessione al database all'avvio del server, garantendo che l'applicazione non proceda se il database risulta non accessibile.

Infine, un ultimo fattore di ottimizzazione è lo **scheduling** di operazioni adibite alla manutenzione nel database stesso. Un esempio è la rimozione di post risalenti a date precedenti e quindi scaduti.

```
const runDeleteOldPosts = async () => {
   try {
     await postController.deleteOldPosts({}, { json: () => console.
        ↪ log('Old posts deleted successfully') });
     console.log('Scheduled job: deleteOldPosts completed
        ↪ successfully');
   } catch (error) { console.error('Scheduled job: deleteOldPosts
      ↪ failed:', error); } };
 const scheduleDeleteOldPosts = () => {
   cron.schedule('0 0 * * *', () => { runDeleteOldPosts(); });
 };
```

3.7 Utilizzo di XAMPP nello sviluppo

Durante la fase di sviluppo dell'applicazione, è stato utilizzato XAMPP, uno strumento utile per creare un ambiente di sviluppo locale prima della distribuzione dell'applicazione sul mercato. In particolare permette di mettere a disposizione dell'utente il server web e il database.

XAMPP (Cross-Platform, Apache, MySQL, PHP and Perl)[9] in particolare utilizza il modulo **Apache** installato automaticamente durante il primo avvio. Questo modulo, quando viene avviato, fornisce un server web locale, in modo tale da testare le funzionalità del backend. Il secondo modulo che viene messo a disposizione da XAMPP è MySQL che garantisce un server di database locale, permettendo di sviluppare e testare le funzionalità del database senza la necessità di una connessione remota.

L'impiego di XAMPP ha offerto diversi vantaggi durante il processo di sviluppo, uno di questi è già stato trattato, ovvero quello relativo all'ambiente di sviluppo che essendo pre-configurato ha ridotto il tempo necessario per la configurazione iniziale. Un altro vantaggio è relativo all'intuitività dell'interfaccia di controllo di XAMPP, che ha permesso di avviare e arrestare facilmente i servizi necessari (Apache e MySQL) secondo le esigenze di sviluppo. Oltre all'interfaccia di controllo di XAMPP, anche quella di **phpMyAdmin** ha notevolmente semplificato la gestione del database attraverso funzionalità che permettono di visualizzare i dati nelle tabelle, eseguire query e modificare rapidamente il database.

Infine, è importante tenere conto che XAMPP garantisce un ambiente di sviluppo facilmente replicabile, importante per lo sviluppo su diverse macchine.

3.7.1 Workflow di sviluppo con XAMPP

Il procedimento di avvio di XAMPP durante lo sviluppo dell'applicazione prevedeva per prima cosa l'avvio dei moduli trattati precedentemente, ovvero Apache e MySQL, tramite il pannello di controllo di XAMPP. Successivamente era necessario avviare l'interfaccia di phpMyAdmin in modo tale da visualizzare e gestire i dati del database, al fine di eseguire il debugging e la verifica delle operazioni sul database. Infine, venivano effettuate ulteriori test delle operazioni tra frontend e backend.

Capitolo 4

Interfaccia Utente e Discussione

4.1 Funzionalità principali dell'applicazione

L'applicazione oggetto di questa tesi si propone come una valida soluzione per coloro che desiderano organizzare o partecipare a convegni accademici e professionali. Tale applicazione offre diverse funzionalità mirate a migliorare l'organizzazione di eventi soprattutto universitari.

4.1.1 Gestione eventi

La funzionalità centrale del progetto riguarda la possibilità di creare gli eventi, richiedendo all'utente la compilazione di un form (Figura 4.1) con tutti i dettagli necessari a definire le informazioni del convegno. Tale funzionalità è disponibile nella pagina chiamata "Event Creation" da cui è possibile visualizzare nel dettaglio tutti i campi richiesti per la creazione dell'evento tra cui quelli obbligatori come: titolo o tipologia evento, descrizione, data e luogo. Questi vengono poi richiesti anche nello specifico per le varie sessioni che si desidera creare. Tra i campi non obbligatori vi sono invece il caricamento di una immagine e l'invito di relatori attraverso l'inserimento delle loro email.

Vi sono poi dei **checkbox** relativi all'abilitazione di commenti o di moderazione dell'evento e la possibilità di caricare un file nel formato csv che consenta la registrazione in massa di più partecipanti contemporaneamente.

Per quanto riguarda la gestione degli eventi vi è inoltre la possibilità di modificarli attraverso la pagina denominata "Account" dalla quale l'utente che ha creato gli eventi può visualizzarli e modificarli (Figura 4.2) premendo sull'icona in alto a destra rispetto alla card del convegno. Nello specifico sarà possibile modificare i dettagli dell'evento, aggiornarne le relative sessioni e, se necessario, eliminarlo.

Infine, tramite la pagina "Program", è possibile visualizzare il programma che raffigura le sessioni organizzate per giorni e fascia orario in modo tale da tenere traccia degli impegni degli eventi a cui si è data la propria partecipazione (Figura 4.3). Inoltre, premendo sulle card delle sessioni è possibile visualizzarne i dettagli e, solo durante la fascia oraria dello svolgimento della sessione, apparirà una icona che permette di effettuare domande e visualizzare quelle fatte.

4.1.2 Sistema di partecipazione

Un'altra funzionalità importante è quella relativa alla partecipazione agli eventi. Tale funzionalità è progettata per essere intuitiva ed efficiente. L'utente, recandosi nella pagina "Home", potrà visualizzare tutti gli eventi programmati con anche la possibilità di commentare attraverso l'interfaccia posta nel lato inferiore della card e decidere di partecipare agli eventi che più gli interessano premendo sul pulsante "Partecipa" in alto a destra della card (Figura 4.4). Una volta premuto se l'evento non richiede moderazione, l'utente sarà considerato "accettato" e potrà da subito visualizzare nel programma le sessioni dell'evento. Se invece è stata richiesta la moderazione l'utente, dovrà aspettare la decisione del creatore dell'evento prima di potersi ritenere un partecipante all'evento. Il proprietario dell'evento riceverà una notifica visualizzabile dalla propria pagina "Account" dalla quale potrà decidere se accettare o rifiutare la richiesta di partecipazione (Figura 4.5). Anche in questo caso, premendo sulla card, l'utente potrà visualizzare maggiori dettagli sull'evento, tra cui le singole sessioni previste.

Il sistema di partecipazione bilancia efficacemente l'intuitività d'uso per i partecipanti con il controllo necessario per gli organizzatori, garantendo un'esperienza fluida e sicura per tutti gli utenti della piattaforma.

4.2 Discussione

4.2.1 Punti di forza dell'applicazione

Confrontando l'applicazione con le soluzioni analizzate nei capitoli precedenti, è possibile notare dei punti di forza che rendono l'applicazione una potenziale soluzione per l'utente capace di differenziarsi dalle altre.

In primo luogo, l'applicazione permette la creazione e la gestione di più convegni attraverso una singola piattaforma, contrapponendosi a servizi che propongono lo sviluppo di applicazioni pensate per un singolo evento.

Un altro punto di forza è sicuramente la flessibilità dell'applicazione, in quanto permette agli organizzatori di adattare ogni evento alle proprie esigenze specifiche, dal tipo di partecipazione (libera o moderata) alla struttura del programma che può basarsi su una o più giornate con la possibilità di molteplici sessione per ogni giornata.

La piattaforma, inoltre, supporta funzionalità interattive come commenti e sezione domande relativa alla sessione in esecuzione in quell'esatto momento, migliorando il coinvolgimento dei partecipanti durante gli eventi.

Infine, da non sottovalutare, è il design user-friendly dell'applicazione che facilita la navigazione e l'utilizzo sia per gli organizzatori che per i partecipanti.

4.2.2 Sfide dello sviluppo

Durante lo sviluppo dell'applicazione si sono presentate alcune sfide stimolanti che hanno reso l'implementazione dell'applicazione un'attività capace di insegnare nuove soluzioni potenzialmente utili per altri progetti.

Una delle sfide principali incontrate durante lo sviluppo è quella relativa alla gestione efficiente di molteplici sessioni riguardanti un singolo evento o convegno. Questa complessità si è manifestata principalmente nel backend, più specificatamente per quel che riguarda l'ottimizzazione delle prestazioni del database e la gestione di query complesse relative a eventi con numerose sessioni. Inoltre, anche l'implementazione frontend delle molteplici sessioni per evento è risultato difficile, soprattutto per le complessità di implementazione all'interno del form di creazione.

4.2.3 Impatto potenziale

L'applicazione sviluppata ha il potenziale per avere un impatto nel settore della gestione dei convegni grazie alla sua capacità di semplificare i processi organizzativi per eventi di varie dimensioni. Grazie anche al miglioramento dell'esperienza dei partecipanti attraverso una piattaforma unificata e interattiva, riducendo, in tal modo, i costi operativi per le organizzazioni che gestiscono molteplici eventi contemporaneamente.

Inoltre, questa piattaforma può adattarsi efficacemente a diversi contesti come quelli maggiormente quotidiani come assemblee di vario genere relegate ad un contesto circoscritto. Oppure ancora in ambito aziendale per la gestione di meeting, percorsi di formazione o di aggiornamento e conferenze interne. Infine, potrebbe trovare applicazione anche nel settore delle fiere ed esposizioni, dove la gestione di molteplici eventi paralleli è molto comune.

Tra le altre cose, la piattaforma sviluppata, potrebbe permettere una maggiore interconnessione tra eventi correlati ma proposti da diverse aziende, favorendo, quindi, la creazione di reti professionali più ampie volte alla collaborazione tra più aziende sviluppando l'economia di ciascuna di esse.

L'applicazione dimostra quindi un potenziale per trasformare l'approccio alla gestione dei convegni, offrendo soluzioni innovative a sfide comuni nel settore. Nonostante le aree di miglioramento siano diverse, i punti di forza dell'applicazione la rendono uno strumento promettente non solo nel suo contesto attuale, ma anche per possibili espansioni in ambiti correlati.

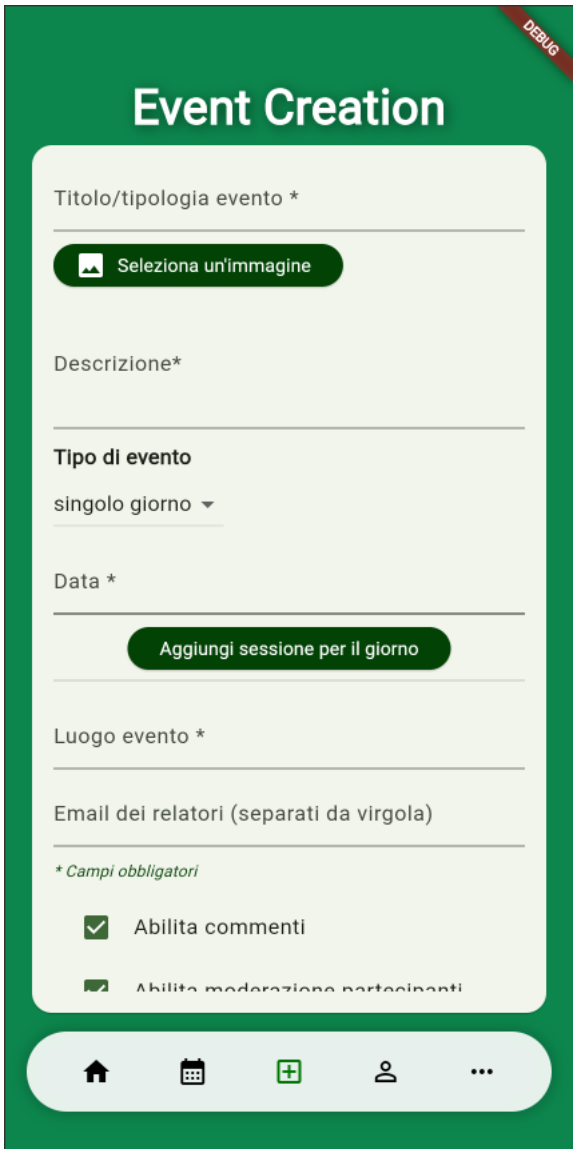

Figura 4.1: Interfaccia di creazione evento

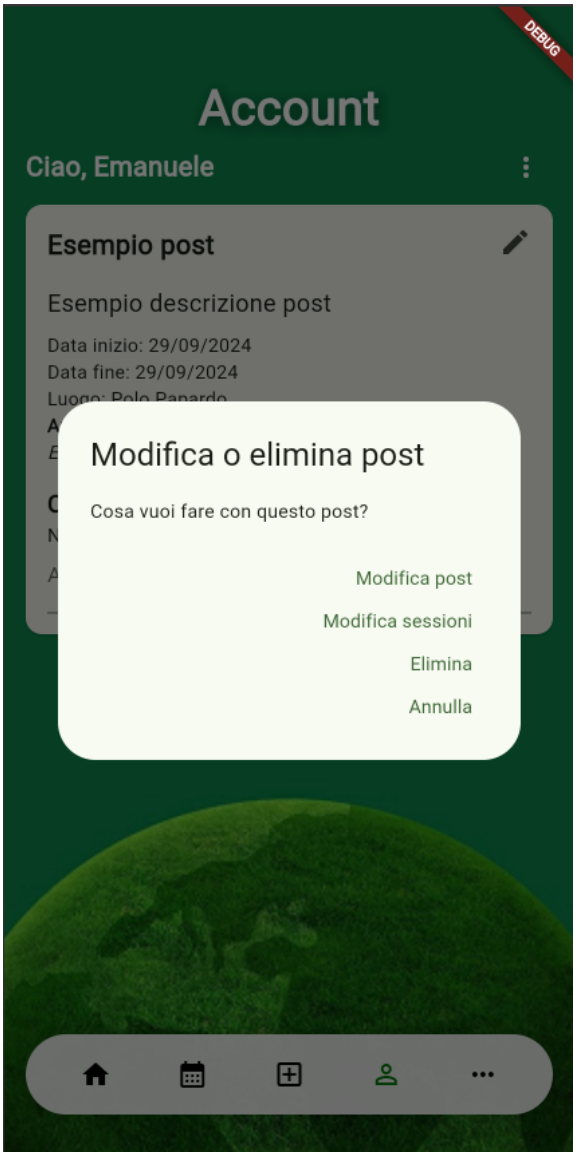

Figura 4.2: Interfaccia di modifica post

Figura 4.3: Interfaccia programma eventi

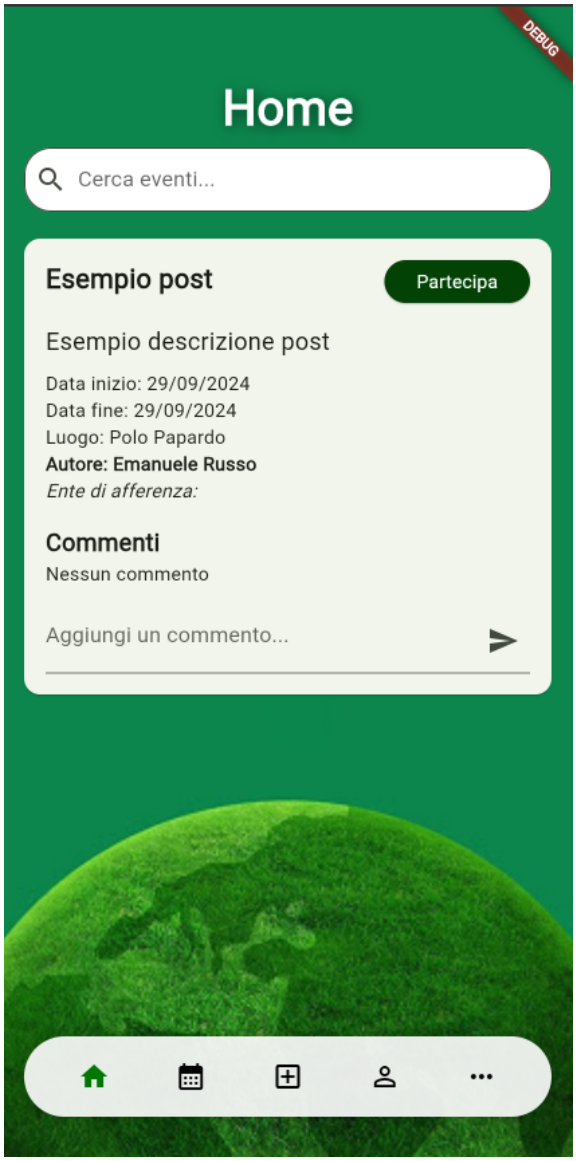

Figura 4.4: Interfaccia di partecipazione all'evento

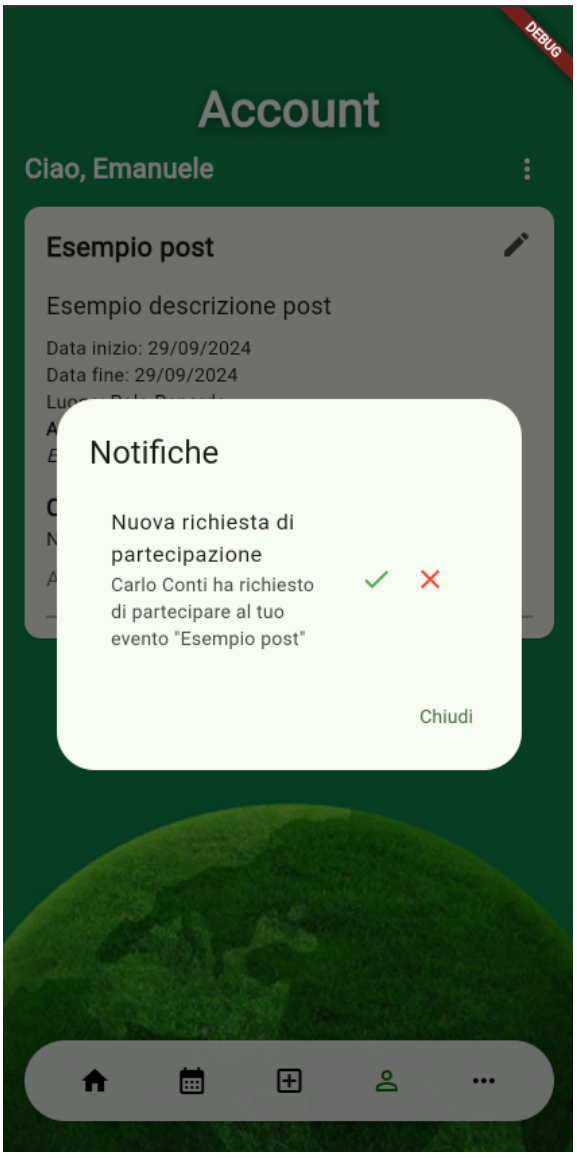

Figura 4.5: Interfaccia notifica di richiesta partecipazione

Capitolo 5

Conclusioni e Sviluppi Futuri

5.1 Principali contributi del progetto

Il progetto di sviluppo dell'applicazione per la gestione di convegni ha il potenziale per portare diversi contributi significativi nel settore.

Uno dei punti di forza principali è la sua capacità di gestire più eventi contemporaneamente all'interno di una singola piattaforma, superando così una delle limitazioni comuni nelle soluzioni esistenti. Questo approccio offre maggiore efficienza e coerenza nella gestione di molteplici convegni contemporaneamente, evitando costi in più per le aziende.

L'applicazione si distingue anche per la sua interfaccia utente intuitiva e flessibile, che facilita la creazione, la gestione e la partecipazione agli eventi, adattandosi a diverse tipologie di convegni e alle esigenze degli organizzatori. Inoltre, il sistema di partecipazione avanzato, con opzioni per la moderazione e l'abilitazione di commenti, migliora il controllo e la sicurezza degli eventi.

Infine, per aumentare il coinvolgimento dei partecipanti, sono state integrate funzionalità interattive come sistemi di commenti, introdotti nel paragrafo precedente, e domande in tempo reale, migliorando così l'esperienza complessiva dei convegni. Inoltre, dal punto di vista tecnico, l'utilizzo di tecnologie moderne come Flutter per il frontend e Node.js per il backend garantisce prestazioni elevate e la possibilità di espandere facilmente le funzionalità in futuro.

5.2 Limitazioni attuali dell'applicazione

Nonostante questi punti di forza, l'applicazione presenta alcune limitazioni che offrono opportunità per futuri miglioramenti.

La principale limitazione è l'assenza di una piattaforma dedicata per la fruizione remota degli eventi. In un'epoca in cui gli eventi ibridi e completamente online sono sempre più comuni, questa lacuna rappresenta un'area critica per lo sviluppo futuro.

Altre limitazioni includono la mancanza di un vero e proprio sistema di comunicazione in tempo reale (come WebSocket), opzioni limitate per la personalizzazione dell'interfaccia utente, che potrebbe non soddisfare pienamente le esigenze di branding e personalizzazione di alcuni organizzatori, funzionalità di networking tra partecipanti non molto sviluppate e l'assenza di integrazione con sistemi esterni come CRM o strumenti di marketing.

5.3 Future implementazioni

Guardando al futuro, ci sono diverse aree in cui l'applicazione potrebbe essere migliorata.

Una priorità potrebbe essere l'implementazione di una piattaforma per eventi virtuali e ibridi, che includa funzionalità per lo streaming live delle sessioni, spazi virtuali per il networking e strumenti per la gestione di Q&A e sondaggi in tempo reale.

L'implementazione di WebSocket, come anticipato, potrebbe migliorare significativamente la comunicazione in tempo reale, aumentando la reattività dell'applicazione per funzionalità come chat live e aggiornamenti istantanei del programma.

Possibilità di creazione e gestione di eventi a pagamento, in cui la partecipazione può essere effettuata tramite pagamenti in app. L'utente creatore dell'evento potrebbe tenere traccia di coloro che effettuano il pagamento per partecipare all'evento e il guadagno totale.

Altre aree di miglioramento potrebbero includere una maggiore personalizzazione dell'interfaccia utente, funzionalità avanzate di networking come un sistema di matchmaking basato su interessi e profili professionali, supporto multilingua e localizzazione, e miglioramenti nella sicurezza e nella privacy.

In conclusione, mentre l'applicazione sviluppata offre già notevoli vantaggi nella gestione di convegni multipli, ci sono numerose opportunità per espandere e migliorare le sue funzionalità. L'implementazione di questi miglioramenti potrebbe posizionare l'applicazione come una soluzione di prim'ordine nel settore, capace di soddisfare le esigenze in continua evoluzione di organizzatori e partecipanti.

Bibliografia

[1] Ahmedin Mohammed Ahmed et al. «Event-Based Mobile Social Networks: Services, Technologies, and Applications». In: *IEEE Access* 2 (2014), pp. 500–513. DOI: 10.1109/ACCESS.2014.2319823.

[2] Alexandra Chalkia e Athina Papageorgiou. «The Management of Conferences and Business Events in Periods of Crisis. The New Digital Paradigm». In: *Transcending Borders in Tourism Through Innovation and Cultural Heritage*. A cura di Vicky Katsoni e Andreea Claudia Şerban. Cham: Springer International Publishing, 2022, pp. 463–473. ISBN: 978-3-030-92491-1.

Sitografia

[3] *Flutter*. URL: https://flutter.dev/.

[4] *What is a REST API?* URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[5] *Fourwaves - Conference Management Software*. URL: https://www.fourwaves.com.

[6] *Ex Ordo - Abstract Management Software*. URL: https://www.exordo.com.

[7] *Cvent - Event Management Software*. URL: https://www.cvent.com.

[8] *ACID*. URL: https://www.databricks.com/it/glossary/acid-transactions#:~:text=Propriet%C3%A0%20A.C.I.D.%3A%20Atomicity%2C%20Consistency%2C,Consistency%2C%20Isolation%2C%20e%20Durability.

[9] *XAMPP*. URL: https://www.apachefriends.org/.