UniReserve

Progetto di Piattaforme Software per Applicazioni su Web

Vincenzo Calogero e Francesco Giacobbe
mat. 250455, 250469

A.A. 2025/2026

1 Introduzione e contesto dell'applicazione

UniReserve è una piattaforma web per la gestione delle prenotazioni di posti in aula nelle lezioni universitarie. Il sistema è concepito per due tipologie di utenti: i *professori*, che pianificano le lezioni assegnando un'aula e delle fasce orarie, e gli *studenti*, che possono consultare il calendario delle lezioni e prenotare un posto specifico.

Il contesto applicativo è prettamente didattico, ma il modello adottato rispecchia requisiti reali di concorrenza e integrità dei dati: più studenti possono tentare di prenotare lo stesso posto contemporaneamente, e un professore può assegnare un'aula già parzialmente o totalmente occupata da un'altra lezione. Queste criticità sono gestite sia tramite meccanismi di locking a livello di backend, e sia tramite vincoli di integrità a livello di database.

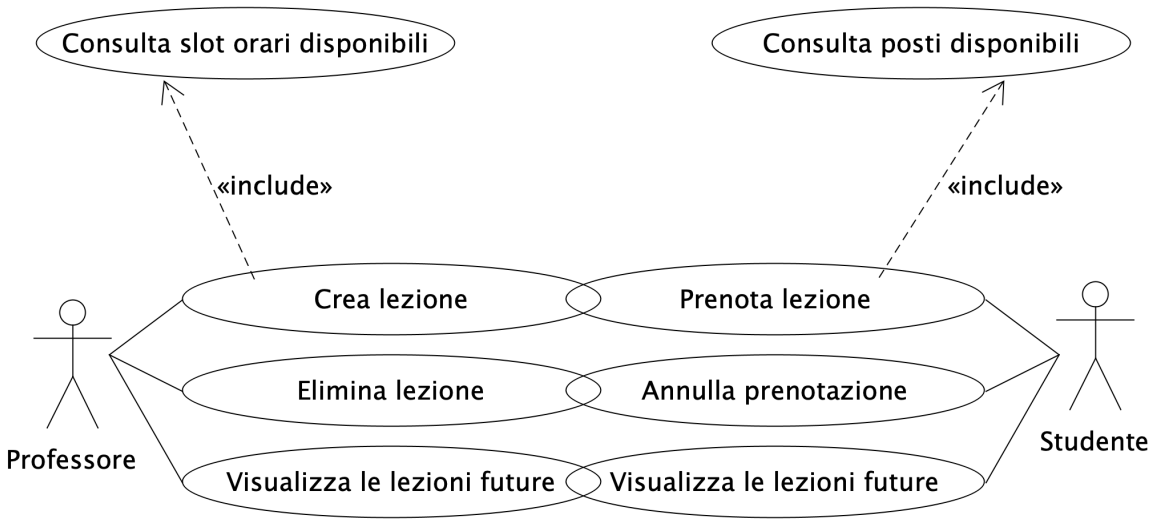


Figura 1: Use case diagram

Trattandosi di un progetto di coppia, l'approccio alla progettazione in toto (full-stack) si è suddiviso come segue:

- **Francesco Giacobbe** si è occupato della progettazione dei casi d'uso dell'attore *Professore*;
- **Vincenzo Calogero** si è occupato della progettazione dei casi d'uso dell'attore *Studente*.

2 Analisi dei requisiti

Requisiti funzionali

- **Gestione lezioni (Professore):** creazione di una lezione con materia, descrizione, data, aula e fasce orarie; eliminazione con cascata su slot e prenotazioni.
- **Prenotazione posti (Studente):** visualizzazione delle lezioni future (filtrabili per dipartimento), scelta del posto disponibile, annullamento della prenotazione.

- **Disponibilità in tempo reale:** i posti occupati e gli slot orari già assegnati sono calcolati dinamicamente a partire dallo stato del database.

Requisiti non funzionali

- **Sicurezza e autenticazione:** l'identità degli utenti è gestita tramite *Keycloak*, un Identity Provider open-source conforme a OAuth 2.0 e OpenID Connect. Il frontend ottiene un JWT firmato dal realm `unireserve` e il backend ne verifica la firma. Il traffico tra client, backend e Keycloak è cifrato con TLS (Let's Encrypt tramite Nginx). L'applicazione così com'è non prevede che l'utente si possa registrare autonomamente; scelta giustificata pensando a una gestione unificata delle utenze da parte della singola università che adotta questa applicazione.
- **Autorizzazione basata su ruoli:** i ruoli `PROFESSOR` e `STUDENT` sono estratti dal claim `resource_access.UniReserveBE.roles` del token JWT e mappati ad authority Spring. Ogni endpoint è protetto con `@PreAuthorize` a seconda del ruolo.
- **Internazionalizzazione:** l'interfaccia (sia frontend che Keycloak) supporta italiano e inglese.

Integrità concorrente lato Student

La prenotazione di un posto usa locking pessimistico (`PESSIMISTIC_FORCE_INCREMENT`, con `@Version` per debugging) sul `Seat`. La scelta del lock pessimistico è giustificata dall'alta probabilità che più studenti possano tentare di prenotare lo stesso posto contemporaneamente. Il lock serializza gli accessi, garantendo che un solo studente alla volta proceda con la prenotazione. Il campo `@Version` sul `Seat` non viene usato come per i lock ottimistici ma è stato aggiunto per rendere visibile a runtime quante transazioni abbiano conteso lo stesso record. Come ulteriore garanzia, il database impone un vincolo di unicità su `(lesson_id, seat_id)`, che funge da ultima linea di difesa anche nel caso in cui il locking a livello applicativo fallisse.

Integrità concorrente lato Professor

In aggiunta ai vincoli di integrità del database che bloccano l'operazione di prenotazione della stessa aula in slot orari sovrapposti, si è implementato un ulteriore meccanismo che rafforza questa *thread-safety*. La creazione di una lezione usa locking ottimistico (`OPTIMISTIC_FORCE_INCREMENT` con `@Version`) sulla `Classroom`. La scelta del lock è giustificata se si assume che sia raro che due professori vadano a prenotare la stessa aula nello stesso momento, ma è stato necessario forzare l'aumento di versione in quanto durante la prenotazione, l'oggetto *Classroom* non viene modificato. Quando un professore richiede i timeslot disponibili per una certa aula, il backend restituisce anche la versione corrente dell'oggetto *Classroom* in modo da poterla confrontare quando viene premuto il tasto di conferma di prenotazione. Se le versioni coincidono la prenotazione va a buon fine, altrimenti viene chiesto di riprovare.

DevOps

L'applicazione in toto è stata deployata su una **VPS** Ubuntu tramite *Oracle Cloud*. Su questo sono in esecuzione tutti i servizi necessari all'applicazione, grazie anche allo SWAP

configurato in modo da aggirare le grosse limitazioni hardware della nostra VPS (1 GB di RAM):

- **MySQL** in ascolto su `unireserver.giacobbe.ch:3306`,
- **Keycloak** in ascolto su `auth.unireserver.giacobbe.ch` e deployato tramite **Docker**,
- **UniReserveBE** Spring Boot application in ascolto su `unireserver.giacobbe.ch`,
- **UniReserveFE** Angular web server in ascolto su `unireserve.giacobbe.ch`.

Il **deploy** avviene in automatico ogni qual volta viene effettuata una `push` sul branch `main`, sia lato frontend che backend, grazie alle *GitHub Actions*. Inoltre, tutte le richieste vengono effettuate tramite protocollo **HTTPS** grazie a un certificato TLS rilasciato da *Let's Encrypt* e richiesto tramite *Nginx*.

3 Back-end e tecnologie impiegate

Stack tecnologico

Il backend è realizzato con **Spring Boot 4** e si appoggia a un database **MySQL** ospitato su `unireserver.giacobbe.ch`. La persistenza è gestita tramite **Spring Data JPA** (Hibernate). La sicurezza è affidata a **Spring Security** con un `JwtAuthenticationConverter` custom che legge i ruoli dal JWT Keycloak.

Architettura a livelli

L'applicazione segue una classica struttura a tre livelli:

1. **Controller REST** (`controllers/rest/`) — punto d'ingresso HTTP; delega tutta la logica ai servizi, mappa i codici di risposta in funzione delle eccezioni lanciate dai services sottostanti.
2. **Service layer** (`services/`) — logica di business, locking, validazioni.
3. **Repository layer** (`repositories/`) — accesso al DB via interfacce Spring Data JPA.

Modello dei dati

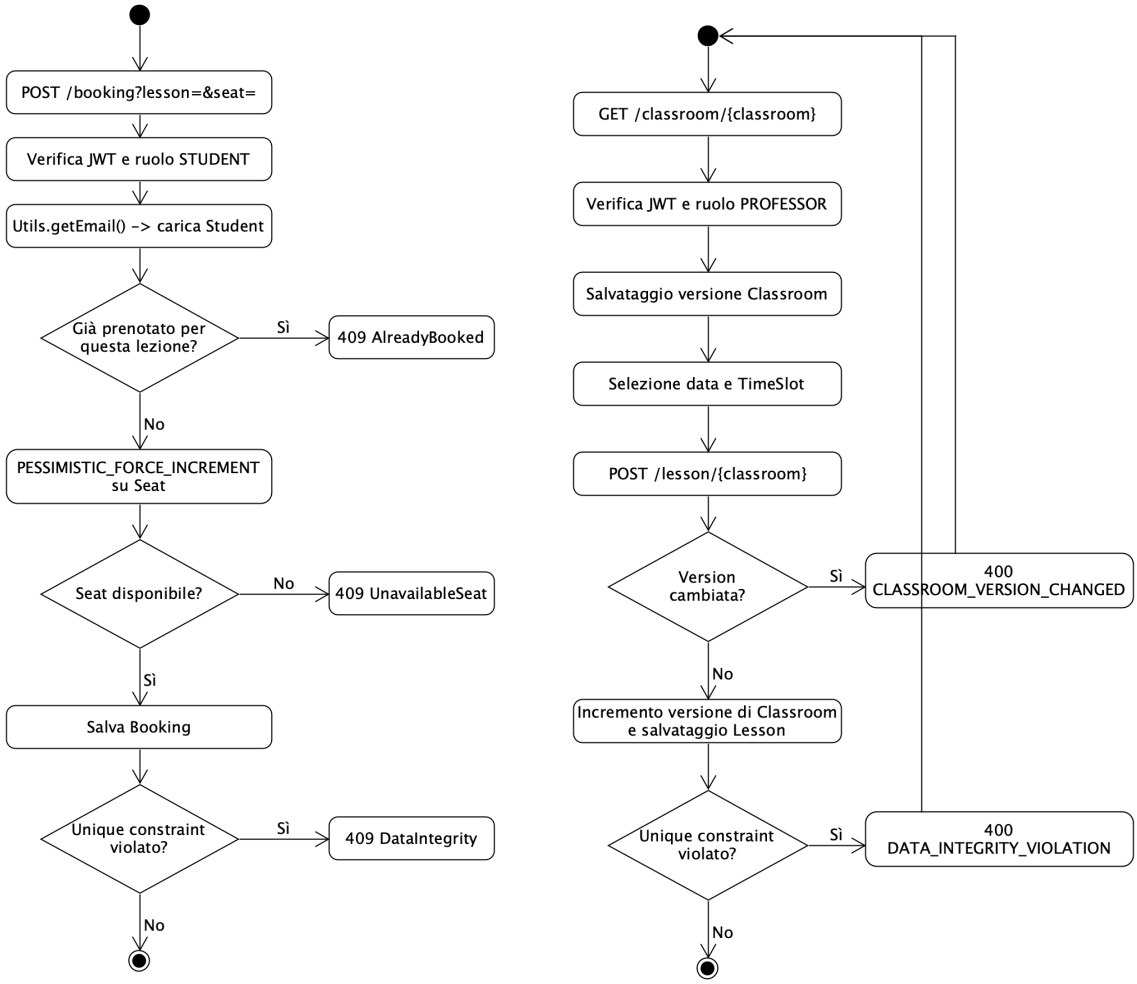
Le entità principali sono: `Department`, `Professor`, `Student`, `Classroom` (con `@Version` per l'optimistic lock), `Seat` (con `@Version` per il pessimistic lock), `Slot` (aula + data + `TimeSlotEnum`, chiave unica), `Lesson` e `Booking` (chiave unica `lesson+seat`). `User` usa ereditarietà JOINED, quindi esiste una sola tabella con le colonne specifiche per le sottoclassi, che sono `Professor` e `Student`. L'`id` di `User` coincide con il `sub` Keycloak, eliminando la necessità di un mapping separato identità $\leftrightarrow$ utente.

Endpoint REST

Controller	Metodo	Path	Ruolo
BookingController	GET	`/bookings`	STUDENT
	GET	`/bookings/available-seats`	STUDENT/PROF
	POST	`/bookings`	STUDENT
	DELETE	`/bookings/{id}`	STUDENT
LessonController	GET	`/lesson`	STUDENT/PROF
	GET	`/lesson/future`	PROFESSOR
	POST	`/lesson/{classroom}`	PROFESSOR
	DELETE	`/lesson/{id}`	PROFESSOR
ClassroomController	GET	`/classroom`	PROFESSOR
	GET	`/classroom/{id}?date=`	PROFESSOR
DepartmentController	GET	`/departments`	STUDENT

Tabella 1: Endpoint REST esposti dal backend

Flusso transazionale



(a) Prenotazione posto studente (lock pessimistico)

(b) Prenotazione aula professore (lock ottimistico)

Figura 2: Activity diagram

4 Front-end e tecnologie impiegate

Stack tecnologico

Il frontend è sviluppato con **Angular 21** in modalità *standalone components*, senza `NgModule` come per convenzione delle nuove versioni. L'interfaccia grafica si basa su **Angular Material** per i componenti UI (card, dialog, button, select). L'autenticazione è gestita dalla libreria **keycloak-js** integrata direttamente tramite il servizio `AuthService`. L'internazionalizzazione usa **@ngx-translate/core** con file JSON separati per italiano e inglese. Il deploy del server web avviene tramite **nginx** su `unireserve.giacobbe.ch`.

Architettura

L'app è organizzata in:

- `core/` — modelli, servizi HTTP, auth guard, auth interceptor, utilities.
- `features/professor/` — Dashboard professore, form creazione lezione.
- `features/student/` — Dashboard prenotazioni, lista lezioni, dettaglio lezione.
- `shared/` — componenti riutilizzabili (es. `ConfirmDialogComponent`).

Flusso di autenticazione

All'avvio, `app.config.ts` inizializza Keycloak in modalità `check-sso` (nessun redirect forzato). Le route sotto `/professor` e `/student` sono protette dall'`authGuard`, che verifica `isLoggedIn()` e il ruolo corretto; in caso contrario reindirizza al login Keycloak. L'`authInterceptor` intercetta ogni chiamata HTTPS, rinnova il token se in scadenza (finestra 30 s) e aggiunge `Authorization: Bearer <token>`.

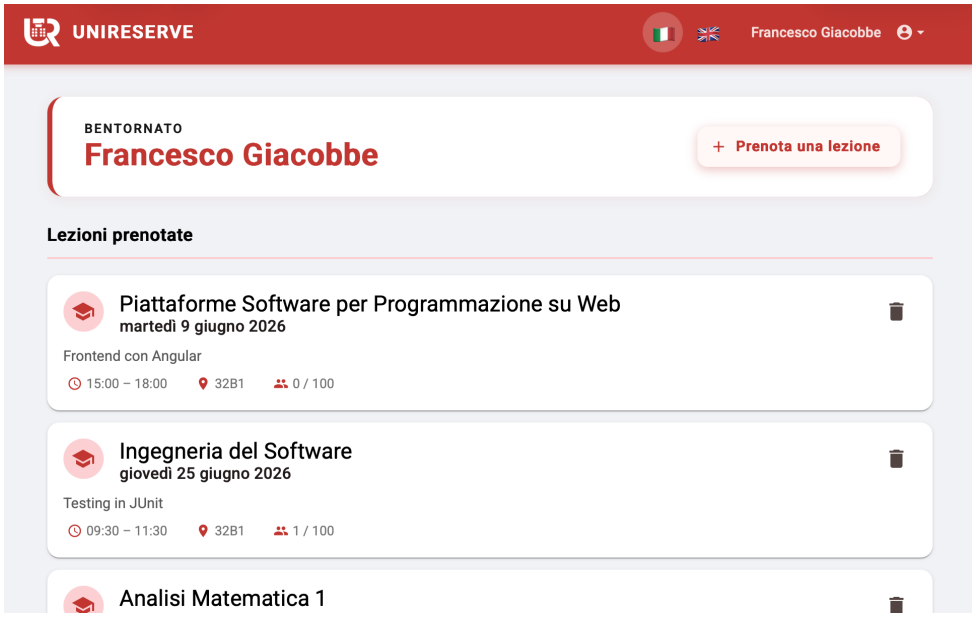


Figura 3: Dashboard professore

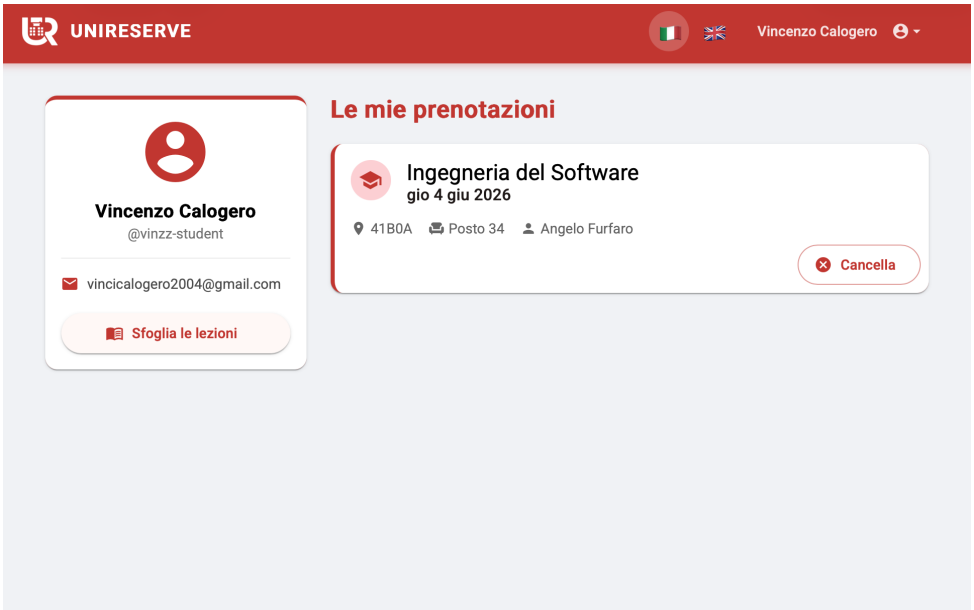


Figura 4: Dashboard studente

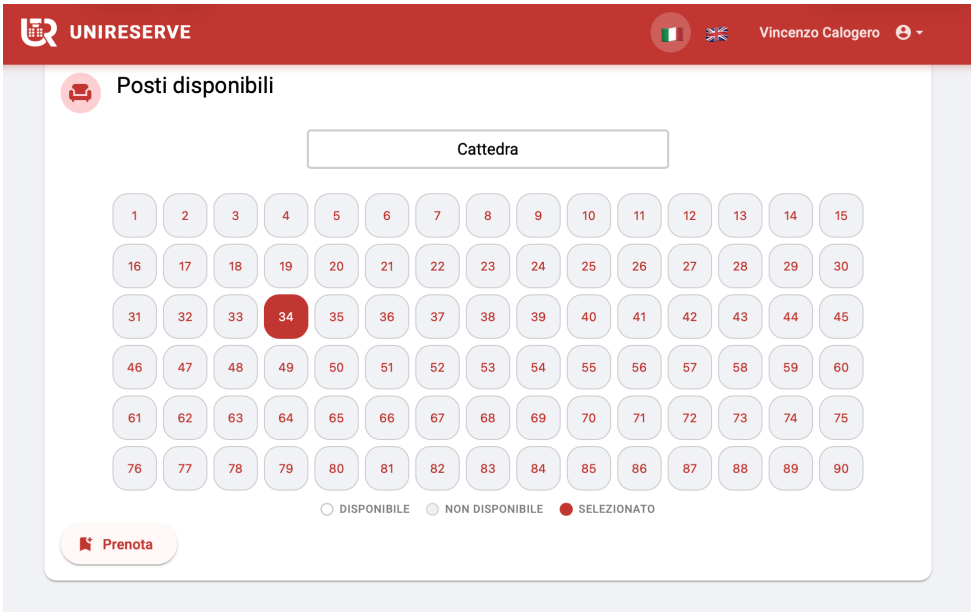


Figura 5: Prenotazione posto

5 Conclusioni

UniReserve costituisce un sistema web completo che integra tecnologie moderne sia lato server che lato client. L'adozione di Spring Boot 4 garantisce un backend robusto e tipizzato, mentre Angular 21 offre un'interfaccia reattiva e modulare. La delega dell'identità a Keycloak semplifica la gestione degli utenti e abilita single-sign-on senza implementare autenticazione proprietaria. I meccanismi di locking (pessimistico per la prenotazione dei posti, ottimistico per l'assegnazione dell'aula) affrontano concretamente i problemi di concorrenza tipici di un sistema di prenotazione reale. La cascata JPA su `Lesson→Slot` e `Lesson→Booking` mantiene la coerenza dei dati in modo dichiarativo.