

# Library Seat Reservation

## SOFTWARE ARCHITECTURES AND METHODOLOGIES

### Studenti

- Mirco Ceccarelli (7042018)
- Edoardo D'Angelis (7027290)
- Pietro Zarri (7042627)

### Docente

Prof. Enrico Vicario

### Supervisori

- Dott. Boris Brizzi
- Ing. Jacopo Parri
- Ing. Samuele Sampietro



## INTRODUZIONE

**LibrarySeatReservation** è una applicazione web per la gestione delle prenotazioni di posti all'interno delle aule studio delle biblioteche di Firenze.

Gli **obiettivi funzionali** di maggiore rilievo sono:

- Progettazione e sviluppo di un **backend** che implementa le API di tipo RESTful e utilizza database Postgres
- Progettazione e sviluppo di un **gateway** che implementa un sistema di gestione “a code” delle richieste pervenute
- Progettazione e sviluppo di due **frontend** (uno per l’utente semplice e uno per l’admin)



## INTRODUZIONE

Gli **obiettivi di sperimentazione tecnologica** di maggiore rilievo sono:

- **TimescaleDB**: estensione di **PostgreSQL** in grado di offrire un insieme di operazioni relative a dati temporali, per memorizzare le prenotazioni
- **WebSocket**: protocollo di comunicazione di **basso livello** basato su Transmission Control Protocol (TCP), per la gestione della coda
- **RSocket**: protocollo di comunicazione a **livello applicativo** bidirezionale, multiplex e duplex, per l'implementazione di un sistema di notifiche *real-time*
- **JWT** per la gestione dell'autenticazione e dell'autorizzazione
- **Docker** per l'isolamento e la divisione in container (Wildfly, database Postgres)



## GESTIONE DELLA CODA (1/2)

Quando si rende necessario l'utilizzo di una **coda**?

- Quando le **risorse** sono **limitate**
- Quando il **traffico web** **aumenta** improvvisamente oltre le capacità di gestione del sistema



Gli scenari **funzionali** possibili sono i seguenti:

- Una **coda singola** per l'accesso ad ogni biblioteca
- Una **coda generale** per l'accesso al sistema di prenotazione

È stata scelta la **seconda opzione**, con l'intento di prevenire sovraccarichi al sistema dovuto ad un grande numero di richieste contemporanee relative a biblioteche diverse.

## GESTIONE DELLA CODA (2/2)

Gli scenari **architetturali** possibili sono i seguenti:

- **Modulo frontend come parte attiva** del funzionamento della coda (es. informato dal gateway reagisce con un redirect)
- **Modulo frontend completamente indipendente** dal modulo gateway (es. il gateway ha un suo frontend specifico per la pagina della coda)



È stata scelta la **prima opzione**, che fornisce maggiore libertà di scelta e consente più facilmente l'aggiunta di funzionalità aggiuntive (es. timer per completare il caso d'uso).

## TIMESTACLEDDB

Database relazionale open-source **Full SQL**. È una estensione di **PostgreSQL** in grado di offrire un insieme di operazioni efficienti relative a dati temporali.

- Basato sul concetto di **hypertable**
- Supporto **nativo** al linguaggio SQL
- Maggiore facilità d'uso per l'analisi delle **time-series**
- Prestazioni **migliorate** per query temporali

In particolare la funzione **time\_bucket()** viene utilizzata per aggregare periodi di tempo di dimensioni arbitrarie (es. 5 minuti, 1 giorno).



```
SELECT time_bucket('1 day', datetime) AS date, count(*)  
FROM reservations  
GROUP BY date  
ORDER BY date ASC
```

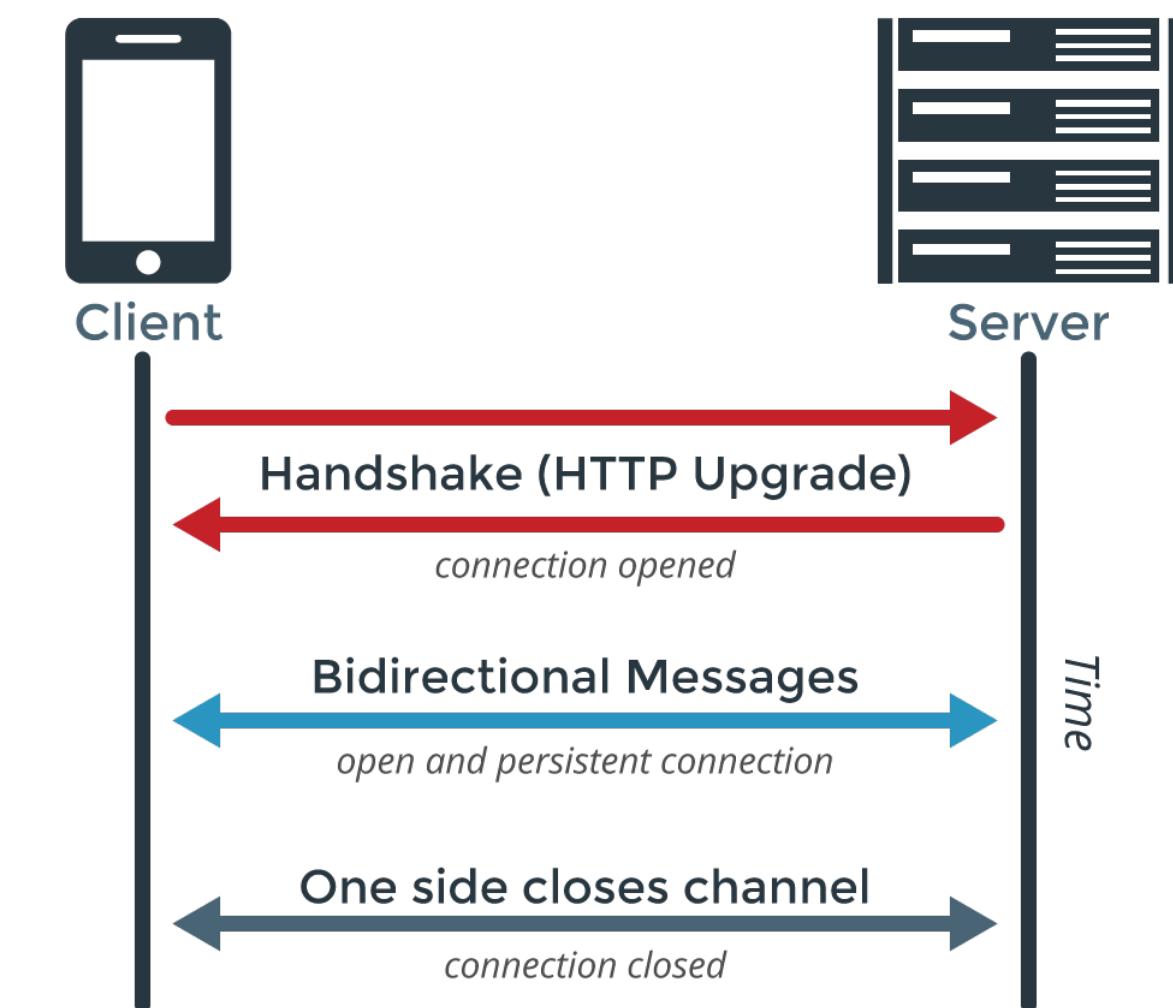
Esempio di query SQL con `time_bucket`

# WEBSOCKET

Protocollo di comunicazione **bidirezionale** in grado di inviare dati da un server ad un client riutilizzando lo stesso **canale di connessione**

Vantaggi rispetto ad HTTP:

- Efficienza
- Riduzione di banda e di latenza
- Semplificazione per le architetture *real-time*



```
import {webSocket} from "rxjs/webSocket";
socket = webSocket("ws://localhost:8080/gateway/queue");
socket.subscribe(message => {
  console.log(message)
})
```

Esempio connessione a endpoint WebSocket



# RSocket

Caratteristiche:

- Protocollo a livello **applicativo**
- Supporto a **feature avanzate** (*framing, session resumption e backpressure non bloccante*)
- **Agnostico** rispetto al livello di trasporto utilizzato (TCP, WebSocket, HTTP/2 ecc...)
- Controllo del **flusso** e riduzione della **latenza**

```
import RSocketClient from "rsocket-core";
import RSocketWebsocketClient from "rsocket-websocket-client";

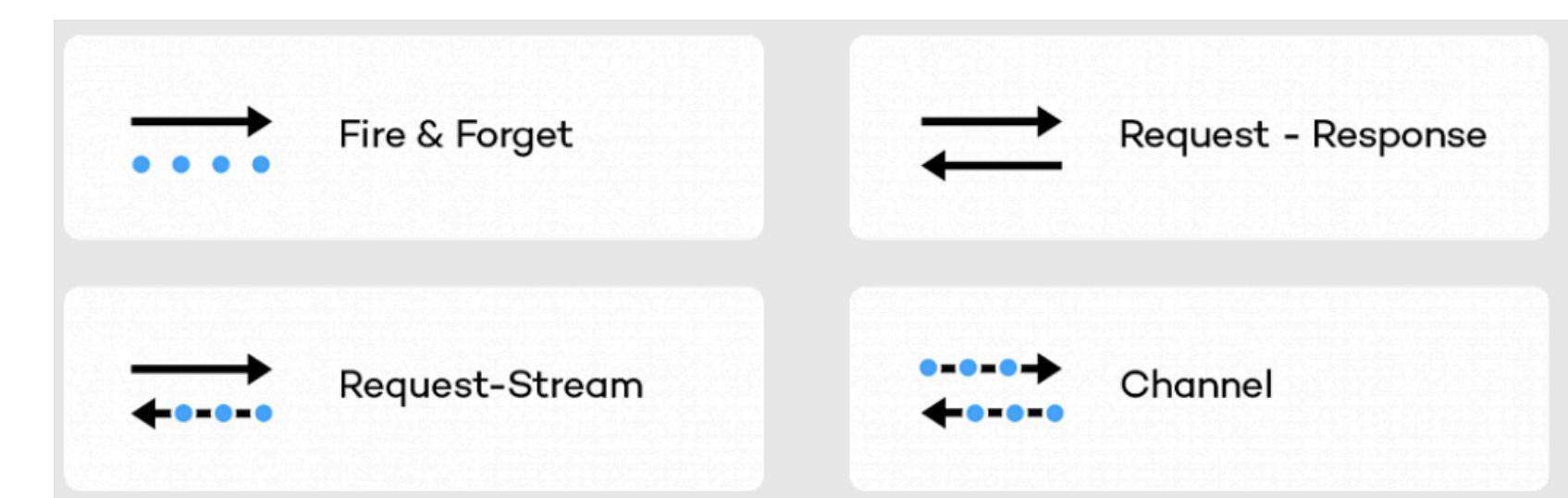
const transport = new RSocketWebsocketClient(
  { url: "ws://localhost:7878" });

client = new RSocketClient(
  { transport: transport, responder: new RSocketResponder() });
client.connect().subscribe({
  onComplete: (rsocket) => {
    rsocket.fireAndForget({data: 'connected!'});
  }
})
```

Esempio di client RSocket

Supporta le seguenti interazioni:

- **Fire-and-forget**
- **Request-Stream**
- **Request-Response**
- **Channel**

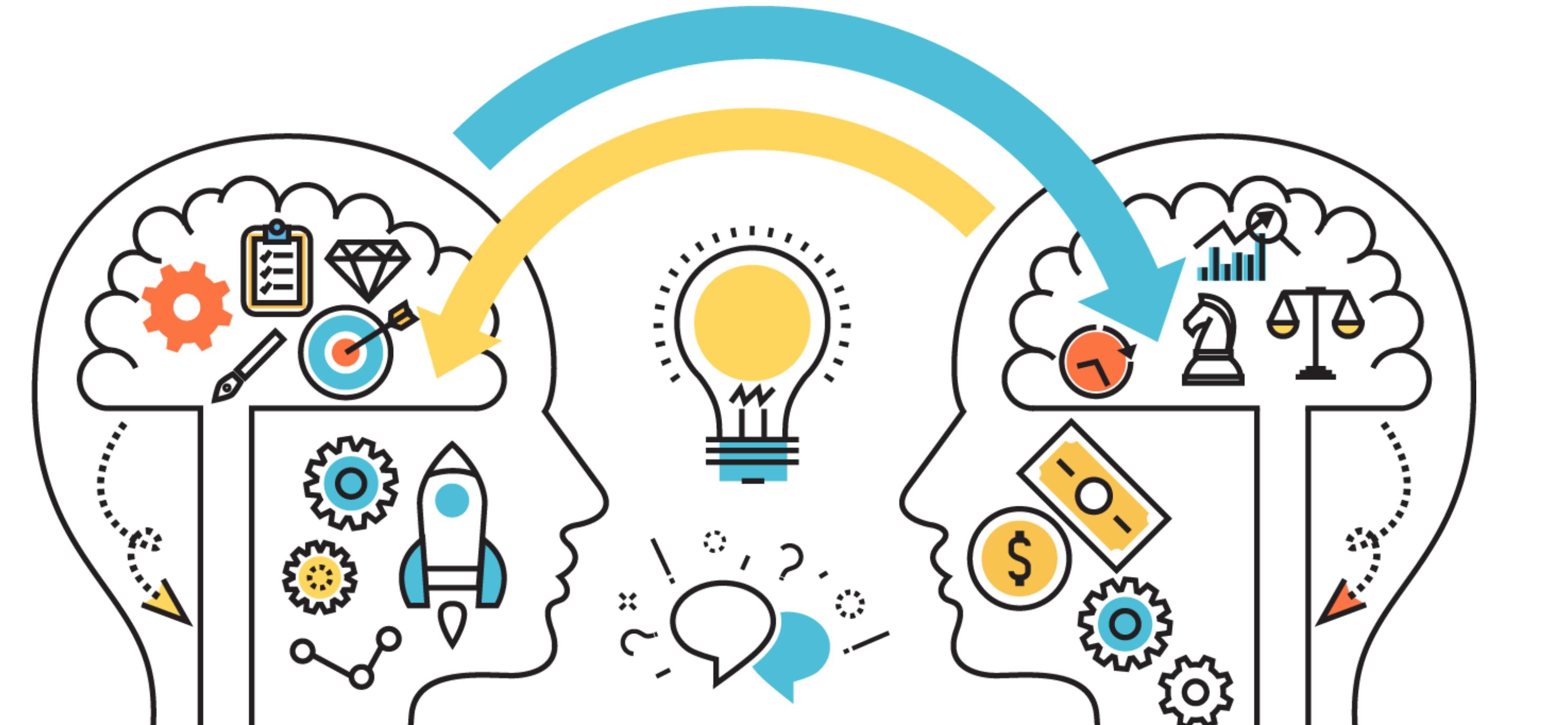


Modelli di interazione RSocket



## FASI DEL PROGETTO

- **Documentazione**
  - Analisi dei requisiti
  - Diagrammi dei casi d'uso
  - Mockup
  - Modelli di dominio
  - Diagrammi di sequenza
- **Implementazione**
  - Frontend
  - Gateway
  - Backend



# ANALISI DEI REQUISITI

## REQUISITI FUNZIONALI

- Gestione degli utenti
- Gestione delle biblioteche
- Gestione delle prenotazioni

## REQUISITI NON FUNZIONALI

- Il sistema dovrà avere una **natura distribuita**
- **Architettura RESTful**
- Logica **gateway** indipendente dal contesto applicativo

## REQUISITI DI DOMINIO

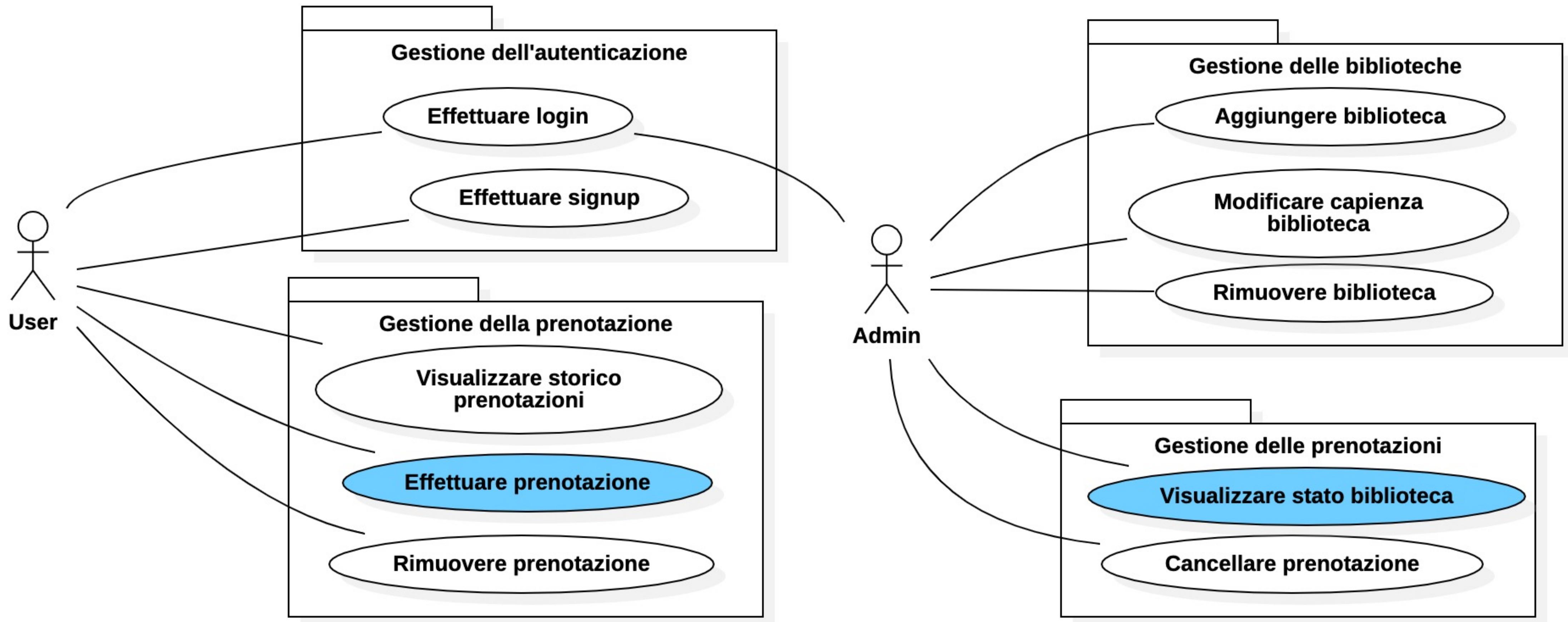
Utente
email
nome
cognome
password

Biblioteca
nome
indirizzo
capienza

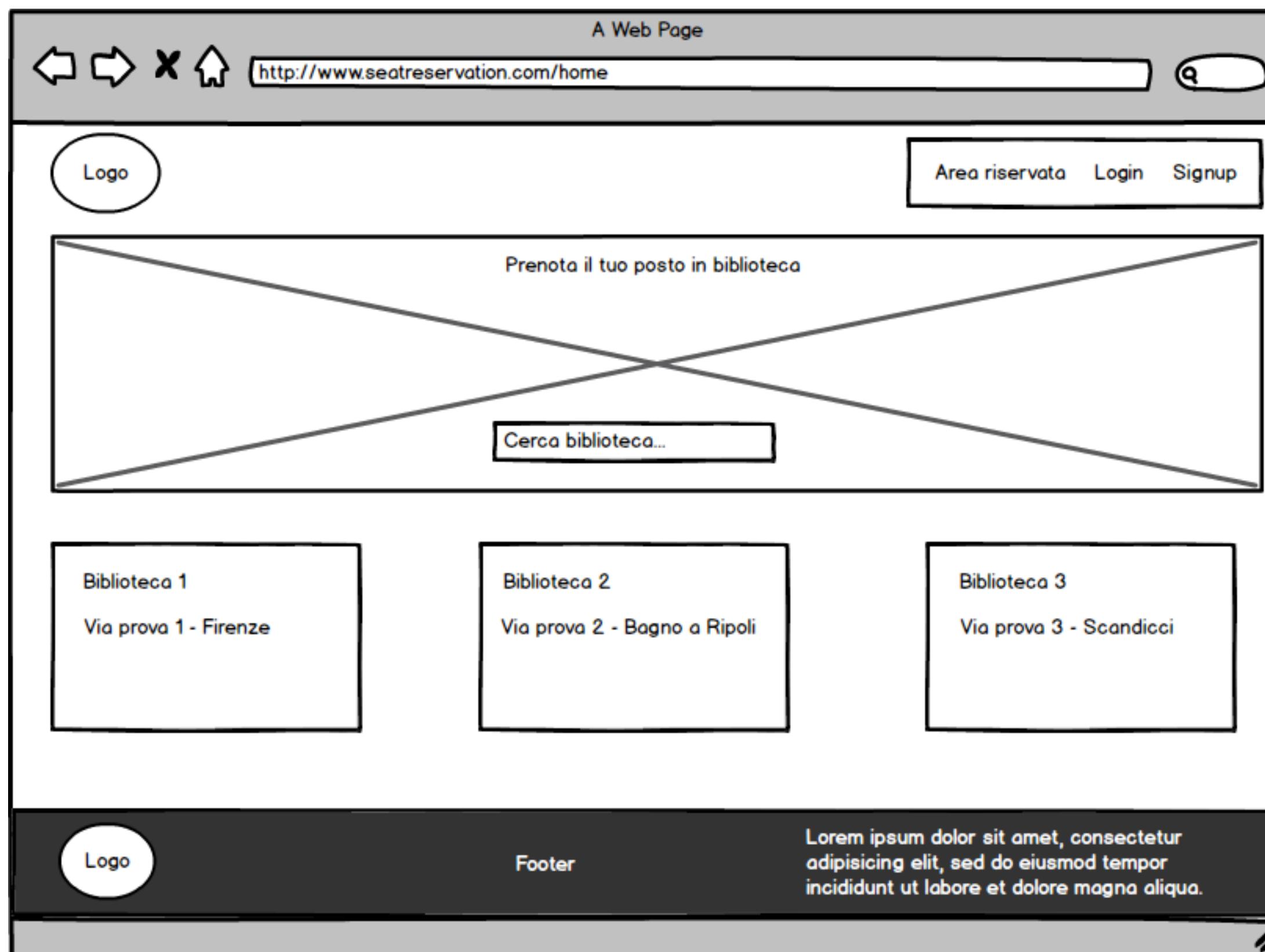
Prenotazione
utente
biblioteca
data
fascia oraria



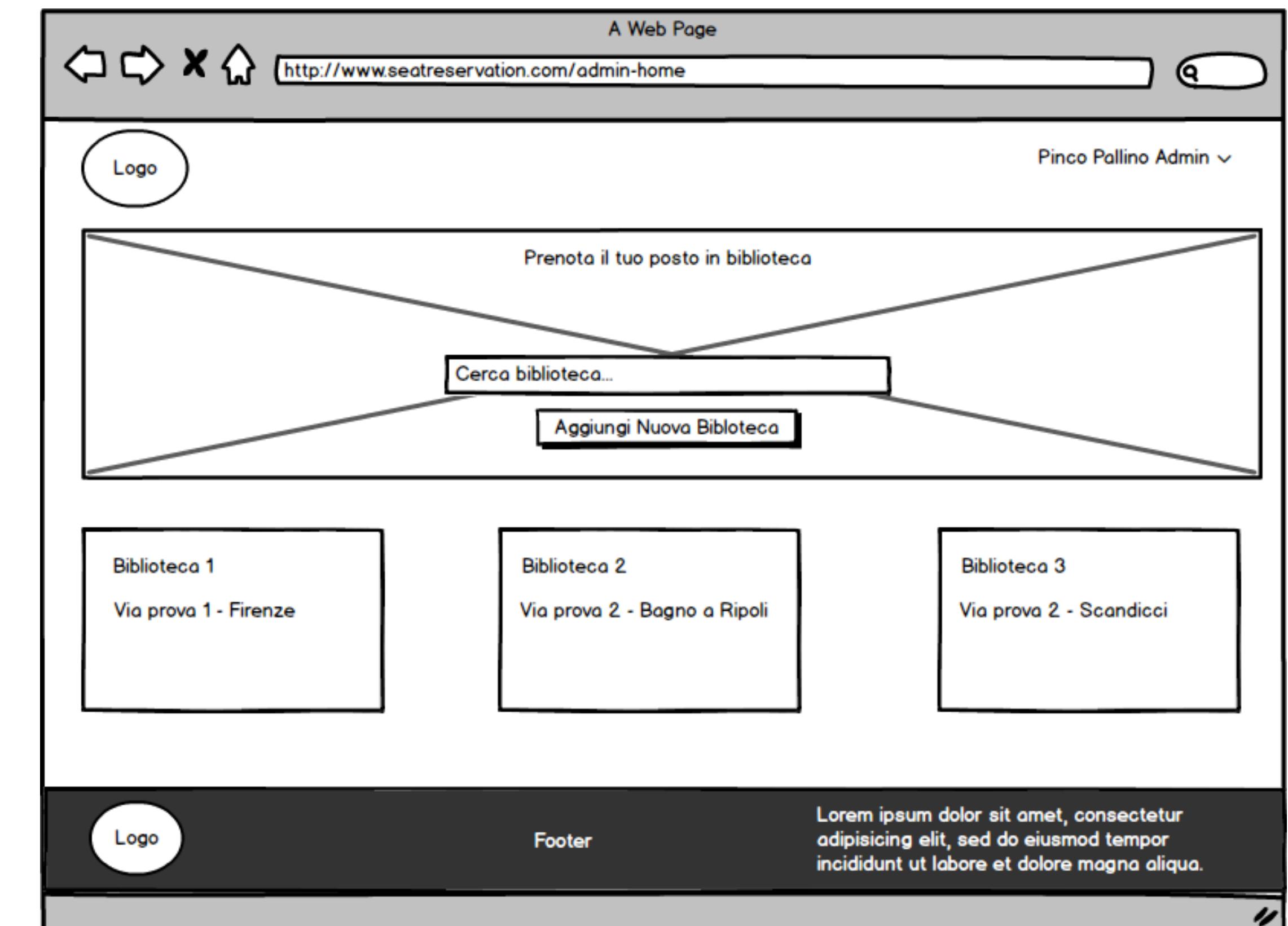
# ANALISI DEI REQUISITI - CASI D'USO



## MOCKUP (1/3)



Mockup home



Mockup home (admin)



## MOCKUP (2/3)

A Web Page  
http://www.seatreservation.com/prenotazione

The page displays a logo, a map showing the location of 'Biblioteca 1' at 'Via prova 1 - Firenze', and a calendar for July 2021. The calendar shows availability for two time slots: 8:00 - 13:00 and 13:00 - 19:00. A 'Prenota' button is present. The footer contains a logo and a placeholder text.

S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Biblioteca 1  
Via prova 1 - Firenze

Area riservata Login Signup

JULY 2021

8:00 - 13:00 12 / 50

13:00 - 19:00 50 / 50

Prenota

Footer

Logo

Lore ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Mockup pagina biblioteca

A Web Page  
http://www.seatreservation.com/admin/prenotazione

The page displays a logo, a map showing the location of 'Biblioteca 1' at 'Via prova 1 - Firenze', and a calendar for July 2021. The calendar shows availability for two time slots: 8:00 - 13:00 and 13:00 - 19:00. A table lists reserved seats for four users. An 'Elimina Biblioteca' button is present. The footer contains a logo and a placeholder text.

S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Biblioteca 1  
Via prova 1 - Firenze

Pinco Pallino Admin ▾

Capienza 60 posti

8:00 - 13:00 13:00 - 19:00

Occupazione 16/60

Pippo Pluto	<input type="button" value="Cancella"/>
Paperino Quo	<input type="button" value="Cancella"/>
Samuele Ceccherini	<input type="button" value="Cancella"/>
Topolino Qua	<input type="button" value="Cancella"/>

Elimina Biblioteca

Footer

Logo

Lore ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Mockup pagina biblioteca (admin)

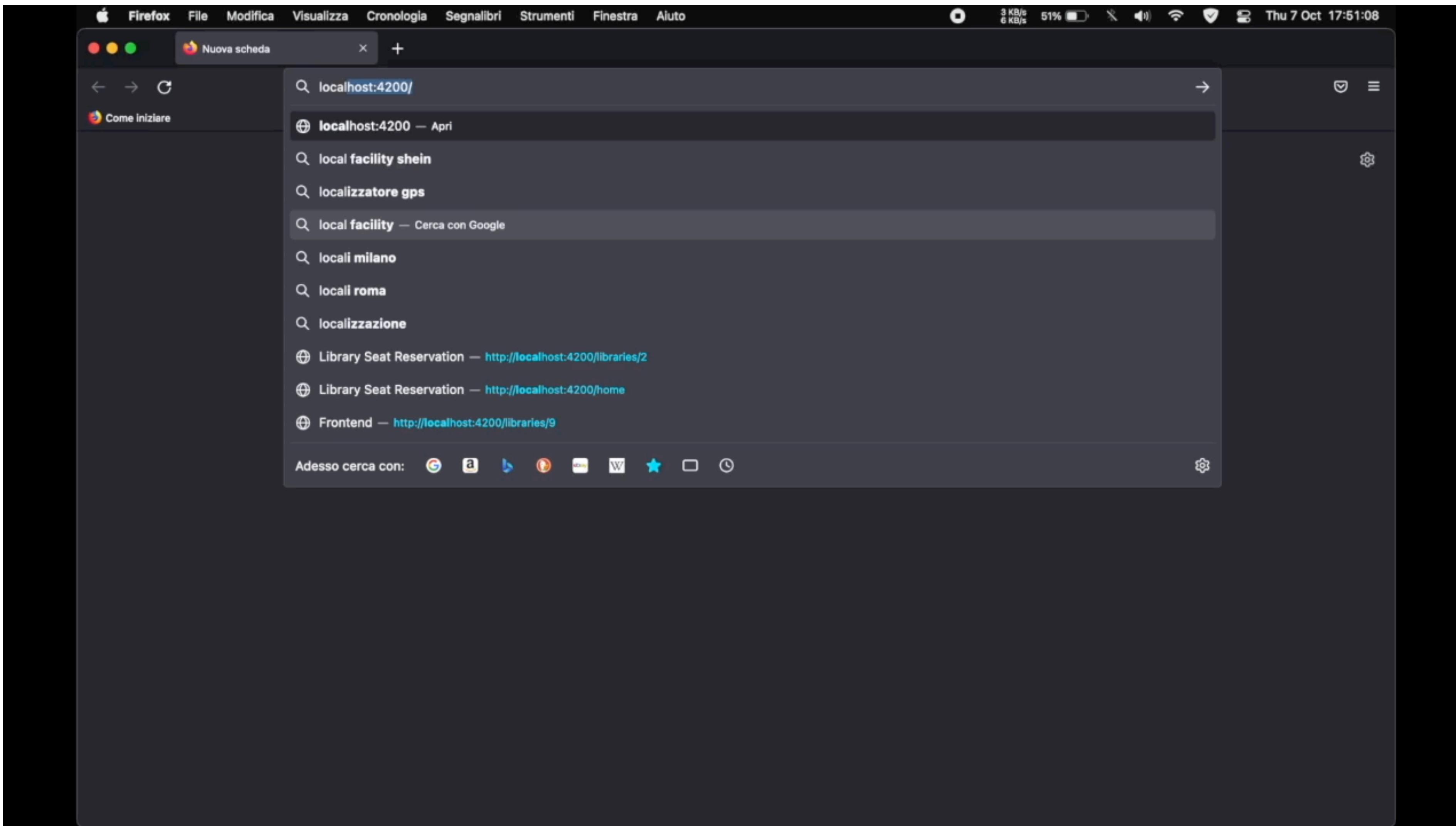


## MOCKUP (3/3)

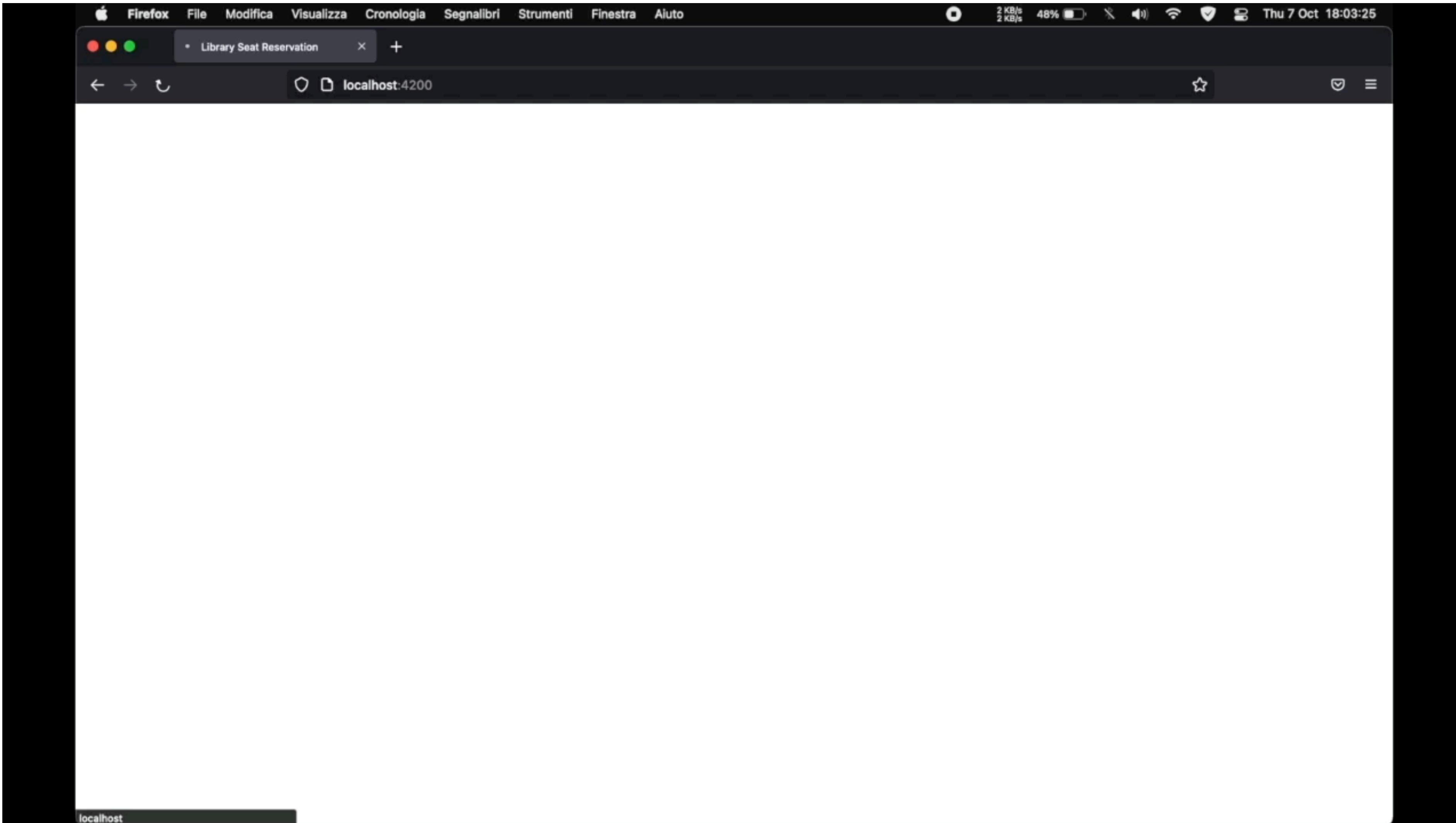


Mockup pagina della coda

# DEMO USER CON ENTRATA IN CODA

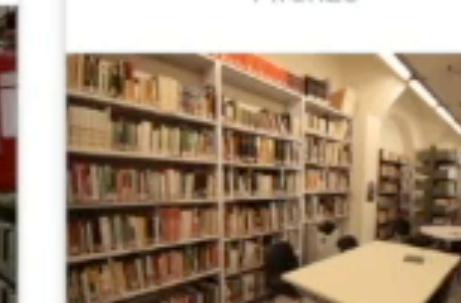


## DEMO ADMIN (1/2)



## DEMO ADMIN (2/2)

The screenshot shows a Firefox browser window displaying the 'Library Seat Reservation' application. The title bar includes the application name, the address 'localhost:4200/home', and the date/time 'Thu 7 Oct 18:05:53'. The top navigation bar has links for File, Modifica, Visualizza, Cronologia, Segnalibri, Strumenti, Finestra, and Aiuto. On the right, there is a user dropdown labeled 'Utente Admin (Admin)'. The main content area features a heading 'Prenota il tuo posto in biblioteca' and a search bar with the placeholder 'Cerca biblioteca...'. Below the search bar, there is a grid of cards representing different library branches:

- Biblioteca del Galluzzo**  
Via Senese, 206, Firenze  
  
Capacità: 70 posti
- Biblioteca delle Oblate**  
Via dell'Oriuolo, 24, Firenze  
  
Capacità: 80 posti
- Biblioteca Dino Pieraccioni**  
Via Nicolodi, 2, Firenze  
  
Capacità: 60 posti
- Biblioteca Fabrizio De André**  
Via delle Carra, 2, Firenze  
  
Capacità: 70 posti

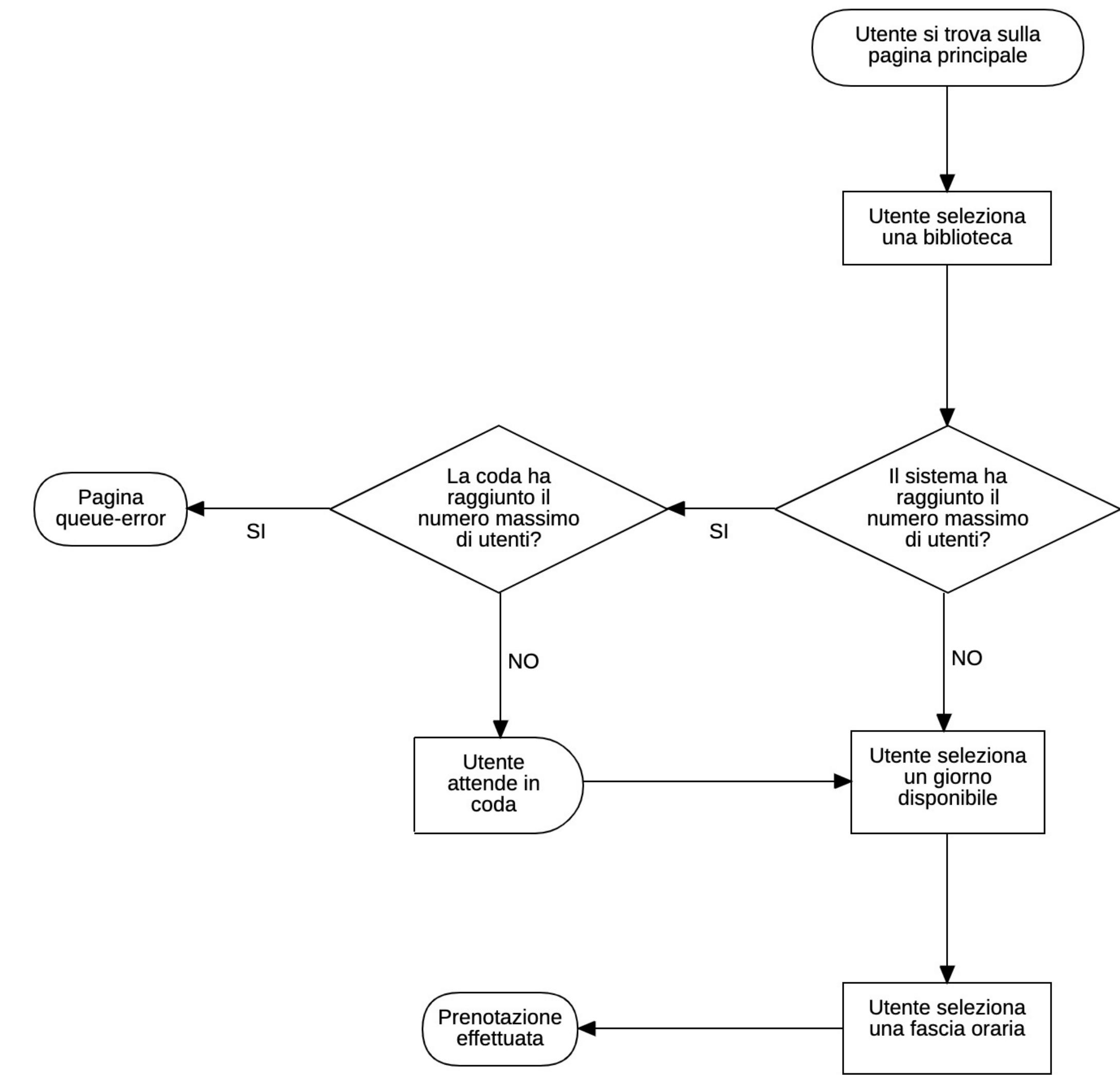
Below this row, there are three more cards partially visible:

- Biblioteca Filippo Buonarroti**
- Biblioteca ISIS Leonardo da Vinci**
- Biblioteca Mario Luzi**
- Biblioteca Orticoltura**

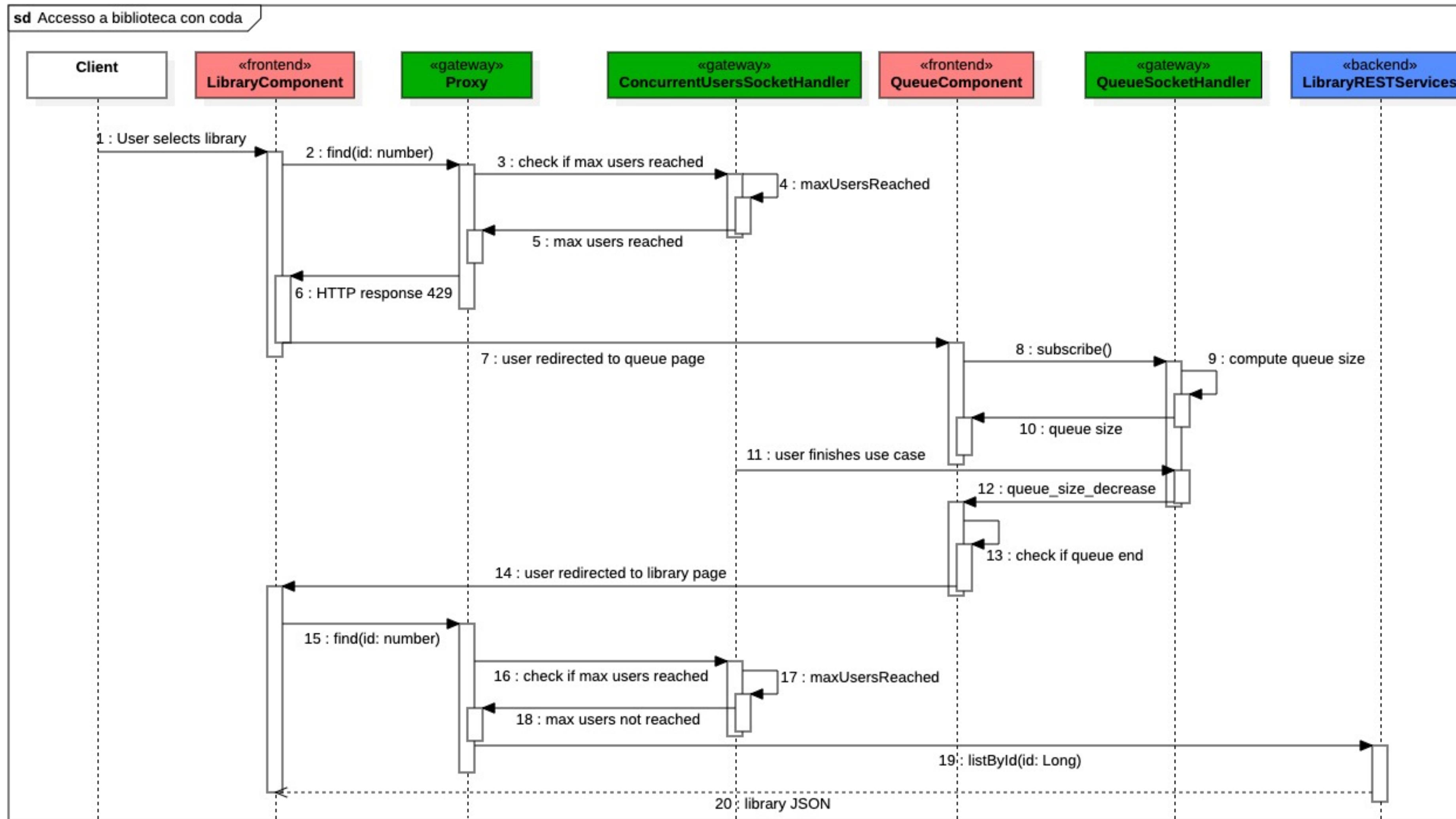


## CASO D'USO: GESTIONE DELLA CODA (1/2)

Diagramma di flusso del caso d'uso  
di un accesso a una biblioteca con  
possibilità di entrata in coda

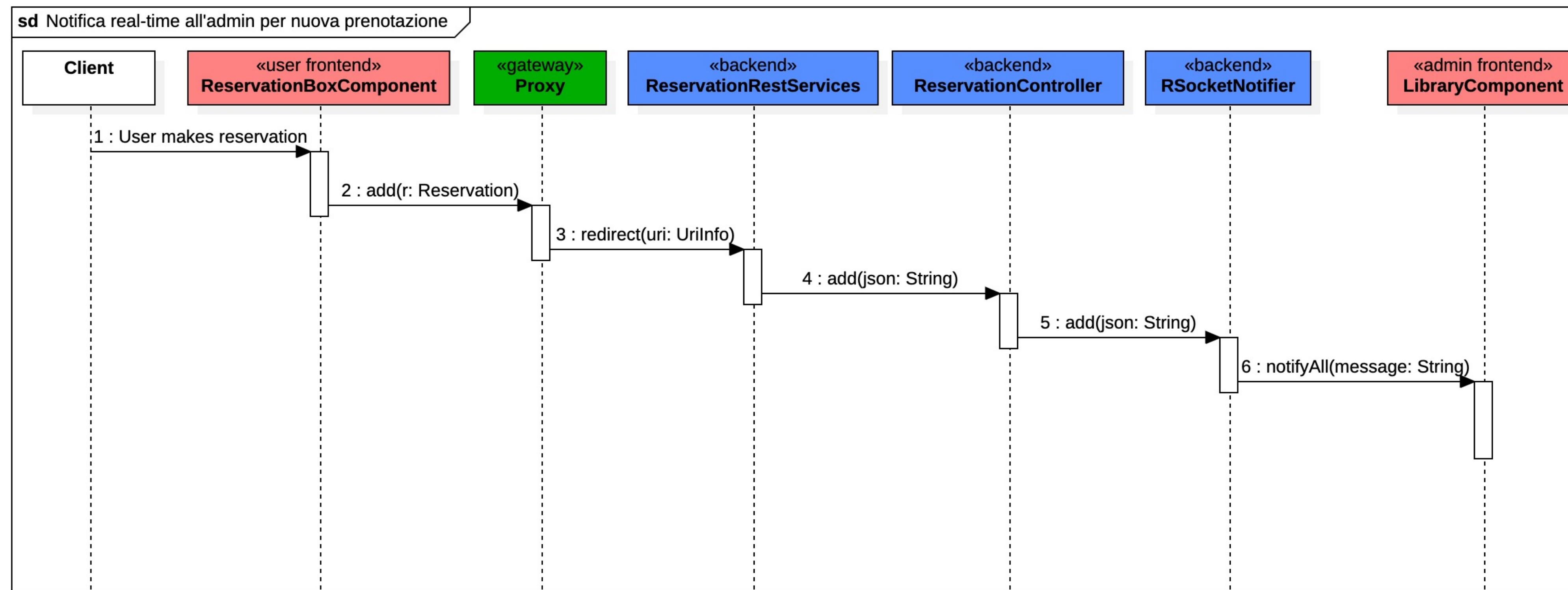


# CASO D'USO: GESTIONE DELLA CODA (2/2)



## CASO D'USO: NOTIFICA ADMIN CON RSOCKET

**Sequence diagram** del caso d'uso dell'arrivo di una **notifica in real-time** all'admin nel caso in cui un utente si sia prenotato per una determinata biblioteca



## DEMO NOTIFICA ADMIN (RSOCKET)

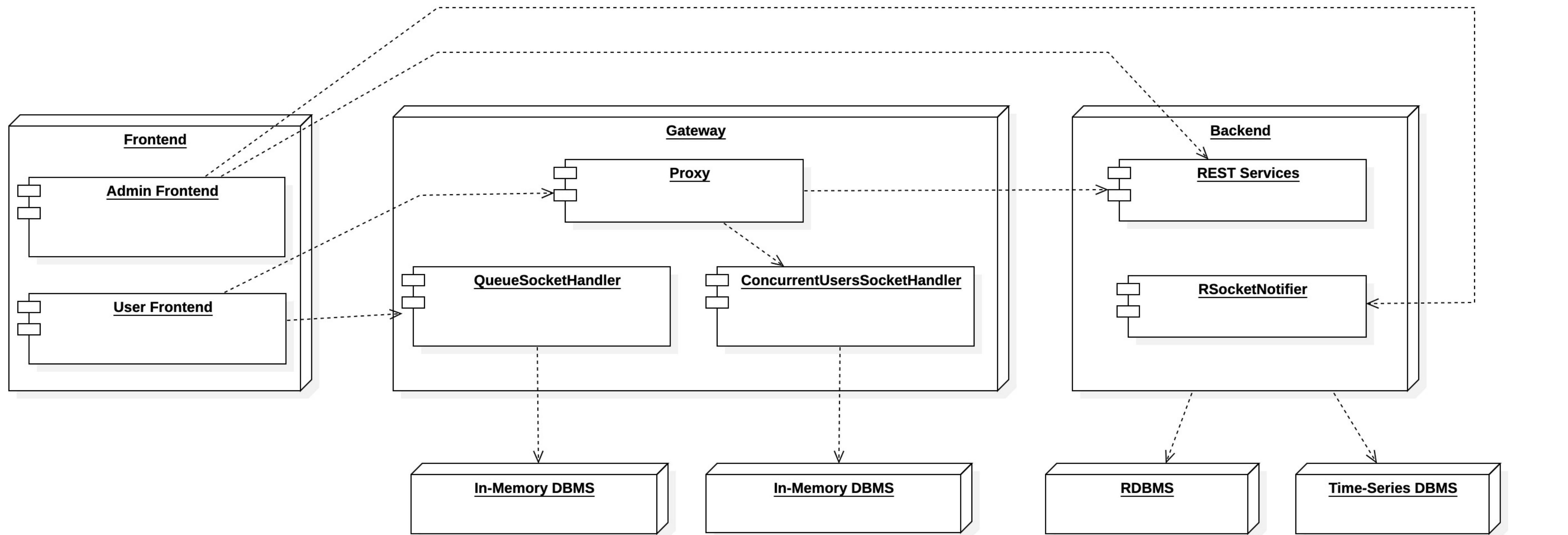
The screenshot shows a Firefox browser window displaying the 'Library Seat Reservation' application. The URL is `localhost:4200/libraries/4`. The interface includes:

- A map of a library location at the top.
- A monthly calendar for October 2021 on the left, with days from 1 to 31. Days 7, 8, 9, 14, 15, 16, 21, 22, 25, 26, 27, 28, 29, and 30 are green circles, while days 23 and 24 are red circles.
- A list titled "Prenotazioni per Giovedì 21 Ottobre" (Reservations for Thursday 21 October) on the right, showing four entries:

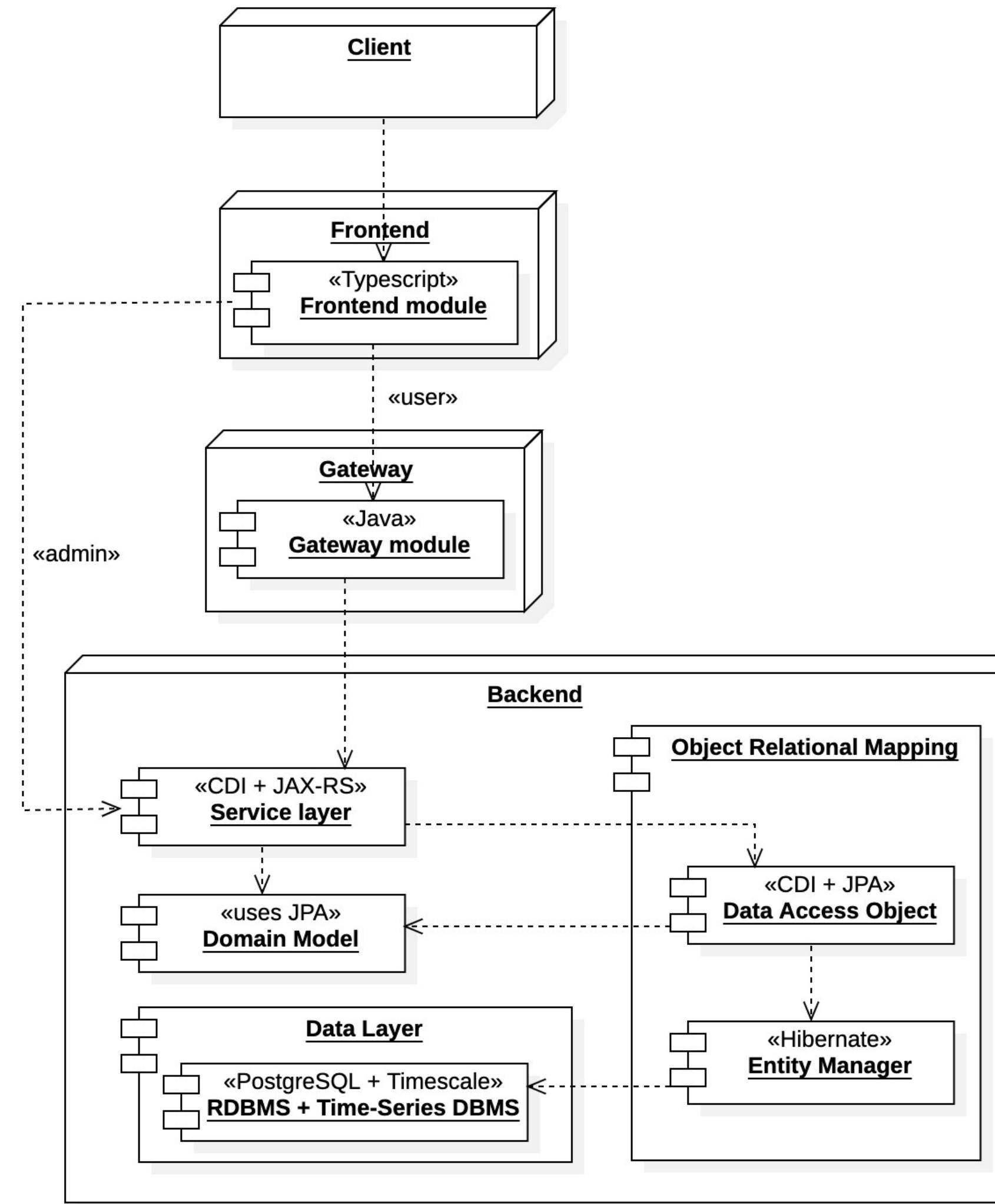
Nome	Email	Elimina
Utente 7856	user7856@email.com	
Utente 9938	user9938@email.com	
Utente 2965	user2965@email.com	
Utente 564	user564@email.com	

- Buttons for "Modifica capacità biblioteca" (Modify library capacity) and "Elimina biblioteca" (Delete library) at the bottom.

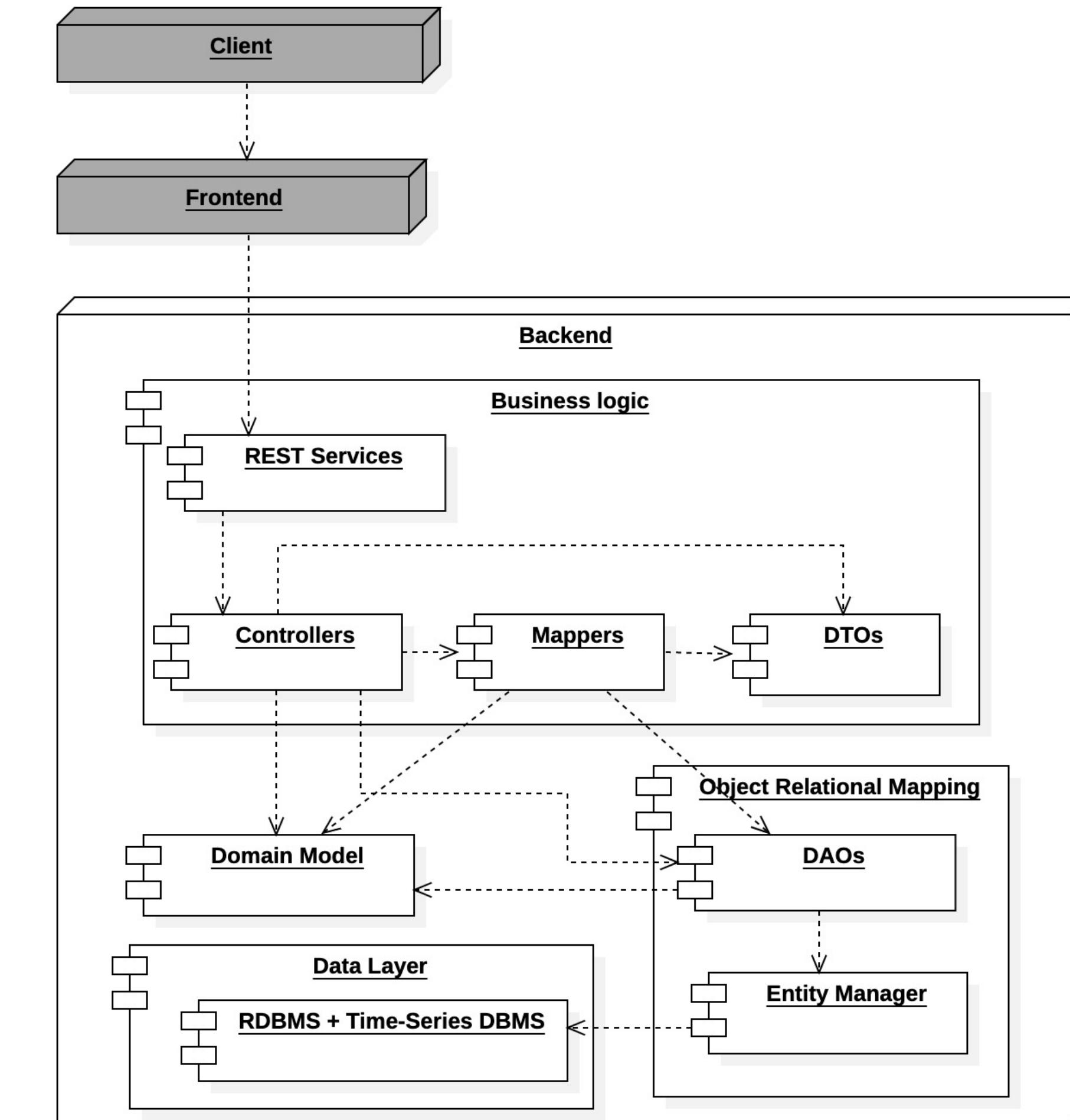
# ARCHITETTURA COMPLESSIVA



# ARCHITETTURA NEL DETTAGLIO



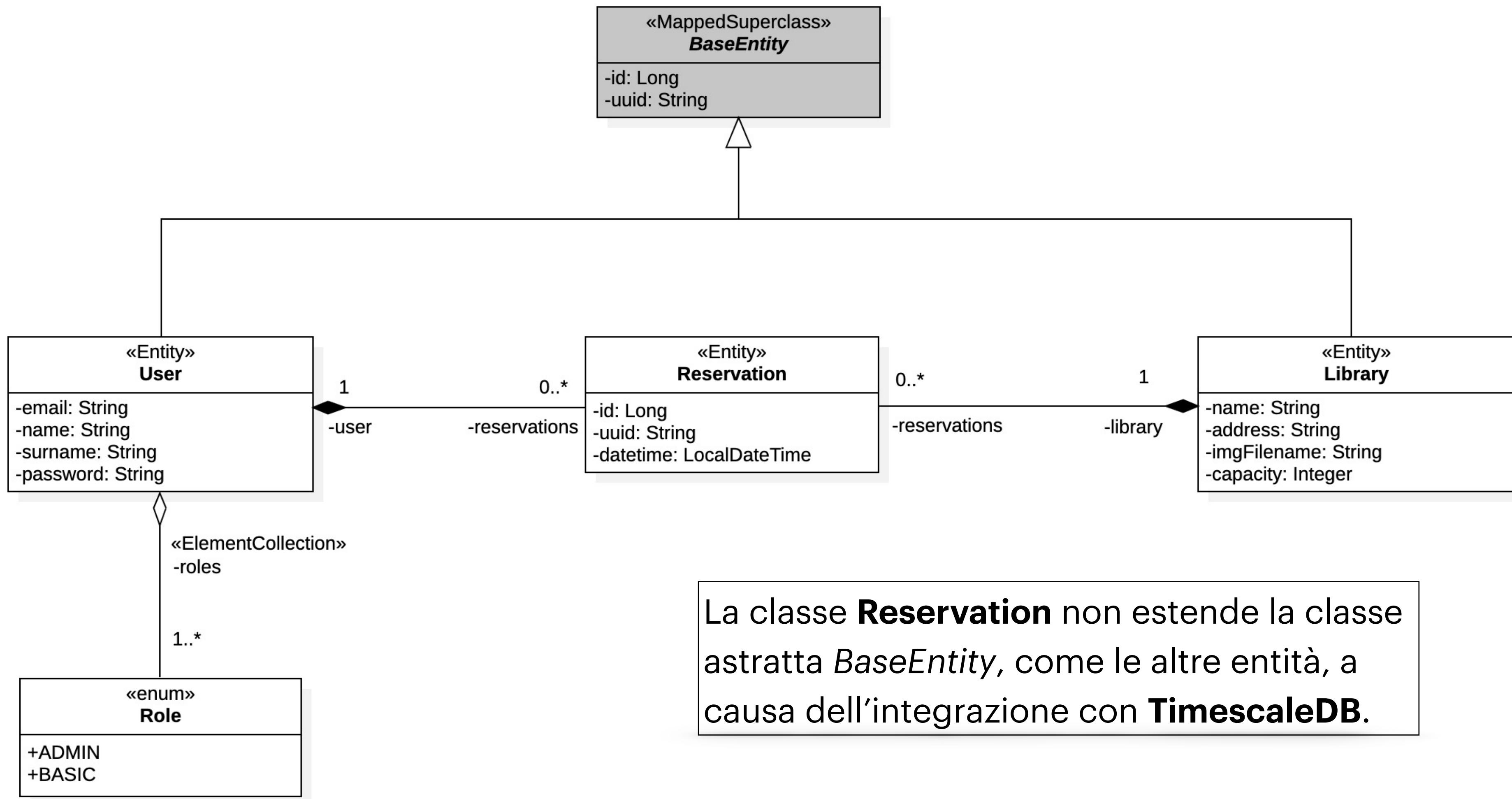
Architettura dettagliata con tecnologie utilizzate



Dettaglio del modulo backend



# BACKEND - DOMAIN MODEL



# BACKEND

## DATA TRANSFER OBJECTS (DTO)

**UserDTO**

```
-id: long
-email: String
-name: String
-surname: String
-password: String
-roles: List<String>
```

**ReservationDTO**

```
-id: long
-userId: long
-userName: String
-userEmail: String
-libraryId: long
-libraryName: String
-datetime: String
```

**LibraryDTO**

```
-id: long
-name: String
-imgFilename: String
-address: String
-capacity: Integer
```

**AdminNotificationDTO**

```
-action: UserAction
-reservationId: Long
-libraryId: Long
-date: String
-notificationMessage: String
```

**ReservationDailyAggregateDTO**

```
-date: String
-countMorning: Integer
-countAfternoon: Integer
```

## DATA ACCESS OBJECTS (DAO)

**BaseDAO**

```
#entityManager: EntityManager
-type: Class<T>

+all(): List<T>
+findById(id: Long): T
+save(entity: T): void
+update(updatedEntity: T, id: Long): void
+delete(id: Long): void
```

**UserDAO**

```
+findByEmail(email: String): User
+verify(email: String, password: String): boolean
+login(email: String, password: String): User
```

**LibraryDAO**

**ReservationDAO**

```
#entityManger: EntityManager

+all(): List<ReservationDto>
+findById(id: Long): ReservationDto
+findByUserId(id: Long): List<ReservationDto>
+findByLibraryId(id: Long): List<ReservationDto>
+findByLibraryIdAndDate(libraryId: Long, year: int, month: int, day: int): List<ReservationDto>
+dailyAggregateByLibraryIdAndMonth(libraryId: Long, year: int, month: int): List<ReservationsDailyAggregateDto>
+save(entity: Reservation): void
+delete(id: Long): void
+enableTimescalePostgresExtensionIfNeeded(): void
+setupHypertable(): void
```



## TESTING

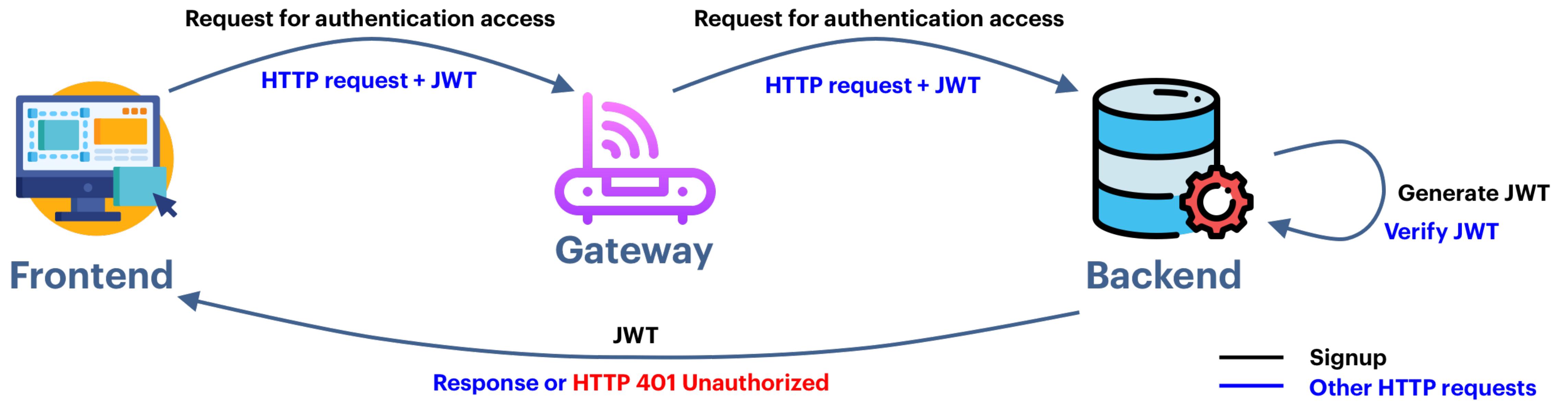
Per il testing abbiamo utilizzato il framework **JUnit5** (per lo unit testing) e **Mockito** (per evitare problemi di dipendenze tra oggetti e aumentare l'isolamento).

Sono stati testati diversi **livelli dell'architettura (Domain Model, Business Logic e DAO)** e i loro componenti critici.

- **Domain Model:** è stata testata la superclasse  *BaseEntity* e  *Reservation* per verificare la corretta identità, uguaglianza ed inizializzazione
- **Business Logic:** i test vengono eseguiti in isolamento, e per eventuali dipendenze esterne sono stati utilizzati i  *mocks*
- **DAO:** sono stati testati i metodi CRUD ed eventuali altri metodi rilevanti. Per testare la persistenza è stato utilizzato un database in-memory (**HyperSQL**)



# AUTENTICAZIONE



Autenticazione tramite JSON Web Token (JWT)

## DEPLOY CON WILDFLY SU DOCKER

Per effettuare il deploy sono stati utilizzati due container Docker:



- Per il **backend** un'immagine **Wildfly** (versione 24.0.0) modificata per il supporto a database **PostgreSQL** (su cui si basa TimescaleDB).
- Per il **gateway** un'immagine standard di **Wildfly** (versione 24.0.0)

```
wildfly:  
  container_name: "wildfly"  
  build:  
    context: .  
    dockerfile: Dockerfile  
  environment:  
    - WILDFLY_USER=admin  
    - WILDFLY_PASS=password  
    - DB_NAME=lsr-db  
    - DB_USER=postgres  
    - DB_PASS=postgres  
    - DB_HOST=db  
    - DB_PORT=5432  
  depends_on:  
    - db  
  volumes:  
    - ./workdir/deploy/wildfly/:/opt/jboss/wildfly/standalone/deployments/:rw  
  ports:  
    - "8080:8080" # application  
    - "9990:9990" # admin console  
    - "5005:5005" # debug port  
    - "7878:7878" # rsocket port
```

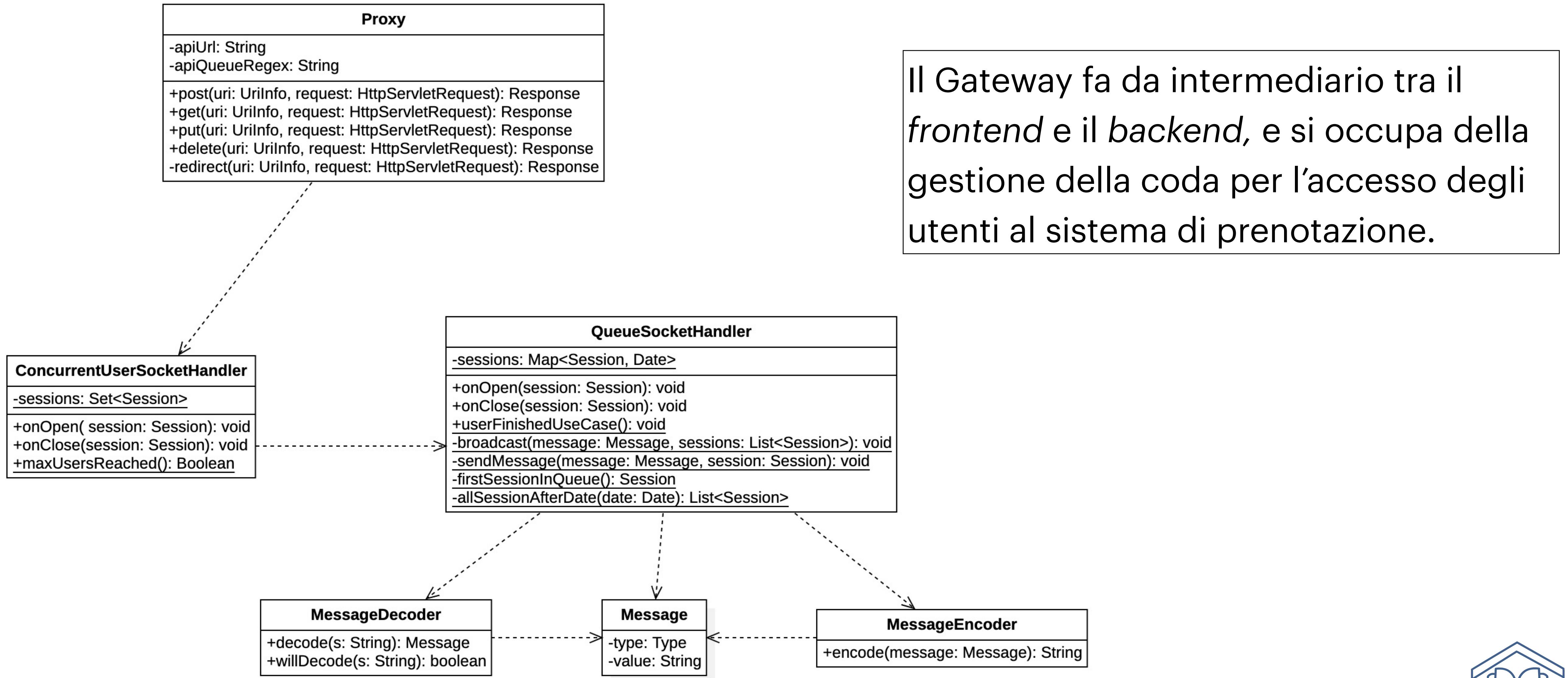
docker-compose backend 1/2

```
db:  
  container_name: "db"  
  image: "timescale/timescaledb:latest-pg13"  
  environment:  
    - POSTGRES_DB=lsr-db  
    - POSTGRES_USER=postgres  
    - POSTGRES_PASSWORD=postgres  
  volumes:  
    - ./workdir/db/init/:/docker-entrypoint-initdb.d/  
    - ./workdir/db/data/:/var/lib/postgresql/  
  ports:  
    - "5432:5432"
```

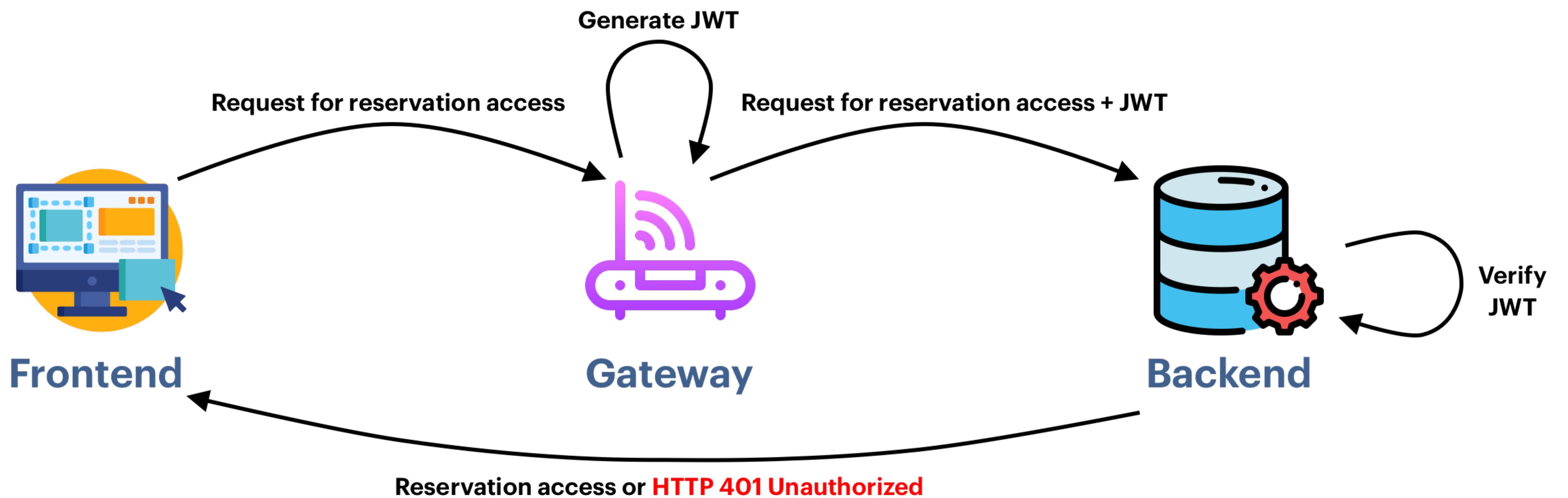
docker-compose backend 2/2



# GATEWAY - DOMAIN MODEL



# AUTORIZZAZIONE

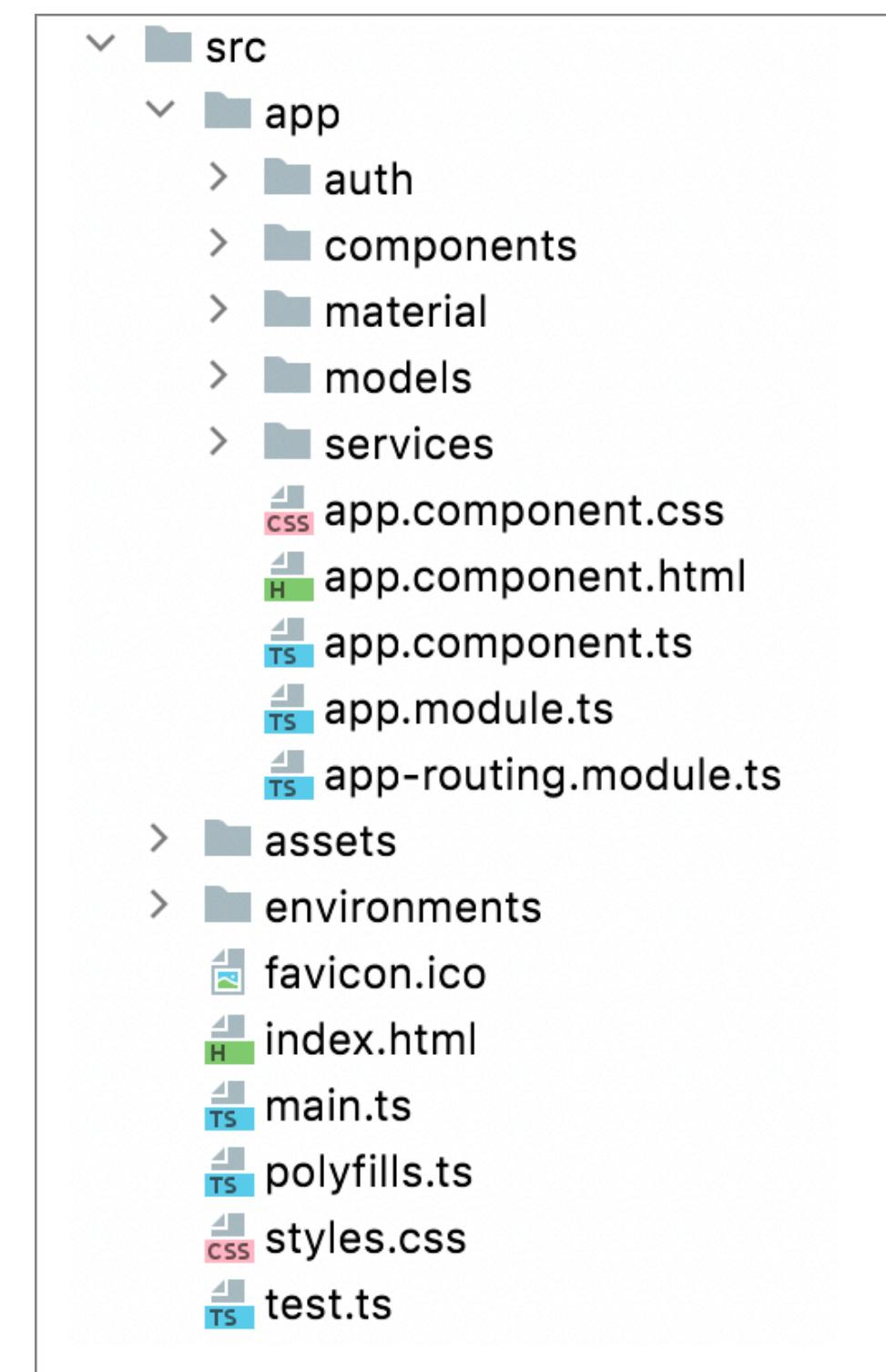


Autorizzazione tramite JSON Web Token (JWT)

# FRONTEND

Il Frontend è stato sviluppato tramite il framework **Angular** con l'aiuto dei componenti forniti dalla libreria **Angular Material**.

- **TypeScript** come linguaggio di base
- L'applicazione è stata realizzata come una **Single Page Application (SPA)**
- Orientata ai servizi
- Con paradigma **Model-View-Controller**



## FRONTEND - DIPENDENZE

Il modulo frontend realizzato fa uso delle seguenti dipendenze esterne:

- **Angular Material**: design system per i componenti UI
- **Leaflet**: libreria JavaScript per mappe interattive
- **angularx-qrcode**: libreria per la generazione di QR code
- **RSocket**: protocollo a livello applicativo per stream reattivi
- **RxJS**: libreria JavaScript che mette a disposizione il tipo Observable e la funzione pipe( ), che permette di compiere più operazioni sul risultato in emissione



# FRONTEND - DIAGRAMMI DI NAVIGAZIONE (1/2)

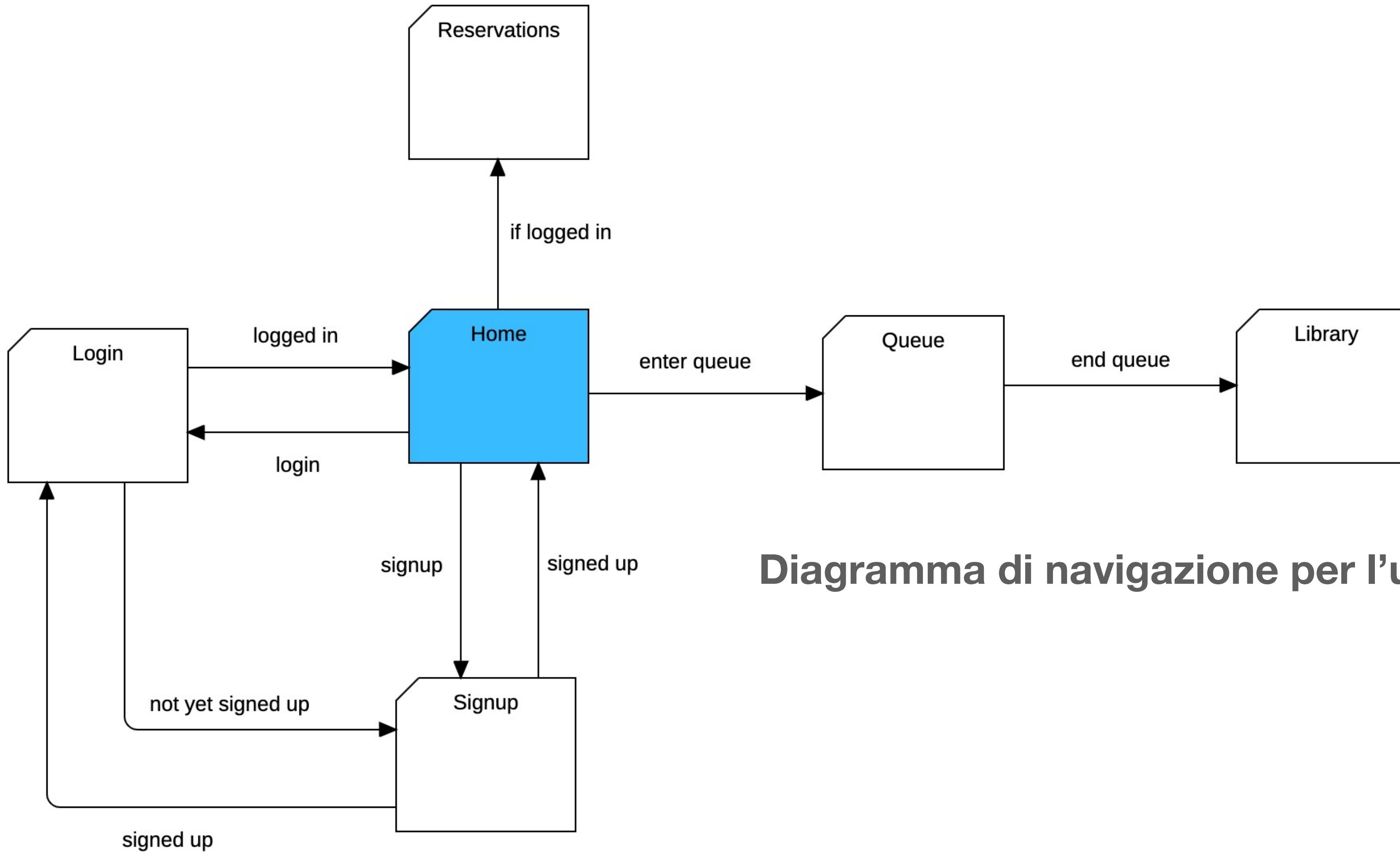


Diagramma di navigazione per l'utente

## FRONTEND - DIAGRAMMI DI NAVIGAZIONE (2/2)

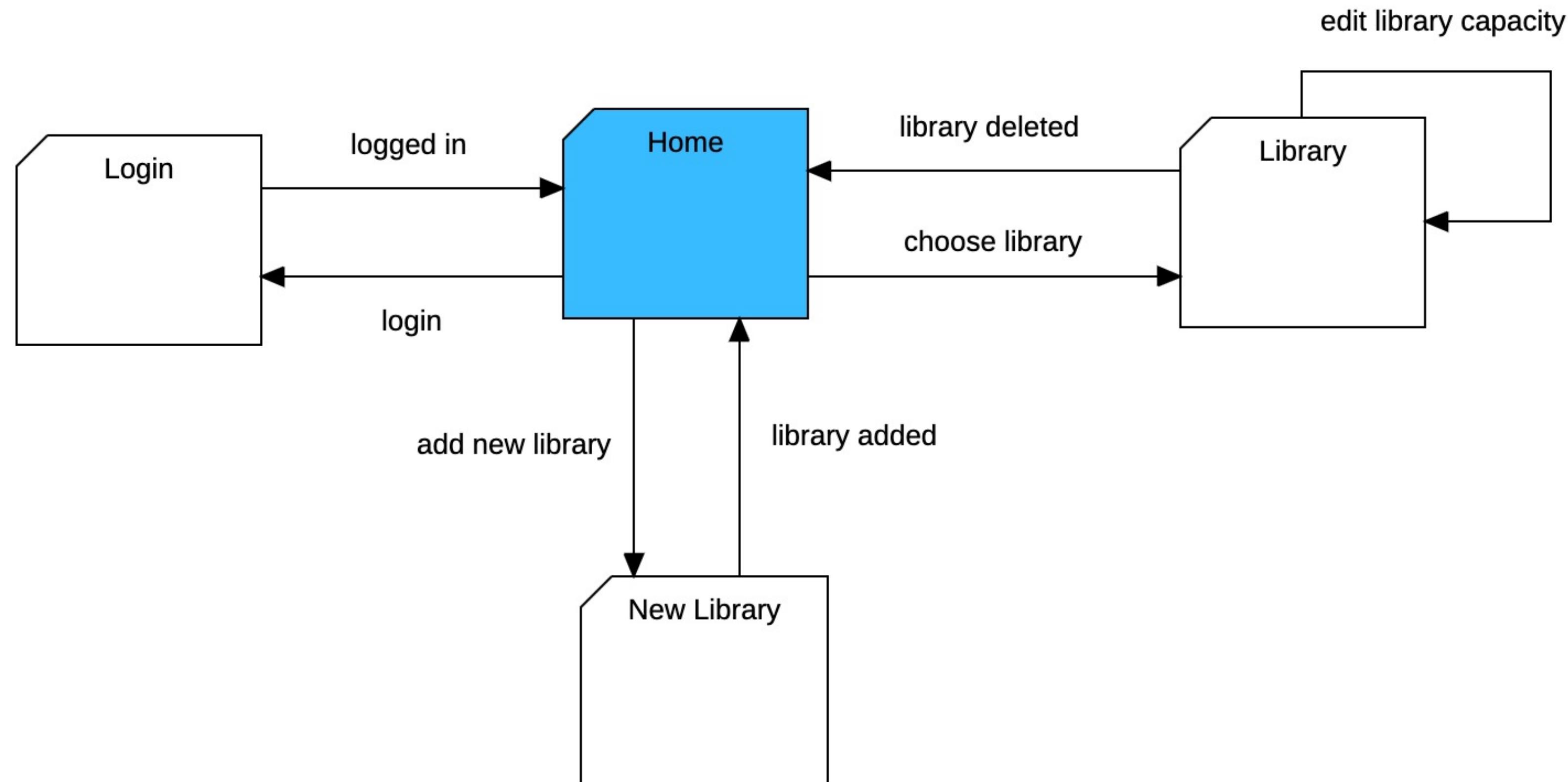


Diagramma di navigazione per l'admin

## UNIVERSALIZZAZIONE DEL GATEWAY

Il modulo **Gateway** potrebbe diventare **indipendente** e **universale**, configurabile e adattabile ai vari casi d'uso.

*Come si può adattare il **modulo gateway** ad un contesto applicativo diverso?*

- Con parametri configurabili da parte dello sviluppatore in base alle necessità:
  - L'URL relativo alle API REST del backend
  - L'espressione regolare che indica le API "critiche" da proteggere con la coda
  - Il numero massimo di utenti che il sistema può ospitare contemporaneamente
  - La lunghezza massima della coda
- Adattando anche il modulo frontend alla gestione della coda (es. se *il gateway comunica che il sistema è pieno, fare redirect alla pagina della coda*)



## CONCLUSIONI E SVILUPPI FUTURI

Il sistema realizzato implementa tutte le specifiche indicate nell'analisi dei requisiti, fornendo all'utente una piattaforma per la **gestione delle prenotazioni di posti all'interno delle aule studio delle biblioteche di Firenze.**

- Estendere l'utilizzo ad un **territorio più ampio** (es. biblioteche della Toscana)
- Aggiunta di **più fasce orarie** (non solo mattina/pomeriggio)
- Offrire una prenotazione **specifica** per una determinata **aula o spazio di lettura** di una biblioteca (es. Aula A della biblioteca *Villa Bandini*)

