

Library Seat Reservation

SOFTWARE ARCHITECTURES AND METHODOLOGIES



INTRODUZIONE

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

Gli **obiettivi** di maggiore rilievo sono:

- Un sistema di **gestione “a code”** delle richieste pervenute
 - ispirandosi al portale di vaccinazione regionale
 - con un componente distribuito (*gateway*) che intermedia le richieste
 - con il risultato di non sovraccaricare il server
- Due **frontend** (uno per l'utente semplice e uno per l'admin)



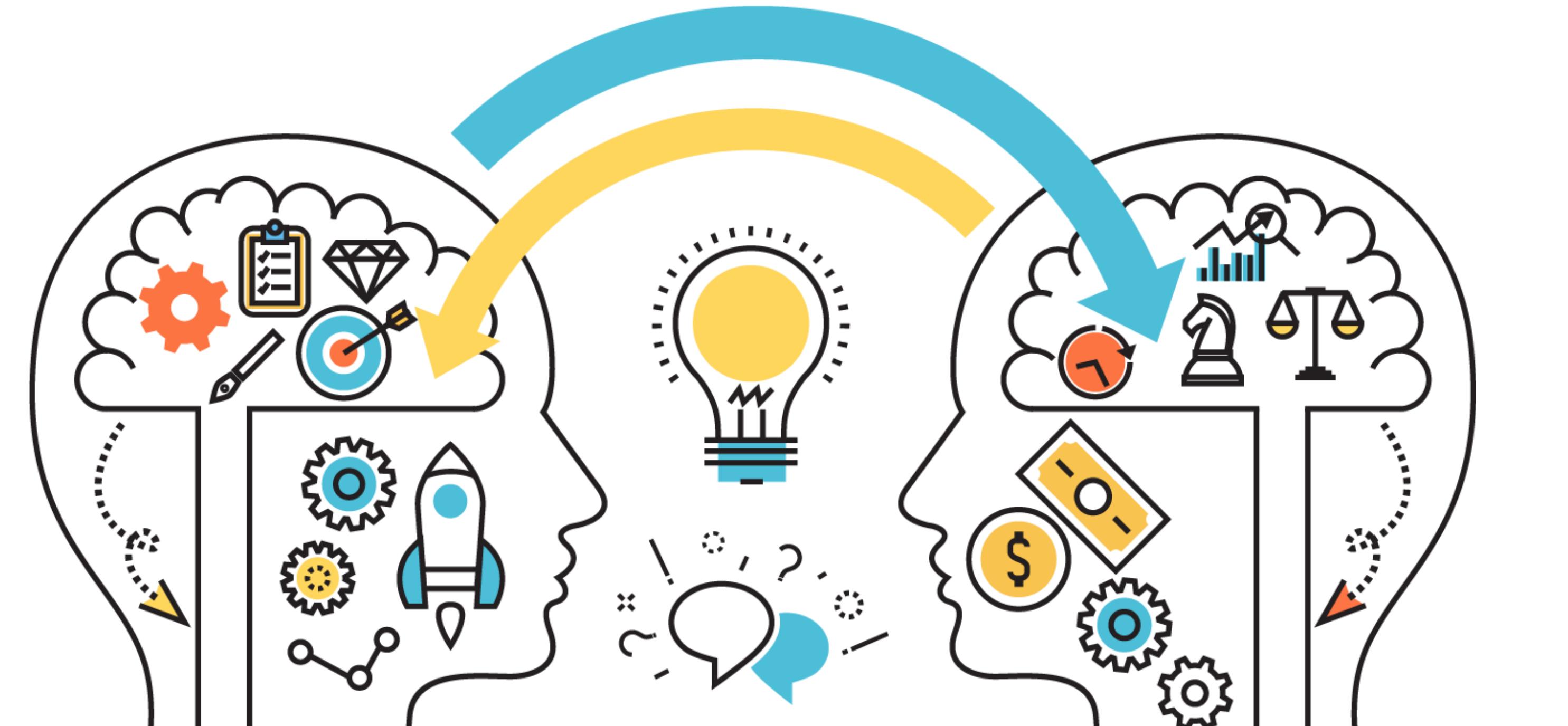
TECNOLOGIE UTILIZZATE

- **Backend e Gateway:** Java Enterprise Edition (JEE)
- **Frontend:** Angular (TypeScript) con Angular Material per il design
- **WebSocket e RSocket** per la comunicazione real-time tra i moduli
- **JWT** per la gestione dell'autenticazione e dell'autorizzazione
- **Docker** per la divisione in container (**Wildfly** e database PostgreSQL)
- **Timescale** per gestione di query temporali in maniera efficiente



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

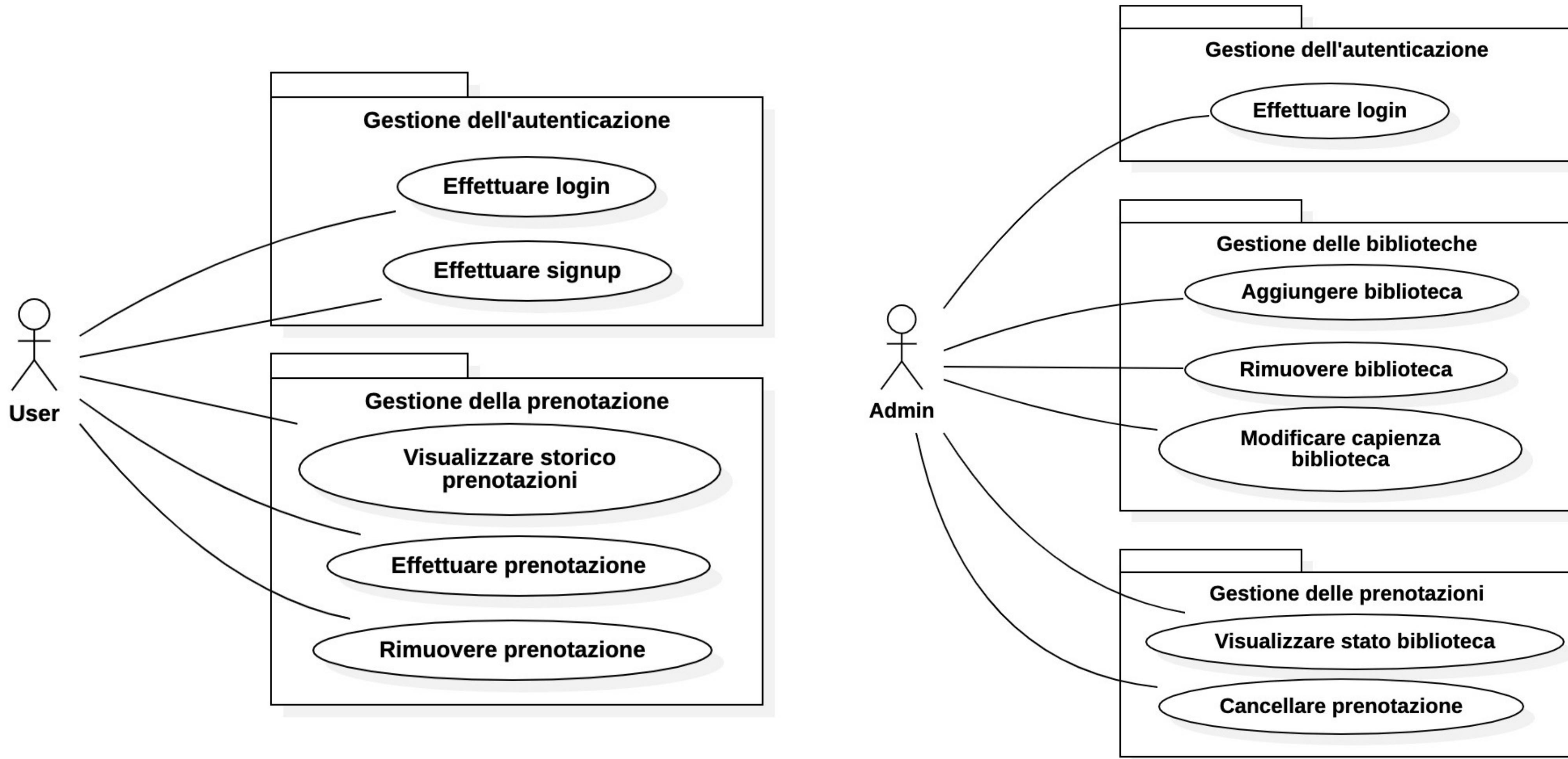
- Il sistema dovrà avere una **natura distribuita**
- **Architettura RESTful**
- Logica **gateway** indipendente dal contesto applicativo

REQUISITI DI DOMINIO

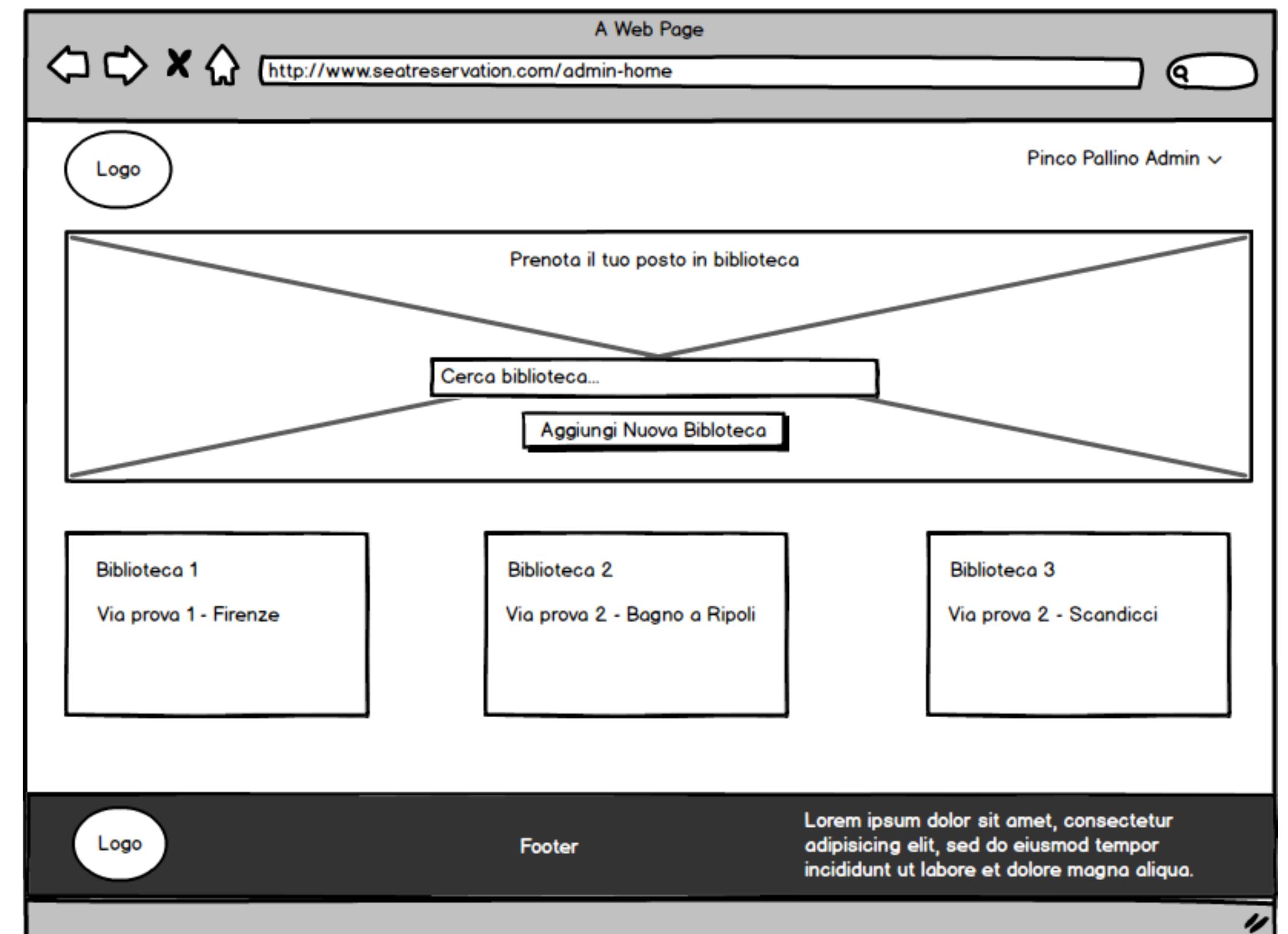
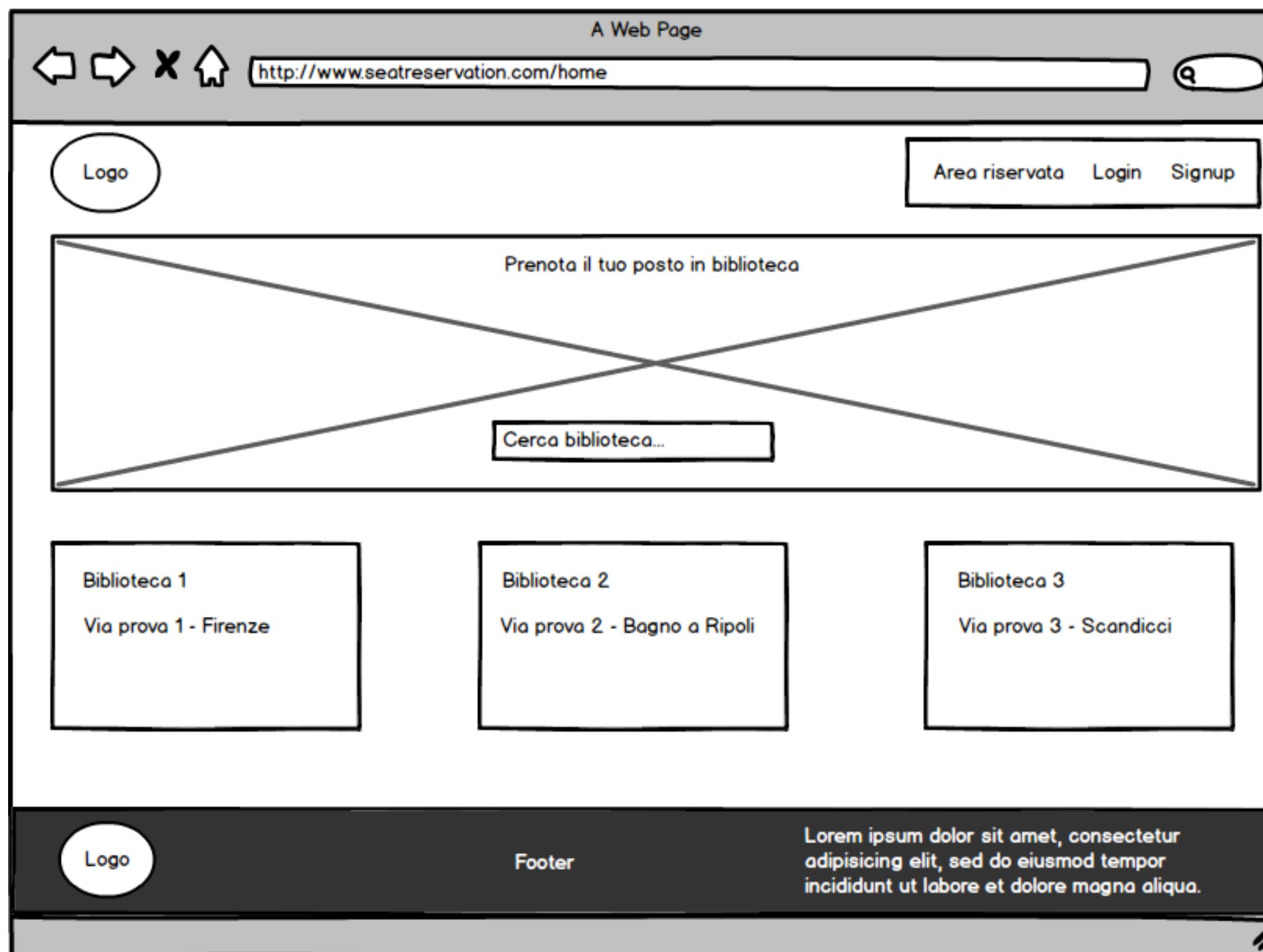
- **Utente**: id, email, nome, cognome, password e lista di ruoli
- **Biblioteca**: id, nome, indirizzo, capienza
- **Prenotazione**: id, id utente, id biblioteca, data e fascia oraria



ANALISI DEI REQUISITI - CASI D'USO

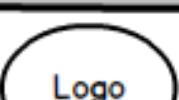


MOCKUP (1/3)



MOCKUP (2/3)

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



Area riservata Login Signup

Biblioteca 1
Via prova 1 - Firenze



JULY 2021						
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

8:00 - 13:00 12 / 50
 13:00 - 19:00 50 / 50

Prenota

Footer

Logo

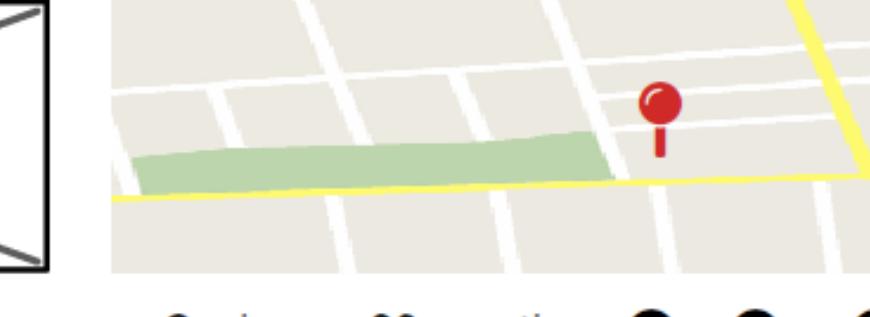
Lore ipsum dolor sit amet, consectetur adipisciing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Pinco Pallino Admin ▾



Biblioteca 1
Via prova 1 - Firenze



Capienza 60 posti - + ↻

JULY 2021						
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

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

Occupazione 16/60

- Pippo Pluto trash
- Paperino Quo trash
- Samuele Ceccherini trash
- Topolino Qua trash

Elimina Biblioteca

Footer

Logo

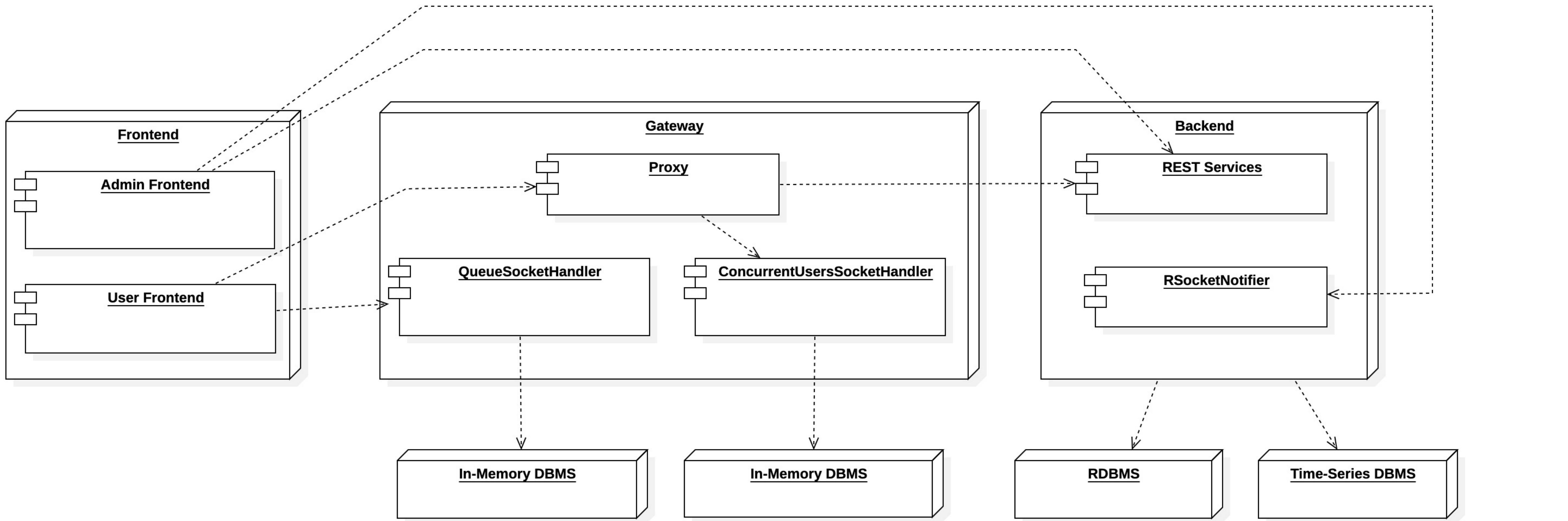
Lore ipsum dolor sit amet, consectetur adipisciing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



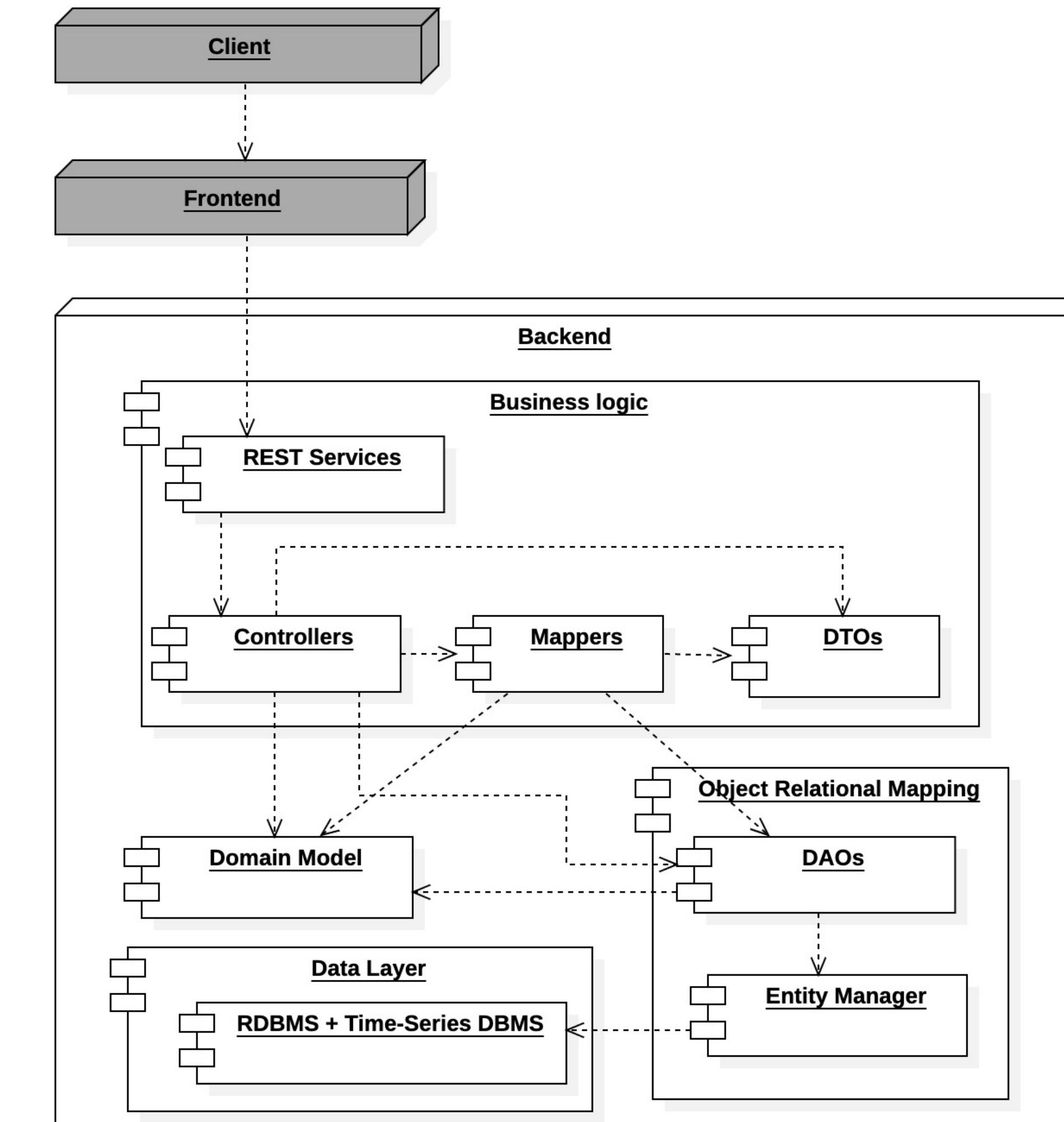
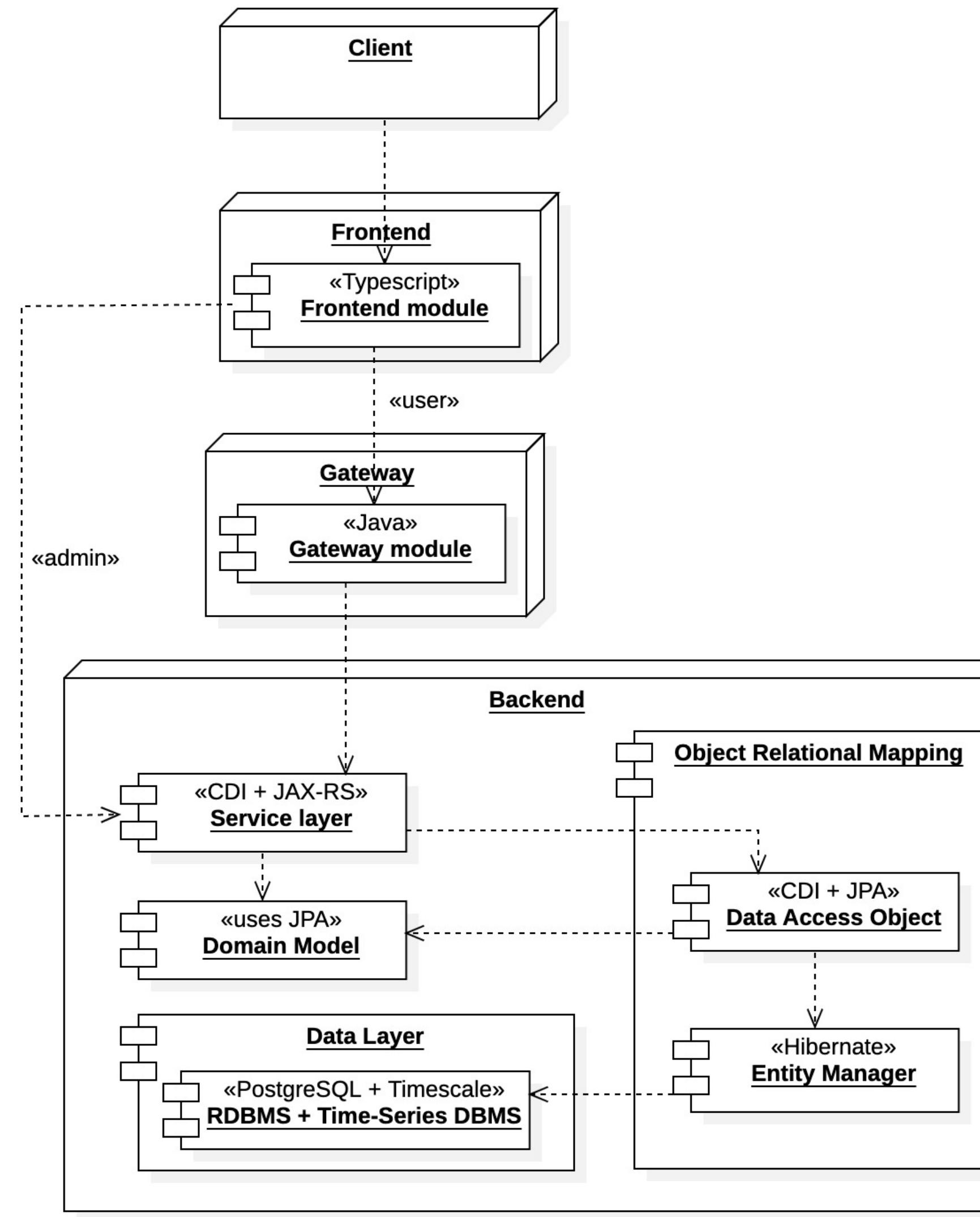
MOCKUP (3/3)



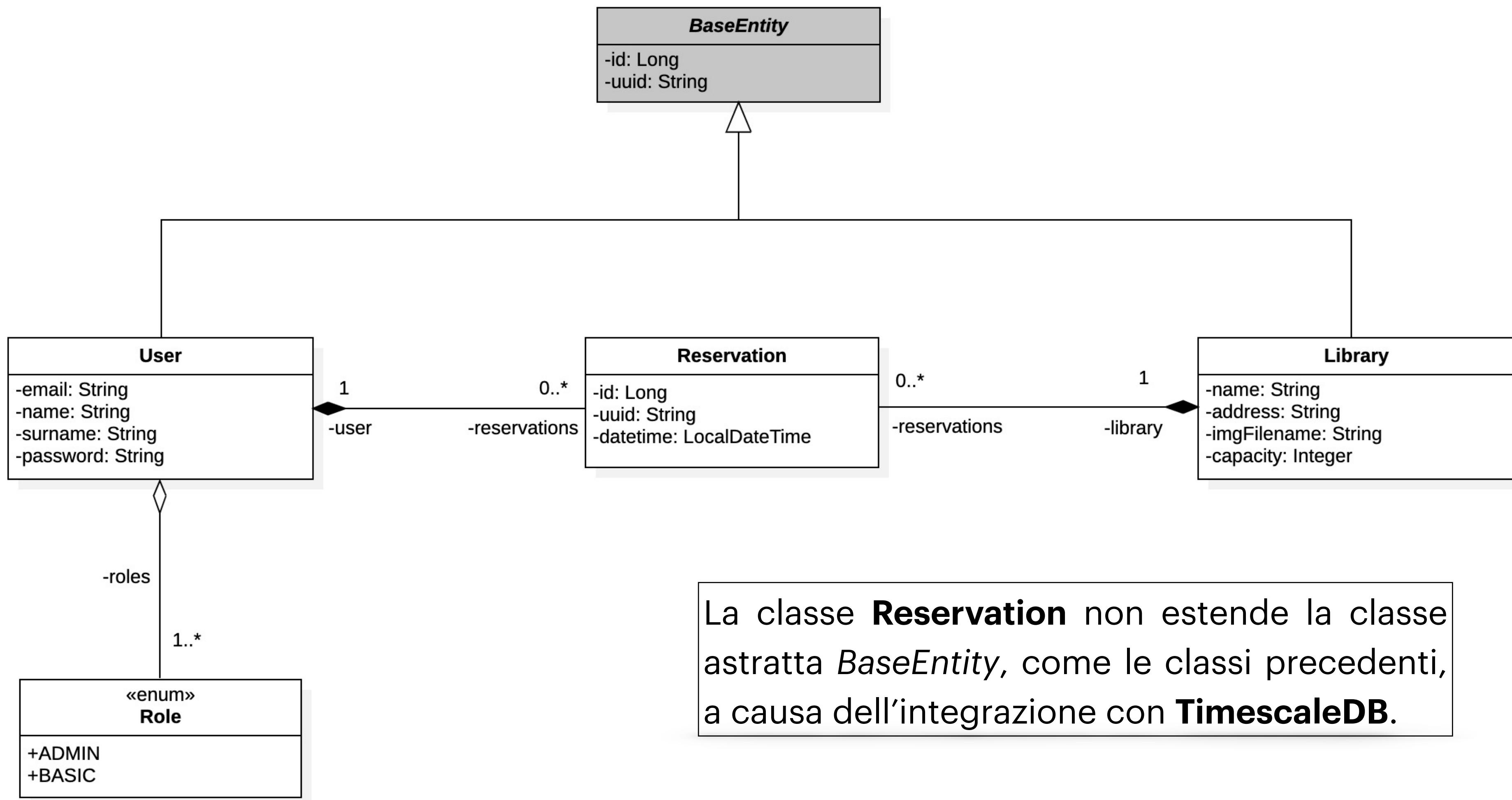
ARCHITETTURA COMPLESSIVA



ARCHITETTURA NEL DETTAGLIO



BACKEND - DOMAIN MODEL



BACKEND - TIMESCALEDB

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

Vantaggi:

- Supporto **nativo** al linguaggio SQL
- Maggiore facilità d'uso per l'analisi delle **time-series**
- Prestazioni **migliorate**



In particolare la funzione `time_bucket()` viene utilizzata per aggregare periodi di tempo di dimensioni arbitrarie (es. 5 minuti, 1 giorno).

BACKEND - DATA TRANSFER OBJECTS (DTO)

Modella un oggetto software che rappresenta una versione semplificata di oggetti riferiti al modello di dominio.

Svolge il ruolo di **trasportatore di dati** tra processi comunicanti, con la finalità di ridurre al quantit  di informazioni trasferite verso i chiamanti

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

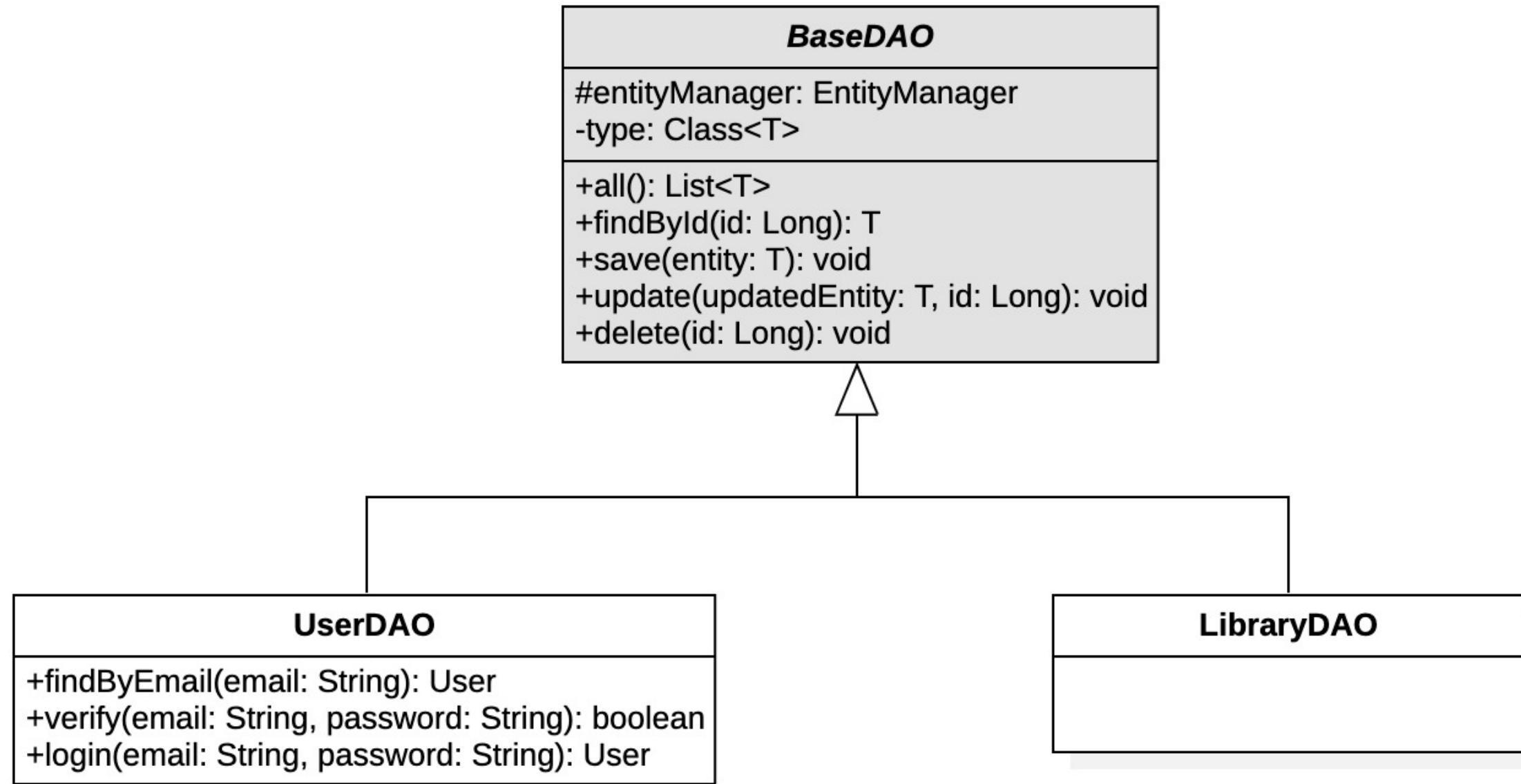
LibraryDTO
-id: long
-name: String
-imgFilename: String
-address: String
-capacity: Integer

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

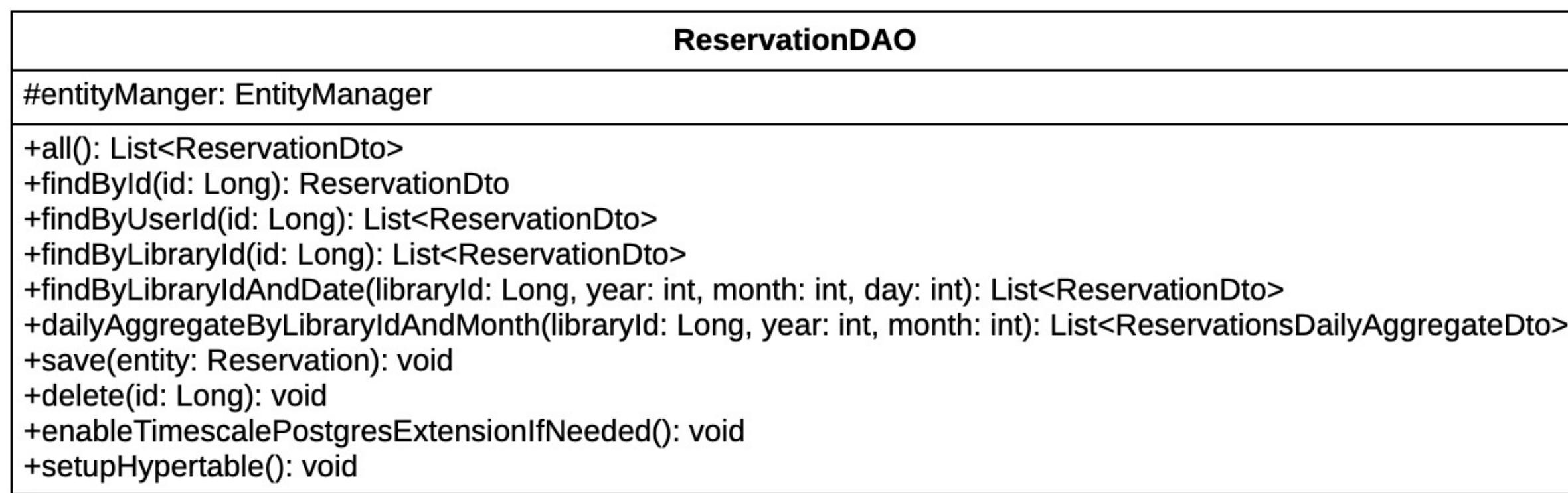
ReservationDailyAggregateDTO
-date: String
-countMorning: Integer
-countAfternoon: Integer



BACKEND - DATA ACCESS OBJECTS (DAO)



Modella un oggetto software che fornisce un'interfaccia astratta verso il livello di persistenza ed espone un insieme di metodi utili per coprire le principali operazioni **CRUD**.



Svolge il ruolo di **intermediario** tra le entità del modello di dominio e le tabelle “reali” del database.



BACKEND - ALTRI COMPONENTI

MAPPER

Modella un oggetto software che espone alcuni metodi per “mappare” gli oggetti Java (entità del domain model) in DTO e viceversa:

- Entità —————> DTO
- DTO —————> Entità

CONTROLLER

Rappresentano dei componenti software in grado di assolvere il caso d'uso invocando o implementando direttamente i metodi caratterizzanti:

- Manipolano le entità del Domain Model per intermediazione dei DAO
- Elaborano ed interpretano istanze di DTO tramite intermediazione dei Mapper



BACKEND - AUTENTICAZIONE

Per l'autenticazione è stato utilizzato lo standard **JSON Web Token** (JWT) del protocollo **OAuth 2.0** tramite la libreria **Java JWT** (JJWT).

Eseguita nei seguenti passi:

- **Login**: viene inviata dal frontend una richiesta HTTP(S) contenente *email* e *password*
- Il backend controlla le credenziali e risponde inviando un *token* al client, valido per sei ore
- Il client inoltrerà il *token* per ogni successiva richiesta HTTP(S)
- Il backend controlla che il *token* sia valido (verifica il digest)
- **Logout**: viene rimosso il *token* lato client



BACKEND - ENDPOINT

Tutti i servizi associati ad una certa risorsa sono gestiti da un **endpoint**, che quindi può essere definito come un contenitore logico che raggruppa e organizza i servizi esposti e che li rende fruibili ai client in modo univoco tramite URI.

Si identificano quindi tre endpoint:

- **UserRestServices** (api/users/)
- **LibraryRestServices** (api/libraries/)
- **ReservationRestServices** (api/reservations/)



BACKEND - 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**)

DEPLOY CON WILDFLY SU DOCKER

Per effettuare il deploy è stato utilizzata un'immagine Docker di **Wildfly** versione 24.0.0, modificata per il supporto a database **PostgreSQL** (su cui si basa TimescaleDB).

In particolare sono stati prodotti i seguenti file:

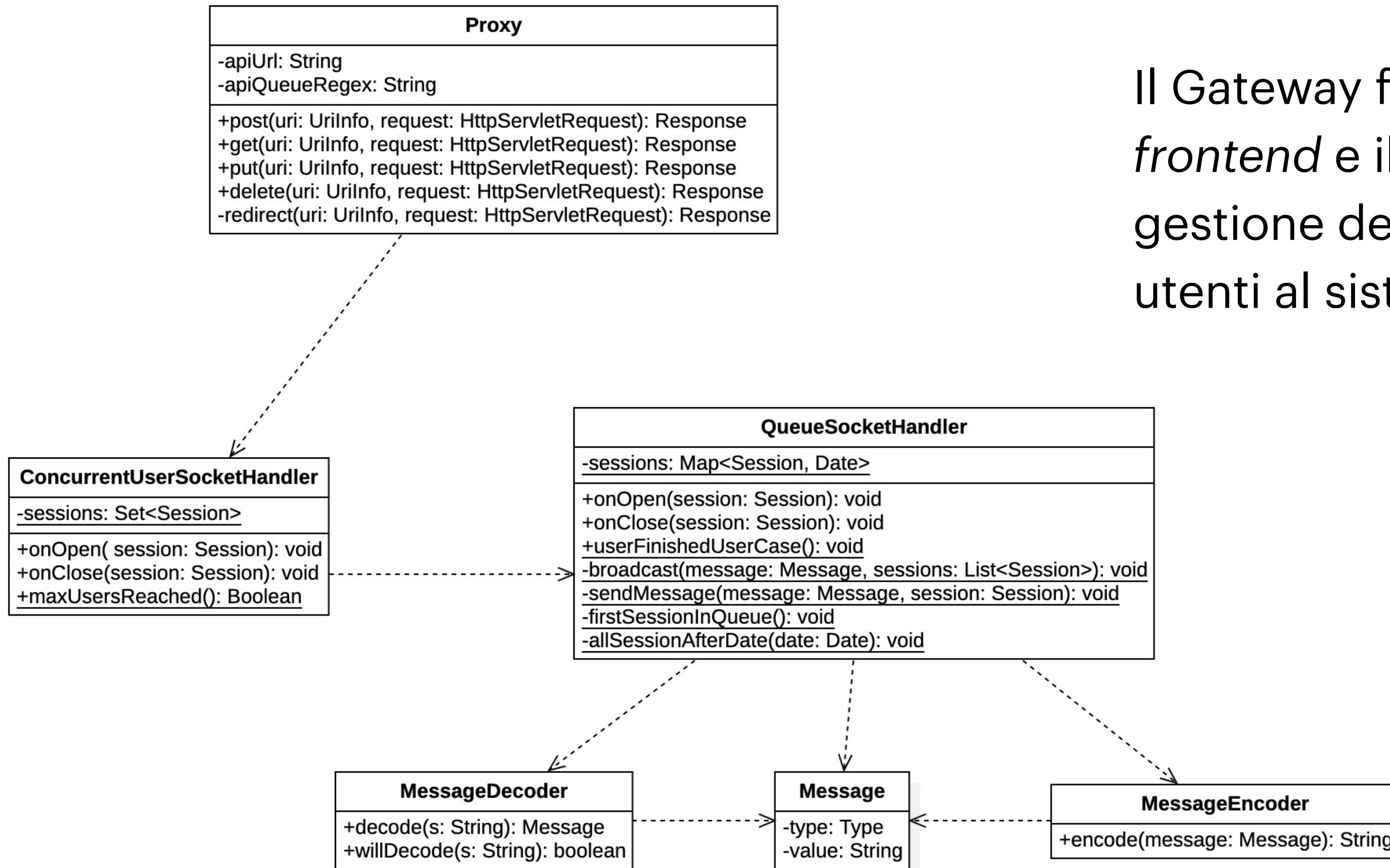
- **Dockerfile**: a partire dall'immagine Wildfly ufficiale, genera un'immagine Docker modificata per il supporto a PostgreSQL
- **docker-compose.yml**: specifica due services, uno per il database PostgreSQL e l'altro per l'application server con l'immagine Wildfly modificata tramite Dockerfile

Le seguenti **porte** vengono esposte:

- **8080** per accedere all'applicazione
- **9990** per la console admin
- **5005** per il debugging
- **7878** per eventuali comunicazioni tramite RSocket



GATEWAY - DOMAIN MODEL



Il Gateway fa da intermediario tra il *frontend* e il *backend*, e si occupa della gestione della coda per l'accesso degli utenti al sistema di prenotazione.



GATEWAY - AUTORIZZAZIONE

Per assicurare che l'utente BASIC non possa bypassare il Gateway inviando richieste non autorizzate direttamente al backend, viene utilizzato un altro token **JWT**:

- Il **Frontend** fa una richiesta al Gateway per poter accedere al servizio di prenotazione
- Il **Gateway** riceve la richiesta e genera un token valido per *due minuti* ed inoltra la richiesta al backend insieme al token creato
- Il **Backend** riceve la richiesta e il token, e controlla che questo sia corretto (verifica il digest con la stessa chiave segreta con cui era stato cifrato nel Gateway)
- Se il token risulta corretto, l'utente può accedere al servizio richiesto. In caso contrario, viene generato un errore **HTTP 401 Unauthorized**



GATEWAY - ENDPOINT

Per identificare gli Endpoint sono presenti due **URL WebSocket**: uno relativo alla coda e l'altro relativo agli utenti presenti all'interno dell'applicazione:

- **QueueSocketHandler** (`ws://<IP>:<PORT>/gateway/queue`)
- **ConcurrentUsersSocketHandler** (`ws://<IP>:<PORT>/gateway/concurrent-users`)



FRONTEND

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



Seguendo il workflow tipico dello sviluppo di un'applicazione Angular, il frontend è stato suddiviso in diversi componenti: ogni componente consiste in una cartella contenente:

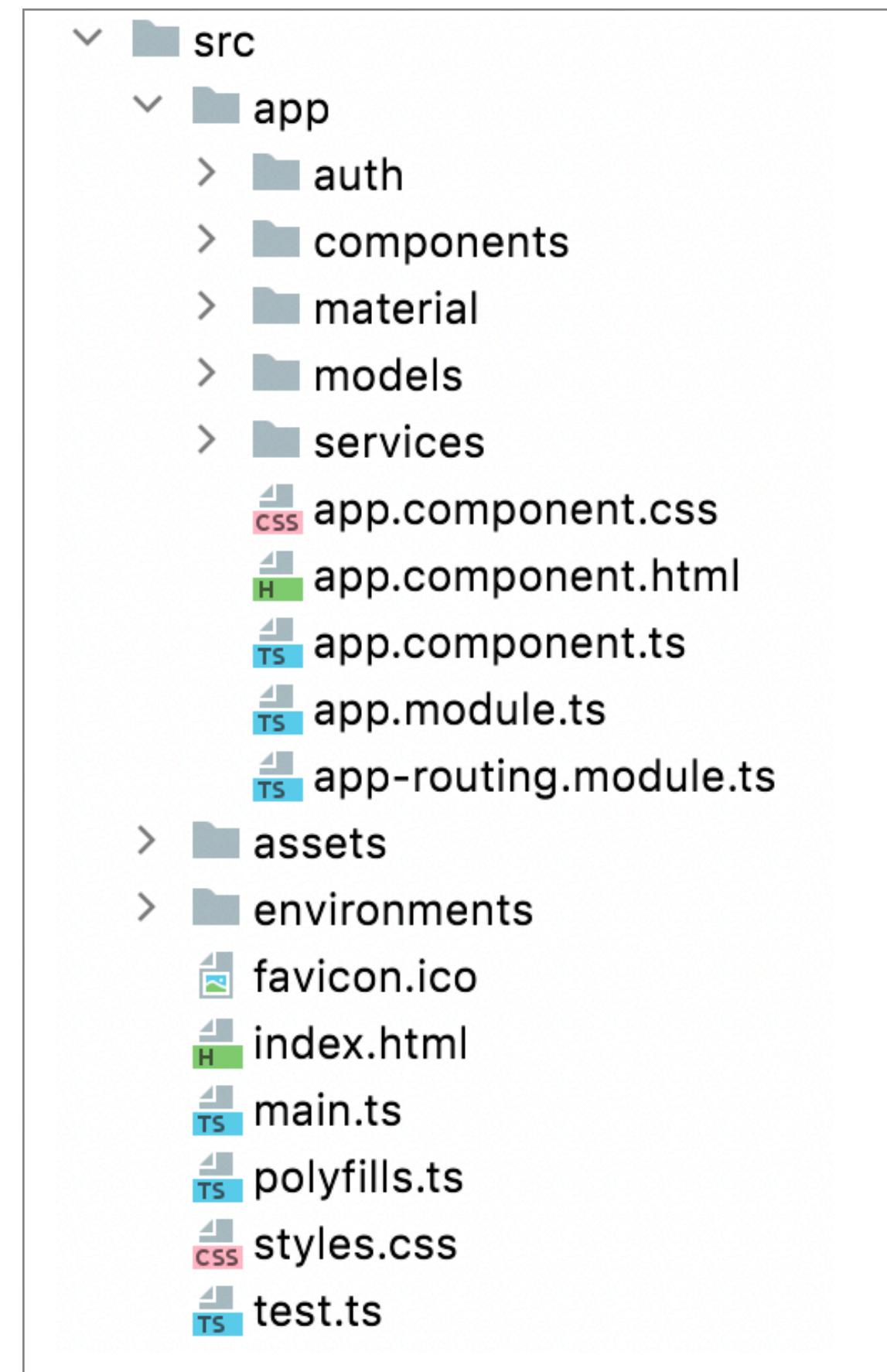
- un **file HTML** che ne definisce la vista;
- un **file CSS** per lo stile;
- un **file TS** che ne definisce il comportamento.

Il linguaggio di base è TypeScript. L'applicazione è stata realizzata come una **Single Page Application** (SPA), orientata ai servizi e con paradigma **Model-View-Controller**.



FRONTEND - ELEMENTI PRINCIPALI

- **Modello**: stato del sistema, legato ai servizi REST del backend
- **Componente**: “controllori” del pattern MVC
- **Template**: documenti HTML che rappresentano le viste (view di MVC)
- **Property Binding**: permette di osservare un attributo di un componente e propagare i cambiamenti nella vista
- **Event Binding**: permette di associare un evento (es. click di un bottone) ad un metodo di un componente
- **Servizi**: classi che mettono a disposizione dei metodi riusabili in vari controller tramite dependency injection



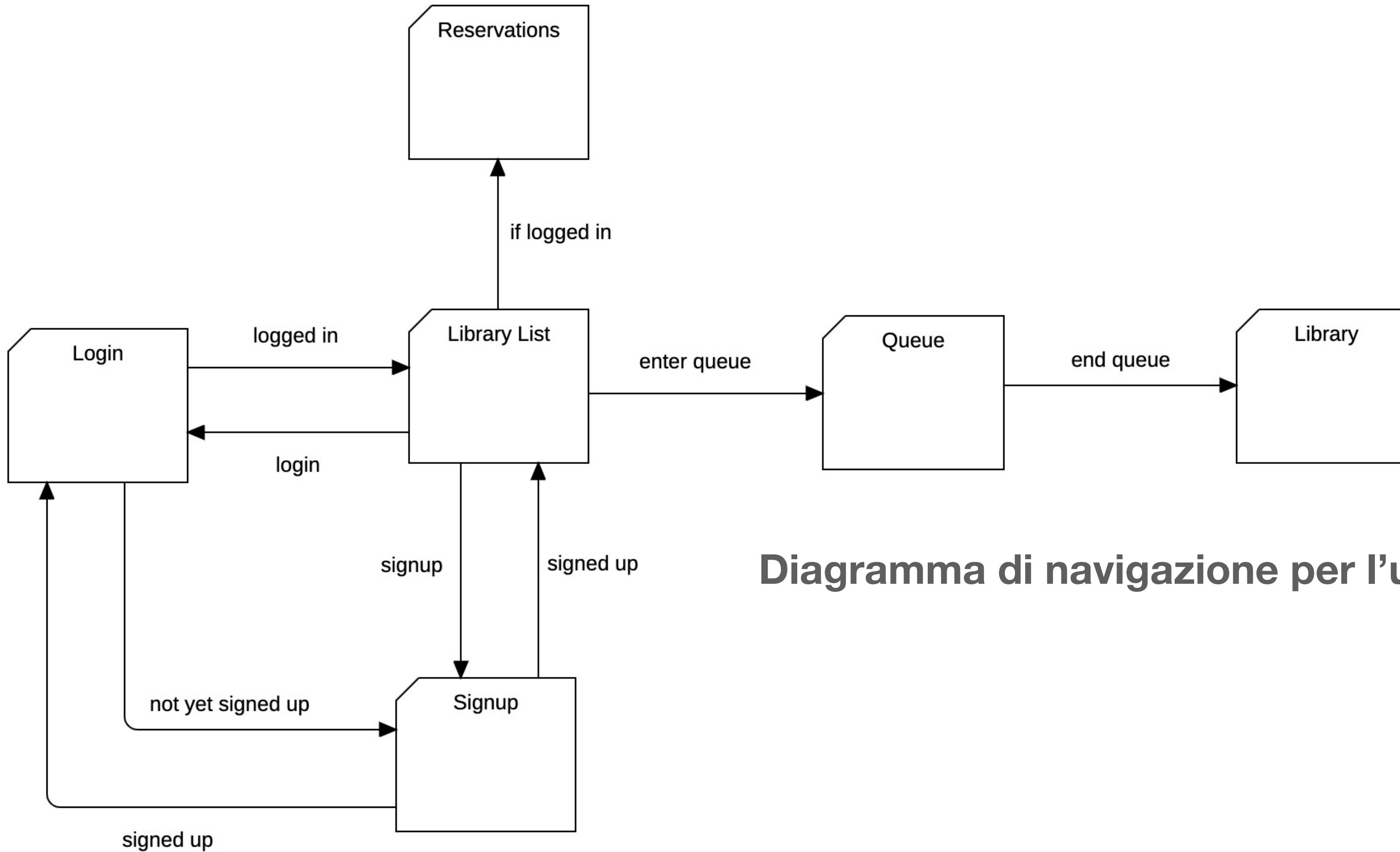
FRONTEND - DIPENDENZE

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)



FRONTEND - DIAGRAMMI DI NAVIGAZIONE (2/2)

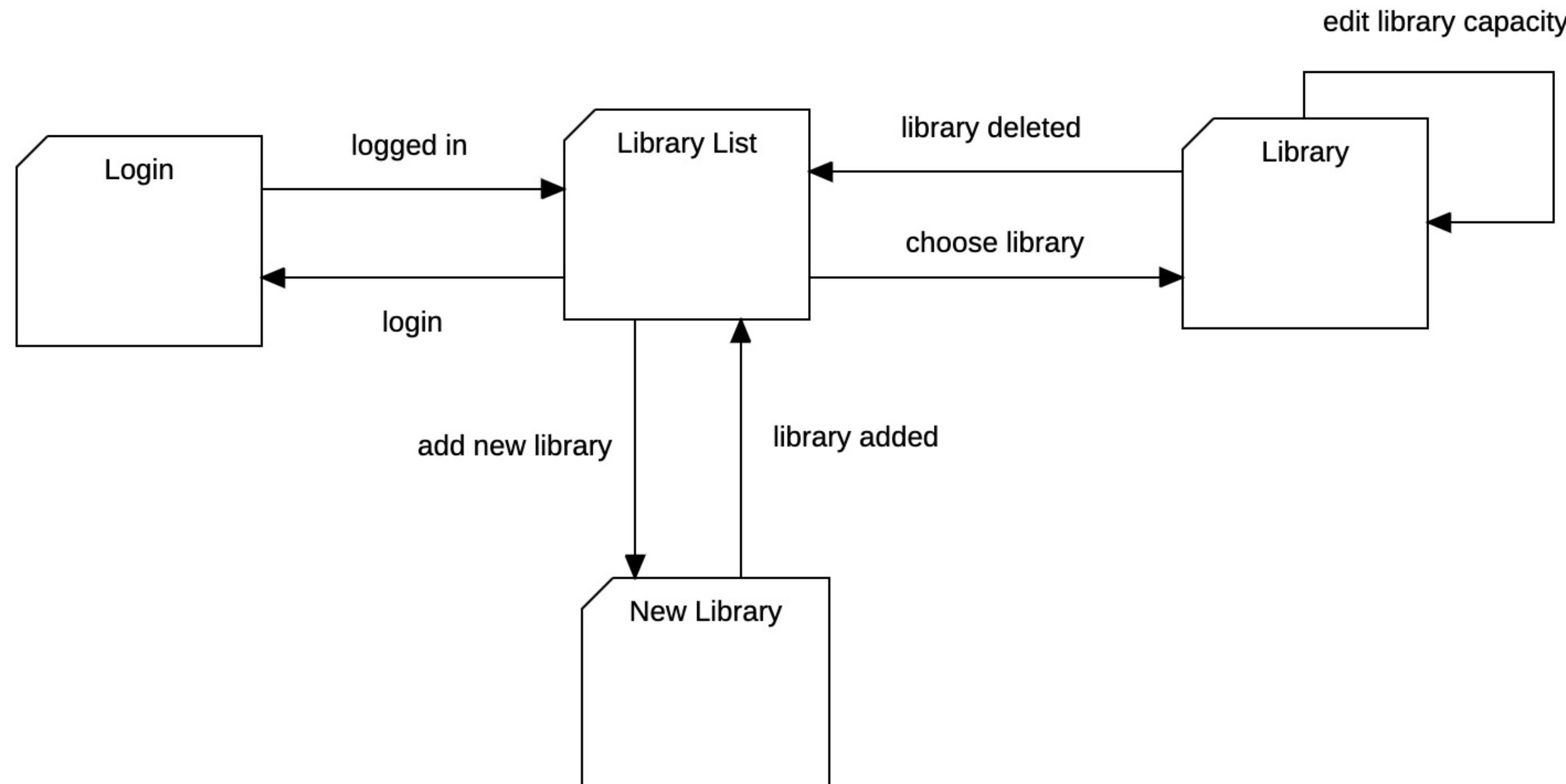
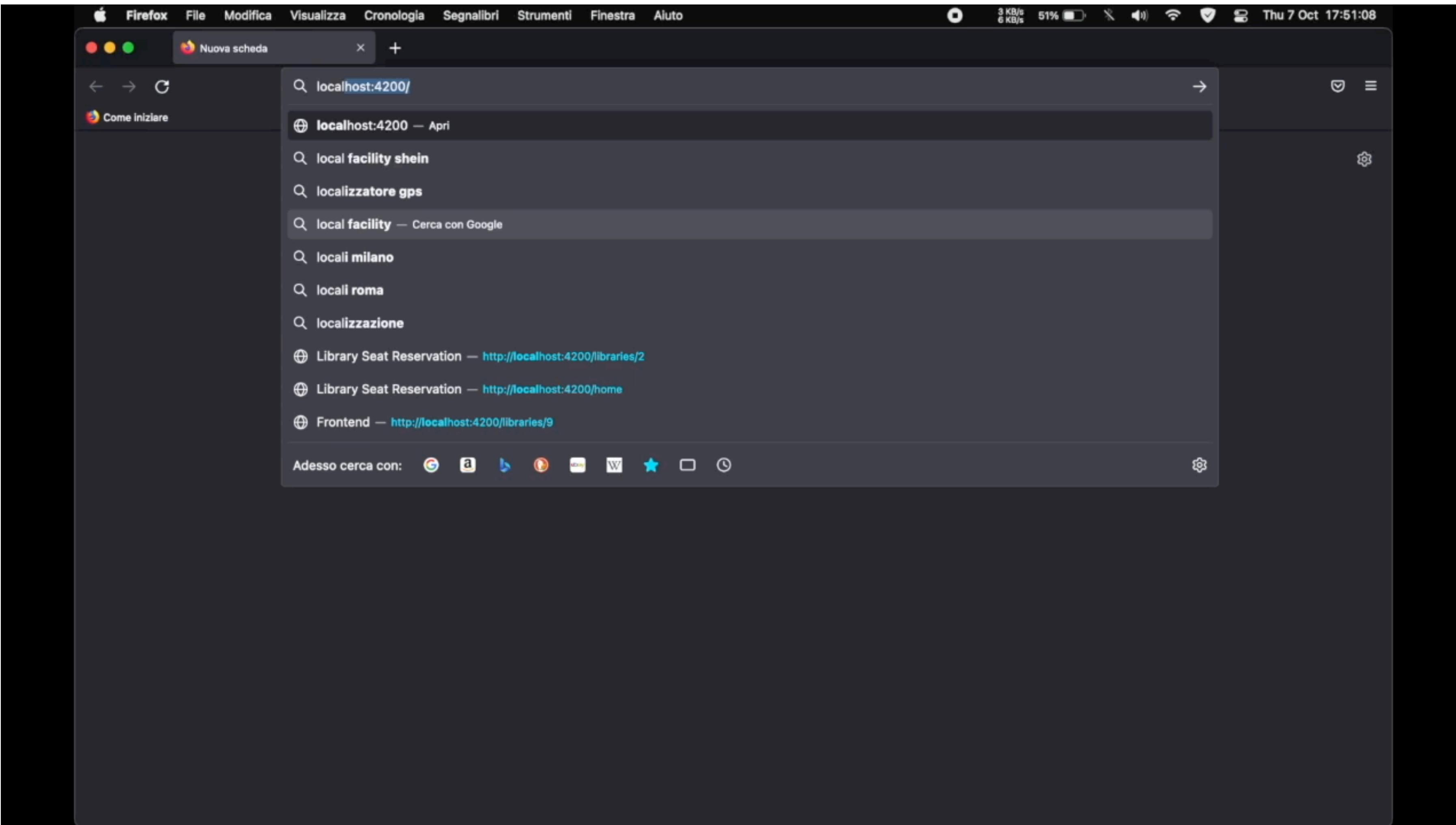
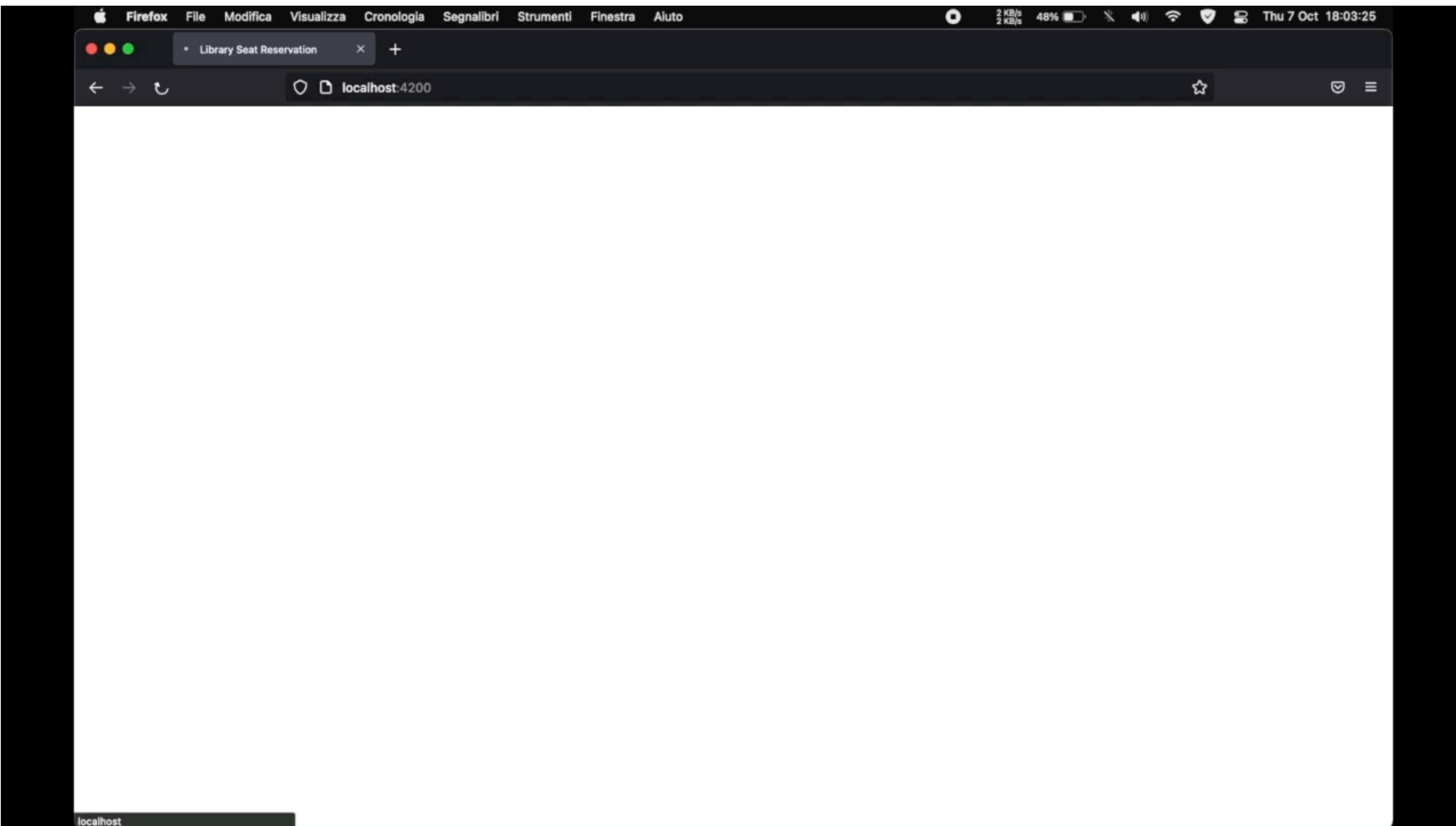


Diagramma di navigazione per l'admin

DEMO USER CON ENTRATA IN CODA



DEMO ADMIN (1/2)



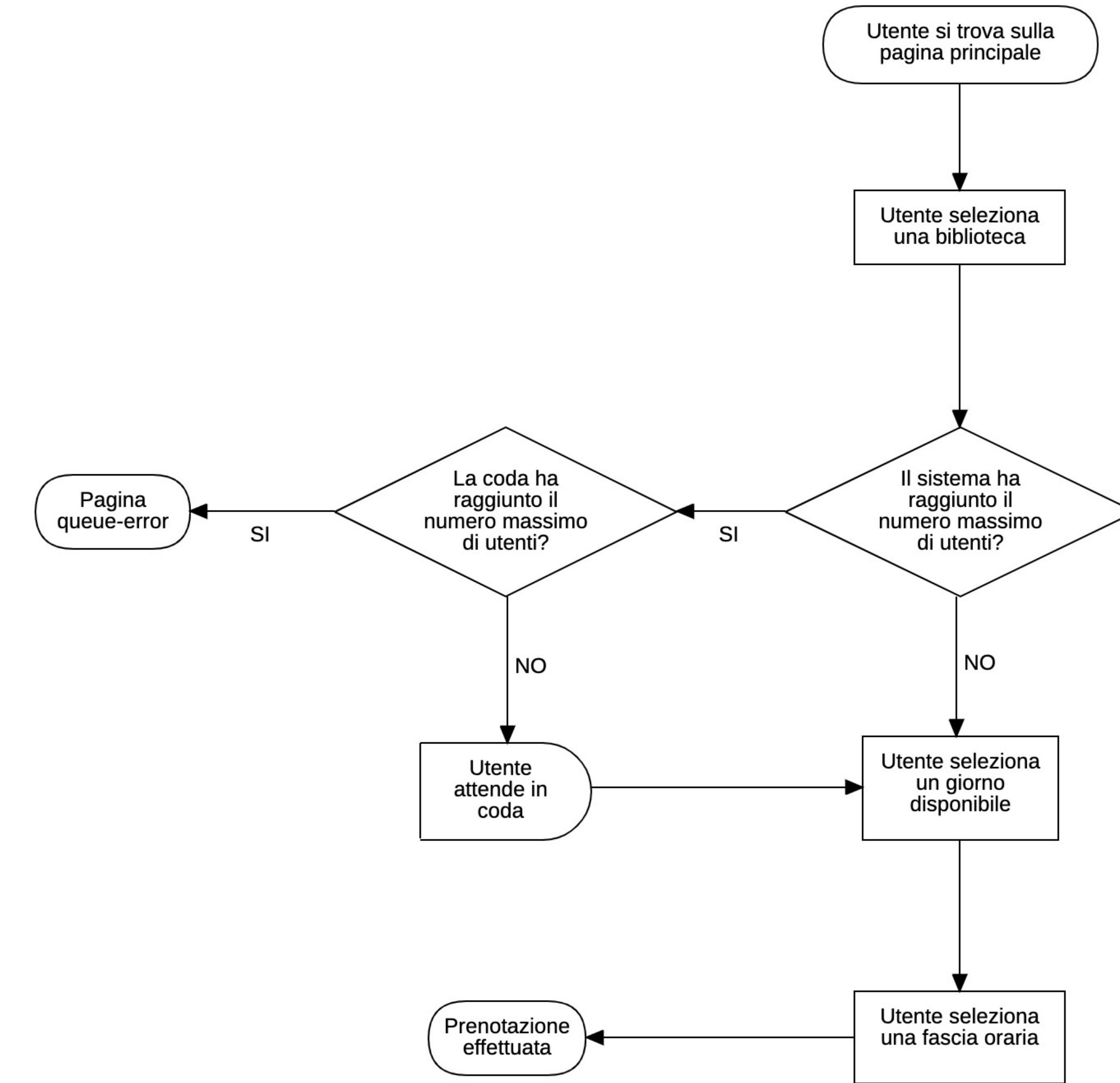
DEMO ADMIN (2/2)

The screenshot shows a Firefox browser window displaying the 'Library Seat Reservation' application. The title bar reads 'Firefox File Modifica Visualizza Cronologia Segnalibri Strumenti Finestra Aiuto' and the address bar shows 'localhost:4200/home'. The top right corner indicates the date 'Thu 7 Oct 18:05:53' and battery level '48%'. A user profile 'Utente Admin (Admin)' is visible in the top right. The main content area has a blue header with the text 'Prenota il tuo posto in biblioteca'. Below it is a search bar with the placeholder 'Cerca biblioteca...' and a magnifying glass icon. A grid of library cards is displayed:

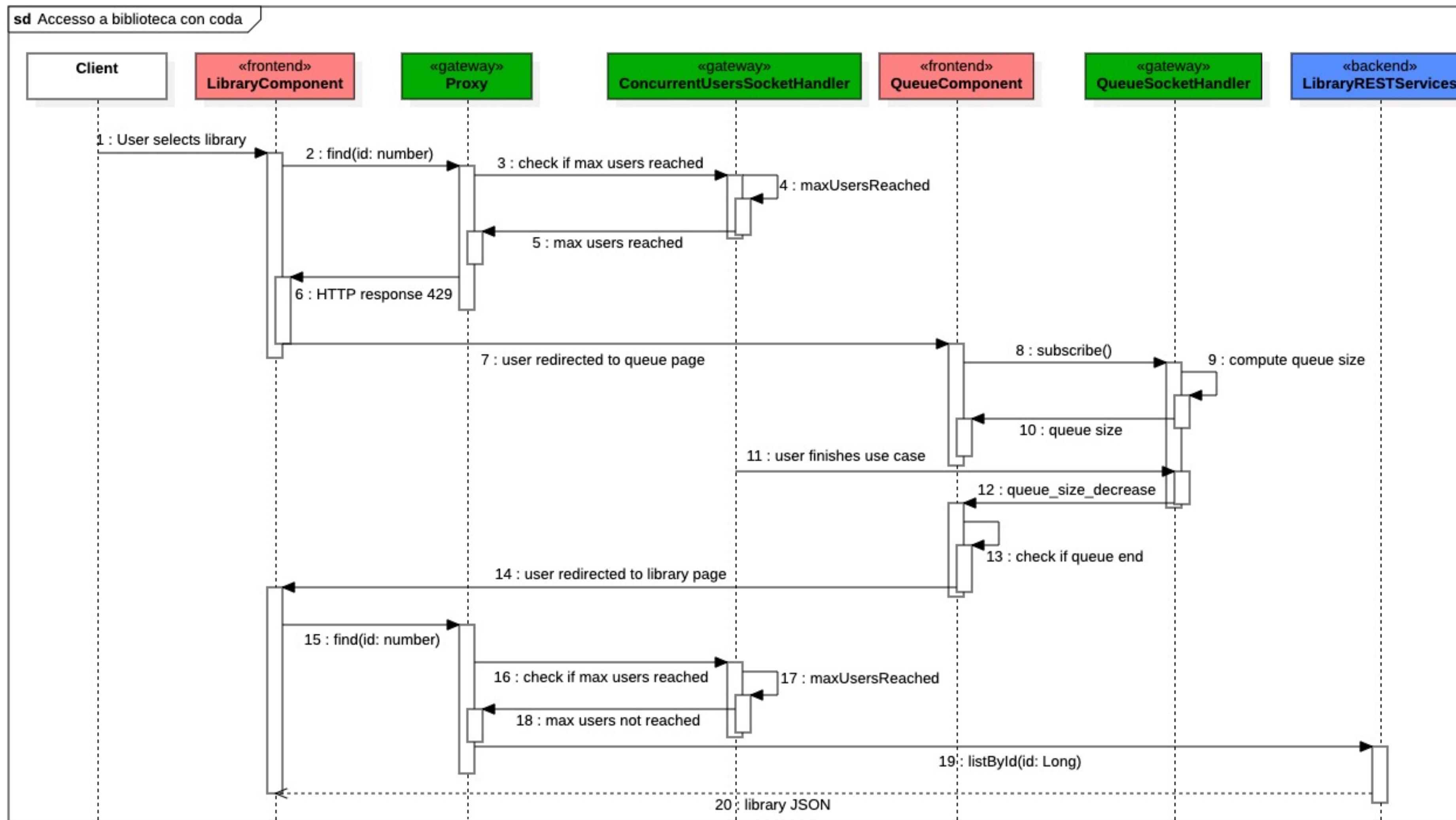
Biblioteca del Galluzzo	Biblioteca delle Oblate	Biblioteca Dino Pieraccioni	Biblioteca Fabrizio De André
Via Senese, 206, Firenze 	Via dell'Oriuolo, 24, Firenze 	Via Nicolodi, 2, Firenze 	Via delle Carra, 2, Firenze
Capacità: 70 posti	Capacità: 80 posti	Capacità: 60 posti	Capacità: 70 posti
Biblioteca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



RSOCKET

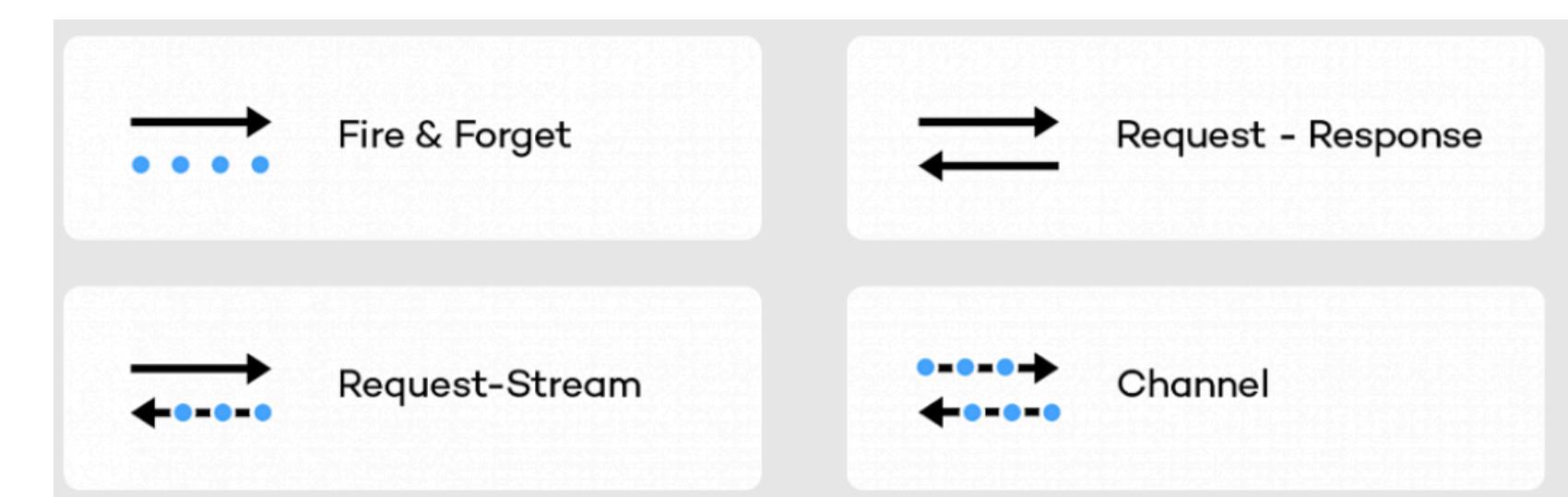
Protocollo di comunicazione bidirezionale, multiplex e duplex, in grado di inviare dati da un server a un client riutilizzando lo stesso canale di connessione.

- Protocollo a livello applicativo
- supporto a caratteristiche avanzate (*framing, session resumption e backpressure non bloccante*)
- agnostico rispetto al livello di trasporto utilizzato
- controllo del flusso e riduzione della latenza

The logo for RSocket, featuring the word "Socket" in a bold, magenta, sans-serif font. The letter "S" has a small black arrow pointing to the right integrated into its top curve.

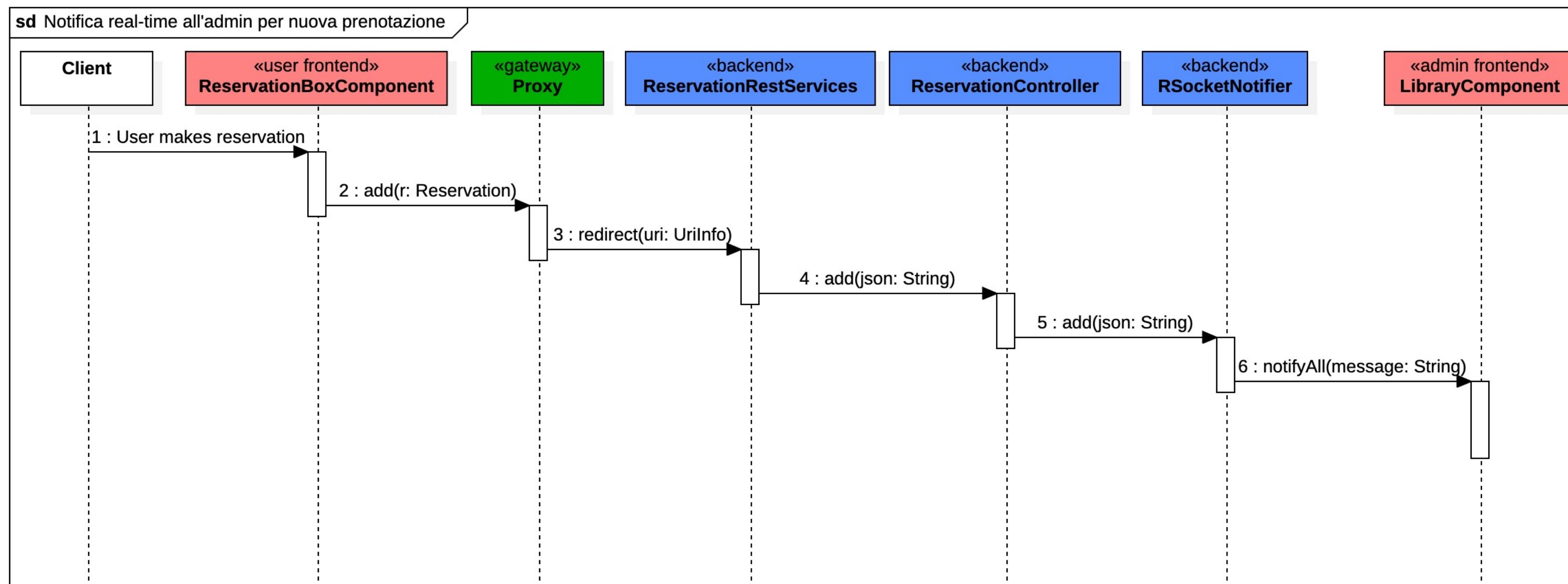
Supporta le seguenti interazioni:

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



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 page includes:

- A map of a city area with various streets labeled.
- A monthly calendar for October 2021, showing 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 10, 11, 12, 13, 17, 18, 19, 20, 23, 24, and 31 are red circles.
- A list titled 'Prenotazioni per Giovedì 21 Ottobre' (Reservations for Thursday 21 October) with two time slots: 'Mattina (8.00-13.00)' and 'Pomeriggio (13.00-19.00)'. It shows 29 reservations out of 30 available seats, with each entry including a name, email, and a delete icon.
- A modal for 'Modifica capacità biblioteca' (Modify library capacity) with a value of 30.
- A modal for 'Elimina biblioteca' (Delete library) with a warning message about deleting all reservations.



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.

- Il modulo **Gateway** potrebbe diventare un modulo **indipendente e universale**, configurabile e adattabile ai vari casi d'uso
- Estendere l'utilizzo ad una **regione più ampia** (es. Toscana)
- Aggiunta di **più fasce orarie** (e non solo mattina/pomeriggio)
- Offrire una prenotazione **specifica** per una determinata **aula** di una biblioteca (es. *Aula A della biblioteca Villa Bandini*).

