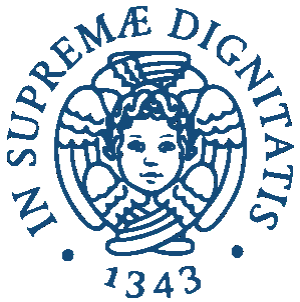

Distributed Systems and Middleware Technologies

Project Specifications Proposal

Jacopo Carlon , Nicola Riccardi

Academic Year 2024-2025

Contents

1 Project Specifications

CinemaBooking is a distributed web-app that allows cinemas to sell tickets for their shows. Registered customers can book any number of available seats for a show and cancel their bookings before a given deadline.

1.1 Use Cases

An *Unregistered User* can:

- Register to the service as a Customer or as a Cinema.

An *Unlogged User* can:

- Login to the service as a Customer or as a Cinema.

A *Logged Cinema* can:

- Logout;
- View list of current Shows;
- Create a new Show.

A *Logged Customer* can:

- Logout;
- View list of current Shows;
- Book some seats for a Show;
- View own booked seats;
- Free some of own booked seats.

The *System* must:

- Remember registered Cinemas and Customers;
- Remember created Shows details;
- Remember booked seats of each Customer;
- Generate a unique history of Customer bookings for each Show;
- Synchronize available seats and history of Customer bookings, for each bookable Show;

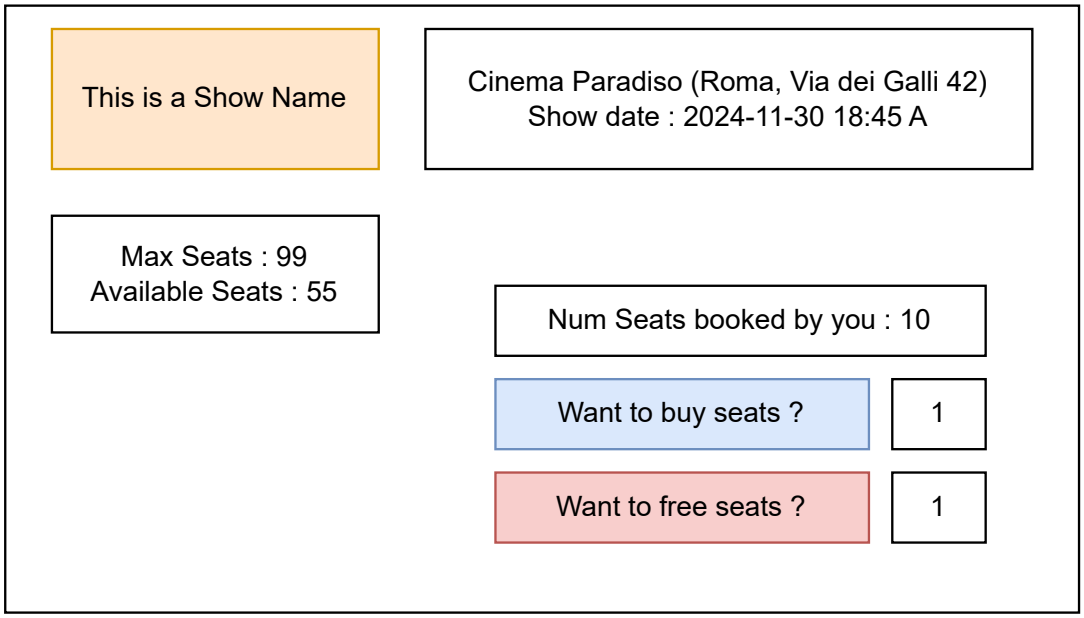

Figure 0: Mock-up of page a Costumer sees regarding a Show that is still bookable

1.2 Synchronization and Communication Issues

On the application we will face the following synchronization and communication issues:

- Client nodes need to be synchronized with the same booking history for each Show and the same list of bookable Shows.
- In case a Customer makes (or cancels) a valid booking for a Show, the server will be in charge of communicating to other clients nodes the updates on the history and available seats for that Show.
- In case a Cinema creates a new Show, the server will be in charge of communicating to other clients nodes the newly created Show.

1.3 Design Ideas

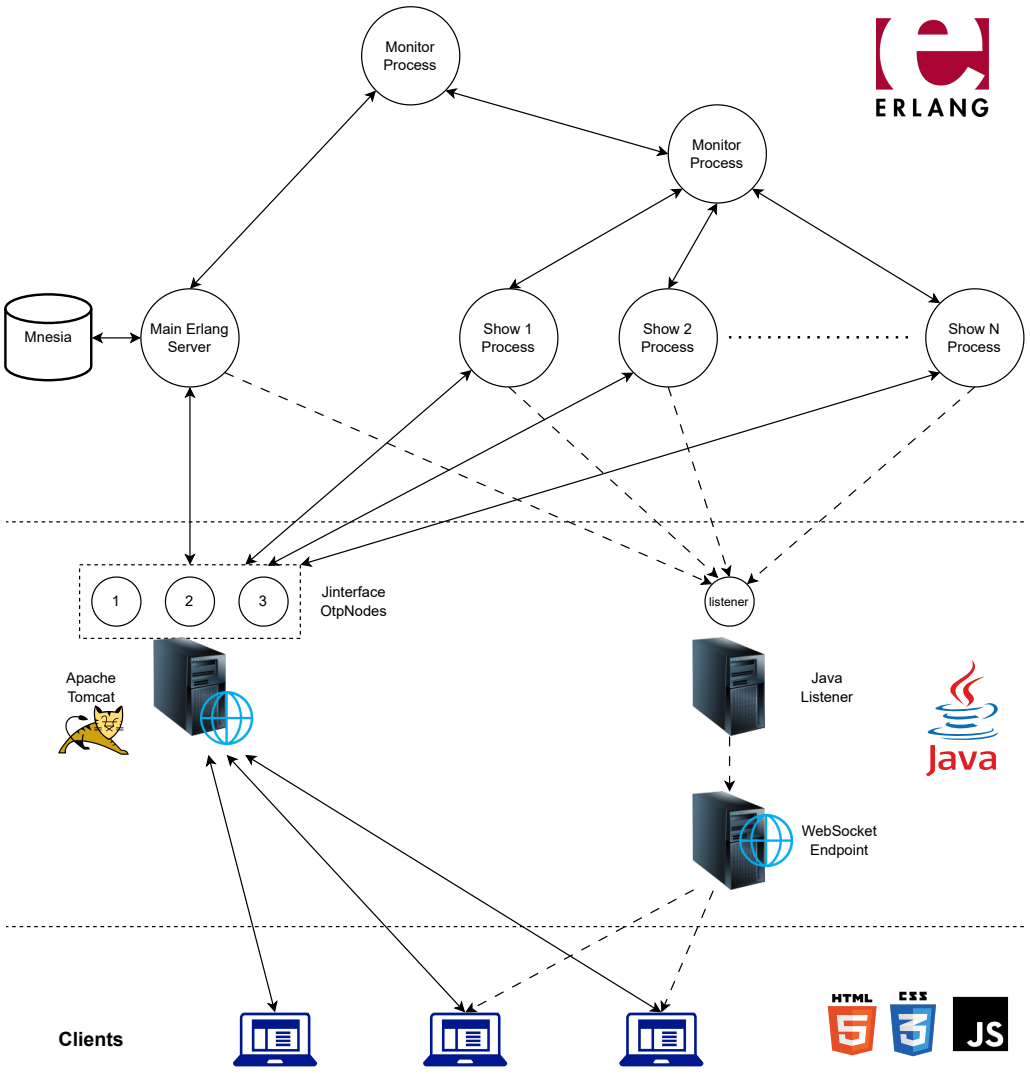

Figure 1: Graphical representation of system architecture

Our proposal for the implementation of this system is as follows:

- **Client Nodes**: User Interface in HTML/CSS, generated with Java Servlets and JSP. For each client node, the web server will spawn a dedicated Erlang node for the communication with the main Erlang server.

- **Server Node** Made in Enrlang. The server will perform persistent data storage and handle communication and synchronization between client nodes. For each Show, a process is spawned in the local node.