UNIVERSITY OF PISA

MSc in Computer Engineering

Project for Distributed Systems and Middleware Technologies

# Distributed Betting System

Professor:

**Prof. Alessio Bechini**

Students:

**Francesco Barcherini (645413)**

**Andrea Di Matteo (641388)**

**Francesco Scarrone (645328)**

ACADEMIC YEAR 2025/2026

# 1 Project Proposal

## 1.1 Requirements

The project implements a website for online betting, both on real events (such as football matches) and virtual ones (such as virtual roulette spins). The quoting system is dynamic, and the betting odds are updated based on other users' bets (the less an option is chosen by previous users, the higher the odds will be).

To avoid significant financial losses for the website, each bet has a maximum cap determined by the total amount currently wagered and its distribution.

The actors are:

- **Unregistered user**: can view the available bets and their current odds.
- **Registered user**: can place bets on available events, fixing the bet at the current dynamic odds (the payout is calculated based on the odds at that moment).
- **Administrator**: can add events, stop them, insert the outcomes of real events, and start the execution of virtual spins. For testing purposes, the administrator can also set a user's balance (in a real scenario, this would be updated through a payment method such as a credit card).

The main coordination challenges of the system are:

- since odds are dynamic, every placed bet must trigger an update of the odds and of the maximum cap, and these updates must be propagated live to all connected clients;
- nodes must be informed of the presence of new betting events (e.g., a newly scheduled match or a new virtual roulette session created by the Administrator), so that all nodes expose the same list of available events;
- once concluded, the system must share the outcome of the event among nodes to process payouts and close the event;
- authentication and betting are handled by different services, therefore the application should use a token-based system to grant access to the game services after login;
- each client must see the same information about balance and bets upon subsequent sessions and logins, regardless of which node serves the request.

## 1.2 Architecture

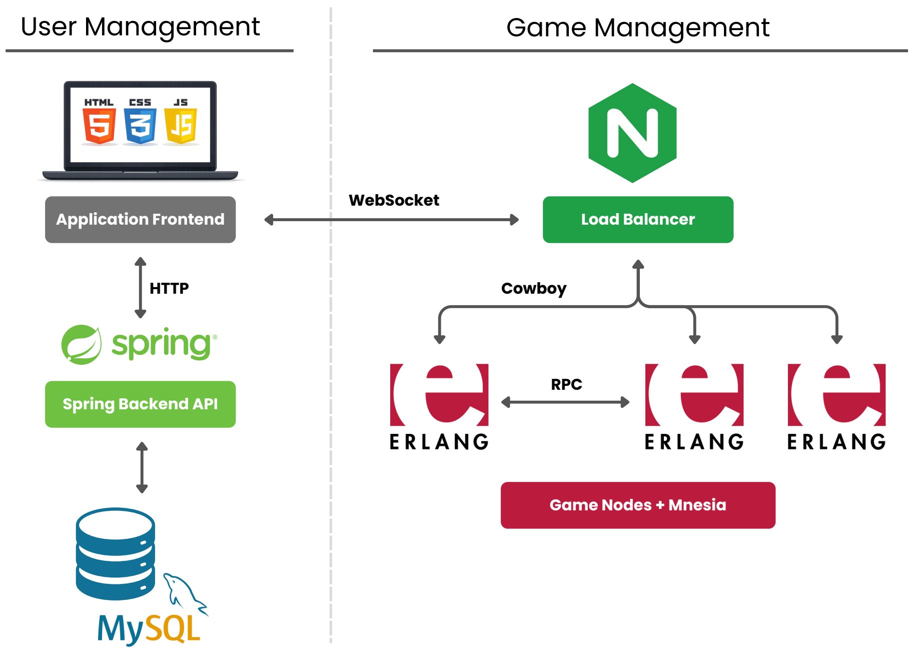

Figure 1: Architecture of the overall system

- A **Spring Boot** server handles user management operations, such as login, logout, and the registration of new users. User data is stored in a MySQL database. Additionally, the Spring Application is responsible for serving the main Web Application to the client.
- A simple front-end, implemented in **HTML**, **CSS**, and **JS**, serves as the primary interface for the User Application. This application also manages the connection and authentication with the game back-end hosted on the Erlang Nodes through **WebSocket**.
- The **Erlang Nodes** constitute the main game back-end where each node hosts a single betting game. **Mnesia** is used as the primary distributed data storage. Communication among nodes through **RPC** ensures the propagation of critical notifications, such as control operations, and placed bets in order to update the odds. Finally, load balancing is managed by an **Nginx server**.