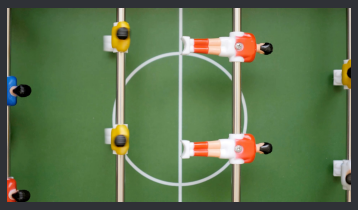
Football central.

A match reservation system

Distributed Systems and Middleware Technologies

Gianluca Serao - Lorenzo Barigliano



1

What is Football central?

- Facilitate the organization of 5v5 football matches by letting players post games to find other players **quickly** and **effortlessly**.
- Let players find games to join.
- Players can interact by **posting messages** on the game dashboard to agree on shirt colours, teams, etc.

2

Why football central?

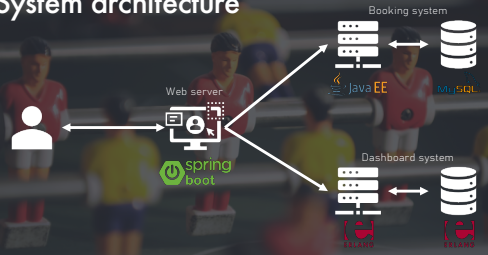
This kind of system requires a particular focus on resources management. It should:

- Manage **concurrent operations** (read/write) on resources
- Avoid **overbooking**
- Manage correctly **delete operations**

In general, it should be free of concurrency and synchronization problems.

3

System architecture



4

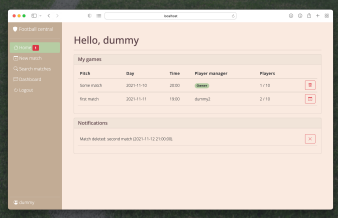
Web server

- Handles clients requests with Spring controllers
- Forwards data access to Spring components (EJBs)
  - One manages data access through JPA
  - One manages a connection with an Erlang server
- Spring container manages multiple clients
- Provides authentication service with Spring security

5

User interface

- Handles clients requests with Spring controllers
- Technologies: Thymeleaf template resolver (HTML5), Bootstrap (CSS3) and JQuery
- Spring framework (MVC)



6

Booking system

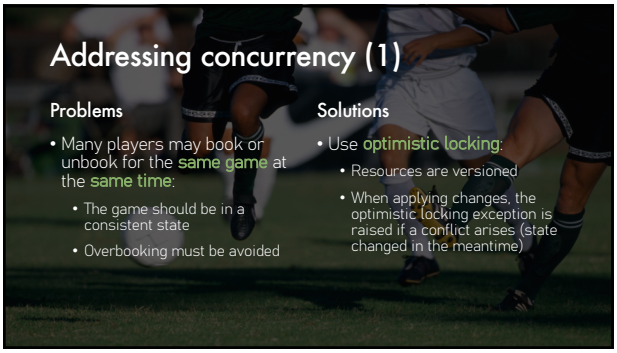
- Implemented with Spring service (CDI)
- Maintains data about games and players
- Data are stored on MySQL and accessed through Spring Data JPA
  - Manages concurrent access with optimistic locking (resource versioning)
  - A scheduled task (method) removes old games

7

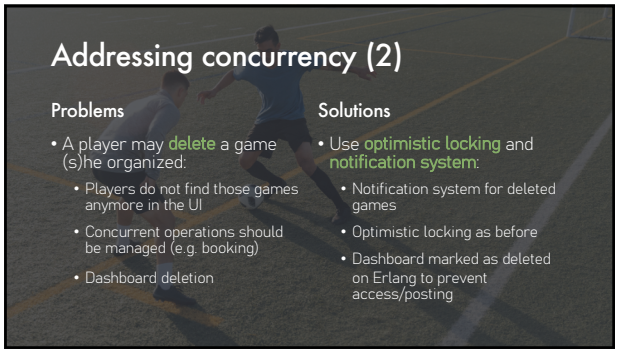
Dashboard system

- Implemented in Erlang with gen_server behaviour
- Maintains data about dashboards (messages)
- Data are stored on Mnesia DB and accessed through the Mnesia library
  - Tables are of type bag (faster dashboard read) and persisted on disk
  - A garbage collector (process) removes deleted dashboards

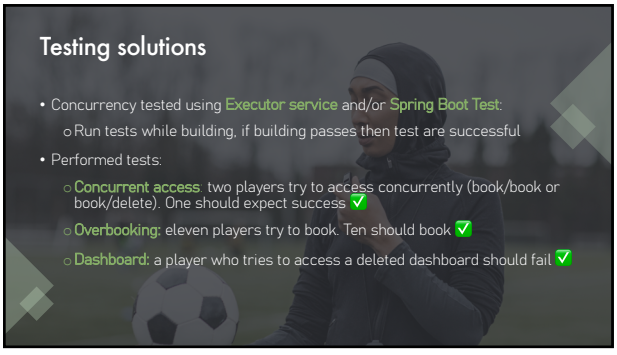
8


Addressing concurrency (1)
Problems
• Many players may book or unbook for the same game at the same time:
• The game should be in a consistent state
• Overbooking must be avoided
Solutions
• Use optimistic locking:
• Resources are versioned
• When applying changes, the optimistic locking exception is raised if a conflict arises (state changed in the meantime)

9


Addressing concurrency (2)
Problems
• A player may delete a game (s)he organized:
• Players do not find those games anymore in the UI
• Concurrent operations should be managed (e.g. booking)
• Dashboard deletion
Solutions
• Use optimistic locking and notification system:
• Notification system for deleted games
• Optimistic locking as before
• Dashboard marked as deleted on Erlang to prevent access/posting

10


Testing solutions
• Concurrency tested using Executor service and/or Spring Boot Test:
o Run tests while building, if building passes then test are successful
• Performed tests:
o Concurrent access: two players try to access concurrently (book/book or book/delete). One should expect success ✅
o Overbooking: eleven players try to book. Ten should book ✅
o Dashboard: a player who tries to access a deleted dashboard should fail ✅

11