Relay Pong

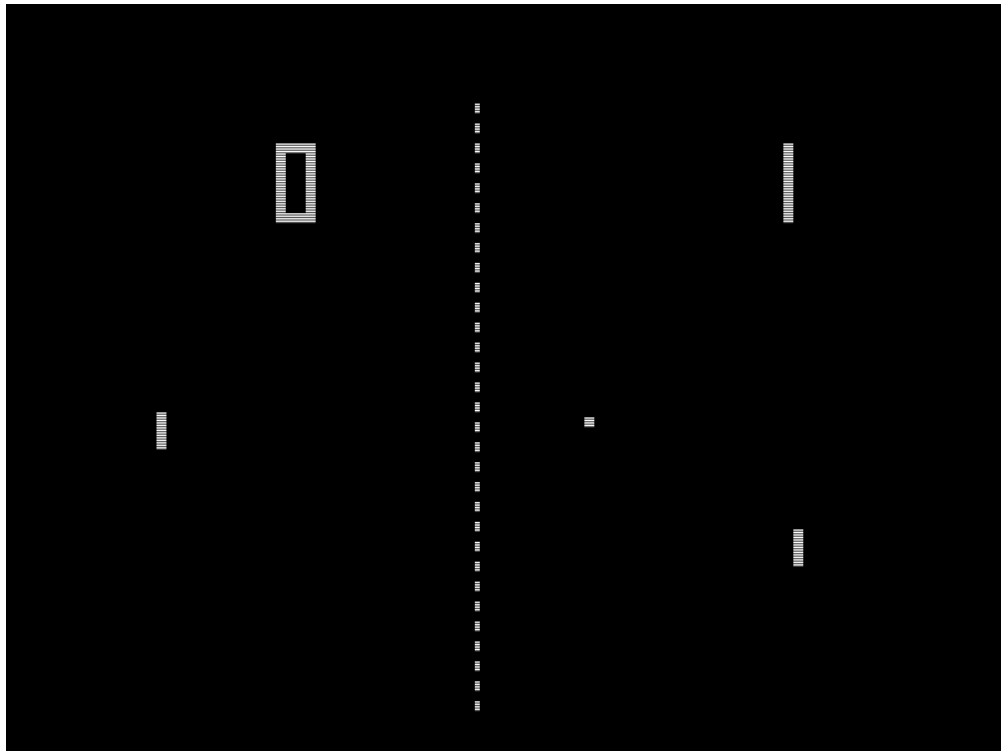
PROJECT FOR THE COURSE OF **DISTRIBUTED SYSTEMS**

Jahrim Gabriele Cesario (0001031535) – jahrim.cesario2@studio.unibo.it

Introduction

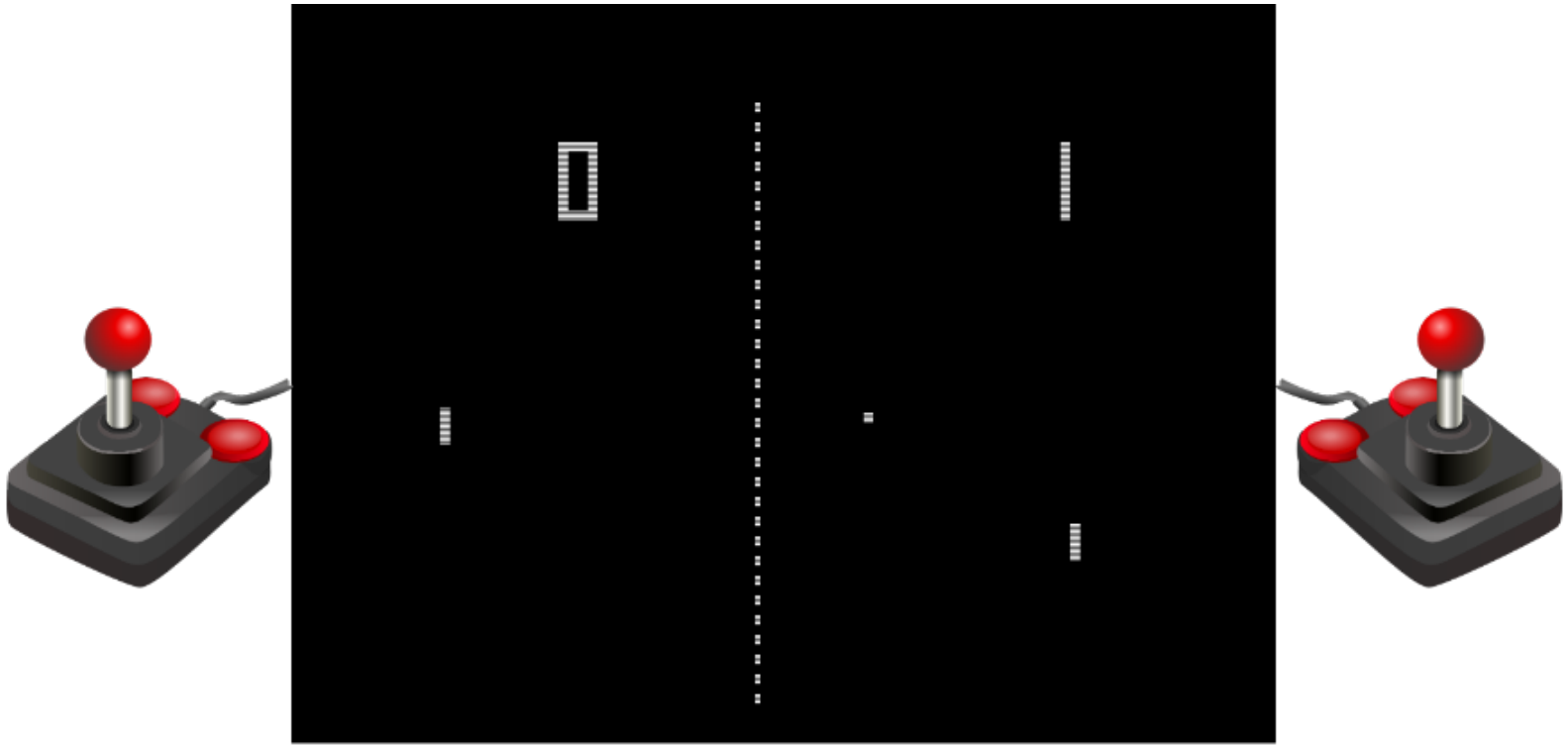
Pong

Pong is the first game developed by Atari, released in 1972.



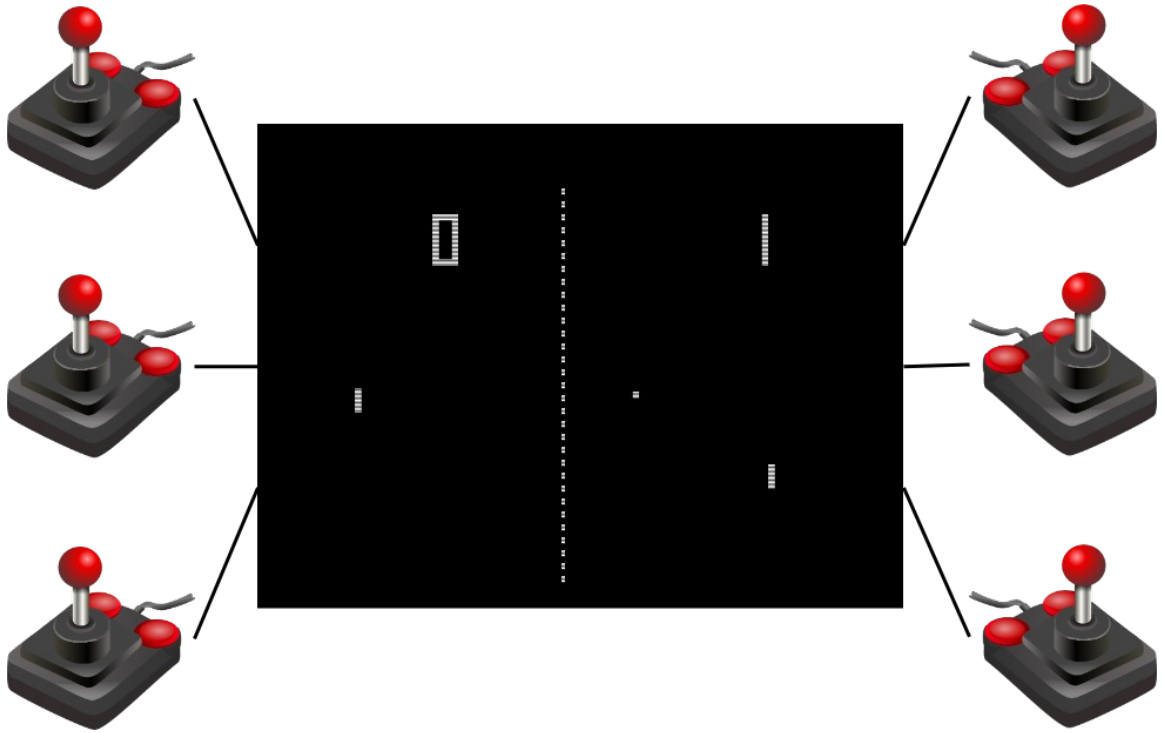
Pong

The game is a **simulation of table tennis**, where **two players** compete against each other, trying to score 11 points before the opponent.



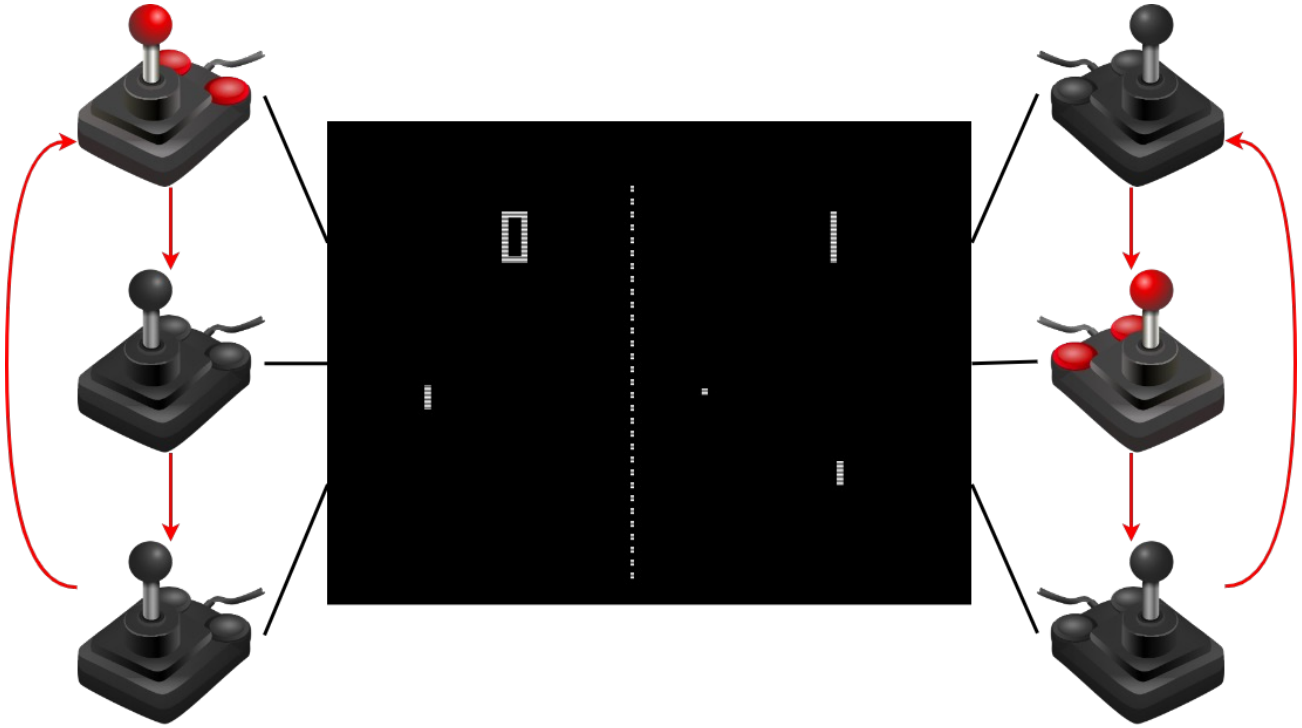
Concept

The idea of this project is to expose an extension of the game through a **web application**. The extension will allow an **indefinite number of players** to join the same game.



Concept

The simplest way to implement such system is to split the players in **two teams**, having the players controlling the racket of their team **in turns**.



Design

Strategy

The system has been designed adopting the following patterns and principles:

- **Service-Oriented Architecture**:
  strictly defining the roles of each component of the system;

- **Outside-In Design**:
  starting from the user's needs, going towards the business logic;

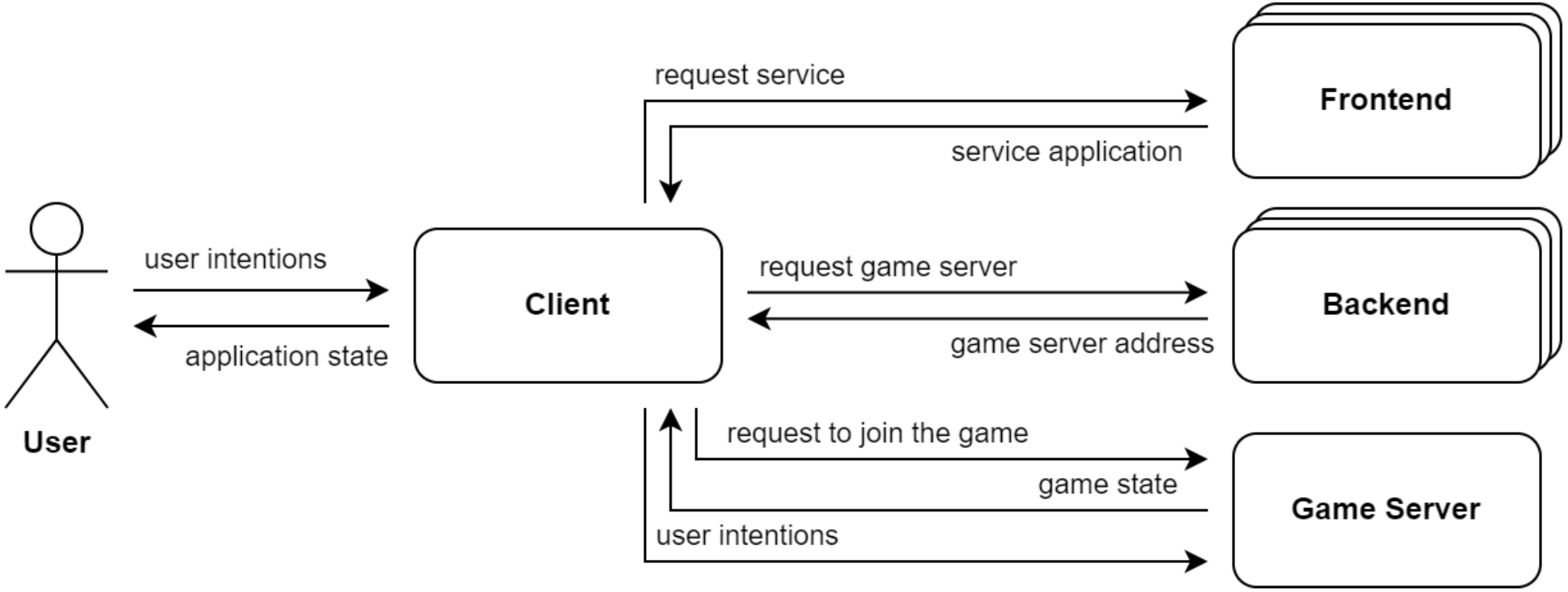
- **Generalization**:
  leveraging different levels of abstractions to describe the same system;

- **KISS (Keep It Simple, Silly)**:
  trying to adopt the simplest solutions when possible;

- **Anticipation of Change**:
  trying to adopt malleable solutions.

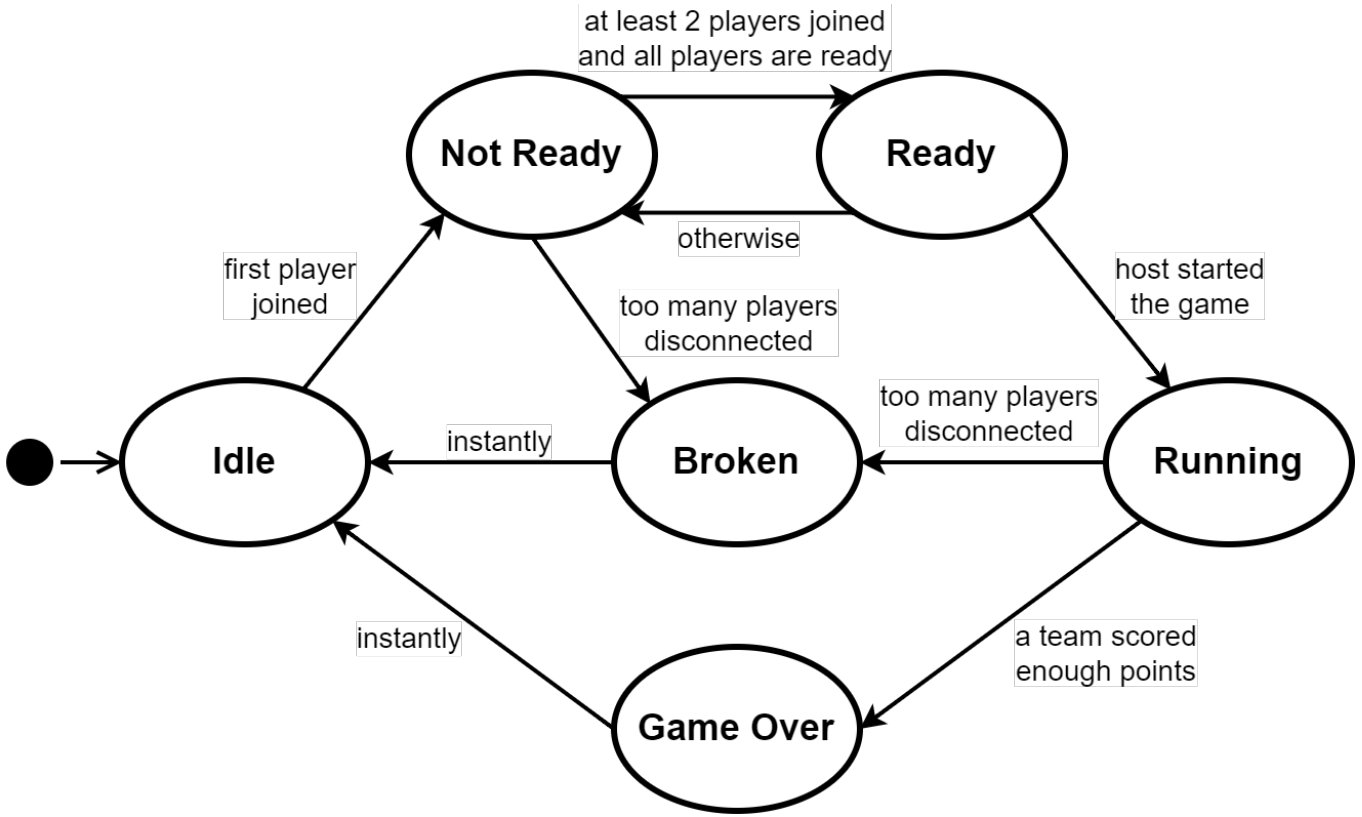
Architectural Design

The system is composed by four main components, whose interactions are showed below.



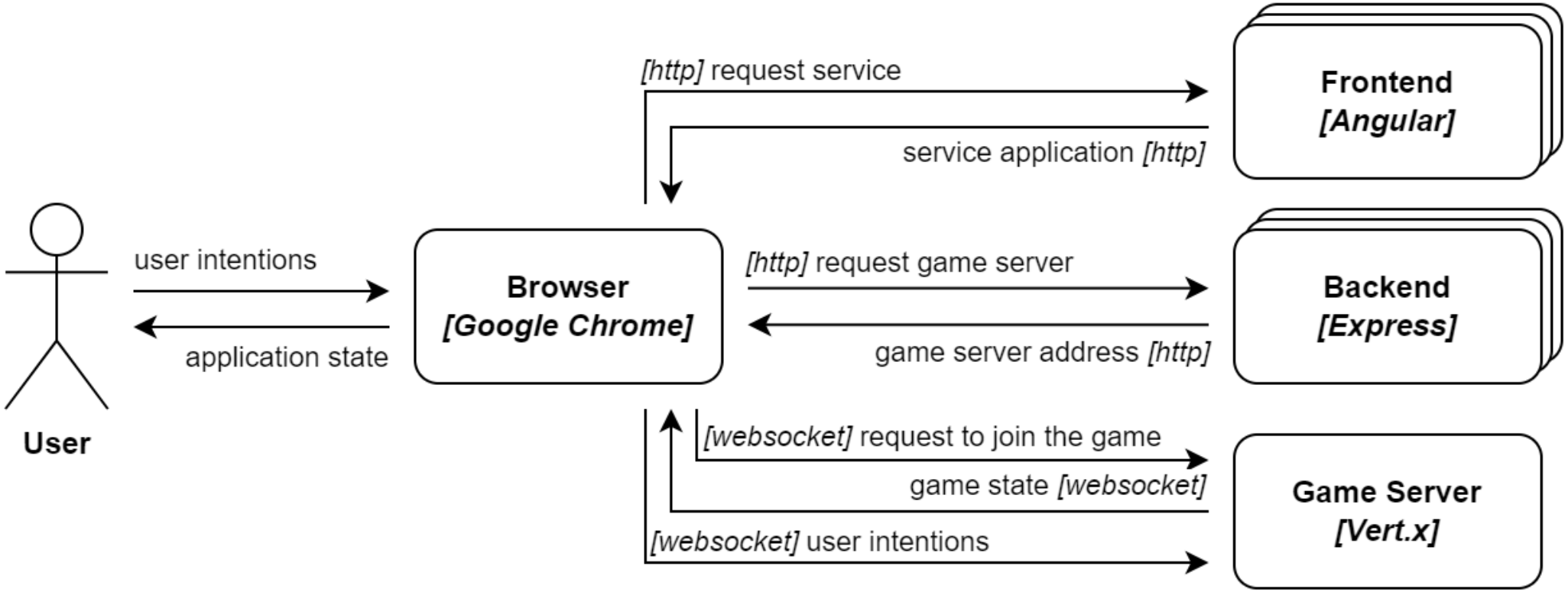
Zoom In: Game Server Lifecycle

After a user connects to a game server, the game server develops through the following lifecycle.



Detailed Design

The system was implemented through the following technologies and communication protocols.



Zoom In: Websockets

Websockets are a core concept of this project, since they enable **real-time communication** between the players and the game servers.

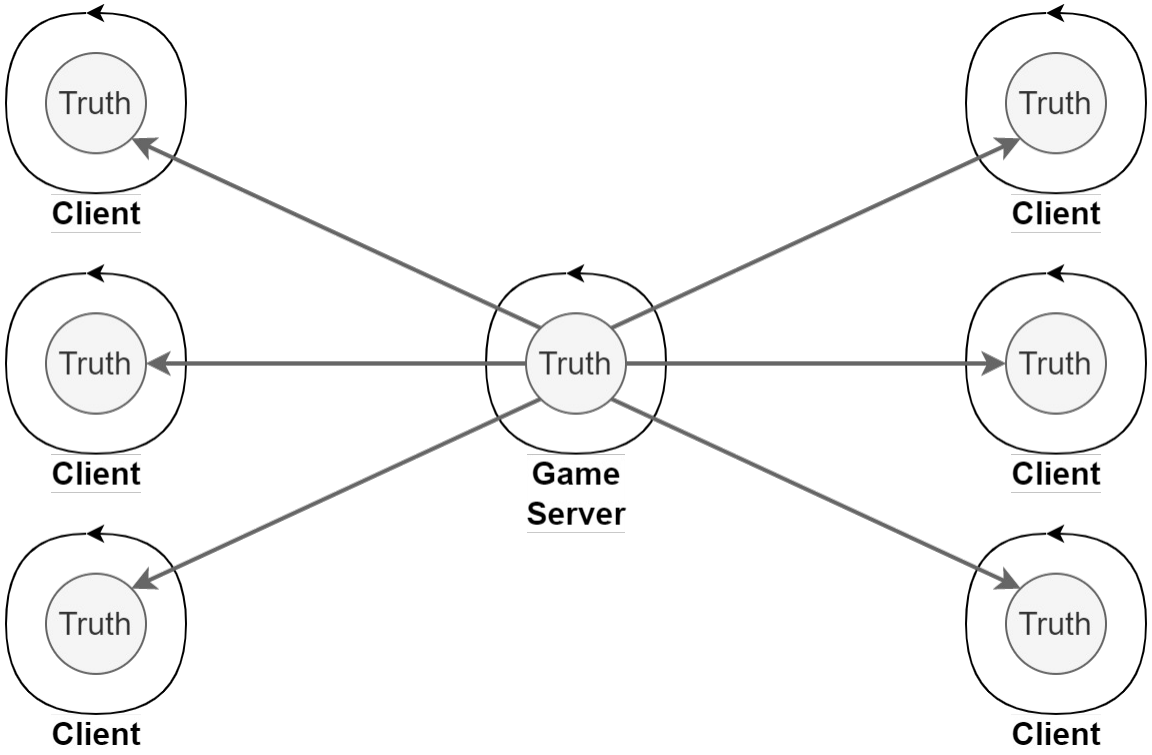
Motivations:

- **Supported by browsers**, which are the best type of clients for exposing an application to a vast audience;
- **Low latency communication**;
- **Push interaction**: the server can send the state of the game to the clients without polling.

Implementation

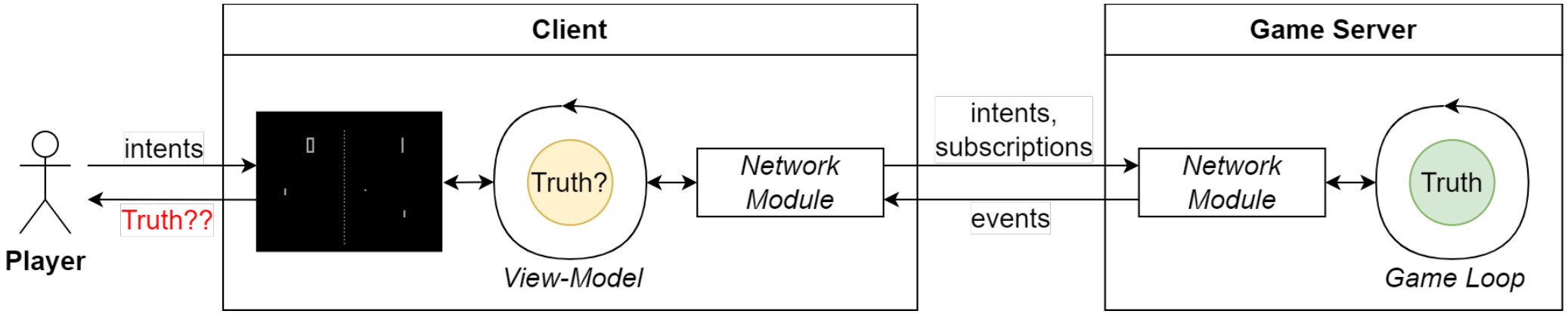
Zoom In: Game State Distribution

The game server manages the **real state of the game**, distributing it to the players as the game develops.

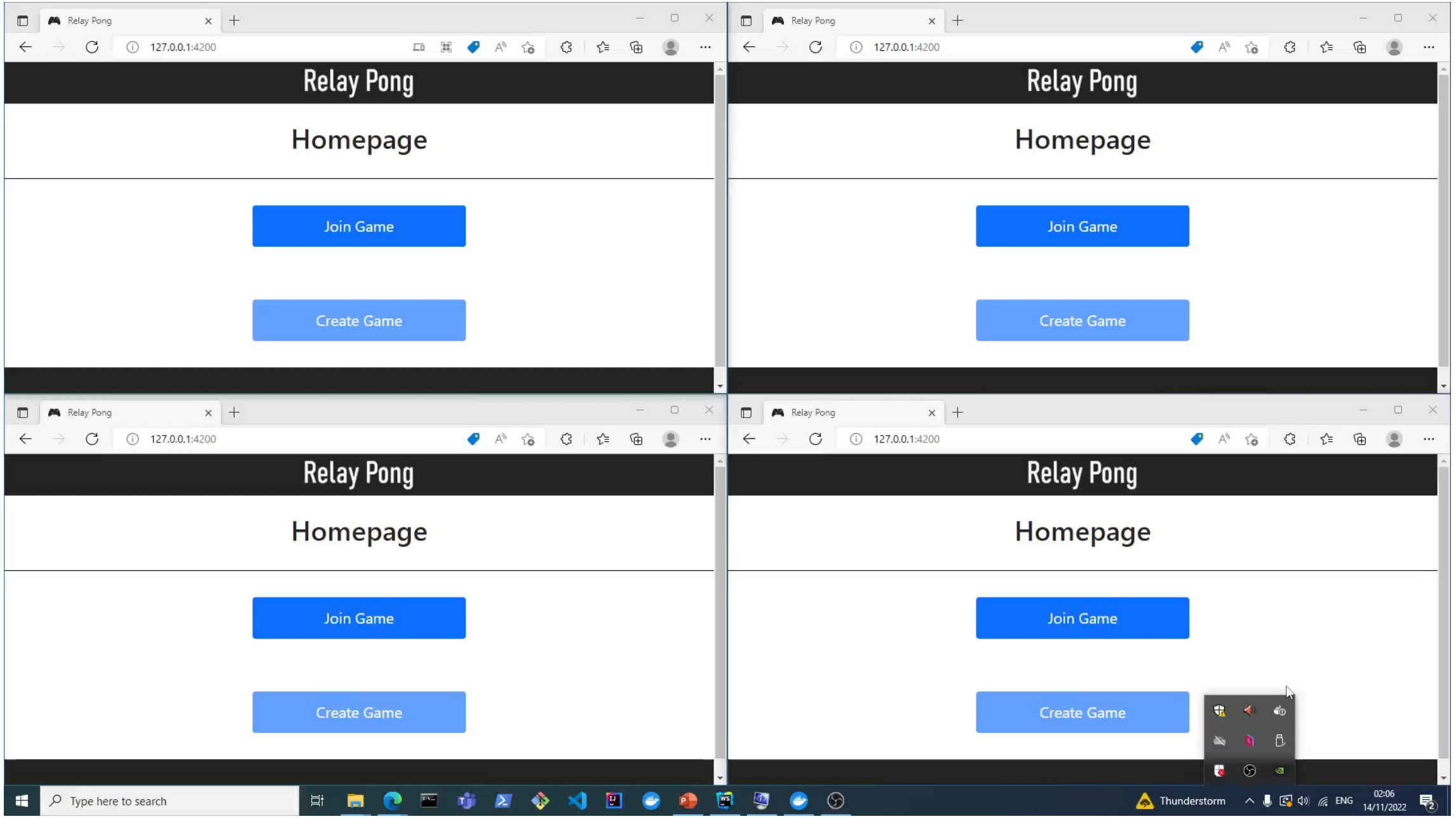


Zoom In: Game State Distribution

Due to the distributed nature of the system, **the state of the game perceived by the clients can differ from that of the game server**, but **eventually they should become the same** as the server keeps pushing updates to the clients.



Demo


Relay Pong
127.0.0.1:4200
Relay Pong
Homepage
Join Game
Create Game
Relay Pong
127.0.0.1:4200
Relay Pong
Homepage
Join Game
Create Game
Relay Pong
127.0.0.1:4200
Relay Pong
Homepage
Join Game
Create Game
Relay Pong
127.0.0.1:4200
Relay Pong
Homepage
Join Game
Create Game
Type here to search
Thunderstorm
ENG
02:06
14/11/2022

Conclusions

Acquired Knowledge

During the development of this project, I have increased my competence in the following topics:

- **Design of service-oriented architectures**;
- **Real time communication on the web**;
- **Technologies**:
  - Protocols: HTTP (REST), Websocket;
  - Frameworks: Angular, Vert.x, Express;
  - Build Automation: gradle, npm;
  - Deployment: Docker.

Considerations

The prototype of the system gave better results than expected. Although, many improvements can still be applied, such as:

- **Optimization of network resources**;

- **Backpressure mechanisms**, in order to prevent clients from being overwhelmed;

- **Redesign of some minor software components**.

Also, the prototype includes only a few of the requirements that should be satisfied by the finalized system.

Thanks!