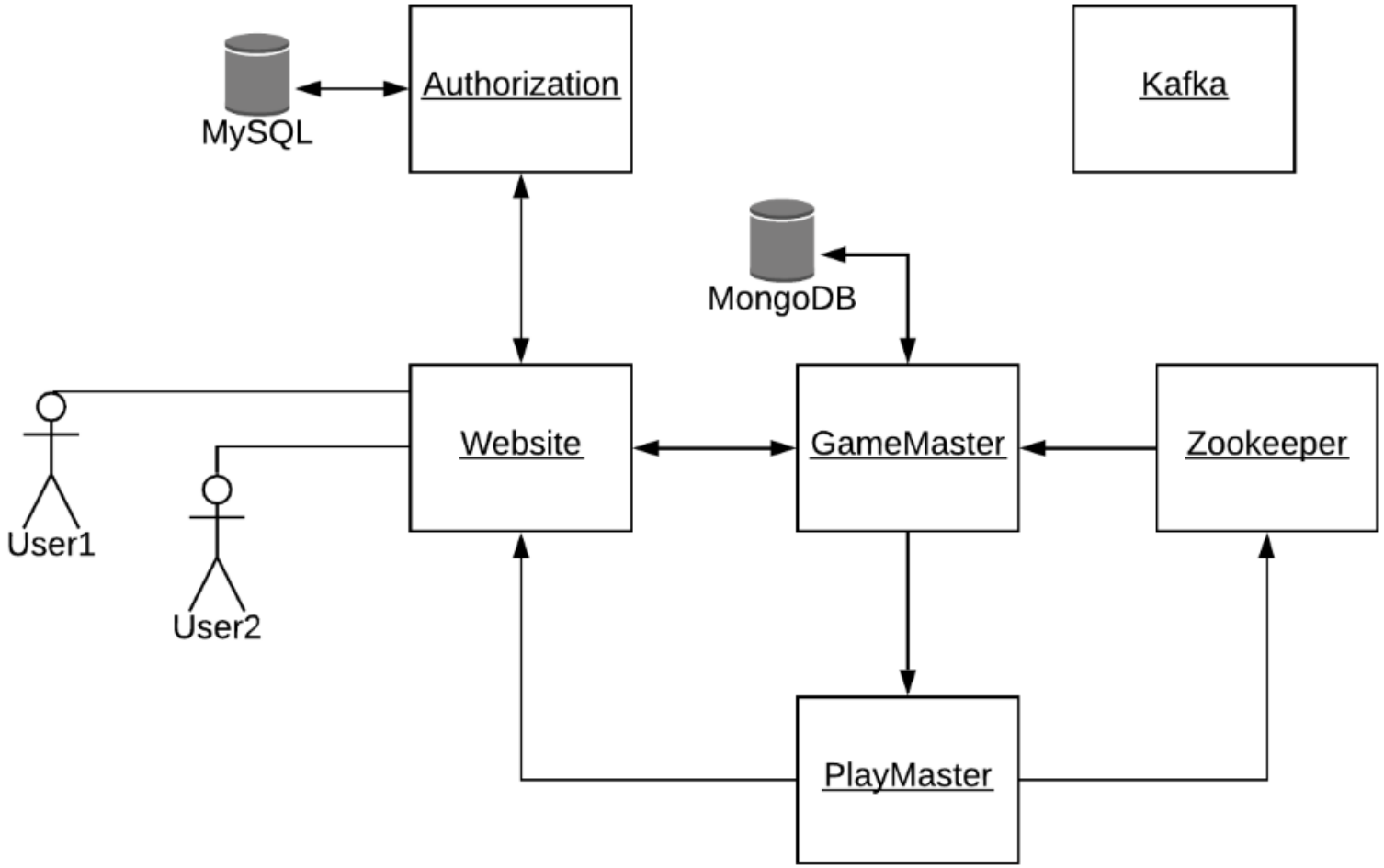
Board Games

Giannakopoulos Panagiotis
Lymperakis Vasilis

System components

- Containers (or Services for Swarm mode)
  - Website
  - Authorization Service
  - Gamemaster
  - Playmaster
  - Apache Kafka
  - Apache Zookeeper
  - Mongo DB
  - MySQL DB

System architecture



Gamemaster

Python

- Zookeeper Client
- Queue Handler
- Score Handler
- API Handler
- Play Recovery

Gamemaster
Queue Handler

- Read from Kafka requests for plays.
- Assign players in practice games.
- Organize tournaments; assign pairs on every round.
- Store to database games and tournaments in progress, and their players
- Send spectator to watch a game upon request.

Gamemaster
Score Handler

- Read from Kafka game results.
- Store practice games.
- Store tournament games and check ties for reassignment, new rounds and if tournament is complete.
- Store to database all of the above information.

Gamemaster API Handler

- Handle users' requests to retrieve data from the database.
- Games:
  - Get: For practice games, return individual plays for the user and the total score of other users. For tournament, returns all individual plays of everyone.
- Spectator:
  - Get: all active games for game type
- Tournaments:
  - Post: tournament creation
  - Get: all tournaments and players connected in it
  - Delete: a tournament

Gamemaster - **Zookeeper Client**

Connect	Connect to Zookeeper
Watch	Add a watch to the path 'games/playmaster'
Save	Save to mongoDB the new list of playmasters.

Gamemaster
Play Recovery

- Listen to port 9000 for client requests.
- Check if the client is valid.
- Retrieve the list of available Playmasters.
- Re-assign the play.
- Return the address of an available Playmaster to the client.

Playmaster

Node JS

Zookeeper Client

Play Listener

Game Server

Playmaster
Play Listener

Start a listener to port 8080 for POST requests.

Request 's body:
{'roundID' : str,
 'spectator': bool,
 'players': array}

The players are added to an array for pending connections.

Playmaster - **Zookeeper Client**

Connect	Connect to Zookeeper
Znode	Create an ephemeral Znode under the path 'games/playmaster/hostname', where hostname is the container name.
Heartbeats	Send heartbeats to Zookeeper

Playmaster - **Game Server (1/3)**

- Listen for WebSocket connection to port 90XX.
- When everyone is connected, initiate the play.
- Serve each play.
- When a play is completed, send the score to Kafka topic 'scores'.
- Score form:

```
{ 'roundID' : str,
  'game' : str ,
  'winner' : str ,
  'players' : array  }
```

Playmaster - **Game Server (2/3)**

Characteristics:

- The server is stateless.
- Arbitrary number of plays can be supported.
- Arbitrary number of players can be supported for the game type.
- Every game which implement the API is supported.

Playmaster **Game Server (3/3)**

Server-Side event handlers

- Connect
- Update
- Disconnect

Client-Side event handlers

- Wait
- Init
- Viewer
- Board
- Disconnect

Game Development
React JS

- The client gets the arguments from the url: Hostname, Playmaster's port, Gamemaster's port, user's token/username.
- The play begins when the connection with Playmaster is established for all players → 'init' message is received.
- When a move is made, the client sends 'update' message:
  { 'roundID': str ,
  'board' : array,
  'progress': 0 (active) | 1 (winner) | 2 (tie) }.
- The client receives the 'board' message with the updated board contents and other variables optionally.
- If the Playmaster fails, the client sends an AJAX call to Gamemaster and then reconnects to a new Playmaster.

Chess Gameplay

Authorization

Python/SQL

Initialization

Requests

Authorization **Initialization**

- Initialize SQL Tables and Events:
- Table users:
  - Contains username(PK), password, email and role
- Table tokens:
  - Contains username(FK), token(PK) and creation date
- Event token_expiration:
  - Checks for tokens more than a week old once a day and deletes them

Authorization Requests

- Users:
  - Post: user creation on sign in; checks for master key and user authorization based on the role of the new user
  - Get: all user info except passwords; checks for user authorization
  - Put: update email and role; checks for user authorization
- Validation:
  - Post: token creation on login; checks for master key
  - Get: check if token exists; checks for user authorization

Website
PHP

Apache Server

Plays Receiver

Website Pages

1. **Index**: Login/Sign up
2. **Portal**: Game selection
3. **Scores**: View scores for practice/tournament games
4. **Official's Panel**: Create tournaments
5. **Admin's Panel**: Install games & edit users

Game Implementation

- The game is built.
- The files of the games are located under public/html/games.
- The game characteristics (name & number of players) are saved in Gamemaster.
- The steps to install a new game include:
  1. Upload the files (in zip) to Website.
  2. Update the DB of Gamemaster.

Game play workflow

1. The user selects a game from the list.
2. An option is selected: Practice/Tournament/Spectate.
3. The Website produce a record to Kafka topic ‘input’.
4. The user is redirected to the loading page.
5. The 'Plays Receiver' process consumes records from the Kafka topic ‘output’ and updates the user session.
6. The user is redirected to the game board and the play begins.
7. When the play is over, the game is redirected to portal or to the loading screen if there is a next round.

Demos

- Sign up & Login: here
- Practice Play: here
- Tournament Play: here
- Spectator Mode: here
- Fault tolerance: here

Thank you