

Design Description *version 1 (November 7th, 2022)*

Project Code Defenders - Robo Tournament
Team Codebenders

Table of content

- Project background (*refer to Project plan document for details*)
 - Project Vision, about Code Defenders
 - Development workflow
- Project requirements (*refer to Requirements definition document for details*)
 - Desired functionalities
 - What the project is not going to address
- System overview and software architecture
 - Streaming component
 - Load Balancer
 - CodeDefenders APIs
 - Database design
- Graphical User Interface
- Detailed software design

Project vision



- Software quality and testing are at the heart of software engineering, but they may not always get enough attention from software engineering education.
- CodeDefenders (web game) proposes the use of **gamification** to teach **mutation testing** and to strengthen code writing and testing skills.
- The game supports **team play and competition** by having Attackers - Defenders teams whose goal is to inject errors into code or write unit tests to catch them.
- The “**CodeDefenders: RoboTournament**” project aims at enriching the game by adding support for students tournaments and games against bots.

Game 115 (Attacker)

Scoreboard Timeline Gradle Export Feedback Editor Mode: default Chat

Existing Mutants

All Alive Killed Claimed Equivalent Equivalent

Mutant	by	Modified line	Points	
Mutant 2131	by grant	4, line 6	45	View
Mutant 2132	by grant killed	Modified line 7	0	View View Killing Test
Mutant 2133	by grant killed	Modified line 9	0	View View Killing Test
Mutant 2134	by grant killed	Modified line 5	0	View View Killing Test
Mutant 2238	by sianico killed	Modified line 4	0	View View Killing Test
Mutant 2239	by sianico killed	Modified line 4, line 6	0	View View Killing Test
Mutant 249	by kJac killed	Modified line 6	1	View View Killing Test
Mutant 251	by akhanfir killed	Modified line 7	1	View View Killing Test

Create a mutant here

Reset Attack

```
1 public class SimpleExamples {
2
3     public static int max(int a, int b, int c){
4         if (a >= b && a >= c)
5             return a;
6         else if (b >= a && b >= c)
7             return b;
8         else
9             return c;
10    }
11
12 }
```

Game 115 (Defender)

Scoreboard Timeline Gradle Export Feedback Editor Mode: default Chat

Class Under Test

```
1 public class SimpleExamples {
2
3     public static int max(int a, int b, int c){
4         if (a >= b && a >= c)
5             return a;
6         else if (b >= a && b >= c)
7             return b;
8         else
9             return c;
10    }
11
12 }
```

Live Killed Claimed Equivalent Equivalent

Mutant restrictions: Moderate

Write a new JUnit test here

Defend

```
1 import org.junit.Test;
2
3 import static org.junit.Assert.*;
4 import static org.hamcrest.MatcherAssert.assertThat;
5 import static org.hamcrest.Matchers.*;
6
7 public class TestSimpleExamples {
8     @Test(timeout = 4000)
9     public void test() throws Throwable {
10         // test here!
11     }
12 }
```

Existing Mutants

All Alive Killed Claimed Equivalent Equivalent

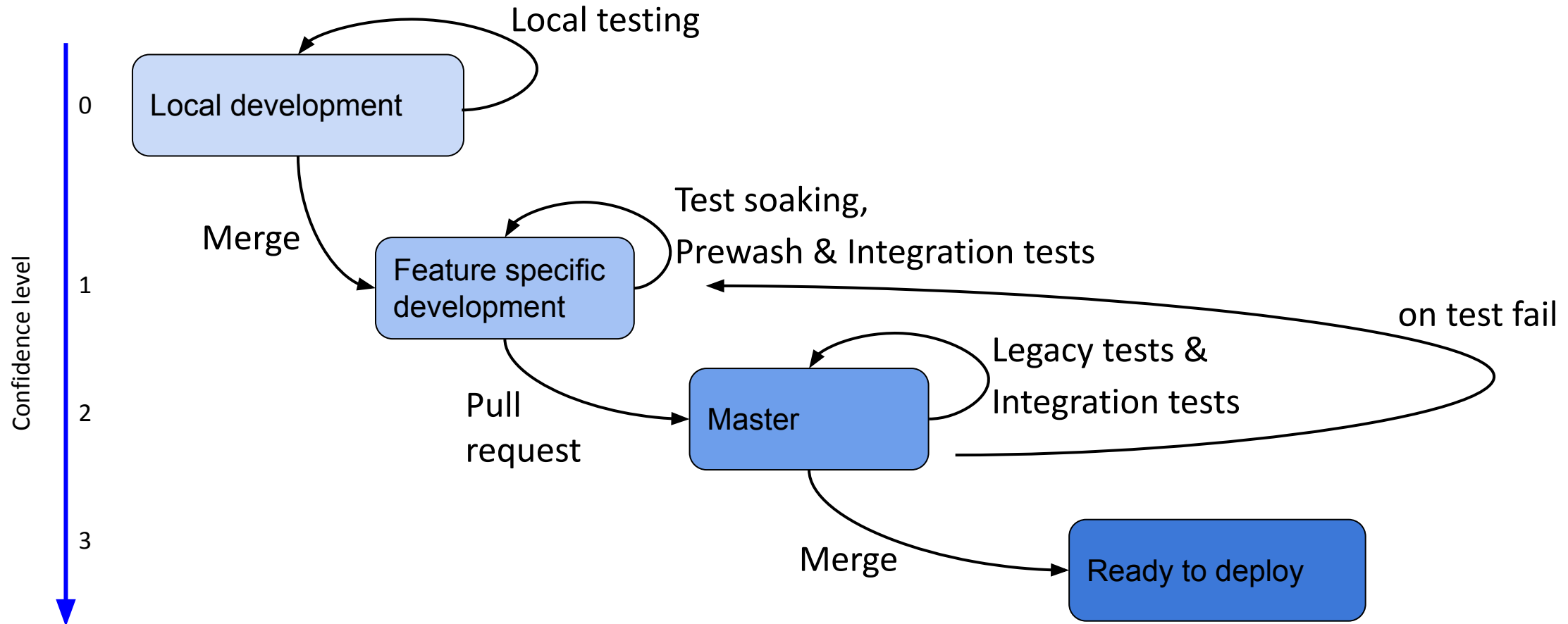
Mutant	by	Modified line	Points	
Mutant 2131	by grant	4, line 6	45	View Claim Equivalent
Mutant 2132	by grant killed	Modified line 7	0	View View Killing Test

JUnit Tests

45 All Tests

44 max(int, int, int)

Development and Testing workflow



Project requirements

- Design and implement a set of **OpenAPIs for CodeDefenders** which can be used from the tournament application to manage games and players.
- Implement the **tournament application**. This application must use CodeDefenders as a remote service (through the APIs) and must include at least two tournaments modalities.
- Implement a **streaming** component which allows users to follow in progress games live. This component can optionally include an “overall tournament view” showing schedule, standings and other information for each tournament.
- Design and implement a set of **APIs** which allows users to train **bots** over past games data and to let those bots play CodeDefenders.

What the project is not going to address

- The tournament application will be an **external application**, developed separately. It won't be a plugin of CodeDefenders nor an application running on the same host.
- The tournament application will implement only the **tournament** and **streaming logic**. It won't reimplement or modify in any way the game logic, which is already coded in CodeDefenders and will be accessible through our APIs.
- We won't implement an **AI** playing CodeDefenders. This project requirement is **optional** and we are not planning to realize it because of the current lack of AI knowledge within our team.
- Streaming component will not respect hard real-time constraints

Desired functionalities (User Stories organized in Epics)

Tournament Management

[CDF-32](#) Login/Register
[CDF-41](#) Display tournaments info
[CDF-33](#) Create Tournament
[CDF-34](#) Join tournament
[CDF-42](#) Matchmaking

Play games in a tournament

[CDF-36](#) Starting games with notification
[CDF-38](#) Return to tournament app from game page

Watch a streamed tournament game

[CDF-39](#) View game stream
[CDF-40](#) Notification of game stream update

Team Management

[CDF-35](#) Team creation
[CDF-54](#) Team management
[CDF-37](#) Join team

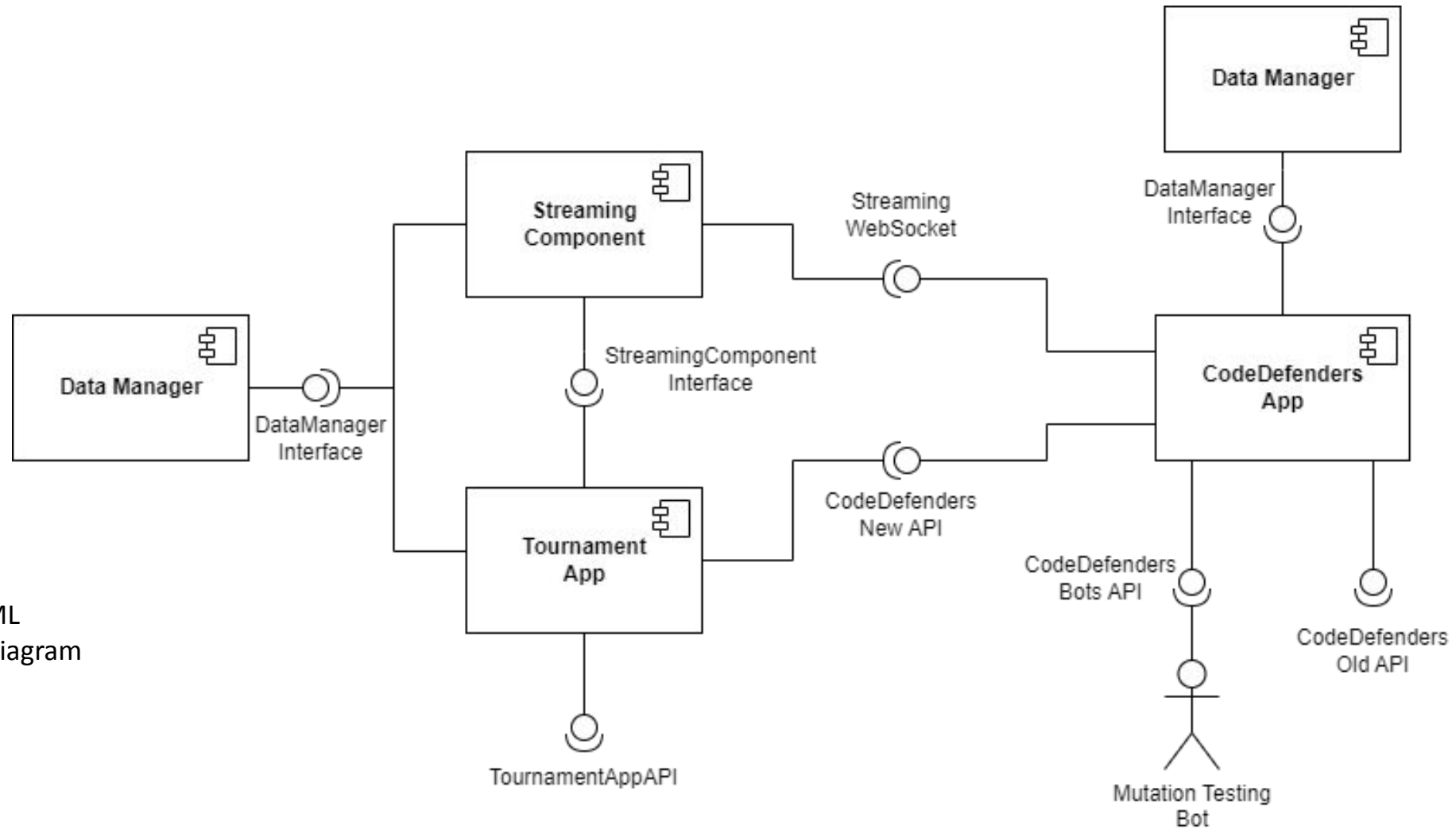
Play with bots in a tournament

[CDF-43](#) Bots can play
[CDF-44](#) Bots can be trained

Avoid CodeDefenders overload

[CDF-31](#) Low latency
[CDF-69](#) Efficient flow of updates

Software architecture



High level UML
component diagram

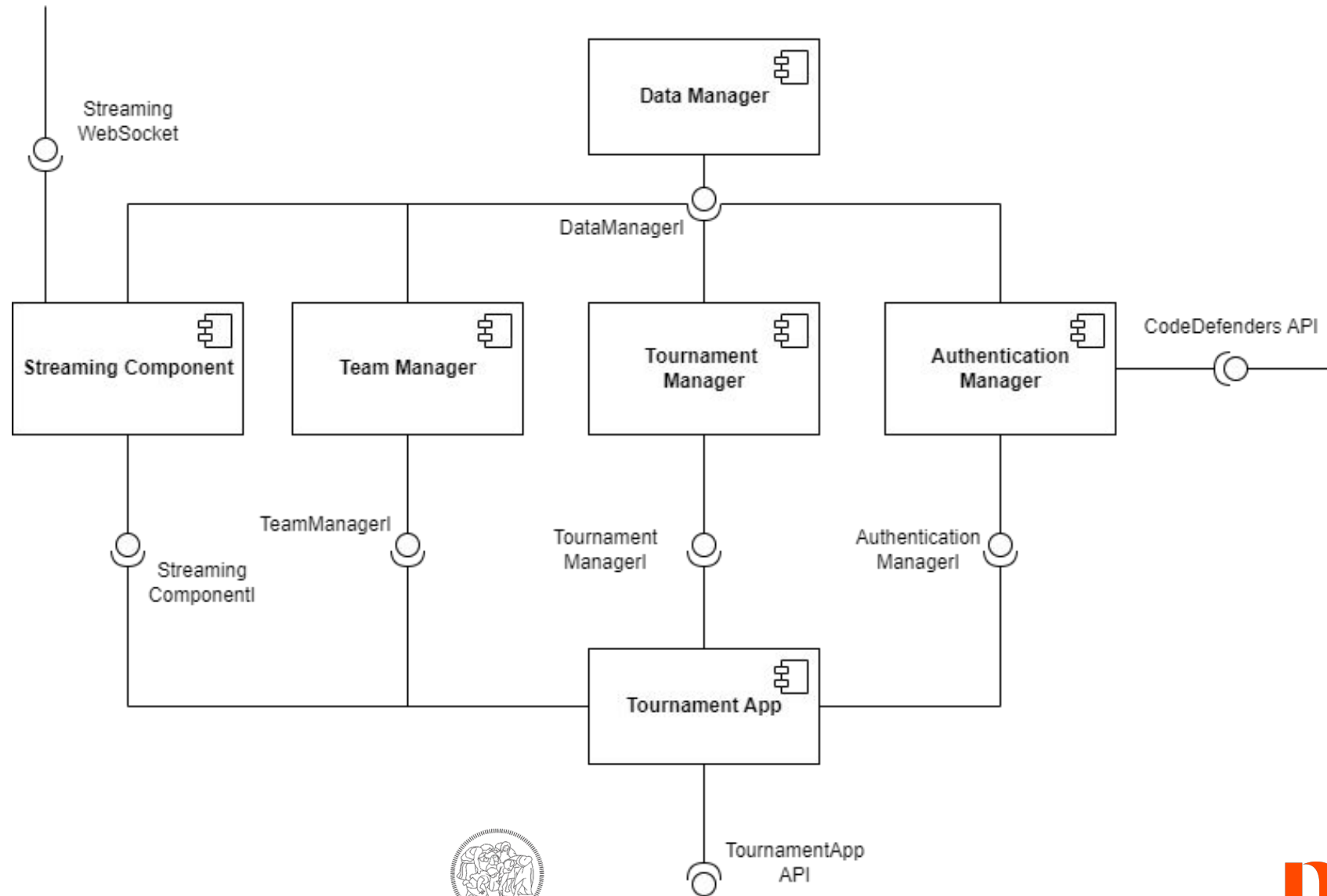
Software architecture

Each software component addresses the following user stories

- Tournament app:
 - [CDF-41](#), [CDF-33](#), [CDF-34](#), [CDF-42](#), [CDF-35](#), [CDF-37](#), [CDF-54](#),
[CDF-36](#), [CDF-38](#)
- Streaming component:
 - [CDF-39](#), [CDF-40](#)
- Code defenders APIs:
 - For bots:
[CDF-43](#), [CDF-44](#)
 - Other:
[CDF-32](#), [CDF-39](#), [CDF-36](#)

Tournament Application

A zoom on the internal components of the tournament application



Software architecture

Each software component addresses the following user stories

- Team manager:
 - [CDF-35](#), [CDF-37](#), [CDF-54](#)
- Tournament manager:
 - [CDF-41](#), [CDF-33](#), [CDF-34](#), [CDF-42](#)
- Streaming component:
 - [CDF-39](#), [CDF-40](#)
- Authentication manager:
 - [CDF-32](#)
- Tournament app, other:
 - [CDF-36](#), [CDF-38](#)

Important design decisions

- Streaming component
- Load Balancer
- APIs for bots
- Database design

Streaming component (option 1)

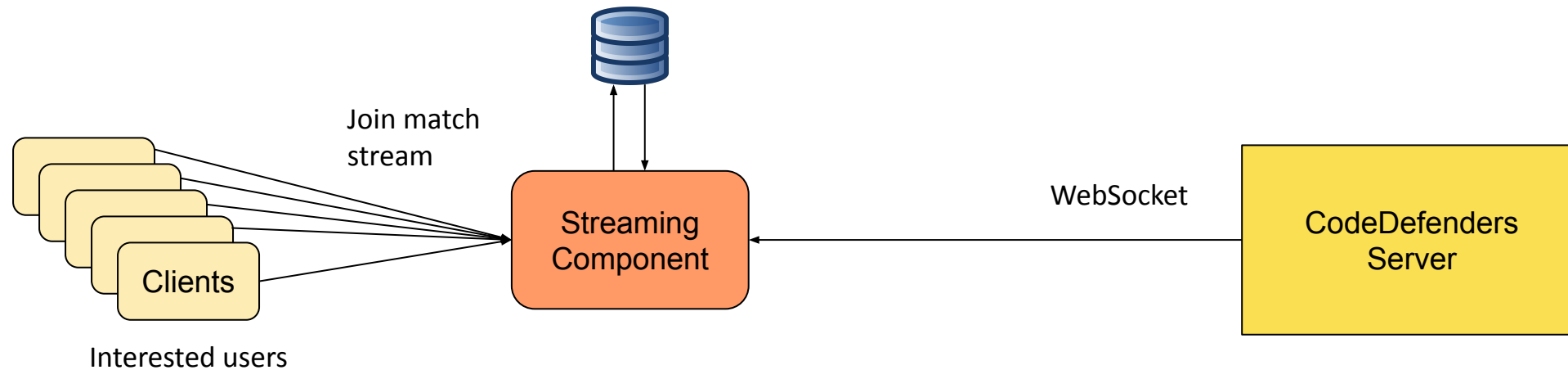


The streaming component periodically updates (frequently to give the impression of real time):

- Polls the CodeDefenders server to retrieve the events happened
- Only for the new events happened notifies the interested users

This solution would be really inefficient (streaming component continuously polling the server by calling the API) and also difficult to implement (we should keep track only of the new events and notifying only them).

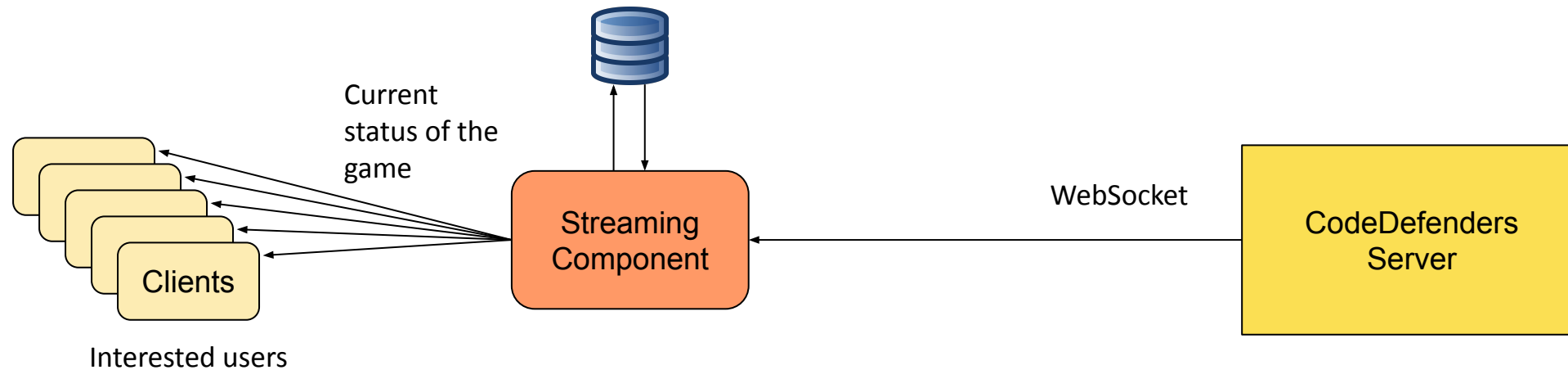
Streaming component (option 2)



CodeDefenders server establishes a WebSocket with the Streaming Component and sends events to it.

Clients interested in an ongoing game join the stream by sending a request to the Streaming Component. It responds by querying its local db and retrieving the current status of the game (e.g. points for each team). Then a WebSocket is established between each client and the Streaming Component.

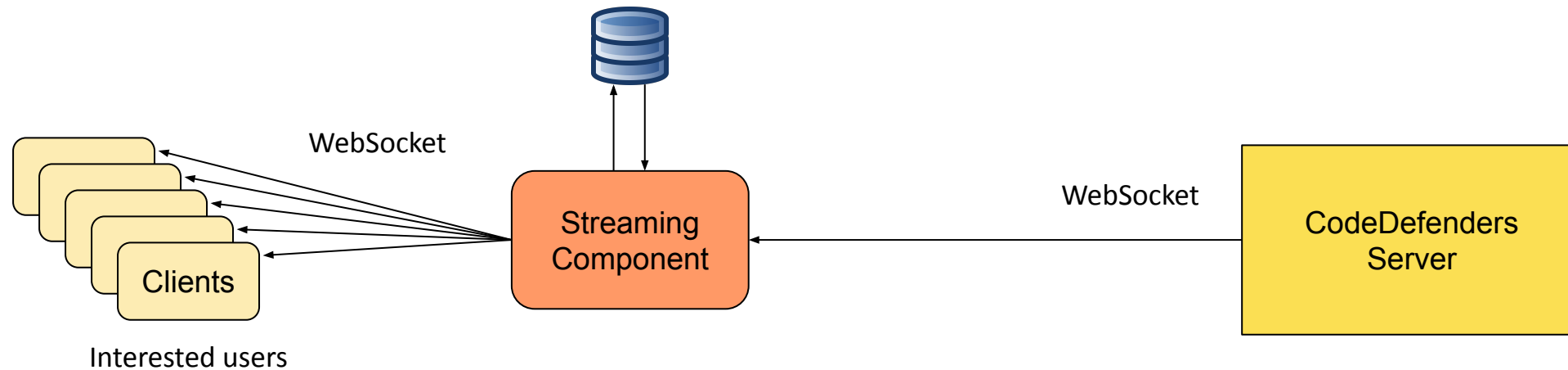
Streaming component (option 2)



CodeDefenders server establishes a WebSocket with the Streaming Component and sends events to it.

Clients interested in an ongoing game join the stream by sending a request to the Streaming Component. It responds by querying its local db and retrieving the current status of the game (e.g. points for each team). Then a WebSocket is established between each client and the Streaming Component.

Streaming component (option 2)



Every time an event takes place during the game, CodeDefenders server sends an update to the Streaming Component through the WebSocket.

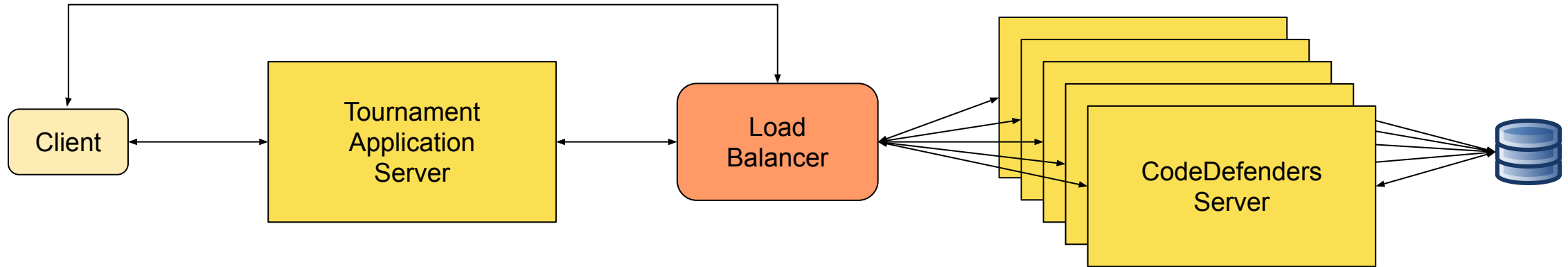
The Streaming Component receives the events and updates its local database (only for important events such as point scored).

The Streaming Component forwards all the events to the interested clients by means of WebSockets.

The client receives the event and displays visual effects if they are active.

This solution would be efficient (events are notified only once) and would also reuse a paradigm already used in CodeDefenders to handle other types of events (i.e. chat messages).

Load Balancer



The load balancer component receives every request addressed to CodeDefenders and redirects it to the CodeDefenders backend server with the minimum number of requests. It is implemented using the existing `mod_proxy_balancer` module for Apache and runs on its own Apache HTTP server.

The load balancer component fulfills the non-functional requirement [CDF-31](#).

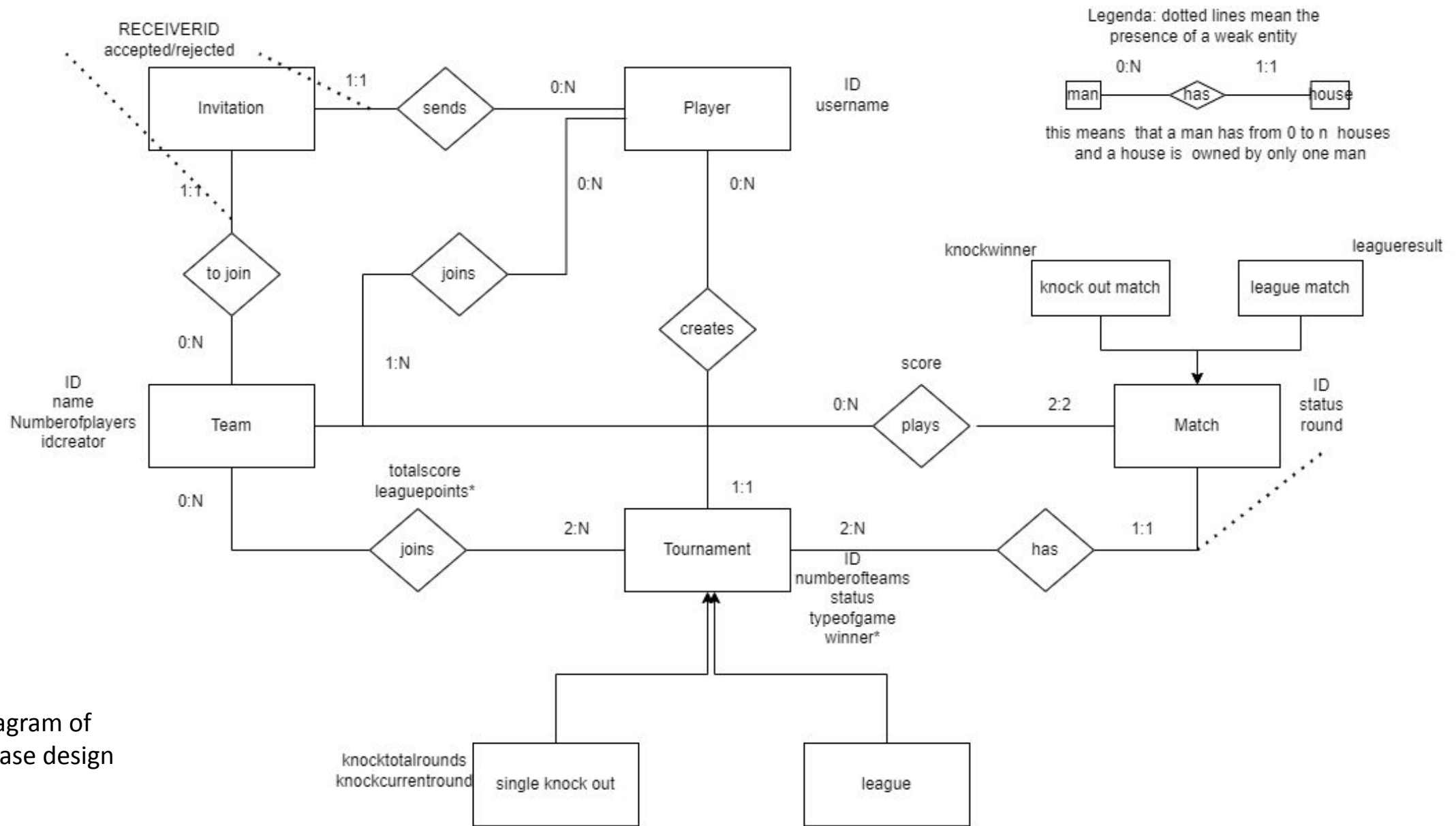
- The load balancer is completely transparent for clients and applications
 - Clients will send requests and receive responses as if they were talking directly to a CodeDefenders server. They are unaware of the architecture behind the load balancer.
- The load balancer exploits the concept of stickiness
 - The first time a client interacts with the load balancer it gets a cookie that lasts until the end of the session.
 - Every time the same client sends a new request, the load balancer recognizes the cookie and sends the request always to the same CD server.
 - This mechanism is necessary so that the web session for one client is managed only by one CD server.

Exposed CodeDefenders APIs

/api/class	GET	Get a class by its ID
/api/test	GET	Get a test by its ID
	POST	Upload a test providing its code and target game
/api/mutant	GET	Get a mutant by its ID
	POST	Upload a mutant providing its code and target game
/api/claimEquivalent	POST	Claim the mutant with the specified ID as equivalent, meaning that it doesn't affect the behavior of the code
/api/pendingEquivalences	GET	Get unresolved equivalence claims for the game with the specified ID
/api/status	GET	Get the status of the game with the specified ID
/api/settings	GET	Get the settings of the game with the specified ID
/api/resolveEquivalence	POST	Resolve the pending equivalence claim for a mutant, either by accepting it as equal or uploading the code for a killing test
/api/history	GET	Get the history of all played games, or filter by class ID and/or user ID

Exposed CodeDefenders APIs

/api/player	GET	Returns a player's username and userId from its playerId
/api/user	GET	Returns a user's informations from its ID
/admin/api/token	GET	Get an authentication token for the Bot API, tied to a specific user
/admin/api/verify	GET	Check if a username and password combination is valid
/admin/api/register	POST	Register a new user with a username, password, email triplet
/api/classes	GET	Get the list of uploaded classes
/admin/api/game	POST	Create a new game with the specified class, settings and teams
/admin/api/game/start	POST	Start a game
/admin/api/game/end	POST	End a game

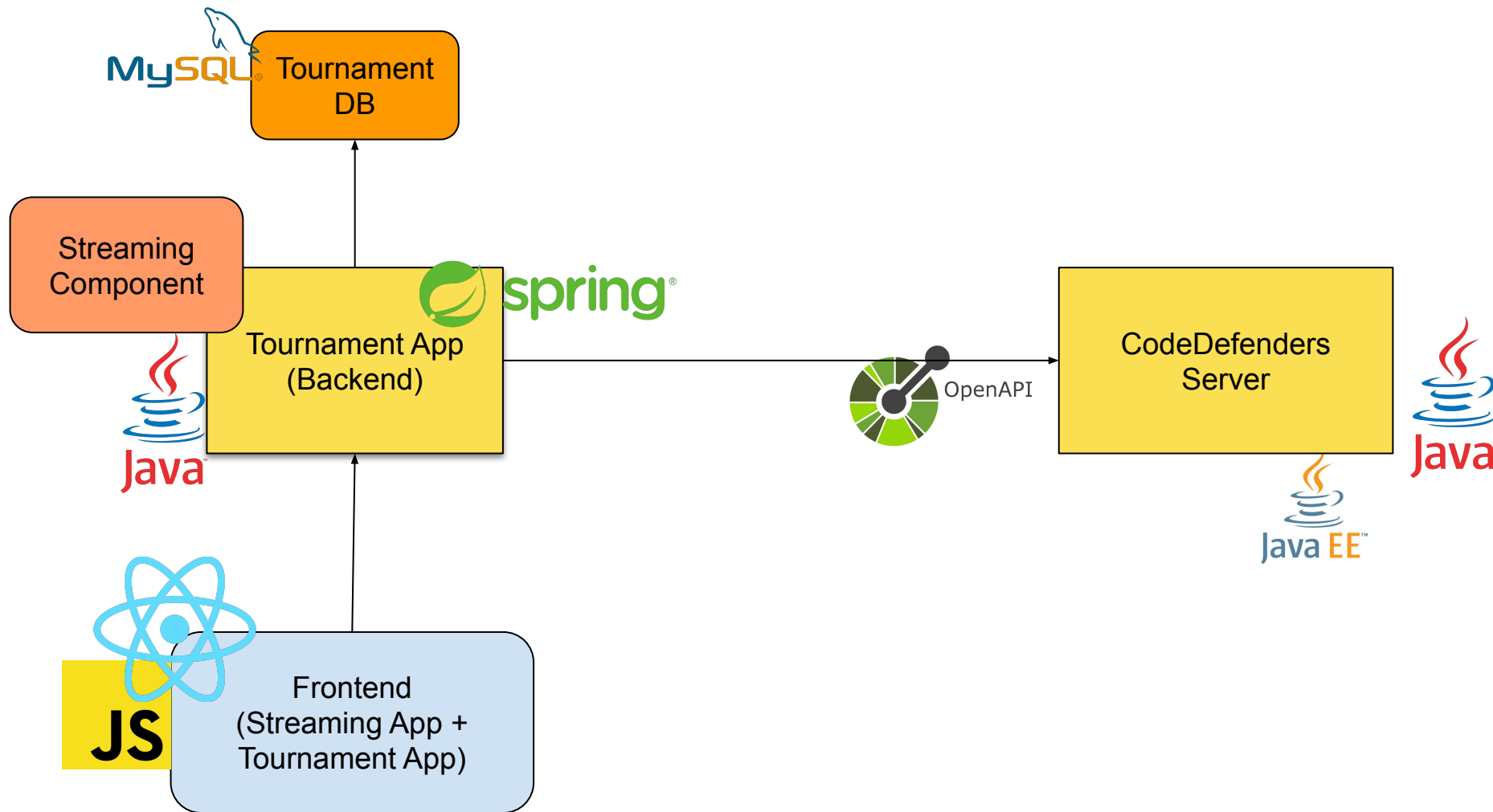


ER Diagram of
database design

Database Design

- Players compete in matches through their teams
- Teams can be of variable size
 - this way tournaments with teams of max size of one player are possible, in other words, 1v1 matches are supported
 - tournament logic can but doesn't need to set conditions for team size
- Each match is assigned to exactly one tournament, and has exactly two teams as competitors (attackers vs defenders)
- Database design does not differentiate human players and bots

Technologies we are going to use



Motivations for technologies

Java + Spring:

- Java knowledge quite already spread among the team
- Integrates very well with concepts taken from JavaEE
- More support than JavaEE on forums and so on (also tutorials online)
- Allows to structure a Java Web application in an easy way

MySQL:

- The structure of the data we need will suit perfectly in a relational database.
- Team already familiar with this technology.

React:

- Free, open-source, and explanatory JavaScript library with simplistic learning curve
- Used for building simple or complex user interfaces, stable front-end framework
- Supports multi-purpose, clean architecture and platform-specific modules

Graphical User Interface

Tournament web app should have user interface for

- Team management
 - CDF-35, CDF-37, CDF-54
- Tournaments overview
 - CDF-34, CDF-36, CDF-41
- Tournaments creation
 - CDF-33, CDF-42
- Watching live score of matches (streaming component)
 - CDF-39, CDF-40

With support for desktop and mobile devices

Home page



Code Defenders

Tournaments

Team
management

Create new
Tournament

FAQ

Log in or
Register

Filter by: Players Teams Date Type Search by name: _____

Name ▲

Starting date ▼ Type

The battle for steaming hot coffee

17.10.2022.

SET

Join

Live score

Peculiar bugs and how to find them

12.10.2022.

RR

Join

Live score

FER championship

16.1.2023.

DET

Join

Live score

...

Join

Live score

...

Join

Live score

...

Home page



Code Defenders Tournaments

Team
management

Create new
Tournament

FAQ

Log in or
Register

Filter by: Players Teams Date Type Search by name: _____

Name ▲

Starting date ▼ Type

The battle for steaming hot coffee

17.10.2022.

SET

Join

Live score

Peculiar bugs and how to find them

12.10.2022.

RR

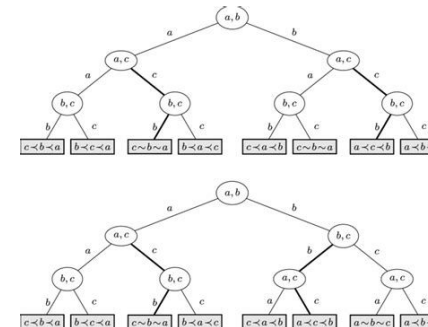
Join

Live score

Currently playing: Johnny vs Sans

Next match on Friday 14. October 14:00 GMT

Bob vs Johnny



Participating:

Bob

Sans

Johnny

...

FER championship

16.1.2023.

DET

Join

Live score



Code Defenders Tournaments -> Team management

Join Team

Create new
Team

Manage your
team

Log out

Filter by: Players

Search by name: _____

Name ▲

Date created ▼ Score

Steaming hot coffee enjoyers

16.10.2022.

0

Join

Guys from the basement

14.10.2022.

32

Join

The quick brown fox

11.10.2022.

69

Join

...

Join

...

Join

...



Code Defenders

Tournaments -> Team management

Join Team

Create new
Team

Manage your
team

Log out

Team Name: _____

Add Players
Name ▲

Search by username: _____
Score ▲

Bob

11

Add to team

SanAndreas

32

Add to team

Sans

420

Add to team

...

Add to team

...

...



Code Defenders Tournaments -> Team management

Join Team

Create new Team

Manage your teams

Log out

Your teams

Search by name: _____

Name ▲

Date created ▼ Score

Steaming hot coffee enjoyers

16.10.2022.

0

Edit

Leave

Guys from the basement

14.10.2022.

32

Edit

Leave

The quick brown fox

11.10.2022.

69

Confirm

Cancel

Player:

Role:

Sans

Leader

Hrvoje

Member

Promote to leader

Kick from the team

.....

.....

...



Code Defenders Tournaments

Team
management

Create new
Tournament

FAQ

Log out

Tournament Name: _HotKey Tournament____
Matchmaking: ☐ random ☒ manual

Type:

DET
SET
RR

Create

Complete Tournament definitions:

FER championship

16.1.2023.

DET

Edit

...



Code Defenders Tournaments

Team
management

Create new
Tournament

FAQ

Log out

Tournament Name: HotKey Tournament

Type: SET

Finish

Cancel



Assign participants manually

Add participants by name: _The Guys____

Team TD

Dream Team

...

Teams that join
tournament will
appear here

OR

Assign automatically



Code Defenders Tournaments

Team
management

Create new
Tournament

FAQ

Log out

Tournament Name: _HotKey Tournament____

Type: SET

Finish

Cancel



Add participants by name: _____

The Guys from the basement

Assign participants manually

OR

Assign automatically



Code Defenders Tournaments

Team
management

Create new
Tournament

FAQ

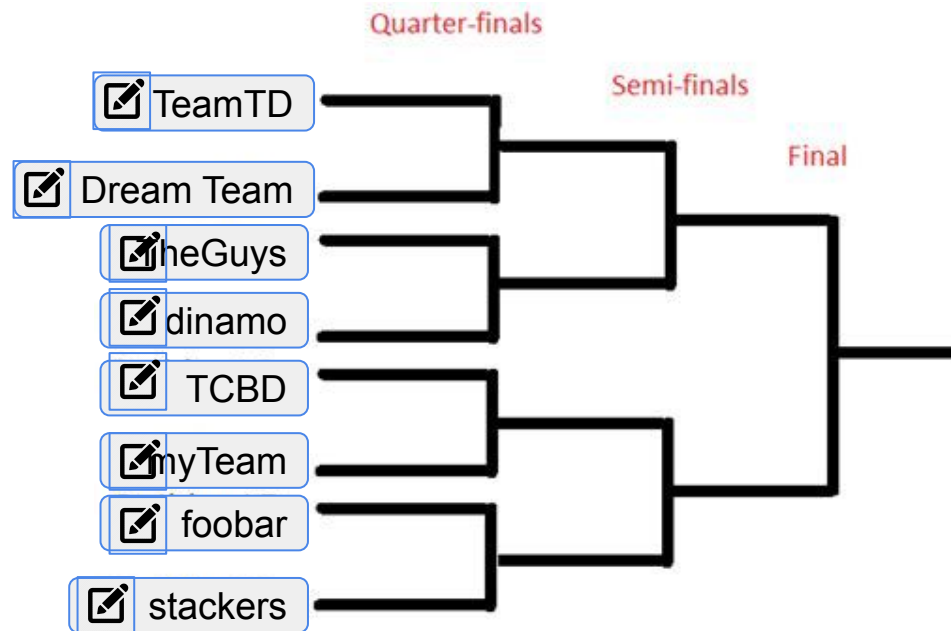
Log out

Tournament Name: _HotKey Tournament_

Type: SET

Finish

Cancel



Add participants by name: _____

Assign participants manually

OR

Assign automatically



Code Defenders Tournaments

Team
management

Create new
Tournament

FAQ

Log out

How to navigate in this app?

- There are buttons to change view on left side panel,
- To return to home page simply click on Tournaments next to CodeDefenders logo
- CodeDefenders logo redirects you to CodeDefenders web game

How to create new tournament?

- 1) Choose type from dropdown list
 - 2) Firstly add participants and then assign automatically or manually by choosing from dropdown list, (one of options is to free that slot)
- *) while adding manually list of participants on the right shows only participants not yet assigned to position on tournament tree

How to manage teams?

....

Live score



Code Defenders

Tournaments

Match: Finals

Tournament: The battle for steaming hot coffee

Attacker: Johnny

9

Animation
on event

Defender: Sans


12

History: Johnny scored 2 points for injecting **error**

....

Home page

Mobile phone support

 **Code Defenders** **Tournaments**

[Team management](#) [Create new Tournament](#) [FAQ](#)

Filter by: [Players](#) [Teams](#) [Date](#) [Type](#)
Search by name: _____

Name ▲

Starting date ▼

Type

The battle for steaming hot coffee

17.10.2022.


SET

Peculiar bugs and how to find them

12.10.2022.

RR

FER championship



Participant 1
Participant 2
Participant 3
Participant 4
Participant 5
Participant 6
Participant 7
Participant 8

Quarter-finals
Semi-finals
Final

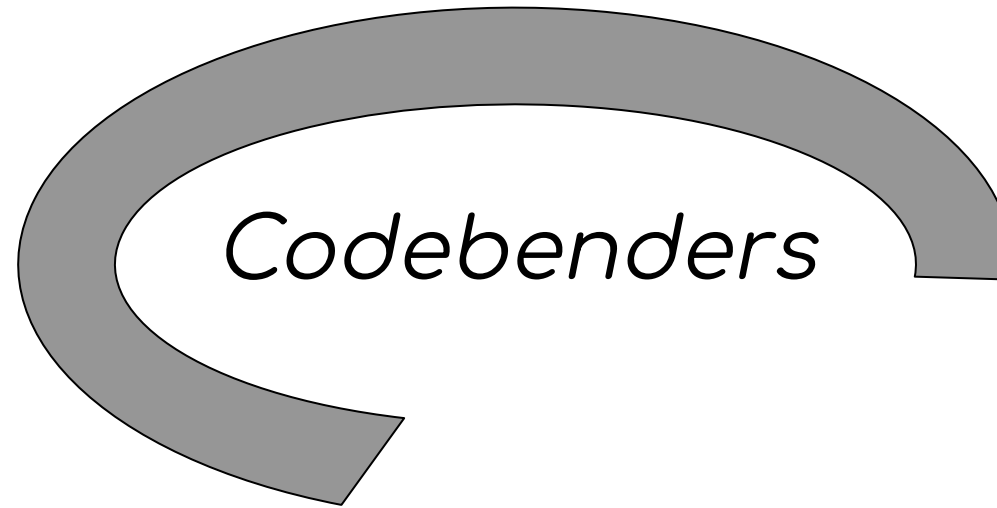
[Join](#) [Live score](#)

...

Detailed software design

A detailed description of each component design will be added later on and updated at the beginning of each sprint to match its current implementation.

The whole design document will be kept constantly updated to reflect the current state of the system.



contact info:

fanny.delnondedieu@fer.hr

dominik.brdar@fer.hr

hrvoje.rom@fer.hr

simone.mezzaro@mail.polimi.it

fabio.patella@mail.polimi.it

andrea2.restelli@mail.polimi.it