

Project Plan

revision B (January 5th, 2023)

Project Code Defenders - Robo Tournament
Team Codebenders

Agenda

- Our Team
- Project vision and plan
 - About Code Defenders
 - Project requirements
 - Our solution
 - Motivation
 - Customer
 - What the project is not going to address
 - Risks
- Our way of working
 - Implementation of SCRUM
 - Tools for communication, project management and development
 - Development and testing workflow

Our Team

Frontend

Product Owner

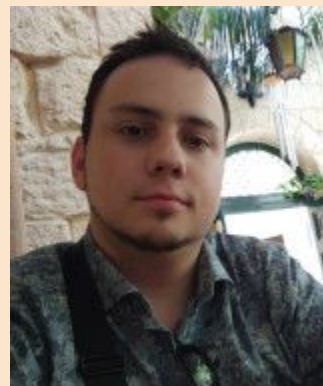


Fanny Delnondedieu



Fabio Patella

Scrum Master



Dominik Brdar

Testers



Hrvoje Rom

Backend



Simone Mezzaro



Riccardo Nava




Andrea Restelli

Project vision



The “**CodeDefenders: RoboTournament**” project aims at enriching the already existing game of CodeDefenders with some new functionalities: students tournaments, games against bots and live score streaming. The tournaments and streaming functions will be implemented by an **external application**.

 Code Defenders Multplayer ▾ Puzzles restp99 ▾

My Games

ID	Creator	Class	Players	Level
You are currently not active in any games.				
Create battleground game Create melee game				

Open Battleground Games

ID	Creator	Class	Attackers	Defenders	Level
> 100	mrpinola	Lift	23 Join	24 Join	Easy
> 118	samira	Lift	5 Join	8 Join	Hard
> 135	sandya123	Lift	1 Join	3 Join	Hard
> 115	rdegiovanni	TAROT2022	16 Join	18 Join	Easy
> 286	sergio	LiftWithMutants	1 Join	1 Join	Easy
> 171	rulands	Lift2WithMutants	5 Join	8 Join	Hard
> 172	rulands	Lift2WithMutants	0 Join	2 Join	Hard
> 173	rulands	Lift2WithMutants	0 Join	0 Join	Hard
> 174	rulands	Lift2WithMutants	0 Join	0 Join	Hard
> 175	rulands	Lift2WithMutants	0 Join	0 Join	Hard
> 176	rulands	Lift2WithMutants	0 Join	0 Join	Hard
> 177	rulands	Lift2WithMutants	0 Join	0 Join	Hard

About Code Defenders

CodeDefenders is a **web game** on mutation **testing**. Each game involves two teams of students, competing against each other, and a piece of code (code under testing).

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

23 All Mutants				
Mutant 2131	by grant	Modified line 4, line 6	Points: 45	View
Mutant 2132	by grant	killed	Modified line 7	Points: 0 View View Killing Test
Mutant 2133	by grant	killed	Modified line 9	Points: 0 View View Killing Test
Mutant 2134	by grant	killed	Modified line 5	Points: 0 View View Killing Test
Mutant 2238	by alerla	killed	Modified line 4	Points: 0 View View Killing Test
Mutant 2239	by slanico	killed	Modified line 4, line 6	Points: 0 View View Killing Test
Mutant 249	by kJac	killed	Modified line 6	Points: 1 View View Killing Test
Mutant 251	by ahanfir	killed	Modified line 7	Points: 1 View View Killing Test

JUnit Tests

45 All Tests

44 max(int, int, int)

Game 115 (Attacker)

Scoreboard

Timeline

Gradle Export

Feedback

Editor Mode: default

Chat

Existing Mutants

All Alive Killed Claimed Equivalent Equivalent

23 All Mutants				
Mutant 2131	by grant	Modified line 4, line 6	Points: 45	View
Mutant 2132	by alerla	killed	Modified line 7	Points: 0 View View Killing Test
Mutant 2133	by grant	killed	Modified line 9	Points: 0 View View Killing Test
Mutant 2134	by alerla	killed	Modified line 5	Points: 0 View View Killing Test
Mutant 2238	by slanico	killed	Modified line 4	Points: 0 View View Killing Test
Mutant 2239	by slanico	killed	Modified line 4, line 6	Points: 0 View View Killing Test
Mutant 249	by kJac	killed	Modified line 6	Points: 1 View View Killing Test
Mutant 251	by ahanfir	killed	Modified line 7	Points: 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 }
```

The attackers team injects faults in the code under testing creating so called mutants, whereas defenders write unit tests to spot the injected problems.

Defenders earn points for each discovered mutant. Attackers earn points for each mutant escaping the tests.

Project requirements

- Implement a **tournament application**. This application must use CodeDefenders as a remote service (through APIs) and must include at least two tournaments modalities.
- Design and implement a set of **OpenAPIs for CodeDefenders** which can be used from the tournament application to manage games and players.
- Implement a **load balancing** mechanism which allows the tournament application to communicate with **multiple CodeDefenders servers** and to always create games on the less loaded server.
- 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.
- Please refer to document *Requirements Definition* for more details

What is our solution?

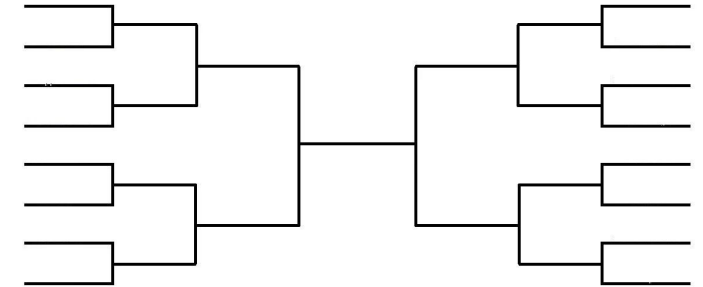
Since CodeDefenders is an open source project deployed on a public GitHub repository, we firstly plan to create a **forked repository** where we can inspect the code, identify which functionalities need to be exposed and implement APIs. CodeDefenders backend is implemented using Java Servlet technology and our APIs will conform to it.

Our tournament application will be a **web based application**. Details about frameworks and technologies employed to realize it can be found in the document *Design Description*. We plan to include the streaming component in the tournament application itself.

Motivation

Software quality and testing are at the heart of software engineering, but they are not always at the heart of software engineering education. CodeDefenders proposes the use of **gamification** to teach **mutation testing** and to strengthen testing skills, by introducing a mutation testing game. CodeDefenders can assist educators in delivering complex mutation testing concepts and is intended to make the learning experience more **enjoyable** and **fruitful** for students.

To make the game even more fun, the scope of the project is to introduce a **tournament mode** in it, so that players can compete and to climb the leaderboard.



Because tournaments are not fun without an audience, we are asked to implement also a **streaming service/application** that let guests and users follow the games of their preferred players live.

Lastly, the design of an API to allow the integration of **bots** in CodeDefenders would make possible to create challenges where students/players/teams compete against artificial intelligence.

Our customer

Professor **Alessio Gambi**

IMC University of Applied Sciences Krems (Austria)

University of Passau (Germany)



Worked on CodeDefenders in the past, now he has not enough time to implement anything.

The project is currently maintained by some students from University of Passau.

However, he will be our **reference point** for the RoboTournament extension.



We will join the students' Slack workspace to ask more technical questions **directly** to the current maintainers of the project.

First meeting with the customer on Wednesday. Two other students will join the meeting together with the Product Owner in order to avoid misinterpretations.

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

Project risks

Risk	Chance of occurring	Impact on project
Poor communication and proactivity	Moderate	High
Lack of technical skills	Low	Moderate
Lack of time	Moderate to high	Moderate
Significant changes in CodeDefenders during our project	Moderate	Low to moderate
Discovering bugs in code too late	Moderate	High
Missing of members working individually on some decoupled parts of the project	Moderate	High

Risk management

- To ensure better cooperation, communication and productiveness of our team we will be using Scrum agile development process. Collaboration between team members will also help mitigate lack of technical skills.
- We agreed with customer to freeze current version of CodeDefenders, by forking the git repository, so there are no interferences between our development and CodeDefenders maintenance by students of University of Passau.
- Commit to test strategy to discover bugs on time.

Risk discovered later in the project

- Missing of members working individually on some decoupled parts of the project
 - This risk was discovered later in the project. It mainly refers to Riccardo being the only one developing CodeDefenders APIs and Hrvoje working alone on testing infrastructure
 - It would be hard to cover for missing of members assigned to that parts
 - To mitigate this risk we tried to increase communication and to provide good documentation for the other team members

Our way of working

Team's focus points

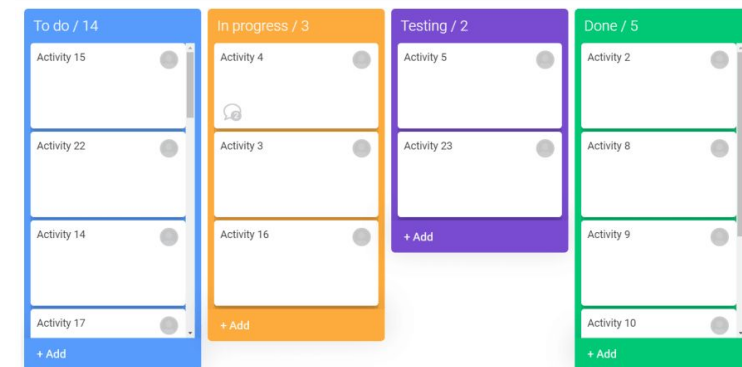
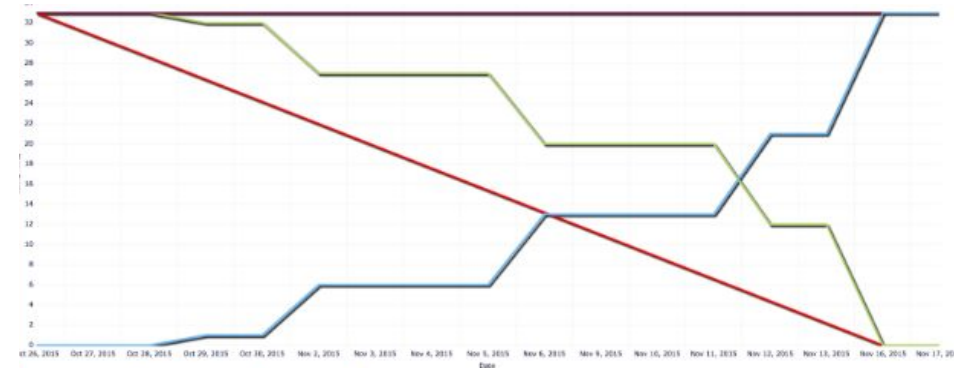
- Agile development
 - Having good communication with customer
 - Self-managing our team using SCRUM and Jira tool
 - Competence buildup
 - Working together
 - Test strategy
 - Analyze requirements and do testing for all use cases
 - Continuous integration, automated tests
- => Quality assurance (0 Trouble Reports to customer)

Implementation of SCRUM

- Sprints of 2 weeks
- Backlog grooming every Tuesday at 21h (CEST)
 - Sprint planning every second Tuesday after backlog grooming
- Sync meetings (daily's) 2-3 times a week (15min max)
 - Follow-up meetings if needed
- Sprint Review on Mondays after Sprint end
 - Retrospective following after the review
- Every-day communication using Discord

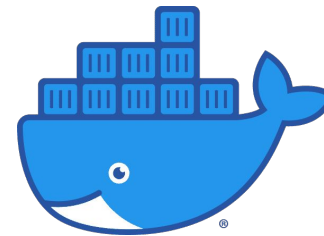
How will we track our progress?

- Estimating working hours needed to complete tasks
- Logging work done on each task
- Using Sprint burndown charts
- Using board to track status of tasks
- Competence matrix

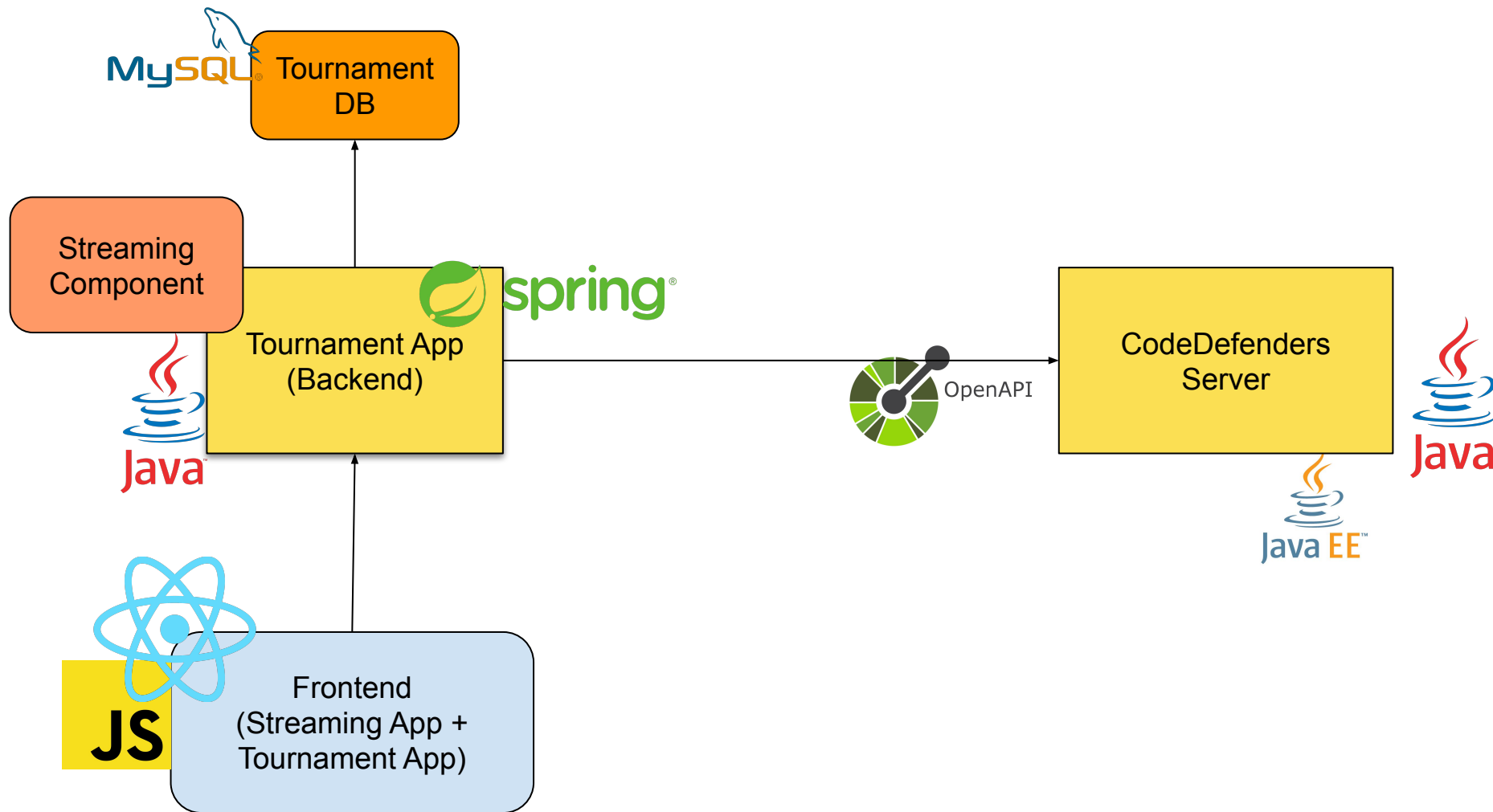


Tools that we will be using

- Communication
 - Discord, Slack, Teams
- Project management
 - Jira
- Collaboration on documentation
 - Google docs, slides, drive
- Development
 - GitHub, Docker, Maven, Vagrant
 - Jenkins



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

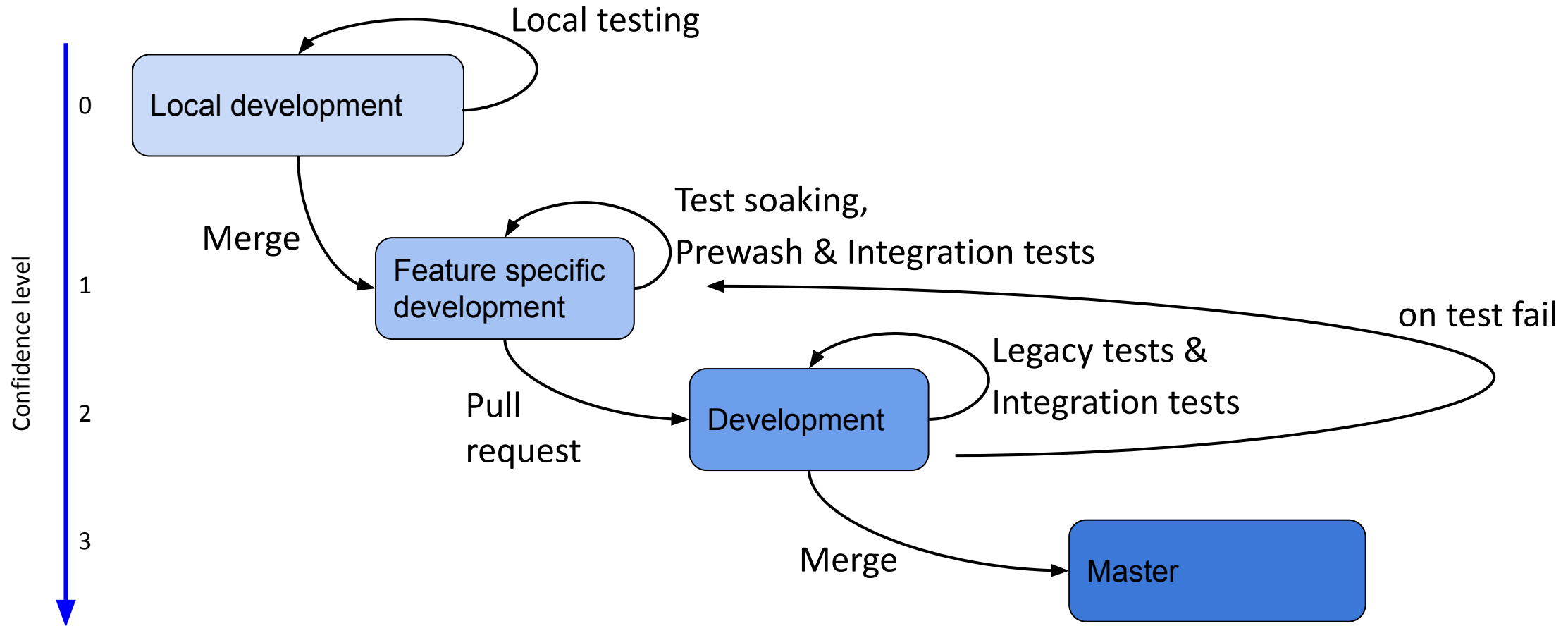
MySQL:

- 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

Development and Testing workflow



Git Branching Strategy

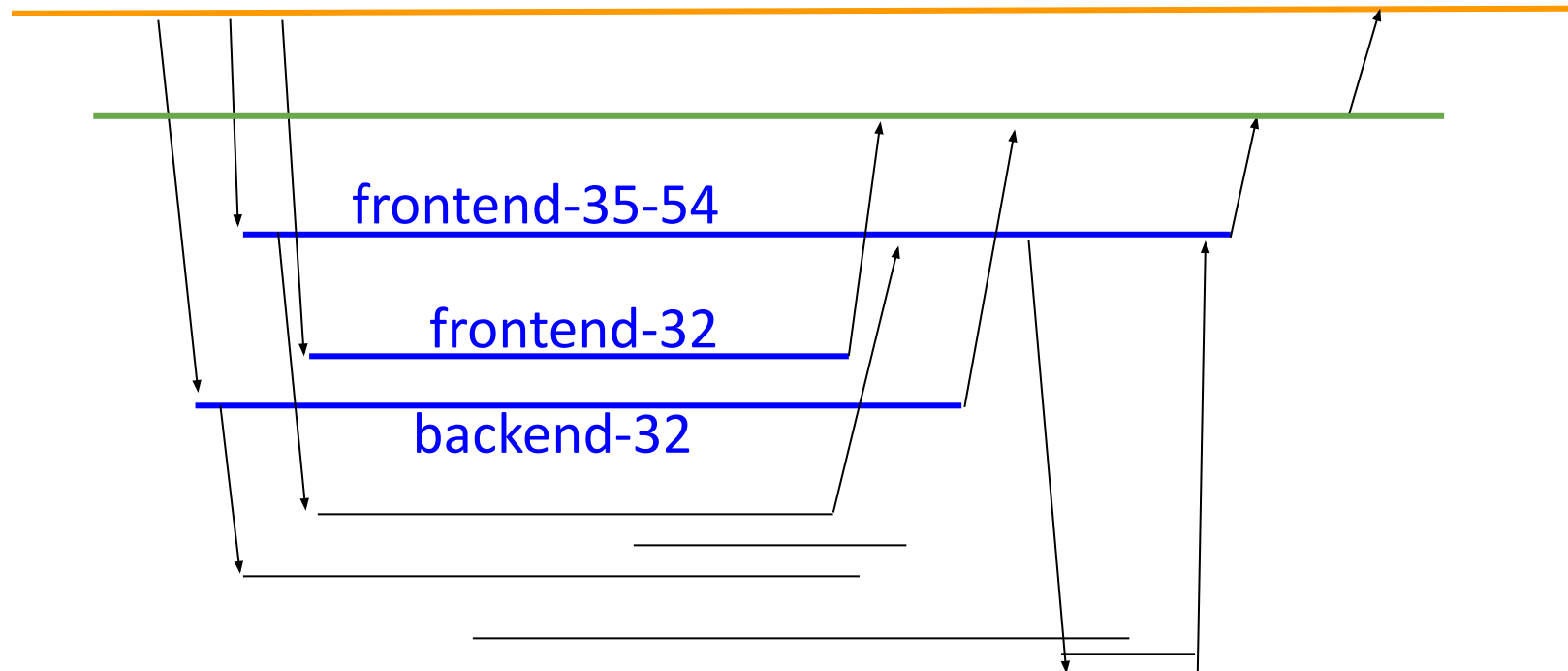
2 repositories -> *CodeDefenders* and *Robo-Tournament-Application*

Master

Dev

Features
development

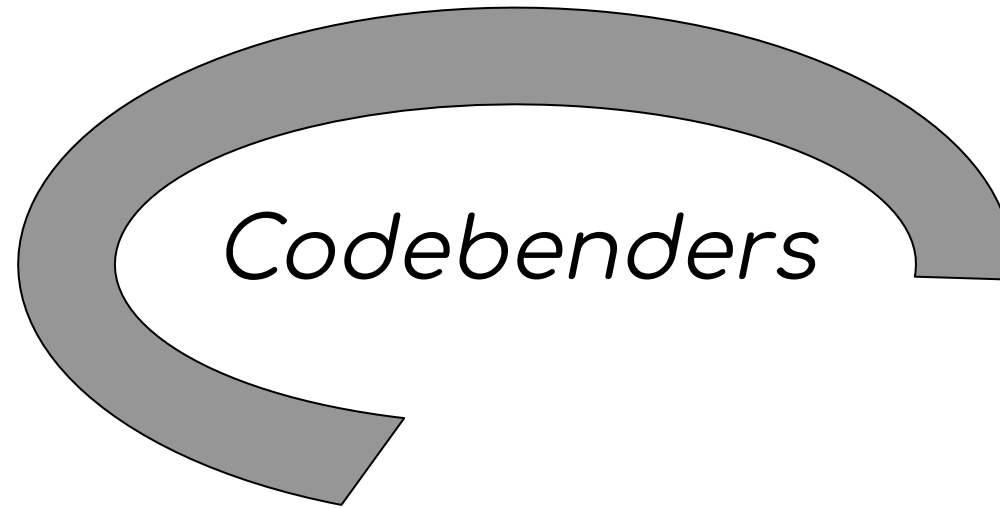
Local branches



example of Robo-Tournament repo branching

Git Branching Strategy - development workflow

- 1) From Development branch out feature specific development branches
 - a) (one branch can be for frontend/backend of one or more user stories)
- 2) Team members branch out their local branches from feature specific ones
- 3) On local branches, developers implement smaller functional requirements and test their code locally (manually or with automated tests)
- 4) Once developers feel confident in their solution, they merge into that specific development branch
- 5) On Feature specific development branches, automated unit tests and integration tests are run after each change
 - a) These are pre-wash tests (testing only the specific features in the branch)
 - b) If there is need for fixes, commits are pushed directly into this branch
 - c) Amend commits for further changes on the same part of code
 - d) Some feature specific branches will be used for soaking tests (to test non-functional requirements)
- 6) When development of the features is done and all tests from the previous step confidently pass, the pull request to Development can be ordered
- 7) Before merging into Development all legacy tests are run, if any fails, pull request is denied
- 8) Once merged into Development, integration tests are run again
 - a) If there are bugs at this point, merge is reverted and development branch rebased for fixing
- 9) When the Development branch is correctly working and ready to be deployed it is merged into Master branch



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