[Date]

Sem Plasmeijer

Fontys Hogescholen ICT

Design Document

ScrimFinder

Contents

[1. Project description 2](#_Toc129007693)

[1.1 challenges 2](#_Toc129007694)

[2. user stories & requirements 3](#_Toc129007695)

[2.1 user stories 3](#_Toc129007696)

[2.2 Non-functional requirements 4](#_Toc129007697)

[2.3 Definition of done 4](#_Toc129007698)

[3. Architecture & C4 models 5](#_Toc129007699)

[3.1 Framework 5](#_Toc129007700)

[3.1.1 Front-end 5](#_Toc129007701)

[3.1.2 Back-end 5](#_Toc129007702)

[3.2 Architecture 6](#_Toc129007703)

[3.2.1 C1 6](#_Toc129007704)

[3.2.2 C2 7](#_Toc129007705)

[3.2.3 C3 8](#_Toc129007706)

[3.2.4 C4 9](#_Toc129007707)

[4. Security 10](#_Toc129007708)

[5. Data 11](#_Toc129007709)

[6. Testing 12](#_Toc129007710)

[7. Wireframes 13](#_Toc129007711)

[8. Documentation 14](#_Toc129007712)

[8.1 scrimfinder-docker 14](#_Toc129007713)

[8.2 scrimfinder-fe 14](#_Toc129007714)

[8.3 scrimfinder-api 14](#_Toc129007715)

[8.4 Scrimfinder-wq 14](#_Toc129007716)

# Project description

Water polo is a niche sport in the Netherlands. Unlike football, hockey, or tennis where nearly each town might have one club of the 3 sports and big cities might have multiple within the same sport. This results in clubs easily finding opponents to play practice matches outside the main competition. Water polo on the other hand besides having a smaller group of athletes also has the problem that it is restricted to towns and cities with a swimming pool which is also competition approved. This combination leads to difficulty finding practice matches, the idea for this project is an integrated system where teams can find each other to play practice matches, contact each other, find relevant personnel like referees, keep track of matches themselves and create an environment which promotes competitive play among  
the regions.

## challenges

The system consists of multiple moving parts, contacting (either through email service or in application communication) and finding a match, agreeing on the terms of a match, a batch system where if more than 3 teams agree can create a league or tournament for practice, a live match updater for tracking match data, an online ranking system like ELO in chess, a location for match data to be stored and lastly if possible for my individual research into AI and ML I might integrate this into the match updating system.

# user stories & requirements

## user stories

* As a user I want to be able to create an account for my club, so that I can separate my team and its players.
* As a user I want to be able to create an account for my player, so that I can separate my team and its players.
* As a user I want to be invited or join my own club, so that I will be notified when the team status changes.
* As a user I want to change or edit teams, so that I am stuck with one club.
* As a user I want to create a match with another team, so we can play each other.
* As a user I want to be able to give specific information regarding a match like location, time and who is available.
* As a user I want to assign people to manage a match, so I can create a fair match environment.
* As a user I want to create multiple matches at once, so I can create a tournament or league.
* As a user I want to be able to change the format of a tournament or league, so I am not stuck with one format.
* As a user I want to be able to see which teams have won the most, so I can plan evenly matched scrims.
* As a user I want to be able to receive a mail for planned events, so I can be notified in time.
* As a user I want to message other teams and users, so I won’t need to use an external tool to communicate.

## Non-functional requirements

* Performance
  + Total time to create 1000 matches is within 10 seconds.
  + Within 24 hours a notification should be send for an upcoming match
  + Within 24 hours a notification should be send regarding the creation of matches.
  + A message should be sent and received within the second.
* Scalability
  + a
* Security
  + The system should carefully handle data regarding players and teams.
* Privacy
  + Player information is stored without made public.
* Maintainability
  + A service will have its own responsibility.
* Reliability
  + Uptime of 1%.
  + Down time during patches of 5 minutes.
  + Test coverage of 80%

## Definition of done

A story or product is finished when:

* All tests have passed.
* Documentation has been added regarding the feature.
* New tests have written for the story if needed.
* Has been tested manually.
* Test coverage of project is still 80%.

# Architecture & C4 models

## Framework

### Front-end

Front end work will be done in Vue.JS as it is what I have been more confident with and allows for easy single page applications. Further due to its flexibility I can more easily change different parts of the system when I find new techniques during the semester.

### Back-end

The decision came down to SpringBoot Java and Django Python, Django allows for quick development cycles and easy deployment. While SpringBoot has a more robust security system and works better within smaller services.

SpringBoot also has integrated dependencies with RabbitMQ. Which will be used for its worker-queue system it was chosen over alternatives due to the aforementioned integration. But also comes with an easy to setup deployment system through Docker.

## Architecture

### C1

The scrimFinder system is a web-based application which allows different users to review, find and connect with others and their matches. An external service will be used to send notification regarding matches.

### C2

### Diagram, schematic Description automatically generatedC3

The system consist out of six parts. An over arcing Rest API which calls the different sources. Which are an authorization system with its own database storage. An worker-queue API connected to a RabbitMQ server. This server creates batches of data which will be stored. An information API will read data regarding the status of batches but also individual match information. Lastly a notification system and messaging system for live match updates.

### C4

This part is separately added to each codebases sub-chapter in [chapter 8: Documentation](DesignDocument.docx#Documentation).

# Security

# Data

# Testing

# Wireframes

# Documentation

## scrimfinder-docker

## scrimfinder-fe

## scrimfinder-api

## Scrimfinder-wq