Project Manual

Mateo Sierens

January 2021

1 Introduction

For this project the task was to make a site based on microservices for the L-V-L. You can run my project by running the install.sh file, which will automatically deploy the docker containers and start the project. If you go to localhost you should be able to see my homepage.

2 Architecture

I decided to split the architecture up into 4 + 1 services: Matches, Clubs, Users, Client + nginx. I said + 1 because I don't consider nginx like the rest of the microservices. For the sake of simplicity I will act as if nginx is not there when talking about the architecture, as the focus lies on the microservices I implemented. In figure 1 you can see my architecture. As you can see every request starts from client and ends in client. I chose for this centralised design specifically so I would have an easy overview of all the API calls to outside. In the end I think it would've been easier to not do that but it definitely has it advantages in certain situations. The double arrows might be a little confusing: client sends the requests, the arrows back are the replies from each service. The services itself don't send any requests.

2.1 Users

This service keeps all data necessary on users. A user has a username, password, email, and can be referenced to a club. I keep 2 boolean values to check whether the user is an admin, and whether the user is a super admin. An admin can perform CRUD operations on all entities except users. The superadmin can perform operations on users as well. The files contain the implementation of the CRUD operations, the database models and some tests.

2.2 Clubs

This service processes everything for clubs and teams. Clubs have a stamnumber that identifies them, a name, an address, a zipcode, a city and a website. Teams keep a stamnumber that references the club they belong to, a suffix and the sweater colors. The files contain CRUD operations on both teams and clubs, the database models and some tests.

2.3 Matches

This service is the biggest one yet, it contains everything needed for matches: status, divisions and referees, as well as matches itself. The status and division tables just contain the names. The referee table keeps first name, last name, address, zipcode, city phone number email and birth date. Lastly, the match table keeps the divisison, matchweek, date, time, home team playing, away team playing, goals from the home team, goals from the away team, the match status and referee. Again, the files include CRUD operations for every entity, the database models and some tests.

2.4 Client

This service does not contain a database. This service stands in for rendering all the front end templates and handling accordingly to the input given by users. It sends requests to all the other services, as well as API calls to the weather service for the upcoming matchweek.
Under "competitie" you can find all the fixtures for every division. It contains the league table for the given division, with filters on matchweek and team. When clicking on more info we can find more info on a specific fixture.
In the "clubs" tab we can get info about every club in the database. Most team only have on team, which is referenced to as "Hoofdteam". When a club has multiple teams, it will address these with their suffix. When clicking on a team you get more info about a team, as well as a fixture for the upcoming matches and results for the last 3 matches under the form of 'LWD'.
When logged in into an account linked to a club, one can go to the score tab. From there a user gets an overview of all matches played at home, and the user can add scores for every match.
Lastly when a user is logged in into an admin account, the user is able to see the "admin" tab. Here all CRUD operations can be performed. Due to lack of time, the operations for referee and clubs are implemented but not supported through the admin interface.

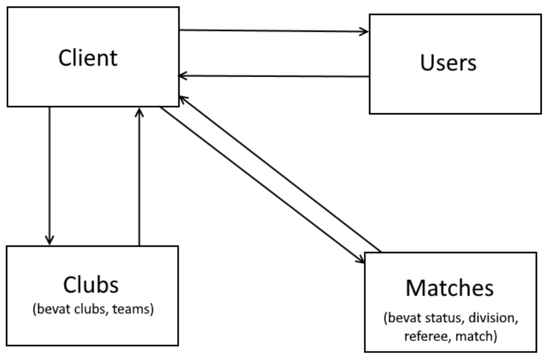

Figure 1: Architecture

3 Quick setup

In the root folder of the project you can find a install.sh file. If you run this it will install docker for you and start it up. It will automatically build the containers, which install the requirements of the requirements.txt of each service. If you go to localhost you should see the homepage of the site. In the right upper corner you can click 'Login' to log in. The following account has all permissions:

```
username: test
password: test
```

4 Endpoints

4.1 Users

- POST /users (success)

```
{
    'status': 'success'
    'message': '{email} was added!'
}
```

- POST /users (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- POST /users (existing email)

```
{
    'status': 'fail'
    'message': 'Sorry. That email already exists'
}
```

- GET /users/<user_id>(success)

```
{
    'status': 'success'
    'data': User object
}
```

- GET /users/<user_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'User does not exist'
}
```

- GET /users/ (success)

```
{
    'status': 'success'
    'data': '{
        users': [list of User objects]
    }
}
```

- PUT /users/<user_id>(success)

```
{
    'status': 'success'
    'message': 'User updated.'
}
```

- PUT /users/<user_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /users/<user_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'User does not exist'
}
```

- PUT /users/<user_id>(double email)

```
{
    'status': 'fail'
    'message': 'Sorry. That email already exists.'
}
```

- PUT /users/<user_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /users/<user_id>(success)

```
{
    'status': 'success'
    'message': 'User deleted'
}
```

- DELETE /users/<user_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'User does not exist'
}
```

- DELETE /users/<user_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete user'
}
```

4.2 Teams

- POST /teams/ping (success)

```
{
    'status': 'success'
    'message': 'pong!'
}
```

- POST /teams (success)

```
{
    'status': 'success'
    'message': 'team successfully created!'
}
```

- POST /teams (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- POST /teams (nonexisting club)

```
{
    'status': 'fail'
    'message': 'No club found with that stamnumber.'
}
```

- GET /teams/<team_id>(success)

```
{
    'status': 'success'
    'data': Team onject
}
```

- GET /teams/<team_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Team does not exist'
}
```

- GET /teams/ (success)

```
{
    'status': 'success'
    'data': '{
        teams': [list of Team objects]
    }
}
```

- PUT /teams/<team_id>(success)

```
{
    'status': 'success'
    'message': 'Team updated.'
}
```

- PUT /teams/<team_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /teams/<team_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Team does not exist'
}
```

- PUT /teams/<team_id>(nonexistant club)

```
{
    'status': 'fail'
    'message': 'No club found with that stamnumber.'
}
```

- PUT /teams/<team_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /teams/<team_id>(success)

```
{
    'status': 'success'
    'message': 'Team deleted'
}
```

- DELETE /teams/<team_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Team does not exist'
}
```

- DELETE /teams/<team_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete team'
}
```

- POST /clubs/ping (success)

```
{
    'status': 'success'
    'message': 'pong!'
}
```

- POST /clubs (success)

```
{
    'status': 'success'
    'message': 'Club successfully created!'
}
```

- POST /clubs (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- GET /clubs/<club_id>(success)

```
{
    'status': 'success'
    'data': Club object
}
```

- GET /clubs/<club_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Club does not exist'
}
```

- GET /clubs/ (success)

```
{
    'status': 'success'
    'data': '{
        clubs': [list of Club objects]
    }
}
```

- PUT /clubs/<club_id>(success)

```
{
    'status': 'success'
    'message': 'Club updated.'
}
```

- PUT /clubs/<club_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /clubs/<club_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Club does not exist'
}
```

- PUT /clubs/<club_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /clubs/<club_id>(success)

```
{
    'status': 'success'
    'message': 'Club deleted'
}
```

- DELETE /clubs/<club_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Club does not exist'
}
```

- DELETE /clubs/<club_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete club'
}
```

4.3 Matches

- POST /referees/ping (success)

```
{
    'status': 'success'
    'message': 'pong!'
}
```

- POST /referees (success)

```
{
    'status': 'success'
    'message': 'Referee successfully created!'
}
```

- POST /referees (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- GET /referees/<referee_id>(success)

```
{
    'status': 'success'
    'data': Referee object
}
```

- GET /referees/<referee_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Referee does not exist'
}
```

- GET /referees/ (success)

```
{
    'status': 'success'
    'data': '{
        referees': [list of Referee objects]
    }
}
```

- PUT /referees/<referee_id>(success)

```
{
    'status': 'success'
    'message': 'Referee updated.'
}
```

- PUT /referees/<referee_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /referees/<referee_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Referee does not exist'
}
```

- PUT /referees/<referee_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /referees/<referee_id>(success)

```
{
    'status': 'success'
    'message': 'Referee deleted'
}
```

- DELETE /referees/<referee_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Referee does not exist'
}
```

- DELETE /referees/<referee_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete referee'
}
```

- POST /divisions/ping (success)

```
{
    'status': 'success'
    'message': 'pong!'
}
```

- POST /divisions (success)

```
{
    'status': 'success'
    'message': 'Division successfully created!'
}
```

- POST /divisions (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- GET /divisions/<division_id>(success)

```
{
    'status': 'success'
    'data': Division object
}
```

- GET /divisions/<division_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Division does not exist'
}
```

- GET /divisions/ (success)

```
{
    'status': 'success'
    'data': '{
        divisions': [list of Division objects]
    }
}
```

- PUT /divisions/<division_id>(success)

```
{
    'status': 'success'
    'message': 'Division updated.'
}
```

- PUT /divisions/<division_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /divisions/<division_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Division does not exist'
}
```

- PUT /divisions/<division_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /divisions/<division_id>(success)

```
{
    'status': 'success'
    'message': 'Division deleted'
}
```

- DELETE /divisions/<division_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Division does not exist'
}
```

- DELETE /divisions/<division_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete division'
}
```

- POST /status/ping (success)

```
{
    'status': 'success'
    'message': 'pong!'
}
```

- POST /status (success)

```
{
    'status': 'success'
    'message': 'Status successfully created!'
}
```

- POST /status (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- GET /status/<status_id>(success)

```
{
    'status': 'success'
    'data': Status object
}
```

- GET /status/<status_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Status does not exist'
}
```

- GET /status/ (success)

```
{
    'status': 'success'
    'data': '{
        status': [list of Status objects]
    }
}
```

- PUT /status/<status_id>(success)

```
{
    'status': 'success'
    'message': 'Status updated.'
}
```

- PUT /status/<status_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /status/<status_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Status does not exist'
}
```

- PUT /status/<status_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /status/<status_id>(success)

```
{
    'status': 'success'
    'message': 'Status deleted'
}
```

- DELETE /status/<status_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Status does not exist'
}
```

- DELETE /status/<status_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete status'
}
```

- POST /matches/ping (success)

```
{
    'status': 'success'
    'message': 'pong!'
}
```

- POST /matches (success)

```
{
    'status': 'success'
    'message': 'Matches successfully created!'
}
```

- POST /matches (wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload.'
}
```

- GET /matches/<match_id>(success)

```
{
    'status': 'success'
    'data': Match object
}
```

- GET /matches/<match_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Status does not exist'
}
```

- GET /matches/ (success)

```
{
    'status': 'success'
    'data': '{
        'matches': [list of match objects]
    }
}
```

- PUT /matches/<match_id>(success)

```
{
    'status': 'success'
    'message': 'Status updated.'
}
```

- PUT /matches/<match_id>(Wrong input)

```
{
    'status': 'fail'
    'message': 'Invalid payload'
}
```

- PUT /matches/<match_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Match does not exist'
}
```

- PUT /matches/<match_id>(Invalid input)

```
{
    'status': 'fail'
    'message': 'Invalid input data'
}
```

- DELETE /matches/<match_id>(success)

```
{
    'status': 'success'
    'message': 'Match deleted'
}
```

- DELETE /matches/<match_id>(wrong ID / invalid ID)

```
{
    'status': 'fail'
    'message': 'Match does not exist'
}
```

- DELETE /matches/<match_id>(Database exception)

```
{
    'status': 'fail'
    'message': 'Failed to delete match'
}
```