Specification - Sports Club App

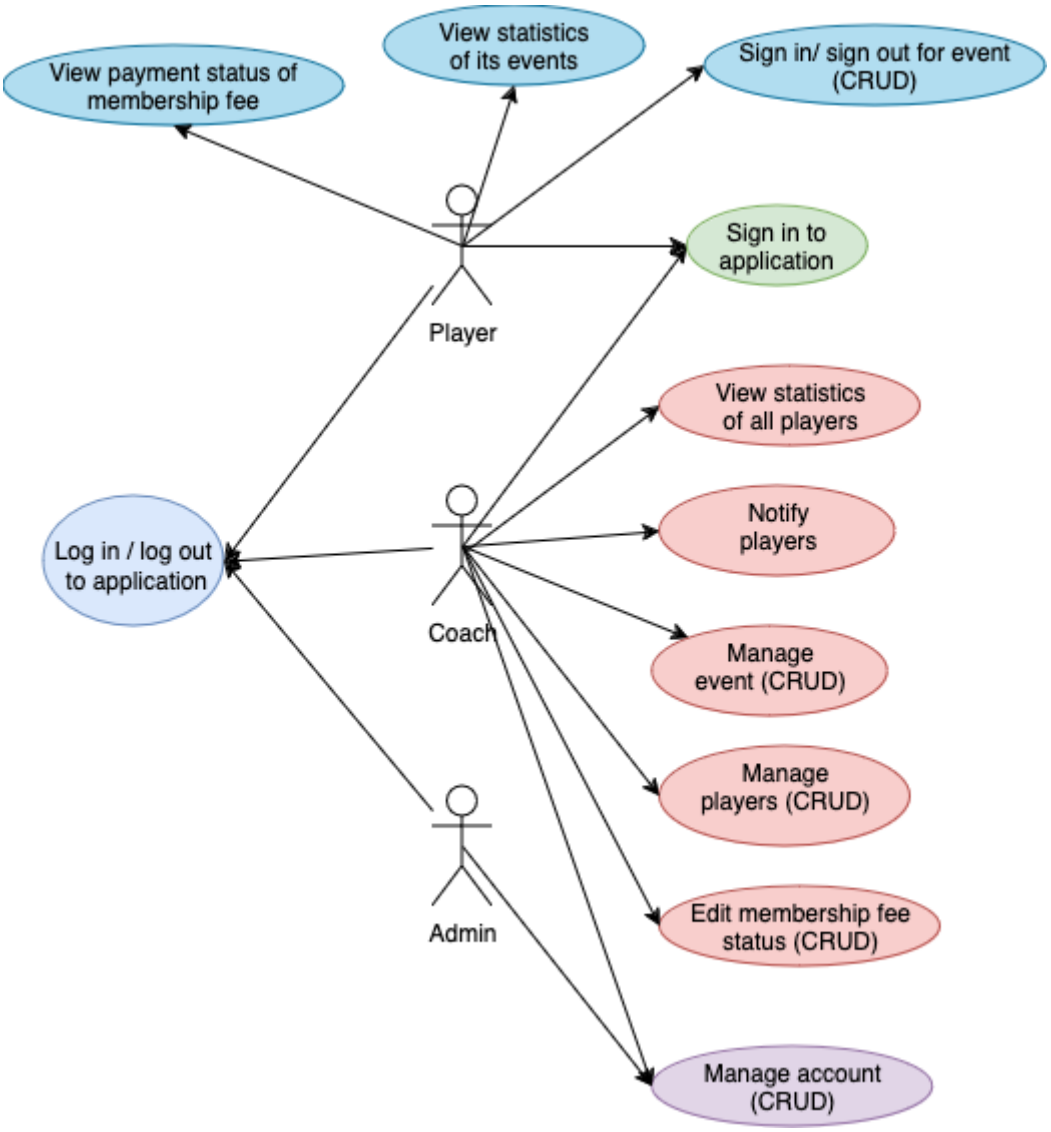
The goal is to create a simple application that would be helpful in the management of a floorball club FBK Raptors. In the app, every player will be able to sign up for training, and matches, also he will be able to see an overview of training and matches where he is and was signed up. Also, the player will see in the app the information about the payment of the membership fee after successful registration and the app will offer the possibility to generate a QR code for payment. Only the user who is allowed by the coach will be able to register in the app. Additionally, the coach will be able to add and change practices and games, mark paid memberships, and have an overall view of all players' participation statistics.

Functional requirements

Roles

- **Player** - club player. Can log in/out of training sessions and matches. He will see an overview of training sessions and matches where he is logged in. Will see information about membership payment (paid/unpaid) and will be able to generate a QR code for payment in case of unpaid membership.
- **Coach** - coach of the club. He will be able to add/remove training sessions and matches, and also change their times. He will be able to send notifications to players if they have not yet expressed their participation. He will have an overview of participation statistics. Mark paid memberships to individual players. The coach can also add and delete team players.
- **Admin** - will change the roles of players/coaches in agreement with the club. In agreement with the coach, will only allow selected people to register, so that not just anyone can register for the club. The team will be able to have several admins, one of them will be the coach himself.

Use Case diagram



Data model

Roles

- list of role types (player/coach/admin):
  - `role_id` = role identifier
  - `description` = role description

EventTypes

- list of event types (training/match):
  - `type_id` = event identifier
  - `description` = event description

Teams

- list of teams:
  - `team_id` = team identifier
  - `name` = team name

Members

- list of specific application users:
  - `member_id` = member identifier
  - `team_id` = the user's team (from table `Teams`)
  - `name`
  - `surname`
  - `birth`
  - `email`
  - `is_active` = True/False whether the player is currently part of the team or not

MembershipFee

- information on payment of the membership fee of registered players:
  - `fee_id` = fee identifier
  - `member_id` = member identifier (from table `Members`)
  - `amount` = how much money has to be/was paid
  - `has_paid` = True/False
  - `date`

Events

- list of specific created events:
  - `event_id` = identifier of the created event
  - `type_id` = type of the event (from table `EventTypes`)
  - `team_id` = team that created the event (from table `Teams`)
  - `name` = name of the event
  - `location` = location of the event
  - `date`

EventsMember

- list of members who have expressed their participation in the event
  - `id` = record identifier
  - `event_id` = identifier of the event (from table `Events`)
  - `member_id` = identifier of the user (from table `Members`)
  - `response` = response of the user (participate/not participate)

Accounts

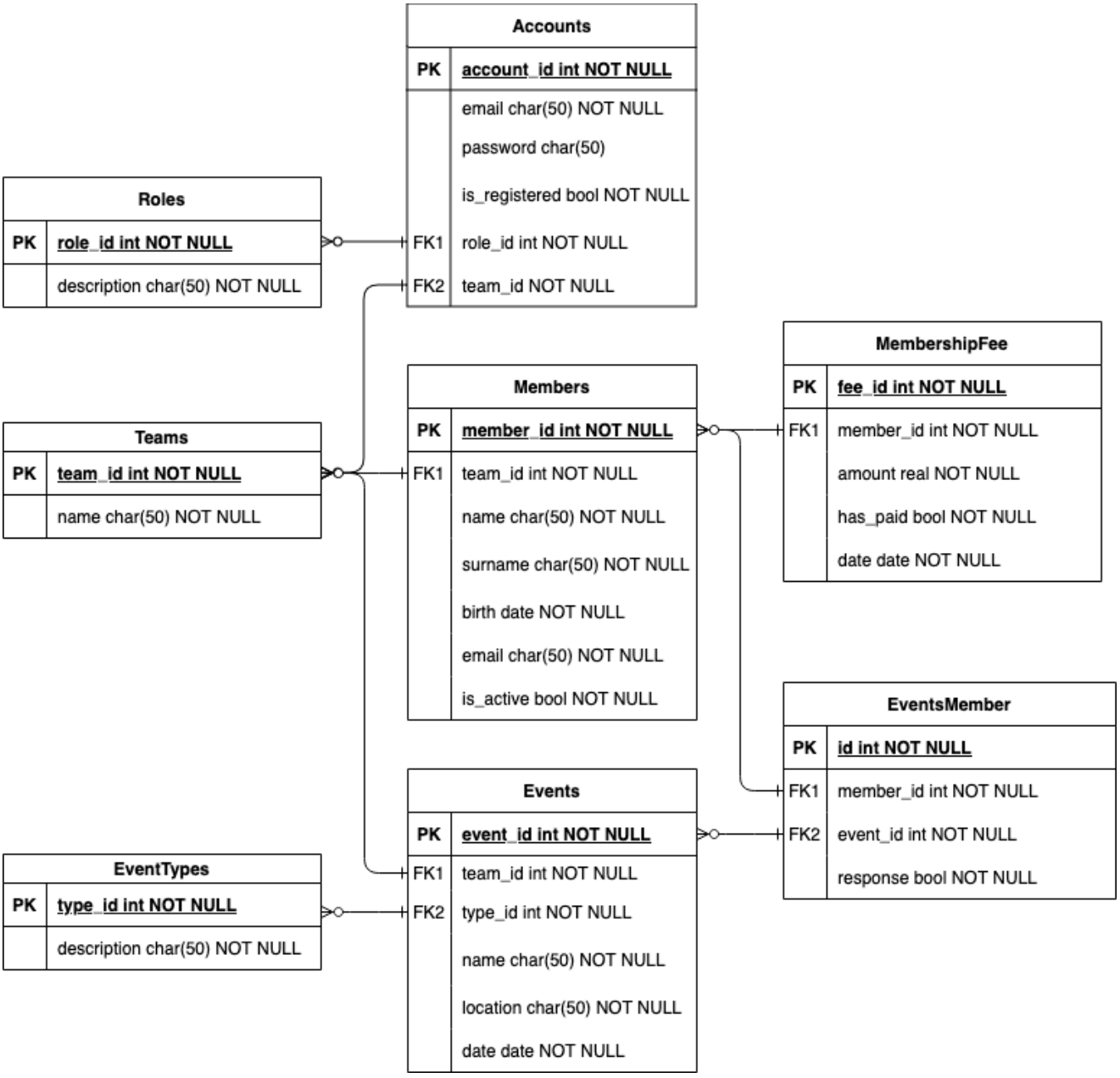
- list of people who can register to the app. Admins added at database level. Players/coach for a specific team (records added by the admin in agreement with the club/coach) added in admin interface. Only people in this table will be allowed to register. This table will be used only for registration and login of members.

- `account_id` = identifier of the account
- `email`
- `password` = contains hash values, `NULL` at the beginning
- `is_registered` = default `False`
- `team_id` = identifier of the team (from table `Teams`), `NULL` at the beginning
- `role_id` = the role of the team member (from table `Roles`)

Notes:

- *When a player is removed from the team, the record `is_active = False`, `is_registered = False` , etc.*
- *Only coach and admin can edit in admin interface. I'll check what mail is logged in, what role it has in the accounts table and if it is admin or coach, it can go to the admin interface.*

ER diagram



Architecture

The application will be based on the client-server architecture and use PWA (Progressive Web App), so we can send the notifications straight to mobile phones.

Technological requirements

- **Client-side**: React (framework), JavaScript, HTML5, CSS3
- **Server-side**: Ruby on Rails/Django (framework), Ruby/Python
- **Database**: PostgreSQL
- **Interface client - server**: Rest API
- **Hosting**: https://sports-club-app.onrender.com/
- **Supported browsers**: Chrome, Firefox, Safari

Notes:

- *I'll see what I choose on the Server-side after studying both frameworks in more detail.*

Timeline

1. week = studying frameworks
2. week = backend -> db models
3. week = backend -> logic
4. week = backend -> prepare views/endpoints
5. week = frontend -> prepare graphic design of the interfaces + interfaces implementation
6. week = frontend -> interfaces implementation
7. week = PWA
8. reserve

Notes:

- *Deployment to the server will be done continuously after each new implementation.*
- *Weeks numbered from 1 after the date 21.3.*

Future work

The application will be created for one specific sports club listed in the introduction, but I will take into account that in the future the application could be useful for other teams and could be easily extended for them (adding more teams to the `Teams` table and possibly other necessary things). If there is enough time, this can be worked into the project as well, but I prefer to think of it as future work.