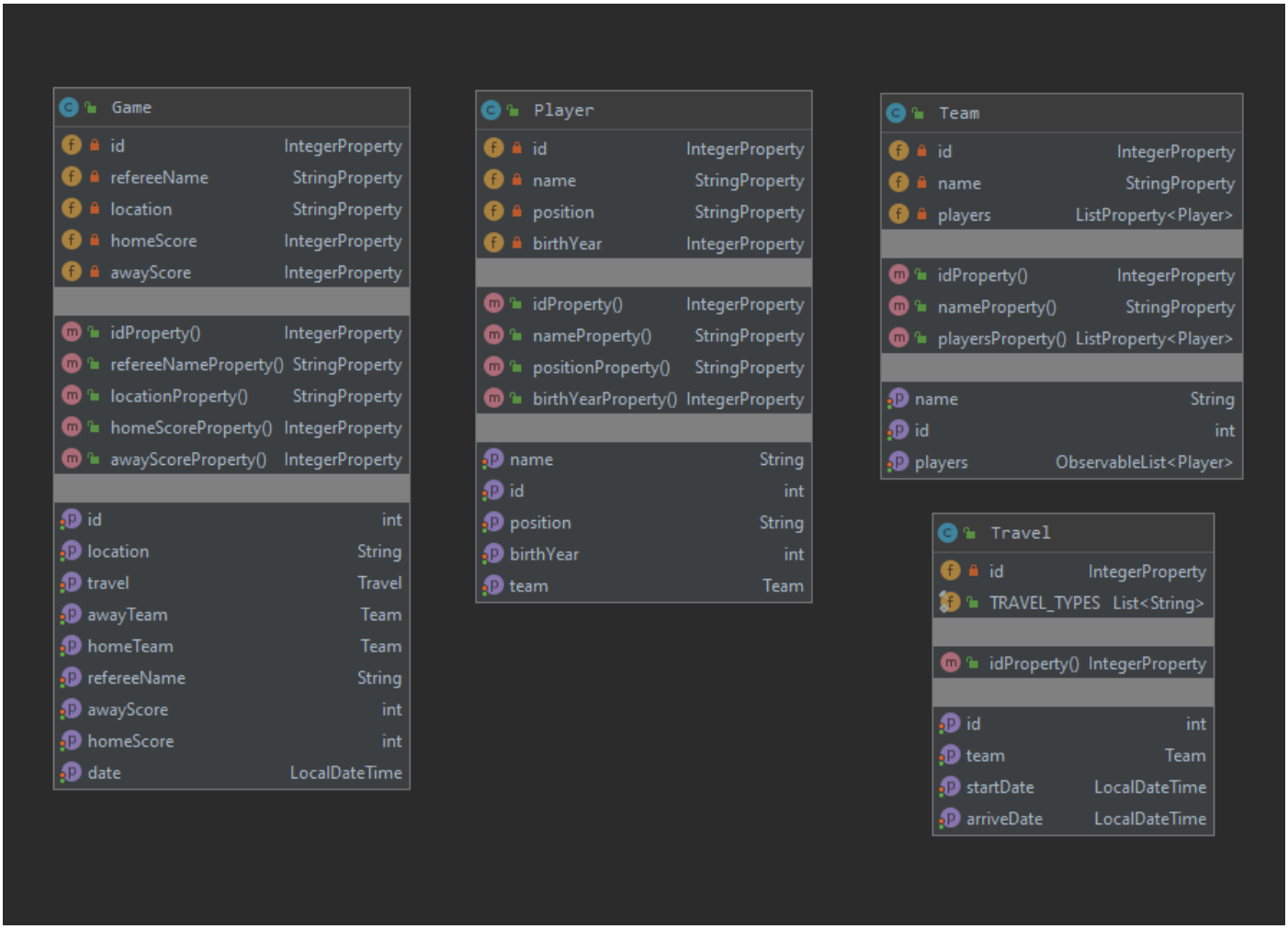
Team manager documentation

I thought about and then gathered what data and the relationships between them could emerge in the models within the project. Based on these, I designed the database and wrote the scripts for the tables. The other scripts (SELECT, INSERT, UPDATE, DELETE) can be found in the DAO part of the applications.

Once I was done with this, I completed a small part of the task, which is responsible for creating, listing, and modifying teams and players. I did this as a C # desktop application using Winforms.

Finally, I create the other half of the task as a JavaFX desktop application. Here I used FXML for the interface. Documentation is provided by in-program comments for both applications.

UML Class Diagram (Models)



Database

The structure of the tables may change during implementation.

A table for storing players. With the team_id data member, I keep track of which player belongs to which team.

```
CREATE TABLE IF NOT EXISTS players (
        id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT,
        position TEXT,
        birth_year INTEGER,
        team_id INTEGER,
        FOREIGN KEY(team_id) REFERENCES teams(id)


);
```

Team storage table.

```
CREATE TABLE IF NOT EXISTS teams (
        id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT


);
```

A table for storing messages between players. With the data member sender_player_id I consider the sender of the message, and with the data member receiver_player_id I consider the recipient of the message.

```
CREATE TABLE IF NOT EXISTS message_player (
        id INTEGER PRIMARY KEY AUTOINCREMENT, sender_player_id INTEGER,
        receiver_player_id INTEGER,
        title TEXT,
        message TEXT,
        FOREIGN KEY(sender_player_id) REFERENCES players(id), FOREIGN
        KEY(receiver_player_id) REFERENCES players(id)
```

Travel storage table. With the team_id data member, I count the team belonging to the trip.

```
CREATE TALBE IF NOT EXISTS travels (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    team_id INTEGER,
    start_date TEXT,
    arrive_date TEXT,
    type TEXT,
    FOREIGN KEY (team_id) REFERENCES teams (id)
);
```

Games storage table. With the data member home_team_id I count the home team, with away_team_id the away team, and with away_team_travel_id I count the travel information of the away team.

```
CREATE TABLE IF NOT EXISTS games (
    id INTEGER PRIMARY KEY AUTOINCREMENT, home_team_id INTEGER,
    away_team_id INTEGER,
    date TEXT,
    referee_name TEXT,
    location TEXT,
    home_score INTEGER,
    away_score INTEGER,
    away_team_travel_id INTEGER,
    FOREIGN KEY (home_team_id) REFERENCES teams (id), FOREIGN KEY (away_team_id)
    REFERENCES teams (id), FOREIGN KEY (away_team_travel_id) REFERENCES travels (id)
);
```

This table counts the active and scoring players in a given game.

```
CREATE TABLE IF NOT EXISTS game_info (
    game_id INTEGER,
    home_team_id INTEGER,
    away_team_id INTEGER,
    player_id INTEGER,
    score INTEGER DEFAULT 0,
    FOREIGN KEY (home_team_id) REFERENCES teams (id),
    FOREIGN KEY (away_team_id) REFERENCES teams (id),
    FOREIGN KEY (player_id) REFERENCES players (id)
```