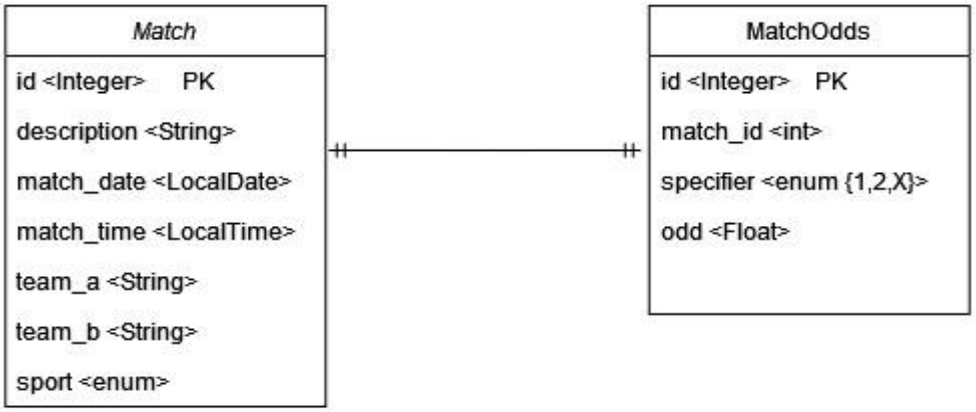
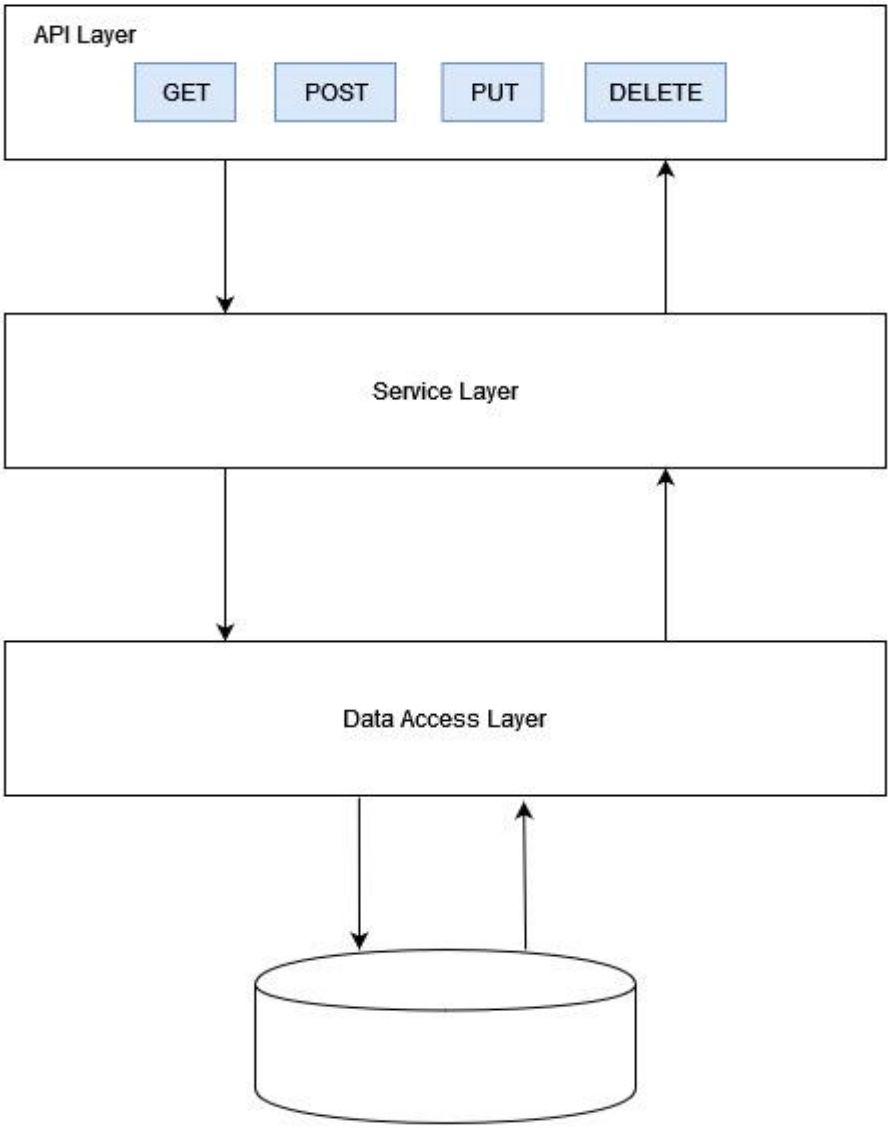
Design Document for the matches-api application

1. Architecture





1.1. API Layer

The API layer is responsible for offering a decoupled interface to the data and the supported functionality of the matches application. It is implemented by the MatchController class which is flagged as a @RestController and used by Spring to handle web requests. It communicates with the Service Layer which implements the business logic and returns JSON responses to the client for all supported operations.

1.1.1. API definition

Method	Path	Body	Explanation
GET	/api/v1/matches	no body	Returns all matches and their betting odds
GET	/api/v1/match/{id}	no body	Returns a specific match and its betting odds
POST	/api/v1/match	Match info provided in request body	Saves a new match and its betting odds
DELETE	/api/v1/match/{id}	no body	Deletes match with id = {id}
DELETE	/api/v1/match?date=<date>	no body	Deletes all matches with match_date before <date>
PUT	/api/v1/match/{id}?date=<date>&time=<time>&odd=<odd>&specifier=<spec>	no body	Updates any of the match_date, match_time, odd, specifier of match with id = {id}. All the query parameters are optional

1.1.2 Example requests

```
###
POST http://localhost:8081/api/v1/match
Content-Type: application/json
{
 "description": "aek-atromitos",
 "team_a": "AEK",
 "team_b": "ATROMITOS",
 "match_date": "2023-12-04",
 "match_time": "17:20:00",
```

```
 "sport": 1,
 "odd": 1.05,
 "specifier": "X"
}
###
DELETE http://localhost:8081/api/v1/match?date=2023-12-04

###
PUT
http://localhost:8081/api/v1/match/3?date=2023-12-04&time=17:20:00&odd=2.0&specifier=A
```

1.2 Service Layer

The Service Layer communicates with the data access layer to fetch the data requested by the API layer. It is also responsible for implementing the business logic and throw custom exceptions on possible violations that are handled by Spring. for example deleting a match that does not exist or adding a duplicate match.

Is is implemented by the MatchService class annotated with the @Service annotation

1.3 Data access Layer

The data access Layer is responsible for communicating with the database. It's implemented by the MatchRepository interface that is an extension of the JPA repository which provides several methods to communicate with postgres. In case of more complex queries, it allows for their definition directly on the interface.

1.4 Entities

The application is modeled with the Match and MatchOdds entities and implemented by the Match and MatchOdds classes respectively, which are annotated with the @Entity and @Table annotations that result in the creation of the respective relations in the database.

Furthermore, Match and MatchOdds are connected with an 1 to 1 relation meaning one record in the db table (match) holding match entities is associated with exactly one record in the db table (matchOdds) holding matchOdds. The match_id field in MatchOdds is a foreign key.

2. Installation

The application can be installed by docker-compose.
You can invoke the utility script provided directly with ./install-app-docker.sh

This produces a fat jar, builds the api application and deploys two containers, one with postgres and one with the matches-api.

The application is publically available at https://github.com/iasonaschr/matches-api

3. Known issues

- The app does not contain any unit tests
- The current document could be more detailed
- Type checking and validation is not that strict and could be enriched using Spring's validation annotations