Name:
Project Name:
Supervised By:

Index

Introduction

This is a simple ticket reservation system for football world cup, from this system you can reserve a chair in the stadium.

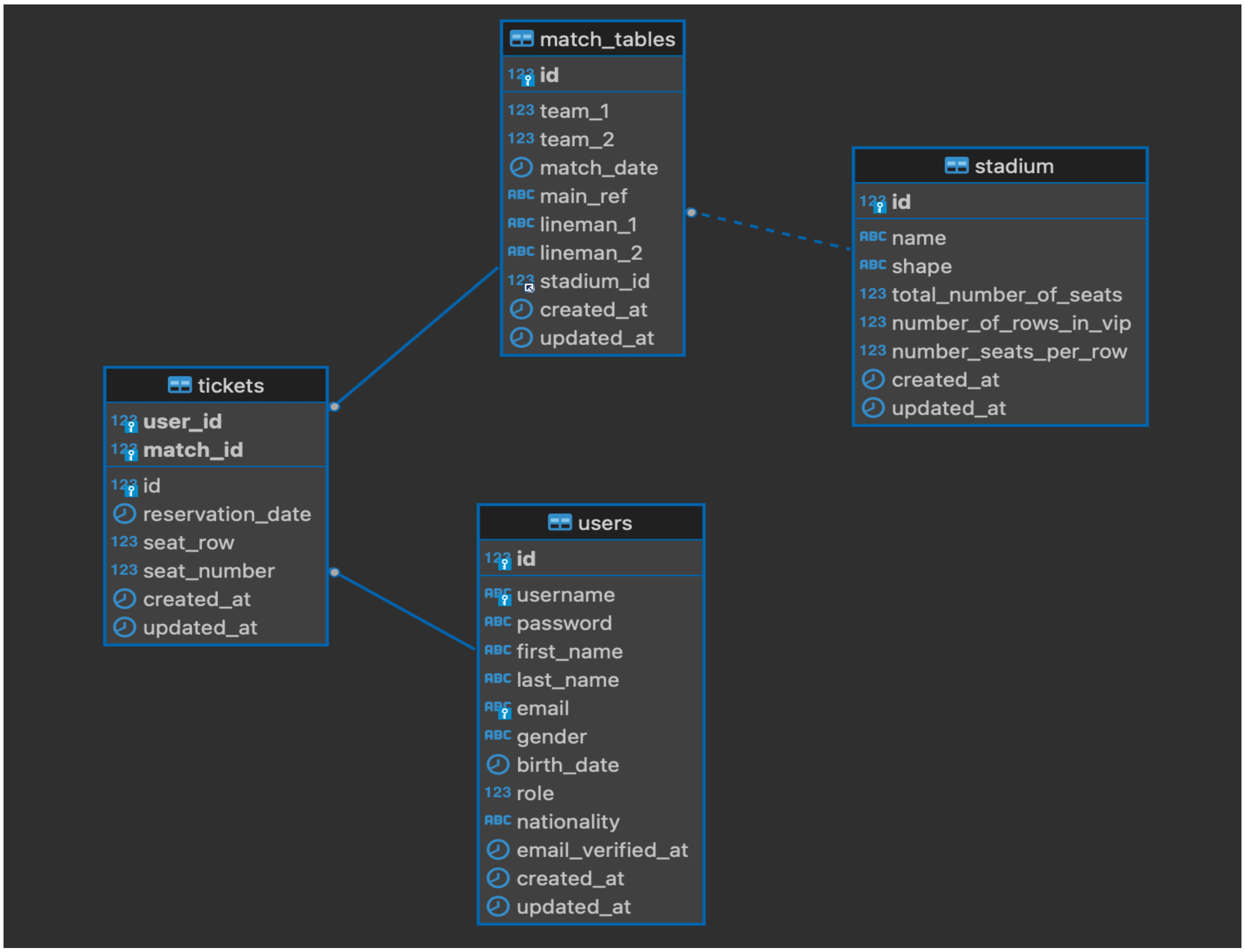
We have 3 roles

Admin: verify users – delete users – see some statistics of system

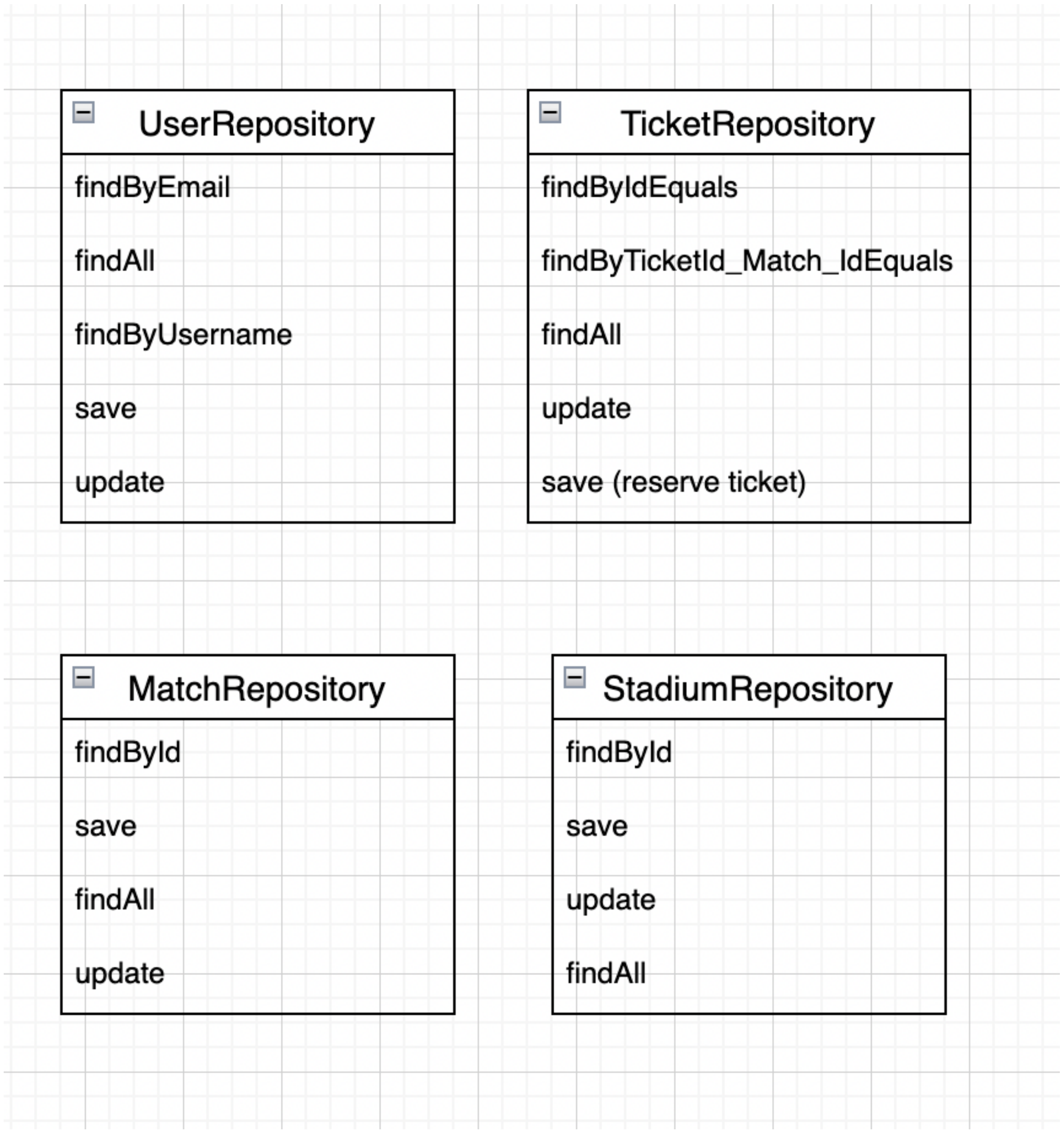
Manager: create matches

Fan: see coming matches – reserve ticket – edit profile

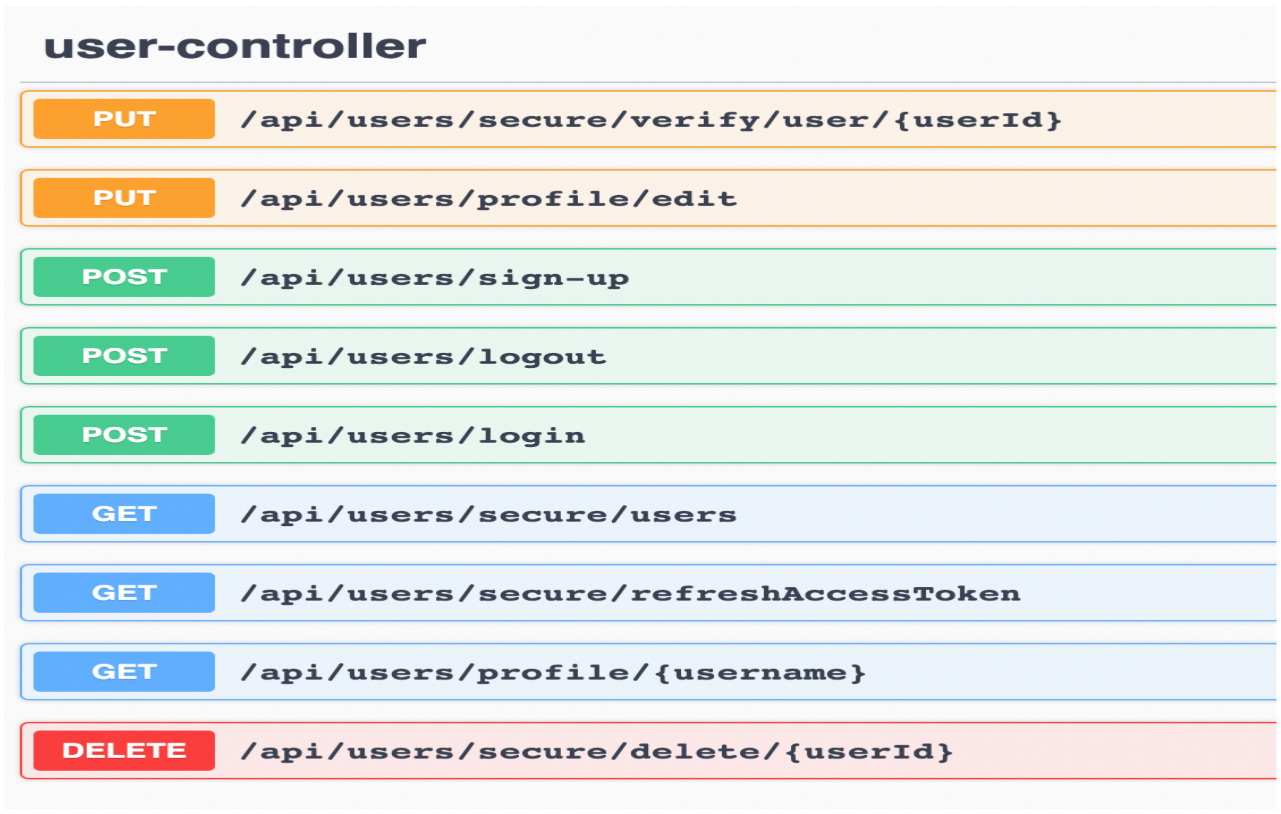
Entities



Repositories



Controllers



user-controller

PUT /api/users/secure/verify/user/{userId}

PUT /api/users/profile/edit

POST /api/users/sign-up

POST /api/users/logout

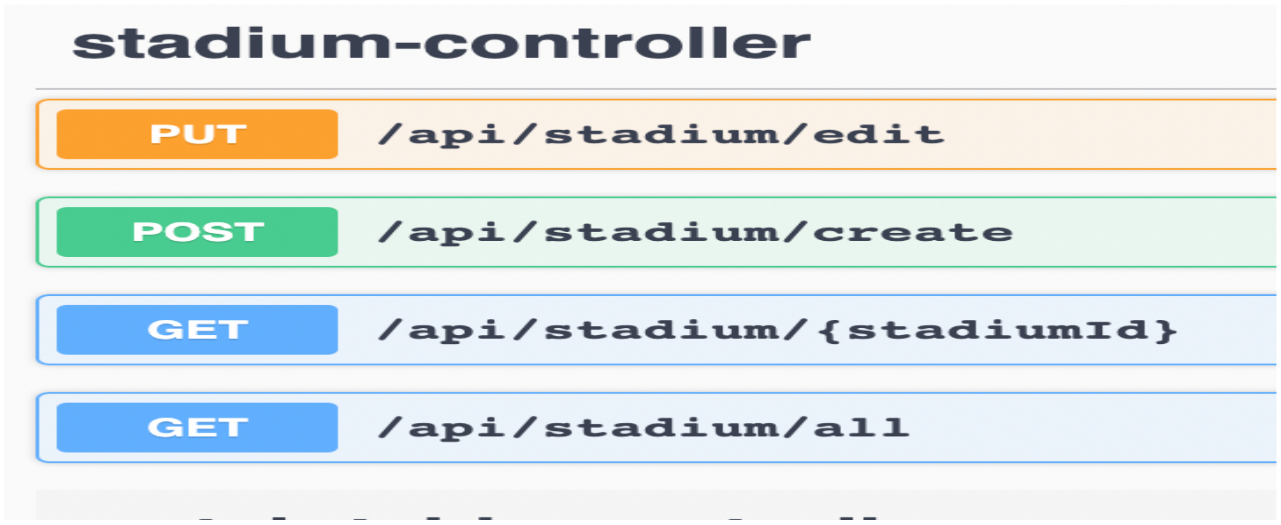
POST /api/users/login

GET /api/users/secure/users

GET /api/users/secure/refreshAccessToken

GET /api/users/profile/{username}

DELETE /api/users/secure/delete/{userId}



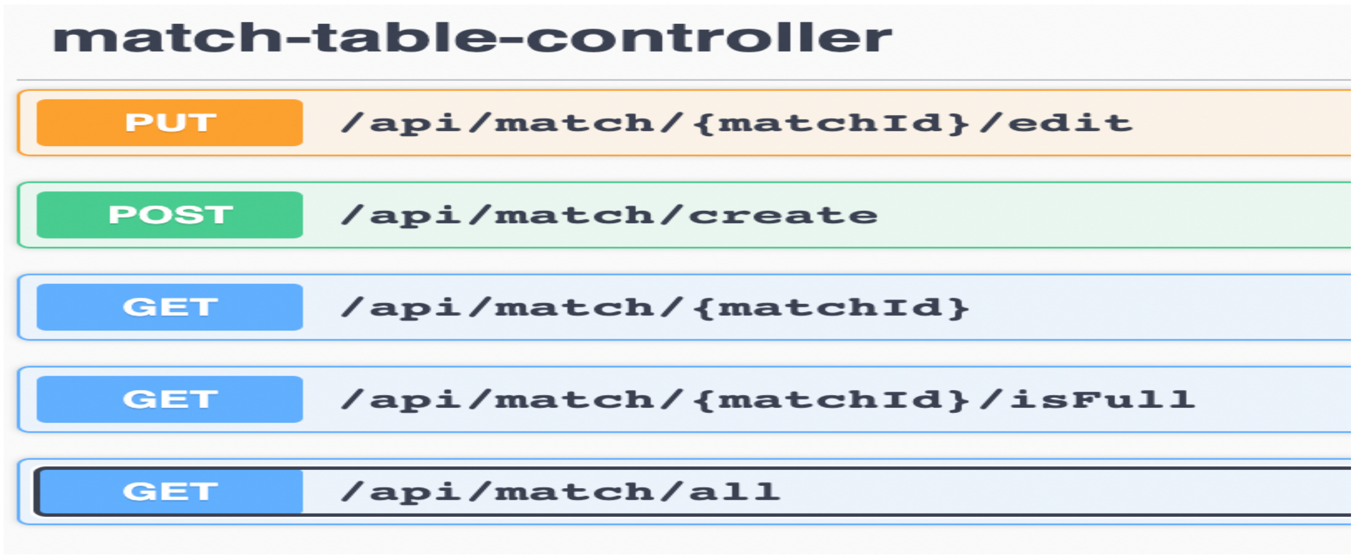
stadium-controller

PUT /api/stadium/edit

POST /api/stadium/create

GET /api/stadium/{stadiumId}

GET /api/stadium/all



match-table-controller

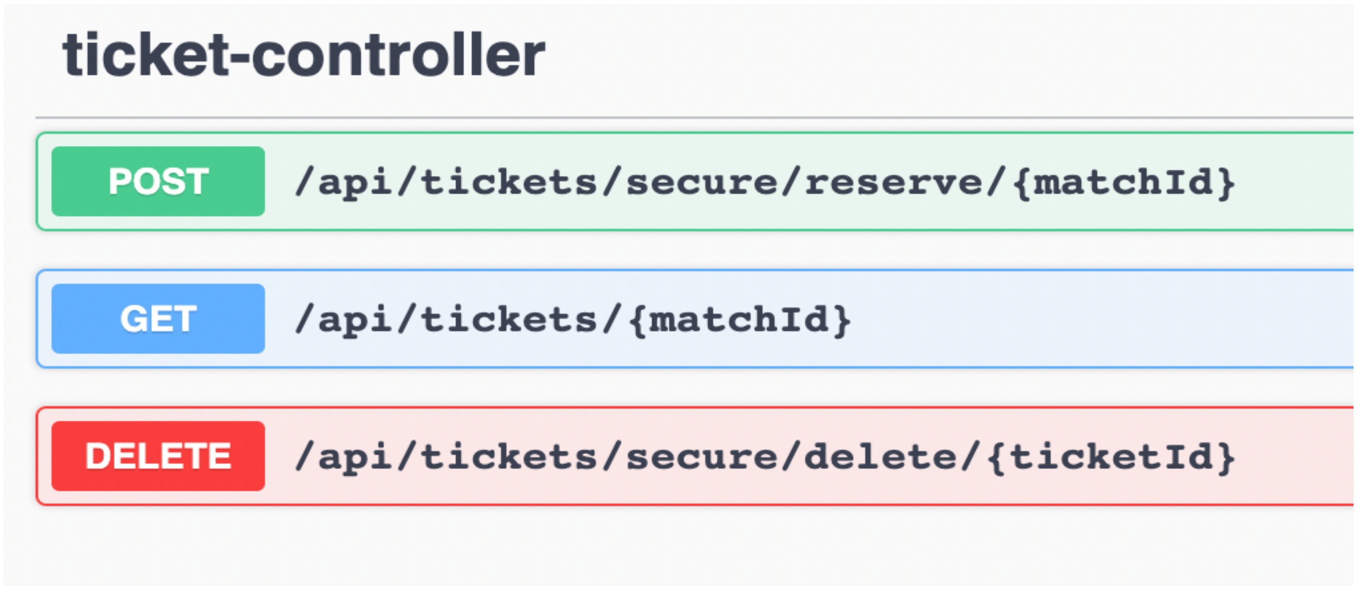
PUT /api/match/{matchId}/edit

POST /api/match/create

GET /api/match/{matchId}

GET /api/match/{matchId}/isFull

GET /api/match/all



ticket-controller

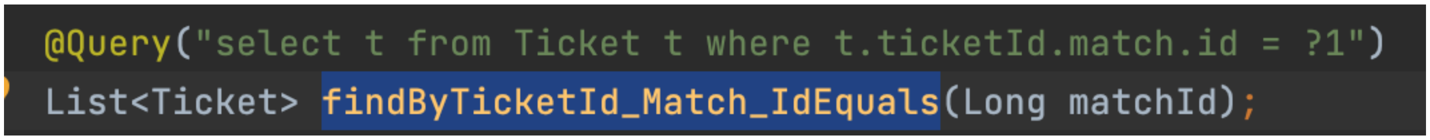
POST /api/tickets/secure/reserve/{matchId}

GET /api/tickets/{matchId}

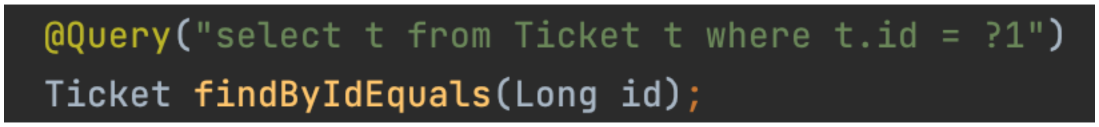
DELETE /api/tickets/secure/delete/{ticketId}

Surpassing the expectations

- Additional generated queries
- Additional native queries (@Query)
- Complex queries (Join,)



```
@Query("select t from Ticket t where t.ticketId.match.id = ?1")
List<Ticket> findByTicketId_Match_IdEquals(Long matchId);
```



```
@Query("select t from Ticket t where t.id = ?1")
Ticket findByIdEquals(Long id);
```

- Additional endpoints
- Use of different Methods (GET, POST, PUT, DELETE, ...)

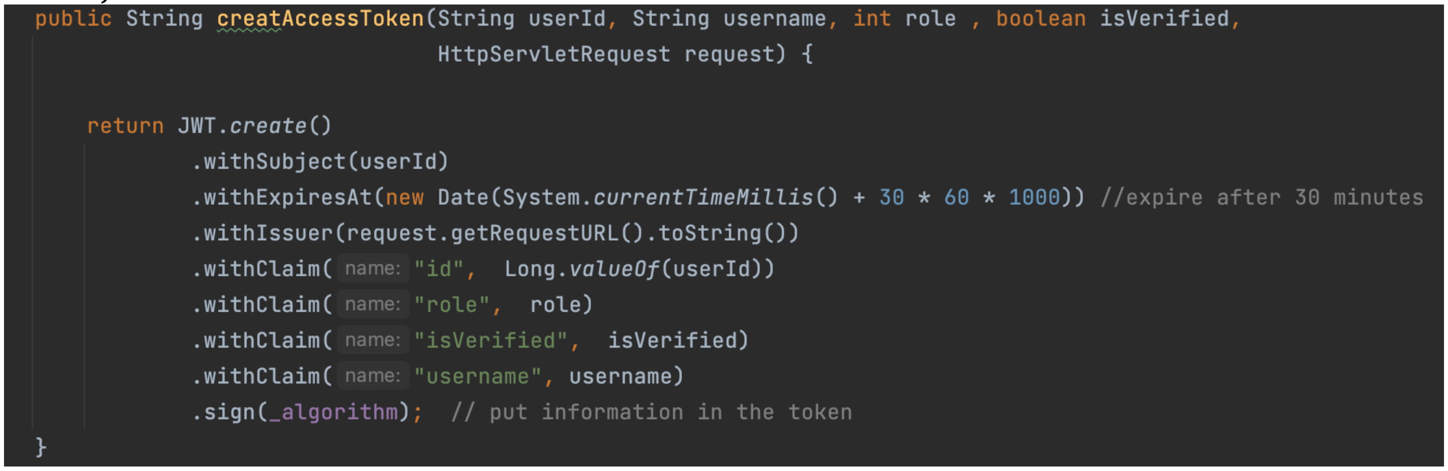
I used more than expected end points in controllers

- Use of additional libraries

1. Lombok (default getter and setter)
2. Auth (for authentications)
3. Spring doc (for swagger controllers)

- Security (Authentication, Access Control, ...)

I used JWT



```
public String creatAccessToken(String userId, String username, int role , boolean isVerified,
                               HttpServletRequest request) {

    return JWT.create()
            .withSubject(userId)
            .withExpiresAt(new Date(System.currentTimeMillis() + 30 * 60 * 1000)) //expire after 30 minutes
            .withIssuer(request.getRequestURL().toString())
            .withClaim( name: "id",  Long.valueOf(userId))
            .withClaim( name: "role",  role)
            .withClaim( name: "isVerified",  isVerified)
            .withClaim( name: "username", username)
            .sign(_algorithm);  // put information in the token
}
```