Applied Research

Date	**:**	**28/03/2022**
Version	**:**	**0.2**
State	**:**	
Author	**:**	**Group 2**

Version history

Version	Date	Author(s)	Changes	State
0.1	10/3/2022	All members	Initial version	Finish
0.2	28/03/2022	All members		Unfinished

Distribution

Version	Date	Receivers

Table of contents

1. Topic

The topic being researched in this document is going to be "*Building a system for buying tickets for NBA games with Spring boot".* To answer this question research will be conducted making use of the DOT framework. By creating a main and sub questions as well as choosing the most suitable research methods I will dive more deeply in this topic.

2. Main question

How do I make a system for buying NBA games tickets which also keeps track of players, teams, games and different statistics?

3. Sub questions

1. What is the best approach for keeping track of live changing statistics and does it suit my project?
2. Best practices in making database designs for saving tickets which are extendable with possible discount codes etc...?
3. Do I need a login system for purchasing a ticket?

4. Research methods

4.1. Sub question 1

In order to answer this question, I will use the following research methods: '**Community research**' and '**Available product analyses**.

4.2. Sub question 2

In order to answer this question, I will use the following research methods: '**Community research**' and '**Best good and bad practices**.

4.3. Sub question 3

In order to answer this question, I will use the following research methods: '**Explore user requirements**' and '**Problem analysis**.

5. Results

5.1. Sub question 1

In an ideal world where I had more time than the deadline for this project the best approach would be to fetch live statistics from an external source like an NBA API. By doing so I ensure the validity of the data and that it is up to date. However, a problem emerges from that solution, in case of a server error in the external source all the live data is lost and the application cannot fetch any statistics. To resolve that I will need to save the data locally (to my database for example) and periodically synchronize my local data with the live data from the external source. After doing some research I found I way to do that using a scheduling support provided by Spring boot. By using the @Scheduled annotation, I can make my application execute tasks every time a certain interval of time passes. The tasks in this case would be the synching of live data with my locally stored data. After that, all I needed was a suitable external source to pull data from. (Rapid API, n.d.):

- Sportspage Feeds
- API-NBA
- Live Sports Odds
- The Rundown
- Free NBA API (balldontlie)
- NBA Stats
- Basketball Data
- Real-Time Basketball Content
- JsonOdds

After looking through the options above I decided to go for Free NBA API (balldontlie) for several reasons. First of all, although all of the options are free to use in the beginning most of them require you to pay subscription after a certain amount of time has passed which is not the case with balldontlie API which is free no matter how much time you have been using it. What is more, this API contains a lot of data such as statistics for players on average or even for different seasons which is more than enough for a full and reliable application which provides all the information the user may need or want to know before eventually buying tickets for games which is the main purpose of the application. The API also has a nice documentation explaining exactly how to connect to it and fetch the data you need.

The next problem comes in the limitations of the balldontlie API. It limits the client so only limited amount of data can be pulled in a single GET request. This means that in order to synchronize all the data I need to make numerous requests to the API which will really slow my application. What is worse is that all these requests need to be executed periodically because the data changes often. That is why I came to a simple solution, which may not be the best one but for this project and the time I have been given to finish it, I think is the best possible one. I will use the external API to fetch the statistics from my front end and won't save them in a database. In case the API is down, I will display an appropriate message to the user. When taking into consideration all there is to this project I think this solution is completely fine given that the statistics are not the main focus of my system.

5.2. Sub question 2

When thinking about storing tickets there are a couple of questions that need to be looked into: What are the tickets for? Is the ticket standard or are there different types of tickets? Until when is the ticket valid etc...

After doing some research I came to the conclusion (as I expected) that there is no specific design or rule to stick to when creating a ticket buying system like that. (Stack overflow, n.d.). For this project I will create several objects, some of which are going to be *User*/*Customer*, *Game*, and *Ticket.* The customer will be able to buy a ticket (or several tickets) for a particular game which means there is going to be a relation between all 3 of the mentioned objects. The database relation should be one to many as one *Ticket* is for one game and one customer only and one *Customer* can buy many tickets. After coming to this conclusion, the database design, I came upon is the following:

- **User/Customer**:
  - id
  - email
  - name
  - etc.

- **Ticket**:
  - ticket_id
  - customer_id
  - game_id
  - date_purchased
  - quantity

From the following design it shows that the connection to the game (game_id) and to the customer (customer_id) is present in the *Ticket* table.

5.3. Sub question 3

A login system in this project would ensure that bought tickets can easily be linked to an account and the user can see what tickets they have purchased in the past. However, given the limited time and knowledge I have in this project it would be smarter to just let the user purchase as a guest and send a confirmation email to the corresponding email. That would save me a lot of time and give me opportunity to focus on other parts of the system instead as implementing a nice login/register system with authentication and authorization is not a simple task. However, the requirements for this project require me to showcase that I posses a certain level of knowledge concerning these topics and they are also part of the study program this project is a part of. In order to satisfy these requirements, I will implement a login/register system. This, although will complicate my work, will extend my application nicely with additional features and show that I acquired a sufficient level of knowledge in this area.

6. Conclusion

All in all, there are a million ways to create such ticketing system as I am aiming to do in this project. The important thing is to evaluate all the conditions, deadlines and restrictions and work out a solution around those and as long as the solution is justifiable and efficient, then my goal should be achieved.