# Applied research

Table of contents

[Applied research 1](#_Toc99615046)

[1. Topic 2](#_Toc99615047)

[2. Main question 2](#_Toc99615048)

[3. Sub questions 2](#_Toc99615049)

[4. Research methods 2](#_Toc99615050)

[4.1. Sub question 1 2](#_Toc99615051)

[5. Results 2](#_Toc99615052)

[5.1. Sub question 1 2](#_Toc99615053)

[5.2. Sub question 2 3](#_Toc99615054)

# 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.

# Main question

How do I make a buying tickets system which uses external source of data and uses that data to store locally?

# Sub questions

1. What external source can I use for live data and why external source?
2. Best practices in making database designs for saving tickets?
3. How to use cache in Spring Boot to store testable data even when the external source is down?

# Research methods

## Sub question 1

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

# Results

## Sub question 1

To answer the why part of this questions I need to first explain what the NBA is in real life. The NBA (Nation Basketball Association) is the primary professional basketball league in North America. It consists of 30 teams located across the country. It is no longer the case that basketball players come only from the United States and Canada. The NBA recruit players from around the world. Connecting to NBA data allows global fans to follow their favorite players in areas where the games are not telecast. Also, a lot of trades are going on and a lot of the data is changing which is going to make it very difficult to keep track of every change and put that in the data if it was to be stored locally in a database for example. For these reasons I decided to try and find an API which is going to provide all the live and accurate data I need for this application.

After researching all options of available external API sources containing live data these are the sources I came upon:

* 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 players statistics, teams, games and more 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.

## 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. What is most important is to establish the connections between objects and determine the relationship between the corresponding table. 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. The *Game* table is not present as it will be taken from external API and only its id will be stored.