|  |
| --- |
| **Read with Panda** |

**Michał Karol Pompa 266494, Michaela Golhova 266099**

**Michal Ciebien 266908, Matej Michalek 266827**

**Daniela Koch 266502**

**Supervisor:**

**Jan Otto Munch Pedersen,**

**Jakob Knop Rasmussen**

**VIA University College**

**[Number of characters]**

**ICT Engineering**

**3rd semester**

**December 2018**

**Table of content**

[Abstract iv](#_Toc532777025)

[1 Introduction 1](#_Toc532777026)

[2 Requirements 3](#_Toc532777027)

[2.1 Functional Requirements 3](#_Toc532777028)

[2.2 Non-Functional Requirements 4](#_Toc532777029)

[3 Analysis 5](#_Toc532777030)

[3.1 System analysis 6](#_Toc532777031)

[3.2 Use cases 7](#_Toc532777032)

[3.3 Security 9](#_Toc532777033)

[3.3.1 Thread Model 10](#_Toc532777034)

[3.3.2 Risk assessment model 13](#_Toc532777035)

[3.4 Choose of GUI 15](#_Toc532777036)

[4 Design 15](#_Toc532777037)

[4.1 Architecture 15](#_Toc532777038)

[4.2 Book Service and Bookstore Service 16](#_Toc532777039)

[4.3 Library Service 17](#_Toc532777040)

[4.4 Database Server 18](#_Toc532777041)

[4.4.1 Database design 18](#_Toc532777042)

[4.4.2 Database access 19](#_Toc532777043)

[4.5 GUI 20](#_Toc532777044)

[4.5.1 Basic Principles 21](#_Toc532777045)

[4.5.2 Technologies 21](#_Toc532777046)

[4.6 Security 21](#_Toc532777047)

[4.7 Communication 21](#_Toc532777048)

[5 Implementation 23](#_Toc532777049)

[5.1 Authentication 23](#_Toc532777050)

[5.2 BookService and Bookstore - search 24](#_Toc532777051)

[5.3 Library - Orders 26](#_Toc532777052)

[5.4 DBServer – accessing the database 29](#_Toc532777053)

[5.5 Website 33](#_Toc532777054)

[5.6 Communication 38](#_Toc532777055)

[6 Test 41](#_Toc532777056)

[6.1 Unit test 41](#_Toc532777057)

[6.2 Requirement Test Descriptions 41](#_Toc532777058)

[7 Results and Discussion 42](#_Toc532777059)

[8 Conclusions 43](#_Toc532777060)

[9 Project future 44](#_Toc532777061)

[10 Sources of information 45](#_Toc532777062)

[11 Appendices 46](#_Toc532777063)

# Abstract

*An abstract is a shortened version of the report and should contain all information necessary for the reader to determine:*

1. *What are the aim and objectives of the project*
2. *What are the main technical choices*
3. *What are the results*

*Frequently, readers of a report will only read the abstract, choosing to read at length those reports that are most interesting to them. For this reason, and because abstracts are frequently made available to engineers by various computer abstracting services, this section should be written carefully and succinctly to have the greatest impact in as few words as possible.*

*Although it appears as the first section in a paper, most report writers write the abstract section last.*

Cf. (Dawson 2009, p.195).

# Introduction

Ernest Hemingway once said: “There is no friend as loyal as a book”. That sentence can be read on a few levels. One aspect of it is being shaped by a book. On that point an association between reading and the level of empathy can be distinguished, namely the less adults read, the smaller they define their empathy to be (Jamil Zaki, 2011). That leads to the statement, that reading boosts one’s quality of empathy, as well as the ability to see the world from other people's perspective (Christopher Ingraham, 2016). What is more, “a book is a device to ignite the imagination”. As Alan Bennett stated, reading develops imagination and creativity (Dennis J. Sumara, 2002). But this is not everything a book has to offer! Among the vast range of its advantages reading has been proven to reduce the stress level (Putai Jin, 1992) and improve one’s communication skills by extending one’s vocabulary (Gery Deer, 2016).

Taking under consideration all the enumerated benefits one can gain from books, reading should be a common habit and should be growing into importance and popularity as time passes. Unfortunately the reality is different. In 1982 the percentage of Americans reading books for pleasure was on the rate of almost 57%. That may seem as a small number, but the perspective changes while comparing with the percentages from 2015, which is only 43% of the American population. More than every second American did not read any work of literature during the year 2015. And that does not only count to the proverbial man in the street, but even the educated citizens, that are supposed to be the intelligence in the country, who have put enough effort in gaining knowledge, to achieve a graduate degree, are on the level of only 68% of “intelligence” who have read any work of literature for pleasure (Christopher Ingraham, 2016).

There are multiple reasons to the issue of the decreasing popularity of books. One of them is being surrounded by various types of distraction. As technology progresses, so does the number of ways of spending free time. Nowadays the act of reading a book has to compete for people’s attention with the internet, movies, or computer games. Unfortunately for the books, people tend to choose the latter ones (Christopher Ingraham, 2016; Michael Kozlowski, 2018). Additionally, is the matter of availability of books. As most of the competitors of books can easily be found on the internet, the case is often looking differently for books. Even though a book can be found in an e-version, many people declare to prefer the classic printed versions. Moreover, paper based books increase one’s reading comprehension more than e-books (Hanho Jeong, 2012). That leads to the issue of availability. While to find a book one has to access multiple sides and search for it, or even (God forbid!) go out and look in a library or a store, still not having the certainty of finding it, it is just simpler to use the time on the internet.

Bearing everything that has been mentioned in mind, it can clearly be seen that the humanity is in a need of a way to increase the popularity of books again. One approach to this issue could be combining the traditional books with technology. As it has been stated, people fancy technology. That is why combining books with technology could attract them, if done in an appealing manner. Another approach could be simplifying the process of looking for a book. Namely, connecting libraries with bookstores. In that case, readers would have a bigger range of artworks to choose from and would save time they would normally have spent on looking for books.

More details can be found in the project description appended in Appendix A.

# Requirements

Considering the description of the needed functionalities of the system, a list of requirements is stated below. The requirements are divided into functional and non-functional.

## Functional Requirements

1. The guest should be able to create an account with a name, surname, address, e-mail address and phone number
2. The user should be able to log in to the system
3. The user should be able to log out from the system
4. The customer should be able to see his/her personal information
5. The customer should be able to borrow books from the library
6. The customer should be able to buy books from the bookstore
7. The customer should be able to see books he/she has borrowed and has not given back in his/her account
8. The user should be able to see details of books
9. The customer and guest should be able to search for books (by inputting the name, author, isbn, year or category, filtered by (specific) bookstore/library)
10. The administrator of an institution should be able to search for books in his/her institution
11. The administrator of an institution should be able to manage books in the institution (add, delete)
12. The administrator of an institution should be able to see the list of borrowed books and mark chosen ones as returned
13. The administrator of an institution should be able to confirm book orders
14. The administrator of the system should be able to manage administrators of the bookstores and libraries
15. The system should send reminder emails 3 days before the return date
16. The system should be sending confirmation emails to the users after the administrator of the bookstore has confirmed the users order

## Non-Functional Requirements

1. The system must be written in Java and C#
2. The system must be using a database for persistence
3. The system must be based on a 3-tier architecture
4. Communication in the system must be done with the web service and sockets (with a communication protocol)
5. The system must include a GUI dedicated for the clients(administrator of the system, administrator of a library, administrator of a bookstore, customer)
6. The system must store passwords in a secure way

# Analysis

The first stage of creating a project is analysis. This is where the needs of the customer are analyzed and defined. It is the most important part of the system development, as it creates the base of it. Therefore, next stages as Design, Implementation and Test were derived from deep analysis of this stage.

As the purpose was to make a system, that would improve the accessibility of books by combining functionalities of bookstores and libraries into one system, the main questions considering the problem are as follows:

* What functionalities does a bookstore provide?
* What functionalities does a library provide?
* What should the user be able to do in the system?
* What would make the user interface attractive?
* What information should be stored about users?
* Is membership needed?
* What work methodology should be used while developing the project?

Extended analysis takes into consideration every issue, but sometimes there are some difficulties that are not possible to overcome and that is why creating a list of delimitations is needed. Due to specific requirements and the lack of time the delimitations are summarized as follows:

* The system will connect only one bookstore with one library. However, the architecture and design should support connecting more institutions.
* The membership in the Library won’t be taken into consideration
* The system will not fulfill the requirements with minor importance and one with normal, which are:
  + As an administrator of the system I want to be able to manage administrators of the bookstores and libraries so that so that I can add new institutions with new administrators and delete the shutdown ones
  + As a customer I want to be able to access my account so that I can see books I have borrowed and not given back and my personal information
  + As a customer I want to receive reminder emails 3 days before the return date so that I will be inform that I have to return the book

## System analysis

Basing on requirements, one can proceed with analyzing the main parts of the system needs. The first thing to start is analyzing the domain. The outcome can be seen on the domain model diagram, Figure 1.

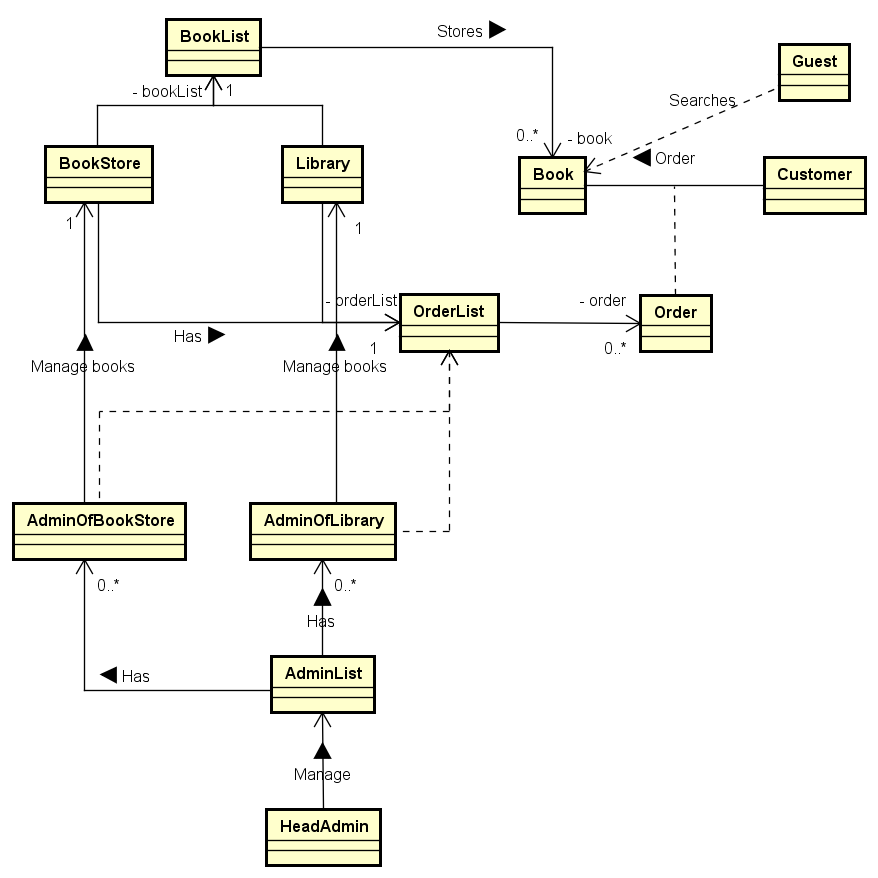
****

Figure 1

It can be seen that the Book is in the center of the domain and everything happens around it. The Library and the BookStore own many books (they are storing them in the BookList). The Customer is ordering books and the orders are going to the OrderList which is managed by Administrators of the Library and the BookStore. Administrators are also managing books in the institutions. The Guest can only search for books. Moreover, the HeadAdministrator (Administrator of the system) is responsible for managing the administrators of institutions and by this, also the institutions.

## Use cases

After stating the requirements, use cases can be created basing on them. The use case diagram below (Figure 2) presents different ways of using the system and also divides the possible actions between actors. The system has following actors: Guest, Customer, Administrator of the Library. Administrator of the Bookstore, Administrator of the System and Time. Each of them can perform different actions in the system. The result of this part of analysis are following use cases:

1. Login into an account
2. Logout from an account
3. Order a book
4. Create an account
5. Search for a book
6. See personal information
7. Return a book
8. Manage books
9. Confirm book orders
10. Manage administrators
11. Send remainder

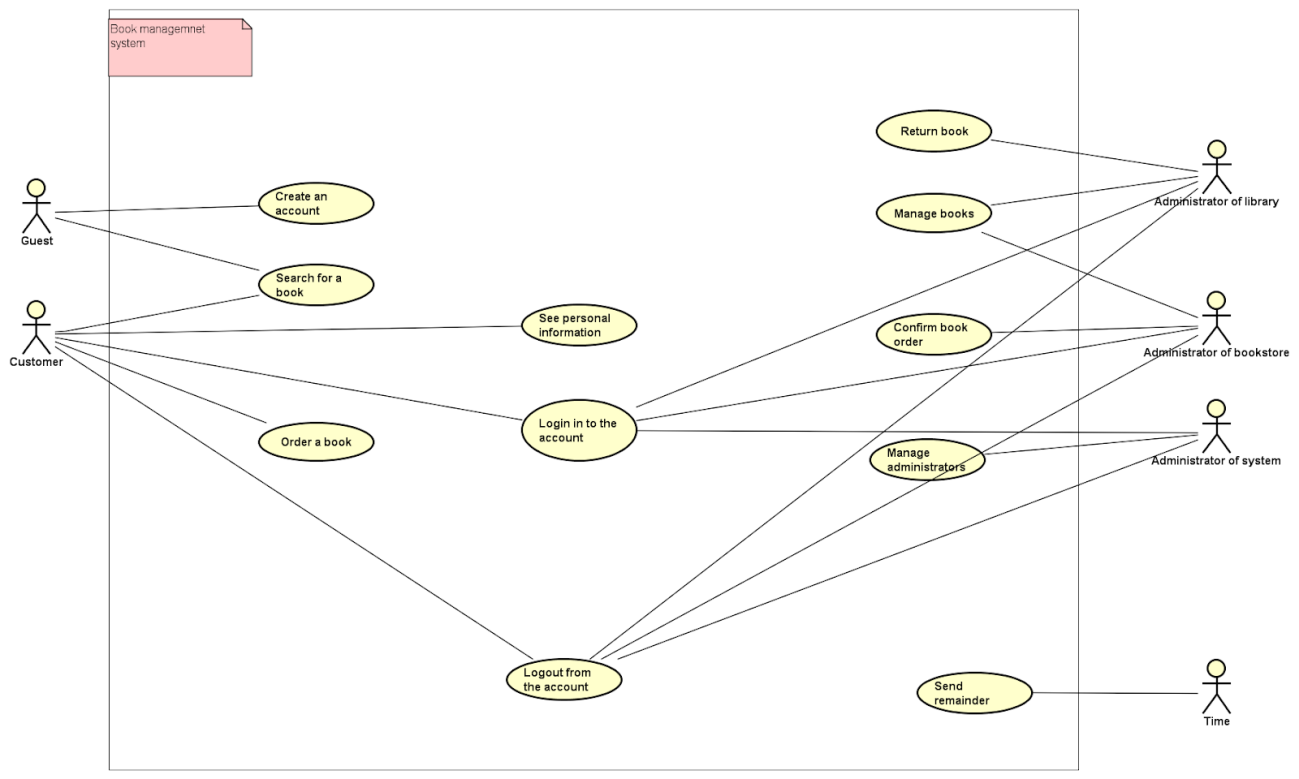
****

Figure 2

One of the core functionalities of the system is ordering a book, which has been analyzed in the Order a book use case. The use case description can be seen in the Figure 3. Other use cases can be found in the Appendix B.

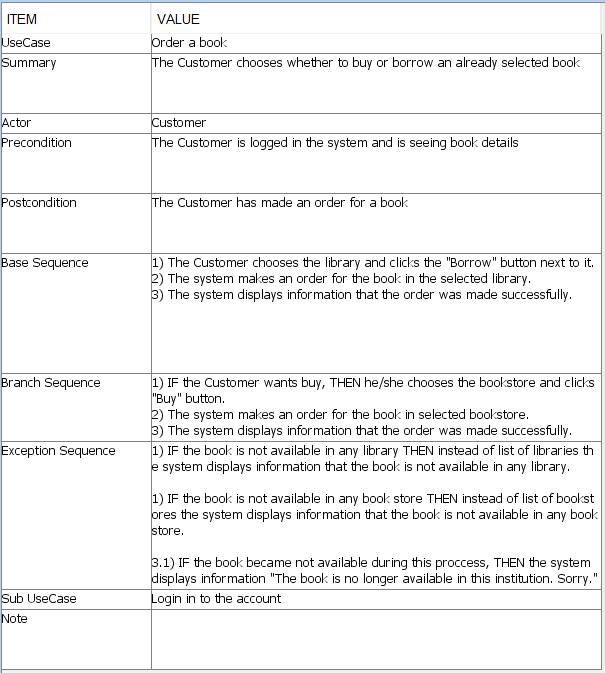


Figure 3

## Security

The system cannot be successful if it is not safe. Users must be sure, that their personal information is not endangered and that only they have the access to their accounts. The topics such as possible threats and vulnerabilities of the system should be analysed. The outcome of that analysis is the Threat Model, which enumerates threats that the system can face. Moreover, the goal and means of the attacker are stated to the every threat. Last part is concerning place and person being able to conduct an attack (expressed in EINOO scale).

### Thread Model

1. **DoS/DDos attack**

The Attacker is sending a large number of search requests to the BookService. When the number of requests reaches the limit, the BookService runs out of resources and is unable to process the requests. The BookService crushes and the system is unable to work as it is a key component of the system.

* 1. **Goal of the attacker**

The goal of the attacker is Denial of Service and objective is Availability of application.

* 1. **Means of the Attacker**

Active - Blocking the application

* 1. **EINOO**
     1. **WHO** - Both Internal and External because the search request can be done by the registered user of a system or a guest.
     2. **WHERE** – Online

1. **Unauthorized access to the Database**

The Attacker is able to acquire the password to the database and log into it through the SSH. When the attacker has access to the DB, he can modify the data or erase all data from database.

* 1. **Goal of the attacker**

Elevation of privileges - the attacker can create an administrator account for himself. - Authorisation

Information Disclosure - the attacker has access to the information that he should not see. - Confidentiality

Tampering - the attacker can modify data without being detected - Integrity.

* 1. **Means of the Attacker**

Active - Modification

* 1. **EINOO**
     1. **WHO** - External - the attacker does not have to be a user of the system.
     2. **WHERE** – Online

1. **Unauthorized access to the admins accounts**

The Attacker is able to acquire the password to the admin account. After that, the attacker is able to modify the data in one specific institution, i.e. confirm the order or return a book.

* 1. **Goal of the attacker**

Elevation of privileges - Authorisation

Information Disclosure - Confidentiality

Tampering - Integrity

* 1. **Means of the Attacker**

Active - Modification

* 1. **EINOO**
     1. **WHO** - Internal, External - if the attacker has access to the admin’s computer in the institution.
     2. **WHERE** – Online, Offline - if the attacker has access to the admin’s computer in the institution.

1. **Unauthorized access to the customer’s accounts**

The Attacker is able to acquire the password to the customer’s account. After that, the attacker is able to make an order as though he was some other customer.

* 1. **Goal of the attacker**

Spoofing Identity - making order on behalf of other customer - Authentication

Tampering - changing the customer personal data - Integrity

* 1. **Means of the Attacker**

Active - Modification

* 1. **EINOO**
     1. **WHO** - Internal.
     2. **WHERE** – Online.

1. **Replay attack on the library (borrowing a book)**

The Attacker intercepts a message with a borrow request sent to the library. Then, the Attacker is able to replay the message and borrow all books with the same isbn as the one in the original message.

* 1. **Goal of the attacker**

Denial of Service - the book is not available for the customers - Availability

Spoofing Identity - making order on behalf of other customer - Authentication

Tampering - modification of the data - Integrity

* 1. **Means of the Attacker**

Active - Modification

* 1. **EINOO**
     1. **WHO** - Internal.
     2. **WHERE** – Online.

1. **Manually shutting down the server**

The Attacker breaks into the room with the server and pull of the plug.

* 1. **Goal of the attacker**

Denial of Service - the system is shut down and unavailable - Availability

* 1. **Means of the Attacker**

Active - Blocking

* 1. **EINOO**
     1. **WHO** - External.
     2. **WHERE** – Offline.

1. **Unauthorized access to the admin’s computer in the institution**

The Attacker breaks into the room with the server and pull of the plug.

* 1. **Goal of the attacker**

Denial of Service - the system is shut down and unavailable - Availability

* 1. **Means of the Attacker**

Active - Blocking

* 1. **EINOO**
     1. **WHO** - External.
     2. **WHERE** – Offline.

### Risk assessment model

After analysing possible threats, one can proceed on creating the Risk Assessment. In this part to each threat are assigned frequency, preventive measures, incident prevention, threat effect and corrective measures. Moreover, to the parts that are needed for risk calculations are marked by Low, Moderate or High (or Good, Moderate, Bad) indicators, that are rating the solution. The risk is calculated basing on the threat frequency with preventive measures (as risk likelihood) and threat effect and corrective measures (as incident consequences).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Threat** | **Threat frequency** | **Preventive Measures/ Vulnerabilities** | **Incident Prevention** | **Threat Effect** | **Corrective Measures** | **Risk** |
| DDoS/DoS attack | Low | Use of an external server with unknown configuration  High | Buy better server hosting | BookService is down and system is unable to work.  High | Buy new server hosting and apologize through email to the customers.  Bad | Potential loss of customers  High |
| Unauthorized access to DB | Low | Posting password to DB on GitHub  High | Storing the password in a secure way; Frequent changes of password; Make frequent backups | Loss or misuse of data or addition of data  High | Restore backups; Change of password  Good | Potential loss or modifications of books or customers data; potential inconsistency between the system and reality  Moderate |
| Unauthorized access to admins accounts | Low | Encrypted passwords in DB  Low | Better authorization; Frequent changes of passwords; Make frequent backups  Digital signature | Loss or misuse of data or addition of data  High | Restore backups; Change of passwords  Good | Potential loss or modifications of books or customers data; potential inconsistency between the system and reality  Low |
| Unauthorized access to customer's account | Moderate | Encrypted passwords in DB  Low | Digital signature and encrypting passwords | Undesired actions taken on behalf of customer  High | Change of password  Moderate | Potential loss of money  Moderate |
| Replay attack on the library (borrowing a book) | Moderate | SSL configured on the server  Low | Use of session keys and nonces or ssl (sequence numbers and nonces), Frequent backups | Books data has changed  Hard to detect  Moderate | Restore backup  Good | Potential unavailability of a book  Moderate |
| Manually shutting down the server | Low | Server owned by third party  Moderate | Buy own server | System is shut down and unable to work  High | Contact server owners  Moderate | System is not available for some time  High |
| Unauthorized access to the admin’s computer in the institution | Low | Computer located in restricted area  Moderate | Decrease session key lifetime  Hire guards to guard the computer  Frequent backups | Undesired actions taken on behalf of administrator  High | Change the password  Restore backup  Moderate | Potential loss of data  Time spend to restore  Moderate |

## Choose of GUI

For the GUI of the system, the decision to use the website was made. This followed the discussion about different ways of making the system easily interactive with users. Website, compared to some other GUI forms is entirely cross-platform and due to its design flexibility, it can provide an excellent User Experience.

# Design

Having done analysis, one can proceed with designing the system. This is the part where specific classes are distinguished and relations between them are being specified. In this part of the working process, the analysis of the problem statements and requirements are converted into an overview of the whole system. This is the last step before formulating the code.

## Architecture

The base of a successful system is a well-designed architecture, as it is a skeleton for all features of the system. The 3-Tier architecture pattern has been chosen, as the system is a distributed system. This pattern allows dividing the responsibilities in the system to different components. Another advantage of choosing this pattern is easy scalability. There are 5 components of the system. Starting from the 1st tier, which is responsible for interacting with the user, the website has been chosen. On this tier, to extend the system, the mobile application can easily be created, basing on the 2nd tier. The topic of this choice is developed in Website section of design. The 2nd tier focuses on the business logic of the system. There are 3 components on this tier. BookStore and Library are services responsible for actions taken by the Administrators of those institutions. The BookService is a service that takes care of general functionalities (like searching for books) and also Customers registrations or making orders for books. Moreover, the system can be extended by adding more Library and BookStores components, representing real institutions. Last, 3rd tier, is responsible for storing in and accessing data from the database. The component that is responsible for this is DBServer. The architecture of the system is presented on the diagram below.(Figure 4)

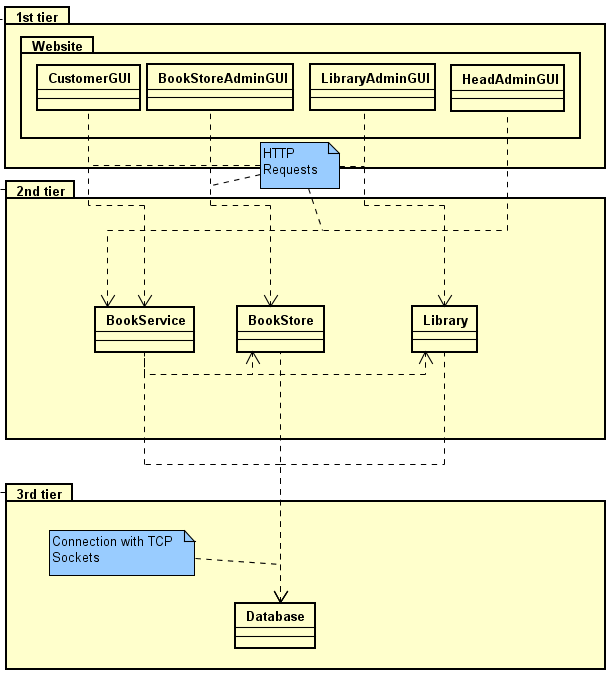


Figure 4

## Book Service and Bookstore Service

The book service consists of three main parts: Requests controllers (API), Model and Controller. The model is an independent part which contains classes describing the business domain regarding functionalities of customer (in the case of Book service) and orders (in Bookstore). The Controller represents a bridge between requests and other components in the system. The request package in Book service consists of requests such as Make order, Create customer, Book details, Login in and Search. Book store is handling Order request and Search which is same as in the book service.

## Library Service

As other services, the Library service also consists of three main parts such as Requests controllers(API), Models and LibraryControllers.

Requests coming to the Library service are handled by 4 controllers in following way:

1. **SearchController** - handles two GET requests (Search,AdvancedSearch) by passing optional parameters from the request to the LibraryController class. Moreover, it handles another GET request (BookDetails). In this case it receives the “isbn” parameter from the URI and passes it to the LibraryController class.
2. **BooksController** - is responsible for two methods regarding books, such as add (POST) and delete (DELETE) book.
3. **OrdersController** - handles GET request (GetOrders) and DELETE request (ReturnBook)
4. **LogOutController** - handles only one DELETE request LogOut that clears local cache and also sends DELETE request to the BookService to clear it’s own cache, too.

The Models namespace contains basic Book class and Category enum that holds all possible book categories in the system. Another classes in Library model namespace are not needed for the reason that Library is a service that only consumes requests from Library administrators and forwards them through Controllers.Connections namespace to the DBServer component that is operating in the 3rd tier.

The Controllers namespace consists of LibraryController singleton class, SessionKeyManager class and the namespace Connections.

LibraryController singleton is a bridge between Requests (API) and Connections namespace responsible for Socket connection to the DBServer. In the same namespace is located another important class called SessionKeyManager. As the Library service can be contacted only by library administrators, all of the requests are sent through authorization process that is handled in SessionKeyManager. If received session key from request is not present in the local cache of Library, the HTTP request is sent from SessionKeyManager to the BookService in order to get the expiration date from original cache storage. After that is needed to check the expirationDate and perform the request if the session key has been evaluated as valid.

The Connections namespace contains one main interface IDatabaseProxy that provides all possible functions that can be called on DBServer through Socket connection. The detailed analysis of Socket connection between DBServer and Services can be found in the section 4.7 - Communication.

What is more, in the Library service can be found also Resources namespace. In this namespace is located ConfigurationLoader singleton class that is responsible for loading data from configuration text file. This data is needed to either open a socket connection to DBServer (host and port) or make HTTP request to the BookService(url). Moreover, the specific id for the Library service is stored in this configuration file. Whole Library class diagram can be found in Appendix B.

## Database Server

### Database design

From the functional requirements the business entities have been derived. Those entities are: Book, Library, BookStore and Customer. Those are strong entities of the system. In purpose to model the relations between them, week entities are needed. For example, when the Book is possessed by the Library, this information is represented by LibraryStorage. Other relation entities are: BookStoreStorage, BookStoreOrder and LibraryOrder. BookStoreAdministrator and LibraryAdministrator are dependent on BookStore and Library respectively. Those entities are responsible for Institution Administrator. The relations between entities can be seen in the Figure 5.

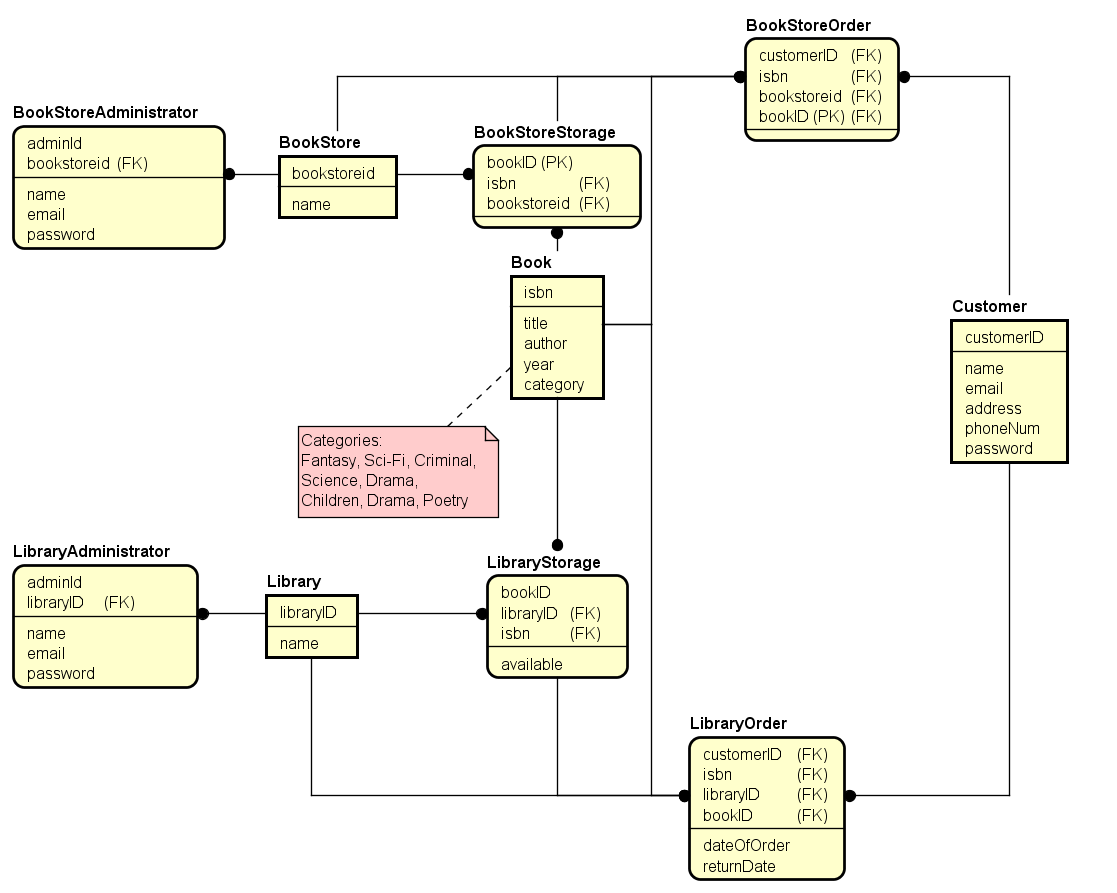
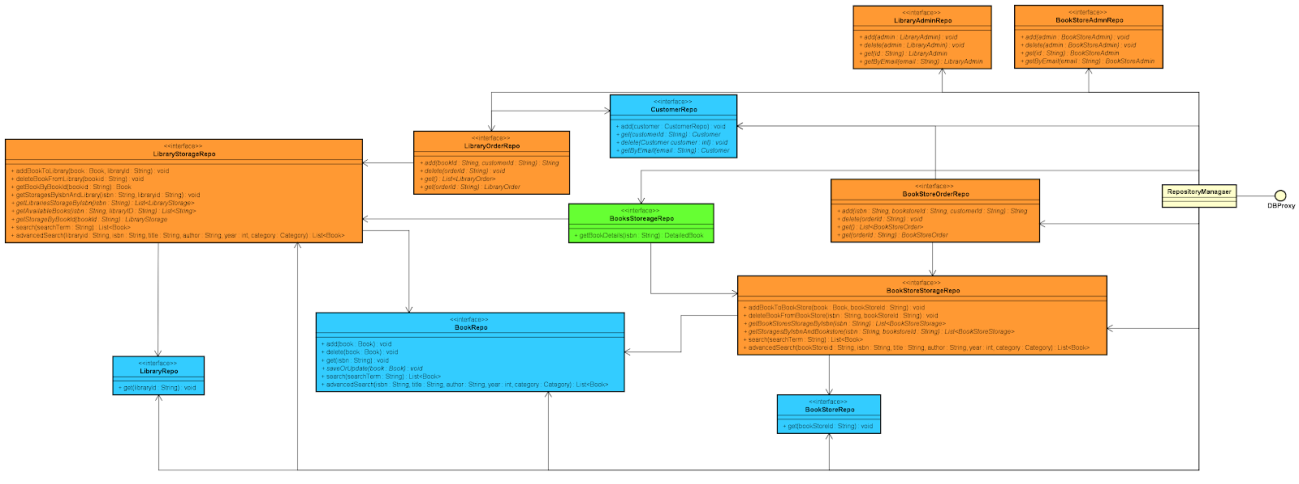


Figure 5

### Database access

Database entities are accessed by Repositories where each entity has its own Repository. The diagram on Figure 6 shows connections and dependencies between repositories. The base of the structure are repositories responsible for CRUD on basic entities like Book, Customer, Library and BookStore. Those repositories are marked with blue color. Higher in hierarchy are repositories for managing storages in Library and BookStore and Orders, marked with orange. Beside CRUD operations, they are also exposing methods for searching for books in particular institutions, getting storages by isbn and institution’s ID. BooksStorageRepo, marked with green color, has the highest place in hierarchy. It is responsible for retrieving information from other repositories and combining it. It is use to get overall view of the book in the system (i.e BookDetails). Repository manager is a facade to the repositories for the Controller.

Each of the repositories has its own interface. That gives a possibility to change the implementation of the specific repository without interfering with other repositories.

[Figure 6 - full size image can be found in Appendix B]

Figure 6

## GUI

The goal of the First Tier architecture (GUI) in the system is to provide means for the user to be able to interact with the system’s underlying business logic hidden in the Second Tiers’ servers. The GUI is to be a place where no business logic will be implemented, but where the data will begin it’s flow through the system’s architecture. In other words, the GUI is to query the business logic servers with data, and display the data to the user.

As stated in Analysis, the choice has been made to use website for above purposes, but similar design could refer also for other GUI choices. The term website and GUI will be used interchangeably further below.

### Basic Principles

The GUI should “not know” how does the internal business logic works. All the internal server operations should be entirely disassociated with data and logic contained in the GUI structure. This will not only simplify the system’s architecture - but also provide more security to the system by giving the attackers no easy means to manipulate any data. One may find it helpful to think of this first-second-tier relationship as if where the second tier behaves as a “black box” for the first tier. The latter one can send data or ask for some data contained in the “black box”, but does not know how this data is generated or used inside it.

As with most websites, the decision has been made to use HTTP protocol requests to fetch data from/to 2nd Tier servers.

### Technologies

Frontend developers environment gives a variety of tools for developers to use to make websites look and work well. The decision was made to use React Javascript framework as a primary tool for making the website modern, up to date with today’s market as well as easy to maintain in the future.

## Security

## Communication

As the system is distributed and consists of many components, to connect their functionalities, there must be a communication amongst them. There are two types of communications technologies used in the system. Between 1st and 2nd tier, the website is making HTTP requests which are handled by REST requests controllers in the 2nd tier components.

The communication with the Database Server is accomplished by TCP sockets connection. Services application and Database Server exchange messages in form of JSON strings containing Request or Response.

Request message has two attributes, operation and arguments. Operation describes the type of the request and arguments supplies necessary information for processing the request by database.

Response message has two attributes, status and content. Status describes if the requested operation has ended with success or an error occurred. The content of the response contains the result of the request. If the request operation is not recognized by the Database Server, it returns a response with Error status and content “Wrong operation”. Examples of the messages and their JSONs can be found in Appendix X.

However, the base of the communication is communication protocol. In the case of this system, the protocol is very simple and works in a request-response manner. The protocol has been represented on the diagram below in the example of searching for a book request. (Figure 7)

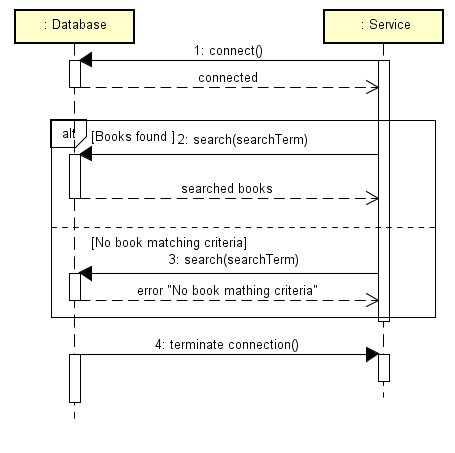


Figure 7

# Implementation

Basing on analysis and design a code can be formulated. The following part shows the most interesting parts of the implementation. However, the whole source code can be found in Appendix X.

## Authentication

One of the security mechanisms implemented in our system is session key validation. This approach requires sending a session key, acquired during login, with requests that should be accessible only by specific type of users of the system.

For generating and controlling session keys is responsible the SessionKeyManager. The generateSK() method generates a new session key using UUID and together with its expiration date it puts it to the HashMap. (Figure 8)



Figure 8

The method checkSessionKey() takes a session key in parameter and checks if the date of the session key is before the expiration day. If the session key’s expiration time has expired the method will throw a SessionKeyIsNotValidExeption and user’s activity is stopped. (Figure 9)



Figure 9

Before each request the checkSessionKey() method is called to validate the session key. If the session key is not out of its expiration date, the request is executed. Otherwise, the exception is thrown and it is caught by advice class (advice classes are described in detail in section 5.2 - BookService and Bookstore - search) and the Unauthorized status is returned. (Figure 10)

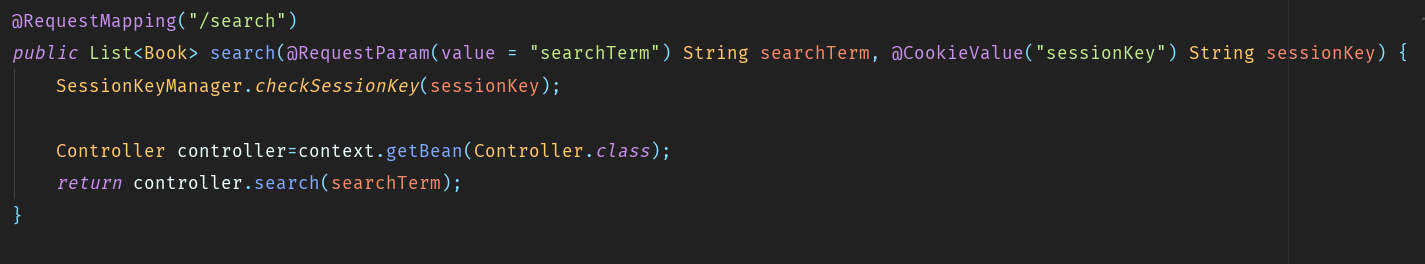


Figure 10

## BookService and Bookstore - search

Those components have been written in Java and are REST web services which are handling requests from the website. The technology that is used to implement REST web service in Java is Spring MVC Framework. This solution has been chosen over the Java EE because the Spring Framework comes in a Spring Boot Framework which is providing also other functionalities like Dependency Injection that has been used in our project. We also found it easier to use than Java EE. Another advantage is that the Spring Framework is currently the most popular web services framework used in Java and learning it will benefit us in the future.

The implementation of the BookService and BookStore is in many places similar, as they are using the same technology and as there is not much business logic in the system, processing of the request looks similar. The request is received by RestController and then passed to the Controller. Controller sends the request to the DBServer through the DatabaseConnection and returns the response depending on the DBServer response.

The search Request in the BookStore will be used as an example. (Figure 11)

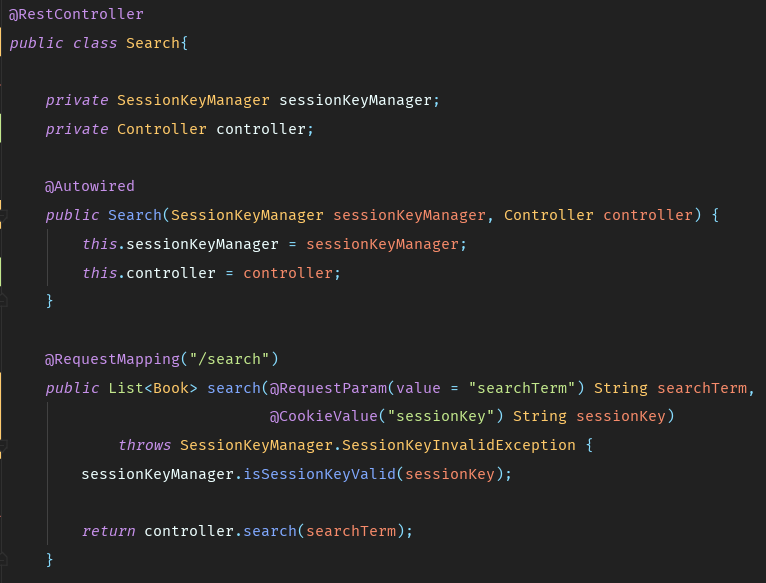


Figure 11

The class with request, Search, is marked with annotation @RestController, which allows the Spring Framework to map it with the requests. The constructor is marked with @Autowired annotation. That means that the parameters should be injected by Dependency Injection and Spring will take care of them. The method search(), that is a request handler, is marked with @RequestMapping annotation and the parameter for annotation is a route to the request. The parameters of the method are marked with two more annotation. The @RequestParam is specifying that this parameter should be in the URL with name searchTerm. The session key is passed in the cookies, so it is marked with @Cookie annotation. First action which is done is to check if the session key is valid. If not, the exception will be thrown and caught by the exception advice, and the 401 (Unauthorized) status code will be returned. The implementation of the advice class can be seen in the Figure 12. After ensuring that the session is valid, the searchTerm is passed to the Controller which is processing the request and prepares the response.

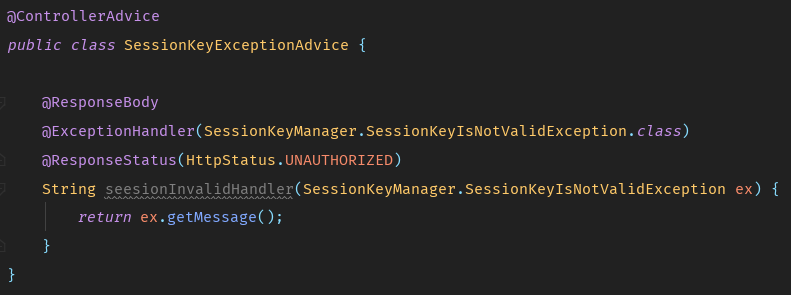


Figure 12

## Library - Orders

The ASP.NET Web API is used as a framework to handle HTTP request to the Library service. Below, can be found the implementation of the flow through the LibraryService on getting the GetOrders request.

As it is stated in design section 4.3 - Library Service, there are 4 request controllers in the Library service. The OrdersController is the one responsible for handling requests regarding orders.

As the Library service is supposed to be reachable only by the corresponding admin, all requests on this service need to be authorized. In the Figure13, can be seen that the GET request is handled by GetOrders() method in the way that firstly the session key from request is checked. This is handled in the CheckSessionKey private method that gets cookie from request and the SessionKeyManager is used in static way in order to validate the session key. Only if the session key is validated then the request is processed and passed further to the \_libraryController and then to the DatabaseProxy class where the corresponding request is send to the DBServer through socket connection. Otherwise, if the session key is not validated, an Unauthorized response with a status code 401 is sent back to the consumer of the Library service.

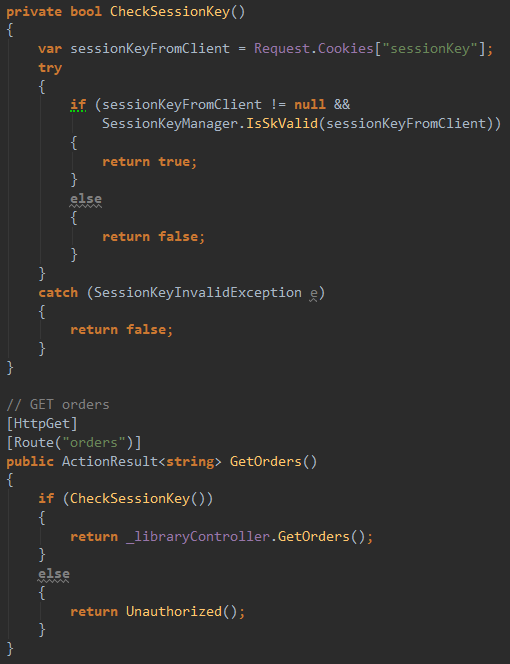


Figure 13

In the Figure 14, can be observed how the authorization process is implemented in the Library service, more precisely in SessionKeyManager class.



Figure 14

From each of the request controllers, the static method IsSKValid(sessionKey) is called before processing the request. In this method, firstly is checked whether the desired session key is present in the library local cache - in this case it is a Dictionary<string,DateTime> field that stores session keys of all library admin accounts that are using this library currently and the session key’s expiration date as a value in this Dictionary. If there is not any data to the corresponding session key in the library local cache, then in the catch block is handled this case by calling CheckInBookService(sessionKey) private static method. In this method is called MakeRequest(url,cookie) method from where the HTTP request is sent to the BookService. After that the response containing the expiration date is parsed and returned. Finally, the expiration date is checked in IsSKValid method and boolean expression is returned back to the request controller.

## DBServer – accessing the database

DBServer is written in Java. Hibernate ORM framework has been used for database manipulating and quarrying.

The class responsible for starting Hibernate connection with the database is HibernateAdapter. This class is also providing some CRUD operations in database for Repositories, because adding, updating and deleting is implemented in the same way for every kind of object. Method addObject() is shown in the Figure 15. In order to add an object to DB using Hibernate a session must be open. Next step is to open transaction and perform a save. Last step is to commit transaction.

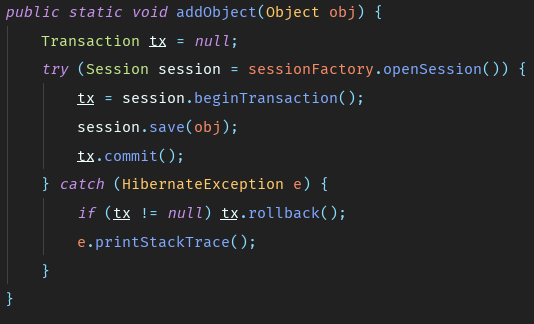


Figure 15

For more advanced queries, custom query must be created. One of such queries is to get a BookStoreStorage by Book isbn and BookStore ID. The process is presented in Figure 16. First, the session is opened and the transaction is begun. Then, the query is created with Hibernate Query Language. To use additional parameters in query, the setParameter method is used, which associates parameter in query with variable. tolist() method executes the query and the result list is retrieved. Then result is returned.

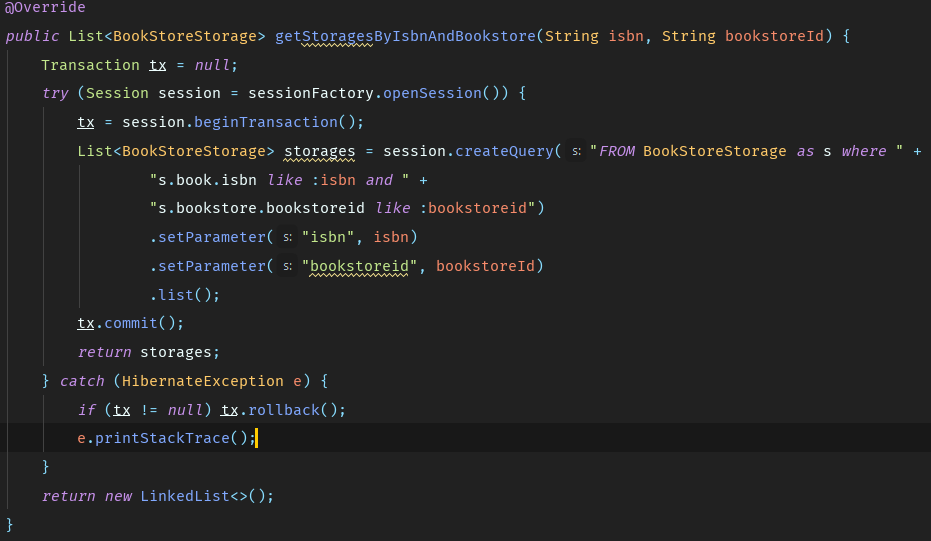


Figure 16

In order to allow Hibernate map desired class to the corresponding table, JPA annotations has been used. @Entity annotation marks the class as the Entity to be managed by Hibernate. @Table specifies the table name. @Column annotation maps the field to the column. The example in for of Book class is shown on the Figure 17.

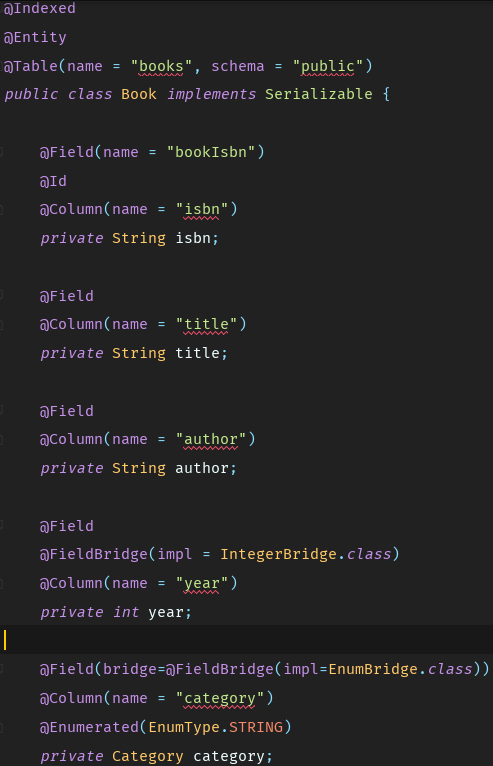


Figure 17

Another functionality of the DBServer is to provide searching for the books. It is accomplished by Hibernate Search library using Apache Lucene index. By adding annotations @Indexed to the class it is marked to be indexed by Lucene. That allows writing a special kind of search query that performs full text searching on specified fields, marked with @Field annotation. The example of such query is presented in Figures 18, 19 and 20.

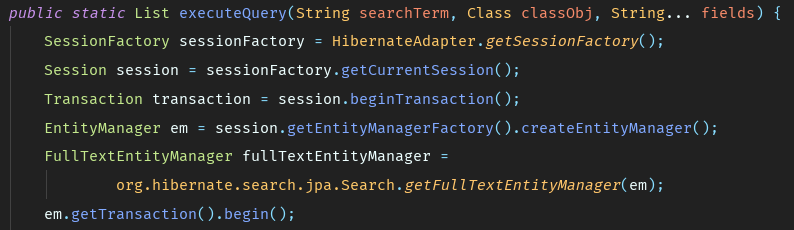


Figure 18

To perform a search, first the session must be opened and the FullTextEnittyManager object must be created. Next step is to open a transaction.

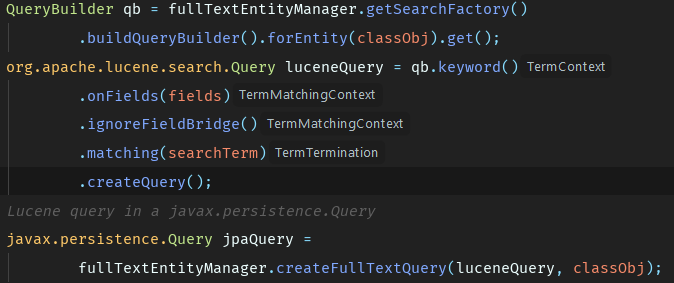


Figure 19

Then, using QueryBuilder the luceneQuery is created with proper configuration. The Class to be searched must specified, fields and searchTerm must be provided. After that, query is wrapped into JPA query. Last step is to execute a query, commit the transaction and return the result.

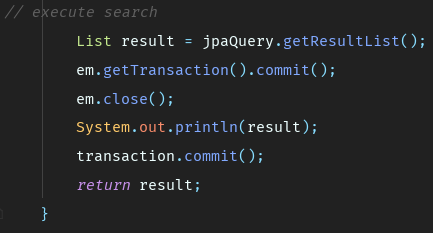


Figure 20

## Website

In this part, we’ll focus on how the website was implemented, how the queries were handled and how the React components were structured.

The basic structure of the React components looks like in the diagram shown below. (Figure 21)

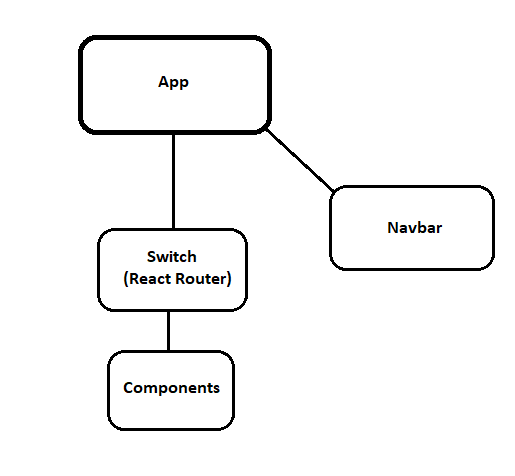


Figure 21

The App component encapsulates both React Router components as well as the Navbar separately, as the Navbar is always displayed on the page. React Router serves the role of changing the ‘root’ of the page regarding the specified URL. This also provides a static-page functionality as the page will not reload the whole window, but just the React Router component to display various data.

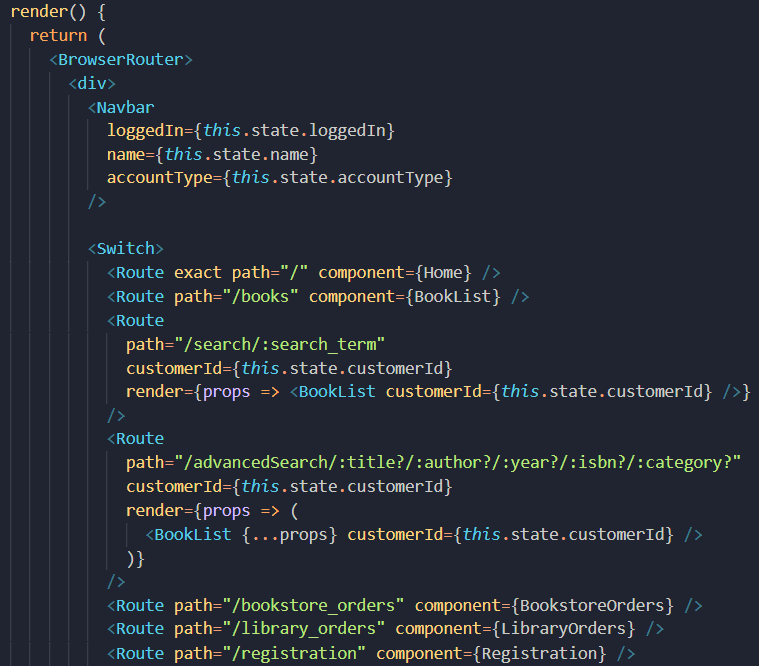


Figure 22

The above code snippet (Figure 22) shows how indeed Routes are defined inside the Switch component, leaving the Navbar outside.

The App component also stores the vital data for the website like:

* cookies
* data of the user

These data are then passed to the different component on call and used in queries to the APIs (2nd Tier servers), which will be covered further.



Figure 23

The above code snippet (Figure 23) shows how the state of the data is changed on the call from *handleLogIn* function.

This architecture simplifies the structure of the data saved on the website, as all of the data are stored in one place, namely the *App* component.

Different queries in the website are handled in different places, before loading the page, after loading the page, but also are triggered by the user by for example clicking a button.

Consuming the API was done using the library called *axios*, which simplified and improved the quarrying process.

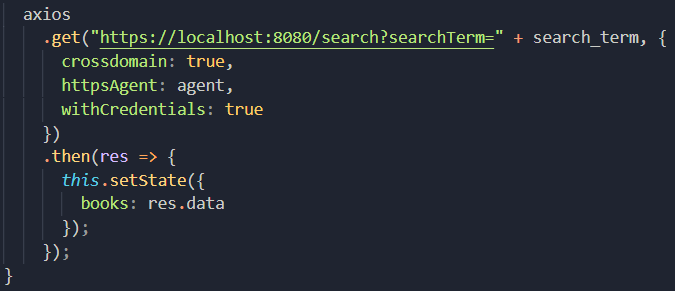


Figure 24

The above snippet (Figure 24) shows how does axios do the query, and how it can update the state of the component. Axios sends a https request to the URI with CORS header enabled (crossdomain: true), and also using cookies stored in the App component, by passing withCredentials option. Then if the response is received - it can work with the acquired data and in this case change the state of the component.

After the state is changed, the corresponding functions are called, so that different data is displayed to the user. This is shown in the code snippet below (Figure 25), where in this particular example, on each change of books state, the mapping function is called and the layout is changed.



Figure 25

## Communication

As stated in Communication section, the communication with DBServer is accomplished by TCP sockets. This requires classes which are receiving and sending messages. On the DBServer side, the Server side is responsible for listening for client connection and passing the request to the Controller, which is processing the request. When the Response is ready, it is sending it back to the client. When the connection is established, the client socket is passed to the HandleClient class, which in a new thread receives a message and after the request is processed, sending back the response. The implementation is shown in the Figure 26.

****

Figure 26

On the other side of the connection are BookService, BookStore or Library and the class responsible for connection is DatabaseConnection. In all those components it is implemented similarly. It is implementing DatabaseProxy interface, where are methods that are corresponds with available requests to the DBServer.

The method that is responsible is sendMessage() method. It is opening the connection with the DBServer, sends the message and receives the response. If the connection could not be established, it is throwing ServerOfflineException which is cough on the higher parts of the system. The implementation is shown in the Figure 27.

****

Figure 27

As the messages are sent in format of JSON Strings, they must be deserialized and translated to the objects. The getResponseStatus and getContent methods are responsible for deserializing the two parts of the response. The more interesting one is a getContent (Figure 28). As the content can be in many forms and represents many formats the method is generic and is able to deserialize any object. It is accomplished by using Gson library for serializing and deserializing JSON. By Type object and passing it to the fromJson method of gson object, the JSON is properly deserialized into required object. The type T of the method is specified during the method call. In the case of library, as it is written in C#, the NewtonSoft Json library has been used.



Figure 28

# Test

Having implemented the system, one has to conduct tests. Tests are needed not only to check whether the system is working and has been implemented correctly, but also to ensure that all requirements have been fulfilled and the system provides the end user with all the desired functionalities. Bearing that in mind, two kinds of tests have been executed: whitebox tests, in this case unit tests, and blackbox test, in this case tests following Requirement Test Descriptions. The results of the tests are shown further in this paragraph.

## Unit test

## Requirement Test Descriptions

For this type of testing requirement test descriptions have been formulated, i.e. scenarios of what output the system should give after entering specific input. The descriptions and results are shown in the table below.

TABLE

# Results and Discussion

# Conclusions

# Project future

The major part of tasks was completed and it can be said that the system is fully working. For the future there are some features left that can be implemented like:

* sending remainders
* creation of new administrators
* customer’s profile
* pagination of search request
* buy domain

# Sources of information

Jamil Zaki, 2011. What, Me Care? Young Are Less Empathetic. Scientific American. [e-journal] https://www.scientificamerican.com/article/what-me-care/

Christopher Ingraham, 2016. The long, steady decline of literary reading. The Washington Post. [e-journal] https://www.washingtonpost.com/news/wonk/wp/2016/09/07/the-long-steady-decline-of-literary-reading/?noredirect=on&utm\_term=.7935d603d3c3

Dennis J. Sumara, 2002. Why Reading Literature in School Still Matters: Imagination, Interpretation, Insight. 1st ed. New York: Routledge

Putai Jin, 1992. Efficacy of Tai Chi, brisk walking, meditation, and reading in reducing mental and emotional stress. Journal of Psychosomatic Research. [e-journal] https://linkinghub.elsevier.com/retrieve/pii/002239999290072A

Gery Deer, 2016. Improve your vocabulary through reading. GLD Enterprises Communications, Ltd. [online] http://geryldeer.com/improve-your-vocabulary-through-reading/

Michael Kozlowski, 2018. Reading books is on the decline. GoodEReader. [online] https://goodereader.com/blog/bookselling/reading-books-is-on-the-decline

Hanho Jeong, 2012. A comparison of the influence of electronic books and paper books on reading comprehension, eye fatigue, and perception. emerald insight. [online] https://www.emeraldinsight.com/doi/pdfplus/10.1108/02640471211241663

# Appendices

**?Appendix A** Project Description – ProjectDescription.pdf

**?Appendix B** Diagrams as Astah projects –

**?Appendix C** Data glossary –

**?Appendix D** User guide –

**?Appendix E** Code –

**?Appendix F** Client application –

**?Appendix G** Server application –

**?Appendix H** SQL code of the database –

**?Appendix I** Unit test coverage report –