

Designing simple web Application in Spring Boot

Before we start ...



Kinga

- Participated in Women In Tech Camp in 2017 , where decided to apply for internships at Ericsson,
- Now works as Senior Software Developer responsible of automation in one of Ericsson's QAs,
- Main technology used at work: Java.

Łukasz

- Has been working at Ericsson for 9 years,
- Senior Software Developer responsible of automation in one of Ericsson's QAs,
- Main technology used at work: Java.



Content of presentation



1

Functional specification

- General Info
- Roles
- Items/Objects in System
- Use Cases
- Scenarios
- Limitations
- Forms/Views

2

Implementation - Backend

- Technologies used
- Spring Boot – what is it?
- Database schema
- Database – Hibernate & MySQL
- MVC
- REST methods
- Spring Security

3

View - Frontend

- Technologies used
- Thymeleaf
- Bootstrap
- JQuery

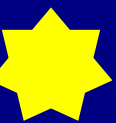
4

Questions





1 Functional specification



General information



Book Rental is a web application written using Spring Boot, Hibernate & Thymeleaf.

Book Rental is designed for librarians, and it is supposed to be used in small libraries. It helps with managing the process of renting books to readers.

Application is designed to be simple and not complicated. Many features are simplified.



User Roles



Librarian



Reader



Librarian



Librarian

- Registering the new users,
- Adding new books,
- Editing data about books / readers,
- Renting books to users,
- Checking how many books are rented by particular user,
- Handling returned books in system.



Reader



Reader

- Searches available books in Library,
- Checks which books are currently rented by him.





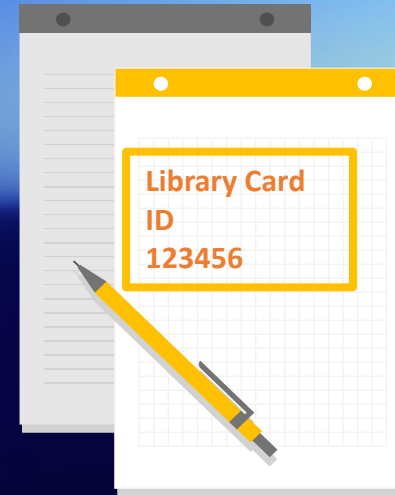
Items / Objects in system



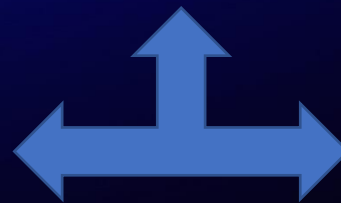
Real-world objects



Book



Library card
ID



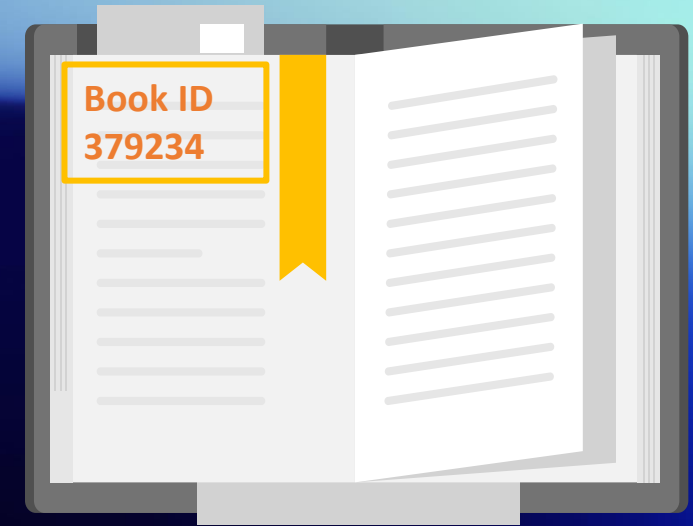
RENTED



Book



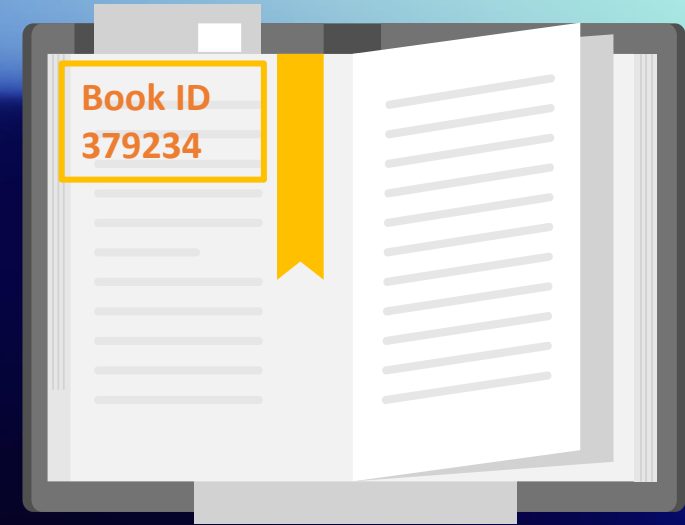
- Usually inside any library we have many books on the shelves,
- Each book **has unique book ID**,
- Each book has **title** and **author**.



Book

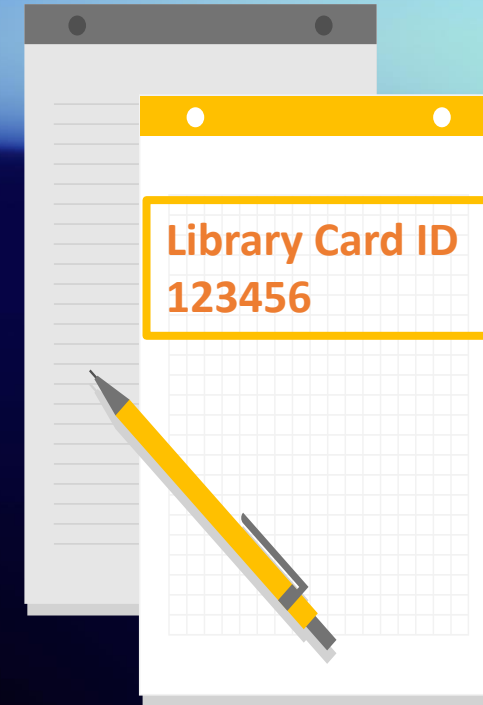


- Other important properties are **ISBN** and **year of publication**,
- Book can have a few **categories** (for example: novel, romance, horror, fantasy, science-fiction...),
- Finding a particular book in big store might be difficult. To help Librarians with this task we added field **shelf**. This field represents physical location of book in library building.



Library card

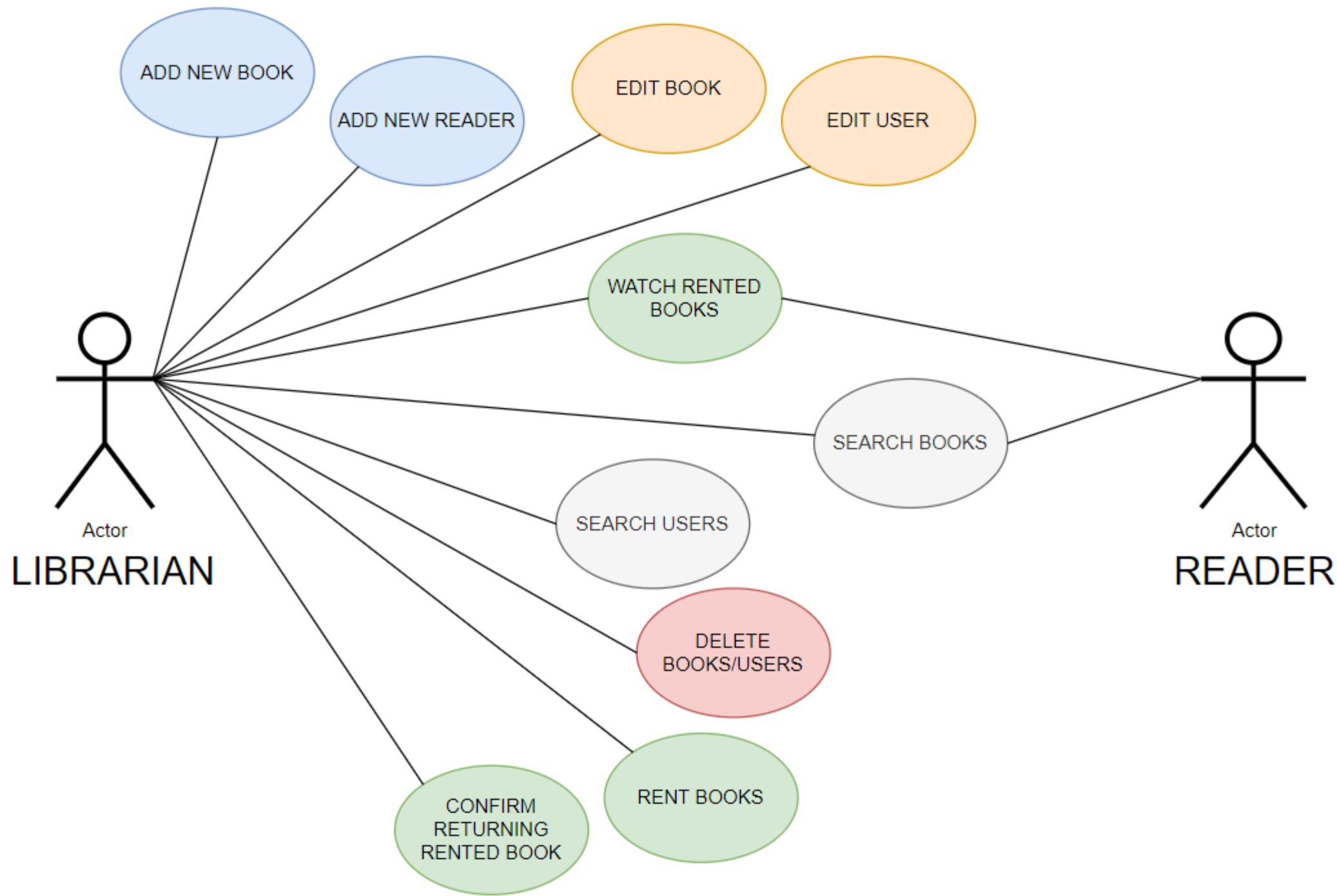
- Each user registered in library gets the printed document: "Library Card",
- Each Library CARD has **unique ID**,
- The same ID is used in our system. Book Rental application identifies users by Library CARD ID.





Use cases







Scenarios





Yellow — activities in "real world", for example
putting book on the shelf

White — virtual activities in Book Rental application



Scenario 1: Library bought a new book



<<INSIDE LIBRARY>>



Librarian

- Librarian puts the new book on the shelf.
- Librarian starts the Book Rental application, logs in and clicks "Add New Book" button.
- Librarian fills in the form and confirms operation. Book is added to the system.



Scenario 2: New reader came to the library ≡

<<INSIDE LIBRARY>>



Reader

- New Reader arrived. The person wants to join library.
- Librarian starts the Book Rental application, logs in and fills in the form "Add New User". Unique ID is generated.
- Library CARD ID is printed by Librarian. Document is given to the user.



Librarian



Scenario 3: Reader wants to rent a book



<<INSIDE LIBRARY>>



Reader

- Reader comes in into the library building. Guest informs librarian about the book he wishes to rent.
- Librarian searches book in System. The book is available.
- Librarian takes the book from the shelf.
- Librarian registers renting book in the system using ID of Library CARD showed by user.



Librarian



Scenario 4: Reader comes to library and returns book ≡

<<INSIDE LIBRARY>>



Reader

- Reader comes in into the library building and returns the book.
- Librarian searches book in System and registers that the book is back.
- Librarian puts the book on the shelf.
- From now on other readers might borrow this book.



Librarian



Scenario 5: Pile of returned books, readers already left ≡

<<INSIDE LIBRARY>>



- In the morning many readers came to the library building. Librarian was very busy...
- In meanwhile a few readers put their books on the table and left the library.
- Now, we have a pile of returned books,
- One by one book, Librarian opens the books and finds the rubber stamp with book id,
- For each book librarian finds record in the system under "Rented Books" and clicks "Return Book" button



Librarian



Scenario 6: Reader logs in to Rental App from home,



<<ONLINE>>



Reader

- Reader needs certain book for his university work.
- Reader logs in to the system,
- Reader goes to "Search Book" and types the query.
- System displays the information about the desired book.
- Book is available in the library and not rented, so user will visit the building in person,





Limitations



Limitations

- Application can be used in libraries, so it is designed to be run on Laptops/Computers,
- Application **is NOT supposed** to be used on cell phones.



Limitations



- Application is designed to be simple and not complicated. Many features are simplified,
- Application was designed for training purpose.
- What is simplified?
 - no registration and same password for each user with the same role
 - no limit of books and no due date for rentals





Forms / Views





Searching

LOGIN

MENU

Search User

User Details

EDIT

REMOVE

Rented Books

RETURN BOOK

Search Book

Book Details

RENT BOOK

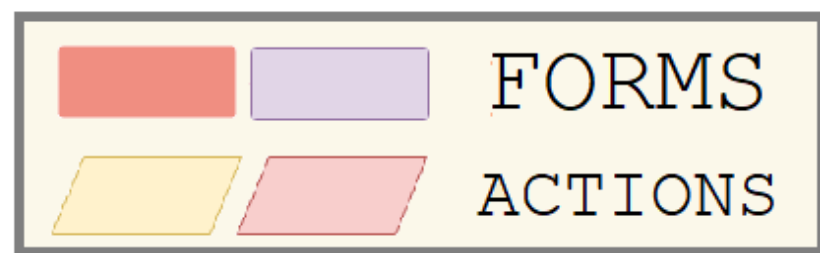
EDIT

REMOVE

Add New User

Add New Book

Adding



Menu for Librarian vs Menu for Reader

Book Rental Management System

Main Menu

Search Books

Search Users

Rented Books

New User

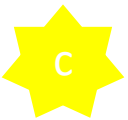
New Book

Book Rental Management System

Main Menu

Search Books

Rented Books





Adding New Book

Book title:

Author:

ISBN:

Description:

Shelf (in library):

Year of publication:

Categories:

☐ SCHOOL_BOOK

☐ DRAMA

☐ NOVEL

☐ THRILLER

☐ CRIME_STORY

☐ FANTASY

☐ ROMANCE

☐ FAIRYTALE

☐ BIOGRAPHY

☐ ADVENTURE

☐ POETRY

☐ MANAGEMENT_AND_ECONOMY

☐ SHORT_STORY

☐ HISTORY

Create Book

Book Rental Management System



[Main Menu](#) [Search Books](#) [Search Users](#) [Rented Books](#) [New Book](#) [New User](#) [Logout](#)

Search Users

Username (name/surname):

[Search](#)

Username:	Library CARD ID:	User Role:	Operations:		
JanKowalski	100000	READER	Rented Books	View User Info	Edit User Info
Anna Kowalska	100001	READER	Rented Books	View User Info	Edit User Info
Maria Komornicka	100003	READER	Rented Books	View User Info	Edit User Info

[Add Library User](#)

Book Rental Management System



[Main Menu](#) [Search Books](#) [Search Users](#) [Rented Books](#) [New Book](#) [New User](#) [Logout](#)

Search Books

Book title:

Category:

Author:

[Search](#)

	Title:	Author:	Book Id:	Shelf:		
[B]	W Pustyni i W Puszczy	Henryk Sienkiewicz	300005	ADVENTURE	Edit	View Info
[F]	W Puszczy i Nie W Puszczy	Tomasz Kanioł	300006	HORROR	LIB CARD ID: <input type="text"/>	Rent Book Edit View Info
[B]	Zaraza	Jan Malkowski	300000	HORROR	Edit	View Info

[Add New Book](#)





<< IT'S SHOWTIME >>

Book Rental Application

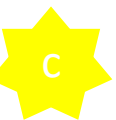




2 Implementation - Backend



Technologies used for backend



General Idea of Project – Make Our Lives Easier!



- No need to manually add dependencies to pom - we can use Spring **Initializr** and choose desired components,
- We are not creating tables/relations - **Hibernate** is doing this for us automatically,
- We are not installing **Tomcat Server** to run web app - it is already included in **Spring Boot**,
- As for Authentication/Authorization, for example preparing login page and displaying certain things for certain roles – **Spring Security** is helping us heavily!





<< IT'S SHOWTIME >>

Initializr



Spring boot – what is it?



- **Open-source java-based** framework used to create micro-Services,
- We can start our work with minimum configuration -> [Spring boot initializer](#),
- No need for installing server – tomcat is already provided,
- **Easy to understand**, lots of free tutorials available,
- Using Spring Boot, we can **eliminate** most of the **boilerplate** configuration.

We can create our applications much easier/faster!





<< IT'S SHOWTIME >>

Project's Structure





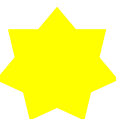
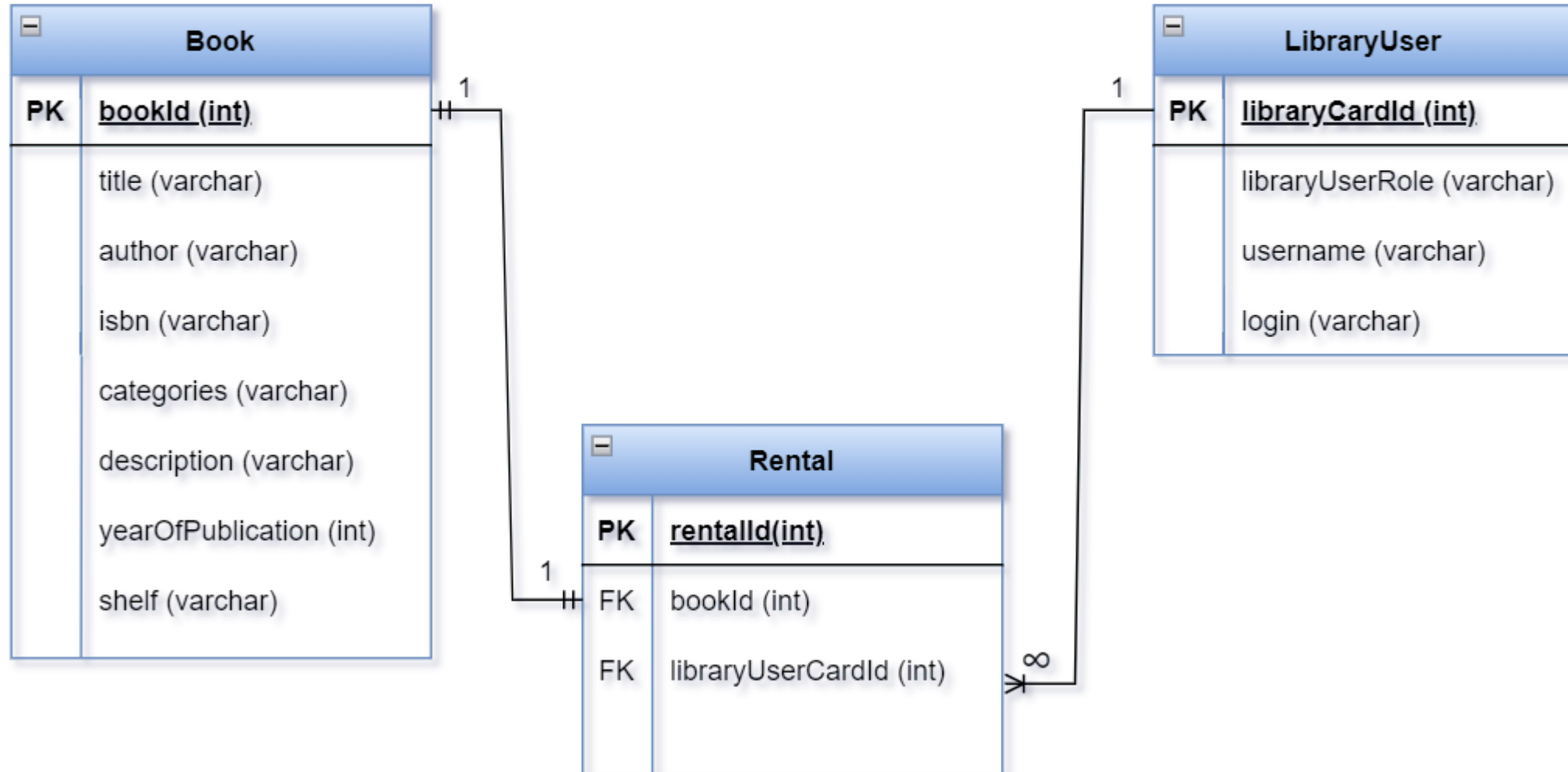
Database schema



Library data base schema



Simplified database schema of Book Rental system
used to implement project for IT FOR SHE
workshops 2022



Database –Hibernate & MySQL



- Hibernate is:
 - One of the most popular implementation of JPA (Java Persistent API),
 - **Object relational mapping (ORM) tool** providing a framework to map object-oriented domain models to relational databases for web applications (maps java classes into database tables and java data types to SQL data types and provides querying and retrieval).

- **Benefits of using hibernate:**



- Open-source,
- Database type independent – connectors available for any database like Oracle, MySQL, PostgreSQL etc
- Automatic table creation – no manual work in database needed,
- Handling database creation if it does not exist,



Database – Hibernate & MySQL



Annotation for model classes:

- All java classes that shall be converted to tables need to be annotated with **@Table**,
- Java fields of mapped classes need to have annotation **@Column**,
- For Primary Key (PK) we can use autogenerated ID within annotations **@Id** and **@GeneratedValue**,
- Relations between tables (FK) are annotated by **@ManyToOne**, **@OneToOne**, **@OneToMany**, **@ManyToMany** tags.



REST methods - overview



REST (**Re**presentational **S**tate **T**ransfer) - an architectural style defining constraints to be used for web services.

REST != HTTP

REST principles:

- Uniform interface,
- Client – server separation,
- Stateless,
- Cacheability,
- Layered system,
- Code on demand.



REST URLs/Endpoints naming – Nouns,



Our application is NOT fully RESTFUL, but we follow the guidelines to some extent:

URL / ENDPOINT	HTTP METHOD	ACTION
/users	GET	List with many users,
/users	POST	Adding new user,
/users/{id}	GET	Details page for one user having certain ID ,
/users/{id}	PUT	Updating one user having certain ID,
/books	GET	List with many books,
/books	POST	Adding new book,
/books/{id}	GET	Details page for one book having certain ID ,
/books/{id}	PUT	Updating one book having certain ID,
/rented-books	GET	List with many books that are rented ,
/books-rented-by-user/{userId}	GET	List with many books rented by certain user (user has certain userId)



Spring security in backend



Spring Security is a framework that supports authentication / authorization / access-control.

Authorization is any mechanism by which a system grants or revokes the right to perform some action or access some data.

Authentication – who are you?

Authorization – what are you allowed to do?



Spring security in backend



```
@Configuration
@EnableWebSecurity
public class SpringSecurityAccessConfiguration {

    @Autowired
    private LibraryUserRepository libraryUserRepository;

    // adding users and passwords:
    @Bean
    public InMemoryUserDetailsManager userDetailsService(PasswordEncoder
        passwordEncoder) {
        ...
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        ...
    }
}
```



Spring Security



```
http.authorizeRequests()  
  
    .antMatchers("/users/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/new-books/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/new-users/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/rented-books/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/return-books/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/edit/books/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/edit/users/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/rent-books/**").access("hasRole('ROLE_LIBRARIAN')")  
    .antMatchers("/delete/users/**").access("hasRole('ROLE_LIBRARIAN')")  
  
    .antMatchers("/my-books/**").access("hasRole('ROLE_READER')")  
    .antMatchers("/books/**").access("hasRole('ROLE_READER')")  
    .antMatchers("/books-rented-by-user/**").access("hasRole('ROLE_READER')")  
  
    .antMatchers("/**").authenticated()  
    .and().formLogin().permitAll();  
  
return http.build();
```





<< IT'S SHOWTIME

>>

Database & Model
Controllers
Thymeleaf (VIEW)





3 View - Frontend




Technologies used for frontend



Thymeleaf – what is it?



 **Thymeleaf** **open-source server-side** Java template engine for both web and standalone environment. It processes and creates HTML, XML, JavaScript, CSS and text. It provides full integration with Spring Framework.

Thymeleaf templates look similar to normal HTML format (much more similar than for example older JSP)

It is **compatible with HTML** and supports **variable expressions** (`${...}`) like Spring Expression Language and executes on model attributes, asterisk expressions (`*{...}`) execute on the form backing bean, hash expressions (`#{...}`) are for internationalization, and link expressions (`@{...}`) rewrite URLs.



Thymeleaf integration with Spring Security

Integration with Spring

Security allows to check if user is authenticated and handle displaying given elements in the form based on user's roles.

```
<div class="collapse navbar-collapse" id="navbarNav">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="#">Main Menu</a>
    </li>
    <li class="nav-item" >
      <a class="nav-link" href="/listBooks">Search Books</a>
    </li>
    <li sec:authorize="hasRole('ROLE_LIBRARIAN')" class="nav-item">
      <a class="nav-link" href="/searchUser">Search Users</a>
    </li>
    <li sec:authorize="hasRole('ROLE_LIBRARIAN')" class="nav-item">
      <a class="nav-link" href="/addBook">New Book</a>
    </li>
    <li sec:authorize="hasRole('ROLE_LIBRARIAN')" class="nav-item">
      <a class="nav-link" href="/addUser">New User</a>
    </li>
  </ul>
</div>
```



Bootstrap



 is **open-source front-end** development **framework** for the creation of websites and web apps.

Bootstrap simplifies creating responsive, mobile-first front-end programs.

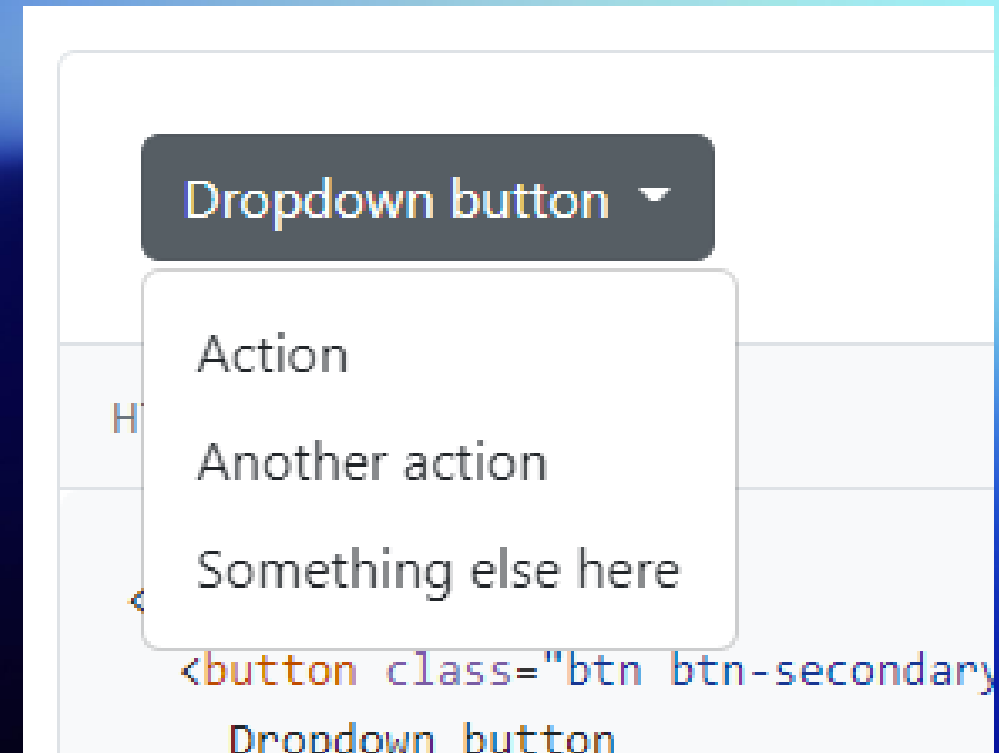
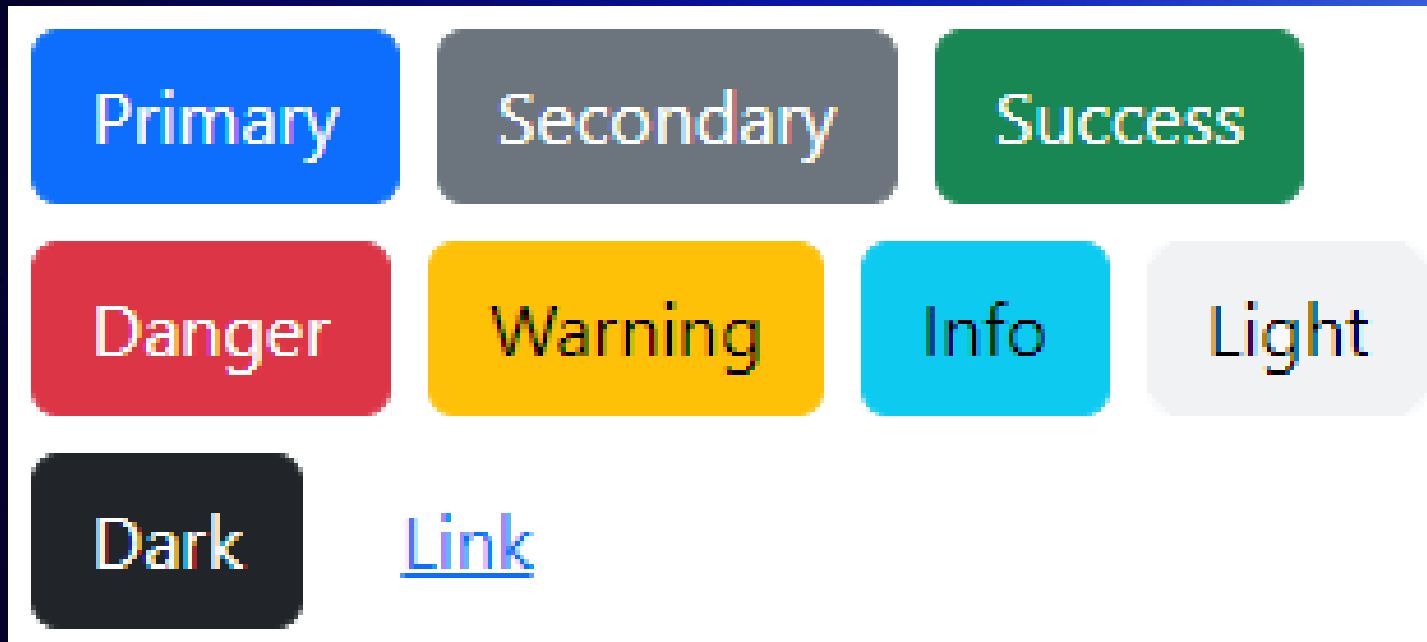
Bootstrap help us with appearance of our applications.

Bootstrap uses HTML, CSS, and JavaScript.

Bootstrap has ready solutions for many elements: buttons, alerts, modals, tabs, navigation bars, toolkits, dropdowns and many more...



Bootstrap



Bootstrap



Icons Themes Blog

Launch demo

Modal title

Woo-hoo, you're reading this text in a modal!

Close Save changes

```
<!-- Button tri
<button type="b
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="exampleModalLabel">Modal title</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"
      </div>
```

On this p

How it wor

Examples

Modal c

Live den

Static ba

Scrolling

Verticall

Tooltips

Using th

Varying

Toggle b

modals

Change

Remove



JQuery



is a **feature-rich JavaScript library**.

It simplifies operations like: manipulation of HTML document (DOM operations), animations, event handling, AJAX.

Usually, no changes in HTML tags are required.

In our Book Rental application **JQuery helps us with form validation on client side.**



JQuery example



```
• $(function() {  
    • $('#add-user-form').on('submit', function() {  
        clearPreviousValidationErrors();  
        var isValidatationOk = true;  
        isValidatationOk = validateFieldNonEmpty("userName", "add-user-form");  
        if(!isValidatationOk){  
            • $("html, body").animate({ scrollTop: 0 }, "slow");  
        }  
        return isValidatationOk;  
    });  
  
    • $('.delete-item-button').click(function() {  
        var result = confirm("Are you sure?");  
        return result;  
    });  
});
```



JQuery validate form



Adding New Book

Book title:

Title

This field cannot be empty!

Author:

Author

This field cannot be empty!

ISBN:

ISBN

This field must have format (digits-digits-digits-digits-digits)!

Description:

Description

Shelf (in library):

Shelf

Year of publication:

1





4

Questions

Bibliography



Images:

- <https://publicdomainvectors.org/pl/wektorow-swobodnych/Grafika-wektorowa-budynku-szko%C5%82y/11750.html>
- <https://openclipart.org/detail/314949/librarian-2>
- <https://freesvg.org/books>
- https://images.rawpixel.com/image_800/czNmcy1wcml2YXRIL3Jhd3BpeGVsX2ltYWdlcy93ZWJzaXRlX2NvbniRlbnQvbHIvam9iNjczLTE1OS14LmpwZWw.jpg?s=l3MJkiQKcO5Q1lqiFPEpZn-HyL-CZmKAjsEnk9XR1bA
- <https://openclipart.org/detail/298193/librarian>

Bibliography (2)



- <https://www.interviewbit.com/blog/difference-between-spring-mvc-and-spring-boot/>
- <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>
- <https://www.wimdeblauwe.com/blog/2021/09/23/todomvc-with-spring-boot-and-thymeleaf-part-2/>
- <https://www.thymeleaf.org/>
- <https://www.baeldung.com/thymeleaf-in-spring-mvc>

