OurLib

Advanced object-oriented programming

Kornel Stefańczyk
Technical University of Varna
Erasmus+ student from:
Wrocław University of Science and Technology

28 January 2021
version: 1.1

Contents

1 Requirements

Design and implementation of an information system - Library. The system must store and processing data for books and readers. The system allows multiple access (Many users in same time). The system supports two types of users - administrator and clients (operators and readers) with different roles to access the functionalities in the system.

1.1 Users

User operations:

- Creation of operators by an administrator
- Creating and unsubscribing readers from an operator
- Form for creating a reader profile
- Renting books

1.2 Books

The system supports operations for working with books:

- Adding new books (Inventory number, title, author, genre. . .)
- Borrowing books (different level of security when borrowing books (reading room, for home))
- Return of a book
- Disposal of damaged books
- Archiving of old editions (using only in the reading room)

1.3 Reports

The system maintains reports for:

- Submitted forms (date, status, content of the form)
- Books (Book status, book information)
- Users (approval date, book list, user information)
- User rating (loyal and disloyal readers)

1.4 Event Notifications

The system supports Event Notifications(working with threads):

- Request for opening a reader profile
- Notification of the need to archive a book
- Delayed return of books

2 System modules

OurLib project can be divided in three main modules.

2.1 Database

Database module is obligatory to store users and books data.

2.2 Backend

Backend module is the bridge between database module and GUI module.

This module is responsible for implementation of business logic. Backend receive data from GUI, validate it and communicate with database to read, create or modify data stored in database module.

2.3 GUI

Graphical user interface module is only available for system user module . This module is responsible for communication with backend module. GUI interface changes according to data get from backend module. It presents all features available for user using graphical elements.

3 System design

3.1 Database

MySQL was selected as database service technology.

ERD schematic is presented at the figure 1.

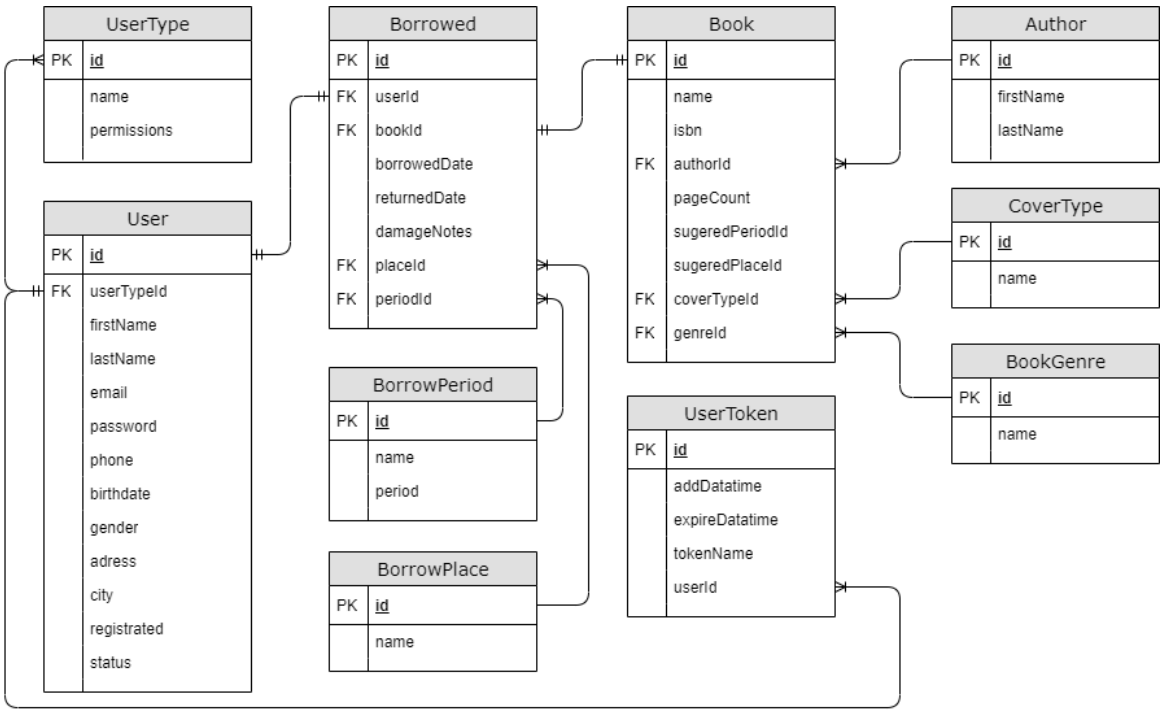

Figure 1: ERD schematic diagram

Database tables:

- *UserType* table should be prepared before system deployment. Data from this table will be not available to change from the system.
- *User* table contains user data. Each of users in the system has separated entry. User type is defined by userTypeId that reefers to the *UserType* table.
- *UserToken* table contains security information about logged in users.
- *BorrowedPeriod* table contains information about how long book can be borrowed by user.
- *BorrowedPlace* table contains the name of place where book can be read.

- *BookGenere* table contains name of the type of the book.
- *CoverType* table contains all possible cover types of the books in library.
- *Author* table store basic personal data of the authors of the books.
- *Book* table store all books existing in library.
- *Borrowed* table store all borrow orders details.

3.2 Backend

Backend of the system is based on two main sub modules:

- Data layer
- Business logic

Backend uses Java SpringBoot technology.

3.2.1 Data layer

Data layer uses Hibernate. It allows to perform all CRUD methods using JPA api. Including setup of database.

3.2.2 Business logic

Services realizes all needed functionality of system. Data models were designed using Swagger API Documentation and Development tool. It allows to generate models of classes containing data and Rest points structure. Source code of Swagger specification can be found in Appendix A

3.3 Graphical user interface

Graphical user interface is based on JavaScript React framework.

4 Implementation

4.1 Database

MySQL database *library_db* was set up and new user account was created. Backend is connected to database through *api* user.

Backend create on startup default values needed for system set up.

4.2 Backend

4.2.1 Models

Models of data were generated using Swagger web app. Definition of all models can be found in section 6. For example definition of User class in Swagger model is presented in listing 1.

Listing 1: User model

```
35    User:
36      type: object
37      properties:
38        id:
39          type: integer
40          format: int64
41        userTypeId:
42          type: integer
43          format: int64
44        firstName:
```

```
45          type: string
46        lastName:
47          type: string
48        email:
49          type: string
50        password:
51          type: string
52        phone:
53          type: string
54        birthdate:
55          type: string
56          format: date-time
57        registrated:
58          type: string
59          format: date-time
60        gender:
61          type: string
62          enum:
63            - male
64            - female
65            - other
66        address:
67          type: string
68        city:
69          type: string
70        status:
71          type: string
72          description: User Status
73          enum:
74            - active
75            - suspended
76            - inactive
77            - to veryfication
78      xml:
79        name: User
```

It was needed to add annotation from *javax.persistence* like: *Entity*, *GeneratedValue*, *GenerationType*, *Id*, *Table* to default generated models. These annotations are using to communicate with database properly. Example of setting up code can be seen in listing 2.

Listing 2: Model persistence annotations

```
1  @Entity
2  @Table(name = "book")
3  @JsonRootName("Book")
4  public class Book   {
5    @Id
6    @GeneratedValue(strategy=GenerationType.AUTO)
7    @JsonProperty("id")
8    private Long id = null;
9
10   @JsonProperty("name")
11   private String name = null;
```

4.2.2 Services

Services are implementation of business logic and all needed manipulation of data in database.

Each of main data segments had implemented needed functionalities. Example book service functionalities are included in listing 3.

Listing 3: Book services interface

```
1  public interface BookService {
2    public Void initBookValues();
3    public int countBooks();
4    public int countAvailableBooks();
5    public Book createBook(Book book);
6    public List<Book> getAllBooks();
7    public Book deleteBookById(Long id);
8    public Book getBookById(Long id);
9    public String getBookNameById(Long id);
10   public Book updateBookById(Long id, Book body);
11   public Boolean updateBookStatus(Long id, StatusEnum status);
12   public List<Book> getAllAvailableBooks();
```

Services were using JPA repositories to manipulate data stored in database like in listing 4.

Listing 4: User service create user

```
1  @Override
2  public User createUser(User user) {
3    OffsetDateTime registrated = OffsetDateTime.now(ZoneId.of("UTC"));
4    user.setRegistrated(registrated);
5
6    Optional<User> od = userRepository.getUserByEmail(user.getEmail());
7    if(!od.isPresent()) {
8        log.info("Create new user:\n"+user.toString());
9        return userRepository.save(user);
10   } else {
11       log.error("Sorry, user with  given email exist. Email: "+user.
             getEmail());
12   }
13   return null;
14 }
```

4.3 Graphical user interface

GUI was written in JavaFX framework.

4.3.1 Connection with backend

Connection with backend was realized by execution service methods. Example of service request is in listing 5.

Listing 5: Connection with backend through *userService*

```
1  public void saveButtonOnAction(ActionEvent actionEvent) {
2    if(getInitialUser() != null ){
3        System.out.println(getDataFromForm().toString());
4        userService.updateUserById(getInitialUser().getId(),
             getDataFromForm());
5    }
6  }
```

4.3.2 Sidebar

Main navigation element of web app is sidebar. It changes according to logged in user. Each of user type has different set of visible buttons on sidebar. Listing 6 contains id's of buttons and allowed role for each button.

Listing 6: Sidebar buttons with allowed user types

```
1  public Boolean isButtonValidForUserType(String buttonId) {
2    if (LoginController.getUser() != null) {
3        if (buttonId.equals("findBooksButton")) {
4            return LoginController.isReader() || LoginController.
                 isLibrarian();
5        }
6        if (buttonId.equals("ordersButton")) {
7            return LoginController.isReader();
8        }
9        if (buttonId.equals("manageBooksButton")) {
10           return LoginController.isLibrarian();
11       }
12       if (buttonId.equals("manageOrdersButton")) {
13           return LoginController.isLibrarian();
14       }
15       if (buttonId.equals("allUsersButton")) {
16           return LoginController.isAdmin();
17       }
18       if (buttonId.equals("readersButton")) {
19           return LoginController.isAdmin() || LoginController.isLibrarian
                 ();
20       }
21       if (buttonId.equals("reportsButton")) {
22           return LoginController.isAdmin();
23       }
24       if (buttonId.equals("accountButton")) {
25           return true;
26       }
27   }
28   return false;
29 }
```

Button will be visible only when user type will be valid else button will be deleted. Listing 7.

Listing 7: Remove sidebar buttons not allowed for user types

```
1    private void removeNotValidButtons() {
2      if (buttonsVBox != null) {
3        ObservableList<Node> nodes = FXCollections.observableArrayList(
             buttonsVBox.getChildren());
4        for (Node node : nodes) {
5            if (!this.mainViewService.isButtonValidForUserType(node.
                 idProperty().get())) {
6                buttonsVBox.getChildren().remove(node);
7            }
8        }
9      }
10   }
```

Different sidebar buttons can be seen at figures:

- Admin - figure 3
- Librarian - figure 6
- Reader - figure 7

4.3.3 Switching context

Beside sidebar buttons main view contains switching context. Right side is defined as *BorderPane* - listing 8.

Listing 8: Switching context of main view - FXML

```
<ScrollPane prefHeight="-1.0" prefWidth="-1.0">
<content>
    <BorderPane fx:id="mainPane" />
</content>
</ScrollPane>
```

After pressing any button of sidebar new context is loaded - listings 9 and 10.

Listing 9: Switching context - button detection

```
public void updateMainPane(ActionEvent actionEvent) {
  String buttonId = ((Button) actionEvent.getSource()).getId();
  String buttonName = ((Button) actionEvent.getSource()).getText();
  setLastChoseButtonAs(buttonName);
  // System.out.println("buttonId: " + buttonId + " buttonName: " +
      buttonName);
  mainPane.setCenter(mainViewService.getMainPane(buttonId));
}
```

Listing 10: Switching context of main view - loading new view

```
public Pane getMainPane(String buttonId) {
  // System.out.println(buttonId);
  try {
      FxWeaver fxWeaver = JavaFxApplication.applicationContext.getBean(
          FxWeaver.class);
      Node node = fxWeaver.loadView(NotImplementedController.class);
      if(buttonId.equals("accountButton")){
          node = fxWeaver.loadView(AccountPaneController.class);
      }
      return (Pane) node;
  } catch (Exception e) {
      e.printStackTrace();
      return null;
  }
}
```

4.3.4 Forms

Forms was created using *GridPane*, *TextField* and *ChoiceBox*. Forms are encapsulate in larger structures like main view 11.

Listing 11: Encapsulation of userDetailsPane in main view Pane

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox alignment="CENTER" prefHeight="100.0" prefWidth="350.0" spacing
    ="10" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="io.swagger.app.controller.
    AccountPaneController">
```

```
 9      <BorderPane fx:id="userDetailsPane" prefHeight="200.0" prefWidth
            ="200.0">
10          <top>
11              <Label text="Personal Info" BorderPane.alignment="CENTER">
12              <font>
13                  <Font size="24.0" />
14              </font></Label>
15          </top>
16      </BorderPane>
17      <BorderPane fx:id="passwordPane" prefHeight="200.0" prefWidth
            ="200.0">
18          <top>
19              <Label text="Change Password" BorderPane.alignment="CENTER">
20              <font>
21                  <Font size="24.0" />
22              </font></Label>
23          </top>
24      </BorderPane>
25  </VBox>
```

User details form is presented in listing 12.

Listing 12: Account personal data form

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2
 3  <?import java.lang.String ?>
 4  <?import javafx.collections.FXCollections ?>
 5  <?import javafx.scene.control.Button ?>
 6  <?import javafx.scene.control.ChoiceBox ?>
 7  <?import javafx.scene.control.DatePicker ?>
 8  <?import javafx.scene.control.Label ?>
 9  <?import javafx.scene.control.Separator ?>
10  <?import javafx.scene.control.TextField ?>
11  <?import javafx.scene.layout.ColumnConstraints ?>
12  <?import javafx.scene.layout.GridPane ?>
13  <?import javafx.scene.layout.HBox ?>
14  <?import javafx.scene.layout.RowConstraints ?>
15  <?import javafx.scene.layout.VBox ?>
16
17  <VBox fx:id="mainVBox" alignment="CENTER" prefWidth="350.0" spacing="10"
        xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/
        fxml/1" fx:controller="io.swagger.app.controller.
        UserDetailsFormController">
18      <GridPane>
19          <columnConstraints>
20              <ColumnConstraints hgrow="SOMETIMES" />
21              <ColumnConstraints hgrow="SOMETIMES" />
22              <ColumnConstraints hgrow="SOMETIMES" />
23              <ColumnConstraints hgrow="SOMETIMES" />
24          </columnConstraints>
25          <rowConstraints>
26              <RowConstraints vgrow="SOMETIMES" />
27              <RowConstraints vgrow="SOMETIMES" />
28              <RowConstraints vgrow="SOMETIMES" />
29              <RowConstraints vgrow="SOMETIMES" />
30              <RowConstraints vgrow="SOMETIMES" />
31              <RowConstraints vgrow="SOMETIMES" />
32          </rowConstraints>
```

```
33          <children>
34              <Label text="First Name" GridPane.columnSpan="2" />
35              <TextField fx:id="firstNameTextField" GridPane.columnIndex
                    ="0" GridPane.rowIndex="1" />
36              <Label text="Last Name" GridPane.columnIndex="1" />
37              <TextField fx:id="lastNameTextField" GridPane.columnIndex="1"
                     GridPane.rowIndex="1" />
38              <Label text="Status" GridPane.columnIndex="2" />
39              <ChoiceBox fx:id="statusChoiceBox" GridPane.columnIndex="2"
                    GridPane.rowIndex="1">
40                  <items>
41                      <FXCollections fx:factory="observableArrayList">
42                          <String fx:value="active" />
43                          <String fx:value="suspended" />
44                          <String fx:value="inactive" />
45                          <String fx:value="to veryfication" />
46                      </FXCollections>
47                  </items>
48              </ChoiceBox>
49              <Label text="User type" GridPane.columnIndex="3" />
50              <ChoiceBox fx:id="userTypeChoiceBox" GridPane.columnIndex="3"
                     GridPane.rowIndex="1">
51                  <items>
52                      <FXCollections fx:factory="observableArrayList">
53                          <String fx:value="Administrator" />
54                          <String fx:value="Librarian" />
55                          <String fx:value="Reader" />
56                      </FXCollections>
57                  </items>
58              </ChoiceBox>
59              <Label text="Phone" GridPane.columnIndex="0" GridPane.
                    rowIndex="2" />
60              <TextField fx:id="phoneTextField" GridPane.columnIndex="0"
                    GridPane.rowIndex="3" />
61              <Label text="Email" GridPane.columnIndex="1" GridPane.
                    rowIndex="2" />
62              <TextField fx:id="emailTextField" GridPane.columnIndex="1"
                    GridPane.rowIndex="3" />
63              <Label text="Password" GridPane.columnIndex="2" GridPane.
                    rowIndex="2" />
64              <TextField fx:id="passwordTextField" GridPane.columnIndex="2"
                     GridPane.rowIndex="3" />
65              <Label text="Registrated" GridPane.columnIndex="3" GridPane.
                    rowIndex="2" />
66              <Label fx:id="registratedLabel" GridPane.columnIndex="3"
                    GridPane.rowIndex="3"  text="Not available"/>
67              <Label text="Birthday" GridPane.rowIndex="4" />
68              <DatePicker fx:id="birthdayDatePicker" GridPane.rowIndex="5"
                    />
69              <Label text="Gender" GridPane.columnIndex="1" GridPane.
                    rowIndex="4" />
70              <ChoiceBox fx:id="genderChoiceBox" GridPane.columnIndex="1"
                    GridPane.rowIndex="5">
71                  <items>
72                      <FXCollections fx:factory="observableArrayList">
73                          <String fx:value="male" />
74                          <String fx:value="female" />
```

```
75                          <String fx:value="other" />
76                      </FXCollections>
77                  </items>
78              </ChoiceBox>
79              <Label text="Adress" GridPane.columnIndex="2" GridPane.
                    rowIndex="4" />
80              <TextField fx:id="adressTextField" GridPane.columnIndex="2"
                    GridPane.rowIndex="5" />
81              <Label text="City" GridPane.columnIndex="3" GridPane.rowIndex
                    ="4" />
82              <TextField fx:id="cityTextField" GridPane.columnIndex="3"
                    GridPane.rowIndex="5" />
83          </children>
84      </GridPane>
85      <HBox alignment="TOP_CENTER" prefHeight="100.0" prefWidth="200.0">
86          <children>
87              <Button mnemonicParsing="false" onAction="#saveButtonOnAction
                    " text="Save" />
88              <Separator orientation="VERTICAL" prefHeight="100.0"
                    prefWidth="25.0" visible="false" />
89              <Button mnemonicParsing="false" onAction="#loadButtonOnAction
                    " text="Load Current User" />
90          </children>
91      </HBox>
92  </VBox>
```

4.3.5 Screenshots of GUI

Screenshots of different application views are presented below.

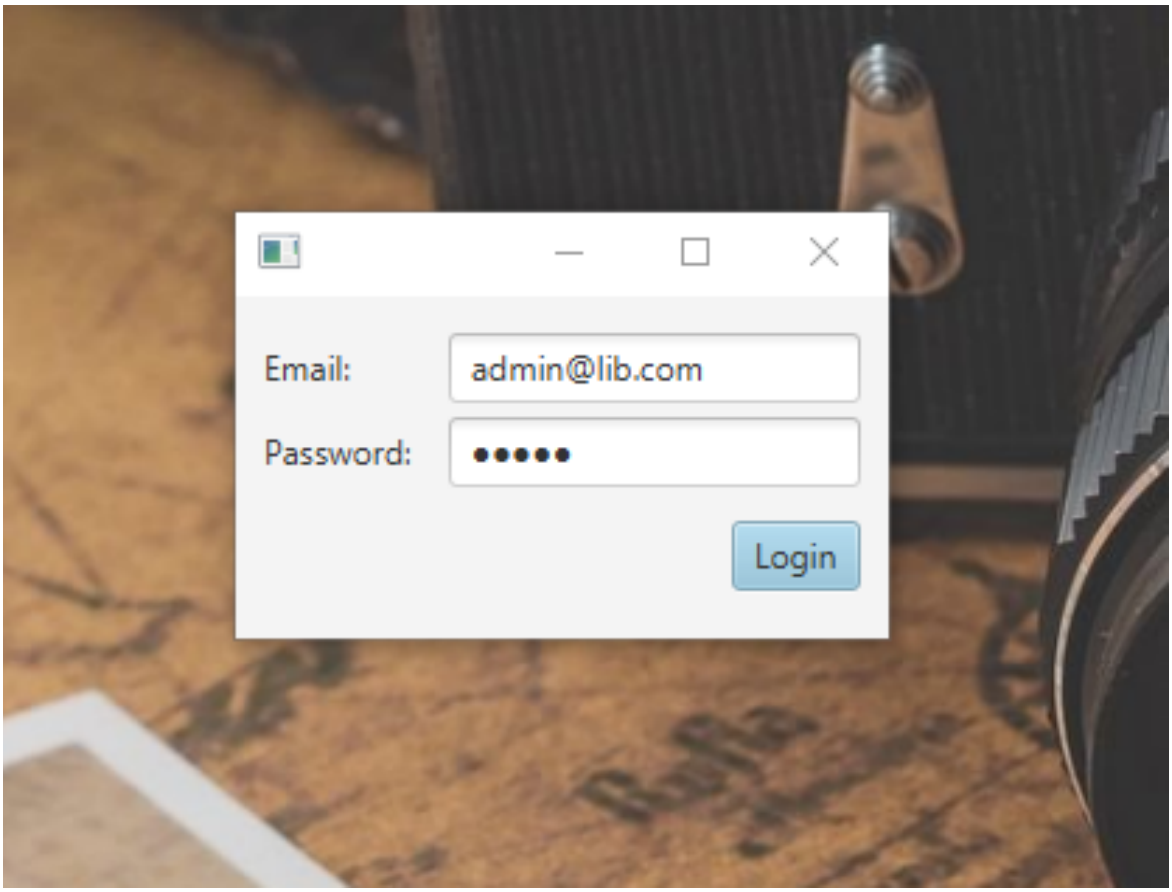

Figure 2: Login window

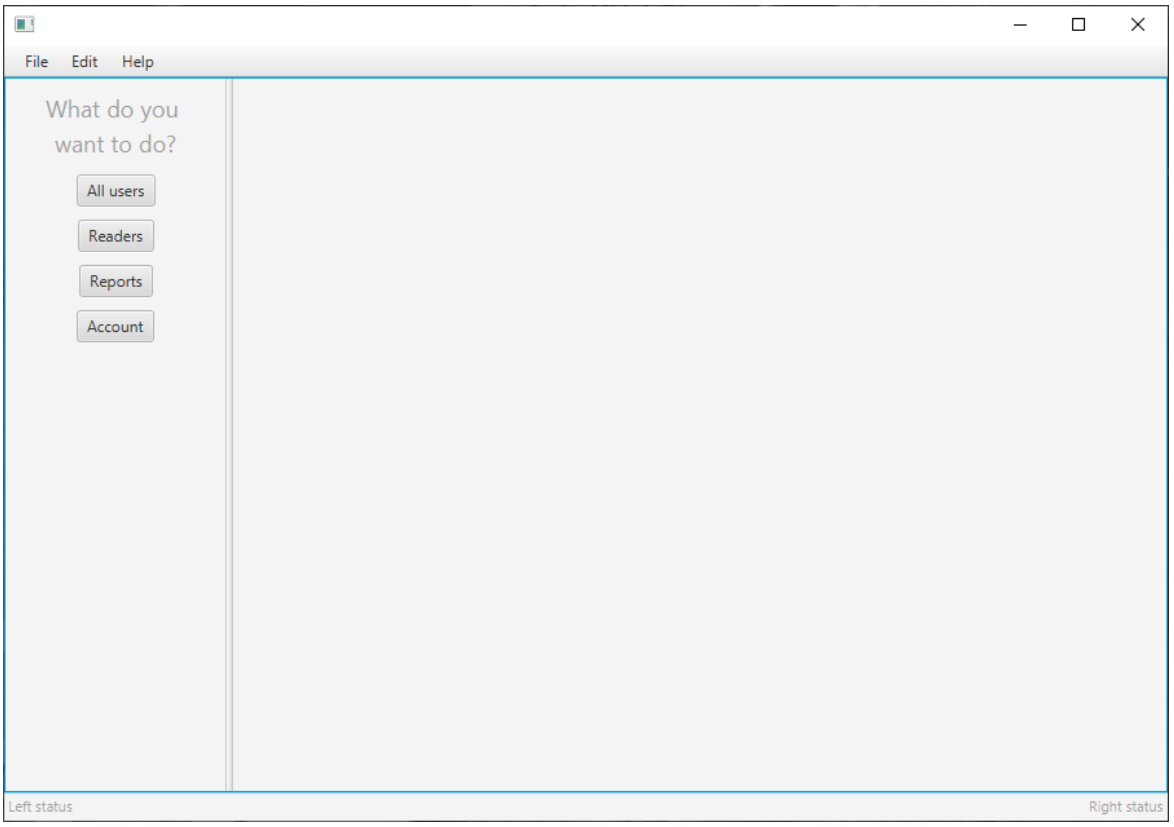

Figure 3: View before selection

Figure 4: View of not implemented section

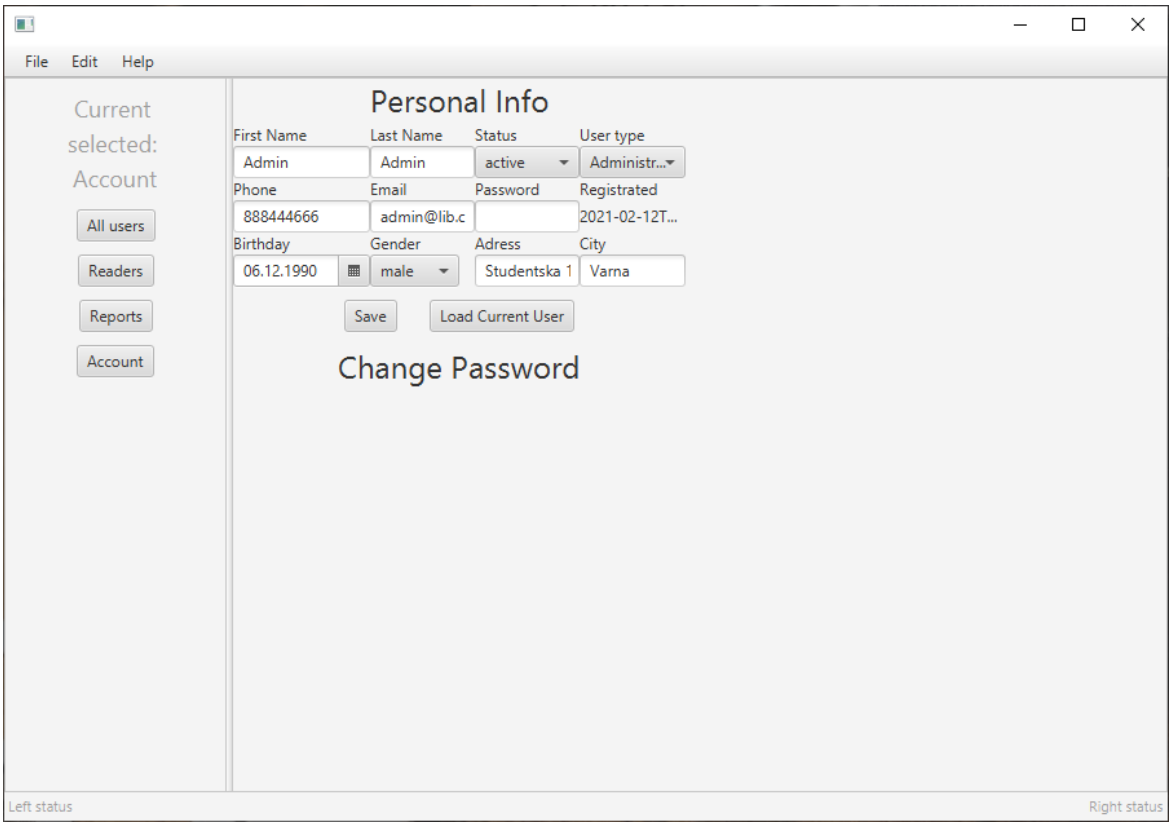

Figure 5: Account section view

Figure 6: Librarian sidebar buttons

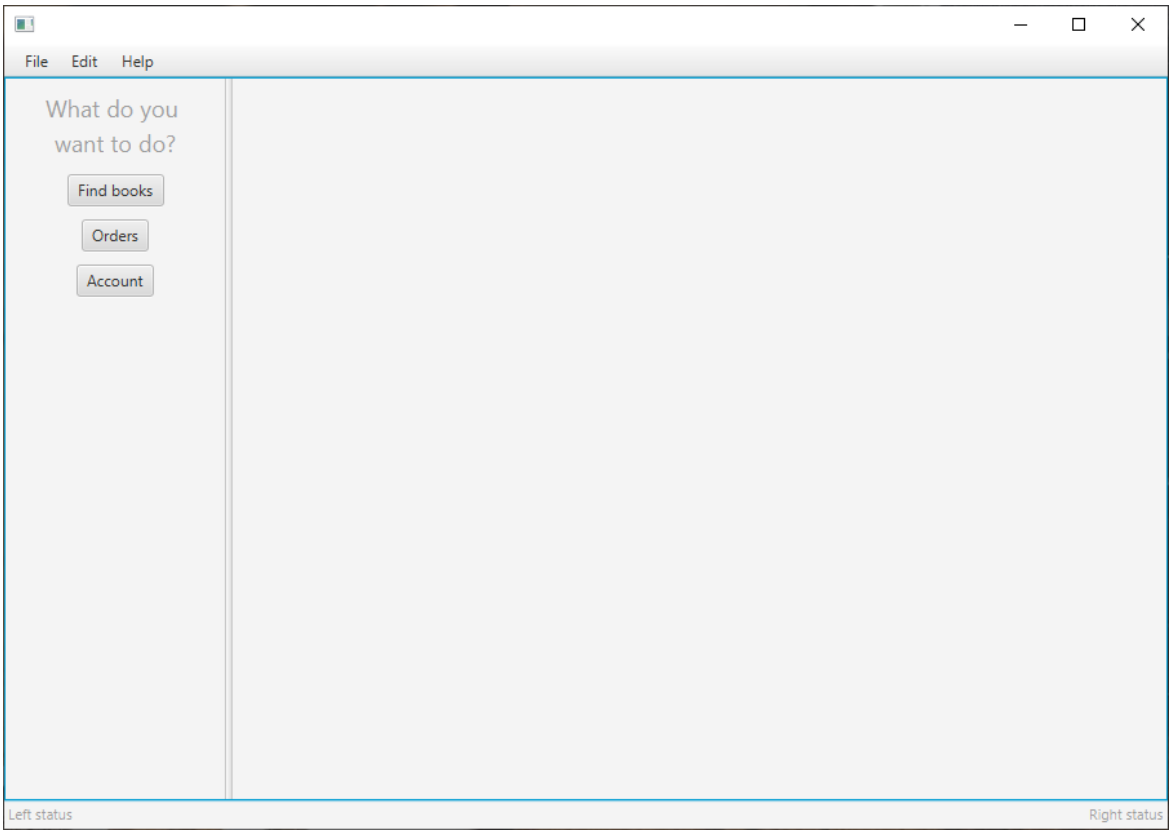

Figure 7: Reader sidebar buttons

5 Developer version deployment

To deploy development version of OurLib system a few steps are needed.

5.1 Database

Initialize database with following steps:

- MySQL server should be installed on your computer.
- Table *library_db* should be set up.
- New database user account with credentials login: *api* and password: *api* should be created.

5.2 Backend and Frontend

Initialize backend with following steps:

- Maven and JDK tools should be installed on your computer.

5.3 Run project

To run project on computer with Windows OS script *run.bat* can be used.

6 Appendix A

```yaml
swagger: '2.0'
info:
  description: Library API
  version: 1.0.3_non_Rest
  title: Library Varna
  termsOfService: http://swagger.io/terms/
  contact:
    email: kornelstefanczyk@wp.pl
  license:
    name: Apache 2.0
    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
host: localhost:8080
# basePath: /
schemes:
- http
paths:
  /rest:
    get:
      operationId: rest
      responses:
        default:
          description: successful operation

definitions:
  UserLoginBody:
    type: object
    properties:
      email:
        description: The email for login
        type: string
        format: email
      password:
        description: The password for login in clear text
        type: string
  User:
    type: object
    properties:
      id:
        type: integer
        format: int64
      userTypeId:
        type: integer
        format: int64
      firstName:
        type: string
      lastName:
        type: string
      email:
        type: string
      password:
        type: string
      phone:
        type: string
      birthdate:
        type: string
        format: date-time
      registrated:
```

```
58          type: string
59          format: date-time
60        gender:
61          type: string
62          enum:
63            - male
64            - female
65            - other
66        address:
67          type: string
68        city:
69          type: string
70        status:
71          type: string
72          description: User Status
73          enum:
74            - active
75            - suspended
76            - inactive
77            - to veryfication
78      xml:
79        name: User
80    UserType:
81      type: object
82      properties:
83        id:
84          type: integer
85          format: int64
86        name:
87          type: string
88        permissions:
89          type: string
90      xml:
91        name: UserType
92    SubmitUserReport:
93      allOf:
94        - $ref: '#/definitions/User'
95        - type: object
96          properties:
97            userTypeName:
98              type: string
99    UserStatusReport:
100     allOf:
101       - $ref: '#/definitions/User'
102       - $ref: '#/definitions/UserStatus'
103       - type: object
104         properties:
105           userTypeName:
106             type: string
107           currentBorrowed:
108             type: array
109             items:
110               type: object
111               properties:
112                 bookId:
113                   type: integer
114                   format: int64
```

```
                bookName:
                  type: string

  UsersRatingReport:
    type: object
    properties:
      loyal:
        type: array
        items:
          type: object
          properties:
            userId:
              type: integer
              format: int64
            userName:
              type: string
      disloyal:
        type: array
        items:
          type: object
          properties:
            userId:
              type: integer
              format: int64
            userName:
              type: string

  Author:
    type: object
    properties:
      id:
        type: integer
        format: int64
      firstName:
        type: string
      lastName:
        type: string
    xml:
      name: Author
  CoverType:
    type: object
    properties:
      id:
        type: integer
        format: int64
      name:
        type: string
    xml:
      name: CoverType
  Borrowed:
    type: object
    properties:
      id:
        type: integer
        format: int64
      userId:
        type: integer
```

```
172           format: int64
173         bookId:
174           type: integer
175           format: int64
176         borrowedDate:
177           type: string
178           format: date-time
179         returnedDate:
180           type: string
181           format: date-time
182         damageNotes:
183           type: string
184         placeId:
185           type: integer
186           format: int64
187         periodId:
188           type: integer
189           format: int64
190       xml:
191         name: Borrowed
192     Book:
193       type: object
194       properties:
195         id:
196           type: integer
197           format: int64
198         name:
199           type: string
200         isbn:
201           type: string
202         authorId:
203           type: integer
204           format: int64
205         pageCount:
206           type: integer
207         coverTypeId:
208           type: integer
209           format: int64
210         genreId:
211           type: integer
212           format: int64
213         sugeredPeriodId:
214           type: integer
215           format: int64
216         sugeredPlaceId:
217           type: integer
218           format: int64
219         status:
220           type: string
221           description: Book Status
222           enum:
223             - available
224             - in use
225             - archived
226       xml:
227         name: Book
228     BookReport:
```

```
229       allOf:
230         - $ref: '#/definitions/Book'
231         - type: object
232           properties:
233             authorTypeName:
234               type: string
235             coverTypeName:
236               type: string
237             genreName:
238               type: string
239             sugeredPeriodName:
240               type: string
241             sugeredPlaceName:
242               type: string
243             statusName:
244               type: string
245
246     BookGenre:
247       type: object
248       properties:
249         id:
250           type: integer
251           format: int64
252         name:
253           type: string
254       xml:
255         name: BookGenre
256     BorrowPeriod:
257       type: object
258       properties:
259         id:
260           type: integer
261           format: int64
262         name:
263           type: string
264         period:
265           type: integer
266       xml:
267         name: BorrowPeriod
268     BorrowPlace:
269       type: object
270       properties:
271         id:
272           type: integer
273           format: int64
274         name:
275           type: string
276       xml:
277         name: BorrowPlace
278     LibraryBooksReport:
279       type: object
280       properties:
281         numberOfBooks:
282           type: integer
283         numberOfAvailableBooks:
284           type: integer
285         books:
```

```
286           type: array
287           items:
288             $ref: '#/definitions/BookReport'
289       xml:
290         name: LibraryBooksReport
291     UserStatus:
292       type: object
293       properties:
294         numberOfAllBorrowedBooks:
295           type: integer
296         numberOfAllDamagedBooks:
297           type: integer
298         numberOfCurrentBorrowedBooks:
299           type: integer
300         numberOfDelayedBooks:
301           type: integer
302       xml:
303         name: UserStatus
304     NotificationForm:
305       type: object
306       properties: # also possible to model with oneOf+discriminator
307         id:
308           type: integer
309           description: my event id
310         event:
311           type: string
312           description: my event type
313         data:
314           type: object # object in case we have a complex structure,
                  otherwise string, integer, ..., as usual
315       xml:
316         name: UserStatus
317     UserToken:
318       type: object
319       properties:
320         id:
321           type: integer
322           format: int64
323         addDatatime:
324           type: string
325           format: date-time
326         tokenName:
327           type: string
328         userId:
329           type: integer
330           format: int64
331         expireDatatime:
332           type: string
333           format: date-time
334
335 externalDocs:
336     description: Find out more about Swagger
337     url: "http://swagger.io"
```