ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

КНИТУ-КАИ им. А.Н. Туполева

Колледж Информационных Технологий

Отчет по лабораторной работе

по дисциплине «Технология разработки программного обеспечения»

Описание программы

Выполнили студенты группы 4341

Зарипов Ислам Радифович

Горбачев Сергей Максимович

**Содержание**

[1. Общие сведения 3](#_Toc530651700)

[1.1. Обозначение и наименование программы 3](#_Toc530651701)

[1.2. Программное обеспечение, необходимое для функционирования программы 3](#_Toc530651702)

[1.3. Языки программирования, на которых написана программа 3](#_Toc530651703)

[2. Функциональное назначение 3](#_Toc530651704)

[2.1. Классы решаемых задач 3](#_Toc530651705)

[2.2. Назначение программы 4](#_Toc530651706)

[2.3. Сведения о функциональных ограничениях на применение 4](#_Toc530651707)

[3. Описание логической структуры 5](#_Toc530651708)

[3.1. Алгоритм программы 5](#_Toc530651709)

[3.2. Используемые методы 8](#_Toc530651710)

[3.3. Структура программы с описанием функций составных частей и связи между ними 9](#_Toc530651711)

[3.4. Связи программы с другими программами 10](#_Toc530651712)

[4. Используемые технические средства 10](#_Toc530651713)

[5. Вызов и загрузка 10](#_Toc530651714)

[6. Входные данные 10](#_Toc530651715)

[7. Выходные данные 11](#_Toc530651716)

[Приложение 1. Модели 12](#_Toc530651717)

[1. Файл Author.java 12](#_Toc530651718)

[2. Файл Book.java 12](#_Toc530651719)

[3. Файл Reader.java 13](#_Toc530651720)

[4. Файл Librarian.java 14](#_Toc530651721)

[5. Файл Relationship.java 14](#_Toc530651722)

[6. Файл Configuration.java 15](#_Toc530651723)

[7. Файл Constants.java 15](#_Toc530651724)

[8. Файл DataBaseEntity.java 15](#_Toc530651725)

[9. Файл DataBase.java 16](#_Toc530651726)

[Приложение 2. Представления. 21](#_Toc530651727)

[1. Файл login.fxml 21](#_Toc530651728)

[2. Файл admin.fxml 24](#_Toc530651729)

[3. Файл add\_librarian\_dialog.fxml 28](#_Toc530651730)

[4. Файл add\_reader\_dialog.fxml 29](#_Toc530651731)

[5. Файл add\_book\_dialog.fxml 30](#_Toc530651732)

[6. Файл add\_author\_dialog.fxml 32](#_Toc530651733)

[7. Файл librarian.fxml 33](#_Toc530651734)

[8. Файл give\_book\_dialog.fxml 35](#_Toc530651735)

[9. Файл accept\_book\_dialog.fxml 37](#_Toc530651736)

[Приложение 3. Контроллеры 39](#_Toc530651737)

[1. Файл LoginController.java 39](#_Toc530651738)

[2. Файл AdminController.java 40](#_Toc530651739)

[3. Файл AdminAddController.java 45](#_Toc530651740)

[4. Файл LibrarianController.java 47](#_Toc530651741)

[5. Файл LibrarianActionController.java 50](#_Toc530651742)

[Приложение 4. Структура базы данных 54](#_Toc530651743)

**Основная часть**

# Общие сведения

## Обозначение и наименование программы

Автоматическая информационная система учета читального зала или «Система учета читального зала» имеет следующие атрибуты:

* Наименование исполняемого файла ReadingRoom.exe
* Размер исполняемого файла 182 КБ
* «Иконка» исполняемого файла 
* Версия файла 1.0
* Версия продукта 1.0
* Внутреннее имя ReadingRoom
* Производитель КНИТУ-КАИ
* Язык интерфейса Русский

## Программное обеспечение, необходимое для функционирования программы

* Операционная система: Windows 7 и выше, Linux, macOS;
* Среда выполнения Java: JRE 9.0 и выше;
* СУБД: MySQL 5.0 и выше.

## Языки программирования, на которых написана программа

Программа «Система учета читального зала» была реализована на языке высокого уровня Java, с использованием платформы JavaFX.

# Функциональное назначение

## Классы решаемых задач

* **Администрирование**: система позволяет управлять определенными структурами, такими как: пользовательские данные читателей, информация о книгах, информация об авторах, а также вести контроль доступа в систему.
* **Обслуживание**: система позволяет определенному кругу лиц выполнять работы по обслуживанию пользователей – читателей (выдача и принятие книг).
* **Анализ**: система позволяет производить анализ текущего состояния для принятия решений (поиск и сортировка данных).

## Назначение программы

Данная программа предназначена для автоматизации учетных работ читального зала, программа может использоваться как система контроля библиотекой.

Программа реализует следующие функции:

1. Управление списками сущностей (автор, книга, библиотекарь, читатель), и включает в себя: добавление, поиск, удаление.
2. Выдача книги читателю
3. Принятие книги от читателя

## Сведения о функциональных ограничениях на применение

Программа не позволяет полностью автоматизировать внутренние процессы библиотеки, такие как: выдача книги, принятие книги, регистрация читателей, а только автоматизирует учет этих процессов.

# Описание логической структуры

## Алгоритм программы

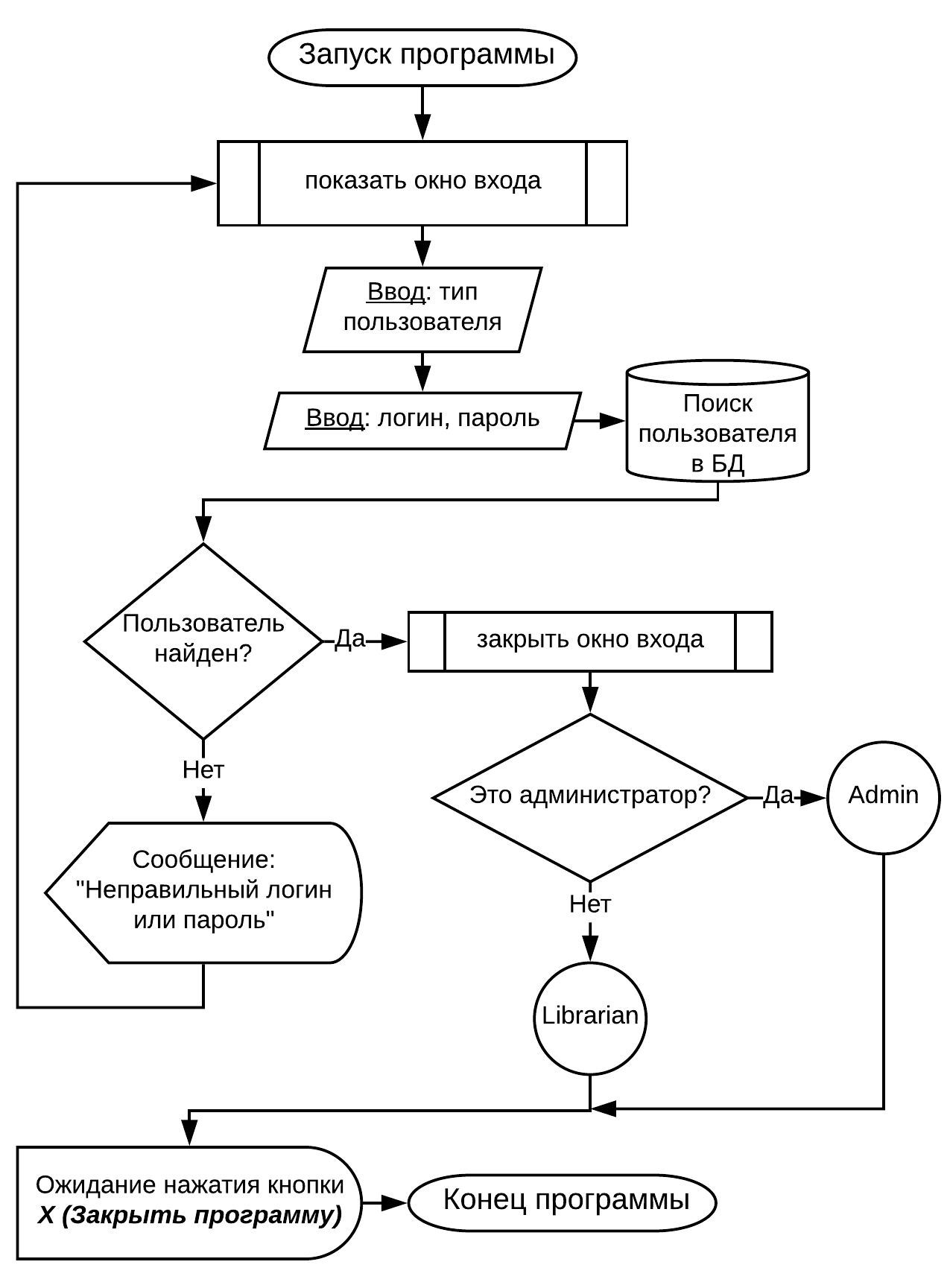


Рисунок 1. Блок схема входа в программу

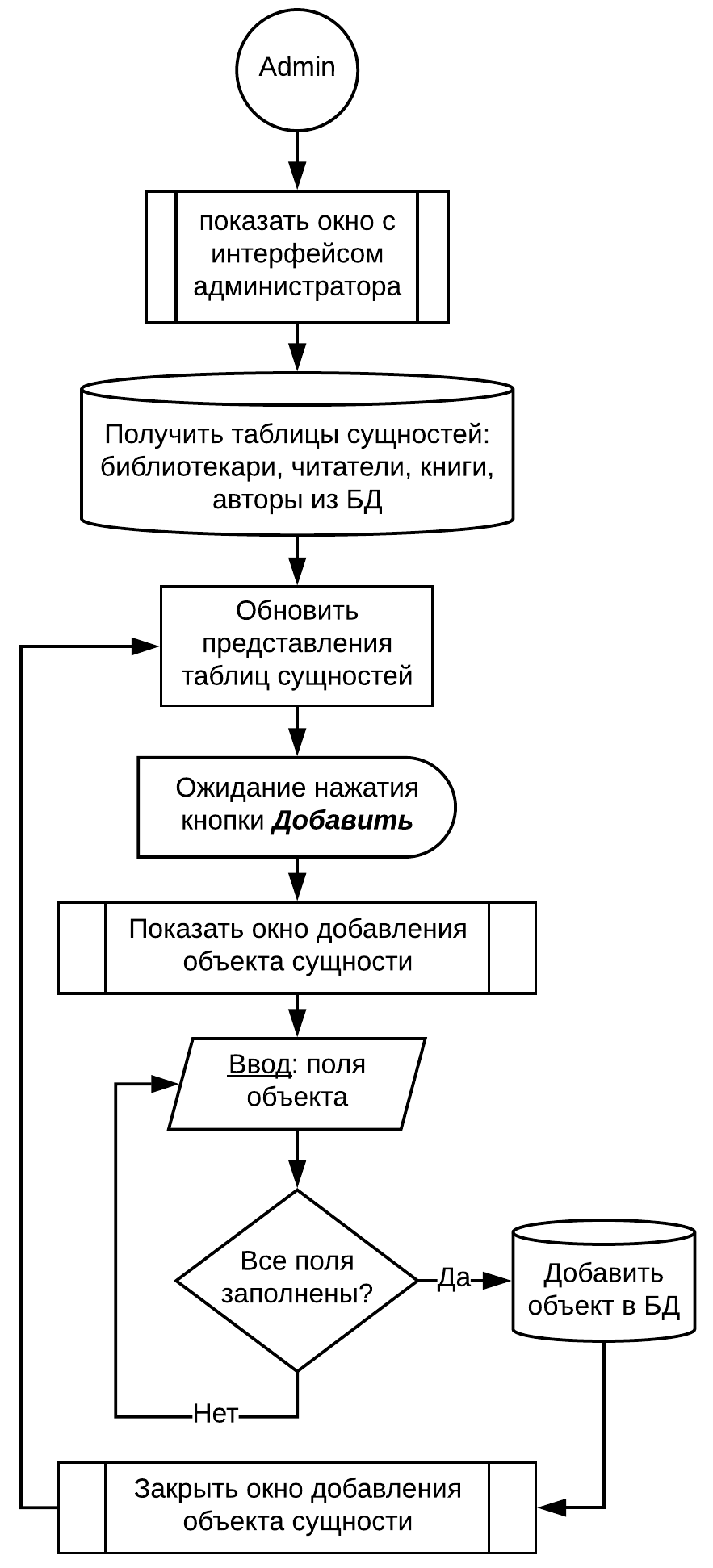


Рисунок 2. Блок схема логики администратора

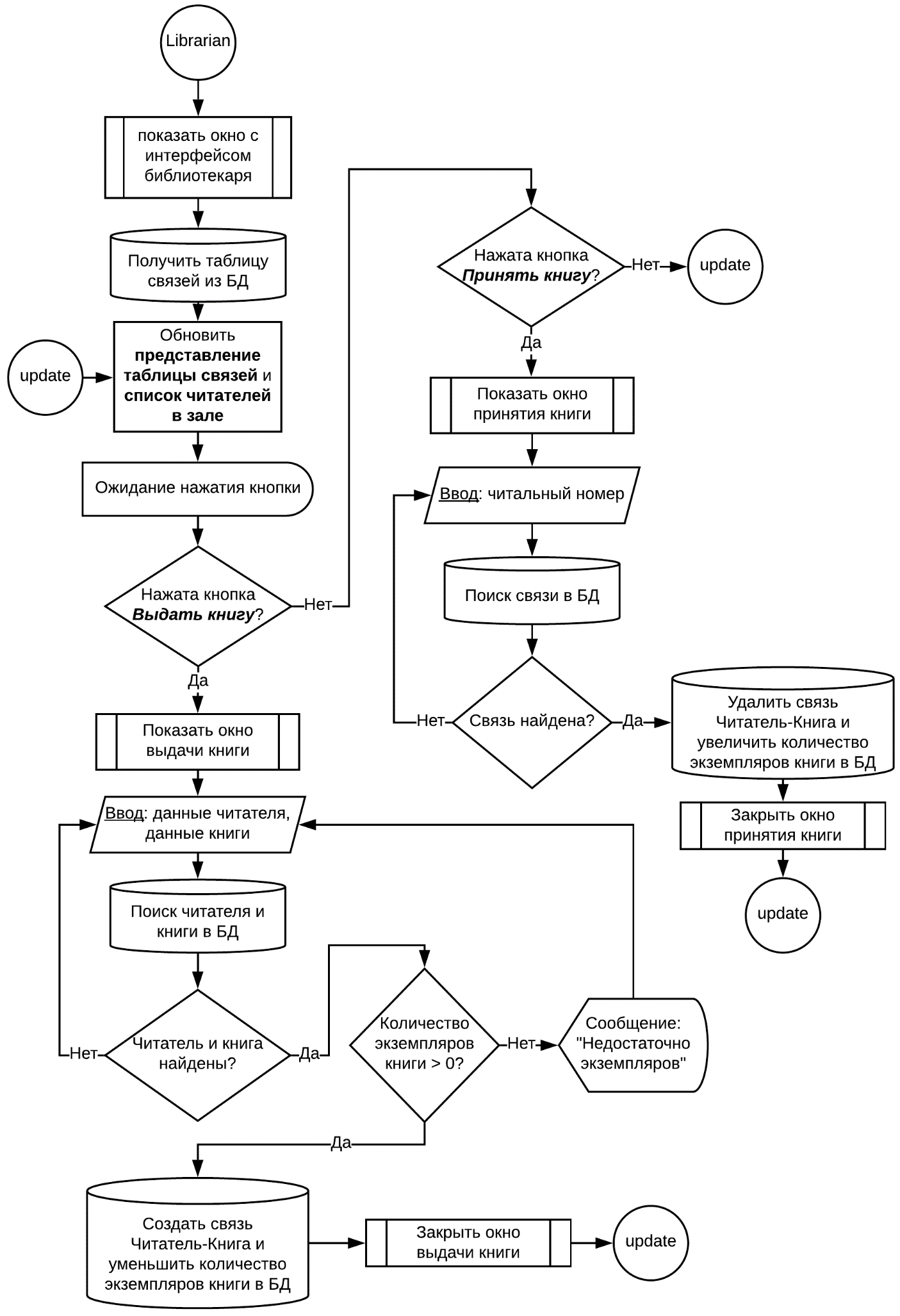


Рисунок 3. Блок схема логики библиотекаря

## Используемые методы

Программа использует следующие методы из библиотеки JavaFX:

* FXMLLoader.load – загружает разметку сцены из fxml файла
* setTitle – устанавливает заголовок окна
* getWindow – получает окно
* getScene – получает сцену окна
* setScene – устанавливает сцену в окно
* setResizable – устанавливает возможность скрыть окно
* show – показывает окно
* hide – скрывает окно
* setContentText – устанавливает содержимое окна с сообщением
* getParent – получить родителя элемента
* getText – получает текст элемента
* setText – устанавливает текст элемента
* getId – получает идентификатор элемента
* setOnAction – устанавливает элементу слушатель действий
* setOnKeyReleased – устанавливает слушатель клавиатуры
* selectedItemProperty – устанавливает слушатель выбора элемента списка
* setRowFactory – устанавливает элементы меню для строк таблицы
* setCellValueFactory – устанавливает соответствия столбцов и значений
* getItems – получить элементы списка или меню
* clear – очищает список элементов

Методы из стандартной библиотеки Java:

* getResource – получает URL ресурса по его имени
* getClass – получить класс объекта
* isEmpty – проверить пустоту строки
* forName – создает объект по имени класса
* exit – завершить программу
* getConnection – получает экземпляр подключения к базе данных
* prepareStatement – подготавливает запрос к БД
* setString, setInt – устанавливают значение в подготовленный запрос
* executeQuery – выполняет запрос

## Структура программы с описанием функций составных частей и связи между ними

Архитектура программы реализована по принципу MVC. И имеет пакетное деление следующим образом:

* models – содержит классы-модели с описанием полей и методов сущностей из базы данных, а также методы для доступа к базе данных;
* views – содержит разметку представлений в формате FXML;
* controllers – содержит классы-контроллеры реализующие отклик на действия пользователя в представлениях, в этих классах содержится основная логика программы;
* resources – содержит файлы-ресурсы графического или иного интерфейса, например изображения.

Содержимое models:

* DataBaseEntity – класс, является родителем классов-сущностей БД
* Librarian – класс, описывает сущность *librarians* из БД
* Reader – класс, описывает сущность *readers* из БД
* Book – класс, описывает сущность *books* из БД
* Author – класс, описывает сущность *authors* из БД
* Relationship – класс, описывает сущность *relationships* из БД
* Configuration – класс со статическими полями, для подключения к БД
* Constants – класс со статическими полями, для хранения констант
* DataBase – класс, реализует методы для осуществления запросов к БД

Содержимое views:

* login.fxml – разметка окна входа в программу
* admin.fxml – разметка основного окна администратора
* add\_librarian\_dialog.fxml – разметка окна добавления библиотекаря
* add\_reader\_dialog.fxml – разметка окна добавления читателя
* add\_book\_dialog.fxml – разметка окна добавления книги
* add\_author\_dialog.fxml – разметка окна добавления автора
* give\_book\_dialog.fxml – разметка окна выдачи книги
* accept\_book\_dialog.fxml – разметка окна принятия книги
* librarian.fxml – разметка основного окна библиотекаря

Содержимое controllers:

* LoginController – класс-контроллер окна входа в программу
* AdminController – класс-контроллер для основного окна администратора
* AdminAddController – класс-контроллер для окна добавления сущности
* LibrarianController – класс-контроллер для основного окна библиотекаря
* LibrarianActionController – класс-контроллер действий библиотекаря

Запуск окна для входа выполняется классом Main.

## Связи программы с другими программами

Программа «ReadingRoom.ехе» в ходе своей работы производит подключение по протоколу TCP/IP к серверу базы данных MySQL. Поля с деталями подключения описаны в классе Configuration (Приложение 1. Файл Configuration.java).

Структура таблиц базы данных находится в Приложении 4.

# Используемые технические средства

В состав используемых технических средств входит: IBM PC совместимый с процессором Intel Pentium CPU G2020 @ 2.90GHz и выше, ОЗУ не менее 150Мбайт, наличие свободного места на жестком диске 200Мбайт.

# Вызов и загрузка

Запуск программы производится путем запуска исполняемого файла «ReadingRoom.ехе». Дополнительная информация изложена в Руководстве пользователя.

# Входные данные

Программа в ходе своей работы использует вводимые пользователем числовые и строковые значения.

Для входа в систему:

* Тип пользователя
* Логин
* Пароль

Поля сущности **библиотекарь**:

* ФИО
* Логин
* Пароль

Поля сущности **читатель**:

* ФИО

Поля сущности **книга**:

* Название
* ISBN
* Год издания
* Количество
* ФИО Автора

Поля сущности **Автор**:

* ФИО

Для выдачи книги:

* № Читательского билета
* Название книги
* ФИО Автора книги

Для принятия книги:

* Читальный номер

# Выходные данные

Характер выходных данных зависит от контекста выполняемых действий. Программа выводит таблицы и списки, которые отображают состояние базы данных. В ходе поиска экземпляров определенных сущностей происходит визуализация поиска – вывод промежуточных результатов. После ввода данных программы может отобразить сообщение об ошибке или неуспешном поиске. Список всех сообщений приведен в Руководстве пользователя.

# Приложение 1

**Модели**

## Файл Author.java

package ru.readingroom.system.models;

public class Author extends DataBaseEntity {

private String name;

public Author(int id, String name) {

super(id);

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return getName() + " (№" + getId() + ")";

}

}

## Файл Book.java

package ru.readingroom.system.models;

public class Book extends DataBaseEntity {

private int quantity;

private String name, author, year, isbn;

public Book(int id, String name, String author, String year, String isbn,

int quantity) {

super(id);

this.name = name;

this.author = author;

this.year = year;

this.isbn = isbn;

this.quantity = quantity;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getYear() {

return year;

}

public void setYear(String year) {

this.year = year;

}

public String getIsbn() {

return isbn;

}

public void setIsbn(String isbn) {

this.isbn = isbn;

}

@Override

public String toString() {

return getName() + ", " + getAuthor() + " (" + getYear() + ")";

}

}

## Файл Reader.java

package ru.readingroom.system.models;

public class Reader extends DataBaseEntity {

private String name;

public Reader(int id, String name) {

super(id);

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "[№ " + getId() + "]: " + getName();

}

}

## Файл Librarian.java

package ru.readingroom.system.models;

public class Librarian extends DataBaseEntity {

private String name, login, password;

public Librarian(int id, String name, String login, String password) {

super(id);

this.name = name;

this.login = login;

this.password = password;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLogin() {

return login;

}

public void setLogin(String login) {

this.login = login;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

## Файл Relationship.java

package ru.readingroom.system.models;

public class Relationship extends DataBaseEntity {

private String reader, book, time;

public Relationship(int id, String reader, String book, String time) {

super(id);

this.reader = reader;

this.book = book;

this.time = time;

}

public String getReader() {

return reader;

}

public void setReader(String reader) {

this.reader = reader;

}

public String getBook() {

return book;

}

public void setBook(String book) {

this.book = book;

}

public String getTime() {

return time;

}

public void setTime(String time) {

this.time = time;

}

}

## Файл Configuration.java

package ru.readingroom.system.models;

public class Configuration {

protected final static String DB\_HOST = "localhost";

protected final static String DB\_PORT = "3306";

protected final static String DB\_USER = "root";

protected final static String DB\_PASSWORD = "";

protected final static String DB\_NAME = "reading\_room";

}

## Файл Constants.java

package ru.readingroom.system.models;

public class Constants {

public static final int USER\_ADMIN = 1;

public static final int USER\_LIBRARIAN = 2;

public static final String ADMINS\_TABLE = "admins";

public static final String LIBRARIAN\_TABLE = "librarians";

public static final String READERS\_TABLE = "readers";

public static final String BOOKS\_TABLE = "books";

public static final String AUTHORS\_TABLE = "authors";

public static final String RELATIONSHIPS\_TABLE = "relationships";

}

## Файл DataBaseEntity.java

package ru.readingroom.system.models;

public class DataBaseEntity {

private int id;

public DataBaseEntity(int id) {

this.id = id;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

}

## Файл DataBase.java

package ru.readingroom.system.models;

import java.sql.\*;

public class DataBase extends Configuration {

public static Connection connection;

public static boolean connect() throws ClassNotFoundException, SQLException {

String connectionStr = "jdbc:mysql://" + DB\_HOST + ":" + DB\_PORT + "/" + DB\_NAME + "?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

Class.forName("com.mysql.cj.jdbc.Driver");

connection = DriverManager.getConnection(connectionStr, DB\_USER, DB\_PASSWORD);

return true;

}

public static boolean checkLogin(String login, String password, int userType) {

String select = "SELECT COUNT(\*) AS count FROM "

+ (userType == Constants.USER\_ADMIN ? Constants.ADMINS\_TABLE : Constants.LIBRARIAN\_TABLE)

+ " WHERE login=? AND password=? LIMIT 1";

try {

PreparedStatement stmt = connection.prepareStatement(select);

stmt.setString(1, login);

stmt.setString(2, password);

ResultSet rs = stmt.executeQuery();

rs.next();

return rs.getInt("count") == 1;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

private static ResultSet getAllFromTable(String table) {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT \* FROM " + table);

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static ResultSet getLibrariansList() {

return getAllFromTable(Constants.LIBRARIAN\_TABLE);

}

public static ResultSet getReadersList() {

return getAllFromTable(Constants.READERS\_TABLE);

}

public static ResultSet getRelationshipsList() {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT id, timestamp, " +

"CONCAT('[№ ', reader\_id, '] ', (SELECT name FROM " + Constants.READERS\_TABLE + " WHERE id="+ Constants.RELATIONSHIPS\_TABLE +".reader\_id)) AS reader, " +

"(SELECT name FROM " + Constants.BOOKS\_TABLE + " WHERE id="+ Constants.RELATIONSHIPS\_TABLE +".book\_id) AS book" +

" FROM " + Constants.RELATIONSHIPS\_TABLE);

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static ResultSet getBooksList() {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT id, name, (SELECT name FROM " + Constants.AUTHORS\_TABLE + " WHERE id=author\_id) AS author, year, isbn, quantity FROM " + Constants.BOOKS\_TABLE);

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static ResultSet getAuthorsList() {

return getAllFromTable(Constants.AUTHORS\_TABLE);

}

public static void insertLibrarian(String name, String login, String password) {

try {

PreparedStatement stmt = connection.prepareStatement("INSERT INTO " + Constants.LIBRARIAN\_TABLE + "(name, login, password) VALUES(?, ?, ?)");

stmt.setString(1, name);

stmt.setString(2, login);

stmt.setString(3, password);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void insertReader(String name) {

try {

PreparedStatement stmt = connection.prepareStatement("INSERT INTO " + Constants.READERS\_TABLE + "(name) VALUES(?)");

stmt.setString(1, name);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

private static ResultSet searchRows(String tableName, String columnName, String searchValue) {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT \* FROM " + tableName + " WHERE "+ columnName + " LIKE **\"**%"+ searchValue +"%**\"**");

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static ResultSet searchAuthors(String searchName) {

return searchRows(Constants.AUTHORS\_TABLE, "name", searchName);

}

public static ResultSet searchReaders(int searchId) {

return searchRows(Constants.READERS\_TABLE, "id", String.valueOf(searchId));

}

public static ResultSet searchBooks(String bookName, String authorName) {

try {

String nameSearch = bookName.isEmpty() ? "" : "name LIKE **\"**%"+ bookName +"%**\"**";

String authorSearch = authorName.isEmpty() ? "" : "(SELECT name FROM " + Constants.AUTHORS\_TABLE + " WHERE id=author\_id) LIKE **\"**%"+ authorName +"%**\"**";

String div = (!bookName.isEmpty() && !authorName.isEmpty() ? " AND " : "");

PreparedStatement stmt = connection.prepareStatement("SELECT id, name, " +

"(SELECT name FROM " + Constants.AUTHORS\_TABLE + " WHERE id=author\_id) AS author, year, isbn, quantity " +

"FROM " + Constants.BOOKS\_TABLE + " WHERE " + nameSearch + div + authorSearch);

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static void insertBook(String name, String isbn, int authorId, String year, int quantity) {

try {

PreparedStatement stmt = connection.prepareStatement("INSERT INTO " + Constants.BOOKS\_TABLE + "(name, author\_id, year, isbn, quantity) VALUES(?, ?, ?, ?, ?)");

stmt.setString(1, name);

stmt.setInt(2, authorId);

stmt.setString(3, year);

stmt.setString(4, isbn);

stmt.setInt(5, quantity);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void insertAuthor(String name) {

try {

PreparedStatement stmt = connection.prepareStatement("INSERT INTO " + Constants.AUTHORS\_TABLE + "(name) VALUES(?)");

stmt.setString(1, name);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void deleteRow(int rowId, String tableName) {

try {

PreparedStatement stmt = connection.prepareStatement("DELETE FROM " + tableName + " WHERE id=?");

stmt.setInt(1, rowId);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void deleteBooksByAuthorId(int authorId) {

try {

PreparedStatement stmt = connection.prepareStatement("DELETE FROM " + Constants.BOOKS\_TABLE + " WHERE author\_id=?");

stmt.setInt(1, authorId);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void bookQuantityPlus(int id, int d) {

try {

PreparedStatement stmt = connection.prepareStatement("UPDATE " + Constants.BOOKS\_TABLE + " SET quantity=quantity+? WHERE id=?");

stmt.setInt(1, d);

stmt.setInt(2, id);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void insertRelationship(int readerId, int bookId, long timestamp) {

try {

PreparedStatement stmt = connection.prepareStatement("INSERT INTO " + Constants.RELATIONSHIPS\_TABLE + "(reader\_id, book\_id, timestamp) VALUES(?, ?, ?)");

stmt.setInt(1, readerId);

stmt.setInt(2, bookId);

stmt.setLong(3, timestamp);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void deleteRelationship(int id) {

try {

PreparedStatement stmt = connection.prepareStatement("DELETE FROM " + Constants.RELATIONSHIPS\_TABLE + " WHERE id=?");

stmt.setInt(1, id);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static int getBookIdByRelationship(int relationshipId) {

String select = "SELECT book\_id FROM " + Constants.RELATIONSHIPS\_TABLE + " WHERE id=?";

try {

PreparedStatement stmt = connection.prepareStatement(select);

stmt.setInt(1, relationshipId);

ResultSet rs = stmt.executeQuery();

rs.next();

return rs.getInt("book\_id");

} catch (SQLException e) {

e.printStackTrace();

return 0;

}

}

public static ResultSet getActiveReaders() {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT id, name " +

"FROM " + Constants.READERS\_TABLE + " WHERE (SELECT COUNT(\*) FROM " + Constants.RELATIONSHIPS\_TABLE + " WHERE reader\_id=" + Constants.READERS\_TABLE + ".id) > 0");

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

}

# Приложение 2

**Представления**

## Файл login.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.PasswordField**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.image.Image**?>**

**<?import** javafx.scene.image.ImageView**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane**

fx:id="rootPane"

prefHeight="400.0"

prefWidth="600.0"

xmlns="http://javafx.com/javafx/8.0.171"

xmlns:fx="http://javafx.com/fxml/1"

fx:controller="ru.readingroom.system.controllers.LoginController"**>**

**<children>**

**<AnchorPane**

fx:id="selectLoginTypePane"

layoutX="-1.0"

prefHeight="410.0"

prefWidth="611.0"

style="-fx-background-color: #3279cd;"**>**

**<children>**

**<Label**

layoutX="161.0"

layoutY="42.0"

text="Войти как"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="59.0" **/>**

**</font>**

**</Label>**

**<Button**

fx:id="enterAsAdminButton"

cancelButton="true"

layoutX="233.0"

layoutY="167.0"

mnemonicParsing="false"

prefHeight="58.0"

prefWidth="205.0"

style="-fx-background-radius: 30;

-fx-background-color: #541e00;"

text="Администратор"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="20.0" **/>**

**</font></Button>**

**<ImageView**

fitHeight="58.0"

fitWidth="58.0"

layoutX="174.0"

layoutY="167.0"

pickOnBounds="true"

preserveRatio="true"**>**

**<image>**

**<Image** url="@../resources/admin.png" **/>**

**</image>**

**</ImageView>**

**<ImageView**

fitHeight="58.0"

fitWidth="69.0"

layoutX="161.0"

layoutY="239.0"

pickOnBounds="true"

preserveRatio="true"**>**

**<image>**

**<Image** url="@../resources/librarian.png" **/>**

**</image>**

**</ImageView>**

**<Button**

fx:id="enterAsLibrarianButton"

cancelButton="true"

layoutX="233.0"

layoutY="239.0"

mnemonicParsing="false"

prefHeight="58.0"

prefWidth="205.0"

style="-fx-background-radius: 30;

-fx-background-color: #00540c;"

text="Библиотекарь"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="20.0" **/>**

**</font>**

**</Button>**

**</children></AnchorPane>**

**<AnchorPane**

fx:id="loginPane"

focusTraversable="true"

layoutX="-1.0"

prefHeight="410.0"

prefWidth="611.0"

style="-fx-background-color: #3279cd;"

visible="false"**>**

**<children>**

**<Label**

layoutX="231.0"

layoutY="40.0"

text="Вход"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="59.0" **/>**

**</font>**

**</Label>**

**<TextField**

fx:id="loginInput"

layoutX="188.0"

layoutY="154.0"

prefHeight="30.0"

prefWidth="230.0"

promptText="Логин"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</TextField>**

**<PasswordField**

fx:id="passwordInput"

layoutX="188.0"

layoutY="198.0"

prefHeight="30.0"

prefWidth="230.0"

promptText="Пароль"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</PasswordField>**

**<Button**

fx:id="loginButton"

layoutX="188.0"

layoutY="258.0"

mnemonicParsing="false"

prefHeight="13.0"

prefWidth="230.0"

text="Войти"**>**

**<font>**

**<Font**

name="System Bold"

size="15.0" **/>**

**</font>**

**</Button>**

**<Button**

fx:id="backButton"

layoutX="14.0"

layoutY="15.0"

mnemonicParsing="false"

text="&lt; Назад"**>**

**<font>**

**<Font**

name="System Bold"

size="11.0" **/>**

**</font>**

**</Button>**

**<Label**

fx:id="userTypeLabel"

alignment="CENTER"

layoutX="188.0"

layoutY="113.0"

prefHeight="24.0"

prefWidth="230.0"

textAlignment="CENTER"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Italic"

size="18.0" **/>**

**</font></Label>**

**</children>**

**</AnchorPane>**

**</children>**

**</AnchorPane>**

## Файл admin.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.Tab**?>**

**<?import** javafx.scene.control.TabPane**?>**

**<?import** javafx.scene.control.TableColumn**?>**

**<?import** javafx.scene.control.TableView**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane**

prefHeight="627.0"

prefWidth="956.0"

xmlns="http://javafx.com/javafx/8.0.171"

xmlns:fx="http://javafx.com/fxml/1"

fx:controller="ru.readingroom.system.controllers.AdminController"**>**

**<children>**

**<TabPane**

fx:id="adminTabPane"

layoutX="-3.0"

layoutY="-2.0"

prefHeight="627.0"

prefWidth="956.0"

tabClosingPolicy="UNAVAILABLE"**>**

**<tabs>**

**<Tab**

fx:id="librariansTab"

text="Библиотекари"**>**

**<content>**

**<AnchorPane**

minHeight="0.0"

minWidth="0.0"

prefHeight="571.0"

prefWidth="929.0"

style="-fx-background-color: #d3e7ff;"**>**

**<children>**

**<TableView**

fx:id="librariansTable"

layoutX="19.0"

layoutY="19.0"

prefHeight="498.0"

prefWidth="880.0"**>**

**<placeholder>**

**<Label** text="Нет библиотекарей"**>**

**<font>**

**<Font** size="20.0" **/>**

**</font>**

**</Label>**

**</placeholder>**

**<columns>**

**<TableColumn**

fx:id="librariansIdColumn"

prefWidth="75.0" text="№" **/>**

**<TableColumn**

fx:id="librariansNameColumn"

prefWidth="295.0"

text="ФИО" **/>**

**<TableColumn**

fx:id="librariansLoginColumn"

minWidth="0.0"

prefWidth="146.0"

text="Логин" **/>**

**<TableColumn**

fx:id="librariansPasswordColumn"

prefWidth="162.0"

text="Пароль" **/>**

**</columns>**

**</TableView>**

**<Button**

fx:id="addLibrarianButton"

layoutX="723.0"

layoutY="531.0"

mnemonicParsing="false"

prefHeight="22.0"

prefWidth="176.0"

style="-fx-background-color: #5cb46f;"

text="+ Добавить"**>**

**<font>**

**<Font**

name="System Bold"

size="17.0" **/>**

**</font>**

**</Button>**

**<Button**

layoutX="19.0"

layoutY="531.0"

mnemonicParsing="false"

onAction="#adminExit"

style="-fx-background-color: #d73638;"

text="Выйти"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="13.0" **/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

**</content>**

**</Tab>**

**<Tab**

fx:id="readersTab"

text="Читатели"**>**

**<content>**

**<AnchorPane**

minHeight="0.0"

minWidth="0.0"

prefHeight="674.0"

prefWidth="1036.0"

style="-fx-min-width: 100;

-fx-background-color: #d3e7ff;"**>**

**<children>**

**<TableView**

fx:id="readersTable"

layoutX="19.0"

layoutY="19.0"

prefHeight="498.0"

prefWidth="880.0"**>**

**<placeholder>**

**<Label** text="Нет читателей"**>**

**<font>**

**<Font** size="20.0" **/>**

**</font>**

**</Label>**

**</placeholder>**

**<columns>**

**<TableColumn**

fx:id="readersIdColumn"

prefWidth="74.99993896484375"

text="№" **/>**

**<TableColumn**

fx:id="readersNameColumn"

minWidth="0.0"

prefWidth="390.0000305175781"

text="ФИО" **/>**

**</columns>**

**</TableView>**

**<Button**

fx:id="addReaderButton"

layoutX="723.0"

layoutY="531.0"

mnemonicParsing="false"

prefHeight="22.0"

prefWidth="176.0"

style="-fx-background-color: #5cb46f;"

text="+ Добавить"**>**

**<font>**

**<Font**

name="System Bold"

size="17.0" **/>**

**</font>**

**</Button>**

**<Button**

layoutX="19.0"

layoutY="531.0"

mnemonicParsing="false"

onAction="#adminExit"

style="-fx-background-color: #d73638;"

text="Выйти"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="13.0" **/>**

**</font>**

**</Button>**

**</children></AnchorPane>**

**</content>**

**</Tab>**

**<Tab** fx:id="booksTab" text="Книги"**>**

**<content>**

**<AnchorPane**

minHeight="0.0"

minWidth="0.0"

prefHeight="180.0"

prefWidth="200.0"

style="-fx-background-color: #d3e7ff;"**>**

**<children>**

**<TableView**

fx:id="booksTable"

layoutX="19.0"

layoutY="19.0"

prefHeight="498.0"

prefWidth="880.0"**>**

**<placeholder>**

**<Label** text="Нет книг"**>**

**<font>**

**<Font** size="20.0" **/>**

**</font>**

**</Label>**

**</placeholder>**

**<columns>**

**<TableColumn**

fx:id="booksIdColumn"

prefWidth="75.0"

text="№" **/>**

**<TableColumn**

fx:id="booksNameColumn"

prefWidth="208.0"

text="Название" **/>**

**<TableColumn**

fx:id="booksAuthorColumn"

minWidth="0.0"

prefWidth="252.0"

text="Автор" **/>**

**<TableColumn**

fx:id="booksYearColumn"

prefWidth="91.0"

text="Год издания" **/>**

**<TableColumn**

fx:id="booksIsbnColumn"

prefWidth="151.0"

text="ISBN" **/>**

**<TableColumn**

fx:id="booksQuantityColumn"

prefWidth="101.0"

text="Количество" **/>**

**</columns>**

**</TableView>**

**<Button**

fx:id="addBookButton"

layoutX="723.0"

layoutY="531.0"

mnemonicParsing="false"

prefHeight="22.0"

prefWidth="176.0"

style="-fx-background-color: #5cb46f;"

text="+ Добавить"**>**

**<font>**

**<Font**

name="System Bold"

size="17.0" **/>**

**</font>**

**</Button>**

**<Button**

layoutX="19.0"

layoutY="531.0"

mnemonicParsing="false"

onAction="#adminExit"

style="-fx-background-color: #d73638;"

text="Выйти"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="13.0" **/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

**</content>**

**</Tab>**

**<Tab**

fx:id="authorsTab"

text="Авторы"**>**

**<content>**

**<AnchorPane**

minHeight="0.0"

minWidth="0.0"

prefHeight="180.0"

prefWidth="200.0"

style="-fx-background-color: #d3e7ff;"**>**

**<children>**

**<TableView**

fx:id="authorsTable"

layoutX="19.0"

layoutY="19.0"

prefHeight="498.0"

prefWidth="880.0"**>**

**<placeholder>**

**<Label** text="Нет авторов"**>**

**<font>**

**<Font** size="20.0" **/>**

**</font>**

**</Label>**

**</placeholder>**

**<columns>**

**<TableColumn**

fx:id="authorsIdColumn"

prefWidth="75.0"

text="№" **/>**

**<TableColumn**

fx:id="authorsNameColumn"

prefWidth="290.0"

text="ФИО" **/>**

**</columns>**

**</TableView>**

**<Button**

layoutX="19.0"

layoutY="531.0"

mnemonicParsing="false"

onAction="#adminExit"

style="-fx-background-color: #d73638;"

text="Выйти"

textFill="WHITE"**>**

**<font>**

**<Font**

name="System Bold"

size="13.0" **/>**

**</font>**

**</Button>**

**<Button**

fx:id="addAuthorButton"

layoutX="723.0"

layoutY="531.0"

mnemonicParsing="false"

prefHeight="22.0"

prefWidth="176.0"

style="-fx-background-color: #5cb46f;"

text="+ Добавить"**>**

**<font>**

**<Font**

name="System Bold"

size="17.0" **/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

**</content>**

**</Tab>**

**</tabs>**

**</TabPane>**

**</children>**

**</AnchorPane>**

## Файл add\_librarian\_dialog.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane**

focusTraversable="true"

prefHeight="282.0"

prefWidth="354.0"

xmlns="http://javafx.com/javafx/8.0.171"

xmlns:fx="http://javafx.com/fxml/1"

fx:controller="ru.readingroom.system.controllers.AdminAddController"**>**

**<children>**

**<Label**

layoutX="10.0"

layoutY="25.0"

text="Добавление библиотекаря"**>**

**<font>**

**<Font** size="27.0" **/>**

**</font>**

**</Label>**

**<TextField**

fx:id="librarianNameInput"

focusTraversable="false"

layoutX="49.0"

layoutY="87.0"

prefHeight="25.0"

prefWidth="265.0"

promptText="Фамилия Имя Отчество"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</TextField>**

**<TextField**

fx:id="librarianLoginInput"

focusTraversable="false"

layoutX="49.0"

layoutY="135.0"

prefHeight="25.0"

prefWidth="265.0"

promptText="Логин"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</TextField>**

**<TextField**

fx:id="librarianPasswordInput"

focusTraversable="false"

layoutX="49.0"

layoutY="182.0"

prefHeight="25.0"

prefWidth="265.0"

promptText="Пароль"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</TextField>**

**<Button**

fx:id="librarianAdd"

focusTraversable="false"

layoutX="228.0"

layoutY="226.0"

mnemonicParsing="false"

text="Добавить"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

## Файл add\_reader\_dialog.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane**

focusTraversable="true"

prefHeight="188.0"

prefWidth="354.0"

xmlns="http://javafx.com/javafx/8.0.171"

xmlns:fx="http://javafx.com/fxml/1"

fx:controller="ru.readingroom.system.controllers.AdminAddController"**>**

**<children>**

**<Label**

layoutX="47.0"

layoutY="26.0"

text="Добавление читателя"**>**

**<font>**

**<Font** size="27.0" **/>**

**</font>**

**</Label>**

**<TextField**

fx:id="readerNameInput"

focusTraversable="false"

layoutX="49.0"

layoutY="87.0"

prefHeight="25.0"

prefWidth="265.0" promptText="Фамилия Имя Отчество"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</TextField>**

**<Button**

fx:id="readerAdd"

focusTraversable="false"

layoutX="230.0"

layoutY="141.0"

mnemonicParsing="false"

text="Добавить"**>**

**<font>**

**<Font** size="15.0" **/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

## Файл add\_book\_dialog.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.ListView**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane** xmlns:fx="http://javafx.com/fxml/1"

focusTraversable="true"

prefHeight="431.0"

prefWidth="354.0"

xmlns="http://javafx.com/javafx/8.0.171"

fx:controller="ru.readingroom.system.controllers.AdminAddController"**>**

**<children>**

**<Label** layoutX="66.0"

layoutY="26.0"

text="Добавление книги"**>**

**<font>**

**<Font** size="27.0"**/>**

**</font>**

**</Label>**

**<TextField** fx:id="bookNameInput"

focusTraversable="false"

layoutX="28.0"

layoutY="84.0"

prefHeight="31.0"

prefWidth="307.0"

promptText="Название"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="bookAdd"

focusTraversable="false"

layoutX="249.0"

layoutY="387.0"

mnemonicParsing="false"

text="Добавить"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Button>**

**<AnchorPane** layoutX="28.0"

layoutY="211.0"

prefHeight="162.0"

prefWidth="307.0"

style="-fx-border-style: solid;

-fx-border-width: 1px;

-fx-border-color: black;

-fx-background-color: #eff0ff;"**>**

**<children>**

**<Label** layoutX="103.0"

layoutY="8.0"

prefHeight="21.0"

prefWidth="101.0"

text="Поиск автора"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Label>**

**<TextField** fx:id="bookAuthorNameSearchInput"

focusTraversable="false"

layoutX="14.0"

layoutY="33.0"

prefHeight="29.0"

prefWidth="206.0"

promptText="Фамилия Имя Отчество"**>**

**<font>**

**<Font** size="13.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="bookAuthorNameClearButton"

focusTraversable="false"

layoutX="225.0"

layoutY="33.0"

mnemonicParsing="false"

prefHeight="29.0"

prefWidth="68.0"

text="Очистить"**/>**

**<ListView** fx:id="bookAuthorsList"

focusTraversable="false"

layoutX="14.0"

layoutY="67.0"

prefHeight="82.0"

prefWidth="279.0"**>**

**<placeholder>**

**<Label** fx:id="bookAuthorsListPlaceholder"

text="Список авторов"**>**

**<font>**

**<Font** size="14.0"**/>**

**</font>**

**</Label>**

**</placeholder>**

**</ListView>**

**</children>**

**</AnchorPane>**

**<TextField** fx:id="bookIsbnInput"

focusTraversable="false"

layoutX="28.0"

layoutY="123.0"

prefHeight="31.0"

prefWidth="307.0"

promptText="ISBN"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</TextField>**

**<TextField** fx:id="bookYearInput"

focusTraversable="false"

layoutX="28.0"

layoutY="168.0"

prefHeight="31.0"

prefWidth="147.0"

promptText="Год издания"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</TextField>**

**<TextField** fx:id="bookQuantityInput"

focusTraversable="false"

layoutX="182.0"

layoutY="168.0"

prefHeight="31.0"

prefWidth="153.0"

promptText="Количество"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</TextField>**

**</children>**

**</AnchorPane>**

## Файл add\_author\_dialog.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane** xmlns:fx="http://javafx.com/fxml/1"

focusTraversable="true"

prefHeight="188.0"

prefWidth="354.0"

xmlns="http://javafx.com/javafx/8.0.171"

fx:controller="ru.readingroom.system.controllers.AdminAddController"**>**

**<children>**

**<Label** layoutX="55.0"

layoutY="26.0"

text="Добавление автора"**>**

**<font>**

**<Font** size="27.0"**/>**

**</font>**

**</Label>**

**<TextField** fx:id="readerNameInput"

focusTraversable="false"

layoutX="49.0"

layoutY="87.0"

prefHeight="25.0"

prefWidth="265.0"

promptText="Фамилия Имя Отчество"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="authorAdd"

focusTraversable="false"

layoutX="230.0"

layoutY="141.0"

mnemonicParsing="false"

text="Добавить"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

## Файл librarian.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.ListView**?>**

**<?import** javafx.scene.control.TableColumn**?>**

**<?import** javafx.scene.control.TableView**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane** xmlns:fx="http://javafx.com/fxml/1"

focusTraversable="true"

prefHeight="619.0"

prefWidth="914.0"

style="-fx-background-color: #d5ffd4;"

xmlns="http://javafx.com/javafx/8.0.171"

fx:controller="ru.readingroom.system.controllers.LibrarianController"**>**

**<children>**

**<TableView** fx:id="relationshipsTable"

focusTraversable="false"

layoutX="14.0"

layoutY="99.0"

prefHeight="500.0"

prefWidth="662.0"**>**

**<placeholder>**

**<Label** text="Нет читающих"**>**

**<font>**

**<Font** size="20.0"**/>**

**</font>**

**</Label>**

**</placeholder>**

**<columns>**

**<TableColumn** fx:id="relationshipsIdColumn"

prefWidth="110.0"

text="Читальный №"**/>**

**<TableColumn** fx:id="relationshipsReaderColumn"

minWidth="0.0"

prefWidth="201.0"

text="Читатель"**/>**

**<TableColumn** fx:id="relationshipsBookColumn"

prefWidth="217.0"

text="Книга"**/>**

**<TableColumn** fx:id="relationshipsTimeColumn"

prefWidth="132.0"

text="С момента выдачи"**/>**

**</columns>**

**</TableView>**

**<Label** fx:id="readersQuantity"

alignment="CENTER"

focusTraversable="false"

layoutX="686.0"

layoutY="99.0"

prefHeight="36.0"

prefWidth="211.0"

style="-fx-background-color: #59b255;"**>**

**<font>**

**<Font** size="19.0"**/>**

**</font>**

**</Label>**

**<Button** fx:id="giveBookButton"

focusTraversable="false"

layoutX="14.0"

layoutY="28.0"

mnemonicParsing="false"

prefHeight="48.0"

prefWidth="436.0"

text="Выдать книгу"**>**

**<font>**

**<Font** name="System Bold"

size="22.0"**/>**

**</font>**

**</Button>**

**<Button** fx:id="acceptBookButton"

focusTraversable="false"

layoutX="461.0"

layoutY="28.0"

mnemonicParsing="false"

prefHeight="48.0"

prefWidth="436.0"

text="Принять книгу"**>**

**<font>**

**<Font** name="System Bold"

size="22.0"**/>**

**</font>**

**</Button>**

**<Button** focusTraversable="false"

layoutX="686.0"

layoutY="563.0"

mnemonicParsing="false"

onAction="#librarianExit"

prefHeight="35.0"

prefWidth="211.0"

style="-fx-background-color: #d73638;"

text="Выйти"

textFill="WHITE"**>**

**<font>**

**<Font** name="System Bold"

size="15.0"**/>**

**</font>**

**</Button>**

**<ListView** fx:id="readersList"

layoutX="686.0"

layoutY="142.0"

prefHeight="412.0"

prefWidth="211.0"**/>**

**</children>**

**</AnchorPane>**

## Файл give\_book\_dialog.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.ListView**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane** xmlns:fx="http://javafx.com/fxml/1"

focusTraversable="true"

prefHeight="501.0"

prefWidth="365.0"

xmlns="http://javafx.com/javafx/8.0.171" fx:controller="ru.readingroom.system.controllers.LibrarianActionController"**>**

**<children>**

**<Label** layoutX="97.0"

layoutY="26.0"

text="Выдача книги"**>**

**<font>**

**<Font** size="27.0"**/>**

**</font>**

**</Label>**

**<Button** fx:id="bookGiveButton"

focusTraversable="false"

layoutX="268.0"

layoutY="422.0"

mnemonicParsing="false"

text="Выдать"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Button>**

**<AnchorPane** layoutX="29.0"

layoutY="227.0"

prefHeight="178.0"

prefWidth="307.0"

style="-fx-border-style: solid;

-fx-border-width: 1px;

-fx-border-color: black;

-fx-background-color: #6aa178;"**>**

**<children>**

**<Label** layoutX="111.0"

layoutY="8.0"

text="Поиск книги"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Label>**

**<TextField** fx:id="bookNameSearchInput"

focusTraversable="false"

layoutX="14.0"

layoutY="33.0"

prefHeight="29.0"

prefWidth="206.0"

promptText="Название"**>**

**<font>**

**<Font** size="13.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="bookNameClearButton"

focusTraversable="false"

layoutX="225.0"

layoutY="33.0"

mnemonicParsing="false"

prefHeight="29.0"

prefWidth="68.0"

text="Очистить"**/>**

**<ListView** fx:id="booksList"

focusTraversable="false"

layoutX="15.0"

layoutY="102.0"

prefHeight="63.0"

prefWidth="279.0"**>**

**<placeholder>**

**<Label** fx:id="booksListPlaceholder"

text="Список книг"**>**

**<font>**

**<Font** size="14.0"**/>**

**</font>**

**</Label>**

**</placeholder>**

**</ListView>**

**<TextField** fx:id="bookAuthorNameSearchInput"

focusTraversable="false"

layoutX="14.0"

layoutY="67.0"

prefHeight="29.0"

prefWidth="206.0"

promptText="Автор"**>**

**<font>**

**<Font** size="13.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="bookAuthorNameClearButton"

focusTraversable="false"

layoutX="225.0"

layoutY="67.0"

mnemonicParsing="false"

prefHeight="29.0"

prefWidth="68.0"

text="Очистить"**/>**

**</children>**

**</AnchorPane>**

**<AnchorPane** layoutX="29.0"

layoutY="80.0"

prefHeight="134.0"

prefWidth="307.0"

style="-fx-border-style: solid;

-fx-border-width: 1px;

-fx-border-color: black;

-fx-background-color: #6b85a1;"**>**

**<children>**

**<Label** layoutX="100.0"

layoutY="8.0"

text="Поиск читателя"

textFill="WHITE"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Label>**

**<TextField** fx:id="readerIdSearchInput"

focusTraversable="false"

layoutX="14.0"

layoutY="33.0"

prefHeight="29.0"

prefWidth="206.0"

promptText="№ Читательского билета"**>**

**<font>**

**<Font** size="13.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="readerIdClearButton"

focusTraversable="false"

layoutX="225.0"

layoutY="33.0"

mnemonicParsing="false"

prefHeight="29.0"

prefWidth="68.0"

text="Очистить"**/>**

**<ListView** fx:id="readersList"

focusTraversable="false"

layoutX="14.0"

layoutY="67.0"

prefHeight="57.0"

prefWidth="279.0"**>**

**<placeholder>**

**<Label** fx:id="readersListPlaceholder"

text="Список читателей"**>**

**<font>**

**<Font** size="14.0"**/>**

**</font>**

**</Label>**

**</placeholder>**

**</ListView>**

**</children>**

**</AnchorPane>**

**</children>**

**</AnchorPane>**

## Файл accept\_book\_dialog.fxml

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

**<?import** javafx.scene.control.Button**?>**

**<?import** javafx.scene.control.Label**?>**

**<?import** javafx.scene.control.TextField**?>**

**<?import** javafx.scene.layout.AnchorPane**?>**

**<?import** javafx.scene.text.Font**?>**

**<AnchorPane** xmlns:fx="http://javafx.com/fxml/1"

focusTraversable="true"

prefHeight="188.0"

prefWidth="354.0"

xmlns="http://javafx.com/javafx/8.0.172-ea"

fx:controller="ru.readingroom.system.controllers.LibrarianActionController"**>**

**<children>**

**<Label** layoutX="78.0"

layoutY="26.0"

text="Принятие книги"**>**

**<font>**

**<Font** size="27.0"**/>**

**</font>**

**</Label>**

**<TextField** fx:id="relationshipIdInput"

focusTraversable="false"

layoutX="49.0"

layoutY="87.0"

prefHeight="25.0"

prefWidth="265.0"

promptText="Читальный номер"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</TextField>**

**<Button** fx:id="bookAcceptButton"

focusTraversable="false"

layoutX="235.0"

layoutY="143.0"

mnemonicParsing="false"

text="Принять"**>**

**<font>**

**<Font** size="15.0"**/>**

**</font>**

**</Button>**

**</children>**

**</AnchorPane>**

# Приложение 3

**Контроллеры**

## Файл LoginController.java

**package** ru.readingroom.system.controllers;

**import** javafx.event.ActionEvent;

**import** javafx.event.EventHandler;

**import** javafx.fxml.FXML;

**import** javafx.scene.control.Alert;

**import** javafx.scene.control.Button;

**import** javafx.scene.control.Label;

**import** javafx.scene.control.TextField;

**import** javafx.scene.layout.AnchorPane;

**import** javafx.stage.Stage;

**import** ru.readingroom.system.App;

**import** ru.readingroom.system.Main;

**import** ru.readingroom.system.models.Constants;

**import** ru.readingroom.system.models.DataBase;

**public** **class** LoginController **implements** EventHandler<ActionEvent> {

@FXML

**private** Label userTypeLabel;

@FXML

**private** AnchorPane selectLoginTypePane, loginPane;

@FXML

**private** Button enterAsAdminButton, enterAsLibrarianButton;

@FXML

**private** Button backButton, loginButton;

@FXML

**private** TextField loginInput, passwordInput;

**public** **int** userType;

@FXML

**void** initialize() {

enterAsAdminButton.setOnAction(**this**);

enterAsLibrarianButton.setOnAction(**this**);

backButton.setOnAction(**this**);

loginButton.setOnAction(**this**);

}

**public** **void** showSelectLoginTypePane() {

loginPane.setVisible(**false**);

selectLoginTypePane.setVisible(**true**);

}

**public** **void** showLoginPane() {

**if** (userType == Constants.USER\_ADMIN)

userTypeLabel.setText("Администратор");

**else**

userTypeLabel.setText("Библиотекарь");

loginInput.setText("");

passwordInput.setText("");

selectLoginTypePane.setVisible(**false**);

loginPane.setVisible(**true**);

}

@Override

**public** **void** handle(ActionEvent event) {

Object src = event.getSource();

**if** (src.equals(enterAsAdminButton)) {

userType = Constants.USER\_ADMIN;

showLoginPane();

} **else** **if** (src.equals(enterAsLibrarianButton)) {

userType = Constants.USER\_LIBRARIAN;

showLoginPane();

} **else** **if** (src.equals(backButton)) {

showSelectLoginTypePane();

} **else** **if** (src.equals(loginButton)) {

**boolean** result = DataBase.checkLogin(loginInput.getText(), passwordInput.getText(), userType);

**if** (result) {

startApp();

}

**else** Main.showMsg("Не удалось войти", "Неправильный логин или пароль", Alert.AlertType.ERROR, **null**);

}

}

**private** **void** startApp() {

**if** (userType == Constants.USER\_ADMIN) {

App.xmlSrc = "views/admin.fxml";

App.title = "Система учета библиотеки | Администратор";

} **else** {

App.xmlSrc = "views/librarian.fxml";

App.title = "Система учета библиотеки | Библиотекарь";

}

**try** {

**new** App().start(**new** Stage());

} **catch** (Exception e) {

e.printStackTrace();

}

Main.stage.close();

}

}

## Файл AdminController.java

**package** ru.readingroom.system.controllers;

**import** javafx.beans.binding.Bindings;

**import** javafx.fxml.FXML;

**import** javafx.fxml.FXMLLoader;

**import** javafx.scene.Parent;

**import** javafx.scene.Scene;

**import** javafx.scene.control.\*;

**import** javafx.scene.control.cell.PropertyValueFactory;

**import** javafx.stage.Modality;

**import** javafx.stage.Stage;

**import** javafx.util.Callback;

**import** ru.readingroom.system.App;

**import** ru.readingroom.system.Main;

**import** ru.readingroom.system.models.\*;

**import** java.io.IOException;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**public** **class** AdminController {

@FXML

**private** TabPane adminTabPane;

@FXML

**private** Tab librariansTab;

@FXML

**private** TableView<DataBaseEntity> librariansTable;

@FXML

**private** TableColumn<Object, Object> librariansIdColumn;

@FXML

**private** TableColumn<Object, Object> librariansNameColumn;

@FXML

**private** TableColumn<Object, Object> librariansLoginColumn;

@FXML

**private** TableColumn<Object, Object> librariansPasswordColumn;

@FXML

**private** Tab readersTab;

@FXML

**private** TableView<DataBaseEntity> readersTable;

@FXML

**private** TableColumn<Object, Object> readersIdColumn;

@FXML

**private** TableColumn<Object, Object> readersNameColumn;

@FXML

**private** Tab booksTab;

@FXML

**private** TableView<DataBaseEntity> booksTable;

@FXML

**private** TableColumn<?, ?> booksIdColumn;

@FXML

**private** TableColumn<?, ?> booksNameColumn;

@FXML

**private** TableColumn<?, ?> booksAuthorColumn;

@FXML

**private** TableColumn<?, ?> booksYearColumn;

@FXML

**private** TableColumn<?, ?> booksIsbnColumn;

@FXML

**private** TableColumn<?, ?> booksQuantityColumn;

@FXML

**private** Tab authorsTab;

@FXML

**private** TableView<DataBaseEntity> authorsTable;

@FXML

**private** TableColumn<Object, Object> authorsIdColumn;

@FXML

**private** TableColumn<Object, Object> authorsNameColumn;

@FXML

**private** Button addLibrarianButton, addReaderButton, addBookButton, addAuthorButton;

**public** Stage addDialogStage;

**public** Tab currentTab;

@FXML

**void** initialize() {

adminTabPane.getSelectionModel().selectedItemProperty().addListener((ov, t, t1) -> {

currentTab = t1;

updateCurrentTable();

});

librariansTable.setRowFactory((tableViewCallback));

readersTable.setRowFactory((tableViewCallback));

booksTable.setRowFactory((tableViewCallback));

authorsTable.setRowFactory((tableViewCallback));

librariansIdColumn.setCellValueFactory(**new** PropertyValueFactory<>("id"));

librariansNameColumn.setCellValueFactory(**new** PropertyValueFactory<>("name"));

librariansLoginColumn.setCellValueFactory(**new** PropertyValueFactory<>("login"));

librariansPasswordColumn.setCellValueFactory(**new** PropertyValueFactory<>("password"));

readersIdColumn.setCellValueFactory(**new** PropertyValueFactory<>("id"));

readersNameColumn.setCellValueFactory(**new** PropertyValueFactory<>("name"));

booksIdColumn.setCellValueFactory(**new** PropertyValueFactory<>("id"));

booksNameColumn.setCellValueFactory(**new** PropertyValueFactory<>("name"));

booksAuthorColumn.setCellValueFactory(**new** PropertyValueFactory<>("author"));

booksYearColumn.setCellValueFactory(**new** PropertyValueFactory<>("year"));

booksIsbnColumn.setCellValueFactory(**new** PropertyValueFactory<>("isbn"));

booksQuantityColumn.setCellValueFactory(**new** PropertyValueFactory<>("quantity"));

authorsIdColumn.setCellValueFactory(**new** PropertyValueFactory<>("id"));

authorsNameColumn.setCellValueFactory(**new** PropertyValueFactory<>("name"));

addLibrarianButton.setOnAction((event -> showAddLibrarianDialog()));

addReaderButton.setOnAction((event -> showAddReaderDialog()));

addBookButton.setOnAction((event -> showAddBookDialog()));

addAuthorButton.setOnAction((event -> showAddAuthorDialog()));

updateLibrariansTable();

}

Callback<TableView<DataBaseEntity>, TableRow<DataBaseEntity>> tableViewCallback = tableView -> {

**final** TableRow<DataBaseEntity> row = **new** TableRow<>();

**final** ContextMenu rowMenu = **new** ContextMenu();

MenuItem removeItem = **new** MenuItem("Удалить");

removeItem.setOnAction(event -> deleteEntity(row.getTableView(), row.getItem().getId()));

rowMenu.getItems().addAll(removeItem);

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty()))

.then(rowMenu)

.otherwise((ContextMenu)**null**));

**return** row;

};

**public** **void** showAddDialog(**int** addEntityType, String fxmlPath, **int** w, **int** h) {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(fxmlPath));

**try** {

Parent parent = fxmlLoader.load();

addDialogStage = **new** Stage();

((AdminAddController) fxmlLoader.getController()).initAddDialog(**this**, addEntityType);

addDialogStage.initModality(Modality.APPLICATION\_MODAL);

addDialogStage.setScene(**new** Scene(parent, w, h));

addDialogStage.setResizable(**false**);

addDialogStage.showAndWait();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** showAddLibrarianDialog() {

showAddDialog(AdminAddController.LIBRARIAN\_TYPE, "../views/add\_librarian\_dialog.fxml", 354, 282);

}

**public** **void** showAddReaderDialog() {

showAddDialog(AdminAddController.READER\_TYPE, "../views/add\_reader\_dialog.fxml", 354, 188);

}

**public** **void** showAddBookDialog() {

showAddDialog(AdminAddController.BOOK\_TYPE, "../views/add\_book\_dialog.fxml", 354, 446);

}

**public** **void** showAddAuthorDialog() {

showAddDialog(AdminAddController.AUTHOR\_TYPE, "../views/add\_author\_dialog.fxml", 354, 188);

}

**public** **void** deleteEntity(TableView table, **int** id) {

**if** (! App.confirm("Подтвердите действие", "Удалить выбранную запись?")) **return**;

**if** (table.equals(librariansTable))

DataBase.deleteRow(id, Constants.LIBRARIAN\_TABLE);

**else** **if** (table.equals(readersTable))

DataBase.deleteRow(id, Constants.READERS\_TABLE);

**else** **if** (table.equals(booksTable))

DataBase.deleteRow(id, Constants.BOOKS\_TABLE);

**else** **if** (table.equals(authorsTable)) {

DataBase.deleteBooksByAuthorId(id);

DataBase.deleteRow(id, Constants.AUTHORS\_TABLE);

}

updateCurrentTable();

}

**public** **void** updateCurrentTable() {

String tabId = currentTab.getId();

**if** (tabId.equals(librariansTab.getId()))

updateLibrariansTable();

**else** **if** (tabId.equals(readersTab.getId()))

updateReadersTable();

**else** **if** (tabId.equals(booksTab.getId()))

updateBooksTable();

**else** **if** (tabId.equals(authorsTab.getId()))

updateAuthorsTable();

}

**public** **void** updateLibrariansTable() {

librariansTable.getItems().clear();

ResultSet rs = DataBase.getLibrariansList();

**try** {

**while**(rs.next())

librariansTable.getItems().add(

**new** Librarian(rs.getInt("id"),

rs.getString("name"),

rs.getString("login"),

rs.getString("password"))

);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **void** updateReadersTable() {

readersTable.getItems().clear();

ResultSet rs = DataBase.getReadersList();

**try** {

**while**(rs.next())

readersTable.getItems().add(**new** Reader(rs.getInt("id"), rs.getString("name")));

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **void** updateBooksTable() {

booksTable.getItems().clear();

ResultSet rs = DataBase.getBooksList();

**try** {

**while**(rs.next())

booksTable.getItems().add(

**new** Book(rs.getInt("id"),

rs.getString("name"),

rs.getString("author"),

rs.getString("year"),

rs.getString("isbn"),

rs.getInt("quantity"))

);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **void** updateAuthorsTable() {

authorsTable.getItems().clear();

ResultSet rs = DataBase.getAuthorsList();

**try** {

**while**(rs.next())

authorsTable.getItems().add(**new** Author(rs.getInt("id"), rs.getString("name")));

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **void** adminExit() {

**if** (App.confirm("Подтвердите действие", "Выйти из учетной записи администратора?")) {

App.stage.close();

**try** {

**new** Main().start(**new** Stage());

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

## Файл AdminAddController.java

**package** ru.readingroom.system.controllers;

**import** javafx.fxml.FXML;

**import** javafx.scene.control.Button;

**import** javafx.scene.control.Label;

**import** javafx.scene.control.ListView;

**import** javafx.scene.control.TextField;

**import** ru.readingroom.system.models.Author;

**import** ru.readingroom.system.models.DataBase;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**public** **class** AdminAddController {

@FXML

**private** TextField librarianNameInput, librarianLoginInput, librarianPasswordInput;

@FXML

**private** TextField readerNameInput;

@FXML

**private** TextField bookNameInput, bookIsbnInput, bookYearInput, bookQuantityInput, bookAuthorNameSearchInput;

@FXML

**private** Button bookAuthorNameClearButton;

@FXML

**private** ListView<Author> bookAuthorsList;

@FXML

**private** Label bookAuthorsListPlaceholder;

@FXML

**private** Button librarianAdd, readerAdd, bookAdd, authorAdd;

**private** AdminController adminController;

**public** **static** **final** **int** LIBRARIAN\_TYPE = 1;

**public** **static** **final** **int** READER\_TYPE = 2;

**public** **static** **final** **int** BOOK\_TYPE = 3;

**public** **static** **final** **int** AUTHOR\_TYPE = 4;

**public** Author selectedAuthor = **null**;

**public** **void** initAddDialog(AdminController adminController, **int** addEntityType) {

**this**.adminController = adminController;

**switch** (addEntityType) {

**case** LIBRARIAN\_TYPE:

librarianAdd.setOnAction((v)-> addLibrarian());

**break**;

**case** READER\_TYPE:

readerAdd.setOnAction((v)-> addReader());

**break**;

**case** BOOK\_TYPE:

bookAuthorNameSearchInput.setOnKeyReleased((v) -> searchAuthor());

bookAuthorsList.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> selectedAuthor = newValue);

bookAuthorNameClearButton.setOnAction((e) -> {

bookAuthorNameSearchInput.setText("");

searchAuthor();

});

bookAdd.setOnAction((v)-> addBook());

**break**;

**case** AUTHOR\_TYPE:

authorAdd.setOnAction((v)-> addAuthor());

**break**;

}

}

**private** **void** addLibrarian() {

String name = librarianNameInput.getText(), login = librarianLoginInput.getText(), password = librarianPasswordInput.getText();

**if** (!name.isEmpty() && !login.isEmpty() && !password.isEmpty()) {

DataBase.insertLibrarian(name, login, password);

adminController.addDialogStage.getScene().getWindow().hide();

adminController.updateLibrariansTable();

}

}

**private** **void** addReader() {

String name = readerNameInput.getText();

**if** (!name.isEmpty()) {

DataBase.insertReader(name);

adminController.addDialogStage.getScene().getWindow().hide();

adminController.updateReadersTable();

}

}

**private** **void** searchAuthor() {

selectedAuthor = **null**;

bookAuthorsList.getItems().clear();

String name = bookAuthorNameSearchInput.getText();

**if** (!name.isEmpty()) {

ResultSet rs = DataBase.searchAuthors(name);

**try** {

**if** (! rs.next())

bookAuthorsListPlaceholder.setText("Ничего не найдено");

**else** {

**do** bookAuthorsList.getItems().add(**new** Author(rs.getInt("id"), rs.getString("name")));

**while** (rs.next());

}

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** bookAuthorsListPlaceholder.setText("Список авторов");

}

**private** **void** addBook() {

**if** (selectedAuthor == **null**) **return**;

String name = bookNameInput.getText();

String isbn = bookIsbnInput.getText();

**int** authorId = selectedAuthor.getId();

String year = bookYearInput.getText();

String quantityStr = bookQuantityInput.getText();

**if** (!name.isEmpty() && !isbn.isEmpty() && !year.isEmpty() && !quantityStr.isEmpty()) {

**int** qty = Integer.valueOf(quantityStr);

**if** (qty < 0) **return**;

DataBase.insertBook(name, isbn, authorId, year, qty);

adminController.addDialogStage.getScene().getWindow().hide();

adminController.updateBooksTable();

}

}

**private** **void** addAuthor() {

String name = readerNameInput.getText();

**if** (!name.isEmpty()) {

DataBase.insertAuthor(name);

adminController.addDialogStage.getScene().getWindow().hide();

adminController.updateAuthorsTable();

}

}

}

## Файл LibrarianController.java

**package** ru.readingroom.system.controllers;

**import** javafx.application.Platform;

**import** javafx.beans.binding.Bindings;

**import** javafx.fxml.FXML;

**import** javafx.fxml.FXMLLoader;

**import** javafx.scene.Node;

**import** javafx.scene.Parent;

**import** javafx.scene.Scene;

**import** javafx.scene.control.\*;

**import** javafx.scene.control.cell.PropertyValueFactory;

**import** javafx.scene.input.MouseEvent;

**import** javafx.stage.Modality;

**import** javafx.stage.Stage;

**import** javafx.util.Callback;

**import** ru.readingroom.system.App;

**import** ru.readingroom.system.Main;

**import** ru.readingroom.system.models.\*;

**import** java.io.IOException;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**public** **class** LibrarianController {

@FXML

**private** Button giveBookButton, acceptBookButton;

@FXML

**private** TableView<DataBaseEntity> relationshipsTable;

@FXML

**private** TableColumn<Object, Object> relationshipsIdColumn;

@FXML

**private** TableColumn<Object, Object> relationshipsReaderColumn;

@FXML

**private** TableColumn<Object, Object> relationshipsBookColumn;

@FXML

**private** TableColumn<Object, Object> relationshipsTimeColumn;

@FXML

**private** Label readersQuantity;

@FXML

**private** ListView readersList;

**public** Stage dialogStage;

**private** Thread updateThread;

@FXML

**void** initialize() {

relationshipsTable.setRowFactory((tableViewCallback));

relationshipsTable.getParent().addEventFilter(MouseEvent.MOUSE\_CLICKED, evt -> {

Node source = evt.getPickResult().getIntersectedNode();

**while** (source != **null** && !(source **instanceof** TableRow)) {

source = source.getParent();

}

**if** (source == **null** || (source **instanceof** TableRow && ((TableRow) source).isEmpty())) {

relationshipsTable.getSelectionModel().clearSelection();

}

});

relationshipsIdColumn.setCellValueFactory(**new** PropertyValueFactory<>("id"));

relationshipsReaderColumn.setCellValueFactory(**new** PropertyValueFactory<>("reader"));

relationshipsBookColumn.setCellValueFactory(**new** PropertyValueFactory<>("book"));

relationshipsTimeColumn.setCellValueFactory(**new** PropertyValueFactory<>("time"));

giveBookButton.setOnAction((event -> showGiveBookDialog()));

acceptBookButton.setOnAction((event -> showAcceptBookDialog()));

update();

updateThread = **new** Thread(() -> {

**while** (**true**) {

**try** {

Thread.sleep(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

Platform.runLater(() -> update());

}

});

updateThread.start();

}

Callback<TableView<DataBaseEntity>, TableRow<DataBaseEntity>> tableViewCallback = tableView -> {

**final** TableRow<DataBaseEntity> row = **new** TableRow<>();

**final** ContextMenu rowMenu = **new** ContextMenu();

MenuItem removeItem = **new** MenuItem("Принять книгу");

removeItem.setOnAction(event -> acceptBook(row.getItem().getId()));

rowMenu.getItems().addAll(removeItem);

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty()))

.then(rowMenu)

.otherwise((ContextMenu)**null**));

**return** row;

};

**public** **void** update() {

updateRelationshipsTable();

updateReadersList();

}

**public** **void** updateRelationshipsTable() {

**int** prevRow = -1;

**if** (relationshipsTable.getSelectionModel().getSelectedItem() != **null**)

prevRow = relationshipsTable.getSelectionModel().getFocusedIndex();

relationshipsTable.getItems().clear();

ResultSet rs = DataBase.getRelationshipsList();

**try** {

**while**(rs.next()) {

**long** currentTimestamp = System.currentTimeMillis() / 1000;

**long** timestamp = rs.getLong("timestamp");

**long** dSeconds = currentTimestamp - timestamp;

**int** dMinutes = (**int**) Math.floor(dSeconds / 60);

**int** dHours = (**int**) Math.floor(dMinutes / 60);

**int** dDays = (**int**) Math.floor(dHours / 24);

**int** dMonth = (**int**) Math.floor(dDays / 30);

String passed = "";

**if** (dMinutes == 0)

passed = dSeconds + " сек.";

**else** **if** (dMinutes < 60)

passed = dMinutes + " мин.";

**else** **if** (dHours < 24)

passed = dHours + " ч.";

**else** **if** (dDays < 30)

passed = dDays + " дн.";

**else** **if** (dMonth < 12)

passed = dMonth + " мес.";

relationshipsTable.getItems().add(**new** Relationship(

rs.getInt("id"),

rs.getString("reader"),

rs.getString("book"),

passed));

}

} **catch** (SQLException e) {

e.printStackTrace();

}

**if** (prevRow >= 0)

relationshipsTable.getSelectionModel().select(prevRow);

}

**public** **void** updateReadersList() {

readersList.getItems().clear();

ResultSet rs = DataBase.getActiveReaders();

**try** {

**int** qty = 0;

**while**(rs.next()) {

readersList.getItems().add(**new** Reader(rs.getInt("id"), rs.getString("name")));

qty++;

}

readersQuantity.setText("Читателей в зале: " + qty);

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **void** showActionDialog(**int** actionType, String fxmlPath, **int** w, **int** h) {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(fxmlPath));

**try** {

Parent parent = fxmlLoader.load();

dialogStage = **new** Stage();

((LibrarianActionController) fxmlLoader.getController()).initActionDialog(**this**, actionType);

dialogStage.initModality(Modality.APPLICATION\_MODAL);

dialogStage.setScene(**new** Scene(parent, w, h));

dialogStage.setResizable(**false**);

dialogStage.showAndWait();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** showGiveBookDialog() {

showActionDialog(LibrarianActionController.GIVE\_ACTION, "../views/give\_book\_dialog.fxml", 354, 460);

}

**public** **void** showAcceptBookDialog() {

Relationship selectedRelationship = (Relationship) relationshipsTable.getSelectionModel().getSelectedItem();

**if** (selectedRelationship != **null**)

acceptBook(selectedRelationship.getId());

**else**

showActionDialog(LibrarianActionController.ACCEPT\_ACTION, "../views/accept\_book\_dialog.fxml", 354, 188);

}

**public** **void** librarianExit() {

**if** (App.confirm("Подтвердите действие", "Выйти из учетной записи библиотекаря?")) {

App.stage.close();

**try** {

**new** Main().start(**new** Stage());

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**public** **void** acceptBook(**int** relationshipId) {

**if** (relationshipId < 1) **return**;

**if** (! App.confirm("Подтвердите действие", "Принять книгу с читальным номером " + relationshipId + "?")) **return**;

**int** bookId = DataBase.getBookIdByRelationship(relationshipId);

DataBase.bookQuantityPlus(bookId, 1);

DataBase.deleteRelationship(relationshipId);

update();

relationshipsTable.getSelectionModel().clearSelection();

}

}

## Файл LibrarianActionController.java

**package** ru.readingroom.system.controllers;

**import** javafx.fxml.FXML;

**import** javafx.scene.control.\*;

**import** ru.readingroom.system.Main;

**import** ru.readingroom.system.models.\*;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**public** **class** LibrarianActionController {

@FXML

**private** TextField readerIdSearchInput, bookNameSearchInput, bookAuthorNameSearchInput, relationshipIdInput;

@FXML

**private** Button readerIdClearButton, bookNameClearButton, bookAuthorNameClearButton, bookGiveButton, bookAcceptButton;

@FXML

**private** ListView<Reader> readersList;

@FXML

**private** ListView<Book> booksList;

@FXML

**private** Label readersListPlaceholder, booksListPlaceholder;

**private** Reader selectedReader;

**private** Book selectedBook;

**public** LibrarianController librarianController;

**public** **static** **final** **int** GIVE\_ACTION = 1;

**public** **static** **final** **int** ACCEPT\_ACTION = 2;

**public** **void** initActionDialog(LibrarianController librarianController, **int** addEntityType) {

**this**.librarianController = librarianController;

**switch** (addEntityType) {

**case** GIVE\_ACTION:

readerIdSearchInput.setOnKeyReleased((v) -> searchReader());

bookNameSearchInput.setOnKeyReleased((v) -> searchBook());

bookAuthorNameSearchInput.setOnKeyReleased((v) -> searchBook());

readersList.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> selectedReader = newValue);

booksList.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> selectedBook = newValue);

readerIdClearButton.setOnAction((event) -> {

readerIdSearchInput.setText("");

searchReader();

});

bookNameClearButton.setOnAction((event) -> {

bookNameSearchInput.setText("");

searchReader();

});

bookAuthorNameClearButton.setOnAction((event) -> {

bookAuthorNameSearchInput.setText("");

searchBook();

});

bookGiveButton.setOnAction((v)-> giveBook());

**break**;

**case** ACCEPT\_ACTION:

bookAcceptButton.setOnAction((v)-> acceptBook());

**break**;

}

}

**private** **void** searchReader() {

selectedReader = **null**;

readersList.getItems().clear();

String idStr = readerIdSearchInput.getText();

**if** (!idStr.isEmpty()) {

**int** id = 0;

**try** {

id = Integer.parseInt(idStr);

} **catch** (NumberFormatException e) { readerIdSearchInput.setText(""); }

**if** (id < 1) **return**;

ResultSet rs = DataBase.searchReaders(id);

**try** {

**if** (! rs.next())

readersListPlaceholder.setText("Ничего не найдено");

**else** {

**do** readersList.getItems().add(**new** Reader(rs.getInt("id"), rs.getString("name")));

**while** (rs.next());

}

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** readersListPlaceholder.setText("Список читателей");

}

**private** **void** searchBook() {

selectedBook = **null**;

booksList.getItems().clear();

String bookName = bookNameSearchInput.getText();

String bookAuthor = bookAuthorNameSearchInput.getText();

**if** (!bookName.isEmpty() || !bookAuthor.isEmpty()) {

ResultSet rs = DataBase.searchBooks(bookName, bookAuthor);

**try** {

**if** (! rs.next())

booksListPlaceholder.setText("Ничего не найдено");

**else** {

**do** booksList.getItems().add(**new** Book(rs.getInt("id"), rs.getString("name"), rs.getString("author"), rs.getString("year"), rs.getString("isbn"), rs.getInt("quantity")));

**while** (rs.next());

}

} **catch** (SQLException e) {

e.printStackTrace();

}

} **else** booksListPlaceholder.setText("Список книг");

}

**private** **void** giveBook() {

**if** (selectedReader != **null** && selectedBook != **null**) {

**if** (selectedBook.getQuantity() > 0) {

DataBase.bookQuantityPlus(selectedBook.getId(), -1);

DataBase.insertRelationship(selectedReader.getId(), selectedBook.getId(), System.currentTimeMillis() / 1000);

librarianController.update();

librarianController.dialogStage.getScene().getWindow().hide();

} **else** Main.showMsg("Ошибка", "Недостаточно экземпляров", Alert.AlertType.ERROR, **null**);

}

}

**private** **void** acceptBook() {

String idStr = relationshipIdInput.getText();

**if** (!idStr.isEmpty()) {

**int** id = 0;

**try** {

id = Integer.parseInt(idStr);

librarianController.acceptBook(id);

librarianController.dialogStage.getScene().getWindow().hide();

} **catch** (NumberFormatException e) {

relationshipIdInput.setText("");

}

}

}

}

# Приложение 4

**Структура базы данных**

**SET** SQL\_MODE="NO\_AUTO\_VALUE\_ON\_ZERO";

**SET** time\_zone = "+03:00";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `admins` (

`id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT**,

`name` **VARCHAR**(128) **NOT** **NULL**,

`login` **VARCHAR**(64) **NOT** **NULL**,

`password` **VARCHAR**(64) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 **AUTO\_INCREMENT**=1;

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `authors` (

`id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT**,

`name` **VARCHAR**(128) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 **AUTO\_INCREMENT**=1;

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `books` (

`id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT**,

`name` **VARCHAR**(300) **NOT** **NULL**,

`author\_id` **INT**(11) **NOT** **NULL**,

`year` **VARCHAR**(4) **NOT** **NULL**,

`isbn` **VARCHAR**(20) **NOT** **NULL**,

`quantity` **INT**(11) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 **AUTO\_INCREMENT**=1;

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `librarians` (

`id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT**,

`name` **VARCHAR**(128) **NOT** **NULL**,

`login` **VARCHAR**(64) **NOT** **NULL**,

`password` **VARCHAR**(64) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 **AUTO\_INCREMENT**=1;

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `readers` (

`id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT**,

`name` **VARCHAR**(128) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 **AUTO\_INCREMENT**=1;

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `relationships` (

`id` **INT**(10) **UNSIGNED** **NOT** **NULL** **AUTO\_INCREMENT**,

`reader\_id` **INT**(11) **UNSIGNED** **NOT** **NULL**,

`book\_id` **INT**(11) **UNSIGNED** **NOT** **NULL**,

`timestamp` **INT**(11) **UNSIGNED** **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB **DEFAULT** CHARSET=utf8 **AUTO\_INCREMENT**=1;