



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

**«Создание приложения для общественной
библиотеки»**

Студент ИУ7-65Б
(Группа)

(Подпись, дата) Е.Б. Гришин
(И.О.Фамилия)

Руководитель курсового проекта

(Подпись, дата) Л.Л. Волкова
(И.О.Фамилия)

2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
И.В. Рудаков
(И.О.Фамилия)
« ____ » _____ 2021 г.

ЗАДАНИЕ на выполнение курсового проекта

по дисциплине Базы данных

Студент группы ИУ7-65Б

Гришин Егор Борисович
(Фамилия, имя, отчество)

Тема курсового проекта Создание приложения для общественной библиотеки

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Разработать программу, предоставляющую интерфейс для получения информации о книгах и посетителях в библиотеке, добавления и удаления книг, составления статистики чтения.

Оформление курсового проекта:

Расчетно-пояснительная записка на 25-30 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту проекта должна быть предоставлена презентация, состоящая из 15-20 слайдов.

На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс, результаты проведенных исследований.

Дата выдачи задания « ____ » _____ 2021 г.

Руководитель курсового проекта

Л.Л. Волкова
(Подпись, дата) (И.О.Фамилия)

Студент

Е.Б. Гришин
(Подпись, дата) (И.О.Фамилия)

Содержание

Введение	6
1 Аналитический раздел	7
1.1 Постановка задачи	7
1.2 Пользователи системы	7
1.2.1 Гость	7
1.2.2 Читатель	7
1.2.3 Библиотекарь	8
1.2.4 Администратор	8
1.3 Анализ моделей баз данных	8
1.3.1 Иерархическая база данных	8
1.3.2 Сетевая модель базы данных	9
1.3.3 Реляционная модель базы данных	9
1.3.4 Выбор модели данных	11
1.4 Вывод	11
2 Конструкторский раздел	12
2.1 Сценарии пользователей	12
2.1.1 Гость	12
2.1.2 Любой авторизованный пользователь	13

2.1.3	Читатель	13
2.1.4	Библиотекарь	15
2.1.5	Администратор	17
2.2	Проектирование базы данных	18
2.2.1	Формализация сущностей системы	18
2.2.2	Ролевая модель	20
2.2.3	Триггеры	22
2.3	Вывод	25
3	Технологический раздел	26
3.1	Анализ СУБД	26
3.1.1	MySQL	26
3.1.2	Microsoft SQL Server	27
3.1.3	PostgreSQL	28
3.1.4	Oracle	28
3.2	Средства реализации поставленной задачи	29
3.3	Создание базы данных	30
3.4	Создание таблиц	31
3.5	Создание пользователей системы	33
3.6	Триггеры	34
3.7	Интерфейс приложения	37

3.8 Вывод	42
Заключение	43
Литература	44
Приложение А. Презентация.	44

Введение

В современном мире каждая сфера деятельности связана с использованием информационных технологий, в частности, сети Интернет. Сейчас уже невозможно представить какую-либо компанию или организацию без своего сайта.

Целью данной работы является разработка базы данных и приложения для библиотеки.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) формализовать задание, выделив соответствующих акторов и их функционал;
- 2) провести анализ СУБД и выбрать наиболее подходящую;
- 3) спроектировать базу данных;
- 4) спроектировать архитектуру приложения;
- 5) разработать приложение.

1 Аналитический раздел

В данном разделе будет поставлена задача, рассмотрены возможные пользователи системы, модели данных и СУБД.

1.1 Постановка задачи

Необходимо разработать программу, предоставляющую интерфейс для получения информации о книгах и совершения действий с ними, доступных в зависимости от роли пользователя.

1.2 Пользователи системы

1.2.1 Гость

Гость — это неавторизованный пользователь. Он может только просматривать список книг и авторов и информацию о них.

1.2.2 Читатель

Читатель - это авторизованный пользователь. Он может подать заявку на взятие книги, если она у него не на руках, и на её сдачу в противном случае. Также ему доступны списки книг, на которые он оставил заявки, которые у него на руках и которые он уже прочел и сдал.

1.2.3 Библиотекарь

Библиотекарь - это авторизованный пользователь, контролирующий выдачу книг. Он одобряет заявки читателей, и добавляет новые книги и авторов в БД.

1.2.4 Администратор

Администратор является авторизованным пользователем с повышенным уровнем полномочий — он может регистрировать новых пользователей и удалять существующих.

1.3 Анализ моделей баз данных

Модель базы данных - это тип модели данных, которая определяет логическую структуру базы данных и в корне определяет, каким образом данные могут храниться, организовываться и обрабатываться.

1.3.1 Иерархическая база данных

Иерархическая модель базы данных подразумевает, что элементы организованы в структуры, связанные между собой иерархическими или древовидными связями. Родительский элемент может иметь несколько дочерних элементов. Но у дочернего элемента может быть только один предок.

Иерархические базы данных графически могут быть представлены как перевернутое дерево, состоящее из объектов различных уровней. Верх-

ний уровень (корень дерева) занимает один объект, второй - объекты второго уровня и так далее.

Такая модель подразумевает возможность существования одинаковых (преимущественно дочерних) элементов. Данные здесь хранятся в серии записей с прикреплёнными к ним полями значений. Модель собирает вместе все экземпляры определённой записи в виде «типов записей» — они эквивалентны таблицам в реляционной модели, а отдельные записи — столбцам таблицы.

1.3.2 Сетевая модель базы данных

Сетевая модель базы данных подразумевает, что у родительского элемента может быть несколько потомков, а у дочернего элемента — несколько предков. Записи в такой модели связаны списками с указателями.

Сетевая модель состоит из множества записей, которые могут быть владельцами или членами групповых отношений. Связь между записью-владельцем и записью-членом также имеет вид 1:N.

Сетевая модель позволяет иметь несколько предков и потомков, формирующих решётчатую структуру.

На рисунке 1 представлен пример сетевой модели базы данных.

1.3.3 Реляционная модель базы данных

В реляционной модели, в отличие от иерархической или сетевой, не существует физических отношений. Вся информация хранится в виде таблиц

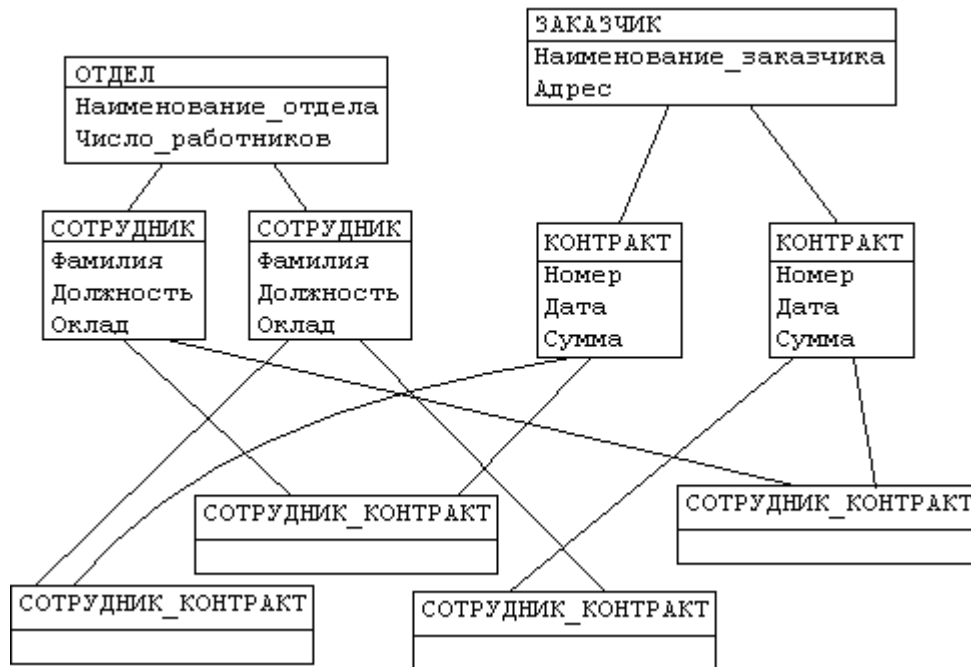


Рисунок 1 – Иерархическая модель базы данных

(отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столбцами, а не физическими ссылками или указателями. Для манипуляций с рядами данных существуют специальные операторы.

Реляционные таблицы обладают следующими свойствами:

- 1) все значения атомарны;
- 2) каждый ряд уникален;
- 3) порядок столбцов не важен;
- 4) порядок рядов не важен;
- 5) у каждого столбца есть своё уникальное имя.

Термин «реляционный» означает, что теория основана на математическом понятии отношение (relation). В качестве неформального синонима

термину «отношение» часто встречается слово таблица. Необходимо помнить, что «таблица» есть понятие нестрогое и неформальное и часто означает не «отношение» как абстрактное понятие, а визуальное представление отношения на бумаге или экране. Некорректное и нестрогое использование термина «таблица» вместо термина «отношение» нередко приводит к недопониманию.

1.3.4 Выбор модели данных

В качестве модели данных была выбрана реляционная модель, так как структура данных однозначно определена, структура данных не является быстроизменяющейся и данные подчиняются строгим правилам и ограничениям.

1.4 Вывод

В данном разделе была поставлена задача, рассмотрены возможные пользователи системы, проанализированы модели данных и СУБД.

2 Конструкторский раздел

В данном разделе будут рассмотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и разработаны триггеры.

2.1 Сценарии пользователей

Необходимо определить функционал каждого из пользователей.

2.1.1 Гость

Гость - это неавторизованный пользователь, у которого есть только одна возможность - авторизоваться. На рисунке 2 представлена Use-Case-диаграмма для гостя.



Рисунок 2 – Сценарии для гостя.

2.1.2 Любой авторизированный пользователь

Функционал, предоставляемый каждому авторизированному пользователю (читателю, библиотекарю, администратору):

- 1) просмотр списка книг;
- 2) просмотр информации о книге;
- 3) просмотр списка авторов;
- 4) просмотр информации о книге.

2.1.3 Читатель

Дополнительная функциональность, предоставляемая пользователю читатель:

- 1) оставить заявку на получение книги;
- 2) оставить заявку на сдачу книги;
- 3) просмотр списка всех заявок;
- 4) просмотр списка всех книг на руках;
- 5) просмотр списка всех прочитанных и сданных книг.

На рисунке 3 представлена Use-Case-диаграмма для читателя.

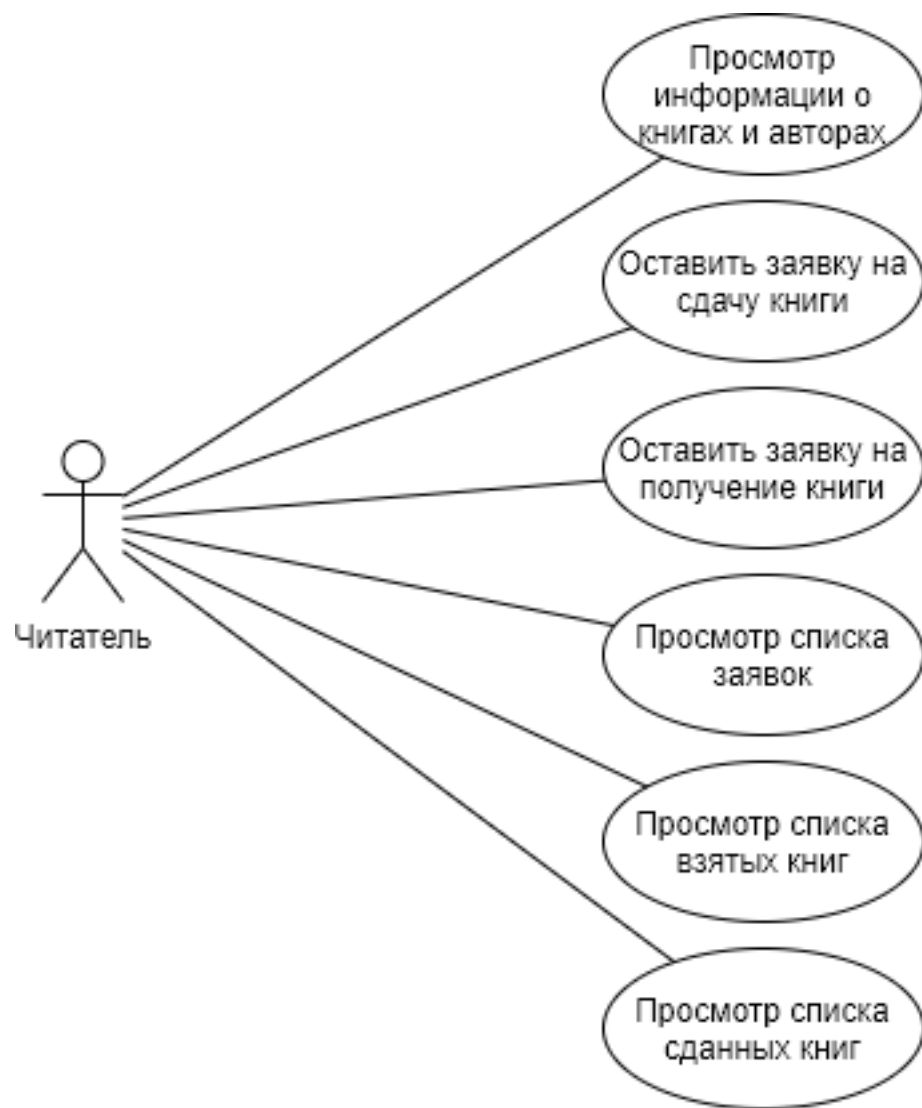


Рисунок 3 – Сценарии для читателя.

2.1.4 Библиотекарь

Дополнительная функциональность, предоставляемая пользователю библиотекарю:

- 1) просмотр и одобрение всех заявок на выдачу книг;
- 2) просмотр и одобрение всех заявок на сдачу книг;
- 3) добавление нового автора;
- 4) добавление новой книги существующему автору.

На рисунке 4 представлена Use-Case-диаграмма для библиотекаря.



Рисунок 4 – Сценарии для библиотекаря.

2.1.5 Администратор

Дополнительная функциональность, предоставляемая пользователю администратор:

- 1) зарегистрировать нового пользователя;
- 2) удалить пользователя.

На рисунке 5 представлена Use-Case-диаграмма для администратора.

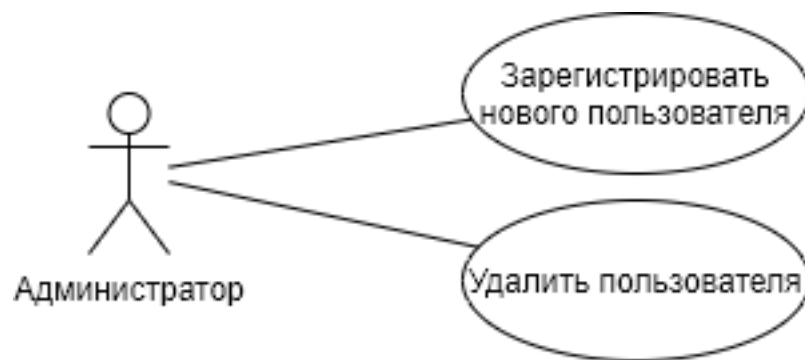


Рисунок 5 – Сценарии для администратора.

2.2 Проектирование базы данных

2.2.1 Формализация сущностей системы

Необходимо выделить сущности предметной области и построить ER-диаграмму.

На рисунке 6 представлена ER-диаграмма системы.

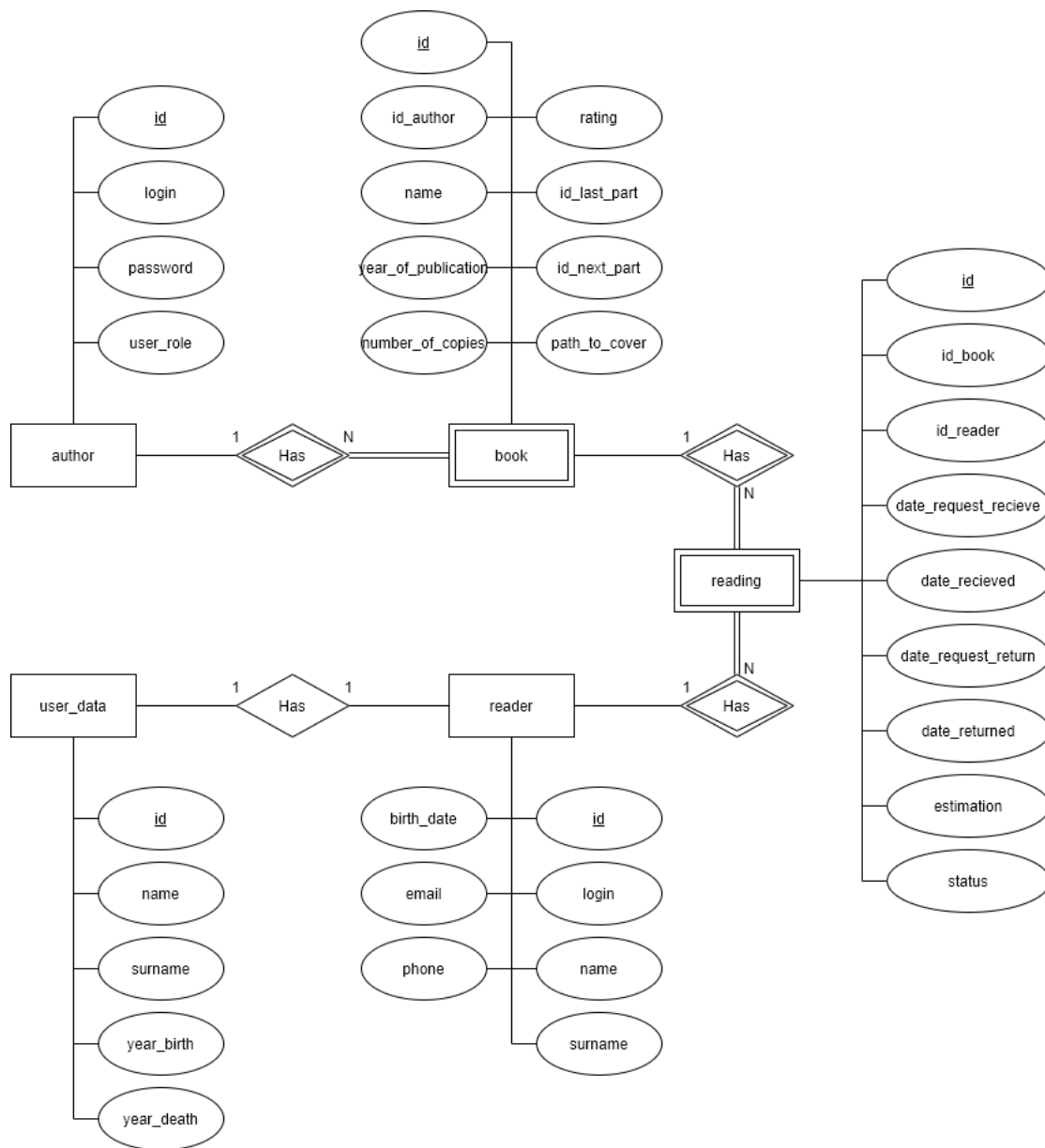


Рисунок 6 – ER-диаграмма системы.

Выделенные сущности:

- 1) author - таблица, в которой хранится информация об авторах;
- 2) book - таблица, в которой хранится информация об книгах;
- 3) user_data - таблица, в которой хранятся данные для авторизации зарегистрированных пользователей;
- 4) reader - таблица, в которой хранится информация о читателях;
- 5) reading - таблица, в которой хранится информация о взятых книгах.

Атрибут status таблицы reading может принимать 4 значения:

- 1) 'request_recieve' - подана заявка на взятие книги;
- 2) 'in_reading' - книга выдана;
- 3) 'request_return' - подана заявка на сдачу книги;
- 4) 'returned' - книга сдана.

2.2.2 Ролевая модель

На уровне базы данных выделена следующая ролевая модель:

- 1) guest - гость;
- 2) reader - читатель;
- 3) librarian - библиотекарь;

- 4) admin - администратор.

Использование ролевой модели на уровне базы данных гарантирует безопасность доступа к объектам базы данных.

Гость

Пользователь с ролью guest имеет следующие права:

- 1) SELECT над таблицей author;
- 2) SELECT над таблицей book;
- 3) SELECT над таблицей user_data.

Читатель

Пользователь с ролью reader имеет следующие права:

- 1) SELECT над таблицей book;
- 2) SELECT, INSERT, UPDATE над таблицей reading;
- 3) SELECT над таблицей author;
- 4) SELECT над таблицей reader;
- 5) SELECT над таблицей user_data.

Библиотекарь

Пользователь с ролью librarian имеет следующие права:

- 1) SELECT, INSERT, DELETE, UPDATE над таблицей book;

- 2) SELECT, INSERT, DELETE, UPDATE над таблицей reading;
- 3) SELECT, INSERT, DELETE, UPDATE над таблицей author;
- 4) SELECT над таблицей reader;
- 5) SELECT над таблицей user_data.

Администратор

Пользователь с ролью admin обладает всеми правами.

2.2.3 Триггеры

Для обеспечения целостности данных необходимо добавить соответствующие триггеры.

При возврате книги, в соответствующей записи в таблице reading полю status присваивается значение 'returned'. Также при возврате книги ей может быть дана оценка - estimation. На основании всех данных оценок строится рейтинг книги. На рисунке 7 представлен алгоритм триггера, обеспечивающего актуальность рейтинга.

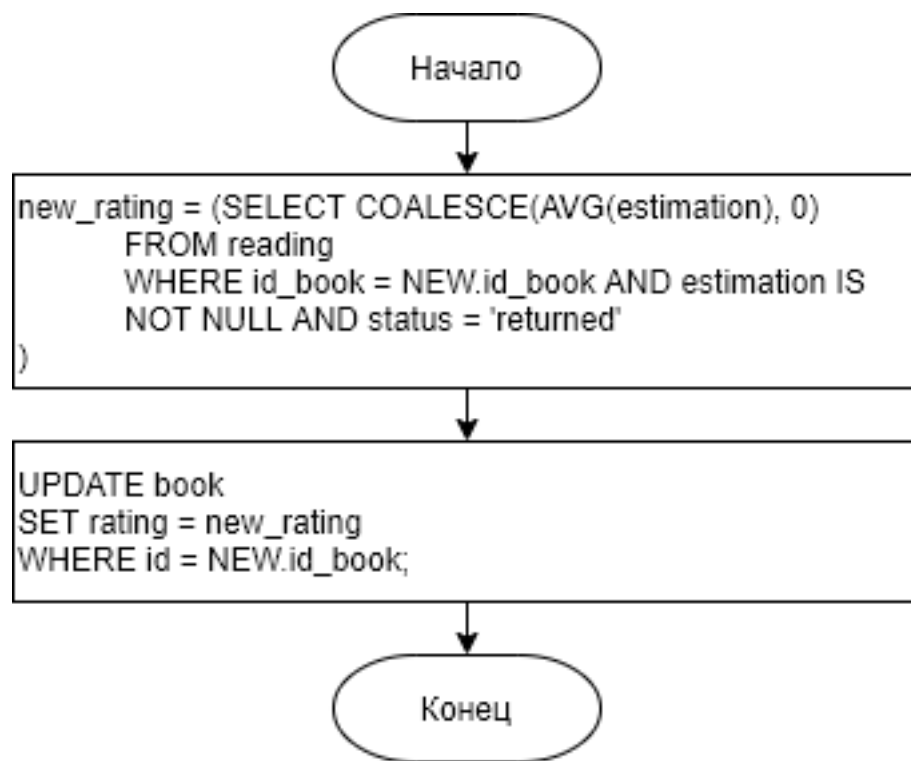


Рисунок 7 – Алгоритм триггера рейтинга.

При выдаче или сдаче книги, в соответствующей записи в таблице reading полю status присваивается значение 'in_reading' или 'returned' соответственно. Триггер должен изменять значение number_of_copies соответствующей книги - количество оставшихся экземпляров книги. На рисунке 8 представлен алгоритм триггера, обеспечивающего актуальность количества оставшихся экземпляров книги.

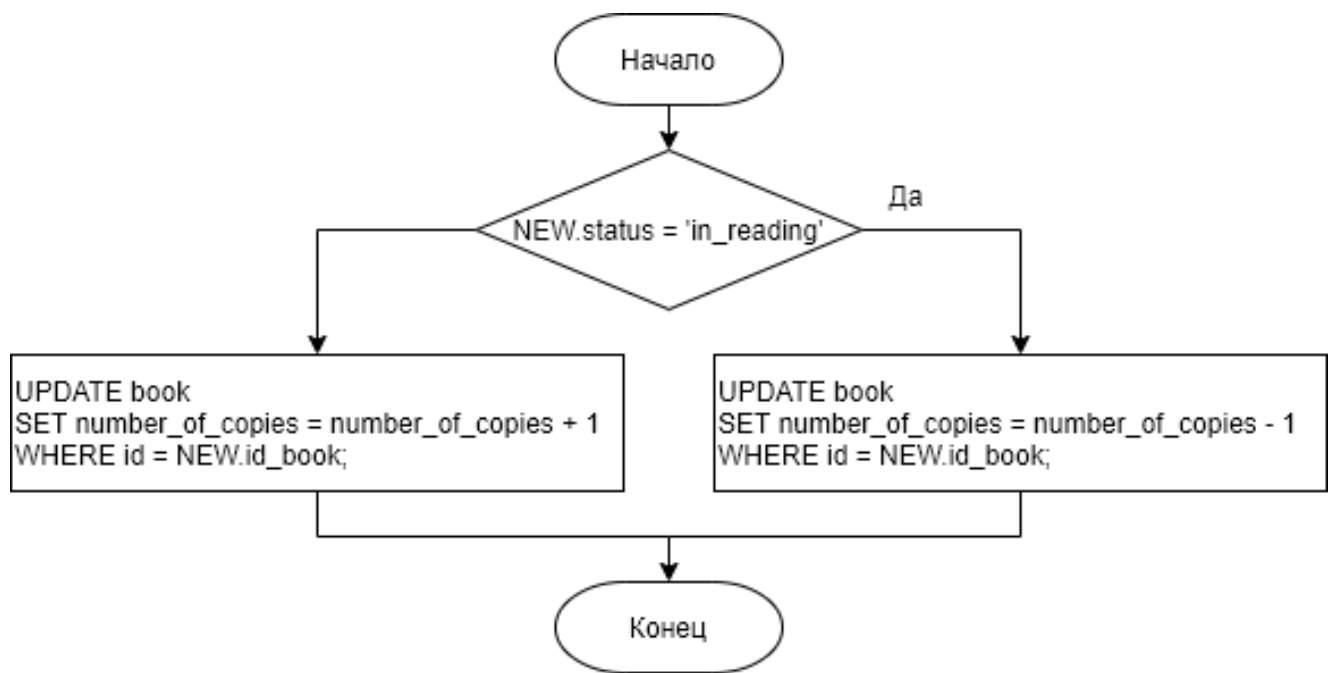


Рисунок 8 – Алгоритм триггера рейтинга.

При добавлении новой книги нужно проконтролировать, что её ссылки на предыдущую и следующую части однозначны. На рисунке 9 представлен алгоритм триггера, обеспечивающего целостность связи частей серии.

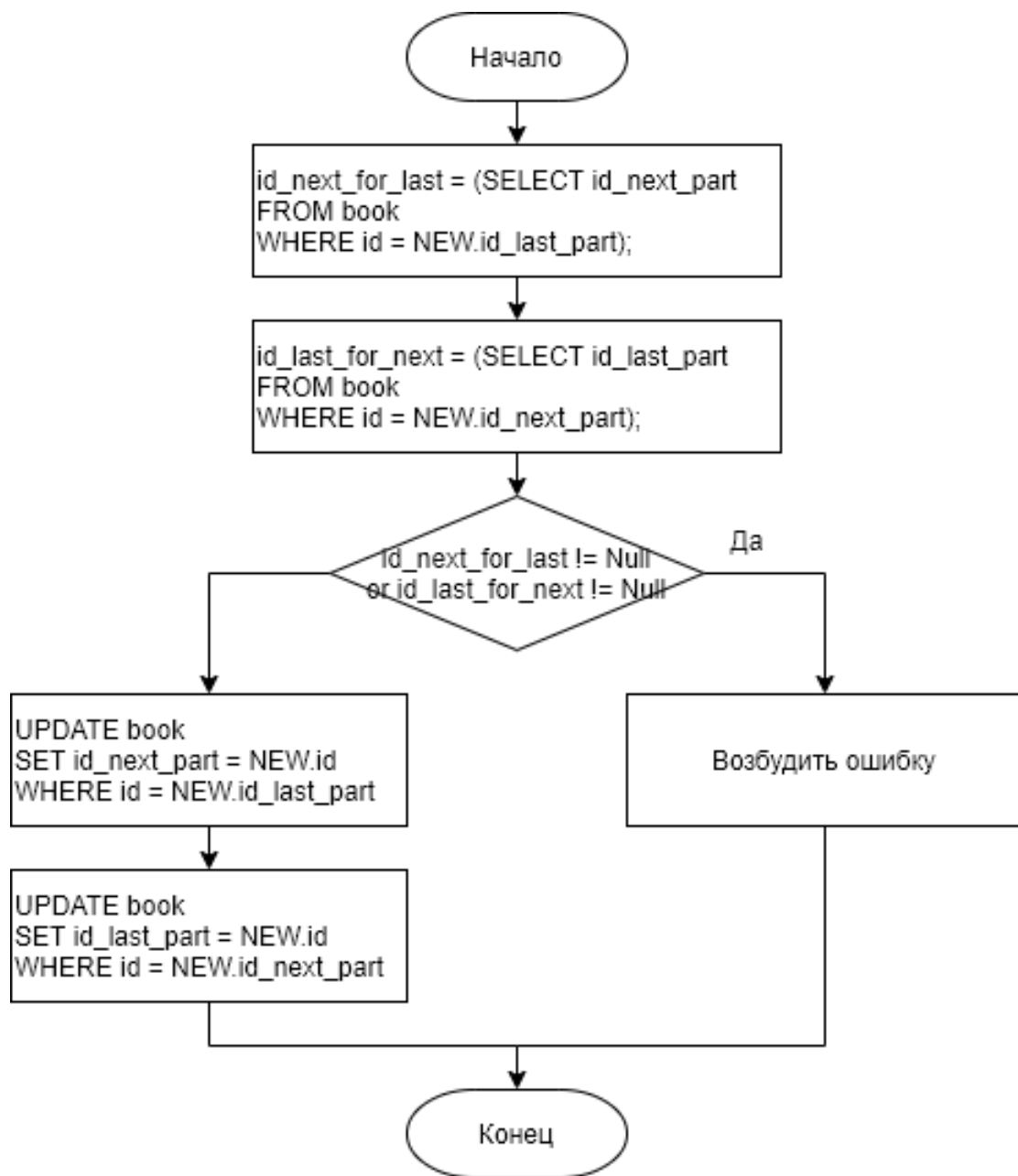


Рисунок 9 – Алгоритм триггера целостности частей серии.

2.3 Вывод

В данном разделе были рассмотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и приведены схемы алгоритмов разработанных триггеров.

3 Технологический раздел

В данном разделе будут выбраны средства реализации поставленной задачи, создана база данных и ролевая модель, разработано приложение.

3.1 Анализ СУБД

СУБД — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД. Самыми популярными СУБД являются MySQL, Microsoft SQL Server, PostgreSQL и Oracle.

3.1.1 MySQL

MySQL — реляционная СУБД с открытым исходным кодом, главными плюсами которой являются ее скорость и гибкость, которая обеспечена поддержкой большого количества различных типов таблиц.

Кроме того, это надежная бесплатная система с простым интерфейсом и возможностью синхронизации с другими базами данных. В совокупности эти факторы позволяют использовать MySQL как крупным корпорациям, так и небольшим компаниям.

Преимущества:

- 1) простота в использовании;

- 2) обширный функционал;
- 3) безопасность;
- 4) масштабируемость;
- 5) скорость.

Недостатки: недостаточная надежность;

3.1.2 Microsoft SQL Server

Система позволяет синхронизироваться с другими программными продуктами компании Microsoft, а также обеспечивает надежную защиту данных и простой интерфейс, однако отличается высокой стоимостью лицензии и повышенным потреблением ресурсов.

Преимущества:

- 1) СУБД масштабируется;
- 2) простота в использовании;
- 3) возможность интеграции с другими продуктами Microsoft.

Недостатки:

- 1) высокая стоимость продукта для юридических лиц;
- 2) возможны проблемы в работе служб интеграции импорта файлов;
- 3) высокая ресурсоемкость SQL Server.

3.1.3 PostgreSQL

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества:

- 1) бесплатное ПО с открытым исходным кодом;
- 2) большое количество дополнений;
- 3) расширения;

Недостатки:

- 1) производительность;
- 2) популярность;

3.1.4 Oracle

Oracle – это объектно-реляционная система управления базами данных.

Преимущества:

- 1) поддержка огромных баз данных;
- 2) быстрая обработка транзакций;
- 3) большой и постоянно развивающийся функционал.

Недостатки:

- 1) высокая стоимость;
- 2) значительные вычислительные ресурсы.

	Oracle	MySQL	Microsoft SQL Server	PostgreSQL
Простота в использовании	+	+	+	+
Бесплатная	-	+	+	+
Безопасность данных	+	-	+	+
Поддержка стандарта SQL	+	+	+	+
Поддержка хранимых процедур и триггеров	+	+	+	+
Кроссплатформенность	+	+	-	+

Рисунок 10 – Анализ СУБД.

3.2 Средства реализации поставленной задачи

Для разработки программы выбран язык Python по причине совмещения нескольких парадигм программирования, а также из-за большого разнообразия представленных библиотек и фреймворков для создания веб-приложения.

В качестве среды разработки был выбран PyCharm, так как с его помощью удобно работать и с python-файлами, и html-страницами. Среда обладает кроссплатформенностью и большим выбором настроек проекта.

Для создания данного проекта в качестве инструмента, который облегчит процесс создания веб-приложения, был выбран фреймворк Flask. Благодаря его простоте и гибкости, разработчик может сам выбрать способ реализации тех или иных задач.

Для данной задачи PostgreSQL[1] выигрывает по многим параметрам и поэтому было решено использовать именно эту СУБД.

3.3 Создание базы данных

Необходимо построить диаграмму БД по выделенным сущностям. На рисунке 11 представлена диаграмма БД.

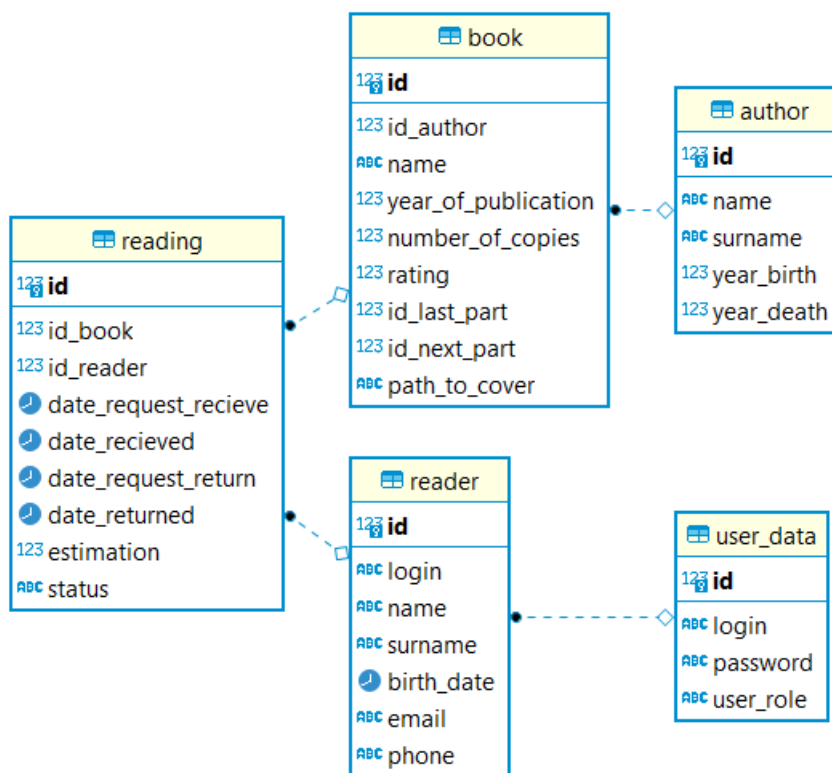


Рисунок 11 – Диаграмма БД.

3.4 Создание таблиц

Далее представлены листинги создания таблиц с указанием типов данных каждого столбца.

Таблица user_data.

Листинг 1 – Создание таблицы user_data.

```
0 CREATE type user_role_t as enum (  
1     'admin',  
2     'librarian',  
3     'reader'  
4 );  
5  
6 CREATE TABLE user_data (  
7     id SERIAL NOT NULL PRIMARY KEY,  
8     login TEXT NOT NULL UNIQUE,  
9     password TEXT NOT NULL,  
10    user_role user_role_t NOT NULL  
11 );
```

Таблица reader.

Листинг 2 – Создание таблицы reader.

```
0 CREATE TABLE IF NOT EXISTS reader (  
1     id SERIAL NOT NULL PRIMARY KEY,  
2     login TEXT REFERENCES user_data(login) ON DELETE CASCADE NOT NULL,  
3     name TEXT NOT NULL,  
4     surname TEXT NOT NULL,  
5     birth_date DATE NOT NULL,  
6     email TEXT NOT NULL UNIQUE CHECK(email SIMILAR TO '%@%.%'),  
7     phone TEXT NOT NULL UNIQUE CHECK(phone SIMILAR TO '[0-9]{11}'))  
8 );
```

Таблица author.

Листинг 3 – Создание таблицы author.

```
0 CREATE TABLE IF NOT EXISTS author (  
1     id SERIAL NOT NULL PRIMARY KEY,  
2     name TEXT NOT NULL,  
3     surname TEXT NOT NULL,  
4     year_birth INT NOT NULL,  
5     year_death INT  
6 );
```

Таблица book.

Листинг 4 – Создание таблицы book.

```
0 CREATE TABLE IF NOT EXISTS book (  
1     id SERIAL NOT NULL PRIMARY KEY,  
2     id_author INT REFERENCES author(id) ON DELETE CASCADE NOT NULL,  
3     name TEXT UNIQUE NOT NULL,  
4     year_of_publication INT NOT NULL CHECK(year_of_publication > 1000),  
5     number_of_copies INT NOT NULL CHECK(number_of_copies >= 0),  
6     rating NUMERIC(3, 1) DEFAULT 0 CHECK(RATING >= 0 AND RATING <= 10) NOT  
7     NULL,  
8     id_last_part INT,  
9     id_next_part INT,  
10    path_to_cover TEXT  
11 );
```

Таблица reading.

Листинг 5 – Создание таблицы reading.

```
0 CREATE type reading_status_t as enum (  
1     'request_recieve',  
2     'in_reading',  
3     'request_return',  
4     'returned'  
5 );  
6
```

```

7 CREATE TABLE IF NOT EXISTS reading (
8     id SERIAL NOT NULL PRIMARY key,
9     id_book INT REFERENCES book(id) ON DELETE CASCADE NOT NULL,
10    id_reader INT REFERENCES reader(id) ON DELETE CASCADE NOT NULL,
11    date_request_recieve TIMESTAMP NOT NULL,
12    date_recieved TIMESTAMP DEFAULT 'infinity' NOT NULL,
13    date_request_return TIMESTAMP DEFAULT 'infinity' NOT NULL,
14    date_returned TIMESTAMP DEFAULT 'infinity' NOT NULL,
15    estimation REAL CHECK(estimation >= 0 and estimation <= 10),
16    status reading_status_t NOT NULL
17 );

```

3.5 Создание пользователей системы

Для того, чтобы обеспечить безопасность доступа к данным необходимо создать пользователей с соответствующими правами.

Гость

Листинг 6 – Создание пользователя guest.

```

0 CREATE ROLE guest WITH LOGIN PASSWORD '1234';
1 GRANT SELECT ON TABLE book TO guest;
2 GRANT SELECT ON TABLE author TO guest;
3 GRANT SELECT ON TABLE user_data TO guest;

```

Читатель

Листинг 7 – Создание пользователя reader.

```

0 CREATE ROLE reader WITH LOGIN PASSWORD '1234';
1 GRANT SELECT ON TABLE book TO reader;
2 GRANT SELECT, INSERT, UPDATE ON TABLE reading TO reader;
3 GRANT SELECT ON TABLE author TO reader;
4 GRANT SELECT ON TABLE reader TO reader;
5 GRANT SELECT ON TABLE user_data TO reader;

```



```

6 GRANT USAGE, SELECT ON SEQUENCE reading_id_seq TO reader;
7 GRANT EXECUTE ON FUNCTION update_rating TO reader;

```

Библиотекарь

Листинг 8 – Создание пользователя librarian.

```

0 CREATE ROLE librarian WITH LOGIN PASSWORD '1234';
1 GRANT SELECT, INSERT, DELETE, UPDATE ON TABLE book TO librarian;
2 GRANT SELECT, INSERT, DELETE, UPDATE ON TABLE reading TO librarian;
3 GRANT SELECT, INSERT, DELETE, UPDATE ON TABLE author TO librarian;
4 GRANT SELECT ON TABLE reader TO librarian;
5 GRANT SELECT ON TABLE user_data TO librarian;
6 GRANT USAGE, SELECT ON SEQUENCE reading_id_seq, book_id_seq, author_id_seq
  TO librarian;

```

Администратор

Листинг 9 – Создание пользователя admin.

```

0 CREATE ROLE admin WITH SUPERUSER LOGIN PASSWORD '1234';

```

3.6 Триггеры

В листинге 10 представлена реализация триггера update_rating_trigger.

Листинг 10 – Реализация триггера update_rating_trigger.

```

0 CREATE FUNCTION update_rating() RETURNS TRIGGER AS
1 $$
2 DECLARE
3     new_rating REAL;
4 BEGIN
5     new_rating = (
6         SELECT COALESCE(AVG(estimation), 0)
7         FROM reading

```

```

8      WHERE id_book = NEW.id_book AND estimation IS NOT NULL AND status
      = 'returned'
9  );
10  UPDATE book
11  SET rating = new_rating
12  WHERE id = NEW.id_book;
13  RETURN NEW;
14 END;
15 $$ LANGUAGE plpgsql;
16
17 CREATE TRIGGER update_rating_trigger
18 AFTER UPDATE ON reading
19 FOR EACH ROW
20 WHEN (NEW.status = 'returned')
21 EXECUTE PROCEDURE update_rating();

```

В листинге 11 представлена реализация триггера update_number_of_copies_trigger.

Листинг 11 – Реализация триггера update_number_of_copies_trigger.

```

0 CREATE FUNCTION update_number_of_copies() RETURNS TRIGGER AS
1 $$
2 BEGIN
3     IF NEW.status = 'in_reading' THEN
4         UPDATE book
5         SET number_of_copies = number_of_copies - 1
6         WHERE id = NEW.id_book;
7     ELSIF NEW.status = 'returned' THEN
8         UPDATE book
9         SET number_of_copies = number_of_copies + 1
10        WHERE id = NEW.id_book;
11    END IF;
12    RETURN NEW;
13 END;
14 $$ LANGUAGE plpgsql;
15

```

```

16 CREATE TRIGGER update_number_of_copies_trigger
17 AFTER UPDATE ON reading
18 FOR EACH ROW
19 WHEN (NEW.status = 'in_reading' OR NEW.status = 'returned')
20 EXECUTE PROCEDURE update_number_of_copies();

```

В листинге 12 представлена реализация триггера parts_connect_trigger.

Листинг 12 – Реализация триггера parts_connect_trigger.

```

0 CREATE FUNCTION check_parts() RETURNS TRIGGER AS
1 $$
2 DECLARE
3     id_next_for_last INT;
4     id_last_for_next INT;
5 BEGIN
6     id_next_for_last = (SELECT id_next_part
7                          FROM book
8                          WHERE id = NEW.id_last_part);
9     id_last_for_next = (SELECT id_last_part
10                        FROM book
11                        WHERE id = NEW.id_next_part);
12
13     IF id_next_for_last IS NOT NULL OR id_last_for_next IS NOT NULL THEN
14         RETURN NULL;
15     ELSE
16         UPDATE book
17         SET id_next_part = NEW.id
18         WHERE id = NEW.id_last_part;
19         UPDATE book
20         SET id_last_part = NEW.id
21         WHERE id = NEW.id_next_part;
22         RETURN NEW;
23     END IF;
24 END;
25 $$ LANGUAGE plpgsql;

```

```

26
27 CREATE TRIGGER parts_connect_trigger
28 BEFORE INSERT ON book
29 FOR EACH ROW EXECUTE PROCEDURE check_parts();

```

3.7 Интерфейс приложения

На рисунках ниже показан интерфейс приложения.

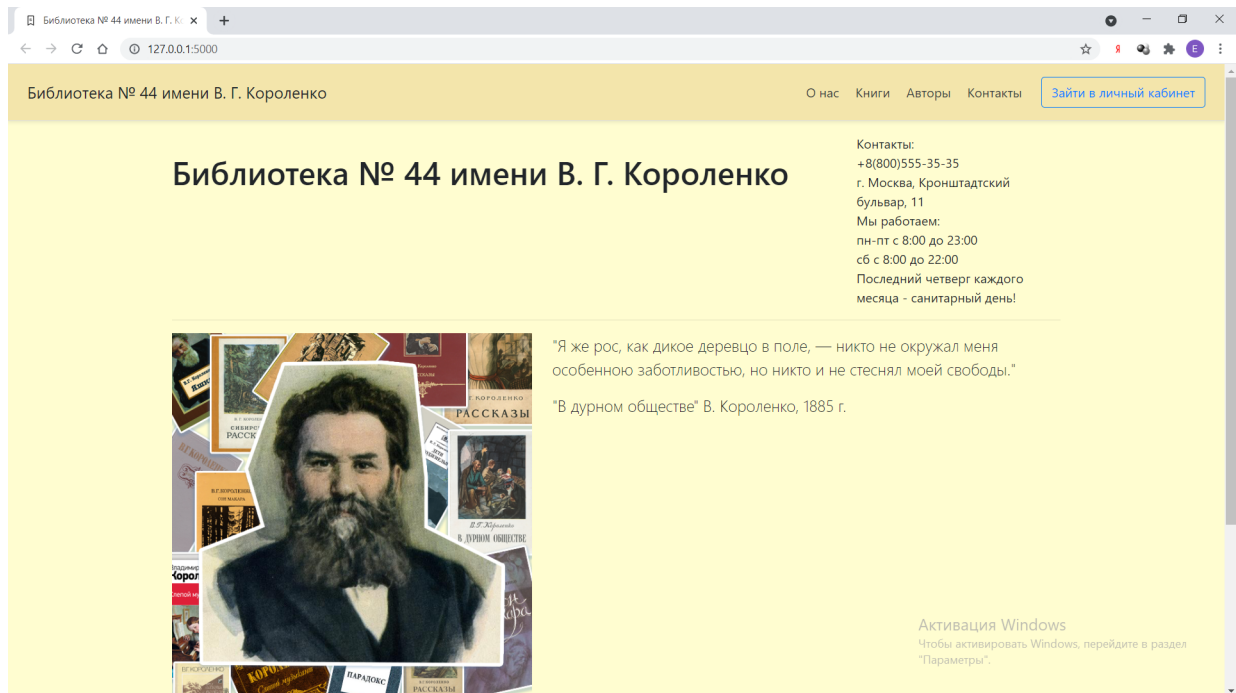


Рисунок 12 – Стартовая страница.

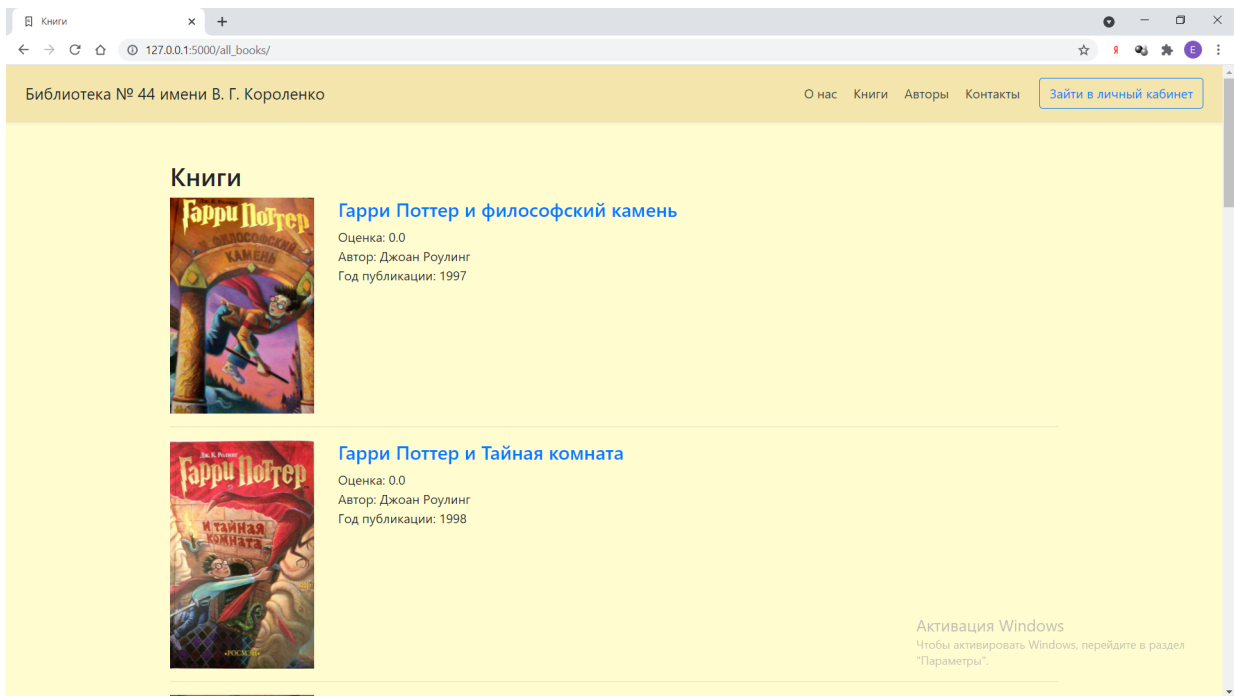


Рисунок 13 – Список книг.

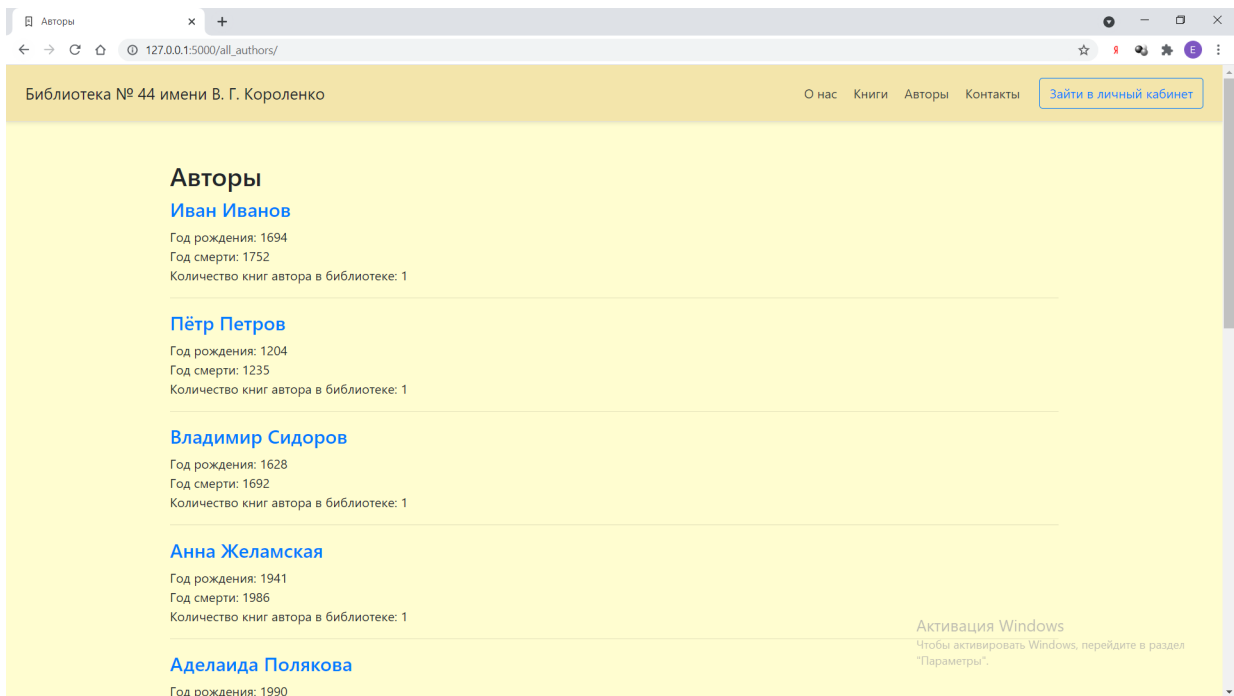


Рисунок 14 – Список авторов.

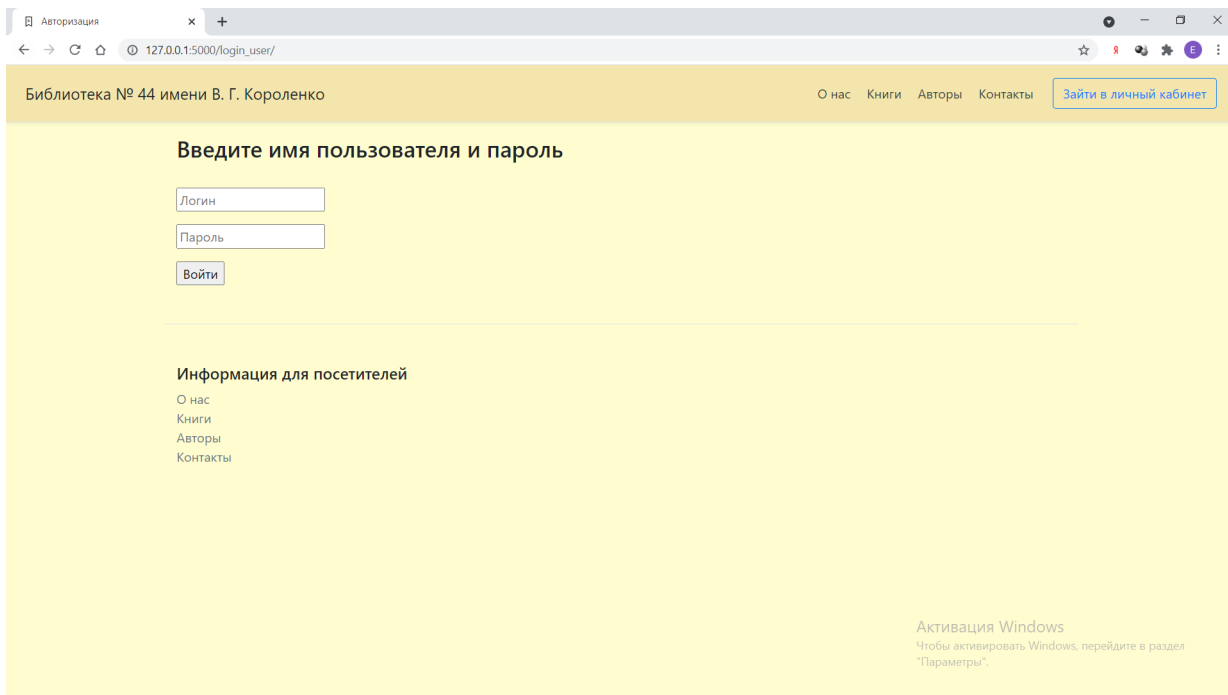


Рисунок 15 – Авторизация.

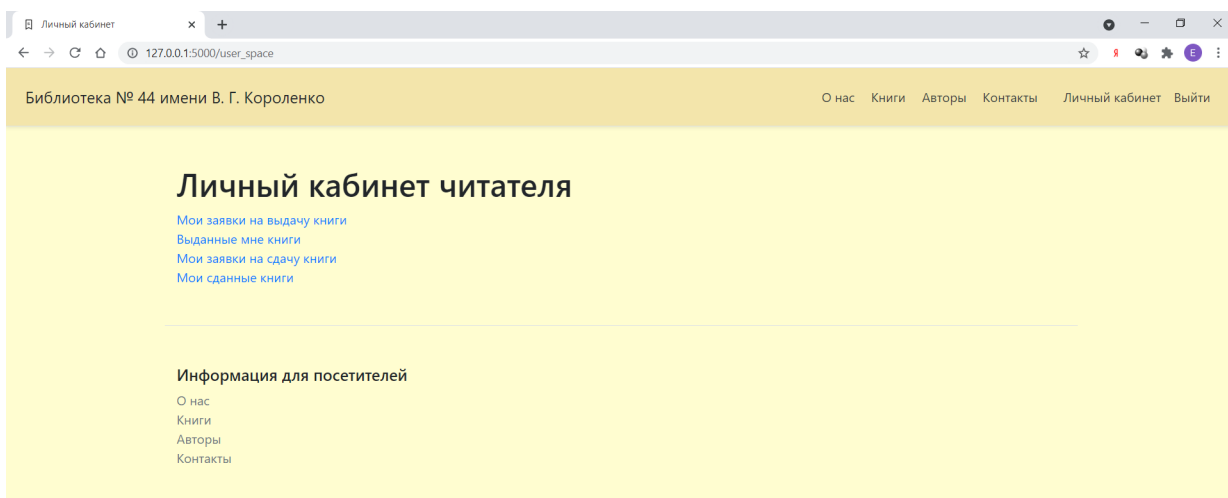


Рисунок 16 – Личный кабинет читателя.

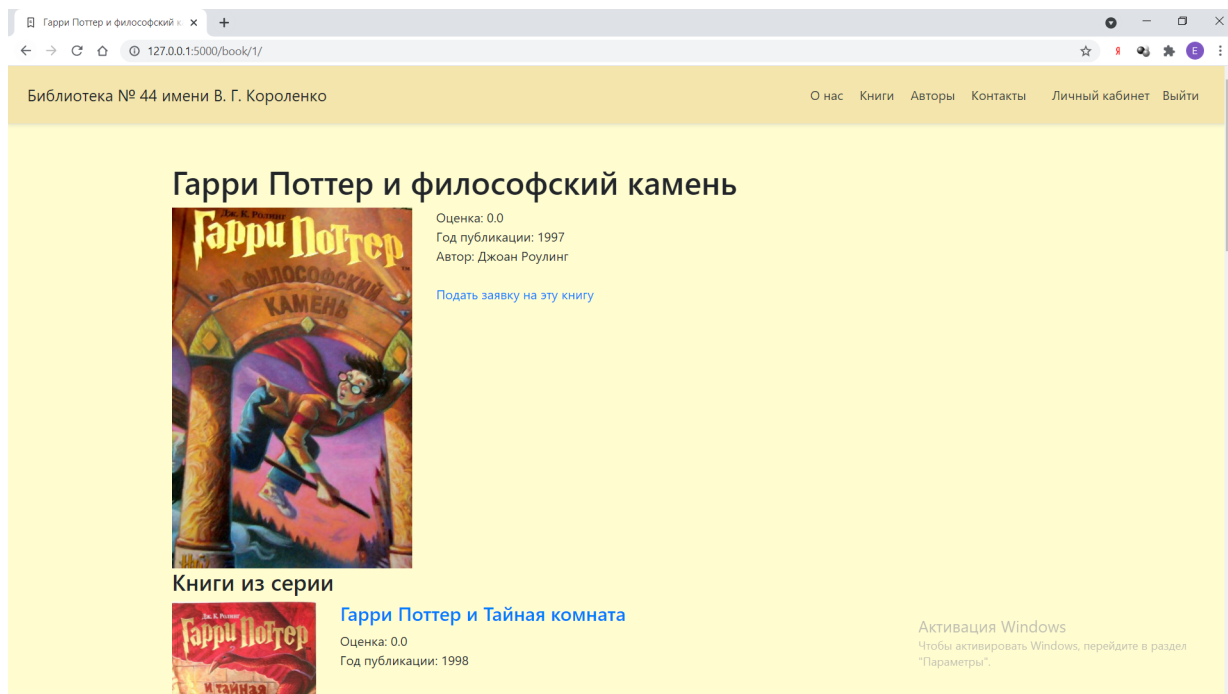


Рисунок 17 – Страница книги.

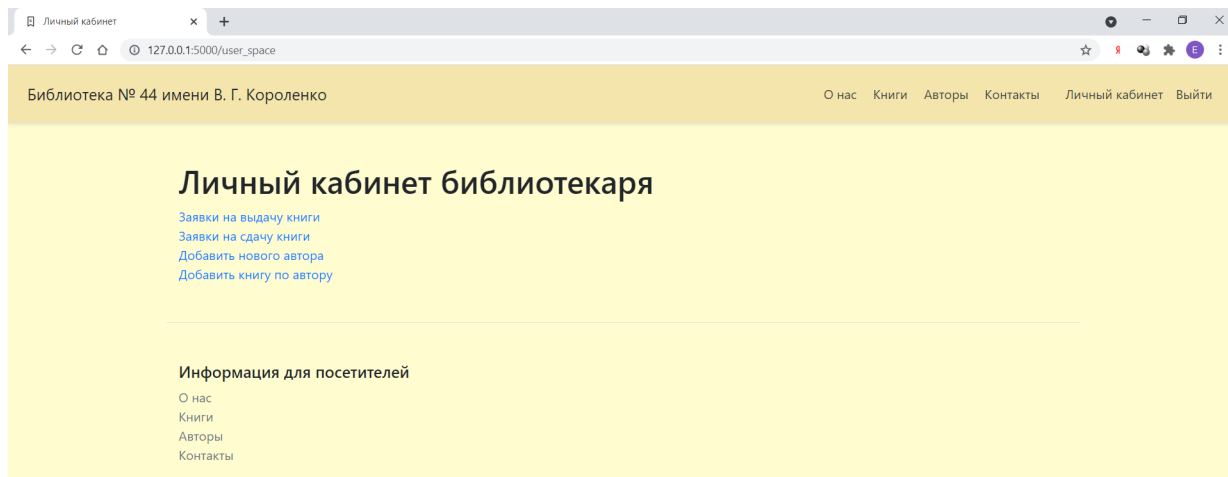


Рисунок 18 – Личный кабинет библиотекаря.

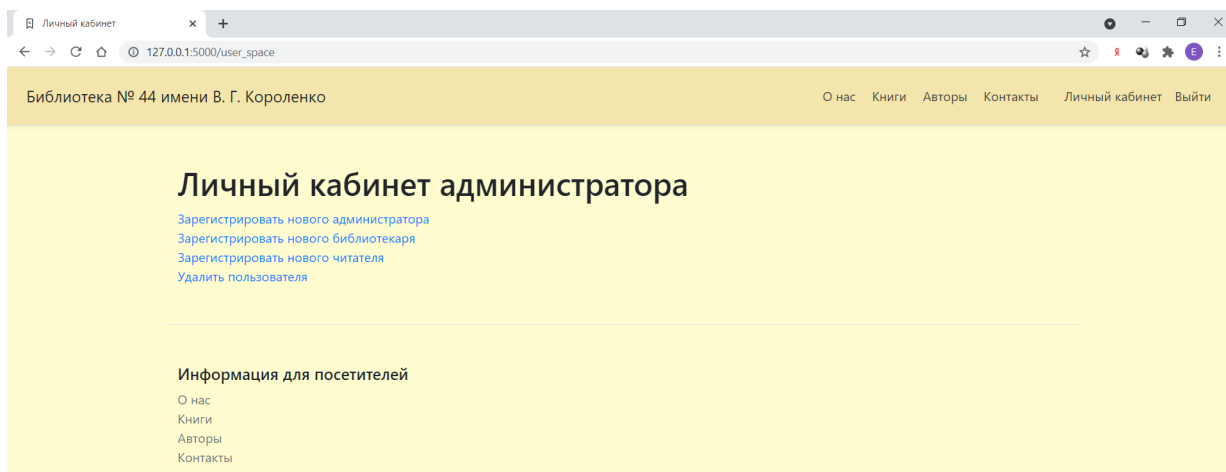


Рисунок 19 – Личный кабинет администратора.

3.8 Вывод

В данном разделе были выбраны средства реализации поставленной задачи, создана база данных и описана ролевая модель на уровне БД, разработан о приложение.

Заключение

Цель курсовой работы достигнута: разработаны база данных и приложение для библиотеки. Все задачи решены:

- 1) задание формализовано, акторы и их функционал выделены;
- 2) проведен анализ СУБД и выбрана наиболее подходящая;
- 3) спроектирована база данных;
- 4) спроектирована архитектура приложения;
- 5) разработано приложение.

Был получен опыт разработки базы данных.

Список литературы

- [1] PostgreSQL : Документация [Электронный ресурс] URL:
<https://postgrespro.ru/docs/postgresql> (дата обращения: 20.09.2021);
- [2] Flask : Документация [Электронный ресурс] URL:
<https://flask.palletsprojects.com/en/1.1.x/> (дата обращения: 20.09.2021);
- [3] Python : Документация [Электронный ресурс] URL:
<https://www.python.org/doc/> (дата обращения: 20.09.2021);
- [4] Bootstrap : Документация [Электронный ресурс] URL:
<https://getbootstrap.com/> (дата обращения: 20.09.2021);

Курсовой проект на тему: «Создание приложения для общественной библиотеки»

Исполнитель:
Гришин Егор Борисович, ИУ7-65Б

Научный руководитель:
Волкова Лилия Леонидовна

Москва
2021

Цель и задачи

Цель: разработка базы данных и приложения для библиотеки.

Задачи:

- формализовать задание, выделив соответствующих акторов и их функционал;
- провести анализ СУБД и выбрать наиболее подходящую;
- спроектировать базу данных;
- спроектировать архитектуру приложения;
- разработать приложение.

ER-диаграмма

Выделенные сущности:

- книга - book;
- автор - author;
- формуляр - reading;
- читатель - reader;
- данные пользователя системы - user_data.

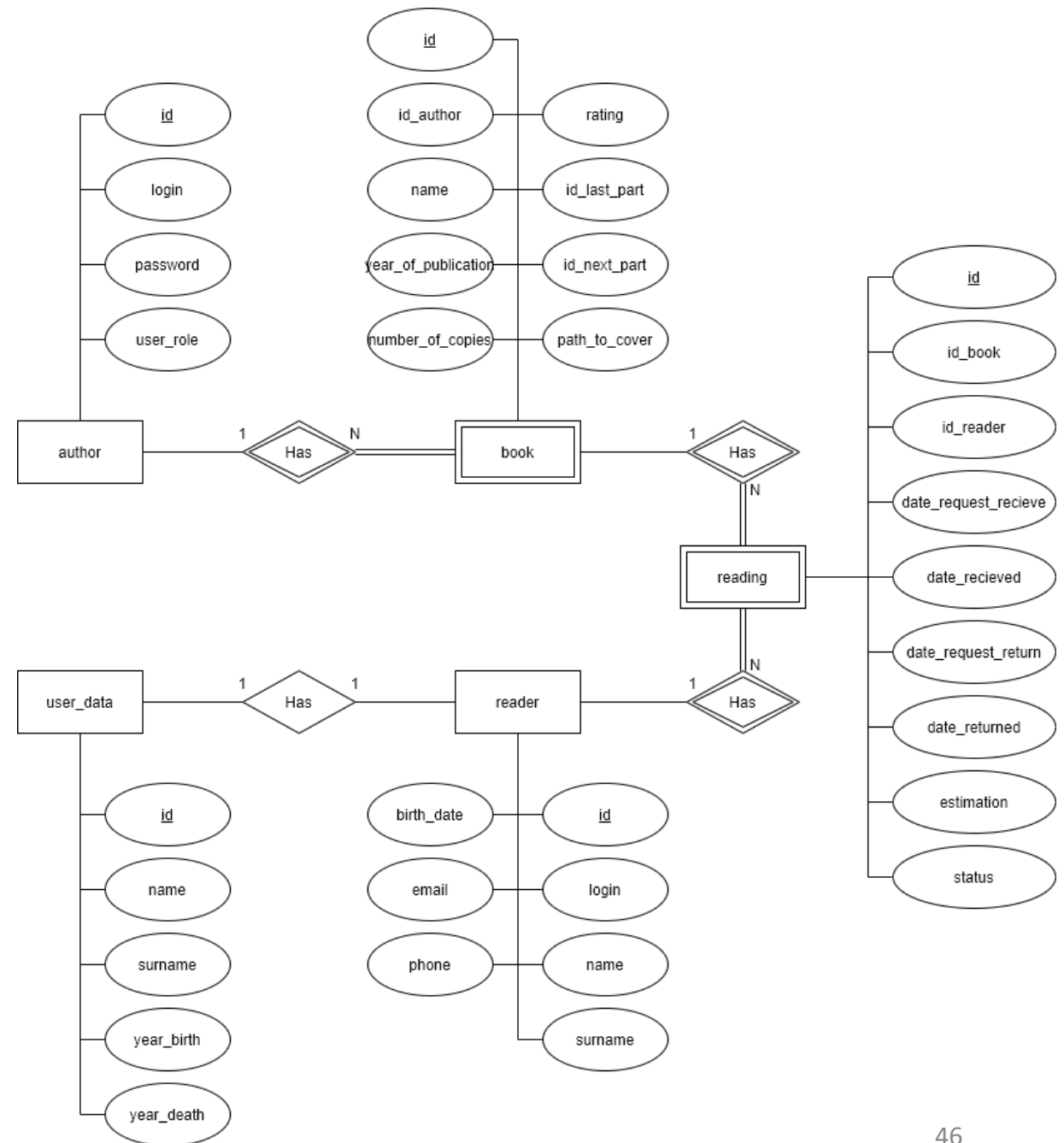
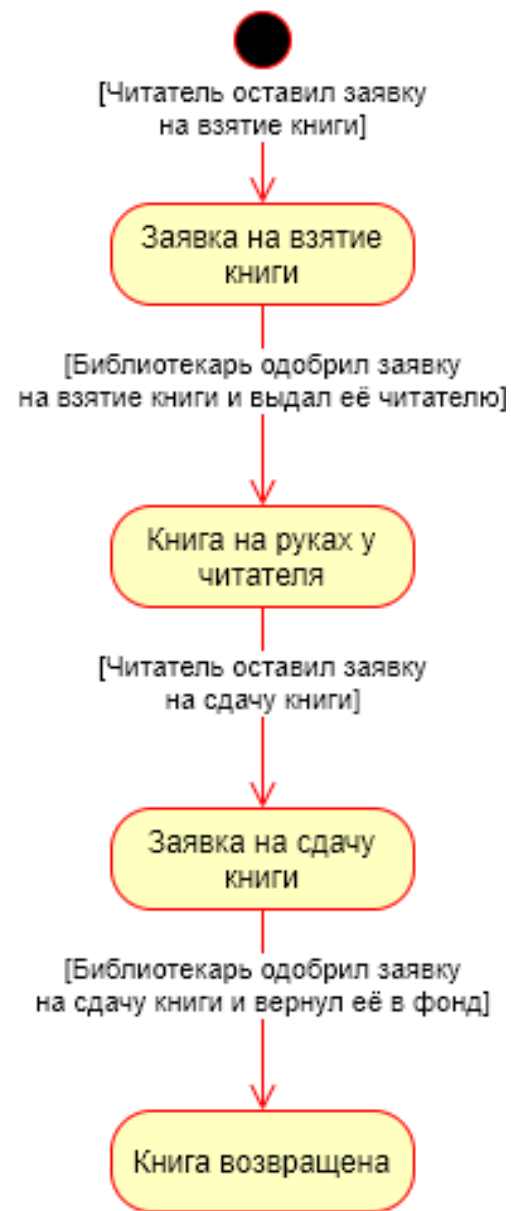


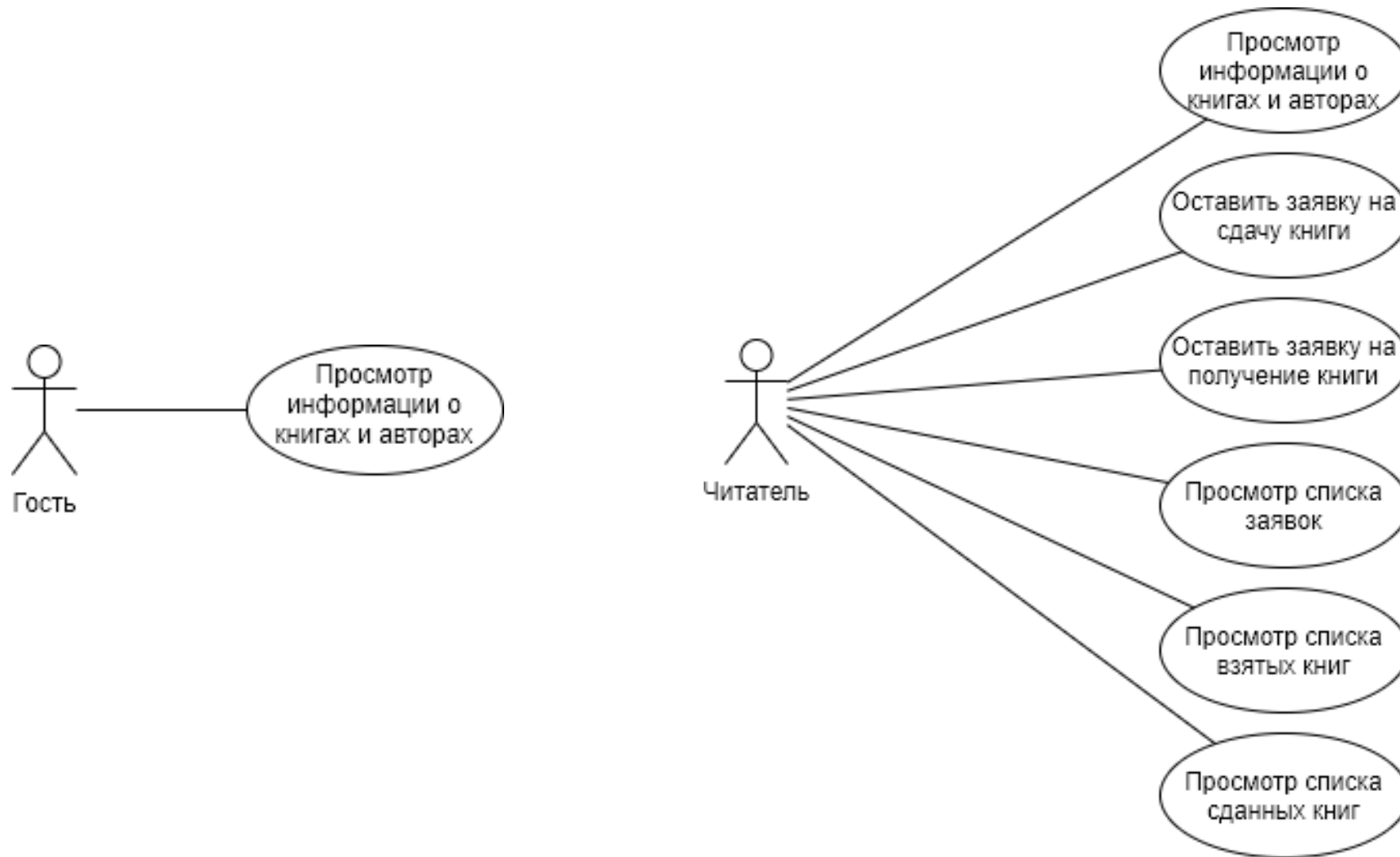
Диаграмма состояний формуляра



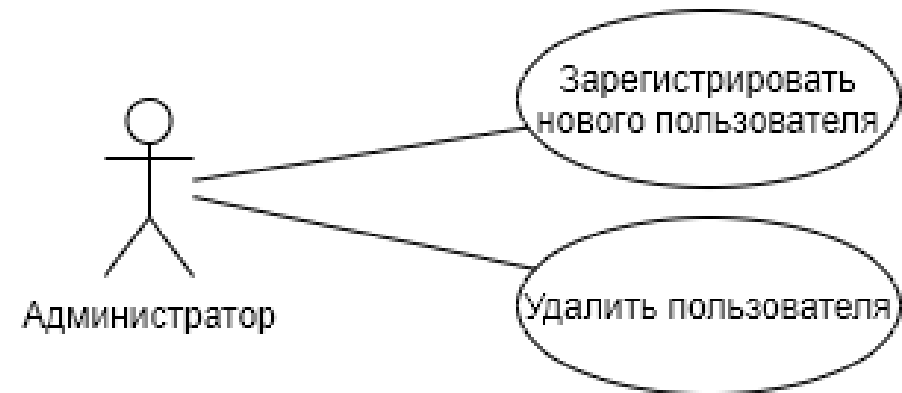
Пользователи системы

- Гость — это неавторизованный пользователь;
- Читатель - это авторизованный пользователь. Он может подать заявку на взятие книги, если она у него не на руках, и на её сдачу в противном случае. Также ему доступны списки книг, на которые он оставил заявки, которые у него на руках и которые он уже прочел и сдал;
- Библиотекарь - это авторизованный пользователь, контролирующий выдачу книг. Он одобряет заявки читателей, и добавляет новые книги и авторов в БД;
- Администратор является авторизованным пользователем с повышенным уровнем полномочий — он может регистрировать новых пользователей и удалять существующих.

Use-Case диаграммы гостя и читателя



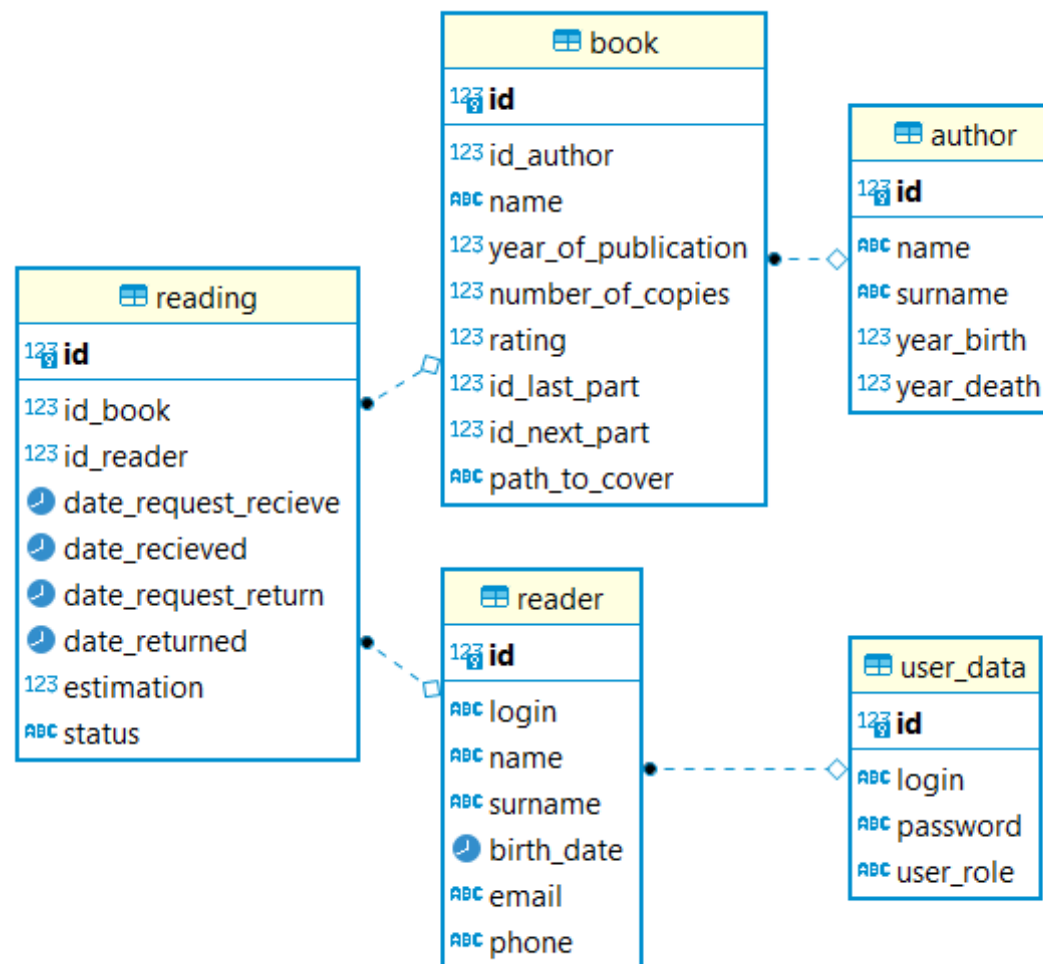
Use-Case диаграммы библиотекаря и администратора



Выбор СУБД

	Oracle	MySQL	Microsoft SQL Server	PostgreSQL
Простота в использовании	+	+	+	+
Бесплатная	-	+	+	+
Безопасность данных	+	-	+	+
Поддержка стандарта SQL	+	+	+	+
Поддержка хранимых процедур и триггеров	+	+	+	+
Кроссплатформенность	+	+	-	+

Диаграмма БД



Спроектированные триггеры

- Триггер поддержания серийности книг;
- Триггер поддержания оценки книг;
- Триггер поддержания количества книг в фонде.

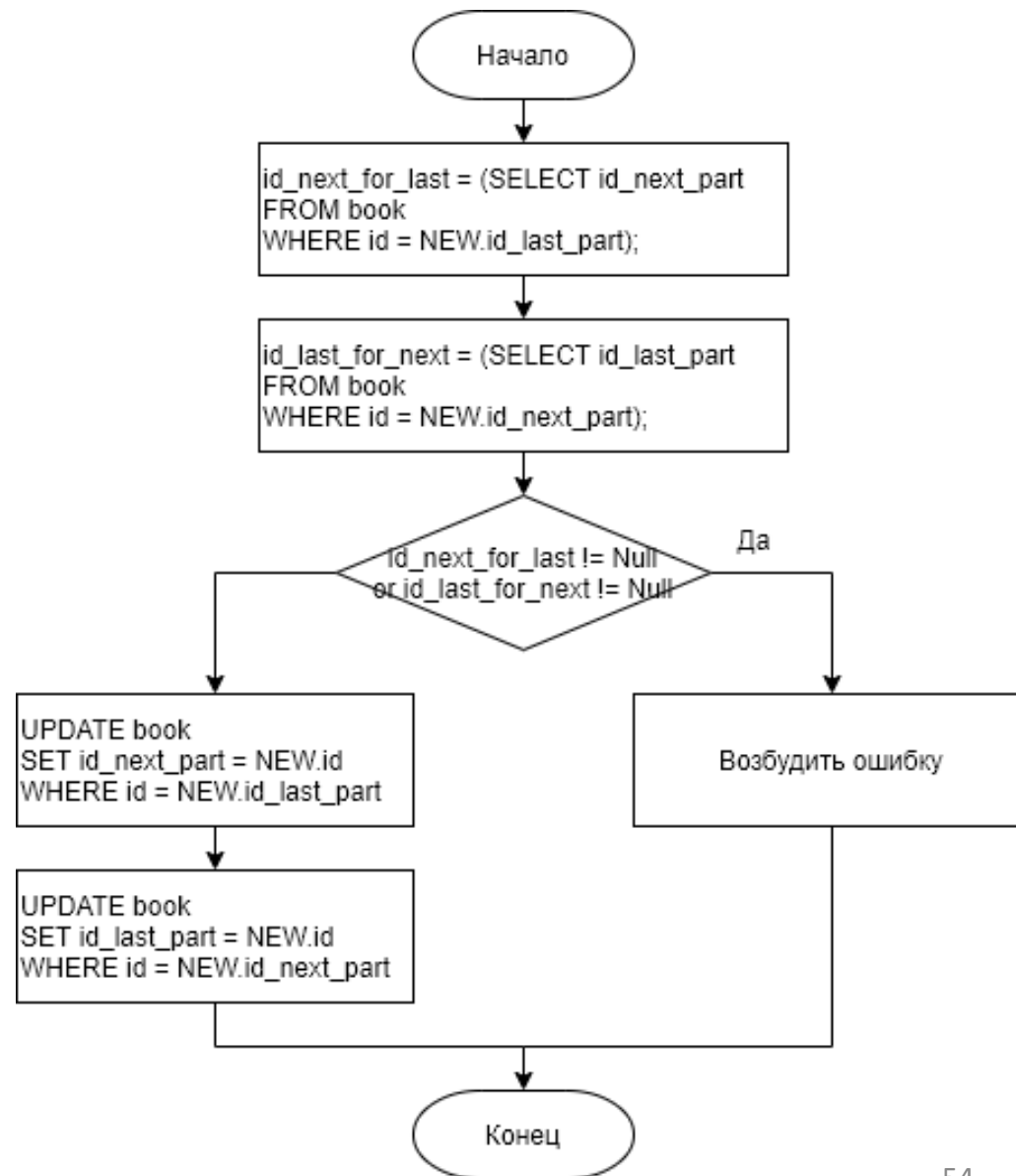
Триггер поддержания серийности книг

При добавлении новой книги срабатывает данный триггер.

Триггер обеспечивает целостность связи частей серии.

Библиотекарю будет показано сообщение об ошибке, ему нужно будет повторить попытку добавления книги, проставив верные ссылки.

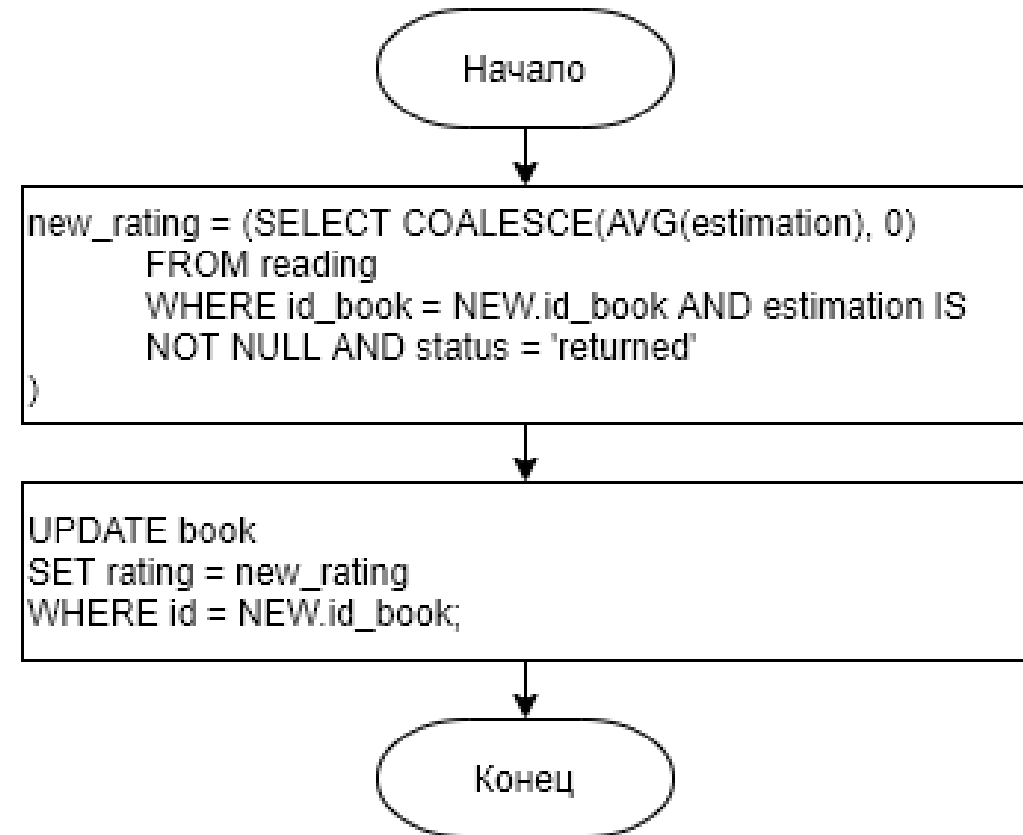
Ограничение триггера: книги добавляются только в начало или в конец серии.



Триггер поддержания оценки книг

При возврате книги, в соответствующей записи в таблице библиотечных карточек reading полю status присваивается значение 'returned' и срабатывает данный триггер.

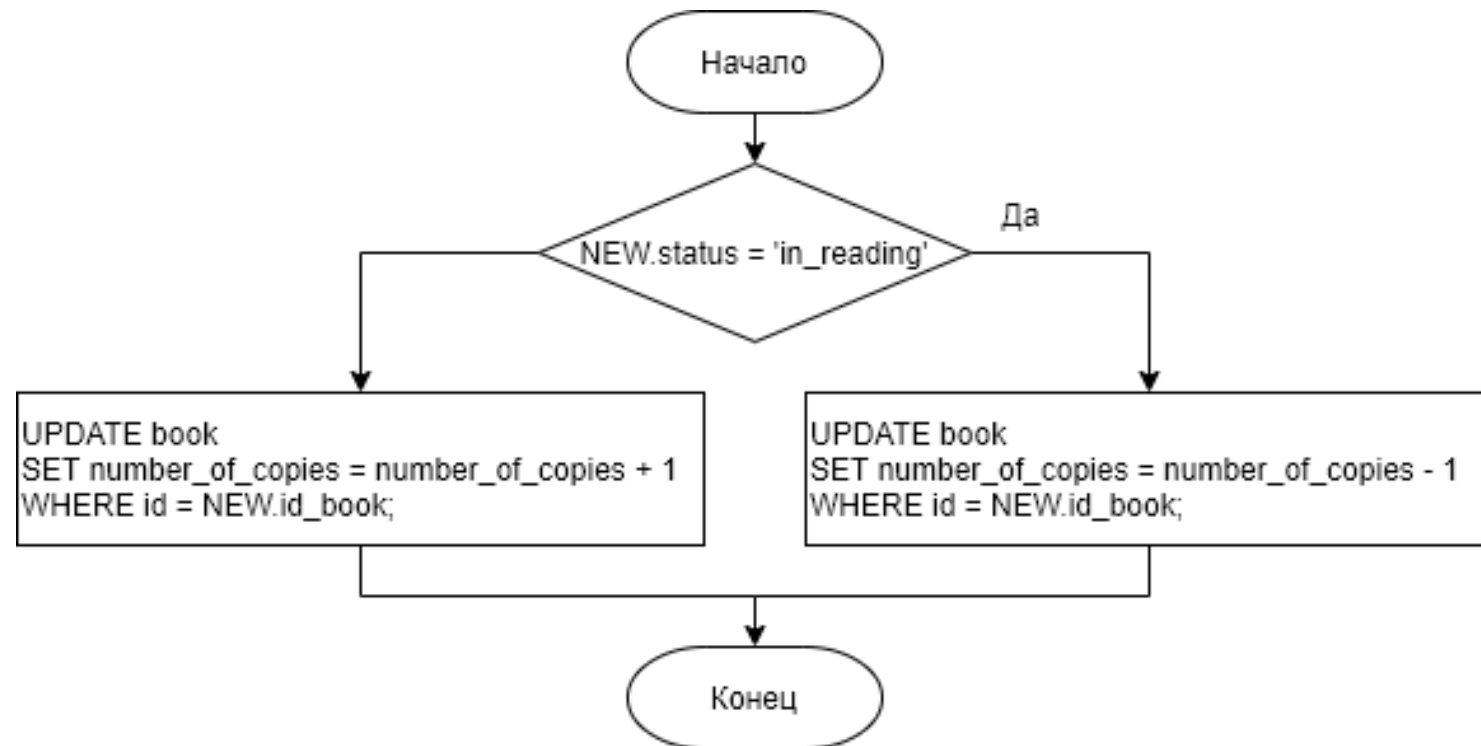
При возврате книги ей может быть дана оценка - estimation. На основании всех данных оценок триггер вычисляет рейтинг книги, обеспечивая его актуальность.



Триггер поддержания количества книг в фонде

При выдаче или сдаче книги, в соответствующей записи в таблице reading полю status присваивается значение 'in_reading' или 'returned' соответственно. При обоих событиях срабатывает этот триггер.

Триггер обеспечивает актуальность количества оставшихся в фонде экземпляров книги.



Сложный запрос выбора всех книг серии

Запрос выдает информацию о
всех книгах из серии, в которую
входит книга с id = ID_BOOK

```
WITH RECURSIVE last_parts AS (  
    SELECT *  
    FROM book  
    WHERE id = <ID_BOOK>  
    UNION ALL  
    SELECT book.*  
    FROM last_parts JOIN book ON last_parts.id_last_part =  
        book.id  
) , next_parts AS (  
    SELECT *  
    FROM book  
    WHERE id = <ID_BOOK>  
    UNION ALL  
    SELECT book.*  
    FROM next_parts JOIN book ON next_parts.id_next_part =  
        book.id  
) , res AS (  
    SELECT * FROM last_parts  
    UNION  
    SELECT * FROM next_parts  
)  
SELECT *  
FROM res  
WHERE id != <ID_BOOK>;
```

Архитектура приложения

Для разработки программы выбран язык Python по причине совмещения нескольких парадигм программирования, а также из-за большого разнообразия представленных библиотек и фреймворков для создания веб-приложения.

Для создания данного проекта в качестве инструмента, который облегчит процесс создания веб-приложения, был выбран фреймворк Flask. Благодаря его простоте и гибкости, разработчик может сам выбрать способ реализации тех или иных задач.

Пример обработки заявок библиотекарем

Для каждой заявки на взятие книги проверяется её наличие в фонде. Если книги в наличии нет, то библиотекарю не будет отображаться кнопка одобрения данной заявки.

Заявки на выдачу книг

Заявка №2

Читатель: Анна Шишихина

Книга: Организация переворота с помощью табакерки

Количество оставшихся экземпляров: 1

[Одобрить заявку](#)

Заявка №1

Читатель: Егор Гришин

Книга: Организация переворота с помощью табакерки

Количество оставшихся экземпляров: 1

[Одобрить заявку](#)

Заявки на выдачу книг

Заявка №1

Читатель: Егор Гришин

Книга: Организация переворота с помощью табакерки

Количество оставшихся экземпляров: 0

Одобрить заявку временно не возможно, т.к. все экземпляры данной книги выданы

Заключение

Цель курсовой работы достигнута: разработаны база данных и приложение для библиотеки. Все задачи решены:

- задание формализовано, акторы и их функционал выделены;
- проведен анализ СУБД и выбрана наиболее подходящая;
- спроектирована база данных;
- спроектирована архитектура приложения;
- разработано приложение.

Получен опыт разработки базы данных.