Оглавление

[1. Аналитический раздел 2](#_Toc133706572)

[1.1. Описание системы 2](#_Toc133706573)

[1.2. Существующие аналоги 2](#_Toc133706574)

[1.3. Общие требования к системе 2](#_Toc133706575)

[1.4. Требования к функциональным характеристикам 3](#_Toc133706576)

[1.5. Входные данные 3](#_Toc133706577)

[1.7. Функциональные требования к системе с точки зрения пользователя 4](#_Toc133706578)

[1.8. Схема вычисления рейтинга пользователя 5](#_Toc133706579)

[1.9. Выходные данные 5](#_Toc133706580)

[1.10. Требования к программной реализации 6](#_Toc133706581)

[1.11. Топология системы 7](#_Toc133706582)

[1.12. Общие требования к подсистемам 8](#_Toc133706583)

[1.13. Функциональные требования к сервисам 8](#_Toc133706584)

[2. Конструкторский раздел 11](#_Toc133706585)

[2.1. Концептуальный дизайн 11](#_Toc133706586)

[2.2. Сценарии функционирования системы 13](#_Toc133706587)

[2.3. Диаграмма прецедентов 15](#_Toc133706588)

[2.4. Спецификация сценариев 17](#_Toc133706589)

[2.5. Логический дизайн 18](#_Toc133706590)

[2.6. Диаграмма последовательности действий 24](#_Toc133706591)

[2.7. Диаграмма потоков данных 25](#_Toc133706592)

[2.8. Архитектура системы 26](#_Toc133706593)

[3. Технологический раздел 27](#_Toc133706594)

[3.1. Схема базы данных 27](#_Toc133706595)

[3.2. Сборка и развертывание системы 27](#_Toc133706596)

[3.3. Тестирование 28](#_Toc133706597)

[Заключение 29](#_Toc133706598)

[Список литературы 30](#_Toc133706599)

1. Аналитический раздел
   1. Описание системы

Система должна представлять из себя портал для поиска книги и ее бронирования. Если у пользователя на руках есть уже некоторое ограниченное количество книг, то он не может взять новую, пока не сдал старые. Если пользователь возвращает книги в хорошем состоянии и сдает их в срок, то максимальное допустимое количество книг у него на руках увеличивается.

Главное назначение разрабатываемой системы – предоставить пользователю возможность посмотреть наличие книг в библиотеке и при необходимости зарезервировать, для дальнейшего использования. Также приложение поможет контролировать прилежность и ответственность пользователя, отслеживая своевременность возврата и состояние книги.

* 1. Существующие аналоги

Среди аналогов можно отметить порталы «Детская библиотека Букля» и VOTUM-book. Данный проект должен иметь следующие преимущества перед существующими аналогами:

* Поиск информации во многих библиотеках, а не в единственной.
* Поиск актуальной информации о точном местоположении библиотек.
* Удобный интерфейс и высокая скорость загрузки страниц портала.
  1. Общие требования к системе

1. Время восстановления системы после сбоя не должно превышать 15 минут;
2. Обеспечить безопасность работоспособности системы за счет отказоустойчивости узлов;
3. Каждый узел должен автоматически восстанавливаться после сбоя;
4. В случае недоступности некритичного функционала, должна осуществляться деградация функциональности;
   1. Требования к функциональным характеристикам

Медиана времени отклика системы на действия пользователя должна быть менее 950мс, задержках между взаимодействующими сервисами менее 320мс и одновременном числе работающих пользователей менее 100 на каждый сервер, обслуживающий внешний интерфейс.

* 1. Входные данные

1. Пользователь

* Идентификатор пользователя;
* ФИО, не более 80 символов;
* Email;

1. Библиотека:

* Идентификатор библиотеки;
* Город;
* Название;
* Адрес;

1. Книга:

* Идентификатор книги;
* Название;
* Автор;
* Жанр;
* Состояние;
* Количество экземпляров;

1. Реквизиты

* Город;
* Адрес библиотеки;
* Название библиотеки;
* Автор;
* Жанр;
* Название книги;
* Состояние книги;
* Количество экземпляров книги;
  1. Входные данные администратора
* Идентификатор
* ФИО, не более 80 символов.
* email
  1. Функциональные требования к системе с точки зрения пользователя

Портал должен обеспечивать реализацию следующих функций:

* + - 1. Система должна обеспечивать разделение пользователей на две роли:
* клиент;
* администратор;
  + - 1. Система должна обеспечивать регистрацию пользователей с валидацией вводимых данных через интерфейс приложения.
      2. Система должна обеспечивать аутентификацию пользователей.
      3. Система должна предоставлять **клиенту** следующие функции:
* Просмотр списка библиотек (названий и адресов) в городе.
* Просмотр по реквизитам списка книг в выбранной библиотеке.
* Получить информацию по бронирования пользователя (идентификатор книги, идентификатор библиотеки, имя пользователя, дата резервации, дата, когда требуется вернуть книгу, статус бронирования).
* Зарезервировать книгу.
* Снять резервацию, вернув книгу.
* Узнать рейтинг пользователя.
  + - 1. Система должна предоставлять **администратору** следующие функции:
* Редактирование списка библиотек.
* Возможность просмотра статистики.
  1. Схема вычисления рейтинга пользователя

Количество звезд пользователя определяем максимальное количество одновременно арендованных книг. У пользователя может быть от 1 до 100 звезд, если изменение выходит за эти границы, то устанавливается граничное значение. Начальное значение равняется 75. Если книгу вернули позднее срока или ее состояние на момент выдачи отличается от состояния, в котором ее вернули, то у пользователя уменьшается количество звезд на 10 за каждое условие (сдача позднее срока и в плохом состоянии). Если книгу вернули в исходном состоянии и в срок, то рейтинг пользователя увеличивается на 1 звезду.

* 1. Выходные данные

Выходными параметрами системы являются веб-страницы. В зависимости от запроса Пользователя они содержат:

* Список библиотек в городе (идентификатор библиотеки, название, адрес).
* Список книг оп реквизитам в выбранной библиотеке (идентификатор книги, название, автор, жанр, состояние, количество экземпляров).
* Информацию по всем взятым в прокат книгам пользователя (идентификатор книги, идентификатор библиотеки, имя пользователя, дата резервации, дата, когда требуется вернуть книгу, статус бронирования).
* Рейтинг пользователя (количество звезд).
* Информацию о статистике использования системы (количество запросов, время запроса, вид запроса).
  1. Требования к программной реализации

1. Все сервисы и фронтэнд должны быть запущены изолированно друг от друга;
2. Требуется использовать СОА (сервис-ориентированную архитектуру) для реализации системы;
3. Система состоит из микросервисов. Каждый микросервис отвечает за свою область логики работы приложения;
4. Взаимодействие между сервисами осуществляется посредством HTTP запросов;
5. Данные сервисов должны храниться в базе данных. Каждый сервис взаимодействует только со своей схемой данных. Взаимодействие сервисов происходит по технологии REST;
6. При недоступности систем портала должна осуществляться деградация функционала или выдача пользователю сообщения об ошибке;
7. Необходимо предусмотреть авторизацию пользователей, как через интерфейс приложения;
8. На сервисе, выполняющем функцию Identity Provider, создать протокол OpenID Connect;
9. Получение токена на Identity Provider и авторизацию пользователя реализовать посредством Authorization Flow;
10. Аутентификация должна быть реализована с помощью Json Web Token;
11. При создании пользовательского интерфейса необходимо использовать CSS;
12. С помощью helm charts произвести деплой сервисов в Managed Kubernetes Cluster;
13. Взаимодействие с внешней средой осуществлять посредством Ingress Controller;
    1. Топология системы

Система будет состоять из фронтэнда и пяти сервисов, что наиболее целесообразно для реализации ее основного назначения. Топология разрабатываемой системы представлена на рисунке 1.

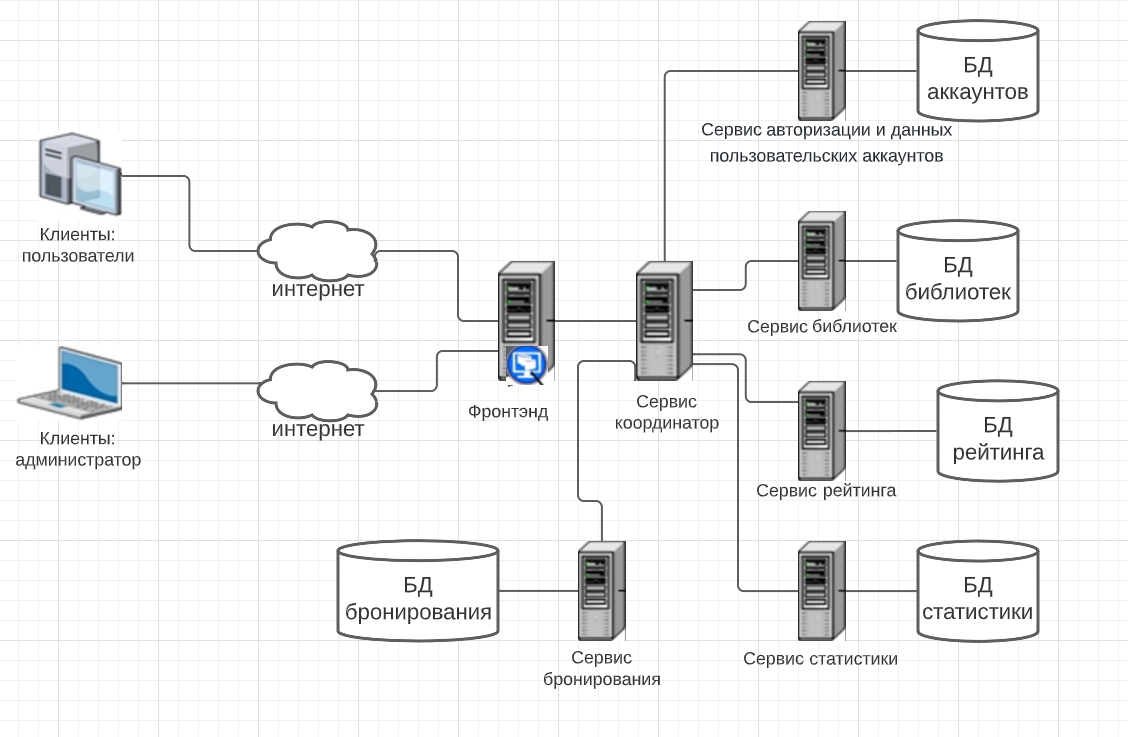


Рисунок 1 – Топология системы

Разрабатываемая система состоит из фронтэнда и 6 подсистем:

* Сервис авторизации и данных пользовательских аккаунтов;
* Сервис библиотек;
* Сервис рейтинга;
* Сервис статистики;
* Сервис бронирования;
* Сервис-координатор;
  1. Общие требования к подсистемам

Фронтэнд – серверное приложение, при разработке которого следует учесть следующие нюансы:

* Фронт-энд должен принимать запросы по протоколу HTTP и формировать ответ пользователю в формате веб-страниц;
* Фронт-энд является посредником между пользователями, передавая их запросы последовательно на сервис агрегации запросов;

К реализации бэкэндов должны быть предъявлены следующие требования:

* Прием и возврат данных должен происходить в формате JSON по протоколу HTTP;
* Если результаты работы сервиса необходимо сохранять в базе данных, то доступ к ней должен осуществляться по протоколу HTTP. Доступ к базе данных может осуществляться только из подсистем, работающих напрямую с данными ее таблиц
  1. Функциональные требования к сервисам

1. Сервис авторизации и данных пользовательских аккаунтов

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Идентификатор пользователя;
* Имя;
* Email;
* Пароль;
* Роль;

Сервис должен реализовывать следующий функционал:

* Регистрация пользователя;
* Аутентификация пользователя;
* Авторизация пользователя;
* Выход из системы.

1. **Сервис библиотек** отвечает за хранение данных библиотек и получение информации о них:

В базе данных хранятся несколько сущностей, ассоциированных с сервисом:

* Библиотека. Имеет следующие обязательные поля:
  + Идентификатор библиотеки;
  + Название;
  + Город;
  + Адрес;
* Книга. Имеет следующие обязательные поля:
  + Идентификатор книги;
  + Название;
  + Автор;
  + Жанр;
  + Состояние;
* Книги в библиотеке. Имеет следующие обязательные поля:
  + Идентификатор библиотеки;
  + Идентификатор книги;
  + Количество экземпляров книги;

Сервис должен реализовывать следующий функционал:

* Получить список библиотек в городе;
* Получить список книг в выбранной библиотеке;
* Получить информацию о библиотеке;
* Получить информацию о книге;

1. **Сервис рейтинга** отвечает за хранение информации о рейтинге пользователя, изменение его показателей, и получение информации о нем.

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Имя;
* Показатель рейтинга;

Сервис должен реализовывать следующий функционал:

* Увеличить рейтинг пользователя (согласно функции вычисления);
* Уменьшить рейтинг пользователя (согласно функции вычисления);
* Получить информацию о рейтинге пользователя;

1. Сервис статистики

Сервис должен реализовывать следующий функционал:

* Получать статистику о выполненных действиях;
* Строить отчет по пришедшим данным;

1. **Сервис бронирования** контролирует действия пользователей с книгами и предоставляет информацию о взятых в прокат сборниках.

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Идентификатор бронирования;
* Имя пользователя;
* Идентификатор книги;
* Идентификатор библиотеки;
* Статус бронирования;
* Дата взятия в прокат;
* Дата, когда следует вернуть книгу;

Сервис должен реализовывать следующий функционал:

* Взять книгу;
* Вернуть книгу;
* Получить информацию по всем взятым в прокат книгам пользователя;

1. **Сервис координатор -** предоставляет пользовательский интерфейс и внешний API системы.

Сервис должен реализовывать следующий функционал:

* Регистрация пользователя;
* Аутентификация пользователя;
* Авторизация пользователя;
* Выход из системы;
* Получение списка библиотек в городе;
* Получение списка книг в выбранной библиотеке;
* Взять книгу в библиотеке;
* Вернуть книгу в библиотеку;
* Получить рейтинг пользователя;
* Просмотр статистики (только для администратора);

1. Конструкторский раздел
   1. Концептуальный дизайн

На рисунке 2 отображена контекстная диаграмма верхнего уровня, которая обеспечивает наиболее общее или абстрактное описание работы системы. Данный вид диаграммы позволяет формализовать описание запросов пользователя и ответов системы на данные запросы, отобразив систему в виде “черного” ящика

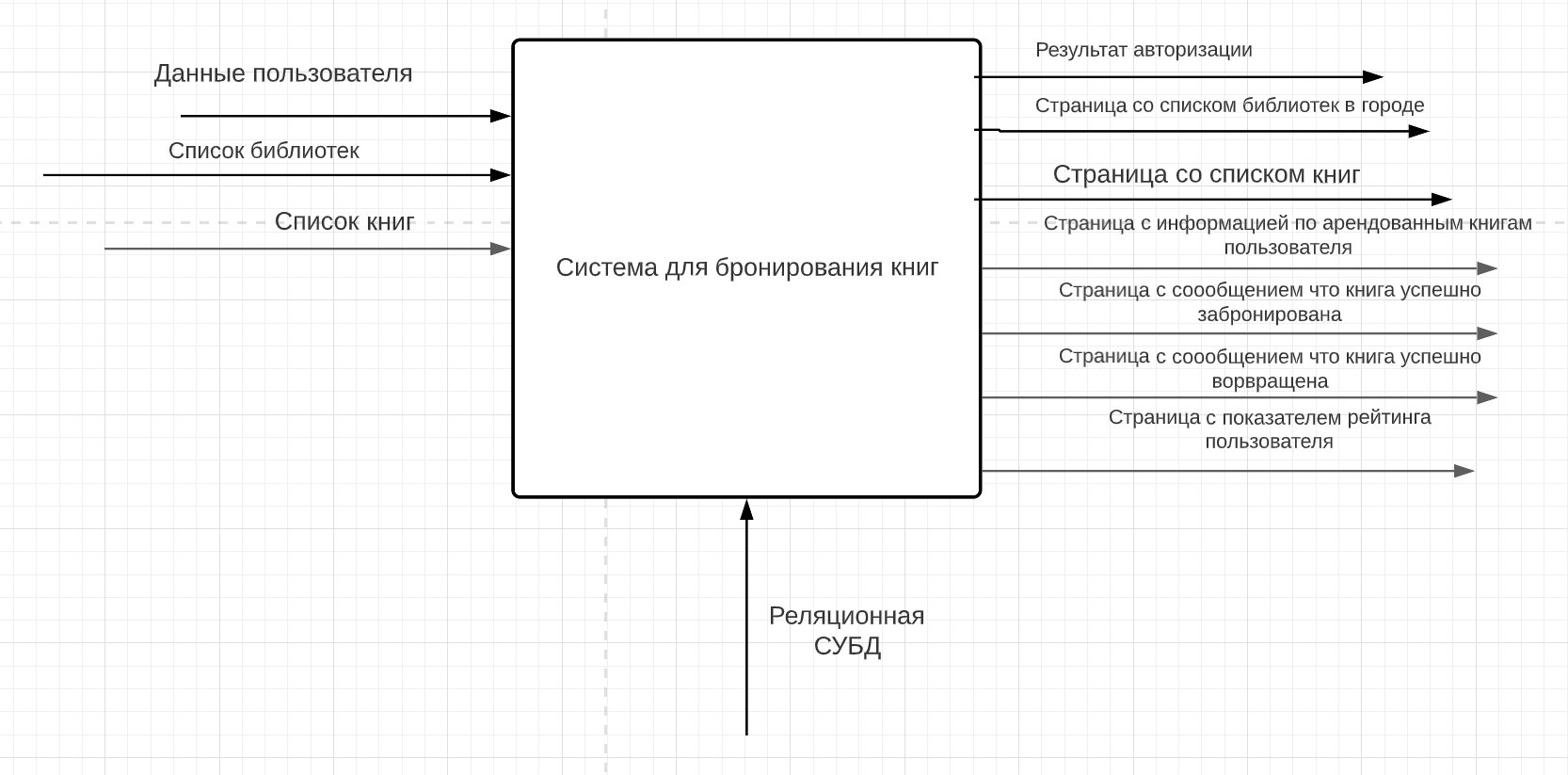


Рисунок 2. Концептуальная модель системы в нотации IDEF0.

Для уточнения деталей работы системы применяется декомпозиция функций,

отображенных на диаграмме верхнего уровня, при помощи создания дочерних

диаграмм. В качестве примера на рисунке 3 изображена дочерняя диаграмма, которая определяет последовательность выполнения операций в системе при обработке запроса пользователя на бронирование книги.

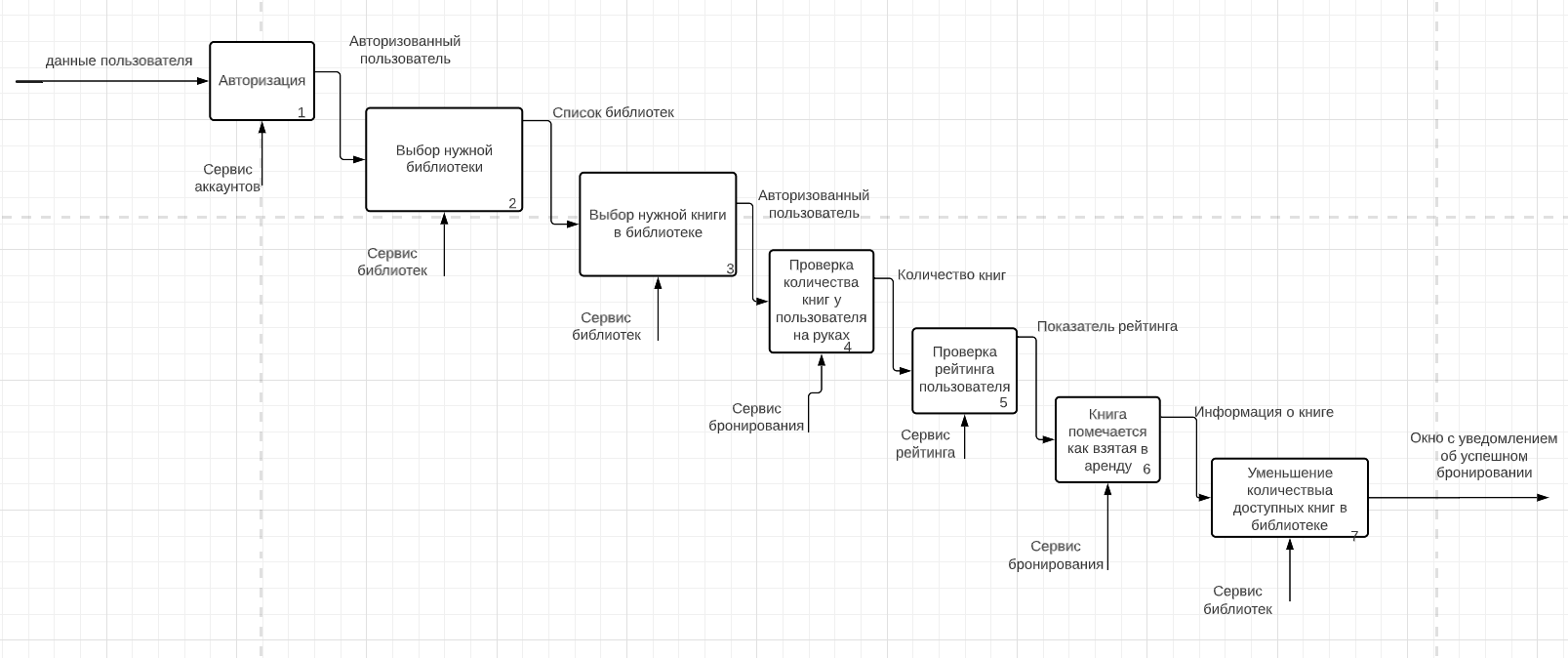


Рисунок 3. Детализированная концептуальная модель системы в нотации IDEF0 для бронирования книги.

* 1. Сценарии функционирования системы

Сценарии функционирования или использования системы описывают конкретную последовательность действий, иллюстрирующую поведение пользователя при работе с приложением. Далее приведены подробные сценарии основных 33 возможных действий пользователя.

Регистрация пользователя:

1. Пользователь нажимает на кнопку «Войти» в интерфейсе приложения;

2. Пользователь перенаправляется на страницу авторизации, которая содержит поля для заполнения его данных;

3. Пользователь вводит данные в форму и для завершения регистрации нажимает на кнопку «Регистрация», тем самым подтверждая верность своих данных, а также согласие на их обработку и хранение.

4. Если пользователь с введенным для регистрации именем уже существует, то клиент перенаправляется на страницу ошибки. При успешной регистрации пользователь перенаправляется на страницу своего профиля в системе

Авторизация пользователя:

1. Пользователь нажимает на кнопку «Войти» в интерфейсе приложения;

2. Пользователь перенаправляется на страницу авторизации, которая содержит поля для заполнения его логина и пароля;

3. Пользователь завершает работу с формой авторизации нажатием кнопки «Войти»;

4. При обнаружении ошибки в данных, пользователь перенаправляется на страницу ошибки; при совпадении данных с записью в базе данных аккаунтов пользователь получает доступ к системе.

Получение списка библиотек в городе:

1. Авторизованный пользователь нажимает на кнопку «Библиотеки»;

2. Пользователь перенаправляется на страницу, содержащую поле для выбора города и после нажатия на «Поиск» получает список библиотек.

Получить список книг в выбранной библиотеке:

1. Авторизованный пользователь после выбора библиотеки нажимает на кнопку «Показать книги»;

2. Пользователь перенаправляется на страницу, со списком книг и фильтрам для поиска по реквизитам.

**Получить информацию по всем взятым в прокат книгам пользователя:**

1. Авторизованный пользователь нажимает на кнопку «Показать арендованные книги»;

2. Пользователь перенаправляется на страницу, со списком действующих бронирований.

**Взять книгу в библиотеке:**

1. Авторизованный пользователь после выбора библиотеки и книги нажимает на кнопку «Взять книгу»;

2. Проверяется информация о рейтинге и бронированиях пользователя и, если все в норме пользователь получает сообщение что книга успешно забронирована.

3. Если операция не завершится успешно пользователю покажется ошибка (о превышении лимита книг на руках) и он будет перенаправлен на главную страницу.

**Вернуть книгу:**

1. Авторизованный пользователь в списке бронирований, напротив соответствующего, нажимает на кнопку «Вернуть книгу»;

2. Пользователь получает сообщение что книга успешно возвращена.

**Узнать рейтинг пользователя:**

1. Авторизованный пользователь нажимает на кнопку «Узнать рейтинг»;

2. Пользователь перенаправляется на страницу, с полем содержащим показатель рейтинга.

**Получение** статистики:

1. Пользователь с ролью администратор нажимает на кнопку “Панель администратора”.

2. Пользователь перенаправляется на страницу просмотра статистики времени обработки запросов.

* 1. Диаграмма прецедентов

В системе выделены 2 основных роли: пользователь и администратор, диаграммы прецедентов для этих ролей изображены на рисунках 4 и 5.

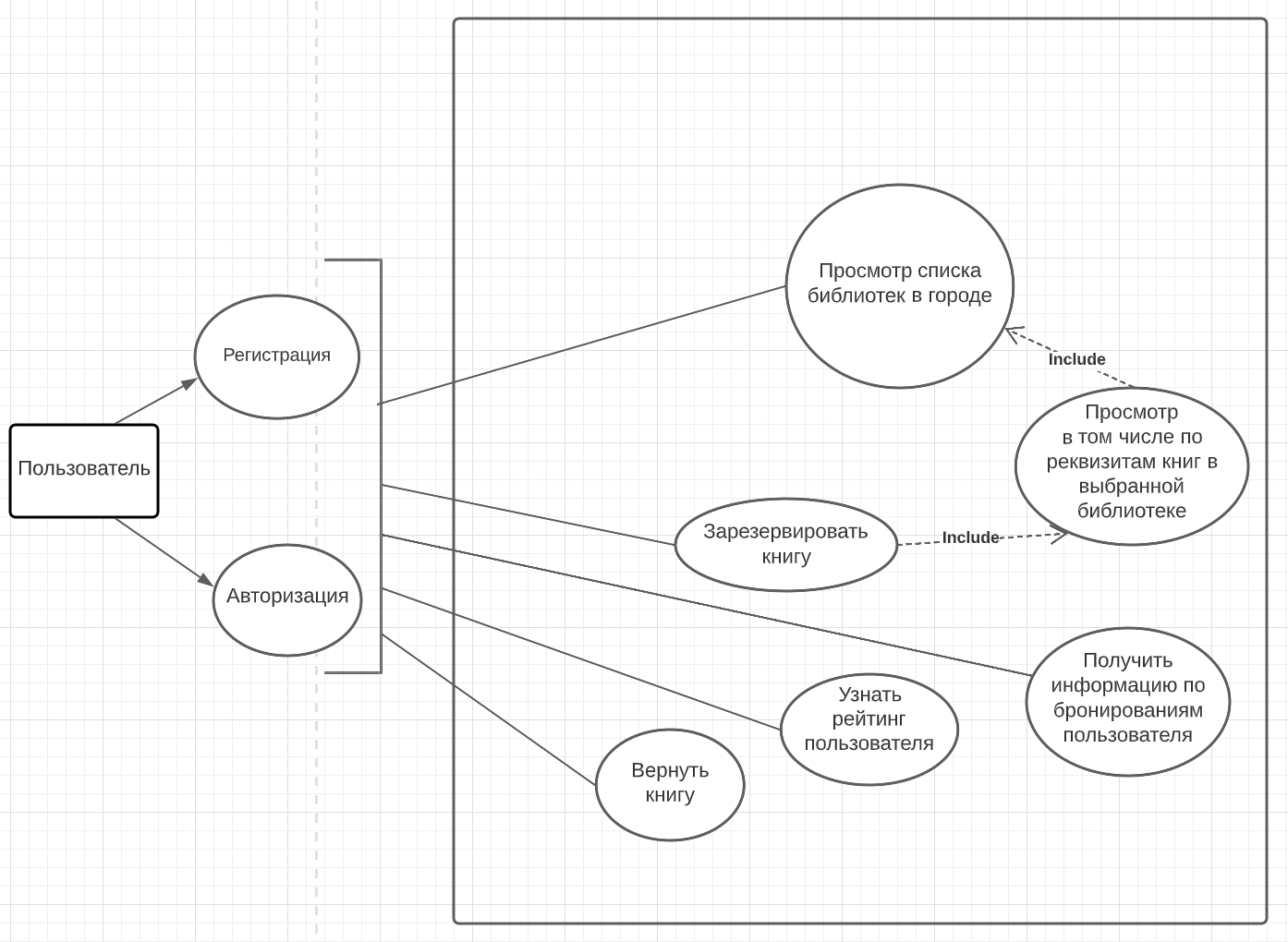


Рисунок 4. Диаграмма прецедентов с точки зрения пользователя.

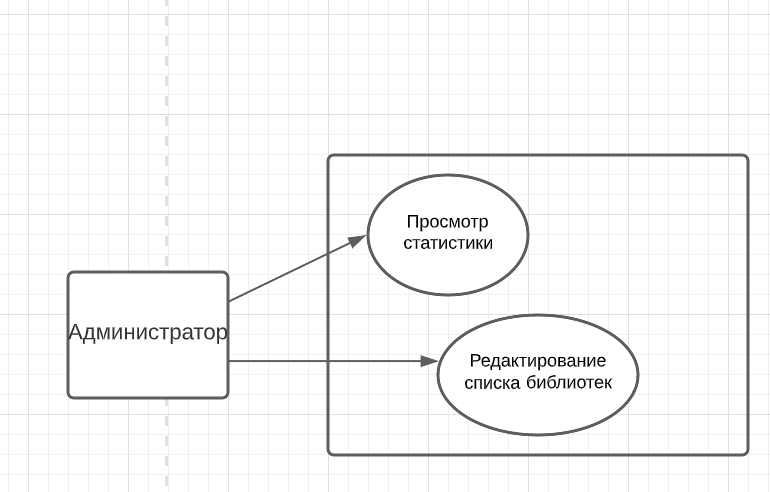


Рисунок 5. Диаграмма прецедентов с точки зрения администратора.

* 1. Спецификация сценариев

Приведенные сценарии могут иметь как основной поток выполнения, который выполняется чаще всего, так и альтернативные потоки, описывающие выполнение запроса при отклонении от основного хода сценария. Все возможные ходы выполнения сценария описываются при помощи спецификаций. Примеры спецификаций для описанных выше сценариев приведены в данном разделе.

**Спецификация сценария «Бронирование книги»**

|  |  |
| --- | --- |
| Нормальный ход сценария | |
| Действие пользователя | Отклик системы |
| Пользователь вводит название города и нажимает на кнопку «Найти библиотеку». | На странице появляется список библиотек в указанном городе. |
| Пользователь нажимает на кнопку «Просмотреть список книг в библиотеке». | На странице появляется список книг в библиотеке. |
| Пользователь нажимает на кнопку «Взять книгу»; | Появляется окно с уведомлением что книга успешно забронирована. |

|  |  |
| --- | --- |
| Альтернативный ход сценария | |
| Действие пользователя | Отклик системы |
| Пользователь вводит название города и нажимает на кнопку «Найти библиотеку». | Открывается окно с сообщением что в данном городе библиотек не найдено. |

|  |  |
| --- | --- |
| Альтернативный ход сценария | |
| Действие пользователя | Отклик системы |
| Пользователь вводит название города и нажимает на кнопку «Найти библиотеку». | На странице появляется список библиотек в указанном городе. |
| Пользователь нажимает на кнопку «Просмотреть список книг в библиотеке». | На странице появляется список книг в библиотеке. |
| Пользователь нажимает на кнопку «Взять книгу»; | Открывается окно с сообщением что достигнут лимит по количеству книг. |

* 1. Логический дизайн

На основе функциональных требований к выделенным подсистемам, а также объектов, о которых необходимо хранить данные в системе, была разработана схема данных приложения. Результат ее проектирования отображен на условной ER-диаграмме, представленной на рисунке 6. На данной схеме овалами обозначены ключевые сущности, а ромбами - связи между ними. Участие сущности в отношении с другой сущностью отмечается линией, соединяющей их. Число, располагающееся около линии, означает тип связи между соединенными сущностями.

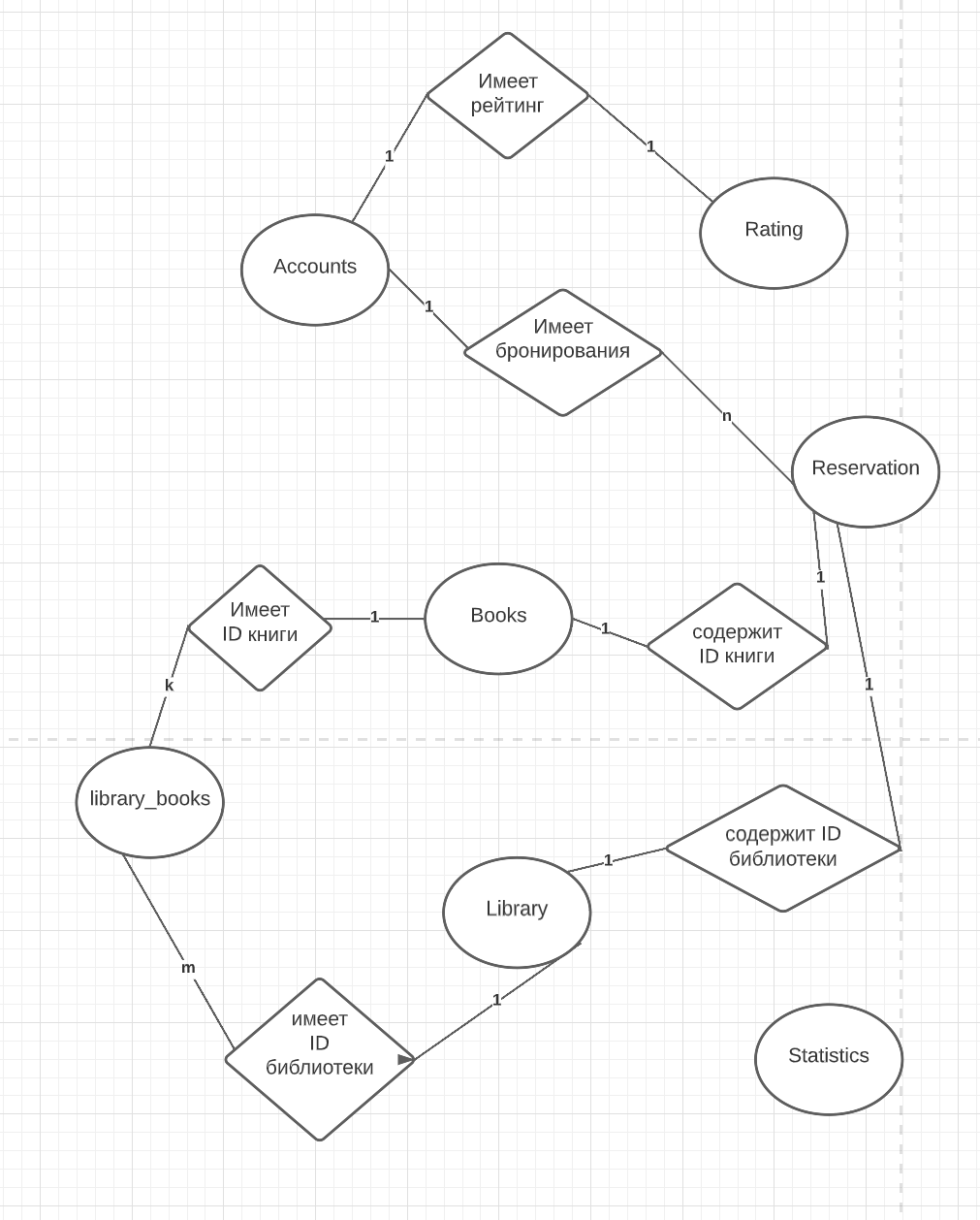


Рисунок 6. ER-диаграмма данных приложения.

Далее приводятся спецификации таблиц базы данных.

**Спецификация таблицы library**

Таблица library содержит данные о библиотеках

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| library\_id | Public : uuid | Идентификатор библиотеки |
| name | Public : VARCHAR(80) | Название библиотеки |
| city | Public : VARCHAR(255) | Город |
| address | Public : VARCHAR(255) | Адрес библиотеки |

**Спецификация таблицы books**

Таблица books содержит данные о книгах

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| book\_id | Public : uuid | Идентификатор книги |
| name | Public : VARCHAR(255) | Название книги |
| author | Public : VARCHAR(255) | Автор книги |
| genre | Public : VARCHAR(255) | Жанр книги |
| condition | Public : VARCHAR(20) | Состояние книги |

**Спецификация таблицы rating**

Таблица rating содержит данные о рейтинге пользователей

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| username | Public : VARCHAR(80) | Имя пользователя |
| stars | Public : int | Рейтинг пользователя |

**Спецификация таблицы library\_books**

Таблица library\_books содержит данные о наличии книг в библиотеках

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| book\_id | Public : uuid | Идентификатор книги |
| library\_id | Public : uuid | Идентификатор библиотеки |
| available\_count | Public : int | Число экземпляров книги |

**Спецификация таблицы reservation**

Таблица reservation содержит информацию о бронированиях

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| reservation\_uid | Public : uuid | Идентификатор бронирования |
| username | Public : VARCHAR(80) | Имя пользователя |
| book\_uid | Public : uuid | Идентификатор книги |
| library\_uid | Public : uuid | Идентификатор библиотеки |
| status | VARCHAR(20) | Статус бронирования |
| start\_date | Public : Date | Дата начала бронирования |
| till\_date | Public : Date | Дата окончания бронирования |

**Спецификация таблицы Accounts**

Таблица Accounts содержит данные о профиле пользователя

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| email | Public: string | Адрес почты пользователя |
| username | Public : VARCHAR(80) | Имя пользователя |
| password | Public: string | Пароль пользователя |
| role | Public: string | Роль (клиент или администратор) |

**Спецификация таблицы Statistics**

Таблица Statistics содержит данные о статистике

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| id | Public : int | Первичный ключ |
| request | Public : string | url запроса |
| answer\_time | Public: double | Время ответа |

**Структура сервиса бронирования**

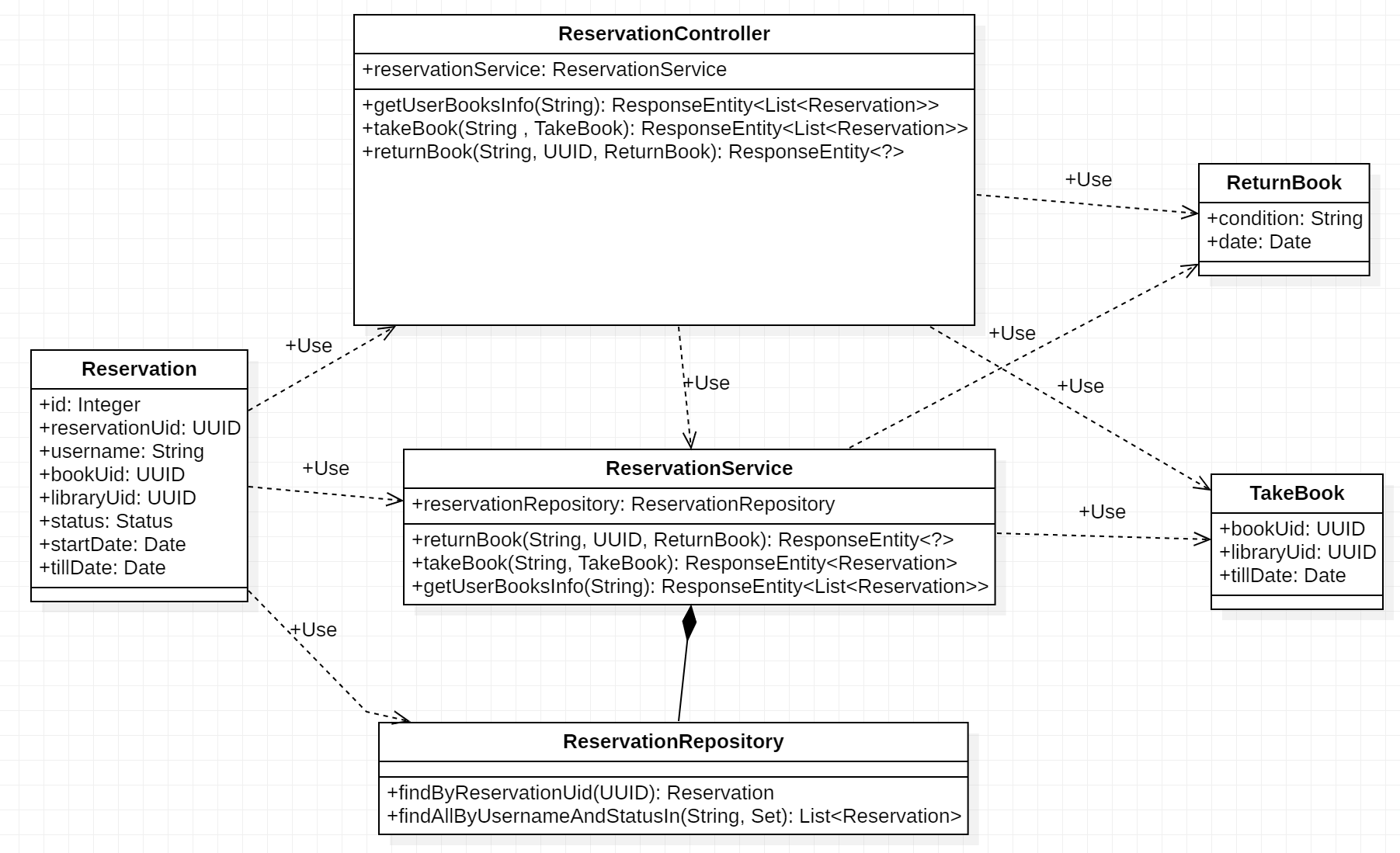


Рисунок 7. Диаграмма классов сервиса бронирования.

Функциональные требования, предъявляемые к сервису публикаций, реализуются при помощи методов контроллера NewsController. Далее приведено описание каждого метода данного контроллера.

**Спецификация класса ReservationController**

|  |  |
| --- | --- |
| Метод | Описание |
| getUserBooksInfo | Получить информацию по арендованным книгам пользователя. |
| takeBook | Взять книгу. |
| returnBook | Вернуть книгу. |

* 1. Диаграмма последовательности действий

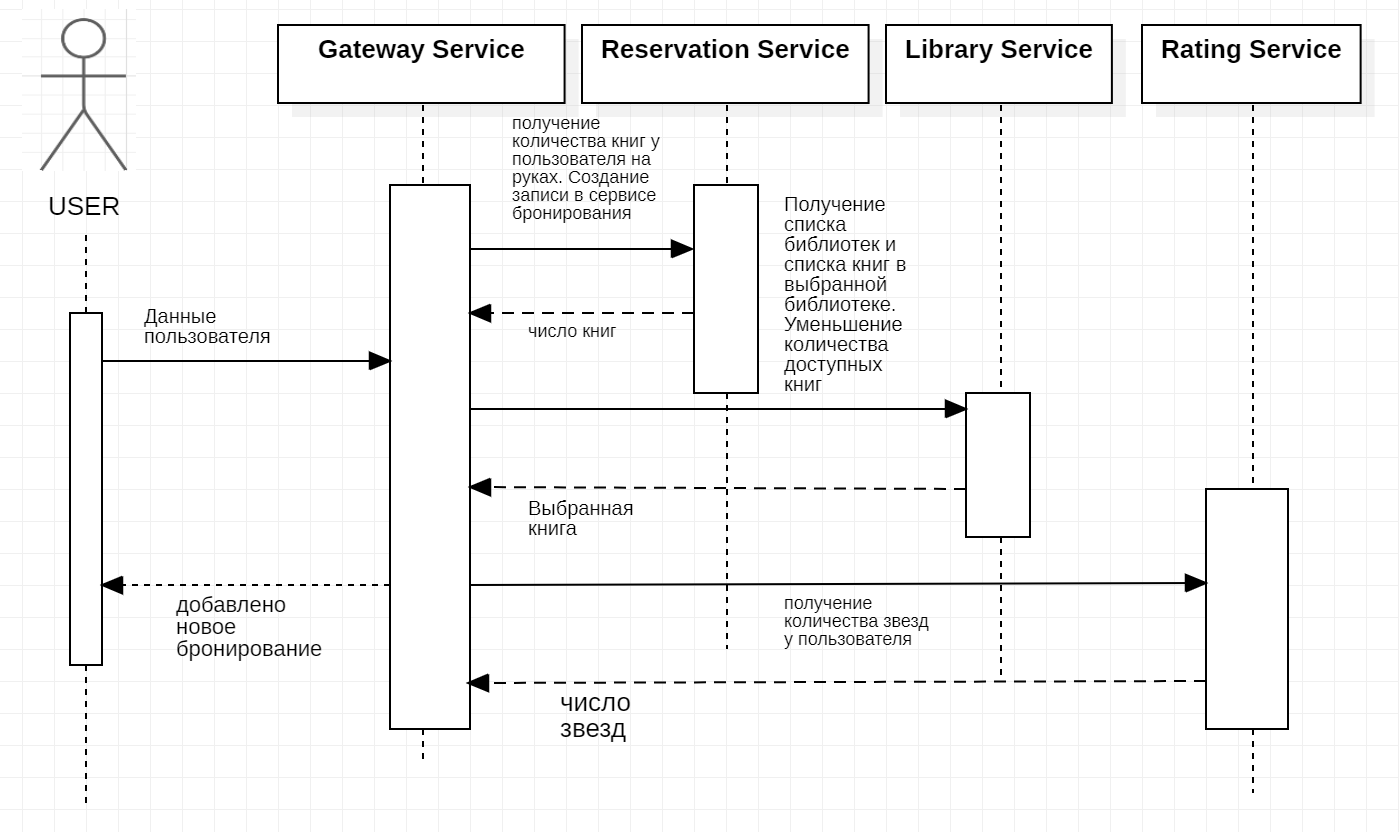


Рисунок 8. Диаграмма последовательности действий при бронировании книги.

Главный сервис приложения (Gateway) отправляет запрос на получение списка библиотек в городе и на получение списка книг в выбранной библиотеке.

Перед выдачей книги, запросом в сервис бронирования проверяется количество книг у пользователя на руках. Далее выполняется запрос в сервис рейтинга и запрашивается количество звезд. Количество звезд определяет максимальное количество книг, которые пользователь может одновременно взять в аренду.

Если условие выполнено, то создается запись в сервисе бронирования в статусе RENTED и в сервис библиотек уменьшает количество доступных книг.

* 1. Диаграмма потоков данных

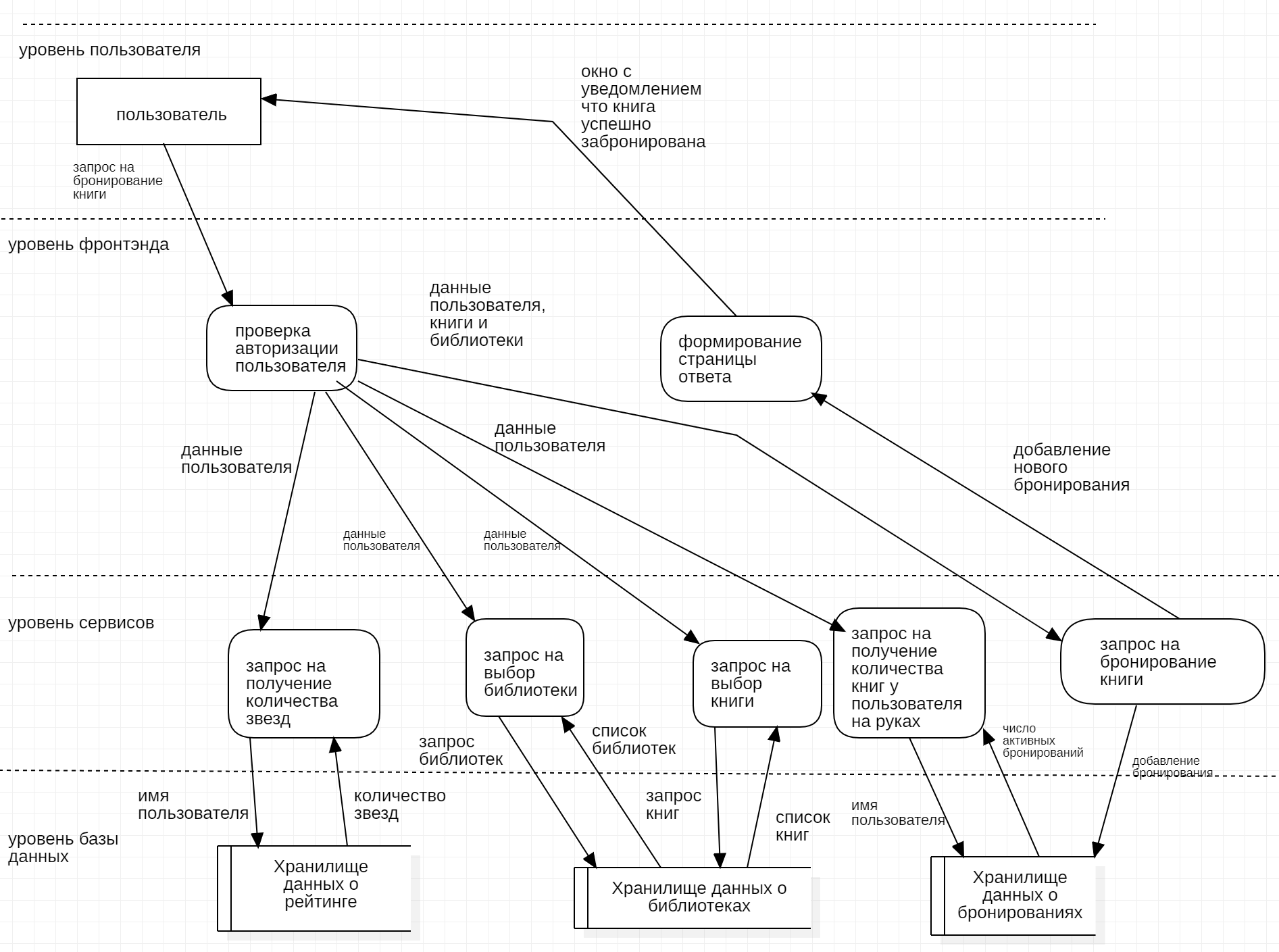


Рисунок 9. Диаграмма потоков данных при бронировании книги.

* 1. Архитектура системы

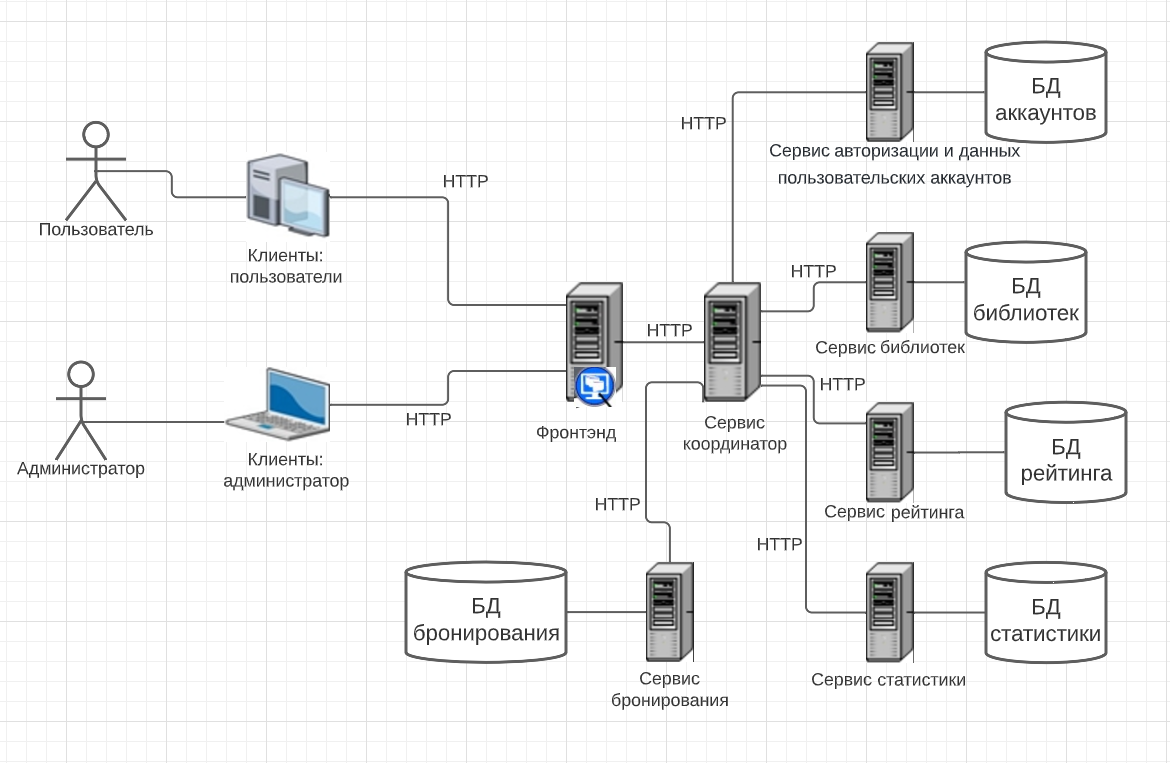


Рисунок 10. Архитектура разрабатываемой системы

1. Технологический раздел

На данном этапе разработки информационного портала будет произведено описание типов и структур данных в нотации IDEF1x, а также тестирование, обработка ошибок и поведение системы в случае отказа.

* 1. Схема базы данных

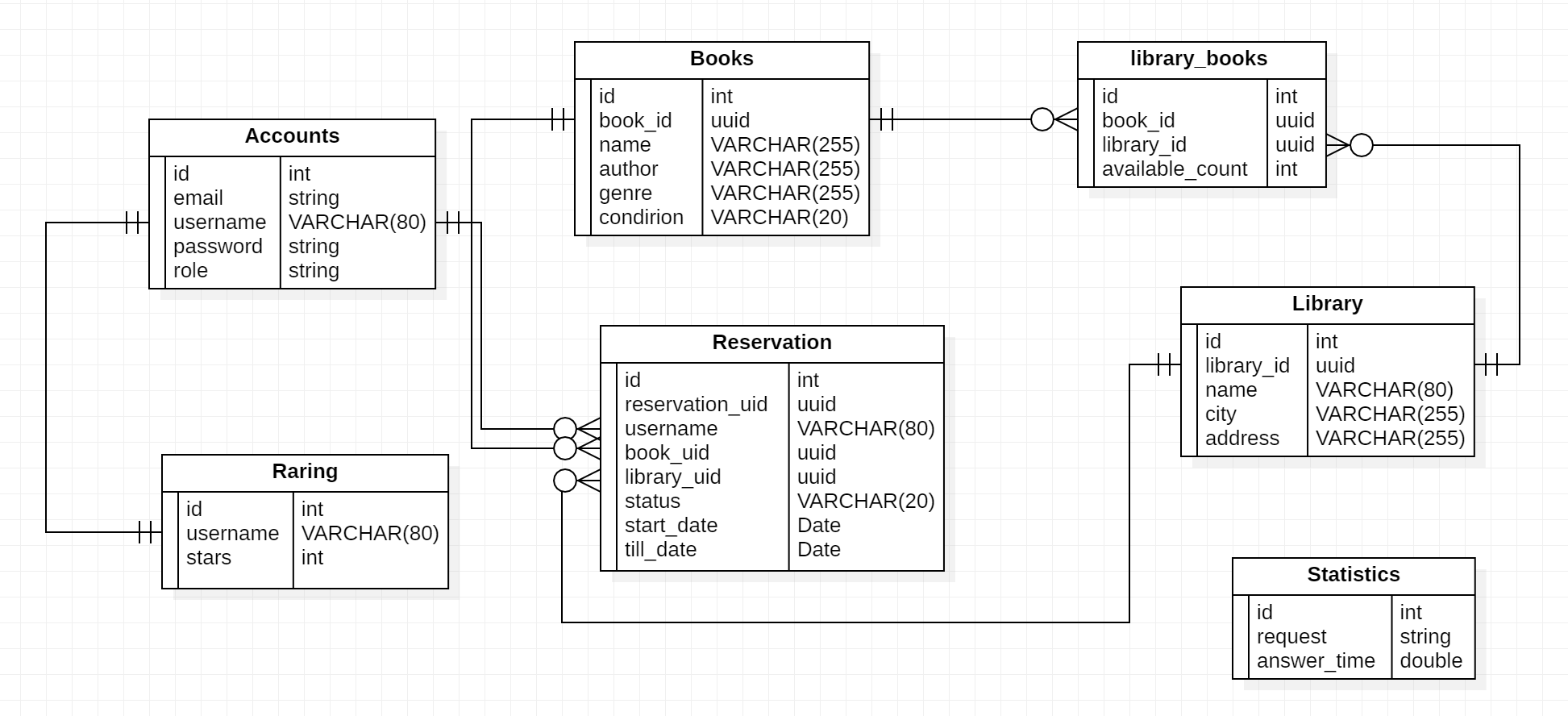


Рисунок 11. Схема базы данных

* 1. Сборка и развертывание системы

Процесс сборки и развертывания системы осуществляется с помощью CI/CD. После прохождения этапов сборки и тестирования сервисы приложения собираются в отдельные докер-контейнеры. Собранные контейнеры публикуются на Docker Hub.

Репозиторий с проектом содержит также файл docker compose, позволяющий развернуть систему локально.

* 1. Тестирование

Тестирование системы производилось методом черного ящика. Тестировались сервисы как по отдельности, так и вместе. В случае недоступности некритичного функционала осуществляется деградация функциональности. Прорабатывались различные сценарии, в том числе:

* Регистрация пользователя;
* Авторизация пользователя;
* Просмотр рейтинга пользователя;
* Просмотр истории бронирований пользователя;
* Просмотр списка библиотек;
* Просмотр списка книг в конкретной библиотеке;
* Бронирование книги;
* Возврат книги;
* Просмотр статистики;

Для тестирования отказоустойчивости отключались некоторые сервисы и проверялось поведение системы.

Заключение

В ходе работы была создана система для бронирования книг в библиотеке. В ее основе лежит микросервисная архитектура.

В первой части данной работы было сформировано описание разрабатываемой системы, её назначение и все необходимые требования для её проектирования и будущей реализации.

В конструкторском разделе, были подробно разобраны все этапы проектирования разрабатываемой архитектуры, определенны механизмы взаимодействия микросервисов между собой, а также описаны на разных уровнях с помощью диаграмм.

В технологическом разделе были описаны способы сборки, развертывания и тестирования системы.

Список литературы

1. Ньюмен С. Создание микросервисов. СПб.: Питер, 2016. 304 с.
2. JavaInUse [Электронный ресурс] // JWT. URL: https://www.javainuse.com/spring/boot-jwt (дата обращения: 16.04.2023).
3. Stackoverflow [Электронный ресурс] // What is secret key for JWT and how to generate it. URL: https://stackoverflow.com/questions/31309759/what-is-secret-key-for-jwt-based-authentication-and-how-to-generate-it (дата обращения: 22.04.2023).
4. Wikipedia [Электронный ресурс] // bcrypt. URL: https://ru.wikipedia.org/wiki/Bcrypt (дата обращения: 15.04.2023).
5. Testengineer [Электронный ресурс] // Что такое use case? Теория и примеры. URL: https://testengineer.ru/chto-takoe-use-case (дата обращения: 30.04.2023).
6. Wikipedia [Электронный ресурс] // Диаграмма последовательности. URL: https://ru.wikipedia.org/wiki/Диаграмма\_последовательности (дата обращения: 25.04.2023).
7. Maccase [Электронный ресурс] // UML Component. URL: https://maccase.ru/android/uml-diagramma-komponentov-opisanie-modelirovanie-na-uml-diagrammy.html (дата обращения: 30.04.2023).
8. Analyst [Электронный ресурс] // IDEF1x. URL: http://analyst.by/diagrams/logicheskaya-model-predmetnoy-oblasti (дата обращения: 20.04.2023).