Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 (ИУ)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка прототипа системы поиска и бронирования отелей на интересующие даты.

Студент группы **ИУ7-23М** _____ **Науменко А. А.**

Руководитель курсовой работы _____ **Ступников А. А.**

2025 г.

РЕФЕРАТ

Отчет 33 с., 1 кн., 11 рис., 6 источн.

РАСПРЕДЕЛЕННАЯ СИСТЕМА, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, БРОНИРОВАНИЕ ОТЕЛЕЙ, OAUTH, KUBERNETES, KAFKA.

Объект исследования — распределенная система. Предмет исследования — система бронирования отелей на интересующие даты.

Цель работы — разработать прототип системы поиска и бронирования отелей на интересующие даты.

В работе формулируются требования к системе и ее реализации. Приводится архитектура системы и описание ее функциональных сервисов. Представлены диаграммы прецендентов, последовательности действий, баз данных. Приведен дизайн интерфейса, особенности реализации, а также описан процесс сборки, развертывания и тестирования системы.

Результатом работы является разработанный прототип системы бронирования отелей на интересующие даты.

Область применения результатов — проектирование распределенных микросервисных систем, разработка системы бронирования и поиска отелей.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Системы бронирования отелей устраняют ключевую проблему туристического рынка — неэффективный поиск и оформление размещения. Они автоматизируют процесс бронирования, предоставляя туристам актуальную информацию о номерах и ценах, а отелям — удобный инструмент управления процессом бронирования. Такие системы упрощают взаимодействие туристов и гостиниц, способствуя развитию туристической отрасли в целом.

Распределенная система — это набор компьютерных программ, использующих вычислительные ресурсы нескольких отдельных вычислительных узлов для достижения одной общей цели. Распределенная система основывается на отдельных узлах, которые обмениваются данными и выполняют синхронизацию в общей сети. Распределенные системы направлены на устранение узких мест или единых точек отказа в системе. Для системы бронирования отелей данный подход является наиболее подходящим, поскольку позволяет обеспечить высокую доступность и надежность. Микросервисная архитектура обеспечивает масштабируемость и повышает безопасность системы в целом.

Цель работы — разработать прототип системы поиска и бронирования отелей на интересующие даты.

Необходимо выполнить следующие задачи.

1) Привести описание разрабатываемой системы, представить обзор существующих сервисов бронирования отелей, описать требования к разрабатываемой распределенной системе.

2) Спроектировать и описать архитектуру системы, а также выделенные в ней сущности. Представить схему взаимодействия сервисов с помощью диаграммы последовательности действий.

3) Разработать программное обеспечение, описать особенности реализации, процесс сборки, развертывания и тестирования системы.

1 Аналитический раздел

1.1 Описание системы

Портал должен представлять собой распределенную систему для бронирования отелей на интересующие даты. Он должен предоставлять пользователям возможность просматривать информацию о зарегистрированных в систему отелях, бронировать их на интересующие даты, просматривать бронирования с возможностью отмены тех, дата заселения в которые еще не наступила. Также пользователь должен иметь возможность просматривать информацию о своем профиле и участвовать в программе лояльности, которая представляет собой систему статусов со скидками:

— бронзовый статус (менее 10 бронирований) — 5% скидки от суммы каждого бронирования;

— серебряный статус (не менее 10 бронирований) – 7% скидки;

— золотой статус (не менее 20 бронирований) – 10% скидки;

Кроме того, у администраторов системы должна быть возможность просматривать информацию о выполненных действиях в системе, а также регистрировать новых пользователей.

1.2 Существующие аналоги

Среди аналогов можно отметить следующие сервисы:

— Островок.ру — российский сервис онлайн-бронирования отелей с акцентом на внутренний туризм [1];

— Яндекс.Путешествия — это сервис для поиска и покупки авиа и ж/д билетов, билетов на автобусы и поиска гостиниц [2];

— Туту.ру — российская онлайн-платформа для организации туристических путешествий. На сервисе можно купить билеты на поезд, самолет и автобус, бронировать номера в отелях, туры и экскурсии [3].

1.3 Требования к системе

Система должна соответствовать следующим требованиям.

1) Разрабатываемое программное обеспечение должно обеспечивать функционирование системы в режиме 24/7/365 со среднегодовым временем доступности не менее 99.9%. Допустимое время, в течение которого система не доступна, за год должна составлять 24*365*0.001=8.76 часа.

2) Время восстановления системы после сбоя не должно превышать 15 минут.

3) Каждый узел системы должен автоматически восстанавливаться после сбоя.

4) Система должна поддерживать возможность добавления нового узла во время работы системы без перезагрузки.

5) В случае недоступности некритичного функционала должна осуществляться деградация функциональности.

6) На каждом из узлов системы должно вестись журналирование.

7) Операции передачи конфиденциальных данных должны соответствовать ГОСТ Р 59407–2021.

1.4 Требования к реализации

К реализации системы предъявляются следующие требования.

1) В разрабатываемой системе пользователи делятся на две роли: Пользователь и Администратор.

2) Требуется использовать сервис-ориентированную архитектуру.

3) Все бекенды и фронтенд должны быть запущены изолированно друг от друга.

4) Для межсервисного взаимодействия использовать HTTP (RESTful API).

5) Для запросов, выполняющих обновление данных на нескольких узлах распределенной системы, в случае недоступности одной из систем, необходимо выполнять полный откат транзакции.

6) Необходимо реализовать пользовательский интерфейс как Single Page Application для фронтенда. Интерфейс должен быть доступен через тонкий клиент — браузер.

7) Серверы бекендов недоступны пользователю, это реализуется их расположением во внутренней сети.

8) Доступ к разрабатываемым базам данных должен осуществляться по протоколу TCP.

9) Валидация входных данных должна производиться на стороне фронтенда. Бекенды не должны валидировать входные данные, так как пользователь не может к ним обращаться напрямую, то есть бекенды должны получать уже отфильтрованные входные данные от фронтенда.

10) Разрабатываемая система должна поддерживать возможность горизонтального и вертикального масштабирования за счет увеличения количества функционирующих узлов и совершенствования технологий реализации компонентов и всей ее архитектуры.

1.5 Функциональные требования

Система должна обеспечивать реализацию следующих функций.

1) Аутентификация и авторизация пользователей.

2) Разделение пользователей на две роли: Пользователь и Администратор.

3) Предоставление **Пользователю** следующих функций:

   — просмотр списка отелей;

   — бронирование отеля на определенные даты;

   — просмотр бронирований;

   — отмена бронирований, дата заселения которых не наступила;

— просмотр личной информации;

— участие в программе лояльности.

4) Предоставление **Администратору** следующих функций:

— функции, доступные **Пользователю**;

— просмотр информации о выполненных действиях в системе;

— регистрация новых пользователей.

1.6 OAuth Authorization Flow

1.6.1 Сценарий авторизации

Authorization code Flow определен в OAuth 2.0 RFC 6749, раздел 4.1. Он включает в себя обмен кода авторизации на токен [4].

1) Пользователь выбирает опцию «Войти» в приложении.

2) Пользователь перенаправляется на сервер авторизации Auth0 (endpoint /authorize).

3) Сервер авторизации Auth0 отображает страницу аутентификации и при необходимости окно согласия с перечнем разрешений, предоставляемых приложению.

4) Пользователь проходит аутентификацию одним из доступных способов.

5) Сервер авторизации Auth0 перенаправляет пользователя обратно в приложение, передавая одноразовый код авторизации.

6) Auth0 отправляет код авторизации, идентификатор клиента и учетные данные приложения (например, client secret или Private Key JWT) на сервер авторизации Auth0 (endpoint /token).

7) Сервер авторизации Auth0 проверяет код авторизации, идентификатор клиента и его учетные данные.

8) В случае успеха сервер авторизации возвращает ID Token, Access Token, при необходимости — Refresh Token.

9) Приложение использует Access Token для вызова API и получения информации о пользователе.

10) API проверяет токен и отвечает приложению запрошенными данными.

1.6.2 PKCE для публичных клиентов

Для клиентов, работающих на стороне фронтенда (например, SPA или мобильных приложений), использование client secret невозможно, так как его нельзя хранить безопасно. В таких случаях применяется расширение PKCE (Proof Key for Code Exchange) [5].

При инициализации запроса авторизации приложение формирует случайную строку — code verifier, из которой вычисляется code challenge (с помощью SHA-256 и Base64URL).

— В параметрах запроса к endpoint /authorize передается code challenge.

— После получения кода авторизации клиент направляет запрос к endpoint /token, указывая исходный code verifier.

— Сервер авторизации сверяет соответствие code challenge и code verifier.

Таким образом, снижается риск подмены кода авторизации злоумышленником.

1.6.3 Проверка токенов с использованием JWKS

Возвращаемые сервером токены (ID Token и Access Token) подписаны приватным ключом. Для проверки их подлинности приложение и API используют открытые ключи, публикуемые сервером авторизации в формате JWKS (JSON Web Key Set) [6].

Проверка токена включает:

— извлечение заголовка токена и определение идентификатора ключа (kid);

— получение соответствующего открытого ключа из JWKS-документа по URL сервера авторизации;

— проверку криптографической подписи токена;

— проверку срока действия (`exp`) и других обязательных полей (например, `aud`, `iss`).

Данный механизм гарантирует, что токен действительно выдан сервером авторизации и не был изменен.

2 Конструкторский раздел

2.1 Архитектура системы

На рисунке 2.1 представлена топология разрабатываемой распределенной системы.

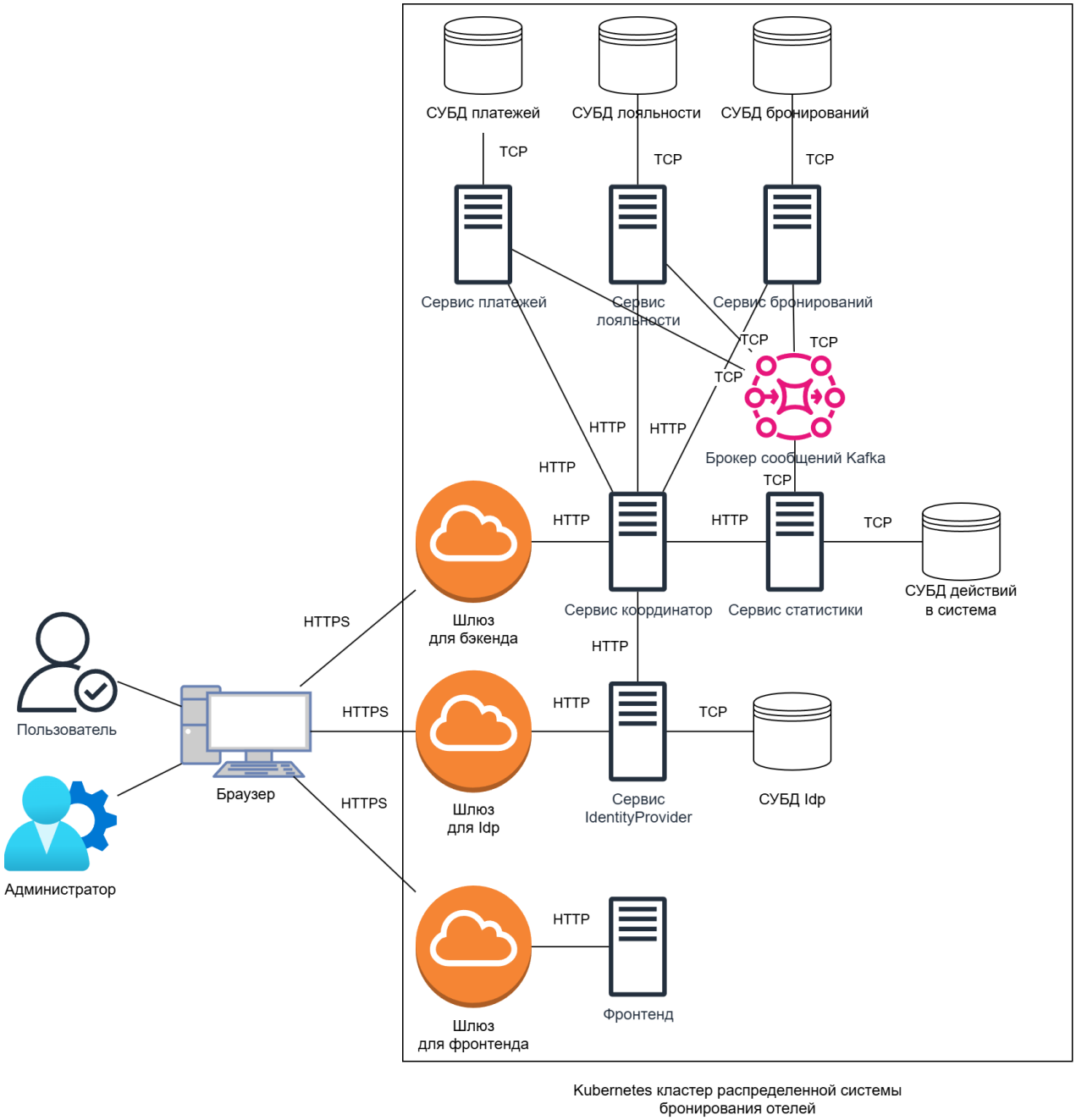

Рисунок 2.1 – Топология распределенной системы бронирования отелей.

2.2 Описание функциональных сервисов системы

Система включает в себя фронтенд, сервис-координатор и 5 подсистем:

1) сервис аутентификации;

2) сервис лояльности;

3) сервис платежей;

4) сервис бронирований;

5) сервис сбора статистики.

Все сервисы должны принимать и отдавать данные в формате JSON по протоколу HTTP.

2.2.1 Фронтенд

Фронтенд — серверное приложение, предоставляет пользовательский интерфейс и внешний API системы. При его разработке нужно учитывать следующие факторы:

— фронтенд должен принимать запросы по протоколу HTTP и формировать ответы пользователям в формате HTML;

— фронтенд должен отправлять поступающие от пользователей запросы в сервис-координатор;

— запросы к бекендам необходимо осуществлять по протоколу HTTP;

— данные необходимо передавать в формате JSON;

— фронтенд должен предоставлять пользовательский интерфейс как Single Page Application.

2.2.2 Сервис координатор

При его разработке нужно учитывать, что сервис координатор должен:

— координировать обмен сообщениями между сервисами внутри системы, направляя запросы к соответствующим бекенд серверам и отправляя их ответы на фронтенд;

— накапливать статистику запросов, и в случае, если какой-то из бекендов не ответил N раз, то на N + 1 раз вместо запроса к нему отдавать fallback-ответ, а через некоторое время выполнять запрос к этому бекенду, чтобы проверить его состояние (реализация паттерна Circuit Breaker).

2.2.3 Сервис аутентификации

При его разработке нужно учитывать, что сервис аутентификации должен:

— выполнять функцию Identity Provider;

— реализовывать протокол OpenId Connect Authorization Flow.

СУБД сервиса аутентификации должна хранить следующие сущности.

1) Код авторизации.

   1.1) Идентификатор кода (guid, генерируется автоматически, первичный ключ).

   1.2) Идентификатор клиента (строка, обязательное поле, ссылка на сущность *Клиент*).

   1.3) Идентификатор пользователя (строка, обязательное поле).

   1.4) Конечная точка перенаправления (строка, не более 256 символов, обязательное поле).

   1.5) Список запрошенных прав доступа (строка, обязательное поле).

   1.6) Срок действия кода (дата и время, обязательное поле).

   1.7) Кодовое значение для PKCE (строка).

   1.8) Метод преобразования для PKCE (строка).

2) Клиент.

   2.1) Идентификатор записи (строка, генерируется автоматически, первичный ключ).

   2.2) Идентификатор клиента (строка, обязательное поле, уникальное значение).

2.3) Получатель (строка, обязательное поле).

2.4) Секрет клиента (строка, указывается только для конфиденциальных клиентов).

2.5) Список разрешенных redirect URI (строка, обязательное поле).

2.6) Разрешенные права доступа (строка, обязательное поле).

2.7) Требуется использование PKCE (логическое значение).

2.8) Признак публичного клиента (логическое значение).

2.2.4 Сервис лояльности

При его разработке нужно учитывать, что он должен:

— выполнять определение величины скидки по идентификатору пользователя;

— выполнять обновление числа забронированных отелей и статуса в программе лояльности (предусмотреть, как повышение, так и понижение в случае отмены бронирования).

— отправлять события о действиях в брокер сообщений kafka.

СУБД сервиса лояльности должна хранить следующие сущности.

1) Лояльность.

1.1) Идентификатор (целое число, генерируется автоматически, первичный ключ).

1.2) Логин пользователя (строка, обязательное поле).

1.3) Количество бронирований (целое число, значение по умолчанию – 0).

1.4) Статус (строка, бронзовый/серебряный/золотой, по умолчанию – бронзовый).

1.5) Размер скидки (целое число).

2.2.5 Сервис платежей

При его разработке нужно учитывать, что он должен:

— предоставлять информацию о платежной операции по ее идентификатору;

— создавать запись об оплате;

— выполнять получение и обновление статуса оплаты.

— отправлять события о действиях в брокер сообщений kafka.

СУБД сервиса платежей должна хранить следующие сущности.

1) Платеж.

   1.1) Идентификатор платежа (целое число, генерируется автоматически, первичный ключ).

   1.2) Идентификатор записи (guid).

   1.3) Сумма (целое число).

   1.4) Статус платежа (строка).

2.2.6 Сервис бронирований

При его разработке нужно учитывать, что он должен:

— предоставлять информацию об отелях;

— создавать запись о бронировании;

— предоставлять информацию о бронированиях конкретного пользователя;

— выполнять получение и обновление статуса бронирования.

— отправлять события о действиях в брокер сообщений kafka.

СУБД сервиса бронирований должна хранить следующие сущности.

1) Отель.

1.1) Идентификатор (целое число, генерируется автоматически, первичный ключ).

1.2) Глобальный уникальный идентификатор отеля (guid, генерируется автоматически, уникальное поле).

1.3) Название (строка, не более 255 символов, обязательное поле).

1.4) Страна (строка, не более 80 символов, обязательное поле).

1.5) Город (строка, не более 80 символов, обязательное поле).

1.6) Адрес (строка, не более 255 символов, обязательное поле).

1.7) Количество звезд (целое число от 1 до 5, может быть пустым).

1.8) Цена (целое число, обязательное поле).

2) Бронирование.

2.1) Идентификатор (целое число, генерируется автоматически, первичный ключ).

2.2) Глобальный уникальный идентификатор бронирования (guid, генерируется автоматически, уникальное поле).

2.3) Имя пользователя (строка, не более 80 символов, обязательное поле).

2.4) Идентификатор платежа (guid, обязательное поле).

2.5) Идентификатор отеля (целое число, внешний ключ на сущность Отель).

2.6) Статус (строка, не более 20 символов, обязательное поле, возможные значения: «Не оплачено», «Оплачено», «Отменено»).

2.7) Дата заселения (дата, обязательное поле).

2.8) Дата выезда (дата, обязательное поле).

2.2.7 Сервис сбора статистики

При его разработке нужно учитывать, что он должен:

— принимать события о действиях в системе с помощью брокера сообщений kafka;

— сохранять и предоставлять информацию о действиях в системе.

СУБД сервиса сбора статистики должна хранить следующие сущности.

1) Действие пользователя.

   1.1) Идентификатор (guid, генерируется автоматически, первичный ключ).

   1.2) Идентификатор пользователя (строка, не более 128 символов, обязательное поле).

   1.3) Имя пользователя (строка, не более 128 символов, обязательное поле).

   1.4) Название сервиса (строка, не более 128 символов, обязательное поле).

   1.5) Действие (строка, не более 128 символов, обязательное поле).

   1.6) Статус (строка, не более 128 символов, обязательное поле).

   1.7) Метка времени (дата и время с часовым поясом, обязательное поле).

   1.8) Дополнительные данные (json).

   1.9) Топик (строка, не более 128 символов, обязательное поле).

   1.10) Раздел (целое число, обязательное поле).

   1.11) Смещение (целое число, обязательное поле, уникальность обеспечивается в связке с топиком и разделом).

2.3 Сценарии функционирования системы

2.3.1 Авторизация пользователя

1) Пользователь нажимает на кнопку «Войти в систему» на фронтенде.

2) Пользователь перенаправляется на сервис авторизации, предоставляющий страницу ввода логина и пароля.

3) Пользователь вводит логин и пароль, затем подтверждает ввод нажатием кнопки «Войти».

4) Если данные указаны неверно, отображается сообщение об ошибке. При корректных данных пользователь получает доступ к системе и попадает на главную страницу.

2.3.2 Просмотр списка отелей и бронирование

1) На главной странице отображается список отелей, разбитый на страницы. Пользователь может выбрать номер страницы и количество записей на ней. Таблица включает 6 столбцов: страна, город, название отеля, количество звезд, цена, а также действие (кнопка «Забронировать»).

2) Доступна сортировка и фильтрация списка в рамках страницы.

   — Сортировка: по стране, городу, названию отеля (по алфавиту), количеству звезд, цене (по возрастанию/убыванию).

   — Фильтрация: по стране, городу и названию отеля (совпадение подстроки), по количеству звезд (не менее заданного), по цене (не более заданной суммы).

3) Для бронирования пользователь нажимает кнопку «Забронировать» в строке нужного отеля. В открывшейся форме он указывает даты заезда и выезда и подтверждает действие кнопкой «Забронировать». При успешном бронировании отображается сообщение с итоговой стоимостью, а также кнопки «Отменить» и «Мои бронирования».

2.3.3 Просмотр списка бронирований

1) После авторизации пользователь нажимает кнопку «Бронирования» в верхней части страницы.

2) Открывается страница со списком бронирований в виде карточек. На карточке отображаются: название отеля, даты заезда и выезда, итоговая стоимость и статус оплаты.

3) Карточки активных бронирований (дата заезда еще не наступила) содержат кнопку «Отменить бронирование». В зависимости от текущей даты карточки имеют разные статусы и визуальное оформление: «Отменено», «Запланировано», «Скоро», «Сейчас в отеле», «Завершено».

2.3.4 Просмотр информации о профиле

1) После авторизации пользователь нажимает кнопку «Профиль».

2) Открывается страница профиля, где отображаются: логин, адрес электронной почты, роль в системе, имя и фамилия.

3) Дополнительно выводится информация о программе лояльности: размер скидки, количество совершенных бронирований и необходимое количество для повышения статуса.

2.4 Диаграммы прецедентов

На рисунке 2.2 представлена диаграмма прецедентов для Пользователя.

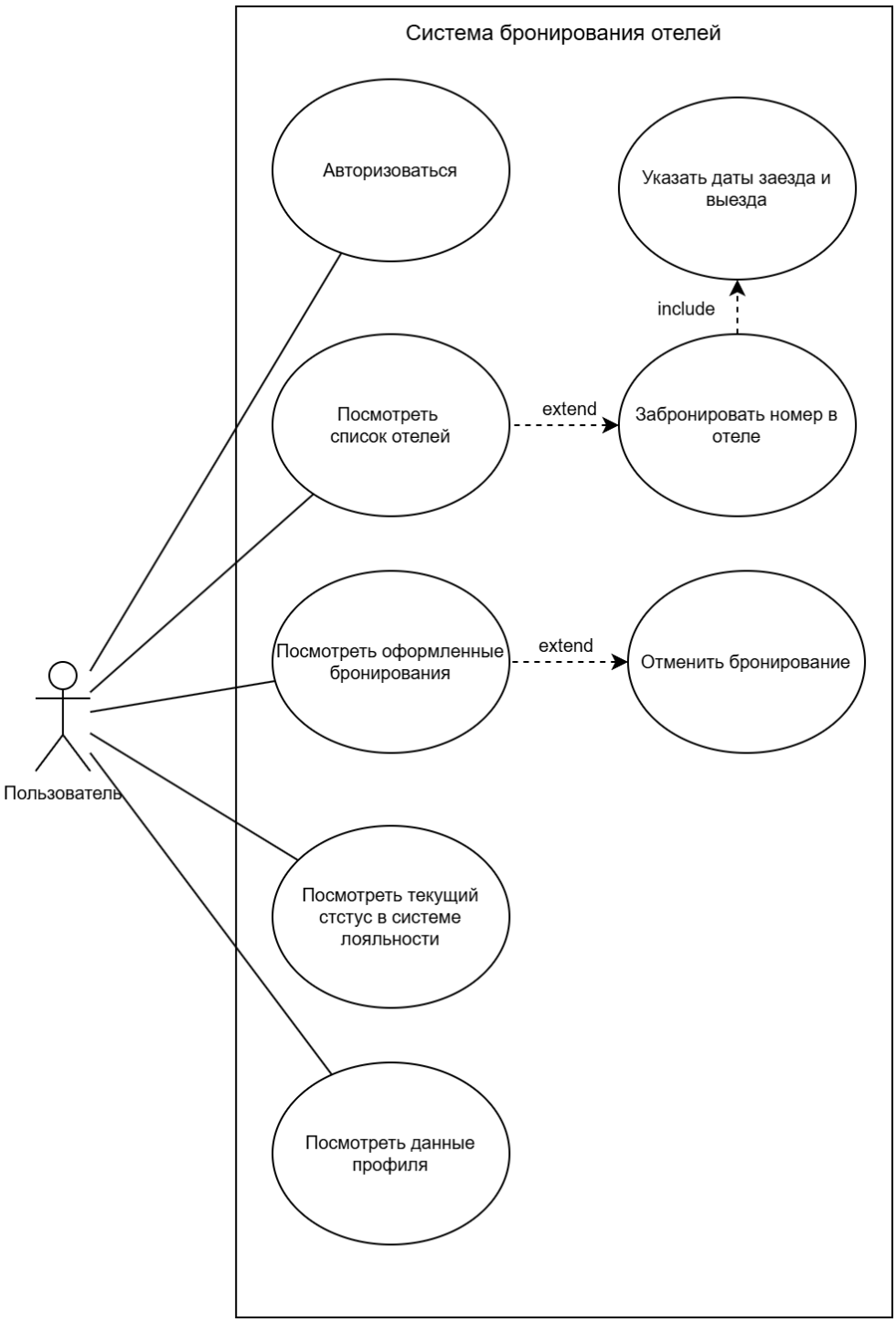

Рисунок 2.2 – Диаграмма прецедентов для Пользователя.

На рисунке 2.3 представлена диаграмма прецедентов для Администратора.

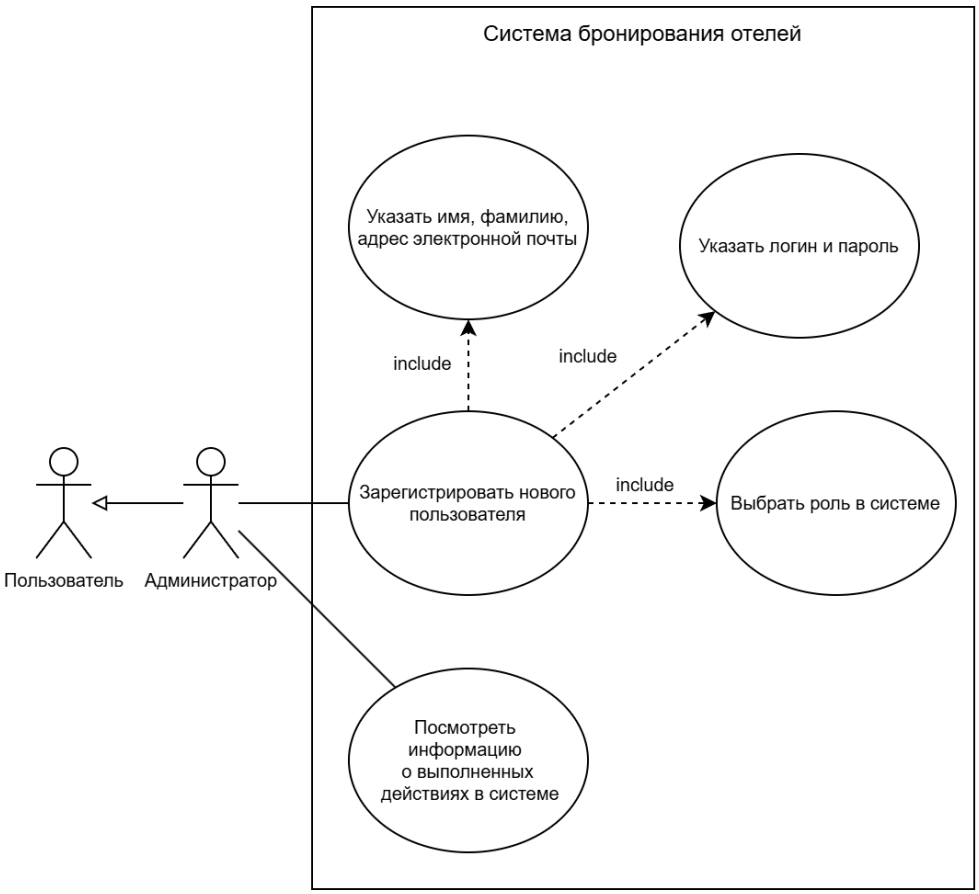

Рисунок 2.3 – Диаграмма прецедентов для Администратора.

2.5 Диаграмма последовательности

На рисунке 2.4 представлена диаграмма последовательности бронирования пользователем отеля.

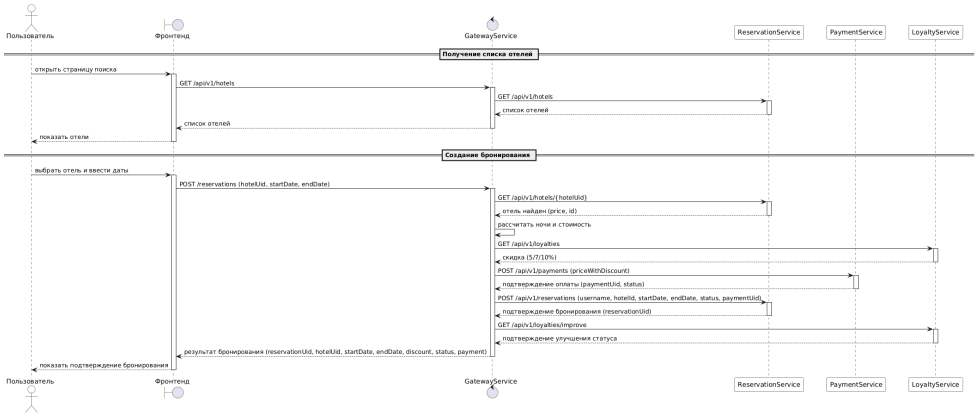

Рисунок 2.4 – Диаграмма последовательности бронирования пользователем отеля.

2.6 Диаграмма баз данных

На рисунке 2.5 представлена диаграмма баз данных распределенной системы бронирования отелей.

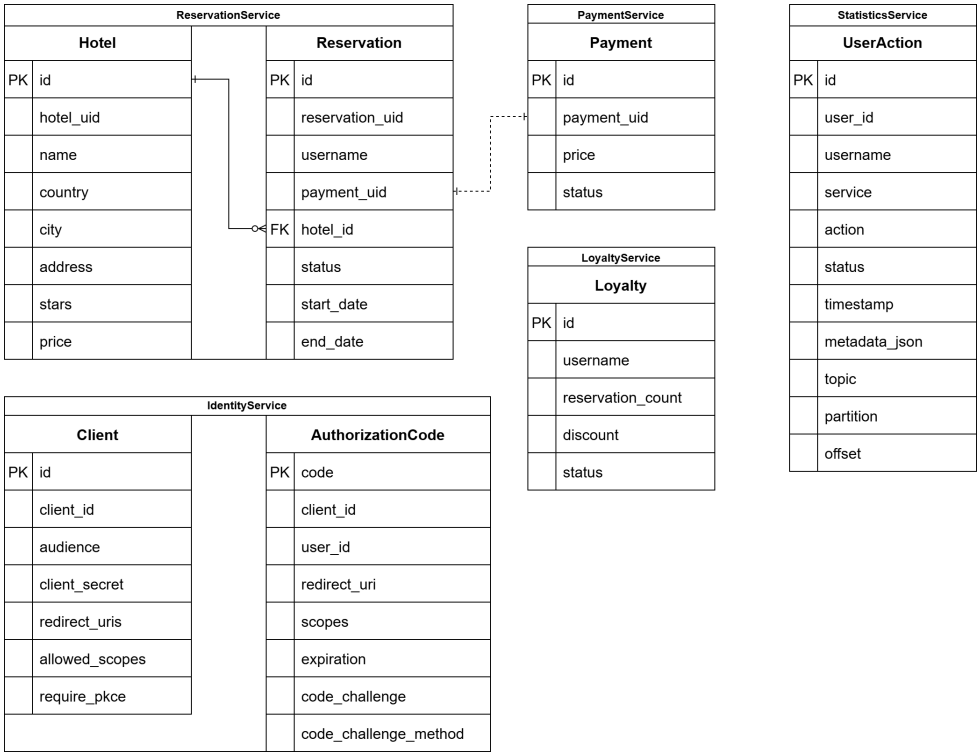

Рисунок 2.5 – Диаграмма баз данных.

3 Технологический раздел

3.1 Высокоуровневый дизайн пользовательского интерфейса

Пользовательский интерфейс в разработанной системе предоставляет собой web-интерфейс, реализованный как Single Page Application, доступ к которому осуществляется через браузер (тонкий клиент). Фронтенд включает в себя следующие страницы:

— страница авторизации;

— страница со списком отелей;

— страница бронирований;

— страница профиля;

— страница статистики системы (только для Администратора);

— страница создания нового пользователя (только для Администратора).

Примеры данных страниц представлены на рисунках 3.1-3.6.

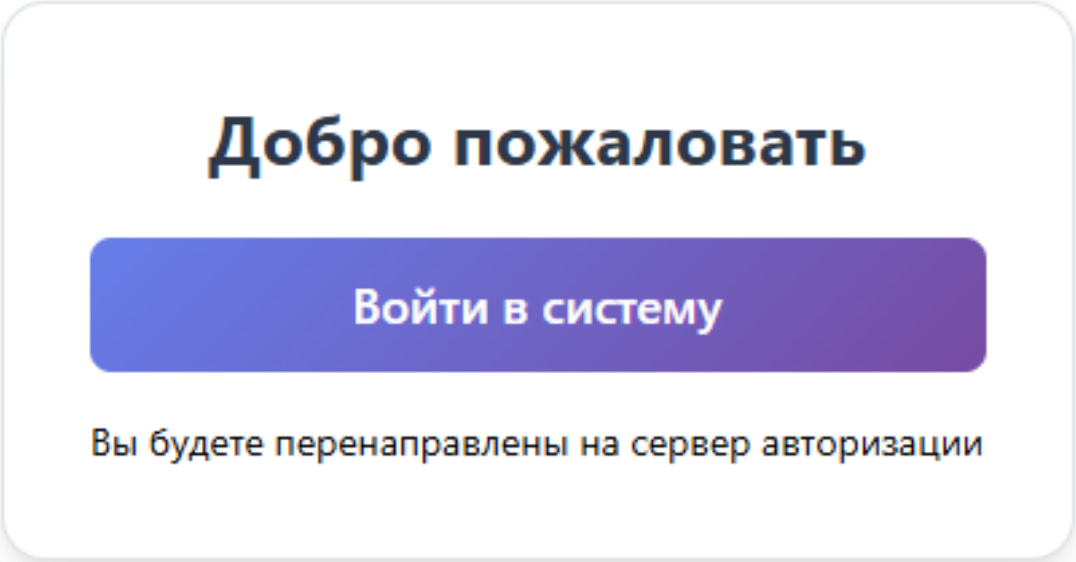

Рисунок 3.1 – Страница авторизации.

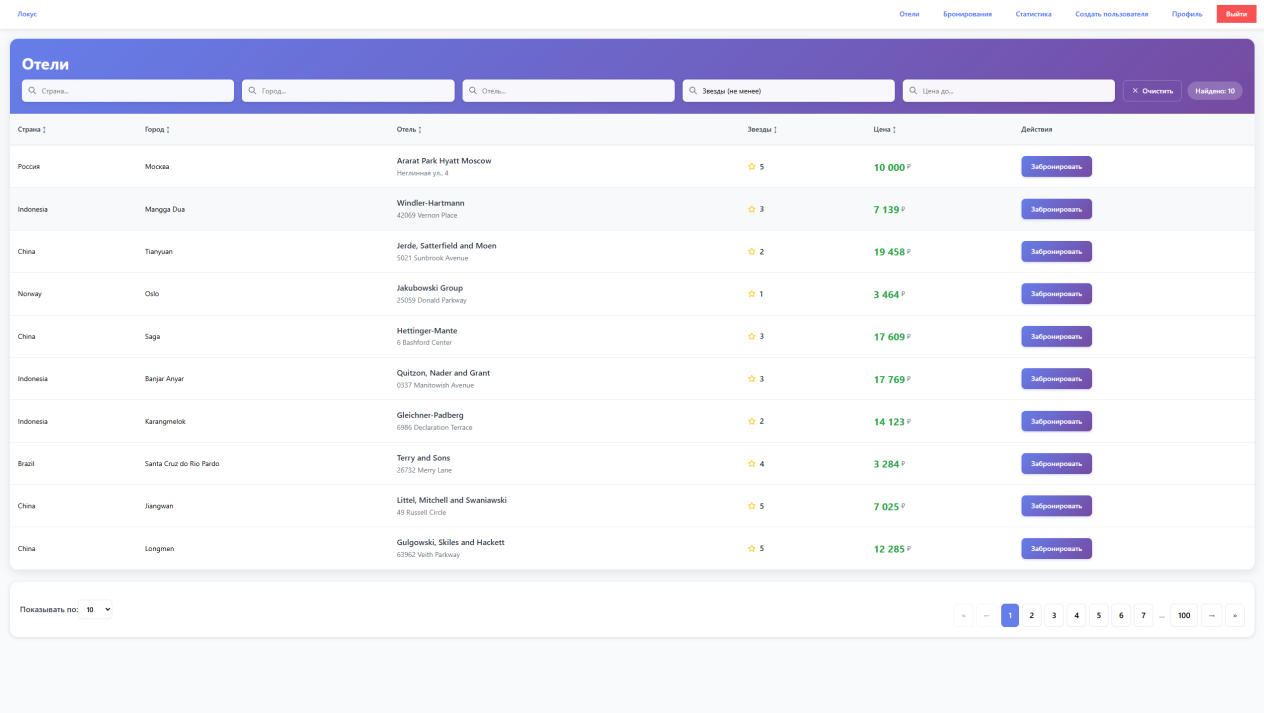

Рисунок 3.2 – Страница со списком отелей.

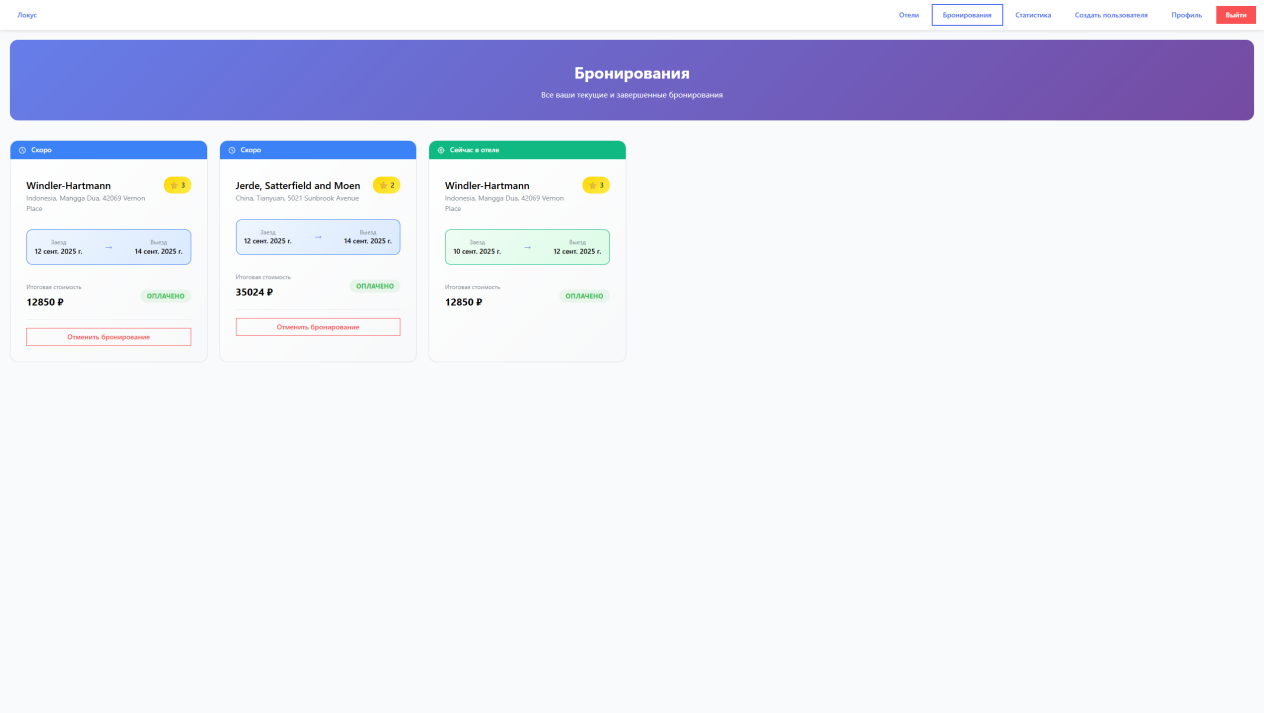

Рисунок 3.3 – Страница бронирований.

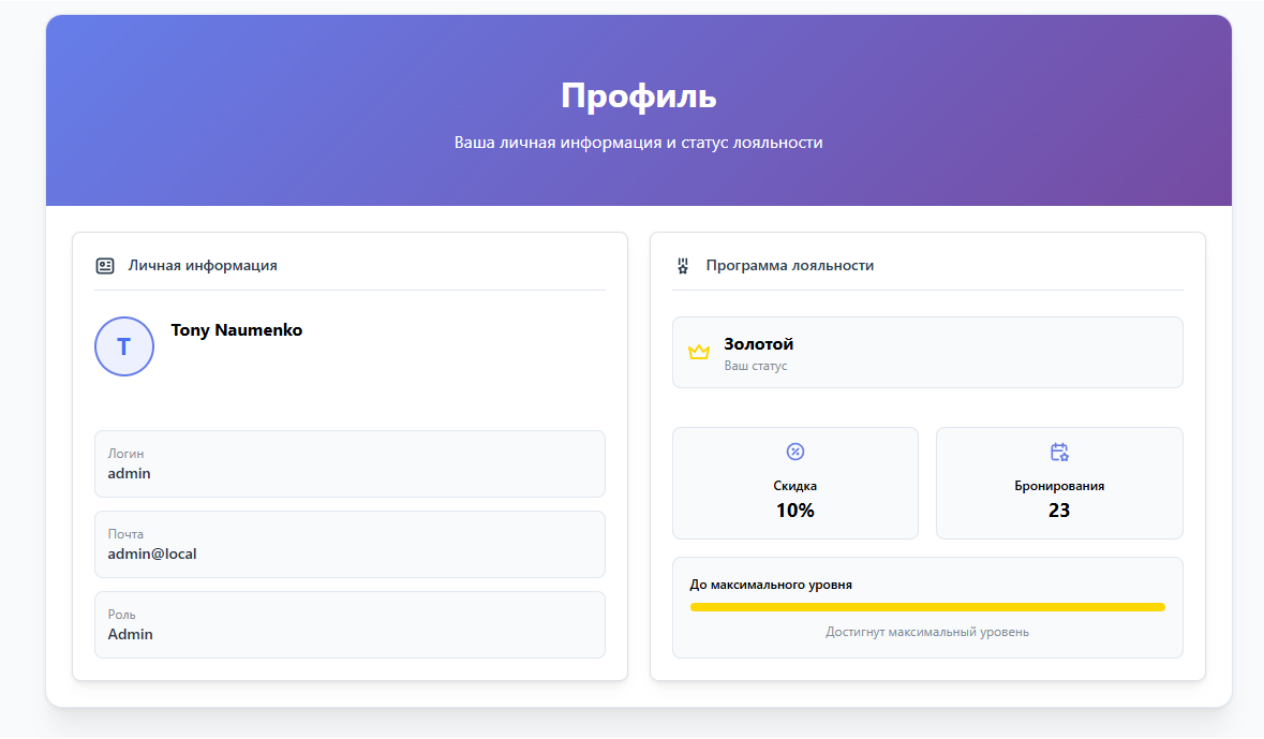

Рисунок 3.4 – Страница профиля.

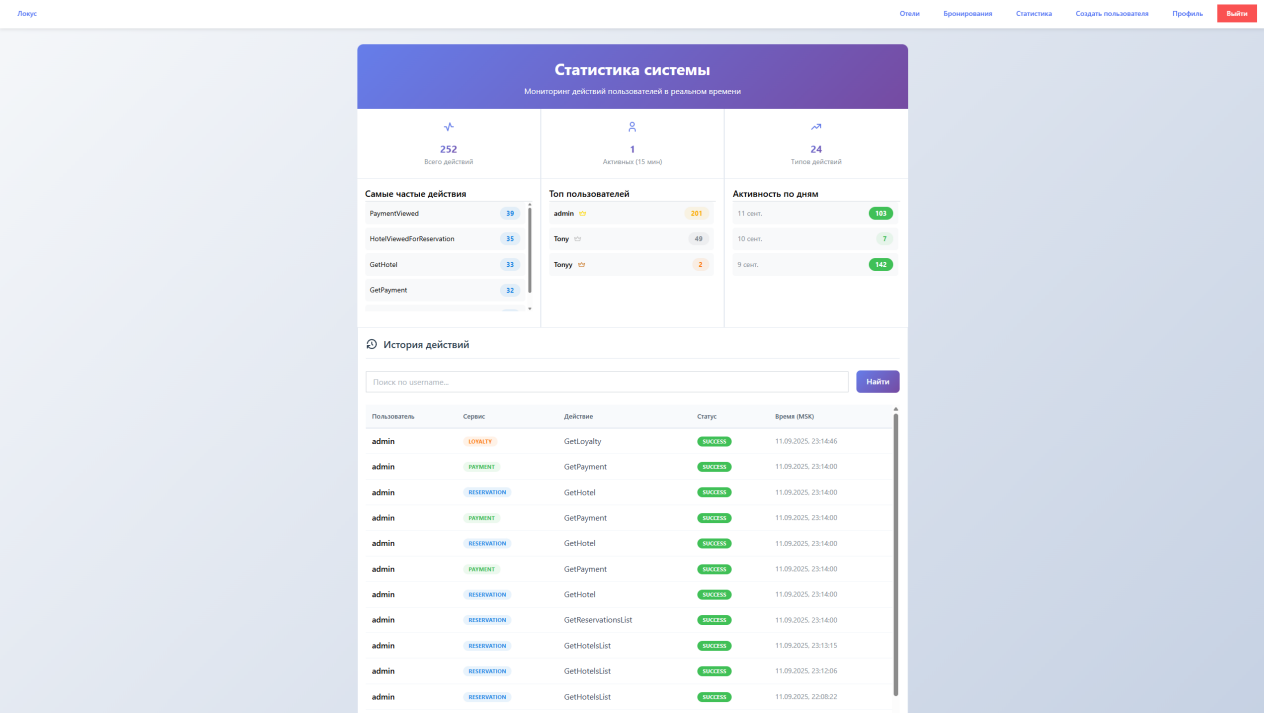

Рисунок 3.5 – Страница статистики системы.

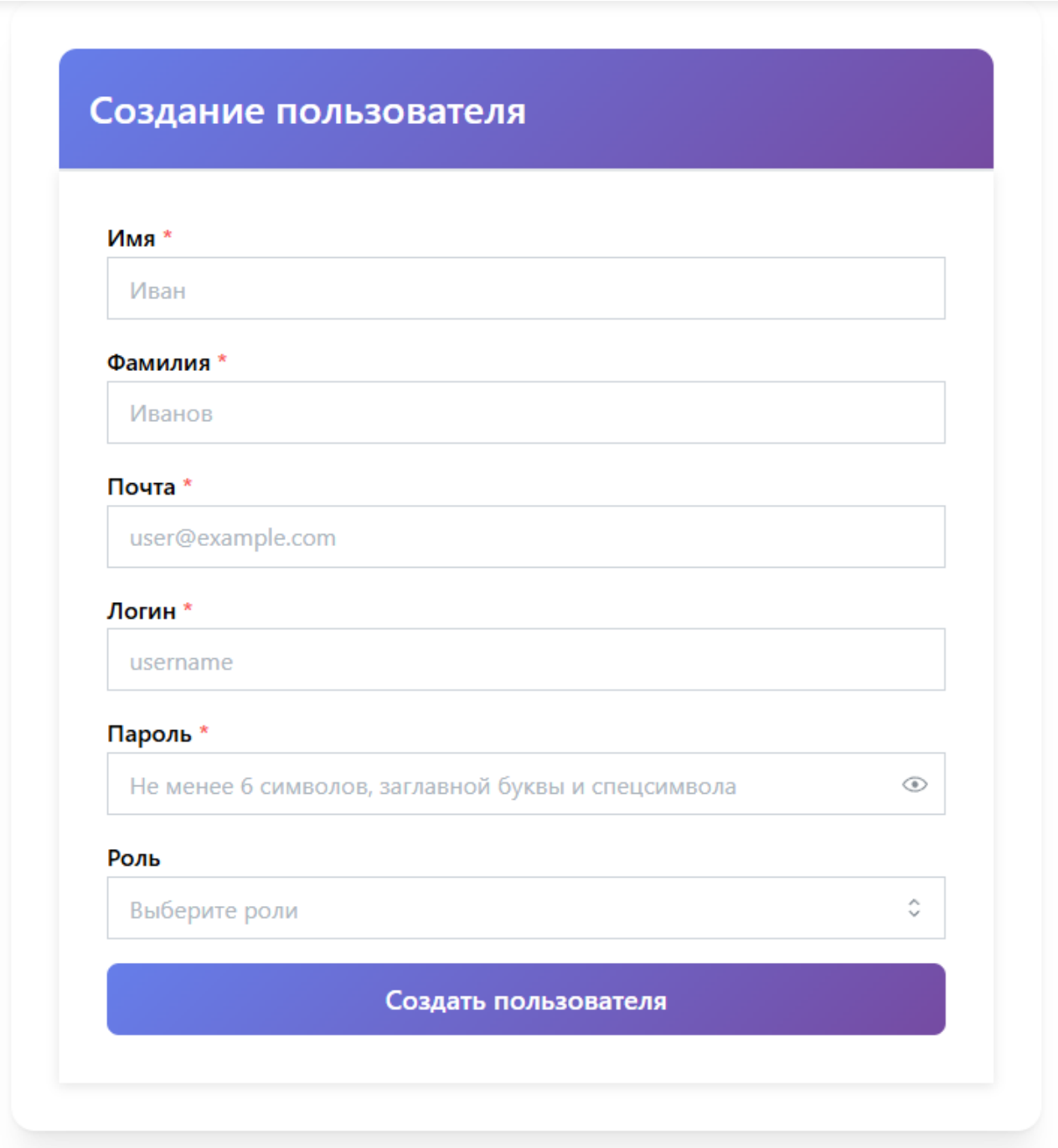

Рисунок 3.6 – Страница создания нового пользователя.

3.2 Поведение системы при отказе компонентов

3.2.1 Отказ сервиса лояльности

В системе лояльности хранится информация о количестве бронирований и статусах клиентов (BRONZE, SILVER, GOLD). При отмене бронирования недоступность данного сервиса не блокирует критические операции, а следовательно запрос на отмену бронирования выполняется. Если сервис лояльности недоступен, пользователю возвращается ответ об успешной отмене. Запрос

в сервис лояльности помещается в очередь внутри сервиса координатора и повторяется до успешного ответа от него.

3.2.2 Отказ Kafka

Все действия пользователей публикуются в брокер сообщений Kafka в виде событий (user-actions). Сервис статистики асинхронно считывает сообщения и сохраняет их в базу данных. Если Kafka недоступен, события не попадают в сервис статистики. При этом основные функции системы продолжают работать без ограничений. После восстановления Kafka накопленные сообщения из ее буфера восстанавливаются и доставляются в сервис статистики.

3.3 Защита от дублирования сообщений

Для предотвращения повторной обработки сообщений реализован следующий механизм:

— Каждое сообщение Kafka содержит поля topic, partition, offset.

— В таблице user_actions создан уникальный составной индекс по этим трем полям.

— При повторном получении сообщения (например в случае, если сервис сохранил сообщение в базу, но не отметил его как полученное) попытка записи приведет к исключению.

— Исключение перехватывается и обрабатывается, сообщение помечается как обработанное, чтобы избежать бесконечной повторной обработки.

3.4 Автоматическое тестирование авторизации

На сервисе координаторе реализован специальный эндпоинт (/api/v1/authorize/directlogin), который позволяет выполнить авторизацию без использования графической формы ввода логина и пароля сервиса авторизации. Вместо этого учетные данные пользователя передаются напрямую в теле HTTP-запроса. Данный эндпоинт необходим для упрощения процесса интеграционного тестирования. При этом остальной поток OAuth выполняется без изменений.

3.5 Тестирование системы

Тестирование системы интегрировано в GitHub Actions и выполняется с использованием Postman и утилиты newman, позволяющей автоматизировать прогон API-тестов.

На первом этапе, после сборки контейнеров и запуска окружения, тесты проверяют как базовую доступность API, так и корректность деградации функциональности при недоступности отдельных сервисов. Таким образом, обеспечивается контроль не только стандартных сценариев работы, но и поведения системы в условиях отказа некритичных компонентов.

На втором этапе, после развертывания микросервисов в кластере Kubernetes, тесты выполняются повторно. Здесь проверяются типичные сценарии использования API в условиях, максимально приближенных к production-среде. Такой подход позволяет убедиться, что система функционирует корректно после развертывания и что ее базовые возможности остаются доступными пользователям.

3.6 Сборка и развертывание системы

Система собирается и разворачивается автоматически с использованием GitHub Actions. Процесс состоит из следующих этапов:

1) Сборка Docker-образов и запуск контейнеров с помощью docker compose.

2) Выполнение интеграционных API-тестов. Если тесты не проходят, пайплайн завершается с ошибкой.

3) В случае успешного прохождения тестов Docker-образы сервисов публикуются в Docker Hub.

4) Деплой в кластер Minikube:

    4.1) установка зависимостей (PostgreSQL для сервисов, требующих наличие СУБД, Redis, Kafka);

    4.2) деплой микросервисов (gateway, loyalty, payment, reservation, identity, statistics, frontend);

    4.3) настройка Ingress-контроллера;

4.4) ожидание и проверка готовности подов и сервисов.

5) Выполнение тестов для развернутой в кластере системы.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана распределенная система для бронирования отелей на интересующие даты. Были рассмотрены существующие аналоги, приведены требования к системе и ее реализации. Также описаны функциональные требования и приведено описание протокола OAuth Authorization Flow.

Приведена архитектура системы и описаны ее функциональные сервисы. Представлены сценарии функционирования, а также диаграммы прецендентов, последовательности, и баз данных.

Приведен пример дизайна пользовательского интерфейса. Описано поведение системы при отказе компонентов, не влияющих на критический функционал приложения. Представлены особенности реализации и процесс тестирования, сборки и развертывания системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Островок! [Электронный ресурс]. — Режим доступа URL: https://ostrovok.ru (дата обращения: 11.09.2025).
2. Яндекс Путешествия [Электронный ресурс]. — Режим доступа URL: https://travel.yandex.ru (дата обращения: 11.09.2025).
3. tutu Отели [Электронный ресурс]. — Режим доступа URL: https://hotel.tutu.ru (дата обращения: 11.09.2025).
4. Authorization Code Flow [Электронный ресурс]. — Режим доступа URL: https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow (дата обращения: 11.09.2025).
5. What is PKCE Flow? [Электронный ресурс]. — Режим доступа URL: https://hemantasundaray.vercel.app/blog/pkce-flow (дата обращения: 11.09.2025).
6. What is JWKS? JSON Web Key Set [Электронный ресурс]. — Режим доступа URL: https://dev.to/fung-authgear/what-is-jwks-json-web-key-set-short-guide-3j9m (дата обращения: 11.09.2025).