|  |  |
| --- | --- |
|  | **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 система управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_«Программное обеспечение ЭВМ и информационные» технологии\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Система для бронирования отелей***

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**Ильясов И.М.\_\_\_

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

*2023 г.*

ТУТ ТЗ

СОДЕРЖАНИЕ

[Глоссарий 5](#_Toc142923308)

[ВВЕДЕНИЕ 6](#_Toc142923309)

[1. Аналитический раздел 7](#_Toc142923310)

[1.1. Постановка задачи 7](#_Toc142923311)

[1.2. Описание системы 7](#_Toc142923312)

[1.3. Общие требования к системе 7](#_Toc142923313)

[1.4. Требования к функциональным характеристикам 8](#_Toc142923314)

[1.5. Функциональные требования к системе с точки зрения пользователя 8](#_Toc142923315)

[1.6. Входные данные 9](#_Toc142923316)

[1.7. Выходные данные 10](#_Toc142923317)

[1.8. Топология системы 10](#_Toc142923318)

[1.9. Требования к программной реализации 12](#_Toc142923319)

[1.10. Функциональные требования к подсистемам 13](#_Toc142923320)

[1.11. Вывод 15](#_Toc142923321)

[2. Конструкторский раздел 17](#_Toc142923322)

[2.1. Концептуальный дизайн 17](#_Toc142923323)

[2.2. Сценарии функционирования системы 18](#_Toc142923324)

[2.3. Диаграммы прецедентов 19](#_Toc142923325)

[2.4. Спецификация сценариев 20](#_Toc142923326)

[2.5. Спецификации классов 23](#_Toc142923327)

[2.6. Структура базы данных 27](#_Toc142923328)

[2.7. Вывод 27](#_Toc142923329)

[3. Технологический раздел 28](#_Toc142923330)

[3.1. Выбор операционной системы 28](#_Toc142923331)

[3.2. Выбор СУБД 29](#_Toc142923332)

[3.3. Выбор языка разработки компонентов 30](#_Toc142923333)

[3.4. Дизайн пользовательского интерфейса 30](#_Toc142923334)

[3.5. Вывод 33](#_Toc142923335)

[ЗАКЛЮЧЕНИЕ 34](#_Toc142923336)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc142923337)

Глоссарий

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| Медиана времени отклика | Среднее время предоставления данных Пользователю. |
| Валидация | Проверка данных на соответствие заданным условиям и ограничениям. |
| REST | Архитектурный стиль взаимодействия компонентов распределённого приложения в сети. |
| SPA | Веб-приложение или веб-сайт, использующий единственный HTML-документ как оболочку для всех веб-страниц и организующий взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript, обычно посредством AJAX. |
| Профиль пользователя | Информация о пользователе портала, хранящаяся в портале, в частности, имя, фамилия, фотография и др. |
| Фронтэнд | Серверное приложение, принимающее запросы от пользователя портала. На каждый из типов запросов от пользователя (показать новости, оплатить работу мастера и др.) фронтэнд определяет, как организовать выполнение запроса. Фронтэнд принимает запросы от пользователя, анализирует их и в соответствии с заложенным алгоритмом выполняет запросы к бекэндам. |
| Бекэнд | Серверное приложение, выполняющее определенную задачу, например, взаимодействие с СУБД. Бекэнды принимают запросы от фронтэнда. |
| Бронирование | Размещение и подтверждение заявки на туристическую услугу. |
| Отель | Заведение, в котором сдаются номера на съем. |
| Узел | Физический элемент системы, существующий во время исполнения и представляющий собой некий вычислительный ресурс. |

ВВЕДЕНИЕ

Согласно исследованию агентства BusinessStat, количество гостиничных учреждений в России за последние годы выросло на 10% [1]. На сегодняшний день путешествия стали обыденностью, нежели чем-то непозволительно дорогим. И перед людьми, планирующими свой отдых, всегда остро стоит вопрос бронирования отелей. Сервисы, предлагающие услуги по поиску и бронированию отелей, являются агрегаторами данных предоставляемых услуг. Такие сервисы зачастую выполняются в виде портала со списком отелей, возможностью выбора необходимого отеля в нужном городе и дальнейшей брони номера в данном отеле.

Целью данной работы разработка распределенной системы, позволяющей производить бронирование отелей. Для достижения поставленной цели необходимо решить следующие задачи:

* сформулировать основные требования к системе;
* описать требования к подсистемам;
* описать архитектуру системы, а также сценарии ее функционирования;
* выбрать и обосновать инструменты для разработки ПО;
* реализовать программный продукт.

1. Аналитический раздел

В данном разделе будет произведена постановка задачи, описана разрабатываемая система, сформулированы общие требования к системе, а также требования к функциональным характеристикам, приведены функциональные требования к системе с точки зрения пользователя, описаны входные и выходные данные системы, сформулированы требования к программной реализации и функциональные требования к подсистемам. Также будет приведена топология системы.

* 1. Постановка задачи

Разрабатываемая система должна предоставлять пользователям возможность бронирования номеров. Должен быть предусмотрен поиск подходящих номеров по таким параметрам, как дата и продолжительность бронирования, стоимость. В зависимости от количества сделанных ранее заказов система должна рассчитывать скидку на новые согласно условиям программы лояльности.

* 1. Описание системы

Разрабатываемый сервис должен представлять собой распределённую систему для бронирования отелей. Если клиент хочет оформить бронь, ему необходимо зарегистрироваться, указав информацию: фамилия, имя, электронная почта. В случае, если зарегистрированному ранее пользователю нужно отменить заказ, получить информацию о его бронированиях или статусе в программе лояльности, ему нужно авторизоваться. Для неавторизованных пользователей доступен только просмотр общей информации.

* 1. Общие требования к системе

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. Вся приходящая на сервер информация должна валидироваться.
4. Каждый узел должен автоматически восстанавливаться после сбоя.
   1. Требования к функциональным характеристикам
5. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы пользователя на получение информации не должна превышать 3 секунд.
6. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы, добавляющие или изменяющие информацию на портале не должна превышать 5 секунд.
7. Портал должен обеспечивать возможность запуска в современных браузерах: не менее 85% пользователей Интернета должны иметь возможность пользоваться порталом без какой-либо деградации функционала.
   1. Функциональные требования к системе с точки зрения пользователя

Портал должен обеспечивать реализацию следующих функций:

1. Система должна обеспечивать регистрацию пользователей с валидацией вводимых данных.
2. Система должна обеспечивать аутентификацию пользователей.
3. Система должна обеспечивать разделение пользователей на три роли:

* пользователь — неавторизованный пользователь;
* клиент — авторизованный пользователь;
* администратор.

1. Система должна предоставлять **администратору** следующие функции:

* просмотр списка отелей, входящих в сеть;
* просмотр информации о возможности бронирования номера;
* получение информации об условиях программы лояльности;
* авторизация в системе;
* получение детальной информации по конкретному бронированию;
* отмена оформленного заказа;

1. Система должна предоставлять **клиенту** следующие функции:
   * просмотр информации об отелях: название, местоположение;
   * просмотр истории бронирований;
   * бронирование номера в выбранном отеле
   * отмена брони номера в отеле;
2. Система должна предоставлять **пользователю** следующие функции:
   * просмотр информации об отелях: название, местоположение, «звездность» отеля, стоимость;
   * регистрация;
   * авторизация.
   1. Входные данные

|  |  |
| --- | --- |
| **Сущность** | **Входные данные** |
| Отель | * + - 1. идентификатор.       2. название не более 256 символов.       3. страна.       4. город.       5. адрес не более 1024 символа.       6. количество звёзд.       7. цена номера. |
| Клиент/администратор | фамилия не более 256 символов  имя не более 256 символов  логин  пароль  почта |
| Бронирование | * + - 1. идентификатор       2. имя пользователя, на которое оформлена бронь       3. идентификатор отеля       4. идентификатор платежной информации       5. статус (PAID/CANCELED)       6. дата въезда       7. дата выезда |
| Оплата | идентификатор  статус (PAID/CANCELED)  сумма |
| Лояльность | 1. имя пользователя 2. количество бронирований 3. статус (BRONZE/SILVER/GOLD) 4. процент скидки в зависимости от количества бронирований |
| Авторизация | 1. имя пользователя 2. пароль |

* 1. Выходные данные

Выходными параметрами системы являются веб-страницы. Они должны содержать следующую информацию:

* страницу регистрации и авторизации пользователя;
* информацию об отеле (название, адрес, «звездность» отеля, стоимость номеров);
* окно бронирования;
* окно для администратора, позволяющее смотреть статистику.
  1. Топология системы

Разрабатываемая система состоит одного веб-фронтэнд приложения и 5 подсистем:

* сервис аккаунтов;
* сервис резервирования;
* сервис лояльности;
* сервис оплаты;
* сервис статистики.

На рисунке 1.1 приведена топология разрабатываемой системы.

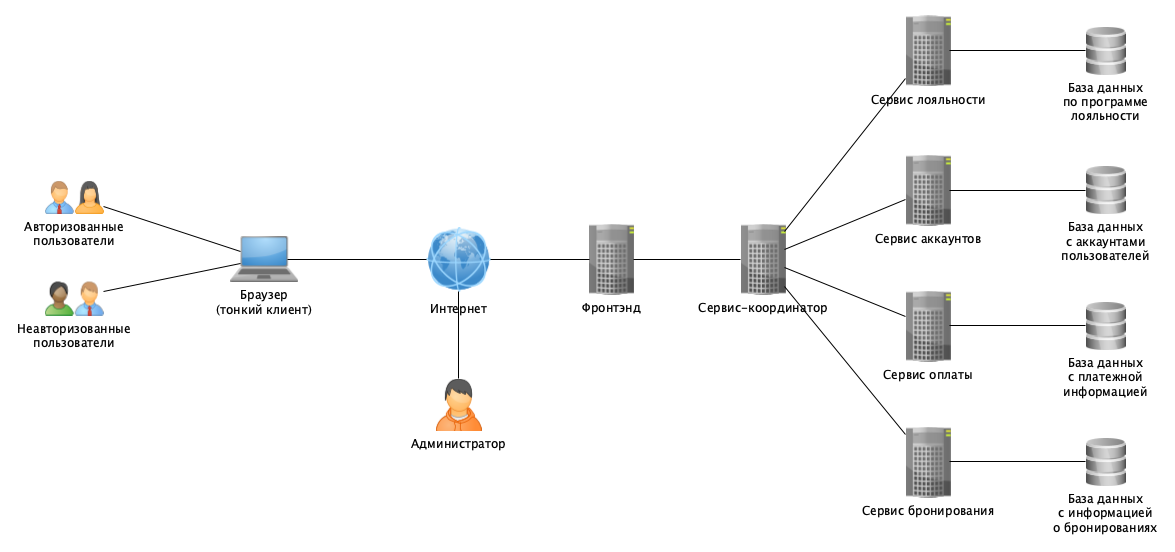


Рисунок 1.1 — Топология системы.

**Фронтэнд-приложение** принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа выполняет запросы к микросервисам бэкенда, агрегирует ответы и отсылает их пользователю.

**Сервис-координатор** — единая точка входа и межсервисной коммуникации.

**Сервис-регистрации и авторизации** отвечает за:

* возможность регистрации нового клиента;
* аутентификацию пользователя (клиента и администратора);
* авторизацию пользователя;
* выход из сессии.

**Сервис бронирования** реализует следующие функции:

* получение списка всех отелей;
* получение информации о конкретном отеле;
* получение, создание, отмена бронирования.

**Сервис оплаты** реализует функции:

* проведение платежа от клиента к системе;
* получение статуса оплаты;
* отмену платежа.

**Сервис лояльности** отвечает за ведение статистики по количеству бронирований всех клиентов, на основе которой для каждого пользователя в индивидуальном порядке предоставляется скидка на будущие заказы.

* 1. Требования к программной реализации

1. Требуется реализовать СОА (сервис-ориентированную архитектуру) [2] для реализации системы.
2. К реализации фронтэнд приложений должны быть предъявлены следующие требования:
   1. Фронтэнд должен является SPA и возвращаться пользователю с сервера в виде одного документа.
   2. Фронтэнд должен является посредником между пользователем и бекэндом. Он должен конструировать запросы, отправлять их, и обрабатывать ответы, выдавая их в виде HTML страниц.
3. К реализации бекэнд-приложений должны быть предъявлены следующие требования:
   1. Прием и возврат данных должен осуществляться в формате JSON [3] по протоколу HTTP (необходимо придерживать RESTful [4]).
   2. Доступ к базам данных должен осуществляться только из подсистем, работающих напрямую с данными ее таблиц.
   3. Выделить Gateway Service как единую точку входа и межсервисной коммуникации. В системе не должно осуществляться горизонтальных запросов.
   4. При недоступности систем портала должна осуществляться деградация функционала или выдача пользователю сообщения об ошибке.
   5. Валидацию входных данных необходимо проводить и на стороне пользователя, и на стороне фронтэнда. Микросервисы бекэнда не должны валидировать входные данные, поскольку пользователь не может к ним обращаться напрямую, они должны получать уже отфильтрованные входные данные.
4. Для запросов, выполняющих обновление данных на нескольких узлах распределенной системы, в случае недоступности одной из систем, необходимо выполнять полный откат транзакции.
5. Приложение должно поддерживать возможность горизонтального и вертикального масштабирования за счет увеличения количества функционирующих узлов и совершенствования технологий реализации компонентов и всей архитектуры системы.
6. Код необходимо хранить на Github.
7. Gateway Service должен запускаться на порту 8080, остальные сервисы

запускать на портах 8030, 8040, 8050, 8060, 8070. Фронтэнд запускается на порту 3000.

1. Каждый сервис должен быть завернут в docker.
   1. Функциональные требования к подсистемам

В разрабатываемой системе существуют следующие подсистемы: фронтэнд, бекэнд-координатор, бекэнд регистрации и авторизации, бекэнд бронирования, бекэнд оплаты, бекэнд лояльности.

**Фронтэнд** – серверное приложение, предоставляет пользовательский интерфейс и внешний API системы, при разработке которого нужно учитывать следующее:

* он должен принимать запросы по протоколу HTTP и формировать ответы

пользователям в формате HTML;

* в зависимости от типа запроса должен отправлять последовательные запросы в соответствующие микросервисы;
* запросы к микросервисам необходимо осуществлять по протоколу HTTP;
* данные необходимо передавать в формате JSON [5].

**Сервис-координатор** – серверное приложение, через которое проходит весь поток запросов и ответов, должен соответствовать следующим требованиям разработки:

* принимать и возвращать данные в формате JSON по протоколу HTTP;
* накапливать статистику запросов в случае, если система не ответила N раз, то в N + 1 раз вместо запроса сразу отдавать fallback. Через некоторое время выполнить запрос к реальной системе, чтобы проверить её состояние;
* выполнять проверку существования клиента, также регистрацию и аутентификацию пользователей;
* получение списка всех отелей;
* получение информации и обновление данных о зарегистрированном пользователе;
* оформление и отмена созданного ранее бронирования;
* получение данных о бронированиях пользователя;

**Сервис-регистрации и авторизации** должен реализовывать следующие функциональные возможности:

* принимать и возвращать данные в формате JSON по протоколу HTTP;
* возможность регистрации нового клиента и обновление данных уже существующего;
* проверка существования клиента;
* обеспечение авторизации пользователя через аккаунт как в системе.

**Сервис бронирования** реализует следующие функции:

* получение и отправка данных в формате JSON по протоколу HTTP;
* получение таких данных, как:

**—**список всех отелей;

**—**информация о конкретном отеле по её идентификатору;

**—**информация о свободных номерах по заданным реквизитам;

**—**все бронирования, зарегистрированные на конкретного клиента;

**—**информация о конкретном бронирования по его идентификатору;

* вычисление стоимости бронирования за указанный период в выбранном отеле;
* создание, редактирование и отмена бронирования.

**Сервис оплаты** реализует функции:

* получение и отправка данных в формате JSON по протоколу HTTP;
* предоставления информации об оплате по её идентификатору;
* проведения оплаты и её отмена;
* получения и обновления статуса оплаты.

**Сервис лояльности** должен реализовывать представленные такие функциональные возможности, как:

* получение и отправка ответов на запросы в формате JSON по протоколу HTTP;
* получение величины скидки по конкретному пользователю;
* получение детальной информации о конкретном участнике программы лояльности;
* обновление числа бронирований и статуса по программе лояльности (предусмотреть, как повышение, так и понижение в случае отмены бронирования);
* внесение изменений в размер скидки по конкретному пользователю.
  1. Вывод

В данном разделе была произведена постановка задачи, описана разрабатываемая система, сформулированы общие требования к системе, а также требования к функциональным характеристикам, приведены функциональные требования к системе с точки зрения пользователя, описаны входные и выходные данные системы, сформулированы требования к программной реализации и функциональные требования к подсистемам. Также была приведена топология разрабатываемой системы.

1. Конструкторский раздел

В данном разделе будет приведен концептуальный дизайн системы, сценарии функционирования, диаграммы прецедентов, спецификация сценариев и классов, структура базы данных.

* 1. Концептуальный дизайн

На рисунке 2.1 представлена IDEF0-диаграмма системы верхнего уровня.

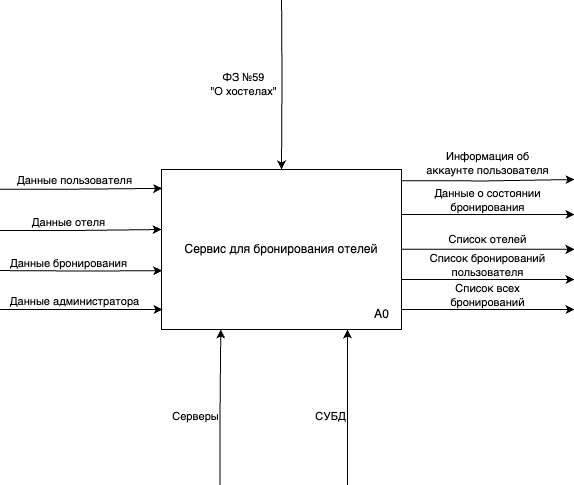


Рис. 2.1 – IDEF0-диаграмма системы для бронирования отелей.

На рисунке 2.2 представлена IDEF0-диаграмма системы первого уровня, описывающая процесс бронирования номера в отеле пользователем.

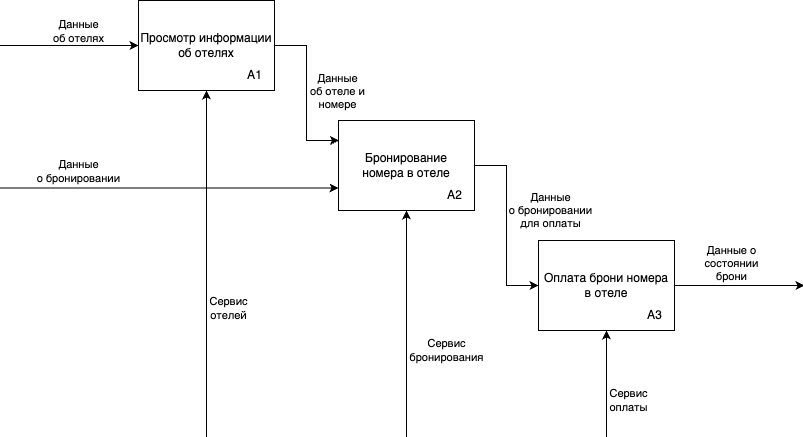


Рис. 2.2 – IDEF0-диаграмма процесса бронирования номера в отеле.

* 1. Сценарии функционирования системы

**Регистрация клиента**

1. Пользователь нажимает на кнопку «Зарегистрироваться» в интерфейсе.
2. Пользователь перенаправляется на страницу, которая содержит поля для заполнения его данных.
3. Пользователь вводит данные в форму и для завершения регистрации нажимает на кнопку «Готово», тем самым подтверждая верность своих данных, а также согласие на их обработку и хранение.
4. Если пользователь с введенным для регистрации логином уже существует, то клиент перенаправляется на страницу ошибки. При успешной регистрации клиент попадает на страницу своего профиля в системе.

**Авторизация клиента**

1. Пользователь нажимает на кнопку «Войти» в интерфейсе.
2. Пользователь перенаправляется на страницу авторизации, которая содержит поля для заполнения логина и пароля.
3. Пользователь завершает работу с формой авторизации нажатием кнопки «Готово».
4. При обнаружении ошибки в данных, пользователю высвечивается сообщение о некорректности введенных данных и возникшей ошибке.

**Бронирование отеля**

1. Клиент открывает страницу с отелями.
2. Клиент выбирает нужный ему отель и нажимает кнопку «бронировать».
3. Открывается окно с вводом нужных дат — клиент вводит даты заезда и выезда и нажимает на кнопку отправки запроса о бронировании.
4. Окно закрывается, клиент может найти созданное бронирование в своем личном кабинете.
5. Если клиент не хочет оформлять бронирование, он при выборе дат заезда и выезда может тыкнуть в свободное от окна пространство — окно закроется и оформление брони завершится.

**Отмена бронирования**

1. Клиент открывает страницу с бронированиями в личном кабинете.
2. Клиент выбирает бронь, которую хочет отменить.
3. Клиент нажимает кнопку «отменить».
4. Происходит отмена бронирования — квитанция об оплате отменяется, количество бронирований в базе данных лояльности уменьшается на единицу.
   1. Диаграммы прецедентов

В системе выделены три роли — пользователь (неавторизованный пользователь), клиент (авторизованный пользователь), администратор. На рисунках 2.3-2.4 представлены диаграммы прецедентов для каждой из ролей. Диаграмма прецедентов для пользователя и клиента объединены в одну. В таблицах 4-5 описаны сценарии функционирования наиболее значимых прецедентов.

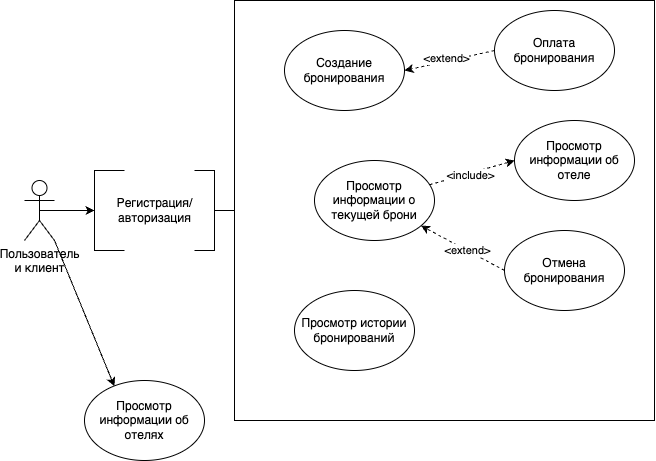


Рис. 2.3 – диаграмма прецедентов для пользователя и клиента.

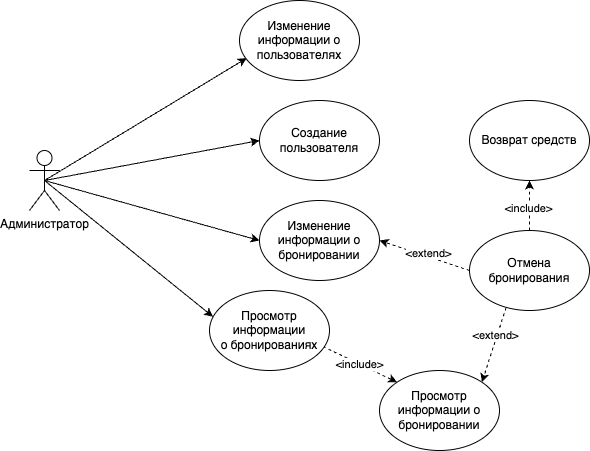


Рис. 2.4 – Диаграмма прецедентов для администратора.

* 1. Спецификация сценариев

**Спецификация сценария регистрации пользователя.**

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь видит кнопку для входа/регистрации и нажимает на нее. | Открывается страница для ввода данных пользователя. |
| Пользователь вводит нажимает на кнопку «Зарегистрироваться» | Открывается форма регистрации. |
| Пользователь вводит свои данные и нажимает «Готово». | Аккаунт пользователя создается, открывается страница входа. |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь видит кнопку для входа/регистрации и нажимает на нее. | Открывается страница для ввода данных пользователя. |
| Пользователь вводит нажимает на кнопку «Зарегистрироваться» | Открывается форма регистрации. |
| Пользователь вводит не все необходимые поля. | Высвечивается сообщение об ошибке. |

**Спецификация сценария авторизации пользователя.**

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь видит кнопку для входа/регистрации и нажимает на нее. | Открывается страница для ввода данных пользователя. |
| Пользователь вводит свои данные для входа | Открывается главная страница со списком отелей для бронирования. |

|  |  |
| --- | --- |
| **Альтернативный ход сценария №1 (только для администратора)** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь видит кнопку для входа/регистрации и нажимает на нее. | Открывается страница для ввода данных пользователя. |
| Пользователь вводит некорректные данные для входа | Высвечивается сообщение об ошибке. |

**Спецификация сценария создания бронирования.**

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь выбирает подходящий ему отель. |  |
| Пользователь нажимает на кнопку «Бронировать» рядом с заинтересовавшим его отелем. | Открывается форма для выбора дат заезда и выезда. |
| Пользователь выбирает даты и нажимает на кнопку отправки запроса. | Форма для выбора дат закрывается, бронирование создано. |

|  |  |
| --- | --- |
| **Альтернативный ход сценария №1** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь выбирает подходящий ему отель. |  |
| Пользователь нажимает на кнопку «Бронировать» рядом с заинтересовавшим его отелем. | Открывается форма для выбора дат заезда и выезда. |
| Пользователь передумал бронировать этот номер и нажимает в свободное от формы пространство. | Форма закрывается. |

**Спецификация сценария отмены бронирования.**

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действия пользователя** | **Отклик системы** |
| Пользователь заходит на web-портал. | Открывается главная страница портала со списком отелей. |
| Пользователь переходит в личный кабинет. | Открывается страница с личным кабинетом. |
| Пользователь нажимает на кнопку с бронированиями. | Открывается форма со всеми бронированиями пользователя. |
| Пользователь выбирает бронирование, которое хочет отменить и нажимает «Отменить». | Бронирование отменяется, счетчик количества бронирований в системе снижается на единицу. |

* 1. Спецификации классов

На приведенном ниже рисунке 2.5 представлена диаграмма классов для разработки микросервиса бронирования.

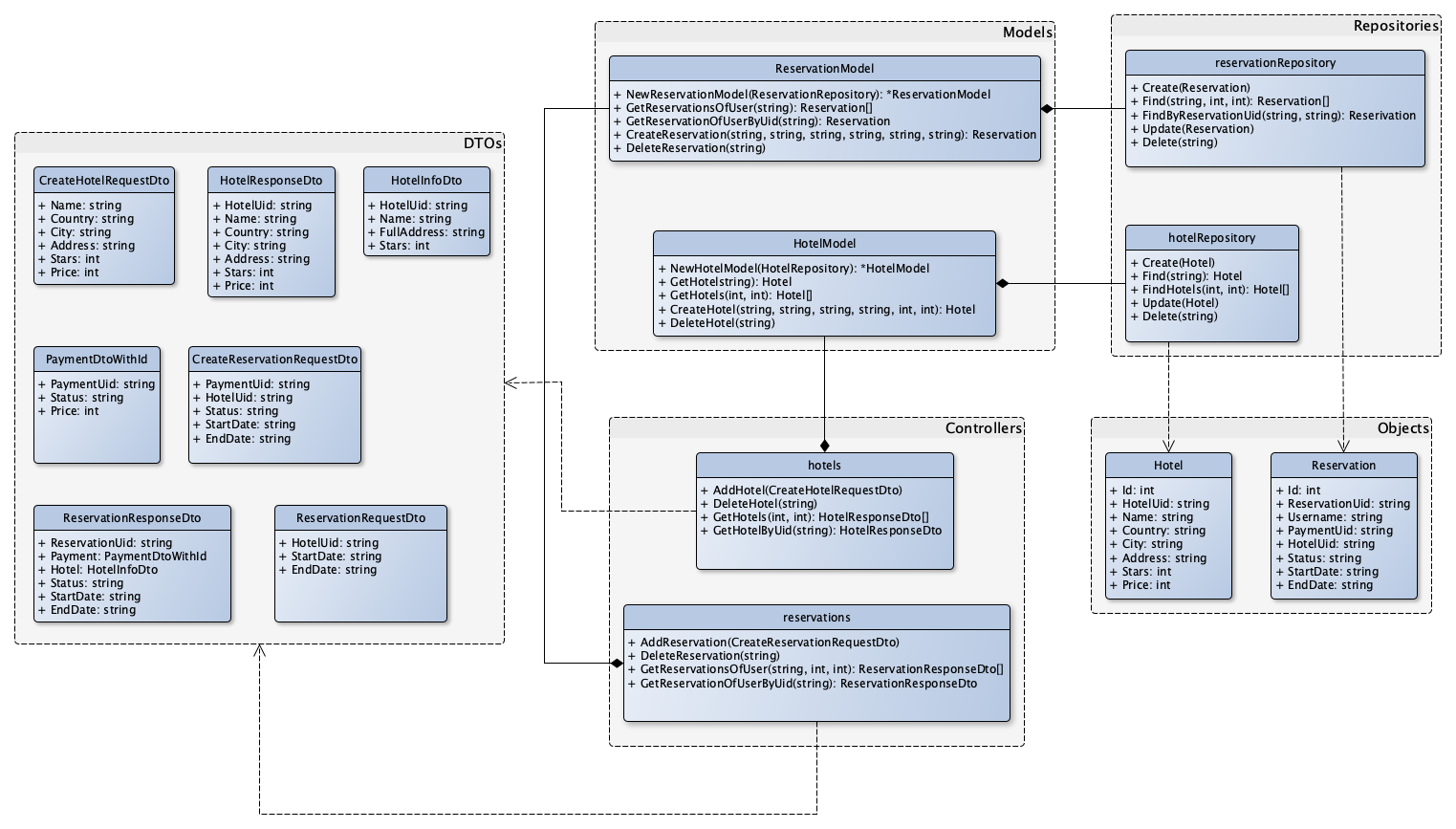


Рис. 2.5 – Диаграмма классов.

Ниже описаны атрибуты классов уровня сущностей БД.

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | | |
| **Имя** | **Тип** | **Описание** |
| Id | integer | Идентификатор (в СУБД) |
| HotelUid | string | Идентификатор отеля |
| Name | string | Название отеля |
| Country | string | Страна |
| City | string | Город |
| Address | string | Адрес |
| Stars | integer | «Звездность» отеля |
| Price | integer | Стоимость бронирования в отеле |

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | | |
| **Имя** | **Тип** | **Описание** |
| Id | integer | Идентификатор (в СУБД) |
| ReservationUid | string | Идентификатор бронирования |
| Username | string | Имя пользователя |
| PaymentUid | string | Идентификатор записи об оплате |
| HotelUid | string | Идентификатор отеля, в котором произведено бронирование |
| Status | string | Статус бронирование |
| StartDate | integer | Дата заезда |
| EndDate | integer | Дата выезда |

Классы CreateHotelRequestDto, HotelResponseDTO, HotelInfoDto, PaymentDtoWithId, CreateReservationRequestDto, ReservationResponseDto, ReservationRequestDto представляют объекты для передачи данных между классами бизнес-логики. Атрибутивный состав CreateHotelRequestDto, HotelResponseDTO, HotelInfoDto, CreateReservationRequestDto, ReservationResponseDto, ReservationRequestDto аналогичен составу ассоциированных с ними сущностей базы данных. Атрибуты вспомогательного класса PaymentDtoWithId приведены ниже.

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | | |
| **Имя** | **Тип** | **Описание** |
| PaymentUId | string | Идентификатор записи об оплате |
| Status | string | Статус оплаты (Оплачено/Отменено) |
| Price | integer | Сумма |

Классы hotelRepository и reservationRepository отвечают за взаимодействие микросервиса с базой данных. Методы каждого приведены в таблицах 10-11.

|  |  |
| --- | --- |
| **Методы** | |
| **Название** | **Описание** |
| Create(Hotel) | param: hotel [Hotel – in] — экземпляр отеля  *Добавить запись об отеле в БД* |
| Find(string): Hotel | param: hotelUid [string – in] — идентификатор отеля  *Найти запись об отеле по идентификатору отеля* |
| FindHotels(int, int): Hotel[] | Param: page [int – in] — страница для пагинации, size [int – in] — количество элементов на странице  *Получить все записи об отелях из БД* |
| Update(Hotel) | param: hotel [Hotel – in] — экземпляр отеля  *Обновить информацию об отеле в БД* |
| Delete(string) | Param: hotelUid [string – in] — идентификатор отеля  *Удалить запись об отеле из БД* |

|  |  |
| --- | --- |
| **Методы** | |
| **Название** | **Описание** |
| Create(Reservation) | param: reservation [Rerservation – in] — экземпляр брони  *Добавить запись о бронировании в БД* |
| Find(string, string): Reservation | param: username [string – in] — имя пользователя, reservationUid [string – in] — идентификатор отеля  *Найти запись о конкретном бронировании пользователя* |
| Find(string, int, int): Reservation[] | Param: username [string – in] — имя пользователя, page [int – in] — страница для пагинации, size [int – in] — количество элементов на странице  *Получить все записи бронирований пользователя из БД* |
| Update(Reservation) | param: reservation [Reservation – in] — экземпляр брони  *Обновить информацию о бронировании в БД* |
| Delete(string) | Param: reservationUid [string – in] — идентификатор брони  *Отменить бронирование* |

* 1. Структура базы данных

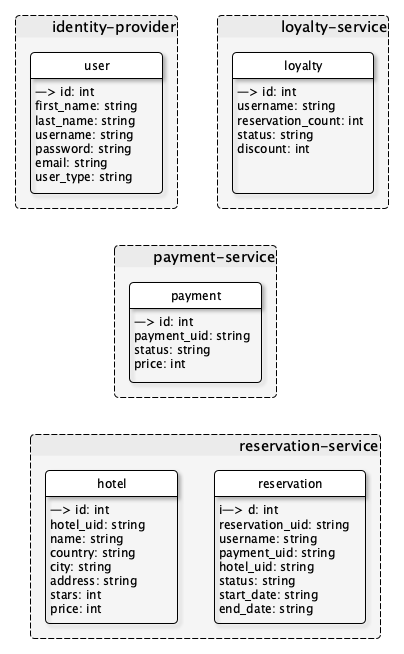


Рис. 2.6 – Схема БД.

* 1. Вывод

В данном разделе был приведен концептуальный дизайн системы, сценарии функционирования, диаграммы прецедентов, спецификация сценариев и классов, а также структура базы данных.

1. Технологический раздел

В данном разделе будет обоснован выбор операционной системы, СУБД, языка разработки компонентов, а также приведен дизайн пользовательского интерфейса.

* 1. Выбор операционной системы

Согласно требованиям технического задания, разрабатываемый портал должен обладать высокой доступностью, работать на типичных архитектурах ЭВМ (Intel x86, Intel x64, ARM), а также быть экономически недорогим для сопровождения. Таким образом, требования к ОС следующие.

* **Распространенность**. На рынке труда должно быть много специалистов, способных администрировать распределенную систему, работающую под управлением выбранной операционной системы.
* **Надежность**. Операционная система должна широко использоваться в стабильных проектах мирового уровня, как Google.com, Amazon.com. Эти компании обеспечивают высокую работоспособность своих сервисов, и на их опыт можно положиться.
* **Наличие требуемого программного обеспечения**. Выбор операционной системы не должен ограничивать разработчиков в выборе программного обеспечения, библиотек.
* **Цена**.

Под данные требования лучше всего подходит ОС Ubuntu [6]. *Ubuntu* —это дистрибутив, использующий ядро Linux. Как и все дистрибутивы Linux, Ubuntu является ОС с открытым исходным кодом, бесплатным для использования. Поставляется как в клиентской (с графическим интерфейсом), так и в серверной (без графического интерфейса) версиями. Ubuntu поставляется с современными версиями ПО. Преимуществом Ubuntu являются низкие требования к квалификации системных администраторов. Однако Ubuntu менее стабильна в работе.

* 1. Выбор СУБД

В соответствии с техническим заданием разработка бекэнда предусматривает следующие требования:

* **Безопасность хранения данных**. Несанкционированный доступ к данным клиентам должен быть невозможен.
* **Транзакционность**. Должен соблюдаться принцип «ACID» (Atomicity— Атомарность, Consistency — Согласованность, Isolation — Изолированность, Durability — Надежность). Атомарность гарантирует, что транзакция не может быть зафиксирована частично. Согласованность — что успешное завершение транзакции оставит систему в согласованном состоянии. Изолированность — что параллельно выполняемые транзакции не будут влиять друг на друга. Надежность — что успешно завершенная транзакция будет зафиксирована, а в случае сбоя, после восстановления системы, результаты транзакции не будут утеряны.
* **Масштабируемость**. Выбранная СУБД должна поддерживать репликацию, шардирование.

PostgreSQL [7] – реляционная система управления базами данных. Она является некоммерческим ПО с открытым исходным кодом. Для работы с этой СУБД существуют библиотеки для таких распространенных языков программирования, как Python, Ruby, Perl, PHP, C, C++, Java, C#, Go. Она работает под управлением многих операционных систем: Linux, MacOS, Windows, FreeBSD, Solaris и др. По сравнению с MySQL система PostgreSQL лучше работает с репликацией, так как в ней существует журнал (средство восстановления системы в случае сбоя) физической модификации страниц. PostgreSQL осуществляет асинхронную репликацию типа «ведущий — ведомый».

Выбор СУБД PostgreSQL для хранения данных разрабатываемой системы обеспечит надежность, безопасность и масштабируемость.

* 1. Выбор языка разработки компонентов

Исходя из приведенных требований к системе, можно выявить следующие требования к языку программирования.

* **Наличие разнообразных библиотек**. Использование готовых библиотек ускоряет разработку программного обеспечения. Также важно, что благодаря использованию распространенных оттестированных библиотек снижается вероятность ошибки. Это повышает надежность программного обеспечения.
* **Совместимость с выбранными ранее технологиями**. Выбранный язык должен уметь взаимодействовать с ОС Linux, СУБД PostgreSQL.
* **Высокая скорость разработки**. На ранних этапах разработки портала технические требования часто меняются. Язык программирования должен позволять как можно быстрее вносить изменения в коды программ.

В качестве языка программирования, соответствующему приведенным выше требованиям, был взят язык программирования Golang [8], разработанный компанией Google, ориентированный на создание микросервисов. Среди его плюсов можно выделить следующие:

* **Простота** – чёткие синтаксические правила и понятная семантика.
* **Объектно-ориентированный подход в формате «утиной» типизации**.
* **Безопасность**. Обойти или взломать механизмы защиты крайне сложно.
* **Производительность**. Язык программирования Golang сочетает в себе скорость языка Си с синтаксической простотой языка Python.
* **Надёжность**. Компилятор способен выявить ошибки ещё до выполнения

кода, то есть на ранних стадиях.

* 1. Дизайн пользовательского интерфейса

Пользовательский интерфейс в разрабатываемой системе представляет собой Web-интерфейс, доступ к которому осуществляется через браузер (тонкий клиент). На рисунках 3.1-3.6 приведен разработанный пользовательский интерфейс.

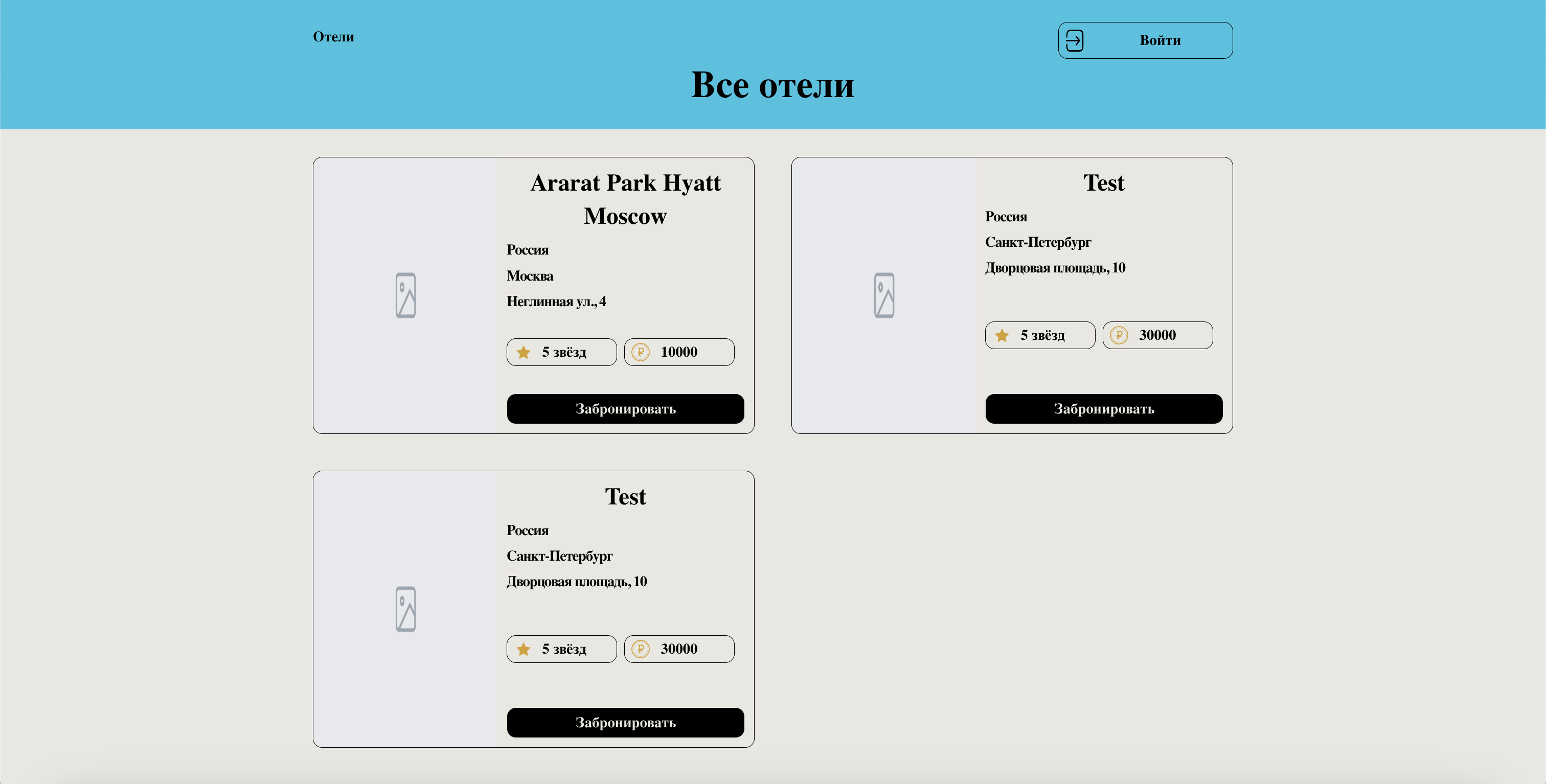


Рис. 3.1 – Главная страница.

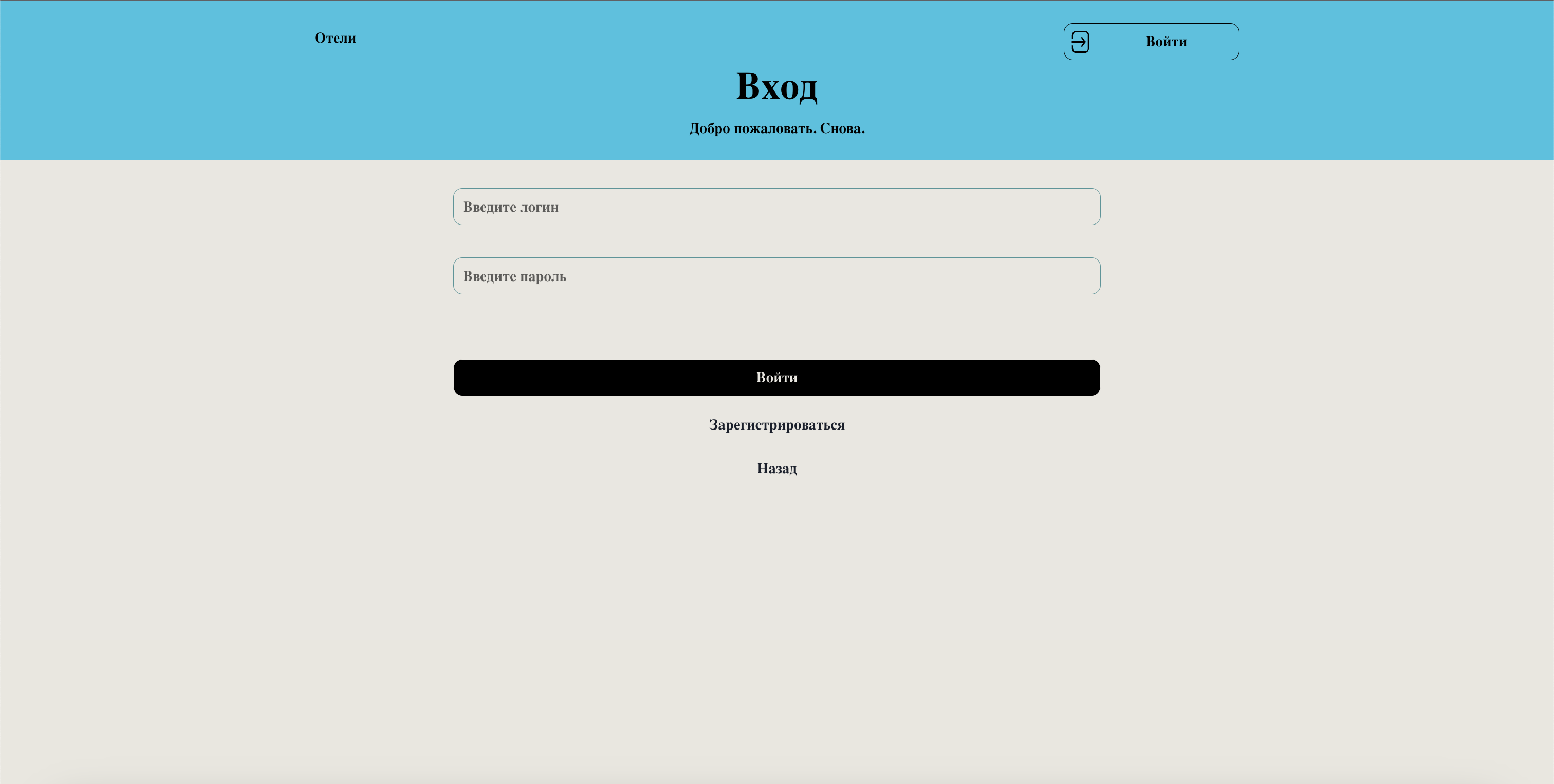


Рис. 3.2 – Страница входа.

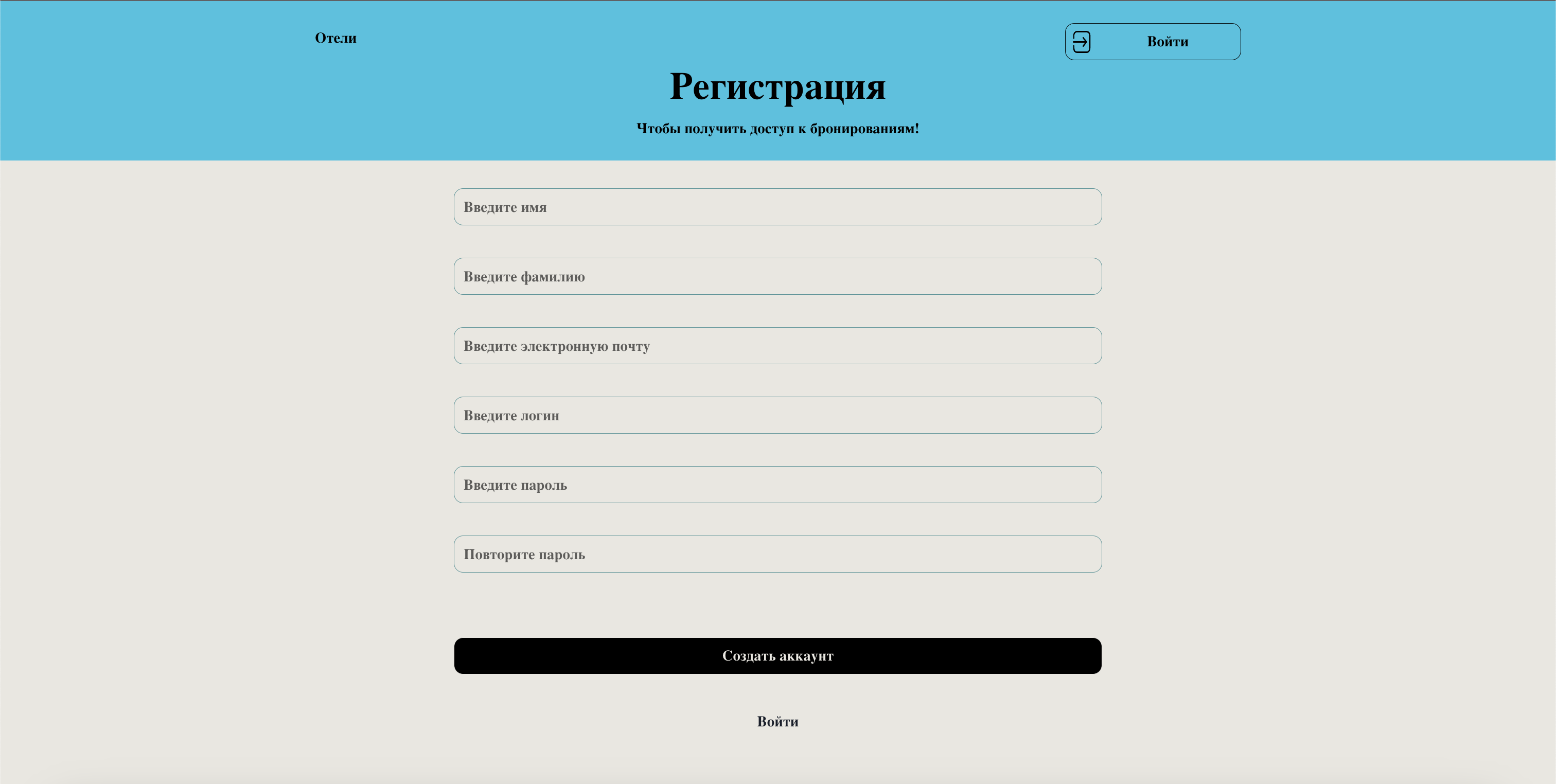


Рис. 3.3 – Страница регистрации.

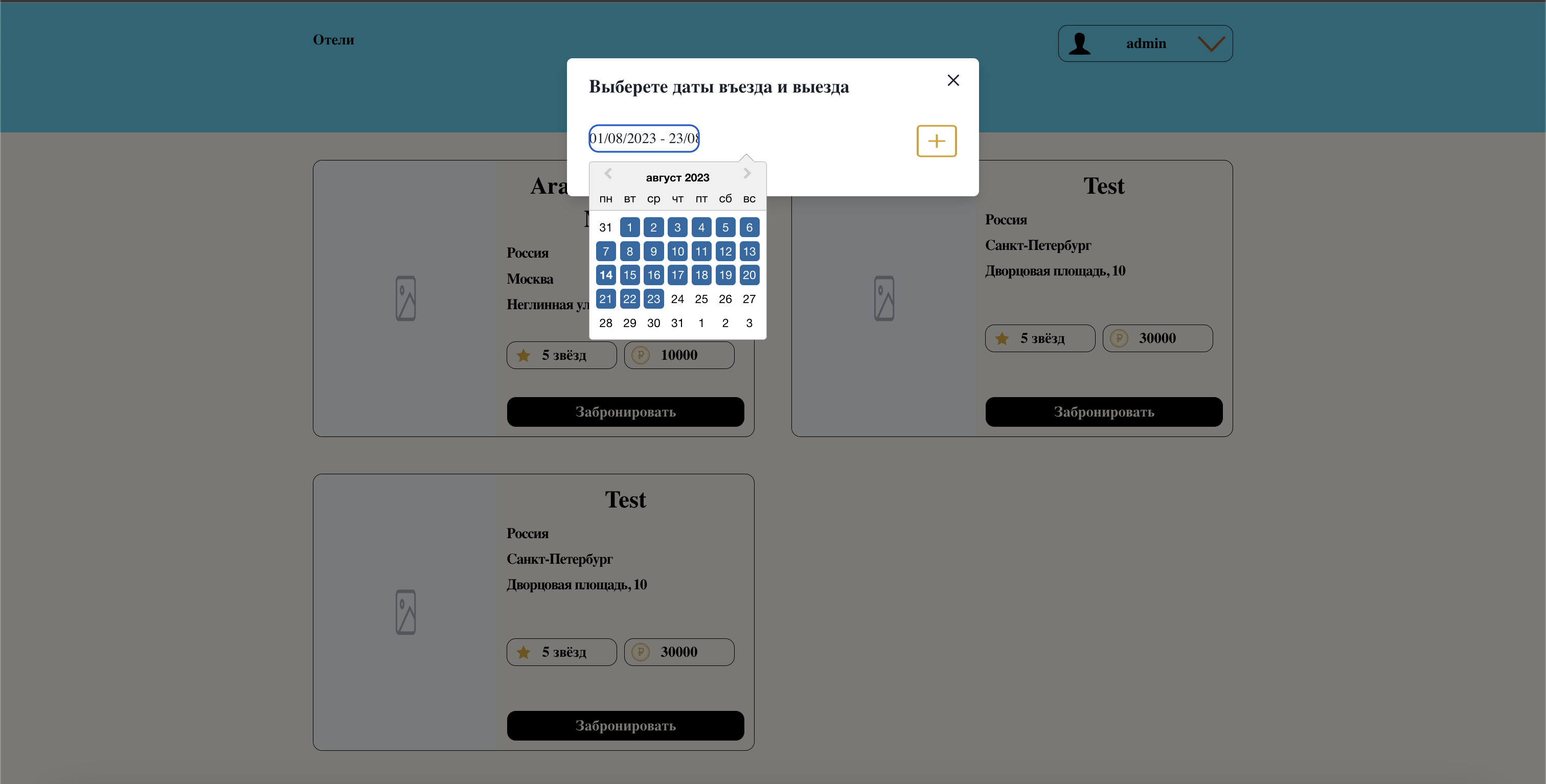


Рис. 3.4 – Окно бронирования отеля.

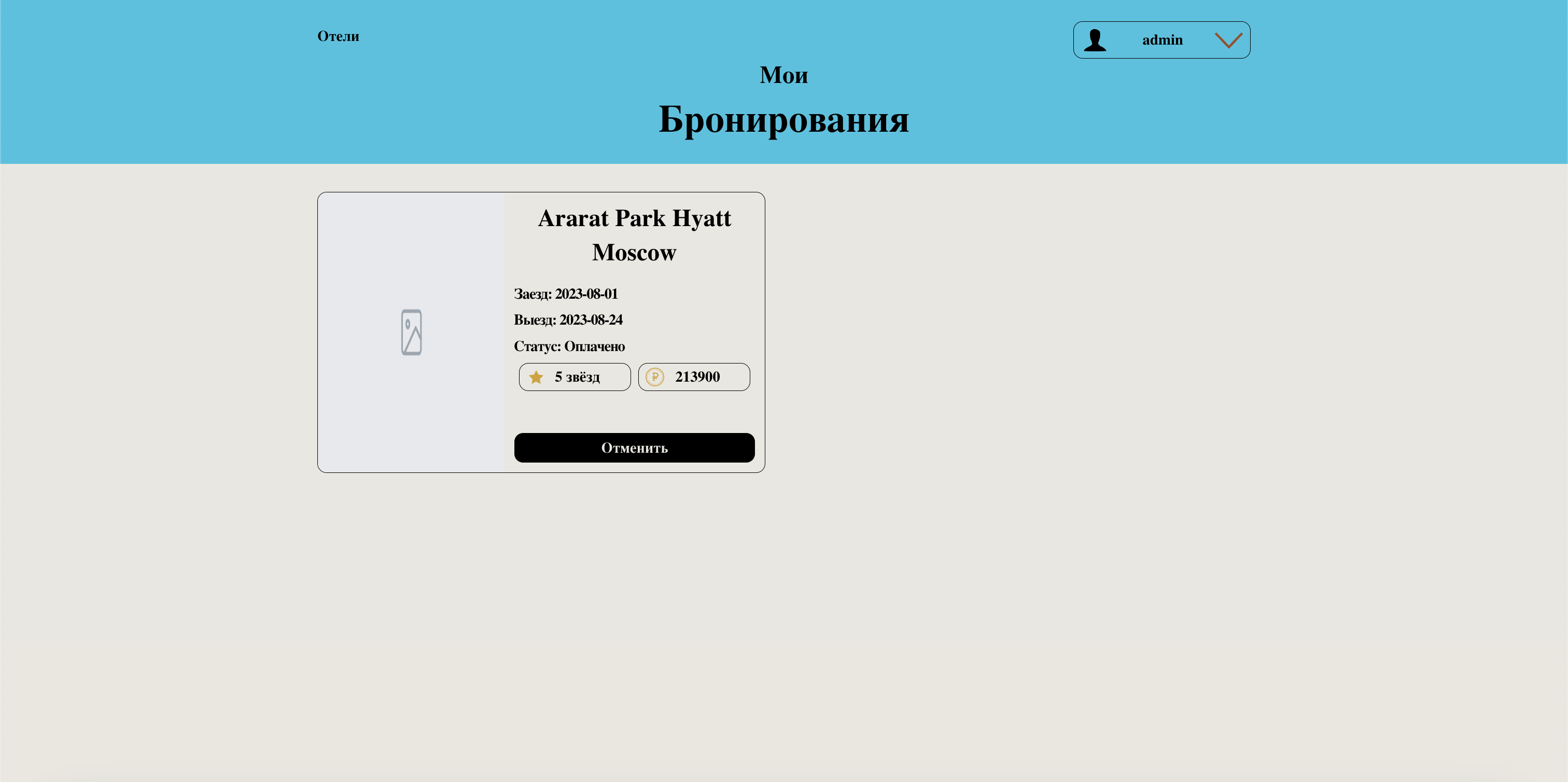


Рис. 3.5 – Страница с бронированиями пользователя.

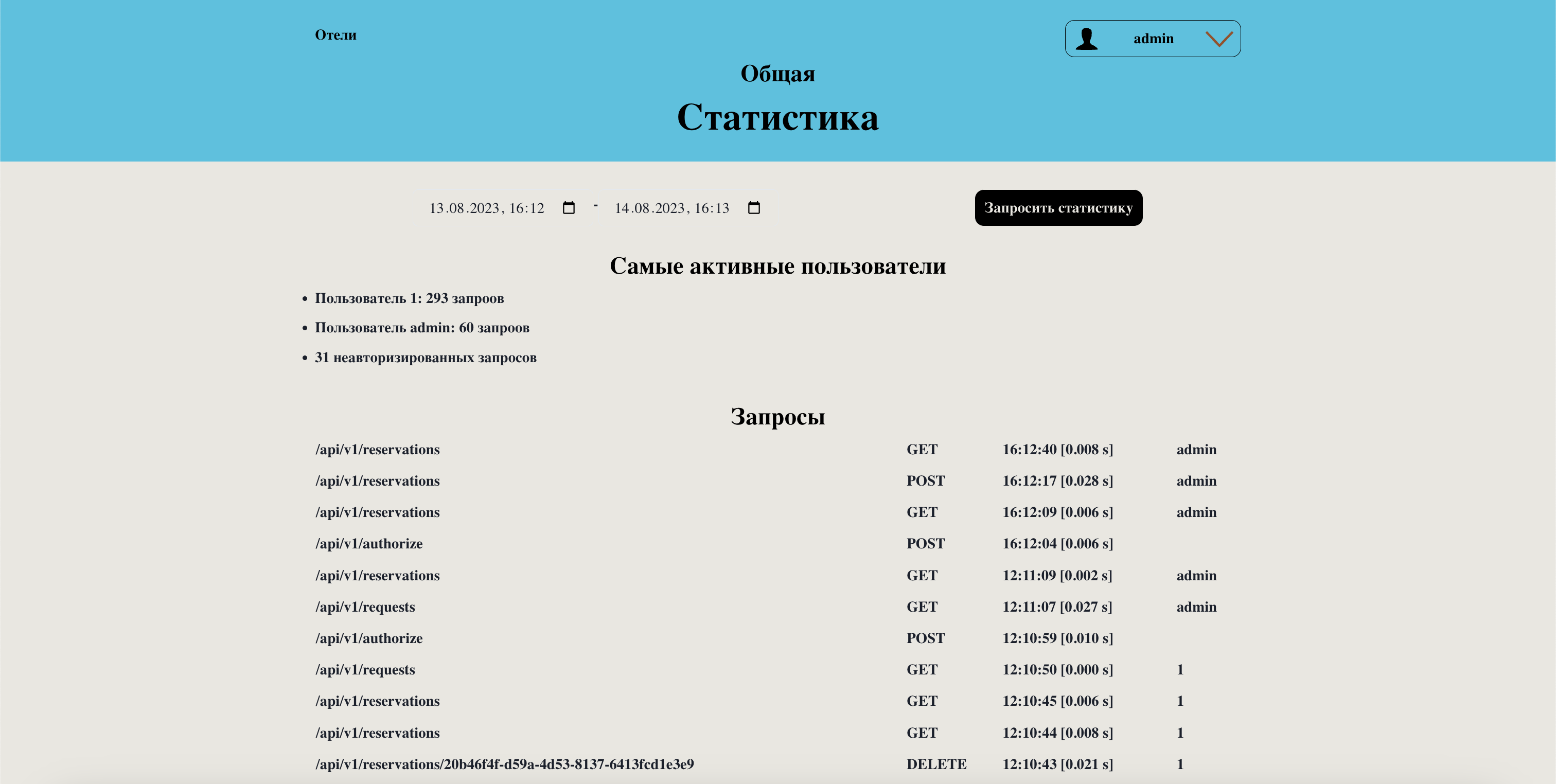


Рис. 3.6 – Страница со статистикой (доступна администратору).

* 1. Вывод

В данном разделе был обоснован выбор операционной системы, СУБД, языка разработки компонентов, а также приведен дизайн пользовательского интерфейса.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана распределенная система, позволяющая производить бронирование отелей. Для достижения поставленной цели были решены следующие задачи:

* сформулированы основные требования к системе;
* описаны требования к подсистемам;
* описана архитектура системы, а также сценарии ее функционирования;
* выбраны и обоснованы инструменты для разработки ПО;
* реализован программный продукт.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. Анализ рынка гостиничных услуг в России в 2018-2022 гг., прогноз на 2023-2027 гг. в условиях санкций [Электронный ресурс]. – Режим доступа: https://marketing.rbc.ru/research/27205/ (Дата обращения: 24.05.2023).
  2. Erl T. SOA design patterns (paperback). — Pearson Education, 2008.
  3. Bourhis P., Reutter J. L., Vrgoč D. JSON: Data model and query languages //Information Systems. – 2020. – Т. 89. – С. 101478.
  4. Ahmad I. et al. Implementation of RESTful API Web Services Architecture in Takeaway Application Development //2021 1st International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS). – IEEE, 2021. – С. 132-137.
  5. Docker Documentation [Электронный ресурс]. – Режим доступа: https://docs.docker.com/ (Дата обращения: 30.05.2023).
  6. Ubuntu Documentation [Электронный ресурс]. – Режим доступа: https://ubuntu.com/ (Дата обращения: 30.05.2023).
  7. PostgreSQL Documentation [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/ (Дата обращения: 30.05.2023).
  8. Golang Documentation [Электронный ресурс]. – Режим доступа: https://go.dev/doc/ (Дата обращения: 30.05.2023).