



**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)**

**РАСПРЕДЕЛЁННАЯ СИСТЕМА БРОНИРОВАНИЯ НОМЕРОВ  
ГОСТИНИЦ СЕТИ APARTLUX**

**Техническое задание**

**Листов 10**

Инв.№ подл.	Подп. и дата	Взаим.инв.№	Инв.№ дубл.	Подп. и дата

## **Введение**

По данным «Анализа рынка гостиничных услуг в России», подготовленного агентством BusinesStat, занимающимся исследованием рынка Российской Федерации и всех стран бывшего СНГ, подобным сервисом в 2022 году воспользовались 62.4 млн чел, что на 63% превысило значение 2020 года (38.3 млн человек). Такой рост объясняется сокращением выездного туризма и развитием внутреннего на фоне геополитической обстановки. Также численность гостиничных учреждений к концу 2022 года достигла 22.01 тыс., в то время как в 2020 она составляла 20.41 тыс.

Ввиду сильной конкуренции для привлечения клиентов каждая компания стремится улучшить свой сервис, особое внимание уделяется системам бронирования.

Данное техническое задание составлено для разработки распределённой системы для бронирования номеров гостиниц сети Apartlux. Техническое задание выполнено на основе ГОСТ 19.201–78 «ЕСПД. Техническое задание. Требования к содержанию и оформлению».

## **Глоссарий**

- 1) Узел системы – региональный сервер, содержащий данные авторов и читателей указанного региона.
- 2) «Горячее» переконфигурирование системы – способность системы применять изменения без перезапуска и перекомпиляции.
- 3) Медиана времени отклика – среднее время предоставления данных пользователю.
- 4) Валидация данных – проверка данных на соответствие заданным условиям и ограничениям.
- 5) REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
- 6) Фронтенд – серверное приложение, принимающее запросы от пользователя. На каждый из типов запросов определяется, как организовать его выполнение. Принимает запросы, анализирует их и в соответствии с заложенным алгоритмом выполняет запросы к бекенду.
- 7) Бекенд – серверное приложение, выполняющее определенную задачу, например, взаимодействие с СУБД. Бекенды принимают запросы от фронтена.
- 8) ПО – программное обеспечение.

## **Основания для разработки**

Разработка ведётся в рамках выполнения лабораторных работ по курсу «Методология программной инженерии» на кафедре «Программное обеспечение ЭВМ и информационные технологии» факультета «Информатика и системы управления» МГТУ им. Н.Э. Баумана.

## **Назначение разработки**

Разрабатываемая система должна предоставлять пользователям возможность бронирования номеров сети Apartlux, в которую входят 14 гостиниц в Москве на таких станциях метро, как Бабушкинская, Рижская, Тверская, Авиамоторная и другие. Должен быть предусмотрен поиск подходящих номеров по таким параметрам, как этаж, дата и продолжительность бронирования, стоимость, число мест, наличие двуспальной кровати. В зависимости от количества сделанных ранее заказов система должна рассчитывать скидку на новые согласно условиям программы лояльности.

## **Существующие аналоги**

У сети Apartlux уже есть действующий с 2011 года сайт для бронирования, который имеет ряд недостатков. При его создании разработчики придерживались подходом монолитной архитектуры, поэтому сейчас компания столкнулась с такими трудностями, как:

- 1) проблематичность масштабирования;
- 2) сложность внедрения появившихся технологий, которые используются повсеместно;
- 3) внесение даже незначительного изменения в функциональность существенно усложняет и замедляет разработку.

В то время как на российском рынке гостиничных услуг появляется всё

больше компаний, например, Radisson, Azimut Hotels, Hilton, остановивших свой выбор на микросервисной архитектуре и внедряющих современные технологии: PostgreSQL, MongoDB, Kafka, ELK stack и т.д. У приведённых сетей отелей и гостиниц есть общий недостаток: непрозрачная программа лояльности, которая направлена лишь на ограниченный круг лиц.

По сравнению с существующим сайтом и указанными аналогами разрабатываемый проект должен иметь следующие преимущества:

- 1) в основе должна лежать микросервисная архитектура, решающая сложности с масштабированием, обслуживанием и внесением изменений в функциональность;
- 2) понятная бонусная программа, ориентированная на каждого из клиентов.

## **Описание системы**

Разрабатываемый сервис должен представлять собой распределённую систему для бронирования номеров гостиниц сети Apartlux. Если клиент хочет оформить бронь, ему необходимо зарегистрироваться, указав информацию: фамилия, имя, отчество, дата рождения, номер телефона, 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. На рисунке 1.1 отображена схема предметной области.

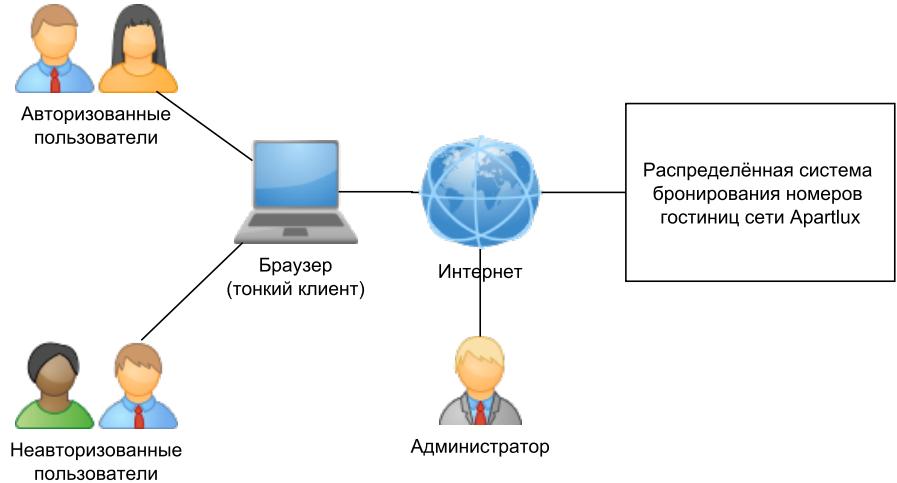


Рисунок 1.1 – Схема предметной области.

## Общие требования к системе

Требования к системе следующие.

- 1) Разрабатываемое ПО должно поддерживать функционирование системы в режиме 24 часов, 7 дней в неделю, 365 дней в году (24/7/365) со среднегодовым временем доступности не менее 99.9%. Допустимое время, в течении которого система недоступна, за год должна составлять  $24 \cdot 365 \cdot 0.001 = 8.76$  ч.
- 2) Время восстановления системы после сбоя не должно превышать 15 минут.
- 3) Каждый узел должен автоматически восстанавливаться после сбоя.
- 4) Система должна поддерживать возможность «горячего» переконфигурирования системы. Необходимо предусмотреть поддержку добавления нового узла во время работы системы без рестарта.
- 5) Обеспечить безопасность работоспособности за счёт отказоустойчивости узлов.

## **Требования к функциональным характеристикам**

- 1) По результатам работы модуля сбора статистики медиана времени отклика системы на запросы пользователя на получение информации не должна превышать 3 секунд.
- 2) По результатам работы модуля сбора статистики медиана времени отклика системы на запросы, добавляющие или изменяющие информацию на портале не должна превышать 7 секунд.
- 3) Медиана времени отклика системы на действия пользователя должна быть менее 0.8 секунд при условии работы на рекомендованной аппаратной конфигурации, задержках между взаимодействующими сервисами менее 0.2 секунды и одновременном числе работающих пользователей менее 100 на каждый сервер, обслуживающий внешний интерфейс.
- 4) Система должна обеспечивать возможность запуска в современных браузерах: не менее 85% пользователей Интернета должны пользоваться ей без какой-либо деградации функционала.

## **Функциональные требования к системе с точки зрения пользователя**

Система должна обеспечивать реализацию следующих функций.

- 1) Регистрация и авторизация пользователей с валидацией вводимых данных как через интерфейс приложения, так и через социальные сети.
- 2) Аутентификация пользователей.
- 3) Разделение всех пользователей на три роли:
  - Пользователь (неавторизированный пользователь);
  - Клиент (авторизированный пользователь);
  - Администратор.
- 4) Предоставление возможностей **Пользователю, Клиенту, Администратору** представленных в таблице 1.

Таблица 1 – Функции пользователей

<b>Пользователь</b>	<ol style="list-style-type: none"><li>1. просмотр списка гостиниц, входящих в сеть;</li><li>2. просмотр информации о возможности бронирования номера гостиницы по заданным реквизитам;</li><li>3. получение информации об условиях программы лояльности;</li><li>4. регистрация в системе;</li><li>5. авторизация в системе;</li></ol>
<b>Клиент</b>	<ol style="list-style-type: none"><li>1. просмотр списка гостиниц, входящих в сеть;</li><li>2. просмотр информации о возможности бронирования номера гостиницы по заданным реквизитам;</li><li>3. получение информации об условиях программы лояльности;</li><li>4. авторизация в системе;</li><li>5. получение и изменение информации текущего аккаунта;</li><li>6. просмотр всех бронирований, зарегистрированных на имя текущего клиента;</li><li>7. получение детальной информации по конкретному бронированию текущего клиента;</li><li>8. бронирование отеля на имя текущего клиента;</li><li>9. отмена заказа, оформленного на имя текущего пользователя;</li><li>10. получение информации о статусе текущего пользователя в программе лояльности.</li></ol>

<b>Администратор</b>	<ol style="list-style-type: none"> <li>1. просмотр списка гостиниц, входящих в сеть;</li> <li>2. просмотр информации о возможности бронирования номера гостиницы по заданным реквизитам;</li> <li>3. получение информации об условиях программы лояльности;</li> <li>4. авторизация в системе;</li> <li>5. получение и редактирование информации о любом клиенте, зарегистрированном в системе;</li> <li>6. просмотр и редактирование всех оформленных бронирований;</li> <li>7. получение детальной информации по конкретному бронированию;</li> <li>8. бронирование отеля на зарегистрированного в системе пользователя;</li> <li>9. отмена любого оформленного заказа;</li> <li>10. получение информации о статусе в программе лояльности любого зарегистрированного в системе клиента;</li> <li>11. изменение доступных дат для бронирования;</li> <li>12. редактирование информации об условиях программы лояльности.</li> </ol>
----------------------	---

### **Входные данные**

Входные параметры системы представлены в таблице 2.

Таблица 2 – Входные данные

<b>Сущность</b>	<b>Входные данные</b>
Клиент / Администратор	<ol style="list-style-type: none"> <li>1. <i>фамилия, имя и отчество</i> не более 256 символов каждое поле;</li> <li>2. <i>дата рождения</i> в формате д/м/гггг;</li> <li>3. <i>логин</i> не менее 10 символов и не более 128;</li> <li>4. <i>пароль</i> не менее 8 символов и не более 128, как минимум одна заглавная и одна строчная буква, только латинские буквы, без пробелов, как минимум одна цифра;</li> <li>5. <i>номер телефона</i>;</li> <li>6. <i>электронная почта</i>;</li> </ol>
Гостиница	<ol style="list-style-type: none"> <li>1. <i>идентификатор</i>;</li> <li>2. <i>название</i> не более 256 символов;</li> <li>3. <i>страна</i>;</li> <li>4. <i>город</i>;</li> <li>5. <i>полный адрес</i> не более 1024 символа;</li> <li>6. <i>контактный телефон</i>;</li> <li>7. <i>электронная почта</i>;</li> <li>8. <i>описание</i> не более 2048 символов</li> <li>9. <i>количество звёзд</i></li> </ol>
Номер	<ol style="list-style-type: none"> <li>1. <i>идентификатор</i>;</li> <li>2. <i>идентификатор</i> соответствующей гостиницы;</li> <li>3. <i>число мест</i>;</li> <li>4. <i>этаж</i>;</li> <li>5. <i>стоимость</i>;</li> <li>6. <i>наличие двуспальной кровати</i>;</li> </ol>

*Продолжение на следующей странице*

<b>Сущность</b>	<b>Входные данные</b>
Бронирование	1. <i>идентификатор</i> ; 2. <i>логин клиента</i> , на которое оно оформлено; 3. <i>идентификатор гостиницы</i> ; 4. <i>идентификатор соответствующей платёжной операции</i> ; 5. <i>статус</i> (APPROVED/UNAPPROVED); 6. <i>дата въезда</i> ; 7. <i>дата выезда</i> ;
Оплата	1. <i>идентификатор</i> ; 2. <i>фамилия, имя, отчество</i> человека, совершившего операцию; 3. <i>статус</i> (PAID/CANCELED); 4. <i>сумма</i> ; 5. <i>дата и время</i> .

### **Выходные параметры**

Выходными параметрами системы являются web-страницы. В зависимости от запроса и текущей роли пользователя они содержат следующую информацию (таблица 3).

Таблица 3 – Выходные параметры

<b>Пользователь</b>	1. список гостиниц, которые входят в сеть Apartlux, указывается: <ul style="list-style-type: none"> <li>• <i>название</i>;</li> <li>• <i>полный адрес</i>;</li> <li>• <i>описание</i>;</li> <li>• контактная информация: <i>телефон</i> и <i>электронная почта</i>;</li> <li>• доступные для бронирования <i>номера</i> по заданным реквизитам;</li> </ul> 2. информация об условиях текущей программы лояльности;
---------------------	--

<b>Клиент</b>	<p>1. список гостиниц, которые входят в сеть Apartlux, указывается:</p> <ul style="list-style-type: none"> <li>• <i>название;</i></li> <li>• <i>полный адрес;</i></li> <li>• <i>описание;</i></li> <li>• контактная информация: <i>телефон</i> и <i>электронная почта;</i></li> <li>• доступные для бронирования <i>номера</i> по заданным реквизитам;</li> </ul> <p>2. информация об условиях текущей программы лояльности;</p> <p>3. детальная информация о пользователе, вошедшем в систему;</p> <ul style="list-style-type: none"> <li>• <i>фамилия, имя, отчество;</i></li> <li>• <i>дата рождения;</i></li> <li>• <i>логин;</i></li> <li>• <i>номер телефона;</i></li> <li>• <i>электронная почта;</i></li> <li>• <i>размер скидки;</i></li> </ul> <p>4. список оформленных бронирований на пользователя, вошедшего в систему, предоставляется информация о:</p> <ul style="list-style-type: none"> <li>• <i>дате въезда/выезда;</i></li> <li>• <i>полный адрес гостиницы и контактные данные;</i></li> <li>• информация о номере: <i>этаж, номер, количество мест;</i></li> <li>• <i>статус оплаты;</i></li> <li>• <i>статус бронирования;</i></li> <li>• <i>сумма;</i></li> </ul>
<b>Администратор</b>	<p>1. список гостиниц, которые входят в сеть Apartlux, указывается:</p> <ul style="list-style-type: none"> <li>• <i>название;</i></li> <li>• <i>полный адрес;</i></li> <li>• <i>описание;</i></li> <li>• контактная информация: <i>телефон</i> и <i>электронная почта;</i></li> <li>• доступные для бронирования <i>номера</i> по заданным реквизитам;</li> </ul>

2. информация об условиях текущей программы лояльности;

3. детальная информация о пользователе, вошедшем в систему;

- *фамилия, имя, отчество;*

- *дата рождения;*

- *логин;*

- *номер телефона;*

- *электронная почта;*

- *размер скидки;*

4. список оформленных бронирований, предоставляются такие данные, как:

- *идентификатор;*

- информация о клиенте, на которое оно оформлено: *логин, номер телефона, электронная почта;*

- *дата въезда/выезда;*

- информация о гостинице: её *идентификатор, полный адрес и контактные данные;*

- информация о номере: *этаж, номер, количество мест;*

- информация об оплате: *идентификатор, статус, ФИО оплатившего, дата и время операции;*

- *статус бронирования;*

- *сумма;*

5. список зарегистрированных в системе клиентов с указанием:

- *ФИО;*

- *логина;*

- *даты рождения;*

- *контактных данных;*

- *персональной скидки;*

- *число предыдущих бронирований.*

## Топология Системы

На рисунке 1.2 изображён один из возможных вариантов топологии разрабатываемой распределенной Системы.

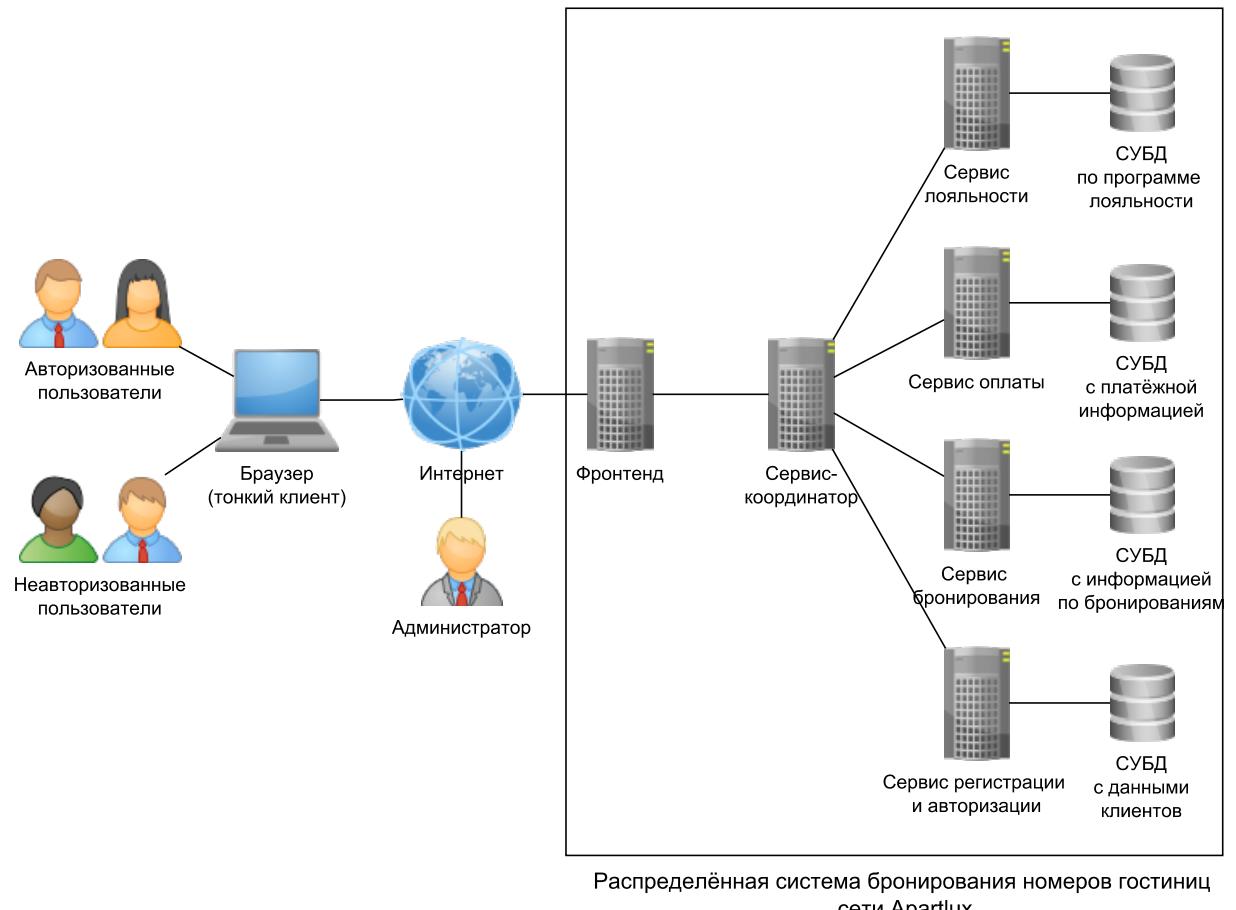


Рисунок 1.2 – Топология системы.

Система будет состоять из фронтенда и 5 подсистем:

- сервис-координатор;
- сервис регистрации и авторизации;
- сервис бронирования;
- сервис оплаты;
- сервис лояльности.

**Фронтенд** принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа выполняет запросы к микросервисам бекенда, 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 гостиниц, входящий в сеть Apartlux;
- получение информации о конкретной гостинице;
- получение, создание, отзыв 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.

## **Требования к программной реализации**

- 1) Требуется использовать СОА (сервис-ориентированную архитектуру) для реализации системы.
- 2) Система состоит из микросервисов. Каждый микросервис отвечает за свою область логики работы приложения и должны быть запущены изолированно друг от друга.
- 3) При необходимости, каждый сервис имеет своё собственное хранилище, запросы между базами запрещены.
- 4) При разработке базы данных необходимо учитывать, что доступ к ней должен осуществляться по протоколу TCP.
- 5) Необходимо реализовать один web-интерфейс для фронтенда. Интерфейс должен быть доступен через тонкий клиент (браузер).
- 6) Для межсервисного взаимодействия использовать HTTP (придерживаться RESTful).
- 7) Выделить Gateway Service как единую точку входа и межсервисной коммуникации. В системе не должно осуществляться горизонтальных запросов.
- 8) При недоступности систем портала должна осуществляться деградация функционала или выдача пользователю сообщения об ошибке.
- 9) Необходимо предусмотреть авторизацию пользователей, как через интерфейс приложения, так и через популярные социальные сети.
- 10) Валидацию входных данных необходимо проводить и на стороне пользователя, и на стороне фронтенда. Микросервисы бекенда не должны валидировать входные данные, поскольку пользователь не может к ним обращаться напрямую, они должны получать уже отфильтрованные входные данные.
- 11) Для запросов, выполняющих обновление данных на нескольких узлах распределенной системы, в случае недоступности одной из систем, необходимо

димо выполнять полный откат транзакции.

- 12) Приложение должно поддерживать возможность горизонтального и вертикального масштабирования за счет увеличения количества функционирующих узлов и совершенствования технологий реализации компонентов и всей архитектуры системы.
- 13) Код хранить на Github, для сборки использовать Github Actions.
- 14) Gateway Service должен запускаться на порту 8080, остальные сервисы запускать на портах 8050, 8060, 8070.
- 15) Каждый сервис должен быть завернут в docker.

### **Функциональные требования к подсистемам**

Подсистемы: фронтенд, бекенд-координатор, бекенд регистрации и авторизации, бекенд бронирования, бекенд оплаты, бекенд лояльности.

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

- 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.

**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 гостиниц с возможными датами для бронирования и внесение изменений в перечень доступных дат (последнее только для администраторов);
- получение информации и обновление данных о зарегистрированном пользователе;
- оформление и отзыв созданного ранее бронирования;
- получение данных о бронированиях пользователя;
- получение статуса конкретного пользователя в программе лояльности и обновление её условий (последнее только для администраторов).

**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, так и через предлагаемые социальные сети.

Хранимая в базе данных сущность, ассоциированная с сервисом, детально представлена в таблице 4.

Таблица 4 – Состав сущностей

Сущность	Поля	Обязательность
Аккаунт	фамилия, не более 256 символов	да
	имя, не более 256 символов	да

*Продолжение на следующей странице*

<b>Сущность</b>	<b>Поля</b>	<b>Обязательность</b>
	<i>отчество</i> , не более 256 символов	нет
	<i>дата рождения</i>	да
	<i>логин</i> , является первичным ключом	да
	<i>захешированный пароль</i>	да
	<i>номер телефона</i>	да
	<i>электронная почта</i>	да

**Сервис бронирования** реализует следующие функции:

- получение и отправка данных в формате JSON по протоколу HTTP;
- получение таких данных, как:
  - список всех гостиниц, входящий в сеть Apartlux;
  - информация о конкретной гостинице по её идентификатору;
  - информация о свободных номерах по заданным реквизитам;
  - информация о конкретном номере по его идентификатору и идентификатору гостиницы, в которой он расположен;
  - все бронирования, зарегистрированные на конкретного клиента;
  - информация о конкретном бронировании по его идентификатору;
- вычисление стоимости бронирования за указанный период в выбранной гостинице за конкретный номер;
- создание, редактирование и отзыв бронирования.

Соответствующая база данных содержит три сущности, описание которых приведено в таблице 5.

Таблица 5 – Состав сущностей

<b>Сущность</b>	<b>Поля</b>	<b>Обязательность</b>
Гостиница	<i>идентификатор</i> , является первичным ключом	да

*Продолжение на следующей странице*

<b>Сущность</b>	<b>Поля</b>	<b>Обязательность</b>
	<i>название</i> не превышает 256 символов	да
	<i>страна</i> не превышает 80 символов	да
	<i>город</i> не превышает 80 символов	да
	<i>адрес</i> не превышает 256 символов	да
	<i>контактный телефон</i>	да
	<i>электронная почта</i>	да
	<i>описание</i> не превышает 2048 символов	нет
	<i>количество звёзд</i> , по умолчанию 0	нет
Номер	<i>идентификатор</i> , является первичным ключом	да
	<i>идентификатор гостиницы</i>	да
	<i>число мест</i>	да
	<i>этаж</i>	да
	<i>стоимость</i>	да
	<i>признак наличия двуспальной кровати</i>	да
Бронирование	<i>идентификатор</i> , является первичным ключом	да
	<i>логин клиента</i>	да
	<i>идентификатор платёжной операции</i>	да
	<i>идентификатор гостиницы</i>	да
	<i>статус</i> , APPROVED/UNAPPROVED, по умолчанию UNAPPROVED	да
	<i>дата въезда</i>	да
	<i>дата выезда</i>	да

**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.

Ассоциированная с этим сервисом база данных содержит сущность, детально представленная в таблице 6.

Таблица 6 – Состав сущностей

<b>Сущность</b>	<b>Поля</b>	<b>Обязательность</b>
Платёж	<i>идентификатор</i> , является первичным ключом	да
	<i>статус</i> , PAID/CANCELED	да
	<i>цена</i>	да
	<i>дата</i>	да

**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.

Соответствующая сущность базы данных имеет поля, представленные в таблице 7.

Таблица 7 – Состав сущностей

Сущность	Поля	Обязательность
Карта лояльности	<i>идентификатор</i> , является первичным ключом	да
	<i>логин клиента</i>	да
	<i>количество оформленных ранее заказов</i> , по умолчанию 0	да
	<i>статус</i> , BRONZE/SILVER/GOLD, по умолчанию BRONZE	да
	<i>скидка</i> , по умолчанию 10	да

### Требования к составу и параметрам технических средств

Все серверные приложения должны потреблять суммарно не более 2 Гбайт оперативной памяти и работать на сервере с процессором Intel(R) Core(TM) i7-10510U CPU 1.80GHz.

### Требования к надёжности

Система должна работать в соответствии с данным техническим заданием без рестарта. Необходимо использовать «зеркалируемые серверы» для всех подсистем, которые будут держать нагрузку в случае сбоя до тех пор, пока основной сервер не восстановится.

### Требования к документации

Исполнитель должен подготовить и передать заказчику руководство:

- для Администратора Системы;
- для Пользователя Системы;

- для Клиента Системы;
- по развёртыванию Системы.

## **Концептуальный дизайн**

Концептуальный дизайн позволяет рассмотреть создаваемую систему с точки зрения пользователей. На рисунке 1.3 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 них, отобразив её в виде «чёрного ящика».

Для уточнения деталей по операции бронирования, отображённой на диаграмме верхнего уровня, используется дочерняя диаграмма, которая изображена на рисунке 1.4. Она определяет последовательность выполнения операций в системе при обработке запроса клиента.

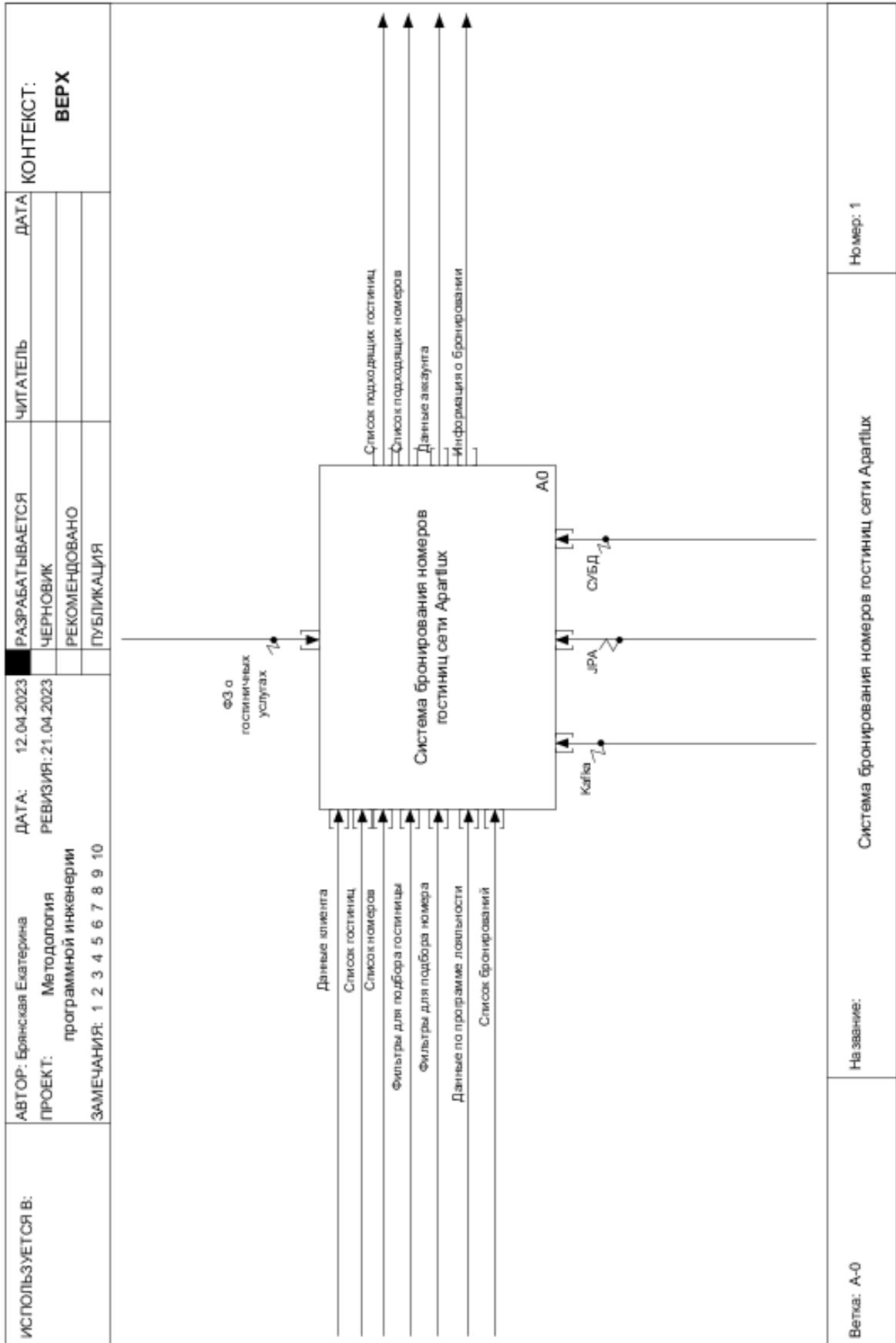


Рисунок 1.3 – Концептуальная модель системы в нотации IDEF0.

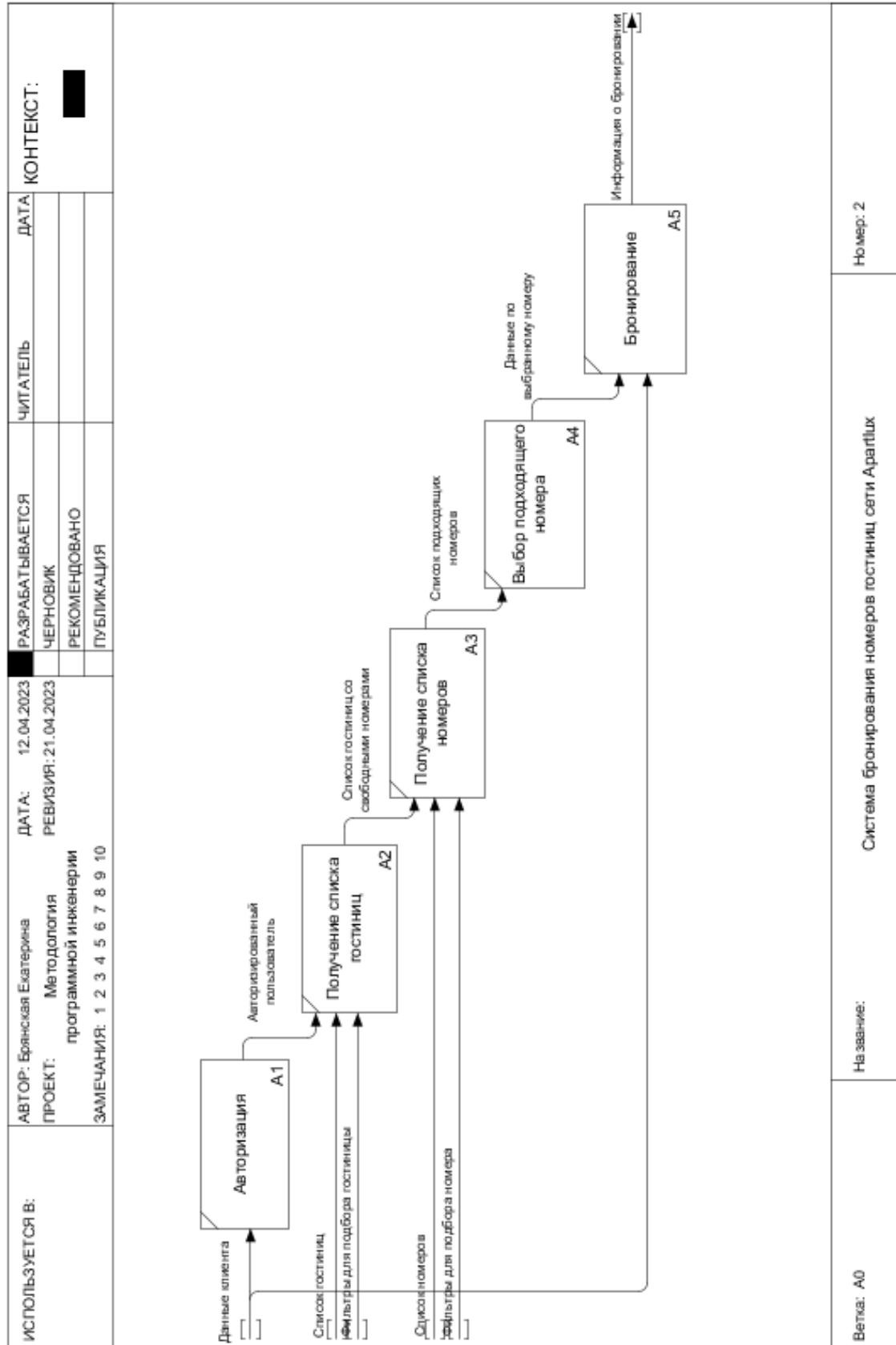


Рисунок 1.4 – Детализированная концептуальная модель системы в нотации IDEF0.

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

### **Бронирование номера**

- 1) Клиент нажимает кнопку «Бронирование».
- 2) Клиент перенаправляется на страницу, которая содержит список гостей.
- 3) Клиент нажимает на понравившуюся позицию и попадает на страницу доступных для бронирования номеров в выбранной гостинице с разными

реквизитами.

- 4) При необходимости выставляет необходимые параметры фильтров (например, адрес, планировка, диапазон цен и дат), нажимает кнопку «Применить». После этого список обновляется, сверху находятся предложения, наиболее соответствующие желанию клиента.
- 5) Клиент нажимает кнопку «Оформить бронирование» напротив нужного номера, на экране появляется всплывающее окно, дублирующее его атрибуты.
- 6) Клиент нажимает кнопку «Готово», выражая своё согласие на оформление бронирования, и перенаправляется на страницу, где вводит проверочный код из смс. В случае успешной операции придёт смс- и email-оповещения.
- 7) Если клиент не хочет оформлять бронь, он нажимает на кнопку выхода – крестик, всплывающее окно пропадает.

### **Диаграммы прецедентов**

В системе выделены три роли: Пользователь, Клиент, Администратор. На рисунках 1.5-1.7 представлены диаграммы прецедентов для каждой из ролей. В таблицах 8-9 описаны сценарии функционирования наиболее значимых прецедентов.

**Система бронирования номеров гостиниц Apartlux**

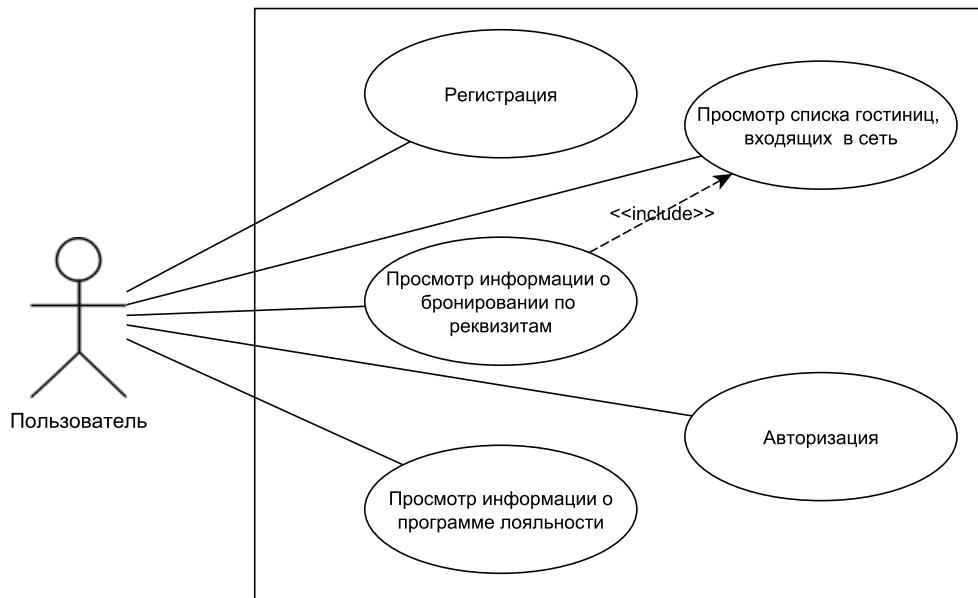


Рисунок 1.5 – Диаграмма прецедентов с точки зрения пользователя.

Таблица 8 – Спецификация сценария регистрации

<b>Нормальный ход сценария</b>	
<b>Действия пользователя</b>	<b>Отклик системы</b>
Регистрация	Система предоставляет пользователю форму для регистрации, в которой нужно заполнить ФИО, дату рождения, логин, пароль, номер телефона, электронную почту
Пользователь заполняет форму и даёт согласие на обработку данных	Данные пользователя регистрируются в системе
<b>Альтернативный ход сценария</b>	

*Продолжение на следующей странице*

<b>Действия пользователя</b>	<b>Отклик системы</b>
Регистрация	Система предоставляет пользователю форму для регистрации, в которой нужно заполнить ФИО, дату рождения, логин, пароль, номер телефона, электронную почту
Пользователь не заполняет форму или не даёт согласие на обработку данных	Пользователь не заносится в клиентскую базу данных

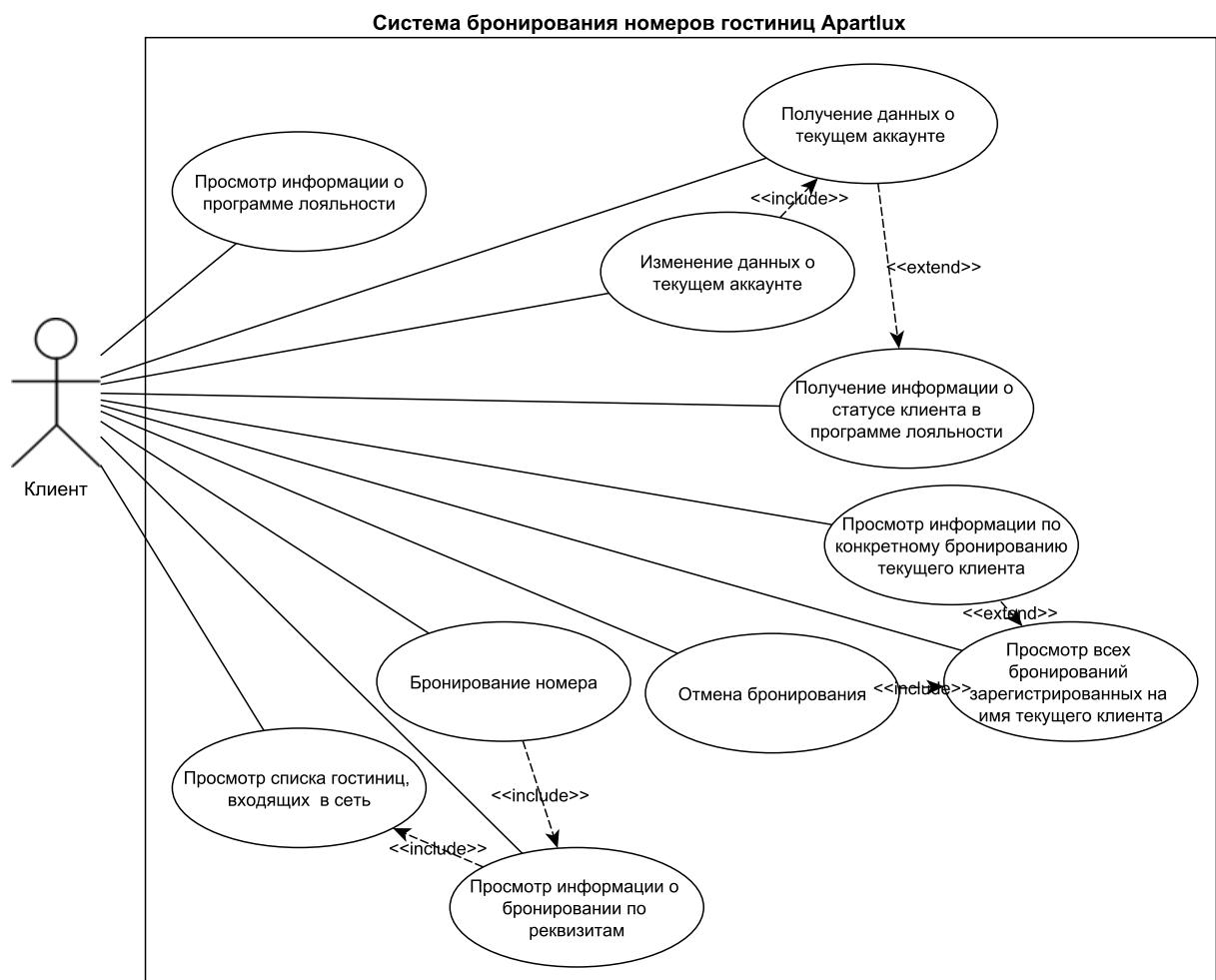


Рисунок 1.6 – Диаграмма прецедентов с точки зрения клиента.

Таблица 9 – Спецификация сценария бронирования

<b>Нормальный ход сценария</b>	
<b>Действия клиента</b>	<b>Отклик системы</b>
Просмотр списка гостиниц, предварительно отфильтрованных по выставленным пользователем параметрам	Система предоставляет клиенту список гостиниц, которые входят в состав сети Apartlux и предоставляющих свободные для бронирования номера
Просмотр информации о возможном бронировании по заданным реквизитам (дата, цена, количество мест и т.д.)	Система предоставляет клиенту ранжированный список номеров, наиболее подходящих под параметры фильтрации, указанные клиентом
Бронирование	Система перенаправляет клиенту на страницу подтверждения по коду из смс, после успешной операции фиксирует выбранный номер за клиентом
<b>Альтернативный ход сценария</b>	
Просмотр списка гостиниц, предварительно отфильтрованных по выставленным пользователем параметрам	Система предоставляет клиенту список гостиниц, которые входят в состав сети Apartlux и предоставляющих свободные для бронирования номера
Клиент не выбирает ни одну гостиницу из предоставленных	Система отправляет письмо-напоминание на почту клиента, содержащее список предлагаемых гостиниц
<b>Альтернативный ход сценария</b>	

*Продолжение на следующей странице*

<b>Действия клиента</b>	<b>Отклик системы</b>
Система не смогла подобрать ни одной гостиницы, которая бы подходила под желания пользователя	Система выводит сообщение об отсутствии позиций, точно подходящих под описание, а также предоставляет список гостиниц, которые частично подходят под реквизиты
<b>Альтернативный ход сценария</b>	
Просмотр списка гостиниц, предварительно отфильтрованных по выставленным пользователем параметрам	Система предоставляет клиенту список гостиниц, которые входят в состав сети Apartlux и предоставляющих свободные для бронирования номера
Просмотр информации о возможном бронировании по заданным реквизитам (дата, цена, количество мест и т.д.)	Система предоставляет клиенту ранжированный список номеров, наиболее подходящих под параметры фильтрации, указанные клиентом
Клиент не выбирает ни одного номера и завершает работу с системой	Система отправляет письмо-напоминание на почту клиента, содержащее список предлагаемых номеров
<b>Альтернативный ход сценария</b>	
Просмотр списка гостиниц, предварительно отфильтрованных по выставленным пользователем параметрам	Система предоставляет клиенту список гостиниц, которые входят в состав сети Apartlux и предоставляющих свободные для бронирования номера

*Продолжение на следующей странице*

<b>Действия клиента</b>	<b>Отклик системы</b>
Система не смогла подобрать ни одного номера, который бы подходил под желания клиента (дата, цена, количество мест и т.д.)	Система предоставляет клиенту ранжированный список номеров, наиболее подходящих под параметры фильтрации, указанные клиентом
<b>Альтернативный ход сценария</b>	
Просмотр списка гостиниц, входящих в сеть	Система предоставляет клиенту список гостиниц, которые входят в состав сети Apartlux и предоставляющих свободные для бронирования номера
Просмотр информации о возможном бронировании по заданным реквизитам (дата, цена, количество мест и т.д.)	Система предоставляет клиенту ранжированный список номеров, наиболее подходящих под параметры фильтрации, указанные клиентом
Бронирование (операции подтверждения по смс завершилась с ошибкой)	Система выводит сообщение об ошибке на экран, из-за проблем с операцией клиенту предлагается повторить попытку, номер на клиента не регистрируется

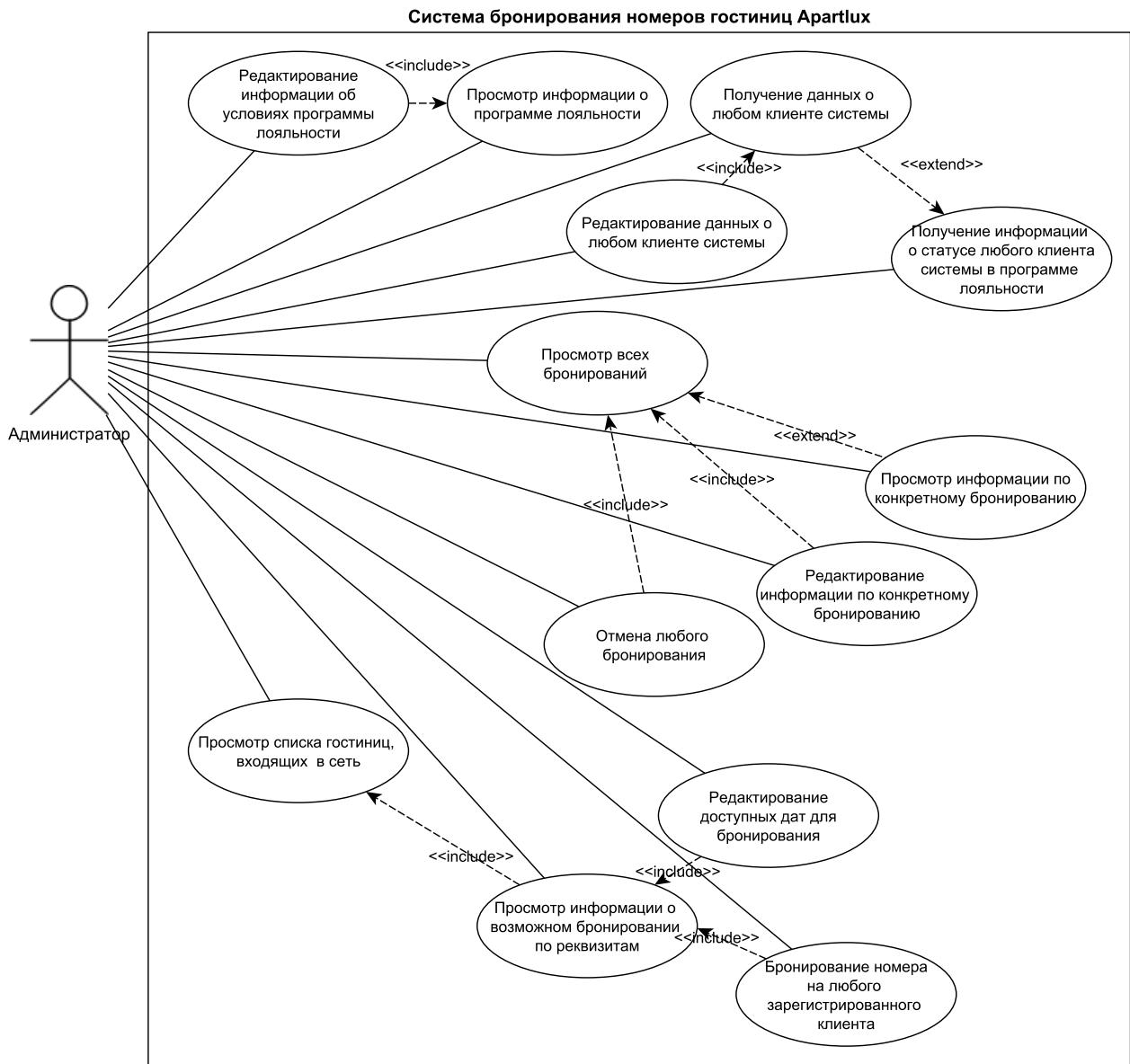


Рисунок 1.7 – Диаграмма прецедентов с точки зрения администратора.

## Спецификация классов

На рисунке 1.8 представлена диаграмма классов для разработки микросервиса бронирования.

Классы `ReservationEntity`, `HotelEntity`, `RoomEntity` представляют легковесные объекты бизнес-логики, ассоциированные с соответствующими сущностями базы данных. Атрибуты указанных классов представлены в таблицах 10-12.

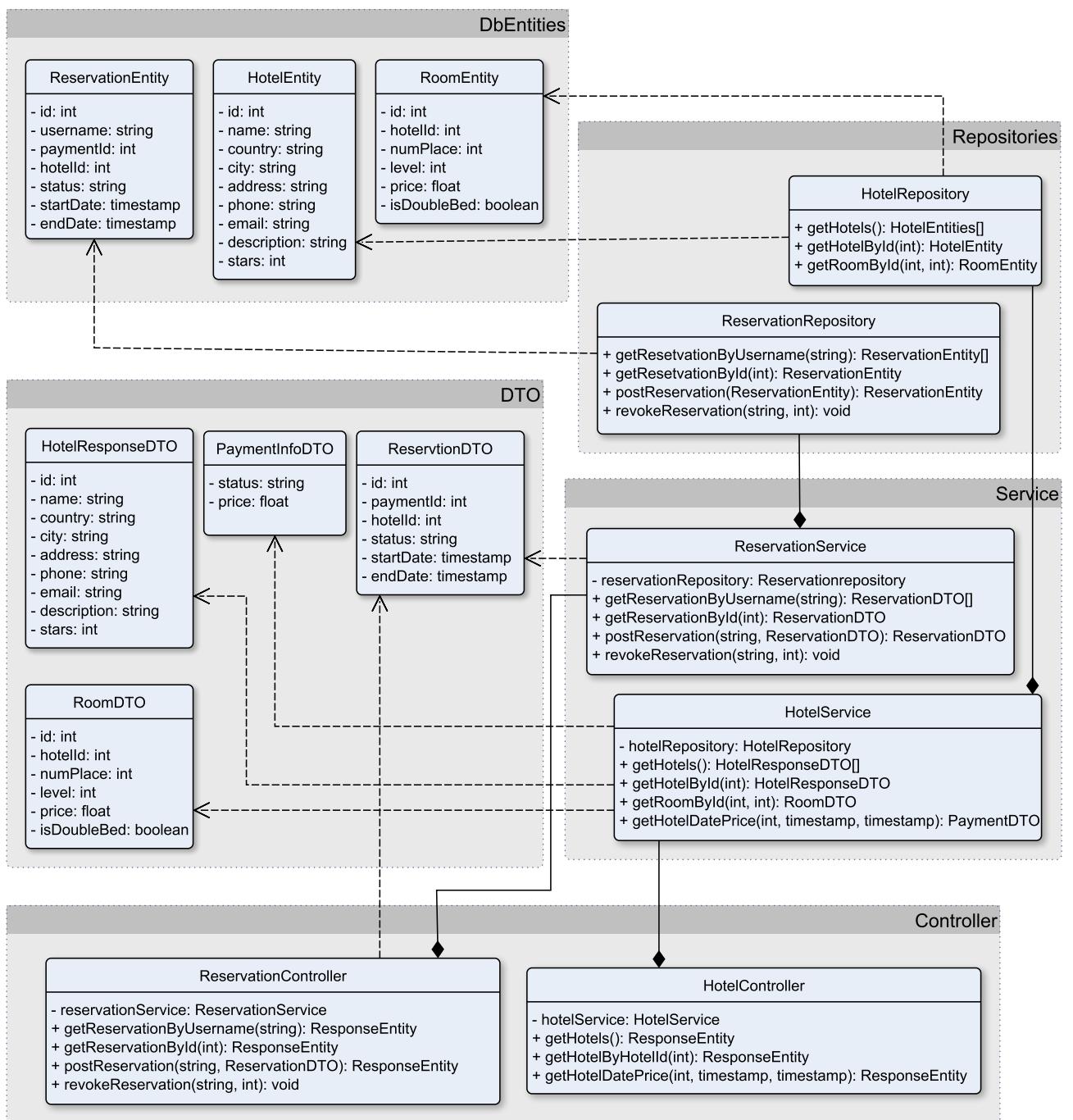


Рисунок 1.8 – Диаграмма классов.

Таблица 10 – Атрибуты класса HotelEntity

Атрибуты		
Имя	Тип	Описание
id	private: int	идентификатор

*Продолжение на следующей странице*

<b>Имя</b>	<b>Тип</b>	<b>Описание</b>
name	private: string	название
country	private: string	страна
city	private: string	город
address	private: string	адрес
phone	private: string	контактный телефон
email	private: string	электронная почта
description	private: string	описание
stars	private: int	количество звёзд

Таблица 11 – Атрибуты класса RoomEntity

<b>Атрибуты</b>		
<b>Имя</b>	<b>Тип</b>	<b>Описание</b>
id	private: int	идентификатор
hotelId	private: int	идентификатор гостиницы
numPlace	private: int	число мест
level	private: int	этаж
price	private: float	стоимость
isDoubleBed	private: boolean	признак наличия двуспальной кровати

Таблица 12 – Атрибуты класса ReservationEntity

<b>Атрибуты</b>		
<b>Имя</b>	<b>Тип</b>	<b>Описание</b>
id	private: int	идентификатор
username	private: string	логин клиента

*Продолжение на следующей странице*

<b>Имя</b>	<b>Тип</b>	<b>Описание</b>
paymentId	private: int	идентификатор платёжной операции
hotelId	private: int	идентификатор гостиницы
status	private: string	статус
startDate	private: timestamp	дата въезда
endDate	private: timestamp	дата выезда

Классы ReservationDTO, HotelResponseDTO, RoomDTO, PaymentInfoDTO представляют объекты для передачи данных между классами бизнес-логики. Атрибутивный состав ReservationDTO, HotelResponseDTO, RoomDTO аналогичен составу ассоциированных с ними сущностей базы данных. Атрибуты вспомогательного класса PaymentInfoDTO представлены в таблице 13.

Таблица 13 – Атрибуты класса PaymentInfoDTO

<b>Атрибуты</b>		
<b>Имя</b>	<b>Тип</b>	<b>Описание</b>
status	private: string	статус операции
price	private: float	стоимость

Классы HotelRepository и ReservationRepository отвечают за взаимодействие микросервиса с базой данных. Методы каждого приведены в таблицах 14-15.

Таблица 14 – Атрибуты класса HotelRepository

Методы	
Название	Описание
getHotelById(int): HotelEntity	param: id [int - in] - идентификатор <i>получение информации о гостинице по её идентификатору</i>
getRoomById(int, int): RoomEntity	param: idHotel [int - in] - идентификатор гостиницы param: idRoom [int - in] - идентификатор номера <i>получение информации о номере по идентификаторам гостиницы и номера</i>

Таблица 15 – Атрибуты класса ReservationRepository

Методы	
Название	Описание
getReservationByUsername(string): ReservationEntity[]	param: [string - in] - логин клиента <i>получение информации о бронированиях по логину клиента</i>
getReservationById(int): ReservationEntity	param: [int - in] - идентификатор бронирования <i>получение информации о бронировании по его идентификатору</i>
postReservation(ReservationEntity): ReservationEntity	param: [ReservationEntity - in] - бронирование <i>создание записи о бронировании</i>

*Продолжение на следующей странице*

<b>Название</b>	<b>Описание</b>
revokeReservation(string, int): void	param: [string - in] - логин клиента param: [int - in] - идентификатор бронирования <i>отмена бронирования</i>

Классы ReservationService и HotelService реализуют основную бизнес-логику микросервиса, преобразование данных и передачу их на последующий слой, непосредственно связанный с базой данных, поэтому предоставляемые ими методы схожи с методами классов ReservationRepository и HotelRepository.

На рисунке 1.9 представлена диаграмма деятельности в режиме «Бронирование» для клиента.

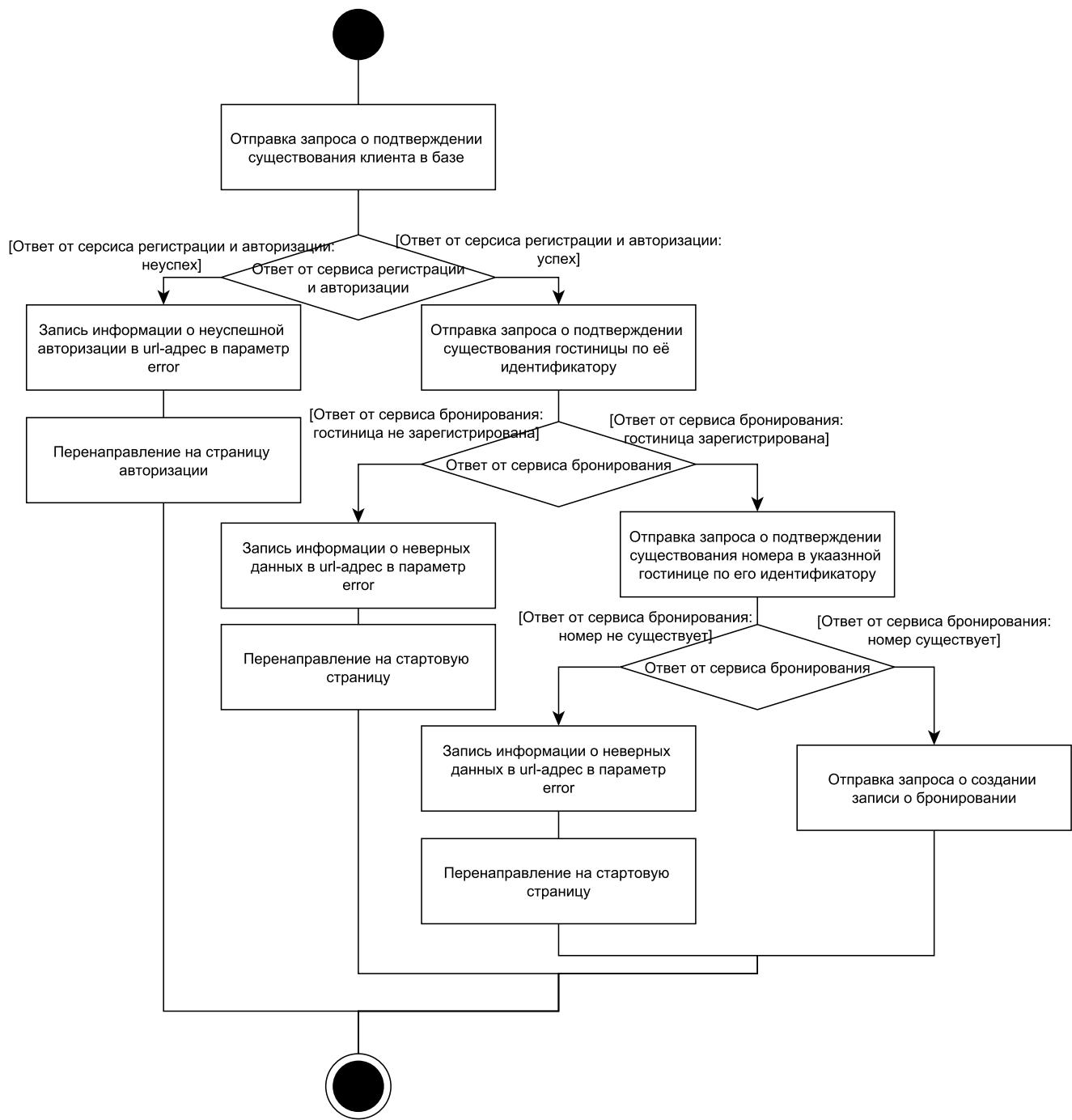


Рисунок 1.9 – Диаграмма деятельности в режиме «Бронирование» для клиента.

На рисунке 1.10 представлена наиболее важная для системы динамическая модель, представленная в виде диаграммы деятельности.

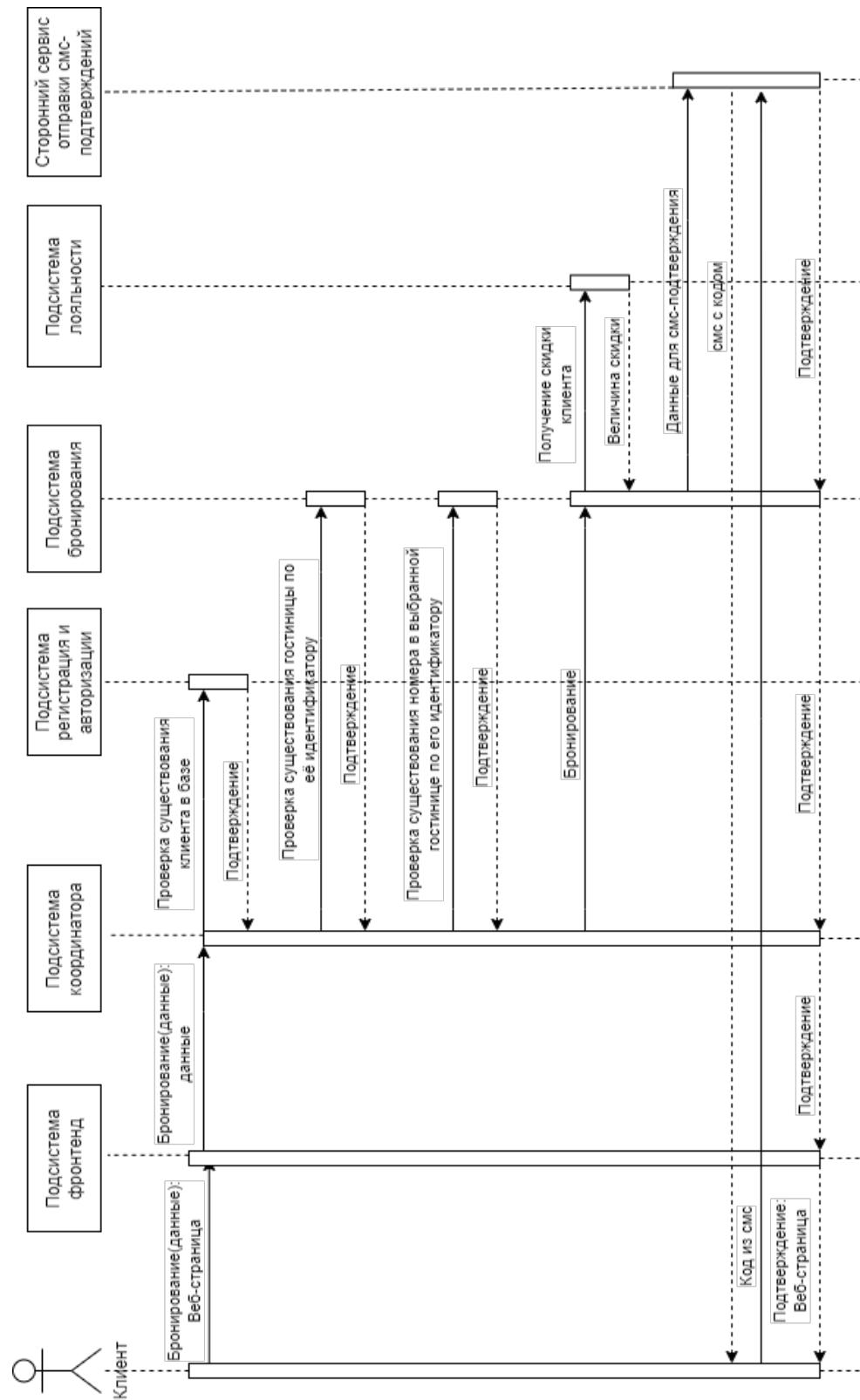


Рисунок 1.10 – Диаграмма последовательности действий при бронировании номера клиентом: концептуальный уровень.

В системе предполагается распределенное хранение данных. Диаграмма потоков данных, представленная на рисунке 1.11, позволяет описать распределение сохраняемой информации в хранилищах данных. Каждое хранилище данных будет реализовано в виде базы данных.

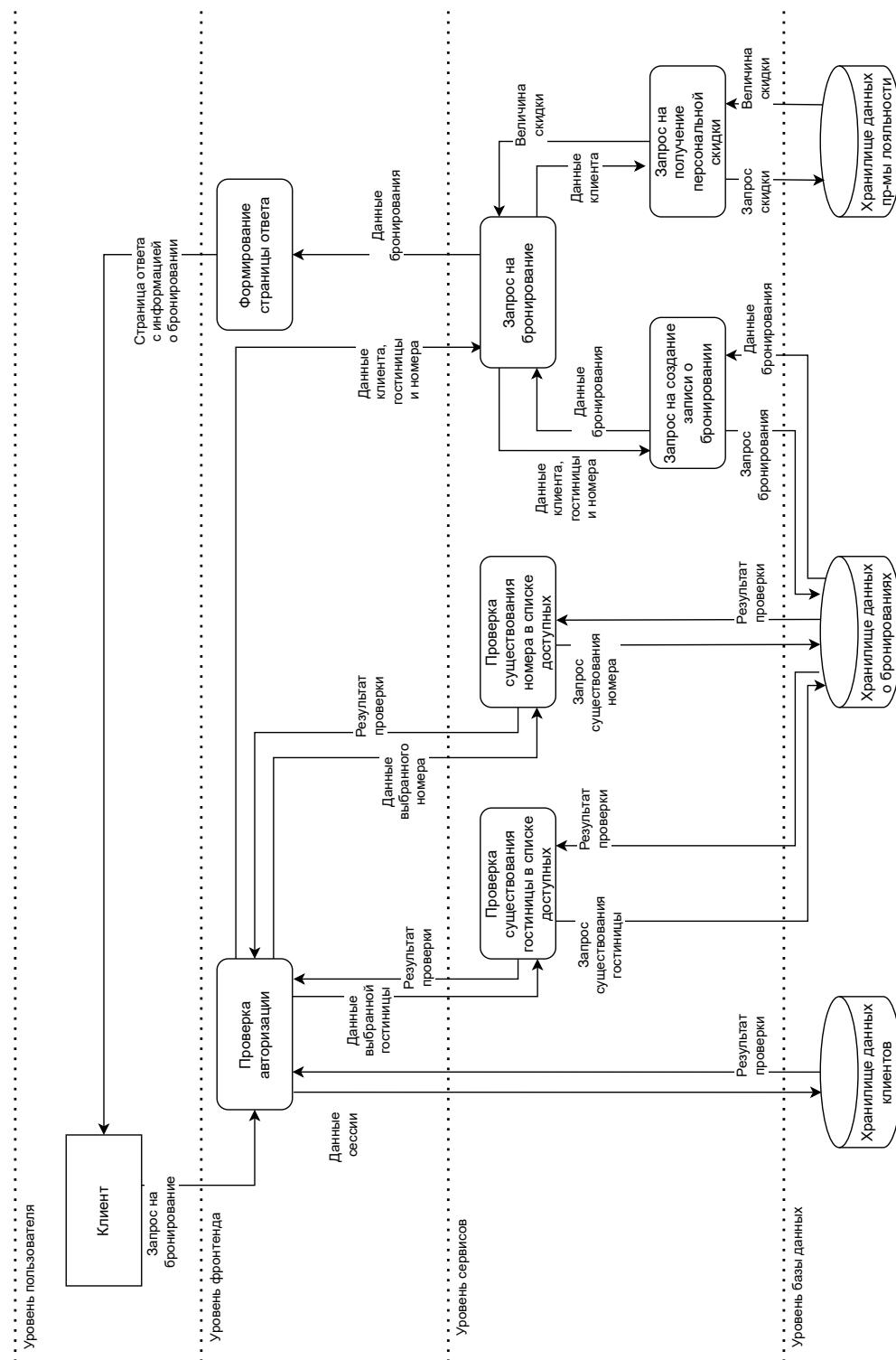


Рисунок 1.11 – Диаграмма потоков данных при бронировании.

## **Высокоуровневый дизайн пользовательского интерфейса**

Пользовательский интерфейс в разрабатываемой системе представляет собой web-интерфейс, доступ к которому осуществляется через браузер (тонкий клиент).

Страница системы состоит из «шапки», основной части и футера. «Шапка» — это верхняя часть страницы, в которой находятся логотип и верхнее меню со ссылками на основные разделы портала. Футер — это нижняя часть страницы, в которой обычно размещают ссылки на редко посещаемые, но необходимые страницы, например, страницы с пользовательским соглашением.

Обобщенно структуру страниц системы можно представить следующим образом:

- главная страница с перечнем наиболее популярных гостиниц сети;
- страница авторизации;
- форма регистрации;
- страница с информацией о гостинице и списком доступных для бронирования номеров;
- страница с детальной информацией по конкретному номеру;
- страница личного кабинета клиента системы;
- о нас
  - контакты;
  - новости;
  - нормативные документы;
  - правила организации.

Ниже приведены такие формы системы, как страницы регистрации нового клиента и главная страница со списком гостиниц, в который есть свободные для бронирования номера. На рисунке 1.12 представлена форма регистрации, на которой пользователю нужно ввести личные данные: ФИО, дата рождения, телефон, электронная почта, также придумать логин и пароль.

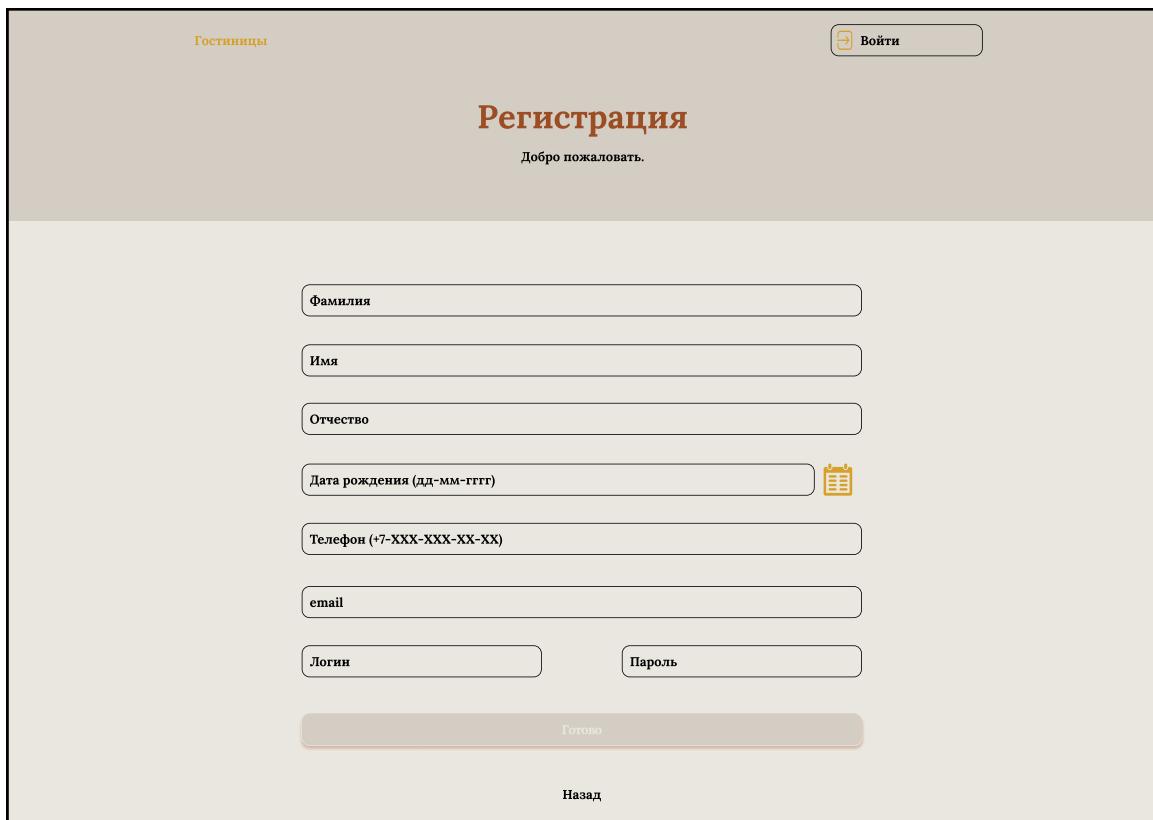


Рисунок 1.12 – Страница регистрации.

На рисунке 1.13 изображена главная страница со списком гостиниц входящих в сеть.

На сайте есть кнопка для входа в систему, поисковая строка, где пользователь может указать свои предпочтения, система будет производить поиск по ключевым словам из этого запроса и формировать подборку наиболее подходящих под описание гостиниц и номеров. Также есть уже готовые фильтры, достаточно нажать на интересующую характеристику.

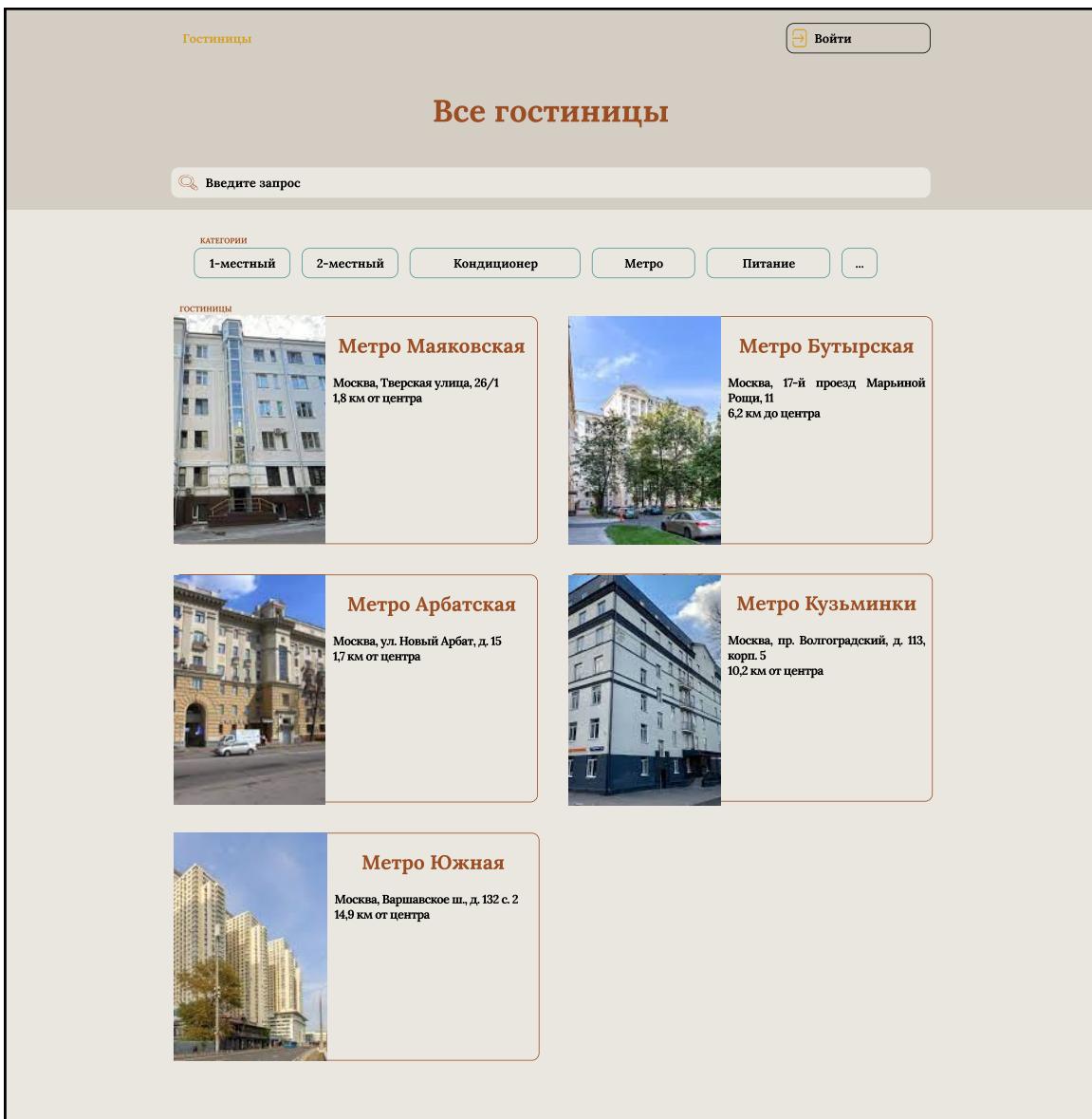


Рисунок 1.13 – Диаграмма потоков данных при бронировании.

## Физический дизайн

### Выбор системы развертывания компонентов распределенной системы

Требования для системы развертывания компонентов.

- **Изоляция.** Компоненты не должны иметь доступа друг к другу, за исключением доступа, предусмотренного протоколом взаимодействия
- **Управление ресурсами.** Компонент должен потреблять ограниченное число ресурсов (процессорного времени, оперативной памяти, места на жест-

ком диске).

- **Отслеживание зависимостей.** Компонент должен разворачиваться монолитно со всеми своими библиотеками. Это требование гарантирует, что компонент будет использоваться с теми же библиотеками, с которыми был протестирован.
- **Низкие накладные расходы.** Использование системы развертывания не должно нести высокие накладные расходы для распределенной системы. Полностью удовлетворяет представленным требованиям такой инструмент визуализации, как программа *Docker* [1]. Согласно определению, Docker — программное обеспечение для автоматизации развертывания и управления приложениями в среде виртуализации на уровне операционной системы Linux. Позволяет «упаковать» приложение со всем его окружением в контейнер, который может быть перенесен на любую Linux-систему, а также предоставляет среду по управлению контейнерами. Разрабатывается и поддерживается одноименной компанией, распространяется как свободное программное обеспечение под лицензией Apache 2.0.

*Контейнером* называется образ изолируемой части Системы, содержащий приложение со всеми его зависимостями. В контейнере могут содержаться приложения и их среда выполнения: нужные библиотеки, конфигурационные файлы. Для облегчения работы используют единый репозиторий (хранилище) образов. Это позволяет применять уже готовые образы, которые необходимо только доработать для нужной задачи. Docker эффективно используют для следующих задач:

- тестирование распределённой системы;
- отслеживание зависимостей между компонентами разрабатываемой системы.

Преимуществами Docker по сравнению с виртуальными машинами (Oracle VM VirtualBox, Microsoft Hyper-V) являются:

- низкие накладные расходы;

- наличие репозитория.

Использование программного обеспечения Docker позволит обеспечить высокую производительность при создании виртуальных вычислительных узлов и уменьшить потребление процессорных ресурсов. Недостатком Docker является то, что он работает только под ОС Linux.

## **Выбор операционной системы**

Согласно требованиям технического задания, разрабатываемый портал должен обладать высокой доступностью, работать на типичных архитектурах ЭВМ (Intel x86, Intel x64), а так же 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, таких как Mail.Ru, Vk.com, Google.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 [2]. *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 ста-

бильна в работе.

## Выбор СУБД

В соответствии с техническим заданием разработки бекенда 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 потеряны.
- **Масштабируемость.** Выбранная СУБД должна поддерживать репликацию, шардирование.

PostgreSQL [3] – 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.

## **Выбор языка разработки компонент портала**

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.

Под данные требования хорошо подходит язык Java [4]. Это универсальный объектно-ориентированный язык со строгой типизацией. В нём реализован принцип WORA (от английского: write once, run anywhere). Это позволяет запускать приложения везде, где есть среда исполнения JRE (от английского: Java Runtime Environment). При этом не имеет значения, какая операционная система установлена на устройстве.

У Java есть ряд преимуществ.

- **Простота** – чёткие синтаксические правила и понятная семантика.
- **Объектно-ориентированный подход.**
- **Безопасность.** Обойти или взломать механизмы защиты крайне сложно.
- **Производительность.** Новые версии динамических компиляторов Java не уступают традиционным из других платформ. При необходимости те или иные приёмы оптимизации включаются или отменяются JIT-компилятором.
- **Надёжность.** Компилятор способен выявить ошибки ещё до выполнения

кода, то есть на ранних стадиях.

- **Независимость от аппаратной части и ОС.** Важно лишь наличие исполняющей среды и JVM. Байт-код легко интерпретируется на любой машине. Кроссплатформенностью отличается также интерфейс, реализованный в системных библиотеках.
- **Динамичность и адаптируемость.** При необходимости можно добавить в библиотеки новые объекты, методы.
- **Удобные и эффективные сетевые возможности.** Приложения умеют находить объекты в сети и открывать к ним доступ. Предоставляется обширная программная библиотека для передачи данных по самым распространённым протоколам: FTP, HTTP, TCP/IP. Работает механизм вызова удалённых методов.

## Выбор фреймворка для разработки портала

К выбору фреймворка предъявляют те же требования, что и к выбору языка программирования:

- большое число стандартных возможностей;
- совместимость с выбранными ранее технологиями;
- высокая скорость разрабатываемого портала.

Для разработки «Распределённой системы бронирования номеров гостей сети Apartlux» должны быть использованы фреймворки, основанные на парадигме MVC:

- модель представляет объекты, хранимые в системе: информацию об аккаунтах, гостиницах, номерах, бронированиях;
- вид отвечает за визуализацию моделей;
- элемент управления отвечает за взаимодействие пользователя и программного обеспечения.

*Spring Boot [5]* — это фреймворк на основе Java с открытым исходным кодом. Благодаря быстродействию и простоте работы он стал популярным ре-

шением для создания развертываний в виде архива веб-приложений (WAR) и автономных Java-приложений.

Он позволяет избавиться от трудоемкой первоначальной установки и настройки среды развертывания. Основные преимущества:

- быстрая и легкая разработка приложений;
- автоконфигурация всех компонентов;
- готовые встроенные серверы, обеспечивающие ускоренное и более продуктивное развертывание приложений;
- отсутствие конфигурации XML;
- большой выбор плагинов, облегчающих работу со встроенными базами данных, легкий доступ к ним и службам очередей.
- подробная документация.

К недостаткам этого фреймворка относятся:

- создает множество неиспользуемых зависимостей, что приводит к большому размеру файла развертывания;
- не подходит для создания монолитных приложений.

Таким образом, в результате проведённого анализа в качестве системы развёртывания компонентов был выбран Docker, ОС – Ubuntu, СУБД – PostgreSQL, язык программирования – Java, фреймворк – Spring Boot.

## **Обеспечение надежности портала**

В рассматриваемой «Распределённой системе бронирования номеров гостиниц сети Apartlux» для обеспечения надежности функционирования СУБД будет применяться репликация типа «ведущий – ведомый» и шардинг. Для обеспечения надежности данных СУБД необходимо разработать скрипт для автоматического создания резервной копии базы данных по расписанию.

Для фронтенда и бекендов целесообразно применить зеркалирование. Это обеспечит отказоустойчивость системы: в случае сбоя любого из ее узлов запросы на чтение данных будут выполняться.

Отказоустойчивость фронтенда возможно за счет его зеркалирования на несколько серверов. Выбор одного из зеркалируемых серверов осуществляется с помощью прописывания в DNS (система доменных имен) записях адресов всех серверов, на которых располагаются фронтенды.

Также должна быть настроена система мониторинга на предмет возникновения ошибок в системе, в случае их появления оповещение должно прийти в Telegram-канал для наиболее оперативного реагирования.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторных работ по курсу «Методология программной инженерии» была разработана распределённая система бронирования номеров гостиниц сети Apartlux.

Перед началом разработки сформулировано техническое задание с общими и функциональными требованиями, а также требованиями к надежности и документации. Помимо требований к системе в техническом задании приведена топология системы, на основе которой разработаны требования к подсистемам.

В конструкторском разделе описано проектирование сервисов по отдельности и системы в целом, проработаны сценарии функционирования и представлена архитектура сервисов.

В технологическом разделе описаны основные принципы реализации ключевого функционала сервисов. Данная работа также включает в себя реализацию описанного проекта согласно разработанному техническому заданию.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Docker Documentation [Электронный ресурс]. – Режим доступа:  
<https://docs.docker.com/> (Дата обращения: 20.04.2023).
2. Ubuntu Documentation [Электронный ресурс]. – Режим доступа:  
<https://ubuntu.com/> (Дата обращения: 20.04.2023).
3. PostgreSQL Documentation [Электронный ресурс]. – Режим доступа:  
<https://www.postgresql.org/docs/> (Дата обращения: 20.04.2023).
4. Java Documentation [Электронный ресурс]. – Режим доступа:  
<https://docs.oracle.com/en/java/> (Дата обращения: 20.04.2023).
5. Spring Boot Documentation [Электронный ресурс]. – Режим доступа:  
<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/> (Дата обращения: 20.04.2023).