

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Разработка приложения для собаководов

Курсовой проект

по дисциплине

Технологии программирования

09.03.02 Информационные системы и технологии

Программная инженерия в информационных системах

6 семестр 2022/2023 учебного года

Зав. кафедрой _____ д. ф.–м. н., доцент С.Д. Махортов

Обучающийся _____ ст. 3 курса оч. отд. А.А. Полев

Обучающийся _____ ст. 3 курса оч. отд. П.О. Федосова

Обучающийся _____ ст. 3 курса оч. отд. К.В. Брюхов

Руководитель _____ В.С. Тарасов, ст. преподаватель

Руководитель _____ И.В. Клейменов, ассистент _____.20__

Воронеж 2023

Содержание

Содержание	2
Введение.....	3
1 Постановка задачи.....	4
1.1 Цель проекта.....	4
1.2 Задачи проекта.....	4
1.3 Требования к разрабатываемой системе	4
Требования к структуре.....	4
Функциональные требования	4
Требования к защите информации.....	5
2 Анализ предметной области	6
2.1 Терминология (гlossарий) предметной области	6
2.2 Обзор аналогов	8
«11pets: Уход за питомцем».....	8
«Dog Health»	9
«Дневник по уходу за домашними».....	11
2.3 Графическое описание работы системы.....	12
Диаграмма IDEF0.....	12
Диаграмма прецедентов	12
Диаграмма последовательности	13
Диаграмма активности	17
Диаграмма состояний	17
Диаграмма классов	19
Диаграмма объектов	20
Диаграмма развертывания	20
Диаграммы сотрудничества.....	20
3 Реализация.....	23
3.1 Средства реализации.....	23
3.2 Реализация Backend	24
База данных	24
Архитектура серверной части приложения	28
3.3 Реализация Frontend.....	31
4 Тестирование	31

Введение

С давних времен собаки считаются верными помощниками и друзьями человека. Наши четвероногие братья меньшие отличаются умом, добротой, смелостью и верностью. Именно поэтому до сих пор собаки занимают особое место в жизни многих людей. Она становится для человека настоящим членом семьи, о котором он с радостью заботится. И основа этой заботы – внимательное отношение к здоровью и питанию питомца.

Актуальность разрабатываемого приложения «Лапки» в том, что оно позволяет следить за многими факторами для поддержания хорошего состояния собаки, например:

- ветеринарными клиниками, предоставляемыми им услугами и их стоимостью;
- оптимальным количеством еды, чтобы питомец не страдал от переедания или, наоборот, голода;
- событиями, такими как поход к ветеринару или грумеру.

В данной курсовой работе были поставлены следующие задачи: реализовать возможности сравнивать цены на услуги в ветеринарных клиниках, вычислять оптимальное количество пищи для питомца и записывать события, связанные с уходом за собакой.

В работе будет более подробно рассмотрен процесс проектирования и разработки, начиная от обзора аналогов и заканчивая созданием базы данных, разработкой интерфейса и реализацией основных функций системы.

1 Постановка задачи

1.1 Цель проекта

Разработка мобильного android – приложения для помощи человеку в уходе за собакой.

1.2 Задачи проекта

Создание приложения для собаководов, обладающего возможностью сравнивать цены на услуги в ветеринарных клиниках, вычислять оптимальное количество пищи для питомца.

1.3 Требования к разрабатываемой системе

1.3.1 Требования к структуре

Приложение должно быть построено на трехуровневой архитектуре: клиент (мобильное приложение) – сервер – база данных.


1.3.2 Функциональные требования

К разрабатываемому приложению выдвинуты следующие функциональные требования:

- разделение пользователей на: неавторизованных, авторизованных (владельцев собак) и администраторов;
- просмотр списка ветеринарных клиник, а также его сортировка по цене на услугу и фильтрация по городу;
- калькулятор для вычисления оптимального количества пищи питомца по его индивидуальным характеристикам;
- добавление информации о своих питомцах и о событиях, связанных с уходом за собакой, а также возможность ее редактирования и удаления;
- изменение личных данных профиля;
- добавление новых ветеринарных клиник от имени администратора;
- редактирование информации об уже существующих ветеринарных клиниках или ее удаление из учетной записи администратора.

1.3.3 Требования к защите информации

Приложение должно обеспечить:

- авторизацию и аутентификацию пользователей;
- защиту от SQL–инъекций;
- шифрование пароля при  записи в БД.

2 Анализ предметной области

2.1 Терминология (гlossарий) предметной области

- Мобильное приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для конкретной платформы (iOS, Android и т. д.);
- Android — это операционная система с открытым исходным кодом, созданная для мобильных устройств на основе модифицированного ядра Linux;
- Android — приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для платформы Android;
- Клиент — это пользовательское устройство или программное обеспечение, которое отправляет запросы к серверу и получает ответы от него. Клиент может быть представлен в различных формах, таких как веб – браузер, мобильное приложение и других. Он предоставляет интерфейс для пользователя, позволяющий взаимодействовать с приложением и отправлять запросы на сервер для выполнения определенных задач;
- Сервер — это центральный компонент, который обрабатывает запросы от клиента и предоставляет соответствующие ответы. Сервер обычно является вычислительным устройством, которое хранит данные и выполняет бизнес–логику приложения. Он прослушивает запросы от клиента, обрабатывает их, взаимодействует с базой данных или другими внешними системами, если это необходимо, и отправляет клиенту результаты выполнения запроса;
- База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном

- виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД);
- SQL – запросы — это наборы команд для работы с реляционными базами данных;
 - Аутентификация — процедура проверки подлинности, например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных;
 - Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий;
 - Фреймворк — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта;
 - SQL – инъекция — внедрении в запрос произвольного SQL-кода, который может повредить данные, хранящиеся в базе данных или предоставить доступ к ним;
 - Пользователь – человек, который использует приложение;
 - Аккаунт или учетная запись — это персональная страница пользователя или личный кабинет, который создается после регистрации;
 - Frontend — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса;
 - Backend — программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней части;
 - HTTP (Hypertext Transfer Protocol) — протокол передачи данных в сети, широко используемым для обмена информацией между клиентом и сервером;
 - REST (Representational State Transfer) — архитектурный стиль взаимодействия компонентов распределённого приложения в сети;

— API (Application Programming Interface) — описание взаимодействия одной компьютерной программы с другой.

2.2 Обзор аналогов

2.2.1 «11pets: Уход за питомцем»

«11pets: Уход за питомцем» – приложение компании «11 PETS». Приложение следит за графиками вакцинаций, дегельминтаций, купания и других процедур. Программа вовремя просигнализирует хозяину, когда животному потребуется ввести препарат или отвезти на плановый осмотр к врачу. Приложение способно одновременно хранить данные сразу о нескольких питомцах, вне зависимости от вида: будь то собака, кошка, кролик или другое животное.

Недостатки:

- Проблемы при входе в учетную запись;
- Ошибки сохранения информации в базу данных.

Интерфейс приложения представлен ниже (Рисунок 1 и Рисунок 2).

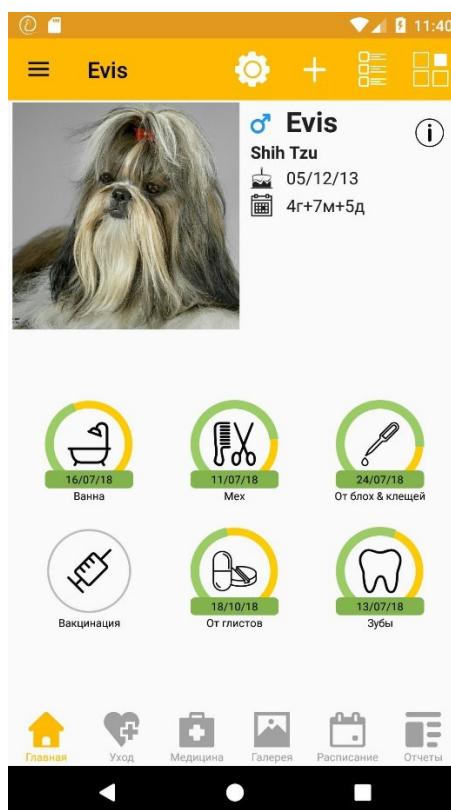


Рисунок 1 - Интерфейс приложения «11pets: Уход за питомцем»

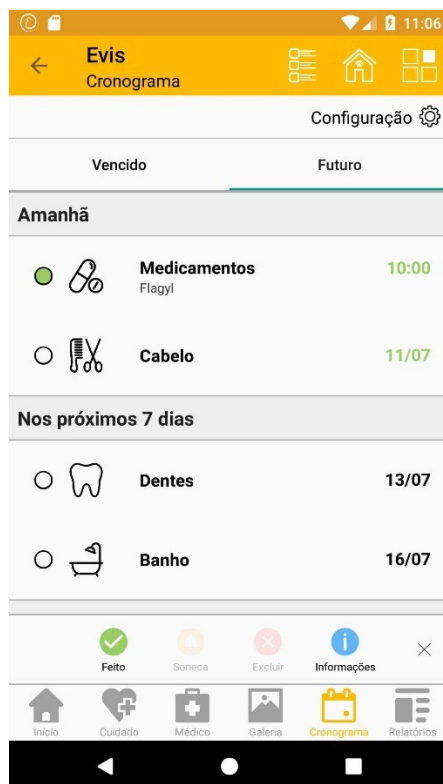


Рисунок 2 - Интерфейс приложения «1 lpets: Уход за питомцем»

2.2.2 «Dog Health»

«Dog Health» — это бесплатное приложение помогает следить за здоровьем домашнего любимца. Его использование не требует предварительной регистрации. Достаточно добавить питомца, указав его личные данные (вес, рост, кличку, дату рождения, номер чипа). Приложение напоминает о предстоящих прививках, противопаразитной обработке, посещениях ветеринара и применении медикаментов. Находит ветеринарные клиники и их контакты.

Недостатки:

- Нет поиска ветеринаров для России;
- Отсутствие Русской локализации.

Интерфейс приложения представлен ниже (Рисунок 3 и Рисунок 4).

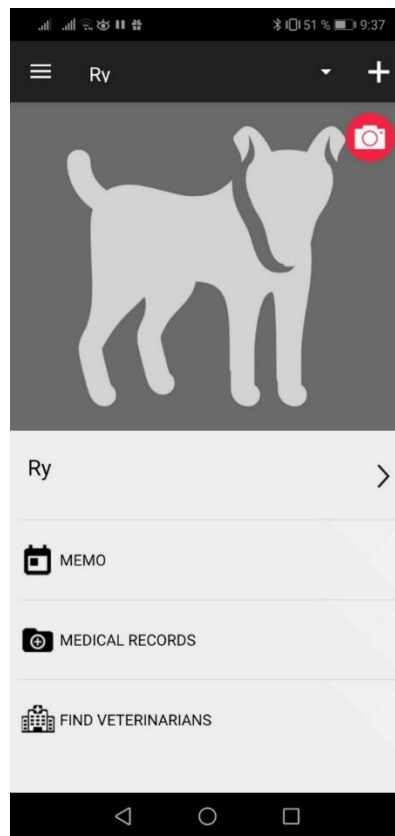


Рисунок 3 - Интерфейс приложения «Dog Health»

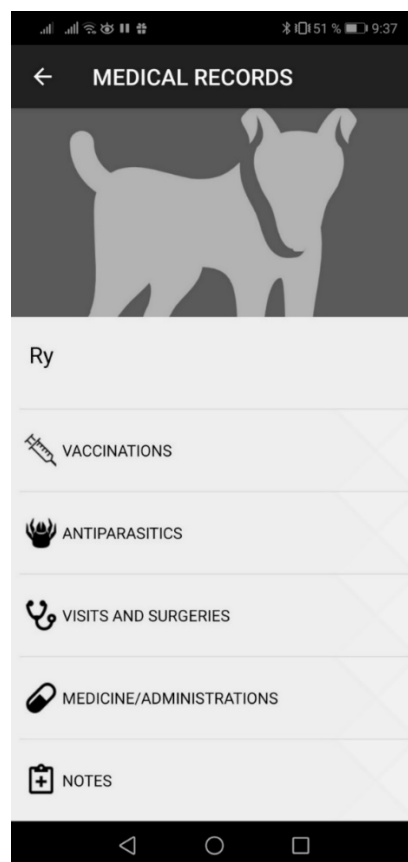


Рисунок 4 - Интерфейс приложения «Dog Health»

2.2.3 «Дневник по уходу за домашними»

«Дневник по уходу за домашними» — это приложение, которое позволит хранить информацию обо всех действиях – питание, вакцинации, посещения ветеринарного врача, покупка корма, ведение заметок и многое другое. Дневник показывает, есть ли запланированные действия – предстоящие, пропущенные или будущие. Кроме запланированных типов действий, можно добавлять новое действие для каждого из домашних питомцев.

Недостатки:

- Нет возможности авторизации и сохранения данных;
- Нет информации о ветеринарных клиниках.

Интерфейс приложения представлен ниже (Рисунок 5).

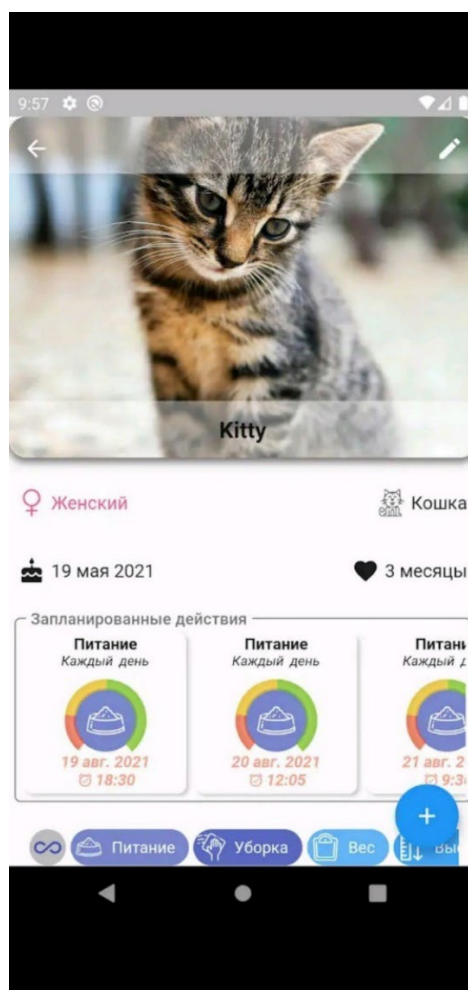


Рисунок 5 - Интерфейс приложения «Дневник по уходу за домашними»

2.3 Графическое описание работы системы

2.3.1 Диаграмма IDEF0

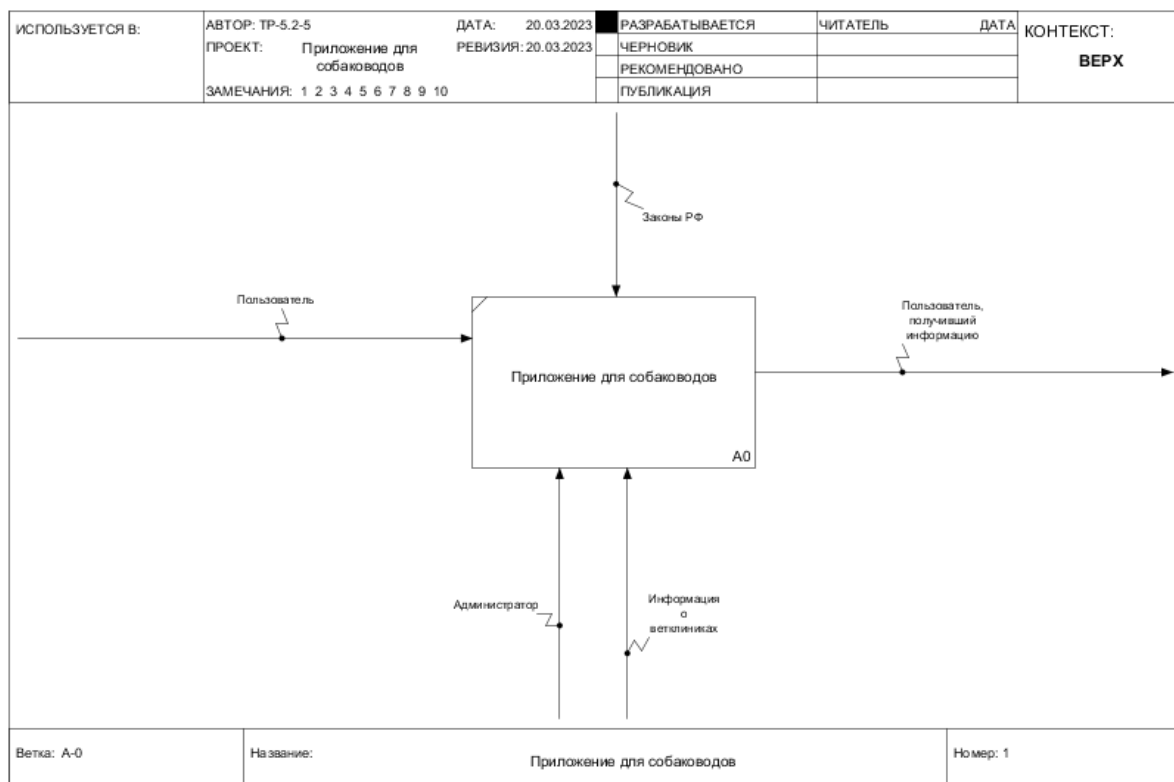


Рисунок 6 - Диаграмма IFEF0

Данная диаграмма используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции.

2.3.2 Диаграмма прецедентов

На Рисунке 7 продемонстрирована диаграмма Use – Case, которая показывает какие сценарии использования приложения доступны различным пользователям веб-приложения.

В этой системе можно выделить следующие группы пользователей:

- незарегистрированный пользователь;
- зарегистрированный пользователь;
- администратор.

Каждая из групп имеет разный доступ к возможностям разрабатываемой системы. Авторизованный пользователь наследуется от неавторизованного, тем самым получая доступ к его функциям также.

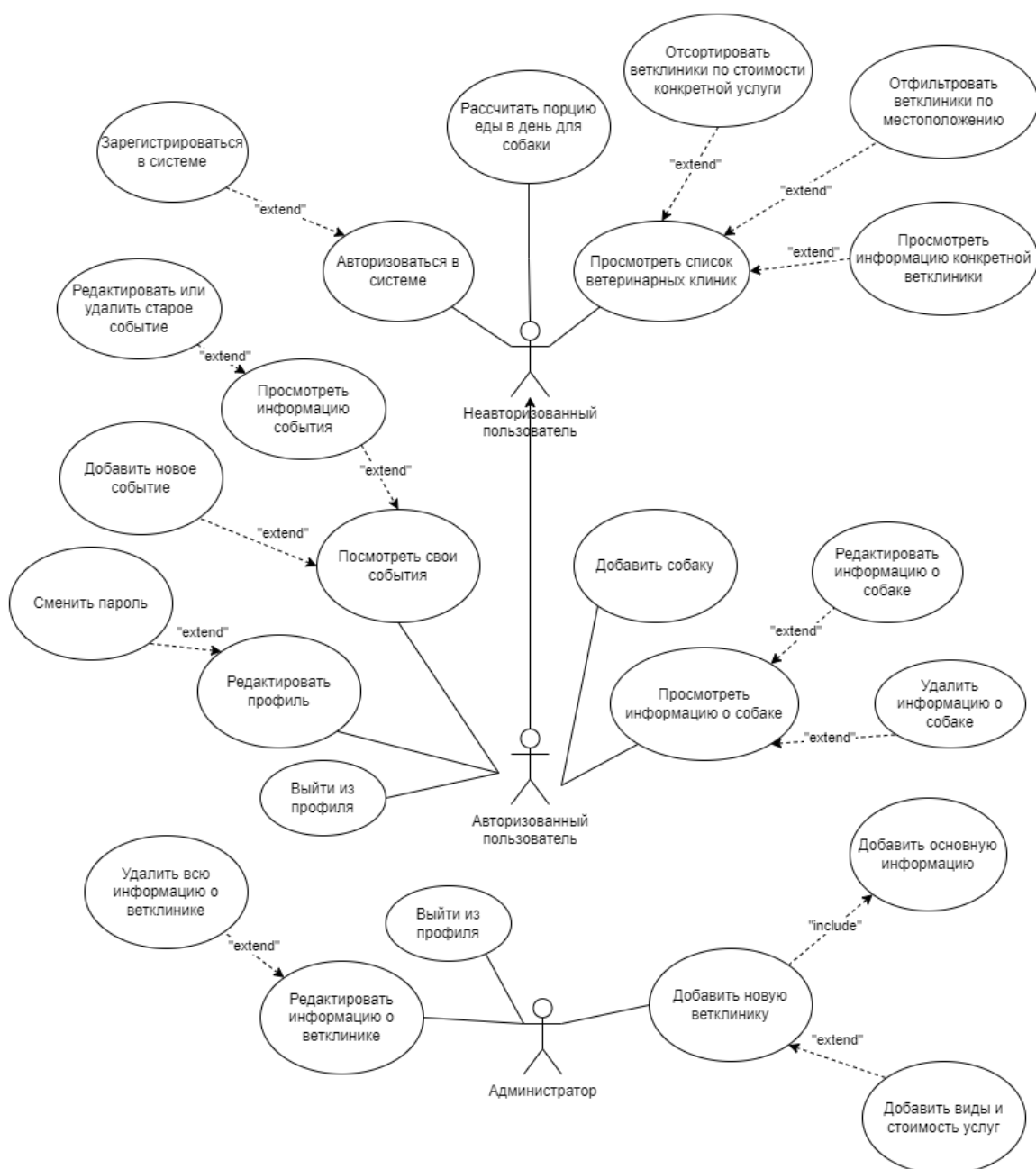


Рисунок 7 - Use – case диаграмма

2.3.3 Диаграмма последовательности

Диаграмма последовательности используется для визуализации последовательности действий, которые выполняются в ходе взаимодействия различных объектов и компонентов в системе. Для удобства восприятия она была разделена на две части – для администратора (Рисунок 8) и для пользователя (Рисунки 9 и 10).

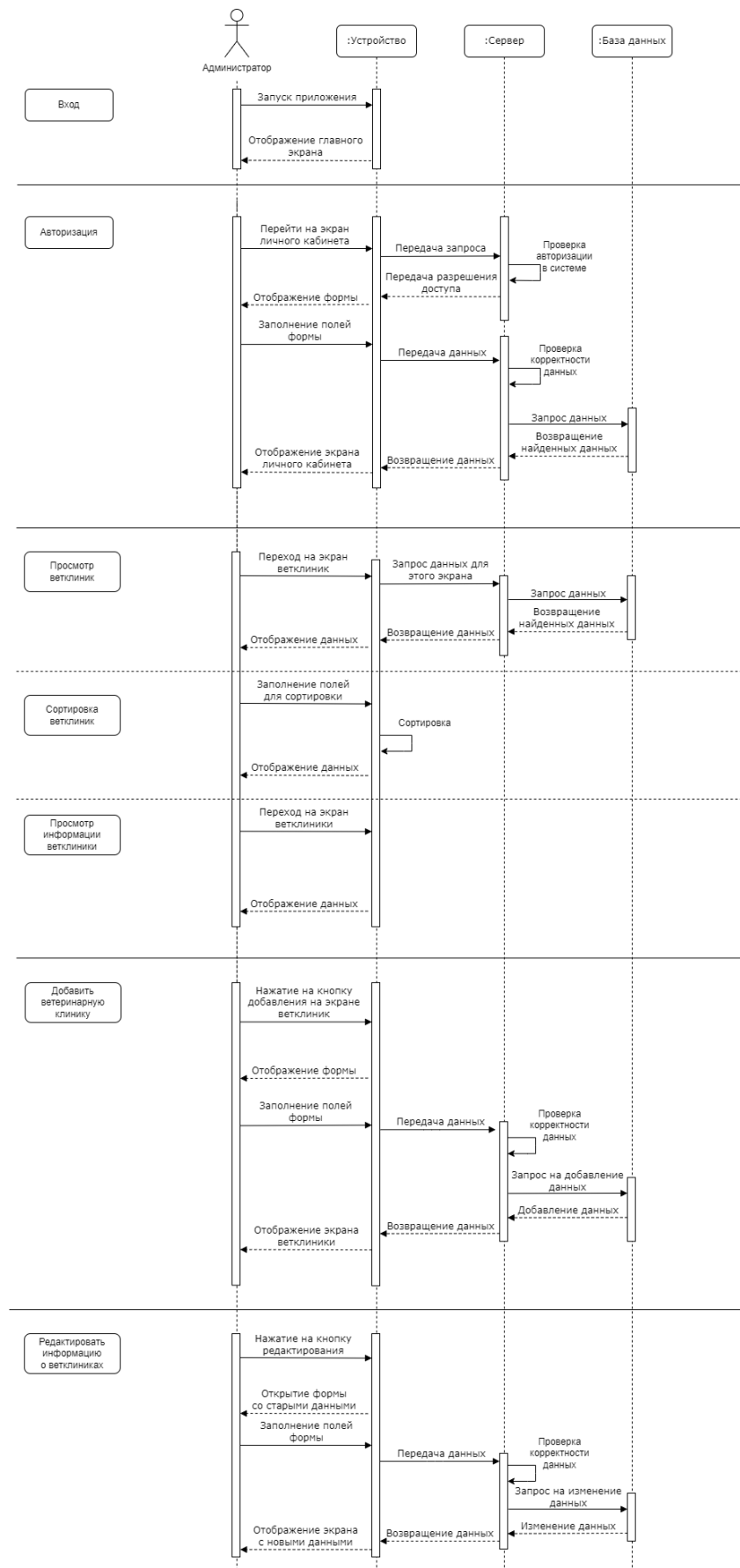


Рисунок 8 - Диаграмма последовательности для администратора

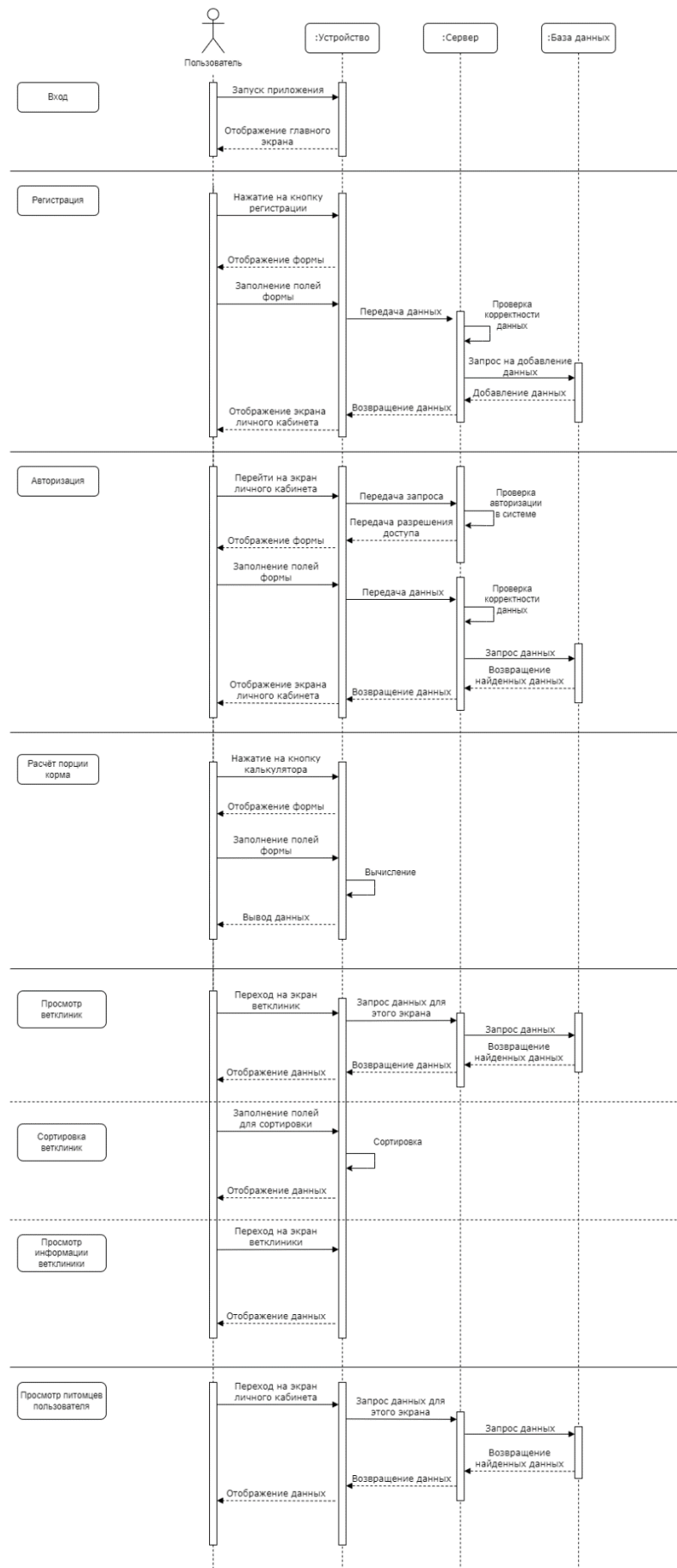


Рисунок 9 - Диаграмма последовательности для пользователя (1 часть)

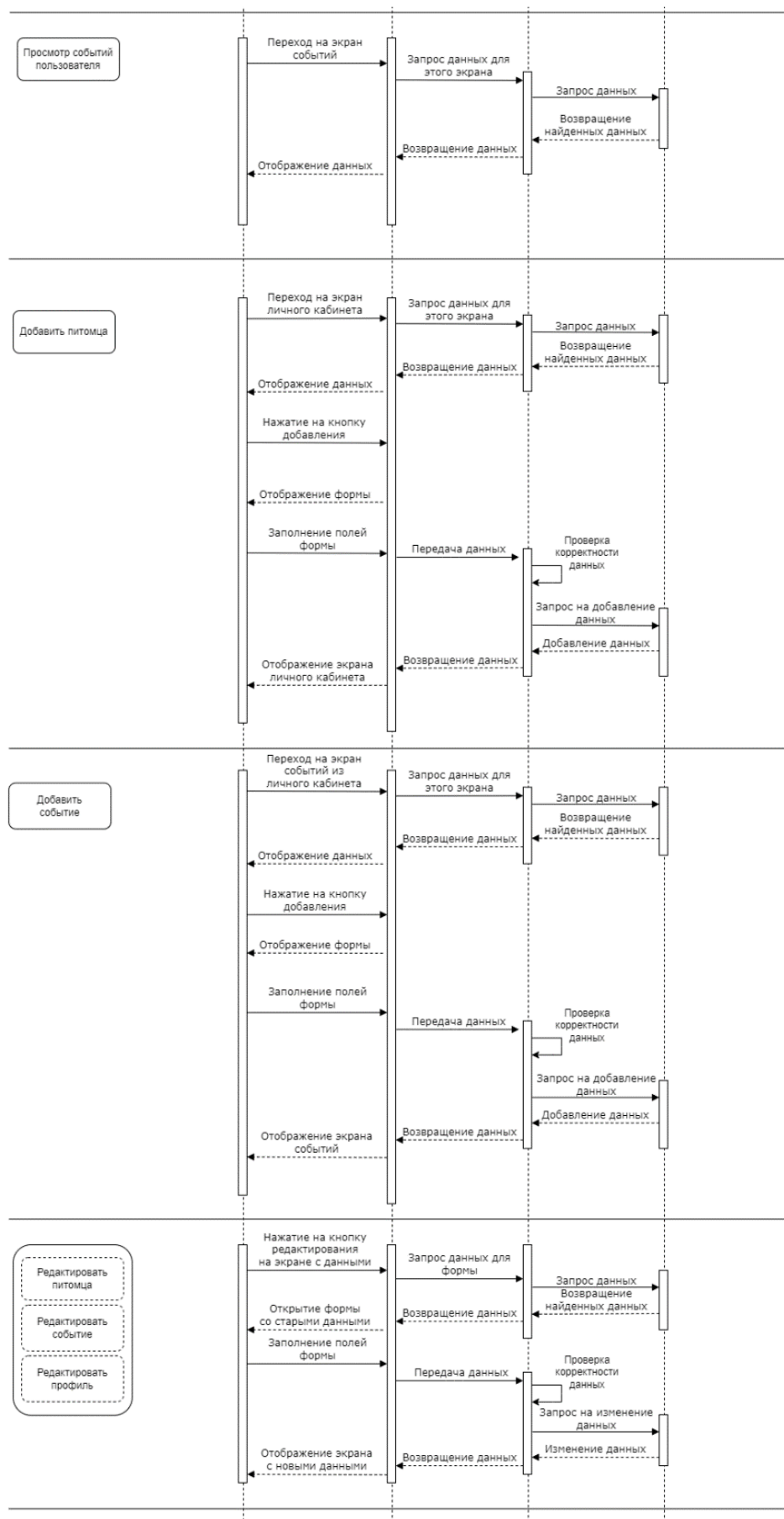


Рисунок 10 - Диаграмма последовательности для пользователя (2 часть)

2.3.4 Диаграмма активности

Диаграмма активности используется для моделирования процессов и последовательностей действий, которые должны быть выполнены для достижения определенной цели (Рисунок 11).

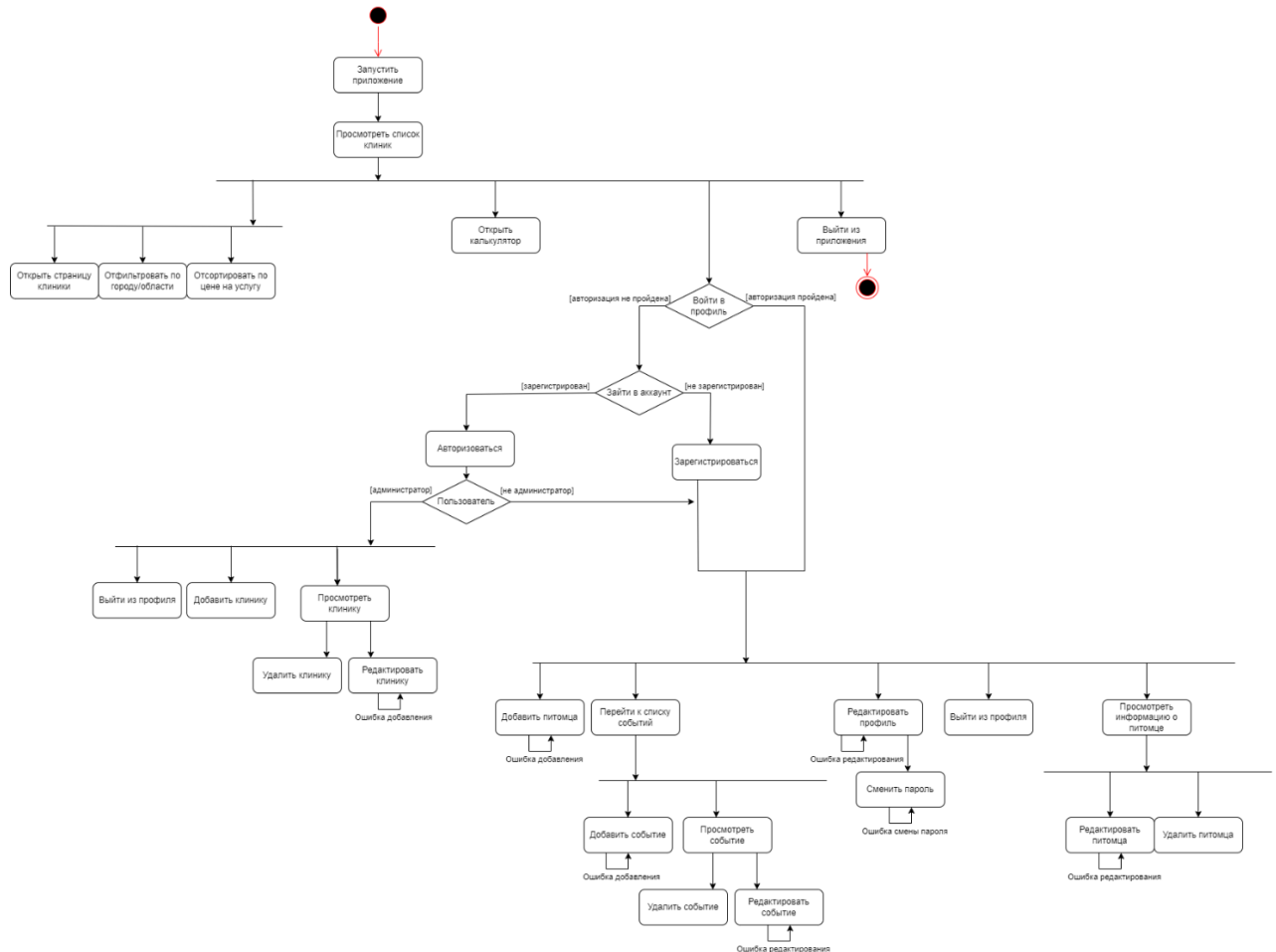


Рисунок 11 - Диаграмма активности

2.3.5 Диаграмма состояний

Диаграмма состояний используется для описания поведения объекта или системы в различных состояниях, а также переходы между ними.

Диаграмма была разделена на три части для удобства восприятия:

- для администратора (Рисунок 12);
- для неавторизованного пользователя (Рисунок 13);
- для авторизованного пользователя (Рисунок 14).



Рисунок 12 - Диаграмма состояний для администратора

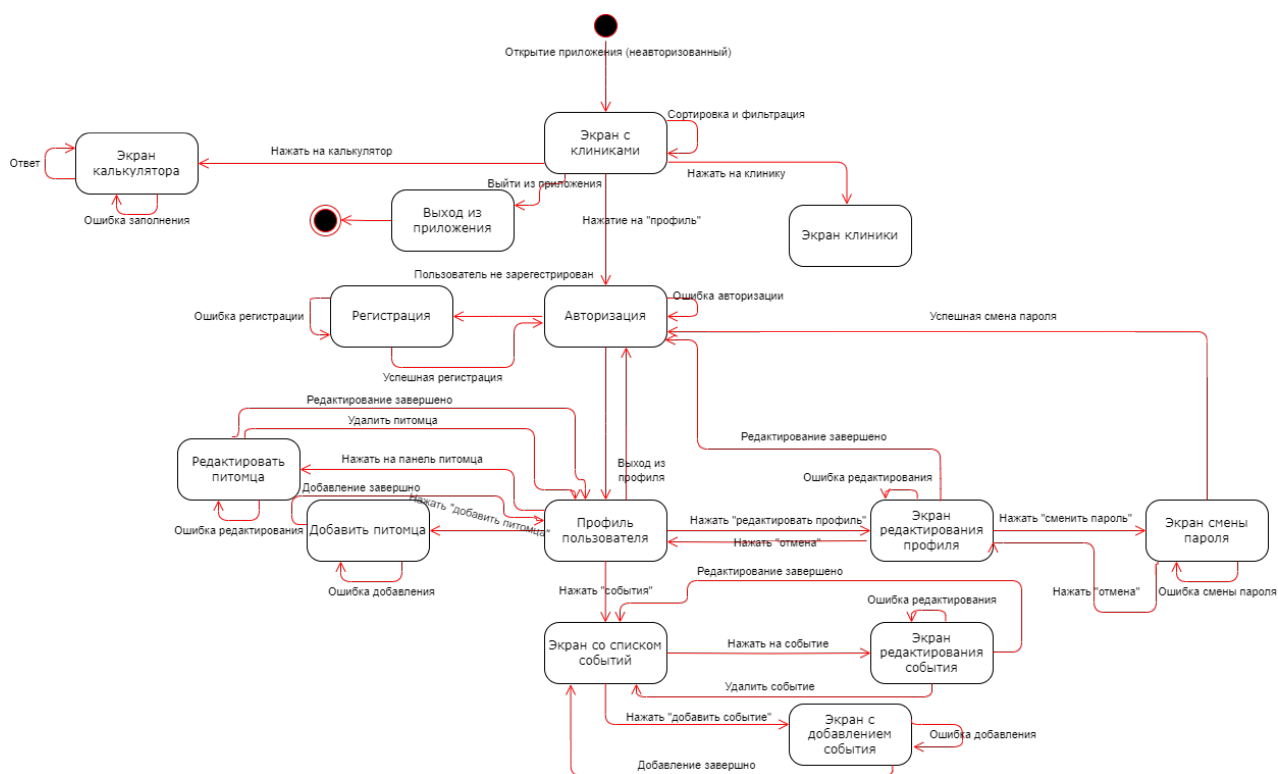


Рисунок 13 - Диаграмма состояний для неавторизованного пользователя

Диаграмма классов используется для описания структуры объектно-ориентированной системы (Рисунок 15). Она показывает классы, их атрибуты и методы, а также связи между классами.

2.3.7 Диаграмма объектов

Диаграммы объектов представляют собой экземпляр диаграммы классов, являются производными от них (Рисунок 16).

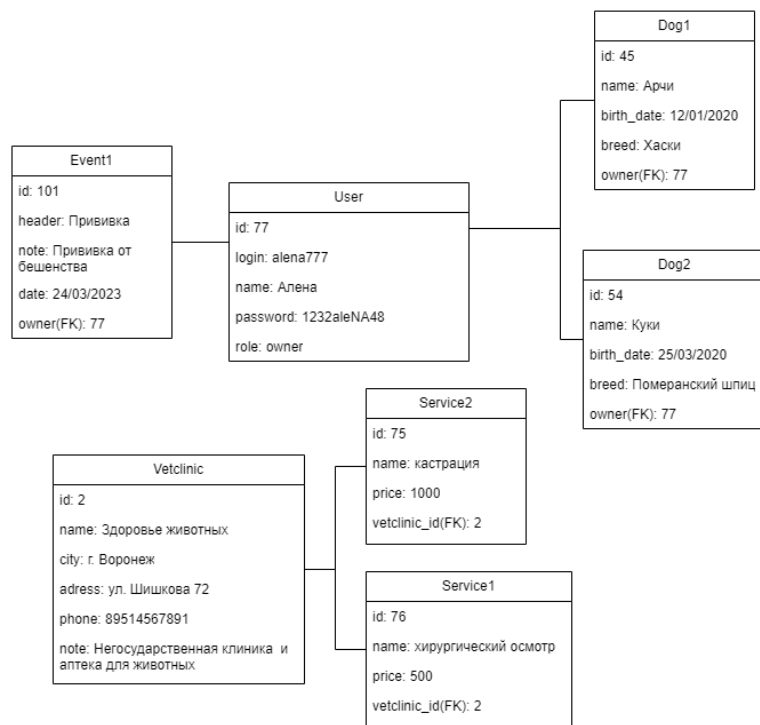


Рисунок 16 - Диаграмма объектов

2.3.8 Диаграмма развертывания

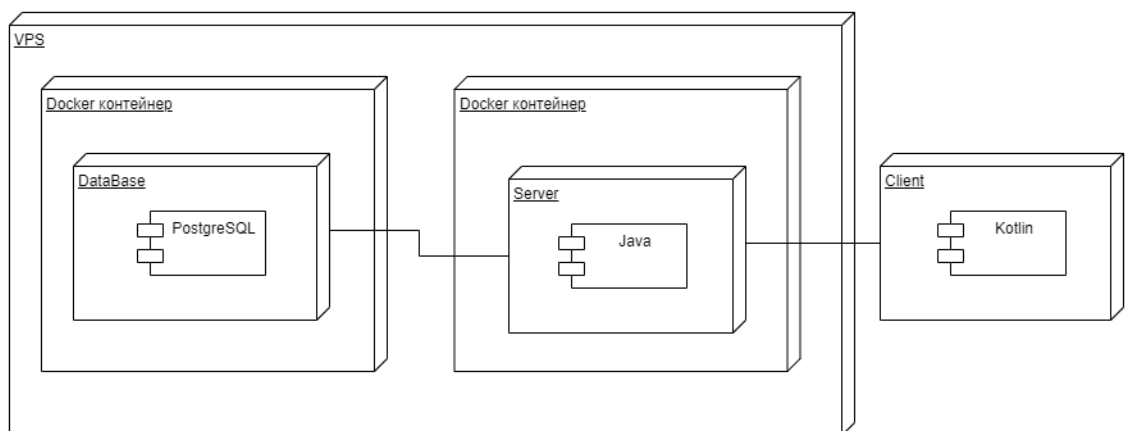


Рисунок 17 - Диаграмма развертывания

2.3.9 Диаграммы сотрудничества

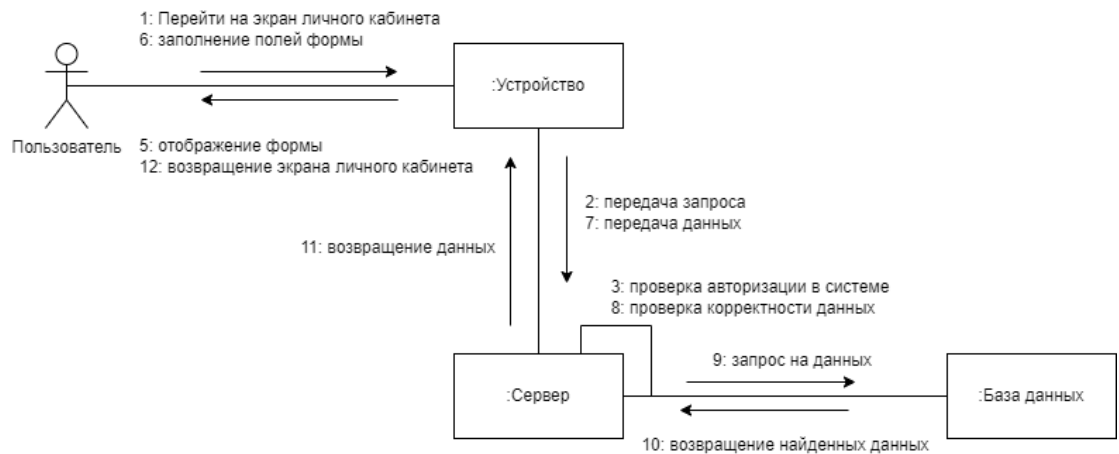


Рисунок 18 - Диаграмма сотрудничества – Авторизация

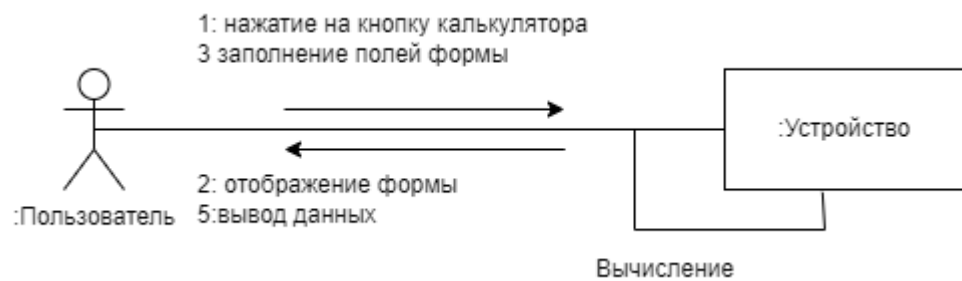


Рисунок 19 - Диаграмма сотрудничества – Расчет порции корма

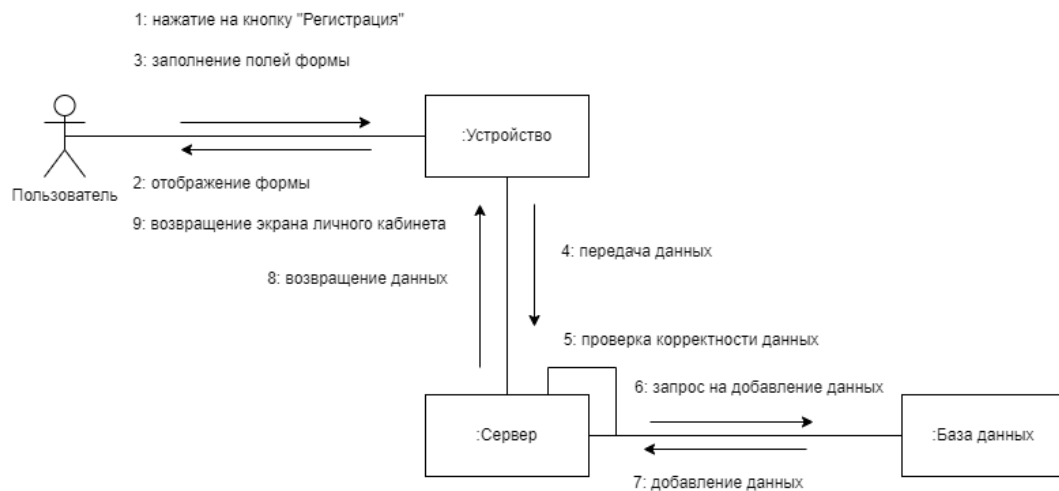


Рисунок 20 - Диаграмма сотрудничества – Регистрация

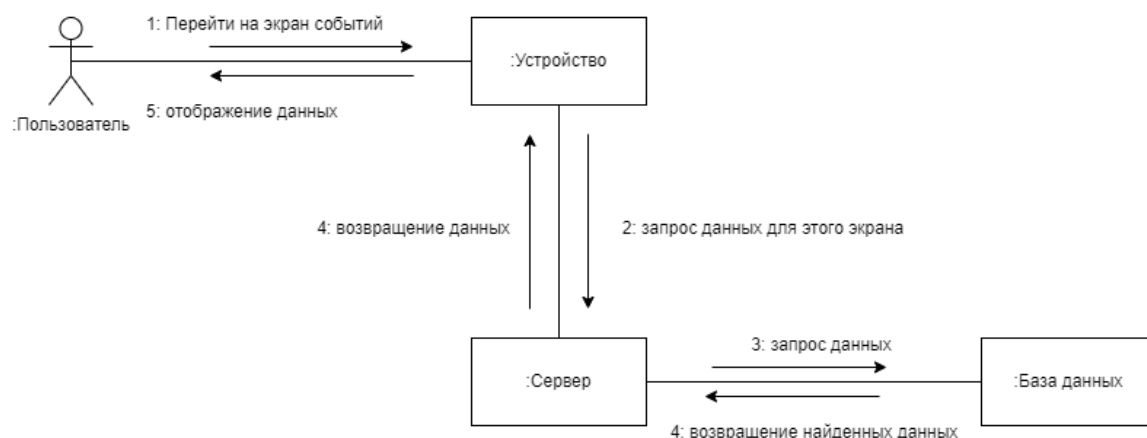


Рисунок 21 - Диаграмма сотрудничества – Просмотр ветеринарных клиник

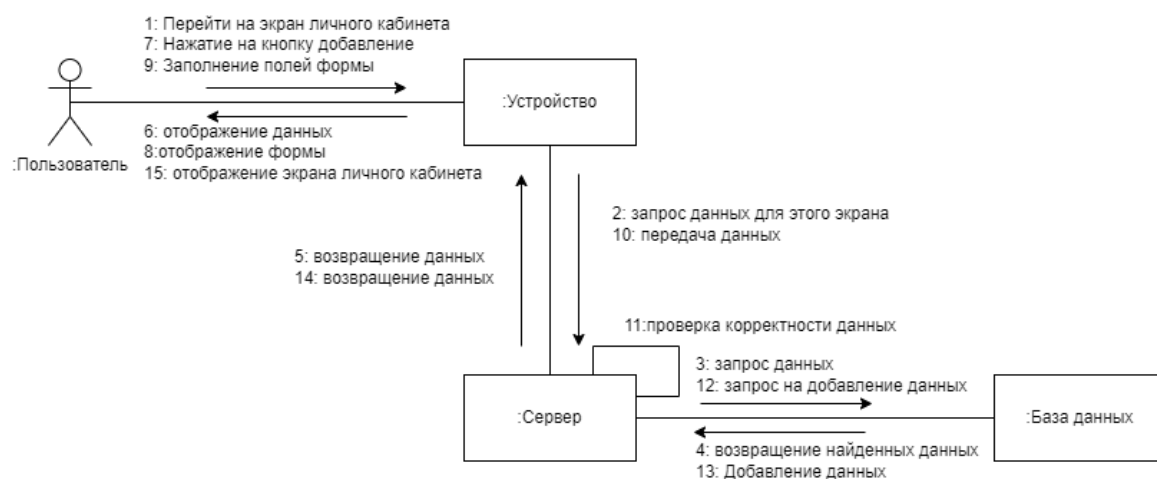


Рисунок 22 - Диаграмма сотрудничества – Добавление питомца

3 Реализация

3.1 Средства реализации

Для разработки приложения будут использоваться:

- Java 17 — строго типизированный объектно – ориентированный язык программирования. Данная версия имеет улучшенную производительность, новые языковые функции, например, усовершенствования в моделях программирования, расширенную поддержку для контейнеров, а также добавление новых ключевых слов и синтаксических конструкций, что делает код более компактным, понятным и легким в разработке;
- Kotlin — статически типизированный язык программирования, который разрабатывался JetBrains с учетом совместимости с Java Virtual Machine (JVM). Был выбран потому, что полностью совместим с Java, что означает возможность использовать существующий код и библиотеки, а также наследовать и использовать Java-классы и интерфейсы в коде Kotlin. Кроме того, данный язык имеет краткий синтаксис, что может значительно сократить объем кода и упростить разработку;
- PostgreSQL — объектно – реляционная система управления базами данных. Является продуктом с открытым исходным кодом, который поддерживается многими серверами, в связи с чем и был выбран. Кроме этого, PostgreSQL надежен, стабилен и предлагает множество функций и возможностей для расширения. Также он поддерживает множество дополнительных стандартов;
- Spring Boot Framework — универсальный фреймворк с открытым исходным кодом для Java-платформы. Был выбран, так как он предоставляет мощные и удобные механизмы построения клиент–серверных приложений, в связи с чем пользуется огромным спросом и является фактически стандартом в построении приложений на Java;

Вспомогательный инструментарий:

- Miro — это онлайн – инструмент для совместной работы и визуализации идей. Он предоставляет возможность создавать диаграммы, прототипы интерфейсов, дизайн – макеты, схемы, планы проектов и многое другое;
- Draw.io — это сервис, предназначенный для формирования диаграмм и схем;
- Swagger — инструмент для разработки, который предоставляет набор инструментов и спецификаций для описания API и создания его интерактивной документации;
- Git — это распределенная система управления версиями (Version Control System, VCS). Он позволяет разработчикам отслеживать изменения в исходном коде, создавать ветки для параллельной работы, сливать изменения и откатывать к предыдущим версиям кода;
- GitHub — это веб – платформа, которая предоставляет хостинг для репозиториях Git и дополнительные функции для совместной работы над проектами;
- Trello — это онлайн – инструмент для управления проектами и задачами, основанный на концепции канбан – доски. Он предоставляет гибкую и интуитивно понятную платформу для организации и отслеживания работы над проектами, управления задачами и координации команды.

3.2 Реализация Backend

3.2.1 База данных

Для описания разработанной базы данных были созданы ER (Entity Relationship) и физическая диаграммы (Рисунок 23 и Рисунок 24 соответственно).

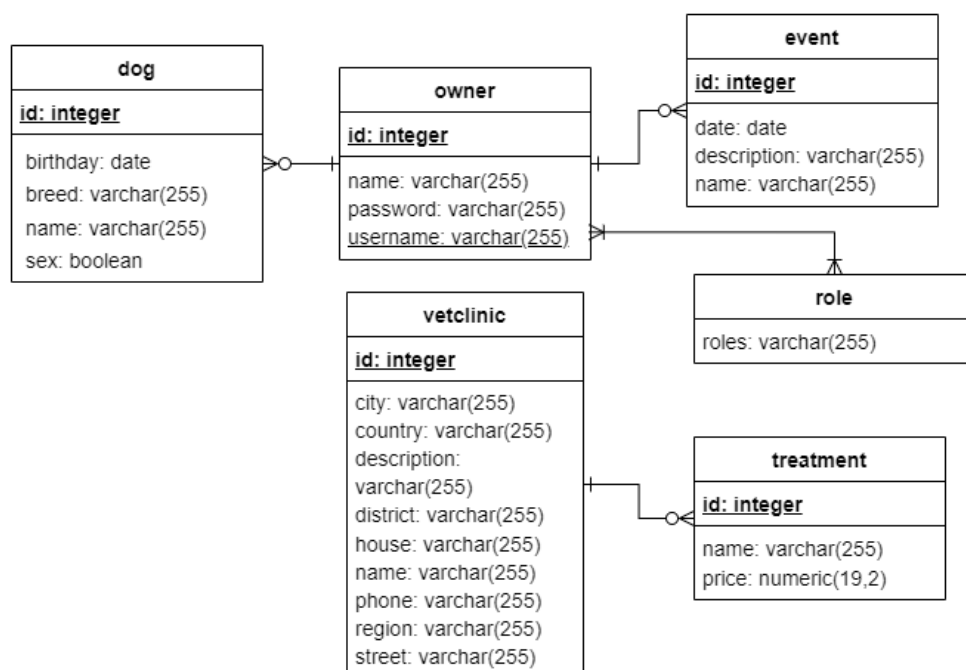


Рисунок 23 - ER – диаграмма базы данных

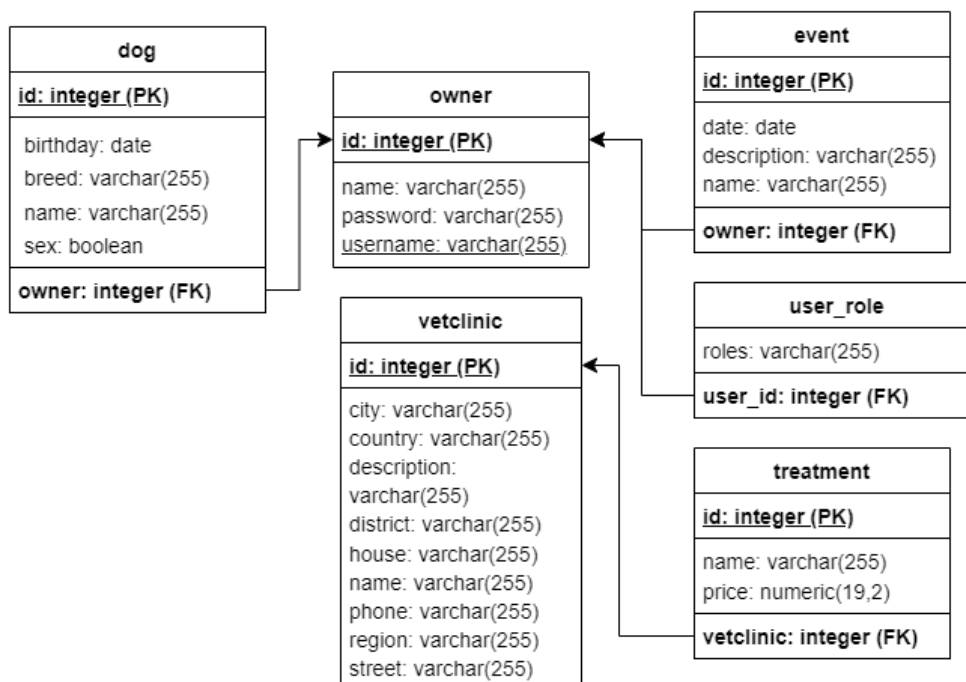


Рисунок 24 - Физическая диаграмма базы данных

База данных состоит из 6 таблиц: «dog», «owner», «event», «vetclinic», «treatment» и «user_role».

Таблица «owner» хранит информацию о пользователях. Она содержит следующие столбцы:

- «id» – идентификационный номер, который генерируется автоматически и является первичным ключом. Тип данных – integer (целое число);
- «name» – имя пользователя собаки. Тип данных – varchar (переменная строка);
- «password» – закодированный пароль пользователя. Тип данных – varchar (переменная строка);
- «username» – логин пользователя. Тип данных – varchar (переменная строка). Является уникальным значением.

Таблица «dog» содержит информацию о собаках, принадлежащих пользователю. Она содержит следующие столбцы:

- «id» – идентификационный номер, генерирующийся автоматически и являющийся первичным ключом. Тип данных – integer (целое число);
- «birthday» – дата рождения собаки. Тип данных – date (дата);
- «breed» – порода собаки. Тип данных – varchar (переменная строка);
- «name» – кличка собаки. Тип данных – varchar (переменная строка);
- «sex» – пол собаки. Тип данных – boolean (логическое значение);
- «owner» – внешний ключ, ссылающийся идентификационный номер таблицы владельцев собак. Тип данных – integer (целое число).

Таблица «event» содержит информацию о событиях владельца собаки. Она содержит следующие столбцы:

- «id» – идентификационный номер, генерирующийся автоматически и являющийся первичным ключом. Тип данных – integer (целое число);
- «date» – дата события. Тип – данных date (дата);

- «description» – описание события. Тип данных – varchar (переменная строка);
- «name» – название события. Тип данных – varchar (переменная строка);
- «owner» – внешний ключ, ссылающийся идентификационный номер таблицы владельцев собак. Тип данных – integer (целое число).

Таблицы «dog» и «event» имеют связь «Многие – к – одному» с таблицей «owner», то есть у владельца собаки может быть одна или несколько собак и одно или несколько событий или не быть их вовсе, а у собаки и события может быть только один пользователь.

Таблица «user_role» хранит информацию о ролях пользователей. Она содержит следующие столбцы:

- «user_id» – внешний ключ, ссылающийся идентификационный номер таблицы владельцев собак. Тип данных – integer (целое число);
- «roles» – роли пользователя. Тип данных – varchar (переменная строка).

У пользователя могут быть роли обычного пользователя («USER») или администратора («ADMIN»).

Таблица «vetclinic» хранит информацию о ветеринарных клиниках. Она содержит следующие столбцы:

- «id» – идентификационный номер, генерирующийся автоматически и являющийся первичным ключом. Тип данных – integer (целое число);
- «name» – название клиники. Тип данных – varchar (переменная строка);
- «phone» – номер телефона клиники. Тип данных – varchar (переменная строка);

- «description» – описание клиники. Тип данных – varchar (переменная строка);
- «city» – город, представленный. Тип данных – varchar (переменная строка);
- «country» – страна. Тип данных – varchar (переменная строка);
- «region»: регион. Тип данных – varchar (переменная строка);
- «district» – район. Тип данных – varchar (переменная строка);
- «street»: улица. Тип данных – varchar (переменная строка);
- «house» – номер дома. Тип данных – varchar (переменная строка).

Таблица «treatment» содержит информацию об услугах ветеринарной клиники. Она содержит следующие столбцы:

- «id» – идентификационный номер, генерирующийся автоматически и являющийся первичным ключом. Тип данных – integer (целое число);
- «name» – название услуги. Тип данных – varchar (переменная строка);
- «price» – стоимость услуги. Тип данных – numeric(19, 2) (число с фиксированной точностью, где 19 – общее количество символов, а 2 – число символов после запятой).
- «vetclinic» - внешний ключ, который ссылается на идентификационный номер в таблице ветеринарных клиник. Тип данных – integer (целое число).

Таблица «treatment» имеет связь «Многие – к – одному» с таблицей «vetclinic», то есть у ветеринарной клиники может быть одна или несколько услуг или не быть их вовсе, а у услуги может быть только одна ветеринарная клиника.

3.2.2 Архитектура серверной части приложения

Структура сервера приложения соответствует архитектуре «Model-View-Controller» (MVC).

В контексте разработанной архитектуры:

- Классы Entity и Repository соответствуют модели (Model) и представляют данные, хранящиеся в базе данных и методы для их доступа и изменения;
- Классы Service содержат бизнес-логику и выполняют операции над данными, предоставляемыми моделью;
- Классы Controller обрабатывают HTTP – запросы, взаимодействуют с сервисами и подготавливают данные для отправки клиенту (View);
- Классы DTO и Mapper служат для передачи данных между клиентом и сервером (View и Controller), а также для преобразования данных между моделью и DTO.

Шаблон MVC обеспечивает разделение ответственности между различными компонентами приложения, что повышает его модульность, упрощает тестирование и поддержку кода. Он также позволяет легко вносить изменения в различные компоненты приложения независимо друг от друга.

Более подробное описание групп классов разрабатываемого мобильного приложения:

- Entity – классы, представляющие объекты, которые связаны с базой данных. Они содержат аннотации JPA, такие как «Entity», «Table», «Id» и другие, чтобы указать ссылку на соответствующую таблицу и столбцы. Entity классы также могут содержать аннотации для определения отношений между таблицами, такие как «ManyToOne», «OneToMany» и другие;
- DTO (Data Transfer Object) – классы, представляющие объекты, которые используются для передачи данных между клиентской и серверной частями приложения. Они содержат только необходимые поля данных и обычно не содержат бизнес – логики;
- Mapper – интерфейсы, которые используются для преобразования объектов между различными типами. В данном случае, они

использовались для перехода экземпляров классов DTO в Entity и наоборот;

- Repository – интерфейсы, которые представляют доступ к базе данных и содержат методы для выполнения операций CRUD (создание, чтение, обновление, удаление) над Entity объектами. Они наследуют интерфейс JpaRepository, который предоставляет базовую функциональность для работы с базой данных;
- Service – классы, которые представляют бизнес-логику приложения. Они содержат методы для обработки запросов от контроллеров, взаимодействия с репозиторием и другие операции, связанные с обработкой данных. Service классы содержат аннотацию «Service» для инъекции зависимостей и управления транзакциями;
- Controller – классы, которые обрабатывают HTTP – запросы от клиента и взаимодействуют с соответствующими Service классами. Они содержат методы – обработчики запросов, которые аннотированы с помощью «RequestMapping» или других аннотаций, указывающих путь и метод запроса;
- Config – классы, которые содержат конфигурацию Spring Boot приложения. В реализации данного проекта были использованы два конфигурационных класса – «WebSecurityConfig», который определяет настройку безопасности (например, использование классов «BCryptPasswordEncoder» и «JwtFilter», работа которых будет описана далее, и настройка доступа к различным URL) и «SwaggerSecurity», использующийся для настройки документации Swagger.

Для соответствия заявленным требованиям безопасности использовались следующие компоненты:

- JwtFilter – фильтр, отвечающий за обработку и проверку JWT (JSON Web Token) для аутентификации и авторизации

пользователей. Сначала он извлекает токен из заголовка «Authorization» запроса с префиксом «Bearer», после проверяет его валидность и целостность, используя секретный ключ, извлекает информацию о пользователе и устанавливает аутентификацию в контексте безопасности Spring Security;

- Использование аннотации «Query», которая определяет пользовательские запросы (в данном случае SQL) в методах репозитория. То есть, доступ к базе данных предоставляется не через генерацию SQL – запросов на основе именования функций. Такой способ является одним из методов защиты от SQL – инъекций: использование параметризованных запросов, где значения передаются через параметры, позволяет правильно обрабатывать значения, предотвращая возможность внедрения злонамеренного SQL-кода;
- BCryptPasswordEncoder – класс в Spring Security, который используется для шифрования паролей с использованием алгоритма BCrypt. BCrypt (Blowfish Crypt) — это хэш – функция, разработанная для хеширования паролей. Она является одним из наиболее надежных и безопасных алгоритмов хеширования паролей, используемых в настоящее время.

Сервер разрабатываемого мобильного приложения также отвечает архитектурному стилю REST API. Он обеспечивает коммуникацию и взаимодействие между клиентскими приложениями и сервером с использованием протокола HTTP для передачи данных и выполнения операций над ними.

3.3 Реализация Frontend

4 Тестирование