

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 \_\_\_\_\_ С.Д. Махортов, д.ф.- м.н., доцент \_\_\_\_\_.20\_\_

Обучающийся \_\_\_\_\_ А.А. Полев, 3 курс, д/о

Обучающийся \_\_\_\_\_ П.О. Федосова, 3 курс, д/о

Обучающийся \_\_\_\_\_ К.В. Брюхов, 3 курс, д/о

Руководитель \_\_\_\_\_ В.С. Тарасов, ст. преподаватель

Руководитель \_\_\_\_\_ И.В. Клейменов, ассистент

Воронеж 2023

## Содержание

Содержание .....	2
Введение .....	5
1 Постановка задачи .....	6
1.1 Цель проекта .....	6
1.2 Задачи проекта .....	6
1.3 Требования к разрабатываемой системе .....	6
1.3.1 Требования к структуре .....	6
1.3.2 Функциональные требования .....	6
1.3.3 Требования к защите информации .....	7
2 Анализ предметной области .....	8
2.1 Терминология (гlossарий) предметной области .....	8
2.2 Обзор аналогов .....	10
2.2.1 11pets: Уход за питомцем .....	10
2.2.2 Dog Health .....	11
2.2.3 Дневник по уходу за домашними .....	13
2.3 Моделирование работы системы .....	14
2.3.1 Диаграмма IDEF0 .....	14
2.3.2 Диаграмма прецедентов .....	15
2.3.3 Диаграмма последовательности .....	16
2.3.4 Диаграмма активности .....	20
2.3.5 Диаграмма состояний .....	20
2.3.6 Диаграмма классов .....	22
2.3.7 Диаграмма объектов .....	23
2.3.8 Диаграмма развертывания .....	23

2.3.9 Диаграммы сотрудничества.....	24
3 Реализация.....	26
3.1 Средства реализации.....	26
3.2 Реализация серверной части .....	28
3.2.1 База данных .....	28
3.2.2 Архитектура серверной части приложения .....	32
3.3 Реализация клиентской части .....	39
3.3.1 Панель навигации .....	40
3.3.2 Экран списка ветеринарных клиник .....	40
3.3.3 Экран ветеринарной клиники .....	42
3.3.4 Экран калькулятора .....	43
3.3.5 Экран авторизации.....	44
3.3.6 Экран регистрации.....	45
3.3.7 Экран профиля .....	46
3.3.8 Экран добавления питомца .....	47
3.3.9 Экран редактирования питомца .....	48
3.3.10 Экран списка событий.....	49
3.3.11 Экран добавления события .....	50
3.3.12 Экран редактирования события .....	51
3.3.13 Экран редактирования профиля .....	52
3.3.14 Экран смены пароля .....	53
3.3.15 Экран личного кабинета администратора.....	54
3.3.16 Экран добавления ветеринарной клиники .....	54
3.3.17 Экран редактирования ветеринарной клиники.....	55
4 Тестирование .....	57

4.1 Дымовое тестирование .....	57
4.2 Интеграционные тесты для серверной части .....	59
4.3 Unit – тесты для клиентской части.....	61
5 Аналитика .....	64
5.1.1 Воронка авторизации.....	64
5.1.2 Воронка регистрации.....	64
5.1.3 Воронка калькулятора .....	65
5.1.4 Воронка просмотра клиник.....	66
5.1.5 Воронка редактирования данных пользователя .....	67
5.1.6 Воронка смены пароля .....	67
5.1.7 Воронка добавления информации о собаке .....	68
5.1.8 Воронка добавления информации о событии .....	69
Заключение .....	70
Список используемых источников.....	72

## **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.

Целью проекта стала разработка мобильного android – приложения для собаководов. 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 рассмотрен процесс проектирования приложения, включая обзор аналогов, анализ предметной области, создание базы данных, разработка интерфейса, реализация серверной и клиентской частей и их тестирование.

## **1 Постановка задачи**

### **1.1 Цель проекта**

Разработка мобильного android – приложения для помощи человеку в уходе за собакой, которое также предоставляет информацию о ветеринарных клиниках и их услугах.

### **1.2 Задачи проекта**

Создание приложения для собаководов, обладающего возможностями:

- сортировать ветеринарные клиники по цене на услугу;
- вычислять оптимальное количество пищи для питомца;
- сохранять информацию о событии (прививка, поход к ветеринару и тому подобное).

### **1.3 Требования к разрабатываемой системе**

#### **1.3.1 Требования к структуре**

Приложение должно быть построено на трехуровневой архитектуре: клиентская часть (мобильное приложение) – серверная часть – 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 [1];
- 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 — приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для платформы Android;
- Android — это операционная система с открытым исходным кодом, созданная для мобильных устройств на основе модифицированного ядра Linux;
- 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**

Для создания программного средства, необходимо изучить представленные аналоги и выделить их основные недостатки и преимущества, определить ключевые особенности тенденций в данном направлении.

В качестве исследуемых аналогов были выбраны программные продукты, связанные с предоставлением и хранением данных о питомце, ветеринарных клиниках и так далее. Основным критерием для выбора служила актуальность данных программных средств, частота их использования, представленные функции. Источником информации послужили электронные базы в сети Интернет. В результате поиска были выявлены 3 программных продукта:

- приложение 11pets: Уход за питомцем [2];
- приложение DogHealth [3];
- приложение Дневник по уходу за домашними [4];

### **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:

- нет возможности сортировать услуги ветеринарных клиник по цене;
- нет возможности просматривать услуги ветеринарных клиник.

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

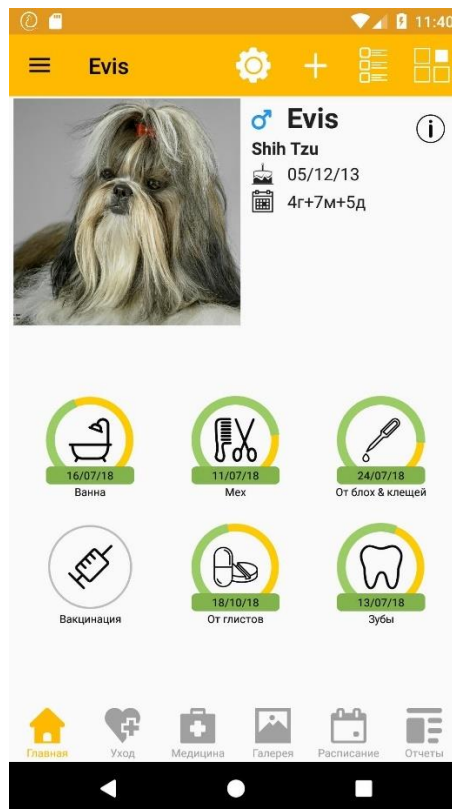


Рисунок 1 - Интерфейс приложения «11pets: Уход за питомцем»

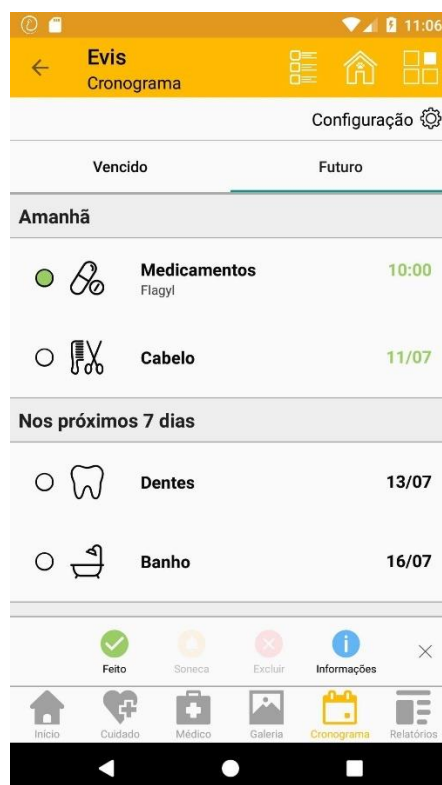


Рисунок 2 - Интерфейс приложения «11pets: Уход за питомцем»

## 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).

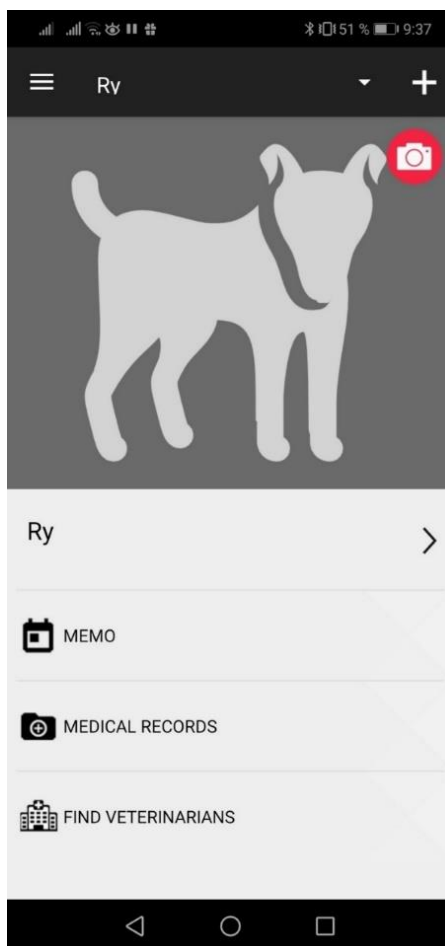


Рисунок 3 - Интерфейс приложения «Dog Health»

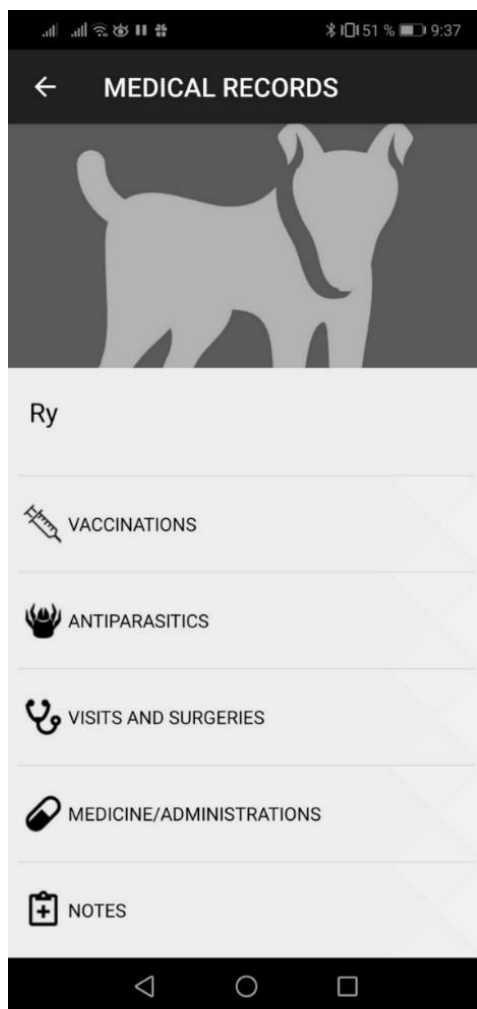


Рисунок 4 - Интерфейс приложения «Dog Health»

### 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 создать учетную запись;
- нет информации о ветеринарных клиниках.

Интерфейс приложения представлен ниже (Рисунок 5).

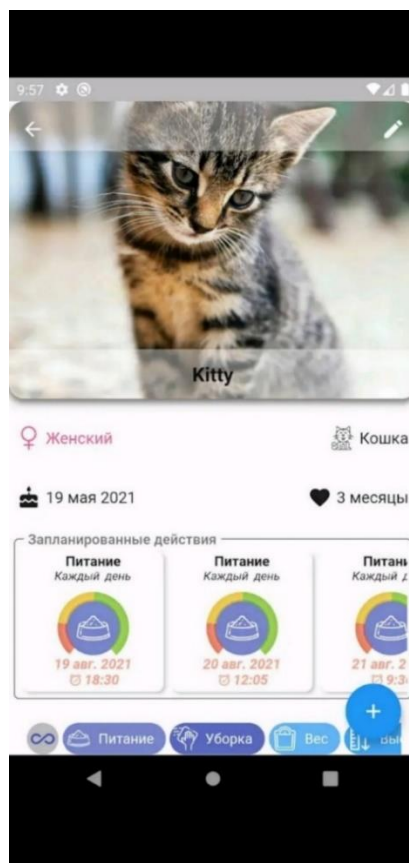


Рисунок 5 - Интерфейс приложения «Дневник по уходу за домашними»

## 2.3 Моделирование работы системы

### 2.3.1 Диаграмма IDEF0

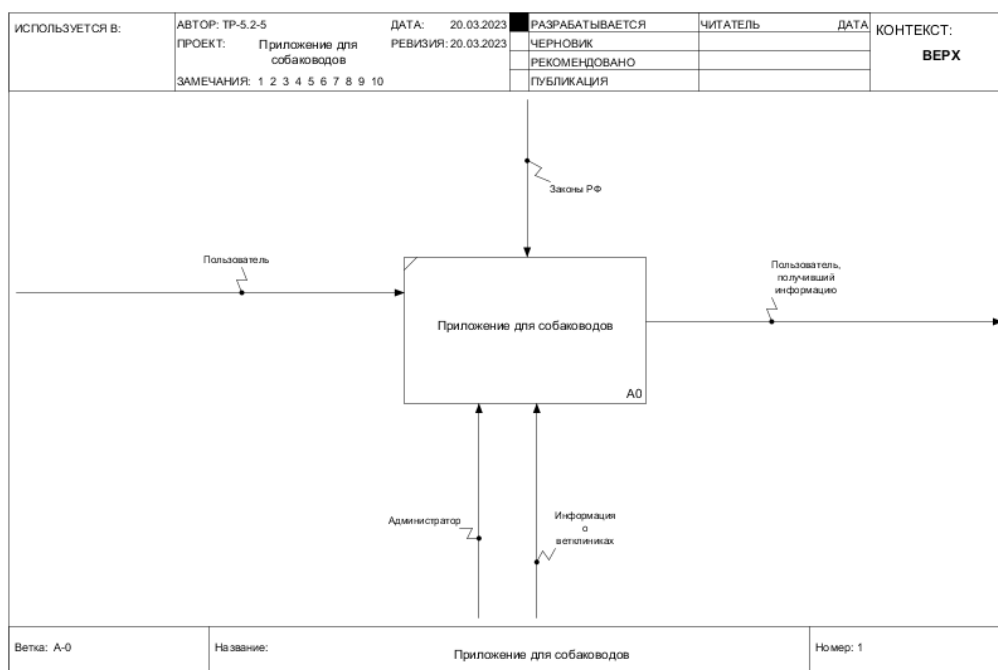


Рисунок 6 - Диаграмма IFEF0

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.

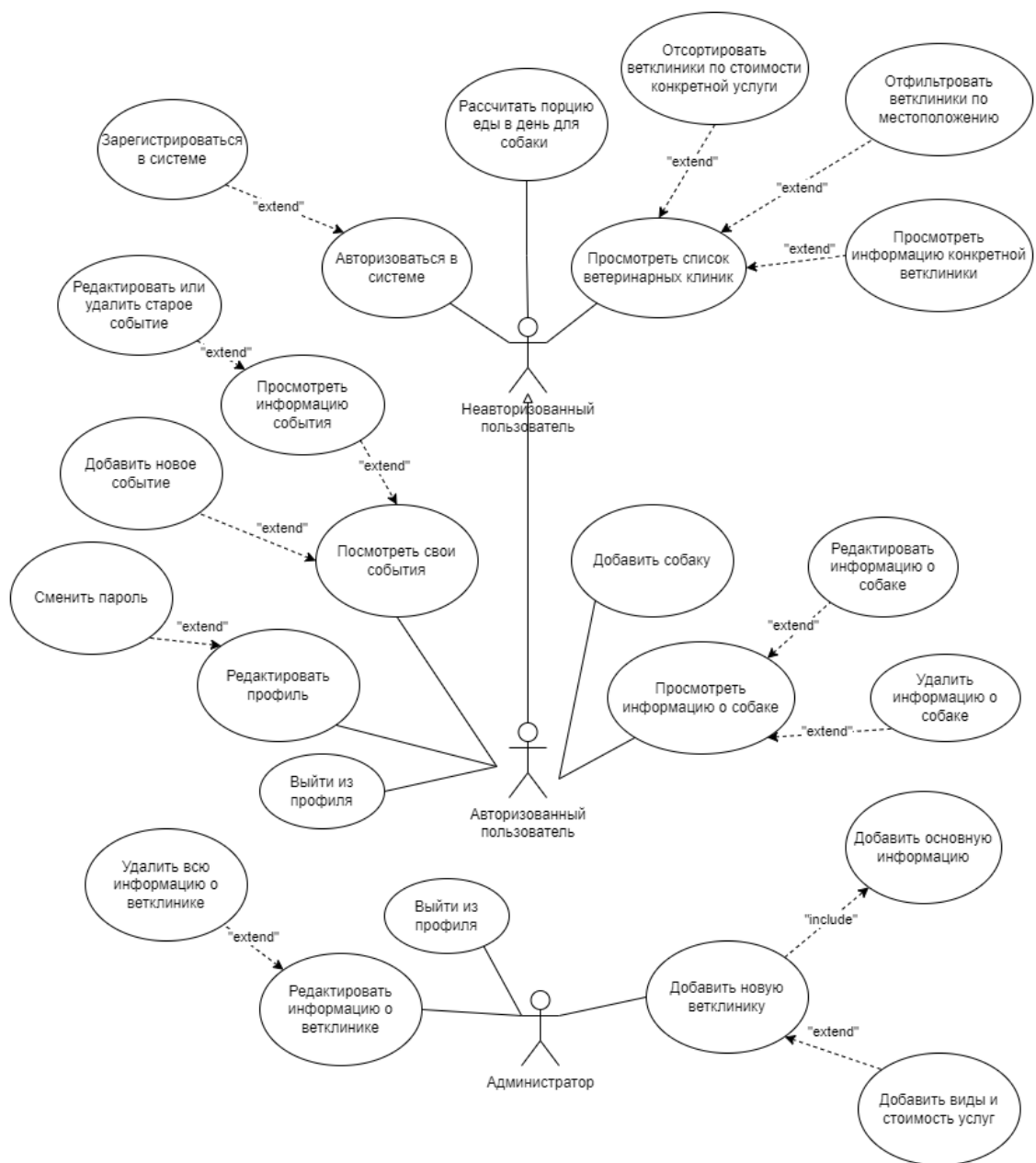


Рисунок 7 - Use – case диаграмма

### 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).



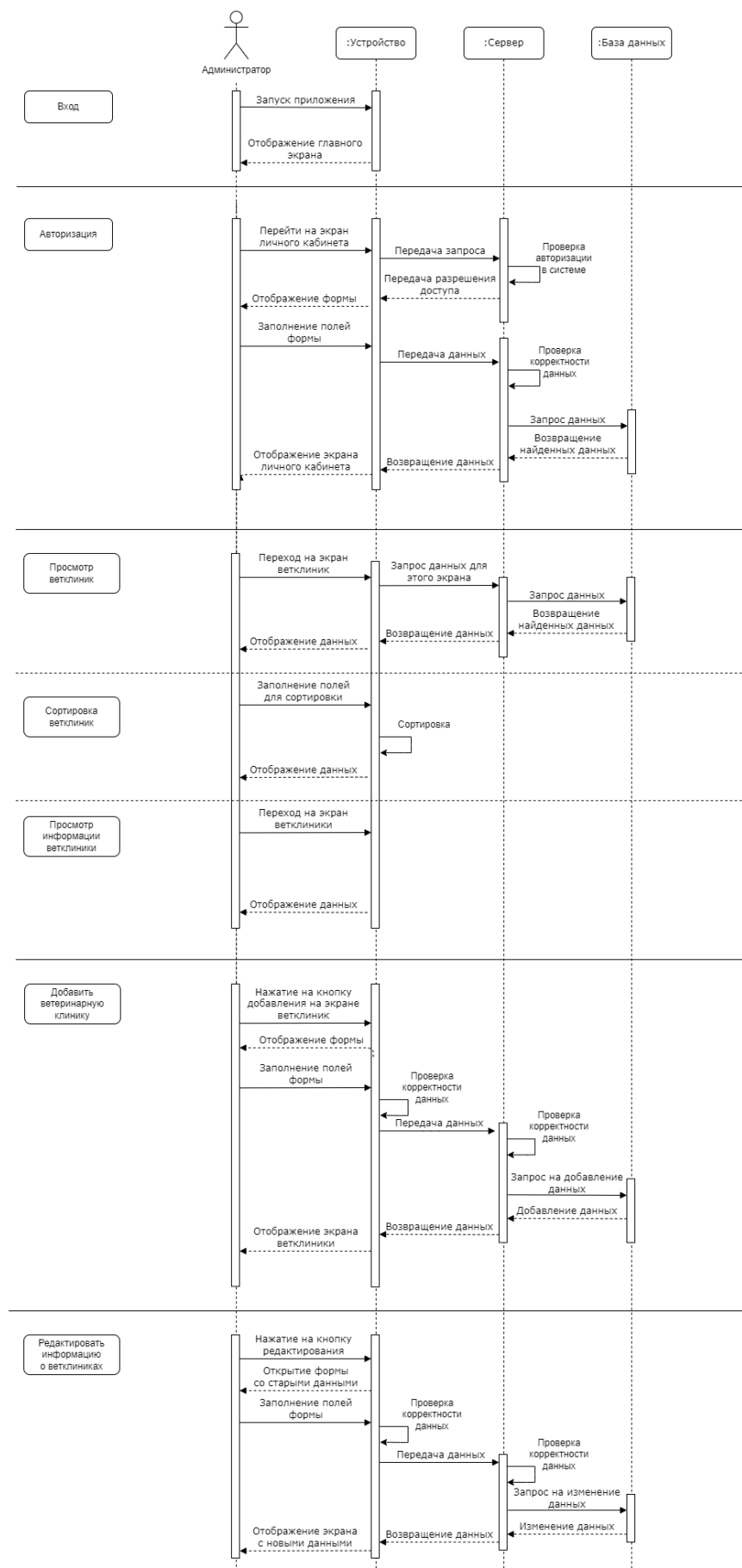


Рисунок 8 - Диаграмма последовательности для администратора

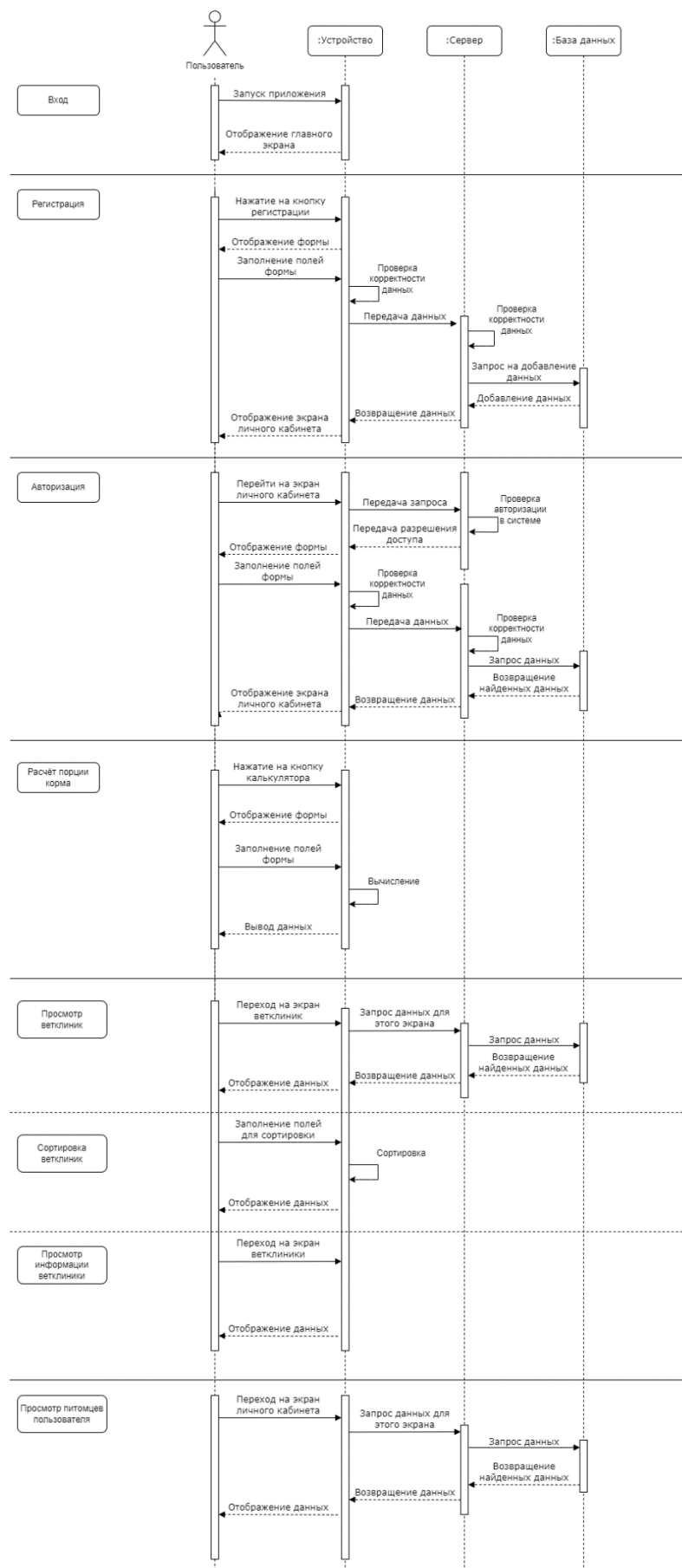


Рисунок 9 - Диаграмма последовательности для пользователя (1 часть)

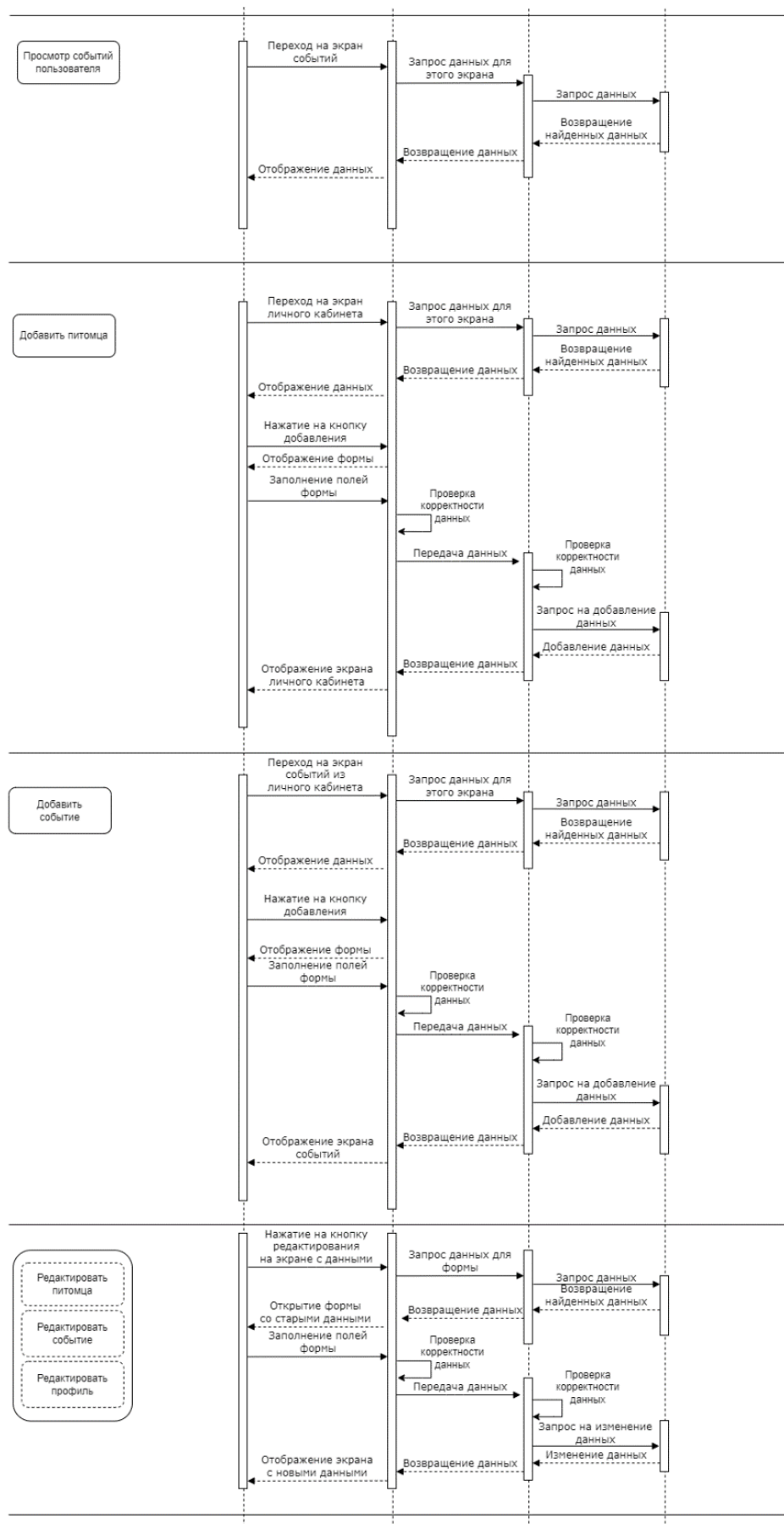


Рисунок 10 - Диаграмма последовательности для пользователя (2 часть)

### 2.3.4 Диаграмма активности

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

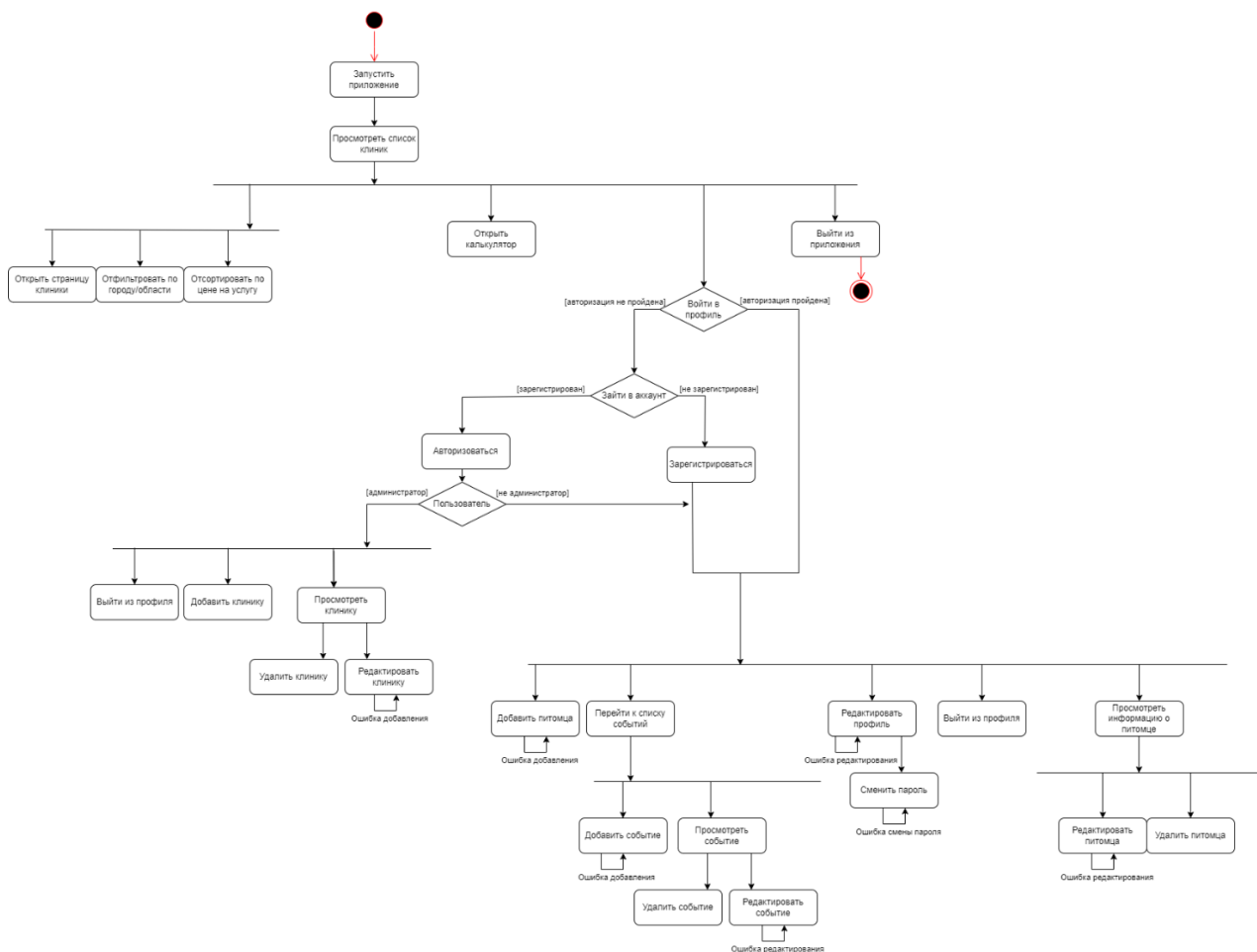


Рисунок 11 - Диаграмма активности

### 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).



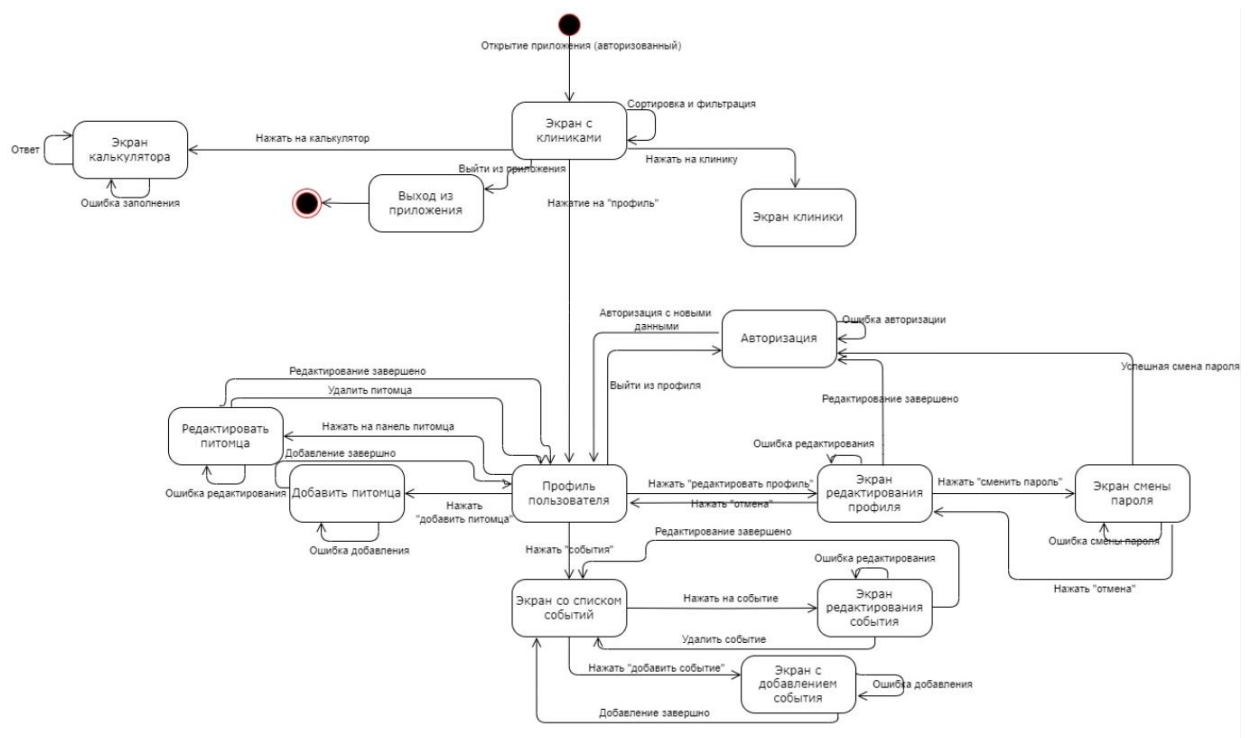


Рисунок 14 - Диаграмма состояний для авторизованного пользователя

### 2.3.6 Диаграмма классов

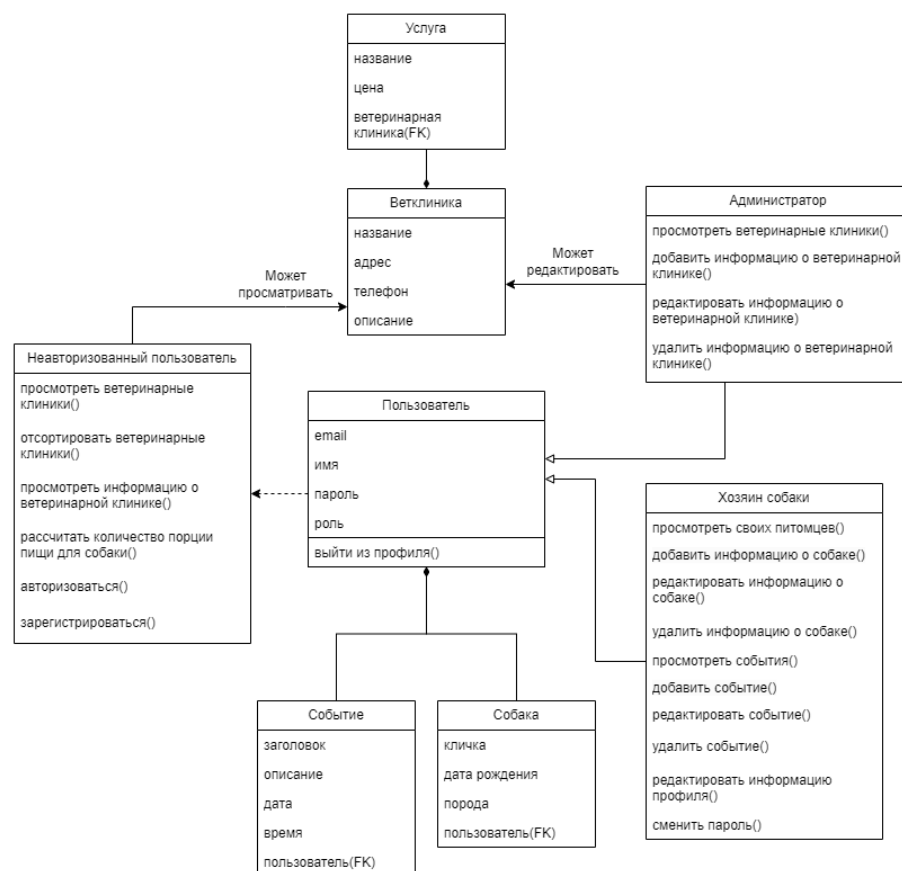


Рисунок 15 - Диаграмма классов

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).

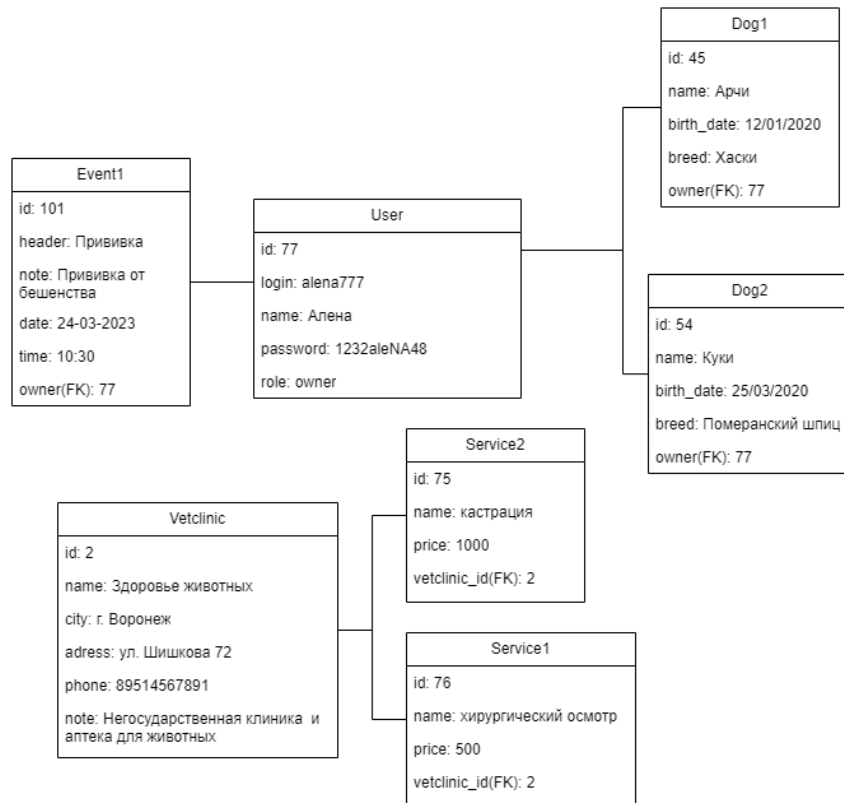


Рисунок 16 - Диаграмма объектов

### 2.3.8 Диаграмма развертывания

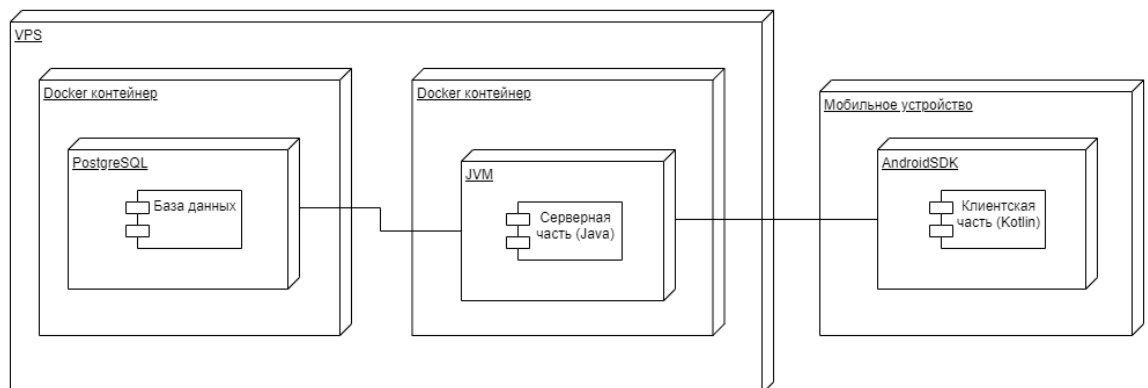


Рисунок 17 - Диаграмма развертывания

### 2.3.9 Диаграммы сотрудничества



Рисунок 18 - Диаграмма сотрудничества – Авторизация

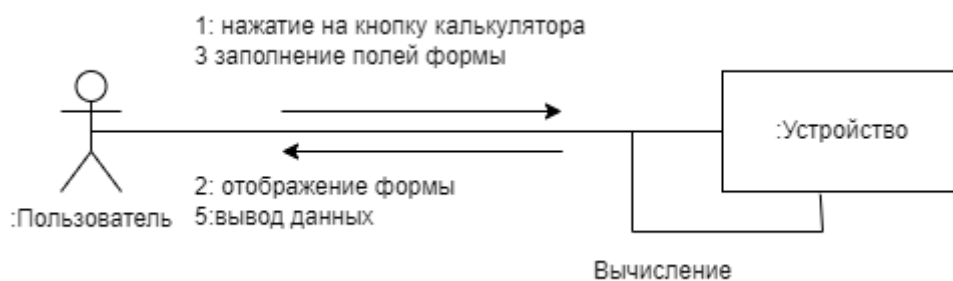


Рисунок 19 - Диаграмма сотрудничества – Расчет порции корма



Рисунок 20 - Диаграмма сотрудничества – Регистрация



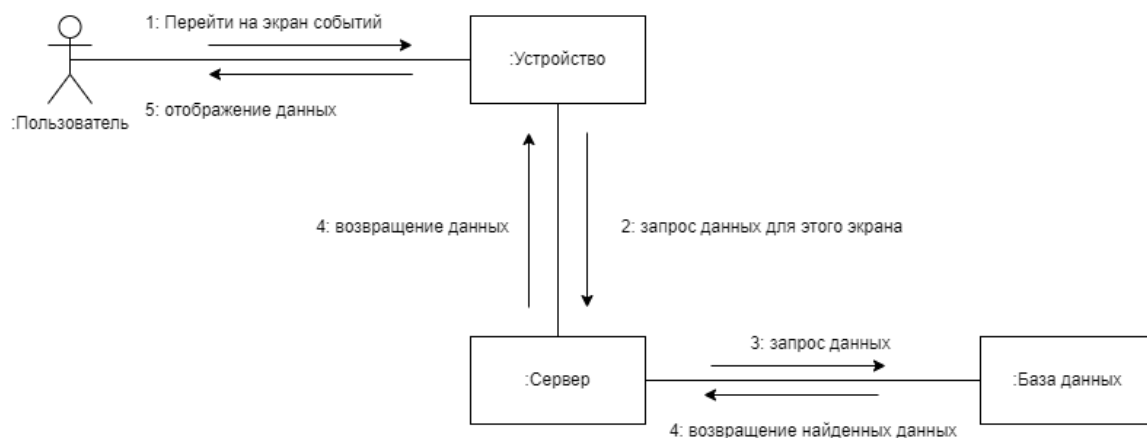


Рисунок 21 - Диаграмма сотрудничества – Просмотр ветеринарных клиник



Рисунок 22 - Диаграмма сотрудничества – Добавление питомца

#### Просмотр питомцев пользователя



Рисунок 23 - Диаграмма сотрудничества – Просмотр питомцев пользователя

### 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;
- Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Был выбран потому, его использование предлагает множество преимуществ, таких как изолированная среда, переносимость, масштабируемость, управление зависимостями и быстрота развёртывания [5].
- 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 [6];
- 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 [7];

- 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 [8];

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;
- Яндекс.метрика — это сервис веб-аналитики, предоставляемый компанией Яндекс. Он позволяет владельцам веб-сайтов и мобильных приложений отслеживать и анализировать данные о посетителях, поведении пользователей, эффективности

рекламных кампаний и другую информацию, связанную с веб-трафиком;

— 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 серверной части

### 3.2.1 База данных

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

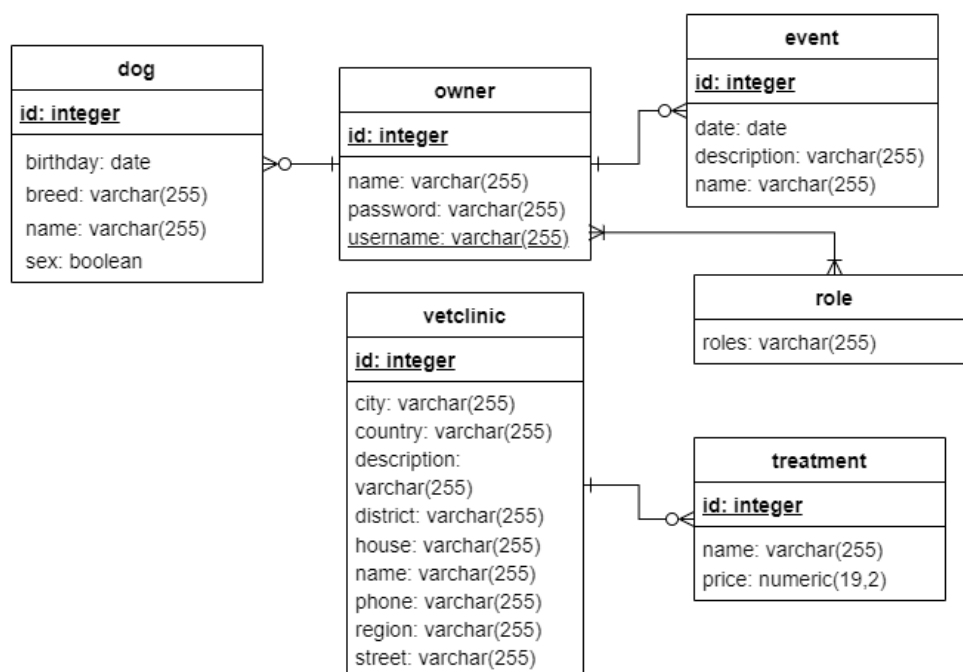


Рисунок 24 - ER – диаграмма базы данных

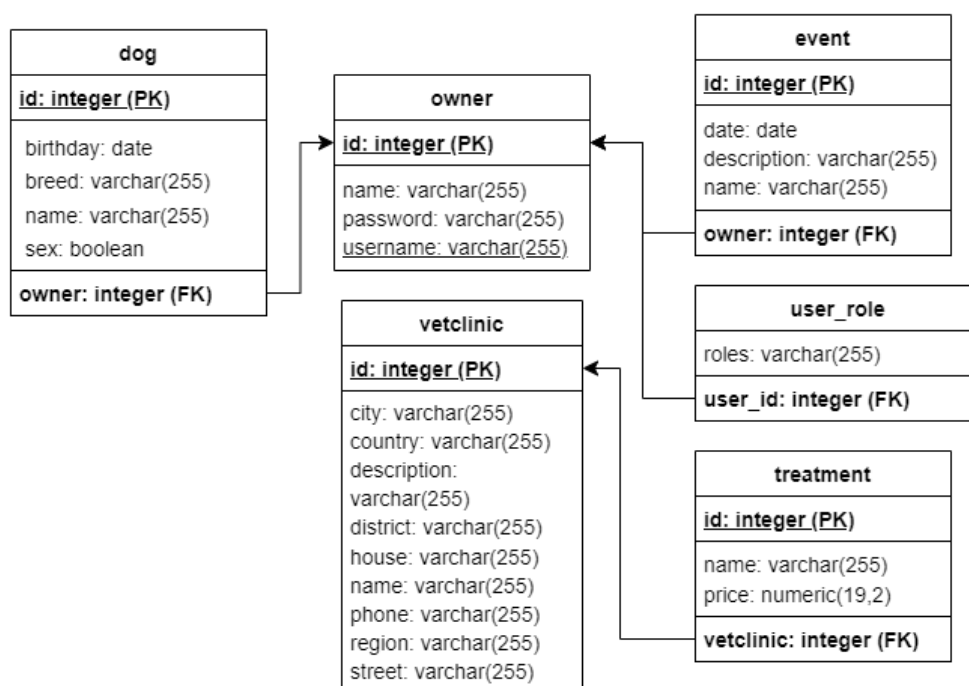


Рисунок 25 - Физическая диаграмма базы данных

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:

- Model – классы с аннотациями Entity и Repository, которые представляют данные, хранящиеся в базе данных и методы для их доступа и изменения;
- Controller – классы с аннотацией Service, которые содержат бизнес – логику и выполняют операции над данными, предоставляемыми моделью;
- View – классы с аннотацией Controller, которые обрабатывают HTTP – запросы, взаимодействуют с сервисами и готовят данные для отправки клиенту;



— Классы DTO и Mapper служат для передачи данных между View и Controller, а также для преобразования данных между моделью и DTO.

Паттерн 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.

Далее представлены описание групп классов разрабатываемого мобильного приложения и их диаграммы классов.

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. Диаграмма классов моделей предоставлена ниже (Рисунок 26).

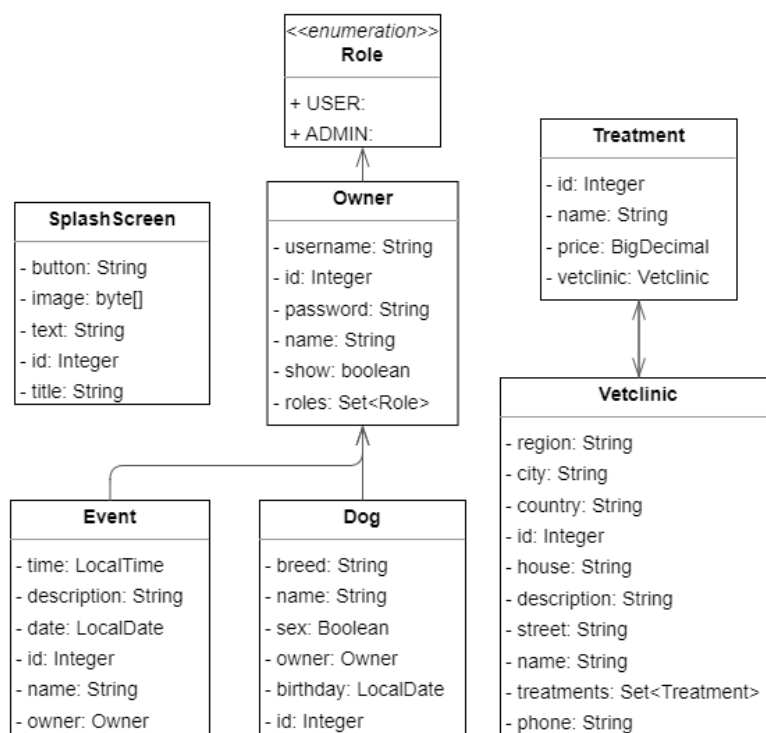


Рисунок 26 - Диаграмма классов моделей

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. Диаграмма классов DTO предоставлена ниже (Рисунок 27).

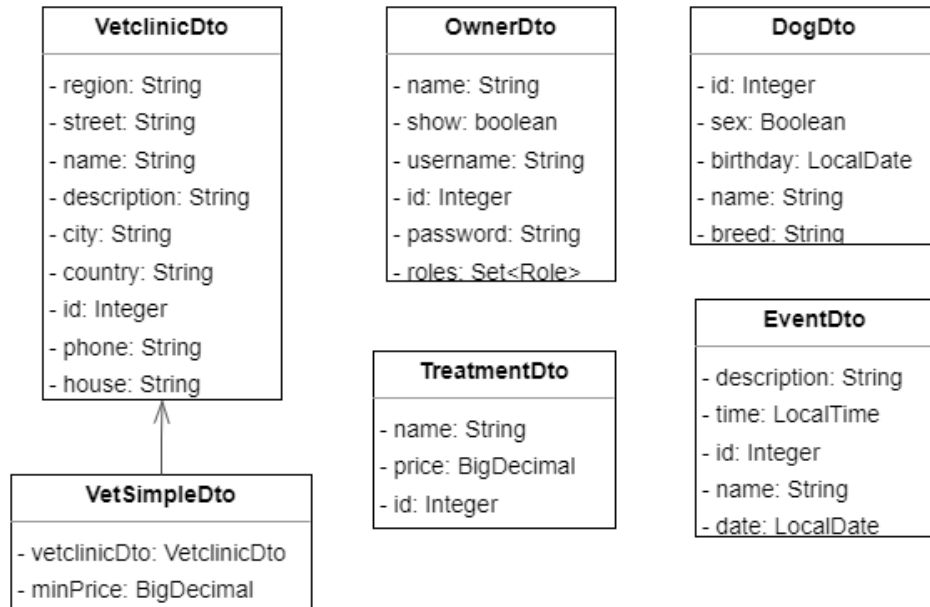


Рисунок 27 - Диаграмма классов DTO

Mapper – интерфейсы, которые используются для преобразования объектов между различными типами, другими словами, сериализаторы. В данном случае, они использовались для перехода экземпляров классов DTO в Entity и наоборот. Диаграмма классов сериализаторов предоставлена ниже (Рисунок 28).

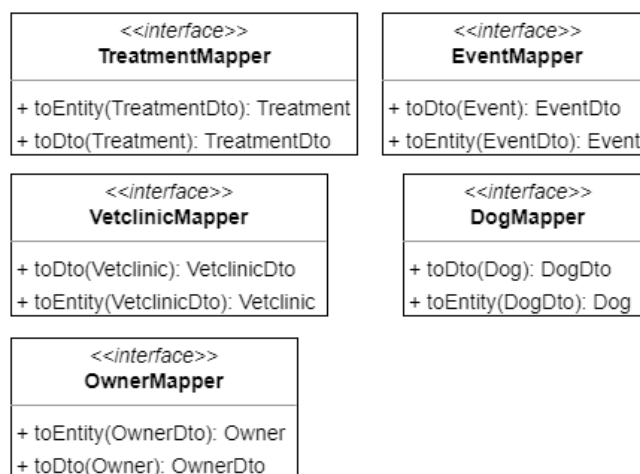


Рисунок 28 - Диаграмма классов сериализаторов

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. Диаграмма классов репозиториев предоставлена ниже (Рисунок 29).

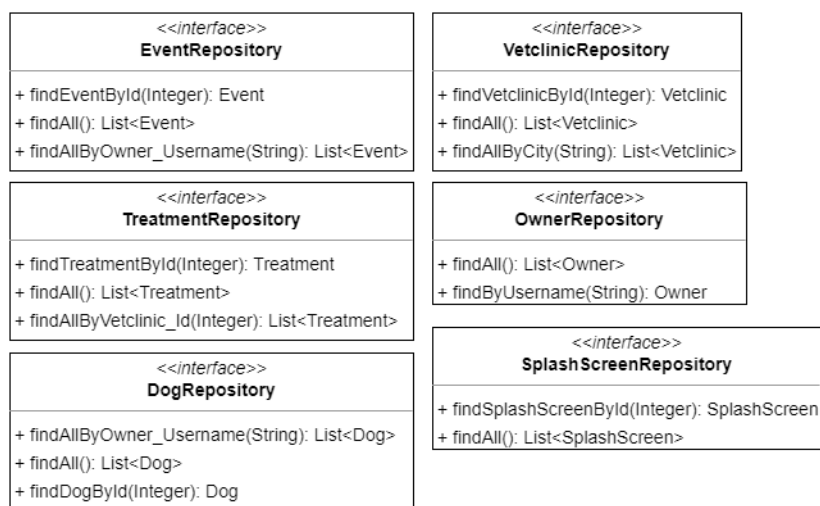


Рисунок 29 - Диаграмма классов репозиториев

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. Диаграмма классов сервисов предоставлена ниже (Рисунок 30).

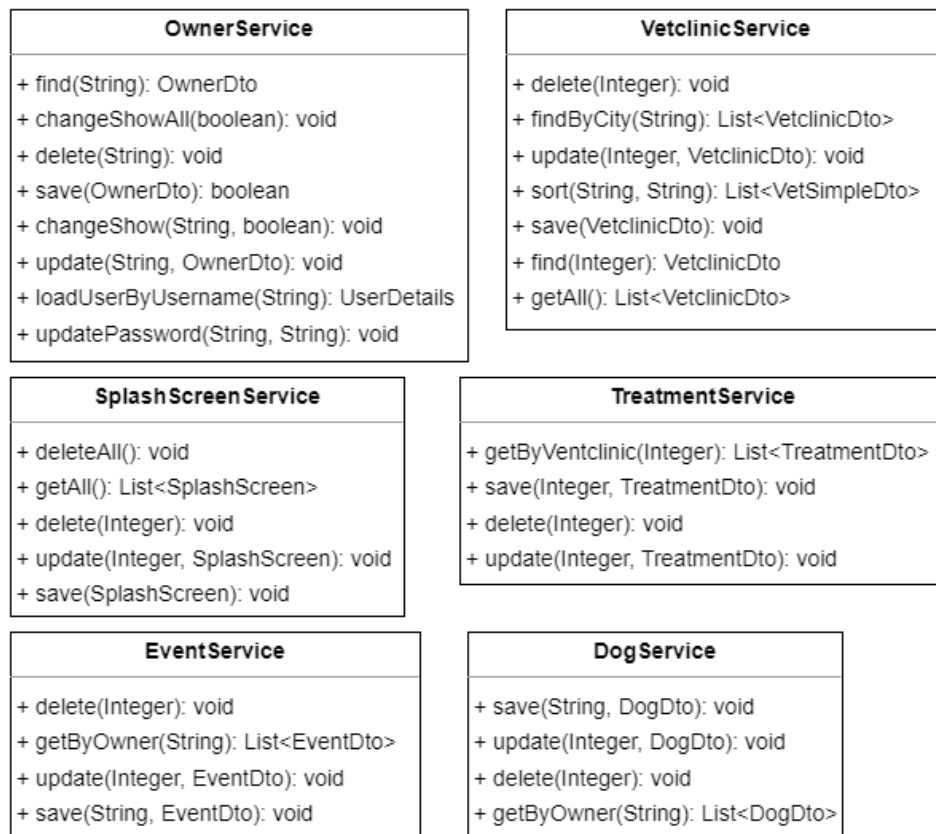


Рисунок 30 - Диаграмма классов сервисов

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. Диаграмма классов контроллеров предоставлена ниже (Рисунок 31).

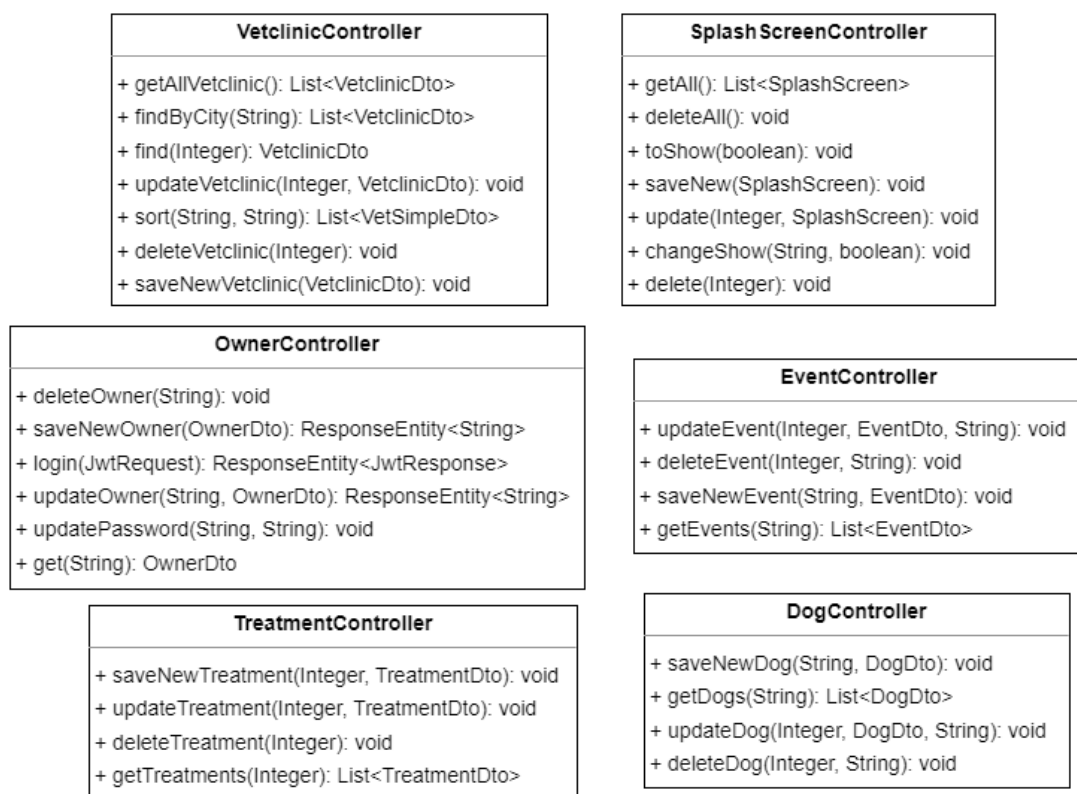


Рисунок 31 - Диаграмма классов контроллеров

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. Диаграмма классов конфигурации предоставлена ниже (Рисунок 31).

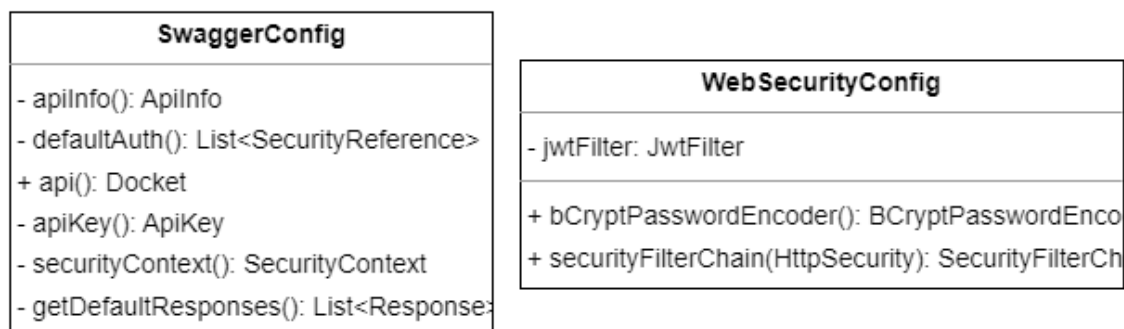


Рисунок 32 - Диаграмма классов конфигурации

Для реализации заявленных требований 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. Диаграмма классов реализации работы JWT токена представлены ниже (Рисунок 32);
- 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.

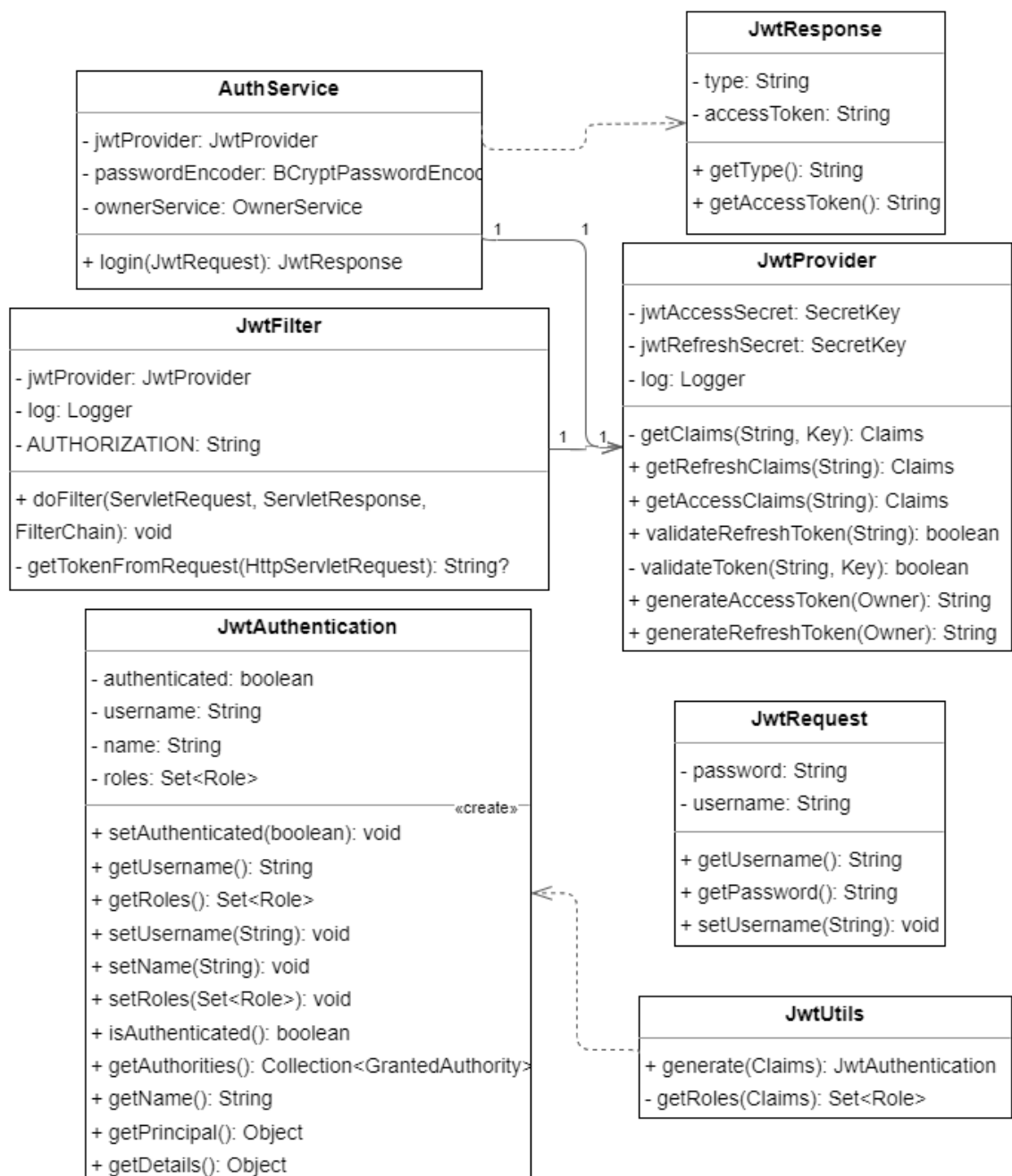


Рисунок 33 - Диаграммы классов, реализующих работу JWT токена

### 3.3 Реализация клиентской части

Для реализации клиентской части использовалась архитектура SAA (Single Activity Architecture) – подход к проектированию и разработке программного приложения, в котором вся функциональность приложения представлена в одной активности (все экраны и пользовательские интерфейсы приложения представлены внутри одного класса MainActivity). Такой способ разработки выбирается, когда проект имеет небольшой объем

функциональности или имеет линейный процесс работы без сложной навигации между экранами. Архитектура SAA упрощает код и уменьшает сложность приложения.

### 3.3.1 Панель навигации

Панель навигации состоит из трех элементов (Рисунок 34):

- «Клиники» – ведет к экрану списка ветеринарных клиник;
- «Калькулятор» – ведет к экрану калькулятора;
- «Профиль» – ведет к экрану авторизации (если пользователь не был авторизован) или экрану личного кабинета (если пользователь уже авторизован).



Рисунок 34 - Панель навигации

### 3.3.2 Экран списка ветеринарных клиник

На экране отображается список ветеринарных клиник в виде элементов с краткой информацией (название клиники и ее адрес), и два поля: для ввода услуги и для ввода города (для сортировки и фильтрации соответственно) (Рисунок 35).

При вводе названия услуги производится сортировка ветеринарных клиник по цене (от меньшего к большему) на нее. Если клиника не имеет услуги, название которой не содержит заданное значение, то она не будет отображаться. При сортировке, в элементах списка, содержащих информацию о клинике, также появляется список услуг, названия которых содержат введенное значение, и наименьшая среди них стоимость в рублях (Рисунок 36).

При вводе города производится фильтрация ветеринарных клиник по данному значению (Рисунок 37).



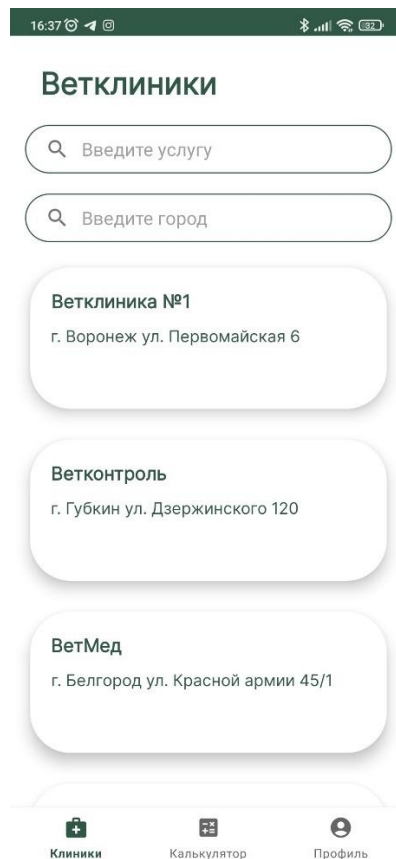


Рисунок 35 - Экран списка ветеринарных клиник

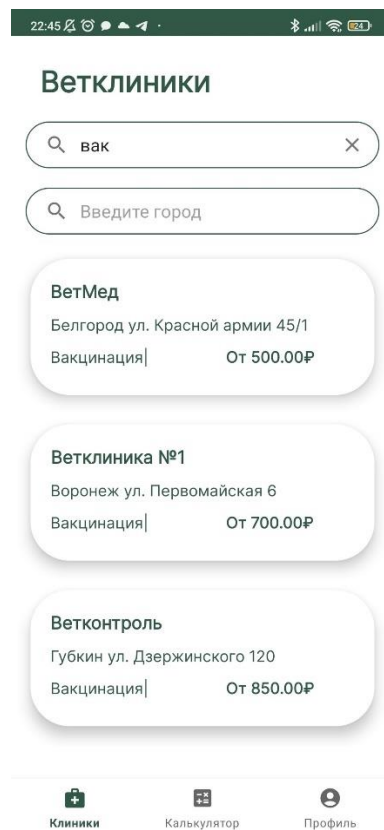


Рисунок 36 - Сортировка списка ветеринарных клиник по услугам

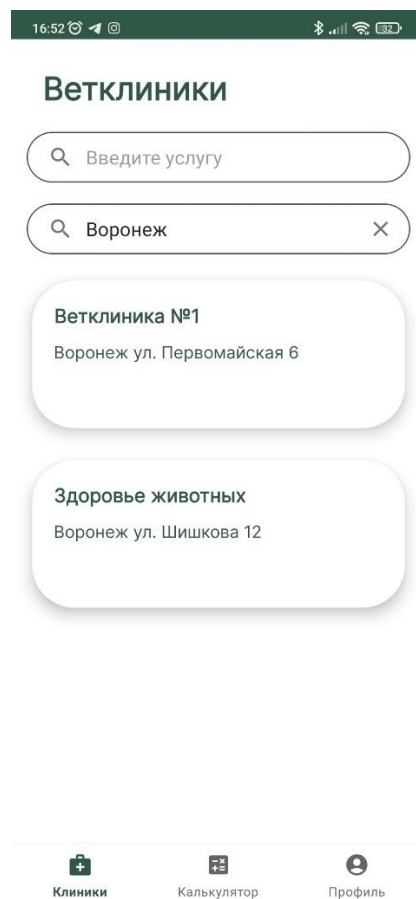


Рисунок 37 - Фильтрация списка по городу

При нажатии на элемент списка ветеринарной клиники пользователь перейдет на экран ветеринарной клиники с более подробной информацией о ней, а также полным списком услуг.

### 3.3.3 Экран ветеринарной клиники

Экран содержит главную информацию о ветеринарной клинике, а именно ее название, описание, адрес, телефон и список предлагаемых услуг, который содержит их название и цену (Рисунок 38).



Рисунок 38 - Экран ветеринарной клиники

### 3.3.4 Экран калькулятора

Экран содержит следующие элементы, необходимые для заполнения, чтобы провести расчет необходимого количества пищи собаки:

- выпадающий список диапазона возраста;
- выпадающий список уровней физической активности;
- поле ввода массы в килограммах.

После заполнения данных, пользователю требуется нажать на кнопку «Рассчитать», чтобы увидеть суточную норму корма в килограммах, а также необходимое количество содержания в ней белковой и растительной пищи в граммах (Рисунок 39).

17:27 100% 4G

## Калькулятор корма

Возраст  
Больше 12 месяцев

Уровень физической активности  
Обычная

4

РАССЧИТАТЬ

Суточная норма (кг): 0.145  
Белковой пищи (гр): 32  
Растительной пищи (гр): 10

Клиники Калькулятор Профиль

Рисунок 39 - Экран калькулятора

### 3.3.5 Экран авторизации

На экране авторизации (Рисунок 40) находятся два поля для ввода логина и пароля, а также две кнопки:

- «продолжить» - при нажатии, если авторизация успешна, выполняется переход на экран личного кабинета;
- «регистрация» - при нажатии выполняется переход на экран регистрации.

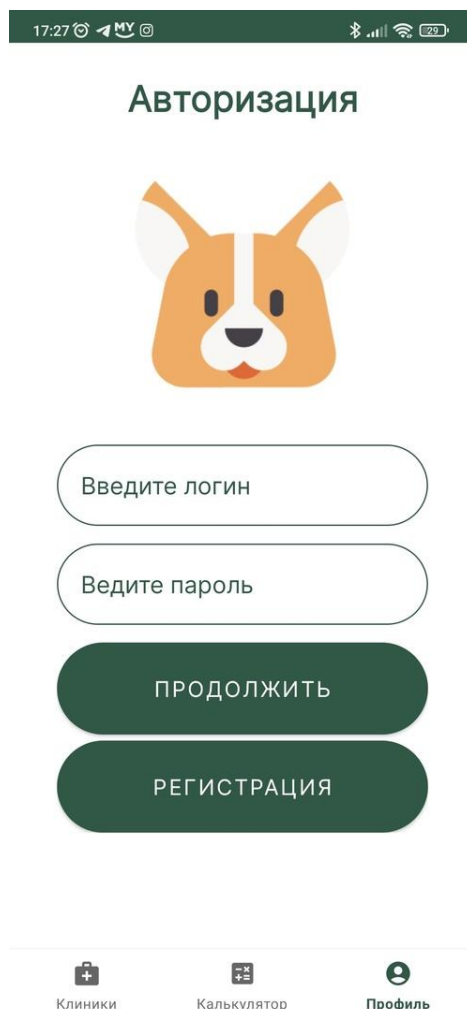


Рисунок 40 - Экран авторизации

### 3.3.6 Экран регистрации

На экране регистрации находятся поля для ввода логина, имени, которое будет отображаться в личном кабинете и пароля, который надо будет продублировать (Рисунок 41). При успешной регистрации кнопка «готово» выполнит переход на экран личного кабинета. Кнопка «назад» возвращает на экран авторизации.

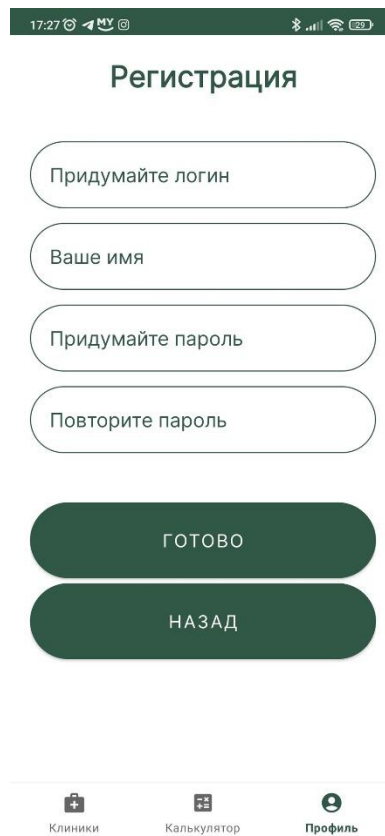


Рисунок 41 - Экран регистрации

### 3.3.7 Экран профиля

На экране профиля (Рисунок 42) находятся следующие элементы:

- имя пользователя;
- кнопка «добавить питомца» - выполняет переход на экран добавления питомца;
- список питомцев, который состоит из элементов, содержащих основную информацию о собаках (кличку, дату рождения и породу). Картинка идет одинаковой для всех по умолчанию. При нажатии на элемент выполняется переход на экран редактирования информации о питомце;
- кнопка «события» - выполняет переход на экран списка событий;
- кнопка «редактировать профиль» - выполняет переход на экран редактирования профиля;
- кнопка «выйти» - выполняет выход из учетной записи и переход на экран авторизации.

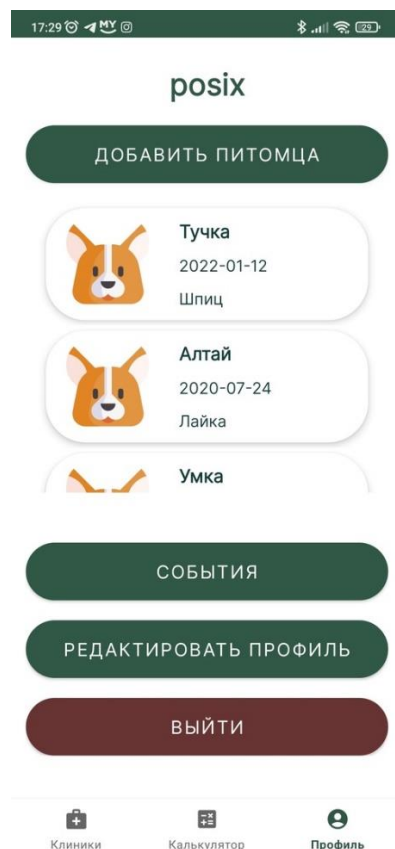


Рисунок 42 - Экран профиля

### 3.3.8 Экран добавления питомца

На экране находятся следующие поля для заполнения информации о собаке (Рисунок 43):

- кличка;
- дата рождения;
- порода (необязательное поле);
- пол (выпадающий список).

Кнопка «готово» сохраняет данные (если они корректны), а кнопка «назад» выполняет переход на экран личного кабинета.

17:28 100% 4G

### Добавление питомца

Тучка

12.01.2022

Шпиц

Пол  
Самка

ГОТОВО

НАЗАД

Клиники Калькулятор Профиль

Рисунок 43 - Экран добавления питомца

### 3.3.9 Экран редактирования питомца

Экран содержит те же поля, что и экран добавления питомца, в которых находятся старые данные (Рисунок 44). Также экран содержит следующие кнопки:

- «редактировать» - сохраняет информацию (если они корректны) и выполняет переход на экран личного кабинета;
- «удалить» - полностью удаляет данные о питомце;
- «назад» - выполняет переход на экран личного кабинета без сохранения данных.



17:28 100% 4G

## Редактирование питомца

Тучка

12.01.2022

Шпиц

Пол  
Самка

РЕДАКТИРОВАТЬ

УДАЛИТЬ

НАЗАД

Клиники Калькулятор Профиль

Рисунок 44 - Экран редактирования питомца

### 3.3.10 Экран списка событий

На экране находится список событий с основной информацией о них: название, дата, время и описание (Рисунок 45), и две кнопки:

- «+» - выполняет переход на экран добавления события;
- «←» - выполняет переход назад на экран личного кабинета.

При нажатии на элемент списка выполняется переход на экран редактирования данного события.

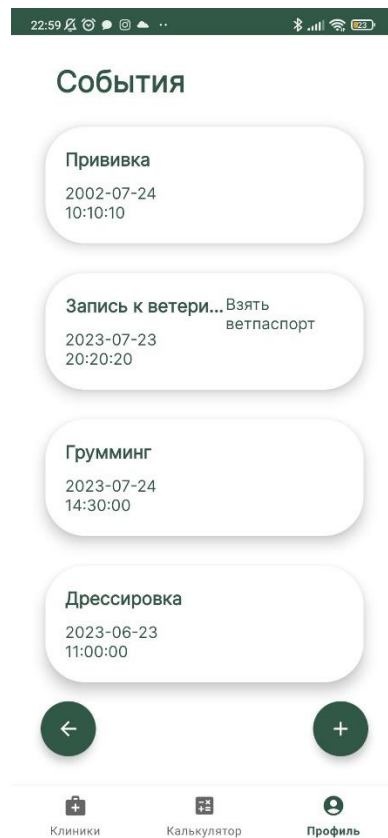


Рисунок 45 - Экран списка событий

### 3.3.11 Экран добавления события

На экране находятся поля, для заполнения информацией о событии (его названия, даты, времени и комментария (описания), а также две кнопки: «готово», которая сохраняет данные, если они корректны, и «назад», которая выполняет переход на экран списка событий (Рисунок 46).

Рисунок 46 - Экран добавления события

### 3.3.12 Экран редактирования события

На экране находятся те же поля, что и на экране добавления события (Рисунок 47), а также кнопки:

- «редактировать» - сохраняет информацию (если они корректны) и выполняет переход на экран списка событий;
- «удалить» - полностью удаляет данные о событии;
- «назад» - выполняет переход на экран списка событий без сохранения данных.

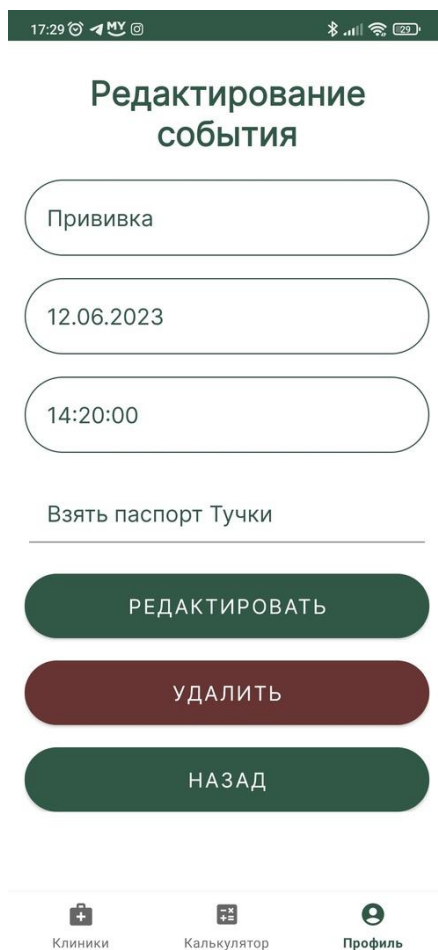


Рисунок 47 - Экран редактирования события

### 3.3.13 Экран редактирования профиля

На экране находятся два поля для ввода нового логина и нового имени пользователя (Рисунок 48), а также кнопки:

- «готово» - сохраняет данные (если они корректны) и выполняет переход на экран личного кабинета;
- «сменить пароль» - выполняет переход на экран смены пароля;
- «отмена» - выполняет переход на экран личного кабинета без сохранения данных.



Рисунок 48 - Экран редактирования данных пользователя

### 3.3.14 Экран смены пароля

Экран содержит два поля для ввода нового пароля и его дублирования, а также кнопки «готово» для сохранения данных (если они корректны) и перехода на экран личного кабинета и «отмена» для перехода на экран редактирования без сохранения данных (Рисунок 49).

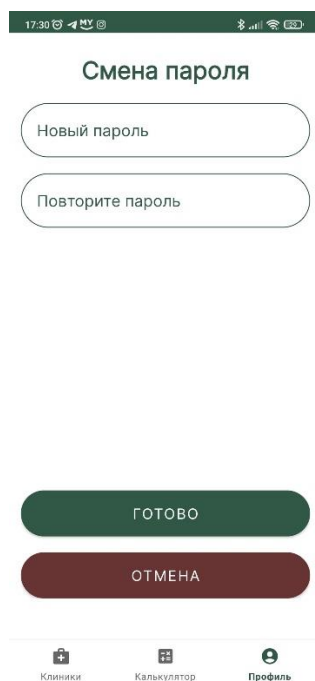


Рисунок 49 - Экран смены пароля

### 3.3.15 Экран личного кабинета администратора

Экран содержит список ветеринарных клиник (Рисунок 50), при нажатии на элемент которого выполняется переход на экран редактирования клиники, а также две кнопки:

- добавления (находится справа), которая выполняет переход на экран добавления новой ветеринарной клиники;
- выход (находится слева), которая выполняет выход из учетной записи.

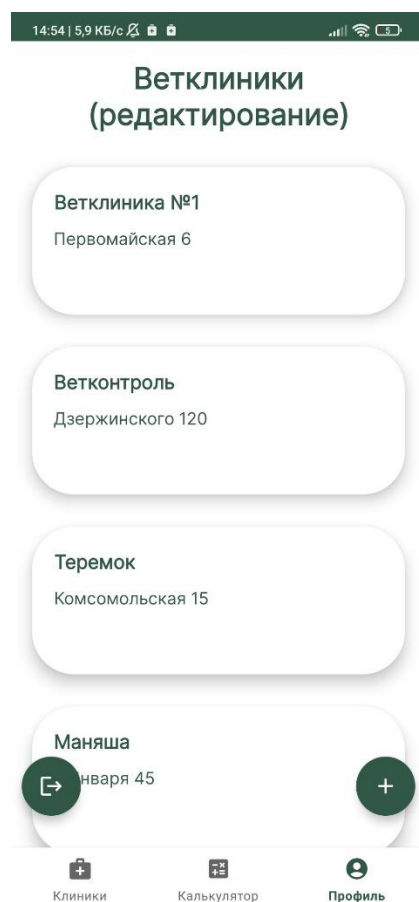


Рисунок 50 - Экран личного кабинета администратора

### 3.3.16 Экран добавления ветеринарной клиники

Экран доступен только для администратора и перейти к нему можно после нажатия на кнопку «+» из личного кабинета. На нем находятся поля для заполнения информации о ветеринарной клинике: названия, адреса, телефона, описания и предоставляемых ею услуг (Рисунок 51), а также кнопки:

- «готово» - сохраняет информацию (если они корректны) и выполняет переход на экран личного кабинета администратора со списком всех ветеринарных клиник;
- «назад» - выполняет переход на экран списка событий без сохранения данных.





Рисунок 51 - Экран добавления ветеринарной клиники





### 3.3.17 Экран редактирования ветеринарной клиники

Экран содержит те же поля, что и экран добавления ветеринарной клиники (Рисунок 52), а также кнопки:

- «редактировать» - сохраняет информацию (если они корректны) и выполняет переход на экран личного кабинета администратора со списком всех ветеринарных клиник;
- «удалить» - полностью удаляет данные о ветеринарной клинике;
- «назад» - выполняет переход на экран списка событий без сохранения данных.

23:13





Редактирование  
клиники

89553245821

Ветконтроль


Белгородская область, Губкин, ул. ↓

Вакцинация, 850.00; Внутрисуставная  
инъекция, 800.00;


РЕДАКТИРОВАТЬ

УДАЛИТЬ


НАЗАД



Клиники



Калькулятор



Профиль

Рисунок 52 - Экран редактирования ветеринарной клиники



## 4 Тестирование

После реализации серверной и клиентской частей, был произведен запланированный набор тестов:

- ручное дымовое тестирование на мобильном устройстве;
- интеграционные тесты для серверной части;
- unit – тесты для клиентской части;

### 4.1 Дымовое тестирование

Дымовое тестирование (smoke testing) — это вид тестирования программного обеспечения, который выполняется с целью быстрой проверки основных функций. Данными тестами была проверена работоспособность следующих основных сценариев:

- регистрация;
- авторизация;
- сортировка ветеринарных клиник по услуге;
- фильтрация ветеринарных клиник по городу;
- расчет оптимального количества пищи;
- добавление информации о питомце;
- редактирование информации о питомце;
- удаление информации о питомце;
- добавление информации о событии;
- редактирование информации о событии;
- удаление информации о событии;
- добавление информации о ветеринарной клинике;
- редактирование информации о ветеринарной клинике;
- удаление информации о ветеринарной клинике;
- редактирование данных пользователя;
- смена пароля.

Дымовое тестирование было проведено ручным способом на мобильном устройстве Redmi Note 8 Pro с операционной системой Android 11. Результаты

тестов приведены в Таблице 1:

Таблица 1 - Результаты дымового тестирования

<b>Сценарий</b>	<b>Результат</b>
Регистрация	Пройден
Авторизация	Пройден
Сортировка ветеринарных клиник по услуге	Пройден
Фильтрация ветеринарных клиник по городу	Пройден
Расчет оптимального количества пищи	Пройден
Добавление информации о питомце	Пройден
Редактирование информации о питомце	Пройден
Удаление информации о питомце	Пройден
Добавление информации о событии	Пройден
Редактирование информации о событии	Пройден
Удаление информации о событии	Пройден
Добавление информации о ветеринарной клинике	Пройден
Редактирование информации о ветеринарной клинике	Пройден
Удаление информации о ветеринарной клинике	Пройден
Редактирование данных пользователя	Пройден
Смена пароля	Пройден

## 4.2 Интеграционные тесты для серверной части

Интеграционные тесты (integration tests) — вид тестирования программного обеспечения, в котором проверяется взаимодействие различных компонентов или модулей в рамках серверной части приложения. Целью интеграционных тестов является проверка, что эти компоненты работают совместно правильно и взаимодействуют друг с другом в соответствии с ожидаемым поведением. Результаты интеграционного тестирования представлены на рисунках 53–58.

<b>VetclinicServiceTest: 6 total, 6 passed</b>		593 ms
		<a href="#">Collapse</a>   <a href="#">Expand</a>
<code>saveTest()</code>	passed	424 ms
<code>findTest()</code>	passed	25 ms
<code>updateTest()</code>	passed	55 ms
<code>getAllTest()</code>	passed	24 ms
<code>findByCityTest()</code>	passed	23 ms
<code>deleteTest()</code>	passed	42 ms

Рисунок 53 - Результат тестирования бизнес-логики для клиник

<b>TreatmentServiceTest: 3 total, 3 passed</b>		550 ms
		<a href="#">Collapse</a>   <a href="#">Expand</a>
<code>saveTest()</code>	passed	440 ms
<code>updateTest()</code>	passed	59 ms
<code>deleteTest()</code>	passed	51 ms

Рисунок 54 - Результат тестирования бизнес-логики для услуг

## SplashScreenServiceTest: 5 total, 5 passed

618 ms

[Collapse](#) | [Expand](#)

<code>saveTest()</code>	passed	455 ms
<code>deleteAllTest()</code>	passed	61 ms
<code>updateTest()</code>	passed	21 ms
<code>getAllTest()</code>	passed	28 ms
<code>deleteTest()</code>	passed	53 ms

Рисунок 55 - Результат тестирования бизнес-логики для сплеш скрина

## OwnerServiceTest: 8 total, 8 passed

846 ms

[Collapse](#) | [Expand](#)

<code>changeShow()</code>	passed	420 ms
<code>saveTest()</code>	passed	61 ms
<code>delete()</code>	passed	107 ms
<code>updateTest()</code>	passed	59 ms
<code>find()</code>	passed	22 ms
<code>loadUserByUsernameTest()</code>	passed	24 ms
<code>changeShowAll()</code>	passed	69 ms
<code>updatePassword()</code>	passed	84 ms

Рисунок 56 - Результат тестирования бизнес-логики для пользователя

## EventServiceTest: 4 total, 4 passed

683 ms

[Collapse](#) | [Expand](#)

<code>saveTest()</code>	passed	463 ms
<code>getByOwnerTest()</code>	passed	103 ms
<code>updateTest()</code>	passed	40 ms
<code>deleteTest()</code>	passed	77 ms

Рисунок 57 - Результат тестирования бизнес-логики для события

<b>DogServiceTest: 4 total, 4 passed</b>		683 ms
<a href="#">Collapse</a>   <a href="#">Expand</a>		
<code>saveTest()</code>	passed	485 ms
<code>getByOwnerTest()</code>	passed	68 ms
<code>updateTest()</code>	passed	50 ms
<code>deleteTest()</code>	passed	80 ms

Рисунок 58 - Результат тестирования бизнес-логики для собак

### 4.3 Unit – тесты для клиентской части

Unit-тесты (unit tests) — вид тестирования программного обеспечения, в котором отдельные компоненты или модули программы тестируются изолированно на предмет правильности их работы. Целью unit – тестов является проверка отдельных частей кода с целью убедиться, что они выполняют свою функциональность правильно. Результаты интеграционного тестирования представлены на рисунках 53–58.

<b>CalculatorTest: 1 total, 1 passed</b>		12 ms
<a href="#">Collapse</a>   <a href="#">Expand</a>		
<code>ru.vsu.cs.tp.paws.CalculatorTest</code>		12 ms
<code>testAns</code>	passed	12 ms

Generated by Android Studio on 05.06.2023, 16:16

Рисунок 59 - Результаты юнит тестов для калькулятора

<b>ApiTreatmentTest: 4 total, 4 passed</b>		441 ms
<a href="#">Collapse</a>   <a href="#">Expand</a>		
<code>ru.vsu.cs.tp.paws.ApiTreatmentTest</code>		441 ms
<code>testGetAllTreatment</code>	passed	381 ms
<code>testSaveTreatment</code>	passed	46 ms
<code>testDeleteTreatment</code>	passed	6 ms
<code>testUpdateTreatment</code>	passed	8 ms

Generated by Android Studio on 05.06.2023, 16:16

Рисунок 60 - Результаты юнит тестов для услуг

**ApiOwnerTest: 6 total, 6 passed**

405 ms

[Collapse](#) | [Expand](#)

ru.vsu.cs.tp.paws.ApiOwnerTest		405 ms
testUpdateUser	passed	353 ms
testFindById	passed	15 ms
testFindByLogin	passed	10 ms
testDeleteUser	passed	11 ms
testSaveNewUser	passed	5 ms
testUpdateUserPassword	passed	11 ms

Generated by Android Studio on 05.06.2023, 16:16

Рисунок 61 - Результаты юнит тестов для владельца

**ApiGlobalConfigTest: 2 total, 2 passed**

385 ms

[Collapse](#) | [Expand](#)

ru.vsu.cs.tp.paws.ApiGlobalConfigTest		385 ms
testGetAllScreens	passed	355 ms
testChangeFlag	passed	30 ms

Generated by Android Studio on 05.06.2023, 16:16

Рисунок 62 - Результаты юнит тестов для сплеш скрина

**ApiEventTest: 4 total, 4 passed**

404 ms

[Collapse](#) | [Expand](#)

ru.vsu.cs.tp.paws.ApiEventTest		404 ms
testDeleteEvent	passed	362 ms
testUpdateEvent	passed	24 ms
testSaveEvent	passed	6 ms
testGetEvents	passed	12 ms

Generated by Android Studio on 05.06.2023, 16:16

Рисунок 63 - Результаты юнит тестов для событий

<b>ApiDogTest: 4 total, 4 passed</b>		420 ms
		<a href="#">Collapse</a>   <a href="#">Expand</a>
ru.vsu.cs.tp.paws.ApiDogTest		420 ms
testUpdateDogs	passed	377 ms
testGetDogs	passed	25 ms
testDeleteDogs	passed	7 ms
testSaveNewDogs	passed	11 ms

Generated by Android Studio on 05.06.2023, 16:15

Рисунок 64 - Результаты юнит тестов для собак

<b>ApiAuthTest (1): 2 total, 2 passed</b>		408 ms
		<a href="#">Collapse</a>   <a href="#">Expand</a>
ru.vsu.cs.tp.paws.ApiAuthTest		408 ms
testGetToken	passed	372 ms
testLogin	passed	36 ms

Generated by Android Studio on 05.06.2023, 16:14

Рисунок 65 - Результаты юнит тестов для авторизации

<b>ApiClinicTest: 6 total, 6 passed</b>		430 ms
		<a href="#">Collapse</a>   <a href="#">Expand</a>
ru.vsu.cs.tp.paws.ApiClinicTest		430 ms
testUpdateClinic	passed	381 ms
testDeleteClinic	passed	7 ms
testSaveClinic	passed	7 ms
testSortByCityClinics	passed	15 ms
testGetClinics	passed	9 ms
testSortClinics	passed	11 ms

Generated by Android Studio on 05.06.2023, 16:15

Рисунок 66 - Результаты юнит тестов для клиник

## 5 Аналитика

Для сбора отчетов был использован сервис веб-аналитики Яндекс.Метрика, который позволяет отслеживать и анализировать информацию, связанную с веб-трафиком. В курсовой работе будут описана аналитика воронок – моделей, используемых для описания и анализа шагов, которые пользователи проходят на пути от начальной точки до целевого действия или конверсии.

Данные были собраны за недельный период.

### 5.1.1 Воронка авторизации

Сценарий воронки:

- запуск приложения;
- переход на экран профиля;
- авторизация.

Конверсия шагов представлена на Рисунке 53.

#### Конверсия шагов

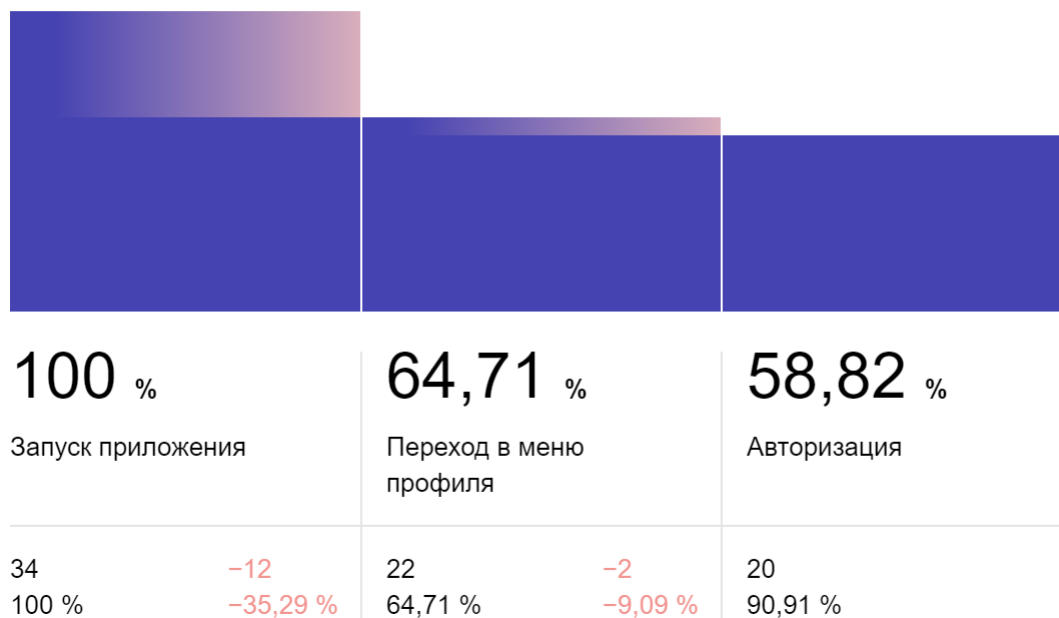


Рисунок 67 - Конверсия шагов воронки авторизации

### 5.1.2 Воронка регистрации

Сценарий воронки:



- запуск приложения;
- переход на экран профиля;
- регистрация.

Конверсия шагов представлена на Рисунке 54.

### Конверсия шагов

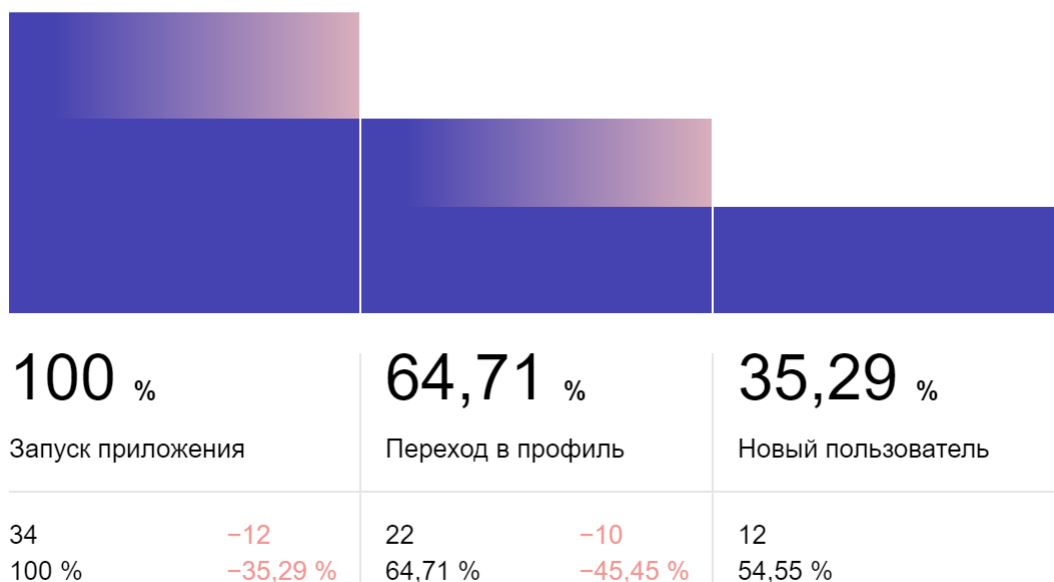


Рисунок 68 - Конверсия шагов воронки регистрации

### 5.1.3 Воронка калькулятора

Сценарий воронки:

- запуск приложения;
- открытие калькулятора.

Конверсия шагов представлена на Рисунке 55.

### Конверсия шагов

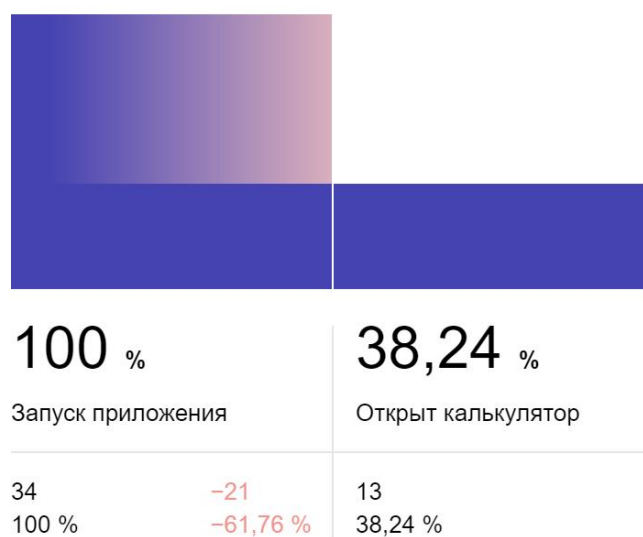


Рисунок 69 - Конверсия шагов воронки открытия калькулятора

### 5.1.4 Воронка просмотра клиник

Сценарий воронки:

- запуск приложения;
- переход на экран ветеринарных клиник.

Конверсия шагов представлена на Рисунке 55.

### Конверсия шагов

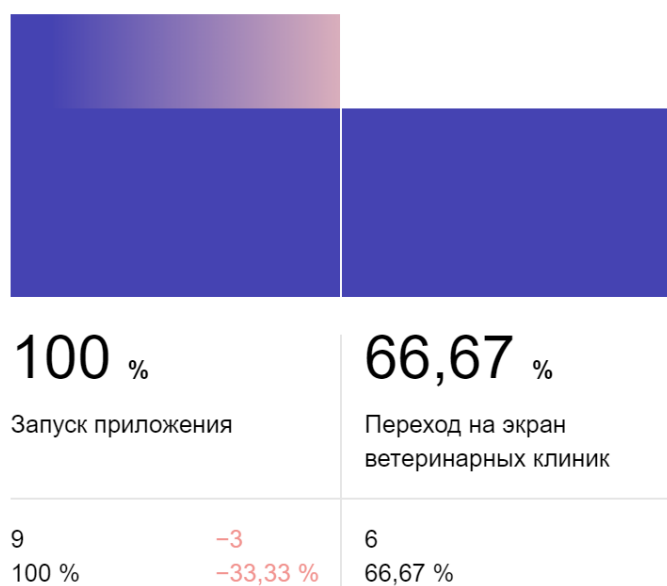


Рисунок 70 - Конверсия шагов воронки просмотра ветеринарных клиник

### 5.1.5 Воронка редактирования данных пользователя

Сценарий воронки:

- запуск приложения;
- переход на экран калькулятора.

Конверсия шагов представлена на Рисунке 55.

#### Конверсия шагов

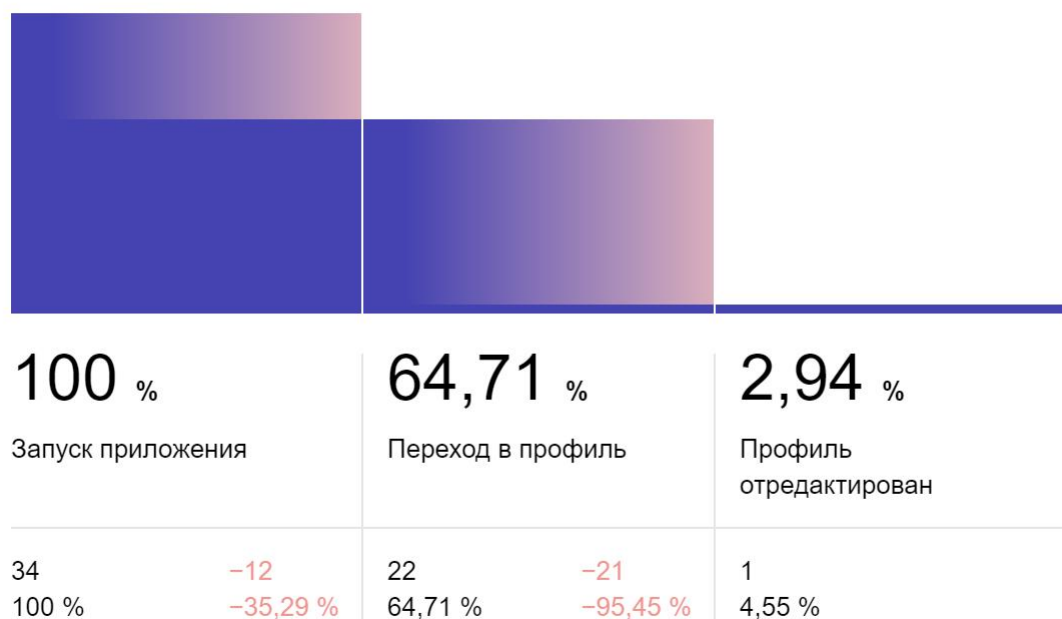


Рисунок 71 - Конверсия шагов воронки редактирования профиля

### 5.1.6 Воронка смены пароля

Сценарий воронки:

- запуск приложения;
- переход в профиль;
- смена пароля.

Конверсия шагов представлена на Рисунке 55.

### Конверсия шагов

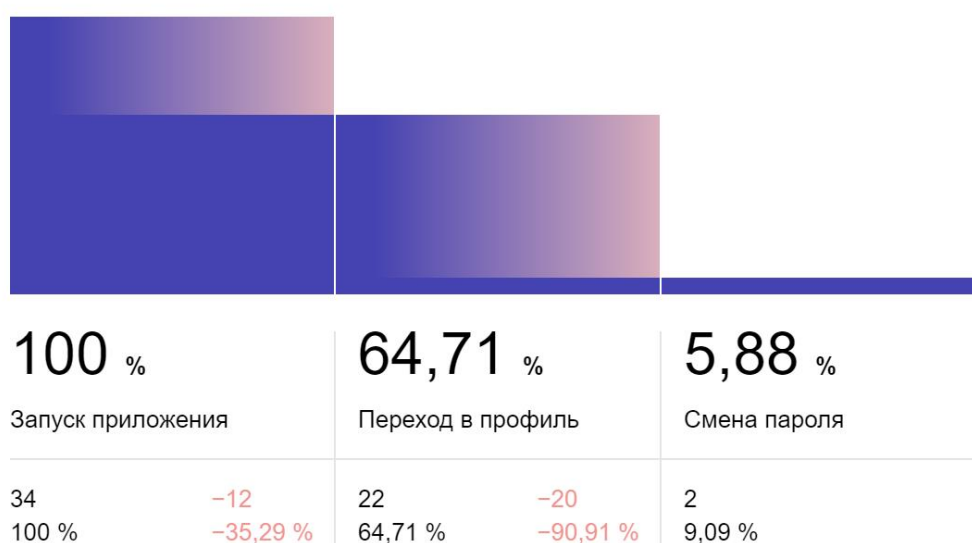


Рисунок 72 - Конверсия шагов воронки смены пароля

### 5.1.7 Воронка добавления информации о собаке

Сценарий воронки:

- запуск приложения;
- переход в профиль;
- добавление собаки.

Конверсия шагов представлена на Рисунке 55.

### Конверсия шагов

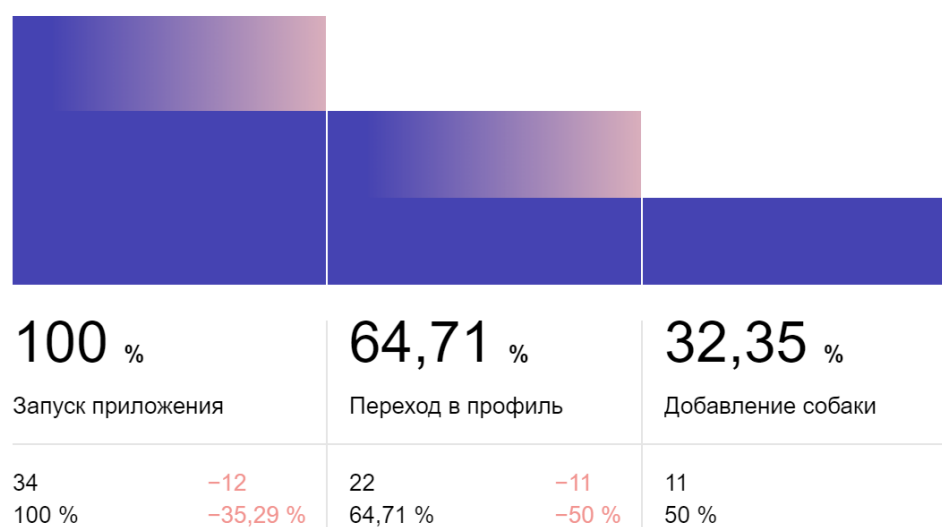


Рисунок 73 - Конверсия шагов воронки добавления собаки

### 5.1.8 Воронка добавления информации о событии

Сценарий воронки:

- запуск приложения;
- переход в профиль;
- добавление события.

Конверсия шагов представлена на Рисунке 55.

#### Конверсия шагов

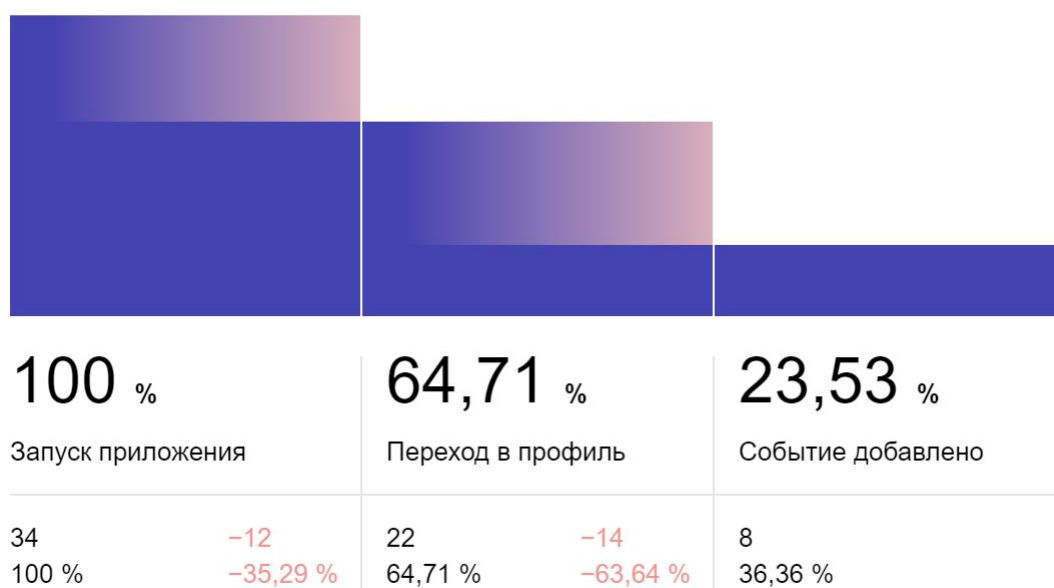


Рисунок 74 - Конверсия шагов воронки добавления события

## Заключение

В ходе выполнения данного курсового проекта был выполнен анализ предметной области и проведен обзор аналогов разрабатываемого приложения.

Для разработки приложения были созданы макеты интерфейса, выбраны средства реализации. Для описания создаваемой системы были построены UML диаграммы.

В процессе разработки было реализовано следующее:

- 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.

Серверная часть приложения и база данных были размещены на удаленном хостинге с помощью Docker – контейнера. Были подключены Яндекс.метрика для сбора данных и составления аналитики и Swagger для ведения документации.

Разработанное приложение удовлетворяет поставленным требованиям.

Все поставленные задачи были выполнены.

В качестве дальнейшего развития проекта рассматриваются следующие

усовершенствования:

- push – уведомления для напоминания о событии;
- добавление карты выгула питомца;
- возможность создавать и просматривать объявления заводчиков о продаже собак.

### Список используемых источников

1. Расчет количества корма для собак // Сайт vashipitomcy.ru URL: [https://vashipitomcy.ru/sobaki/soderzhanie\\_i\\_ukhod\\_29/raschet-kolichestva-korma-dlya-sobak-tablica/](https://vashipitomcy.ru/sobaki/soderzhanie_i_ukhod_29/raschet-kolichestva-korma-dlya-sobak-tablica/) (дата обращения: 23.03.2023).
2. 11pets: pets care // Сайт 11pets.com URL: <https://www.11pets.com/en/petcare/tutorials-for-11pets-pet-care> (дата обращения: 16.03.2023).
3. Doghealth // Сайт doghealthapp.it URL: <https://doghealthapp.it/> (дата обращения: 16.03.2023).
4. Pet Diary // Сайт sensortronic.net URL: <https://sensortronic.net/applications/pet-diary/> (дата обращения: 16.03.2023).
5. Документация к Docker // Сайт docker.com URL: <https://docs.docker.com/> (дата обращения: 22.03.2023).
6. Документация Kotlin кода // Сайт kotlinlang.ru URL: <https://kotlinlang.ru/docs/reference/kotlin-doc.html> (дата обращения: 22.03.2023).
7. Документация к PostgreSQL // Сайт postgrespro.ru URL: <https://postgrespro.ru/docs/postgresql/> (Дата обращения: 23.03.2023).
8. Справочная документация Spring Boot // Сайт docs.spring.io URL: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/> (дата обращения: 23.03.2023).
9. Документация к Git // Сайт git-scm.com URL: <https://git-scm.com/doc/> (Дата обращения: 24.03.2023).