

**МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Приложение для аренды инвентаря на спортивной базе “Renty”

Курсовой проект

09.03.02 Информационные системы и технологии

Программная инженерия в информационных системах

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

Обучающийся \_\_\_\_\_ С.А. Волченко, 3 курс, д/о

Обучающийся \_\_\_\_\_ В.Ю. Шафоростов, 3 курс, д/о

Обучающийся \_\_\_\_\_ А.Г. Линкина, 3 курс, д/о

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

Руководитель \_\_\_\_\_ В.С. Тарасов, ст. преподаватель \_\_.\_\_.20\_\_.

Воронеж 2023

# СОДЕРЖАНИЕ

СОДЕРЖАНИЕ .....	2
ВВЕДЕНИЕ.....	4
1 Постановка задачи .....	6
1.1 Цель .....	6
1.2 Требования к системе .....	6
1.3 Задачи .....	6
2 Анализ предметной области .....	8
2.1 Анализ рынка приложений по прокату спортивного инвентаря .....	8
2.2 Анализ существующих решений.....	9
2.2.1 Прокаты.Ru.....	9
2.2.2 Getski .....	11
2.2.3 Swissrent .....	14
2.2.4 Ski-iD.....	17
3 Реализация .....	21
3.1 Средства реализации .....	21
3.2 IDEF0 .....	23
3.3 Диаграмма прецедентов .....	24
3.4 Диаграмма состояний .....	26
3.5 Диаграмма последовательностей .....	26
3.6 Диаграмма деятельности.....	28
3.7 Методы разработки.....	29
3.7.1 Kanban .....	29
3.7.2 Contract-first подход.....	29

3.7.3	Git-flow .....	30
3.7.4	CI/CD .....	30
3.8	Реализация серверной части приложения .....	31
3.8.1	Схема базы данных .....	31
3.8.2	Архитектура .....	33
3.8.3	Классы сущностей .....	33
3.8.4	Классы репозиториев .....	33
3.8.5	Классы контроллеров .....	34
3.8.6	Классы сервисов .....	35
3.8.7	Бизнес-логика .....	36
3.8.8	Аутентификация .....	37
3.9	Реализация клиентской части приложения .....	38
3.9.1	Архитектура .....	38
3.9.2	Экраны .....	40
ЗАКЛЮЧЕНИЕ .....		48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....		49

## ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

1. База данных – это организованная коллекция данных, хранящихся в электронном виде, которые могут быть получены, обработаны, сортированы и связаны друг с другом при помощи соответствующих инструментов.
2. Сервер (back-end) – программа, выполняющая функции обработки запросов от клиента и предоставления данных в ответ на эти запросы. Например, сервер может обрабатывать базу данных и генерировать ответы на запросы клиентов, используя бэкэнд-фреймворки или другие технологии.
3. Клиент (front-end) – это пользовательский интерфейс, непосредственно взаимодействующий с пользователем. Клиент может иметь средства для взаимодействия с сервером, например, отправлять запросы и получать ответы через REST API.
4. REST API – это интерфейс программирования приложений, который предоставляет клиентским приложениям доступ к функционалу сервера. REST API может быть использован для получения и отправки данных, выполнения операций и взаимодействия с другими приложениями или сервисами. Он может использовать различные методы запросов, включая GET, POST, PUT и DELETE.
5. Фреймворк - это набор предопределенных классов и функций, которые можно использовать для разработки программного обеспечения. Фреймворки упрощают процесс разработки программного обеспечения, предоставляя стандартный набор функций и средств для решения определенных задач. Они включают в себя инструментарий для разработки, тестирования и развертывания приложений.

## ВВЕДЕНИЕ

Спорт – это жизнь. Постоянные тренировки улучшают настроение, познавательную способность, ментальное здоровье, иммунитет, самочувствие человека. Этот список можно продолжать долго.

Существуют виды спорта, не требующие большого количества снаряжения, например:

- бег;
- фитнес;
- йога;
- игры с мячом (футбол, баскетбол, волейбол и так далее).

Но есть и такие, занятия которыми невозможны без специализированного инвентаря. К данным видам спорта можно отнести:

- настольный теннис;
- большой теннис;
- бадминтон;
- хоккей;
- гольф;
- горные лыжи, сноуборд.

И что делать человеку, который только начинает знакомиться с “затратным” видом спорта? Или тому, кто не может тренироваться часто? А если нет возможности привезти свою экипировку до места занятий? Что же делать этим людям? Есть решение – организовать базу для проката спортивного инвентаря.

В рамках курсовой работы было решено разработать мобильное приложение, которым смогут пользоваться как сотрудники, так и клиенты базы проката спортивного инвентаря. Оно позволит искать товар среди ассортимента базы, а также автоматизировать процессы проката инвентаря и оплаты аренды.

# **1 Постановка задачи**

## **1.1 Цель**

Целью курсового проекта является создание мобильного приложения, выполняющего функции по аренде спортивного инвентаря.

## **1.2 Требования к системе**

### **6. Автоматизация поиска товара среди ассортимента базы:**

- просмотр списка товаров, находящихся на базе;
- просмотр информации о конкретном товаре.

### **7. Автоматизация процесса проката инвентаря:**

- добавление товара в корзину;
- бронирование товаров;
- продление срока аренды товаров;
- уведомление об освобождении занятого товара;
- учет сроков аренды товара.

### **8. Автоматизация процесса оплаты аренды:**

- формирование счета для оплаты аренды;
- учет штрафов за просрочку аренды;
- генерация QR-кода для оплаты аренды.

### **9. Изменение ассортимента базы:**

- добавление и удаление товара;
- добавление и удаление категорий товаров;
- редактирование количества и размеров товаров.

## **1.3 Задачи**

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ существующих решений с целью определения достоинств и недостатков систем, реализующих схожие требования;
- спроектировать приложение с учётом данных, полученных в результате выполнения анализа;
- реализовать приложение, соответствующее требованиям, представленным выше;
- описать результат разработки.

Для реализации приложения необходимо выполнить следующие подзадачи:

- разработать back-end часть приложения, развернуть её на удалённом сервере;
- разработать front-end часть приложения, развернуть её на удалённом сервере;
- создать связь между front-end и back-end с помощью REST API;
- разработать базу данных, развернуть её на удалённом сервере.

## **2 Анализ предметной области**

### **2.1 Анализ рынка приложений по прокату спортивного инвентаря**

Сегодня, когда спорт стал неотъемлемой частью нашей жизни, все больше людей начинают заниматься фитнесом и активными видами спорта. Однако, не каждый может позволить себе купить дорогостоящее спортивное оборудование или инвентарь. Поэтому рынок проката спортивного инвентаря становится очень востребованным.

Согласно исследованию, проведенному Apparel Market Research, мировой рынок спортивного оборудования и одежды оценивался в 340.6 млрд долларов в 2020 году и, по прогнозам, достигнет 930.5 млрд долларов к 2031 году, а среднегодовой темп роста составит 8.3% с 2022 по 2031 год [1]. Здоровый образ жизни является текущим трендом, и ожидается, что этот тренд останется актуальным еще долгое время. Сегодня люди стали очень хорошо осведомлены о здоровье и фитнесе и начали инвестировать в различные виды деятельности, такие как йога, тренажерные залы и многие другие формы тренировок, чтобы оставаться в тонусе. Ожидается, что это будет способствовать росту рынка спортивного инвентаря.

Сегодня существует множество приложений, которые помогают людям арендовать спортивный инвентарь в любой точке мира. Они предлагают огромный выбор товаров для различных видов спорта и позволяют пользователям выбирать наиболее удобный способ получения и возврата инвентаря.

Рынок приложений для проката спортивного инвентаря очень разнообразен и востребован. Он предлагает огромный выбор товаров для различных видов спорта и позволяет пользователям выбирать наиболее удобный способ получения и возврата инвентаря. С развитием технологий и увеличением числа людей, занимающихся спортом, этот рынок будет только расти и развиваться в будущем.



## 2.2 Анализ существующих решений

Прежде всего, необходимо провести анализ уже существующих проектов, рассмотреть их преимущества и недостатки. На этой основе можно будет определить, что нужно учесть при разработке приложения.

### 2.2.1 Прокаты.Ru

Приложение, которое объединило в себе точки прокатов горнолыжного оборудования. Обладает простым и понятным интерфейсом (рисунки 1 - 4).

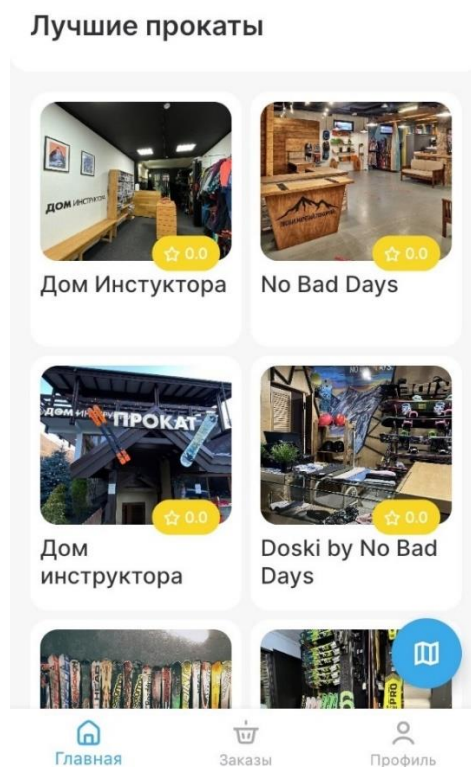


Рисунок 1 - Каталог в приложении Прокаты.Ru

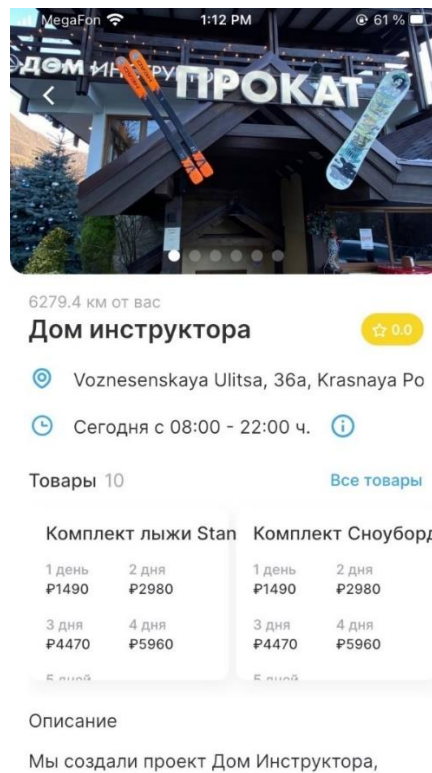


Рисунок 2 - Страница с товаром на определенной базе в приложении Прокаты.Ру

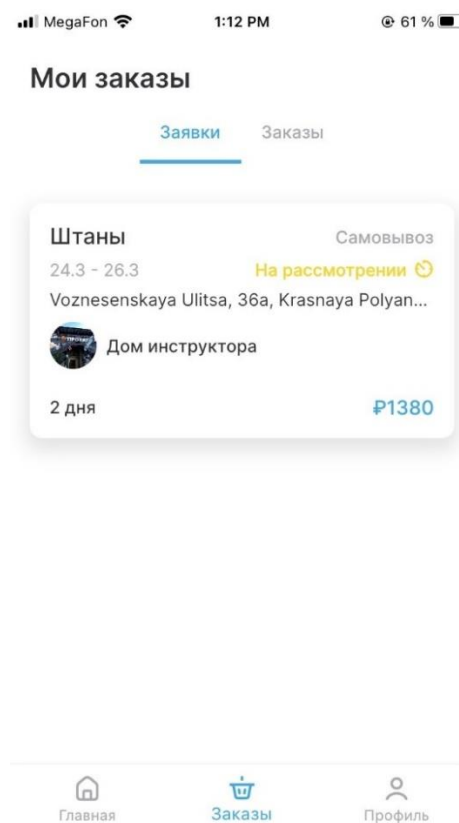


Рисунок 3 - Заказы в Прокаты.Ру

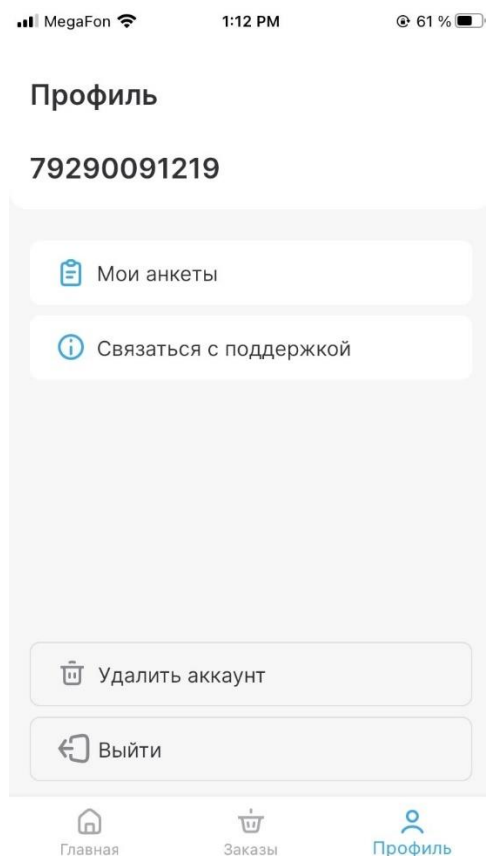


Рисунок 4 - Личный кабинет в Прокаты.Ru

К преимуществам приложения Прокаты.Ru можно отнести:

- доступность по Android и iOS;
- поддержка нескольких анкет с персональными данными;
- возможность связаться с Поддержкой по любым вопросам.

Недостатки:

- бронирование только на целый день, то есть нельзя забронировать на несколько часов, а значит, при аренде на несколько часов пользователь все равно заплатит за целый день;
- нельзя забронировать несколько товаров разных размеров сразу;
- нельзя удалить заявку на аренду.

### 2.2.2 Getski

Сервис по прокату горнолыжного оборудования по всей России: Красная Поляна, Архыз, Домбай, Эльбрус, Москва. Интерфейс приложения представлен на рисунках 5 – 7.



Рисунок 5 - Меню приложения Getski

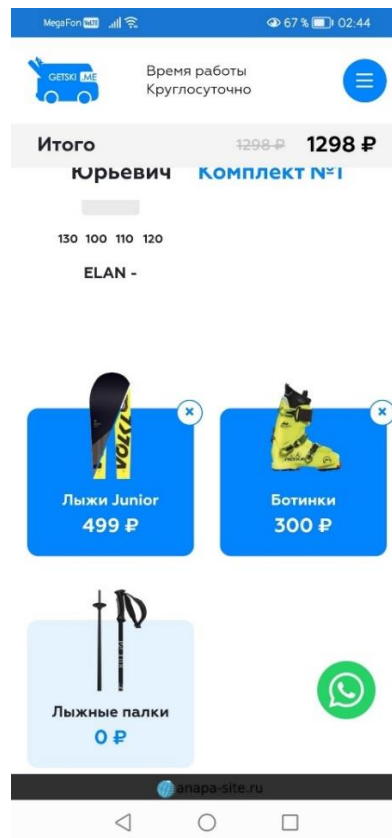


Рисунок 6 - Аренда в Getski

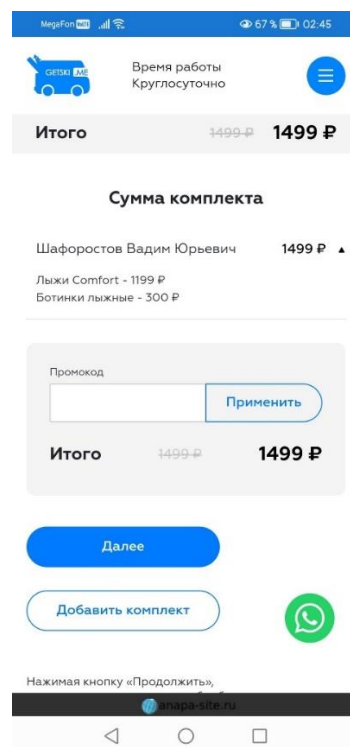


Рисунок 7 - Оплата аренды в Getski

Преимущества:

- есть возможность выбрать свои параметры: рост, вес, пол, размер обуви. Приложение будет подбирать товары под эти параметры;
- работает система скидок и купонов;
- возможность доставки, бронирования авто для самовывоза.

Недостатки:

- работает только под Android;
- регистрация сразу же при запуске с указанием не нужных для бронирования данных (например, дата рождения);
- слишком много ненужной информации (при бронировании лыж на одной странице предлагают арендовать авто);
- нельзя выбрать количество инвентаря для аренды (например, взять 2 пары лыж, обуви и т.д.);
- нет личного кабинета;
- нет возможности посмотреть остаток времени забронированных товаров.

### **2.2.3 Swissrent**

Swissrent – сервис по аренде спортивного оборудования в Швейцарии.

Интерфейс приложения представлен на рисунках 8 – 11.

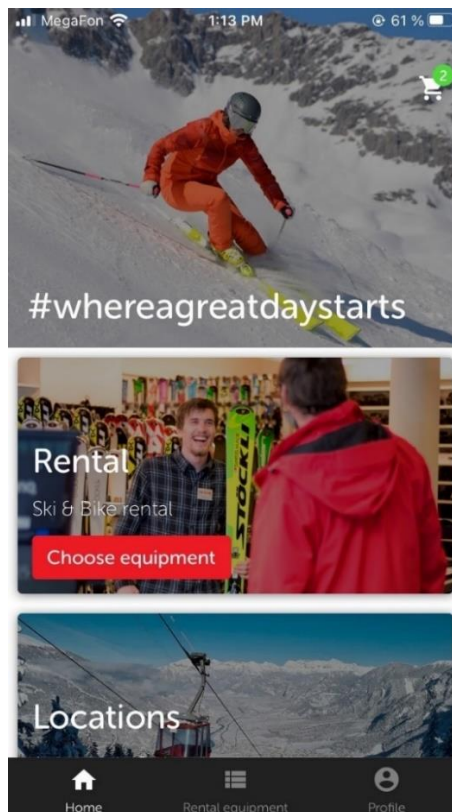


Рисунок 8 - Домашняя страница в Swissrent

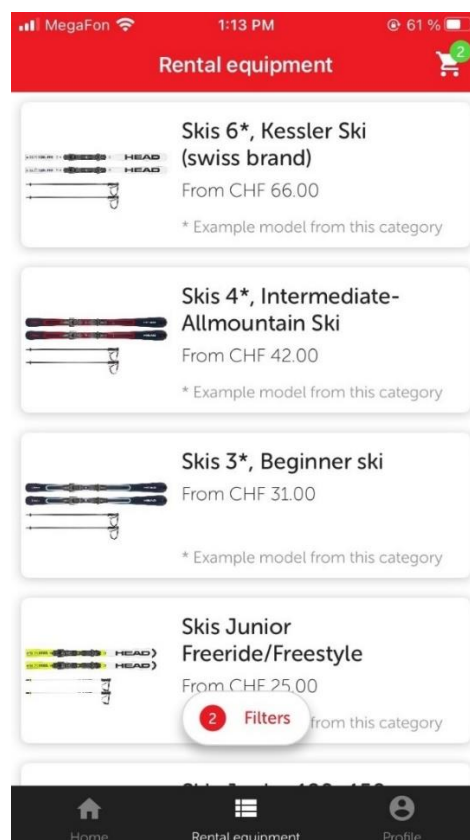


Рисунок 9 - Каталог товаров в приложении Swissrent

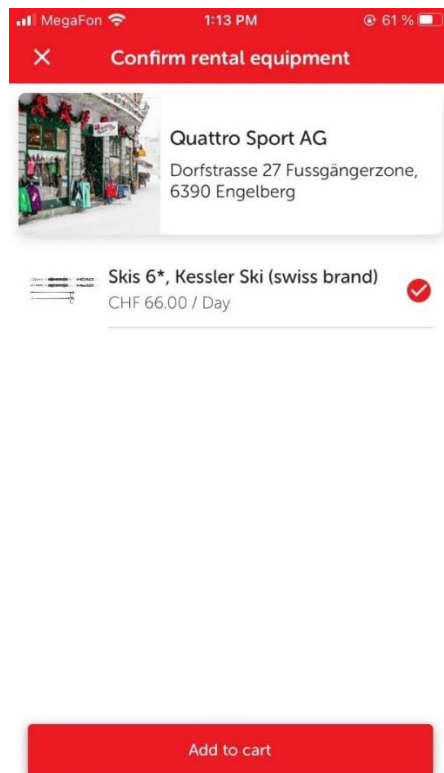


Рисунок 10 - Добавление товара в корзину в Swissrent

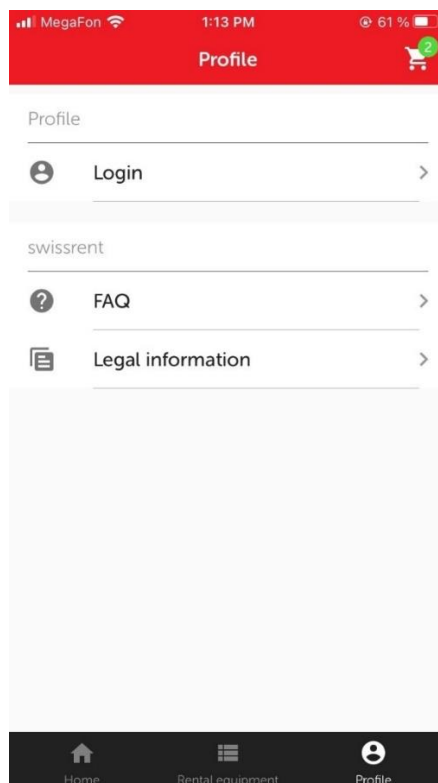


Рисунок 11 - Профиль пользователя в приложении Swissrent



Преимущества сервиса Swissrent:

- доступность по Android и iOS;
- приложение представляет собой систему, объединяющую несколько различных баз, находящихся в разных локациях;
- удобные фильтры для поиска.

Недостатки:

- урезанный функционал для неавторизованного пользователя, например, нет информации о размерах просматриваемых товаров;
- при регистрации нужно указывать страну, город и улицу проживания;
- пользователи отмечают частые сбои.

#### **2.2.4 Ski-iD**

Приложение SKI-ID от SPORTRUDI - цифровая программа лояльности, в которой пользователь может арендовать, починить и даже купить горнолыжный инвентарь. Предлагает своим клиентам систему бонусов, которые можно получать за активность и различные действия. Интерфейс приложения представлен на рисунках 12- 15.

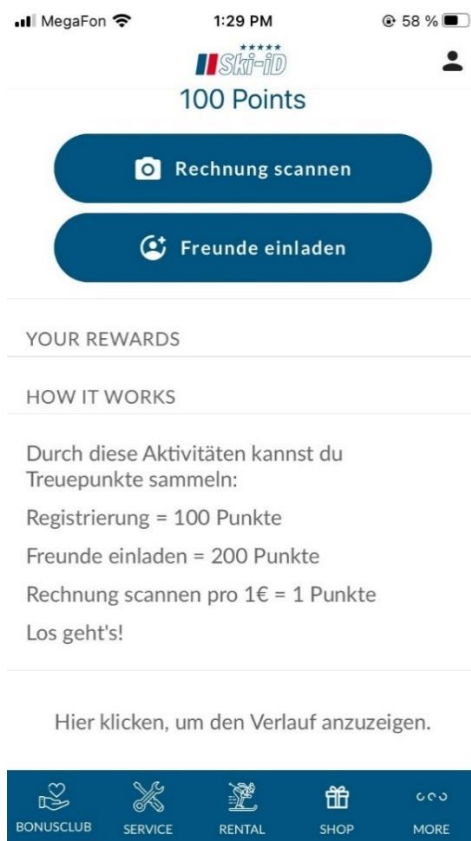


Рисунок 12 - Клуб бонусов в Ski-iD



Рисунок 13 - Сервис по ремонту в Ski-iD

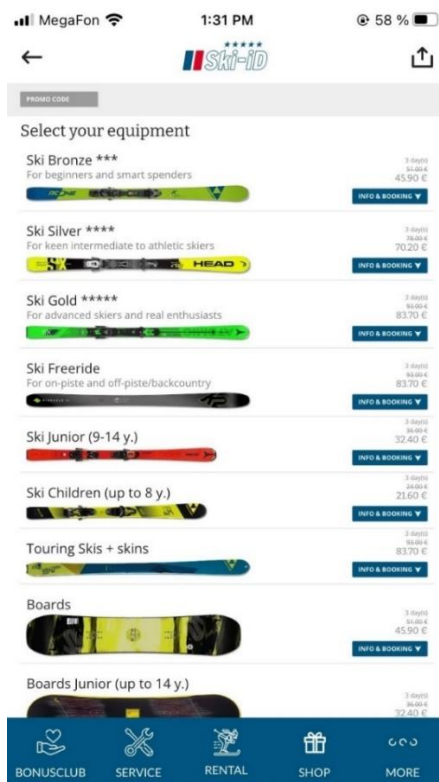


Рисунок 14 - Список товаров для аренды в Ski-iD

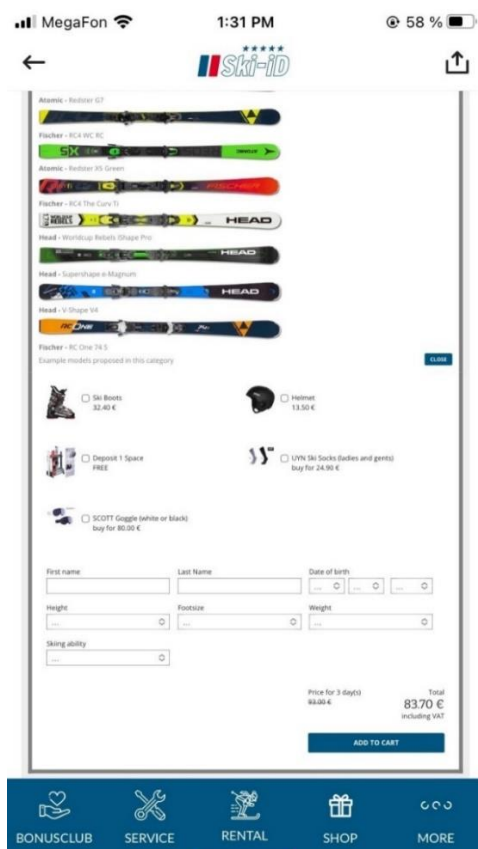


Рисунок 15 - Форма для ввода личных данных при бронировании в Ski-iD

#### Основные преимущества:

- доступность по Android и iOS;
- отличная система бонусов за различные активности;
- интегрирован сервис для ремонта и покупок инвентаря;
- удобное меню для аренды товара – содержит всё необходимое в одном месте и сразу же выводится стоимость и можно добавить карту для оплаты.

#### Недостатки:

- для просмотра товаров нужно зарегистрироваться;
- страница с доступными для аренды товарами очень большая и ее загрузка занимает много времени;
- на экране до 6 дюймов элементы крайне малы;
- много лишней информации на экране, которую можно было бы вынести на отдельную страницу.

## 3 Реализация

### 3.1 Средства реализации

Backend:

- Java – строго типизированный объектно-ориентированный язык программирования. Был выбран в качестве основного, т.к. за много лет существования успел зарекомендовать себя как надежная и легко масштабируемая платформа разработки и до сих пор не потерял своей актуальности. К тому же существует огромное количество фреймворков и библиотек, написанных для Java, которые, в перспективе, можно легко интегрировать в проект;
- Spring Framework – универсальный фреймворк с открытым исходным кодом для Java-платформы. Был выбран, т.к. 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;
- Liquibase – продукт с открытым исходным кодом для обеспечения миграций баз данных. Был выбран, т.к. легко интегрируется со Spring Framework и поддерживает PostgreSQL;
- Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации;
- PostgreSQL – объектно-реляционная система управления базами данных. Является продуктом с открытым исходным кодом, который поддерживается многими серверами, в связи с чем и был выбран.

Frontend:

- Flutter – это платформа с открытым исходным кодом, который разработан и поддерживается Google. Flutter позволяет разработать

приложение с единой кодовой базой, которое будет работать на всех платформах - Web, iOS, Android, Window, Mac, Linux;

- Dart – язык программирования общего назначения от компании Google, который предназначен прежде всего для разработки прикладных приложений. Основной сферой применения языка Dart на сегодняшний день является разработка графических приложений с помощью кроссплатформенного фреймворка Flutter.

Инструменты для ведения документации:

- Miro – быстрая и безопасная интерактивная доска для совместной работы распределенных команд;
- Figma – это графический онлайн-редактор для совместной работы. В нём можно создать прототип сайта, интерфейс приложения и обсудить правки с коллегами в реальном времени;
- Draw.io – бесплатное кроссплатформенное программное обеспечение с открытым исходным кодом для рисования графиков, создания диаграмм, таких как блок-схемы, каркасы, диаграммы UML, организационные диаграммы и сетевые диаграммы;
- Ramus – это программа, в которой можно проектировать визуальные диаграммы. Софт обеспечивает создание наглядных бизнес процессов в виде диаграмм;
- Swagger – это набор инструментов, который позволяет автоматически описывать API на основе его кода.

Дополнительный инструментарий:

- Git – распределённая система управления версиями;
- GitHub – это сервис для совместной разработки и хостинга проектов, с помощью которого над кодом проекта может работать неограниченное количество программистов из любых точек мира;

— Trello – это визуальный инструмент, который позволяет вашей команде управлять проектами, рабочими процессами и заданиям любых типов.

3.2 IDEF0

На рисунках 16 и 17 представлена IDEF0 диаграмма, которая отображает процесс проката спортивного инвентаря. Она помогает лучше понять, какие шаги необходимо выполнить для аренды оборудования, начиная с регистрации пользователя и заканчивая оплатой.

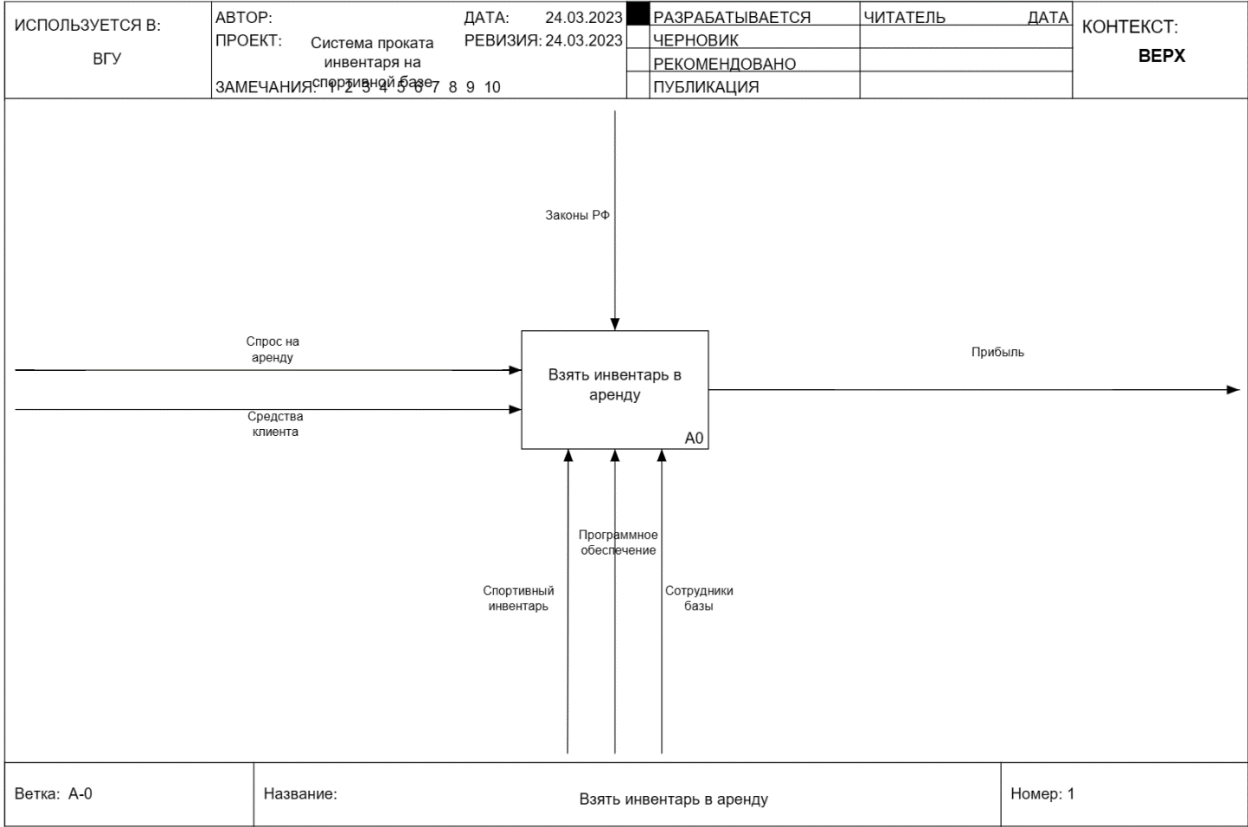


Рисунок 16 - IDEF0

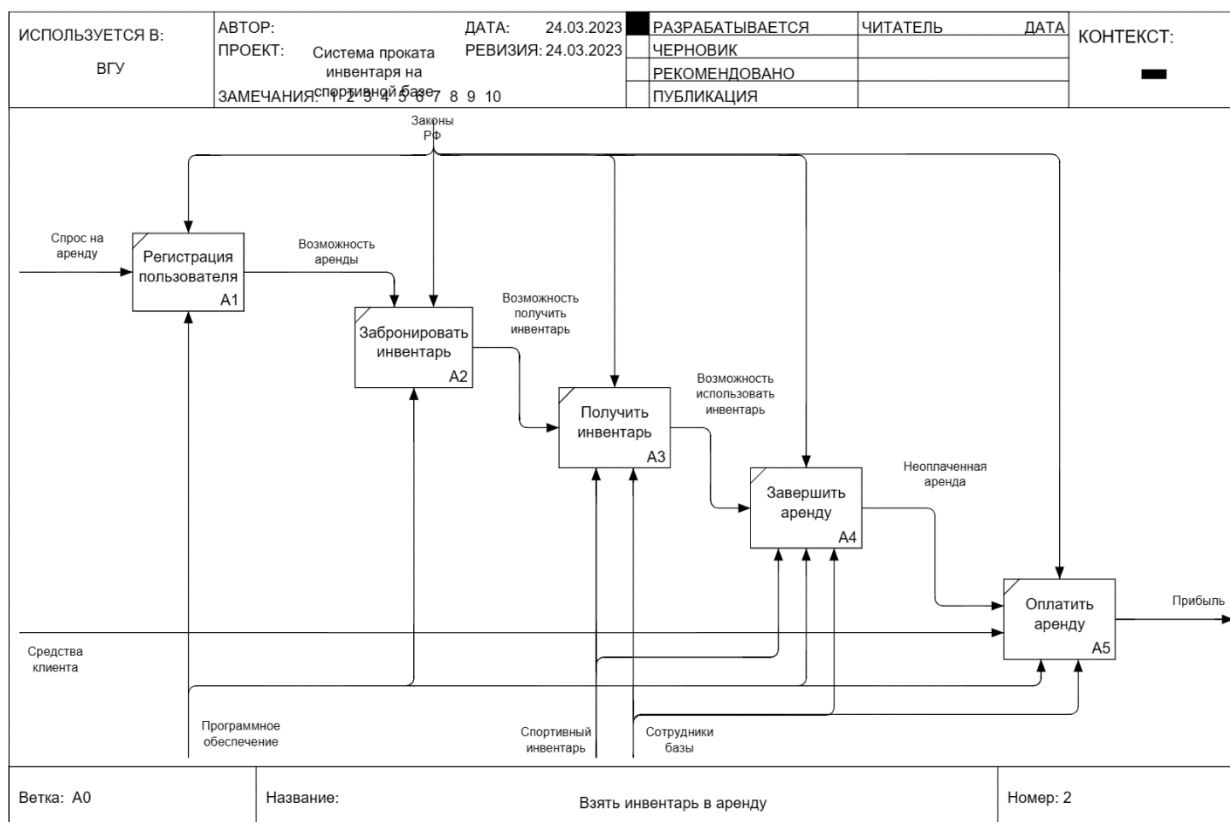


Рисунок 17 - IDEF0

### 3.3 Диаграмма прецедентов

Use-case диаграмма, представленная на рисунке 18, описывает, как различные актеры будут использовать систему и какие действия смогут выполнять. Здесь присутствуют три основных актера: неавторизованный пользователь, авторизованный пользователь и сотрудник.

Неавторизованный пользователь — это человек, который еще не зарегистрировался в системе и не имеет доступа к ее функционалу. Он может просматривать общедоступную информацию, но не может выполнять какие-либо действия.

Авторизованный пользователь — это человек, который зарегистрировался в системе и имеет доступ к ее функционалу. Он может выполнять различные действия, такие как добавление товаров в корзину, бронирование и т.д.



Сотрудник — это человек, который работает в компании и имеет доступ к определенным функциям системы, которые не доступны обычным пользователям. Он может выполнять задачи, связанные с управлением арендами, обработкой платежей и т.д.

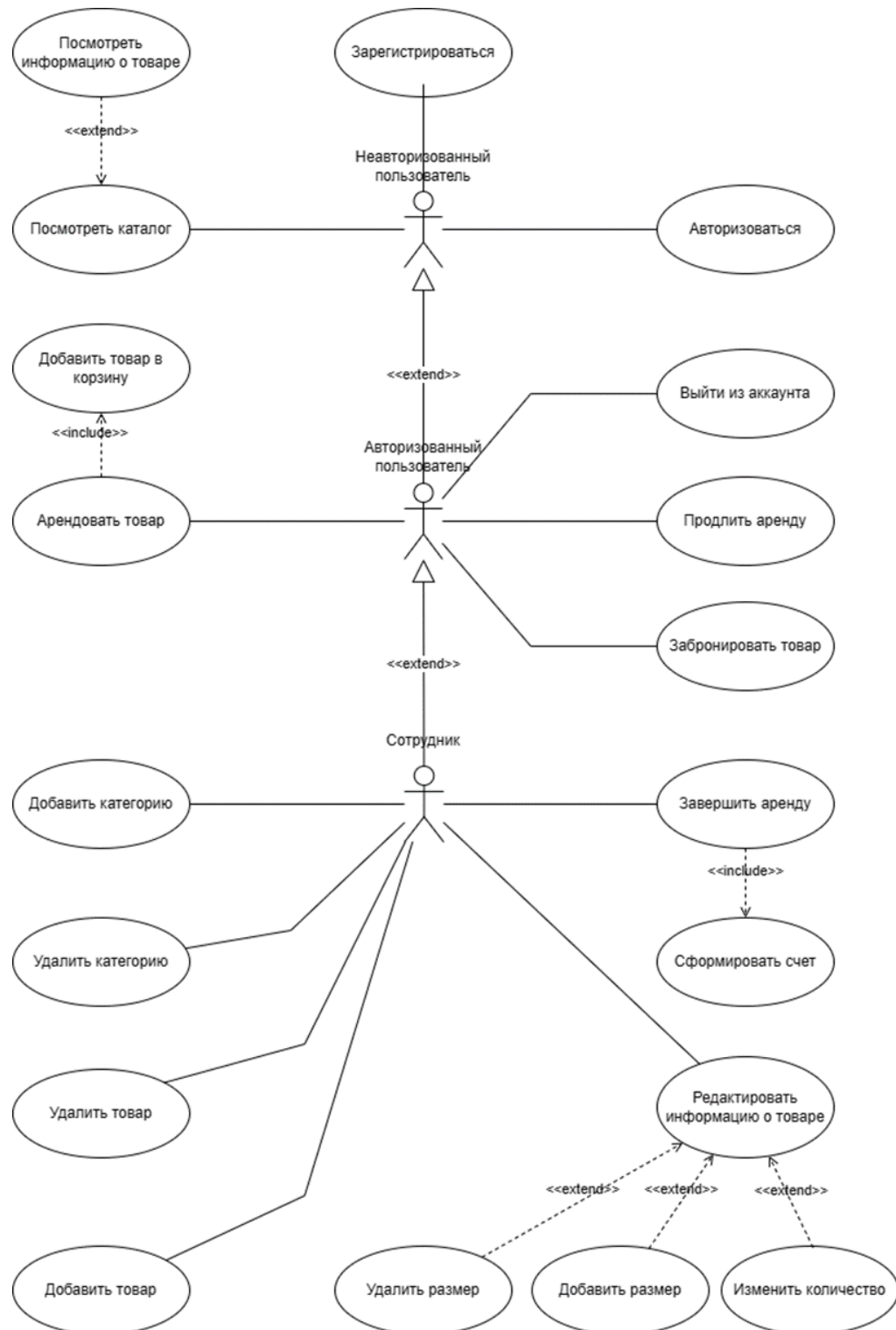


Рисунок 18 - Диаграмма прецедентов

### 3.4 Диаграмма состояний

Диаграмма состояний (рисунок 19) показывает процесс работы приложения по прокату спортивного инвентаря. Она описывает, как пользователь взаимодействует с системой на разных этапах аренды, а также какие действия выполняет приложение в ответ. Данная диаграмма поможет создать наиболее удобный интерфейс для пользователей.

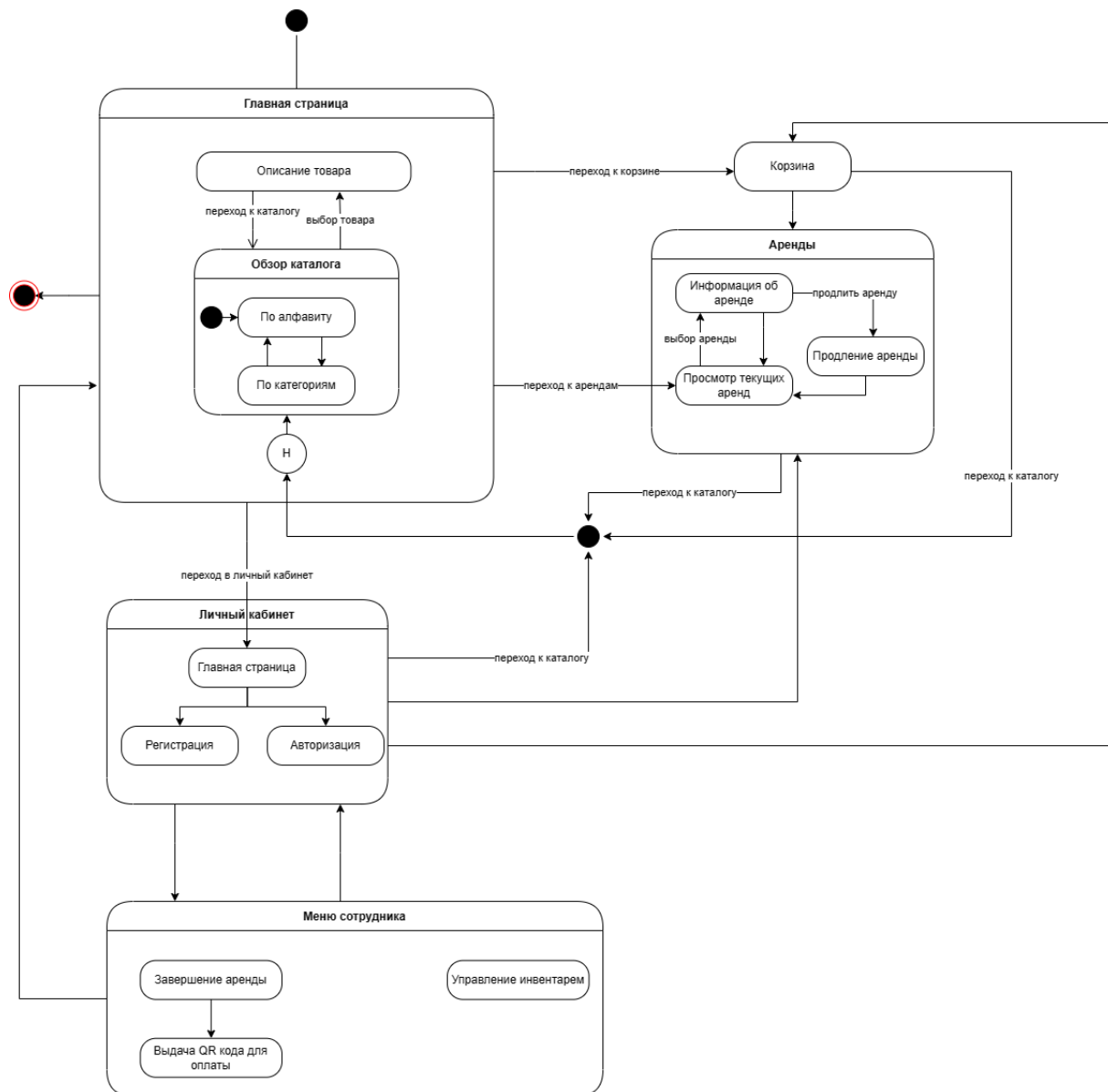


Рисунок 19 - Диаграмма состояний

### 3.5 Диаграмма последовательностей

Данная диаграмма последовательностей (рисунок 20) описывает взаимодействие между объектами приложения на разных этапах процесса аренды. Она показывает, как объекты взаимодействуют друг с другом и какие действия выполняют в ответ на запросы.

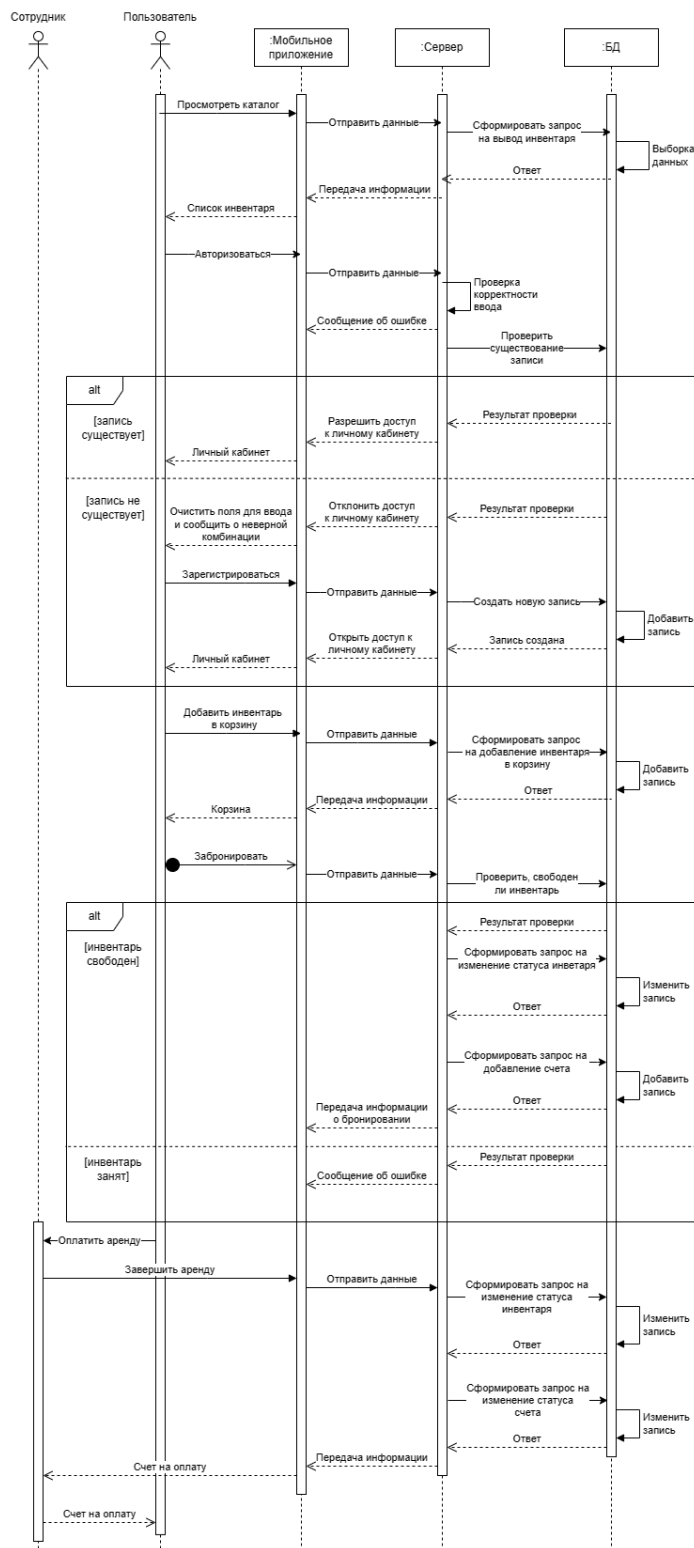


Рисунок 20 - Диаграмма последовательностей

### 3.6 Диаграмма деятельности

Диаграмма деятельности, представленная на рисунке 21, описывает последовательность действий, которые выполняет пользователь приложения.

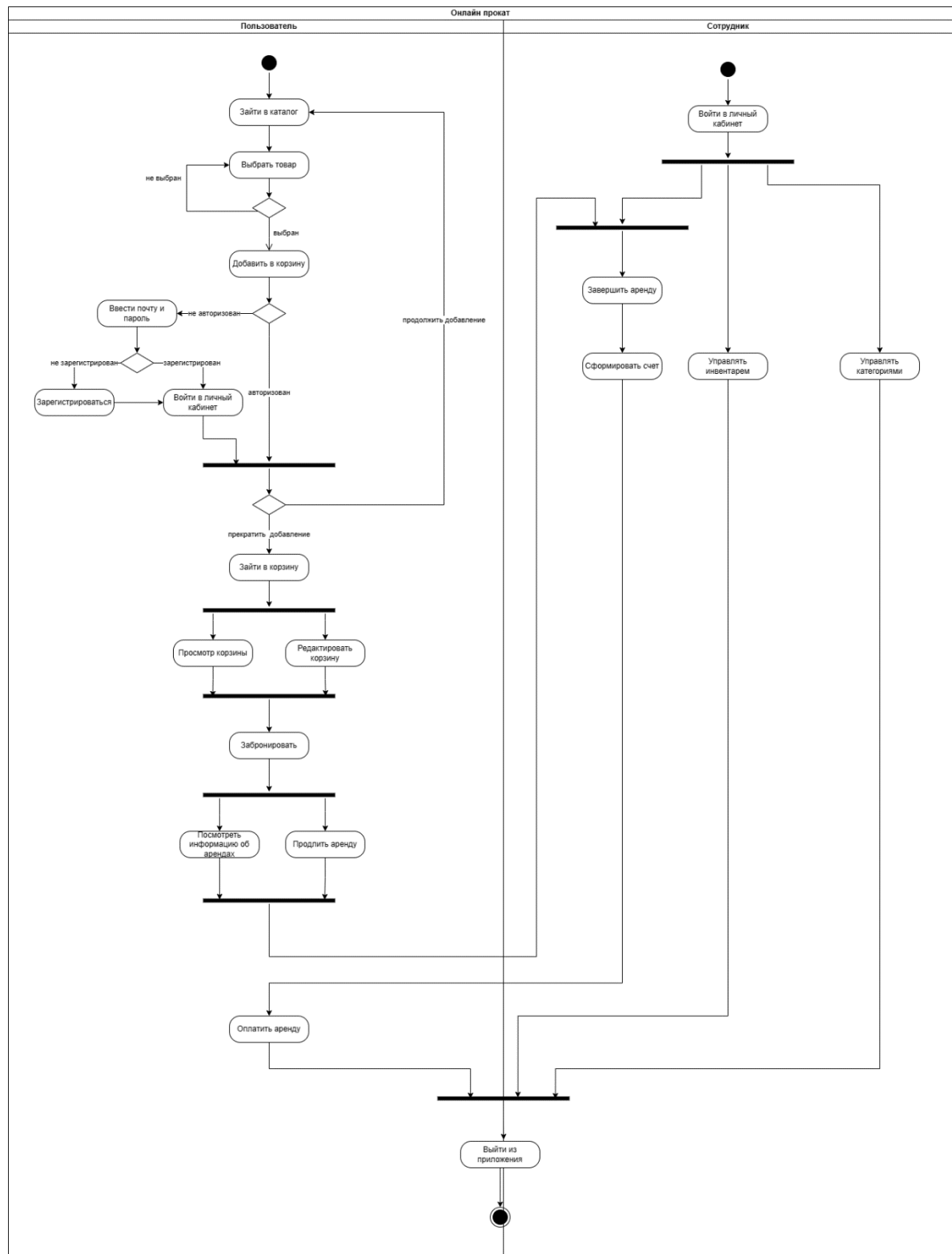


Рисунок 21 - Диаграмма деятельности

## **3.7 Методы разработки**

### **3.7.1 Kanban**

Kanban-разработка — это методология управления проектами, которая основывается на использовании доски с задачами и работой с ними по принципу потока [3]. Она позволяет сократить время на выполнение задач, упростить коммуникацию между участниками проекта и повысить качество работы.

Kanban-разработка использовалась для управления задачами, которые необходимо было выполнить в рамках проекта. На доске Trello помещались все задачи, которые нужно было выполнить, и каждый участник команды мог видеть, какие задачи еще остались, какие уже выполнены, и какие находятся в работе.

### **3.7.2 Contract-first подход**

Contract-first разработка — это подход к разработке программного обеспечения, при котором сначала создается спецификация (контракт) API, а затем уже на ее основе создается код. Этот подход позволяет избежать проблем с несоответствием между API и кодом, а также ускоряет процесс разработки [2].

В проекте contract-first разработка использовалась с помощью инструмента Swagger. Сначала была создана спецификация API в формате OpenAPI, которая описывала все доступные эндпойнты, параметры запросов и ответы сервера. Затем на основе этой спецификации осуществлялось написание код на языке программирования, который реализовывал логику работы API. Такой подход позволил избежать ошибок и ускорил процесс разработки.

### 3.7.3 Git-flow

В проекте использовалась git-flow методология ветвления и слияния [4]. Были созданы три основные ветки: main, client и server. В первой ветке хранился стабильный код, который был готов к релизу, а в ветке server/develop и client/develop - код, который находился в разработке.

В ветке feature, которая была создана от ветки develop, разрабатывались соответствующие части проекта.

После завершения работы изменения сливались обратно в ветку develop и в ветку master, которая была обновлена до последней версии проекта. Такой подход позволил эффективно разрабатывать части проекта и помог разделить процесс разработки клиента и сервера.

### 3.7.4 CI/CD

CI/CD (Continuous Integration/Continuous Delivery) — это методология разработки программного обеспечения, которая позволяет автоматизировать процесс сборки, тестирования и доставки кода в производственную среду. CI/CD ускоряет процесс разработки и повышает качество кода, так как позволяет быстро выявлять ошибки и исправлять их [5].

В данном проекте CI/CD использовался для автоматизации процесса развертывания приложения на сервере. Каждый раз, когда разработчик отправлял изменения в централизованный репозиторий, автоматически запускался процесс сборки и тестирования кода. Если тесты проходили успешно, автоматически создавался Docker-контейнер с приложением.

В проекте Java Spring Boot и Hibernate CI/CD использовался для автоматизации процесса развертывания приложения на сервере. С помощью этих инструментов была настроена непрерывная интеграция и доставка приложения в производственную среду. Каждый раз, когда разработчик отправлял изменения в централизованный репозиторий на GitHub, GitHub

Actions автоматически запускал процесс сборки и тестирования кода. Если тесты проходили успешно, GitHub Actions автоматически создавал Docker-контейнер с приложением и доставлял его в Kubernetes-кластер на платформе Okteto.

Таким образом, использование CI/CD с инструментами GitHub Actions, Kubernetes и Okteto позволило автоматизировать процесс развертывания приложения на сервере и ускорить процесс разработки, что повысило качество кода и улучшило опыт пользователей.

## **3.8 Реализация серверной части приложения**

### **3.8.1 Схема базы данных**

База данных является важной составляющей любого приложения, ведь именно она хранит и обрабатывает всю необходимую информацию. В данном курсовом проекте была выбрана PostgreSQL, так как это мощная и надежная СУБД с открытым исходным кодом. На рисунке 22 представлена схема структуры базы данных разрабатываемого приложения по прокату спортивного инвентаря.

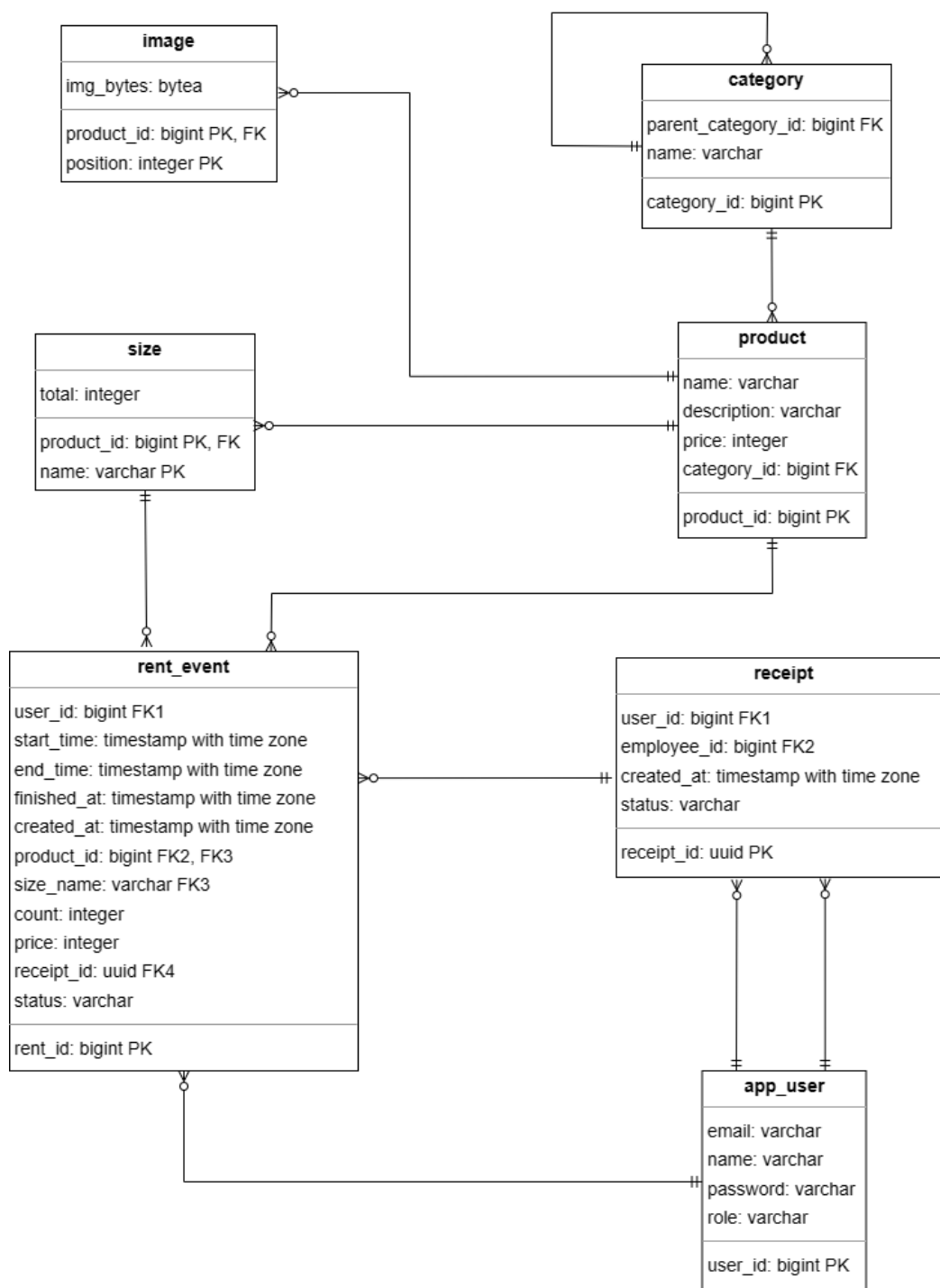


Рисунок 22 - ER диаграмма

Таблица product содержит информацию о продуктах, которые могут быть арендованы, size - о размерах продуктов, image - об изображениях продуктов, category - категориях продуктов, receipt – о заказах пользователей, app\_user - пользователях приложения и rent\_event содержит информацию об аренде продуктов.



### 3.8.2 Архитектура

В проекте использовался Java Spring Boot и Hibernate API для обмена данными между клиентской и серверной частями приложения. В частности, созданы RESTful API, которые позволяют клиентам отправлять запросы на сервер и получать ответы в формате JSON. Для создания API использованы различные инструменты, включая Spring MVC, Spring Data JPA и Hibernate ORM [6]. С помощью этих инструментов были созданы контроллеры, сервисы и репозитории, которые обеспечивают взаимодействие с базой данных и обработку запросов от клиентов.

### 3.8.3 Классы сущностей

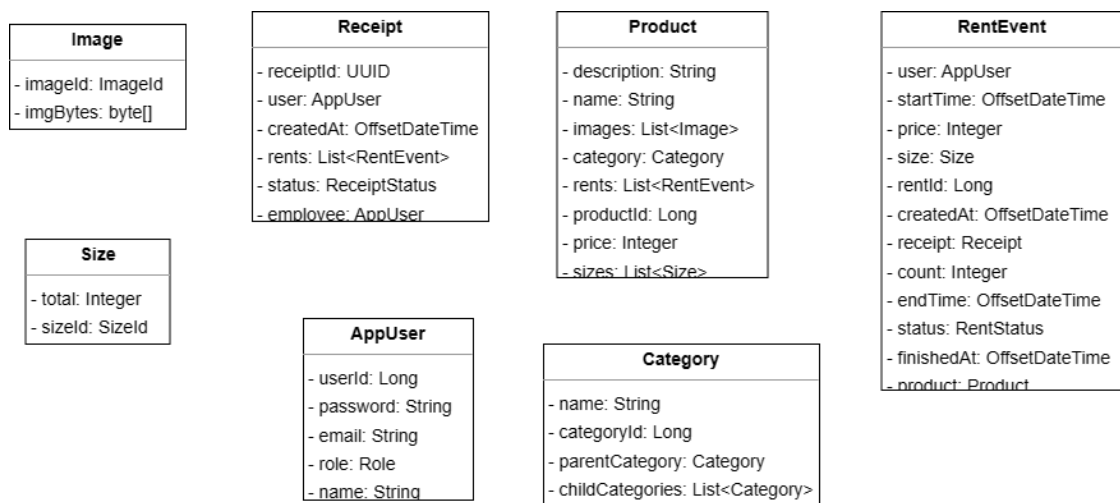


Рисунок 23 - Диаграмма классов сущностей

Поля каждого из этих классов эквивалентны атрибутам одноименных таблиц в БД.

### 3.8.4 Классы репозиториев

Для каждой из сущностей был реализован интерфейс-репозиторий, который является наследником JpaRepository. Такой подход позволил существенно уменьшить количество написанного кода, путем использования

возможностей Spring Data, а именно больше количество уже реализованных методов для генерации запроса в базу данных и получения ответа и возможность объявления своих методов, при правильном написании названий которых Spring Data сам сгенерирует запрос в базу.

### **3.8.5 Классы контроллеров**

Контроллеры – такие классы, каждый метод из которых обрабатывает запрос с клиента на определенный маппинг и возвращает ответ в виде `ResponseEntity` в Rest API. Для каждой из сущностей были написаны контроллеры (рисунок 24), методы которых отвечают за необходимые приложению действия с этими сущностями. Помимо этого, был реализован контроллер аутентификации, который отвечает за вход и регистрацию пользователей.

Здесь важно учитывать права доступа пользователей к определенным действиям. Например, при создании, обновлении или удалении данных, необходимо ограничить доступ только для сотрудников, которые имеют соответствующие права. Для этого используются механизмы авторизации и аутентификации, чтобы определить, какие пользователи имеют доступ к определенным действиям. Например, можно создать роли "пользователь" и "сотрудник", где пользователи могут только получать данные (GET), а сотрудники могут выполнять все действия (POST, PATCH, DELETE). Для реализации такой системы прав доступа используются аннотации Spring Security, которые позволяют задавать различные правила доступа для каждого метода контроллера.



Рисунок 24 - Диаграмма классов контроллеров

### 3.8.6 Классы сервисов

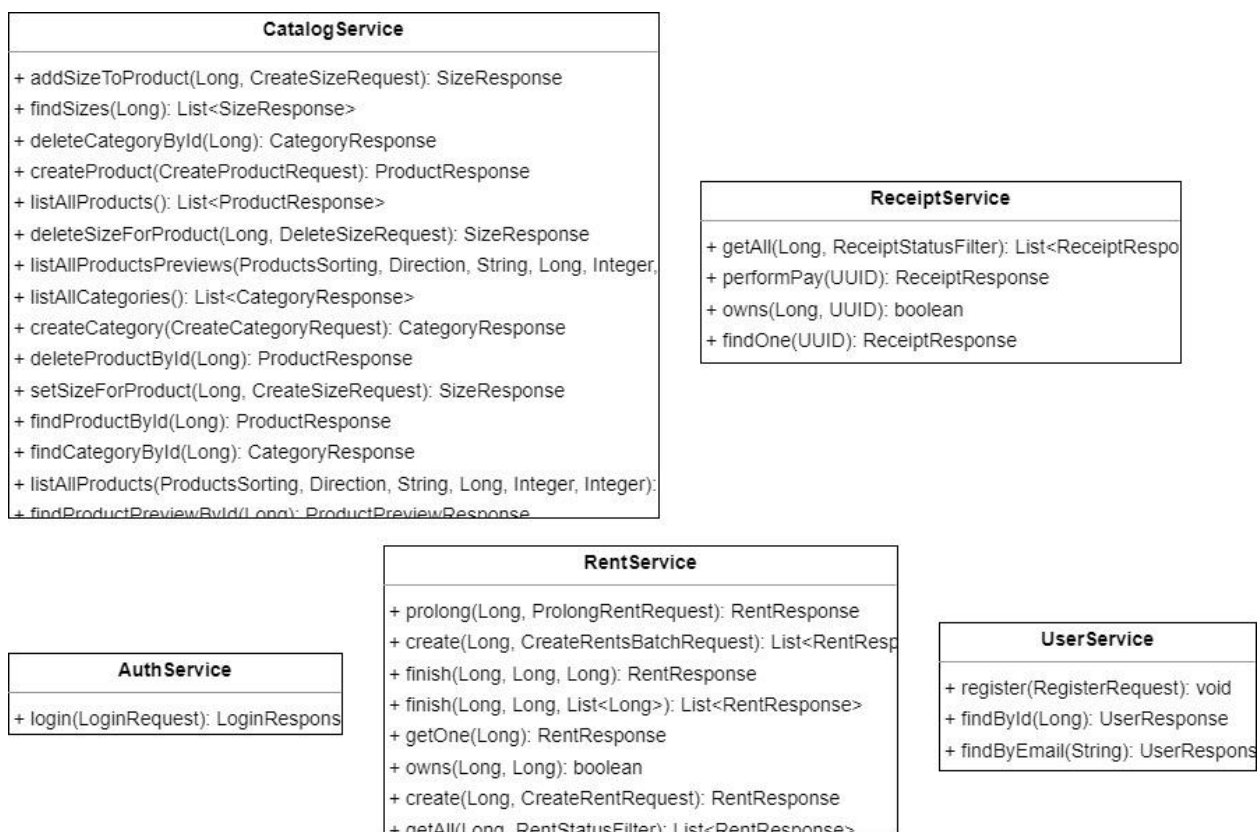


Рисунок 25 - Диаграмма классов сервисов

Каждый из этих классов является частью слоя бизнес-логики, т.е. выступают связующим звеном между слоем доступа к данным и контроллерами. Все алгоритмы находятся в этих классах.

### **3.8.7 Бизнес-логика**

Кроме стандартных CRUD операций, сервер непосредственно отвечает за реализацию бизнес-логики, связанной с учетом доступных товаров в определенный промежуток времени. Для этого используется класс `RentProcessor`, который содержит функцию `countAvailableAt`. Эта функция позволяет определить количество доступных товаров в заданный промежуток времени, учитывая уже забронированные товары и текущее количество на складе. Например, если клиент хочет забронировать товар на два дня, функция проверит наличие свободных товаров на этот период, и если товары доступны, то произойдет успешная бронь.

Класс `RentUpdaterScheduler` создан для реализации бизнес-логики, связанной с учетом времени и изменения статуса аренды. Его функционал позволяет отслеживать бронирования товаров и автоматически изменять их статус в зависимости от времени, на которое они были забронированы. Например, если клиент забронировал товар на два дня, `RentUpdaterScheduler` автоматически изменит его статус после истечения этого периода. Это позволяет автоматически управлять доступностью товаров и предотвращать конфликты при бронировании одного и того же товара на пересекающийся период времени.

Кроме учета времени и изменения статуса аренды, реализована логика завершения аренды. Для изменения статуса `RentEvent` на `CREATED`, `ONGOING`, `EXPIRED`, `FINISHED` `AWAITING_PEMENT` в зависимости от текущего времени и времени окончания аренды. Например, если аренда была создана, но клиент еще не забрал товар, статус будет `CREATED`. Когда клиент забирает товар, статус изменится на `ONGOING`. Если время аренды истекло,

статус изменится на EXPIRED. Когда клиент вернет товар, статус изменится на FINISHED. Если товар уже забронирован другим клиентом на пересекающийся период времени, статус изменится на AWAITING\_PAYMENT. Также формируется Receipt из RentEvent'ов, которые были завершены. Receipt содержит информацию о продукте, цене, дате начала и окончания аренды, а также информацию о клиенте.

Также в конфигурации задается процент за просроченную аренду, который будет автоматически начисляться в зависимости от времени просрочки. Например, если задано 10 процентов за 10 минут, то при просрочке на 10 минут будет начислен штраф в размере 10 процентов от стоимости аренды. Это обеспечивает дополнительную мотивацию для клиентов возвращать товары вовремя и позволяет управлять рисками связанными с просроченными арендами.

### **3.8.8 Аутентификация**

Аутентификация — это процесс проверки подлинности пользователя при попытке доступа к защищенным ресурсам приложения. Spring Security — это мощный фреймворк для управления аутентификацией и авторизацией в веб-приложениях на основе Java. JSON Web Token (JWT) — это стандарт для создания токенов, которые могут использоваться для аутентификации и авторизации в приложениях. [7]

Токен содержит информацию о пользователе, а также подпись для проверки целостности данных. Использование JWT токена в Spring Security позволяет более безопасно передавать информацию о пользователе между клиентом и сервером. При авторизации пользователь получает токен, который потом можно использовать для доступа к защищенным ресурсам без повторной аутентификации.

Реализация аутентификации с помощью Spring Security и JWT токенов включает в себя конфигурацию Spring Security и создание контроллера для обработки запросов аутентификации и генерации токенов.

JWT-токены используются для авторизации пользователей в данном приложении. Когда пользователь успешно аутентифицируется, сервер создает JWT-токен и отправляет его клиенту. Клиент сохраняет токен и отправляет его в каждом запросе на сервер, чтобы доказать свою легитимность. Сервер проверяет подпись токена, расшифровывает полезную нагрузку и принимает решение о предоставлении доступа к запрашиваемому ресурсу.

### **3.9 Реализация клиентской части приложения**

#### **3.9.1 Архитектура**

Приложение реализовано на Flutter с использованием архитектурного подхода Bloc. Это означает, что приложение разделено на три основных слоя: пользовательский интерфейс (UI), бизнес-логику (Bloc) и слой доступа к данным (Repository) [8].

UI-слой отвечает за отображение данных и взаимодействие с пользователем. Он содержит виджеты, которые отображают информацию о продуктах, заказах и пользователях, а также позволяет пользователю совершать действия, такие как добавление товаров в корзину, оформление заказа и т.д.

Бизнес-логика (Bloc) отвечает за обработку логики приложения и управление состоянием приложения. Он получает данные из слоя доступа к данным (Repository), обрабатывает их и передает обновленное состояние в UI-слой. Благодаря этому подходу, мы можем отделить бизнес-логику от пользовательского интерфейса и сделать наше приложение более модульным и масштабируемым.

Слой доступа к данным (Repository) отвечает за получение данных с сервера и передачу их в бизнес-логику. Он содержит методы для получения информации о продуктах, заказах и пользователях, а также для добавления новых заказов и обновления информации о существующих.

Таким образом, Bloc-классы играют ключевую роль в нашем приложении, отвечая за управление состоянием приложения и обработку бизнес-логики. Они получают данные из слоя доступа к данным (Repository), обрабатывают их и передают обновленное состояние в пользовательский интерфейс (UI). Благодаря этому подходу, можно создать приложение, которое будет легко масштабироваться и поддерживать в будущем.

### **3.9.2 Изменение цветовой схемы**

RemoteConfig - это инструмент Firebase, который позволяет изменять параметры приложения без необходимости выпуска новой версии приложения. С помощью RemoteConfig можно изменять различные параметры, такие как цветовую схему, текст на экранах, макеты и остальные параметры, связанные с цветовой палитрой.

В дальнейшем описывается с чего начинается знакомство пользователя с клиентской частью программного обеспечения.

Онбординг - это процесс знакомства пользователя с приложением. Обычно он состоит из нескольких экранов, на которых пользователю показываются основные функции приложения. С помощью RemoteConfig можно изменять текст и стиль на экранах онбординга, чтобы сделать его более привлекательным и информативным для пользователей.

Например, если мы хотим изменить цветовую схему приложения, мы можем использовать RemoteConfig, чтобы изменить цвет фона или шрифта на всех экранах. Пользователи могут увидеть эти изменения мгновенно, без необходимости обновления приложения.

### 3.9.3 Экраны

Пользователь при наличии аккаунта может войти в систему, введя свои данные на экране авторизации: адрес электронной почты и пароль (рисунок 26).

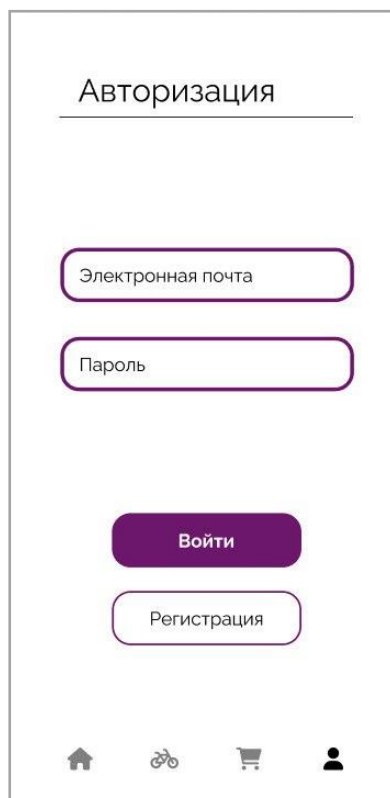
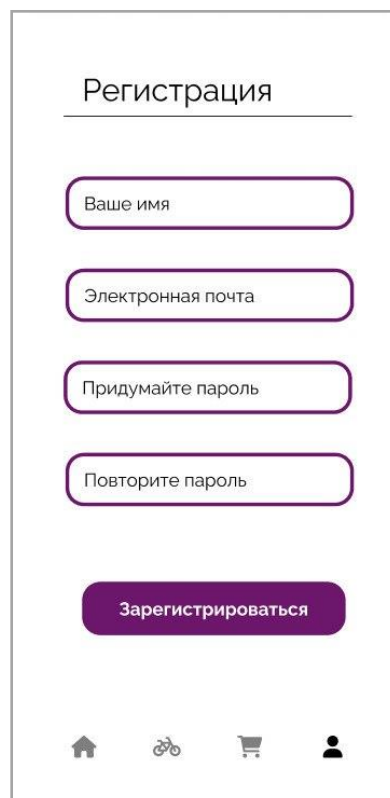


Рисунок 26 - Авторизация

В ином случае пользователь может зарегистрировать свой аккаунт в системе, введя имя, адрес электронной почты и пароль с повторным вводом для подтверждения (рисунок 27).





Регистрация

Ваше имя

Электронная почта

Придумайте пароль

Повторите пароль

Зарегистрироваться

Home, Bike, Cart, User icons

The registration form is titled 'Регистрация' (Registration). It contains four input fields: 'Ваше имя' (Your name), 'Электронная почта' (Email), 'Придумайте пароль' (Create a password), and 'Повторите пароль' (Repeat password). Below these fields is a purple button labeled 'Зарегистрироваться' (Register). At the bottom of the form are four icons: a house, a bicycle, a shopping cart, and a person.

Рисунок 27 - Регистрация

Любой пользователь имеет возможность просматривать информацию об ассортименте базы в виде списка товаров с возможностью фильтрации и сортировки (рисунок 28).



Поиск

Сноуборд LIB TECH DYNASWORD 2.000 Р/час

Велосипед горный Stels Focus 1.000 Р/час

Лыжные палки Cober 500 Р/час

Шлем Briko sistmic x 900 Р/час

Home, Bike, Cart, User icons

The product catalog interface features a search bar at the top labeled 'Поиск'. Below it are four product cards arranged in a 2x2 grid. Each card displays an image of the product, its name, and its price per hour. The products are: 'Сноуборд LIB TECH DYNASWORD' (2.000 Р/час), 'Велосипед горный Stels Focus' (1.000 Р/час), 'Лыжные палки Cober' (500 Р/час), and 'Шлем Briko sistmic x' (900 Р/час). At the bottom are four icons: a house, a bicycle, a shopping cart, and a person.

Рисунок 28 - Каталог

Пользователь может выбрать конкретный товар и перейти на страницу с его более подробным описанием, где ему доступно выбрать время начала и количество часов аренды (рисунок 29). Если в данный момент товар уже арендован, то доступна функция “Уведомить по освобождении”, при активации которой пользователь получит уведомление после того, как товар сдадут сотруднику.

### Информация о товаре



Название

Велосипед Горный Stels Focus

Описание

Горный велосипед Navigator-500 V 26" F020 (2022) отлично подойдет для начинающих и заядлых велосипедистов.

Цена

1000 руб./час

Размер

детский

Всего доступно

3

Количество

- 1 +

Время начала аренды

Часов аренды

- 1 +

Уведомить по освобождению

Добавить в корзину




Рисунок 29 - Информация о товаре

Авторизованный пользователь имеет возможность добавить товар в корзину. При попытке неавторизованного пользователя добавить товар в корзину он будет перенаправлен на страницу авторизации.

Авторизованный пользователь может просматривать список товаров, добавленных в корзину (рисунок 30). Для каждого товара в корзине можно изменять размер и количество арендуемых единиц. Также пользователь может забронировать товары, находящиеся в корзине.



Рисунок 30 - Корзина

Авторизованный пользователь имеет возможность просмотреть информацию о всех своих текущих арендах (рисунок 31). При выборе конкретной аренды пользователь увидит полную информацию об аренде. Есть возможность продлить аренду при условии, что товар свободен на новое время.

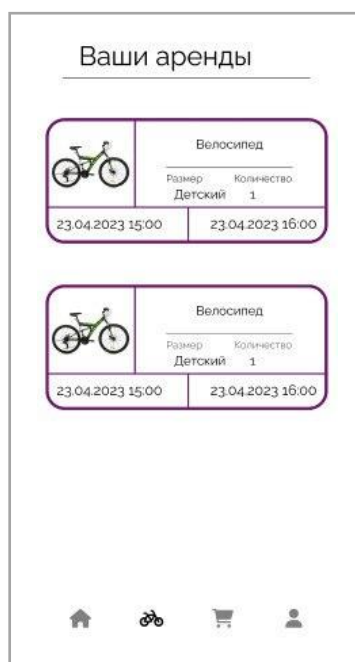


Рисунок 31 - Аренды

Авторизованный пользователь имеет доступ в личный кабинет (рисунок 32), в котором содержится личная информация: имя и адрес электронной почты. У сотрудника в личном кабинете доступен переход на экран «Меню сотрудника».

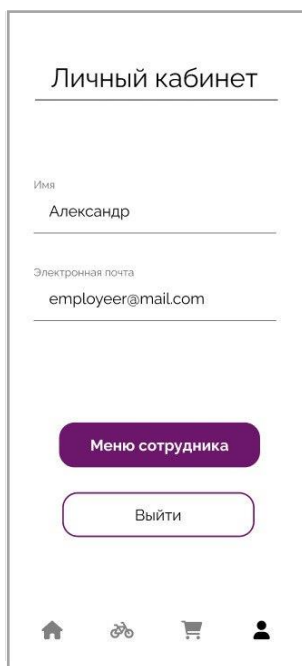


Рисунок 32 - Личный кабинет

Из меню сотрудника (рисунок 33) можно попасть в два меню: “Завершение аренды”, “Управление каталогом”.

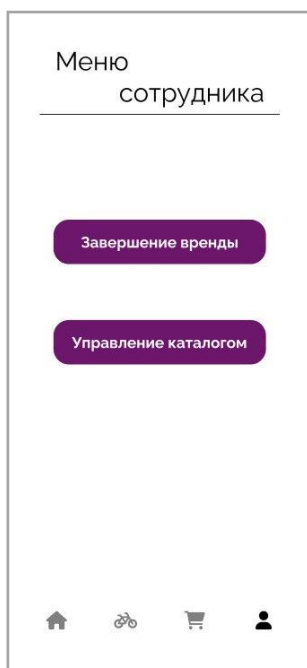


Рисунок 33 - Меню сотрудника

Сотрудник, после ввода адреса электронной почты клиента, получает доступ к действующим арендам клиента, с возможностью завершить их и сформировать счет для оплаты (рисунок 34). После завершения аренды товар считается доступным для аренды другими пользователями.

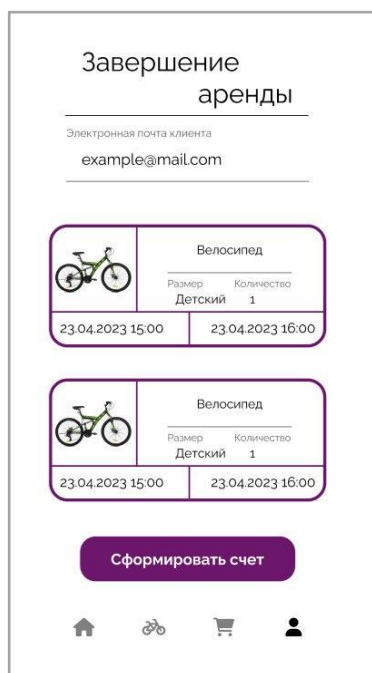


Рисунок 34 - Завершение аренды

Сотрудник завершает аренду и приложение выводит общий чек за аренду и автоматически генерируется QR-код, отсканировав который, арендатор сможет оплатить аренду (рисунок 35).



Рисунок 35 - Оплата аренды

Сотрудник может посмотреть весь ассортимент, удалить товар и добавить новый товар, указав его параметры (рисунок 36). Выбрав товар, сотрудник попадет в меню «Редактирование количества».



Рисунок 36 - Управление инвентарем

Для выбранного товара сотрудник может редактировать доступные размеры и количества каждого размера, также добавлять новые размеры, указав размер и количество (рисунок 37).



Рисунок 37 - Редактирование количества

Сотрудник может посмотреть список всех категорий, удалить категорию и добавить новую категорию, указав название и родительскую категорию (рисунок 38).



Рисунок 38 - Управление категориями

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано приложение, соответствующее всем ранее поставленным требованиям.

Были проведены анализ существующих решений и проектирование приложения, учитывающее полученные данные. Реализация приложения включала в себя создание back-end и front-end частей, а также связь между ними с помощью REST API. Была разработана база данных и развернута на удалённом сервере.

Дальнейшее развитие приложения может включать в себя добавление новых функций и улучшение существующих, например:

- интеграции с платежными системами для обеспечения реальных финансовых операций;
- реализации аккаунтов общего пользования (семейного, корпоративного) для учета инвентаря, используемого группой людей;
- возможности авторизации пользователя, используя сторонние сервисы;
- подтверждения электронной почты пользователя;
- возможности заполнения антропометрических данных пользователя для автоматизации подбора размера товара.

В целом, приложение для проката спортивного инвентаря имеет большой потенциал для успеха на растущем рынке проката спортивных товаров. Увеличение числа людей, занимающихся спортом, а также повышение популярности экологически чистых видов транспорта, таких как велосипеды и самокаты, являются благоприятными факторами для развития этого бизнеса.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Apparel Market Research Reports & Industry Analysis [Электронный ресурс]. - Режим доступа: <https://www.marketresearch.com/Consumer-Goods-c1596/Consumer-Goods-Retailing-c80/Apparel-c612/>
2. Разработка REST API [Электронный ресурс]. - Режим доступа: <https://habr.com/ru/articles/483206/>
3. Kanban: краткое знакомство [Электронный ресурс]. - Режим доступа: <https://www.atlassian.com/ru/agile/kanban>
4. Git-flow Documentation [Электронный ресурс]. - Режим доступа: <https://buildmedia.readthedocs.org/media/pdf/git-flow/latest/git-flow.pdf>
5. Continuous Integration and Continuous Delivery (CI/CD) Fundamentals [Электронный ресурс]. - Режим доступа: <https://resources.github.com/ci-cd/>
6. Документация Spring Boot [Электронный ресурс]. - Режим доступа: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#documentation.first-step>
7. REST API с использованием Spring Security и JWT [Электронный ресурс]. - Режим доступа: <https://habr.com/ru/articles/545610/>
8. Flutter documentation [Электронный ресурс]. - Режим доступа: <https://docs.flutter.dev/>