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
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

Обучающийся _____ С. А. Волченко, ст. 3 курса оч. отд.

Обучающийся _____ В. Ю. Шафоростов, ст. 3 курса оч. отд.

Обучающийся _____ А. Г. Линкина, ст. 3 курса оч. отд.

Руководитель _____ И. В. Клейменов, ассистент

Руководитель _____ В.С. Тарасов, ст. преподаватель __.__.20__

Воронеж 2023

Содержание

Введение

Считается, что спорт является неотъемлемой частью здорового образа жизни, способствующей улучшению настроения, когнитивных способностей, психологического здоровья, иммунитета и общего самочувствия человека.

Существуют виды спорта, не требующие значительного количества специализированного оборудования: бег, фитнес, йога, игры с мячом. Однако есть и такие виды спорта, которые требуют специализированного инвентаря: настольный теннис, бадминтон, хоккей, гольф, горные лыжи, сноуборд.

Для тех, кто только начинает, заниматься спортом или не может тренироваться регулярно, а также для тех, у кого нет возможности привезти свою экипировку на место занятий, существует возможность воспользоваться услугами базы проката спортивного инвентаря. Организация такой базы позволяет людям арендовать необходимое оборудование для занятий спортом, не приобретая его самостоятельно.

В рамках данной курсовой работы было разработано мобильное приложение, которое позволяет пользователям базы проката спортивного инвентаря искать товары, автоматизировать процессы проката и оплаты аренды.

1 Постановка задач

Целью данного проекта является создание мобильного приложения для аренды спортивного инвентаря.

Основными задачами проекта являются:

— онлайн бронирование инвентаря;

— онлайн продление аренды;

— выдача счета на оплату по QR-коду.

Для достижения поставленных целей необходимо:

— иметь представление о разрабатываемой системе, представленное необходимыми UML-диаграммами и разработанным дизайном веб-приложения, как в целом, так и в отдельных сценариях;

— провести обзор аналогов 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.

Также, необходимо провести тестирование системы и ее аналитику.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

Автоматизация поиска товара среди ассортимента базы:

— просмотр списка товаров, находящихся на базе;

— просмотр информации о конкретном товаре.

Автоматизация процесса проката инвентаря:

— добавление товара в корзину;

— бронирование товаров;

— продление срока аренды товаров;

— учет сроков аренды товара.

Автоматизация процесса оплаты аренды:

— формирование счета для оплаты аренды;

— учет штрафов за просрочку аренды;

— генерация QR-кода для оплаты аренды.

Изменение ассортимента базы:

— добавление и удаление товара;

— добавление и удаление категорий товаров;

— редактирование количества и размеров товаров.

1.1.2 Технические требования

Программный продукт должен обеспечить:

— авторизацию пользователей;

— хранение данных в БД.

1.2 Требования к интерфейсу

Интерфейс разрабатываемого мобильного приложения должен иметь единую цветовую палитру на всех экранах.

Все текстовые блоки должны быть четко читаемыми, а элементы управления должны быть единообразными по размеру и стилю, чтобы выделяться на фоне содержимого экранов.

Информация для пользователя должна быть представлена только в необходимом объеме и располагаться в соответствующих местах приложения.

Основные элементы управления должны быть легко заметны для пользователя.

1.3 Задачи, решаемые в процессе разработки

Перед проектом были поставлены следующие задачи:

— анализ предметной области;

— обзор аналогов;

— написание технического задания;
— описание разрабатываемой системы UML-диаграммами;
— разработка схемы БД;
— разработка функциональных возможностей мобильного приложения;
— создание макета дизайна и его реализация;
— реализация интерфейса;
— проведение тестирования;
— описание процесса разработки и результата.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Мобильное приложение – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для конкретной платформы (iOS, Android, Windows Phone и т. д.).

Клиент – это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

Сервер – выделенный или специализированный компьютер для выполнения сервисного программного обеспечения.

База данных – это упорядоченный набор структурированной информации, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД).

Аутентификация – процедура проверки подлинности, например, проверка подлинности пользователя путем сравнения введённого им пароля с паролем, сохраненным в базе данных.

Авторизация – предоставление определённому лицу или группе лиц прав на выполнение определенных действий.

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Аккаунт (учетная запись) – это персональная страница пользователя или личный кабинет, который создается после регистрации в приложении.

Пользователь – человек, который использует приложение.

QR-код – двухмерный штриховой код, предоставляющий информацию для быстрого ее распознавания с помощью камеры устройства.

Концептуальная схема – карта понятий и их отношений, используемых для баз данных.

Java – строго типизированный объектно-ориентированный язык программирования.

Spring Framework – универсальный фреймворк с открытым исходным кодом для Java-платформы.

Liquibase – продукт с открытым исходным кодом для обеспечения миграций баз данных.

Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

PostgreSQL – объектно-реляционная система управления базами данных.

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

Dart – язык программирования общего назначения от компании Google, который предназначен прежде всего для разработки прикладных приложений.

REST API (REST) – стиль архитектуры программного обеспечения для построения масштабируемых веб–приложений.

JUnit 5 – библиотека для написания юнит-тестов, в данном тесте используется для написания тестовых методов и проверки ожидаемых результатов.

Spring Test – модуль фреймворка Spring, который предоставляет инфраструктуру для тестирования приложений, основанных на Spring, в коде используются аннотации, позволяющие полноценно запустить приложение и использовать внедрение зависимостей, также аннотации для использования тестовой конфигурации приложения.

Mockito – библиотека для создания заглушек (mock objects), которые позволяют эмулировать поведение объектов в тестах.

Hamcrest – библиотека для написания удобочитаемых и гибких проверок (matchers) в тестах, в данном тесте используется для написания проверок на ожидаемые результаты.

Подход "Arrange-Act-Assert" – это стандартный подход к написанию юнит-тестов, который заключается в том, чтобы разбить тест на три части: подготовка (arrange), выполнение действия (act) и проверка результата (assert).

2.2 Обзор аналогов

2.2.1 Getski [1]

Мобильное приложение "Getski" – это сервис по прокату горнолыжного оборудования в таких местах, как Красная Поляна, Архыз, Домбай, Эльбрус и даже Москва.

"Getski" – это удобный сервис, который позволяет пользователям быстро и легко арендовать спортивный инвентарь для зимних видов спорта, таких как лыжи, сноуборды.

Интерфейс мобильного приложения приведен в соответствии с рисунком 1, рисунком 2 и рисунком 3.

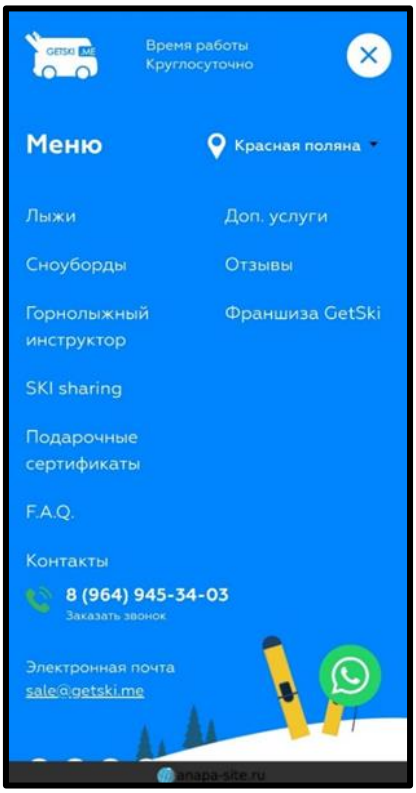

Рисунок 1 - Меню приложения Getski

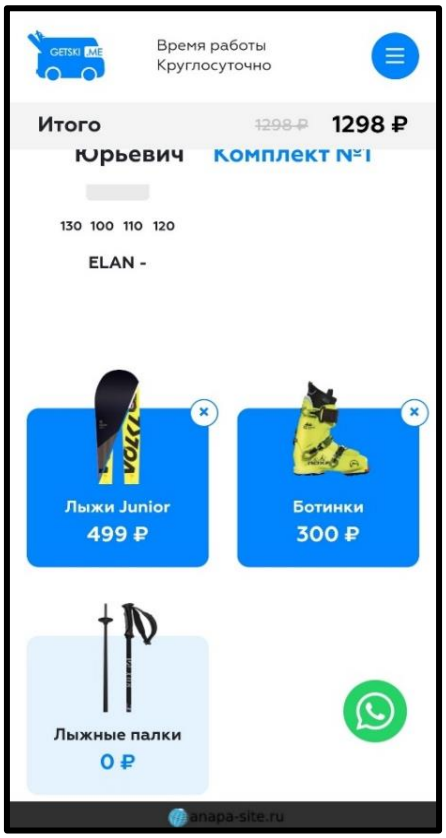

Рисунок 2 - Аренда в Getski

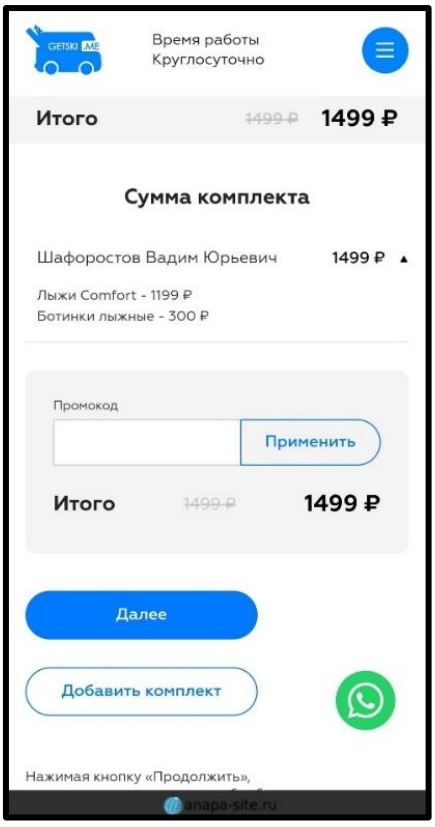

Рисунок 3 - Оплата аренды в Getski

Достоинствами приложения "Getski" были выделены:

— широкий выбор спортивного инвентаря от разных производителей и разных уровней сложности;
— возможность выбрать место и время получения и возврата инвентаря;
— удобная система оплаты через приложение;
— возможность оставить отзыв о качестве обслуживания и инвентаря;
— наличие возможности выбора параметров: рост, вес, пол, размер обуви, и подбора товаров под конкретные параметры;
— работает система скидок и купонов;
— возможность доставки, бронирования авто для самовывоза.

Недостатками приложения "Getski" были выделены:

— регистрация сразу же при запуске приложения;
— наличие ненужной информации (при бронировании лыж на одной странице предлагают арендовать авто);
— невозможность выбрать необходимое количество инвентаря для аренды (например, взять 2 пары лыж, обуви и т.д.);
— отсутствие наличия личного кабинета;
— нет возможности посмотреть остаток времени бронирования для товаров.

2.2.2 Спортмастер [2]

Мобильное приложение "Спортмастер" для аренды спортивного инвентаря – это удобный инструмент для любителей спорта, которые хотят арендовать необходимое оборудование без лишних хлопот.

Интерфейс мобильного приложения приведен в соответствии с рисунком 4, рисунком 5 и рисунком 6.

Рисунок 4 - Стартовый экран

Рисунок 5 - Стартовый экран

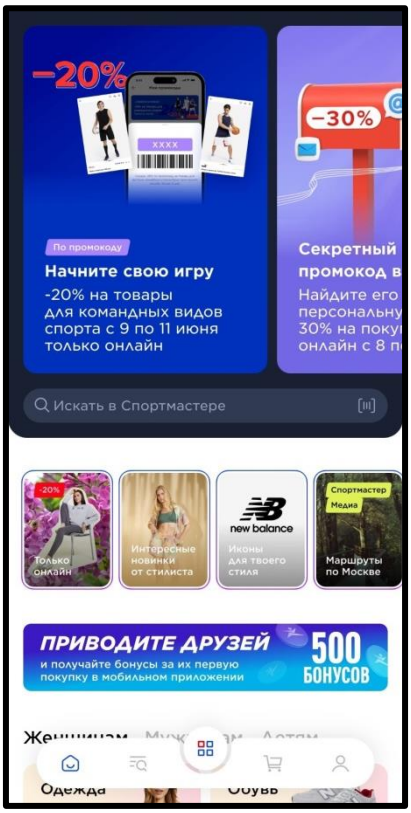

Рисунок 6 - Интерфейс приложения

Достоинствами приложения "Спортмастер" были выделены:

— широкий выбор спортивного инвентаря в каталоге;

— быстрый процесс аренды;

— наличие splash screeen’ов перед экраном регистрации;

— наличие фильтрации товаров по цене, категориям и производителям;

— возможность выбрать аренду по дням и по месяцам;

— наличие функций: доставка, возврат (отмена аренды) и защита от кражи.

Достоинства мобильного приложения приведены в соответствии с рисунком 7, рисунком 8, рисунком 9 и рисунком 10.

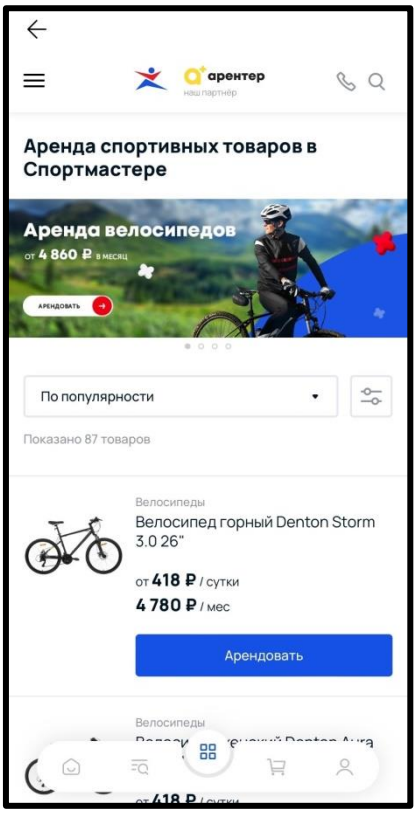

Рисунок 7 - Товары для аренды

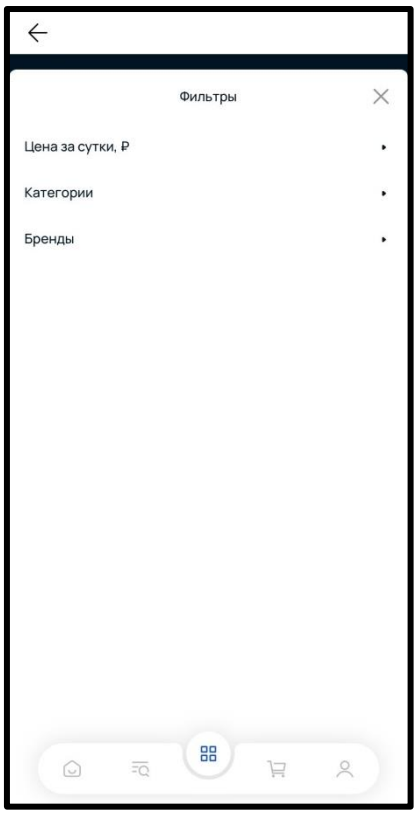

Рисунок 8 - Фильтры

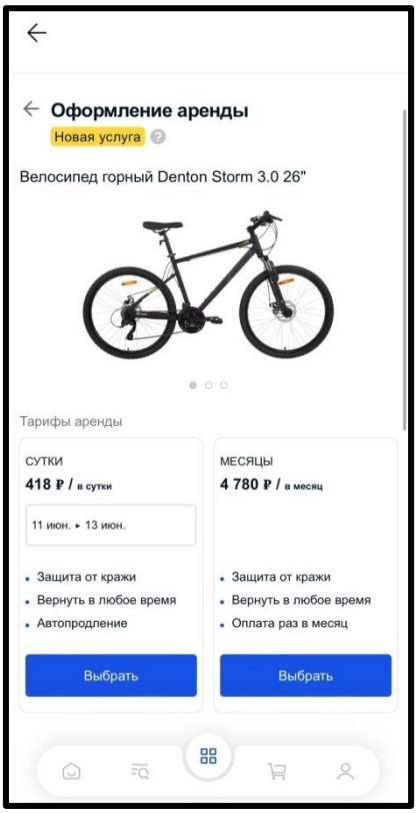

Рисунок 9 - Оформление аренды

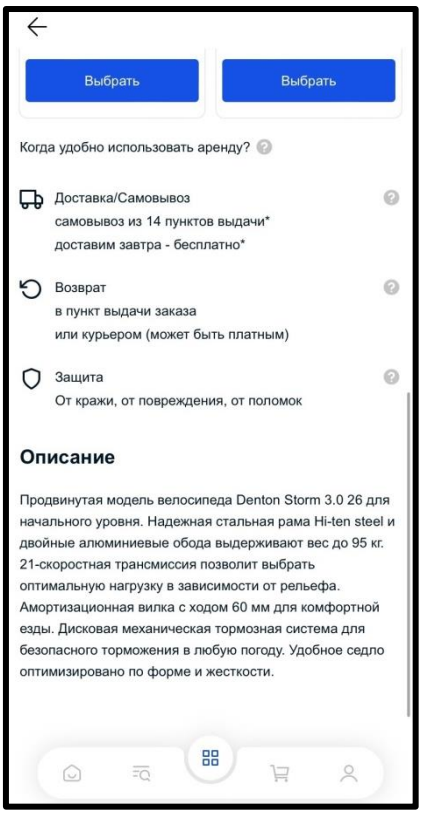

Рисунок 10 - Описание при аренде

Недостатками приложения "Спортмастер" были выделены:

— ограниченный выбор мест для аренды (приложение работает только с определенными магазинами "Спортмастер”);

— нет доступа к блоку аренды спортивного инвентаря через само приложение "Спортмастер": переход возможен только через стороннюю ссылку.

2.2.3 Спортивные города [3]

Мобильное приложение "Спортивные города" для аренды спортивного инвентаря - это онлайн-сервис, который позволяет пользователям арендовать спортивный инвентарь в различных городах.

Мобильное приложение "Спортивные города" представляет собой удобный и функциональный сервис для аренды спортивного инвентаря, который может быть полезен как для любителей спорта, так и для профессиональных спортсменов.

Интерфейс мобильного приложения приведен в соответствии с рисунком 11, рисунком 12 и рисунком 13.

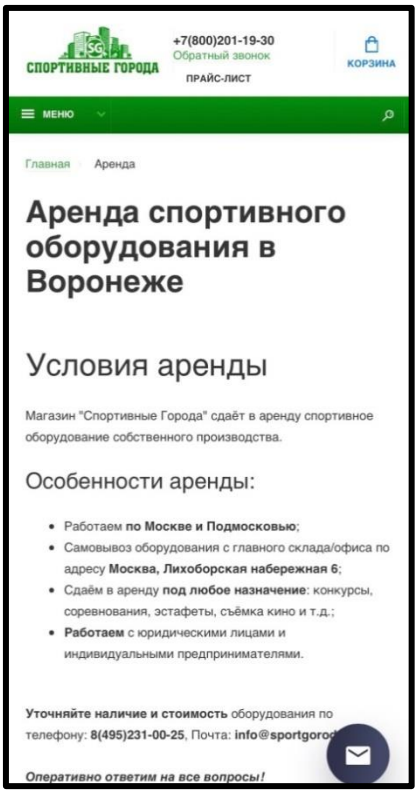

Рисунок 11 - Условия аренды

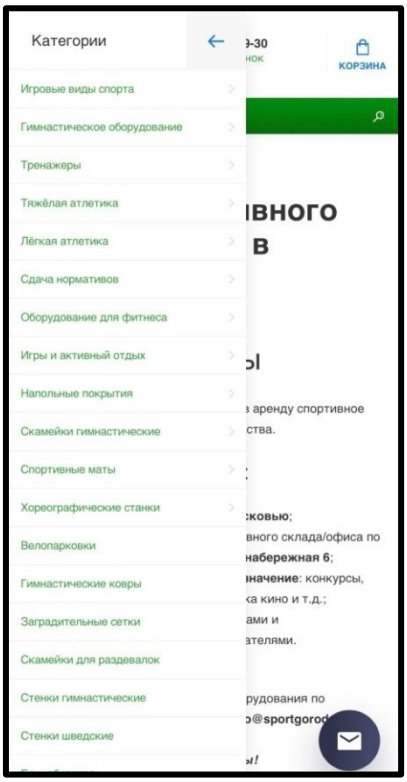

Рисунок 12 - Выбор инвентаря

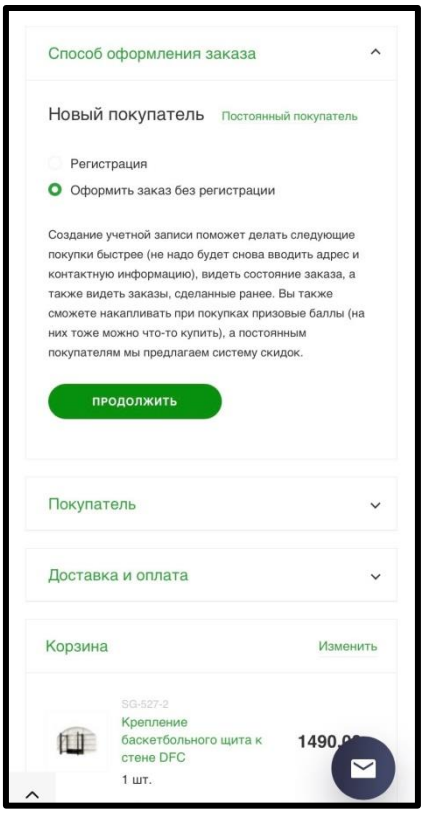

Рисунок 13 - Оформление заказа

Достоинствами приложения "Спортивные города" были выделены:

— широкий выбор спортивного инвентаря в различных городах;

— возможность быстро и удобно забронировать нужный инвентарь;

— возможность забронировать оборудования без регистрации в самом мобильном приложении

Недостатками приложения "Спортивные города" были выделены:

— малое количество информации по конкретным товарам для аренды;

— разницы между “арендой с регистрацией” и “аренды без регистрации”, как таковой, нет.

2.3 Моделирование системы

2.3.1 Диаграмма в стиле методологии IDEF0

Благодаря диаграммам в стиле методологии IDEF0 можно увидеть работу системы, когда пользователь взаимодействует с приложением на этапах процесса аренды спортивного инвентаря.

В соответствии с рисунком 14 представлена контекстная диаграмма в стиле методологии IDEF0.

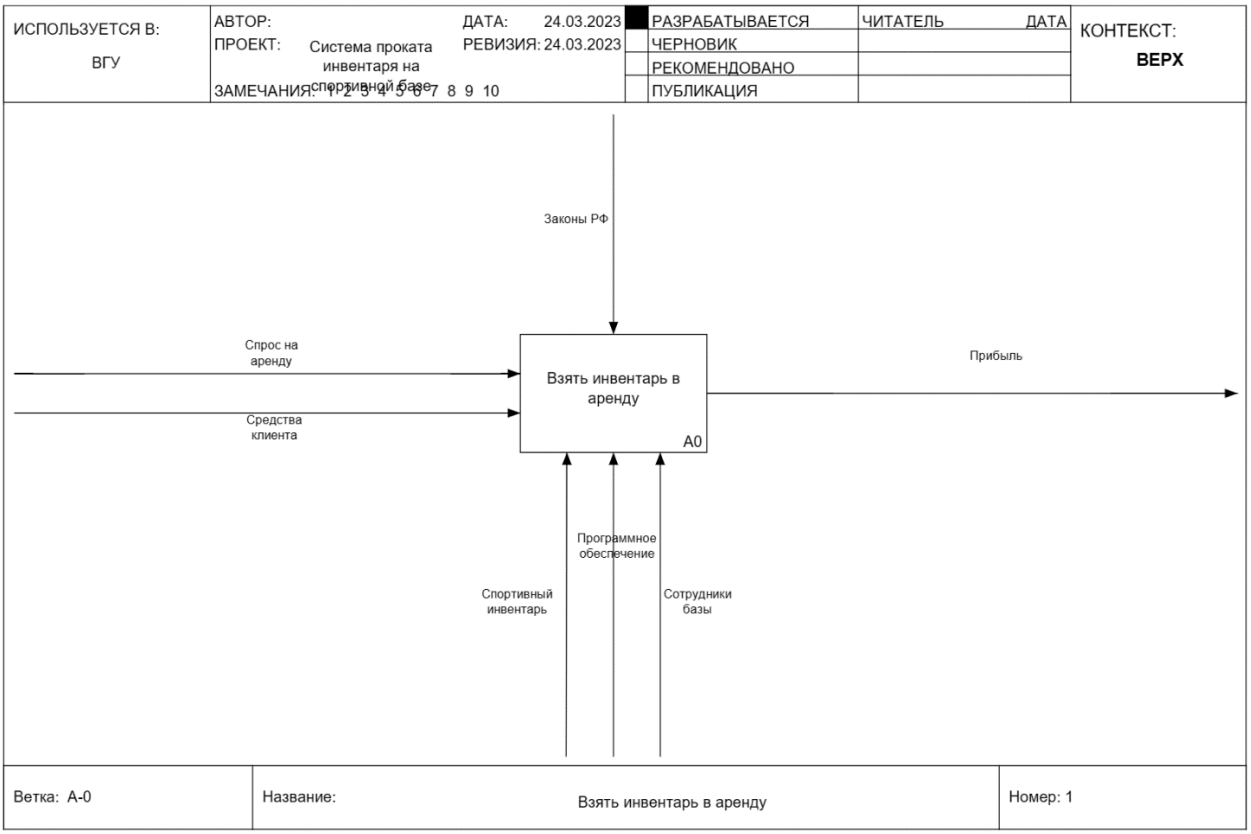

Рисунок 14 - Контекстная диаграмма

В соответствии с рисунком 15 представлена дочерняя диаграмма в стиле методологии IDEF0.

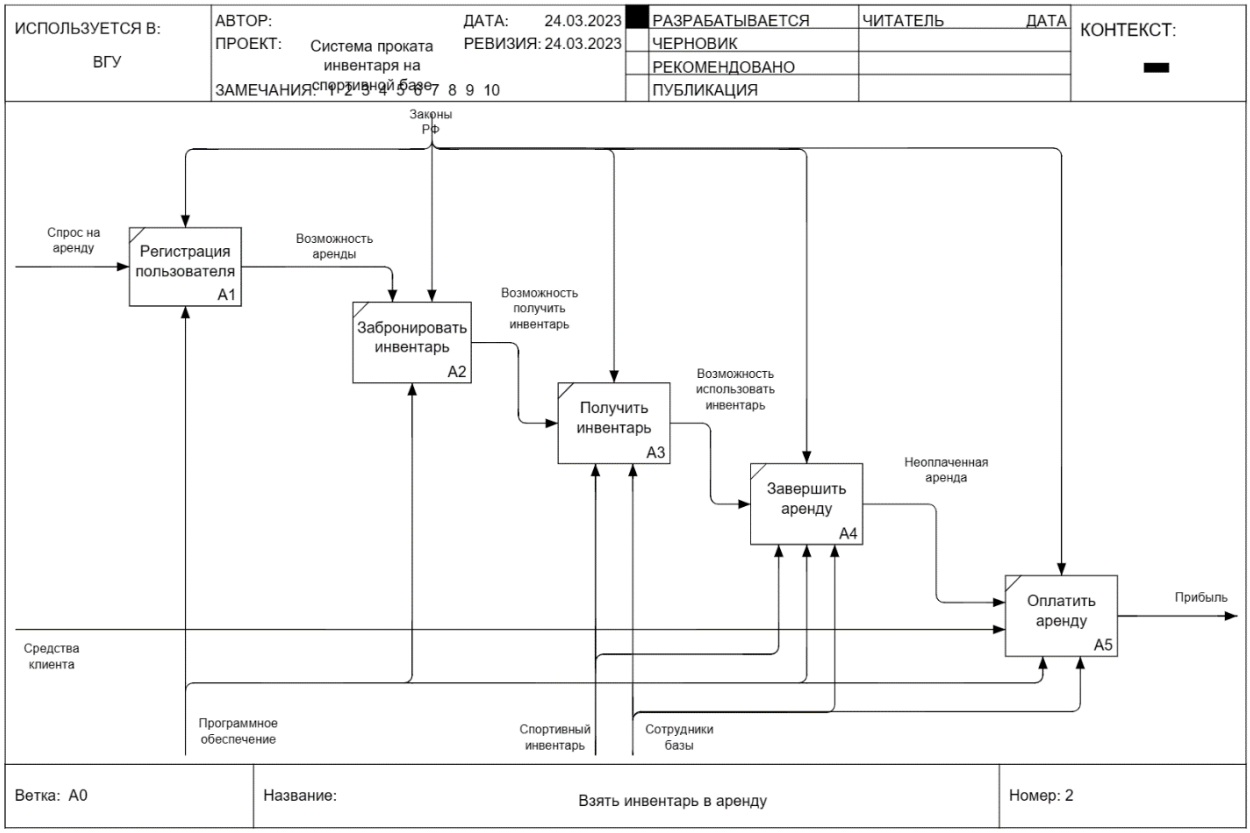

Рисунок 15 - Дочерняя диаграмма

2.3.2 Диаграмма прецедентов

Диаграмма прецедентов показывает, как пользователь взаимодействует с приложением и какие функции доступны для использования. Это помогло команде разработчиков понять требования пользователей.

Диаграмма прецедентов приведена в соответствии с рисунком 16.

Рисунок 16 - Диаграмма прецедентов

2.3.3 Диаграмма состояний

Диаграмма состояний показывает, как пользователь взаимодействует с приложением на каждом этапе процесса аренды спортивного инвентаря.

Диаграмма состояний приведена в соответствии с рисунком 17.

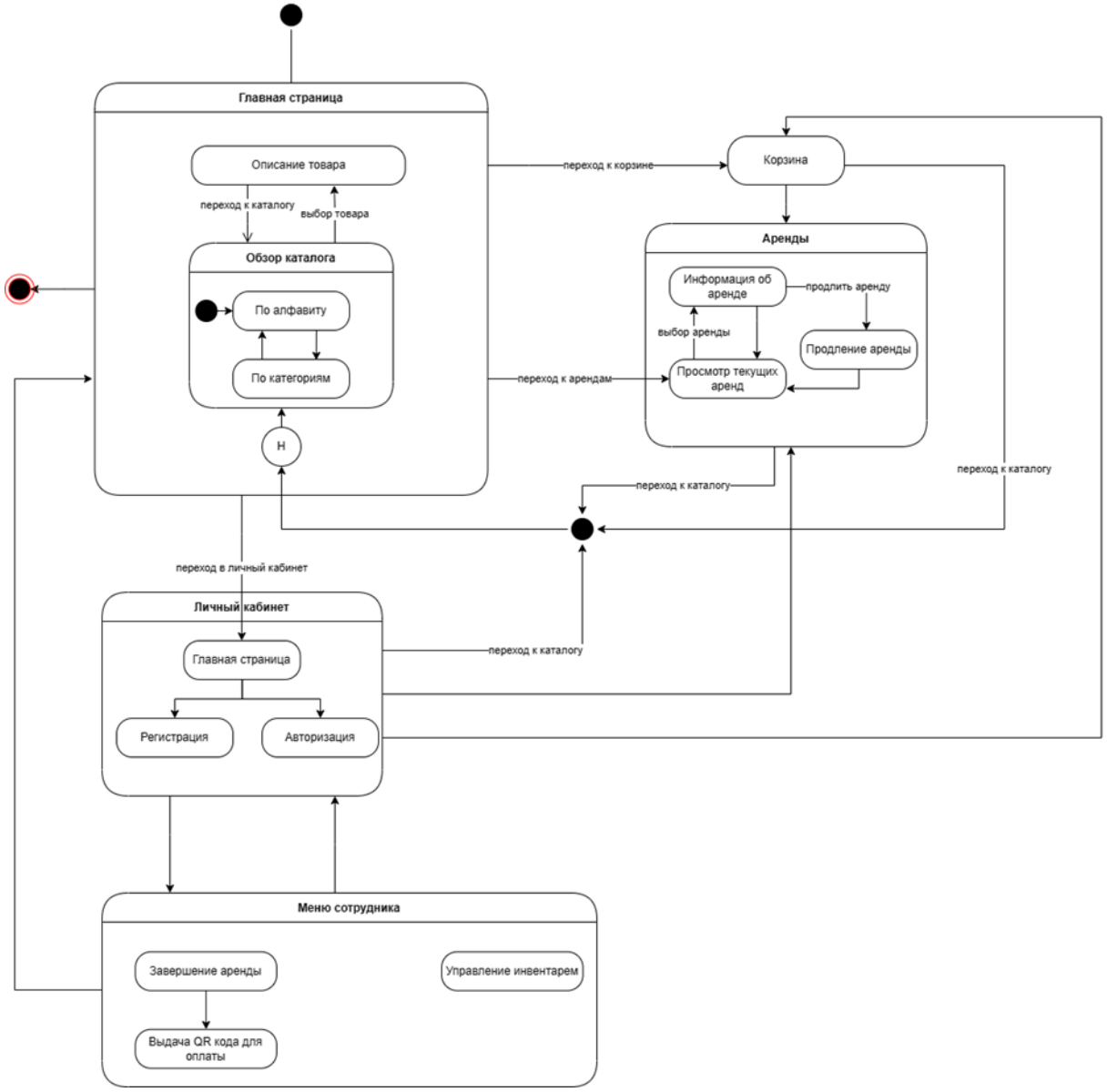

Рисунок 17 - Диаграмма состояний

2.3.4 Диаграмма последовательности

Диаграмма последовательности на единой временной оси показывает жизненный цикл системы, когда пользователь взаимодействует с приложением в процессе аренды спортивного инвентаря.

Диаграмма последовательности приведена в соответствии с рисунком 18.

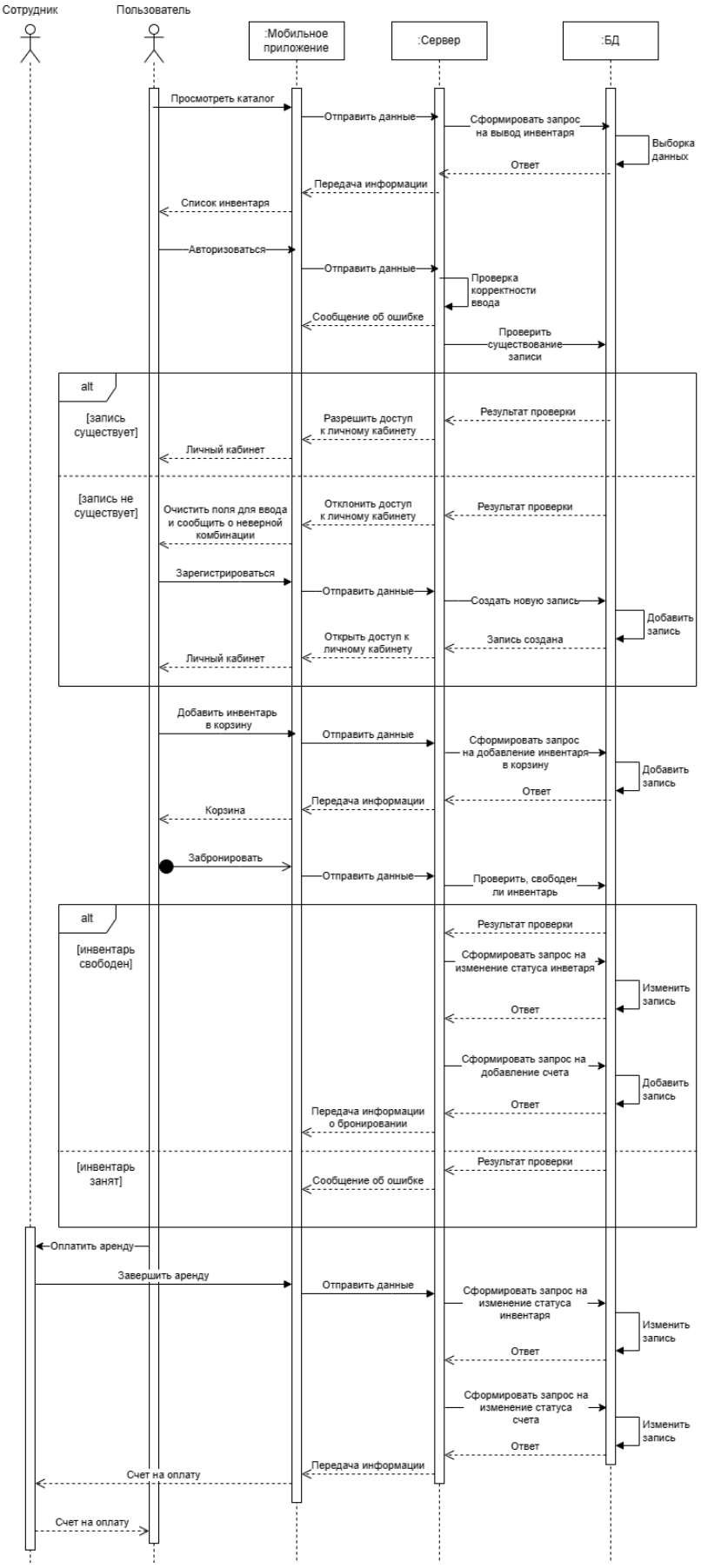

Рисунок 18 - Диаграмма последовательности. Регистрация

2.3.5 Диаграмма деятельности

Благодаря диаграмме деятельности можно увидеть действия, состояния которых описаны на диаграмме состояний во время процесса взаимодействия пользователя с приложением на этапах аренды спортивного инвентаря.

Диаграмма деятельности приведена в соответствии с рисунком 19.

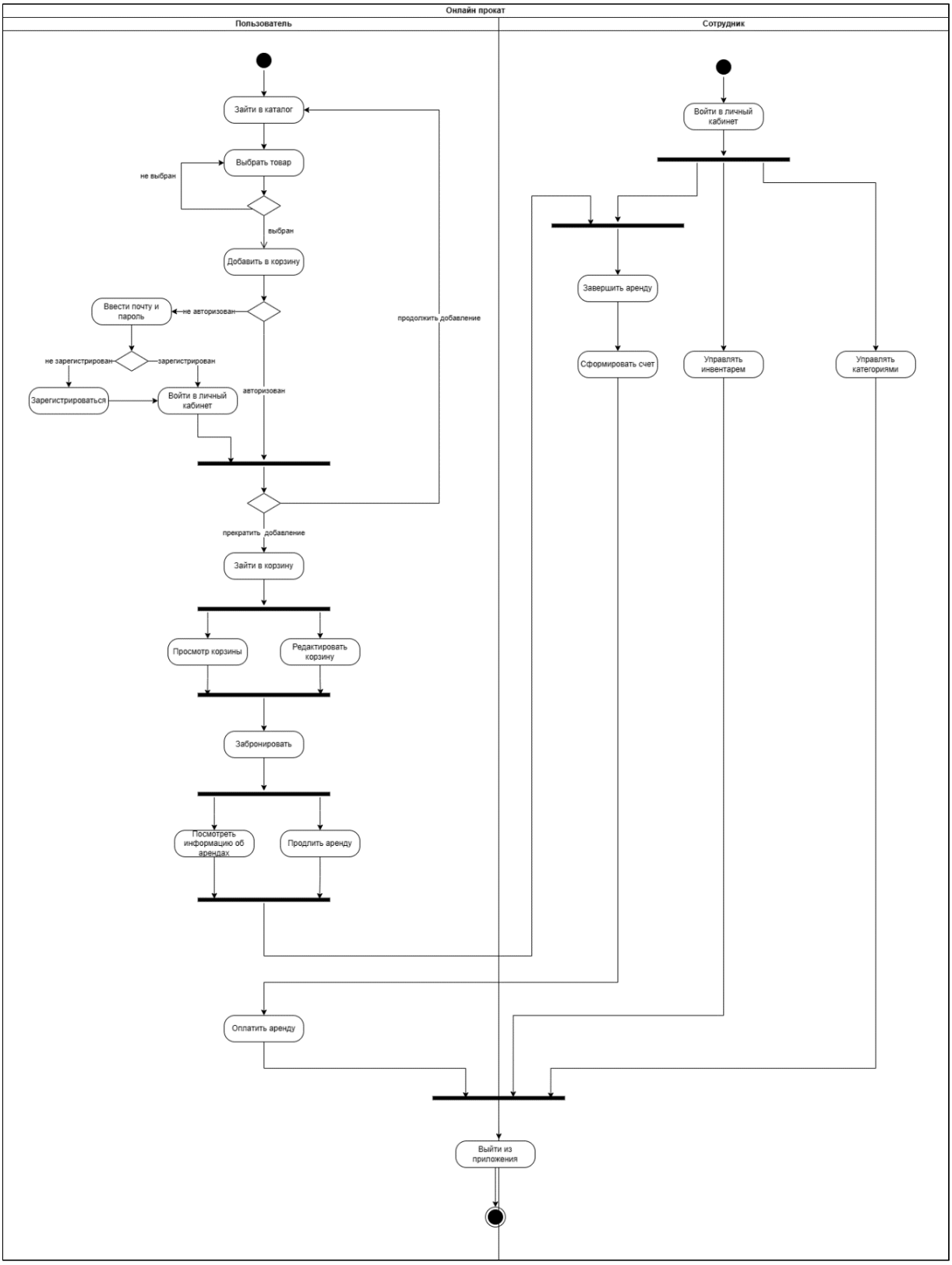

Рисунок 19 - Диаграмма деятельности

2.3.6 Диаграмма классов

ER-диаграмма позволяет описать концептуальную схему предметной области разрабатываемой системы.

ER-диаграмма приведена в соответствии с рисунком 20.

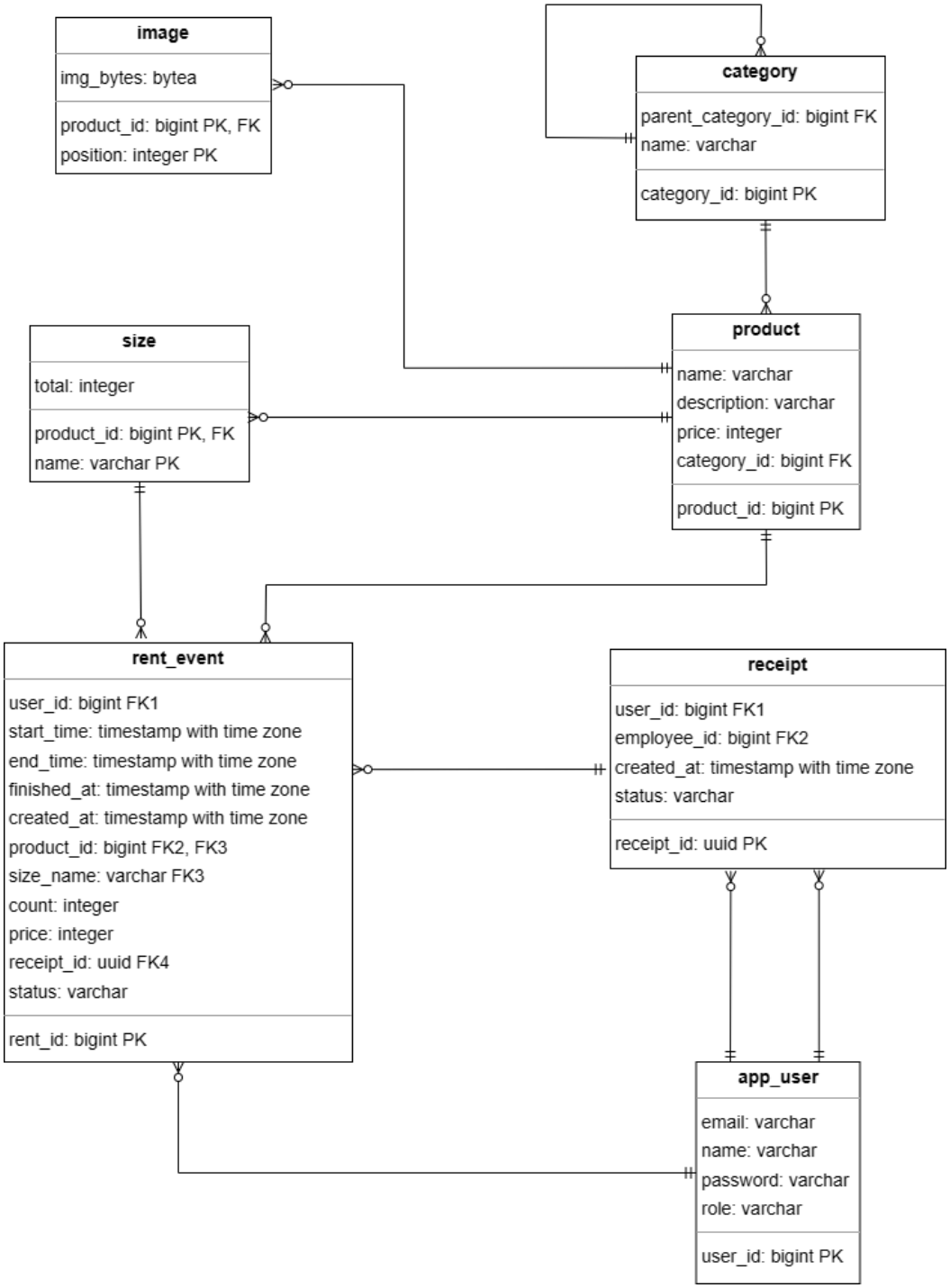

Рисунок 20 - ER-диаграмма

2.4 Аналитика веб-приложения

Для сбора метрик для приложения было выбрано AppMetrica от Яндекс. Это система для сбора данных об использовании приложения пользователями. Она отличается высокой скоростью и удобством настройки метрик для мобильных приложений. Кроме того, она предлагает интуитивный интерфейс и понятное руководство по использованию.

С помощью сервиса «AppMetrica» удалось собрать данные, которые приведены на рисунке 21 и рисунке 22.

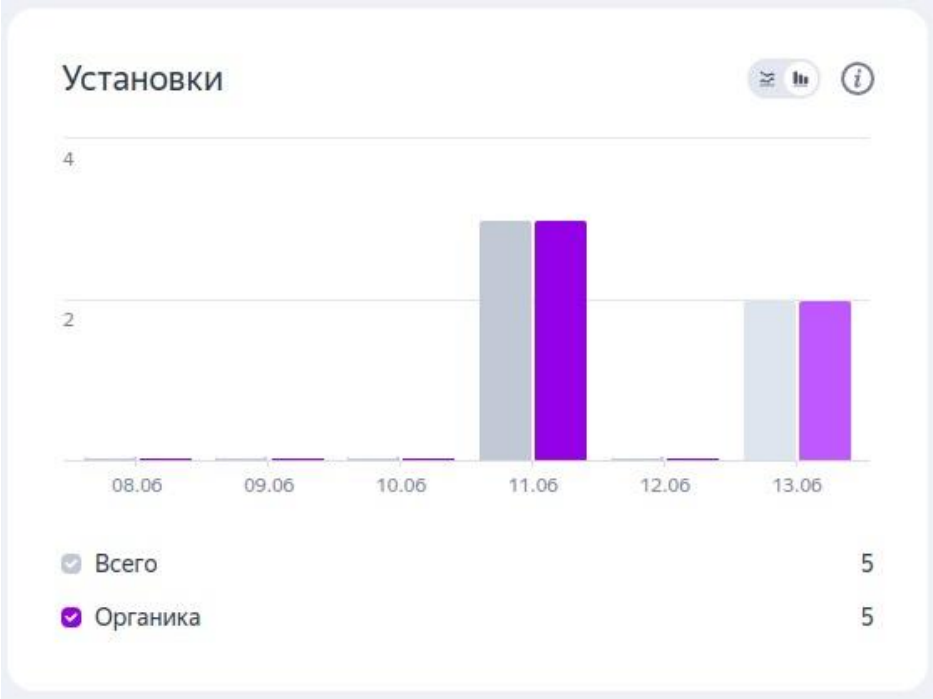

Рисунок 21 - Данные сервиса «AppMetrica»

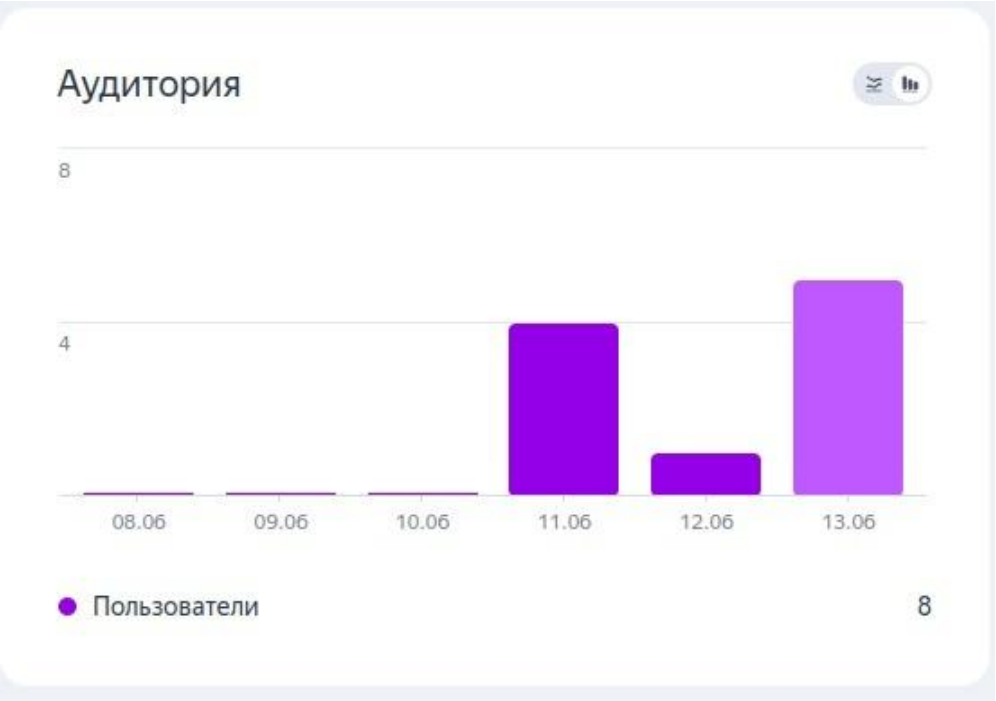

Рисунок 22 - Данные сервиса «AppMetrica»

Была составлена воронка конверсии «Аренда товара», которая позволяет увидеть использование основного сценария мобильного приложения. Результаты приведены в соответствии с рисунком 23.

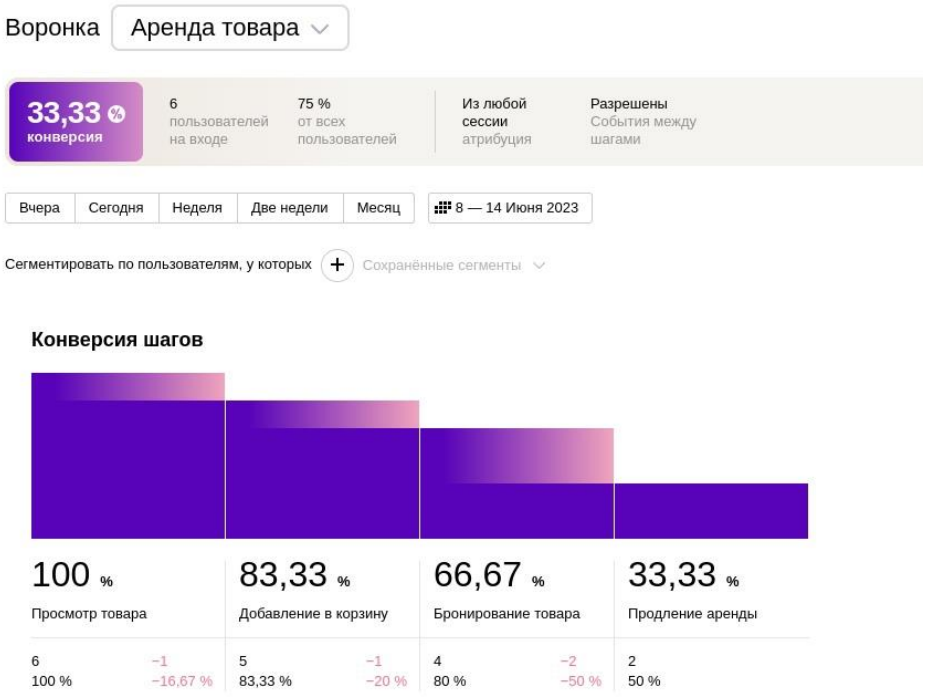

Рисунок 23 - Воронка «Аренда товара»

2.5 Границы проекта

Основными границами проекта являются:

— ограниченный бюджет на разработку и продвижение проекта;

— ограниченный выбор спортивного инвентаря на начальной стадии проекта;

— ограниченные сроки разработки и запуска проекта;

— регистрация, авторизация и аутентификация пользователей;

— просмотр доступного инвентаря, который можно арендовать;

— бронирование инвентаря пользователями на определенное время;

— оплата за аренду спортивного инвентаря посредством QR-кода.

На фоне основных сценариев и функциональных требований, можно сформулировать то, что точно выходит за основные границы проекта:

— аренда другой техники, кроме спортивного инвентаря, представленного в каталоге мобильного приложения;

— разработка и продажа спортивного инвентаря;

— физическая доставка и забор спортивного инвентаря;

— обслуживание и ремонт спортивного инвентаря;

— организация мероприятий и тренировок;

— предоставление инструкторов и тренеров;

— организация транспорта для перевозки спортивного инвентаря;

— предоставление мобильным приложением страховки на спортивный инвентарь.

3 Реализация

3.1 Средства реализации

Мобильное приложение имеет архитектуру, соответствующую шаблону клиент-серверного приложения и разделяется на back-end и клиентскую часть, связанных между собой посредством REST API.

Для реализации программно-аппаратной части были выбраны:

— язык программирования Java [4];

— фреймворк Spring [5];

— библиотека Liquibase [6];

— ПО Docker [7];

— СУБД PostgreSQL [8].

Для реализации клиентской части были выбраны:

— язык программирования Dart [9];

— фреймворк Flutter [10].

3.2 Реализация серверной части веб-приложения

Архитектура серверной части представляет собой стандартное Spring Boot приложение, где реализованы слои контроллеров, сервисов и репозиториев.

Присутствуют конфигурации безопасности и подсчета цен и штрафов за аренду спортивного инвентаря.

Бизнес-логика серверной части мобильного приложения заключается в том, чтобы пользователь не имел возможности арендовать уже занятый товар, то есть, товар, который полностью закончился в наличии на складе.

Также, присутствует внедрение Swagger, реализованное таким образом, что Swagger UI генерируется автоматически.

3.3 Реализация клиентской части веб-приложения

Архитектура клиентской части мобильного приложения основывается на BLoC-архитектуре [11], которая позволяет отделить бизнес-логику от пользовательского интерфейса.

Директория Lib хранит в себе код, описывающий сущности во всем мобильном приложении. Модуль api хранит в себе код сущностей, использующихся в приложении. Модуль request хранит в себе данные, которые отправляются на сервер. Модуль response хранит в себе данные, которые приходят с сервера.

Директория common хранит общие для всего продукта модули. В модуле routes реализуется навигация приложения, где конкретному экрану задаются маршрут и блок. В модуль service помещено хранилище для данных, которое необходимо сохранить на устройстве: в разработанном мобильном приложении сохраняются данные пользователя после его авторизации, а также товары, добавленные в корзину, для того, чтобы при закрытии мобильного приложения данные не были потеряны. Модуль utils хранит в себе общие методы, использующиеся в модуле api. Модуль values хранит в себе переменные-константы, которые используются во всем приложении и базовый URL сервера мобильного приложения. Модуль widgets хранит виджеты, которые используются во всем приложении.

Модуль repository используется для обработки данных, которые приходят с сервера или отправляются на него.

Промежуточный слой между BLoC-ом и пользовательским интерфейсом controller взаимодействует с ивентами.

Директория screens содержит данные, отображающиеся на экранах мобильных устройств.

4 Тестирование

4.1 Интеграционное тестирование

Для проведения интеграционного тестирования, которое проверяет взаимодействие между приложением и базой данных, использовалась библиотека TestContainers, которая позволяет создавать и запускать изолированные контейнеры Docker, которые используются для тестирования приложений. На время тестирования запускается контейнер PostgreSQL и данные для подключения именно к этому контейнеру используются для тестов.

Для тестирования каждого класса использовались следующие библиотеки и подходы:

— JUnit 5;

— Spring Test;

— Mockito;

— Hamcrest;

— подход "Arrange-Act-Assert".

В коде приведены примеры тестовых методов, которые проверяют различные варианты работы методов класса CatalogService. Например, метод testListAllProducts() проверяет, что метод listAllProducts() возвращает корректный список всех товаров из базы данных, включая связанные с ними сущности (например, категории). Метод testFindProductById() проверяет, что метод findProductById() возвращает корректный товар по его идентификатору.

Метод testFindProductByIdWithProductNotExists() проверяет, что метод findProductById() генерирует исключение NoSuchElementException, если товар с указанным идентификатором отсутствует в базе данных.

Все тесты запускаются при запуске команды mvn package, которая, в свою очередь запускается во время стадии build автоматизированного процесса интеграции и доставки приложения на удаленный сервер.

Результаты интеграционного тестирования приведены в соответствии с рисунком 24.

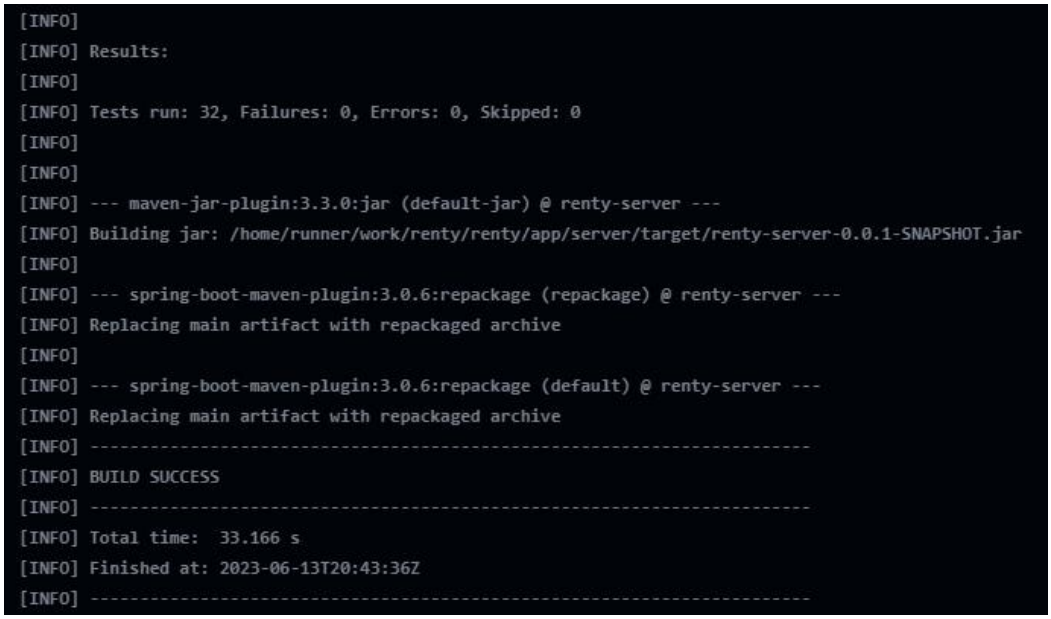

Рисунок 24 - Результаты интеграционного тестирования

4.2 UI–тестирование

UI-тестирование – это тестирование всех визуальных индикаторов и иконок, меню, переключателей, текстовых полей, флажков, панелей инструментов, цветов, шрифтов и других элементов управления и принятия решений в цифровой среде.

Мы решили остановиться на ручном UI-тестировании.

В работе были рассмотрены следующие аспекты данного тестирования:

— визуальные характеристики;

— композиция

— взаимодействие;

— доступность;

— пользовательские сценарии.

Для каждого тестирования были введены критерии успешности прохождения UI–тестирования для определенного экрана:

— 1 – не пройден;

— 2 – пройден частично;

— 3 – пройден.

Тестирование экранов основного сценария приведено в соответствии с таблицей 1.

Таблица 1 - Тестирование экранов основного сценария

Кейс	Каталог товаров (домашний экран)	Экран информации о товаре	Экран списка аренд	Экран информации об аренде
Визуальные характеристики	3	3	3	3
Композиция	3	3	3	3
Взаимодействие	3	2	3	3
Доступность	3	3	3	3
Пользовательский сценарий	3	3	3	3

Тестирования экранов для сценария управления инвентарем сотрудником базы приведено в соответствии с таблицей 2 и таблицей 3.

Таблица 2 - Тестирования экранов для сценария управления инвентарем

Кейс	Экран с инвентарем	Экран редактирования количества товаров	Экран добавления нового размера
Визуальные характеристики	3	2	3
Композиция	3	3	3
Взаимодействие	3	3	3
Доступность	3	3	3
Пользовательский сценарий	3	3	3

Таблица 3 - Тестирования экранов для сценария управления инвентарем

Кейс	Экран добавления нового товара	Экран категорий	Экран добавления новой категории
Визуальные характеристики	3	3	3
Композиция	3	3	3
Взаимодействие	3	3	3
Доступность	3	3	3
Пользовательский сценарий	3	3	3

Тестирование экранов для завершения аренды сотрудником приведено в соответствии с таблицей 4.

Таблица 4 - Тестирование экранов для завершения аренды сотрудником

Кейс	Экран поиска по email	Экран со списком найденных аренд	Экран оплаты аренды
Визуальные характеристики	3	3	3
Композиция	3	3	3
Взаимодействие	3	3	3
Доступность	3	3	3
Пользовательский сценарий	3	3	3

Тестирование экранов авторизации, регистрации и личного кабинета приведено в соответствии с таблицей 5.

Таблица 5 - Тестирование экранов авторизации, регистрации и личного кабинета

Кейс	Экран авторизации	Экран регистрации	Экран личного кабинета
Визуальные характеристики	3	3	3
Композиция	3	3	3
Взаимодействие	3	3	3
Доступность	3	3	3
Пользовательский сценарий	3	3	3

Заключение

В рамках данной курсовой работы было разработано мобильное приложение, которое позволяет пользователям базы проката спортивного инвентаря искать товары, автоматизировать процессы проката и оплаты аренды.

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе, определены основные сценарии мобильного приложения и пользовательские истории.

По результатам разработки проводился ряд тестов с целью проверки работоспособности системы.

В процессе работы были реализованы следующие задачи проекта:

— просмотр списка товаров, находящихся на базе;

— просмотр информации о конкретном товаре;

— добавление товара в корзину;

— бронирование товаров;

— продление срока аренды товаров;

— уведомление об освобождении занятого товара;

— учет сроков аренды товара;

— формирование счета для оплаты аренды;

— учет штрафов за просрочку аренды;

— генерация QR-кода для оплаты аренды;

— добавление и удаление товара;

— добавление и удаление категорий товаров;

— редактирование количества и размеров товаров.

Список используемых источников

1. Getski [Электронный ресурс]. Режим доступа: https://getski.me/ - Заглавие с экрана. (Дата обращения: 24.03.2023).
2. Спортмастер [Электронный ресурс]. Режим доступа: https://arenter.sportmaster.ru/ - Заглавие с экрана. (Дата обращения: 24.03.2023).
3. Спортивные города [Электронный ресурс]. Режим доступа: https://voronezh.sportgoroda.ru/аренда.html - Заглавие с экрана. (Дата обращения: 24.03.2023).
4. Документация языка программирования Java [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javame/8.3/index.html
5. Документация фреймворка Spring Boot [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-boot
6. Документация библиотеки Liquibase [Электронный ресурс]. – Режим доступа: https://docs.liquibase.com/home.html
7. Документация библиотеки Docker [Электронный ресурс]. – Режим доступа: https://docs.docker.com/get-started/
8. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.
9. Документация языка программирования Dart [Электронный ресурс]. – Режим доступа: https://dart.dev/guides
10. Документация фреймворка Flutter [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/
11. Документация BLoC-архитектуры [Электронный ресурс]. – Режим доступа: https://inostudio.com/blog/articles-develop/razdelenie-biznes-logiki-i-ui-vo-flutter-s-pomoshchyu-bloc-arkhitektury/
12. ГОСТ 7.32-2001 [Электронный ресурс]. Режим доступа:https://kpfu.ru/portal/docs/F1867381138/gost7_32_2001.pdf