

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

Факультет компьютерных наук
Кафедра программирования и информационных технологий

Курсовая работа
Разработка мобильного приложения для индивидуальных домашних
тренировок «FitHub»

09.03.02 Информационные системы и технологии
Программирование и информационные технологии

Зав. кафедрой _____ Махортов С.Д., д.ф.-м.н., профессор
Обучающийся _____ Таранцов И.Ю., 3 курс, д/о
Обучающийся _____ Тернавский Д.О., 3 курс, д/о
Обучающийся _____ Путилин М.Д., 3 курс, д/о
Руководитель _____ Тарасов В.С., ст. преподаватель

Воронеж 2024

СОДЕРЖАНИЕ

Обозначения, определения и сокращения	5
Введение.....	7
1 Постановка задачи.....	9
1.1 Цели создания приложения.....	9
1.2 Задачи приложения	9
1.3 Требования к приложению.....	10
1.3.1 Требования к приложению в целом	10
1.3.2 Требования к функциям (задачам), выполняемым приложением	10
1.3.3 Требования к структуре	11
1.3.4 Требования к программному обеспечению	12
1.3.5 Требования к оформлению и верстке страниц	14
1.3.6 Требования к защите информации.....	15
1.4 Задачи, решаемые в процессе разработки	15
2 Анализ предметной области	17
2.1 Обзор аналогов	17
2.1.1 Freeletics: Fitness Workouts.....	17
2.1.2 PUMATRAC Run, Train, Fitness	18
2.1.3 Fitness Online	20
2.1.4 Strong Workout Tracker Gym Log.....	23
2.1.5 Nike Training Club	26
3 Реализация.....	30
3.1 Средства реализации.....	30
3.2 Реализация базы данных	30
3.3 Реализация клиентской части	31
3.3.1 Загрузочный экран	31
3.3.2 Приветственные экраны.....	32
3.3.3 Экраны личной информации	33
3.3.4 Главная страница	35
3.3.5 Страница списка курсов	37
3.3.6 Страница курса.....	38
3.3.7 Страница тренировки	38

3.3.8 Страницы упражнения	39
3.3.9 Страница пройденной тренировки.....	41
3.3.10 Прогресс.....	42
3.3.11 Комьюнити	43
3.3.12 Фильтрация.....	44
3.3.13 Страница пользовательского курса.....	45
3.3.14 Страница комментариев.....	46
3.3.15 Страница создания комментария	48
3.3.16 Страница создания жалобы	48
3.3.17 Профиль	49
3.3.18 Страница настроек	52
3.3.19 Вход/регистрация.....	53
3.3.20 Страница восстановления пароля	55
3.3.21 Создание курса.....	57
3.4 Описание архитектуры клиентской части.....	62
3.4.1 Слой моделей.....	62
3.4.2 Слой сервисов.....	64
3.4.3 Слой репозитория.....	65
3.4.4 Слой представления.....	66
3.5 Серверная часть.....	67
3.5.1 Архитектура серверной части приложения	68
3.5.2 Структура серверной части приложения.....	69
4 Моделирование системы	72
4.1 Диаграмма прецедентов	72
4.2 Диаграмма прецедентов	73
4.3 Диаграмма развертывания	74
4.4 Диаграммы состояния.....	74
4.5 ER-диаграмма	75
4.6 Диаграммы активности.....	76
5 Тестирование	78
5.1 Unit-тестирование	78
5.2 Тест планы	79

6 Аналитика	81
Заключение	83
Список используемых источников.....	85

Обозначения, определения и сокращения

В настоящем отчете о ВКР применяют следующие термины с соответствующими определениями:

Мобильное приложение – программное изделие, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных (портативных, переносных, карманных) устройствах [1].

Frontend – презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты [2].

Клиент (клиентская сторона) – аппаратный или программный компонент вычислительной системы, посылающий запросы серверу [3].

Backend – Логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя [4].

Сервер (серверная сторона) – программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам [5].

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта [6].

Dart – доступный, портативный и продуктивный язык для создания высококачественных приложений на любой платформе [7].

Flutter – это фреймворк с открытым исходным кодом от Google для создания красивых, изначально скомпилированных мультиплатформенных приложений на основе единой кодовой базы [8].

API – программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими [9].

JSON Web Token – Открытый стандарт для создания токенов доступа, основанный на формате JSON [16].

Docker – Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений [15].

Python – Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью [14].

PostgreSQL – Реляционная база данных с открытым кодом [13].

GitHub – Веб-сервис для хостинга IT-проектов и их совместной разработки [12].

Введение

Современный ритм жизни часто оставляет мало времени для систематических посещений спортзала, однако значимость регулярных физических упражнений для поддержания здоровья и благополучия человека неоспорима. В условиях ограниченного времени и возможностей, все большее число людей обращаются к домашним тренировкам как альтернативному способу сохранения физической активности. В этом контексте, наличие хорошо структурированной и доступной платформы для создания и деления тренировочных программ становится крайне важным. Томас Джефферсон когда-то сказал: «Физические упражнения и спорт — это неделимые части здоровья».

Развитие приложений, таких как FitHub, приходит на помощь тем, кто ищет удобный и эффективный способ планирования и отслеживания индивидуальных тренировок. Создание персональных тренировочных курсов и возможность делиться ими с сообществом предоставляет пользователю не только мотивацию, но и необходимую поддержку со стороны других участников. Особенно это важно в городах, где ограничен доступ к спортивным учреждениям, или для людей, у которых нет времени на регулярные посещения фитнес-центров.

Современные технологии позволяют создать условия, при которых пользователь может не только следить за своими успехами через удобный интерфейс, но и корректировать свои программы в зависимости от получаемых результатов. Такая систематизация подходит для тех, кто стремится к максимальной эффективности своих усилий.

Тем не менее, отсутствие унифицированного приложения значительно усложняет процесс организации и выполнения тренировок, а также взаимодействие внутри фитнес-сообществ. Поиск нужной информации в различных источниках может стать настоящей проблемой для тех, кто предпочитает систематизированный подход к тренировкам.

Актуальность обусловлена необходимостью автоматизации процесса создания и управления персональными тренировочными программами и повышения удобства функционирования платформы для домашних тренировок. На текущий момент существует проблема распределения и доступности качественных ресурсов для самостоятельных тренировок, что осложняет поддержание регулярной физической активности у многих людей. Приложение FitHub позволит пользователям не только создавать и модифицировать тренировочные планы, но и делиться своими успехами и находками с широким сообществом, что способствует формированию здоровых привычек и поддержанию мотивации.

Целью курсовой работы является разработка мобильного приложения FitHub для организации и выполнения домашних тренировок. Это приложение предложит пользователям инструменты для создания индивидуальных тренировочных планов, отслеживания прогресса и обмена опытом с другими участниками сообщества. Основная задача — обеспечить легкость в использовании и функциональность, позволяющую пользователям эффективно управлять своими физическими упражнениями, адаптируя их под личные цели и возможности.

1 Постановка задачи

1.1 Цели создания приложения

Целями создания приложения являются:

- Реализация приложения, которое позволит пользователям тренироваться дома, проходя специальные курсы, а заказчику получать прибыль от приобретения пользователями премиум подписок;
- Создание комьюнити, позволяющее пользователям выкладывать свои созданные курсы и проходить, комментировать, оценивать курсы других пользователей.

1.2 Задачи приложения

Приложение позволяет решать следующие задачи:

- Просматривать и проходить базовые тренировочные курсы разного уровня сложности;
- Просматривать и проходить уникальные тренировочные курсы, посредством покупки премиума;
- Просматривать и проходить созданные другими пользователями тренировочные курсы;
- Оценивать и комментировать тренировочные курсы других пользователей;
- Просматривать свой прогресс;
- Создавать свои уникальные тренировочные курсы;
- Осуществлять редактирование данных своего аккаунта после регистрации или авторизации.

1.3 Требования к приложению

1.3.1 Требования к приложению в целом

Данное приложение должно удовлетворять следующим основным требованиям:

- приложение должно корректно работать на устройствах, работающих на операционной системе Android 8.0 и новее;
- реализовывать все поставленные задачи.

1.3.2 Требования к функциям (задачам), выполняемым приложением

Разрабатываемое приложение должно соответствовать следующим функциональным требованиям:

Неавторизованный пользователь должен обладать возможностью:

- Просматривать и проходить базовые тренировочные курсы;
- Просматривать и проходить рекомендуемые тренировочные курсы;
- Сортировать список тренировочных курсов;
- Указать информацию о себе;
- Авторизоваться/зарегистрироваться в приложении;
- Посмотреть политику приватности;
- Включить/выключить уведомления;
- Связаться с нами.

Авторизованный пользователь должен обладать возможностями неавторизованного, а также:

- Просматривать список всех тренировочных курсов;
- Проходить все (кроме заблокированных) тренировочные курсы;

- Просматривать и проходить тренировочные курсы, созданные другими пользователями;
- Оценивать и комментировать пользовательские курсы;
- Сортировать пользовательские курсы;
- Отправлять жалобы на пользовательские курсы;
- Просматривать и продолжать выполнение активных тренировочных курсов;
- Просматривать и проходить рекомендуемые тренировочные курсы;
- Создавать собственные уникальные тренировочные курсы;
- Делиться/не делиться созданными курсами с комьюнити;
- Редактировать собственные тренировочные курсы;
- Удалять собственные тренировочные курсы;
- Приобрести премиум и проходить заблокированные курсы;
- Редактировать свой профиль;
- Восстановить пароль;
- Выйти из профиля.

1.3.3 Требования к структуре

Для Frontend:

Приложение должно быть реализовано в соответствии с подходом MVVM (Model – View – ViewModel). MVVM — это паттерн разработки, позволяющий разделить приложение на три функциональные части:

- Model – основная логика программы;
- View – вид или представление (пользовательский интерфейс);
- ViewModel – модель представления, которая служит прослойкой между View и Model.

Шаблон MVVM помогает четко отделять бизнес-логику приложения и логику презентации от пользовательского интерфейса. Поддержание четкого разделения между логикой приложения и пользовательским интерфейсом помогает решить многочисленные проблемы разработки и упрощает тестирование, обслуживание и развитие приложения.

Для Backend:

Приложение должно быть реализовано в соответствии с подходом MVT (Model – View – Template) — паттерн разработки, разделяющий архитектуру приложения на три модуля: модель (Model), представление или вид (View), шаблон (Template).

— Model – этот компонент отвечает за бизнес-логику и данные приложения. Модели представляют собой структуры данных, которые определяют, как данные должны быть организованы и как они взаимодействуют друг с другом;

— View – отвечает за отображение данных пользователю. В контексте MVT, виды обрабатывают запросы от пользователя и возвращают ответы, используя данные, полученные от моделей. Виды могут быть представлены в виде функций или классов, которые определяют, какие данные должны быть отображены и как;

— Template – используется для определения структуры и формата представления данных. Шаблоны позволяют разделить логику представления данных от бизнес-логики и логики приложения, обеспечивая гибкость и повторное использование кода. Шаблоны могут включать в себя HTML, CSS и JavaScript для создания пользовательского интерфейса.

1.3.4 Требования к программному обеспечению

Для реализации серверной части приложения будут использоваться следующие средства:

- Язык программирования Python 12 версия;
- Фреймворк Django Rest Framework;
- СУБД PostgreSQL;
- Инструмент для создания документации API Swagger.

Для реализации клиентской части приложения будут использоваться следующие средства:

- Язык программирования Dart версия 2.19.0;
- Flutter SDK версия 3.7.6.

Для развертывания приложения будут использоваться следующие средства:

- Docker для автоматизации развертывания приложения;
- Nginx – прокси-сервер с поддержкой SSL;
- uWSGI – сервер для запуска веб-приложений на Python.

Инструменты для ведения документации:

- Miro – платформа для совместной работы распределенных команд;
- Swagger – это фреймворк для спецификации REST API;
- Draw.io – Бесплатное кроссплатформенное программное обеспечение для рисования графиков с открытым исходным кодом. Его интерфейс можно использовать для создания диаграмм, таких как блок-схемы, каркасы, диаграммы UML;
- Figma – онлайн-сервис для дизайнеров, веб-разработчиков и маркетологов. Он предназначен для создания прототипов сайтов или приложений, иллюстраций и векторной графики.

Дополнительный инструментарий:

- Git – распределённая система управления версиями;
- GitHub – платформа разработки программного обеспечения с открытым исходным кодом, представляющая систему управления репозиториями программного кода для Git;

— Asana – визуальный инструмент, обеспечивающий эффективность командной работы на любом проекте.

В качестве преимуществ выбранных технологий можно отметить следующее:

Для Python и Django:

- Готовые решения для реализации RESTful архитектуры;
- Удобные инструменты для работы с PostgreSQL
- Готовые встроенные серверы (Tomcat), обеспечивающие ускоренное и более продуктивное развертывание приложений.

Для PostgreSQL:

- Функциональность;
- Высокая надежность и производительность;
- Бесплатное и открытое ПО.

Для Flutter:

- Мультиплатформенность;
- Понятная и полная документация;

Возможность быстро проектировать мобильные приложения.

1.3.5 Требования к оформлению и верстке страниц

Оформление и верстка страниц должны удовлетворять следующим требованиям:

- приложение должно быть оформлено в едином стиле;
- должно быть разработанное название, присутствующее в оформлении страниц;
- приложение должно быть разработано в одной цветовой палитре с использованием ограниченного набора шрифтов;
- цветовая палитра должна быть контрастной;

— необходимо корректное и одинаковое отображение страниц на экранах различного размера.

1.3.6 Требования к защите информации

Для защиты информации будут использоваться JWT. Даже если злоумышленник получит этот токен, с помощью которого он может получить доступ ко всем функциям приложения, через заданное количество времени токен не будет действителен (обычно это от 2 до 10 минут) и ему придется вылавливать новый.

1.4 Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

- Анализ предметной области;
- Обзор аналогов;
- Постановка задачи;
- Создание репозитория GitHub и доски в Asana;
- Разработка требований: к приложению в целом, к функциям, к структуре, к программному обеспечению, к оформлению и верстке страниц, к защите информации;
- Создание диаграмм: use case, состояний, активностей, последовательностей, IDEF0, сотрудничества, ER, классов, объектов, развертывания.
- Разработка дизайна приложения;
- Написание технического задания в соответствии с ГОСТ 34.602 – 2020;
- Реализация интерфейса приложения;
- Реализация серверной части приложения;

- Развертывание приложения;
- Написание курсовой работы.

2 Анализ предметной области

2.1 Обзор аналогов

На рынке есть достаточно много приложений для домашних индивидуальных тренировок. Мы выберем наиболее схожие по функционалу. Далее мы рассмотрим их достоинства и недостатки, чтобы разработать функционал приложения, основываясь на том, чего не хватает пользователю в существующих решениях.

2.1.1 Freeletics: Fitness Workouts

«Freeletics: Fitness Workouts» — мобильное приложение для планирования и выполнения индивидуальных тренировок. Эта платформа позволяет пользователям проходить тренировки, следить за своим прогрессом. Приложение включает в себя разнообразные функции для поддержания мотивации и эффективности тренировок. Но главной особенностью является ИИ-тренер, который может создавать персональные тренировки.

Достоинства:

- Использование ИИ-тренера для создания индивидуальных тренировок;
- Встроенные инструменты для мониторинга физических показателей и прогресса;
- Большая база упражнений с подробными инструкциями и видео-гидами;
- Поддержка приложения на различных мобильных платформах, включая Android и iOS.

Недостатки:

- Полноценное использование приложения, включая многие персональные тренировки и аналитику, требует покупки подписки;

- Отсутствует возможность взаимодействия с тренером или получения профессиональной обратной связи в стандартной версии;
- Интерфейс может оказаться сложным для новичков из-за большого количества функций и настроек;
- Недостаток социальных функций для общения и соревнований с другими пользователями, что могло бы повысить мотивацию.

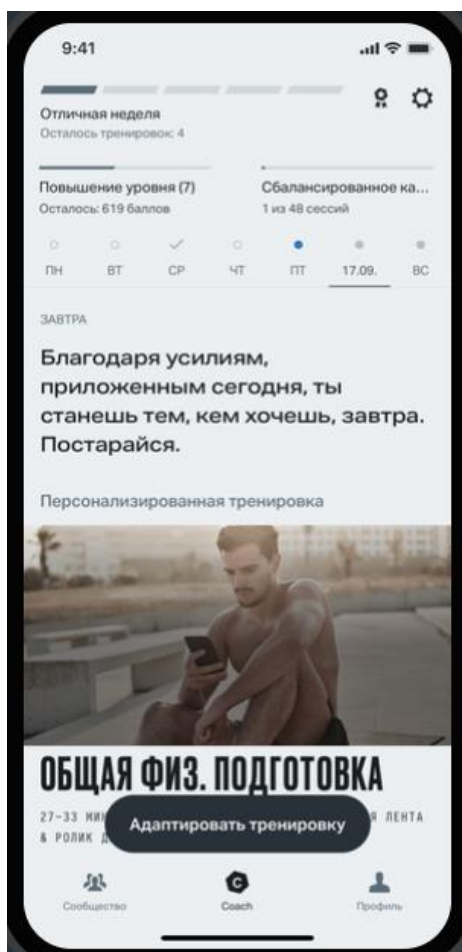


Рисунок 1 - Главная страница в приложении Freeletics

2.1.2 PUMATRAC Run, Train, Fitness

«PUMATRAC» — это мобильное приложение, разработанное компанией PUMA для фитнеса и тренировок, которое предоставляет пользователям уникальные и адаптивные тренировочные планы, основываясь на их личных данных и предпочтениях. Приложение включает в себя

различные виды фитнес-активностей, такие как йога, бег, силовые тренировки и многое другое. Оно также использует данные о погоде и времени суток для предложения оптимальных тренировок, адаптированных под текущие условия и настроение пользователя.

Достоинства:

- Аудио забеги;
- Возможность поделиться своими достижениями;
- Тренеры мирового класса;
- Испытания для пользователей.

Недостатки:

— Ограниченная интеграция с другими фитнес-приложениями и устройствами, что может быть неудобно для пользователей, уже использующих другие платформы для отслеживания своих тренировок и здоровья.

— Отсутствие персонализированной обратной связи от тренеров или экспертов, что может ограничить эффективность тренировок для более продвинутых пользователей, ищущих профессиональное руководство.

— Интерфейс приложения может показаться перегруженным новичкам, что усложняет процесс адаптации к многочисленным функциям.

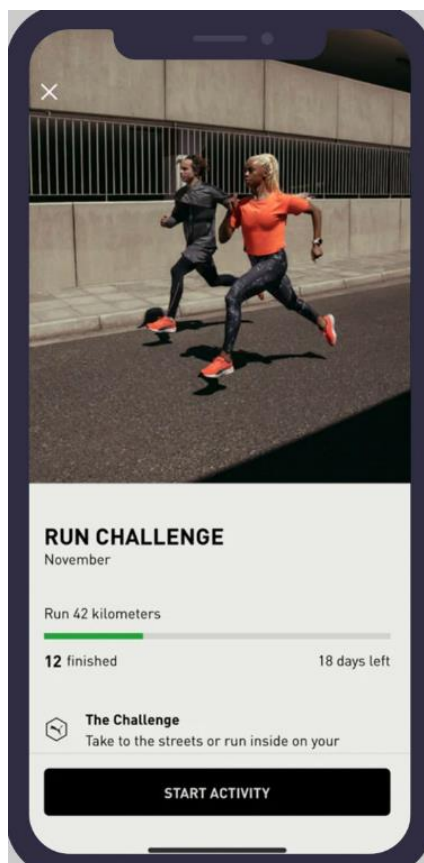


Рисунок 2 - Страница испытания приложения PUMATRAC

2.1.3 Fitness Online

Fitness Online - спортивное приложение, с помощью которого вы сможете вести личный дневник тренировок, найти себе персонального тренера из любого города для онлайн занятий фитнесом. Также можно общаться с другими пользователями и делиться успехами.

Достоинства:

— Большое разнообразие тренировочных программ, подходящих как для новичков, так и для продвинутых спортсменов.

— Возможность работы с профессиональными тренерами через видеосвязь, что обеспечивает более высокий уровень индивидуального подхода.

— Интеграция с умными устройствами и приложениями для отслеживания здоровья, что позволяет пользователям мониторить свои достижения и физическое состояние в реальном времени.

— Предоставление планов питания, созданных диетологами, что помогает улучшить результаты тренировок благодаря сбалансированному питанию.

Недостатки:

— Высокая стоимость подписки по сравнению с некоторыми другими аналогичными платформами, что может быть барьером для пользователей с ограниченным бюджетом.

— Ограниченная функциональность бесплатной версии, большинство полезных функций доступно только после покупки подписки.

— Необходимость стабильного и высокоскоростного интернет-соединения для эффективного использования видеосвязи и потоковых тренировок.

— Возможные технические проблемы и ошибки в приложении, которые могут мешать проведению тренировок и отслеживанию прогресса.



Рисунок 3 - Упражнение в приложение Fitness Online

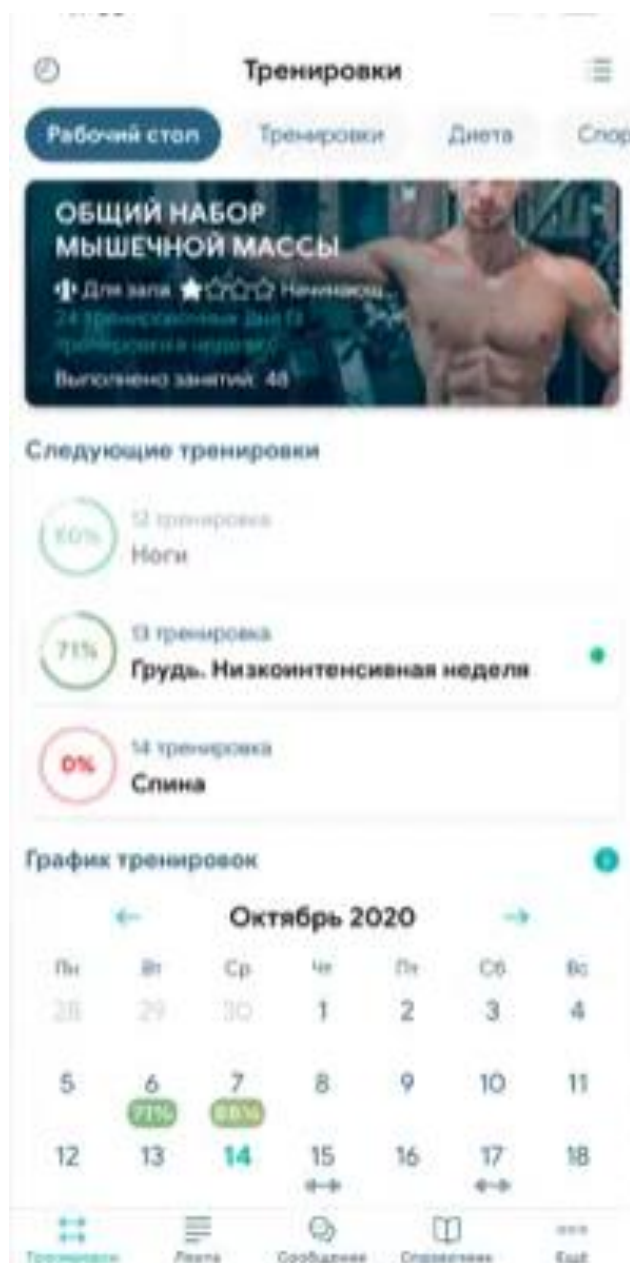


Рисунок 4 - Главная страница в Fitness Online

2.1.4 Strong Workout Tracker Gym Log

Strong Workout Tracker Gym Log – это приложение для отслеживания и логирования тренировок в зале, которое помогает пользователям эффективно планировать и анализировать свои занятия. Это приложение особенно популярно среди тех, кто занимается силовыми тренировками и стремится к улучшению своих результатов.

Достоинства:

— Удобный интерфейс для логирования: Простота в использовании интерфейса позволяет легко записывать каждую тренировку, включая упражнения, количество подходов и повторений, вес и время отдыха.

— Автотаймер отдыха: Встроенный таймер для отслеживания времени отдыха между подходами помогает поддерживать оптимальный ритм тренировки.

— Калькулятор веса: Возможность легко рассчитывать необходимый вес для различных упражнений, включая вес дисков и гантелей.

— Диаграммы и графики: Пользователи могут создавать графики и диаграммы для визуального отслеживания своего прогресса, что помогает мотивировать и анализировать тренировки.

Недостатки:

— Ограниченная бесплатная версия: Основные функции, такие как анализ прогресса и доступ к шаблонам тренировок, доступны только в платной версии.

— Отсутствие видео-инструкций: Приложение не предоставляет видео-инструкций, что может быть неудобно для новичков, не уверенных в правильности выполнения упражнений

— Фокус на силовые тренировки: Приложение в основном ориентировано на силовые тренировки, что может не подойти пользователям, предпочитающим кардио, йогу или другие виды фитнеса.

— Нет социальной составляющей: в отличие от некоторых конкурентов, приложение не имеет функции социальной сети, что исключает возможность взаимодействия с другими пользователями и обмена опытом.

Profile



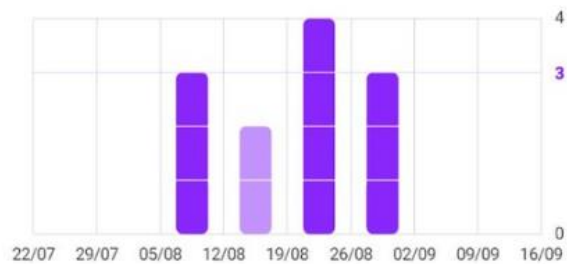
Adam

234 workouts

DASHBOARD



Workouts per week




Bench Press (Barbell)





BEST SET (EST. 1RM)



 Profile

 History

 Workout

 Exercises


 Measurements

Рисунок 5 - Графики в Strong Workout

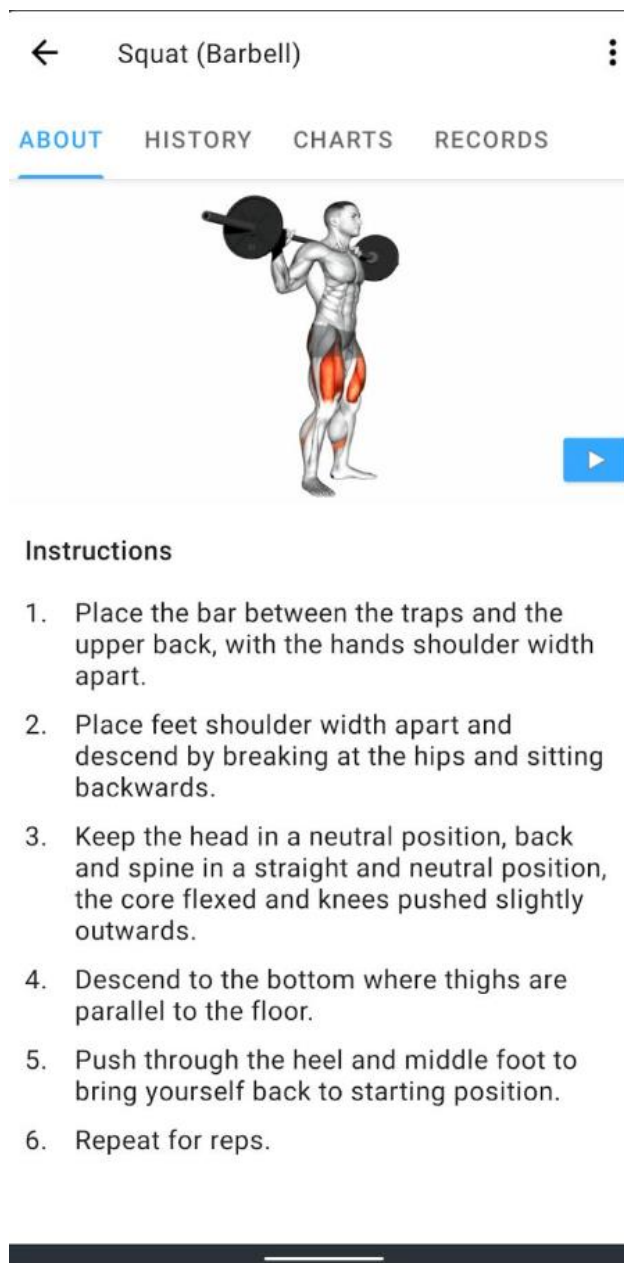


Рисунок 6 - Упражнение в Strong Workout

2.1.5 Nike Training Club

Nike Training Club (NTC) – это популярное фитнес-приложение, разработанное компанией Nike, которое предоставляет пользователям доступ к широкому спектру тренировок и программ, адаптированных под различные уровни подготовки и цели.

Достоинства:

— Большое разнообразие тренировочных программ, подходящих как для новичков, так и для продвинутых спортсменов.

— NTC адаптирует тренировочные планы под индивидуальные цели пользователя, будь то похудение, набор мышечной массы или улучшение общей физической формы.

— Видео инструкции.

— Пользователи могут выбирать тренировки по времени, интенсивности и типу, что позволяет легко интегрировать занятия в ежедневный график.

Недостатки:

— Технические проблемы: Некоторые пользователи сообщают о возможных технических сбоях и ошибках в приложении, что может мешать проведению тренировок.

— Зависимость от интернет-соединения: для просмотра видео инструкций и синхронизации данных требуется стабильное интернет-соединение, что может быть проблемой в условиях ограниченного доступа к сети.

— Отсутствие персонализированных диетических планов: в отличие от некоторых конкурентов, NTC не предоставляет планов питания, созданных диетологами, что может ограничить возможности улучшения результатов тренировок через сбалансированное питание.

— Ограниченные функции в бесплатной версии: хотя приложение предлагает множество бесплатных тренировок, некоторые функции и контент доступны только в платной версии.

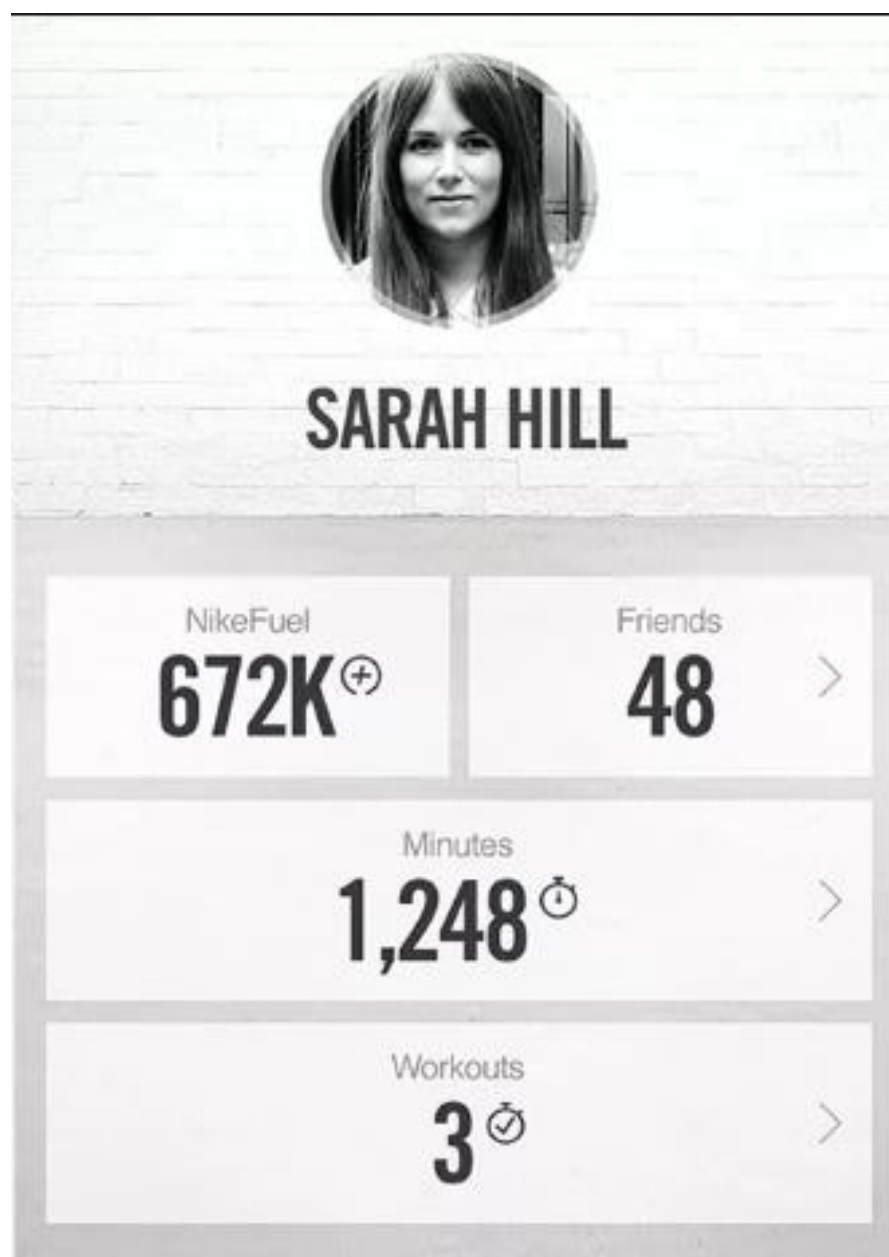


Рисунок 7 - Профиль в NTC



Christy Self

© 14m



Like Cheer

Comment



Рисунок 8 - Сообщество в NTC

3 Реализация

3.1 Средства реализации

Ниже приведен перечень используемых технологий.

Backend

- Python;
- Django Rest Framework;
- PostgreSQL;
- uWSGI;
- Docker.

Frontend:

- Flutter SDK;
- Dart.

Инструменты для ведения документации:

- Miro;
- Swagger;
- Draw.io;
- Figma.

Дополнительный инструментарий:

- Git;
- GitHub;
- Asana.

3.2 Реализация базы данных

Для хранения данных была выбрана база данных PostgreSQL. Она является продуктом с открытым исходным кодом, который поддерживается многими серверами. PostgreSQL поддерживает множественные типы данных,

такие как числа разной точности, тексты с различными кодировками, изображения, звуки, видео, XML-документы, JSON-объекты и многие другие.

3.3 Реализация клиентской части

Для реализации клиентской части приложения было выбрано сочетание языка Dart и Flutter SDK. Этот набор предоставляет разработчикам множество возможностей для создания кроссплатформенных мобильных приложений. Они могут использовать SDK для доступа к аппаратным возможностям устройства, взаимодействия с различными сервисами и API, разработки пользовательского интерфейса и многого другого. Dart, в свою очередь, обеспечивает надежный и мощный язык программирования, который позволяет разработчикам создавать сложные и высокопроизводительные приложения.

Клиентская часть имеет архитектуру, основанную на модели MVVM. MVVM (Model-View-ViewModel) — это архитектурный шаблон, который разделяет разработку графического интерфейса и бизнес-логику приложения.

Все эти компоненты взаимодействуют друг с другом, чтобы обеспечить функциональность приложения. Представление обновляется на основе данных, предоставляемых ViewModel, которая получает данные из модели. Когда пользователь взаимодействует с приложением, ViewModel обрабатывает действия, обновляет модель, и представление автоматически обновляется.

3.3.1 Загрузочный экран

Страница доступна всем пользователям.

В центре экрана находится лого приложения «FitHub».

Ниже загрузочная полоса.



Рисунок 9 - Загрузочный экран

3.3.2 Приветственные экраны

При первом входе пользователь должен просмотреть приветственные экраны.

На экране в центре располагаются тематическая картинка и небольшая мотивационная фраза. Ниже расположены панели. При свайпе будут переходы на другие экраны.

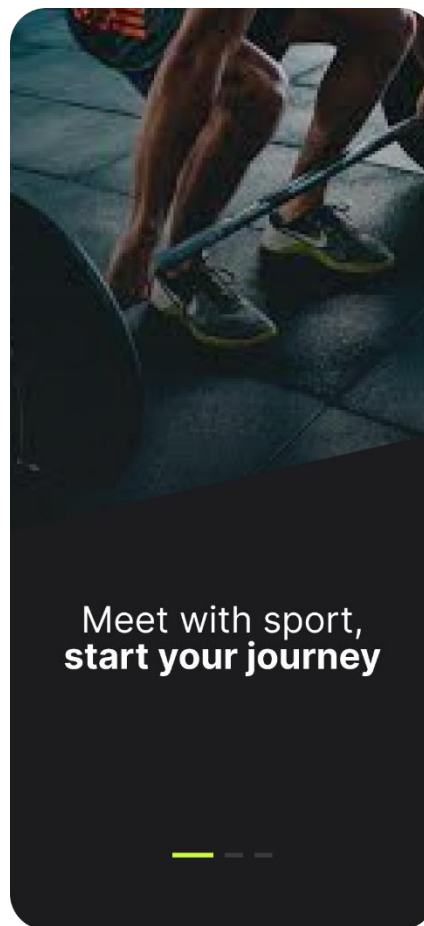


Рисунок 10 - Приветственный экран

3.3.3 Экраны личной информации

Страница доступна пользователям, впервые запустившим приложение. Сверху расположена фраза для пользователя.

На первой странице находятся:

- Две кнопки выбора гендера;
- Кнопка перехода на следующую страницу.

На второй странице находятся:

- Scrollbar для выбора возраста;
- Кнопка перехода на предыдущую страницу;
- Кнопка перехода на следующую страницу.

На третьей странице находятся:

- Scrollbar для выбора цели пользователя;
- Кнопка перехода на предыдущую страницу;
- Кнопка перехода на следующую страницу.

На четвертой странице находятся:

- Scrollbar для выбора уровня физической подготовки пользователя;
- Кнопка перехода на предыдущую страницу;
- Кнопка перехода на страницу «Главный экран».

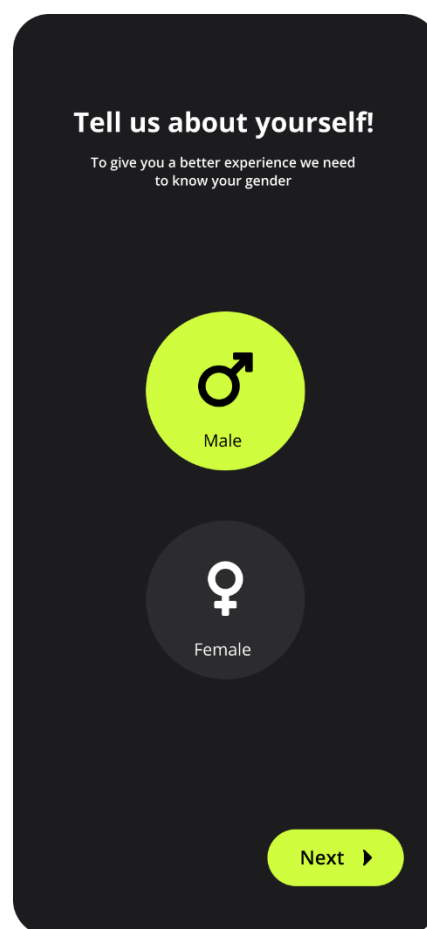


Рисунок 11 - Экран личной информации



Рисунок 12 - Экран личной информации

3.3.4 Главная страница

Страница доступна всем пользователям.

В шапке страницы находится приветствие.

На странице находится список активных, рекомендуемых и всех курсов, оформленный в виде карточек с обложкой и названием. На страницу курса можно перейти по нажатию на карточку.

Внизу страницы находится навигационная панель.

Для неавторизованного пользователя на страницах «Прогресс» и «Комьюнити» будут находиться надпись, сообщающая, что необходимо авторизоваться, а также кнопка, перенаправляющая пользователя на страницу для авторизации/регистрации.

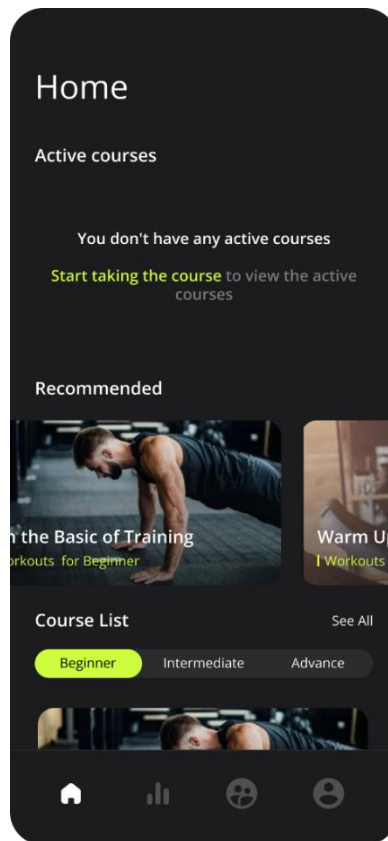


Рисунок 13 - Главный экран для неавторизованного пользователя

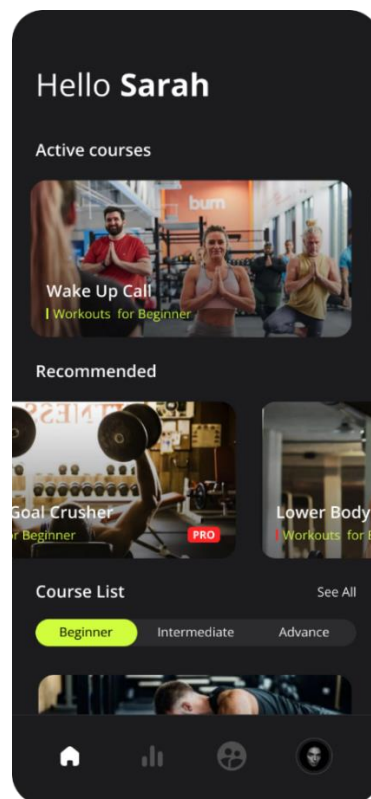


Рисунок 14 - Главный экран для авторизованного пользователя

3.3.5 Страница списка курсов

Страница доступна всем пользователям.

В шапке страницы находятся:

- В левом углу кнопка в виде стрелочки, чтобы вернуться к экрану главная страница.

На странице находятся:

- Название страницы;
- Панель фильтрации;
- Список курсов.

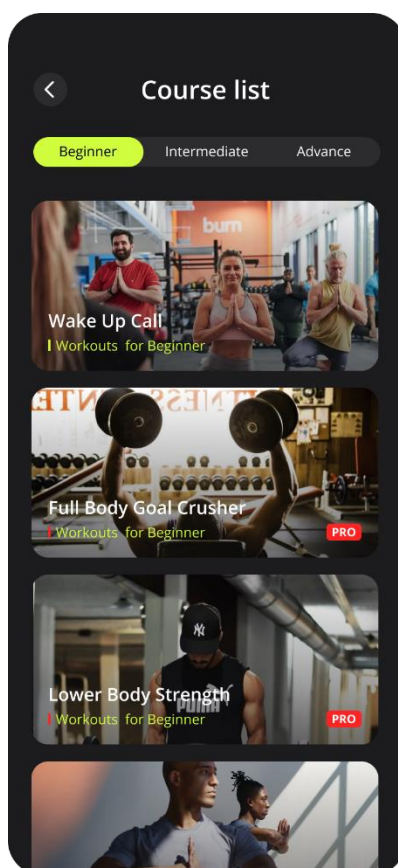


Рисунок 15 - Страница списка курсов

3.3.6 Страница курса

Страница доступна всем пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу.

На странице находится название курса, время прохождения, описание курса, список тренировок и кнопка запуска курса. Каждый элемент списка состоит из фотографии и названия тренировки.

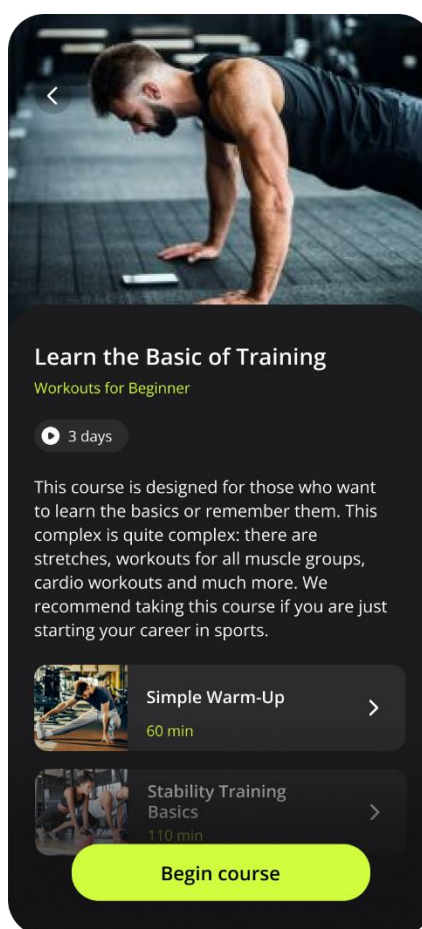


Рисунок 16 - Страница курса

3.3.7 Страница тренировки

Страница доступна всем пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу.

На странице находится название тренировки, время прохождения, список упражнений и кнопка запуска тренировки. Каждый элемент списка состоит из фотографии и названия упражнения.

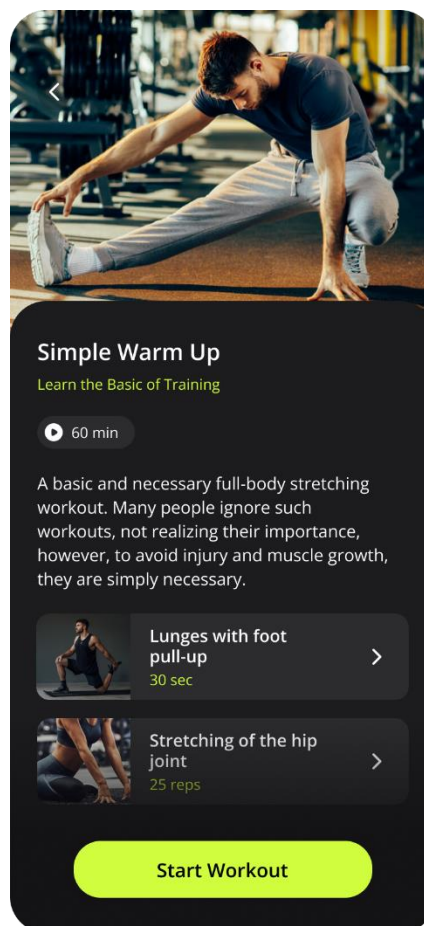


Рисунок 17 - Страница тренировки

3.3.8 Страницы упражнения

Страница доступна всем пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу.

На странице находится название тренировки, время прохождения или количество повторений, описание упражнения, картинка или гифка выполнения упражнения. Также на странице может быть расположен таймер, и две кнопки:

- Начать упражнение;
- Следующее упражнение.

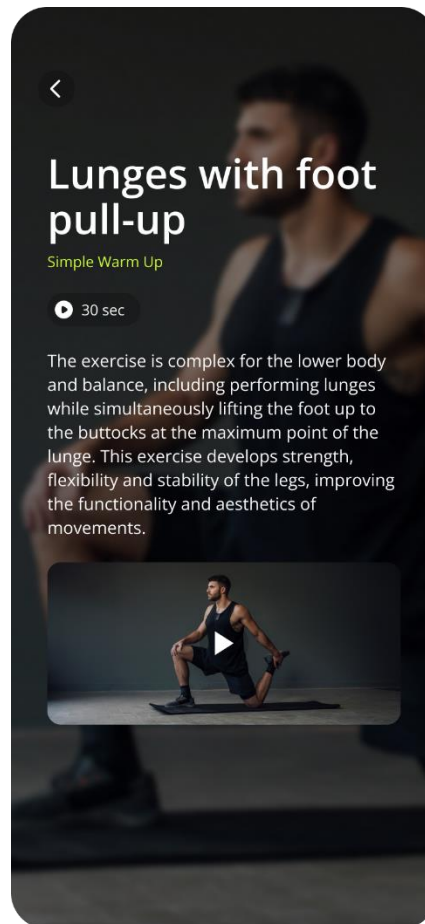


Рисунок 18 - Страница упражнения в режиме просмотра

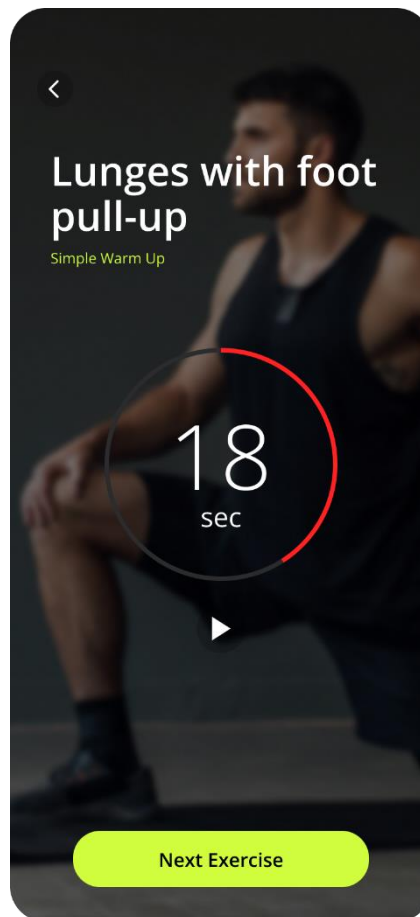


Рисунок 19 - Страница упражнения в режиме прохождения

3.3.9 Страница пройденной тренировки

Страница доступна всем пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу.

На странице находится название тренировки, поздравления с завершением тренировки, время прохождения. Внизу расположена кнопка завершения тренировки.

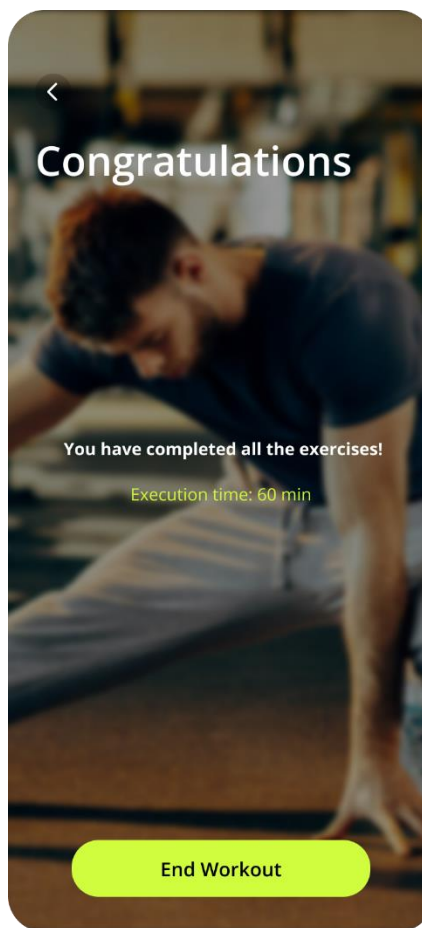


Рисунок 20 - Страница пройденной тренировки

3.3.10 Прогресс

Страница доступная всем пользователям.

В шапке страницы расположен календарь с днями недели и числами месяца, кнопки пролистывания вперед и назад, название месяца и год.

На странице находятся:

- Круг, показывающий время, проведенное в тренировках;
- График активности;
- Панель с тренировочными днями, пройденными курсами и количеством активных курсов.

Снизу расположена навигационная панель.

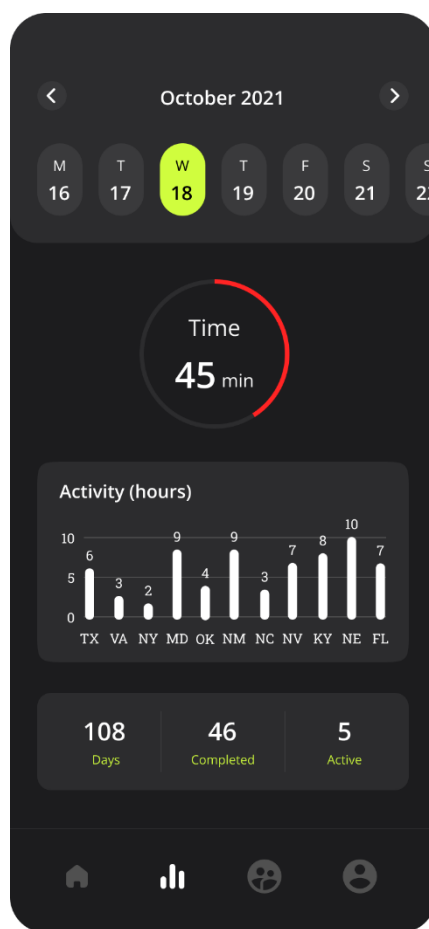


Рисунок 21 - Прогресс

3.3.11 Комьюнити

Страница доступная авторизованным пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу, название страницы, кнопка открывающая страницу “Фильтрация”.

На странице находятся панель, сортирующая курсы, список курсов. Каждый элемент списка состоит из фотографии, названия курса, фотографии пользователя, оценка курса. При удержании курса появится панель с кнопкой пожаловаться, после нажатия откроется страница «Создания жалобы».

Снизу расположена навигационная панель.

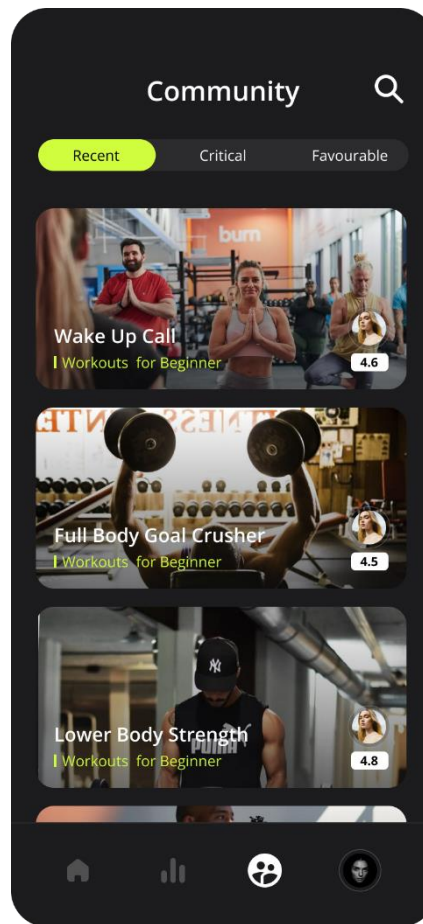


Рисунок 22 - Комьюнити

3.3.12 Фильтрация

Страница доступная авторизованным пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу, название страницы, поисковая строка.

Ниже расположены теги, которые можно выбрать.

Ниже расположена панель для сортировки по времени.

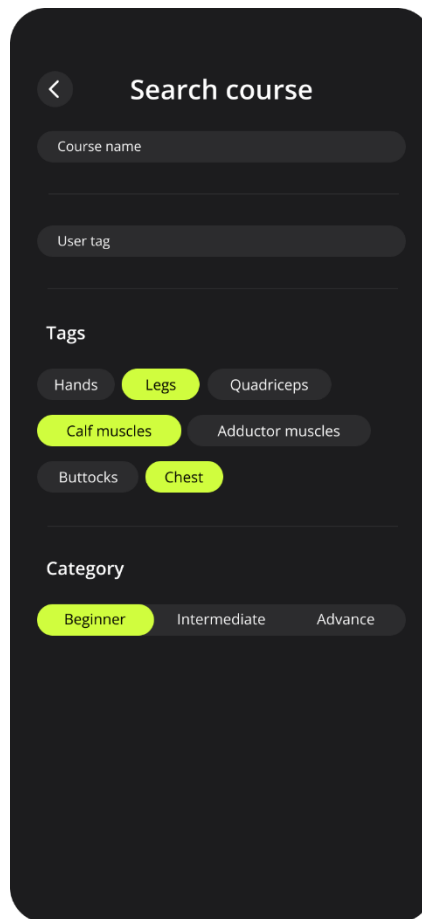


Рисунок 23 - Фильтрация

3.3.13 Страница пользовательского курса

Страница доступная авторизированным пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу.

На странице расположены:

- Название курса;
- Время прохождения;
- Описание;
- Оценка;
- Профили людей, оставивших комментариев;
- Кнопка, чтобы увидеть все комментарии;

- Панель с фотографией пользователя, его имя, оценка, дата написания и сам комментарий;
- Список тренировок, на каждой панели название и фотография тренировки.

Ниже расположена кнопка для начала курса.

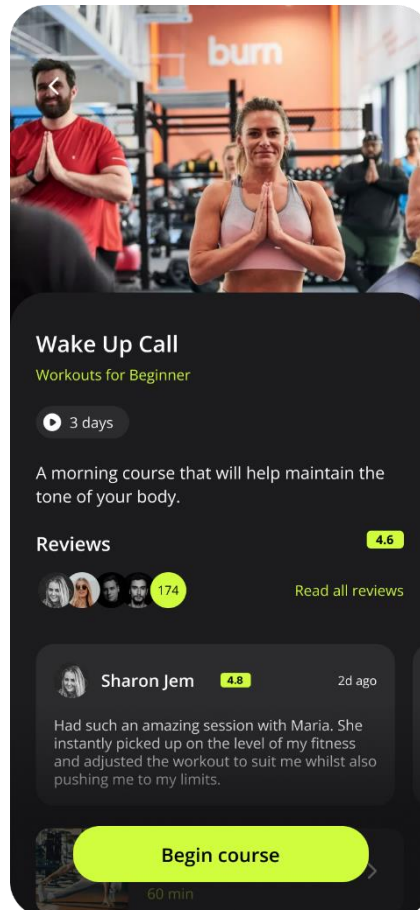


Рисунок 24 - Страница пользовательского курса

3.3.14 Страница комментариев

Страница доступная авторизованным пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу, название страницы.

На странице расположены:

- Панель сортировки комментариев (недавние, негативные и положительные);
- Оценка курса;
- График оценок;
- Количество оценок;
- Список панелей с фотографией пользователя, его именем, оценкой, датой написания и самим комментарием.

Ниже расположена кнопка для написания комментария.

При удержании комментария появится панель с кнопкой «пожаловаться», после нажатия на нее откроется страница «Создание жалобы».

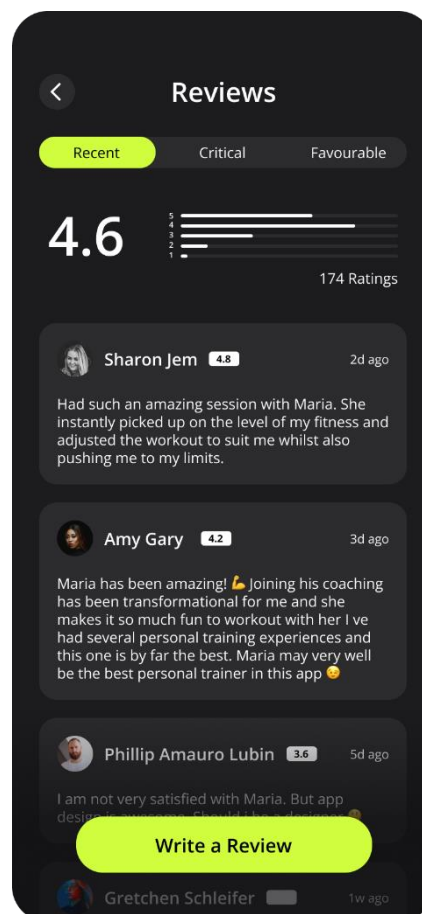


Рисунок 25 - Страница комментариев

3.3.15 Страница создания комментария

Страница доступная авторизованным пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу, название страницы.

На странице расположены:

- Панель для выставления оценки курсу;
- Область для набора текста;
- Кнопка отправки.

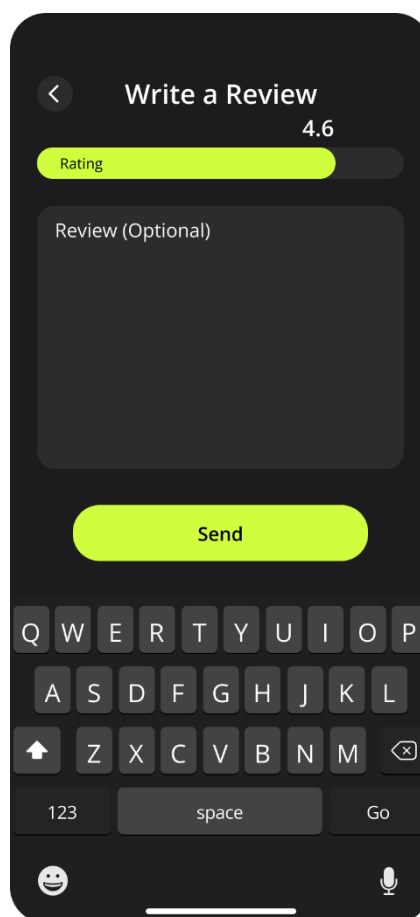


Рисунок 26 - Страница создания комментария

3.3.16 Страница создания жалобы

Страница доступная авторизованным пользователям.

В шапке страницы находится кнопка в виде стрелочки для возвращения на предыдущую страницу, название страницы.

На странице расположены:

- Область для набора текста;
- Кнопка отправки.

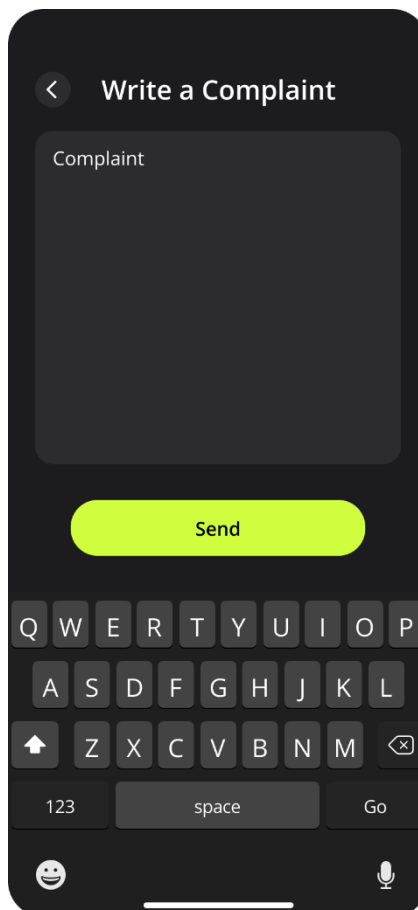


Рисунок 27 - Страница создания жалобы

3.3.17 Профиль

Страница доступная авторизированным пользователям.

Если пользователь не авторизован:

На странице расположена 3 кнопки: «Политика приватности», «Настройки» и кнопка, открывающая экран «Регистрация».

Если пользователь авторизован:

В шапке страницы находятся надпись «Профиль» и кнопка возврата на предыдущую страницу.

На странице расположены:

- Фотография пользователя;
- Дата его регистрации;
- Имя пользователя;
- Кнопка редактирования профиля;
- Политика приватности;
- Настройки;
- Кнопка «Мои курсы»
- Кнопка улучшения профиля до премиума;
- Кнопка выхода из аккаунта.

В шапке страницы редактирования профиля расположены надпись «Редактирование профиля» и кнопка в виде стрелочки для возврата на предыдущую страницу. На странице расположены фото пользователя и заполненные поля для ввода, доступные для изменения, а также кнопка сохранения.

На странице премиум пользователя будут добавлены иконка «Pro» на фотографию и дата окончания подписки. А кнопка покупки премиума будет убрана.



Рисунок 28 - Профиль

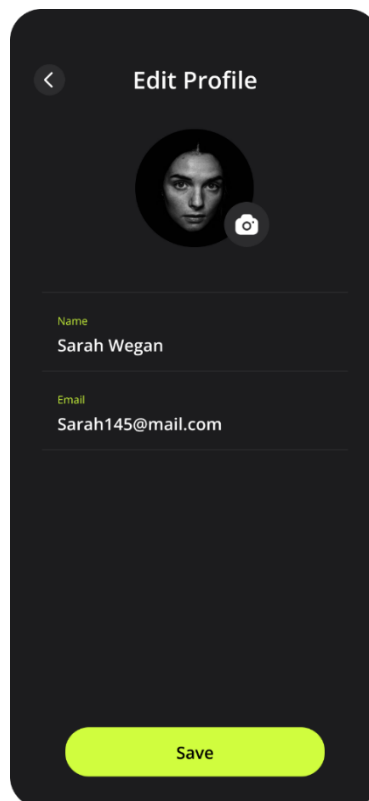


Рисунок 29 - Редактирование профиля

3.3.18 Страница настроек

Страница доступная всем пользователям.

В шапке страниц находится кнопка в виде стрелочки для возвращения на предыдущую страницу, название страницы.

На странице настроек расположены:

- Кнопка настройки уведомлений;
- Кнопка смены языка;
- Кнопка для связи с нами.

На странице уведомлений расположены переключатели для получения уведомлений. А внизу надпись с выделенным текстом, который перенаправляет пользователя в настройки телефона.

На странице смены языка 2 поля с названием языка и галочкой.

На странице связи с нами область для ввода текста и кнопка отправки.

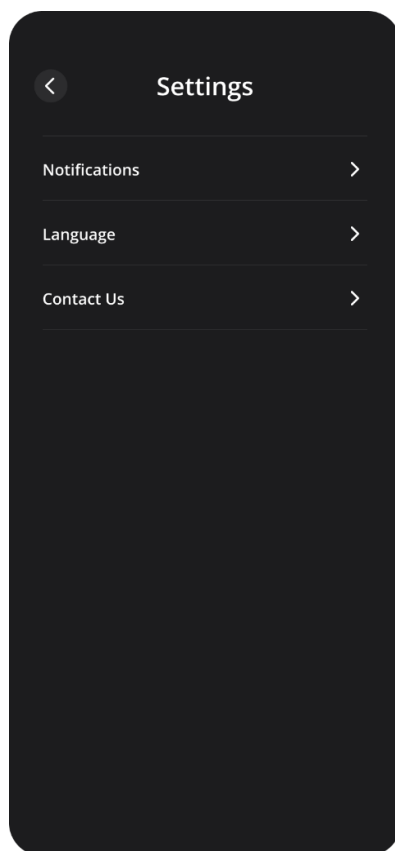


Рисунок 30 - Страница настроек

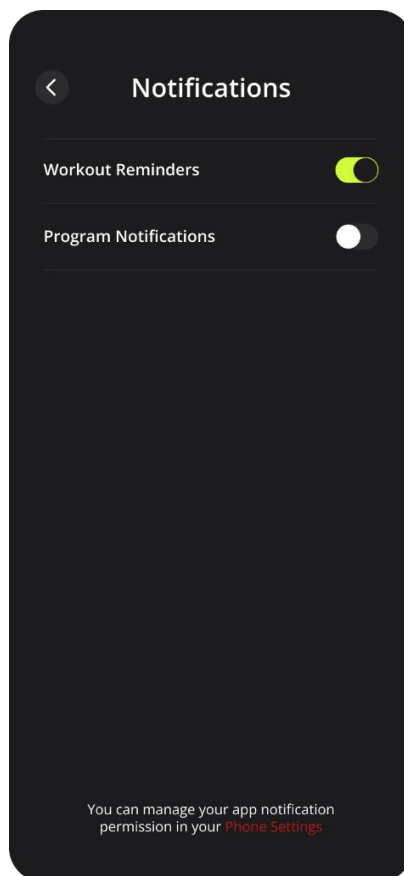


Рисунок 31 - Страница настроек уведомлений

3.3.19 Вход/регистрация

Страница доступна неавторизированным/незарегистрированным пользователям.

В шапке страницы расположен переключатель вход/регистрация

На странице «Вход» расположены:

- Картинка на фоне;
- Приветствие;
- Поля для ввода логина и пароля;
- Кнопка «Войти»;
- Кнопка «Забыл пароль»;
- Кнопка «Может позже».

На странице «Зарегистрироваться» расположены:

- Картинка на фоне;
- Приветствие;
- Поля для ввода логина и пароля (2 раза);
- Кнопка «Зарегистрироваться»;
- Кнопка «Может позже».

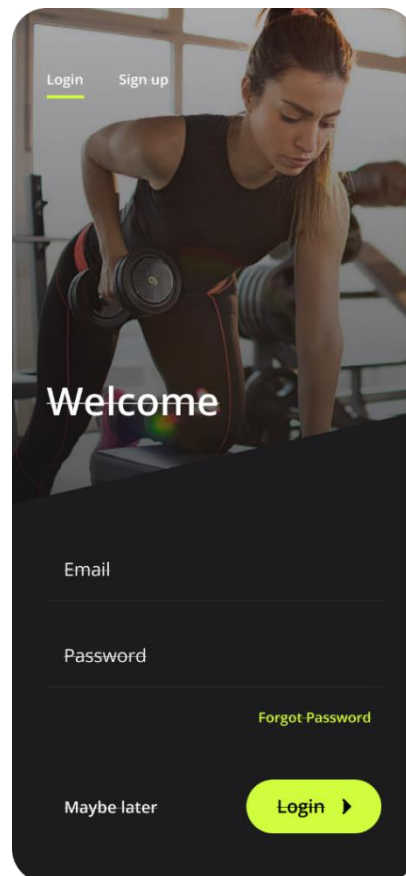


Рисунок 32 - Вход

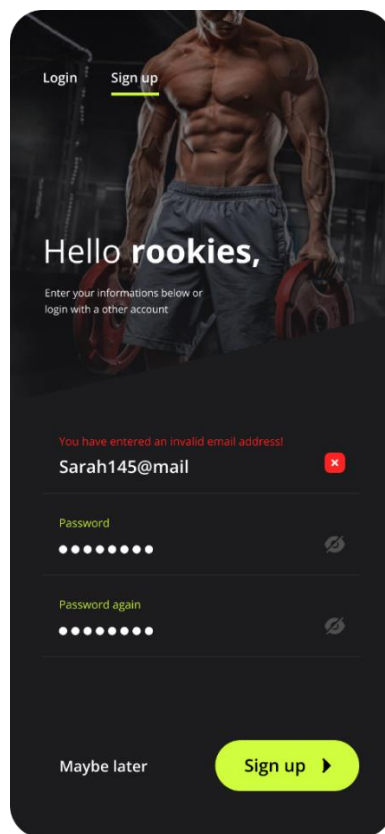


Рисунок 33 - Регистрация

3.3.20 Страница восстановления пароля

Страница доступная неавторизованным пользователям.

В шапке страницы расположен переключатель вход/регистрация.

На странице «Отправления почты» расположены поле для ввода почты и кнопка отправки.

На странице «Ввода кода» расположены поля для ввода цифр, кнопка «проверить» и кнопка отправить код еще раз.

На странице ввода нового пароля расположены поля для ввода нового пароля и кнопка «принять».

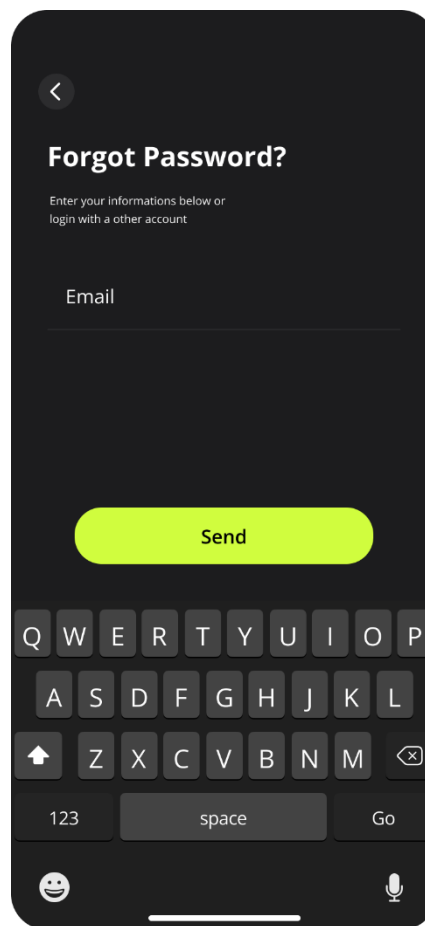


Рисунок 34 - Страница отправления почты

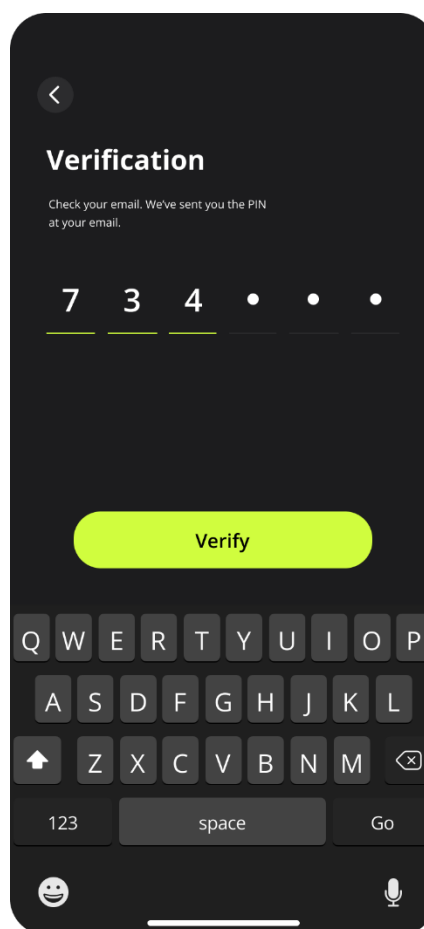


Рисунок 35 - Страница ввода кода

3.3.21 Создание курса

Страница доступная авторизованным пользователям.

В шапке страницы находятся кнопка для возвращения на предыдущую страницу, название страницы.

На странице расположены:

- Кнопка «+», открывающая страницу «создание курса»;
- Список созданных курсов.

На странице создания курса расположены:

- такая же шапка, но еще добавлена шестеренка для настройки курса, она открывает страницу, где можно выбрать теги и настроить приватность;

- Картинка курса, при нажатии на которую открывается страница со списком картинок, из которого можно выбрать нужную;
- Область для ввода текста, где можно добавить описание;
- Кнопка «+», которая открывает страницу «Добавление тренировок в курс»;
- Список добавленных тренировок.

На странице «Добавление тренировок в курс» есть список всех тренировок, если нажать на тренировку, то откроется страница «Редактирования тренировки».

На странице «Редактирование тренировки» есть:

- Картинка курса, при нажатии открывается страница со списком картинок, из которого можно выбрать картинку;
- Область для ввода текста, где можно добавить описание;
- Кнопка «+», которая открывает страницу «Добавление упражнения в курс»;
- Список добавленных упражнений.

На странице «Добавление упражнения в курс» есть список упражнений, при нажатии на упражнение откроется страница «Редактирование упражнения».

На странице «Редактирование упражнения» есть описание упражнения, картинка или гифка и панель, при нажатии на которую откроется страница для изменения времени или количества повторений.

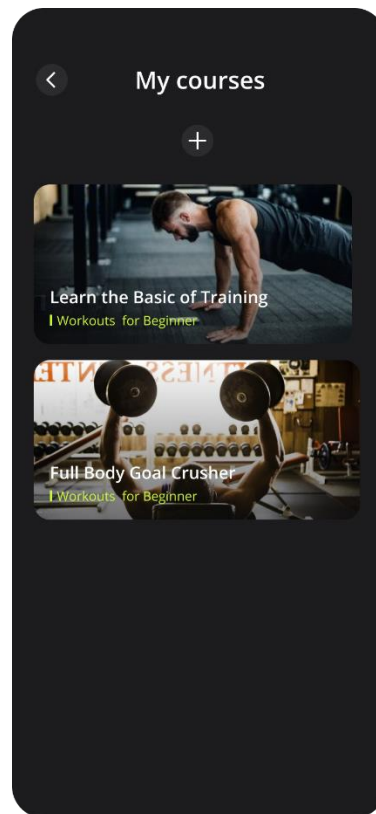


Рисунок 36 - Страница создания курсов

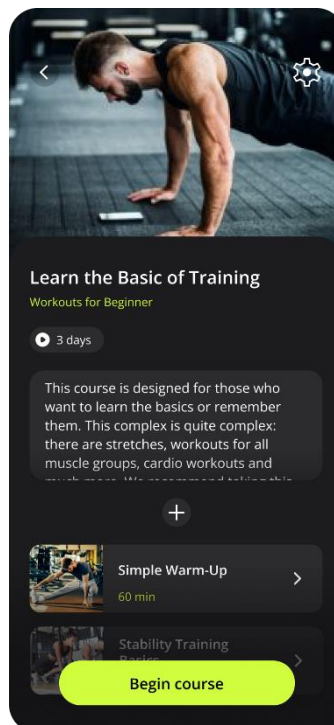


Рисунок 37 - Создание/редактирование курса

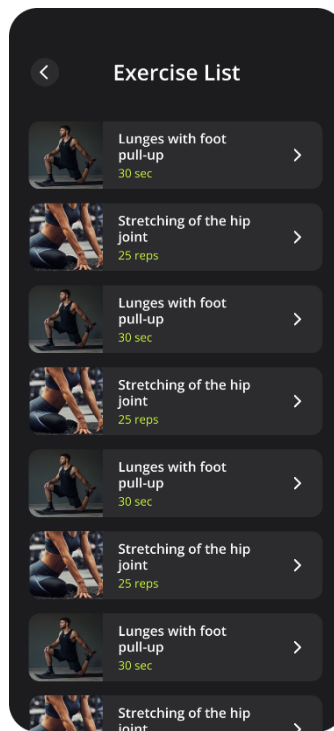


Рисунок 38 - Список упражнений для добавления в курс

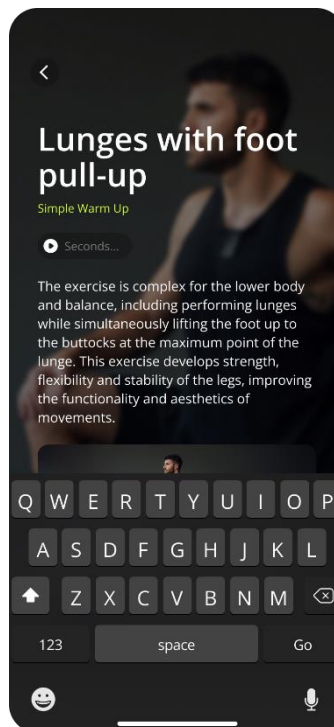


Рисунок 39 - Редактирование упражнения

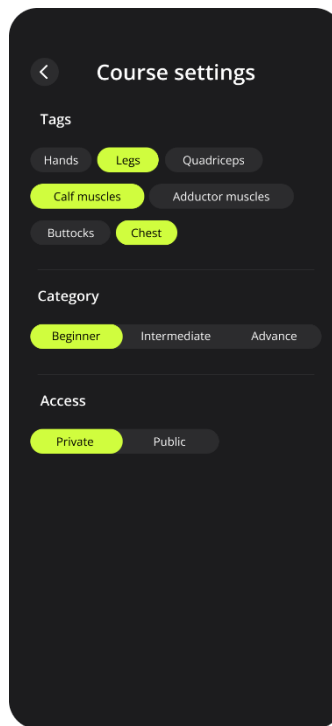


Рисунок 40 - Настройки курса

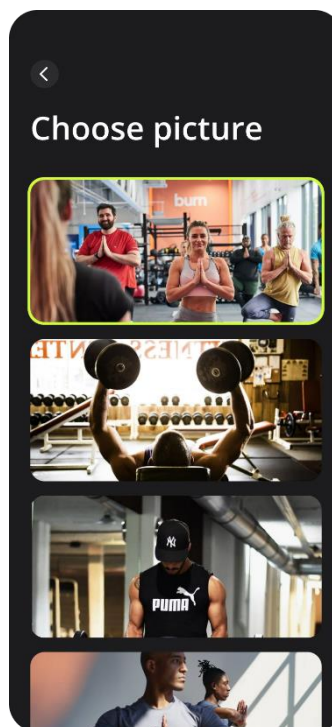


Рисунок 41 - Страница выбора картинки/гифки

3.4 Описание архитектуры клиентской части

Код разделен на 3 части:

- Модели;
- Сервисы;
- Представление.

Все слои разделены соответственно в файловой системе:

- для моделей: файлы содержат request / response;
- для сервисов: файлы содержат service;
- для представления: Model, Provider, Screen (реализация паттерна MVVM).

3.4.1 Слой моделей

Слой моделей является ответственным за сохранение данных. Модели были разработаны для каждого возможного JSON-файла, который приходит с сервера или отправляется на него. Полученный с сервера JSON-файл обрабатывается в объект соответствующей сущности или в список сущностей.

```

@freezed
You, 10 seconds ago | 1 author (You)
class UserLoginRequest with _$UserLoginRequest {
  @JsonSerializable(
    explicitToJson: true,
    includeIfNull: false,
  )
  factory UserLoginRequest({
    required String email,
    required String password,
  }) = _UserLoginRequest;

  factory UserLoginRequest.fromJson(Map<String, dynamic> json)
    => _$UserLoginRequestFromJson(json);
}

```

Рисунок 42 - Пример класса модели

Файловая структура классов моделей: lib → domain → основной репозиторий для моделей.

Для сериализации и десериализации данных используется Dart библиотека JsonSerializable, методы toJson, fromJson, экземпляры моделей создаются в фабричном конструкторе.

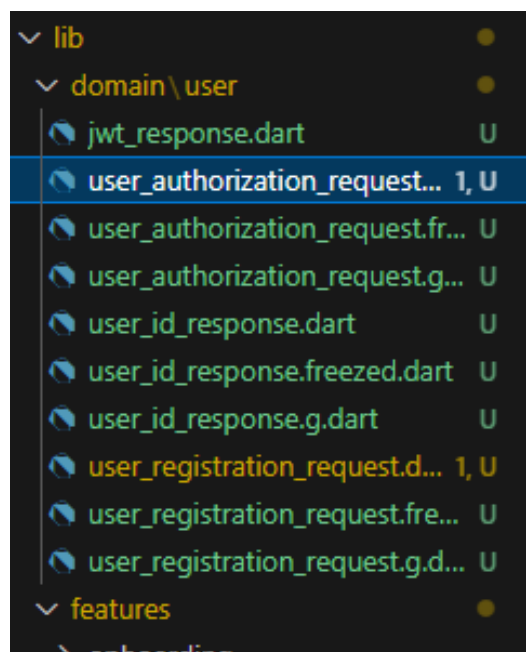


Рисунок 43 - Файловая структура классов модели

3.4.2 Слой сервисов

Слой сервисов является ответственным для работы с запросами. В нем прописываются адрес запроса, тела, query и path параметры для методов получения данных с сервера (со Swagger). Реализован с помощью Retrofit и Dio. Retrofit / Retrofit Annotation для автоматизации развертывания сервисов, Dio расширяет возможности HTTP клиента.

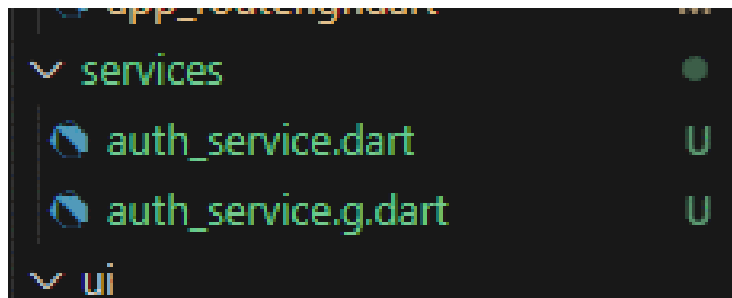


Рисунок 44 - Файловая структура классов сервисов

Структура моделей, сервисов совпадают с моделями из Swagger. Никакие данные не мапятся.

```
@RestApi()
You, yesterday | 1 author (You)
abstract class AuthService {
  factory AuthService(
    Dio dio
  ) = _AuthService;

  @POST('/auth/login/')
  Future<JwtResponse> loginUser({
    @Body() required UserLoginRequest request,
  });
}
```

Рисунок 45 - Пример реализации класса сервиса по авторизации

3.4.3 Слой репозиториев

Слой репозиториев отвечает за доступ к данным и взаимодействие с сервером. Он выполняет все необходимые запросы на сервер, такие как: добавление или удаление товаров из избранного, добавление или удаление товаров из корзины, получение информации о пользователе и другие. Обращается к слою сервисов и получает из них данные, которые в последствии используются в модели представления в виде удобного для клиента.

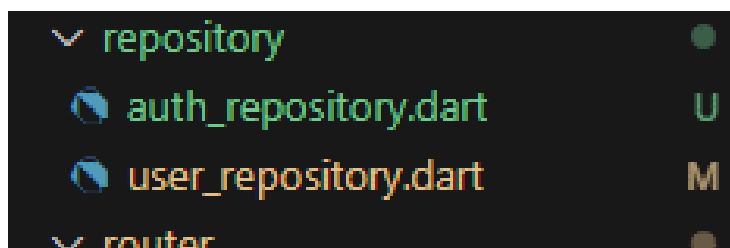


Рисунок 46 - Файловая структура класса репозиториев

```
You, yesterday | 1 author (You)
class AuthRepository {
  final AuthService _authService;

  AuthRepository(
    this._authService,
  );

  Future<JwtResponse> login({
    required UserLoginRequest request,
  }) async {
    try {
      final result = await _authService.loginUser(
        request: request,
      );
      return result;
    } catch (e, s) {
      throw Error.throwWithStackTrace(e, s);
    }
  }
}
```

Рисунок 47 - Пример реализации класса для репозитория авторизации

3.4.4 Слой представления

Слой представления является прямым отражением паттерна MVVM. Вместо привычного ViewModel используется Provider, который позволяет определять состояние экрана, и изменять его под конкретные нужды.

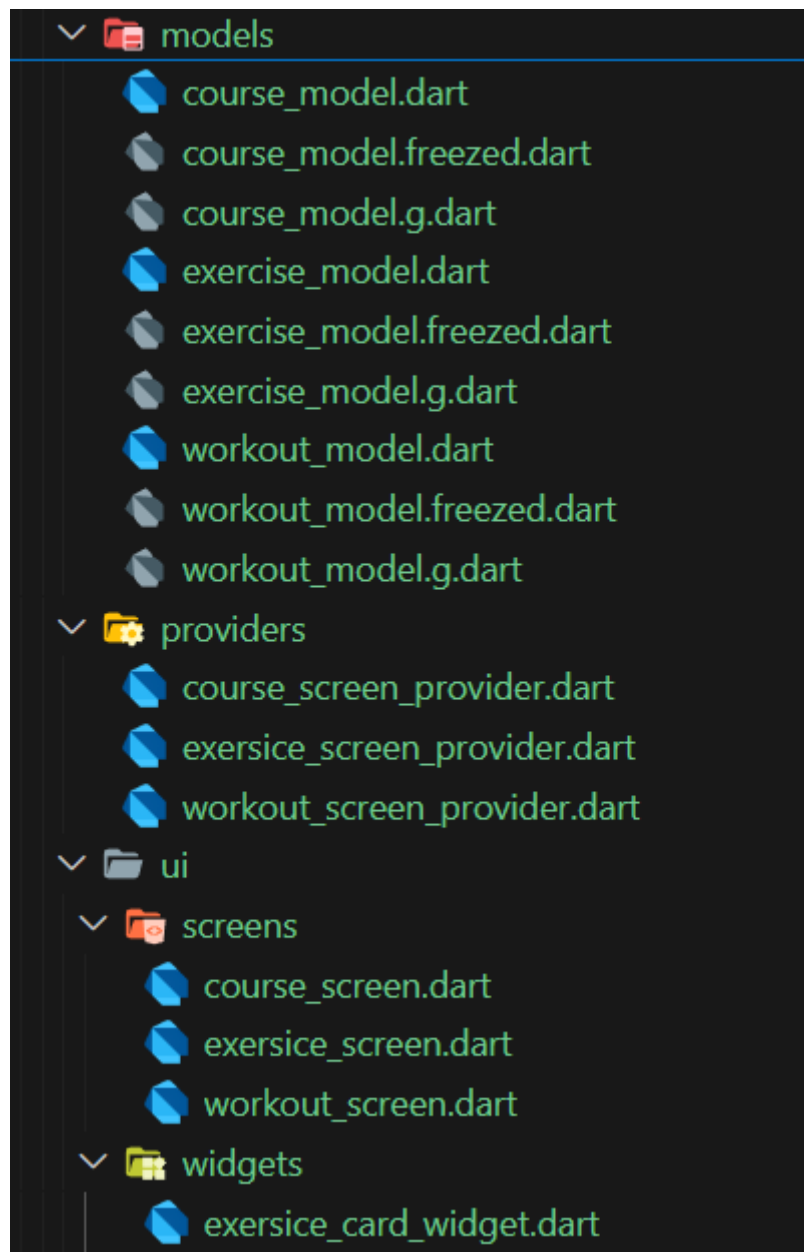


Рисунок 48 - Файловая структура основных страниц

```

class AuthorizationScreenProvider extends ChangeNotifier {
  final AuthRepository _authRepository;
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final GlobalKey<FormState> formKey = GlobalKey<FormState>();

  AuthorizationScreenProvider(this._authRepository);

  bool _isLoading = false;
  bool get isLoading => _isLoading;

  Future<void> login(BuildContext context) async {
    if (formKey.currentState?.validate() ?? false) {
      _setLoading(true);
      try {
        final request = UserLoginRequest(
          email: emailController.text,
          password: passwordController.text,
        );

        await _authRepository.login(request: request);
        _setLoading(false);
        AutoRouter.of(context).pushAndPopUntil(const HomeRoute(), predicate: (_) => false);
      } catch (e) {
        _setLoading(false);
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Login failed: $e'))
        );
      }
    }
  }
}

```

Рисунок 49 - Пример реализации класса провайдера экрана авторизации

3.5 Серверная часть

Серверная часть, помимо описанной выше базы данных, состоит из следующих компонентов:

- Файл settings.py, содержащий настройки базы данных, настройки для подключения к ней, валидации полей, настройки JWT, секретное слово, список разрешённых хостов и т.д.
- Сериализаторы для подготовки данных к отправке и наоборот, для загрузки полученных данных на сервер.

— Модели данных, с которыми будет работать Django ORM. Также определены некоторые дополнительные поля (например, для генерации пути к изображению).

— Файл маршрутизации (`urls.py`) – определяет пути к конечным точкам API. Также определены контроллеры для конечных точек.\

— Файл `views.py` – основная логика работы с базой данных.

3.5.1 Архитектура серверной части приложения

Серверная часть приложения реализована соответственно трехслойной архитектуре MVT (Model – View – Template) с использованием фреймворка Django REST Framework (DRF). Django REST Framework предоставляет мощные возможности для разработки веб-API, обеспечивая гибкость и масштабируемость приложений.

— Model (Модель): Модель представляет собой слой данных, который отвечает за взаимодействие с базой данных и определение логики бизнес-процессов. В нашем приложении модели описывают структуру данных, включая поля и отношения между таблицами. Использование Django ORM (Object-Relational Mapping) позволяет легко управлять базой данных и выполнять сложные запросы, минимизируя количество написанного вручную SQL-кода. Это повышает производительность разработки и снижает вероятность возникновения ошибок.

— View (Представление): Представления обрабатывают запросы от клиента, взаимодействуют с моделями для получения данных и готовят ответ в удобном формате. В случае нашего приложения, разработанного на Django REST Framework, представления также могут быть реализованы в виде классовых или функциональных API представлений, что обеспечивает гибкость в выборе подхода к разработке. Классовые представления облегчают создание

многоразового кода и логики, тогда как функциональные представления могут быть более простыми и легковесными для выполнения специфических задач.

— **Template (Шаблон)**: Шаблоны отвечают за представление данных в удобном для пользователя виде. В контексте веб-API, разрабатываемого с помощью Django REST Framework, шаблоны могут включать в себя сериализаторы, которые преобразуют сложные типы данных, такие как запросы и модели, в легко читаемые форматы JSON или XML. Это обеспечивает взаимодействие между клиентом и сервером в стандартизированном формате, упрощая интеграцию с различными фронтенд-фреймворками и мобильными приложениями.

Django REST Framework предоставляет мощные возможности для разработки веб-API, обеспечивая гибкость и масштабируемость приложений. Он включает в себя такие функции, как встроенная пагинация, фильтрация, аутентификация и авторизация, что значительно упрощает реализацию сложных функциональностей. Кроме того, DRF поддерживает создание многоразовых и расширяемых компонентов, что позволяет быстро адаптировать приложение к изменяющимся требованиям бизнеса.

Таким образом, использование трехслойной архитектуры MVT в сочетании с возможностями Django REST Framework позволяет нам создавать надежные и масштабируемые серверные приложения, обеспечивающие высокую производительность и удобство использования для конечных пользователей.

3.5.2 Структура серверной части приложения

Структура серверной части нашего Django проекта тщательно организована и разделена на три ключевых приложения: `authorization`, `community` и `courses`. Каждое из этих приложений отвечает за свою часть

функциональности и взаимодействует с другими компонентами системы, обеспечивая целостность и логическую связность всего проекта.

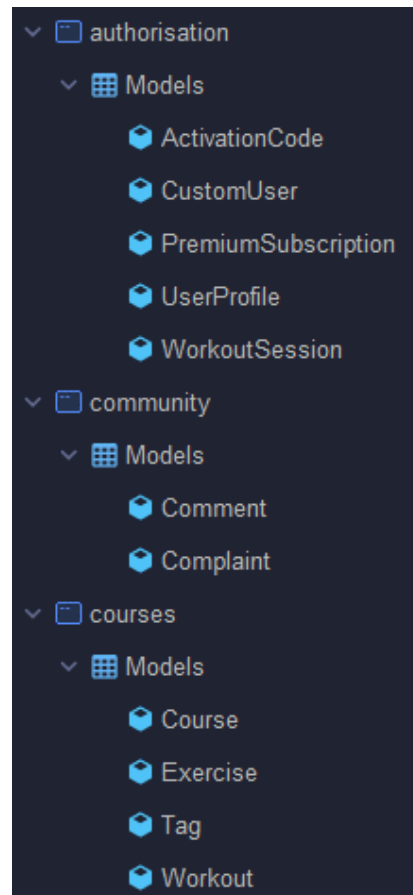


Рисунок 50 - Структура Django проекта

В этих приложениях основная логика обработки запросов находится в файлах views.py.

```
class RegisterAPIView(generics.GenericAPIView):
    serializer_class = MyUserSerializer

    8 usages (8 dynamic)  Shish-ai-ai

    def post(self, request):
        serializer = self.get_serializer(data=request.data)
        if serializer.is_valid():
            user = serializer.save()
            return response.Response(data: {
                "user": MyUserSerializer(user).data,
                "message": "User Created Successfully."
            }, status=status.HTTP_201_CREATED)
        return response.Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Рисунок 51 - Пример метода для регистрации

```
2 usages  Shish-ai-ai
class LoginAPIView(generics.GenericAPIView):
    serializer_class = MyUserSerializer

8 usages (8 dynamic)  Shish-ai-ai
def post(self, request):
    email = request.data.get('email')
    password = request.data.get('password')

    user = authenticate(request, username=email, password=password)
    if user:
        refresh = RefreshToken.for_user(user)
        return response.Response( data: {
            "user": MyUserSerializer(user).data,
            "refresh": str(refresh),
            "access": str(refresh.access_token),
            "message": "User Logged In Successfully."
        }, status=status.HTTP_200_OK)
    return response.Response( data: {"message": "Invalid Credentials"}, status=status.HTTP_401_UNAUTHORIZED)
```

Рисунок 52 - Пример метода для авторизации

4 Моделирование системы

4.1 Диаграмма прецедентов

Рассмотрим полную диаграмму использования приложения различными типами пользователей (Рисунок 5). В данном контексте составление диаграммы прецедентов обусловлено необходимостью моделирования системы и понимания её функциональности и потребностей пользователей. Эти диаграммы помогают определить основные действия, которые пользователь должен совершить в системе для достижения определенных целей. Они также помогают выявить возможные риски и проблемы, которые могут возникнуть в процессе использования системы.

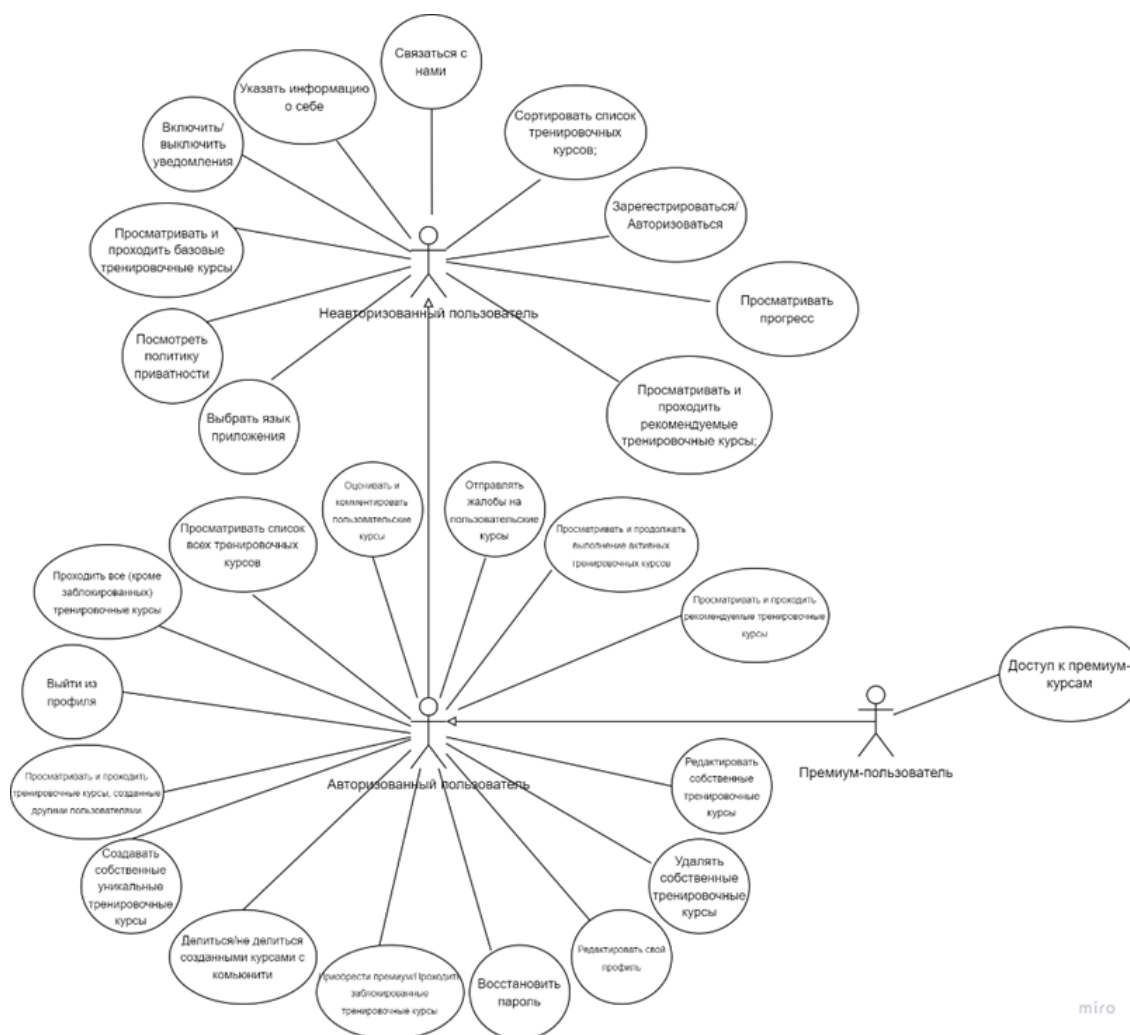


Рисунок 53 - Диаграмма прецедентов (use-case)

4.2 Диаграмма прецедентов

Диаграмма последовательности (Рисунок 6) играет важную роль в проекте, помогая более глубоко понять процесс, повысить его эффективность и упростить взаимодействие.

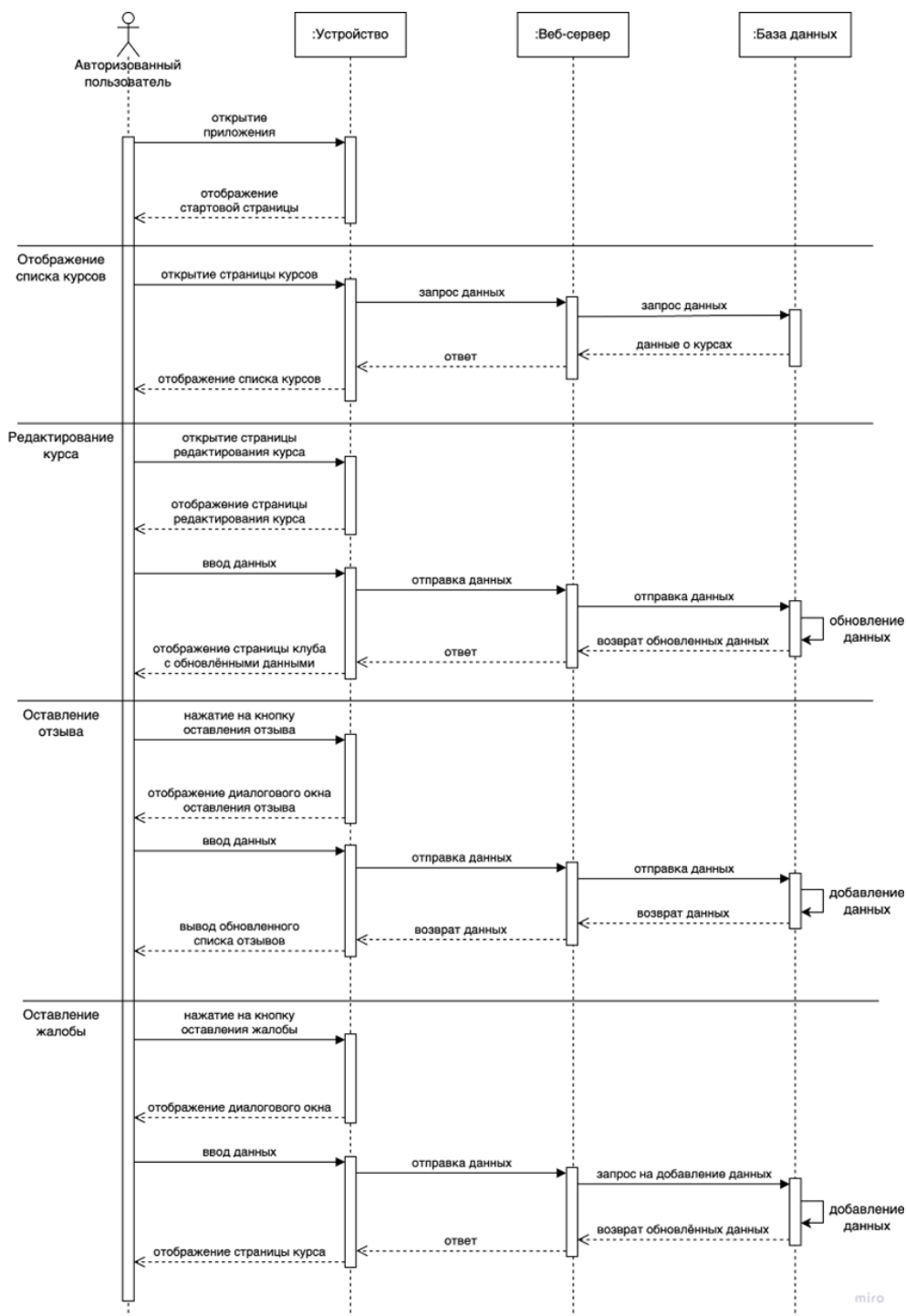


Рисунок 54 - Диаграмма последовательностей (sequence)

4.3 Диаграмма развертывания

Диаграмма развертывания (Рисунок 7) помогает определить необходимости в аппаратном обеспечении, спланировать установку и настройку компонентов системы, а также оценить её производительность и масштабируемость.

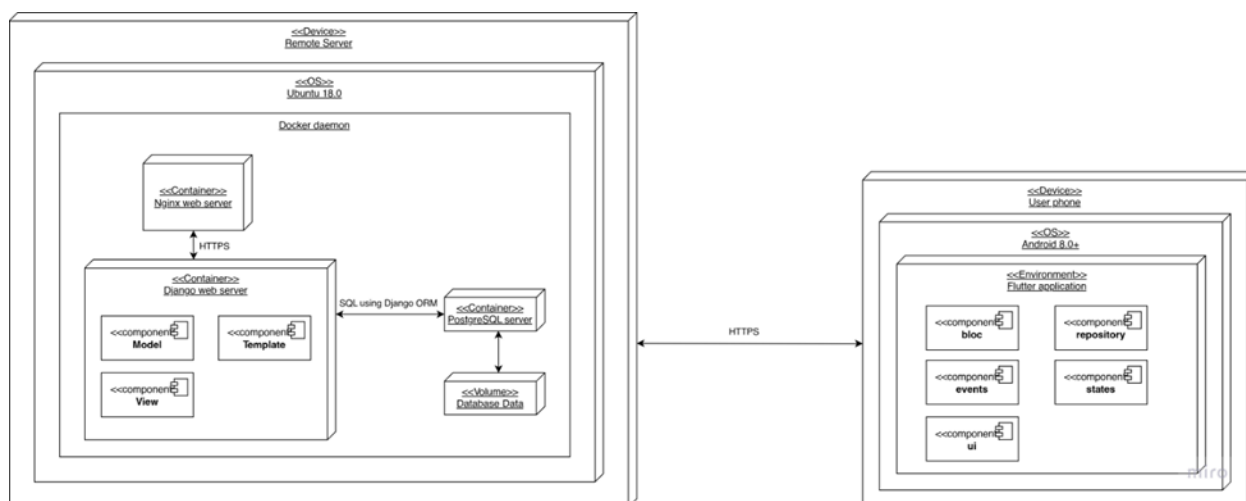


Рисунок 55 - Диаграмма развертывания приложения

4.4 Диаграммы состояния

Диаграмма состояния (Рисунок 8) позволяет анализировать возможные сценарии поведения системы, выделять ключевые состояния и переходы между ними, а также оценивать её надежность и устойчивость к ошибкам. В рамках нашего проекта были разработаны три диаграммы состояний для администратора, зарегистрированного пользователя и незарегистрированного пользователя.

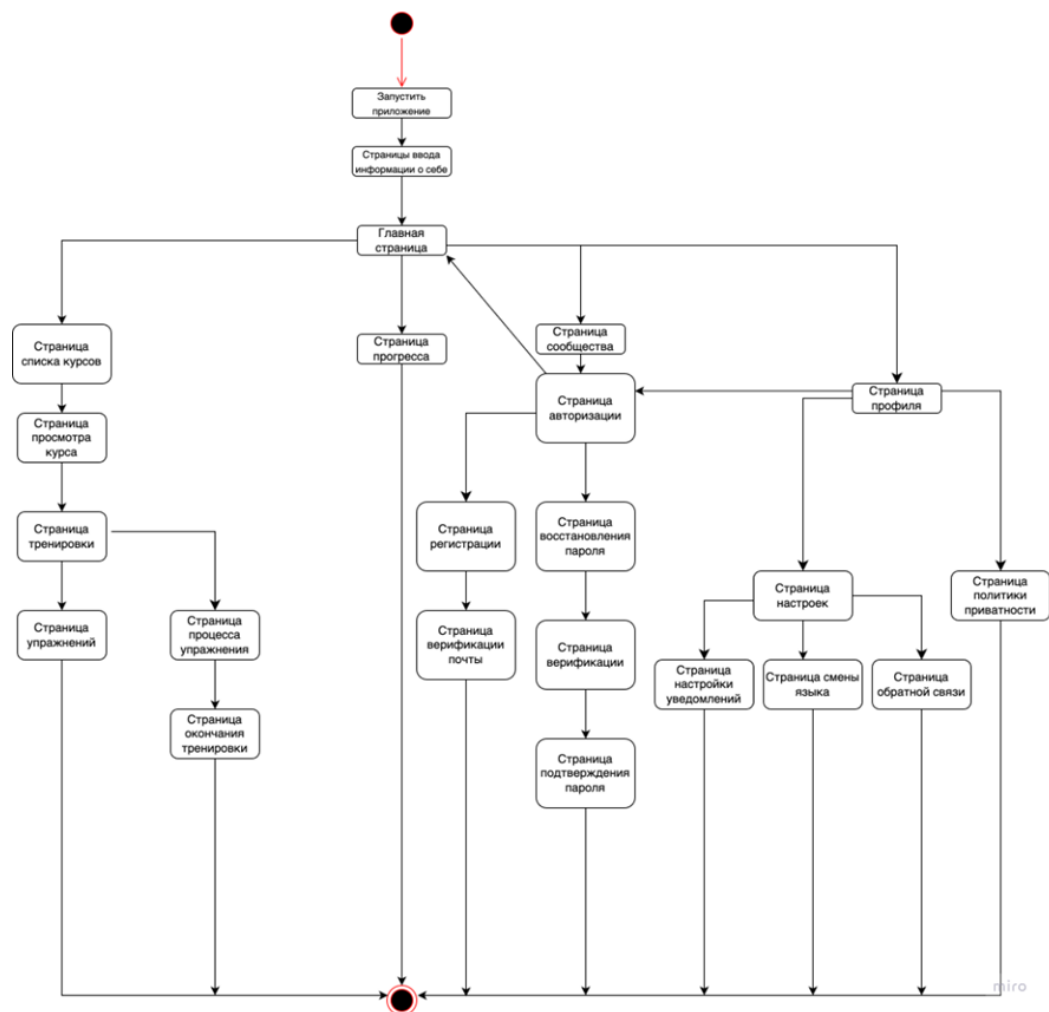


Рисунок 56 - Диаграмма состояния

4.5 ER-диаграмма

ER-диаграмма (Рисунок 9) предоставляет структурное представление данных, иллюстрируя сущности (объекты) в системе и их взаимосвязи. Она помогает определить основные сущности, их атрибуты и отношения между ними, что облегчает проектирование базы данных и анализ требований к системе.

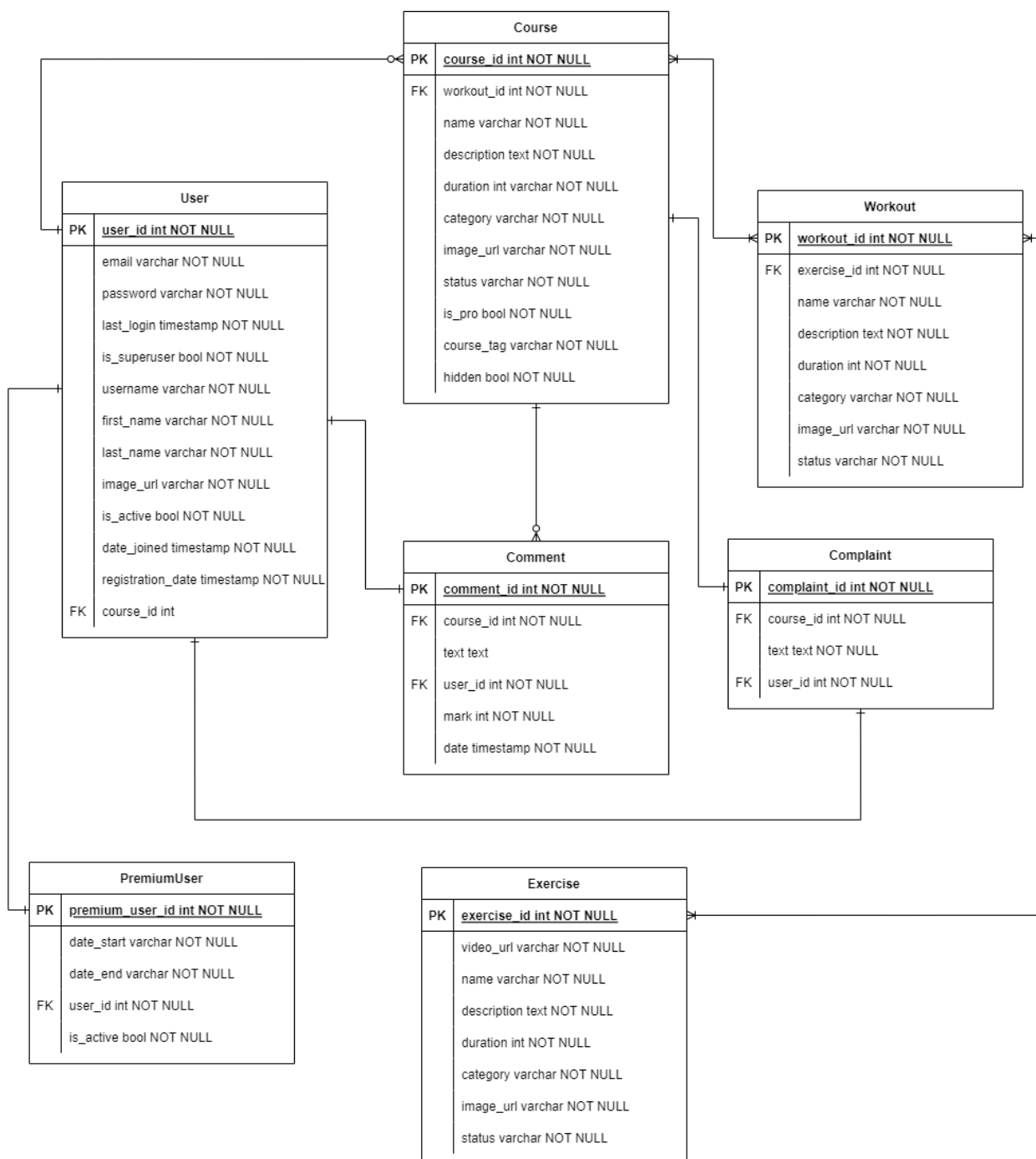


Рисунок 57 - ER-диаграмма

4.6 Диаграммы активности

Диаграмма активности (Рисунок 10) помогает улучшить понимание системных процессов, выявить и оптимизировать узкие места. Кроме того, она

применяется для описания бизнес-процессов и управления проектами. Для данного проекта было спроектировано 5 диаграмм активности: для администратора, преподавателя, учащегося, зарегистрированного пользователя и незарегистрированного пользователя.

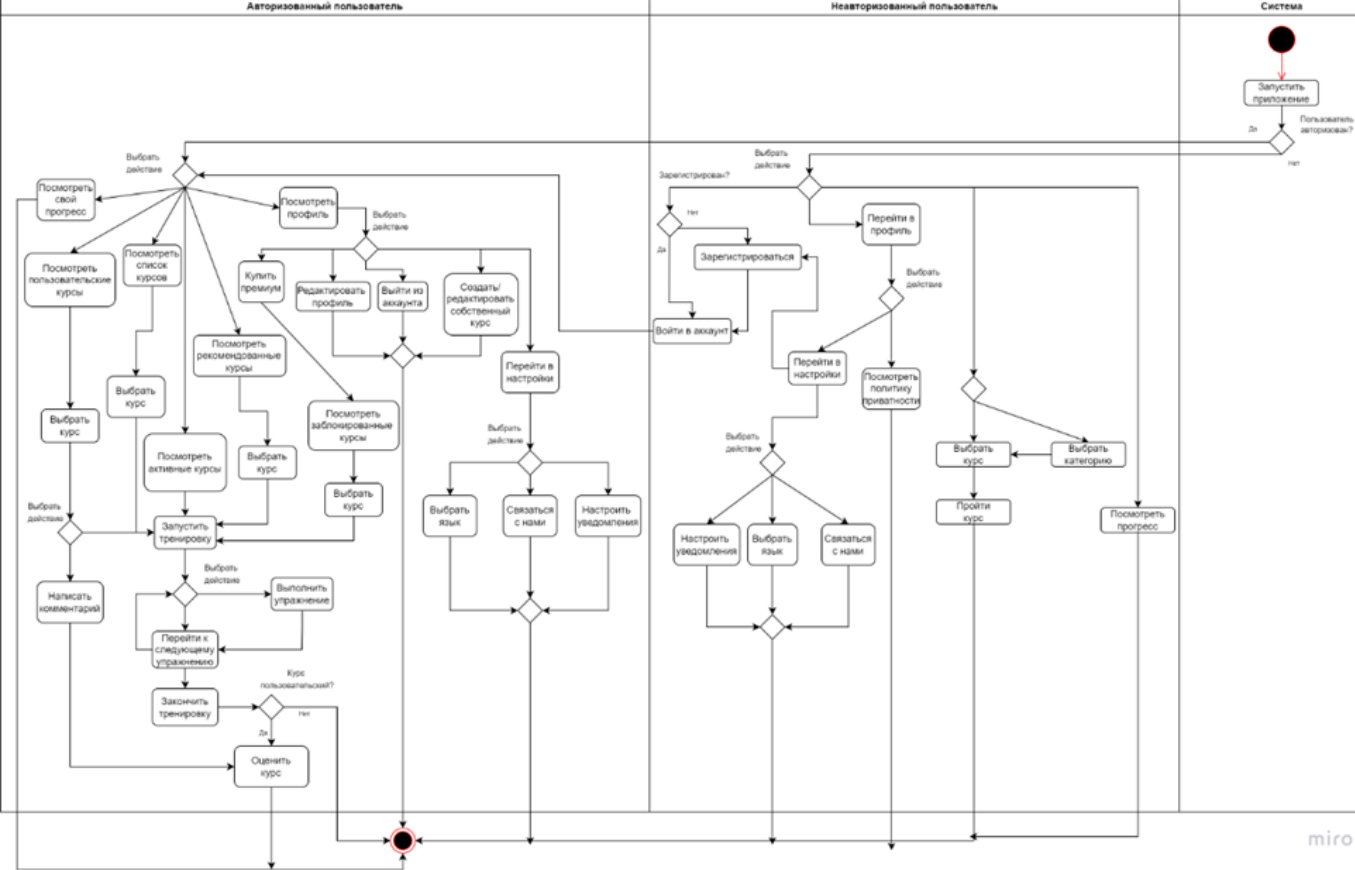
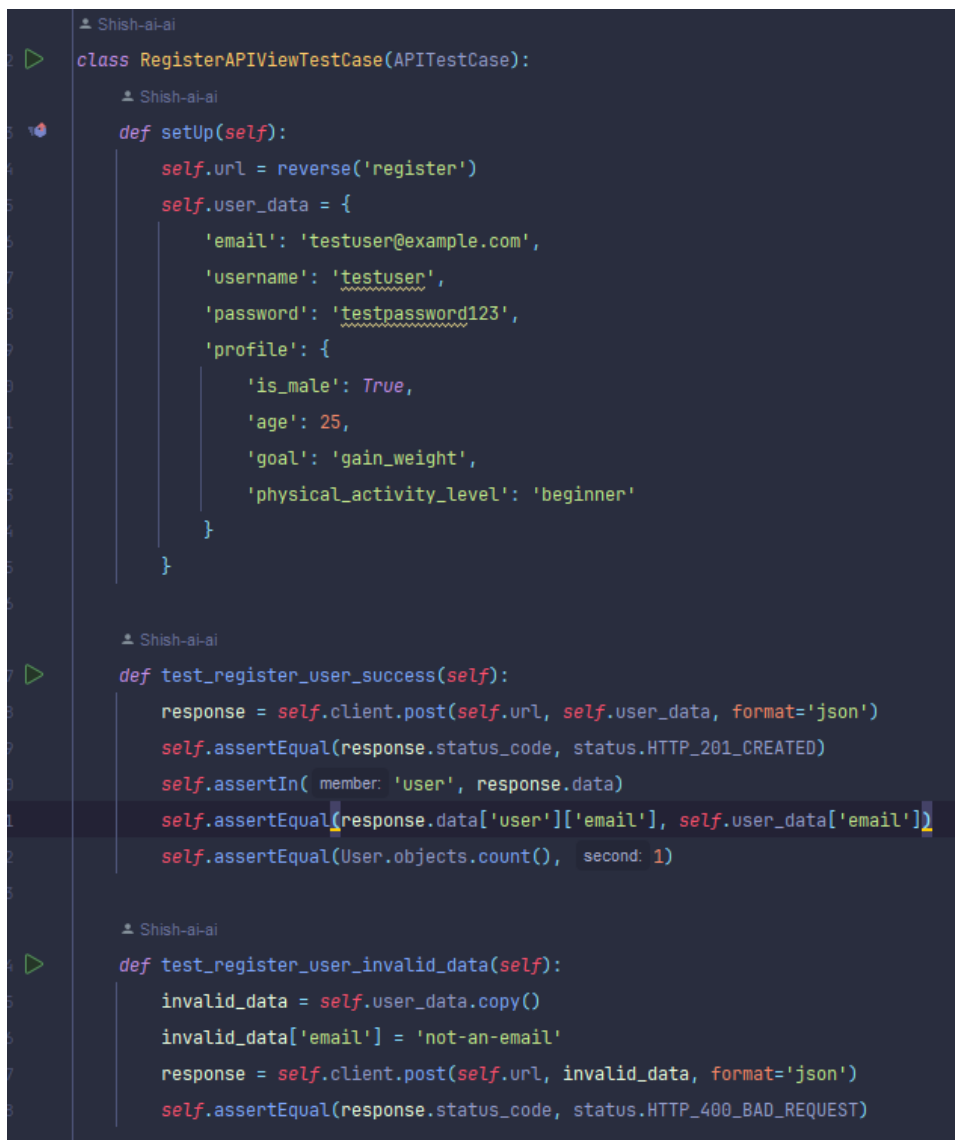


Рисунок 58 - Диаграмма активности

5 Тестирование

5.1 Unit-тестирование

Юнит-тестирование (unit testing) — это метод тестирования программного обеспечения, при котором отдельные модули или компоненты программы проверяются на правильность работы. Эти модули могут быть отдельными функциями, методами, классами или объектами. Основной целью юнит-тестирования является убедиться, что каждый элемент кода работает согласно ожиданиям в изоляции от других частей системы.



```
class RegisterAPIViewTestCase(APITestCase):
    def setUp(self):
        self.url = reverse('register')
        self.user_data = {
            'email': 'testuser@example.com',
            'username': 'testuser',
            'password': 'testpassword123',
            'profile': {
                'is_male': True,
                'age': 25,
                'goal': 'gain_weight',
                'physical_activity_level': 'beginner'
            }
        }

    def test_register_user_success(self):
        response = self.client.post(self.url, self.user_data, format='json')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertIn('user', response.data)
        self.assertEqual(response.data['user']['email'], self.user_data['email'])
        self.assertEqual(User.objects.count(), second=1)

    def test_register_user_invalid_data(self):
        invalid_data = self.user_data.copy()
        invalid_data['email'] = 'not-an-email'
        response = self.client.post(self.url, invalid_data, format='json')
        self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
```

Рисунок 59 - Пример unit-теста

5.2 Тест планы

Тест-план — это документ, описывающий все аспекты процесса тестирования программного обеспечения. Он служит руководством для тестировщиков и включает в себя стратегии, цели, объем и ресурсы, необходимые для проведения тестирования. Основная цель тест-плана — обеспечить организованный и систематический подход к тестированию, минимизировать риски и гарантировать, что продукт соответствует требованиям и ожиданиям пользователей.

```
1. **критический** - дефект блокирует дальнейшее тестирование приложения, должен быть устранен немедленно.
2. **Важный** - дефект негативно влияет на основные функции системы, из-за чего они начинают работать некорректно.
3. **Обычный** - дефект вносит минимальное отклонение от требований к системе.
4. **Малозначительный** - минимальное нефункциональное влияние на работу системы и на ход тестирования.

## Функциональное тестирование

Цель функционального тестирования - убедиться, что весь программный продукт работает в соответствии с требованиями в Цел

### Критерии входа

- Наличие подготовленных тестовых сценариев.

### Критерии выхода

- Все тесты пройдены, отсутствуют дефекты в статусе выше, чем малозначительный.

## Риски в функционале приложения
```

Риск	Вероятность	Ущерб	Минимизация
Нарушение подключения к серверу (PostgreSQL, uWSGI)	Высокий	Высокий	Перед проведением тестирования у
Неправильная сериализация / десериализация данных	Средний	Высокий	Исправление ошибки разработчиком
Некорректная или неработающая бизнес-логика	Средний	Средний	Исправление ошибки разработчиком

Рисунок 60 - Пример тест-плана для серверной части приложения

```

## Классификация дефектов

Всем найденным дефектам необходимо присвоить один из четырех уровней важности:

1. **Критический** – дефект блокирует дальнейшее тестирование приложения, должен быть устранен немедленно.
2. **Важный** – дефект негативно влияет на основные функции системы, из-за чего они начинают работать некорректно.
3. **Обычный** – дефект вносит минимальное отклонение от требований к системе.
4. **Малозначительный** – минимальное нефункциональное влияние на работу системы и на ход тестирования.

## Функциональное тестирование

Цель функционального тестирования – убедиться, что весь программный продукт работает в соответствии с требованиями, описанными в пункте Классификация дефектов.

### Критерии входа

- Наличие подготовленных тестовых сценариев.

### Критерии выхода

- Все тесты пройдены, отсутствуют дефекты в статусе выше, чем малозначительный.

## Риски в функционале приложения



| Риск                                                            | Вероятность | Ущерб   | Минимизация              |
|-----------------------------------------------------------------|-------------|---------|--------------------------|
| Нарушение подключения к серверу (API, серверная часть) клиента. | Высокий     | Высокий | Перед проведением тестов |
| Неправильная обработка данных сервера                           | Средний     | Высокий | Исправление ошибки       |
| Некорректная или неработающая бизнес-логика клиента             | Средний     | Средний | Исправление ошибки       |



## Необходимые инструменты

- **Flutter SDK** для разработки и тестирования приложения.
- **Dart** для написания кода и тестов.
- **Emulators/Physical devices** для запуска и тестирования приложения.
- **Integration testing tools** для проверки взаимодействия приложения с серверной частью.
- **CI/CD pipeline** для автоматизации тестирования и деплоя приложения.

## Заключение

```

Рисунок 61 - Пример тест-плана для клиентской части приложения

6 Аналитика

Используя инструмент AppMetrica был проведён анализ количества пользователей, которые, открыв приложение, прошли весь пользовательский путь по регистрации, авторизации и первому входу в приложение.

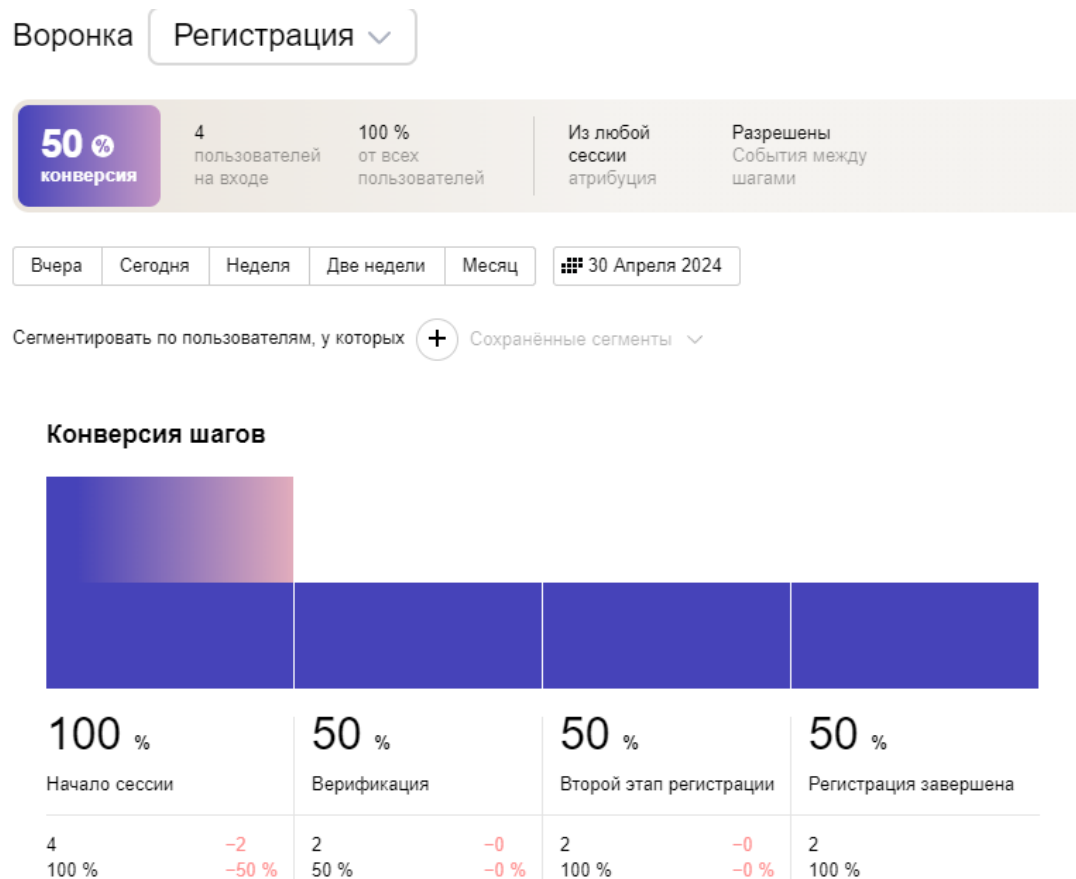


Рисунок 62 - Воронка «Регистрация» с диаграммой конверсии шагов

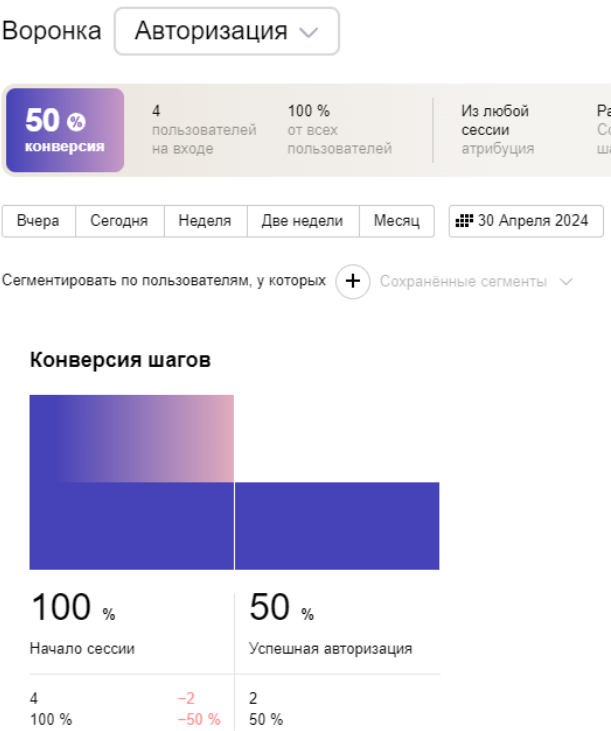


Рисунок 63 - Воронка «Авторизация» с диаграммой конверсии шагов

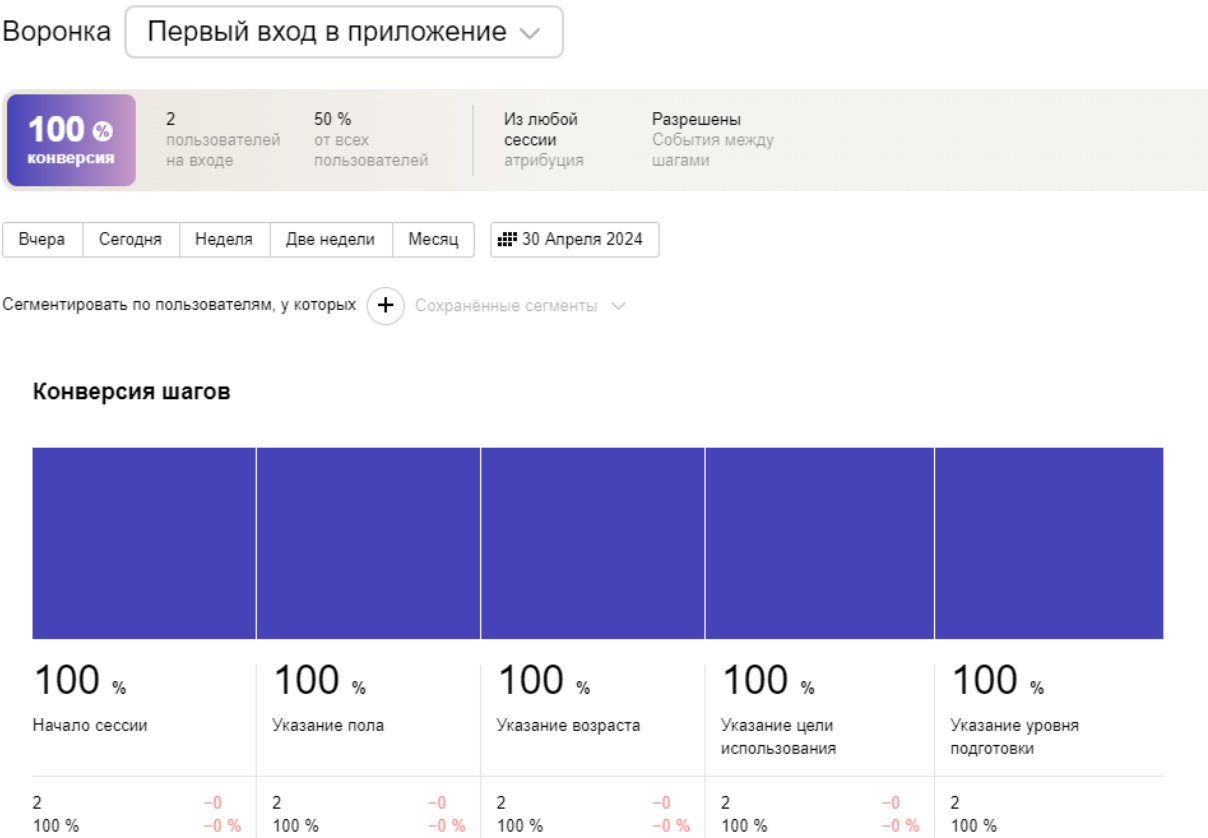


Рисунок 64 - Воронка «Первый вход в приложение»

Заключение

Введение новых функций и возможностей в наше приложение FitHub для индивидуальных домашних тренировок является стратегически важным шагом для расширения его функциональности и привлекательности на рынке. В современном мире фитнеса и здорового образа жизни цифровые платформы становятся неотъемлемой частью повседневной жизни пользователей, и наше приложение стремится предоставить удобную и многофункциональную платформу для тренировок и улучшения физической формы.

Одним из ключевых нововведений в нашем приложении является система индивидуальных тренировочных курсов, что открывает возможности для привлечения пользователей с разными уровнями подготовки и целями. Это значительно расширяет нашу базу пользователей и способствует увеличению вовлеченности. Индивидуальные курсы особенно привлекательны для тех, кто хочет получать персонализированные рекомендации и планы тренировок для достижения своих целей.

Интеграция сообщества пользователей в наше приложение является еще одним важным шагом в развитии платформы. Это позволяет пользователям делиться своими достижениями, комментариями и оценками тренировок, а также просматривать и выполнять тренировки, созданные другими пользователями. Такой подход способствует развитию социальной компоненты и мотивации пользователей, обеспечивая им поддержку и вдохновение.

Анализ рынка конкурентов и выбор технологического стека являются неотъемлемой частью нашей стратегии. Мы провели анализ конкурентного окружения и тщательно выбрали оптимальные технологии для разработки приложения. Такой подход позволяет нам создать конкурентоспособный продукт, отвечающий требованиям рынка и обеспечивающий эффективное функционирование платформы.

В заключение, разработанное нами приложение FitHub для индивидуальных домашних тренировок представляет собой инновационное и перспективное решение на рынке цифрового фитнеса. Мы стремимся предоставить пользователям удобство и широкие возможности для занятий спортом и улучшения физической формы, а также способствовать развитию и поддержке фитнес-сообщества. Мы уверены, что наше приложение будет успешно конкурировать на рынке и удовлетворит потребности современных пользователей.

Список используемых источников

1. Мобильное приложение – Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Мобильное_приложение (дата обращения: 06.06.2024).
2. Фронтенд – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Фронтенд> (дата обращения: 06.06.2024).
3. Клиент (информатика) – Википедия [Электронный ресурс] – Режим доступа: [https://ru.wikipedia.org/wiki/Клиент_\(информатика\)](https://ru.wikipedia.org/wiki/Клиент_(информатика)) (дата обращения: 06.06.2024).
4. Бэкенд – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Бэкенд> (дата обращения: 06.06.2024).
5. Сервер (программное обеспечение) – Википедия [Электронный ресурс] – Режим доступа: [https://ru.wikipedia.org/wiki/Сервер_\(программное_обеспечение\)](https://ru.wikipedia.org/wiki/Сервер_(программное_обеспечение)) (дата обращения: 06.06.2024).
6. Фреймворк – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Фреймворк> (дата обращения: 06.06.2024).
7. Dart programming language | Dart [Электронный ресурс] – Режим доступа: <https://dart.dev/> (дата обращения: 06.06.2024).
8. Flutter - Build apps for any screens [Электронный ресурс] – Режим доступа: <https://flutter.dev/> (дата обращения: 06.06.2024).
9. API – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/API> (дата обращения: 06.06.2024).
10. Документация Flutter [Электронный ресурс]. – Режим доступа: <https://docs.flutter.dev/> (дата обращения 10.05.2024).
11. Вигерс К. Разработка требований к программному обеспечению/ Карл Вигерс, Джой Бити. — М.: Изд-во Русская редакция, 2014. — 736 с.
12. GitHub – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/GitHub> (дата обращения: 06.06.2024).

13. PostgreSQL – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/PostgreSQL> (дата обращения: 06.06.2024).
14. Python – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Python> (дата обращения: 06.06.2024).
15. Docker – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Docker> (дата обращения: 06.06.2024).
16. JSON Web Token – Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/JSON_Web_Token (дата обращения: 06.06.2024).

