

Федеральное государственное образовательное бюджетное
учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финансовый университет)

Факультет социальных наук и массовых коммуникаций
Департамент массовых коммуникаций и медиабизнеса

Выпускная квалификационная работа
на тему
«Разработка мобильного приложения для автоматизации процессов
взаимодействия клиентов с фитнес-клубом.»

Направление подготовки: 09.03.03 Прикладная информатика
Профиль: «ИТ-сервисы и технологии обработки данных в экономике и
финансах»

Выполнил студент учебной группы
ПИ19-3
Данилин А. А.

(подпись)

Руководитель доцент, к.т.н.
Калажоков З.Х.

(подпись)

**ВКР соответствует предъявляемым
требованиям**

Руководитель Департамента:
К. Т. Н.

(подпись) Петросов Д. А.

«____» _____ 2023 г.

Москва – 2023 г.

Оглавление

| | |
|--|----|
| ВВЕДЕНИЕ..... | 3 |
| 1. АНАЛИЗ ВАЖНОСТИ ПОСТАВЛЕННОЙ ЗАДАЧИ..... | 5 |
| 1.1 Почему важны приложения..... | 5 |
| 1.2 Важность спорта в современном мире | 6 |
| 1.3 Обзор ситуации на рынке | 6 |
| 1.4 Преимущества созданного решения | 7 |
| 2. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ | 10 |
| 2.1 Разбор поставленной цели..... | 10 |
| 2.2 Python..... | 12 |
| 2.3 PIP | 13 |
| 2.4 Django | 15 |
| 2.5 Django REST..... | 17 |
| 2.6 SQLite..... | 19 |
| 2.7 Android studio..... | 20 |
| 2.8 Java..... | 22 |
| 2.9 Volley | 23 |
| 2.10 Bluestacks..... | 24 |
| 3. РАЗРАБОТКА BACKEND-СЛУЖБЫ..... | 26 |
| 3.1 Разработка базы данных | 26 |
| 3.2 Разработка обработчиков | 32 |
| 3.3 Общая структура backend-службы. | 35 |
| 3.4 Панель администрации | 36 |
| 4. РАЗРАБОТКА FRONTEND-СЛУЖБЫ..... | 40 |
| 4.1 Окно новостей..... | 41 |

| | |
|---|----|
| 4.2 Окно записной книжки | 43 |
| 4.3 Окно создания записей | 45 |
| 4.4 Окно расписания | 50 |
| 4.5 Окно профиля | 52 |
| 4.6 Регистрация и вход в приложение..... | 53 |
| ЗАКЛЮЧЕНИЕ | 56 |
| СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ | 57 |

ВВЕДЕНИЕ

В мире информационных технологий повседневная жизнь человека стала удобной как никогда. Человек обладает доступом к информации и контролем над своей жизнью из любой точки мира. Благодаря развитию технологий и интернета, зачастую нет нужды выходить из дома, и даже куда – то звонить, чтобы получить, узнать или сделать то, что нужно, все уже доступно удаленно. Человек всеми доступными способами пытается сократить необходимость движения и контактов с людьми, при этом сохраняя качество жизни, поэтому и создаются всевозможные веб-сайты и приложения, позволяющие делать все, от чтения новостей, до покупки дорогостоящего имущества, что раньше занимало много времени и было целым событием. Мир становится удобнее и доступнее с каждым днем.

Однако не все сферы жизни могут похвастаться таким развитием. Во многих областях все еще наблюдается либо полное отсутствие какого – либо удаленного доступа к услугам, либо оно крайне ограничено настолько, что не имеет никакого смысла и требует дальнейшего взаимодействия с другими элементами цепи, которые человек так активно пытается устранить. Одной из таких сфер является сфера фитнеса и спорта.

Целью данной работы была разработка мобильного приложения для автоматизации процессов взаимодействия клиентов с фитнес-клубом. Таким образом, была поставлена необходимость в создании не просто рекламного агрегатора, которыми являются приложения большинства фитнес-клубов, а инструмента, которым клиенты будут пользоваться ежедневно. Созданное решение должно содержать весь функционал, который может потребоваться большинству посетителей клуба

Для достижения цели было произведено исследование рынка мобильных приложений фитнес клубов России и анализ их функционала для дальнейшего создания проекта, способного конкурировать при текущей ситуации как по

функциональному наполнению, так и по производительности, используя современные технологии и инструменты.

1. АНАЛИЗ ВАЖНОСТИ ПОСТАВЛЕННОЙ ЗАДАЧИ

1.1 Почему важны приложения

Создание приложений спортивных залов имеет множество преимуществ, которые становятся все более важными сейчас, в эру цифровизации и удаленной работы. Создание приложений крайне необходимо, потому что:

- В последние годы люди стали более заботиться о своем здоровье, и фитнес стал неотъемлемой частью этой тенденции. Следовательно, спрос на услуги спортивных залов и фитнес-тренеров постоянно растет.
- Приложения спортивных залов позволяют создавать персонализированные тренировки, учитывающие индивидуальные потребности и цели каждого клиента. Это может быть особенно полезно для начинающих и тех, кто хочет достичь конкретной цели, такой как похудение или набор мышечной массы.
- Приложения спортивных залов могут помочь улучшить опыт клиентов, предоставляя им информацию о расписании занятий, доступных тренерах, технике выполнения упражнений и т. д. Кроме того, многие приложения позволяют клиентам легко отслеживать свой прогресс и достигать своих целей.
- Создание приложений спортивных залов может помочь увеличить доходы, предоставляя дополнительные услуги, такие как продажа спортивного оборудования, диетических продуктов и т. д. Кроме того, приложения могут быть использованы для продвижения специальных предложений и акций, которые помогут привлечь новых клиентов.
- Приложения спортивных залов позволяют клиентам бронировать занятия и связываться с тренерами в режиме онлайн, что упрощает процесс и сокращает время и затраты на административные задачи.

1.2 Важность спорта в современном мире

Спорт – это неотъемлемая часть современного мира, которая оказывает огромное влияние на различные аспекты нашей жизни. Он стал не только физической, но и культурной составляющей нашего общества, а его влияние на экономику, политику, науку, медиа и на человека неизмеримо.

Один из важнейших аспектов роли спорта в современном мире – это его влияние на здоровье населения. Занимаясь спортом, человек улучшает физическое и психическое состояние, повышает иммунитет и продлевает свою жизнь. Кроме того, спортивные соревнования являются одним из самых популярных мероприятий для туристов, что способствует развитию туризма и экономики в целом.

Спорт также играет важную роль в социальной жизни общества. Он объединяет людей, независимо от их возраста, пола, социального статуса и национальности, создавая единство и духовное согласие. Спортивные мероприятия могут стать отличной площадкой для общения и укрепления дружеских связей.

В современном мире спорт имеет также большое значение для различных бизнес-секторов. Спортивная индустрия – это многомиллиардный бизнес, который охватывает множество областей, начиная от производства спортивного инвентаря и заканчивая организацией масштабных спортивных событий.

Поэтому крайне важно создание всех возможных условий для комфортного доступа к спорту и наличия всего самого необходимого под рукой именно сейчас, потому что эта сфера растет бурными темпами, и чем раньше получится на ней остановиться, тем выше будет лояльность клиентов к сети.

1.3 Обзор ситуации на рынке

Сегодня рынок приложений для спортивных залов переживает бурный рост. Согласно исследованиям, количество пользователей мобильных приложений в этой нише продолжает расти. Это связано с тем, что все больше

людей стремятся к здоровому образу жизни и активной физической нагрузке. Спортивные залы становятся более популярными, а приложения помогают сделать тренировки более эффективными и удобными, однако, несмотря на это, в России на текущий момент, большинство спортивных залов и комплексов либо вовсе не имеют своих сайтов и приложений, либо имеют совсем базовые решения, которые вовсе не получают трафика, потому что функциональное наполнение минимально и отстает от современных стандартов и клиентских ожиданий на десятки лет,

Те же, кто все – таки инвестировал в комфорт своих клиентов, все равно не имеют требуемого функционала, заставляя своих пользователей искать другие решения, которые могут удовлетворить все их потребности, или вовсе отказываться от приложений, что в итоге вредит в том числе и спортивным комплексам, потому что теряется вовлеченность клиентов в спортивные мероприятия, которые пользователи могли бы заинтересовать, если бы они были представлены в более доступной манере

Одним из основных преимуществ приложений для спортивных залов должна быть возможность получать персонализированные рекомендации по тренировкам и питанию, а также возможность отслеживать свой прогресс. Клиенты должны иметь возможность выбирать индивидуальные программы тренировок и следить за своими достижениями. Это не только поможет сделать тренировки более эффективными, но и повышает мотивацию клиентов.

1.4 Преимущества созданного решения

В отличие от большинства приложений на Российском рынке спортивных залов, приоритетом разработанного приложения была обратная связь и взаимодействие.

До сих пор, ни одно приложение на российском рынке не предоставляет обратной связи, потому что единственный их функционал – показ рекламных предложений и акций, которые клиенту, уже купившему членство в клубе

просто не интересно, а для получения доступа к приложению потенциальным клиентам нужен абонемент, что в принципе убивает весь смысл создания и существования таких приложений.

Если посетители все же захотят получить обратную связь, например создание расписание собственных тренировок и отслеживание прогресса, им приходится прибегать к помощи сторонних решений, таких как:

- Fitbit - один из лидеров в области фитнес-технологий, который предлагает различные устройства и мобильные приложения для отслеживания физической активности и здоровья.
- MyFitnessPal - приложение для отслеживания питания и физической активности, которое позволяет пользователям записывать свои ежедневные приемы пищи, тренировки и прогресс в достижении целей.
- Nike Training Club - приложение, созданное Nike, которое предлагает пользователю персонализированные тренировки, а также советы по здоровому образу жизни и питанию.
- Adidas Training by Runtastic - приложение, разработанное Adidas, которое позволяет пользователям отслеживать свои тренировки, установить цели и получать персонализированные рекомендации.
- FitOn - приложение для тренировок, которое предлагает бесплатные и платные тренировки с инструкторами известных фитнес-брендов.
- Peloton - компания, предлагающая оборудование и приложение для тренировок в домашних условиях, которое позволяет пользователям
- Apple Fitness+ - новое приложение от Apple, которое позволяет пользователям получить доступ к множеству видеотренировок, а также отслеживать свой прогресс и достижения. Также имеет интеграцию со своими умными часами, способными в реальном времени отслеживать физическое состояние пользователя и

предоставлять соревновательный аспект тренировкам для создания стимула продолжать занятия.

Одной из главных тенденций на рынке мобильных приложений фитнес клубов является переход к более персонализированным и интерактивным решениям. Компании стараются предоставить пользователям индивидуальный подход к тренировкам, а также использовать технологии и данные, чтобы улучшить опыт пользователя. И по данным многих исследований, пользователи предпочитают именно такой подход к созданию приложений, что из-за чего решения на российском рынке просто не могут никого удовлетворить и ставят под вопрос смысл своего существования.

Поэтому, мной было разработано приложение, предоставляющее тот список возможностей, которые пользователи так активно ищут на рынке мобильных приложений. Оно было создано как приложение для паритета, чтобы удовлетворить потребности как новичков, так и уже опытных пользователей.

Данное решение позволит владельцам фитнес клубов предоставлять клиентам рекламные предложения с наименьшей периодичностью, в то же время у клиентов будет стимул посещать данное приложение, не зависимо от того пользуются ли они особыми предложениями клуба, или же являются рядовыми посетителями, каждый найдет то, что его интересует.

По моему мнению именно такого подхода не хватает владельцам спортивных комплексов к разработке мобильных приложений в наше время.

2. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

2.1 Разбор поставленной цели

Целью данной работы являлась Разработка мобильного приложения для автоматизации процессов взаимодействия клиентов с фитнес-клубом. Для создания наилучшего пользовательского опыта взаимодействия было принято решение создать клиент-серверное приложение.

Клиент-серверная архитектура является одной из основных моделей в современной компьютерной сетевой технологии. Она представляет собой распределенную систему, в которой компьютеры взаимодействуют друг с другом, обмениваясь информацией и ресурсами.

В модели клиент-сервер система разбивается на две основные части: клиентскую и серверную (рис. 1). Клиент – это программа или устройство, которое запрашивает информацию или услуги у сервера, а сервер – это программа или устройство, которое предоставляет запрашиваемую информацию или услуги клиенту. Взаимодействие между клиентом и сервером происходит посредством сетевого протокола, который обеспечивает передачу данных между устройствами.

Клиент-серверная архитектура используется во многих областях, в том числе в сетевых играх, приложениях, ориентированных на работу с базами данных, веб-сайтах, облачных вычислениях и т. д. Она позволяет создавать сложные системы, которые могут масштабироваться и быть устойчивыми к сбоям.

Одним из преимуществ клиент-серверной архитектуры является возможность централизованного управления данными и ресурсами. Это означает, что данные могут храниться на сервере, а клиенты могут запрашивать их по мере необходимости. Это упрощает управление и сокращает количество дублирования данных, что повышает эффективность работы системы в целом.

Кроме того, клиент-серверная архитектура обеспечивает безопасность и защиту данных. Все данные, которые передаются между клиентом и сервером,

могут быть зашифрованы, что обеспечивает конфиденциальность и защиту от несанкционированного доступа.

Таким образом, клиент-серверная архитектура является одной из самых распространенных моделей в современном компьютерном мире. Она обеспечивает эффективное и безопасное взаимодействие между клиентами и серверами, а также позволяет создавать сложные системы, которые могут масштабироваться и быть устойчивыми к сбоям.

Для удовлетворения потребностей клиент – серверной архитектуры был составлен следующий список инструментов: Python, PIP, Django, Django-REST, SQLite, Android Studio, Java, Volley, Bluestacks

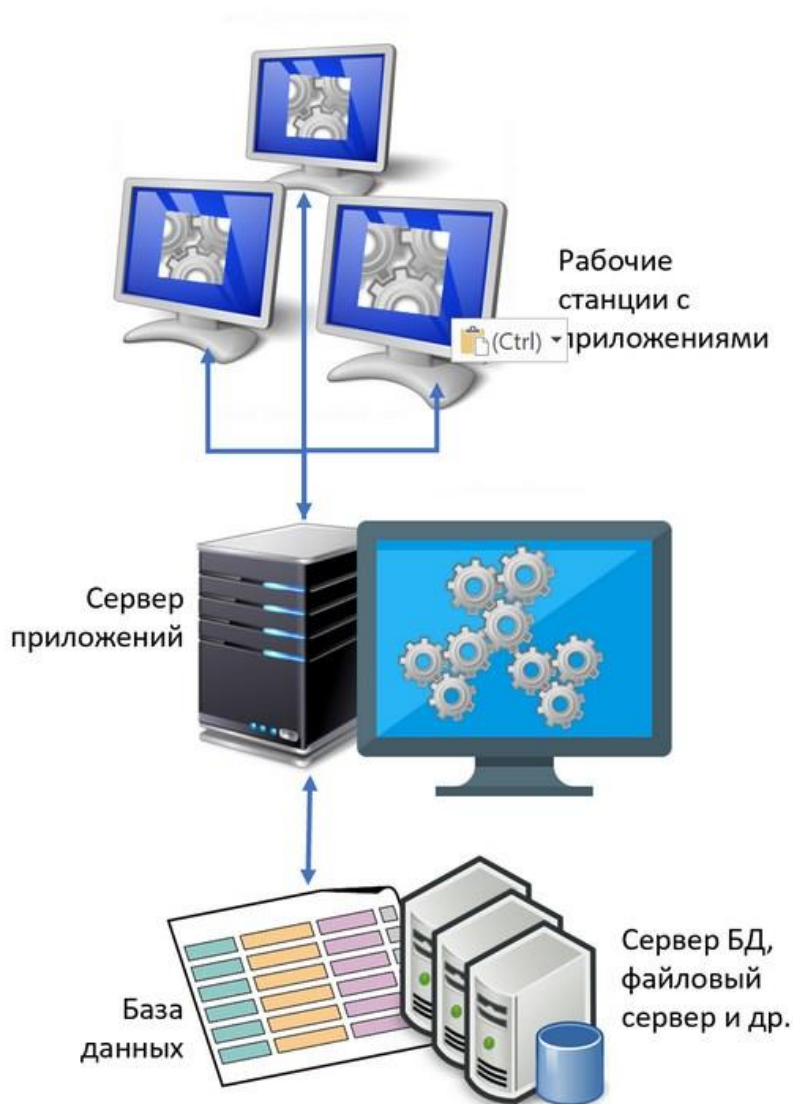


Рисунок 1.

2.2 Python

Для разработки backend-службы был выбран python.

Python - это интерпретируемый, высокоуровневый, объектно-ориентированный язык программирования с динамической типизацией, он является одним из наиболее популярных языков программирования для разработки серверного backend-кода, благодаря ряду преимуществ, которые он предоставляет, таких как:

- Простота и читаемость кода: простой и понятный синтаксис языка позволяет разработчикам создавать высококачественный код с минимальными усилиями.
- Широкий набор библиотек и фреймворков для разработки серверного backend-кода. Один из самых популярных фреймворков - Django - предоставляет готовое решение для создания высокопроизводительных веб-приложений и API.
- Масштабируемость. Python серверы могут быть легко масштабированы на большие нагрузки благодаря использованию распределенных систем и облачных сервисов.
- Высокую скорость разработки. Благодаря большому количеству высококачественных готовых модулей и библиотек, разработчики могут быстро создавать и разворачивать приложения, что особенно важно для бизнес-задач, где время имеет решающее значение.
- Асинхронное программирование. Асинхронное программирование в Python — это методика написания кода, которая позволяет создавать приложения, которые могут выполнять несколько задач одновременно, не блокируя главный поток выполнения. Она основана на использовании событийно-ориентированной архитектуры и концепции неблокирующего ввода/вывода. В асинхронном программировании операции выполняются параллельно в нескольких потоках, но каждая из них работает

независимо, не блокируя другие потоки и не прерывая основной поток приложения, что позволяет создавать более отзывчивые и производительные приложения, особенно при работе с сетью и базами данных.

Выбранной версией python была версия 3.10. Важными особенностями данной версии являются множество усовершенствований в различных областях, таких как улучшенная поддержка синтаксиса и операций с декораторами, улучшения в обработке ошибок и исключений, а также новые возможности для работы с файловой системой.

Также, немаловажным нововведением была повышенная скорость выполнения. Новая версия языка программирования включает множество оптимизаций и улучшений, которые делают его более быстрым и эффективным в работе с большими объемами данных, что является крайне необходимым для серверов.

2.3 PIP

Для установки модулей и управления ими был выбран менеджер пакетов `pip`.

Менеджер пакетов — это инструмент, который позволяет управлять установкой, обновлением и удалением программных пакетов на компьютере.

Пакеты представляют собой наборы кода, библиотек, модулей и зависимостей, которые позволяют создавать приложения, веб-сайты и другие программы. При этом важно иметь возможность легко устанавливать, обновлять и удалять эти пакеты, чтобы разработка была более эффективной и удобной.

Менеджеры пакетов предназначены для того, чтобы упростить процесс управления пакетами и зависимостями. Они обычно включают в себя следующие функции:

- Установка пакетов - позволяет легко устанавливать нужные пакеты и их зависимости на компьютер.
- Обновление пакетов - позволяет обновлять уже установленные пакеты до последней версии.
- Удаление пакетов - позволяет удалить пакеты и их зависимости с компьютера.
- Управление зависимостями - позволяет управлять зависимостями пакетов и установкой необходимых зависимостей.
- Поиск пакетов - позволяет искать нужные пакеты в репозитории и устанавливать их.

PIP (Python Package Installer) является стандартным пакетным менеджером, который используется для управления сторонними пакетами и их зависимостями. PIP позволяет удобно и быстро устанавливать, обновлять и удалить пакеты Python в вашей системе.

PIP был введен в 2008 году для замены старой системы установки пакетов EasyInstall. С тех пор он стал стандартом в сообществе Python и широко используется для управления зависимостями в большинстве проектов на Python.

PIP позволяет быстро и легко устанавливать сторонние пакеты с помощью команды `pip install`. Эта команда автоматически загружает пакеты из репозитория PyPI (Python Package Index) и устанавливает их в систему. PyPI — это крупнейший репозиторий Python-пакетов, где разработчики могут опубликовывать свои пакеты для использования другими.

Кроме того, PIP позволяет управлять зависимостями пакетов. При установке новых пакетов PIP автоматически загружает и устанавливает все необходимые зависимости. Это позволяет упростить процесс их установки и обеспечить совместимость между разными версиями.

PIP также позволяет обновлять и удалять модули. Команда `pip install --upgrade` позволяет обновлять их до последней версии, а команда `pip uninstall` - удалить из системы.

2.4 Django

В качестве фреймворка для разработки backend-службы был выбран фреймворк Django.

Фреймворк (англ. framework) - это программный инструмент, предназначенный для упрощения разработки программного обеспечения. Фреймворк предоставляет набор готовых компонентов и библиотек, которые можно использовать в своем приложении. Он определяет структуру и правила для разработки, устанавливает определенные ограничения и рекомендации, которые позволяют быстрее и эффективнее создавать программное обеспечение.

Фреймворки разрабатываются, чтобы облегчить разработку программного обеспечения, ускорить процесс разработки, снизить количество ошибок и повысить безопасность. Они могут быть написаны на разных языках программирования и использоваться для создания разных видов приложений: от веб-сайтов и мобильных приложений до игр и настольных приложений.

Основными компонентами фреймворка являются:

- Библиотеки: предоставляют готовый код для решения определенных задач, например, работу с базами данных или отправку электронных писем.
- Инструменты разработки: облегчают процесс разработки, такие как средства отладки и тестирования, утилиты сборки и документации.
- Архитектура: определяет структуру приложения и правила взаимодействия между его компонентами.
- Паттерны проектирования: определенные подходы к разработке приложений, которые повышают качество и снижают время разработки.

Фреймворки могут быть общего назначения, предназначенные для разных типов приложений, или специализированные для конкретных задач,

таких как разработка веб-приложений или игр. Кроме того, существуют фреймворки с открытым исходным кодом, такие как Django и Flask для веб-разработки на Python, и фреймворки с закрытым исходным кодом, такие как .NET Framework от Microsoft.

Основными преимуществами использования фреймворков являются повышение производительности разработки, повторное использование кода, упрощение обслуживания приложения и обновления, а также снижение рисков ошибок и повышение безопасности.

Django — это высокоуровневый веб-фреймворк на языке Python, который позволяет быстро и удобно создавать веб-приложения. Он основан на принципах модели-представления-контроллера (MVC), которые позволяют разделять приложение на три части: модели, которые представляют данные, представления, которые отображают данные на странице, и контроллеры, которые обрабатывают запросы и связывают модели и представления.

Он предоставляет разработчикам множество инструментов и библиотек, упрощающих и ускоряющих создание веб-приложений, помимо этого, в него включены инструменты для обработки HTTP-запросов, работы с базами данных, создания шаблонов и много других.

Одной из ключевых особенностей Django является его ORM (Object-Relational Mapping), который позволяет работать с базами данных без написания SQL-запросов. Django поддерживает различные базы данных, включая SQLite, MySQL и PostgreSQL.

Основными принципами фреймворка являются скорость разработки, повторное использование кода, разделение кода и верстки, обеспечение безопасности и простота в использовании.

Помимо того, django предлагает множество встроенных функций, таких как аутентификация, управление сессиями, администрирование и многое другое. Это позволяет разработчикам сосредоточиться на разработке бизнес-логики и функциональности приложения, не тратя много времени на написание базовых функций.

Преимущества выбора данного фреймворка:

- Быстрая разработка: Django позволяет быстро создавать веб-приложения, благодаря готовому функционалу и инструментам, таким как ORM (Object-Relational Mapping), шаблонизаторы, административный интерфейс и многие другие.
- Масштабируемость: Django поддерживает горизонтальное и вертикальное масштабирование, что позволяет увеличивать производительность и обрабатывать большое количество запросов.
- Безопасность: Django имеет множество инструментов для обеспечения безопасности веб-приложений, таких как защита от CSRF-атак, XSS-атак и SQL-инъекций.
- Гибкость: Django позволяет разрабатывать разнообразные веб-приложения, от простых блогов до сложных социальных сетей и электронной коммерции.
- Маршрутизация: Django обладает мощной системой маршрутизации URL, которая позволяет управлять тем, как обрабатываются запросы, и связывать URL-адреса с определенными представлениями, что упрощает процесс написания и поддержки кода, а также повышает его читабельность.

2.5 Django REST

Несмотря на все преимущества Django, его инструментария все равно недостаточно, поэтому было принято решение использовать фреймворк для Django, обладающий переработанным функционалом – Django REST.

Django REST framework (DRF) - это набор инструментов, созданный на основе фреймворка Django для разработки RESTful API веб-сервисов. DRF предоставляет множество готовых инструментов и методов для быстрой и

эффективной разработки API, позволяя разработчикам создавать API без необходимости создавать все с нуля.

Одним из ключевых преимуществ DRF является возможность разработки API в соответствии со стандартами RESTful архитектуры, которая облегчает коммуникацию между клиентской и серверной сторонами. Для этого DRF предоставляет встроенную поддержку сериализации и десериализации данных, а также возможность обработки запросов и ответов в различных форматах данных, таких как JSON, XML и других.

Сериализация в DRF это инструмент, который позволяет преобразовывать сложные структуры данных, такие как объекты Django модели, в форматы, которые могут быть переданы через API. Сериализатор также может преобразовывать полученные данные обратно в объекты модели.

Основная цель сериализатора - преобразование объектов модели в формат, который может быть передан по сети.

В Django REST Framework, Serializer представляет собой класс, который определяет модель, которую необходимо сериализовать, а также описывает формат, в котором необходимо представить данные. Класс Serializer предоставляет методы для сериализации и десериализации данных в объекты модели, валидации данных и многих других функций.

Serializer в Django REST способен возвращать внешние ключи словарем с полями и значениями, таким образом достаточно сделать один запрос к Serializer и получить все нужные значения без необходимости расписывать запросы.

DRF также предоставляет множество инструментов для авторизации, аутентификации и управления доступом, позволяя разработчикам создавать безопасные API, которые могут использоваться только авторизованными пользователями. Также есть возможность подключения различных плагинов и расширений, которые облегчают создание API и предоставляют больше функциональности.

2.6 SQLite

Архитектурой базы данных был выбран SQLite.

Архитектура базы данных (Database architecture) - это общая концепция, описывающая организацию базы данных (БД) и способы взаимодействия с ней. Она включает в себя не только физическую структуру БД, но и логическую модель данных, механизмы хранения, методы обработки данных, механизмы безопасности и резервного копирования данных.

Архитектура БД описывает, как данные хранятся и организованы, как они передаются между приложениями и базами данных, как они обрабатываются и какие механизмы обеспечивают безопасность данных. Архитектура БД важна для эффективной работы с данными, масштабируемости приложений и обеспечения безопасности данных.

Существует несколько типов архитектур БД:

- Клиент-серверная архитектура. В этом случае, БД расположена на сервере, который обрабатывает запросы от клиентов и возвращает результаты. Это позволяет многим пользователям работать с данными одновременно, а также обеспечивает масштабируемость и централизованное управление данными.
- Распределенная архитектура. Эта архитектура позволяет распределять данные на несколько серверов, чтобы обеспечить высокую доступность и масштабируемость. Каждый сервер может обрабатывать запросы от клиентов и передавать данные между собой.
- Централизованная архитектура. В этом случае, все данные хранятся в одном месте, что позволяет управлять ими централизованно и обеспечивать безопасность данных. Это также обеспечивает высокую доступность и эффективное управление данными.
- Распределенно-централизованная архитектура. Эта архитектура сочетает в себе преимущества централизованной и распределенной

архитектур. Данные хранятся в централизованной БД, но могут быть распределены на несколько серверов для обработки запросов.

SQLite — это компактная и быстрая встроенная реляционная база данных, которая обрабатывает транзакции без использования отдельного сервера. Она создана как библиотека на языке C, и может использоваться с большинством языков программирования, в том числе с Python.

SQLite отличается от большинства других СУБД тем, что не имеет отдельного серверного процесса, и вместо этого база данных хранится в одном файле на диске, что облегчает процесс установки и обслуживания.

Одним из основных преимуществ SQLite является его скорость и эффективность. Благодаря своей архитектуре она может обрабатывать тысячи транзакций в секунду, и может быть использована для хранения и обработки больших объемов данных.

Большой плюс SQLite в том, что ее можно использовать совместно с другими технологиями, такими как Django, Flask, Android и другие фреймворки и библиотеки, что делает ее идеальным выбором для разработки веб-приложений и мобильных приложений.

Также следует отметить, что SQLite обладает высокой надежностью и устойчивостью, так как при каждой транзакции происходит запись на диск, что уменьшает риск потери данных в случае сбоя системы.

2.7 Android studio

Android Studio - это интегрированная среда разработки (IDE) для создания приложений под операционную систему Android. Android Studio был создан на основе IntelliJ IDEA и предоставляет разработчикам множество инструментов и функций для создания мощных приложений.

Одна из главных особенностей Android Studio — это простота использования и интуитивно понятный интерфейс. Она предоставляет широкий

выбор инструментов и библиотек, которые упрощают процесс разработки приложений. Android Studio включает в себя интегрированные средства разработки, которые позволяют легко отлаживать и тестировать приложения. Она также включает в себя инструменты для создания макетов пользовательского интерфейса, а также поддержку многих языков программирования, включая Java, Kotlin и C++.

Android studio использует IntelliJ IDEA в качестве своего редактора, главным преимуществом которого является обеспечение разработчикам множество инструментов и функций для создания высококачественных и производительных приложений для Android. В IDE включены различные компоненты, такие как редактор кода, компилятор, отладчик, графический макетер, инструменты анализа кода, средства управления версиями и т. д. Android Studio также интегрирована с Android SDK (Software Development Kit), что позволяет разработчикам легко получать доступ к необходимым инструментам и ресурсам для создания приложений.

Одной из ключевых функций Android Studio является возможность создания APK-файлов, которые можно загружать в Google Play Store. Android Studio также предоставляет разработчикам возможность создания приложений с использованием различных платформ, таких как Android Wear, Android TV и Android Auto.

Android Studio также имеет открытый исходный код, что позволяет разработчикам вносить изменения и улучшать его. Интеграция с Git и другими системами контроля версий облегчает совместную работу над проектом.

Кроме того, Android Studio активно поддерживается сообществом разработчиков, которые обновляют его и добавляют новые функции. Благодаря этому Android Studio является одной из наиболее популярных IDE для создания приложений под Android.

2.8 Java

В качестве языка разработки frontend-службы был выбран язык Java.

Java - это объектно-ориентированный язык программирования, разработанный в 1995 году компанией Sun Microsystems. Язык был создан для написания кроссплатформенных приложений, которые могут выполняться на любой операционной системе, поддерживающей виртуальную машину Java (JVM).

Java стал одним из самых популярных языков программирования в мире благодаря своей простоте и универсальности. Он используется для разработки широкого спектра программного обеспечения, от веб-приложений до мобильных приложений и игр. Он был выбран, потому что:

- Большой выбор инструментов. Java имеет обширную библиотеку классов, которая предоставляет множество инструментов для разработки приложений. Библиотека Java содержит классы для работы с сетями, базами данных, графикой, звуком и многими другими функциями, что позволяет разработчикам сосредоточиться на разработке бизнес-логики приложения, не заботясь о реализации базовых функций.
- Безопасность. Java является одним из самых безопасных языков программирования. Благодаря своей строгой типизации и обширной библиотеке классов, которая предоставляет инструменты для обработки ошибок и исключений, Java позволяет разработчикам создавать надежные и безопасные приложения.
- Надежность и стабильность. Java прочно закрепила свое место как одного из популярнейших языков разработки для мобильных устройств за счет непревзойденной надежности и стабильности. Он предлагает широкий спектр возможностей и гибкость, позволяющие разработчикам создавать приложения любого уровня сложности, от простых до масштабных проектов.

- **Объектная ориентированность.** Java является объектно-ориентированным языком программирования, что означает, что он предоставляет удобный и гибкий способ организации кода, что упрощает разработку масштабных и сложных проектов. Java также имеет встроенную систему управления памятью, что позволяет упростить работу с памятью и избежать многих проблем, связанных с утечкой памяти.

2.9 Volley

Volley - это библиотека сетевых запросов для Android, разработанная компанией Google. Она облегчает выполнение HTTP-запросов и обработку ответов на сервере в асинхронном режиме. Volley предоставляет удобный интерфейс для работы с сетевыми запросами, а также обеспечивает автоматическое кэширование и очередь запросов.

Volley позволяет выполнять запросы в фоновом потоке, не блокируя основной поток пользовательского интерфейса. Это важно, так как сетевые запросы могут занять много времени и могут привести к задержкам в работе приложения. Она предоставляет простой интерфейс для определения обратных вызовов, которые будут вызваны после выполнения запроса.

Кроме того, встроенная поддержка кэширования результатов запросов позволяет сохранять результаты запросов в локальном хранилище и использовать их повторно при следующих запросах. Это улучшает производительность приложения и снижает нагрузку на сеть.

Еще одно преимущество Volley - возможность отмены запросов. Это позволяет избежать ненужных запросов и сократить использование сетевых ресурсов. Также имеется возможность задавать приоритеты для запросов, что позволяет управлять порядком их выполнения.

Volley работает на основе очереди запросов. Она позволяет управлять порядком выполнения запросов и задавать приоритеты для каждого запроса.

Это позволяет более гибко управлять сетевыми запросами и повышать производительность приложения.

2.10 Bluestacks

В качестве инструмента тестирования был использован эмулятор Bluestacks.

BlueStacks предоставляет возможность запуска Android – приложений на персональных компьютерах с операционной системой Windows.

BlueStacks использует технологию виртуализации, которая эмулирует аппаратное и программное обеспечение мобильных устройств на компьютере. Это позволяет приложениям работать на компьютере так же, как на мобильном устройстве, но с использованием более мощных ресурсов, которые доступны на ПК.

Удобный интерфейс и обширный набор функций, включающий поддержку многозадачности, синхронизацию с мобильными устройствами и многое другое позволяют пользователям удобно запускать и использовать мобильные приложения на ПК, что может быть полезно для разработчиков и тестировщиков.

BlueStacks предоставляет ряд преимуществ для мобильной разработки:

- Тестирование приложений на ПК: BlueStacks позволяет разработчикам тестировать и отлаживать мобильные приложения на ПК, что может значительно ускорить процесс разработки.
- Удобство работы с мобильными приложениями: BlueStacks позволяет запускать мобильные приложения на ПК, что делает работу с ними более удобной и эффективной, особенно если нужно работать с несколькими приложениями одновременно.
- Работа с мобильными приложениями на большом экране: запуск мобильных приложений на ПК с помощью BlueStacks позволяет работать с ними на большом экране, что может быть особенно

удобно для тонкой настройки внешнего вида приложения и верстки интерфейса.

- Синхронизация между устройствами: BlueStacks позволяет синхронизировать данные между ПК и мобильными устройствами, что может быть полезно для разработки мобильных приложений и игр, которые работают на разных платформах.
- Доступ к Google Play: BlueStacks предоставляет доступ к магазину приложений Google Play, что позволяет быстро и удобно скачивать, и устанавливать мобильные приложения на ПК.

3. РАЗРАБОТКА BACKEND-СЛУЖБЫ

3.1 Разработка базы данных

В качестве backend-службы было разработано приложение на языке Python 3.10 с использованием фреймворка Django REST.

В состав приложения входят база данных, обработчики запросов и ссылки.

База данных содержит:

Activities – таблица упражнений. В ней хранятся как упражнения персональные, так и групповые. Содержит следующие атрибуты:

- **id** – уникальный числовой идентификатор упражнения
- **name** – строковое поле, хранящее информацию о названии упражнения. Максимальная длина - 100 символов, обязано быть уникальным
- **beginner_friendly** – логическое поле, содержащее информацию, подходит ли данное занятие для новичков. Имеет базовое значение **False**(ложь), означающее, что упражнение для новичков не подходит
- **crossfit** – логическое поле, содержащее информацию, является ли данное занятие кроссфитом(программа упражнений на силу и выносливость, состоящая в основном из анаэробных упражнений, гимнастики и тяжёлой атлетики). Имеет базовое значение **False**(ложь), означающее, что упражнение кроссфитом не является
- **general_workout** – логическое поле, содержащее информацию, является ли данное занятие общей тренировкой, рассчитанной на нагрузку всех или большинства групп мышц. Имеет базовое значение **False**(ложь), означающее, что упражнение таковым не является
- **cardio** – логическое поле, содержащее информацию, является ли данное упражнение кардио(вид физической активности, при

котором сердце работает в учащенном режиме, а источником энергии выступает подкожный жир и гликоген). Имеет базовое значение False(ложь), означающее, что упражнение кардио нагрузкой не является

- поля back, legs, chest, shoulders, biceps, triceps, abs - логические поля, содержащее информацию, является ли данное занятие упражнением на спину, ноги, грудь, плечи, бицепс, трицепс и пресс соответственно. Имеет базовое значение False(ложь), означающее, что упражнение не направленно на соответствующую группу мышц.
- is_group – логическое поле, содержащее информацию, является ли данное занятие групповой тренировкой. Имеет базовое значение False(ложь), означающее, что упражнение рассчитано на персональное исполнение
- is_competition – логическое поле, содержащее информацию, является ли данное занятие соревнованием, проводимым клубом. Имеет базовое значение False(ложь), означающее, что упражнение соревнованием не является

Clients – таблица клиентов. В ней хранится информация о клиентах фитнес – клуба, содержит в себе следующие атрибуты:

- id – уникальный числовой идентификатор клиента
- login – уникальный строковый логин клиента. Используется клиентом для входа в систему. Максимальная длина – 100 символов.
- username – строковое поле, хранящее имя клиента. Максимальная длина – 100 символов, стандартное значение – default_name

- `last_name` – строковое поле, хранящее фамилию клиента. Максимальная длина – 100 символов, стандартное значение – `default_last_name`
- `email` – поле для уникального Email – адреса пользователя. Имеет автоматическую верификацию на правильность введенного адреса от Django. Максимальная длина – 255 символов.

Таблица клиентов заменяет стандартную таблицу пользователей, представленную в Django, таким образом для объявления этой модели было необходимо создать проверки на уровень доступа (имеет ли пользователь права администратора) и создать менеджер клиентов.

Менеджер клиентов – встроенный класс Django, позволяющий создать и установить объекты любой выбранной таблицы как таблицы пользователей. Преимущества такого решения в том, что в дальнейшем, при разработке, фреймворк будет сам проводить все необходимые проверки на принадлежность пользователя, что сильно повысит безопасность сервера и уменьшит объем необходимого кода.

Trainer – таблица тренеров. Содержит в себе информацию о тренерах – работниках фитнес клуба, проводящих групповые и персональные тренировки. Содержит в себе следующие атрибуты:

- `id` – уникальный числовой идентификатор тренера.
- `name` – строковое поле, хранящее имя тренера. Максимальная длина – 100 символов. Данное поле может быть пустым.

Schedule – таблица расписания. Содержит в себе информацию о проведении групповых тренировок и соревнования. Содержит в себе следующие атрибуты:

- `date` – поле даты, хранящее информацию о дате проведения события. Может быть пустым.
- `startTime` – поле времени, хранящее информацию о времени начала события. Может быть пустым.

- endTime – поле времени, хранящее информацию о времени конца события. Может быть пустым.
- people_limit – числовое поле, хранящее информацию о количестве людей, которые могут принять участие в событии за один раз. Имеет стандартное значение - 20.
- people_enlisted – числовое поле, хранящее информацию о количестве людей, записавшихся на событие. Стандартное значение – 0.
- place – строковое поле, хранящее информацию о месте проведения данного мероприятия (как правило зал). Максимальная длина – 100 символов, может быть пустым.
- price – числовое поле, хранящее информацию о цене за вход на данное мероприятие. Стандартное значение – 0.
- leader – поле, хранящее информацию о тренере, проводящем занятие. Является внешним ключом на таблицу Trainer. Если объект тренера удаляется, внешнему ключу присваивается значение null.
- description – строковое поле, хранящее информацию о данном мероприятии. Максимальная длина – 1000 символов, может быть пустым.
- activity – поле, хранящее информацию о типе мероприятия. Является внешним ключом на таблицу Activities. При удалении объекта activity объект расписания удаляется. Имеет стандартное значение 1 (activity с индексом 1)

Appointments – таблица записавшихся. Хранит в себе информацию о людях, записавшихся на события и групповые тренировки. Содержит в себе следующие атрибуты:

- id – уникальный числовой идентификатор записи.

- **client** – поле, хранящее информацию о клиенте, записавшемся на занятие. Является внешним ключом таблицы **Clients**, при удалении объекта клиента, соответствующие записи удаляются. Имеет стандартное значение 1.
- **schedule_position** – поле, хранящее информацию о событии, на которое была сделана запись. Является внешним ключом к таблице **Schedule**, при удалении объекта расписания, соответствующие записи удаляются. Имеет стандартное значение 1.

News – таблица новостей. Хранит информацию о новостях клуба.

Содержит следующие атрибуты:

- **id** – уникальный числовой идентификатор новости.
- **title** – строковое поле, хранящее заголовок новостной записи. Максимальная длина – 100 символов. Может быть пустым.
- **sub-title** – строковое поле, хранящее подзаголовок новостной записи. Максимальная длина – 100 символов. Может быть пустым.
- **text** – текстовое поле, хранящее основной текст новостной записи. Максимальная длина – 500 символов. Может быть пустым.
- **date** – поле даты, хранящее информацию о дате выпуска новостной записи. Поле может быть пустым.

Workouts – таблица, в которой хранится история тренировок клиента.

Содержит следующие атрибуты:

- **id** – уникальный числовой идентификатор тренировки.
- **user** – поле, в котором содержится информация о клиенте, проходившем тренировку. Является внешним ключом таблицы **Clients**. При удалении объекта клиента, соответствующая запись удаляется. Имеет стандартное значение 1.

- name – строковое поле, в котором содержится название тренировки. Максимальная длина – 100 символов. Может быть пустым.
- year – числовое поле, в котором содержится год проведения тренировки. Имеет стандартное значение 2001.
- month - числовое поле, в котором содержится месяц проведения тренировки. Имеет стандартное значение 1.
- day - числовое поле, в котором содержится день проведения тренировки. Имеет стандартное значение 1.
- startTime – поле времени, в котором содержится информация о времени начала тренировки. Имеет стандартное значение 0:00:00.
- endTime - поле времени, в котором содержится информация о времени конца тренировки. Имеет стандартное значение 1:00:00.

Exercises – таблица, в которой хранится информация об упражнениях, выполненных пользователем во время тренировки. Содержит следующие атрибуты:

- id – уникальный числовой идентификатор упражнения.
- weight – числовое значение, в котором содержится вес, с которым проводилось упражнение. Может быть пустым
- reps – числовое поле, в котором содержится количество повторений упражнения. Может быть пустым
- workout_id – поле, в котором содержится информация о тренировке, в которой проводилось данное упражнение. Является внешним ключом таблицы Workouts. При удалении объекта тренировки, соответствующее упражнение удаляется.
- activity – поле, в котором содержится информация об упражнении. Является внешним ключом таблицы Activities. При удалении объекта activity, его место остается пустым.

3.2 Разработка обработчиков

Для того, чтобы Frontend-служба могла взаимодействовать с базой данных, были созданы url patterns. urlpatterns является списком объектов URLconf, которые определяют, как URL-адреса должны быть обрабатываться в Django. Каждый элемент urlpatterns определяет соответствие между URL-адресом и функцией представления, которая будет обрабатывать запрос. Для корректной работы проекта были созданы следующие паттерны: “login/”, “register/”, “add_workout/”, “remove_workout/”, “enroll/”, “check_for_appointment”, “return_client”, “return_schedule_all”, “return_schedule”, “return_appointments”, “return_news”, “return_news/<int:since>”, “return_exercise_activities”, “return_searched_exercise_activities”, “return_all_activities”, “return_workouts/”, “return_workouts/<int:year>/<int:month>”, “return_workout_years”, “return_workout_months/<int:year>”.

По вызову паттернов “return_schedule”, “return_news”, “return_all_activities”, “return_exercise_activities”, “return_workouts” вызываются простейшие обработчики, делающие запрос к базе данных на получение всех объектов и, обрабатывая их через простейший Serializer, возвращает их объектом JSON.

Django REST предоставляет сильный инструмент для проверки прав пользователя, сделавшего запрос, таким образом простейшие запросы, описанные ранее имеют разрешение IsAdminUser, что не позволит пользователю сделать вызов данной ссылки, если он не вошел как админ. Такие запросы были бы крайне неэффективны для простого пользователя, потому что ему нет необходимости получать все поля одновременно, однако для отладки и модерации такие запросы необходимы.

Запросы, которые будут выполнять обычные пользователи, должны быть эффективны и возвращать только необходимый объем информации, таким образом они требуют более сложных обработчиков. Ниже представлено краткое описание обработчиков:

- **return_schedule_week** – обработчик, вызываемый по паттерну “return_schedule”. Он возвращает расписание групповых занятий и мероприятий на 2 недели вперед. Он является POST – запросом, потому что ему необходимо получать текущую позицию пользователя в ленте расписания и возвращать следующие недели. Достигается это получением смещения от пользователя. Оно указывает сколько недель уже запрошено и локальную дату. Далее на основе этих двух параметров вычисляются начало и конец необходимой недели и делается запрос к таблице расписания, который фильтруется по дате начала и конца. В качестве результата возвращаются информация о неделях, первый и последний дни этого промежутка.
- **check_for_appointment** – обработчик, вызываемый по паттерну "check_for_appointment". Он возвращает информацию, записан ли пользователь на выбранную активность.
- **return_news_month** – обработчик, вызываемый по запросу “return_news/drawn”. Он возвращает следующие 10 новостей отсчитывая от параметра drawn. Достигается это фильтрацией по количеству от drawn.
- **return_appointments** – обработчик, вызываемый по паттерну “return_appointments”. Данный запрос возвращает список всех групповых занятий и мероприятий, на которые записался пользователь и время которых еще не прошло.
- **return_client** – обработчик, вызываемый по паттерну “return_client”. Данный запрос возвращает информацию о клиенте, сделавшем запрос. В этот список входят имя, фамилия и email.
- **add_workout** – обработчик, вызываемый по паттерну “add_workout”. Данный запрос позволяет пользователю сохранить тренировку, которую он провел. Для этого клиент в запросе передает название тренировки, дату проведения, время начала и

конца и информацию об упражнениях, в которую входят вес каждого упражнения, количество повторений и ссылку на объект таблицы Activities, которая хранит в себе информацию об упражнении.

- **remove_workout** – обработчик, вызываемый по паттерну “remove_workout”. Данный запрос позволяет пользователю удалить тренировку, записанную ранее. Он нужен на случай появления ошибочно добавленных тренировок, или если есть желание исправить что – то в информации о тренировке.
- **return_exercise_activities** – обработчик, вызываемый по паттерну “return_exercise_activities”. Данный запрос вызывается при создании тренировки и возвращает пользователю список всех упражнений в алфавитном порядке. Достигается это при помощи фильтрации объектов Activities по параметру is_exercise, описанном ранее.
- **return_selected_workouts** – обработчик, вызываемый по паттерну “return_selected_workouts”. Данный запрос принимает на вход месяц и год и возвращает список всех тренировок, которые были проведены пользователем в этот год и этот месяц. Достигается это путем фильтрации объектов Workouts по пользователю, а затем по дате. Объекты также сортируются по дате для более удобного отображения.
- **return_workout_years** – обработчик, вызываемый по паттерну “return_workout_years”. Данный запрос возвращает список всех лет, когда пользователь проводил тренировки.
- **return_workout_months** – обработчик, вызываемый по паттерну “return_workout_months”. Данный запрос возвращает список всех месяцев внутри заданного года, когда пользователь проводил тренировки.

- **register** – обработчик, вызываемый по паттерну “register”. Данный запрос позволяет пользователю пройти регистрацию в системе. Реализована регистрация при помощи встроенных инструментов фреймворка DRF: был создан сериализатор регистрации, который обрабатывал переданные параметры и сохранял их в новом объекте таблицы Clients.
- **enroll** – обработчик, вызываемый по паттерну “enroll”. Данный запрос проверяет, записан ли пользователь на групповое занятие, и, если он записан, то удаляет запись, а если нет – создает запись в таблице Appointments.

3.3 Общая структура backend-службы.

Проект сервера имеет стандартную структуру Django – приложения (рис.2). Оно содержит:

- Директорию .vscode, хранящую инструкции для локального запуска сервера
- Директорию diplom, являющуюся основной директорией приложения и хранящей в себе настройки сервера и файл со ссылками на сегменты сервера. Этот файл позволяет приложению знать о всех своих сегментах и взаимодействовать с ними.
- Директорию pract, содержащую:
 - models.py, в котором объявляются все таблицы базы данных
 - admin.py, регистрирующий таблицы в базе данных
 - serializers.py, в котором содержатся все сериализаторы моделей
 - views.py, в котором содержатся все запросы и обработчики базы данных
 - urls.py, в котором содержатся ссылки на запросы
 -

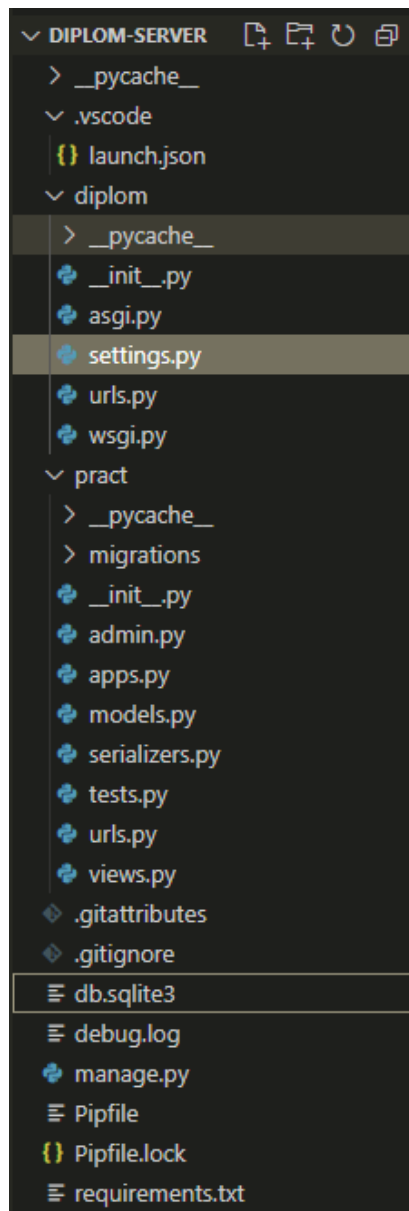


Рисунок 2.

3.4 Панель администратии

Django предоставляет разработчикам удобную панель администратии, позволяющую взаимодействовать с данными на сервере при помощи понятного

интерфейса (рис.3).

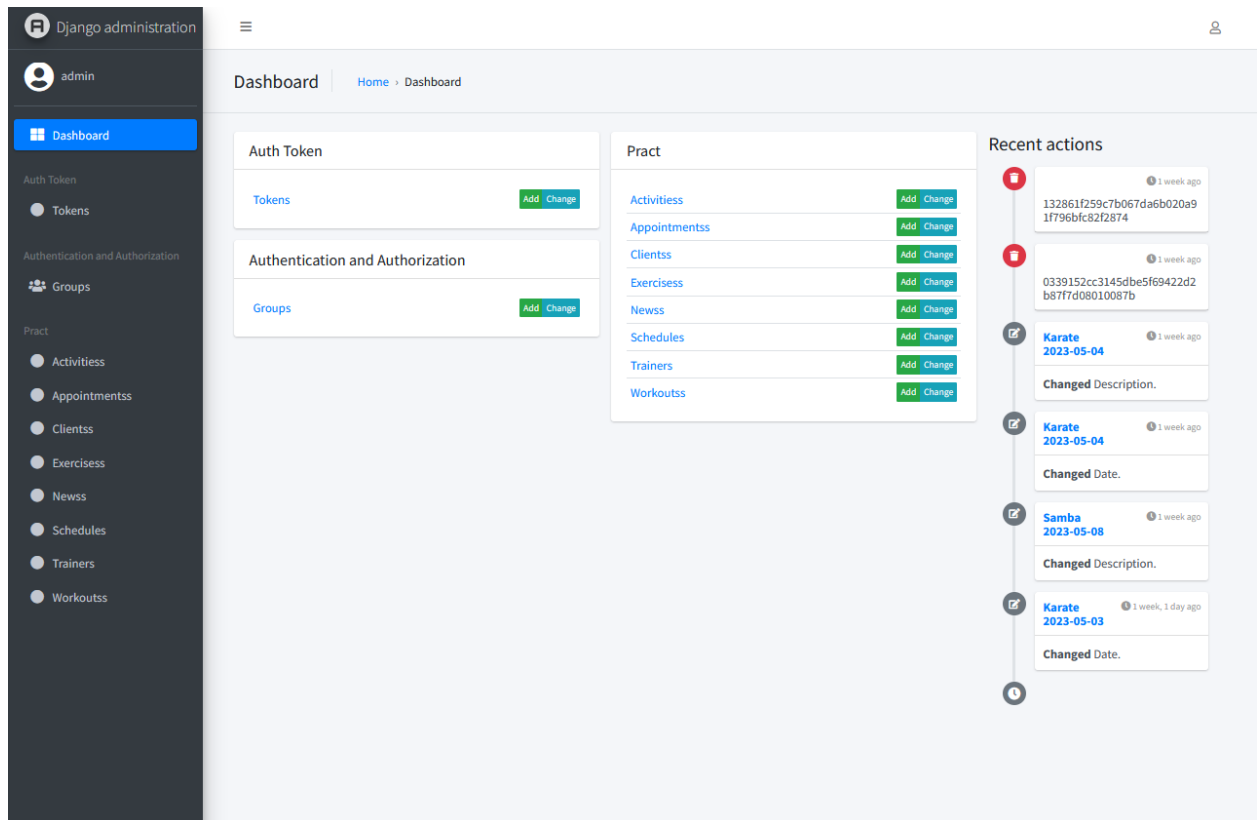


Рисунок 3.

Данная панель предоставляет функционал для взаимодействия с токенами (рис. 4), таблицами (рис. 5), а также просматривать историю изменений внутри панели администрации (рис. 6). Благодаря этому разработка утилиты для менеджмента не является первостепенной задачей.

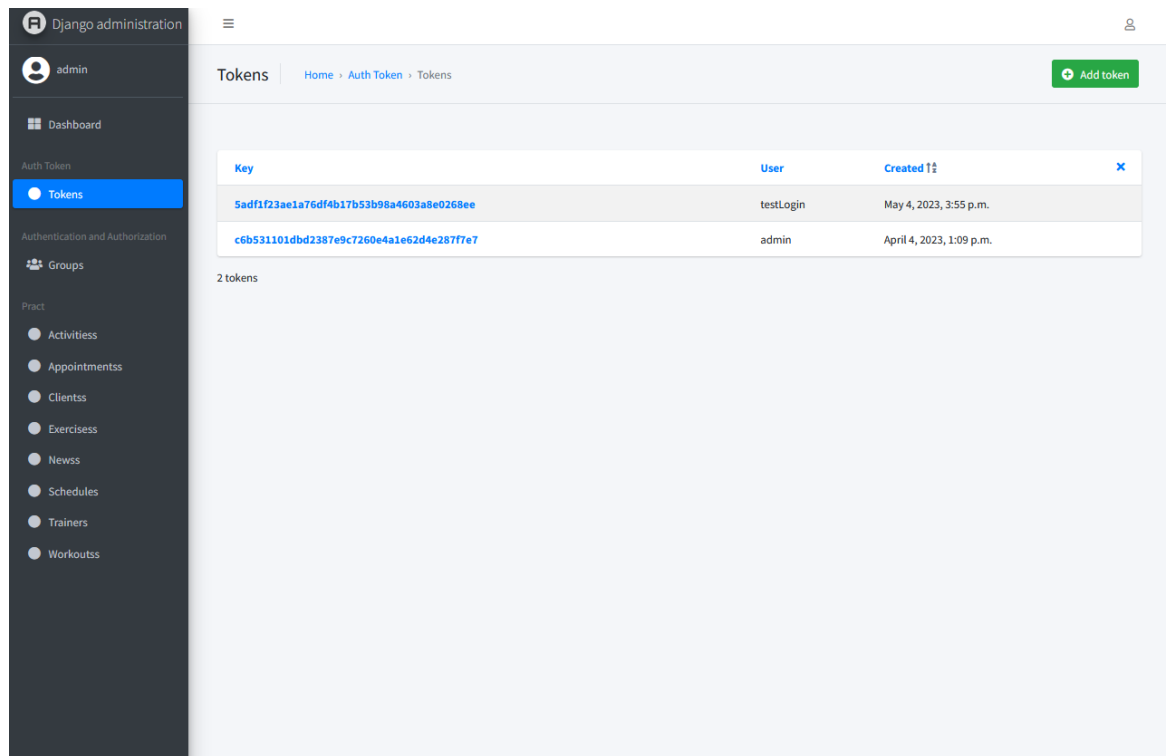


Рисунок 4.

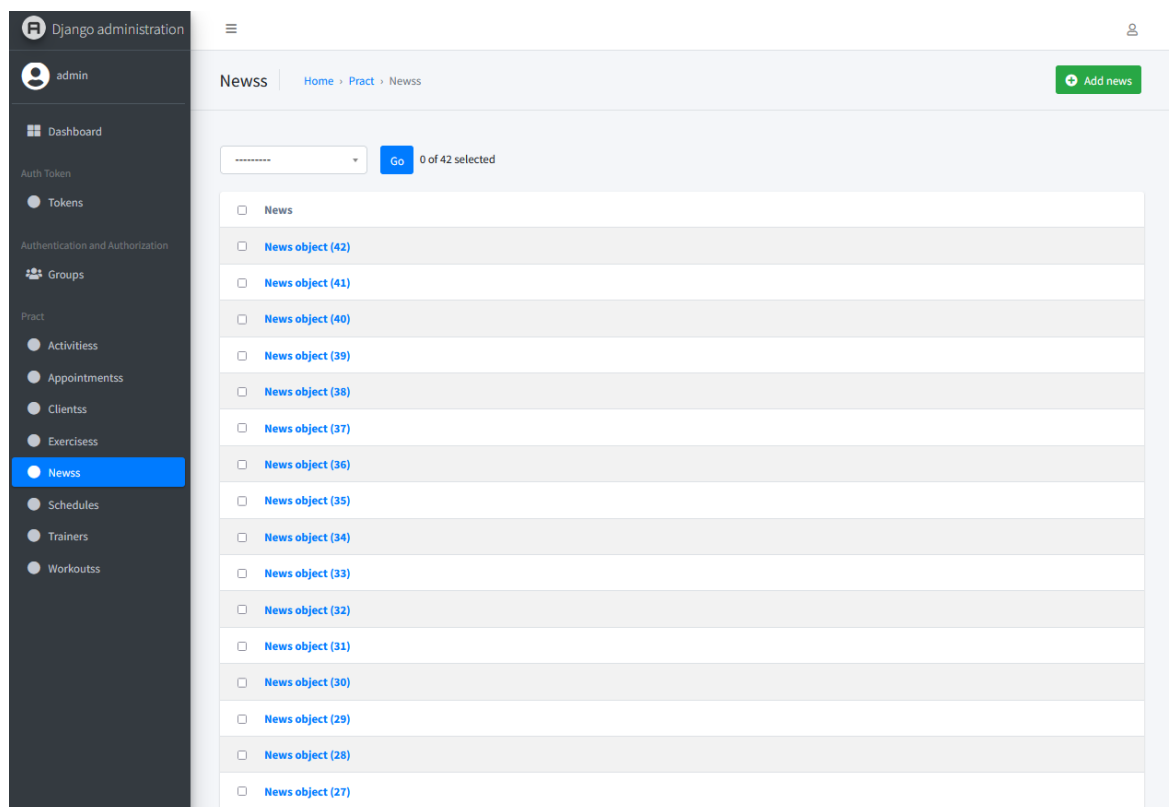


Рисунок 5.

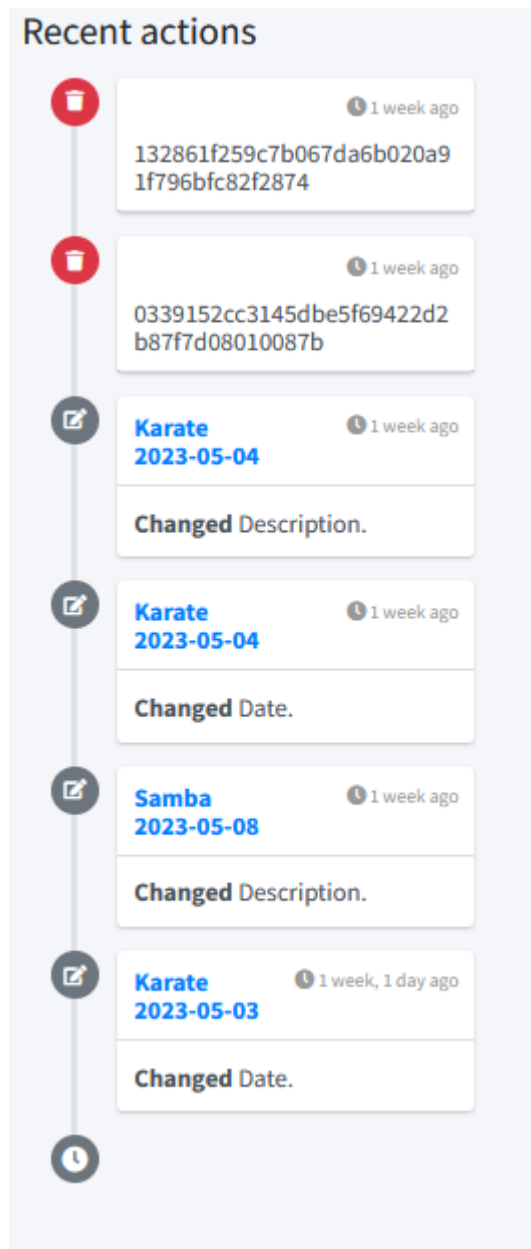


Рисунок 6.

4. РАЗРАБОТКА FRONTEND-СЛУЖБЫ

В данном разделе речь пойдет о интерфейсе программы, разработке каждого из интерфейсов и опыте взаимодействия клиента с приложением.

Всего в приложении представлено 5 главных окон:

- окно новостей
- окно записной книжки
- окно создания записей
- окно расписания
- окно профиля

Ниже представлена диаграмма взаимодействия пользователя с приложением (рис. 7)

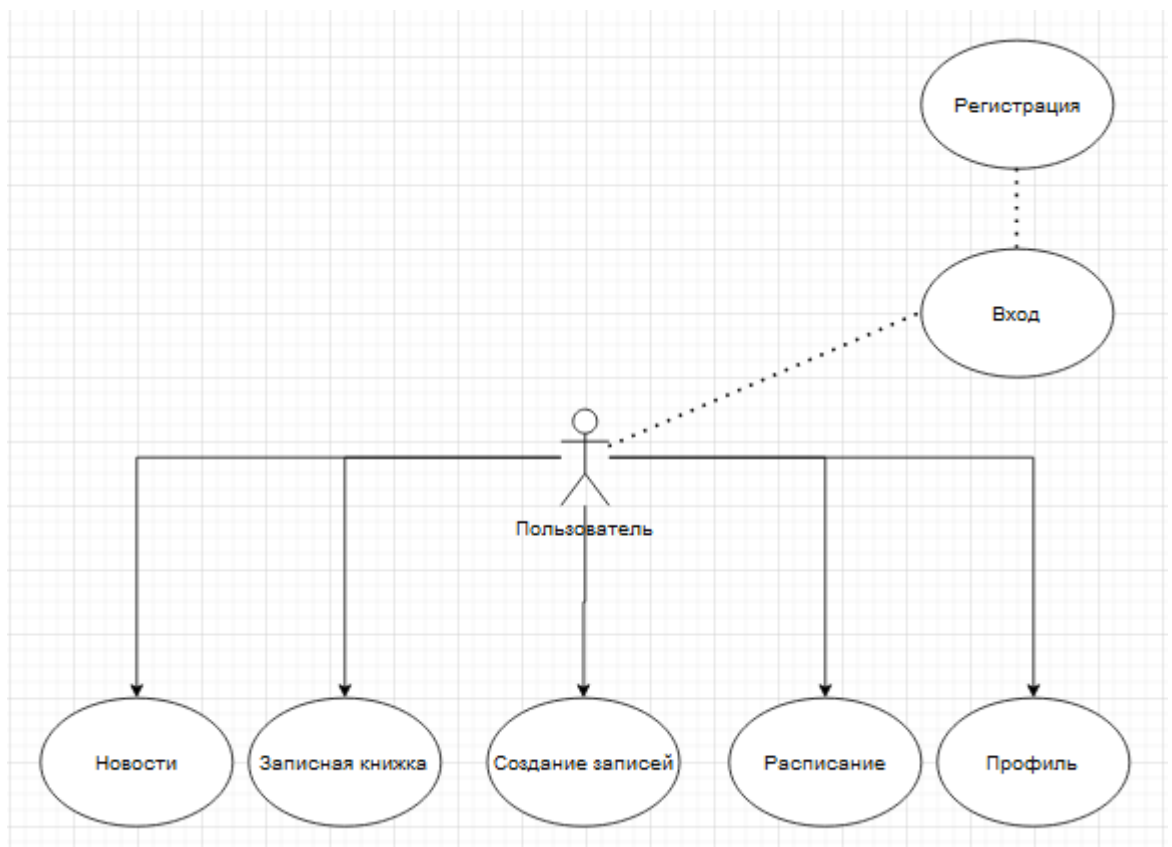


Рисунок 7.

4.1 Окно новостей

Окно новостей (рис. 8) является стандартным окном, которое видит пользователь при запуске приложения. Оно содержит список событий, происходящих в фитнес клубе. Каждая из записей состоит из даты создания новости, заголовка, подзаголовка и небольшого отрывка текста. Единоновременно на окне загружается не более 10 записей, однако если пользователь окажется в конце ленты, будут загружены следующие 10 записей. Данный функционал был сделан для оптимизации работы, потому что теперь клиенту нет необходимости загружать все и сразу, а также хранить тысячи новостных записей, которые он даже не будет читать.



Рисунок 8.

Если клиента заинтересовала одна из новостей, у него есть возможность нажать на запись. Это действие развернет ее на весь экран (рис. 9) и предоставит возможность ознакомиться со всем текстом.



Рисунок 9.

Для того, чтобы вернуться назад к ленте, достаточно нажать в любую область экрана. Это действие свернет запись и вернется к предыдущему виду (рис. 8).

Также, на протяжении всего взаимодействия с новостями, на экране остается Navigation bar.

Navigation Bar — это стандартная панель навигации, расположенная в нижней части экрана, которая позволяет быстро переходить между различными разделами приложения. Она может содержать различные элементы, такие как

кнопки, иконки и текст, в зависимости от дизайна и функциональности приложения. Этот элемент обычно используется для предоставления быстрого доступа к основным функциям приложения, таким как меню, поиск, настройки и т. д. Это может быть особенно полезно для приложений с большим количеством разделов или функций.

Таким образом, прочитав новость, пользователь мгновенно может перейти в другие разделы приложения.

4.2 Окно записной книжки

Следующее окно, в которое может перейти пользователь – окно записной книжки. В данном окне содержится история пользовательских тренировок с сортировкой по годам и месяцам (рис. 10). Единоновременно загружаются только тренировки, проведенные за последний записанный месяц последнего года. Данное решение сделано с целью оптимизации, потому что даже внутри одного месяца может содержаться несколько десятков записей, а прогрессивная загрузка позволит снизить требовательность приложения.

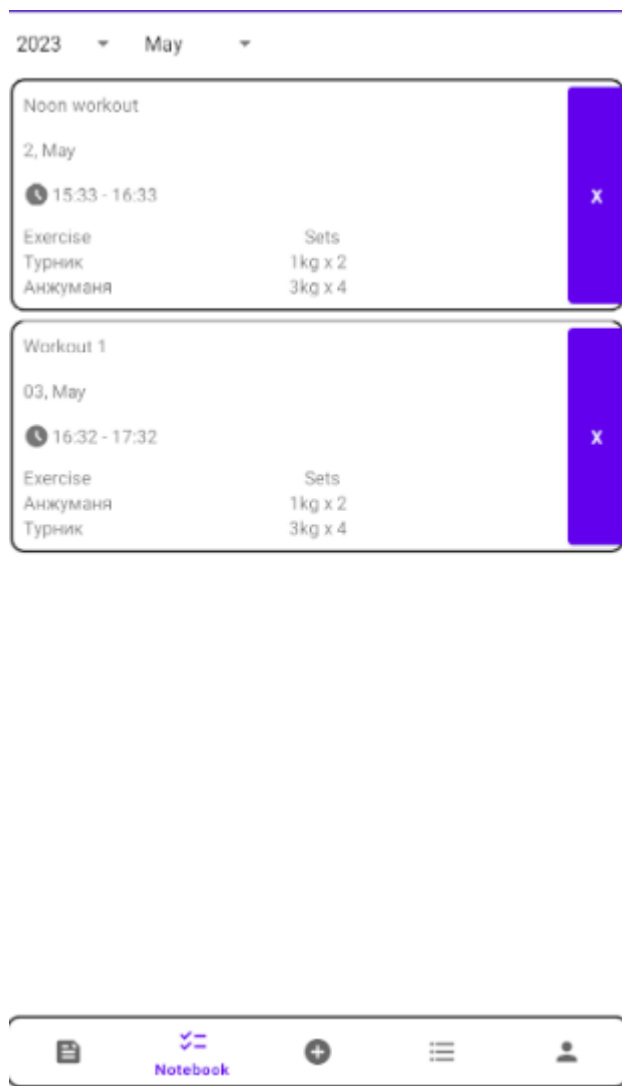


Рисунок 10.

Интерфейс состоит из кнопок для выбора месяца и года (сверху), а также списка дней, в которые были проведены тренировки, где каждый день выделен рамкой для удобства.

Каждая тренировка содержит в себе название тренировки, дату проведения, время начала и конца, а также список всех упражнений по порядку с весом и количеством повторений.

Напротив каждой записи также имеется кнопка для удаления тренировки, нажатие на которую вызовет окно подтверждения (рис. 11), которое, после нажатия на «ок», позволит приложению удалить тренировку.

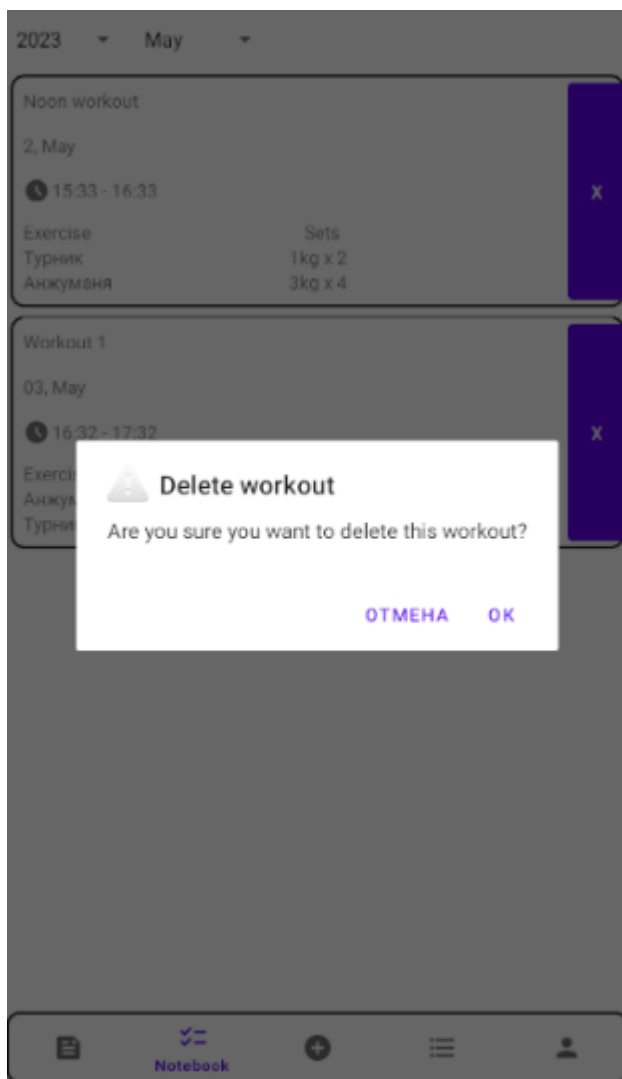


Рисунок 11.

4.3 Окно создания записей

Данное окно напрямую связано с предыдущим. В то время, как в окне записной книжки пользователь мог смотреть все свои тренировки, в окне создания записей (рис. 12) пользователь может их создавать.

Create workout

Workout name

04:16

05:16

04 May 2023



ACCEPT



Рисунок 12.

Данное окно состоит из:

- Надписи, описывающей краткую суть окна
- Текстового поля, где пользователь может ввести название своей тренировки. Если такового введено не будет, будет применено стандартное.
- Два поля времени. Слева – время начала тренировки, справа – конца. По нажатию на них будут открываться соответствующие спиннеры для удобного времени часов и минут (рис. 13). По стандарту эти поля имеют значения текущего часа и следующего часа.

- Поле даты. По нажатию на поле открывается календарь (рис. 14) для удобного выбора дня тренировки. По стандарту это поле имеет значение текущего дня.
- Кнопка для добавления упражнений. По нажатию откроется дополнительное окно (рис. 15), содержащее все тренировки и строку поиска, для удобной навигации по списку. Каждое из упражнения имеет снизу строку с кратким описанием специфики упражнения. При нажатии на любое из упражнений оно будет добавлено в список, при нажатии кнопки принять (ассерт) весь список будет добавлено на главное окно (рис. 16).
- Поля веса и повторений тоже интерактивны. При нажатии на одно из них будет открыто текстовое поле, где пользователь сможет ввести свое количество веса и повторений (рис. 17).
- Кнопка “Delete” напротив каждого из упражнений удалит его из списка.
- При нажатии кнопки “Ассерт” все изменения, сделанные пользователем, будут сохранены, а данная запись будет добавлена в записную книжку и сохранена в базе данных.

Create workout

Workout name

04:16

05:16

3

15

4

16

5

17

04 May 2023

+

ACCEPT



Add



Рисунок 13.

Create workout

Workout name

04:16

05:16

04 May 2023



Май 2023 г.



пн

вт

ср

чт

пт

сб

вс

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

+

ACCEPT



Add



Рисунок 14.

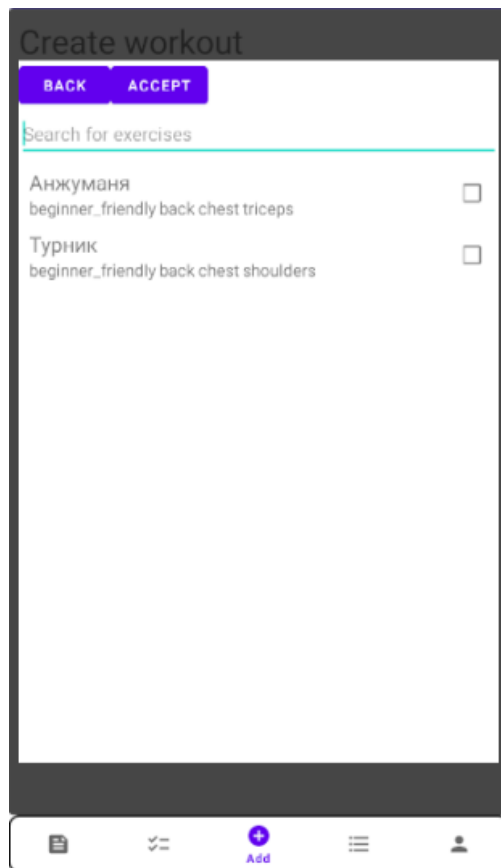


Рисунок 15.

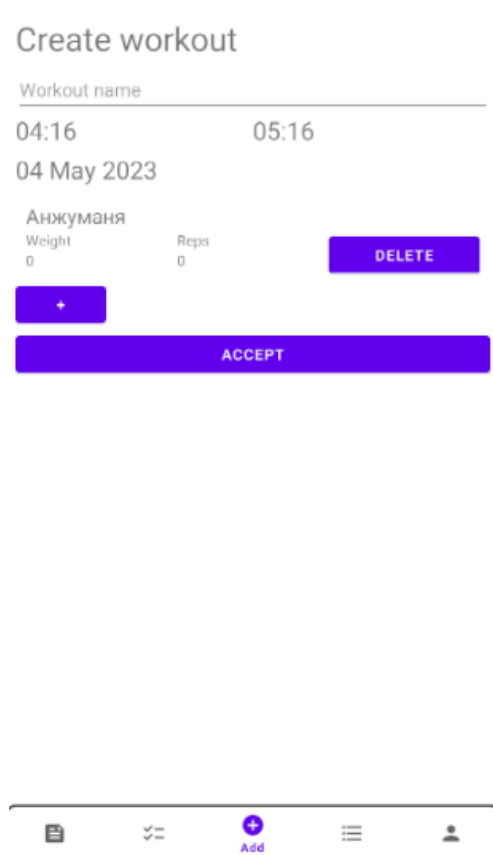


Рисунок 16.

Create workout

Workout name

04:16 05:16

04 May 2023

Анжуманя

| Weight | Reps | |
|------------------------|------|-------------------|
| <div><div></div></div> | 0 | <div>DELETE</div> |

+

ACCEPT



Рисунок 17.

4.4 Окно расписания

Данное окно содержит расписание групповых занятий клуба с сортировкой от текущего дня и далее (рис. 18.). Единоновременно загружаются 14 дней начиная от текущего с целью оптимизации.

Schedule

04 May 2023

Karate

🕒 18:44 - 19:44

Available spots: 20



SS

Trainer 1

05 May 2023

Nothing today

06 May 2023

Nothing today

07 May 2023

Nothing today

08 May 2023

Samba

🕒 16:06 - 17:06

Available spots: 20



Hall 1

Trainer 1

09 May 2023

Samba

🕒 10:34 - 11:34

Available spots: 20



Hall 1

Trainer 1



Рисунок 18.

Каждое событие сортируется по времени внутри одного дня. Если в определенный день нет никаких занятий, там будет соответствующая надпись.

Каждая из позиций в расписании содержит:

- Название занятия
- Время начала и конца
- Количество доступных мест
- Информацию является ли занятие платным (перечеркнутый значок обозначает бесплатное занятие).
- Место проведения занятия
- Тренера, проводящего занятие

Если пользователь захочет записаться на мероприятие, он может нажать на соответствующую позицию в расписании. Данное действие откроет окно выбранного занятия (рис. 19), где будет представлена подробная информация о занятии, а также будет доступна кнопка записаться. При нажатии на кнопку пользователь будет записан на занятие, а кнопка изменится на отмену записи. При нажатии на нее запись пользователя будет удалена.

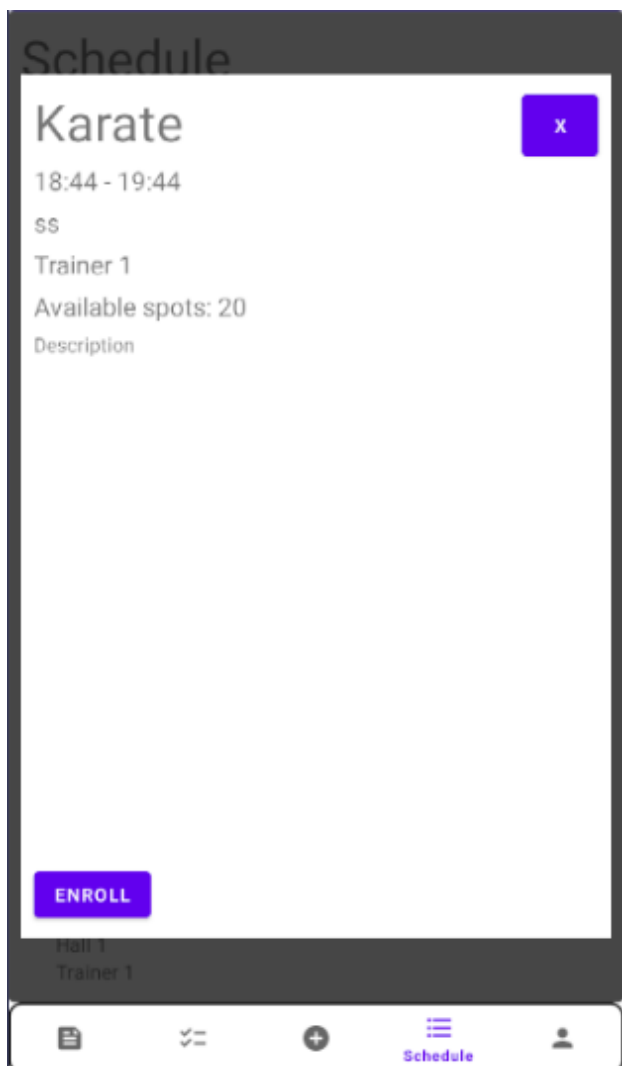


Рисунок 19.

4.5 Окно профиля

Данное окно содержит базовую информацию о пользователе, а также список всех занятий, на которые он записан (рис. 20).

Содержит следующие поля:

- Имя
- Фамилия
- Электронная почта
- Список актуальных записей с названием, датой и временем начала

Все записи для удобства отсортированы по дате и времени.

testName

testLastName

test@test.com

List of appointments

| | | |
|--------|------------|----------|
| Samba | 2023-05-08 | 16:06:47 |
| Samba | 2023-05-09 | 10:34:54 |
| Karate | 2023-06-03 | 12:50:49 |

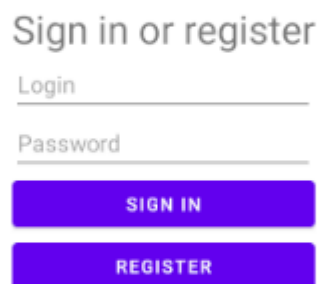


Рисунок 20.

4.6 Регистрация и вход в приложение

Все вышеперечисленные окна очень плотно взаимодействуют с клиентом и информацией о нем, хранящейся в базе данных. Однако для того, чтобы данное взаимодействие было возможным, необходимо иметь информацию о

том, кто взаимодействует с приложением. Поэтому, при первом включении на экране включается окно входа (рис. 21).



Sign in or register

Login

Password

SIGN IN

REGISTER

Рисунок 21.

В этом окне присутствуют следующие элементы:

- Текст заголовка, объясняющий назначение окна
- Поле логина
- Поле пароля
- Кнопка входа, при нажатии на которую будет произведена верификация введенной информации с последующим входом, если все верно
- Кнопка регистрации, при нажатии на которую будут созданы дополнительные поля (рис. 22):
 - Поле имени

- Поле фамилии
- Поле электронной почты
- Поле повтор пароля

После заполнения всех вышеуказанных полей пользователь будет зарегистрирован и сможет начать пользоваться приложением.

Sign in or register

Form fields and buttons:

- Login
- Password
- Name
- Last name
- E-mail
- Repeat password
- SIGN IN**
- REGISTER**

Рисунок 22.

ЗАКЛЮЧЕНИЕ

Таким образом, было разработано мобильное приложение, способное удовлетворять потребности как обычных посетителей спортивных залов, так более продвинутых спортсменов. Оно сочетает в себе качества информационного и интерактивного инструмента, позволяющего оставаться в курсе всего происходящего в спортзале, а также наблюдать за собственным прогрессом.

На текущий момент, данное приложение является уникальным в своем роде, потому что ни один спортивный зал не предоставляет такого функционала, полностью убирая необходимость в сторонних инструментах для комфортного спортивного времяпровождения.

Данный проект сочетает в себе современные средства разработки для обеспечения максимального комфорта использования приложения и наилучшего пользовательского опыта.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ

1. ГОСТ 2.316-2008. Правила нанесения надписей, технических требований и таблиц на графических документах.
2. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
3. ГОСТ 7.32-2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
4. Лутц. М. Программирование на Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
5. Гришаева. О., Ширшова. Е. Исследование рынка мобильных приложений для оценки двигательной активности человека
6. Денисов А. А. Современные средства разработки мобильных приложений //Вестник Воронежского института высоких технологий. – 2019. – №. 2. – С. 64–67.
7. Ким В. Ю. Особенности разработки дизайна пользовательского интерфейса для мобильного приложения //Новые информационные технологии в автоматизированных системах. – 2015. – №. 18. – С. 479-481.
8. Python. К вершинам мастерства / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2016. – 768 с.: ил.
9. Документация Django – [Электронный ресурс]. URL:
<https://docs.djangoproject.com>
10. Документация Python – [Электронный ресурс]. URL:
<https://docs.python.org/3.10/>
11. Документация Django REST – [Электронный ресурс]. URL:
<https://www.django-rest-framework.org/>
12. Документация SQLite – [Электронный ресурс]. URL:
<https://www.sqlite.org/docs.html>

13. Документация Java – [Электронный ресурс]. URL:
<https://docs.oracle.com/javase/8/docs/>