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. звание)

Калажоков З.Х. \_\_\_\_\_\_\_\_

(фамилия, имя, отчество полностью) (подпись)

**ВКР соответствует предъявляемым**

**требованиям**

Руководитель Департамента:

к. т. н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Петросов Д. А.

(подпись)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва – 2023 г.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc133590842)

[1. АНАЛИЗ ВАЖНОСТИ ПОСТАВЛЕННОЙ ЗАДАЧИ 5](#_Toc133590843)

[1.1 Почему важны приложения 5](#_Toc133590844)

[1.2 Обзор ситуации на рынке 6](#_Toc133590845)

[1.2 Преимущества созданного решения 7](#_Toc133590846)

[2. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ 10](#_Toc133590847)

[2.1 Разбор поставленной цели 10](#_Toc133590848)

[2.2 Python 10](#_Toc133590849)

[2.3 Django 11](#_Toc133590850)

[2.3 Django REST 13](#_Toc133590851)

[2.4 SQLite 13](#_Toc133590852)

[2.5 Android studio 14](#_Toc133590853)

[2.6 Java 15](#_Toc133590854)

[2.7 Volley 16](#_Toc133590855)

[2.8 Bluestacks 16](#_Toc133590856)

[3. РАЗРАБОТКА ПРИЛОЖЕНИЯ 19](#_Toc133590857)

[3.1 РАЗРАБОТКА BACKEND-СЛУЖБЫ 19](#_Toc133590858)

[3.2 РАЗРАБОТКА FRONTEND-СЛУЖБЫ 28](#_Toc133590859)

[ЗАКЛЮЧЕНИЕ 29](#_Toc133590860)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ 30](#_Toc133590861)

[ПРИЛОЖЕНИЯ 31](#_Toc133590862)

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 ограниченно 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 фитнесс 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. 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. 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 пользователя 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.

1.2 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 неменьшей 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.

1. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ
   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-серверное приложение – это тип архитектуры программного обеспечения, который предполагает разделение функционала приложения на две части: клиентскую и серверную.

Клиентская часть – это часть приложения, которая выполняется на устройстве пользователя (например, на компьютере, смартфоне или планшете). Она обеспечивает пользовательский интерфейс и интерактивность, а также обрабатывает локальные запросы пользователя.

Серверная часть – это часть приложения, которая запущена на сервере. Она обрабатывает запросы от клиентов, осуществляет доступ к базам данных и другим ресурсам сервера, а также обеспечивает обработку данных и обратную связь с клиентской частью приложения.

Клиент-серверная архитектура позволяет создавать более сложные приложения, которые могут быть масштабированы для обработки большого количества запросов от клиентов. Также это позволяет обеспечить более безопасный обмен данными между клиентами и серверами, так как сервер может контролировать доступ к ресурсам и данные, а клиенты могут работать только с теми ресурсами, к которым у них есть права доступа.

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

* 1. Python

Для разработки backend-службы был выбран python.

Python - это интерпретируемый, высокоуровневый, объектно-ориентированный язык программирования с динамической типизацией. Для разработки сервера он был выбран, потому что:

* Python имеет простой и понятный синтаксис, который делает его легко читаемым и понятным. Это упрощает разработку и поддержку кода сервера.
* Большое количество библиотек и фреймворков: Python имеет огромную библиотеку сторонних модулей и фреймворков, которые могут ускорить разработку сервера.
* Высокая производительность: Python имеет множество инструментов для оптимизации производительности, позволяющих ускорить выполнение кода и сделать его более эффективным.
* Асинхронная обработка запросов позволяет серверу обрабатывать множество запросов одновременно и увеличивает производительность.
* Масштабируемость: Python серверы могут быть легко масштабированы на большие нагрузки благодаря использованию распределенных систем и облачных сервисов.

2.3 Django

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

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.
  1. 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 мощный и гибкий фреймворк на базе Django для создания веб-сервисов RESTful API (Application Programming Interface). Он предоставляет разработчикам инструменты для создания REST API с минимальным количеством кода, что упрощает и ускоряет процесс разработки.

DRF предоставляет множество встроенных функций для создания REST API, таких как аутентификация, авторизация, сериализация(процесс перевода структуры данных в битовую последовательность) и десериализация объектов, валидация данных, маршрутизация запросов и многое другое.

Одной из ключевых особенностей DRF является его возможность быстрого создания API для моделей Django. Он автоматически создает сериализаторы и представления для моделей Django, что упрощает процесс создания API и уменьшает количество кода, необходимого для реализации.

2.4 SQLite

Архитектурой базы данных был выбран SQLite.

SQLite — это легковесная встраиваемая реляционная база данных, которая позволяет сохранять и управлять данными. Она используется во многих приложениях для хранения и обработки структурированных данных, таких как контакты, изображения, тексты, настройки и многое другое.

Одно из главных преимуществ SQLite заключается в том, что она не требует отдельного сервера для управления базой данных. Вместо этого, все данные хранятся в одном файле, что делает ее легко интегрируемой в различные приложения. Кроме того, SQLite поддерживает стандартный язык SQL (Structured Query Language), что делает ее удобной для использования и для профессионалов, и для новичков.

Помимо того, SQLite – одна из самых быстрых систем. Она обеспечивает быстрый доступ к данным, что делает ее идеальным выбором для многих мобильных приложений и веб-приложений. База данных также обладает хорошей масштабируемостью, что позволяет использовать ее как для небольших проектов, так и для больших приложений с множеством пользователей.

2.5 Android studio

Android Studio — это интегрированная среда разработки (IDE) для создания мобильных приложений под операционную систему Android. Она была разработана компанией Google и основана на среде разработки IntelliJ IDEA.

Одним из главных преимуществ данной среды разработки 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.

Помимо того, она предоставляет удобный и интуитивно понятный пользовательский интерфейс для разработки приложений. Это включает в себя редактор кода с подсветкой синтаксиса, автодополнение и быстрым доступом к документации и справке. Android Studio также имеет множество готовых шаблонов и компонентов, которые могут быть использованы для ускорения процесса разработки.

Android Studio также обеспечивает широкие возможности для отладки приложений. Встроенный отладчик позволяет разработчикам легко находить и исправлять ошибки в коде. IDE также позволяет создавать виртуальные устройства для тестирования приложений на различных устройствах и версиях Android.

2.6 Java

В качестве языка разработки frontend-службы был выбран язык Java, потому что:

* Java также обладает широкой функциональностью и мощными возможностями, что делает её привлекательным языком программирования для разработки многих типов приложений. Java поддерживает многопоточность, что позволяет создавать приложения, которые могут одновременно выполнять несколько задач. Кроме того, Java предоставляет различные структуры данных и алгоритмы для решения различных задач, а также богатую стандартную библиотеку, которая содержит множество полезных классов и методов.
* Java является одним из самых безопасных языков программирования благодаря своей встроенной системе безопасности. Он предоставляет механизмы проверки типов, управление памятью и проверку безопасности для защиты приложения от вредоносных программ.
* Java является одним из самых производительных языков программирования, благодаря чему даже самые устаревшие и слабые устройства могут стабильно запускать приложения, написанные на этом языке.
* Java обладает сильными инструментами интеграции, позволяющими использовать множество сторонних библиотек и API без каких либо проблем.

2.7 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.

Основные преимущества использования Volley для Java включают:

* Простота использования: Volley обладает простым и понятным API, что облегчает работу с сетевыми запросами. Он позволяет отправлять запросы в несколько строк кода и автоматически обрабатывает ответы.
* Асинхронный режим работы: Volley позволяет выполнять запросы в фоновом режиме, что делает приложение более отзывчивым и предотвращает блокировку пользовательского интерфейса.
* Ускоренный запрос: Volley автоматически выполняет оптимизации, такие как использование многопоточности и кэширование, что позволяет ускорить процесс запроса и ответа.
* Поддержка очередей запросов: Volley позволяет создавать очереди запросов, что облегчает управление процессом отправки запросов.

2.8 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.

1. РАЗРАБОТКА ПРИЛОЖЕНИЯ
   1. РАЗРАБОТКА BACKEND-СЛУЖБЫ

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 фитнесс – 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 фитнесс 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
* wokout\_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.

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/”, "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.

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

Основная цель сериализатора - преобразование объектов модели в формат, который может быть передан по сети. Сериализаторы могут использоваться для сериализации данных как в формат JSON, так и в формат XML.

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.

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 неделю вперед. Он является POST – запросом, потому что ему необходимо получать текущую позицию пользователя в ленте расписания и возвращать следующую неделю. 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 неделе, первый и последний дни этой недели.
* **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.

3.2 РАЗРАБОТКА FRONTEND-СЛУЖБЫ

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ

1. Garrigos F., Gil I., Narangajavana Y. The impact of social networks in the competitiveness of the firms //Competitiveness: Psychology, Production, Impact and Global Trends. Nova Science Publishers, Inc., Hauppauge. – 2011.
2. Garrigos‐Simon F. J., Alcamí R. L., Ribera T. B. Social networks and Web 3.0: their impact on the management and marketing of organizations //Management Decision. – 2012.

ПРИЛОЖЕНИЯ