**МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**РОСТОВСКОЙ ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РОСТОВСКОЙ ОБЛАСТИ**

**РОСТОВСКИЙ-НА-ДОНУ КОЛЛЕДЖ СВЯЗИ И ИНФОРМАТИКИ**

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

**УП.01.01 «Разработка модулей программного обеспечения для компьютерных систем»**

Специальность **09.02.07 «Информационные системы и программирование»**

Студента 3-го курса, группы ИС-34

форма обучения: очная

Артур Фурт Иванович

ФИО

Место практики **Ростовский-на-Дону колледж связи и информатики**

Срок практики с «25» марта 2025 г. по «04» апреля 2025 г.

Руководитель практики Кошкина А. А.

подпись ФИО

Итоговая оценка по практике\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ростов-на-Дону

2025 г.

**МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**РОСТОВСКОЙ ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕРОСТОВСКОЙ ОБЛАСТИ**

**РОСТОВСКИЙ-НА-ДОНУ КОЛЛЕДЖ СВЯЗИ И ИНФОРМАТИКИ**

**Аттестационный лист**

**по учебной практике**

Студент Артур Фурт Иванович

обучающийся на **3** курсе в группе **ИС-34**

форма обучения: **очная**

по специальности: **09.02.07 «Информационные системы и программирование»**

прошел(ла) учебную практику

**УП.01.01 Учебная практика «Разработка модулей программного обеспечения для компьютерных систем»**

в объеме **78** часов с «**25**» марта **2025** г. по «0**4**» апреля **2025**г.

в ГБПОУ РО «РКСИ»

Виды и объем работ, выполненные студентом в период практики, формирующие у обучающегося умения и приобретение первоначального практического опыта (таблица 1).

Таблица 1 – Виды и объем работ

|  |  |  |  |
| --- | --- | --- | --- |
| Вид работ | Объём работ, час | Отметка о выполнении | |
| оценка | подпись руководителя |
| Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием | **8** |  |  |
| Разрабатывать программные модули в соответствии с техническим заданием | **32** |  |  |
| Выполнять отладку программных модулей с использованием специализированных программных средств | **16** |  |  |
| Выполнять тестирование программных модулей | **16** |  |  |
| Осуществлять рефакторинг и оптимизацию программного кода | **6** |  |  |

Качество выполнения работ в соответствии с технологией и предъявляемыми требованиями.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_ /Кошкина А.А. /«\_\_\_» \_\_\_\_\_\_\_\_\_\_2025г.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Содержание  Введение 3  1.Проектирование программного продукта 4  1.1 Диаграмма связей 4  1.2 Диаграмма сценариев использования 5  1.3 Прототипирование и дизайн 6  1.4 Проектирование БД 9  2 Разработка программного продукта 10  2.1 Описание средств разработки 10  2.2 Разработка пользовательского интерфейса 11  2.3 Создание базы данных 16  2.4 Взаимодействие с базой данных 20  3 Тестирование программного продукта 22  3.1 Описание используемых видов тестирования 22  3.2 Тест-план 22  3.3 Тест-кейсы 24  Заключение 28  Приложение А 29  Приложение Б 30  Приложение В 33 | | | | | | | | | | |
|  |  |  |  |  | РКСИ.УП25.09.02.07.УП.04.01.4422.00ПЗ | | | | | |
|  |  |  |  |  |
| *изм* | *лист* | *№ докум.* | *Подпись* | *Дата* |
| *Разраб.* | | *Фурт А. И.* |  |  | УП 04 | *Лит* | | | *Лист* | *Листов* |
| *Провер.* | | *Кошкина А. А.* |  |  |  |  |  | *2* | *39* |
| *Реценз.* | |  |  |  | *ИС-34* | | | | |
| *Н. контр* | |  |  |  |
| *Утверд.* | |  |  |  |

Введение

В рамках данной практической работы разработан Telegram-бот под названием «GymBroBot», который позволяет пользователям просматривать программу питания и тренировок. В условиях современного общества, где здоровый образ жизни становится все более актуальным, создание такого приложения представляет собой важный шаг к упрощению доступа к персонализированным рекомендациям в области питания и физической активности.

Актуальность проекта обусловлена растущим интересом к здоровью и фитнесу среди различных возрастных категорий. Многие люди стремятся улучшить свое физическое состояние, но часто сталкиваются с трудностями в выборе подходящей программы питания и тренировок, что может привести к неэффективным результатам или даже вреду для здоровья. «GymBroBot» предоставляет пользователям возможность легко и удобно получать рекомендации, основанные на их индивидуальных потребностях и целях.

Целью разработки приложения является создание эффективного инструмента, который поможет пользователям планировать свое питание и тренировки, учитывая их физические данные и предпочтения. Это позволит не только улучшить физическую форму, но и повысить уровень осведомленности о здоровом образе жизни.

Целевая аудитория приложения включает людей, стремящихся к улучшению своего здоровья и физической формы, а также тех, кто хочет получить профессиональные рекомендации, но не имеет возможности обратиться к специалистам. Это могут быть как новички в фитнесе, так и опытные спортсмены, желающие оптимизировать свои тренировки и питание.

Основные функции «GymBroBot» включают регистрацию и авторизацию пользователей, ввод персональных данных (таких как возраст, вес, пол), рекомендации индивидуальных программ тренировок.

Этап Тестирование программного продукта включает в себя всестороннюю проверку функциональности, производительности и пользовательского интерфейса. Различные виды тестирования помогают выявить и устранить ошибки, а также гарантируют, что приложение соответствует требованиям пользователей и работает без сбоев.1 Проектирование программного продукта

Проектирование программного обеспечения – это процесс создания программных систем, который включает анализ требований, разработку архитектуры, детальное проектирование, программирование, тестирование и документацию. Он направлен на создание эффективных и надежных решений, соответствующих потребностям пользователей. Процесс требует знаний в области программирования, системного анализа и управления проектами.

* 1. Диаграмма связей

Диаграмма связей – это графическое представление взаимосвязей между различными элементами системы, такими как объекты, классы или компоненты. Она используется для визуализации и анализа отношений, взаимодействий и зависимостей между этими элементами. Диаграммы связей помогают понять структуру системы, выявить ключевые связи и упростить процесс проектирования и разработки.

На рисунке 1 можно увидеть диаграмму связей для Телеграмм-бота. Процесс взаимодействия с системой начинается со стартовой точки – первоначальный этап. После этого осуществляется авторизация пользователя, что позволяет безопасно входить в систему. Далее пользователю открывается доступ к различным услугам, таким как «Настройка Тренировок», «Настройка питания» и «Связаться с админом». Данная навигационная диаграмма показывает структуру Телеграмм-бота.

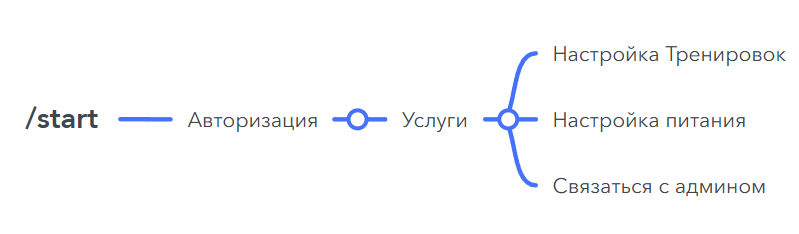


Рисунок 1 – Диаграмма связей для Телеграмм-бота

1.2 Диаграмма сценариев использования

Диаграмма сценариев – это визуальное представление последовательности действий и взаимодействий пользователей с системой для достижения определенной цели. Она иллюстрирует, как пользователи (или другие системы) взаимодействуют с функциональностью программного обеспечения в различных сценариях использования.

На рисунке 2 можно увидеть диаграмму сценария использования Телеграмм-бота. Диаграмма показывает варианты взаимодействия пользователей с системой. «Гость» может начать с «Начать регистрацию» и пройти процесс «Авторизации» и получив доступ к «Услугам, может использовать функции «Настройка Тренировок», «Настройка питания», «Связаться с админом».

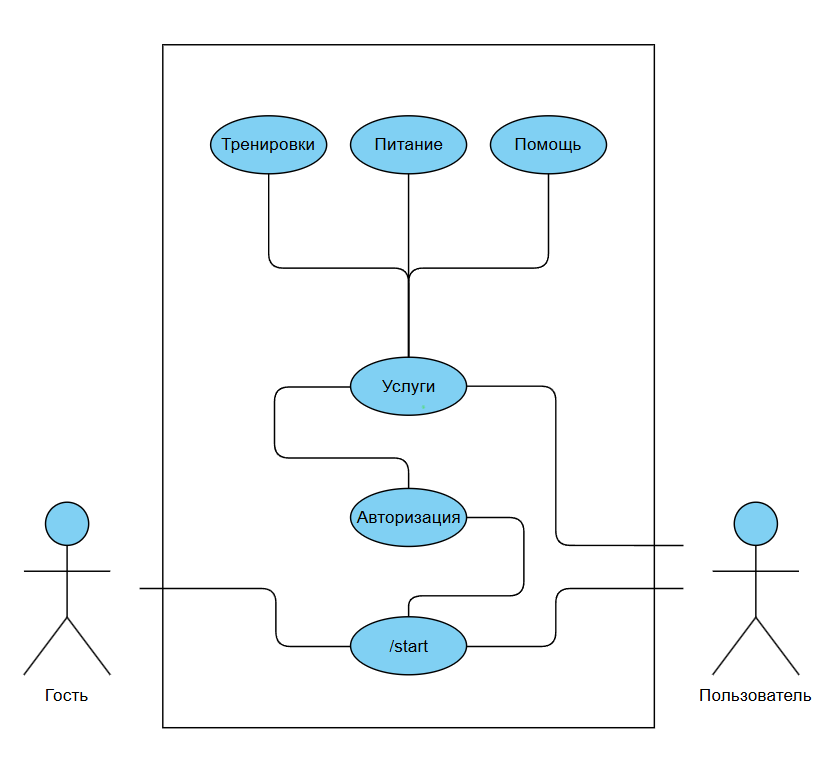


Рисунок 2 – Диаграмма сценариев использования

1.3 Прототипирование и дизайн

Дизайн экранов приложения представлен на рисунке 3.

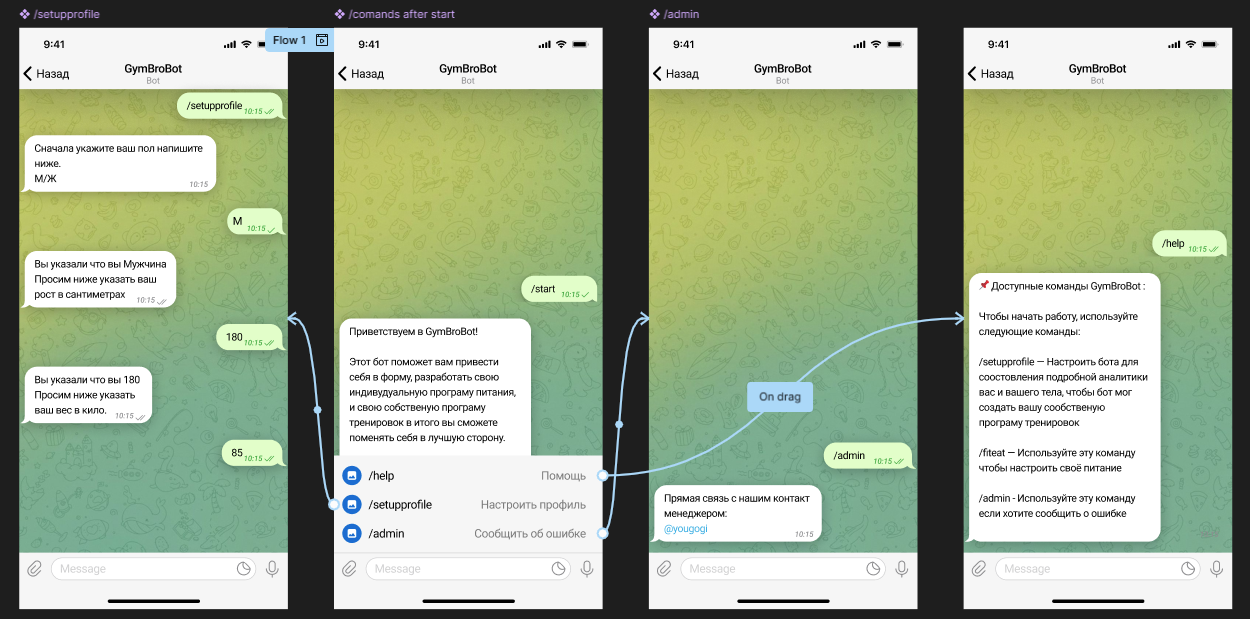


Рисунок 3 - Дизайн главных экранов

Рисунки 4 и 5 показывают, как выглядят экраны помощи и настройки профиля.

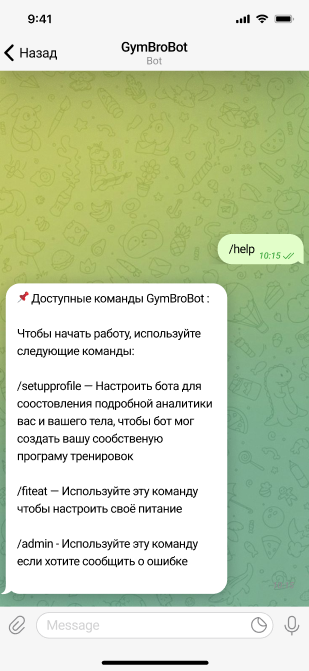


Рисунок 4 - Макет экрана помощи

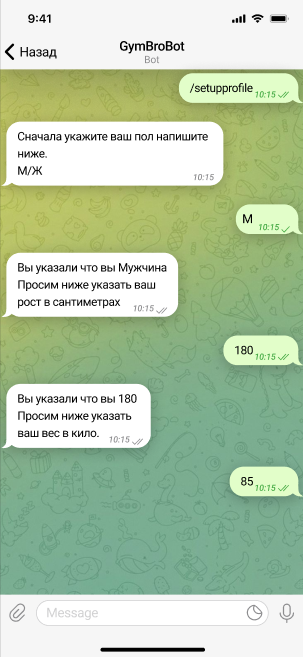


Рисунок 5 - Макет экрана настройки профиля

На рисунке 6 представлен QR-код, который можно отсканировать, чтобы получить доступ к полному дизайну.



Рисунок 6 - QR-код, для доступа к дизайну

1.4 Проектирование БД

Для реализации функционала Telegram-бота была спроектирована база данных, состоящая из трех основных таблиц: Пользователи (users), упражнения (exercies), категории тренировок (categories).

На рисунке 7 изображена таблица users, которая хранит информацию о пользователях с полями: id tg\_id, gender, height, weight, и is\_registered, где tg\_id – это идентификатор пользователя в Telegram gender, пол – height, рост – weight, вес и is\_registered – статус регистрации.

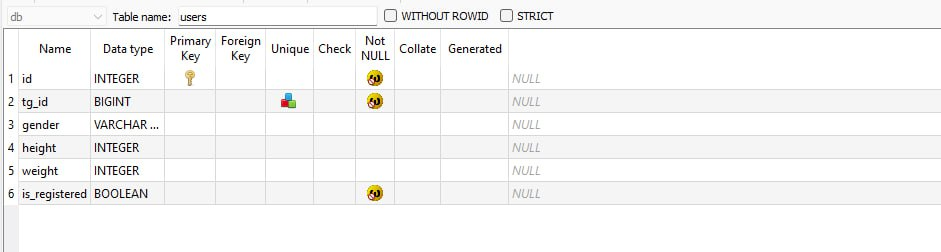


Рисунок 7 – Таблица о пользователях

На рисунке 8 изображена таблица, которая содержит полную информацию о всех доступных упражнениях: id name description, videotraining и category, где id это первичный ключ, name – название упражнения до 30 символов, description – описание до 100 символов, videotraining – ссылка на видео, а category – внешний ключ связывающий с таблицей Categories.

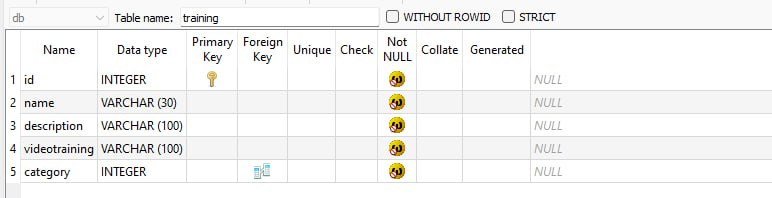


Рисунок 8 – Таблица об упражнениях

На рисунке 9 показана таблица Categories включает id и name, где id – первичный ключ, а name – название категории тренировки до 30 символов, например, Тренировка кора или Тренировка ног.

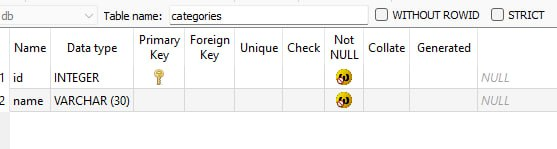


Рисунок 9 – Таблица о категориях тренировок

2 Разработка программного продукта

В данном разделе представлены инструменты и технологии, использованные в процессе разработки проекта.

2.1 Описание средств разработки

Проект был разработан на Python версии 3.9+, благодаря его лаконичности и богатой библиотечной экосистеме. Для Telegram API применялся асинхронный фреймворк Aiogram 3.x, предоставляющий полный инструментарий для разработки интерактивных ботов, включая FSM-реализацию сложных диалогов.

Разработка велась в профессиональной среде Visual Studio Code, что значительно повысило эффективность написания кода. Visual Studio Code предоставил возможности интеллектуального автодополнения, встроенного отладчика и удобной работы с виртуальными окружениями, обеспечивающими изоляцию зависимостей проекта.

Взаимодействие с платформой Telegram обеспечивалось через официальный Bot API, позволяющее не только обмениваться текстовыми сообщениями, но и предоставлять структурированные данные, интерактивные меню и кнопки для удобной навигации по функциям бота. На этапе проектирования применялись инструменты визуализации: MindMeister для создания карты функциональности бота и Figma для прототипирования пользовательского интерфейса, включая элементы управления командами и навигацией.

2.2 Разработка пользовательского интерфейса

Взаимодействие с GymBroBot реализовано через систему текстовых команд, обеспечивающую простое и интуитивно понятное управление. Основной функционал бота доступен через ключевые команды, показанные на рисунке 7:

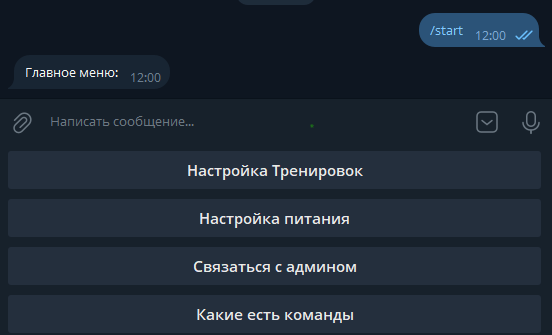


Рисунок 7 - Основные команды

На рисунке 8 представлена команда /start, которая запускает диалог с ботом. Бот предлагает зарегистрироваться, нажав на «Начать регистрацию» для получения списка доступных команд и инструкций по началу работы. Это является отправной точкой для новых пользователей, желающих воспользоваться функционалом GymBroBot.

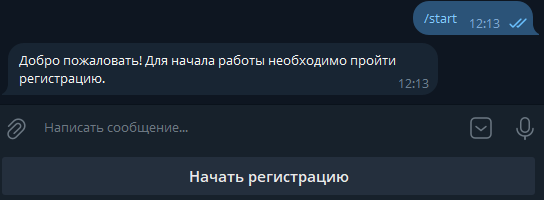


Рисунок 8 - Команда /start

На рисунке 9 показана команда «Начать регистрацию», после нажатия которой бот начинает процесс сбора пользовательских данных, необходимых для персонализации тренировок. Пользователь вводит данные, отвечая на вопросы бота. Бот проверяет, чтобы данные были введены правильно (например, вес и рост должны быть числами).

Если при вводе данных возникнут ошибки (например, пользователь введет текст вместо числа), бот сообщит об этом и предложит исправить ошибку.

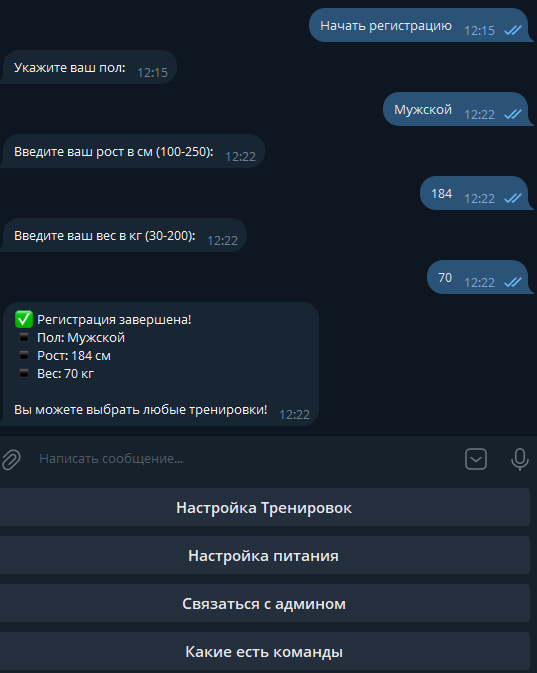
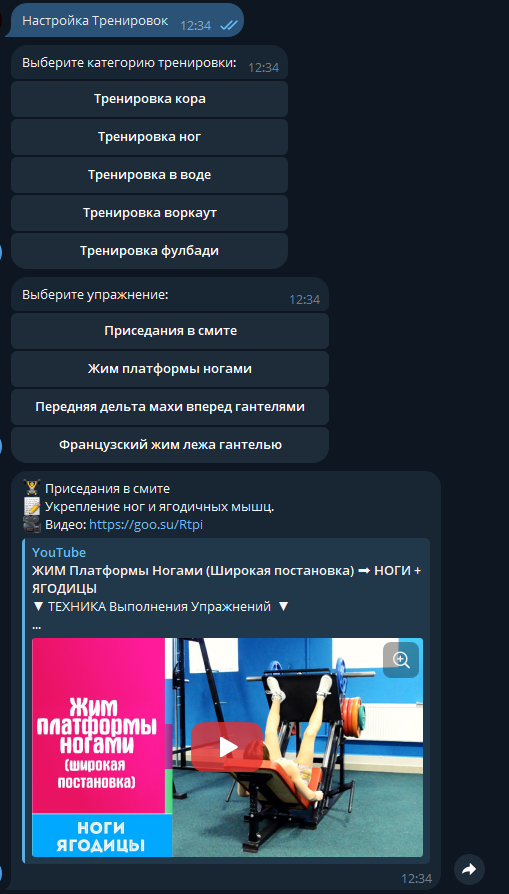


Рисунок 9 - Команда «Начать регистрацию»

На Рисунке 10 представлена команда «Настройка тренировок» позволяет пользователю выбрать подходящую тренировочную программу из предложенных вариантов. При нажатии на соответствующую кнопку бот отправляет список доступных категорий тренировок, среди которых тренировка кора, тренировка ног, тренировка в воде, тренировка воркаута и тренировка в формате фуллбади.

Каждая категория содержит заранее подготовленный набор упражнений, которые бот рекомендует в зависимости от введенных пользователем данных, таких как пол, возраст, рост и вес. Это система рекомендаций, помогающая выбрать наиболее подходящий вариант из уже существующих шаблонов.

После выбора категории бот отправляет список упражнений, входящих в эту программу. Каждое упражнение представлено отдельной кнопкой, нажав на которую пользователь получает видеоинструкцию из базы данных. Это позволяет изучить правильную технику выполнения упражнений.



Рисунки 10 - Команда «Настройка тренировок»

На рисунке 11 показана команда «Настройка питания» позволяет пользователю выбрать наиболее подходящий тип питания из предложенных вариантов. При нажатии на эту кнопку бот открывает раздел питания, где доступны следующие категории: сбалансированное, кето, веганское, высокобелковое и низкоуглеводное питание.

После выбора одного из вариантов бот отправляет подробную информацию о выбранном типе питания. Например, если пользователь нажимает «Сбалансированное», он получает описание этого режима, где указывается оптимальное соотношение белков, жиров и углеводов (БЖУ), основные принципы питания, а также советы по составлению рациона.

Дополнительно бот предлагает примерное меню на день, включающее завтрак, обед, ужин и перекусы. Это помогает пользователю лучше ориентироваться в рационе и соблюдать выбранный режим питания.

Таким образом, функция «Настройка питания» предоставляет удобный способ выбора и изучения различных пищевых режимов. Пользователь может ознакомиться с принципами питания, получить рекомендации и примерное меню, что делает процесс питания более структурированным и понятным.

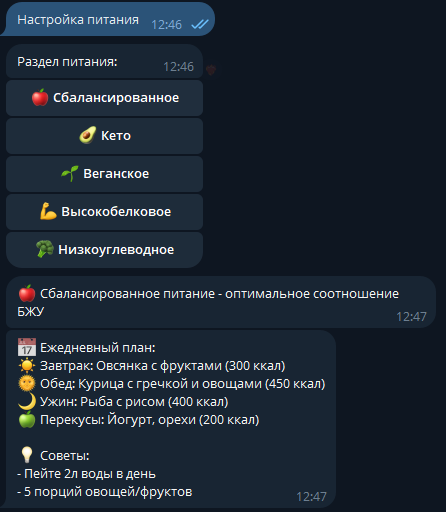


Рисунок 11 - Команда «Настройка питания»

На рисунке 12 показана команда  «Связаться с админом», которая предоставляет пользователю способ связаться с администратором GymBroBot через Telegram. После ввода команды бот отправляет пользователю username администратора в Telegram. Пользователь может использовать этот username, чтобы написать администратору напрямую и задать вопросы о работе бота, сообщить о проблемах или оставить предложения по улучшению.

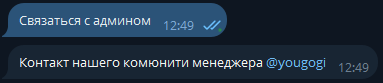


Рисунок 12 - Команда «Связаться с админом»

2.3 Создание БД

База данных (БД) – это ключевой компонент Telegram-бота для управления тренировками. Она хранит информацию о пользователях, упражнениях. В проекте используется MySQL – реляционная СУБД, обеспечивающая надежное хранение данных.

На рисунке 1 показана настройка подключения к базе данных, где настраиваем подключение к SQLite базе данных, используя асинхронный движок SQLAlchemy. База данных будет хранится в файле db.sqlite3, создание фабрик синхронных сессий для работы с БД.

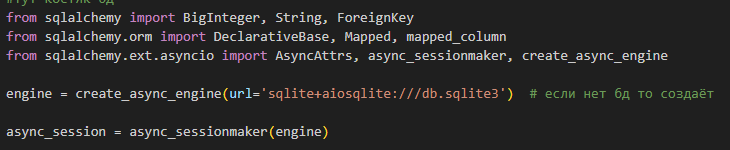


Рисунок 1 – Настройка подключения к базе данных

На рисунке 2 показан базовый класс для моделей, где наследуется от AsyncAttrs для асинхронной работы с атрибутами и от DeclarativeBase для декларативного стиля ORM. Все модели таблиц будут наследоваться от этого класса.

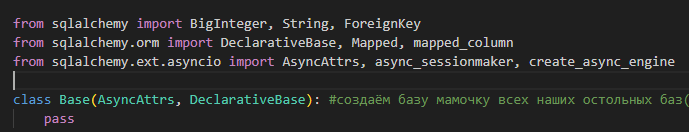


Рисунок 2 – Базовый класс для моделей

На рисунке 3 показана модель пользователя user, где id – первичный ключ (автоинкремент), tg\_id – уникальный ID пользователя в Telegram (BigInteger), gender – пол пользователя (1 символ, может быть NULL), height – рост в см (может быть NULL), weight – вес в кг (может быть NULL), is\_registered – флаг завершения регистрации (по умолчанию False).

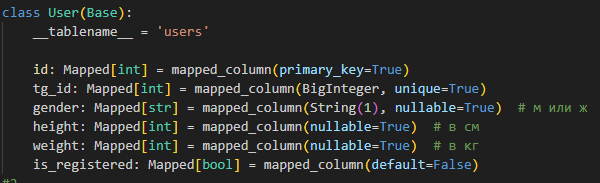


Рисунок 3 – Базовый класс для моделей

На рисунке 4 показана категория тренировок category, где id – первичный ключ, name – название категории (максимум 30 символов).

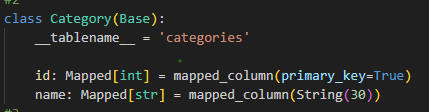


Рисунок 4 – Категория тренировок

На рисунке 5 показана модель тренировок workout, где id – первичный ключ, name – название категории (максимум 30 символов), description – описание (100 символов), videotraining – ссылка на видео (100 символов), category –внешний ключ к таблице категорий.

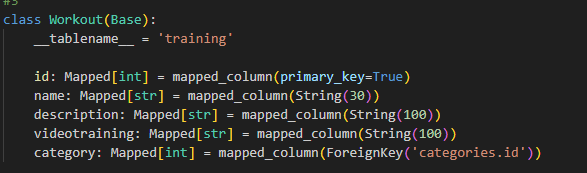


Рисунок 5 – Модель тренировок

На рисунке 6 показана инициализация базы данных, где функция async\_main создает все таблицы при первом запуске и использует асинхронный контекст, вызывает синхронный метод create\_all через run\_sync.

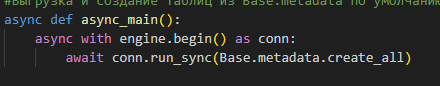


Рисунок 6 – Инициализация базы данных

На рисунке 7 показана инициализация сессии и импорты. Импортируется фабрика асинхронных сессий (async\_session), модели данных (User, Category, Workout) и функция select из SQLAlchemy для построения запросов. Это основа для всех последующих операций с БД.

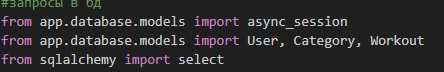


Рисунок 7 – Инициализация сессии и импорты

На рисунке 8 изображено получение пользователя. Создается синхронная сессия через async with, выполняется запрос на поиск пользователя по telegram ID (tg\_id), session.scalar() возвращает один результат или None, типизированный возврат User помогает при разработке.

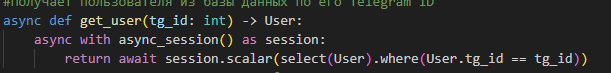


Рисунок 8 – Получение пользователя

На рисунке 9 изображено создание или обновление пользователя, где проверяется существование пользователя, либо обновляет существующие поля, либо создает новую запись. session.add() добавляет новый объект в сессию, session.commit() фиксирует изменения атомарно. Все поля пользователя обновляются единообразно.

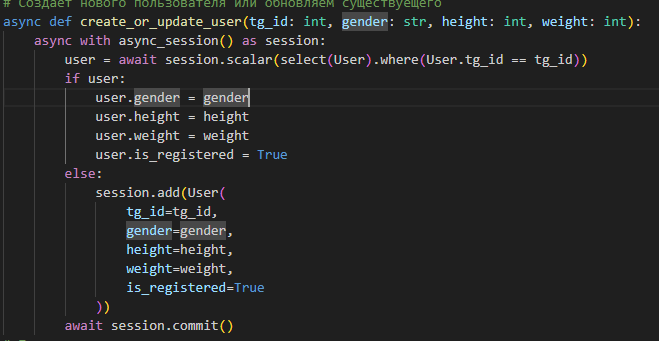


Рисунок 9 – Создание или обновление пользователя

На рисунке 10 показано получение категорий. Простейший запрос без условий, scalars() возвращает итератор объектов category, используется для отображения меню выбора категорий.

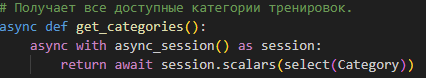


Рисунок 10 – Получение категорий

На рисунке 11 изображено получение тренировок по категориям. Фильтрация через where по внешнему ключу category. Возвращает все упражнения выбранной категории.

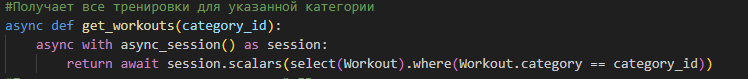


Рисунок 11 – Получение тренировок по категориям

На рисунке 12 изображено получение конкретной тренировки. Точечный запрос по первичному ключу id, возвращает полные данные упражнения, включая видео-ссылку.

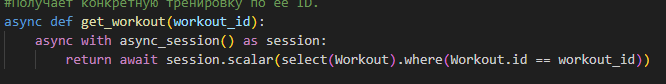


Рисунок 12 – Получение конкретной тренировки

2.4 Взаимодействие БД

На рисунке 1 изображен фрагмент кода из файла keyboards.py содержит два основных типа клавиатур, которые играют ключевую роль в организации взаимодействия пользователя с базой данных. Главное меню (main) реализовано через ReplyKeyboardMarkup и включает четыре основные кнопки: "Настройка Тренировок", "Настройка питания", "Связаться с админом" и "Какие есть команды". Особенность этого меню в том, что первые две кнопки напрямую связаны с запросами к базе данных. При нажатии на "Настройка Тренировок" бот выполняет проверку регистрации пользователя через функцию check\_registration(), которая отправляет запрос get\_user() к базе данных. После успешной проверки загружается список категорий тренировок через get\_categories(), что демонстрирует тесную интеграцию этой клавиатуры с системой хранения данных.

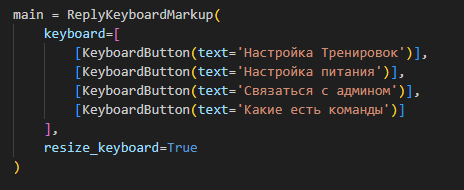


Рисунок 1 – Первая клавиатура

Вторая клавиатура - nutrition\_types - реализована как InlineKeyboardMarkup и предоставляет выбор типа питания. Хотя текущая версия использует статические данные (словарь NUTRITION\_PLANS), эта реализация легко может быть модифицирована для работы с базой данных. Каждая кнопка содержит callback-данные в формате nutrition\_\*, которые в перспективе могут быть связаны с записями в базе данных, позволяя хранить и загружать персонализированные планы питания для разных пользователей. Это открывает возможности для масштабирования системы без изменения базовой логики работы клавиатур.

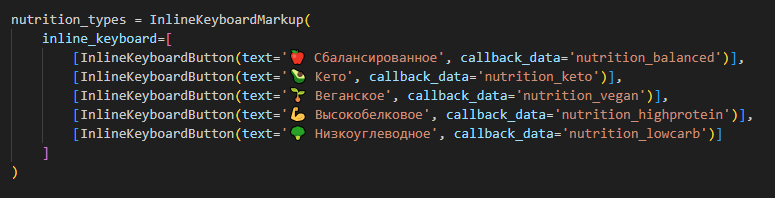


Рисунок 2 – Вторая клавиатура

Основной файл проекта реализует сложную систему взаимодействия с базой данных через несколько ключевых механизмов. В первую очередь стоит отметить процесс регистрации пользователя, который построен на машине состояний (FSM) и включает три последовательных этапа. При обработке команды /start бот проверяет наличие пользователя в базе через функцию check\_registration(), которая выполняет запрос get\_user() к таблице users. Если пользователь не найден, запускается процесс регистрации с поэтапным сбором данных: пол, рост и вес. Каждый этап валидирует ввод и сохраняет промежуточные данные в FSM-контексте. Финальный этап регистрации демонстрирует комплексное взаимодействие с БД - функция create\_or\_update\_user() либо создает новую запись, либо обновляет существующую, используя переданные параметры (tg\_id, gender, height, weight). Особенностью реализации является генерация персонализированных рекомендаций на основе введенного веса, что показывает пример бизнес-логики, работающей с данными из БД. Пример кода обработки веса и сохранения в БД показана на рисунке 3.

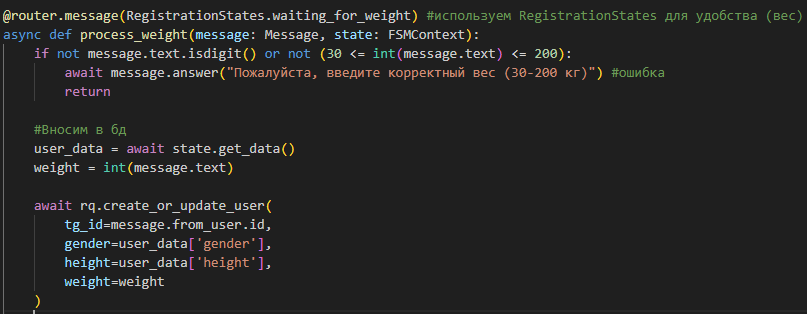


Рисунок 3 – Обработка веса

Работа с тренировками организована через два уровня вложенности: категории и конкретные упражнения. При выборе "Настройка Тренировок" бот загружает список категорий через get\_categories(), затем для выбранной категории подгружает упражнения через get\_workouts(category\_id). Это демонстрирует типичный пример связки таблиц в БД, где workouts связаны с categories через внешний ключ. Каждое упражнение содержит название, описание и ссылку на видео, что формирует полноценную карточку тренировки. Динамическое создание инлайн-клавиатур на основе данных из БД позволяет гибко управлять контентом без изменения кода бота. Пример загрузки и отображения тренировок изображено на рисунке 4.

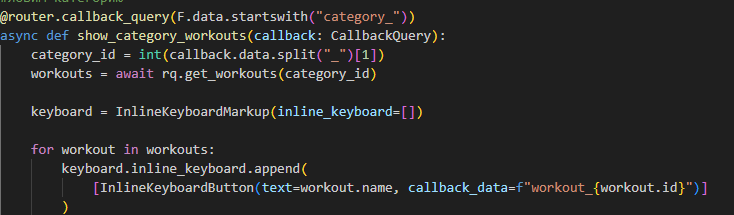


Рисунок 4 – Загрузка и отображение тренировок

3 Тестирование ПП

Тестирование - это процесс оценки качества программного обеспечения путем выявления ошибок, дефектов и несоответствий требованиям.

3.1 Описание видов тестов

Программный продукт был протестирован вручную, без использования автоматизации. Тестировщик выполнял сценарии, как обычный пользователь, и внимательно следил за работой системы, фиксируя все ошибки. Этот подход дал возможность лучше понять потребности пользователей, вовремя реагировать на нестандартные ситуации и заметить те детали, которые сложно выявить автоматизированными средствами.

Таблица 1 – Тест-план.

|  |  |  |  |
| --- | --- | --- | --- |
| Тест-кейс, № | Название | Описание | Метод  тестирования |
| 1. | Нажатие на кнопку /start | Пользователь нажимает кнопку Start для начала работы с ботом. | Ручное |
| 1. | Нажатие на кнопку «Начать регистрацию» | Пользователь нажимает кнопку Начать регистрацию для ввода данных. | Ручное |
| 3. | **Ввод данных о пользователе** | Пользователь вводит пол, рост, вес (в пределах допустимых значений). | Ручное |
| 4. | Завершение регистрации | Проверка отправки username администратора. | Ручное |
| 5. | Настройка тренировки | Пользователь выбирает настройку тренировки (например, тренировка кора, тренировка ног и т.д.). | Ручное |
| 6. | Настройка питания | Пользователь выбирает тип питания (например, сбалансированное, кето, веганское). | Ручное |
| 6. | Проверка работы кнопки «Начать регистрацию» | Проверка работы кнопки «Начать регистрацию» для зарегистрированного пользователя. | Ручное |

3.3 Тест-кейсы

В таблице 2 представлен тест-кейс, который проверяет, что при нажатии на кнопку Start бот правильно реагирует и отправляет пользователю сообщение с кнопкой для начала регистрации.

Таблица 2 – Тест-кейс №1 «Нажатие на кнопку /start».

|  |  |  |
| --- | --- | --- |
| Название | Нажатие на кнопку /start | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | - Пройден  - Провален |
| Предусловие | Аккаунт не зарегистрирован | |
| Шаги теста |  | |
| Запуск программы с помощью /start | Успешный запуск программы. Вывод программы для запуска с кнопкой для дальнейшей работы | Пройден |

В таблице 3 представлен тест-кейс, который проверяет, что при нажатии на кнопку «Начать регистрацию» бот правильно реагирует и отображает запрос на ввод данных для регистрации пользователя.

Таблица 3 – Тест-кейс №2 «Нажатие на кнопку «Начать регистрацию»».

|  |  |  |
| --- | --- | --- |
| Название | Нажатие на кнопку «Начать регистрацию» | |
| Тип тестирования | Функциональное | |
| Действие | Ожидаемый результат | - Пройден  - Провален |
| Предусловие | Пользователь нажал на кнопку Start и бот отобразил сообщение с кнопкой Начать регистрацию. | |
| Шаги теста |  | |
| Пользователь нажимает на кнопку «Начать регистрацию» | Бот должен запросить у пользователя следующие данные: пол, рост, вес в рамках допустимых значений. | Пройден |

В таблице 4 представлен тест-кейс в котором тестируется функционал добавления резюме от нового пользователя.

Таблица 4 – Тест-кейс №3 «**Ввод данных о пользователе**».

|  |  |  |
| --- | --- | --- |
| Название | **Ввод данных о пользователе** | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | - Пройден  - Провален  - Заблокирован |
| Предусловие | Пользователь авторизован | |
| Шаги теста |  | |
| Пользователь вводит свои данные | Бот должен корректно принять все введенные данные, проверить их на соответствие допустимым значениям и отобразить сообщение о завершении регистрации. | Пройден |
| Пользователь вводит некорректные данные | Бот должен отклонить неверные данные (например, рост 350 или вес 500) и запросить корректные данные, отправив сообщение о том, что данные введены неправильно. | Пройден |

В таблице 5 представлен тест-кейс, который должен завершить процесс регистрации и уведомить пользователя о том, что регистрация прошла успешно.

Таблица 5 – Тест-кейс №4 «Завершение регистрации».

|  |  |  |
| --- | --- | --- |
| Название | Завершение регистрации | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | - Пройден  - Провален  - Заблокирован |
| Предусловие | Пользователь ввел все данные | |
| Шаги теста |  | |
| Пользователь завершает ввод данных | Бот должен подтвердить успешное завершение регистрации и уведомить пользователя о том, что регистрация прошла успешно | Пройден |

В таблице 6 представлен тест-кейс в котором тестируется функционал «Настройки тренировок».

Таблица 6 – Тест-кейс №5 «Настройка тренировок».

|  |  |  |
| --- | --- | --- |
| Название | Настройка тренировки | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | - Пройден  - Провален  - Заблокирован |
| Предусловие |  | |
| Шаги теста |  | |
| Пользователь нажимает кнопку «Настройка тренировок». | Бот должен предложить выбор тренировок | Пройден |
| Пользователь выбирает тренировку кора | Бот должен отправить список упражнений для тренировки кора | Пройден |
| Пользователь нажимает на одно из упражнений | Бот отправляет подробности о выбранном упражнении | Пройден |

В таблице 7 представлен тест-кейс в котором тестируется проверка работы кнопки .

Таблица 7 – Тест-кейс №6 «Начать регистрацию».

|  |  |  |
| --- | --- | --- |
| Название | Начать регистрацию | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | - Пройден  - Провален  - Заблокирован |
| Предусловие |  | |
| Шаги теста |  | |
| Пользователь нажимает кнопку «Настройка питания». | Бот должен предложить выбор раздел питания. | Пройден |
| Пользователь выбирает «Сбалансированное питание». | Бот должен отправить план питания | Пройден |

В таблице 8 представлен тест-кейс в котором тестируется проверка кнопки «Начать регистрацию» для уже зарегистрированного пользователя.

Таблица 8 – Тест-кейс №7 «Проверка работы кнопки «Начать регистрацию»».

|  |  |  |
| --- | --- | --- |
| Название | Проверка работы кнопки «Начать регистрацию» | |
| Тип тестирования | Ручное | |
| Действие | Ожидаемый результат | - Пройден  - Провален  - Заблокирован |
| Предусловие | Пользователь зарегистрирован | |
| Шаги теста |  | |
| Открыть бота и ввести «Начать регистрацию» и убедиться что регистрация не начинается заново. | Бот должен отобразить, что пользователь уже зарегистрирован. | Провален |

Заключение

Проект GymBroBot успешно завершен как платформа, предоставляющая пользователям доступ к персонализированным планам тренировок. С применением современных технологий, включая Python (aiogram) и MySQL. Мы создали удобный инструмент, который предлагает широкий выбор программ тренировок. GymBroBot позволяет пользователям указать свои параметры, такие как вес, рост и возраст, для получения доступа к наиболее подходящим планам тренировок. Пользователи получают готовые детальные программы тренировок, что обеспечивает удобство просмотра и использования. Тщательное функциональное тестирование подтвердило соответствие GymBroBot заявленным требованиям, гарантируя надежный доступ к персонализированным планам тренировок и стабильную работу.Приложение А

Проект GymBroBot - это удобный инструмент, разработанный для помощи пользователям в выборе и получении индивидуальных планов тренировок. Его основная задача - предоставить пользователю доступ к персонализированным планам тренировок, которые соответствуют его параметрам и фитнес-целям.

Система работает следующим образом: пользователь указывает свои основные данные, такие как рост, вес и возраст. Далее система предоставляет ему список планов тренировок, которые наилучшим образом подходят под его параметры. Пользователь может выбрать понравившийся план и начать тренировки.

Таким образом, GymBroBot помогает пользователям получить доступ к персонализированным программам тренировок, разработанным специально для них, и тем самым повысить эффективность своих занятий.

Листинг 1 – main.py

import asyncio  
import os  
from aiogram import Bot, Dispatcher  
from app.handlers import router  
from app.database.models import async\_main  
from pathlib import Path  
async def main():  
    await async\_main()  
    bot = Bot(token='7806482724:AAHM2Q-HEfUDryBEok2yysmSloEc37L-iYc') #токен  
    dp = Dispatcher()  
    dp.include\_router(router)  
    try:  
        await dp.start\_polling(bot)  
    except Exception as e:  
        print(f"Error: {e}")  
    finally:  
        await bot.session.close()  
if \_\_name\_\_ == '\_\_main\_\_':  
    try:  
        asyncio.run(main())  
    except KeyboardInterrupt:  
        print('Bot stopped')

Приложение Б

Данный код реализует взаимодействие Telegram-бота с базой данных SQLite через асинхронные запросы SQLAlchemy. Основные функции включают регистрацию пользователей (хранение пола, роста, веса), управление тренировками (категории и упражнения) и проверку доступа для зарегистрированных пользователей. Код организован по принципу разделения ответственности: models.py определяет структуру БД, requests.py содержит CRUD-операции, а handlers.py реализует бизнес-логику бота.

Листинг 1 – requests.py

from app.database.models import async\_session  
from app.database.models import User, Category, Workout  
from sqlalchemy import select  
  
async def get\_user(tg\_id: int) -> User:  
    async with async\_session() as session:  
        return await session.scalar(select(User).where(User.tg\_id == tg\_id))  
  
async def create\_or\_update\_user(tg\_id: int, gender: str, height: int, weight: int):  
    async with async\_session() as session:  
        user = await session.scalar(select(User).where(User.tg\_id == tg\_id))  
        if user:  
            user.gender = gender  
            user.height = height  
            user.weight = weight  
            user.is\_registered = True  
        else:  
            session.add(User(  
                tg\_id=tg\_id,  
                gender=gender,  
                height=height,  
                weight=weight,  
                is\_registered=True  
            ))  
        await session.commit()  
async def get\_categories():  
    async with async\_session() as session:  
        return await session.scalars(select(Category))

async def get\_workouts(category\_id):  
    async with async\_session() as session:

        return await session.scalars(select(Workout).where(Workout.category == category\_id))  
  
async def get\_workout(workout\_id):  
    async with async\_session() as session:  
        return await session.scalar(select(Workout).where(Workout.id == workout\_id))

Листинг 2 – models.py

from sqlalchemy import BigInteger, String, ForeignKey  
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped\_column  
from sqlalchemy.ext.asyncio import AsyncAttrs, async\_sessionmaker, create\_async\_engine  
engine = create\_async\_engine(url='sqlite+aiosqlite:///db.sqlite3')

async\_session = async\_sessionmaker(engine)  
from sqlalchemy import BigInteger, String, ForeignKey  
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped\_column  
from sqlalchemy.ext.asyncio import AsyncAttrs, async\_sessionmaker, create\_async\_engine  
class Base(AsyncAttrs, DeclarativeBase):    
  
class User(Base):  
    \_\_tablename\_\_ = 'users'  
     
    id: Mapped[int] = mapped\_column(primary\_key=True)  
    tg\_id: Mapped[int] = mapped\_column(BigInteger, unique=True)  
    gender: Mapped[str] = mapped\_column(String(1), nullable=True)   
    height: Mapped[int] = mapped\_column(nullable=True)     
    weight: Mapped[int] = mapped\_column(nullable=True)     
    is\_registered: Mapped[bool] = mapped\_column(default=False)  
  
class Category(Base):  
    \_\_tablename\_\_ = 'categories'  
     
    id: Mapped[int] = mapped\_column(primary\_key=True)  
    name: Mapped[str] = mapped\_column(String(30))  
  
class Workout(Base):  
    \_\_tablename\_\_ = 'training'  
     
    id: Mapped[int] = mapped\_column(primary\_key=True)  
    name: Mapped[str] = mapped\_column(String(30))

    description: Mapped[str] = mapped\_column(String(100))  
    videotraining: Mapped[str] = mapped\_column(String(100))  
    category: Mapped[int] = mapped\_column(ForeignKey('categories.id'))

async def async\_main():  
    async with engine.begin() as conn:  
        await conn.run\_sync(Base.metadata.create\_all)

Листинг 3 – handlers.py

from aiogram import F, Router  
from aiogram.filters import CommandStart  
from aiogram.fsm.context import FSMContext  
from app.database.requests import get\_user, create\_or\_update\_user, get\_categories, get\_workouts, get\_workout  
from app.keyboards import main, nutrition\_types

router = Router()

@router.message(CommandStart())  
async def cmd\_start(message: Message):  
    if not await get\_user(message.from\_user.id):  
        await message.answer("Для начала работы зарегистрируйтесь.", reply\_markup=...)  
    else:  
        await message.answer("Главное меню:", reply\_markup=main)

@router.message(F.text == 'Настройка Тренировок')  
async def workout\_setup(message: Message):  
    categories = await get\_categories()  
    keyboard = InlineKeyboardMarkup(inline\_keyboard=[  
        [InlineKeyboardButton(text=cat.name, callback\_data=f"category\_{cat.id}")   
        for cat in categories  
    ])  
    await message.answer("Выберите категорию:", reply\_markup=keyboard)

Приложение В

Аннотация

Настоящее руководство предназначено для пользователей Telegram-бота GymBroBot, разработанного для предоставления персонализированных планов тренировок. GymBroBot анализирует введенные пользователем данные и предоставляет доступ к наиболее подходящим программам тренировок в удобном формате.

Цель GymBroBot – облегчить пользователям процесс выбора подходящей программы тренировок, предоставляя готовые решения, основанные на индивидуальных параметрах. Руководство содержит описание функциональности бота, инструкции по использованию, а также рекомендации по решению возможных проблем.

Документ предназначен для всех пользователей Telegram, заинтересованных в получении персонализированных планов тренировок для достижения своих фитнес-целей.

Оформление программного документа «Руководство пользователя» произведено по требованиям ЕСПД (ГОСТ 19.101-77 [[1]](#footnote-1)1), ГОСТ 19.103-77 [[2]](#footnote-2)2), ГОСТ 19.104-78\* [[3]](#footnote-3)3), ГОСТ 19.105-78\* [[4]](#footnote-4)4), ГОСТ 19.106-78\* [[5]](#footnote-5)5), ГОСТ 19.505-79\* [[6]](#footnote-6)6), ГОСТ 19.604-78\* [[7]](#footnote-7)7)).

1 Введение

1.1 Область применения

Настоящее руководство пользователя предназначено для ознакомления с функциональными возможностями телеграм-бота GymBroBot, а также для получения практических рекомендаций по его использованию.

Руководство ориентировано на всех, кто заинтересован в получении доступа к персонализированным планам тренировок, основанным на их индивидуальных параметрах и целях.

1.2 Краткое описание возможностей

Основные возможности бота включают:

* регистрация пользователя;
* ввод основных данных, таких как рост, вес и возраст;
* получение доступа к списку подходящих планов тренировок;
* удобная и интуитивно понятная навигация через команды и кнопки.

1.3 Уровень подготовки пользователя

Telegram-бот GymBroBot разработан для пользователей с любым уровнем технической подготовки. Для его использования достаточно базового понимания работы Telegram и умения вести переписку в чатах. Специальных знаний или навыков не требуется. Интерфейс бота интуитивно понятен и прост в использовании.

2 Назначение и условия применения

2.1 Назначение

Telegram-бот GymBroBot разработан для предоставления пользователям персонализированных планов тренировок, основанных на их индивидуальных параметрах. GymBroBot служит удобным инструментом для быстрого доступа к подходящим программам тренировок, помогая пользователям достигать своих фитнес-целей.

2.2 Условия применения

Для использования Telegram-бота GymBroBot необходимо устройство с доступом в интернет и установленным приложением Telegram. Бот работает в стандартном интерфейсе Telegram и не требует установки дополнительного программного обеспечения. Использовать бота можно в любом месте и в любое время при наличии стабильного интернет-соединения. GymBroBot предназначен для индивидуального использования.

3 Подготовка к работе

3.1 Состав и содержание носителя данных

Telegram-бот GymBroBot размещен на удаленном сервере и доступен через Telegram. Данные, используемые ботом, хранятся в базе данных, связанной с серверной частью приложения. Это облачное хранилище содержит информацию о пользователях (рост, вес, возраст), а также перечень доступных планов тренировок.

3.2 Порядок загрузки программ и данных

Для начала работы с Telegram-ботом GymBroBot необходимо открыть Telegram и перейти по ссылке на бота или найти его через строку поиска. Дополнительная установка не требуется, так как бот работает в облачной среде и доступен напрямую через интерфейс Telegram.

При первом запуске бот предложит зарегистрироваться и ввести основные данные: рост, вес и возраст. Вся последующая загрузка и обновление данных осуществляется через команды и кнопки внутри диалога, без необходимости каких-либо дополнительных действий со стороны пользователя.

3.3 Порядок проверки работоспособности

Проверка работоспособности Telegram-бота GymBroBot осуществляется непосредственно через взаимодействие с ним в приложении Telegram. После запуска бота пользователь должен получить приветственное сообщение и доступ к основным функциям, таким как регистрация, ввод параметров и получение доступа к планам тренировок. Успешный отклик бота на команды и корректное отображение информации свидетельствуют о его работоспособности. В случае отсутствия ответа, появления ошибок или некорректного поведения необходимо проверить подключение к интернету, убедиться в актуальности версии Telegram и повторно запустить бота.

4 Описание операций

Взаимодействие с Telegram-ботом GymBroBot происходит в диалоговом режиме внутри мессенджера Telegram. Пользователь начинает работу, отправляя команду или выбирая пункт из меню. Каждая операция представляет собой последовательность шагов для ввода данных, получения планов тренировок или получения информации.

Первым этапом является запуск бота. После перехода по ссылке или поиска в Telegram бот отправляет приветственное сообщение и предлагает зарегистрироваться. Пользователь вводит основные данные о себе: рост, вес, возраст. Эти данные используются для рекомендации по плану тренировок наиболее подходящих планов тренировок.

Следующей ключевой операцией является получение доступа к списку подходящих планов тренировок. Бот анализирует введенные пользователем данные и предлагает наиболее релевантные программы тренировок. Пользователь может просматривать планы, выбирать подходящий и начинать тренировки.

5 Аварийные ситуации

В процессе работы с Telegram-ботом GymBroBot могут возникнуть некоторые проблемы, вызванные различными факторами. Ниже приведены наиболее распространенные ошибки и способы их устранения.

Отсутствие реакции на команды может быть вызвано проблемами с подключением к интернету, устаревшей версией Telegram или ошибками в написании команды. В этом случае проверьте подключение к интернету, обновите приложение Telegram до последней версии и убедитесь, что команда введена правильно.

Некорректное отображение информации может быть связано с ошибками в данных, переданных пользователем, или проблемами на сервере. Проверьте правильность введенных данных и попробуйте перезапустить бот. Если проблема сохраняется, обратитесь в службу поддержки.

Невозможность загрузки планов тренировок может возникнуть из-за проблем с подключением к интернету или временных сбоев на сервере. В этом случае проверьте подключение к интернету и попробуйте повторить запрос позже. Если проблема не исчезает, обратитесь в службу поддержки.

Если бот не отвечает (не работает), это может быть связано с техническими проблемами на стороне сервера или временными сбоями. Подождите некоторое время и попробуйте снова. Если проблема сохраняется, перезапустите бот или обратитесь в службу поддержки.

В случае возникновения любых проблем, рекомендуется в первую очередь проверить подключение к интернету и убедиться в актуальности версии Telegram. Если проблема не исчезает, обратитесь в службу поддержки, используя контактную информацию, предоставленную в интерфейсе бота.

6 Рекомендации по освоению

Освоение работы с Telegram-ботом GymBroBot не займет много времени. Для начала ознакомьтесь с приветственным сообщением бота.

Пройдите регистрацию и введите свои основные данные. Это позволит боту подобрать для вас рекомендации по плану тренировок.

Попробуйте разные варианты, чтобы найти те планы тренировок, которые вам больше всего нравятся. Обратите внимание на описание каждого плана, чтобы выбрать наиболее подходящий для ваших целей.

В случае возникновения вопросов, обратитесь к данному руководству пользователя. Регулярное использование GymBroBot поможет вам достичь ваших фитнес-целей.

1. 1) ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. 2) ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. 3) ГОСТ 19.104-78\* ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. 4) ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. 5) ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. 6) ГОСТ 19.505-79\* ЕСПД. Руководство пользователя. Требования к содержанию и оформлению [↑](#footnote-ref-6)
7. 7) ГОСТ 19.604-78\* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)