**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**(Университет ИТМО)**

Факультет **Прикладной информатики**

Направление подготовки **09.03.03 Прикладная информатика**

Образовательная программа **Мобильные и сетевые технологии**

**К У Р С О В О Й   П Р О Е К Т**

Тема: «Мобильное приложение для отслеживания достижения целей личностного развития»

Обучающийся: Петрова Мария Валерьевна К3139

Санкт-Петербург 2024

[**ВВЕДЕНИЕ 3**](#_w1jg8pf6qgmo)

[Актуальность темы 3](#_ur5fnj4hc5gy)

[Цель проекта 3](#_tpbr4ptbd5m2)

[Задачи проекта 3](#_ibmhg3haevuj)

[**ОСНОВНАЯ ЧАСТЬ 4**](#_k1d6cshygurl)

[Суть проекта 4](#_jtpsmhcucfx1)

[Процессы работы над проектом 4](#_uizxsfy1zgu4)

[Раскрытие сути проблемы 8](#_wl1ry1xizeaj)

[Описание решения задачи 9](#_lwwuqks6g7wr)

[Анализ работы 11](#_u8q43ki8pfu1)

[Взаимодействие с командой 12](#_91mgexmx3g23)

[Взаимодействие с руководителем проекта 12](#_hwggkdhgfeib)

[Оценка работы руководителя 12](#_orfmast4f39b)

[**ЗАКЛЮЧЕНИЕ 13**](#_7spn02fwf0qs)

[Оценка выполнения всего проекта 13](#_6fo3j657ph37)

[Вклад в достижение цели 14](#_o59ojccdp7xy)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16**](#_2siepeukd4zy)

[**ПРИЛОЖЕНИЕ 17**](#_9c75n7zc4yh7)

# **ВВЕДЕНИЕ**

### **Актуальность темы**

В современном мире личностное развитие становится важной составляющей жизни каждого человека. Технологии и мобильные приложения могут значительно упростить процесс постановки и достижения целей, предоставляя пользователям удобные инструменты для планирования, мониторинга и визуализации прогресса. Создание приложения для отслеживания личных достижений является актуальной задачей, способствующей популяризации методов саморазвития и повышения мотивации.

### **Цель проекта**

Цель проекта заключается в разработке мобильного приложения, позволяющего пользователям ежедневно отслеживать прогресс по достижению своих целей, создавать отчеты, просматривать результаты других пользователей и визуализировать достижения в удобной форме.

### **Задачи проекта**

Для реализации цели были поставлены следующие задачи:

1. Подготовка и утверждение технического задания.
2. Проектирование и разработка серверной части с API-эндпоинтами.
3. Создание пользовательского интерфейса и разработка дизайна.
4. Реализация модулей: просмотр отчетов, создание отчетов, визуализации прогресса.
5. Проведение комплексного тестирования приложения.
6. Презентация проекта и его защита.

# **ОСНОВНАЯ ЧАСТЬ**

### **Суть проекта**

Проект представляет собой разработку мобильного приложения, позволяющего пользователям отслеживать их ежедневный прогресс в достижении личных целей, создавать отчеты о выполненной работе, а также просматривать результаты других пользователей. Дополнительной функцией приложения является возможность визуализации прогресса, что позволяет пользователю видеть свои достижения в наглядной графической форме. Основная цель проекта — предоставить инструмент для мониторинга и оценки достижений, способствующий мотивации пользователей к выполнению поставленных задач.

Приложение было разработано с использованием современных технологий: серверная часть написана на языке программирования Go, который известен своей высокой производительностью и возможностью построения масштабируемых решений. Клиентская часть была реализована с помощью фреймворка Flutter, предоставляющего удобный способ создания кроссплатформенных мобильных приложений.

### **Процессы работы над проектом**

Работа над проектом велась в несколько этапов, начиная с подготовки технического задания и заканчивая реализацией финальных модулей и тестированием. Разработка была разделена между backend и frontend частями, которые интегрировались через REST API.

1. Подготовка технического задания

На начальном этапе были определены основные цели проекта, функциональные требования и предполагаемые этапы разработки. В результате согласования технического задания были выделены следующие ключевые функции:

* Создание, удаление и просмотр отчетов.
* Авторизация пользователей через Google OAuth.
* Визуализация прогресса пользователя в графической форме.
* Интерактивный интерфейс для удобного взаимодействия с приложением.

2. Разработка серверной части

Серверная часть была разработана с использованием языка Go и базы данных PostgreSQL[1]. Архитектура приложения строилась по принципам слоистой структуры[2], как показано на рисунке 1:

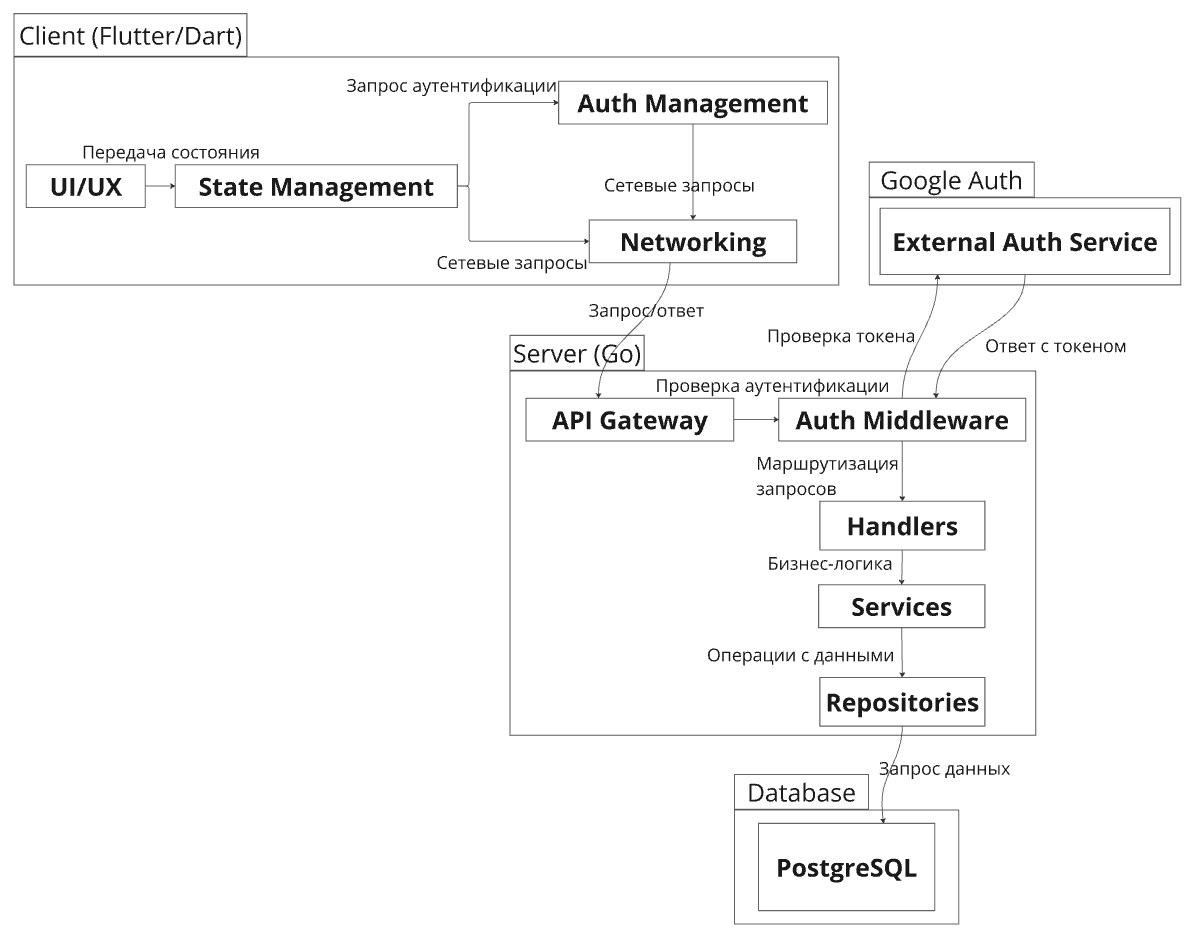


Рисунок 1 – Пример Архитектуры

* **Handlers (обработчики):** Этот слой отвечал за прием HTTP-запросов от клиентской части, передачу данных в сервисы и возвращение ответа в формате JSON.
* **Services (сервисы):** Слой бизнес-логики, который обрабатывал данные, выполнял валидацию запросов, проверку прав доступа и логику работы приложения.
* **Repositories (репозитории):** Слой, ответственный за взаимодействие с базой данных. Здесь выполнялись SQL-запросы, такие как добавление, удаление и получение данных из PostgreSQL.

Для аутентификации пользователей был реализован механизм Google OAuth, который позволял пользователям регистрироваться и входить в приложение через Google-аккаунт. Для защиты данных использовались JWT-токены, которые проверялись при каждом запросе к защищенным маршрутам.

REST API был реализован для обеспечения взаимодействия между клиентом и сервером. Были созданы следующие эндпоинты:

* **GET /reports:** Получение списка отчетов пользователя.
* **POST /reports:** Добавление нового отчета.
* **DELETE /reports/{id}:** Удаление отчета по идентификатору.
* **GET /user:** Получение информации о текущем авторизованном пользователе.

Каждый эндпоинт обрабатывал данные в формате JSON, что делало API универсальным и удобным для использования.

3. Проектирование базы данных

База данных была спроектирована с учетом надежности и удобства хранения данных. Были реализованы следующие таблицы:

* **Users:** Содержит информацию о пользователях (ID, email, имя).
* **Reports:** Содержит данные об отчетах пользователей (ID, описание, дата, оценка, связь с пользователем через user\_id).

Схема базы данных предусматривала автоматическое создание таблиц при запуске приложения, что исключало возможность ошибок в развертывании и упрощало настройку системы.

4. Разработка клиентской части

Клиентская часть была разработана на Flutter, что позволило создать кроссплатформенное приложение с удобным пользовательским интерфейсом. Основные модули включали:

* **Модуль просмотра отчетов:** Отображение списка всех отчетов пользователя.
* **Модуль создания отчетов:** Пользователь мог добавлять новые отчеты с указанием описания, даты и оценки прогресса.
* **Модуль визуализации прогресса:** Графическое отображение прогресса пользователя на основе введенных данных, например, в виде диаграмм или шкал.

Для интеграции клиентской части с серверной использовался REST API. Дизайн интерфейсов был разработан с учетом удобства для пользователей, а все данные синхронизировались с сервером в реальном времени.

5. Интеграция и тестирование

После реализации всех модулей приложения была проведена интеграция серверной и клиентской частей. Особое внимание было уделено тестированию:

* Тестирование API: проверка корректности обработки запросов и ответов.
* Тестирование пользовательского интерфейса: выявление багов и проверка удобства использования.
* Нагрузочное тестирование: проверка производительности сервера при увеличении числа пользователей.

**Раскрытие сути проблемы**

Основная проблема, поставленная передо мной в рамках выполнения проекта, заключалась в разработке надежной и производительной серверной части мобильного приложения, способной обеспечить:

* Обработку пользовательских данных,
* Создание и управление отчетами,
* Реализацию безопасной аутентификации пользователей через Google OAuth,
* Эффективное взаимодействие с клиентской частью через REST API.

Сложности заключались в необходимости соблюдения высоких требований к производительности, безопасности данных и масштабируемости. Так как приложение должно было поддерживать многопользовательский режим, передо мной стояла задача разработать архитектуру, способную без сбоев работать с большим объемом данных. Дополнительно требовалось учесть требования к валидации вводимых данных и реализации логики API.

### **Описание решения задачи**

Для решения поставленных задач был выбран язык Go[6], который подходит для высокопроизводительных и масштабируемых приложений. Основные этапы моей работы включали следующее:

1. **Проектирование базы данных**Я создала схему базы данных на основе PostgreSQL[3]. База данных включала следующие таблицы:

* Таблица **users**, хранящая данные о пользователях (идентификатор, имя, email).
* Таблица **reports**, в которой сохранялись данные отчетов (идентификатор, описание, дата, связь с пользователем через user\_id).

Таблица 1 – Таблица с данными о пользователях

| Колонка | Тип данных | Описание |
| --- | --- | --- |
|  | INT | Уникальный id для каждого пользователя |
| Name | VARCHAR | Имя пользователя |

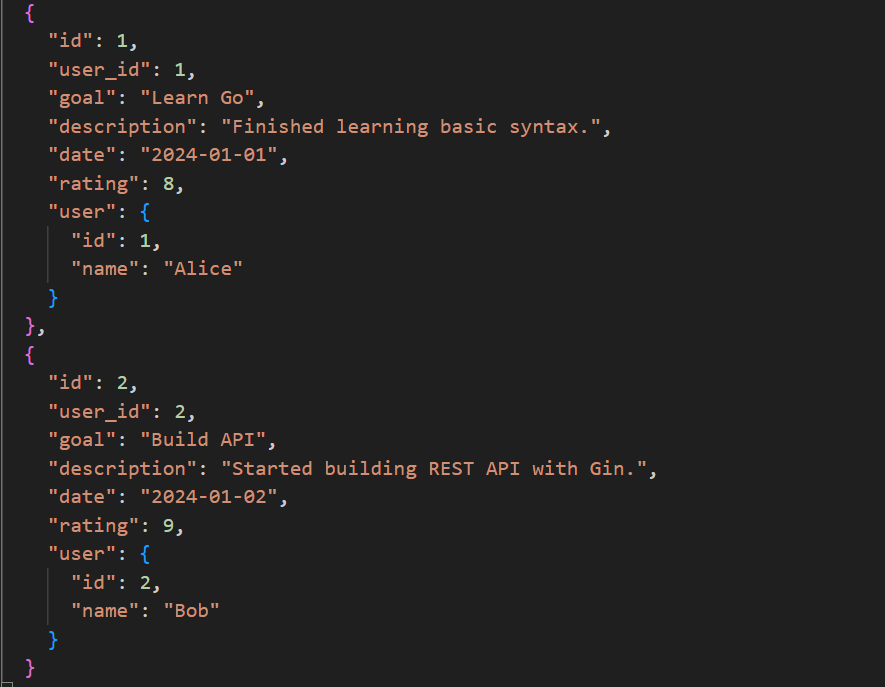
Таблица 2 – Таблица с данными об отчетах и целях

| Колонка | Тип данных | Описание |
| --- | --- | --- |
| ID | INT | Уникальный id для каждого отчета |
| User id | INT | id пользователя |
| goal | VARCHAR | Описание цели |
| description | TEXT | Описание отчета |
| date | DATE | Дата создания отчета |
| rating | INT | Оценка по 10-бальной шкале |

При проектировании я использовала внешние ключи для обеспечения связей между таблицами, что позволило реализовать структурированное хранение данных.

1. **Реализация API для работы с отчетами**Были разработаны три ключевых эндпоинта[4]:

* **GET /reports**: GET-эндпоинт возвращает список всех отчетов из базы данных. Эндпоинт успешно извлекает данные из базы данных и возвращает их в формате JSON. Для каждого отчета передаются описание, дата создания и оценка. Результаты работы эндпоинта соответствуют требованиям, пример JSON-ответа представлен на рисунке 2.

Рисунок 2 – Пример JSON ответа

* **POST /reports**: реализован POST-эндпоинт, позволяющий добавлять новые отчеты в базу данных. Для каждого создаваемого отчета проводится обязательная валидация данных, включая проверку корректности длины описания и диапазона значений оценки. Также была добавлена обработка ошибок, возникающих при передаче некорректных данных. Эндпоинт протестирован, он корректно обрабатывает запросы на создание отчетов и возвращает соответствующие ответы клиенту.
* **DELETE /reports/{id}**: Реализован DELETE-эндпоинт, который позволяет удалять отчеты из базы данных по заданному идентификатору. Была проработана обработка возможных ошибок, включая ситуации, когда указанный отчет отсутствует в базе данных. Эндпоинт успешно протестирован и корректно выполняет удаление отчетов, обеспечивая надежную работу системы.

1. **Интеграция с фронтендом через REST API**Для обеспечения связи с клиентской частью, написанной на Flutter, был реализован REST API. Это позволило легко интегрировать бэкенд с фронтендом, обеспечив обмен данными в формате JSON.
2. **Аутентификация и безопасность данных**Я настроила аутентификацию пользователей через Google OAuth и JWT[5]. После авторизации пользователь получал JWT-токен, который использовался для доступа к защищенным маршрутам приложения. Это решение обеспечило высокий уровень безопасности данных.

### **Анализ работы**

В ходе выполнения проекта я успешно справилась с основной задачей разработкой серверной части приложения. Ключевые результаты включали:

* Рабочий API с поддержкой основных операций над отчетами,
* Продуманную архитектуру базы данных,
* Надежный механизм аутентификации.

Сложности возникли при интеграции Google OAuth, так как потребовалось изучить документацию и протестировать различные сценарии работы. Однако благодаря последовательному подходу мне удалось успешно решить эту проблему.

### **Взаимодействие с командой**

Моя работа была тесно связана с фронтенд-разработчиками. Мы регулярно обменивались информацией о форматах запросов и ответов для корректной интеграции API. Совместная работа позволила сократить время на отладку и улучшить качество конечного продукта.

### **Взаимодействие с руководителем проекта**

Руководитель проекта предоставил четкие задачи и своевременную обратную связь. Его рекомендации позволили улучшить структуру API и избежать потенциальных ошибок.

### **Оценка работы руководителя**

Руководитель проявил высокий профессионализм, четко обозначил цели и поддерживал на всех этапах разработки. Благодаря его подходу я смогла сконцентрироваться на выполнении своих задач и добиться запланированных результатов.

# **ЗАКЛЮЧЕНИЕ**

### **Оценка выполнения всего проекта**

В ходе реализации проекта удалось достичь поставленной цели — разработать мобильное приложение для Android, которое позволяет пользователям создавать ежедневные отчеты о прогрессе, просматривать отчеты других пользователей и визуализировать свой прогресс. Все поставленные задачи были выполнены в полном объеме:

* Создана надежная серверная часть приложения с использованием языка Go.
* Реализован REST API для взаимодействия между клиентской и серверной частью, включая операции создания, просмотра и удаления отчетов.
* Разработана структура базы данных, отвечающая всем требованиям проекта. Таблицы были успешно спроектированы для удобного хранения пользовательских данных и отчетов.
* Настроена аутентификация через Google OAuth с использованием JWT-токенов для обеспечения безопасности пользовательских данных.
* Фронтенд разработан на Flutter и успешно интегрирован с серверной частью.

Проект завершился успешным тестированием и проверкой всех функциональных модулей, включая API-эндпоинты и авторизацию. Тесты подтвердили стабильность и производительность серверной части, а также ее готовность к масштабируемой эксплуатации.

Определенные сложности возникли на этапе интеграции Google OAuth и настройки валидации данных. Эти трудности были успешно преодолены благодаря тщательному анализу документации и тестированию различных сценариев работы системы.

Таким образом, можно сделать вывод, что цель проекта была достигнута. Приложение полностью соответствует заявленным требованиям и готово к использованию.

### **Вклад в достижение цели**

Мой вклад в реализацию проекта заключался в разработке серверной части приложения, которая является ключевым компонентом всей системы. Я выполняла следующие задачи:

1. **Проектирование базы данных:** Создание структурированной схемы для хранения данных пользователей и отчетов.
2. **Реализация API:** Разработка и тестирование API-эндпоинтов для работы с отчетами (GET, POST, DELETE) и предоставления функциональности для взаимодействия с клиентской частью.
3. **Интеграция аутентификации:** Настройка Google OAuth для авторизации пользователей, включая генерацию и проверку JWT-токенов.
4. **Настройка безопасности:** Обеспечение защиты данных пользователей через валидацию вводимой информации и ограничение доступа к защищенным маршрутам.
5. **Взаимодействие с командой:** Постоянная коммуникация со всеми участниками команды для согласования всех идей по проекту и работы над задачами.
6. **Тестирование серверной части:** Проведение функционального и нагрузочного тестирования для проверки стабильности и производительности системы.

Благодаря этим усилиям серверная часть стала надежным фундаментом приложения, обеспечивающим его корректную работу и взаимодействие с клиентской частью. Моя работа позволила обеспечить безопасность, производительность и масштабируемость приложения, что стало важным шагом на пути к успешной реализации проекта.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. СУБД PostgreSQL[Электронный ресурс]. – URL:/ https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/ (дата обращения: 15.11.2024).
2. Архитектура на Go [Электронный ресурс]. – URL: https://telegra.ph/Clean-Architecture-on-Golang-04-07 (дата обращения: 23.11.2024).
3. Проектирование базы данных Go [Электронный ресурс]. – URL: https://metanit.com/go/tutorial/10.3.php (дата обращения: 20.11.2024).
4. Разработка RESTful API [Электронный ресурс]. – URL: https://go.dev/doc/tutorial/web-service-gin (дата обращения: 23.11.2024).
5. JWT авторизация [Электронный ресурс]. – URL: https://ru.hexlet.io/courses/go-web-development/lessons/auth/theory\_unit (дата обращения: 10.12.2024).
6. Документация Go [Электронный ресурс]. – URL: https://go.dev/doc/

(дата обращения: 15.11.2024).

# **ПРИЛОЖЕНИЕ**

1. Наименование проекта: мобильное приложение для отслеживания достижения целей личностного развития
2. Цель (назначение): разработать мобильное приложение для Android, позволяющее пользователям создавать ежедневные отчеты о прогрессе, просматривать отчеты других пользователей, а также визуализировать свой прогресс по достижению цели
3. Сроки выполнения: 01.11.2024 – 20.12.2024
4. Исполнитель проекта (руководитель проекта): Алибеков Олег Олегович
5. Термины и сокращения:

«Приложение» — Мобильное приложение для платформы Android, разрабатываемое с использованием фреймворка Flutter.

«Отчет» — Запись о прогрессе пользователя, содержащая описание действий за день, оценку достижения цели по 10-балльной шкале и дату.

«График прогресса» — Визуализация прогресса пользователя в виде графика, отображающая динамику достижения целей.

«API» — (Application Programming Interface) Интерфейс программирования приложений, позволяющий взаимодействовать между клиентской и серверной частями приложения.

«Flutter» — Фреймворк для разработки кроссплатформенных мобильных приложений на языке Dart.

«RESTful API» — Архитектурный стиль взаимодействия клиент-сервер, основанный на протоколе HTTP.

* Технические требования:

| **Техническое требование** | **Язык разработки** | **Потребители** |
| --- | --- | --- |
| Платформа: Android, версия 9 и выше | Dart (Flutter). | Пользователи Android, заинтересованные в отслеживании прогресса своих целей |
| Архитектура: клиент-серверная с использованием RESTful API (Стандарты HTTP/1.1 для передачи данных, JSON для обмена данными.) | Dart (клиент), Go (сервер). | (Flutter и Backend)-разработчики |
| Создание отчетов: • Пользователь может публиковать один отчет в день для каждой цели. • Отчет включает: • Описание (текст, до 500 символов). • Оценку по 10-балльной шкале. • Дату (устанавливается автоматически). | Dart (Flutter для интерфейса), Go (сервер для обработки запросов). | Конечные пользователи |
| Просмотр отчетов: • Возможность просматривать: Свои отчеты и отчеты других пользователей в одной ленте (Отчет включает: • Описание (текст, до 500 символов). • Оценку по 10-балльной шкале. • Дату (устанавливается автоматически). | Dart (для отображения данных), Go (серверная логика). | Конечные пользователи |
| Визуализация прогресса (Линейный график (оценка, выставленная пользователем по достижению своей цели по дням), Использование библиотеки для графиков (fl\_chart).) | Dart | Конечные пользователи |
| Приложение должно загружаться не более чем за 3 секунды. | Dart (для клиента), Go (серверная оптимизация). | (Flutter и Backend)-разработчики, конечные пользователи. |
| Время отклика на действия пользователя — не более 1 секунды | Dart (для клиента), Go (серверная оптимизация) | (Flutter и Backend)-разработчики, конечные пользователи |
| Поддержка русского языка интерфейса (Использование пакетов Flutter (flutter\_localizations для мультиязычности). | Dart | Конечные пользователи |

* Содержание работы

| **Этапы работы** | **Сроки выполнения** | **Ответственный за этап** |
| --- | --- | --- |
| Сформировать требования к функциональности | 01.11.2024 – 05.11.2024 | Алибеков Олег Олегович |
| Описать архитектуру и технические характеристики | 05.11.2024 – 08.11.2024 | Алибеков Олег Олегович |
| Утвердить техническое задание | 09.11.2024 – 10.11.2024 | Алибеков Олег Олегович |
| Спроектировать структуру базы данных | 11.11.2024 – 13.11.2024 | Петрова Мария Валерьевна |
| Реализовать API для просмотра отчетов | 13.11.2024 – 24.11.2024 | Петрова Мария Валерьевна |
| Реализовать API для создания и удаления отчетов | 20.11.2024 – 05.12.2024 | Петрова Мария Валерьевна |
| Реализовать прототип экрана просмотра отчетов | 11.11.2024 - 17.11.2024 | Коновалова Кира Романовна |
| Реализовать прототип экрана создания отчетов | 18.11.2024 - 24.11.2024 | Коновалова Кира Романовна |
| Реализовать прототип экрана визуализации прогресса | 25.11.2024 - 01.12.2024 | Коновалова Кира Романовна |
| Реализовать UI просмотра отчетов пользователей | 11.11.2024 - 24.11.2024 | Гашимов Ильхам Фаррух оглы |
| Подключить API для просмотра отчетов пользователей | 25.11.2024 – 08.12.2024 | Гашимов Ильхам Фаррух оглы |
| Реализовать UI создания отчетов | 01.11.2024 - 24.11.2024 | Блинова Полина Вячеславовна |
| Подключить API для создания отчетов пользователей | 25.11.2024 - 08.12.2024 | Блинова Полина Вячеславовна |
| Реализовать UI визуализации прогресса | 11.11.2024 - 24.11.2024 | Сусликова Вероника Денисовна |
| Подключить API для визуализации прогресса | 25.11.2024 - 08.12.2024 | Сусликова Вероника Денисовна |
| Исправить найденные ошибки в ходе тестирования | 11.12.2024 - 13.12.2024 | Гашимов Ильхам Фаррух оглы |
| Подготовить презентацию проекта | 14.12.2024 - 14.12.2024 | Сусликова Вероника Денисовна |

* Основные результаты работы

| Результаты работы (наименование) | Формы представления |
| --- | --- |
| Техническое задание | Утвержденное техническое задание, содержащее цели проекта, требования к функциональности и этапы разработки |
| Бэкенд с API | Рабочий сервер с настроенной базой данных и API-эндпоинтами для взаимодействия с клиентской частью |
| UI/UX Дизайн и прототипы | Готовые дизайн-макеты и прототипы экранов приложения, включая интерфейсы для создания отчетов, просмотра целей и визуализации прогресса |
| Функциональные модули приложения | Модуль просмотра отчетов. Модуль создания отчетов. Модуль визуализации прогресса (графическая визуализация прогресса пользователя на основе данных по 10-балльной шкале) |
| Презентация проекта | Подготовленная презентация проекта, включающая описание целей и способы достижения целей |