1. **Обновленный план реализации с учетом новых данных**

Этот план включает:  
✅ **Учет анализа старой БД** (Шаг 1.1).  
✅ **Расширение структуры БД** с новыми таблицами.  
✅ **Интеграцию внешних API**.  
✅ **Пошаговый процесс выполнения**.

**📌 Этап 1. Улучшение структуры базы данных**

**🔹 Шаг 1.1 Анализ текущей структуры БД**

(✅ Этот шаг уже выполнен).

**🔹 Шаг 1.2 Внесение улучшений в схему БД**

* 1. **Добавляем таблицу users**
     + Разделяем пользователей и перевалы.
     + **Email** — уникальный (CONSTRAINT UNIQUE).
     + id — первичный ключ.
  2. **Оптимизируем pereval\_added**
     + Убираем raw\_data и images (JSON).
     + Добавляем отдельные поля (beautyTitle, title, other\_titles, connect, add\_time).
     + Добавляем status (модерация).
     + Добавляем route\_description (описание маршрута).
     + Добавляем hazards (опасности).
     + Добавляем related\_objects (связанные объекты).
  3. **Создаем таблицу coords**
     + В pereval\_added заменяем координаты на coord\_id.
  4. **Создаем таблицу pereval\_images**
     + Храним пути к изображениям, а не бинарные данные.
  5. **Создаем таблицу pereval\_gps\_tracks**
     + Храним ссылки на **GPX/KML треки** пользователей.
  6. **Создаем таблицу pereval\_history**
     + Храним историю прохождений перевала.
  7. **Создаем таблицу related\_objects**
     + Связываем перевалы с вершинами, хребтами.
  8. **Создаем таблицу weather\_info**
     + Храним погоду в районе перевала.

**🔹 Шаг 1.3 Развертывание новой схемы БД**

* 1. Создаем SQL-скрипты для новых таблиц.
  2. Разворачиваем базу в PostgreSQL.
  3. Тестируем базу (вставляем тестовые данные).

**📌 Этап 2. Разработка класса работы с БД**

**🔹 Шаг 2.1 Настройка подключения к БД**

* 1. Класс **Database** для работы с PostgreSQL.
  2. Читаем параметры из **.env** (FSTR\_DB\_HOST, FSTR\_DB\_PORT).

**🔹 Шаг 2.2 Методы работы с таблицами**

* 1. add\_pass() – добавление перевала.
  2. add\_images() – загрузка фото.
  3. add\_gps\_track() – загрузка треков.
  4. add\_history() – добавление прохождения.
  5. add\_related\_objects() – связывание перевала с другими объектами.
  6. get\_weather() – получение погоды.

**🔹 Шаг 2.3 Тестирование класса работы с БД**

* 1. Проверяем CRUD-операции для всех методов.
  2. Проверяем обработку ошибок.

**📌 Этап 3. Реализация REST API**

**🔹 Шаг 3.1 Создание эндпоинта POST /submitData**

* 1. Принимаем JSON-запрос.
  2. Проверяем обязательные поля.
  3. Записываем в БД.
  4. Возвращаем ID.

**🔹 Шаг 3.2 Реализация API для загрузки изображений**

* 1. POST /uploadImage/ – принимает файл.
  2. Сохраняет его в хранилище.
  3. Возвращает URL.

**🔹 Шаг 3.3 Реализация API для загрузки GPS-треков**

* 1. POST /uploadTrack/ – принимает GPX/KML файл.
  2. Сохраняет его в базе.
  3. Возвращает URL.

**🔹 Шаг 3.4 Интеграция внешних API**

* 1. **Google Maps API** – привязка перевалов к картам.
  2. **OpenWeather API** – получение погоды.
  3. **Strava/Wikiloc API** – интеграция треков.

**📌 Этап 4. Тестирование API**

**🔹 Шаг 4.1 Написание автоматических тестов**

* 1. Проверяем POST /submitData.
  2. Проверяем POST /uploadImage.
  3. Проверяем POST /uploadTrack.

**🔹 Шаг 4.2 Запуск тестов**

* 1. Используем **Pytest**.
  2. Проверяем **логирование ошибок**.

**📌 Этап 5. Работа с Git**

**🔹 Шаг 5.1 Создание ветки submitData**

* 1. git checkout -b submitData

**🔹 Шаг 5.2 Регулярные коммиты**

✅ init db structure – структура БД.  
✅ add db class – класс работы с БД.  
✅ implement POST submitData – API.  
✅ add image upload – загрузка фото.  
✅ write tests – тестирование API.  
✅ fix issues – исправления.

**🔹 Шаг 5.3 Завершение работы**

* 1. git merge submitData → master.
  2. git push origin master.

**📌 Итог**

Теперь наш план **учитывает все новые данные** и интеграцию API.

📍 **Следующий шаг:**  
**Этап 1. Шаг 1.2 – Внесение улучшений в схему БД**.

🔹 **Жду твоего подтверждения!** 🚀

Top of Form

Bottom of Form