**Модели, ORM и миграциии**

1. Создать пустой пакет **core** в папке **src**. В данном пакете создать пакет **models,** в нем создать файл **base.py**. задать настройки базовой модели **id**, названия полей, при необходимости задать ограничения для полей. В директориях сервисов создать пакет **models,** внутри пакета определить питоновские файлы для моделей и задать модели согласно методичке. В файле **\_\_init\_\_.py** прописать необходимые импорты.
2. Создать пользователя и базу данных **Postgresql** для реализации в приложении. Создать файл .env и прописать все необходимые переменные окружения( ***DB\_PORT, DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME***). Установить библиотеки **python-dotenv** и **psycopg2-binary**. Добавить файл **settings.py** и задать настройки конфигурации вашего проекта, при необходимости используя переменные окружения.
3. Установить **alembic** для миграций. Инициализировать алембик. Настроить все необходимые файлы для создания миграций **alembic.ini env.py.** Создать миграцию( обязательно проверяем созданную ревизию) и накатить миграцию. Проверить создание таблиц в базе согласно вашей схеме.
4. Создать файл **database.py** для реализации движка подключения к базе данных и создания активной сессии.
5. Создайте файл **schemas.py**. Реализуйте все необходимые схемы **pydantic** моделей для валидации данных вашей бд.
6. Реализуйте **CRUD** запросы для пользователей, используя файл **service.py** для написания orm запросов к вашей бд.

**Дополнительные задания**

1. Дополнительно реализуйте метод **patch**, для частичного обновления данных вашей модели.
2. Реализуйте мульти создание, обновление и удаление ваших пользователей, передавая массив изменённых или новых объектов при **post** и **put** запросе и массив с **id** при **delete** запросе.