**Разработка сервиса создания заданий/дел (TODO сервис)**

**Цель:**

Познакомиться с backend разработкой на Python: изучить Fast-API, схемы данных и валидацию при помощи Pydantic, поработать с SQL базой.

**Постановка задачи:**

Необходимо разработать API для TODO сервиса *(сервис для создания, отслеживание и выполнения задач, см.* [*todolist*](https://todoist.com/ru) *и другие).*

В сервисе должен быть реализован следующий функционал:

* Регистрация нового пользователя *(по логину и паролю)*.
* Авторизация пользователя *(по его логину и паролю)*.
* CRUD *(создание, чтение, обновление, удаление)* для заметок.
* CRUD *(создание, чтение, обновление, удаление)* для группы заметок. Группа заметок – сущность, позволяющая объединять множество заметок в группу.

Сам сервис должен быть разработан при помощи веб фреймворка FastAPI. Для схем данных должна быть использована входящая в стандартную поставку библиотека Pydantic. Сервис должен хранить всю свою информацию в любой реляционной базе данных *(например, MySQL или PostgreSQL).* Взаимодействие с базой данных может быть любым: голые SQL запросы или использование какой-либо ORM *(например, SQLAlchemy).* Аналогично подойти к выбору процесса миграций в базе данных: голые SQL скрипты или миграции при помощи специальных библиотек *(например, alembic).*

**Примерный план выполнения работы:**

* Изучить необходимый стек технологий/фреймворков/понятий *(FastAPI, Pydantic, базы данных, SQL vs ORM, миграции, авторизация и аутентификация, CRUD, REST-API и т.д.)*.
* Продумать объектную модель сервиса, нарисовать её ER-диаграмму, согласовать её.
* Исходя из ER-диаграммы спроектировать таблицы в базе данных, согласовать их.
* Научиться работать с выбранной СУБД из python.
* Продумать какие необходимы REST API эндпоинты для реализации сервиса, согласовать их.
* Научиться работать с FastAPI и Pydantic.
* Для согласованных эндпоинтов сделать входящие и исходящие Pydantic схемы.
* Для согласованных эндпоинтов реализовать методы-заглушки при помощи FastAPI, показать полученный результат.
* Добавить в методы-заглушки необходимую бизнес-логику по работе с базой данных.
* Проверить работоспособность приложения.

**Будет круто, если:**

* Проект будет написан на самом свежем Python и будет использовать последние версии библиотек.
* Код будет разделён на логические модули *(модуль для работы с бд, модуль с API-шкой, модуль с бизнес логикой и т.д.).*
* Код будет полностью типизирован.
* Доступы к базе данных находятся не в коде, а в конфигурационном файле или переменных окружения.
* В базе данных будут храниться не сами пароли, а только их хеш.
* Проект будет на poetry *(т.е. в корне будет pyproject.toml вместо requirements.txt)*
* В проекте будут присутствовать unit-тесты.
* Присутствует README.md проекта *(о чём проект, как развернуть локально, какие есть методы, как пользоваться и т.д.).*
* Автогенерируемая pydantic swagger-схема будет дополнена пояснениями, примерами и подсказками *(что делает её более читаемой и понятной)*.
* При любых input/output операциях будет использован ассинхронный код.

**Литература:**

* [**Диджитализируй**](https://m.youtube.com/channel/UC9MK8SybZcrHR3CUV4NMy2g?ysclid=lj3timttgp670613092) **– канал в целом о Python и всё что вокруг него.**
* [**Уроки FastAPI**](https://www.youtube.com/playlist?list=PLaED5GKTiQG8GW5Rv2hf3tRS-d9t9liUt) **– короткий курс, рассказывающий о возможностях фреймворка.**
* [**FastAPI курс**](https://www.youtube.com/playlist?list=PLeLN0qH0-mCVQKZ8-W1LhxDcVlWtTALCS) **– более подробный курс об освоении фреймворка с нуля.**
* [**Руководство по веб фреймворку FastAPI**](https://metanit.com/python/fastapi/?ysclid=lj3tjnr9od813184611) **– серия статей о фреймворке.**

Дополнительную необходимую литературу найти и освоить самостоятельно. В качестве источников советую: статьи на хабре, статьи на metanit, Youtube уроки, официальную документацию фрейморков/библиотек. Также классным учителем и наставником может стать ChatGPT (при правильном его использовании).