Тестовое задание Python Developer.

> Важно: Все задачи должны быть выполнены с использованием Python 3.11. Типизация и docstrings обязательны для всех функций и методов. Коммиты должны быть четко структурированы и иметь понятные сообщения.

Задание состоит из трех подзадач, после выполнения которых вам потребуется объединить их в один общий проект.

Задача 1: Создание API с использованием Flask и SQLAlchemy

1. Создайте простой API с использованием Flask, который будет работать с базой данных (SQLite) с помощью SQLAlchemy.
2. В базе данных должна быть таблица `User` с полями `id`, `username`, `email` и `registration_date`.
3. Реализуйте CRUD-операции для пользователей: создание, чтение, обновление и удаление.
4. Добавьте поддержку пагинации для получения списка пользователей.

Задача 2: Работа с данными и структурирование

1. Используя данные из таблицы `User`, напишите функцию, которая подсчитывает количество пользователей, зарегистрированных за последние 7 дней.
2. Напишите функцию, которая возвращает топ-5 пользователей с самыми длинными именами.
3. Создайте функцию, которая определяет, какая доля пользователей имеет адрес электронной почты, зарегистрированный в определенном домене (например, "example.com").

Задача 3: Модели анализа данных

1. Создайте модель, которая будет предсказывать, с какой вероятностью пользователь будет активным на сайте в течение следующего месяца, основываясь на его истории взаимодействия с сайтом. Вы можете использовать любые данные, которые считаете необходимыми.
2. Напишите функцию, которая будет использовать эту модель для расчета вероятности активности пользователя в будущем месяце.
3. Включите эту информацию в ответ API при запросе информации о пользователе.

Задача 4: Объединение проекта

1. Объедините все предыдущие задачи в один проект с использованием Flask.
2. Создайте API-метод, который будет возвращать данные из задачи 2 (количество пользователей за последние 7 дней, топ-5 пользователей с самыми длинными именами, доля пользователей с определенным доменом электронной почты).
3. Добавьте к API-методу из задачи 1 возможность получения информации о вероятности активности пользователя в будущем месяце (из задачи 3).

Задача 5: Тестирование и документация

1. Напишите юнит-тесты для проверки корректности работы всех функций и API-методов, которые были реализованы в рамках этого тестового задания.
2. Убедитесь, что ваш код соответствует стандартам PEP8 и использует типизацию.
3. Создайте документацию для вашего API с использованием, например, Swagger или другого инструмента документирования, удобного для вас.

Оценка выполнения:

Тестовое задание состоит из нескольких задач и подзадач. Вы можете выбрать, какие из них выполнить, но учтите, что выполнение всех задач повысит ваши шансы на успех. Каждая задача и подзадача имеют свою оценку в баллах. Ниже представлены возможные диапазоны баллов и соответствующие статусы:

- "Не прошел": 0-20 баллов
- "Почти прошел": 21-40 баллов
- "Прошел": 41-60 баллов
- "Отлично": 61-80 баллов
- "Идеально": 81-100 баллов

Баллы за задачи:

Задача 1: Создание API с использованием Flask и SQLAlchemy – 30 баллов

- Создание таблицы `User` – 5 баллов
- Реализация CRUD-операций – 15 баллов
- Поддержка пагинации – 10 баллов

Задача 2: Работа с данными и структурирование – 25 баллов

- Подсчет пользователей за последние 7 дней – 8 баллов
- Топ-5 пользователей с самыми длинными именами – 8 баллов
- Доля пользователей с определенным доменом электронной почты – 9 баллов

Задача 3: Модели анализа данных – 25 баллов

- Создание модели предсказания активности пользователя – 15 баллов
- Функция для расчета вероятности активности – 10 баллов

Задача 4: Объединение проекта – 10 баллов

- Интеграция задач в одном проекте – 10 баллов

Задача 5: Тестирование и документация – 10 баллов

- Юнит-тесты – 5 баллов
- Документация – 5 баллов

Основные требования:

1. Вы можете использовать стандартный пакетный менеджер pip или выбрать poetry для управления зависимостями проекта. Если вы используете poetry, убедитесь, что ваш проект корректно настроен для работы с ним.

2. Реализуйте понятную и простую файловую структуру проекта. Организуйте код таким образом, чтобы он был легко читаемым и расширяемым.
3. Добавьте README-файл, который будет содержать информацию о проекте, его целях, основных функциях, а также инструкции по установке и использованию. Убедитесь, что README содержит все необходимые сведения, чтобы другие разработчики могли легко разобраться в вашем проекте.
4. Соблюдайте понятную и последовательную коммит-историю. Каждый коммит должен иметь четкое и информативное сообщение, отражающее суть внесенных изменений.
5. Если у вас возникнут вопросы или необходима помощь, не стесняйтесь обращаться к Егору в Telegram. Он является контактным лицом и с радостью поможет вам разобраться с заданием.

Выполняйте задачи и подзадачи в соответствии с вашими навыками и умениями, чтобы продемонстрировать свои компетенции. Удачи!