**Техническое задание: Первая часть разработки системы управления задачами**

**Цель проекта:** Создать Backend для упрощенной системы управления задачами, которая позволит пользователям создавать и управлять проектами и задачами, назначать исполнителей и отслеживать статус выполнения.

**Сроки выполнения: 05.11.2024 – 25.11.2024**

**Стек технологий:**

* Язык программирования: **Python 3.x**
* Фреймворк: **Django Rest Framework** или **FastAPI**
* База данных: **PostgreSQL**
* Аутентификация: **JWT (JSON Web Tokens)**
* Контейнеризация (опционально): **Docker**
* Realtime сообщение клиента и сервера: **Websocket**

**Функциональные требования**

**1. Пользователи и аутентификация**

1. **Регистрация пользователей:**
   1. Возможность создания аккаунта с уникальным именем пользователя и паролем.
   2. Пользователь на сайте должен иметь следующие параметры:
      * Имя пользователя
      * Фамилия пользователя
      * Аватаре пользователя (если не задан то, стандартный)
      * Роль на платформе
      * Текущие проекты
      * \*Дополнительные поля при необходимости
2. **Аутентификация:**
   1. Реализация входа в систему и проверку авторизации с использованием JWT.
3. **Профиль пользователя:**
   1. Возможность просмотра информации на странице “профиль” о:
      * Имени пользователя на платформе
      * Фамилии пользователя на платформе
      * Аватаре пользователя
      * Роль в текущих проектах, на которых задействован пользователь
      * Истории проектов (проекты на которых пользователь был когда-либо задействован)

**2. Проекты**

1. **CRUD операции для проектов:**
   1. Возможность создания, чтение, обновление и удаление проектов.
   2. Участники проекта**:**
      * Возможность добавления и удаления пользователей из проекта.
      * Назначение ролей внутри проекта.
   3. Поля проекта**:**
      * Название
      * Описание
      * Дата создания
      * Дата обновления
      * Статус (Активен, Архивирован)
2. **Фильтрация и сортировка задач:**
   1. Возможность сортировки проектов по:
      * Времени создания\обновления (от старых к новым и от новых к старым)
      * По названию (от а до я)
   2. Возможность фильтрации задач:
      * От и до (созданные/обновленные/сроки выполнения от 00.00.2000 до 00.00.2001)

**3. Задачи**

1. **CRUD операции для задач:**
2. Создание, чтение, обновление и удаление задач.
   1. **Поля задачи:**
      * Название
      * Описание
      * Проект (ссылка на проект)
      * Исполнитель (ссылка на пользователя)
      * Статус (Grooming, In Progress, Dev, Done)
      * Приоритет (Низкий, Средний, Высокий)
      * Дата создания
      * Дата обновления
      * Срок выполнения
      * Ответственный за тестировку
3. **Комментарии к задачам:**
   1. Добавление, редактирование и удаление комментариев.
   2. Просмотр истории комментариев.
4. **Фильтрация и сортировка задач:**
   1. Возможность сортировки задач по:
      * Статусу
      * Приоритету
      * Исполнителю
   2. Возможность фильтрации задач:
      * От и до (созданные/обновленные/сроки выполнения от 00.00.2000 до 00.00.2001)
      * По дате создания/обновления (от старых к новым и от новых к старым)
      * По названию (от а до я)

**4. Развертывание**

1. **Docker:**
   1. Развернуть локальное приложение с использованием технологии Docker. Создание Dockerfile и docker-compose.yml для приложения и базы данных.
2. Инструкция по развертыванию:
   1. Подробное руководство по запуску приложения.

**5. Уведомления**

1. **Email уведомления:**
   1. Отправка уведомлений при:
      * Назначении вас в новый проект
2. **Websocket уведомления на сайте:**
   1. Отправка уведомлений при:
      * Назначении вас как ответственного на задачу
      * Назначении вас в новый проект
      * Изменении статуса в вашей задаче
      * Добавлении комментария к вашей задаче

**Требования к API**

1. **Структура эндпоинтов:**
   1. Должны соответствовать RESTful архитектуре.
2. **Стандартизация ответов:**
   1. Единый формат ответов API (успешных и ошибочных).
3. **Версионирование API:**
   1. Включение версии в URL (например, /api/v1/).

**Ожидаемые результаты**

1. **Исходный код:**
   1. Размещен в репозитории (GitHub, GitLab) с понятной структурой коммитов.
2. **Документация:**
   1. README.md с описанием проекта и инструкцией по запуску.
   2. Документированное API (Swagger, OpenAPI).
3. **Тесты:**
   1. Покрытие тестами основных модулей и функций.
4. **Демонстрация:**
   1. Работающее приложение, готовое к запуску и проверке. Имеется полноценная postman коллекция для тестированя эндпоинтов во время презентации

**Рекомендации для стажеров**

1. **Планирование:**
   1. Перед началом разработки составить план работ и определить приоритеты.
2. **Обратная связь:**
   1. Регулярно сообщать о прогрессе и возникающих проблемах.
3. **Качество:**
   1. Уделять внимание чистоте кода и оптимизации.
4. **Требования к визуальному оформлению(Опционально)**
5. **Реализация frontend части проекта:**
   1. Реализация интерфейса приложения на основе технологий Ninja2/Django Templates
6. **UI/UX интерфейс:**
   1. Продумать флоу и пути обеспечения удобства пользования приложения.

**Примечание:**

Студентам приветствуется реализация дополнительного функционала, не описанного в техническом задании