Техническое задание: Разработка бэкенд части приложения

Описание приложения:

Приложение обеспечивает взаимодействие пользователей с сервером, предоставляя доступ к данным и функциональности через API.

Основная цель — создание высокопроизводительного, надежного и масштабируемого бэкенда, обеспечивающего обработку запросов, управление данными и взаимодействие с фронтендом.

Задача в проекте:

Разработка бэкенд части приложения на JavaScript с использованием современных технологий и фреймворков для обеспечения надежного взаимодействия с базой данных, реализации бизнес-логики, а также кроссплатформенной и масштабируемой работы API.

Дизайн платформы:
https://www.figma.com/design/uK0DTgBMSQ3aJKCsdhDsHW/Performance-%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0-%D0%A2%D0%9C438?t=uPqGW35qFYzEHJoB-0

Технологический стек:

- ***Node.js (ES6+)*** *– серверная платформа для написания бэкенд-логики, обеспечения высокопроизводительных асинхронных операций.*
- ***Express.js/Nest.js*** *– фреймворки для организации REST API, маршрутизации и обработки запросов.*
- ***MongoDB/PostgreSQL/MySQL*** *– база данных для хранения и управления данными приложения.*
- ***Mongoose/Sequelize/TypeORM*** *– ORM для взаимодействия с базой данных.*
- ***JWT/OAuth 2.0*** *– для реализации системы аутентификации и авторизации пользователей.*
- ***Redis*** *– кэширование для ускорения обработки повторяющихся запросов и уменьшения нагрузки на базу данных.*
- ***Docker*** *– контейнеризация для упрощения развертывания и масштабирования приложения.*
- ***Git*** *– для управления версиями и совместной разработки.*
- ***Mocha/Chai/Jest*** *– для тестирования API и функционала приложения.*

Логика реализации:

- Настройка базового окружения с использованием Node.js и Express.js/Nest.js.
- Подключение выбранной базы данных (MongoDB/PostgreSQL/MySQL) и настройка

ORM для работы с данными.

- Определение REST API для основных операций: создание, чтение, обновление и удаление данных (CRUD).
- Реализация маршрутов для обработки запросов от фронтенда (POST, GET, PUT, DELETE).
- Организация работы с сессиями и аутентификацией (JWT или OAuth 2.0).
- Реализация middleware для обработки ошибок и логгирования.
- Создание моделей данных с использованием Mongoose/Sequelize/TypeORM.
- Обеспечение взаимосвязей между сущностями (например, связь один ко многим, многие ко многим).
- Валидация данных перед их сохранением в базу.
- Реализация API для взаимодействия с фронтенд частью приложения (Axios/Fetch запросы).
- Обработка ошибок и отправка корректных ответов фронтенду.
- Реализация систем уведомлений и логики обратной связи на основе полученных данных.
- Настройка кэширования с использованием Redis для уменьшения времени отклика на повторяющиеся запросы.
- Реализация защиты от избыточных запросов (rate limiting), а также системы логгирования.
- Настройка Docker для контейнеризации и развертывания на сервере.

Тестирование:

- Написание тестов для API с использованием Mocha/Chai/Jest для проверки корректности работы функционала.
- Проведение нагрузочного тестирования для оценки производительности и масштабируемости.
- Тестирование безопасности, включая проверку уязвимостей в аутентификации и обработке данных.
- Подготовка API-документации с использованием Swagger/OpenAPI для облегчения интеграции фронтенда.
- Настройка автоматической сборки и деплоя проекта через CI/CD (например, с использованием Jenkins, GitLab CI).

Дедлайн: *25.10, 19:00*

Ежедневно предоставлять отчет о проделанной работе для согласования прогресса и внесения возможных изменений в план проекта.