Техническое задание (микросервисная структура по управлению задачами) – ООО «СистемаКонтроля» по работе с строительными объектами

Цель разработки — собрать рабочий бэкенд с тремя компонентами, привести микросервисы и API-шлюз к единому стандарту качества кода, повысить надежность, наблюдаемость и безопасность (см. пункты Надежность, Наблюдаемость, Безопасность), помните что все действия должны выполняться на удаленном рабочем сервере, подготовьте спецификации API и тестовое покрытие для последующей доработки.

Система предназначена для:

- инженеров (регистрация дефектов, обновление информации);
- менеджеров (назначение задач, контроль сроков, формирование отчётов);
- руководителей и заказчиков (просмотр прогресса и отчётности).

Документ описывает техническое задание по доработке серверной части. Цель работы состоит в унификации архитектуры и качества кода, повышении надежности и наблюдаемости, а также в упрощении запуска и развёртывания. Планируется оформление спецификаций API и тестового покрытия.

Анализ

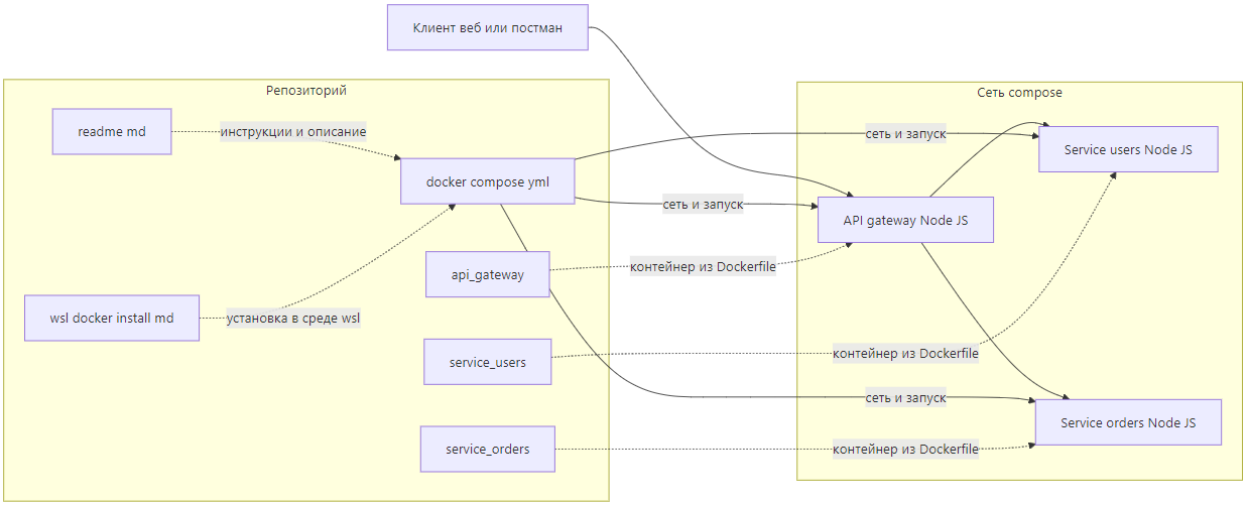

Рис. 1 Схема проекта

В составе решения присутствуют три компонента на Node.js. В корне проекта лежит файл docker compose для оркестрации. Для каждого сервиса и шлюза подготовлены Dockerfile и описания зависимостей. В репозитории имеются руководства по установке в среде WSL и по настройке Docker. Подробности исходного кода уточняются после предоставления доступа.

Ссылка на задание (gitverse).

Работы касаются только серверной части. Клиентские приложения остаются за пределами данного документа. Инфраструктура ограничивается контейнерами и базовым CI.

Обзор текущего состояния

Система предоставляет регистрацию и вход пользователя, управление профилем, операции с заказами, а также единый вход через API-шлюз. Все действия записываются в логи и сопровождаются трассировкой. Решение

готовится к трём окружениям разработки, тестирования и промышленной эксплуатации.

Решение включает в себя:

- Три узла решения представлены как минимум тремя директориями api_gateway, service_users, service_orders;
- Технологический стек по метаданным JavaScript и Dockerfile;
- Оркестрация предполагается через docker-compose.yml;
- Документация и инструкции установки WSL и Docker присутствуют, но содержимое недоступно без входа;
- Подтверждение по навигации и метаданным на GitVerse.

План работы над проектом

Вам необходимо собрать рабочий бэкенд с тремя компонентами:

- Шлюз
- Сервис пользователей
- Сервис заказов

Обеспечить регистрацию и вход, работу с профилем, полный жизненный цикл заказа, единый вход через шлюз, логирование, трассировку и готовность к трём средам разработка тест прод

Объём работ

- Реализовать маршруты и проверку прав в сервисах
- Настроить проксирование и защиту в шлюзе задание со звёздочкой*
- Включить логи и трассировки во всех компонентах задание со звёздочкой*
- Подготовить окружения для трёх профилей swagger или postmat
- Описать интерфейсы в OpenAPI и приложить файл спецификации swagger или postmat

Требования к функционалу **(REST-запросы)**

Шлюз

- Прокидывает пути users и orders на соответствующие сервисы
- Проверяет JWT на защищённых путях
- Ограничивает частоту запросов и сглаживает всплески
- Корректно обрабатывает CORS и заголовок X Request ID с прокидыванием вглубь

Сервис пользователей

- Регистрация нового пользователя с валидацией входных полей
- Вход и выдача JWT
- Получение текущего профиля и обновление профиля
- Список пользователей для роли администратор с пагинацией и фильтрами

Сервис заказов

- Создание заказа с проверкой входных данных и существования пользователя
- Получение заказа по идентификатору
- Список заказов текущего пользователя с пагинацией и сортировками
- Обновление статуса и отмена заказа
- Проверка прав на уровне сервиса для каждой операции

Доменные события

- Публикация события создан заказ
- Публикация события обновлён статус
- Заготовка для подключения брокера сообщений в следующих итерациях

Формат интерфейсов

- Ответы в JSON с полями: success и data error
- Единая ошибка с полями: code и message
- Версионирование путей через префикс v1
- Авторизация через заголовок Authorization со схемой Bearer и токеном JWT
- На свободном доступе только регистрация и вход

Требования к данным

Пользователь	Заказ
• Идентификатор UUID	• Идентификатор UUID

• Электронная почта • Хэш пароля • Имя • Роли массив строк • Дата создания • Дата обновления	• Идентификатор пользователя UUID • Состав позиции товар количество • Статус создан в работе выполнен отменён • Итоговая сумма • Дата создания • Дата обновления

Что сдать студенту (git)

Артефакты

– Исходный код трёх компонентов в отдельных папках

– Файл спецификации OpenAPI в директории docs

Чек лист самопроверки

- Регистрация создаёт пользователя и возвращает успешный ответ
- Вход выдаёт JWT и последующие защищённые вызовы проходят
- Профиль читается обновляется с сохранением
- Заказ создаётся отображается меняет статус удаляется если разрешено
- Пользователь без прав не видит админские функции
- Логи видны и содержат идентификатор запроса
- Трассы собираются и показывают связку шлюз пользовательский сервис сервис заказов
- Запуск в трёх профилях работает с разными переменными окружения

Тест кейсы минимальный набор (unit или хотя бы POSTMAN)

Пользователи (на оценку 3)

- Успешная регистрация с валидными полями ожидаем статус успех и созданный идентификатор
- Повторная регистрация с той же почтой ожидаем контролируемую ошибку
- Вход с правильными данными ожидаем выдачу токена
- Доступ к защищённому пути без токена ожидаем отказ

Заказы (+ Пользователи – оценка 4)

- Создание заказа для авторизованного пользователя ожидаем успех и статус создан
- Получение своего заказа ожидаем успех
- Список своих заказов с пагинацией ожидаем корректные поля и навигацию по страницам

(+ Пользователи + Заказы – оценка 5)

- Попытка обновить чужой заказ ожидаем отказ
- Отмена собственного заказа ожидаем статус отменён и отсутствие побочных эффектов

Подсказки по реализации

– Для JWT удобно взять библиотеку jsonwebtoken
– Для логов подойдёт pino
– Для валидации данных удобны zod или joi
– Для ограничения частоты запросов подойдёт express rate limit

Вы можете взять проект и переписать его на Python, C# и скидывать по своему стеку.