команда undefined

прототип сайта цодд смоленск

ульянов роман
павлов даниил
золотарёв иван
гёзалов али
рябова дарья

о команде

золотарёв иван
–
фронтенд – разработчик

ульянов роман
–
фронтенд – разработчик

павлов даниил
–
тимлид, бэкенд-разработчик

рябова дарья
–
ux/ui дизайн

гёзалов али
–
ux/ui дизайн

проблематика

1. Недостаточная информированность граждан и отсутствие эффективной коммуникации.

2. Разрозненность данных и отсутствие инструментов для их анализа.

архитектура решения

1. Задача, которую мы решаем
2. Ключевая идея
3. Используемые технологии
4. Инновационные аспекты
5. Модульность и структура данных
6. Развитие продукта

Задача, которую решаем

- Обеспечить гражданам удобный доступ к информации и онлайн-услугам ЦОДД.
- Визуализировать данные (графики, карты, фото до/после) и показатели работы.
- Обеспечить масштабируемость и надежность за счет микросервисной архитектуры.

ключевая идея

- Разделение системы на независимые микросервисы: frontend, backend, database.
- Единый REST API для взаимодействия фронтенда и бекенда, автогенерация OpenAPI.
- Контейнеризация и изоляция сервисов для быстрого развертывания и масштабирования.

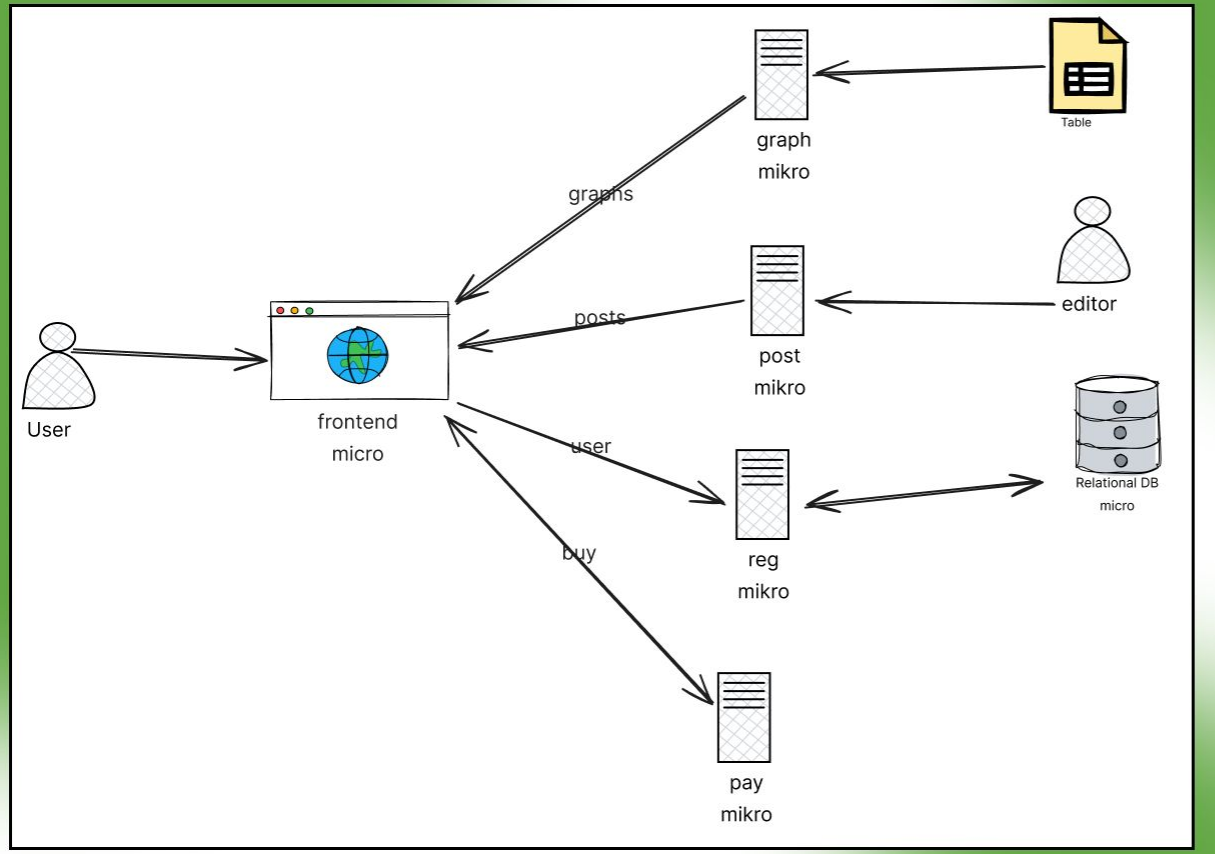
используемые технологии

- Frontend: React 19, Vite, Tailwind CSS v4, React Router.
- Backend: Python 3.13, FastAPI, SQLAlchemy, Uvicorn.
- Database: PostgreSQL (реляционная модель), связи users/roles/orders/services.
- Инфраструктура: Docker/Compose, Nginx (проксирование/HTTPS).

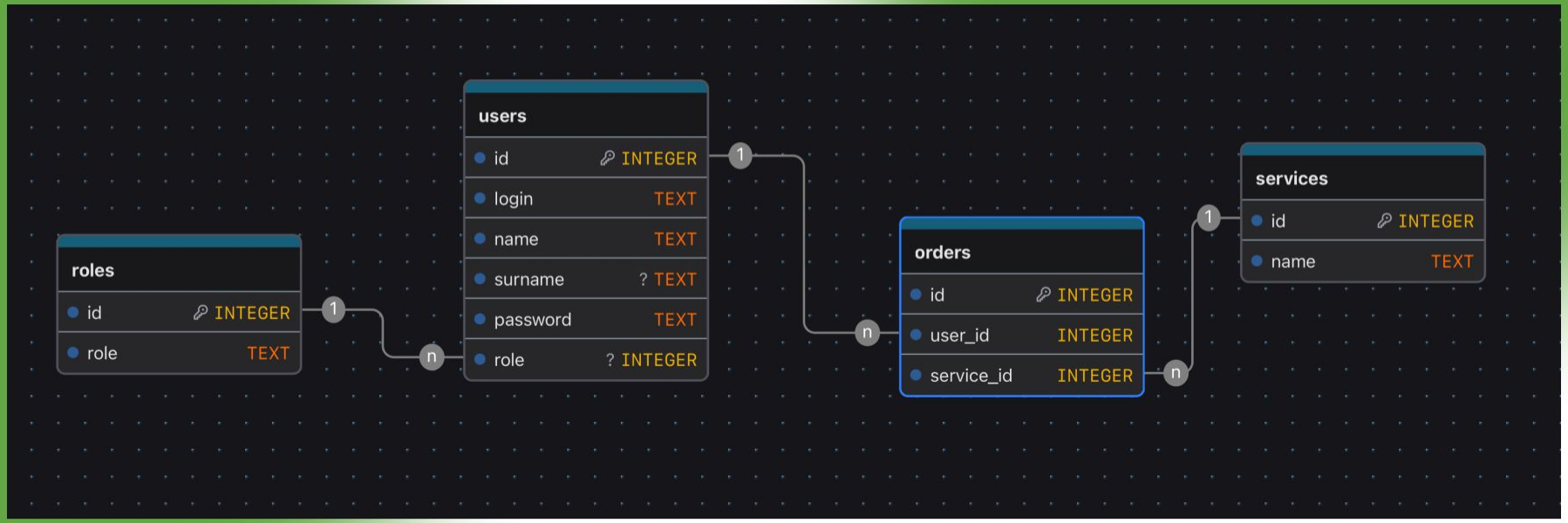
инновационные аспекты

- Адаптивный дизайн (mobile-first), высокая производительность (Vite, оптимизация ассетов).
- Data-driven дашборды и аналитика, в том числе импорт данных из XLSX и транзакций.
- Безопасность: аутентификация по JWT, разграничение доступа.
- Масштабирование по сервисам, готовность к CI/CD и кэшированию.

модульность и структура данных



- **service frontend** — отвечает за UX/UI и служит связкой между пользователем и сервером.
- **service graph** — парсит данные из таблиц и строит графики на их основе.
- **service post** — используется администраторами для редактирования данных и создания постов.
- **service users** — управление пользователями: регистрация, удаление, обновление, просмотр и т. д.
- **billing account service** — обрабатывает платежи пользователей, предоставляет и тарифицирует услуги.



Сущности и их назначение:

- roles (Роли): Список доступных ролей (например, Администратор, Клиент).
- users (Пользователи): Хранит основные данные пользователей, каждый из которых может иметь одну роль (связь n:1 с roles).
- services (Сервисы): Каталог услуг, которые можно заказать.
- orders (Заказы): Связующая таблица для фиксации того, какой пользователь заказал какой сервис (связи 1:n с users и services).

Ключевые принципы:

- Все связи используют внешние ключи (user_id, service_id, role) для обеспечения ссылочной целостности.
- Таблица orders реализует отношение "многие ко многим" между пользователями и сервисами.

развитие продукта

UX и основа

- **WYSIWYG Редактор: Интеграция визуального редактора в React Frontend для форматирования новостей.**
- **Менеджер Медиа: Централизованное хранилище изображений, связанное с Posts Service.**
- **Оптимизация Чтения: Настройка Redis-кэширования (через Docker/Compose) для ускорения загрузки и снижения нагрузки на PostgreSQL**

функционал и автоматизация

- Теги и Категории: Система тегирования для удобной навигации и фильтрации.
- Ролевой Доступ: Разграничение прав пользователей (Админ, Редактор, Гость) на уровне FastAPI и SQLAlchemy.
- Планировщик: Функции черновиков и автоматической публикации по расписанию через Posts Service.

аналитика и интеграции

- Аналитический Дашборд: Сводная статистика (топ-новости, графики просмотров) для принятия решений.
- API Интеграция: Подключение к внешним данным (трафик/ДТП) для автоматизации новостей.
- Глубокий Анализ: Интеграция с внешними системами (Метрика) для отслеживания поведения и источников трафика.

скринкаст