

Веб-приложение для записи на проекты

Результаты первой итерации

План итерации 1:

- **Бэкенд:**
 - Инициализация проекта NestJS, настройка TypeScript, линтера и форматтера.
 - Разработка OpenAPI спецификации для основных сущностей (курсы, проекты, пользователи).
 - Реализация мок-данных и заглушек API для параллельной разработки фронтенда.
- **Фронтенд:**
 - Настройка роутинга и базового компонента с проверкой работоспособности.
 - Настройка линтера и форматера.
 - Интеграция с мок-бэкендом и отображение данных

План итерации 1:

- **База данных:**
 - Проектирование ER-диаграммы и создание SQL-скриптов инициализации.
- **DevOps:**
 - Подготовка Docker-окружения для разработки (docker-compose с MySQL, бэкендом и фронтендом на nginx).
 - Настройка .env.example и инструкций по запуску.
- **Документация:**
 - Составление use-case диаграммы и сценариев использования.
 - Создание инструкций для GitHub Copilot.
 - Обновление корневого README.md с инструкцией по запуску без Docker.

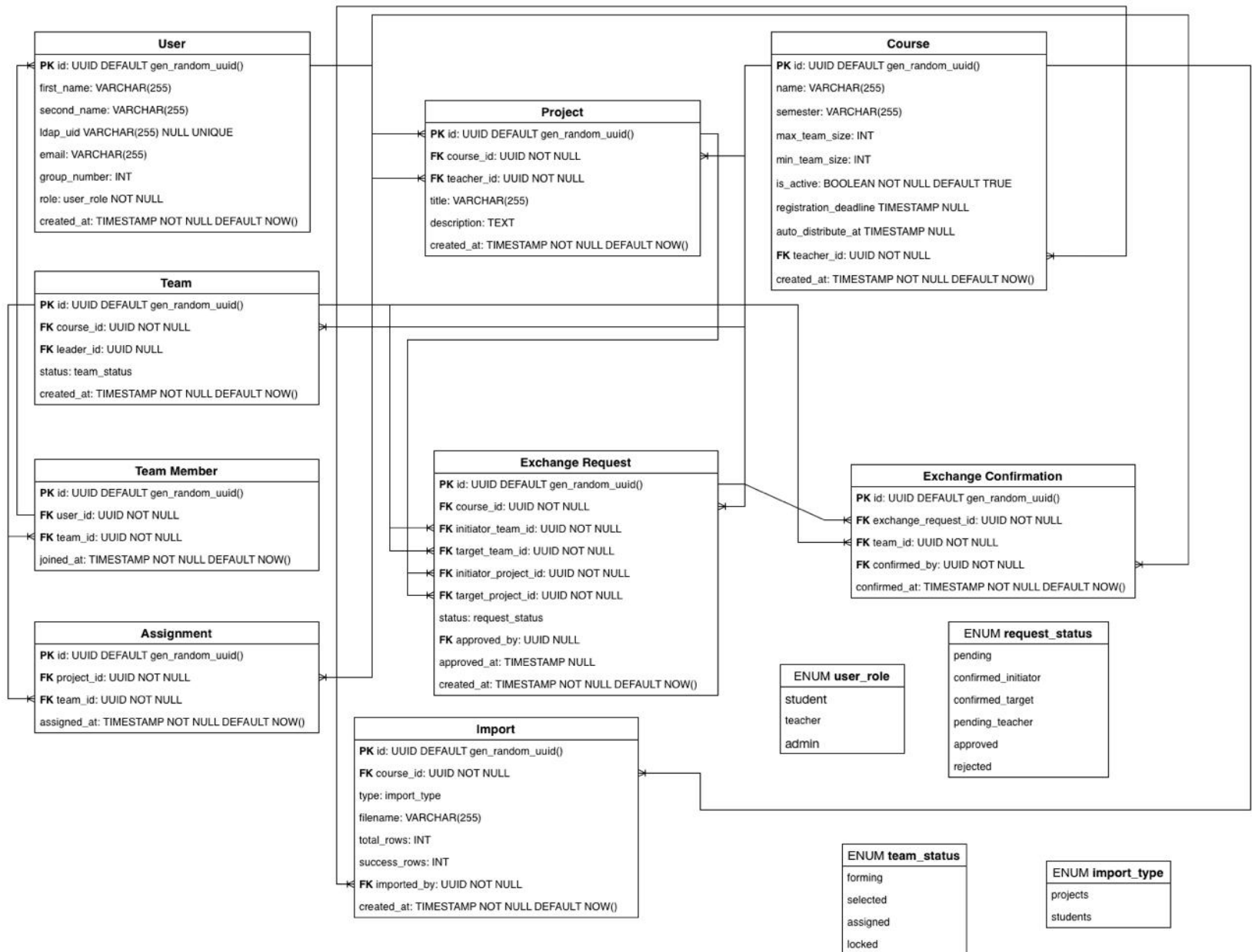
Результаты итерации 1

- **Бэкенд:**
 - Создан проект на NestJS, настроен TypeScript, ESLint, Prettier.
 - Разработана OpenAPI спецификация (файл backend/openapi.yaml) с описанием сущностей и эндпоинтов авторизации, курсов, проектов. [backend/openapi.yaml](#)
 - Подготовлены мок-данные для пользователей, курсов, проектов; настроены заглушки для всех запланированных эндпоинтов.
- **База данных:**
 - Подготовлены SQL-скрипты для создания таблиц с индексами и внешними ключами.
 - Разработана ER-диаграмма (сущности: пользователи, курсы, проекты, назначения, запросы на обмен).

```

    erDiagram
        Project ||--o{ Course : "has"
        Project ||--o{ Exchange_Request : "has"
        Exchange_Request ||--o{ Exchange_Confirmation : "has"
        Exchange_Confirmation ||--o{ Import : "has"
        Exchange_Confirmation ||--o{ User_Role : "has"
        Exchange_Confirmation ||--o{ Request_Status : "has"
        Exchange_Confirmation ||--o{ Team_Status : "has"
        Exchange_Confirmation ||--o{ Import_Type : "has"

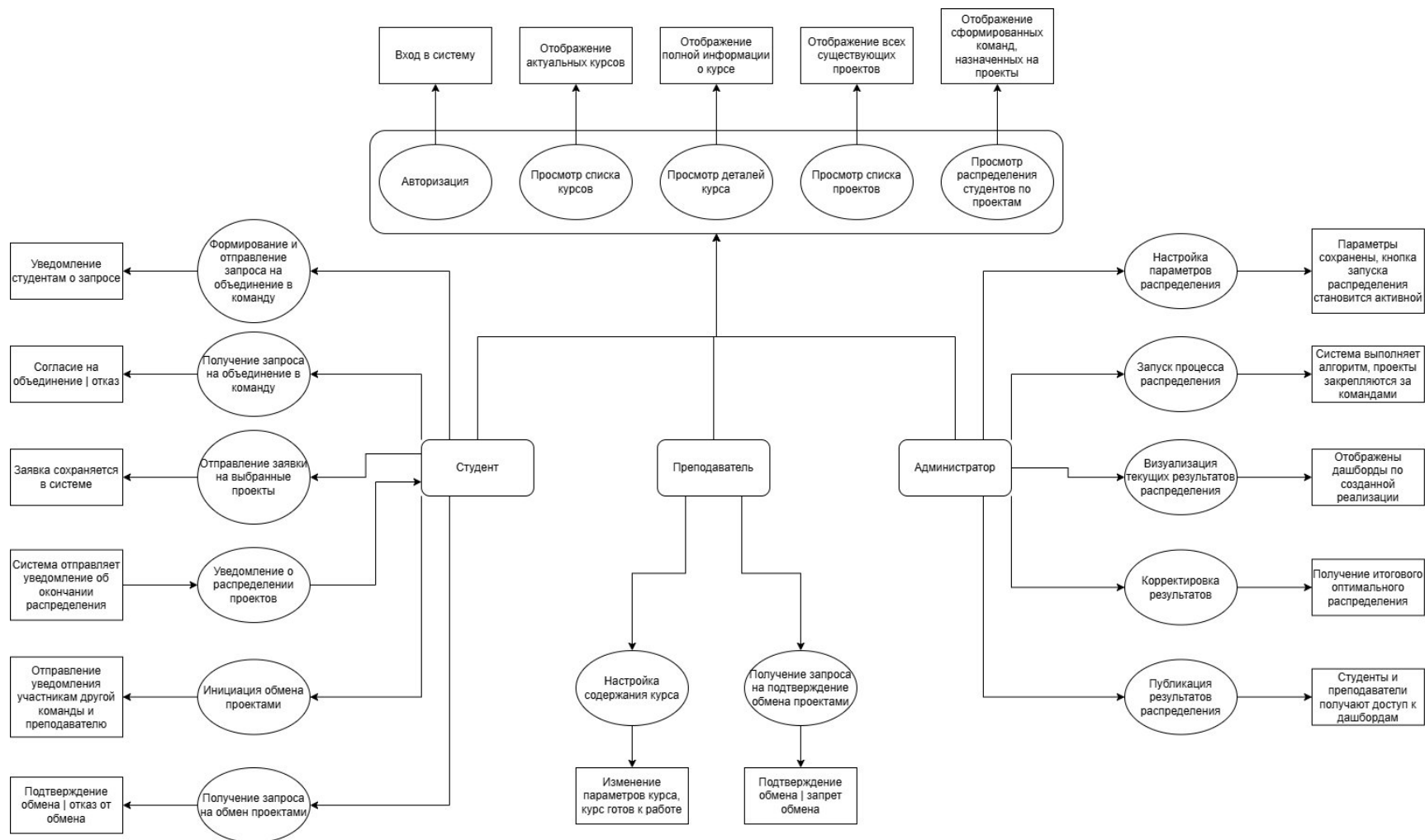
        Project {
            string PK_id PK
            string course_id FK
            string teacher_id FK
            string title
            string description
            timestamp created_at
        }
        Course {
            string PK_id PK
            string name
            string semester
            int max_team_size
            int min_team_size
            bool is_active
            timestamp registration_deadline
            timestamp auto_distribute_at
            string FK_teacher_id FK
            timestamp created_at
        }
        Exchange_Request {
            string PK_id PK
            string course_id FK
            string initiator_team_id FK
            string target_team_id FK
            string initiator_project_id FK
            string target_project_id FK
            string status
            string FK_approved_by FK
            timestamp approved_at
            timestamp created_at
        }
        Exchange_Confirmation {
            string PK_id PK
            string exchange_request_id FK
            string team_id FK
            string FK_confirmed_by FK
            timestamp confirmed_at
        }
        Import {
            string PK_id PK
            string course_id FK
            string type
            string filename
            int total_rows
            int success_rows
            string FK_imported_by FK
            timestamp created_at
        }
        User_Role {
            string student
            string teacher
            string admin
        }
        Request_Status {
            string pending
            string confirmed_initiator
            string confirmed_target
            string pending_teacher
            string approved
            string rejected
        }
        Team_Status {
            string forming
            string selected
            string assigned
            string locked
        }
        Import_Type {
            string projects
            string students
    
```



Результаты итерации 1

- **Фронтенд:**
 - Инициализирован проект на Vue 3 + Vite, подключены Naive UI, Pinia, Vue Router. [/main/frontend](#)
- **DevOps:**
 - Настроен docker-compose с сервисами db (MySQL), backend, frontend (через nginx). Инструкция по запуску: [main/README.md](#)
 - Добавлен .env.example с необходимыми переменными.
- **Документация:**
 - Создана use-case диаграмма и описание сценариев использования.
 - Добавлены инструкции для GitHub Copilot в [.github/copilot-instructions.md](#).
 - Обновлён корневой README.md с подробной инструкцией по запуску без Docker.

Use-case диаграмма



План на итерацию 2

- **Бэкенд**
 - Реализовать JWT-авторизацию (модуль Auth): эндпоинты /auth/login, /auth/register, защита маршрутов через Guards.
 - Подключить Prisma ORM к существующей MySQL, создать модели на основе ER-диаграммы.
 - Разработать CRUD для курсов: GET /courses, POST /courses, GET /courses/{id}, PUT /courses/{id} с валидацией и проверкой прав.
 - Написать юнит-тесты для сервисов курсов.
- **Фронтенд**
 - Реализовать формы логина и регистрации с валидацией, сохранять токен в Pinia store.
 - Создать страницу управления курсами (только для преподавателя): список, создание, редактирование.
 - Интегрировать страницы с реальным API (замена моков).
 - Добавить уведомления об успехе/ошибке (через Naive UI).

План на итерацию 2

- **База данных**
 - Написать миграции для всех таблиц и наполнить тестовыми данными.
 - Создать seed-скрипт для первоначального наполнения (преподаватель, несколько курсов, проектов).
- **DevOps**
 - Доработать Docker-окружение: добавить сценарий для разработки с hot-reload (тома для кода).
 - Настроить GitHub Actions для автоматической проверки линтера и сборки при пушах в main.
- **Документация**
 - Обновить OpenAPI спецификацию с учётом новых эндпоинтов.
 - Дополнить README разделом «Запуск с Docker» и инструкцией по настройке переменных окружения.