**Ticket app**

Това е приложение за продажба на билети за събития, където потребителят може да създава, управлява, закупува и изтегля билети за събития.

**Технологии**

* **Frontend**: React 18, React Router, CSS
* **Backend**: Node.js, Express.js, PostgreSQL, JWT удостоверяване
* **Infrastructure**: Docker, Terraform (локално)

**Основни функции**

* Удостоверяване на потребители (с JWT)
* Преглед на събития с търсене и филтриране по категории
* Закупуване на билети с обработка на плащания
* Потребителско табло и управление на билети

**Архитектура**

* **Frontend**: SPA приложение с компонентна структура (React)
* **Backend**: RESTful API с Express.js
* **Database**: PostgreSQL с оптимизирани заявки
* **Deployment**: Контейнери чрез Docker и инфраструктура в облака

**Основни компоненти:**

**Функции на фронтенда**

* **Откриване на събития**: Преглед и търсене на събития с филтриране в реално време и избор по категории
* **Удостоверяване**: Сигурна регистрация и вход със сесийно управление
* **Закупуване на билети**: Цялостен процес на покупка с валидиране и обработка на поръчки
* **Лично табло**: Управление на билети, изтегляне и преглед на история на покупки
* **Адаптивен дизайн**: Подходящ за мобилни устройства, с лесна навигация и социално споделяне

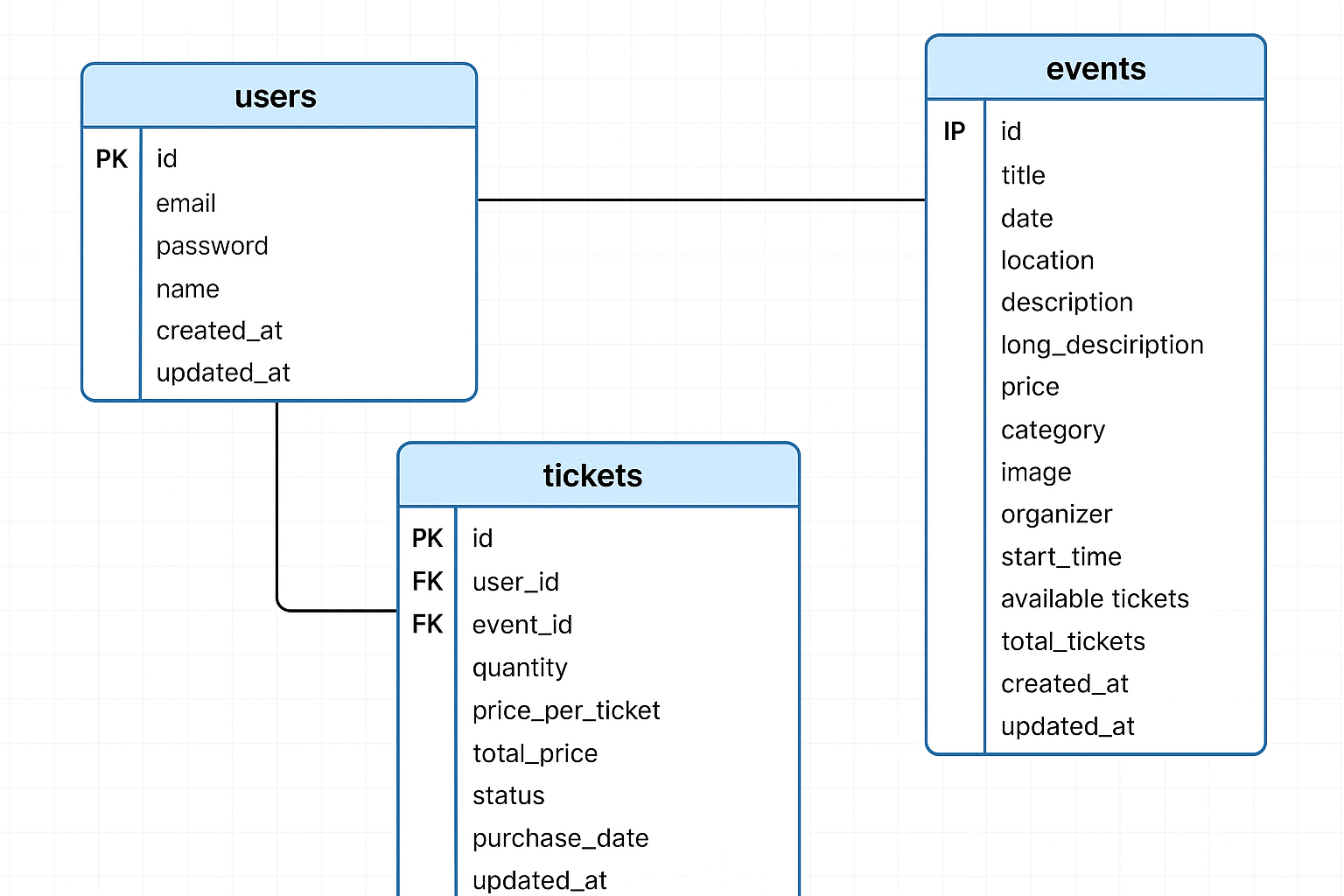
**Бекенд услуги**

* **Удостоверяване**: JWT базирано удостоверяване, с хеширане на пароли и валидиране на токени
* **Управление на събития**: CRUD операции за събития с търсене, филтриране и управление на категории
* **Обработка на билети**: Закупуване и управление на наличности
* **Валидиране на данни**: Санитизация на входни данни, обработка на грешки и форматирани отговори от API

**Документация на API**

Пълната документация на API е достъпна чрез Swagger UI на адрес /api-docs, предоставяща интерактивна документация на всички крайни точки, включително схеми на заявки/отговори и изисквания за удостоверяване.

**Схема на базата данни**



**Процес на удостоверяване**

1. JWT токени с валидност 7 дни
2. Защитени маршрути изискват Bearer токен
3. Съхранение на сесии при покупка
4. Автоматично пренасочване за неавторизирани потребители

**Deployment**

**Env от .env.example от github репото**

**Контейнеризиране с Docker**

* Мулти-контейнерна конфигурация чрез Docker Compose
* PostgreSQL с volumes
* Nginx за обслужване на статични файлове от фронтенда
* Готово за продукция с health checks

**Deployment с Terraform**

Приложението се деплойва локално с помощта на Terraform.

**CI/CD & Code Quality**

Unit тестове с Jest за конролерите.

**Основни мерки за сигурност**

* Хеширане на пароли с bcryptjs
* Удостоверяване с JWT токени
* Защитени API маршрути
* Валидация и санитизация на форми
* Управление на тайни чрез променливи на средата

**Use cases:**

**Use Case 1: Регистрация на потребител**

* **Title**: Регистрация на потребител
* **Primary Actor**: Новият потребител
* **Goal**: Създаване на профил
* **Preconditions**: Потребителят не е регистриран
* **Trigger**: Потребителят използва функционалността "Sign Up"

**Main Success Scenario**:

1. Потребителят въвежда имейл и парола.
2. Системата валидира данните.
3. Системата хешира паролата и запазва данните в базата.
4. Системата връща потвърждение и генерира JWT токен.
5. Потребителят е автоматично логнат.

**Postconditions**: Потребителят има активен профил и достъп до защитени функционалности.

**Alternative Paths**:

* Ако имейлът вече съществува – показва се грешка.
* Ако има празни или невалидни полета – потребителят получава съобщение за грешка.

**Use Case 2: Вход в профил (Login)**

* **Title**: Вход в профил
* **Primary Actor**: Регистриран потребител
* **Goal**: Получаване на достъп до защитените секции
* **Preconditions**: Потребителят вече е регистриран
* **Trigger**: Потребителят използва формата за вход

**Main Success Scenario**:

1. Потребителят въвежда имейл и парола.
2. Системата валидира данните.
3. При успешно съвпадение се издава JWT токен.
4. Потребителят е логнат и пренасочен.

**Postconditions**: Потребителят е влязъл успешно.

**Alternative Paths**:

* Грешен имейл или парола – извежда се съобщение за грешка.

**Use Case 3: Преглед на събития**

* **Title**: Преглед на събития
* **Primary Actor**: Логнат потребител и гост
* **Goal**: Откриване на интересни събития
* **Preconditions**: Няма
* **Trigger**: Посещение на страницата "Events"

**Main Success Scenario**:

1. Зарежда се списък с налични събития.
2. Потребителят може да търси и филтрира по категория.
3. При избор на събитие, се отварят детайли.

**Postconditions**: Потребителят избира събитие за повече информация или покупка.

**Alternative Paths**:

* Няма събития – показва се съобщение.

**Use Case 4: Покупка на билет (Checkout)**

* **Title**: Покупка на билет
* **Primary Actor**: Логнат потребител
* **Goal**: Успешно закупуване на билет
* **Preconditions**: Потребителят трябва да е логнат
* **Trigger**: Клик върху "Купи билет"

**Main Success Scenario**:

1. Потребителят избира брой билети.
2. Системата валидира избора.
3. Билетът се създава и свързва с потребителя.
4. Системата връща потвърждение.

**Postconditions**: Билетът е запазен и достъпен в Mytickets.

**Alternative Paths**:

* Ако няма налични билети – показва се съобщение.
* Ако не е логнат – пренасочване към Login.

**Бъдещи подобрения**

* **Платежни системи**: Поддръжка на Stripe, PayPal и други
* **Система за известия**: Имейл и SMS нотификации при покупкии QR код валидация на билети
* **Административен панел**: Управление на потребители, събития и отчети от страна на администратор
* **Интеграция с календар**: Добавяне на закупени събития в Google Calendar / iCalendar