**АРХИТЕКТУРА ПРИЛОЖЕНИЯ**

Проект представляет собой веб-приложение, реализующее отображение и управление мероприятиями с картой, авторизацией и реакциями пользователей. Архитектура построена по модели "клиент-сервер" и включает следующие ключевые компоненты:

**1. Клиентская часть (HTML + JS):**  
- Отвечает за отображение форм регистрации, входа и карты с событиями.  
- Реализует взаимодействие с сервером через fetch-запросы с JWT-аутентификацией.

**2. Серверная часть (Django + DRF):**  
- Обрабатывает все HTTP-запросы.  
- Реализует REST API и HTML-шаблоны.  
- Обеспечивает JWT-аутентификацию и управление пользователями, событиями и реакциями.

**3**. **База данных (PostgreSQL):**  
- Хранит информацию о пользователях, мероприятиях, категориях и реакциях.

**4. Docker-контейнеризация**:  
- Отдельные сервисы для Django и PostgreSQL, управляемые через docker-compose.

**ВЗАИМОДЕЙСТВИЕ КОМПОНЕНТОВ**

- Пользователь заходит на страницу входа/регистрации.  
- После авторизации клиент получает JWT токены и сохраняет их в localStorage.  
- На странице карты JavaScript запрашивает мероприятия через API (`/api/events/`) с токеном в заголовке.  
- Пользователь может создавать, просматривать и удалять мероприятия, а также оставлять реакции.  
- Все операции происходят через защищённые API-запросы.

**СТРУКТУРА БАЗЫ ДАННЫХ**

1. User:  
- id, username, email, password, avatar, created\_at, is\_staff, is\_active

2. Category:  
- category\_id, name

3. Event:  
- event\_id, title, description, latitude, longitude, datetime, category\_id, creator\_id

4. EventToCategory:  
- event\_id, category\_id (уникальная пара)

5. Reaction:  
- reaction\_id, user\_id, event\_id, type (going / not\_going)

**СТРУКТУРА API**

- POST /api/auth/login/ — вход по имени и паролю  
- POST /api/auth/register/ — регистрация  
- POST /logout/ — выход из сессии

- GET /api/users/ — текущий пользователь  
- PUT /api/users/<id>/ — редактирование пользователя

- GET/POST /api/categories/ — работа с категориями  
- GET/POST/DELETE /api/events/ — мероприятия  
- POST /api/reactions/ — установка реакции

Все API требуют JWT токен (в заголовке Authorization: Bearer).

**СТЕК ТЕХНОЛОГИЙ**

- Backend: Django 4.2, Django REST Framework, JWT (SimpleJWT), PostgreSQL  
- Frontend: HTML, CSS, JavaScript (Vanilla JS)  
- Карты: Яндекс.Карты API v2.1  
- DevOps: Docker, Docker Compose, Gunicorn, Whitenoise

**КОНТЕЙНЕРИЗАЦИЯ И ДОКЕРИЗАЦИЯ**

- backend (Django):  
 \* Использует Dockerfile с установкой зависимостей и запуском gunicorn  
 \* При старте автоматически выполняются миграции и сбор статики

- db (PostgreSQL):  
 \* Использует официальный образ postgres:17  
 \* Настройки передаются через .env

- docker-compose.yml:  
 \* Управляет порядком запуска, подключением к сети и общим томом для данных

**ДЕМОНСТРАЦИЯ (USE CASES)**

1. Регистрация и вход:  
- Пользователь переходит на /register/ и создаёт учётную запись.  
- Получает access\_token и redirect'ится на /map/.

2. Отображение карты и событий:  
- Загружается Яндекс.Карта.  
- События с координатами и описаниями отображаются как метки.

3. Создание мероприятия:  
- Клик по карте — открывается модальное окно.  
- Пользователь вводит данные, подтверждает — новое мероприятие появляется на карте.

4. Реакции на события:  
- В балуне можно выбрать "Пойду" или "Не пойду".  
- Отправляется POST /api/reactions/ с нужным типом.

5. Удаление мероприятия:  
- Кнопка "Удалить" доступна только автору события.  
- DELETE /api/events/<id>/ с обновлением карты.

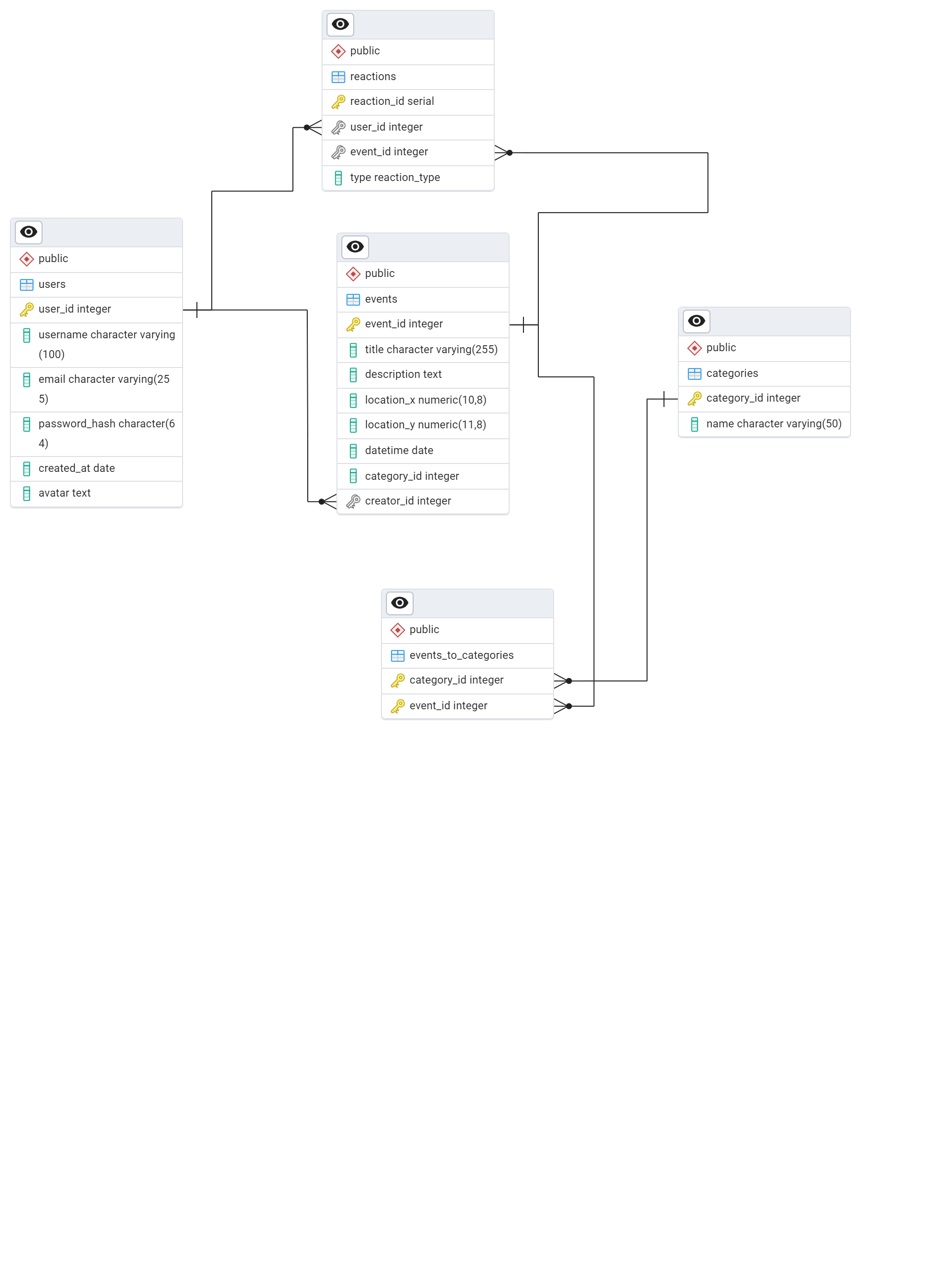
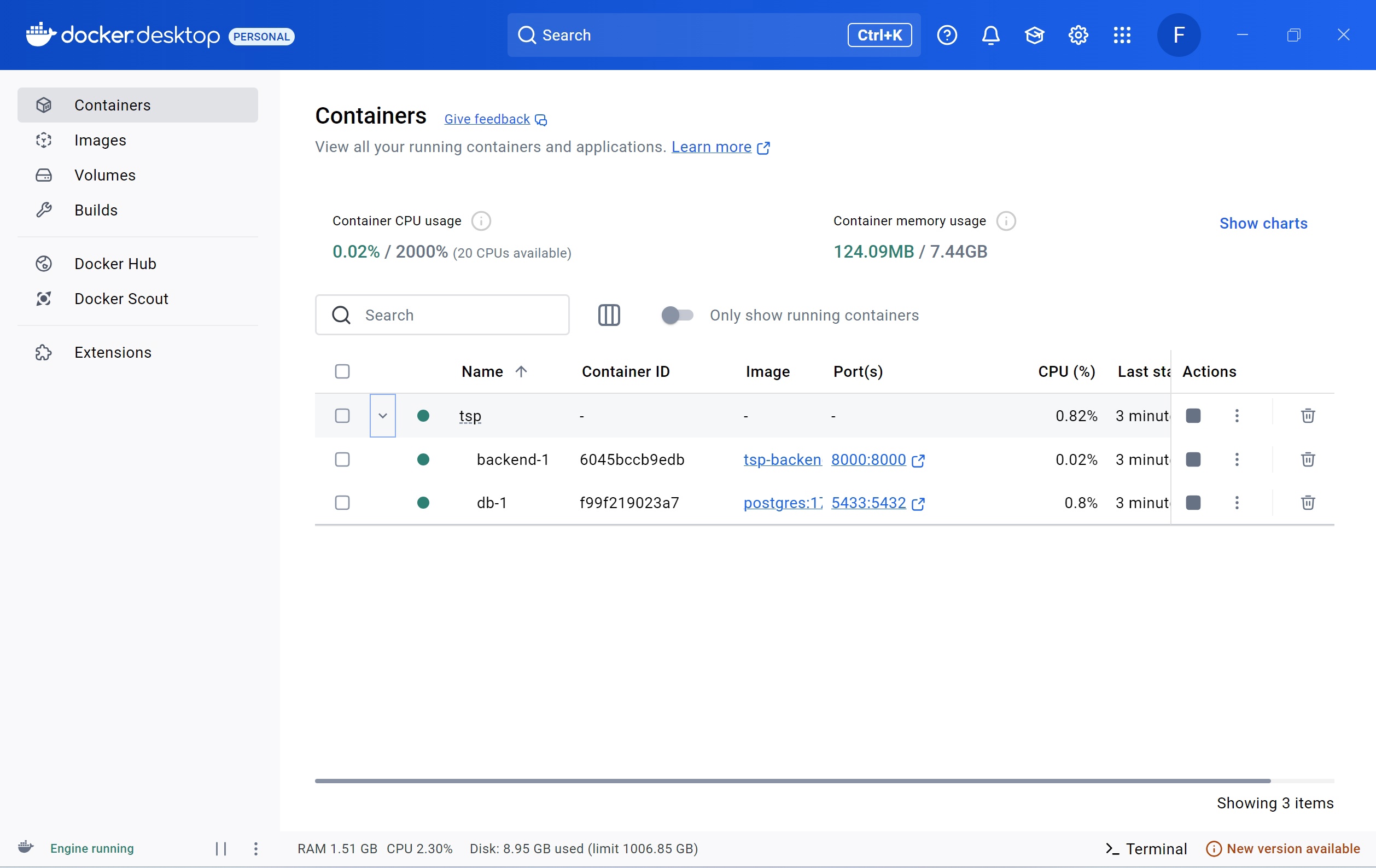


Рисунок 1 ̶ Диаграмма БД

  
Рисунок 2 ̶ Работающие контейнеры

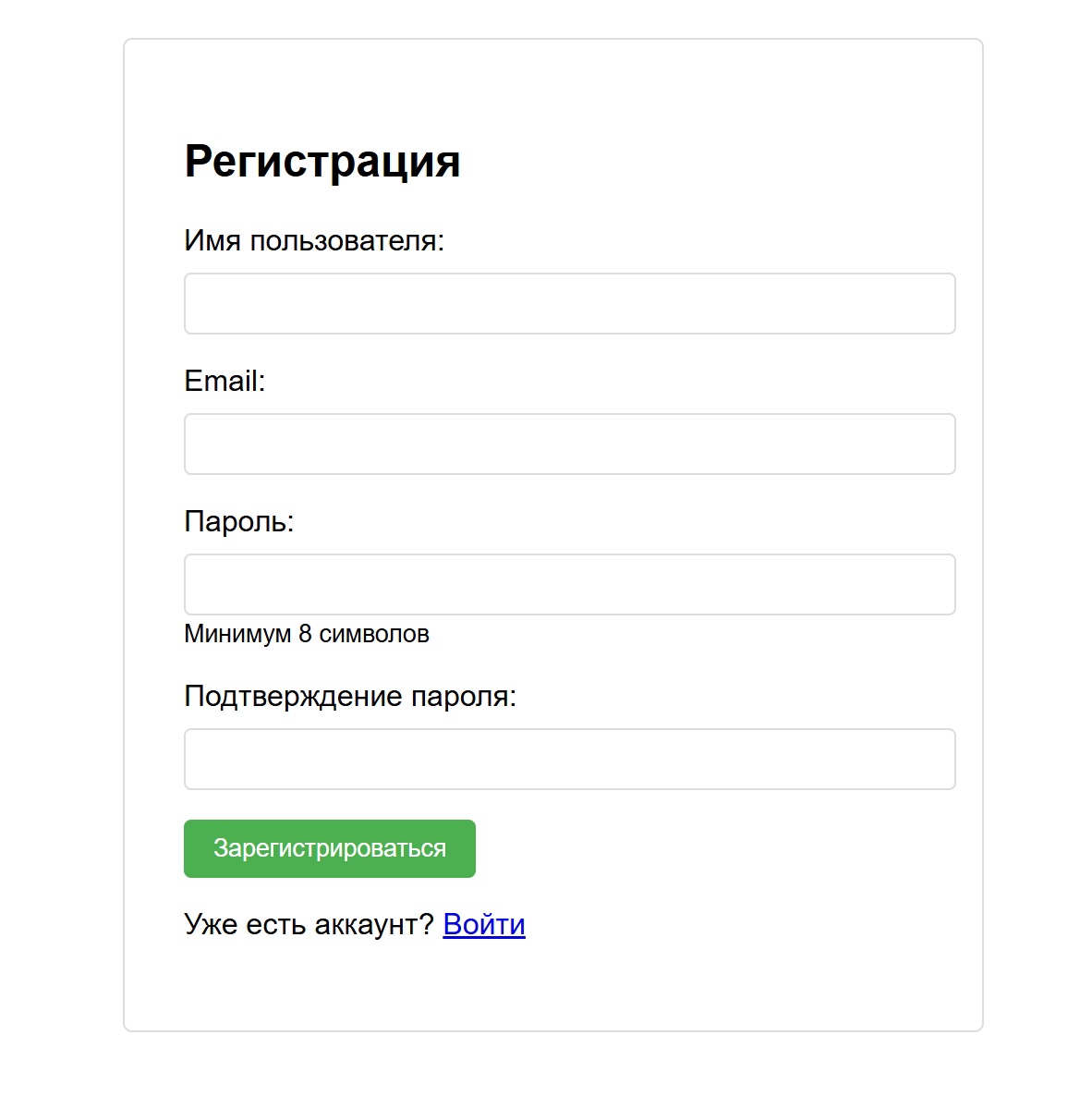


Рисунок 3 ̶ Страница регистрации

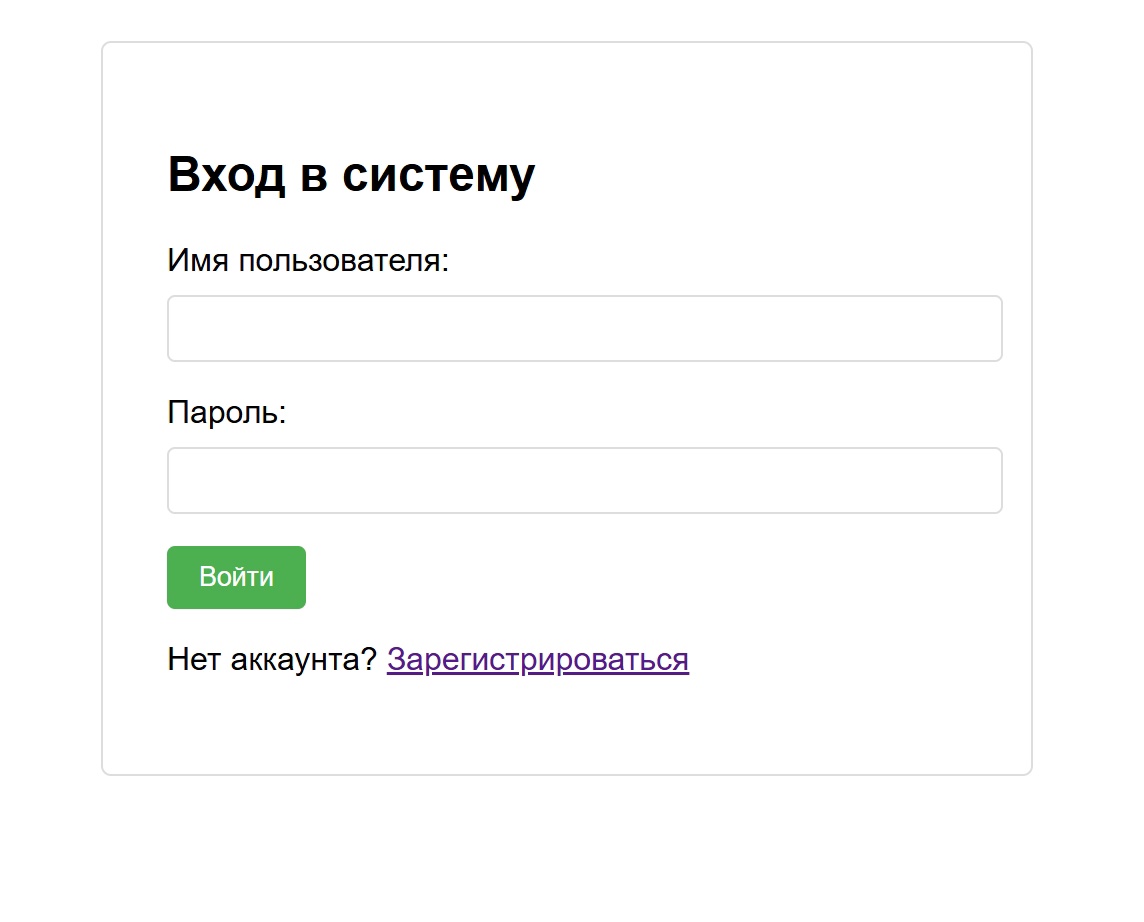


Рисунок 4 ̶ Страница входа



Рисунок 5 ̶ Главная страница

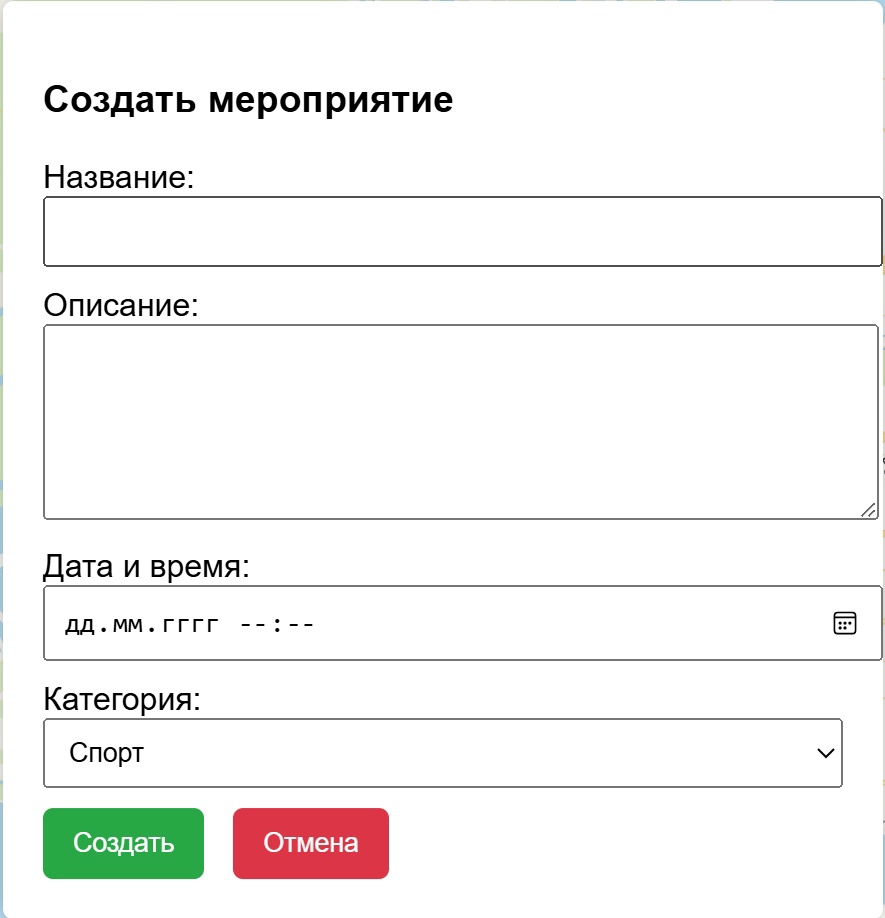


Рисунок 6 ̶ Окно создания мероприятия

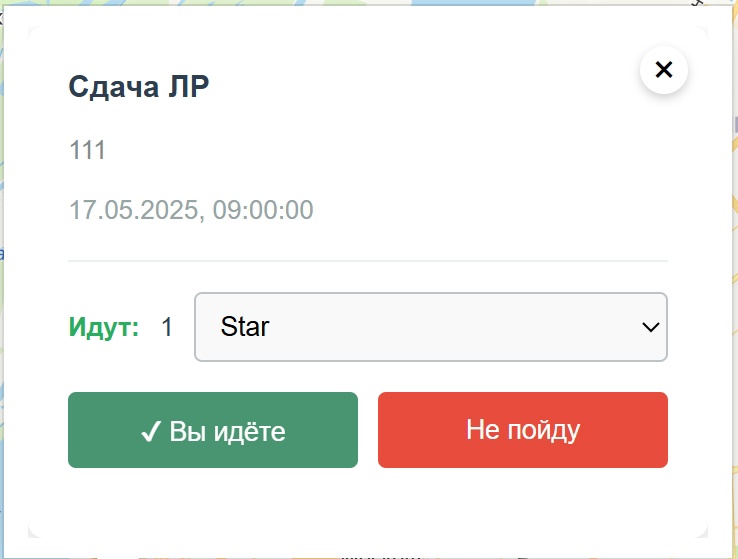


Рисунок 7 ̶ Информационное окно мероприятия