1. **Выбор и обоснование средств проектирования и реализации**

Для успешной реализации волонтёрского портала, важно правильно выбрать технологический стек, который обеспечит высокую производительность, безопасность и устойчивость системы. В рамках проекта мы используем FastAPI для серверной части, PostgreSQL для управления данными, а также NGINX в качестве веб-сервера, что позволит создать эффективную и масштабируемую платформу.

Backend реализован на базе FastAPI — легковесного Python-фреймворка для создания API. FastAPI поддерживает асинхронные запросы, что обеспечивает высокую скорость обработки данных и позволяет обрабатывать большое количество одновременных подключений. Этот фреймворк также предлагает встроенные механизмы для валидации данных и генерации документации API, что упрощает разработку и тестирование. Это также облегчает интеграцию с внешними сервисами, что особенно важно для портала, где возможны интеграции с другими волонтёрскими платформами.

Для хранения данных мы выбрали реляционную базу данных PostgreSQL. Этот СУБД является одним из самых надёжных решений для хранения больших объёмов данных, обеспечивая при этом высокую производительность и безопасность. PostgreSQL отлично поддерживает транзакционную целостность и сложные запросы, что позволит обрабатывать большое количество данных и сохранять целостность системы. Кроме того, её расширяемость позволяет добавлять новые функции и типы данных, что обеспечит гибкость при возможном развитии портала.

В качестве веб-сервера мы используем NGINX, который зарекомендовал себя как надёжное и производительное решение для обработки HTTP-запросов. NGINX эффективно управляет трафиком, распределяя нагрузку между серверами и увеличивая устойчивость системы. Благодаря встроенным функциям кэширования и поддержки обратного прокси, NGINX позволяет ускорить загрузку страниц и обрабатывать большое количество запросов, что особенно важно при росте числа пользователей. Это обеспечит бесперебойную работу портала в периоды высокой активности.

1. **Проектирование архитектуры приложения**

На рисунке (?) представлена схема архитектуры веб-приложения для волонтёрского портала:

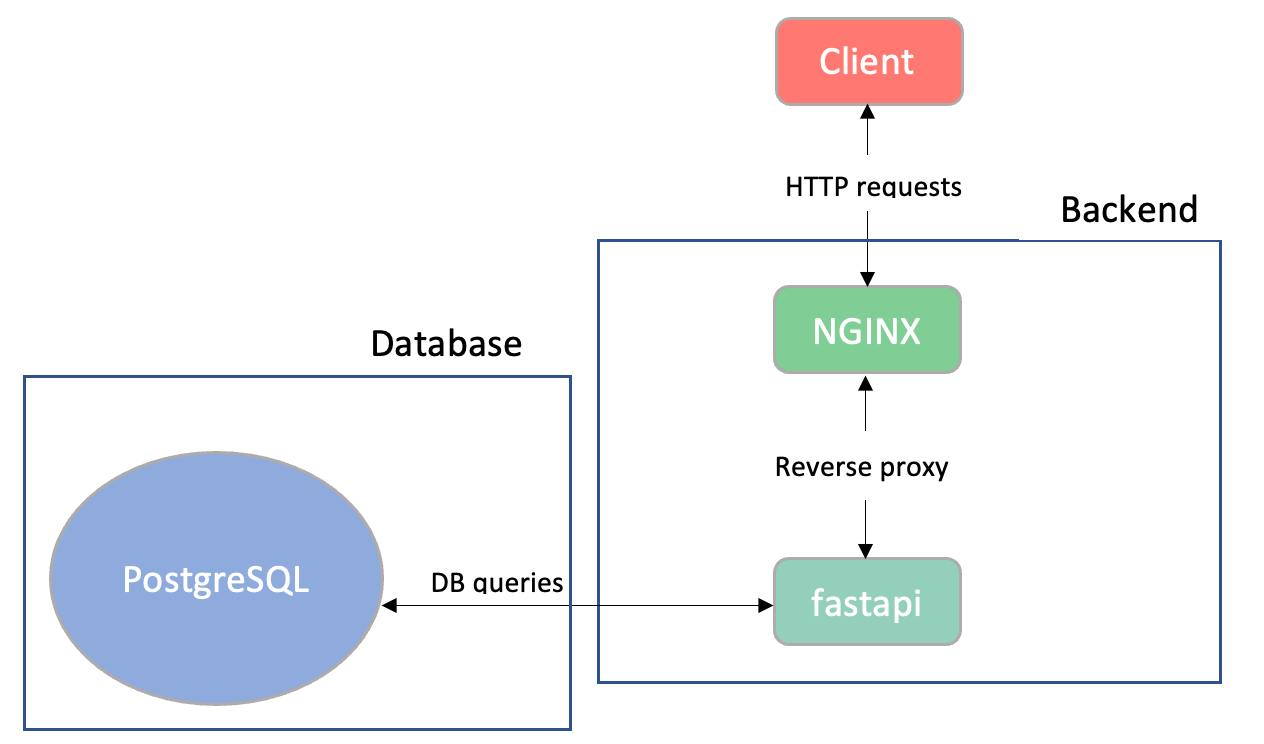


Рисунок (?) - схема архитектуры веб-приложения для волонтёрского портала

Схема включает в себя следующие компоненты:

* клиентская сторона представлена фреймворком React.js, который позволяет пользователям взаимодействовать с порталом в реальном времени. React.js отправляет HTTP-запросы на серверную часть, чтобы получить данные и выполнить различные операции;
* серверная часть представлена веб-сервером nginx и fastapi, которые отвечают за всю бизнес-логику портала: прием и обработка HTTP-запросов от клиентской части и взаимодействие с базой данных;
* база данных используется для хранения и управления данными приложения. Серверная часть обращается к базе данных через SQL-запросы, чтобы совершить действия над информацией

1. **Проектирование хранилища данных**

**Основные сущности и атрибуты**

В таблице 1 приведены основные сущности с описанием их атрибутов.

Таблица 1 – сущности и их атрибуты

|  |  |  |
| --- | --- | --- |
| **Сущность** | **Атрибуты** | **Описание** |
| alembic\_version | version\_num | Номер версии для миграций базы данных |
| category | category\_id, category\_name | Категории событий |
| country | country\_id, country\_name | Страны |
| city | city\_id, city\_name, country\_id | Города |
| volunteer\_org | vol\_id, vol\_name, vol\_contact\_person, vol\_email, vol\_phone, city\_id, country\_id, vol\_foundation\_date, vol\_status, vol\_count\_members | Волонтерские организации |
| user\_metadata | user\_metadata\_id, email, hashed\_password, city\_id, country\_id, status, isActive, avatar\_image, user\_name, user\_surname, age | Метаданные пользователей |
| user | user\_id, user\_metadata\_id, email, is\_superadm, is\_regionadm, is\_cityadm, is\_volunteer | Пользователи системы |
| event | event\_id, event\_name, category\_id, city\_id, country\_id, start\_date, end\_date, participation\_points, required\_volunteers, rewards, longitude, latitude, image, creation\_date, event\_status | События, организованные в системе |
| user\_volunteer\_org | user\_volunteer\_org\_id, user\_id, vol\_id, join\_date | Участие пользователя в волонтерской организации |
| event\_registration | registration\_id, user\_id, event\_id, registration\_date | Регистрация пользователей на события |
| chat\_messages | id, sender\_id, recipient\_id, message, time, delivered | Сообщения в чатах |

**Логическая модель данных**

Логическая модель данных представлена ключевыми связями между сущностями. Сущности связаны через внешние ключи, которые определяют связь между таблицами.

1. **Связи между странами, городами и пользователями:**

* Таблица city связана с country через атрибут country\_id.
* Таблица user\_metadata связана с city и country через city\_id и country\_id соответственно.

1. **Связи для пользователей и их метаданных:**

* user\_metadata хранит данные о пользователях, включая город, страну, статус и т.д.
* Основная таблица user содержит user\_metadata\_id и связана с user\_metadata через user\_metadata\_id.

1. **Связи для событий и волонтерских организаций:**

* volunteer\_org связывается с city и country через city\_id и country\_id, определяя местоположение организации.
* event связан с category, city, country, а также с пользователем (user\_id), который организует событие.

1. **Регистрация на события и участие в организациях:**

* event\_registration связывает user с event по user\_id и event\_id, отслеживая регистрацию на событие.
* user\_volunteer\_org связывает user с volunteer\_org для отслеживания членства пользователей в организациях.

1. **Связи для системы сообщений:**

* chat\_messages связывается с user\_metadata для отслеживания отправителя (sender\_id) и получателя (recipient\_id) сообщений.

**Инфологическая модель данных**

В инфологической модели описываются основные сущности и их взаимосвязи:

* Пользователи связаны с метаданными пользователей и могут быть участниками волонтерских организаций и событий.
* Города и страны играют роль географических данных и связаны с пользователями и организациями.
* Категории служат для классификации событий.
* Сообщения обеспечивают связь между пользователями через чаты.

**Дополнительные ограничения целостности**

1. **Уникальные ограничения:**

* Поле email в таблицах user и user\_metadata должно быть уникальным.

1. **Каскадные операции:**

* При обновлении данных таблицы city происходит каскадное обновление в зависимых таблицах (user\_metadata, volunteer\_org).
* При удалении записи в таблице user\_metadata и связанных данных в chat\_messages или event\_registration записи помечаются как NULL или удаляются каскадно.

1. **Проверки на пустые значения и обязательные поля:**

* Поля, которые участвуют в связях или являются ключевыми, должны быть не NULL (например, country\_id в таблице city должен быть заполнен для указания страны).

Схема базы данных показана на рисунке Х:

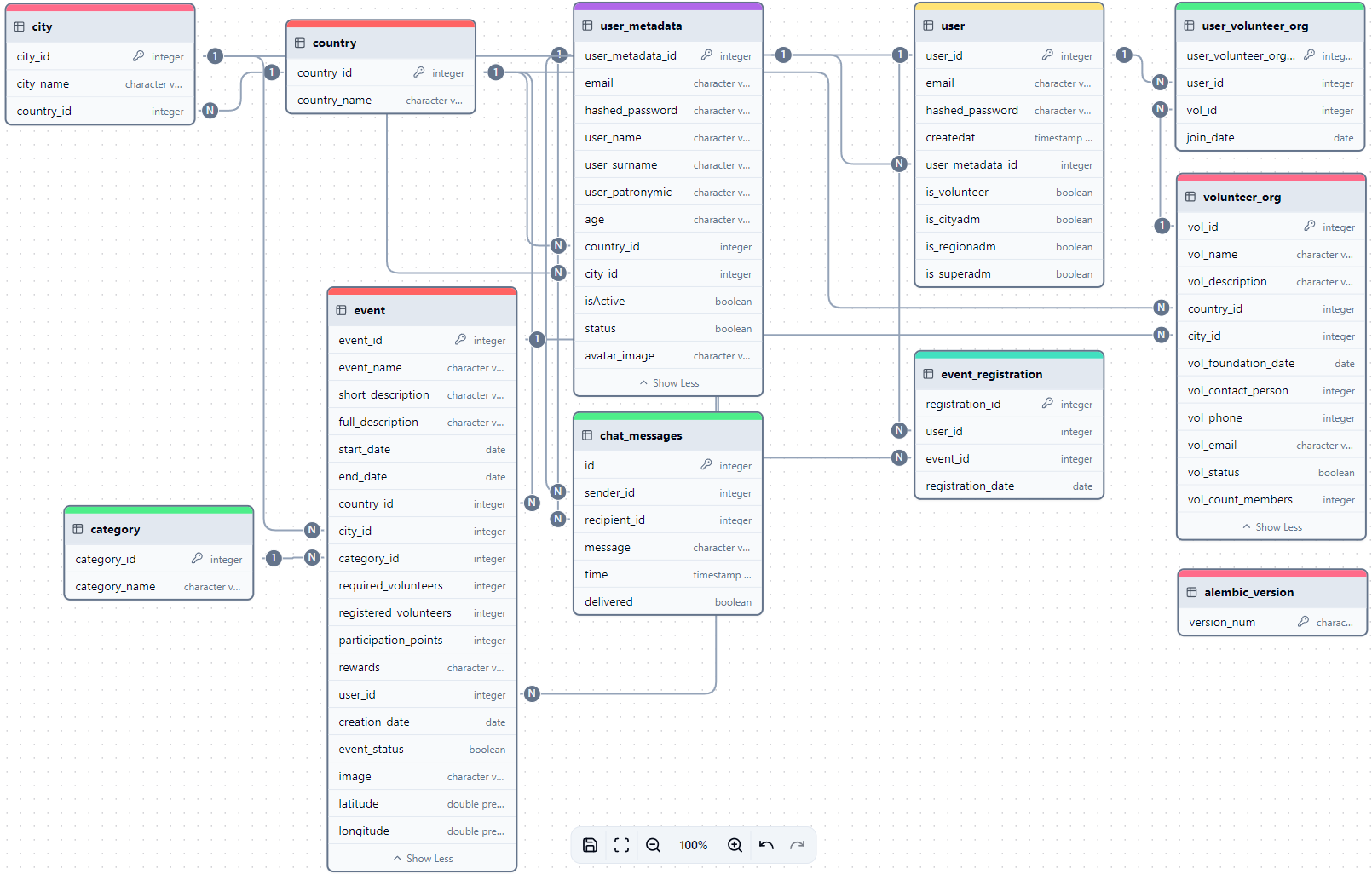


Рисунок Х – Схема БД

**Проектирование пользовательского интерфейса**

тостер

текст