РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

1 Общие сведения о программе

1.1 Введение

Руководство системного программиста предоставляет всестороннюю информацию о программе "TuneMyBand" — веб-приложении, написанном на Django 5.0 и TypeScript. Программа разработана для управления музыкальными группами, обеспечивая функционал от создания профиля пользователя до планирования выступлений и управления репетициями.

1.2 Общие сведения о программе

1.2.1 Назначение программы

Программа "TuneMyBand" создана с целью обеспечения эффективного взаимодействия музыкантов и коллективов. Она предоставляет средства для управления информацией о пользователях, музыкальных групп, репетициях, выступлениях, оборудовании и других сущностях, связанных с миром музыки.

1.2.2 Технологический стек

Программа написана на Django 5.0, обеспечивая стабильность и безопасность на серверной стороне. Фронтенд разработан с использованием TypeScript, что обеспечивает масштабируемость и поддержку современных веб-стандартов. Программа ориентирована на использование в различных операционных системах, поддерживающих протоколы TCP/IP.

1.3 Основные функциональности

Программа предоставляет широкий спектр функциональностей, включая:

- Регистрацию и управление профилем пользователя.
- Формирование музыкальных групп и управление их составом.
- Планирование и отслеживание репетиций и выступлений.
- Управление оборудованием и инструментами музыкантов.
- Создание и администрирование мероприятий.

1.4 Требования к техническим средствам

Для корректной работы программы требуется наличие веб-браузера, поддерживающего современные стандарты, такие как Netscape Navigator, MS Internet Explorer, или аналогичные. Технические средства должны поддерживать протокол TCP/IP и обеспечивать доступ к HTTP-серверу, на котором размещено приложение.

1.5 Установка и настройка

Инструкции по установке и настройке программы приведены в соответствующем разделе руководства. Пользователь должен обладать достаточными правами доступа для установки файлов и настройки необходимых параметров.

2 Структура программы

2.1 Основные компоненты

Программа "TuneMyBand" на базе Django с PostgreSQL и REST API включает следующие компоненты:

- **Back-End на Django (Web-приложение):**
    - **Модели Django:** Определены для каждой сущности системы (Пользователи, Музыкальные Группы, Оборудование, Мероприятия и др.) с использованием ORM для взаимодействия с PostgreSQL.
    - **Django REST Framework (DRF):** Используется для создания RESTful API, предоставляющего доступ к данным из базы данных.
- **Front-End на TypeScript и React:**
    - **React-компоненты:** Разработаны для пользовательского интерфейса, взаимодействующего с REST API через HTTP-запросы.
    - **Context API:** Используется для управления глобальным состоянием приложения, таким как данные пользователя и состояние приложения.
- **База данных:**
    - **PostgreSQL:** Используется в качестве базы данных для хранения данных системы. Модели Django автоматически создают таблицы в PostgreSQL.
- **REST API:**
    - **DRF Serializers:** Преобразуют данные из формата Django-моделей в формат, понятный для API.
    - **Views (Представления) с использованием DRF:** Обрабатывают запросы от фронтенда и возвращают данные в формате JSON.
    - **URL-маршруты для API:** Определены в `urls.py`, определяют, какие views обрабатывают запросы на определенных URL.

2.2 Зависимости и библиотеки

Для успешного развертывания программы необходимо удовлетворить следующие зависимости:

- **Django:** `pip install django`
- **Django REST Framework:** `pip install djangorestframework`
- **psycopg2 (для работы с PostgreSQL):** `pip install psycopg2`
- **axios (HTTP-клиент):** `npm install axios`

3 Установка и настройка

3.1 Установка программы

- Клонировать репозиторий с Django-приложением.
- Установить Python-зависимости: `pip install -r requirements.txt`.
- Произвести миграции базы данных: `python manage.py migrate`.

3.2 Настройка программы

3.2.1 Конфигурация Базы Данных

В файле `settings.py` вашего Django-приложения определите параметры подключения к базе данных PostgreSQL. Найдите секцию DATABASES и укажите следующие параметры:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'название_базы_данных',
        'USER': 'пользователь',
        'PASSWORD': 'пароль',
        'HOST': 'localhost',  # Может потребоваться изменить в зависимости от вашей конфигурации
        'PORT': '5432',       # Порт PostgreSQL
    }
}
```

3.2.2 Определение URL-маршрутов

В файле `urls.py` Django-приложения определите URL-маршруты для основного приложения и API. Пример:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('your_app.api.urls')),  # Убедитесь, что у вас есть модуль api с определенными эндпоинтами
    # Добавьте другие URL-маршруты для вашего приложения
]
```

3.2.3 Создание Суперпользователя Django

Суперпользователь Django позволяет управлять данными через админ-панель Django. Создайте суперпользователя следующей командой:

```
python manage.py createsuperuser
```

Следуйте инструкциям, введите имя пользователя, электронную почту и пароль для нового суперпользователя.

3.2.4 Применение Миграций

Перед первым запуском приложения необходимо применить миграции для создания таблиц в базе данных. Введите следующую команду:

```
python manage.py migrate
```

Это создаст необходимые таблицы согласно определенным моделям в вашем Django-приложении.

3.2.5 Запуск Django-приложения

Теперь вы можете запустить ваше Django-приложение:

```
python manage.py runserver
```

После успешного запуска, ваше приложение будет доступно по адресу http://127.0.0.1:8000/ (или другому, если вы укажете параметры при запуске сервера).

3.3 Front-End

- Установить TypeScript-зависимости: `npm install`.
- Запустить TypeScript сборку: `npm run build`.

3.4 REST API

- Проверить доступность API по определенным эндпоинтам (например, `/api/users/` или `/api/events/`).

4. Проверка программы

Проверка программы в контексте веб-приложения Django с использованием PostgreSQL и RESTful API осуществляется через следующие шаги:

4.1 Заполнение Базы Данных тестовыми данными

Перед проверкой удостоверьтесь, что база данных содержит достаточное количество тестовых данных. Используйте Django-команду для заполнения базы данных:

```
python manage.py loaddata your_test_data.json
```

Здесь `your_test_data.json` - файл с тестовыми данными в формате JSON.

4.2 Запуск Django-сервера

Запустите ваш Django-сервер:

```
python manage.py runserver
```

После запуска сервера вы сможете отправлять запросы к вашему веб-приложению.

4.3 Проверка API-эндпоинтов

Используйте инструменты для отправки HTTP-запросов (например, curl, Postman или HTTPie) для проверки функционала вашего API. Пример запроса:

```
curl -X GET http://127.0.0.1:8000/api/your_endpoint/
```

Замените `your_endpoint` на фактический эндпоинт вашего API.

4.4 Обработка сообщений об ошибках

При возникновении ошибок в процессе проверки, обратите внимание на сообщения об ошибках, которые могут включать в себя коды состояния HTTP, а также текстовые описания ошибок. Они помогут вам идентифицировать и устранить проблемы.

5. Дополнительные возможности

5.1 Интеграция с другими сервисами

При необходимости интеграции с другими сервисами или расширения функционала вашего веб-приложения, создайте дополнительные эндпоинты API и управляйте ими в соответствии с требованиями вашего проекта.

5.2 Динамическое управление данными

Реализуйте возможности динамического управления данными через админ-панель Django. Добавьте, изменяйте и удаляйте записи, следя за целостностью данных в вашей системе.

6. Сообщения системному программисту

В случае возникновения ошибок или проблем при настройке, проверке и использовании программы, обратитесь к таблице сообщений системного программиста. В данной таблице представлены тексты сообщений, их содержание и предполагаемые действия системного программиста для устранения проблем. Основывайтесь на этих сообщениях для диагностики и решения возможных неполадок в веб-приложении.