# РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

# 1 Общие сведения о программе

## 1.1 Введение

Руководство системного программиста предоставляет всестороннюю информацию о программе "MusicBand" — веб-приложении, написанном на Flask. 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 "MusicBand" 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 Flask, 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 "MusicBand" на базе Flask с PostgreSQL и REST API включает следующие компоненты:

* Back-End на Flask (Web-приложение):
  + Модели Flask: Определены для каждой сущности системы (Пользователи, Музыкальные Группы, Оборудование, Мероприятия и др.) с использованием ORM для взаимодействия с PostgreSQL.
* Front-End на TypeScript:
  + 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 Flask автоматически создают таблицы в PostgreSQL.
* REST API:
  + DRF Serializers: Преобразуют данные из формата Flask-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:

* Flask: pip install flask
* psycopg2 (для работы с PostgreSQL): pip install psycopg2

# 3 Установка и настройка

## 3.1 Установка программы

* Клонировать репозиторий с Flask-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 Flask-приложения определите параметры подключения к базе данных PostgreSQL. Найдите секцию connect и укажите следующие параметры:

connect = psql.connect(

dbname='название\_бд',

user='пользователь',

password='пароль'

)

### 3.2.2 Определение URL-маршрутов

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

from user import \*

from admin import \*

from music import \*

app = Flask(\_\_name\_\_)

app.secret\_key = ' '

app.register\_blueprint(user)

app.register\_blueprint(admin, url\_prefix='/admin')

app.register\_blueprint(music, url\_prefix='/music')

### 3.2.3 Создание Суперпользователя Flask

Суперпользователь Flask позволяет управлять данными через админ-панель Flask. 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 Flask-приложении.

### 3.2.5 Запуск Flask-приложения

Теперь вы можете запустить ваше Flask-приложение:

python manage.py

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 Flask 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 Flask-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 Flask-сервера

Запустите ваш Flask-сервер:

python manage.py

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 Flask. 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.