**Документация по работе бэкенда приложения**

**Общая информация**

**Бэкенд приложения реализован на основе фреймворка FastAPI и использует базу данных PostgreSQL для хранения данных. Приложение предоставляет REST API для работы с организациями, пользователями (волонтерами), QR-кодами, скидками и историей операций. Для аутентификации используется JWT (JSON Web Token).**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1. Настройки и конфигурация**

**Переменные окружения**

**Все конфиденциальные данные и настройки хранятся в .env файле:**

**DB\_HOST=localhost**

**DB\_PORT=5432**

**DB\_USER=postgres**

**DB\_PASSWORD=0809**

**DB\_NAME=Dobrodon**

**SECRET\_KEY=EJJYL1xhbn5feEuypsC8xbdqSFUcnCZQGLQY4UEwPvKm6zDCDzeMp8VjSfOt4wE3**

**ALGORITHM=SHA-256**

**ACCESS\_TOKEN\_EXPIRE\_MINUTES=15**

**SMTP\_EMAIL=noreplydobrodon@rambler.ru**

**SMTP\_PASSWORD=Gr@3\_yR4q**

**ADMIN\_NAME=admin**

**ADMIN\_EMAIL=admin@admin**

**ADMIN\_PASSWORD=admin**

**Инициализация базы данных**

**При запуске приложения автоматически выполняется создание таблиц в базе данных (если они отсутствуют). Используется функция init\_db().**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2. Модели базы данных**

**Основные таблицы:**

**1. organizations**

**Хранит данные об организациях.**

**Поля: id, name, inn, email, phone, hashed\_password, address, description, category.**

**2. people**

**Хранит данные о пользователях (волонтерах).**

**Поля: id, full\_name, inn, phone, email, birth\_date, achievement, rating, rating\_ranga, role.**

**3. history**

**Хранит архивные данные пользователей.**

**Поля: id, full\_name, inn, phone, email, birth\_date, achievement, rating, archived\_at, file\_name.**

**4. qr\_hashes**

**Хранит информацию о QR-кодах.**

**Поля: id, hash\_value, created\_at, created\_by, rating, used, used\_at, used\_location.**

**5. discounts**

**Хранит информацию о скидках.**

**Поля: id, description, discount\_percentage, start\_date, end\_date.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**3. API-ручки**

**Аутентификация и регистрация**

**1. Регистрация организации**

**Метод: POST /register**

**Описание: Регистрация новой организации.**

**Возвращает токен доступа и сообщение об успешной регистрации.**

**2. Вход в систему**

**Метод: POST /login**

**Описание: Аутентификация пользователя или администратора.**

**Возвращает токен доступа и роль пользователя.**

**3. Подтверждение кода**

**Метод: POST /verify-code**

**Описание: Подтверждение кода, отправленного на email.**

**Возвращает токен доступа и сообщение об успешной проверке.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Управление пользователями (волонтерами)**

**1. Загрузка CSV-файла**

**Метод: POST /upload-csv/**

**Описание: Загрузка данных о пользователях из CSV-файла.**

**Переносит текущие данные в таблицу history и очищает таблицу people.**

**2. Получение списка пользователей**

**Метод: GET /people**

**Описание: Получение списка пользователей с возможностью фильтрации по email и ИНН.**

**3. Обновление данных пользователя**

**Метод: PUT /update-person**

**Описание: Обновление данных пользователя.**

**Возвращает сообщение об успешном обновлении.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QR-коды**

**1. Генерация QR-кода**

**Метод: POST /generate-qr-hash**

**Описание: Генерация уникального QR-кода для пользователя.**

**Сохраняет операцию в таблице qr\_hashes.**

**2. Проверка QR-кода**

**Метод: POST /check-qr-hash**

**Описание: Проверка статуса QR-кода.**

**Возвращает уровень рейтинга и сообщение о действительности.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Скидки**

**1. Создание скидки**

**Метод: POST /create-discount**

**Описание: Создание новой записи о скидке.**

**Проверяет корректность дат и сохраняет скидку в базу данных.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**История операций**

**1. Получение всех QR-кодов**

**Метод: GET /qr-hashes**

**Описание: Возвращает список всех записей из таблицы qr\_hashes.**

**2. Получение истории QR-кодов пользователя**

**Метод: GET /user-qr-history**

**Описание: Возвращает историю операций QR-кодов для указанного пользователя по его email.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4. Функции и утилиты**

**Аутентификация**

**• Хэширование паролей: Используется библиотека passlib с алгоритмом bcrypt.**

**• JWT: Генерация токенов доступа выполняется с использованием библиотеки PyJWT.**

**Отправка email**

**• SMTP: Используется сервер Rambler для отправки писем с кодами подтверждения.**

**• HTML-шаблон: Код подтверждения вставляется в HTML-шаблон письма.**

**Генерация уникальных хешей**

**• Алгоритм: SHA-256 для генерации уникальных хешей QR-кодов.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5. Обработка ошибок**

**• HTTP-статусы: Используются стандартные HTTP-статусы для обработки ошибок (например, 400, 401, 404, 500).**

**• Логирование: Ошибки логируются для последующего анализа.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6. Зависимости**

**• База данных: PostgreSQL.**

**• ORM: SQLAlchemy.**

**• Валидация данных: Pydantic.**

**• Асинхронность: FastAPI.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**7. Требования к окружению**

**• Установленный Python версии 3.8+.**

**• Настроенный SMTP-сервер для отправки email.**

**• База данных PostgreSQL с предварительно созданной схемой.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8. Дополнительные замечания**

**• Все секретные данные должны храниться в переменных окружения.**

**• Рекомендуется использовать HTTPS для защиты передаваемых данных.**

**• Регулярно обновляйте зависимости для обеспечения безопасности.**