# Документация для бекэнда на Express

## Введение

Данный документ предоставляет информацию о структуре и функциональности бекэнда, написанного на Express.js. Express.js — это минималистичный и гибкий фреймворк веб-приложений для Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

## Настройка окружения

1. Установите [Node.js](https://nodejs.org/).
2. Установите Express.js с помощью npm: npm install express --save.
3. Создайте файл server.js и импортируйте express: const express = require('express');.

## Детальное описание файла index.js

### Настройка и зависимости

В файле index.js используются следующие модули:

* express для создания и настройки сервера.
* express-session для работы с сессиями.
* Маршрутизаторы userRouter и authRouter для обработки маршрутов, связанных с пользователями и аутентификацией соответственно.

### Настройка порта

Приложение будет слушать порт, указанный в переменной окружения PORT, или порт 8070 по умолчанию.

### Настройка Express

Приложение настроено для обработки JSON и URL-кодированных тел запросов. Также настроены сессии с использованием express-session.

### Маршрутизация

Настроены два основных маршрута:

* /api для маршрутов, связанных с пользователями.
* /auth для маршрутов, связанных с аутентификацией.

### Запуск сервера

В конце файла сервер запускается на указанном порту, и в консоль выводится сообщение о его успешном запуске.

## Детальное описание файла user.routes.js из папки routes

### Зависимости

В файле user.routes.js используются следующие модули:

* express: для создания и настройки маршрутов.
* db: для работы с базой данных.
* store: для работы с загрузкой и хранением файлов.
* fs: для работы с файловой системой.
* path: для работы с путями файлов.
* userController: контроллер, содержащий логику обработки запросов, связанных с пользователями.
* userMiddleware и roleMiddleware: middleware для проверки прав доступа и ролей пользователей.
* validator: для валидации данных.

### Маршруты

В файле определены следующие маршруты:

1. /user\_list: Получение списка пользователей.
2. /user\_id/:email: Получение ID пользователя по его email. Доступно для ролей "Ученик", "Эксперт" и "Учитель".
3. /user\_inf\_email/:email: Получение информации о пользователе по его email с проверкой прав доступа через userMiddleware.
4. /user\_inf/:user\_id: Получение информации о пользователе по его ID с проверкой прав доступа через userMiddleware.
5. /addition\_data: Добавление дополнительных данных для пользователя. Доступно для ролей "Ученик", "Эксперт" и "Учитель".
6. /add\_level\_1\_test: Добавление теста уровня 1. Доступно только для роли "Учитель".
7. /getTasksForExpert: Получение задач для эксперта. Доступно только для роли "Эксперт".
8. /uploads/: Маршрут для ручной загрузки файлов. Доступно для ролей "Ученик", "Эксперт" и "Учитель".
9. /add\_level\_1\_test\_with\_files: Добавление теста уровня 1 с файлами. Доступно только для роли "Учитель".

(Примечание: некоторые маршруты, связанные с загрузкой и скачиванием файлов, были закомментированы.)

### Экспорт

В конце файла экспортируется объект router, который содержит все определенные маршруты.

## Файл auth.routes.js в папке routes

Этот файл отвечает за маршрутизацию запросов, связанных с аутентификацией пользователей.

### Зависимости:

* express: Используется для создания экземпляра маршрутизатора.
* authMiddleware: Промежуточное ПО для проверки и валидации запросов аутентификации.
* authController: Контроллер, содержащий логику обработки запросов аутентификации.

### Маршруты:

1. **POST /registration**
   * Промежуточное ПО: authMiddleware.validateRegistration
   * Контроллер: authController.registration
   * Описание: Регистрация нового пользователя.
2. **POST /login**
   * Промежуточное ПО: authMiddleware.validateLogin
   * Контроллер: authController.login
   * Описание: Вход в систему существующего пользователя.

Этот файл экспортирует настроенный маршрутизатор для использования в других частях приложения.

## Файл auth\_controller.js в папке controller

Этот файл отвечает за контроллеры, связанные с аутентификацией пользователей.

### Зависимости:

* express-validator: Используется для валидации данных.
* bcryptjs: Для хеширования паролей.
* jsonwebtoken: Для создания JWT токенов.
* express: Используется для работы с JSON.
* db: Для работы с базой данных.
* config: Для доступа к секретному ключу.
* express-session: Для работы с сессиями.

### Методы:

1. **generateAccesToken(user\_id, type\_user, email)**: Генерирует JWT токен на основе данных пользователя.
2. **async registration(req, res)**: Регистрирует нового пользователя. Проверяет наличие пользователя с таким же email, хеширует пароль и сохраняет нового пользователя в базе данных.
3. **async login(req, res)**: Аутентифицирует пользователя. Проверяет совпадение email и пароля, генерирует JWT токен и сохраняет его в сессии.

Этот файл экспортирует экземпляр класса Auth\_controller для использования в других частях приложения.

### Папка: controller

#### Файл: user\_controller.js

**Описание:**

Этот файл содержит контроллеры для обработки запросов, связанных с пользователями.

**Класс:** User\_controller

**Методы:**

* **getUserList(req, res)**: Возвращает список всех пользователей.
* **additionalData(req,res)**: Обновляет дополнительные данные пользователя.
* **getUserInformation(req, res)**: Возвращает информацию о пользователе по его ID.
* **getUserIDForEmail(req, res)**: Возвращает ID пользователя по его email.
* **getUserDataForEmail(req, res)**: Возвращает данные пользователя по его email.
* **add\_level\_1\_test(req,res)**: Добавляет тест уровня 1.
* **getTasksForExpert(req, res)**: Возвращает задания для эксперта.
* **uploads(req, res)**: Загружает файлы.
* **download(req, res)**: Скачивает файлы.

## Детальное описание файла config.js

### Описание файла

Файл config.js содержит конфигурационные параметры, которые могут быть использованы в различных частях вашего приложения.

### Секретный ключ

В файле определена переменная secret, представляющая собой секретный ключ. Этот ключ может быть использован для различных целей, таких как подпись и верификация токенов или шифрование данных.

(Примечание: В целях безопасности рекомендуется не раскрывать секретные ключи в публичных местах. Вместо этого используйте переменные окружения или другие методы безопасного хранения для чувствительной информации.)

### Экспорт

В конце файла экспортируется объект с секретным ключом.

## Детальное описание файла db.js

### Зависимости

В файле db.js используется модуль pg для работы с PostgreSQL базой данных в Node.js.

### Настройка соединения с базой данных

Создан объект Pool для установления соединения с базой данных. Параметры соединения:

* Имя пользователя: volckovni2
* Хост: pg3.sweb.ru
* Порт: 5432
* Название базы данных: volckovni2

(Примечание: В целях безопасности пароль и другие чувствительные данные были опущены в этом описании.)

### Экспорт

В конце файла экспортируется объект pool, который может быть использован для выполнения запросов к базе данных.

## Детальное описание файла store.js

### Зависимости

В файле store.js используются следующие модули:

* multer для обработки загрузки файлов.
* path для работы с путями файлов.
* fs для работы с файловой системой.

### Класс Store

Этот класс предоставляет функциональность для хранения и обработки загруженных файлов. Основные методы и свойства класса:

* **getUploadsPath**: Функция, генерирующая путь к папке для сохранения загруженных файлов.
* **storage**: Объект для задания папки и имени файла при сохранении.
* **upload**: Объект Multer для обработки загрузки файлов с фильтром поддерживаемых типов файлов.
* **checkReferer**: Middleware для проверки referer адреса.
* **transliterate**: Функция для транслитерации текста.

### Экспорт

В конце файла экспортируется новый экземпляр класса Store.

## Документация по Middleware

### 1. Файл: user\_controller.js

Этот файл содержит класс User\_controller, который предоставляет методы для обработки запросов, связанных с пользователями. Основные функции включают:

* Получение списка пользователей.
* Обновление дополнительных данных пользователя.
* Получение информации о пользователе по его ID.
* Получение ID пользователя по его email.
* Добавление теста уровня 1.
* Загрузка и скачивание файлов.

### 2. Файл: authMiddleware.js

Этот файл содержит middleware для валидации данных при регистрации и входе в систему. Основные функции включают:

* Валидация данных регистрации.
* Валидация данных входа.

### 3. Файл: roleMiddleware.js

Этот файл содержит middleware для проверки роли пользователя на основе JWT (JSON Web Token). Он проверяет, имеет ли пользователь право доступа к определенным ресурсам на основе его роли.

### 4. Файл: userMiddleware.js

Этот файл содержит middleware для проверки авторизации пользователя на основе JWT (JSON Web Token). Он проверяет, авторизован ли пользователь, и извлекает информацию о пользователе из действительного токена.

Этот middleware обычно используется в маршрутах, где необходимо удостовериться, что пользователь авторизован, прежде чем предоставить доступ к определенным ресурсам или функциям.

## Заключение

Express.js — это мощный инструмент для создания веб-приложений. При правильной настройке и документировании он может служить надежной основой для вашего проекта.