### Задание: Разработка веб-сервера на Node.js с несколькими маршрутами

#### Цель:

Научиться создавать веб-сервер на Node.js с использованием фреймворка Express. Сервер должен поддерживать несколько маршрутов, позволяющих выполнять различные операции. Клиент или инструменты, такие как Postman, могут использоваться для отправки запросов к серверу.

### Описание задания:

Необходимо разработать веб-сервер на Node.js, который предоставляет API для управления списком задач (Todo List). Сервер должен поддерживать следующие функции:

1. **Получение списка задач** : Пользователь может получить список всех задач.
2. **Добавление новой задачи** : Пользователь может добавить новую задачу.
3. **Обновление статуса задачи** : Пользователь может отметить задачу как выполненную или незавершённую.
4. **Удаление задачи** : Пользователь может удалить задачу по её идентификатору.

Данные о задачах можно хранить в памяти сервера или в файле JSON.

### Требования к функционалу:

#### 1. Маршрут: Получение списка задач

* **Метод** : GET
* **URL** : /tasks
* **Описание** : Возвращает список всех задач в формате JSON.
* **Пример ответа** :

json

[

{

"id": 1,

"description": "Сделать домашнее задание",

"completed": false

},

{

"id": 2,

"description": "Купить продукты",

"completed": true

}

]

#### 2. Маршрут: Добавление новой задачи

* **Метод** : POST
* **URL** : /tasks
* **Описание** : Добавляет новую задачу.
* **Тело запроса** :

json

{

"description": "Выполнить проект"

}

* **Пример ответа** :

json

{

"message": "Задача успешно добавлена",

"task": {

"id": 3,

"description": "Выполнить проект",

"completed": false

}

}

#### 3. Маршрут: Обновление статуса задачи

* **Метод** : PUT
* **URL** : /tasks/:id
* **Описание** : Обновляет статус задачи (выполнена/не выполнена).
* **Тело запроса** :

json {

"completed": true

}

* **Пример ответа** :

json:

{

"message": "Статус задачи обновлён",

"task": {

"id": 1,

"description": "Сделать домашнее задание",

"completed": true  
}

}

#### 4. Маршрут: Удаление задачи

* **Метод** : DELETE
* **URL** : /tasks/:id
* **Описание** : Удаляет задачу по её идентификатору.
* **Пример ответа** :

json: {

* + "message": "Задача успешно удалена"

}

### Требования к реализации:

1. **Сервер** :
   * Использовать Node.js и фреймворк Express для создания сервера.
   * Сервер должен запускаться на порту 3000 или другом порту по вашему выбору.
   * Все маршруты должны быть документированы в коде с помощью комментариев.
2. **Хранение данных** :
   * Данные о задачах можно хранить в памяти сервера (например, в массиве) или в файле JSON.
   * Если используется файл JSON, убедитесь, что данные сохраняются между запусками сервера.
3. **Тестирование** :
   * Протестировать каждый созданный маршрут с помощью Postman или другого инструмента для отправки HTTP-запросов.
   * Сделать скриншоты результатов тестирования для каждого маршрута.
4. **Обработка ошибок** :
   * Обрабатывать ситуации, когда пользователь пытается получить/обновить/удалить несуществующую задачу.
   * Предусмотреть сообщения об ошибках для клиента.

### Пример структуры проекта:

/project

│

├── app.js # Основной файл сервера

├── package.json # Конфигурационный файл проекта

├── tasks.json # Файл для хранения задач (опционально)

└── README.md # Описание проекта (необязательно)

### Критерии оценки:

1. **Функциональность** :
   * Все маршруты (GET, POST, PUT, DELETE) работают корректно.
   * Обработка ошибок реализована правильно.
2. **Структура кода** :
   * Код читаемый, содержит комментарии.
   * Используются лучшие практики программирования.
3. **Тестирование** :
   * Все маршруты протестированы.
   * Коллекция запросов сохранена и приложена к решению.
4. **Отчёт** :
   * Отчёт подробно описывает работу программы.
   * Содержит все необходимые скриншоты.

### Инструкция по сдаче:

1. Создайте архив (zip или tar.gz) с файлами проекта, включая:
   * Исходный код сервера.
   * Экспорт коллекции запросов из Postman.
   * Отчёт в формате .docx или .pdf.