**Несложный API для обработки графа на Node.js**

**Как я подружил API на Node.js и обработку графа.**

Почему стоит выбрать Node.js для разработки API? Высокая производительность Node.js обеспечивает низкий уровень нагрузки на процессор благодаря асинхронной обработке запросов. Это позволяет обрабатывать большое количество параллельных запросов без затормаживания системы. Так же, единая языковая среда поможет в решении проблем и даст возможность быстро найти ответы на ваши вопросы.

**Архитектура API на Node.js**

Для разработки архитектуры API на Node.js, используются веб-фреймворк Express и обработка графа на языке JavaSрcript:

API (Application Programming Interface) — набор определенных методов и инструментов, предоставляемых программным обеспечением для взаимодействия с другими программами. Разработка API является важной частью процесса создания веб-приложений и сервисов, и Node.js является одной из популярных платформ, используемых для разработки API.

Node.js — среда выполнения JavaScript, построенная на базе движка V8 JavaScript Engine от Google. Благодаря своей асинхронности и событийно-ориентированной архитектуре, Node.js обеспечивает высокую производительность и масштабируемость, что делает его идеальным выбором для разработки API.

npm (Node Package Manager) — стандартный менеджер пакетов, автоматически устанавливающийся вместе с Node.js. Он используется для скачивания пакетов из облачного сервера npm, либо для загрузки пакетов на эти сервера.

Express - минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

Endpoint - название адреса, который будет принимать сообщения от клиента. Обычно представлен URL и портом. Пример: http:/127.0.0.1:5000

**Разработка API**

Перед началом разработки API, необходимо задаться вопросом: Какую задачу оно будет решать? Конечно же обработку каких-либо данных. Для примера я выбрал обработку неориентированного графа, графа у которого ребра не имеют направления. Для примера покажу реализацию получения соседей выбранного узла.

**Начало разработки на Node.js**

Прежде чем приступить к разработке API на Node.js, убедитесь, что у вас установлен Node.js на вашей системе. Вы можете скачать и установить последнюю версию Node.js с официального сайта (https://nodejs.org).

Как только Node.js установлен, вы можете создать новую директорию для вашего проекта и инициализировать его с помощью npm. Откройте командную строку (или терминал) и выполните следующие команды:

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

При выполнении команды `npm init` вам будет предложено задать несколько вопросов о вашем проекте. Вы можете просто нажать Enter, чтобы принять значения по умолчанию, или ввести свои значения.

Сразу установим необходимые пакеты для проекта:

1. express – пакет фреймворка Express
2. express-fileupload – дополнительный пакет для загрузки файлов в Express
3. nodemon – утилита которая отслеживает файловую систему приложения Node и автоматически перезапускает процесс, если есть такая необходимость

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

После установки пакетов настраиваем скрипты запуска сервера Node в файле package.json

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Далее создаем точку входа в приложение

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

На скриншоте выше создается экземпляр приложения и в нем регистрируются функции загрузки файлов через API и парсинг JSON формата данных.

Запускаем приложение командой:

Изображение выглядит как снимок экрана, Шрифт, текст, Графика

Автоматически созданное описание

Заходим по адресу http:/127.0.0.1:5000 и видим всеми нами любимое сообщение:

Изображение выглядит как текст, снимок экрана, Шрифт, белый

Автоматически созданное описание

**Декомпозиция API.**

Для того чтобы сохранить читаемость кода нашего API и не плодить много функций в одном месте подготовим папки и файлы для его декомпозиции.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

1. controllers – директория для контроллера ненаправленного графа
2. module\_gm – директория для класса графа на языке js
3. routes – директория для endpoint`ов
4. storage – директория для сохранения загруженных файлов и результатов выполнения API
5. test\_files – директория для файлов с тестовыми данными графа
6. utils – директория для переиспользуемых функций.

Вспомогательные функции для работы с csv и json

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Класс UnDirectedGraph

Реализуем простой класс ненаправленного графа с базовыми функциями получения узла или ребра, а также функцию с алгоритмом нахождения соседей выбранного узла на языке js

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

После создания класса ненаправленного графа, напишем контроллер, который будет его использовать, и endpoint`ы API для вызова функций контроллера.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

После создания контроллера и endpoint`ов модифицируем index.js файл, подключая написанные роуты.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Подготовка тестовых данных.

Изобразим вот такой граф:

Изображение выглядит как зарисовка, круг, рисунок, графическая вставка

Автоматически созданное описание

Слева список узлов, справа список ребер.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Тестирование. Postman.

Postman - Это инструмент для работы с API, который позволяет тестировщику посылать запросы к сервисам и работать с их ответами. С его помощью можно протестировать бекенд и убедиться, что он корректно работает.

Для начала подготовим запросы, с которыми нам придется иметь дело.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. UploadFilesNodesAndEdges – Post запрос загрузки файлов с узлами и ребрами на бэкенд.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Возвращает сообщение об успешной загрузки файлов и в директории storage появляются два файла в формате json.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

1. GetGraph – Get запрос считывающий тестовые данные из файлов графа.

Изображение выглядит как текст, линия, Шрифт, белый

Автоматически созданное описание

Возвращает объект графа со списками объектов узлов и ребер.

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Автоматически созданное описание

1. GetAllNodes – Get запрос получения списка объектов узлов графа.

Изображение выглядит как текст, линия, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

1. GetOneNode – Get запрос получения объекта узла по его id.

Изображение выглядит как текст, линия, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

1. GetAllEdges и GetOneEdge – Get запросы работающие аналогично get запросам получения узлов.
2. GetNeighbors – Get запрос получения соседей узла графа.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

В заключении можно отметить, что разработка API на Node.js предлагает разработчикам удобное и эффективное средство для создания современных веб-приложений. Благодаря использованию JavaScript как основного языка программирования, Node.js обеспечивает единое окружение для серверной и клиентской части приложения.

Одной из главных преимуществ Node.js является его асинхронная и неблокирующая архитектура, которая позволяет обрабатывать большое количество запросов одновременно и эффективно использовать ресурсы сервера. Это делает его идеальным выбором для создания масштабируемых и высокопроизводительных API.

Node.js также обладает обширной экосистемой пакетов (npm), которая предлагает множество готовых решений для обработки запросов, роутинга, авторизации и аутентификации, валидации данных и других задач, которые часто возникают при разработке API.

Кроме того, Node.js позволяет разрабатывать API с использованием современных технологий, таких как Express.js. Эти фреймворки облегчают разработку и поддержку API, упрощая маршрутизацию запросов, обработку ошибок и взаимодействие с базой данных.

В целом, разработка API на Node.js — это мощный и гибкий инструмент, который позволяет создавать современные и высокопроизводительные веб-приложения. Благодаря активной и поддерживаемой сообществом разработчиков, Node.js остается популярным выбором в мире веб-разработки.