Express Package

O que é o Express?

- Express é um framework para Node.js que ajuda a criar servidores web de forma simples e rápida.
- Facilita a criação de APIs e sites.
- Simplifica isso com abstrações prontas, deixando o código mais limpo e fácil de entender.

CRIAR PROJETO COM EXPRESS

Criar arquivo app.js com servidor básico

- app.js

```
const express = require('express');
//app é o servidor em si
// configuraremos as rotas e middlewares
const app = express();
```

Middleware **express.json**

- Middleware é um código que executa entre o recebimento da requisição e a resposta.

- É uma etapa do processamento da requisição.

- Pode:
  - Modificar a requisição antes que ela chegue nas rotas;
  - Fazer alguma lógica extra (autenticação, logs, etc);
  - Preparar dados para facilitar o uso dentro das rotas.

express.json()

- Quando um cliente (navegador ou app) envia uma requisição HTTP, às vezes os dados vão no corpo da requisição (body).
- Se esses dados estiverem no formato JSON, o Express, por padrão, não interpreta automaticamente esse JSON.
- express.json() lê esse corpo JSON da requisição, converte para um objeto JavaScript e coloca dentro de req.body.
- Facilita trabalhar com APIs que recebem dados via POST, PUT, PATCH etc.

Como enviar JSON para Express?

- Usar curl no terminal
  - Via linha de comando
  - Permite enviar GET, POST, PUT, DELETE e configurar cabeçalhos, corpo, etc.
- Usar Postman ou REST Client
  - **Postman** (app gratuito para testar APIs) — permite criar requisições POST, configurar o corpo JSON, headers, etc.
- Usar JavaScript fetch no navegador

Sem middleware

```
app.post('/dados', (req, res) => {
  console.log(req.body); // undefined ou vazio
  res.send('Sem middleware, req.body não é interpretado');
});
```

```
const express = require('express');
const app = express();

app.post('/dados', (req, res) => {
  console.log('req.body:', req.body);
  res.send('Recebi os dados, veja o console');
});

app.listen(3000, () => console.log('Servidor rodando na porta 3000'));
```

Testando

- Rode o servidor:

node app.js

- Envie a requisição curl:

curl -X POST http://localhost:3000/dados -H "Content-Type: application/json" -d '{"disciplina":"Web", "Turma":4}'

Com middleware

```
app.use(express.json());


app.post('/dados', (req, res) => {
  console.log(req.body); // aqui aparece o objeto JSON enviado
  res.send('JSON recebido e interpretado');
});
```

Testando

- Rode o servidor:

node app.js

- Envie a requisição curl:

curl -X POST http://localhost:3000/dados -H "Content-Type: application/json" -d "{\"Disciplina\":\"WEB\",\"Turma\":2}"

curl

- Ferramenta de linha de comando que permite enviar requisições HTTP (e outros protocolos) para um servidor.
- -X POST
  - Define o método HTTP da requisição
  - Se não colocar –X, o curl usa o GET por padrão
- http://localhost:3000/dados
  - URL para onde a requisição será enviada
  - /dados -> rota criada no servidor para receber os dados via POST
- -H "Content-Type: application/json“
  - Indica o tipo de dado e que estamos enviando JSON
- -d "{\"Disciplina\":\"WEB\",\"Turma\":2}
  - -d dados e o conteúdo está no formato JSON

Por que usar app.use(express.json())?

- Faz o Express **entender o corpo da requisição** quando o conteúdo for JSON.
- Sem ele, req.body é undefined e não é possível acessar os dados enviados.
- É essencial para APIs que recebem dados via POST, PUT, PATCH.

CRIAR ROTAS GET SIMPLES

O que é uma rota?

- É um caminho/endereço no servidor que responde a requisição HTTP

- No navegador, ao acessar http://localhost:3000/ é feita uma requisição para rota /

O que é HTTP GET e HTTP POST?

- GET
  - Solicita dados do servidor
  - Requisição que **pede** informação
  - Busca uma página, carrega dados
  - Dados enviados na URL (como parâmetros ?chave=valor)
  - Usado para obter dados, não para modificar nada
  - Visível na barra de endereços.

O que é HTTP GET e HTTP POST?

- POST
  - Envia dados para o servidor
  - Requisição que **envia** dados no corpo
  - Enviar formulários, criar novo registro
  - Dados são enviados no corpo da requisição (JSON, formulário)
  - Envia dados e modifica o servidor
  - Precisa de middleware para interpretar o corpo
  - Os dados vão **no corpo (body) da requisição**, não aparecem na URL.

Como criar uma rota GET no Express

app.get (caminho, callback);

- caminho: string com o caminho da rota (ex: '/', '/sobre')
- callback: função que será executada quando a rota for acessada
- O callback recebe dois parâmetros:
  - req (request) — objeto com informações da requisição feita pelo cliente
  - res (response) — objeto que usamos para enviar resposta ao cliente

O que é **req**?

- Representa a requisição feita pelo cliente
- Contém dados como:
  - Parâmetros da URL
  - Query strings (?nome=teste)
  - Cabeçalhos HTTP
  - Corpo da requisição (em POST, PUT, etc — mas não em GET)

O que é **res**?

- Representa a resposta que o servidor vai enviar de volta
- Tem vários métodos para enviar dados
  - res.send(): envia texto simples, HTML, ou outros dados
  - res.json(): envia dados em JSON
  - res.status(): define o código HTTP da resposta

Exemplo – index.js

```
const express = require('express');
const app = express();

// Exemplo 1
app.get('/', (req, res) => {
  res.send('Olá, mundo!');
});

// Exemplo 2: rota /sobre (JSON)
app.get('/sobre', (req, res) => {
  res.json({ projeto: 'Aula_02', disciplina: ‘WEB' });
});

// Servidor ouvindo na porta 3000
app.listen(3000, () => {
  console.log('Servidor rodando em http://localhost:3000');
});
```

No terminal

npm init -y

npm install express

node index.js

- Abra o navegador:

http://localhost:3000/

http://localhost:3000/sobre

- **Criar um projeto Node.js**
- **Instalar pacotes**
  - lodash
  - axios
  - express
- **Consumir uma API pública**
- **Processar dados com Lodash**
  - Ordenar alfabeticamente pelo nome da API, por exemplo
- **Criar servidor com Express**
  - Criar rota **GET** /apis que retorne os dados filtrados e ordenados.
- **Testar no navegador**
  - Verificar se o GET retorna o JSON esperado.