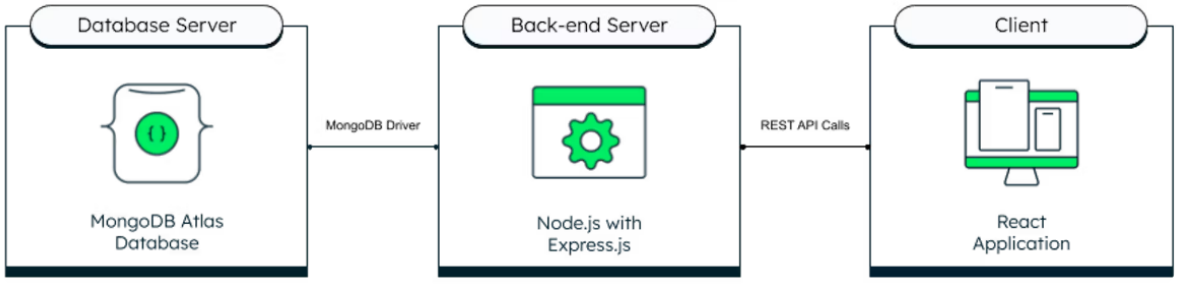
Introdução ao Express - Rotas com Express JS

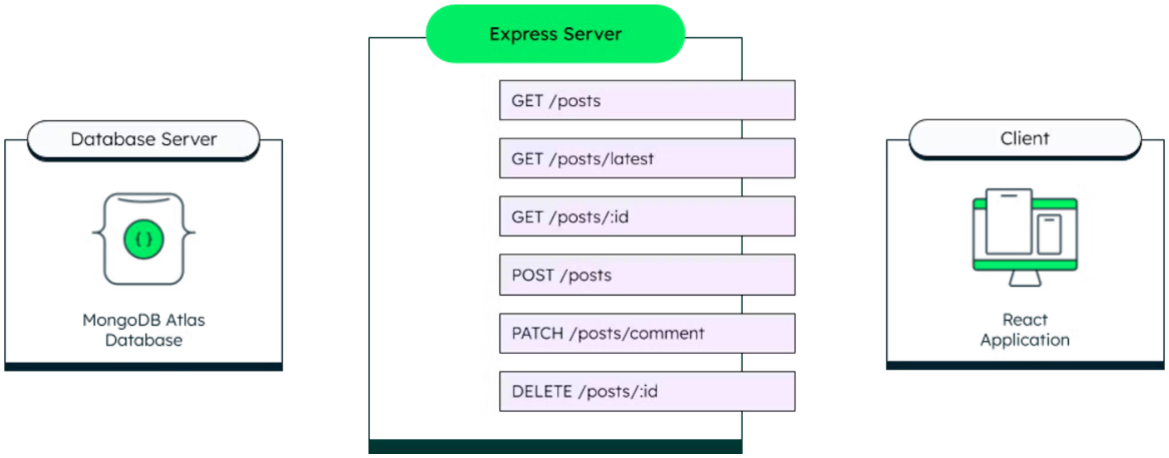
Conhecimentos	6. Interface de desenvolvimento: ferramentas bibliotecas e ambientes de programação (IDE) 7. Desenvolvimento de aplicação web: características da linguagem escolhida comandos e funções orienta
Data da aula	@13 de dezembro de 2023
Tipo	Atividade em Classe Aula Expositiva

O que é Express.js?

O Express.js é um Framework rápido e um dos mais utilizados em conjunto com o Node.js, facilitando no desenvolvimento de aplicações back-end e até, em conjunto com sistemas de templates, aplicações full-stack.



O Express facilita ao criar rotas para aplicações web, principalmente para as requisições HTTP.



Configurações Iniciais

- Configurar ambiente Node.js

```
$ npm init
```

- Instalar o Express com o NPM

```
$ npm install express --save
```

- Importar o Express e iniciar o servidor na nossa aplicação

```
// Importando o Express
import express from 'express';

// Criando a função
const app = express();

.
.
.
// Aplicação
.
.
.

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

OBS: Para utilizar o import express, que é o padrão mais atual, temos que especificar isso no arquivo package-json, adicionado: "type": "module", logo o arquivo fica:

```
{
  "name": "express-routes",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
```

```
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

Rota com Requisição GET

Dentro do arquivo principal .js, index.js, por exemplo, precisamos criar a **rota** raiz, pra isso vamos utilizar o método (ou verbo) GET, caso contrário ao entrarmos em localhost:3000 (endereço de nosso servidor local), vamos obter a seguinte mensagem no navegador: **Cannot GET /.**

```
import Express from 'Express';

const app = Express();

app.get('/', (req, res) =>
res.send("<h3>Rotas no Express</h3><p>Rota '/'")
);

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

Podemos criar outras rotas, como por exemplo:

```
import Express from 'Express';

const app = Express();

app.get('/', (req, res) =>
res.send("<h3>Rotas no Express</h3><p>Rota '/'")
);

app.get('/sobre', (req, res) =>
res.send("<h3>Rotas no Express</h3><p>Vamos aprender a utilizar Rotas com Express")
);

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

Rota com Requisição POST

Antes de tudo, vamos criar um array

```
var alunos = ['gabriel', 'beatriz'];
```

Em seguida, vamos criar uma rota GET para consultar os dados desse array

```
app.get('/students/:id', (req, res) => {
    let id = req.params.id;
```

```
    return res.json([alunos[id]])
});
```

Se fizer uma consulta passando o índice 0, tem-se o seguinte retorno:

```
['gabriel']
```

Agora vamos cadastrar novos alunos utilizando o verbo POST. O Express podemos facilmente utilizar junto ao método `app.post()`. Então vamos criar uma rota e enviar o nome do novo carro pela requisição post da seguinte forma:

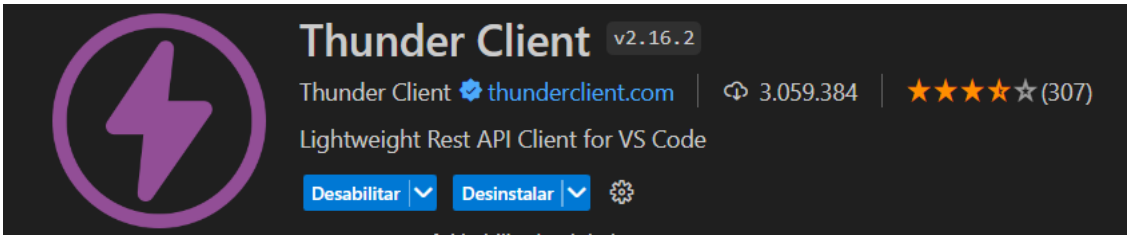
```
.
.
.
var alunos = ['gabriel', 'beatriz'];
app.use(Express.urlencoded({ extended: true }));
.
.
.
```

OBS: perceba que adicionamos a linha `app.use(Express.urlencoded({ extended: true }));`. Como estamos trabalhando com JSON, precisamos fazer o parsing do conteúdo que recebemos nas requisições. Para isso utilizamos o urlenconded, e no caso o extended: true, para selecionar a biblioteca compatível com JSON. Para mais informações sobre a função `urlenconded();`

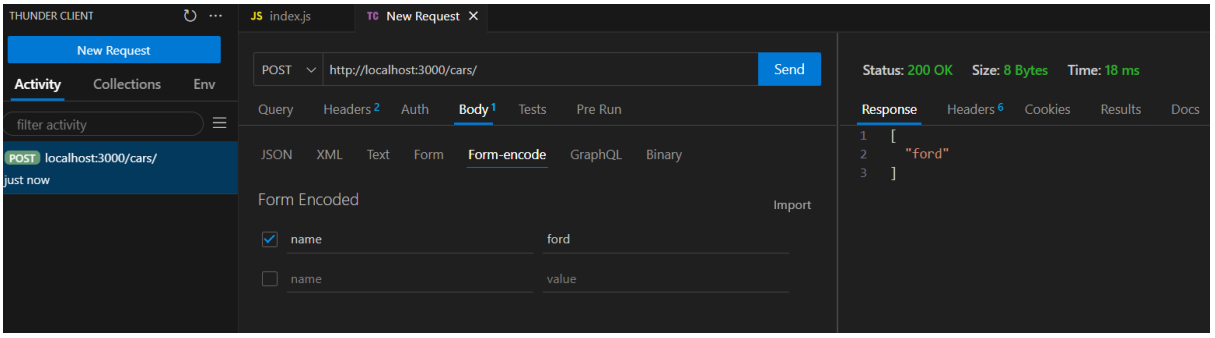
E então, utilizar o método `app.post();`.

```
app.post('/students/', (req, res) => {
    let name = req.body.name;
    alunos[(alunos.length)] = name;
    return res.json([alunos[(alunos.length - 1)]]);
});
```

Instalando a Extensão Thunder Client



A interface é parecida com a que se segue:

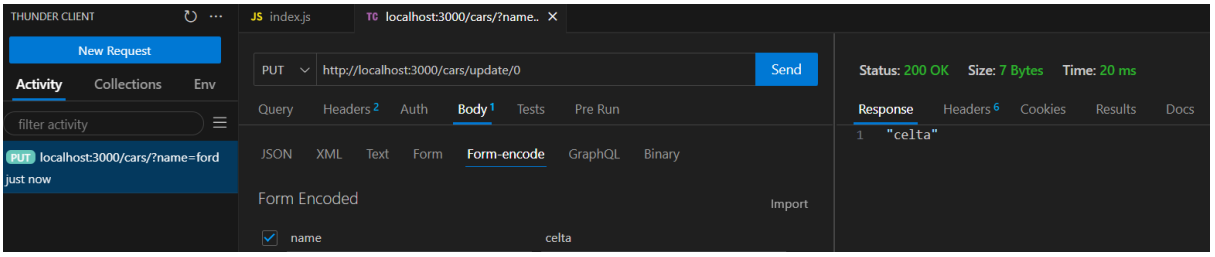


Rota com Requisição PUT

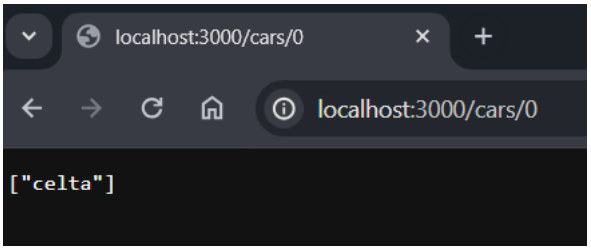
Podemos criar também uma rota com express para atualizar os dados da nossa aplicação, para isso podemos utilizar a rota junto ao método `app.put()`. A requisição PUT segue o mesmo conceito da requisição POST, com a diferença que vamos atualizar uma informação.

```
app.put('/students/update/:id', (req, res) => {
    let name = req.body.name;
    alunos[req.params.id] = name;
    return res.json(alunos[req.params.id]);
});
```

No Thunder Cliente, a resposta é como se segue:



Com o método GET, podemos observar a mudança no próprio navegador:

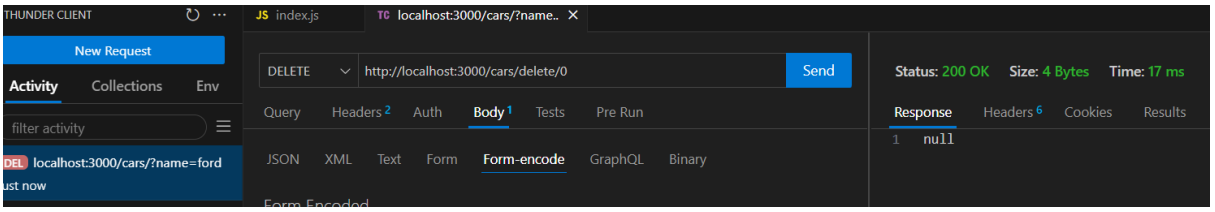


Rota com requisição DELETE

Agora, vamos criar uma rota para deletar algum dado da nossa aplicação, para isso vamos utilizar o método `app.delete();`

```
app.delete('/cars/delete/:id', (req, res) => {
    let id = req.params.id;
    carros[id] = null; //deletar item
    return res.json(carros[id]);
});
```

No Thunder Client:



Atividade

1. Crie um novo projeto
2. Configure com a instalação do npm, express e nodemon
3. Crie um array com 10 itens para salvar os dados com uma aplicação específica (alunos, carros, filmes)
4. Configure uma rota GET de inicialização (Hello World da sua aplicação)
5. Configure uma segunda rota GET para receber o valor de um item específico dado o índice como parâmetro
6. Configure uma rota POST para adicionar valores no array. Adicione 5 valores
7. Configure uma rota PUT para atualizar os três primeiros itens
8. Configure uma rota DELETE para apagar os 2 últimos itens