

# Api localhost Utilizando Node.JS & Express

## Criando o projeto

```
npm init -y
```

## Instalando as dependências do projeto

```
npm install express nodemon
```

npm: nodemon

Simple monitor script for use during development of a Node.js app.. Latest version: 2.0.22, last published: 2 months ago. Start using nodemon in your project by running `npm i nodemon`.

<https://www.npmjs.com/package/nodemon>



npm: express

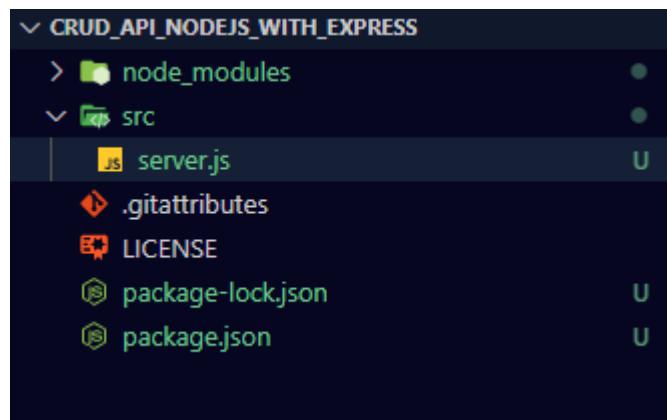
Fast, unopinionated, minimalist web framework. Latest version: 4.18.2, last published: 7 months ago. Start using express in your project by running `npm i express`. There are 71972 other

<https://www.npmjs.com/package/express>



## Criando o Arquivo principal

Após a instalação das dependências, você deverá criar uma pasta na raiz do projeto chamada src, onde deverá conter o arquivo server.js



## Criando a estrutura inicial do servidor

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

const main = ()=>{
    const app = express();
    const port = 3000;

    app.listen(port, ()=> console.log('started server - localhost:3000'))
}

main();
```

## Iniciando o servidor

```
{
  "name": "crud_api_nodejs_with_express",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon src/server.js"
  },
}
```

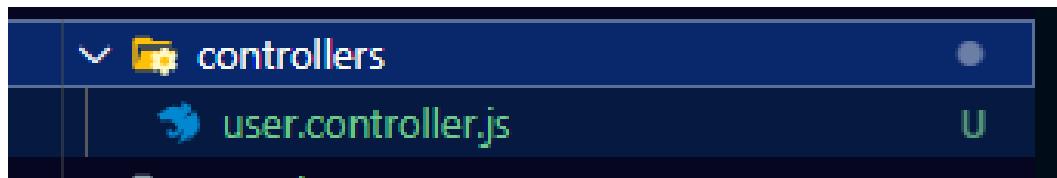
```
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
  "express": "^4.18.2",
  "nodemon": "^2.0.22"
}
}
```

Ao rodar no terminal o seguinte comando [npm start], ele irá startar o servidor na porta 3000, por conta da seguinte linha do arquivo acima

```
"start": "nodemon src/server.js"
```

## Criando a Controller de manipulação de dados

Na raiz do projeto iremos criar a pasta controller/user.controller.js



```
const { userDataRepository } = require("../repositorys/users.data.repository");

class UserController{

  getUsers(request, response){
    return response.json(userDataRepository.usersData);
  }

  putUsers(request, response){
    const {name} = request.body;
    const newUser = {
      id: userDataRepository.usersData.length + 1,
      name,
    };

    userDataRepository.usersData.push(newUser);

    return response.json(newUser);
  }

  deleteUsers(request, response){

    let jsonApiMessage = [];
  }
}
```

```

try {
    const {id} = request.params;

    console.log(id);

    const getAllUsers = userDataRepository.userData;

    userDataRepository.userData = getAllUsers.filter((item) => item.id !== id);

    jsonApiMessage = {
        "code" : "200",
        "message" : `user ${id} deleted`
    }
}

} catch (error) {
    jsonApiMessage = {
        "code" : "500",
        "message" : "Internal Server Error"
    }
}

return response.json(jsonApiMessage);
}

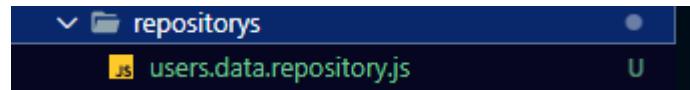
}

const userController = new UserController();

module.exports = {
    userController,
}

```

## Agora crie outra pasta na raiz do projeto, relacionado ao repositório



e para que iremos utilizar? Nesse arquivo iremos criar um variável do tipo lista, com dados pré setados, e com isso já iremos iniciar a nossa api com dados.

```

let userData = [];

class UserDataRepository{
    userData = [
        {
            "id" : 1,
            "name": "Miguel Henrique"
        },
        {

```

```

        "id" : 2,
        "name": "Gabriel"
    },
    {
        "id" : 3,
        "name": "Felipe"
    },
    {
        "id" : 4,
        "name": "Carlos"
    }
]

}

const userDataRepository = new UserDataRepository();

module.exports = {
    userDataRepository,
}

```

**Agora iremos criar o arquivo de rotas da api, que irá ficar na raiz do projeto também**



```

const {Router} = require('express');
const { userController } = require('../controllers/user.controller');

const userRouter = Router();

userRouter.use(
    '/users',
    userRouter.get('/all',userController.getUsers),
    userRouter.post('/',userController.putUsers),
    userRouter.delete('/:id',userController.deleteUsers),
)

module.exports = {
    userRouter,
}

```

**E para finalizar iremos atualizar o arquivo server.js, para o express ler o json**

```

const express = require('express');
const { userRouter } = require('./routes/user.router');

```

```

const main = ()=>{
  const app = express();
  const port = 3000;

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

  app.listen(port, ()=> console.log('started server - localhost:3000'))
}

main();

```

## Com isso ao realizar as requisições nos caminhos das rotas utilizando o insomnia

The screenshot shows the Insomnia application interface. At the top, the title bar reads "Insomnia / API - Node & Express Local no DB". The main window has a dark theme. On the left, there's a sidebar with environment selection ("No Environment"), a cookie section, and a list of requests. The list includes a "Delete User By ID" (DELETE), a "Create New User" (POST), and a selected "GetAll Users" (GET). The main panel shows a "Send" button, a status bar indicating "200 OK", "14.0 ms", and "111 B", and a timestamp "Just Now". Below these are tabs for "Body", "Auth", "Query", "Headers", and "Docs". The "Body" tab is active, showing a JSON response with four user objects:

```

1: [
  {
    "id": 1,
    "name": "Miguel Henrique"
  },
  {
    "id": 2,
    "name": "Gabriel"
  },
  {
    "id": 3,
    "name": "Felipe"
  },
  {
    "id": 4,
    "name": "Carlos"
  }
]

```

Below the response, there's a text input field with placeholder "Enter a URL and send to get a response" and a note "Select a body type from above to send data in the body of a request". At the bottom, there's a link "Introduction to Insomnia" and a footer with "Made with ❤ by Kong".

<http://localhost:3000/users/all>

The screenshot shows the Insomnia REST client interface. In the top bar, the title is "Insomnia / API - Node & Express Local no DB". On the right, it says "A. Miguel Pereira". The main area has a "POST" method selected for "http://localhost:3000/users". The "Send" button is purple and says "200 OK". Below the URL, there's a JSON input field containing:

```
1: {  
2:   "name": "Michael"  
3: }
```

The "Preview" tab shows the response: a single user object with id 5 and name "Michael". The "Headers" tab shows the response headers: "Content-Type: application/json", "Content-Length: 25", and "Date: Mon, 10 Dec 2018 14:45:10 GMT". The "Cookies" and "Timeline" tabs are also visible.

<http://localhost:3000/users>

The screenshot shows the Insomnia REST client interface. In the top bar, the title is "Insomnia / API - Node & Express Local no DB". On the right, it says "A. Miguel Pereira". The main area has a "DELETE" method selected for "http://localhost:3000/users/4". The "Send" button is purple and says "200 OK". Below the URL, there's a "Body" dropdown menu. The "Preview" tab shows the response: a success message with code 200 and message "user 4 deleted". The "Headers" tab shows the response headers: "Content-Type: application/json", "Content-Length: 41", and "Date: Mon, 10 Dec 2018 14:45:10 GMT". The "Cookies" and "Timeline" tabs are also visible.

<http://localhost:3000/users/4>