

# Proposta de arquitetura para aplicações em Node.js e express

---

## Pré requisitos

---

Node instalado e configurado na computador.

## Inicialização do projeto

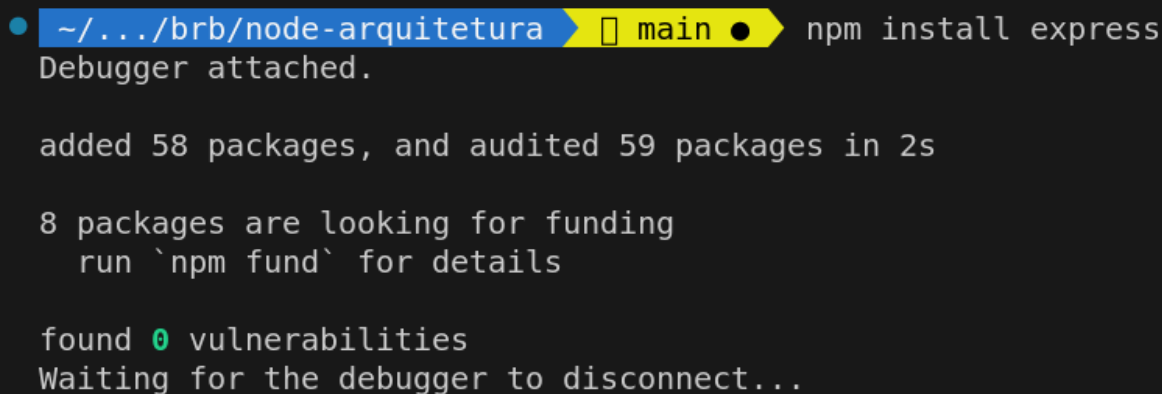
Na sequencia deve-se criar o arquivo package.json, esse é o arquivo de ponto de partida de projetos Node.js. Execute o comando abaixo para inicializar um projeto Node.js.

```
npm init -y
```

## Express

O Express é um framework web para Node.js que simplifica a criação de aplicativos web e APIs. Ele permite definir rotas, manipular solicitações e respostas HTTP, gerenciar sessões e criar middlewares. O Express é amplamente adotado devido à sua simplicidade, flexibilidade e grande comunidade de desenvolvedores. Ele suporta plugins e middlewares, oferecendo recursos adicionais, como autenticação, manipulação de arquivos estáticos e muito mais. Execute o comando abaixo para adicionar o express a um projeto Node.js.

```
npm install express
```

A terminal window with a dark background. The prompt is '~/.../brb/node-arquitetura' followed by a yellow arrow pointing to 'main' and a black circle. The command 'npm install express' has been entered. The output shows 'Debugger attached.', 'added 58 packages, and audited 59 packages in 2s', '8 packages are looking for funding', 'run `npm fund` for details', 'found 0 vulnerabilities', and 'Waiting for the debugger to disconnect...'.

```
• ~/.../brb/node-arquitetura main • npm install express
Debugger attached.

added 58 packages, and audited 59 packages in 2s

8 packages are looking for funding
  run `npm fund` for details

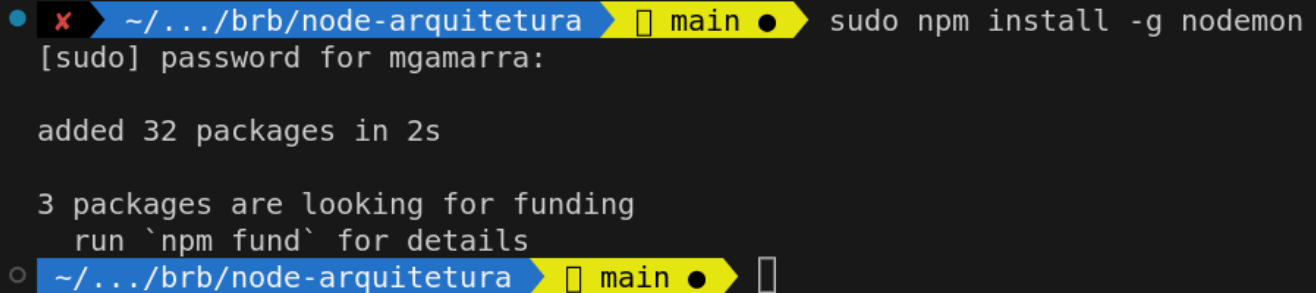
found 0 vulnerabilities
Waiting for the debugger to disconnect...
```

## Nodemon

O Nodemon é uma ferramenta de desenvolvimento para aplicações Node.js. Ele monitora os arquivos do projeto em tempo real e reinicia automaticamente o servidor sempre que ocorre uma alteração nos arquivos. Isso elimina a necessidade de parar e reiniciar manualmente a aplicação a cada alteração no código, tornando o processo de desenvolvimento mais eficiente.

Execute o comando abaixo para instalar o nodemon no projeto.

```
npm install -g nodemon
```



A terminal window with a dark background. The prompt is `~/.brb/node-arquitetura` with a yellow bar and `main` next to it. The command `sudo npm install -g nodemon` is entered. The output shows the password for `mgamarra`, followed by `added 32 packages in 2s`, `3 packages are looking for funding`, and `run 'npm fund' for details`. The prompt returns to `~/.brb/node-arquitetura` with the yellow bar and `main`.

Adicione o comando abaixo na seção scripts do package.json do projeto.

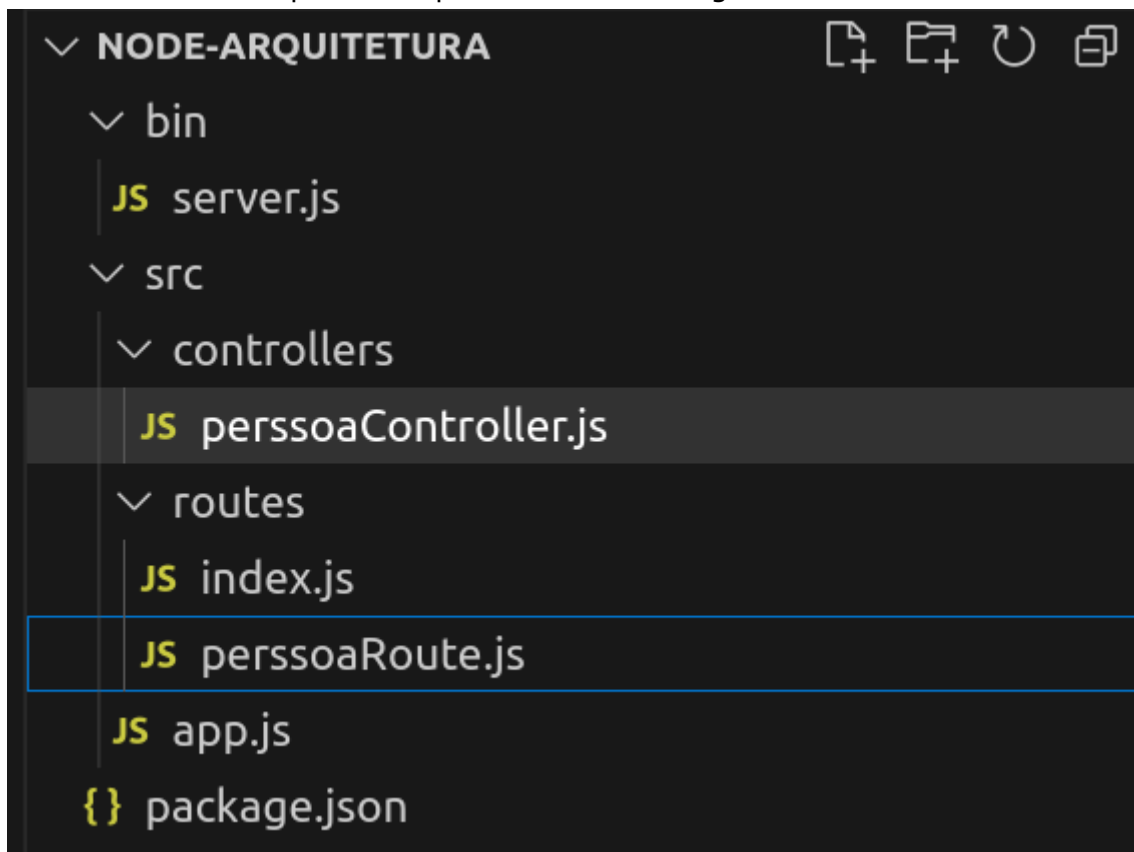
```
start": "nodemon ./bin/server.js"
```

O arquivo package.json deve estar parecido com o exemplo abaixo.

```
{
  "name": "node-arquitetura",
  "version": "1.0.0",
  "description": "#",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon ./bin/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "nodemon": "^2.0.22"
  }
}
```

## Estrutura do projeto

Crie uma estrutura de pastas e arquivos conforme a imagem a baixo:



## Server

Arquivo é responsável pela inicialização e execução da aplicação.

### bin/server.js

```
const app = require('../src/app');
const port = normalizaPort(process.env.PORT || '3000');

function normalizaPort(val) {
  const port = parseInt(val, 10);
  if (isNaN(port)) {
    return val;
  }

  if (port >= 0) {
    return port;
  }

  return false;
}

app.listen(port, function () {
  console.log(`app listening on port ${port}`)
})
```

No código a cima estão definidos os seguintes passos:

- Importação de um modulo a ser criado nos próximos passos;
- Definição da porta em que a aplicação será executada. O valor é lido da variável de ambiente. Caso não esteja definida essa variável é utilizado um valor padrão.
- Passagem para o método `app.listen` da porta que da que a aplicação será executada um `console.log` com ela.

## Controllers

No padrão MVC, os controllers são responsáveis por gerenciar as interações do usuário, atualizar o modelo de dados e atualizar a visualização correspondente.

### **src/controllers/perssoaController.js**

```
exports.get = (req, res, next) => {
  res.status(200).send('Requisição recebida com sucesso!');
};

exports.getById = (req, res, next) => {
  res.status(200).send('Requisição recebida com sucesso!');
};

exports.post = (req, res, next) => {
  res.status(201).send('Requisição recebida com sucesso!');
};

exports.put = (req, res, next) => {
  let id = req.params.id;
  res.status(201).send(`Requisição recebida com sucesso! ${id}`);
};

exports.delete = (req, res, next) => {
  let id = req.params.id;
  res.status(200).send(`Requisição recebida com sucesso! ${id}`);
};
```

## Routes

No padrão MVC, as rotas são responsáveis por mapear as URLs das requisições HTTP para os controladores apropriados.

São apresentados dois arquivos: `index.js` e `personRoute.js`. O arquivo `index.js` seria para passar a versão que esta a nossa API ou para que possamos passar para um balanceador (Load Balancer) verificar se a nossa API está no ar, o `personRoute.js` contem as rotas que iremos utilizar para nossa `PessoaController`.

### **src/routes/index.js**

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

```
const router = express.Router();

router.get('/', function (req, res, next) {
  res.status(200).send({
    title: "Node Express API",
    version: "0.0.1"
  });
});

module.exports = router;
```

### src/routes/perssoaRoute.js

```
const express = require('express');
const router = express.Router();
const controller = require('../controllers/perssoaController')

router.get('/', controller.get);
router.get('/:id', controller.getById);
router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/:id', controller.delete);

module.exports = router;
```

## Configurações

O arquivo app.js é responsável pelas configurações do projeto. Nele devem ser adicionadas as configurações necessárias a execução da aplicação. Ex: configurações de banco de dados, rotas, etc.

Em Node.js, o uso process.env para configuração de aplicações é uma boa prática devido à flexibilidade de que oferece, permitindo que as configurações sejam facilmente ajustadas em diferentes ambientes de execução. Além disso, o process.env proporciona segurança para informações sensíveis, como senhas e chaves de API, evitando que sejam expostas no código-fonte. Com a utilização de variáveis de ambiente, o aplicativo se torna mais portátil e escalável, possibilitando a configuração em diferentes instâncias sem a necessidade de modificar o código. É uma maneira conveniente de ajustar as configurações do aplicativo sem a necessidade de recompilar ou reiniciar o servidor.

### src/app.js

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

//Rotas
const index = require('../routes/index');
const perssoaRoute = require('../routes/perssoaRoute');
```

```
app.use(express.json())
app.use(express.urlencoded({ extended: true }))

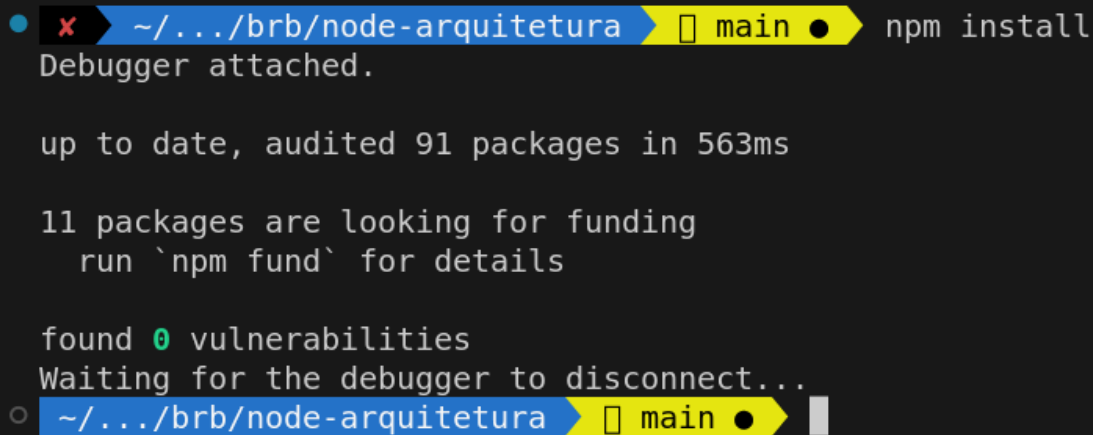
app.use('/', index);
app.use('/perssoa', perssoaRoute);

module.exports = app;
```

## Download de dependências e execução

Para a execução da aplicação, digite o comando `npm install` na console para importar os pacotes necessários para a aplicação e assim que ele finalizar execute o comando `npm start`.

```
npm install
```

A terminal window with a dark background. At the top, there's a status bar with a red 'x' icon, a blue bar containing the path '~/.brb/node-arquitetura', a yellow bar with a folder icon and 'main', and a yellow bar with a circle icon. The terminal text shows the command 'npm install' being executed. The output includes 'Debugger attached.', 'up to date, audited 91 packages in 563ms', '11 packages are looking for funding', 'run `npm fund` for details', and 'found 0 vulnerabilities'. It ends with 'Waiting for the debugger to disconnect...'. At the bottom, there's a second status bar similar to the first one, with a circle icon on the left.

```
● x ~/.brb/node-arquitetura main ● npm install
Debugger attached.

up to date, audited 91 packages in 563ms

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Waiting for the debugger to disconnect...
○ ~/.brb/node-arquitetura main ●
```

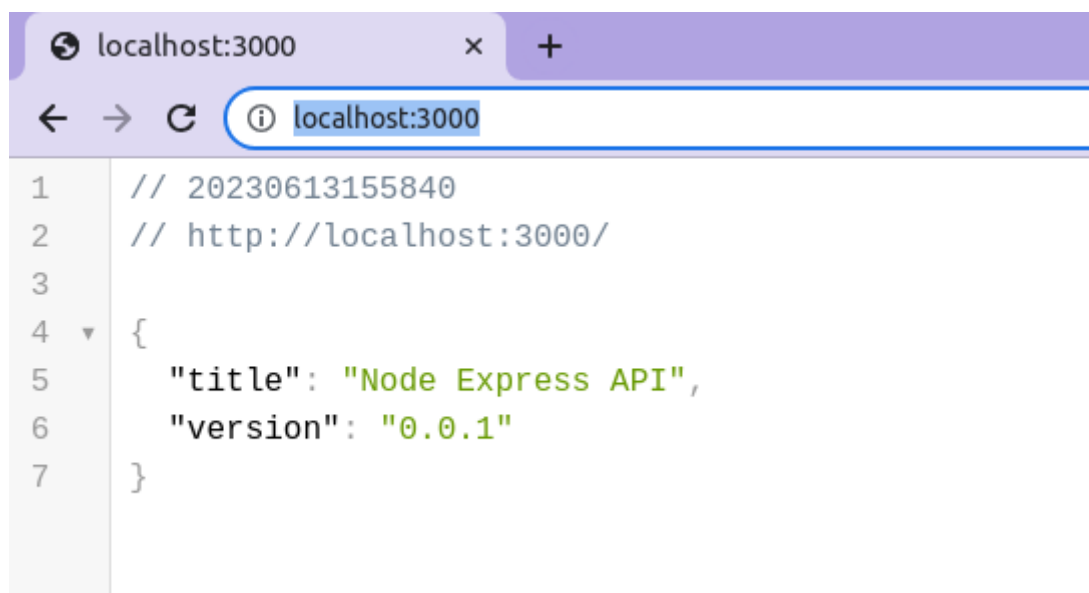
```
npm start
```

```
~/.../brb/node-arquitetura main npm start
Debugger attached.

> arquitetura@1.0.0 start
> nodemon ./bin/server.js

Debugger attached.
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
Debugger attached.
app listening on port 3000
```

Abra no seu navegador o endereço <http://localhost:3000/>. Ele deve apresentar a mensagem a baixo como retorno da nossa rota Index.



```
localhost:3000 x +
< > ↻ ⓘ localhost:3000
1 // 20230613155840
2 // http://localhost:3000/
3
4 {
5   "title": "Node Express API",
6   "version": "0.0.1"
7 }
```