

# Guia de Como Criar uma APP Node.js/Express

prof. Leo Fernandes @ IFAL

---

---

## 1: Usando Node e o NPM

### ### Passo 1: Instalar o Node.js

Certifique-se de que você tenha o Node.js instalado. Você pode verificar isso executando o comando:

```
$ node -v
```

Se não estiver instalado, faça o download e instale o Node.js a partir do site oficial:

<https://nodejs.org/>

### ### Passo 2: Criar uma pasta para o projeto

Crie uma nova pasta para o projeto e navegue até ela:

```
mkdir meu-projeto-node  
cd meu-projeto-node
```

### ### Passo 3: Inicializar o projeto com npm

Inicialize o projeto e crie o arquivo `package.json` executando:

```
npm init -y
```

Isso vai gerar um arquivo `package.json` com as configurações básicas do projeto.

### ### Passo 4: Instalar dependências

Instale as dependências principais: Express e EJS:

```
npm install express ejs
```

Isso instalará o Express e o EJS como motor de template.

### ### Passo 5: Criar a estrutura de pastas

Organize as pastas do projeto da seguinte forma:

```
meu-projeto-node/  
├── views/  
│   └── index.ejs  
├── public/  
│   └── css/  
│       └── styles.css  
├── app.js  
├── package.json  
└── package-lock.json
```

Aqui, a pasta `views` será usada para armazenar os arquivos `.ejs` e a pasta `public` conterá os arquivos estáticos (como CSS, imagens, etc.).

### ### Passo 6: Criar o arquivo `app.js`

Crie o arquivo principal `app.js` e configure o servidor Express:

```
// app.js
const express = require('express');
const app = express();
const port = 3000;

// Configurar EJS como template engine
app.set('view engine', 'ejs');

// Servir arquivos estáticos da pasta 'public'
app.use(express.static('public'));

// Rota principal
app.get('/', (req, res) => {
  res.render('index', { title: 'Minha Aplicação Express' });
});

// Iniciar o servidor
app.listen(port, () => {
  console.log(`Servidor rodando em http://localhost:${port}`);
});
```

### ### Passo 7: Criar o template EJS

Crie o arquivo `index.ejs` dentro da pasta `views/` com o seguinte conteúdo:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
  <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
  <h1>Bem-vindo à <%= title %>!</h1>
  <p>Esta é uma página gerada com EJS.</p>
</body>
</html>
```

### ### Passo 8: Criar o arquivo de estilo CSS (opcional)


Crie o arquivo `styles.css` dentro da pasta `public/css/` com o seguinte conteúdo:

```
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
  color: #333;
  text-align: center;
}
```

### ### Passo 9: Rodar a aplicação

Agora você pode iniciar o servidor executando:

```
node app.js
```



Abra o navegador e acesse `http://localhost:3000` para ver a aplicação em execução.

### ### Passo 10: Adicionar mais rotas (opcional)

Você pode adicionar outras rotas da seguinte forma:

```
app.get('/sobre', (req, res) => {  
  res.render('index', { title: 'Sobre Nós' });  
});
```

Adicione outra página no template EJS para esta rota, se necessário.

---

## 2: Usando Express Generator

O `Express Generator` é uma ferramenta prática para criar rapidamente a estrutura básica de uma aplicação Express. Ele gera automaticamente a estrutura de diretórios e arquivos necessários para iniciar um projeto com Express, incluindo o suporte a EJS (ou qualquer outra) como engine de template.

### ### Passo 1: Instalar o Node.js e o Express Generator

Se ainda não tiver o Node.js instalado, instale a partir do site oficial: <https://nodejs.org/>. Depois, instale o `Express Generator` globalmente com o comando:

```
npm install -g express-generator
```

### ### Passo 2: Criar a aplicação com o Express Generator

Use o `express-generator` para criar a estrutura do projeto. O parâmetro `--view ejs` define o EJS como engine de template:

```
express meu-projeto --view=ejs
```

Esse comando irá criar a pasta `meu-projeto` com a estrutura básica de um projeto Express, configurada para usar EJS.

### ### Passo 3: Navegar até o diretório do projeto

Entre no diretório do projeto recém-criado:

```
cd meu-projeto
```

### ### Passo 4: Instalar dependências

Execute o seguinte comando para instalar as dependências necessárias, que já estão listadas no arquivo `package.json`:

```
npm install
```

### ### Passo 5: Estrutura do projeto gerado

A estrutura básica do projeto gerado será semelhante a esta:

```
meu-projeto/  
|  
├── bin/  
│   └── www  
├── public/  
└── images/
```

```
|
| ├── javascripts/
| ├── stylesheets/
| │   └── style.css
| ├── routes/
| │   ├── index.js
| │   └── users.js
| ├── views/
| │   ├── error.ejs
| │   ├── index.ejs
| │   └── layout.ejs
| ├── app.js
| ├── package.json
| └── package-lock.json
```

- A pasta `views/` contém os arquivos EJS para renderizar as páginas.
- A pasta `public/` contém arquivos estáticos como CSS e JavaScript.
- A pasta `routes/` contém os arquivos de rotas do projeto.
- O arquivo `app.js` é o ponto de entrada da aplicação.

### ### Passo 6: Configurar e entender a aplicação

No arquivo `app.js`, observe que o Express Generator já configurou o EJS como o motor de template:

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
```



Ele também já incluiu algumas rotas básicas para `index` e `users` dentro da pasta `routes/`. Por exemplo, a rota principal (`/`) já renderiza o arquivo `index.ejs`:


```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

app.use('/', indexRouter);
app.use('/users', usersRouter);
```

### ### Passo 7: Personalizar as views

Você pode modificar o arquivo `views/index.ejs` para personalizar a página inicial. Exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
  <link rel="stylesheet" href="/stylesheets/style.css">
</head>
<body>
  <h1>Bem-vindo ao Express com EJS!</h1>
  <p>Esta é uma aplicação gerada com o Express Generator.</p>
</body>
</html>
```



No arquivo de rota `routes/index.js`, o título que será passado para o template `index.ejs` é definido assim:

```
/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Minha Aplicação' });
});
```

### ### Passo 8: Executar a aplicação

Depois de personalizar, você pode iniciar o servidor executando o comando:

```
npm start
```

O servidor estará rodando em `http://localhost:3000`.

### ### Passo 9: Adicionar novas rotas e views (opcional)

Se você quiser adicionar mais rotas, siga este exemplo:

1. Crie um novo arquivo de rota, como `routes/about.js`:

```
var express = require('express');
var router = express.Router();

/* GET about page. */
router.get('/', function(req, res, next) {
  res.render('about', { title: 'Sobre Nós' });
});

module.exports = router;
```

2. No arquivo `app.js`, inclua a nova rota:

```
var aboutRouter = require('./routes/about');
app.use('/about', aboutRouter);
```

3. Crie uma nova view `views/about.ejs` para a página "Sobre":

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
</head>
<body>
  <h1><%= title %></h1>
  <p>Esta é a página sobre nós.</p>
</body>
```

```
</html>
```

Agora, a rota `/about` estará disponível em `http://localhost:3000/about`.

### ### Passo 10: Personalizar estilos e conteúdo

Você pode modificar o arquivo `public/stylesheets/style.css` para personalizar os estilos da sua aplicação, bem como adicionar imagens, arquivos JavaScript e outros recursos na pasta `public/`.

---