



Design Pattern MVC

Prática integradora

Objetivo

Bem, agora temos que reorganizar um pouco nosso código e seguir melhorando a arquitetura do site, agregando padrões do MVC como o Model, View e Controller que nos permite, entre outras coisas, estruturar um site mais limpo.

Nos seguintes desafios, construiremos passo a passo um site baseado em MVC.

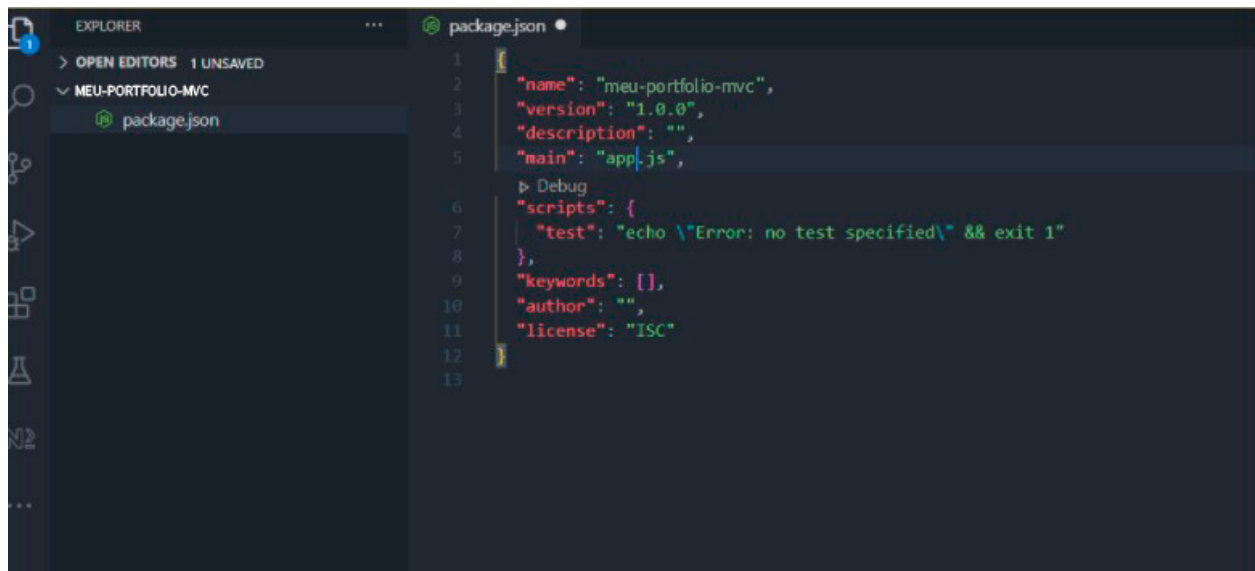
Bom trabalho!

Micro desafio - Passo 1

Criaremos um novo projeto baseado na arquitetura MVC, portanto, em um diretório vazio. Por exemplo: "meu-portfolio-mvc". Executaremos o seguinte comando:

```
C:\meu-portfolio-mvc> npm init -y
```

Lembre-se que este comando inicializa o arquivo **package.json**, ou seja, o arquivo em formato *json* onde especificamos as informações de configuração de nosso projeto.

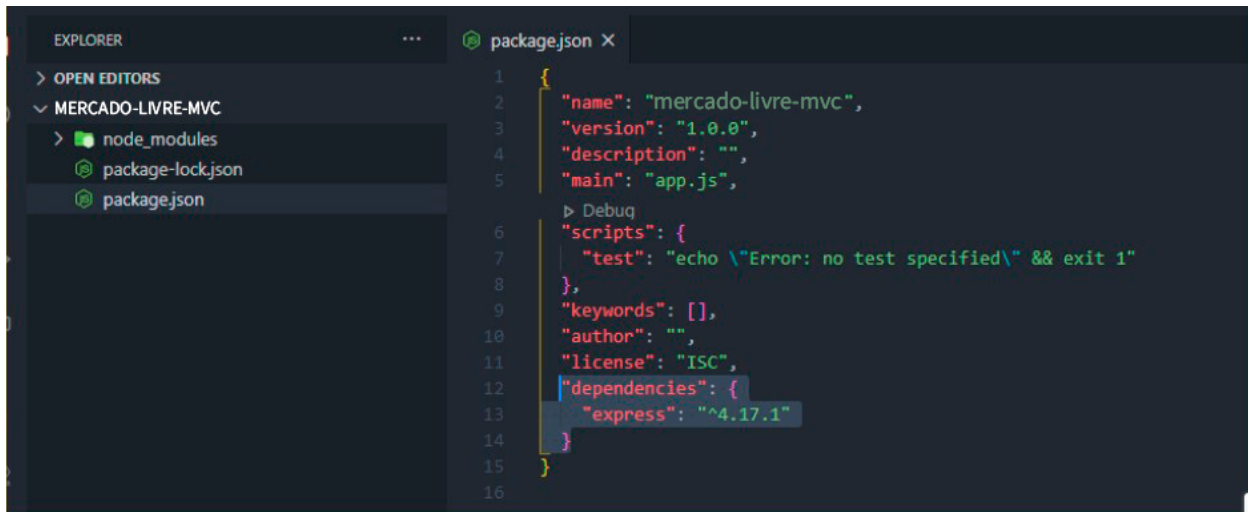


Para nosso *site* vamos definir que o ponto de entrada para a aplicação é o arquivo **app.js**, ao invés de *index.js* como era por padrão 😊.

Micro desafio - Passo 2

```
C:\meu-porfolio-mvc> npm install express
```

Quando executamos este comando, ele gera, por um lado, uma referência dentro de nosso arquivo **package.json** para nos informar que **vamos usar o Express** neste projeto e, por outro lado, cria uma pasta chamada **node_modules**, onde todas as dependências de pacotes de terceiros que precisamos para executar nosso projeto serão armazenadas. Esta pasta não é carregada em Git*.



```
1 {
2   "name": "mercado-livre-mvc",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "express": "^4.17.1"
14  }
15 }
```

(*) Se estamos clonando um projeto do GitHub que já foi criado com o Express, o que precisamos fazer, uma vez terminado o processo de clonagem, é entrarmos na pasta do projeto e executar **npm install**. Este comando, tomando o arquivo package.json como referência, instalará na pasta **node_modules** **tudo** o que precisamos para executar o projeto localmente no nosso computador.

Micro desafio - Passo 3

Dentro da IDE, criamos nosso ponto de acesso, ou seja, o arquivo `app.js`. Nele, indicamos que usaremos o Express com a seguinte linha:

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

Como `require(express)` nos retorna uma função, devemos chamá-la.

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

Agora, na constante **app**, teremos todos os métodos do *framework* disponíveis.

Micro desafio - Passo 4

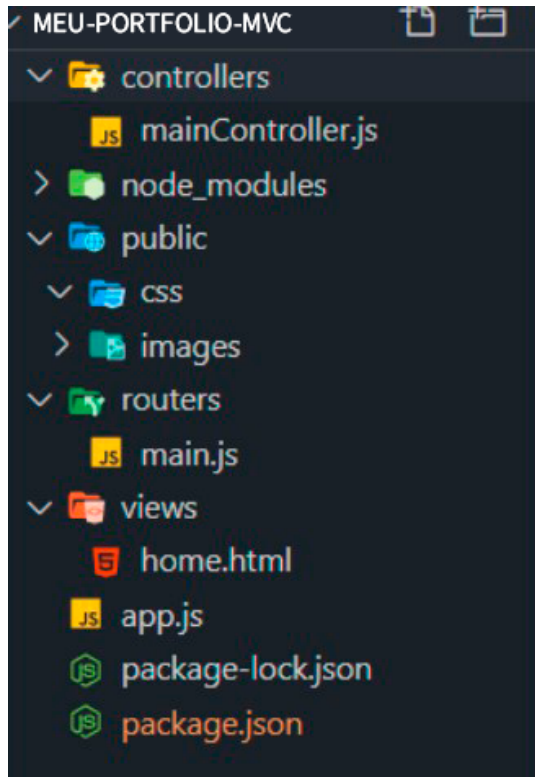
Agora já estamos prontos para configurar nosso servidor.

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



Micro desafio - Passo 5

O próximo passo é ter a estrutura de nosso *site* pronta.:



Os seguintes arquivos foram deixados para este fim:: [images](#), [style.css](#), [home.html](#), [about.html](#) e só temos que colocá-los na pasta correspondente. Quando tivermos a estrutura pronta, criaremos nossos *controllers*, para esta prática criaremos o **mainController.js**:

- Nosso desafio agora é criar a lógica necessária para enviar a *view* **home.html**

quando ela for necessária dentro do *controller*.

Micro desafio - Passo 6

Criar o sistema de roteamento para renderizar a *home.html* quando digitamos *http://localhost:3000/*

Conclusão

O **padrão de projeto MVC** nos permite **manter nossa lógica agrupada de acordo com as responsabilidades**. Por exemplo, toda a lógica que tem a ver com a apresentação vai para a camada de **views** e toda a lógica referente ao negócio de nosso site é colocada na camada de *controllers*. Finalmente, o sistema de roteamento nos permite fornecer uma camada especificamente dedicada ao tratamento de nossos pedidos.

Até a próxima!