03

Gerando o projeto

Transcrição

Vamos dar início ao nosso projeto. No entanto, "dar início" a um projeto pode ser algo um tanto demorado e complicado. Precisamos baixar os scripts que serão usados, decidir a estrutura de pastas e até configurar um servidor web. Imagine fazer todo esse processo antes de escrevermos sequer uma linha de código da nossa aplicação!

Automatizando a construção do projeto com Vue CLI

A boa notícia é que o Vue possui um CLI (command line interface) que nada mais é do que uma ferramenta que automatiza a construção da infraestrutura do projeto. Mas de onde baixaremos o CLI do Vue? Acessaremos algum site? Com certeza não, aliás, baixar dependências de sites é uma prática cada vez mais extinta no mundo front-end. Utilizaremos o gerenciador de pacotes do Node.js, o npm, para baixar o Vue CLI, inclusive todas as dependências da aplicação.

Agora você deve estar entendendo o motivo pelo qual o Node.js é pré-requisito obrigatório para desenvolver em Vue, no entanto essa obrigatoriedade existe apenas em ambiente de desenvolvimento. Quando terminamos uma aplicação, o uso do Node.js não é necessário, pois cabe ao desenvolvedor escolher qual servidor utilizar para hospedá-la, seja ele um servidor Php, Java, .Net entre outros. E claro, nada o impede de querer usar o Node.js como servidor da sua aplicação.

Considerações sobre terminal/prompt de comando

Para utilizarmos o Vue CLI, é necessário um mínimo de traquejo no terminal (Linux/Mac) ou no prompt de comando (Windows) do seu sistema operacional. Não é exclusividade do Vue CLI ser utilizado através do terminal, e um conhecimento básico o ajudará bastante ao longo deste curso e dos demais da Alura que dependem do terminal. Bom, vamos começar os trabalhos.

Instalando o CLI do Vue

Através do terminal e com o Node.js devidamente instalado, vamos instalar o Vue CLI através de um comando do npm. Mas atenção, é necessário ter privilégio de administrador para que o comando funcione corretamente. Dependendo de como seu sistema operacional esta configurado, talvez não seja necessário. Mas se houver algum problema durante a instalação você já sabe qual a possível causa.

No terminal (ou prompt de comando, Se você usa Windows), vamos executar o seguinte comando:

```
npm install -g vue-cli@2.7.0
```

Estamos instalando Vue CLI globalmente através do parâmetro `-g` para que possamos acessar o CLI de qualquer pasta através do terminal. Dentro de instantes tudo será baixado. Assim que a instalação terminar, vamos chamar o CLI e solicitar que seja impresso no console sua versão para sabermos se a instalação foi realizada com sucesso:

```
vue --version
```

Como instalamos a versão `2.7.0`, será essa versão impressa no console. Aliás, peço que usem a mesma versão que estou usando neste treinamento. Ela já foi homologada por mim. É comum o aluno querer atualizar para a versão mais nova, um desejo justo. No entanto, novas versões podem causar bugs que podem atrapalhar o processo de aprendizagem do aluno. Sendo assim, depois que concluir o projeto e tudo estiver funcionando, se quiser atualizar para a versão mais nova e um erro acontecer, você saberá que o erro é da versão mais nova (bug, incompatibilidade) e não do seu código.

Excelente, agora que cliente de linha de comando esta instalado, podemos gerar nosso projeto.

Novo projeto a partir de um template

Através do terminal, vou até a minha área de trabalho (Desktop) para em seguida e gerar o projeto `alurapic` através do comando:

```
vue init webpack-simple alurapic
```

Algumas perguntas serão feitas (nome do projeto, autor, versão) e podemos teclar `ENTER` para todas elas tranquilamente para adotarmos valores padrões. O exemplo acima usa como template o `webpack-simple`. Há outros templates mais simples e mais sofisticados, no entanto, este template é mais do que suficiente para o escopo da nossa aplicação.

O resultado do comando criará a pasta `alurapic`, no entanto, temos apenas a estrutura do projeto e uma lista de todas as suas dependências. Essas dependências não são baixadas automaticamente na construção do projeto. Precisamos entrar na pasta `alurapic` ainda no terminal e executarmos o comando:

```
npm install
```

Este comando baixará todas as dependências listadas no arquivo `alurapic/package.json`. Entenda esse arquivo como um catálogo de todos os recursos que nosso projeto precisa para funcionar. Aliás, como você já deve ter inferido, quem criou esse `package.json` automaticamente para nós foi o CLI.

Levantando um servidor e acessando a aplicação

Quando todas as dependências forem baixadas, já podemos subir nossa aplicação através do comando:

```
npm run dev
```

O comando `npm run dev` executa um script criado em `alurapic/package.json` criado pelo próprio CLI. Há muita coisa envolvida nesse comando, mas o mais importante é saber por agora que ele levanta um servidor local servindo nosso projeto e abrirá automaticamente o navegador padrão do seu sistema operacional apontando para o endereço do projeto no servidor. Fantástico, não? Veremos uma página genérica criada pelo template que utilizamos ao criar nosso projeto.

Com o projeto criado, vamos verificar sua estrutura.