**(Obrigatório) Baixando o projeto e instalação do Node.js**

[Sugerir melhoria](https://cursos.alura.com.br/course/angularjs-mvc/task/9014)

**Preparando o ambiente**

Nosso foco é aprender Angular, porém alguns recursos do framework necessitam de um servidor web rodando localmente em sua máquina. Para que você não perca o foco do Angular e não caia em questões de infraestrutura que dizem respeito a um servidor web, disponibilizamos o projeto [alurapic (baixe agora)](https://s3.amazonaws.com/caelum-online-public/angular-1/stages/01-alurapic.zip) com tudo necessário para subir um servidor web localmente, inclusive com os arquivos do angular já baixados.

**É importante destacar que o uso do projeto inicial alurapic não é opcional**, pois ele já possui registrado todos os endpoints que serão consumidos pela nossa aplicação em Angular. Porém, para que o servidor funcione, é necessário ter o Node.js instalado em sua máquina.

O [Node.js](https://nodejs.org/) é um ambiente JavaScript multiplataforma disponível para Linux, Mac e Windows. Para instalá-lo, siga as instruções abaixo referentes a sua plataforma (versão 4.X):

**Linux (Ubuntu)**

No Ubuntu, através do terminal (permissão de administrador necessária) execute o comando abaixo:

sudo apt-get install -y nodejs

**ATENÇÃO:** em algumas distribuições Linux, pode haver um conflito de nomes quando o Node é instalado pelo apt-get. Neste caso específico, no lugar do binário ser node, ele passa a se chamar **nodejs**. Isso gera problemas, pois a instrução **npm start** não funcionará, pois ela procura o binário node e não nodejs. Para resolver, use a seguinte instrução no terminal para subir o servidor:

nodejs server

Ë uma pena haver essa discrepância, mas fica aqui essa dica!

**Windows**

Baixe o instalador clicando no grande botão install diretamente da página do [Node.js](https://nodejs.org/). Durante a instalação, você apenas clicará botões para continuar o assistente. Não troque a pasta padrão do Node.js durante a instalação a não ser que você saiba exatamente o que está fazendo.

**MAC**

O [homebrew](http://brew.sh/) é a maneira mais recomendada para instalar o Node.js em sua máquina, através do comando:

brew update

brew install node

Não usa homebrew? Sem problema, baixe o instalador clicando no grande botão install diretamente da página do [Node.js](https://nodejs.org/).

**Rodando o servidor**

Depois do Node.js ter sido instalado, **dentro da pasta do projeto alurapic** que você descompactou anteriormente, busque todas as dependências do projeto através do seu terminal (prompt de comando, no caso do Windows) favorito com o comando.

npm install

**ATENÇÃO USUÁRIOS DE WINDOWS**: se por acaso mensagem de erro serem exibidas, procure pelo texto **npm ERR! self signed certificate**. Se ele existir, isso indica um problema no certificado do seu roteador (proxy). Não se preocupe, basta rodar o comando no terminal **npm set strict-ssl false** que resolvera este problema.

Em menos de um minuto, todas as dependências para rodar o servidor terão sido baixadas. Para subi-lo utilizamos o comando:

npm start

Repare que seu terminal ficará aguardando indefinidamente, sinal de que o servidor está escutando. Agora é só abrir o navegador no endereço http://localhost:3000. Uma página de boas-vindas será exibida.

**Curiosidade**

O projeto Alurapic utiliza [Express](http://expressjs.com/) para criar endpoints REST e o [NeDB](https://github.com/louischatriot/nedb), um banco de dados totalmente feito em Node.js e que não necessita qualquer configuração especial.

# (Obrigatório) Failed to parse SourceMap

[Sugerir melhoria](https://cursos.alura.com.br/course/angularjs-mvc/task/9021)

Se no Chrome, durante o treinamento o console do navegador exibir a mensagem

Failed to parse SourceMap: http://localhost:3000/js/lib/angular.min.js.map

Não se preocupe, você não fez nada de errado. As últimas versões do Chrome podem exibir essa mensagem quando arquivos minificados forem carregados. No caso do projeto do Angular que você baixou, esses arquivos não existem. Aliás, para que servem esses arquivos? Quando recebemos uma mensagem de erro do um script minificado, muitas vezes temos dificuldade em analisar a linha que disparou o erro porque o processo de minificação altera e trunca muitos nomes. O que o source map faz é permitir que seja usado o arquivo minificado, mas que a mensagem do debugger do navegador exiba algo mais legível. Como não somos mantenedores do projeto do Angular, a ausência desse arquivo não nos causará problemas.