***Curso de NPM***

***( Node pack Manager )***

Para começarmos a entender **NPM** precisamos instalar o Node.Js e entender sobre ele, pois o NPM faz parte deste.

***Node.js:***

Node.js é um ambiente de execução da linguagem de programação JavaScript que roda no lado do servidor. Ele permite que os desenvolvedores criem suas aplicações backend fora do navegador, possibilitando a criação de APIs e oferecendo todas as funcionalidades que um servidor backend pode fornecer ao usuário, como autenticação e até mesmo segurança.

Dentro do Node.js, temos um facilitador de aplicações chamado NPM, que é um Gerenciador de Pacotes. O NPM pode conter diversos tipos de bibliotecas e ferramentas que simplificam o desenvolvimento de aplicações.

Para começar, siga o link abaixo para instalar o Node.js e acessar sua documentação. Leia e saiba mais:

**Node.js →** <https://nodejs.org/pt>

**Instalando o NPM.**

Por fim agora para instalar o nosso tão querido Gerenciador de Pacotes, navegamos até a pasta do projeto no terminal e digitamos os seguintes comandos:

**npm init** → Comando utilizado para iniciar um projeto.

Aparecerá para você após isso o seguinte formulário:  
  
**package name:** nome\_do\_projeto

**version:** versão\_do\_projeto

**description:** descrição\_do\_projeto

**entry point:** arquivo\_principal

**test command:** arquivo\_de\_teste

**git repository:** repositório\_no\_git

**keywords:** palavra\_chave\_do\_projeto

**author:** nome\_do\_autor

**license:** nome\_da\_liçensa

**Is this OK?** Digite “yes” se estiver tudo certo e “no” para cancelar a operação.

Caso não queira digitar todos esses comandos é possível criar o projeto sem responder as perguntas usando comando:

**npm init -y**

Esse é o mesmo comando anterior porém ele digita apenas **yes** para todas as perguntas.

Com o projeto criado será criado um arquivo na pasta do projeto chamado **package.json** que é um arquivo no formato **JSON** (JavaScript Object Notation) que é geralmente usado para transmitir dados, até mesmo para envio de requisições HTTP. Porém aqui ele será um facilitador para gerenciar e organizar o projeto. Aqui podemos alterar o código, criar scripts, dependencias e muito mais.

No arquivo bloco de **scripts** podemos definir comando que vamos utilizar em nosso projeto. Por exemplo se você quiser usar o comando que é chato as vezes, você pode colocar ele no bloco de scripts e digitar o comando:

**npm run** key\_do\_script

**Dependências.**

Como exemplo de dependências vamos instalar o **React** e o **Next** e vamos prestar atenção como o nosso package.json vai se comportar. Para isso digite o seguinte comando.

**npm install react**

**npm intall next**

Dessa maneira é possível instalar qualquer pacote em nosso projeto.

Vale ressaltar que temos agora outros arquivos em nosso projeto que foram instalados automaticamente. Um deles se chama **node\_modules**.

Essa é uma das pastas que você vai ignorar em seu **.gitignore**, pois essa pasta é extremamente grande; é nesta pasta que está instalado todos os seus pacotes de dependências.

Já o arquivo chamado **package-lock.json** é um arquivo que não deve ser alterado; o propósito desse arquivo é instalar todas as dependências do projeto em sistemas cujo o projeto será clonado. Alterar esse arquivo pode atrapalhar na instalação das dependências quando o projeto for clonado.

Agora para remover pacotes do nosso projeto usaremos o comando:

**npm remove** nome\_do\_pacote

**Exemplo:**

**npm remove react**

**npm remove next**

Assim o pacote do React e do Next serão removidos completamente do projeto. Você vai perceber que eles irão sumir das dependências do package.json.

O node\_modules continuará no projeto porém suas pastas não terão nenhum arquivo.

Outra maneira de remover os pacotes é remove-los manualmente. Basta retirar a dependência no package.json e digitar o comando:

**npm install**

Este comando lerá todas depedências do package.json e atualizará o projeto.

**Dependências Locais.**

Também podemos criar dependências locais, ou seja... Dependências que vão rodar somente na máquina específica. E para fazer isso é super fácil, basta apenas seguir a seguinte linha de comandos...

**npm install** nome\_do\_pacote **--save-dev**

E para remover seguimos a mesma lógica porém como no novo comando...

**npm remove** nome\_do\_pacote **--save-dev**

**Yarn.**

O **Yarn** é outro Gerenciador de Pacotes que segue a mesma linha de raciocínio que o **NPM**, e conforme o **NPM** evolui o **Yarn** evolui também. E para instalarmos ele é bem fácil. Basta começarmos instando o **NPM** e utilizarmos o seguinte comando...

**npm install --global yarn**

Esse tipo de instalação é uma instalação global no Gerenciador de Pacotes NPM.

É possível também instalar o Yarn sem instalar necessariamente o NPM, usando o seguinte comando...

**yarn init -2**

Agora, ao contrário do **NPM** onde tínhamos o **package-lock.json** vamos ter o arquivo **yarn.lock**, e assim como no **NPM** não devemos alterar nada nesse arquivo.

Para instalar um pacote como dependencias globais no Yarn utilizamos o seguinte comando...

**yarn add react**

**yarn add next**

Para instalar um pacote como dependência local no Yarn utilizamos o seguinte comando...

**yarn add react -D**

**yarn add next -D**

Para remover arquivos no Yarn utilizamos o seguinte comando...

**yarn remove react**

**yarn remove next**

Agora basta você escolher um dos Gerenciadores de Pacotes e subir seus projetos.

**ATENÇÃO:** E caso esteja trabalhando com um projeto já pronto continue utilizando o mesmo Gerenciador de Pacotes que já está sendo usado para não haver conflitos.