

//ARQUIVOS PACKAGE JSON:

//Os arquivos package JSON são arquivos criados por através do Node com o objetivo de descrever um projeto criado em Node quanto também de atribuir alguns tipos de comportamento para o nosso projeto, como bibliotecas utilizadas, como controle de versionamento da biblioteca, palavras chave do projeto, quem é o autor do projeto, licença e outras coisas do tipo.

//Outra grande vantagem dos arquivos package.json é a atribuição de dependências dentro do arquivo. Pois o nosso arquivo package.json irá acompanhar o nosso projeto para onde quer que ele for, assim quando vários desenvolvedores estão trabalhando num projeto que precisa de determinadas bibliotecas, o nosso package.json irá providenciar que essas bibliotecas sejam baixadas, e ainda melhor, podemos até escolher a versão de biblioteca, para evitar conflitos de versionamento caso um desenvolvedor baixe uma versão diferente da biblioteca que usamos atualmente no projeto.

//Existem formas diferentes de se criar um arquivo package JSON:

//CRIANDO ARQUIVO DE FORMA CONVENCIONAL:

//Para criar um arquivo package.json convencional com todas as alternativas básicas, basta o comando abaixo...

// npm init -y

// Esse comando quer dizer que estamos criando um arquivo normal e respondendo yes a todas as perguntas do node sobre os atributos do arquivo package.json...

//CRIANDO ARQUIVO PACKAGE.JSON DETALHADAMENTE:

//Podemos gerar um package.json colocando detalhadamente tudo o que queremos no arquivo, nesse método o Node para a execução da criação do arquivo em partes específicas para nos perguntar coisas como nome do arquivo, versão, descrição, arquivo de entrada do projeto, teste de comando, repositório no git, palavras chave para chamar pelo arquivo, autor e licença.

//Para gerarmos uma criação de package.json detalhada usamos o comando:

// npm init

//INCLUÍDO DEPENDÊNCIA A UM ARQUIVO PACKAGE.JSON:

//Para que possamos incluir uma biblioteca dentro do arquivo package.json podemos usar o comando --save quando formos baixar uma determinada biblioteca.

//OBS: Sempre certifique-se de que o arquivo package.json já existe dentro do seu projeto, e certifique-se de que o path esteja encaminhado para a biblioteca correta.

//Podemos incluir uma biblioteca assim:

```
//      npm i --save bibliotecaTal
```

// Depois de fazer o download da biblioteca, perceba que a biblioteca foi inclusa em dependences no arquivo "package.json"

//INCLUINDO DEPENDENCIA SOMENTE EM PERÍODO DE DESENVOLVIMENTO:

//Podemos optar por incluir uma dependência que será utilizada somente durante o período de desenvolvimento do sistema, por exemplo dependências de testes, para isso podemos usar o comando "--save-dev", ele indica que após o desenvolvimento, quando o programa estiver rodando normal, essa dependência não será mais necessária.

//Usamos esse comando assim:

```
//      npm i --save-dev bibliotecaTal
```

//CONTROLE DE VERSIONAMENTO SEMÂNTICO DAS BIBLIOTECAS DO NODE:

//Quando baixamos uma biblioteca Node externa no nosso projeto e ela é inclusa nas nossas dependências, podemos observar que por padrão nossas bibliotecas estão configuradas para aceitar atualizações de versionamentos. Porém existem 3 tipos de versionamento diferentes nas bibliotecas:

```
/*      * MAJOR: Quando as mudanças geram incompatibilidade entre a versão anterior e superior;
      * MINOR: Quando uma biblioteca teve melhorias, mas a compatibilidade entre versão anterior e superior são mantidas;
      * PATCH (ou FIX): Quando uma biblioteca teve correções, mas a compatibilidade entre versão anterior e superior são mantidas;
```

Por padrão os números seguem a sequência: MAJOR.MINOR.PATCH apresentada em números: 0.0.0

```
*/
```

//VERSIONAMENTO NO MINOR:

//Por padrão, quando incluímos uma biblioteca como dependência no nosso package.json ela permitirá atualizações na MINOR, ou seja, permite que novas melhorias implementadas na biblioteca sejam baixadas. No package.json isso é simbolizado pelo símbolo de "^", por exemplo: "^0.22.0". Quando está assim, queremos dizer que só aceitamos atualizações de bibliotecas, não aceitamos atualizações de API MAJOR, onde as mudanças são incompatíveis, nem correções de bugs pelo FIX.

//VERSIONAMENTO NO PATCH(FIX):

//Podemos mudar o estilo de versionamento de biblioteca para aceitar somente mudanças no "fix", para isso usamos o símbolo de "~", por exemplo: "~0.22.0". Quando deixamos assim, estamos dizendo para o Node que não aceitamos atualizações nem na MAJOR nem na MINOR, somente correções de bug na versão atual da API.

//VERSÃO EXATA:

//Quando desejamos que um projeto pegue somente uma versão exata, que ela não aceite versões atualizadas de forma alguma, basta não colocarmos símbolo nenhum, da seguinte forma: "0.22.0". Melhor ainda, podemos atribuir isso no momento que baixamos a API, por através da flag "-E", essa flag significa que deverá pegar uma versão exata que também precisa ser passada no comando, dessa forma:

```
// npm i --save bibliotecaTal@.17.1 -E
```

// Note que incluímos ao nome da biblioteca o "@" seguido pelo número de versão da biblioteca, obrigatoriamente temos que informar qual é o número de versão, e ao final a flag "-E" (Sempre em maiúsculo) pois ela que vai dizer que o arquivo deve ser o Exato!

//Para ver um exemplo prático disso entre na aula "135-Ex-JSON" na pasta "Exercicios/135-Funcionarios"

ARQUIVO PACKAGE.JSON...

```
{
  "name": "135-funcionarios",
  "version": "1.0.0",
  "description": "Aqui segue uma descrição fictícia para o projeto",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "palavraTal",
    "palavraTal",
    "palavraTal"
  ],
  "author": "Gabriel F. Nobre",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.24.0"
  }
}
```

ARQUIVO PACKAGE-LOCK.JSON...

```
{
  "name": "135-funcionarios",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "axios": {
      "version": "0.24.0",
      "resolved": "https://registry.npmjs.org/axios/-/axios-0.24.0.tgz",
      "integrity": "sha512-Q6cWsys88HoPgAaFAVUb0WpPk008iTeisR9IMqy9G8Ab04NlpVknrnQS03zzF9PGAWg03cglet03VjV/P7VztA==",
      "requires": {
        "follow-redirects": "^1.14.4"
      }
    },
    "follow-redirects": {
      "version": "1.14.5",
      "resolved": "https://registry.npmjs.org/follow-redirects/-/follow-redirects-1.14.5.tgz",
      "integrity": "sha512-w01Y8Ze0DqU17qzw2+5xL5Iy3DyGsq9RNySsq5TJw8Q7471H7QRYPj/61KdVU04hmZTMVIXQ2oVrEP1+K1A=="
    }
  }
}
```