npm

INTRODUÇÃO AO NPM

Introdução ao npm (Node Package Manager)

- npm é gerenciador padrão do Node.js

- É formado por:
  - Registry: onde ficam publicados os pacotes (https://www.npmjs.com).
  - CLI: ferramenta de linha de comando para instalar, remover, atualizar e publicar pacotes.
  - package.json: arquivo que descreve o projeto, dependências e *scripts*.

- O npm é muito mais do que instalar pacotes — ele é uma plataforma para **criar, compartilhar, manter e evoluir software em colaboração**.

O que você pode fazer com o npm?

- Gerenciar e usar pacotes:
  - Adaptar pacotes de código para seus apps ou usar como estão.
  - Baixar ferramentas independentes e usar imediatamente.
  - Executar pacotes sem instalar, usando npx.
- Compartilhar e colaborar:
  - Compartilhar código com qualquer usuário do npm, em qualquer lugar.
  - Restringir o uso do código a desenvolvedores específicos.
  - Criar organizações para coordenar manutenção, desenvolvimento e equipes.
  - Formar times virtuais usando organizações.
- Manter e evoluir projetos:
  - Gerenciar múltiplas versões de código e dependências.
  - Atualizar aplicações facilmente quando o código base é atualizado.
  - Descobrir diferentes maneiras de resolver o mesmo problema.
  - Encontrar outros desenvolvedores trabalhando em problemas e projetos semelhantes.

Documentação

- https://www.npmjs.com
  - Pacotes
  - Avaliação
  - Downloads
  - README
- https://docs.npmjs.com
  - Documentação
  - Guias e comandos

O que é um pacote no npm?

- É um conjunto de código + metadados que descrevem esse código (*package.json*).
- Os metadados incluem:
  - Nome do pacote;
  - Versão;
  - Autor;
  - Licença;
  - Dependência que ele precisa para funcionar;
  - Scripts que podem ser executados;
  - Outras informações.

Pacotes populares

- express
  - Framework minimalista para criar servidores web e APIs em Node.js.
  - Facilita a criação de rotas, middlewares e manipulação de requisições HTTP.

- cors
  - Middleware para habilitar CORS (Cross-Origin Resource Sharing).
  - Permite que seu servidor aceite requisições vindas de outros domínios (útil para APIs).

- dotenv
  - Carrega variáveis de ambiente de um arquivo .env para process.env.
  - Facilita o gerenciamento de configurações sensíveis (senhas, tokens, URLs) fora do código.

Pacotes populares

- axios
  - Cliente HTTP baseado em Promises para fazer requisições web (GET, POST, etc).
  - Facilita consumir APIs externas e fazer chamadas HTTP.

- nodemon
  - Ferramenta que reinicia automaticamente o servidor Node.js quando arquivos são alterados.
  - Acelera o desenvolvimento sem precisar parar e reiniciar manualmente o app.

- bcrypt
  - Biblioteca para hashing e comparação segura de senhas.
  - Usada para proteger senhas de usuários armazenadas em banco de dados.

Pacotes populares

- jsonwebtoken
  - Biblioteca para criar e verificar JSON Web Tokens (JWT).
  - Autenticação e autorização usando tokens em APIs.

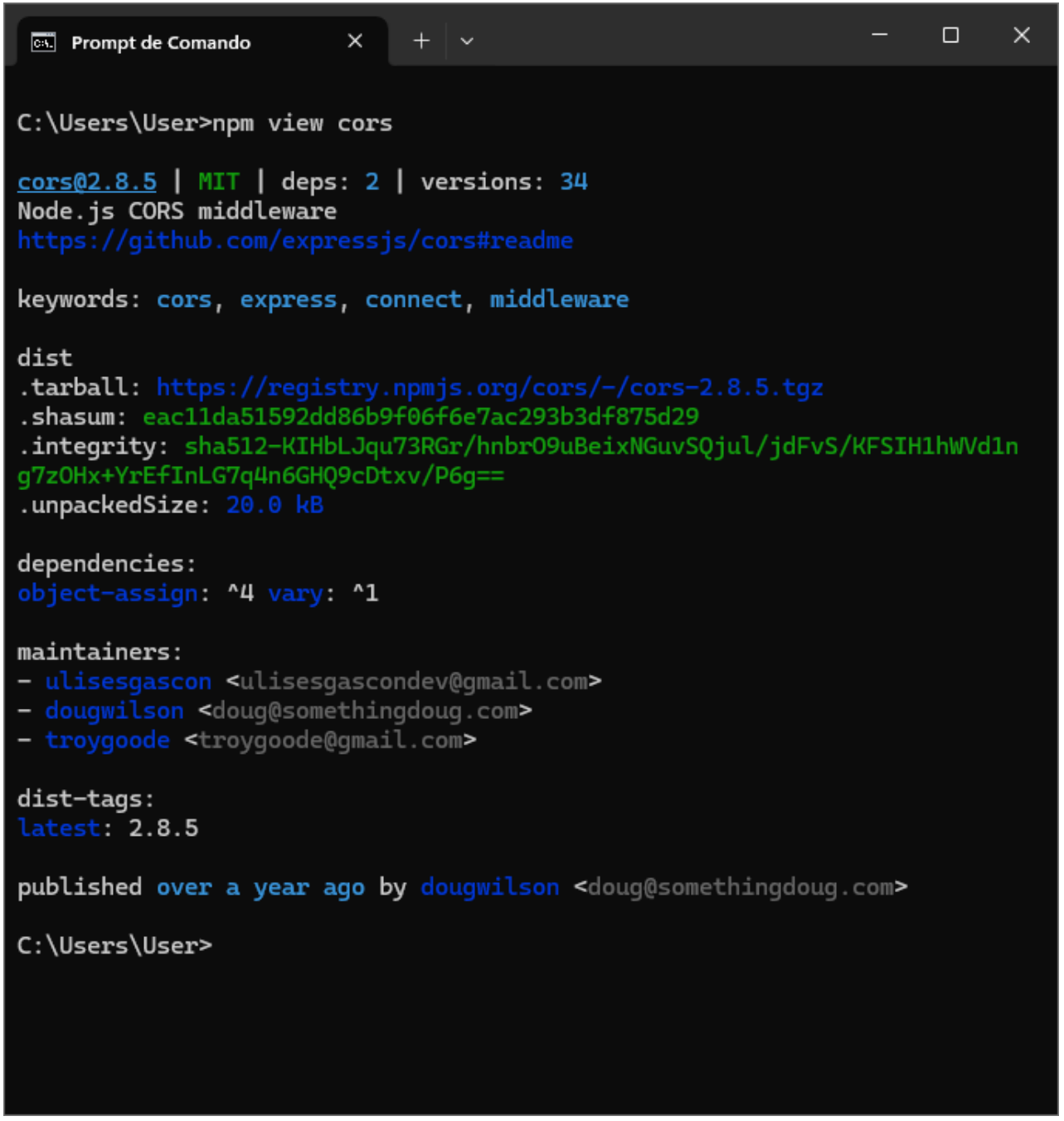
- mongoose
  - ORM (Object-Relational Mapping) para MongoDB em Node.js.
  - Facilita modelar dados e interagir com banco MongoDB usando objetos JS.
- lodash
  - Biblioteca com utilitários para manipulação de arrays, objetos, strings, etc.
  - Facilita operações comuns com dados, tornando o código mais simples e legível.

- Helmet
  - Middleware para configurar cabeçalhos HTTP de segurança.
  - Ajuda a proteger sua aplicação contra algumas vulnerabilidades web conhecidas.

Pacote CORS

https://www.npmjs.com/package/cors

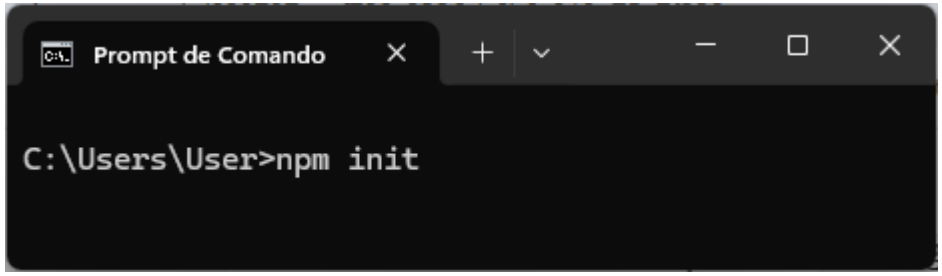
- O CORS é um mecanismo que diz ao navegador que o servidor permite requisições vindas de outros domínios.
  - Configura automaticamente os headers HTTP necessários (Access-Control-Allow-Origin, etc.).
  - Facilita permitir ou restringir quais origens podem acessar sua API.
  - Permite definir métodos e tipos de conteúdo aceitos.
  - Evita que você precise escrever manualmente todo o código de configuração.

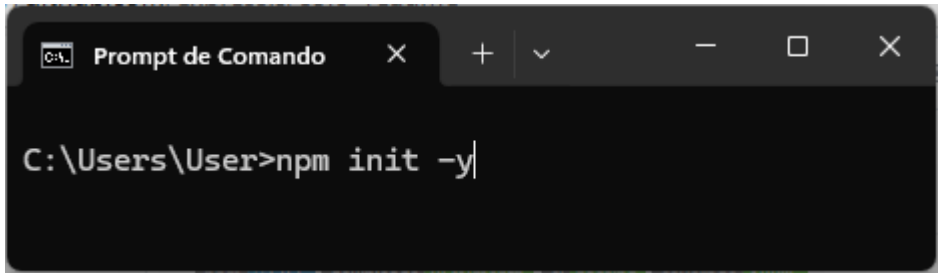

Prompt de Comando
C:\Users\User>npm view cors
cors@2.8.5 | MIT | deps: 2 | versions: 34
Node.js CORS middleware
https://github.com/expressjs/cors#readme
keywords: cors, express, connect, middleware
dist
.tarball: https://registry.npmjs.org/cors/-/cors-2.8.5.tgz
.shasum: eac11da51592dd86b9f06f6e7ac293b3df875d29
.integrity: sha512-KIHbLJqu73RGr/hnbrO9uBeixNGuvSQjul/jdFvS/KFSIH1hWVd1n
g7zOHx+YrEfInLG7q4n6GHQ9cDtxv/P6g==
.unpackedSize: 20.0 kB
dependencies:
object-assign: ^4 vary: ^1
maintainers:
- ulisesgascon <ulisesgascondev@gmail.com>
- dougwilson <doug@somethingdoug.com>
- troygoode <troygoode@gmail.com>
dist-tags:
latest: 2.8.5
published over a year ago by dougwilson <doug@somethingdoug.com>
C:\Users\User>


Prompt de Comando
C:\Users\User>npm docs cors

O que é o package.json ?

- É o arquivo principal que descreve o seu projeto e as dependências que ele precisa.
- Declara o que o projeto usa e scripts.
- Pode ser criado automaticamente com:






{} package.json > ...
1 {
2 "name": "meu-projeto",
3 "version": "1.0.0",
4 "description": "Um projeto de exemplo",
5 "main": "index.js",
Debug
6 "scripts": {
7 "start": "node index.js"
8 },
9 "dependencies": {
10 "express": "^4.19.2"
11 },
12 "devDependencies": {
13 "nodemon": "^3.1.0"
14 }
15 }

package-lock.json

- Criado automaticamente quando pacotes são instalados.
- Registra exatamente a versão instalada de cada pacote e de suas dependências internas;
- Serve para:
  - Garantir que todos usem a mesma versão;

Dependências vs. Dependências de desenvolvimento

- dependencies
  - Necessárias para rodar o app em produção;

  > npm install pacote
- devDependencies
  - Necessárias **apenas** no desenvolvimento (ex.: ferramentas de teste)

  > npm install pacote --save-dev

  > npm install pacote -D

CRIANDO UM PROJETO NODE E USANDO NPM

Criando um projeto

1. Criar a pasta do projeto e entrar nela
2. Criar o arquivo **package.json** básico com **npm init –y**
3. Instalar o pacote **lodash**
    1. npm install lodash
4. **package.json** atualizado
5. Criar o arquivo **index.js** com exemplo usando **lodash**
6. Adicionar script **start** no **package.json**
7. Rodar o script com **npm start**

Criar a pasta do projeto

- Via linha de comando (terminal) ou pela interface gráfica;

  > mkdir meu_projeto

  > cd meu_projeto

- Cada projeto deve ficar em sua própria pasta:
  - Isolamento de arquivos;
  - Facilidade de manutenção;
  - Controle de dependências;
  - Versionamento e Git
  - Portabilidade

Criar o arquivo **package.json**

>npm init –y

- Cria o arquivo **package.json** com as configurações padrão
- Guarda informações do projeto, como nome, versão e dependências

Alterar informações **package.json**

- É possível editar manualmente via editor de texto ou usar comandos **npm**

> npm set init-author-name “José Maria”
> npm pkg set author=“José Maria”
> npm pkg set description=“Projeto 01”
> npm pkg set license=“MIT”
> npm pkg set keywords[0]=npm
> npm pkg set keywords[1]=node
> npm pkg set keywords[2]=aula

Instalar pacote **lodash**

- Esta biblioteca possui várias funções para manipular dados (strings, arrays, objetos, etc).

> npm install lodash

Instalar pacote **lodash**

- Esta biblioteca possui várias funções para manipular dados (strings, arrays, objetos, etc).

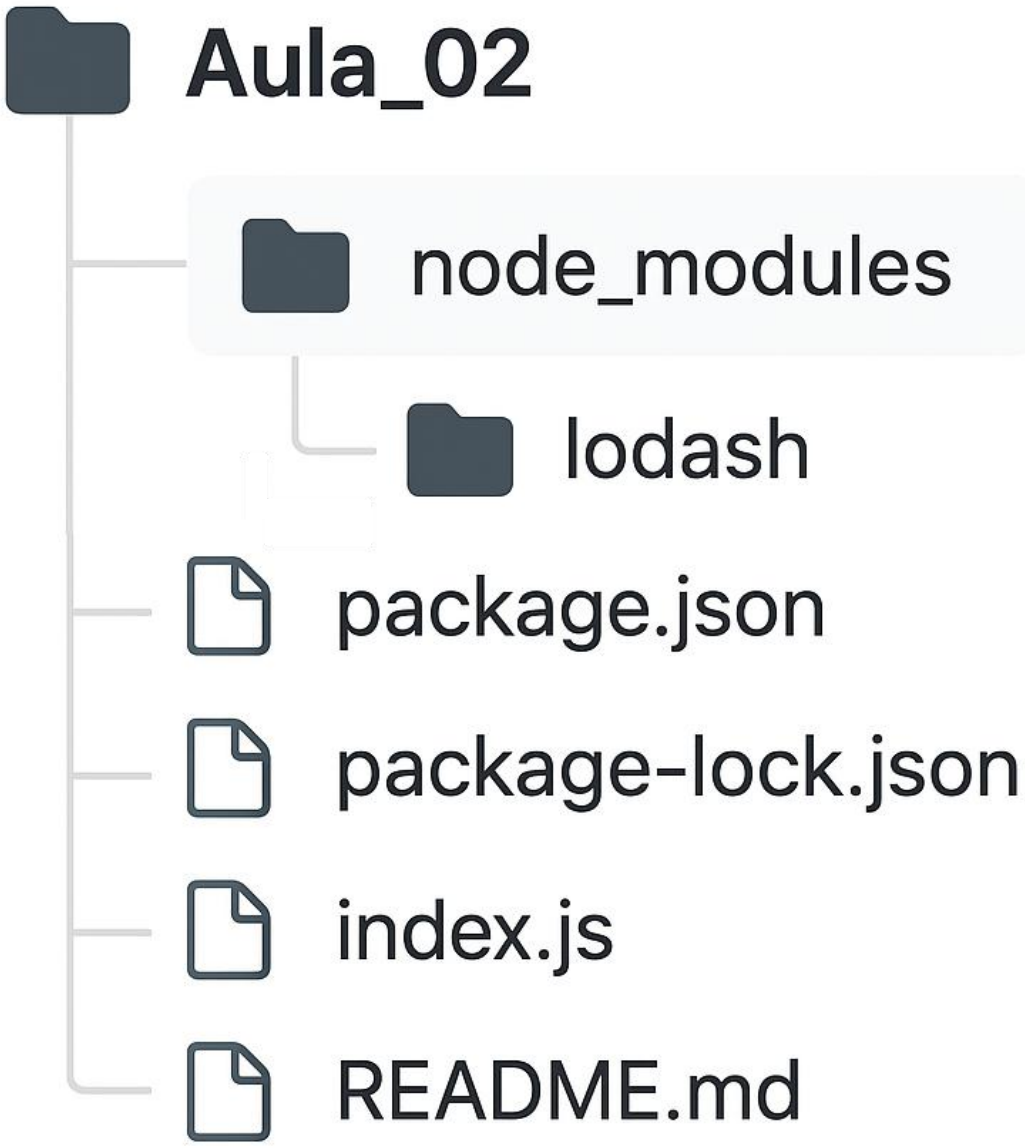
> npm install lodash

- O comando baixa e instala a biblioteca dentro da pasta node_modules
- Atualiza o arquivo package.json automaticamente na seção "dependencies“
- Cria/atualiza o arquivo package-lock.json com as versões exatas instaladas

Inclusão das dependências

```
C:\Users\User\Desktop\Aula_02>type package.json
{
  "name": "aula_02",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "Ariadne Arrais",
  "license": "ISC",
  "dependencies": {
    "lodash": "^4.17.21"
  }
}
```

```
C:\Users\User\Desktop\Aula_02>type package-lock.json
{
  "name": "aula_02",
  "version": "1.0.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "aula_02",
      "version": "1.0.0",
      "license": "ISC",
      "dependencies": {
        "lodash": "^4.17.21"
      }
    },
    "node_modules/lodash": {
      "version": "4.17.21",
      "resolved": "https://registry.npmjs.org/lodash/-/lodash-4.17.21.tgz",
      "integrity": "sha512-v2kDEe57lecTulaDIuNTPy3Ry4gLGJ6Z1O3vE1krgXZNrsQ+LFTGHVxVjcXPs17LhbZVGedAJv8XZ1tvj5FvSg==",
      "license": "MIT"
    }
  }
}
```


Aula_02
node_modules
lodash
package.json
package-lock.json
index.js
README.md

Criar o arquivo index.js com exemplo usando lodash

```
const _ = require('lodash');

const texto = 'ola mundo';
const textoCapitalizado = _.capitalize(texto);

console.log(textoCapitalizado);
```

Adicionar script **start** no **package.json**

- No package.json, localizar a chave "scripts" (criar se não existir) e adicionar:

```
"scripts": {
  "start": "node index.js"
}
```

Rodar o script

- No terminal, executar:

> npm start