

# Desenvolvimento Web

Trilha JavaScript com Angular e Node

Instrutor: Júlio Pereira Machado ([julio.machado@pucrs.br](mailto:julio.machado@pucrs.br))





Link para o  
Material

# O que iremos explorar?



# Tópicos

- Desenvolvimento de sistemas para Web
  - Arquitetura e padrões para Web
- HTML
- CSS
- JavaScript / TypeScript
  - Lado-cliente
  - Lado-servidor
  - Frameworks: Angular e Node
- Serviços Web

# Arquitetura de Software



# Arquitetura de Software

- Dois termos em comum:
  - Decomposição em alto nível de um sistema em suas partes
  - Decisões difíceis de alterar

# Arquitetura de Software

- Classificar as arquiteturas que são usadas nos sistemas nos permite reusar esse conhecimento.
- Essa classificação levou a identificação de famílias de arquiteturas que caracterizam estilos arquiteturais.
- Cada estilo arquitetural oferece suporte a um conjunto de requisitos não funcionais e atributos de projeto

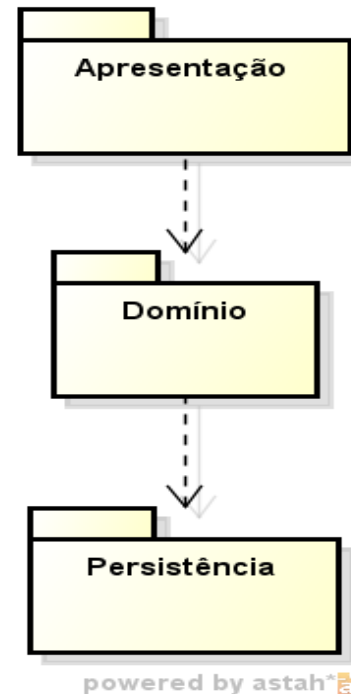
# Arquitetura de Software

- Existem muitas arquiteturas que combinam as arquiteturas básicas
- Existem arquiteturas que tem como foco um determinado tipo de aplicação, as chamadas arquiteturas de domínio específico
- Um domínio específico bastante conhecido são os sistemas de informação
- Para a área de sistemas de informação existem alguns padrões arquiteturais muito utilizados:
  - Arquitetura cliente-servidor
  - Arquitetura multicamadas
  - Arquitetura orientada a serviços



# Arquitetura em Camadas

- Camada (*layer*) é um agrupamento de granularidade grossa de classes, pacotes ou subsistemas que tem responsabilidade coesiva sobre um tópico importante do sistema



# Arquitetura em Camadas

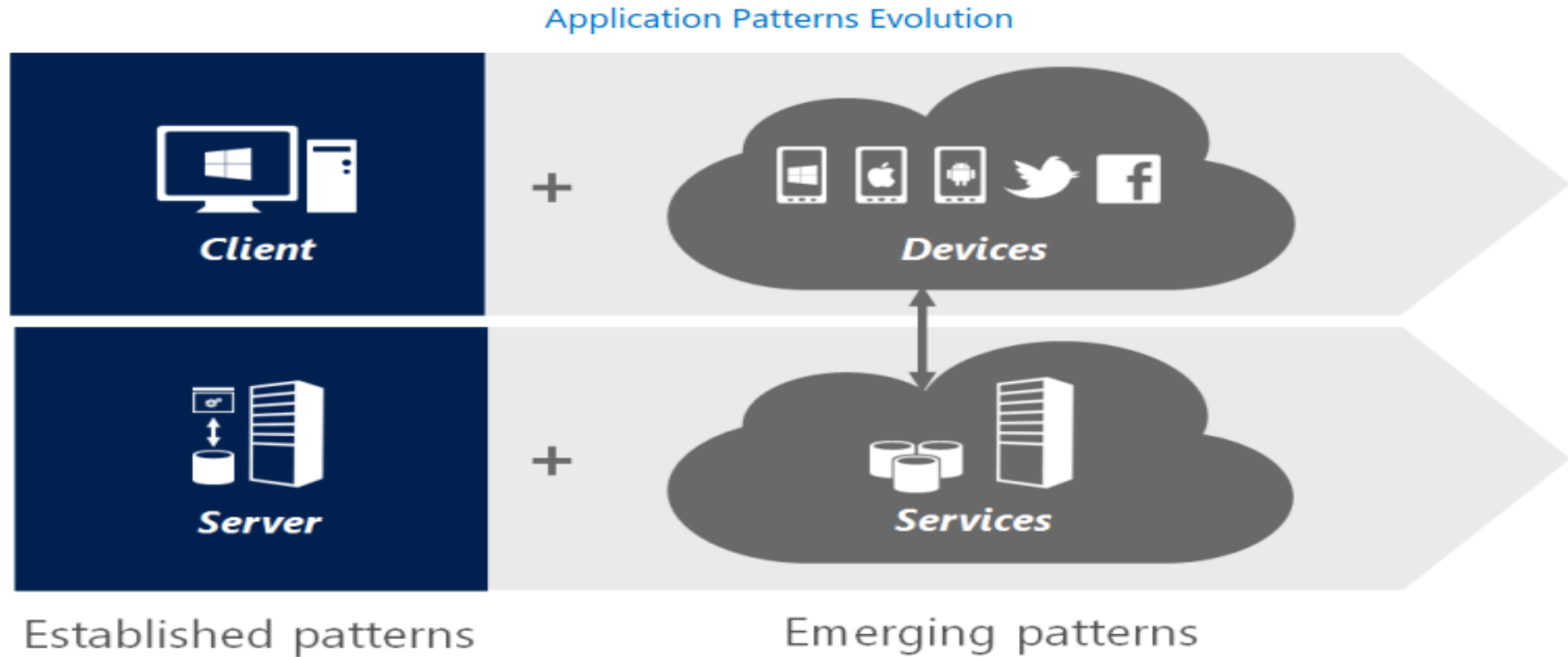
## Camadas usuais de um sistema em 3-camadas:

- Camada de apresentação ou de interface com o usuário
  - Interação com o usuário
  - Responsável por apresentar informações para o usuário e interpretar comandos do usuário em ações sobre a camada de domínio
  - Ex.: uma interface de cliente rico em HTML e JavaScript
- Camada de domínio ou de negócio
  - Objetos de software representando conceito do domínio que satisfazem requisitos da aplicação
  - Envolve cálculos, regras de validações, regras de negócio
- Camada de persistência ou de dados
  - Objetos de propósito geral e subsistemas que fornecem serviços para a aplicação
  - Geralmente são independentes da aplicação e reusáveis entre diversos sistemas
  - Ex.: uma fonte de dados a partir de um Sistema Gerenciador de Banco de Dados

# Arquitetura em Camadas

- Outro conceito de camada vem do termo *tier*
  - Representa a separação física das camadas lógicas em múltiplos nodos computacionais
  - Ex.: um sistema cliente-servidor pode ser visto como 2-tier, onde o cliente é um aplicativo e o servidor é um SGBD, por exemplo

# Arquitetura para Sistemas Web



# Arquitetura para Sistemas Web

Devices, Social, Services, Data, and Cloud



Devices and mobile applications

yammer



Microsoft account

Social

Services

Web APIs (REST, OData)



Multitier



Service orientation



Web



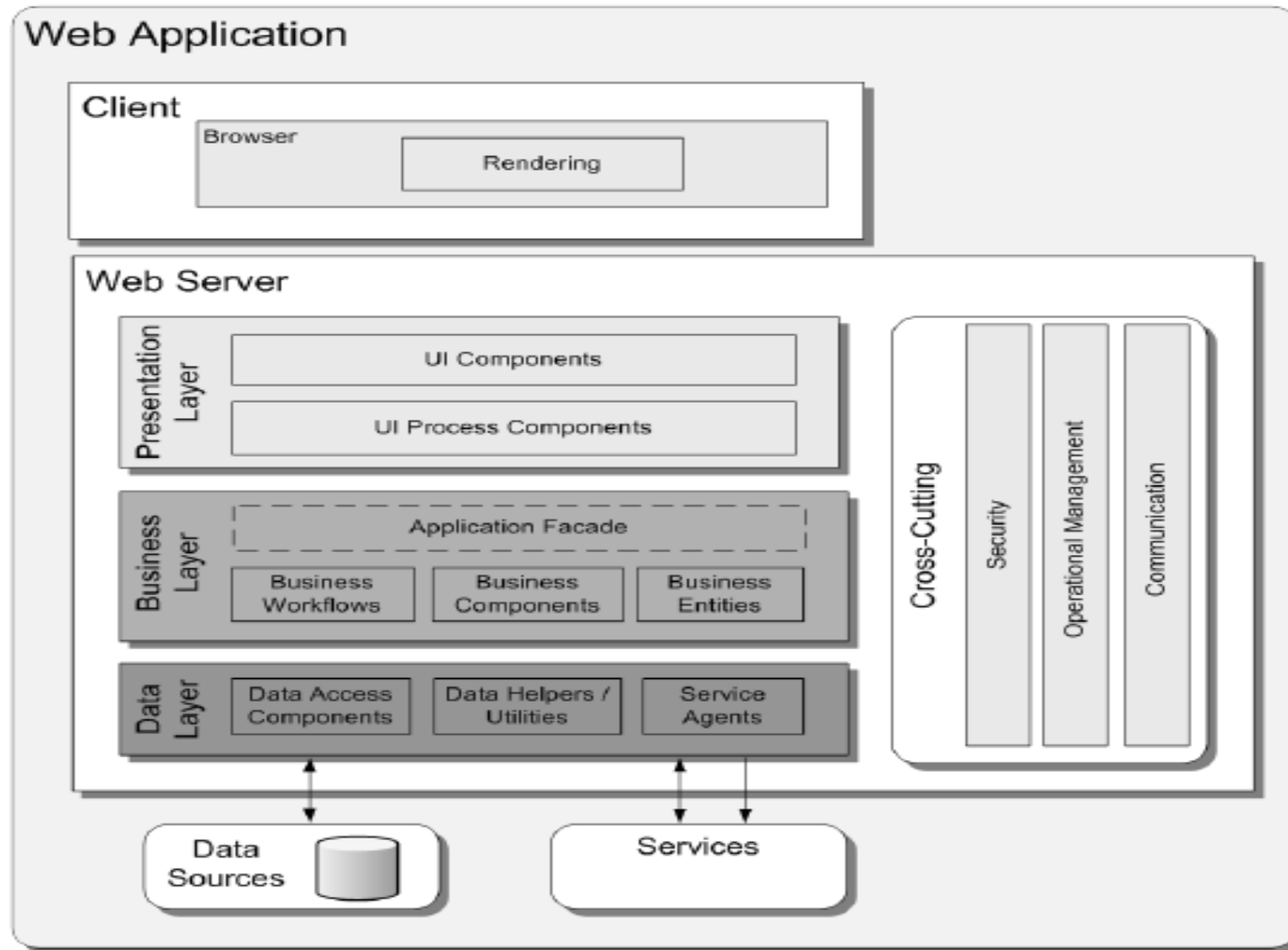
Data

Data, unified management and operations



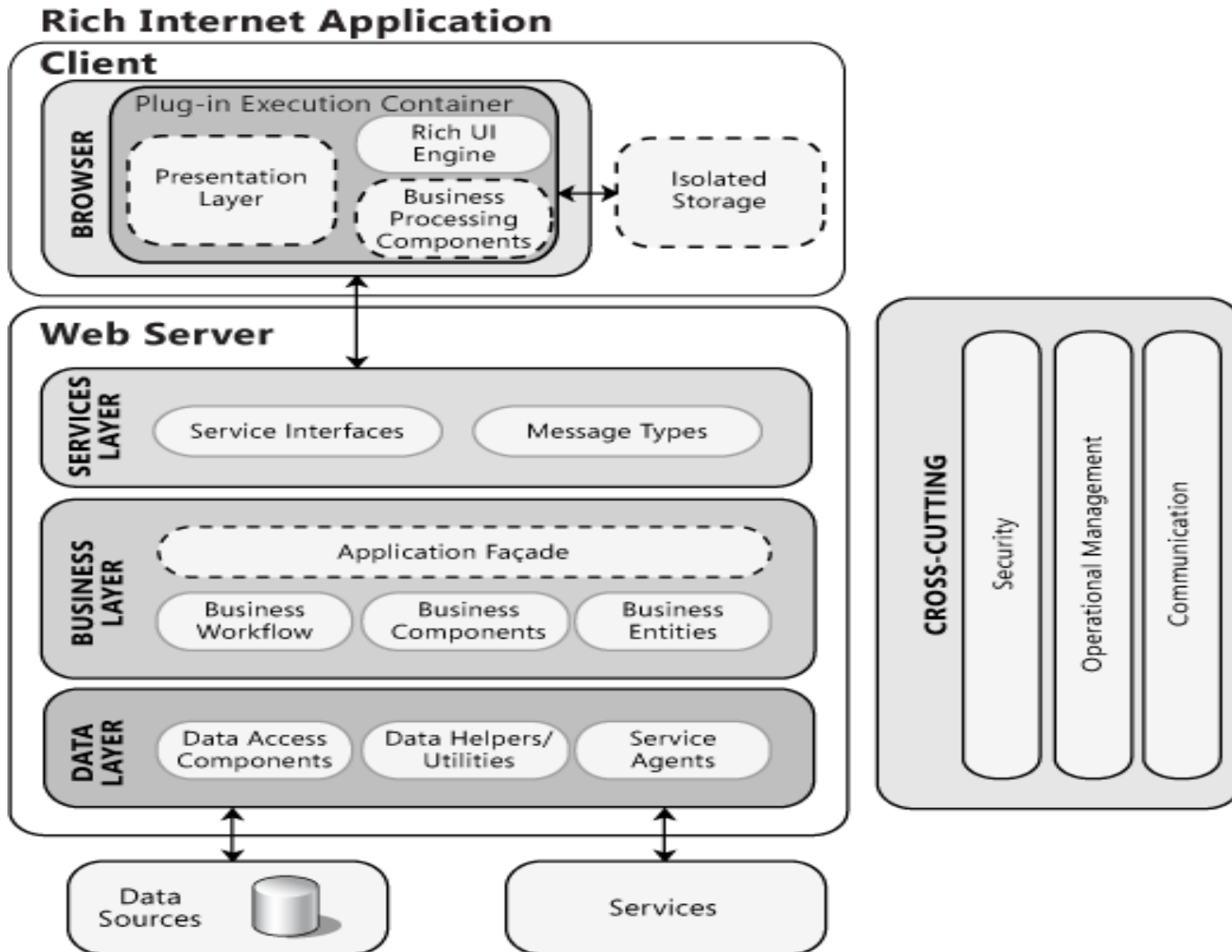
Public cloud infrastructure

# Arquitetura para Sistemas Web



- Cliente-servidor
- Cliente navegador (HTML+CSS+JavaScript)
- Camada de apresentação gerada no lado servidor

# Arquitetura para Sistemas Web



- Cliente-servidor
- Cliente navegador (HTML+CSS+JavaScript)
- Camada de apresentação gerada no lado cliente
- Camada de negócio pode ser distribuída nos lados cliente e servidor
- Servidor expõe camada de negócio via serviços web

# Arquétipos - PetClinic

<https://github.com/spring-petclinic/spring-petclinic-angular>

<https://github.com/spring-petclinic/spring-petclinic-rest>





# MEAN Stack



# MEAN Stack

- MEAN Stack é a solução de desenvolvimento para Web com as seguintes tecnologias:
  - MongoDB
  - Express
  - Angular
  - Node



express



# MongoDB

- Base de dados do tipo NoSQL
- Código aberto

<https://www.mongodb.com/>



# Express

- Framework para aplicações Web baseadas em Node.js
- Abstrai vários elementos da Infraestrutura
  - Por exemplo, a manipulação das requisições e respostas do HTTP
- Ajuda a organizar uma aplicação Node.js dentro do padrão Model-View-Controller (MVC)
- Código aberto

<http://expressjs.com/>

express

# Angular

- Framework para o lado cliente de aplicações Web que seguem o padrão MVC
- Código aberto

<https://angular.io/>



# Node.js

- Ambiente de execução de JavaScript no lado servidor
- Código aberto

<https://nodejs.org>



# TypeScript

- Superconjunto da linguagem JavaScript
- Código aberto



<https://www.typescriptlang.org/>

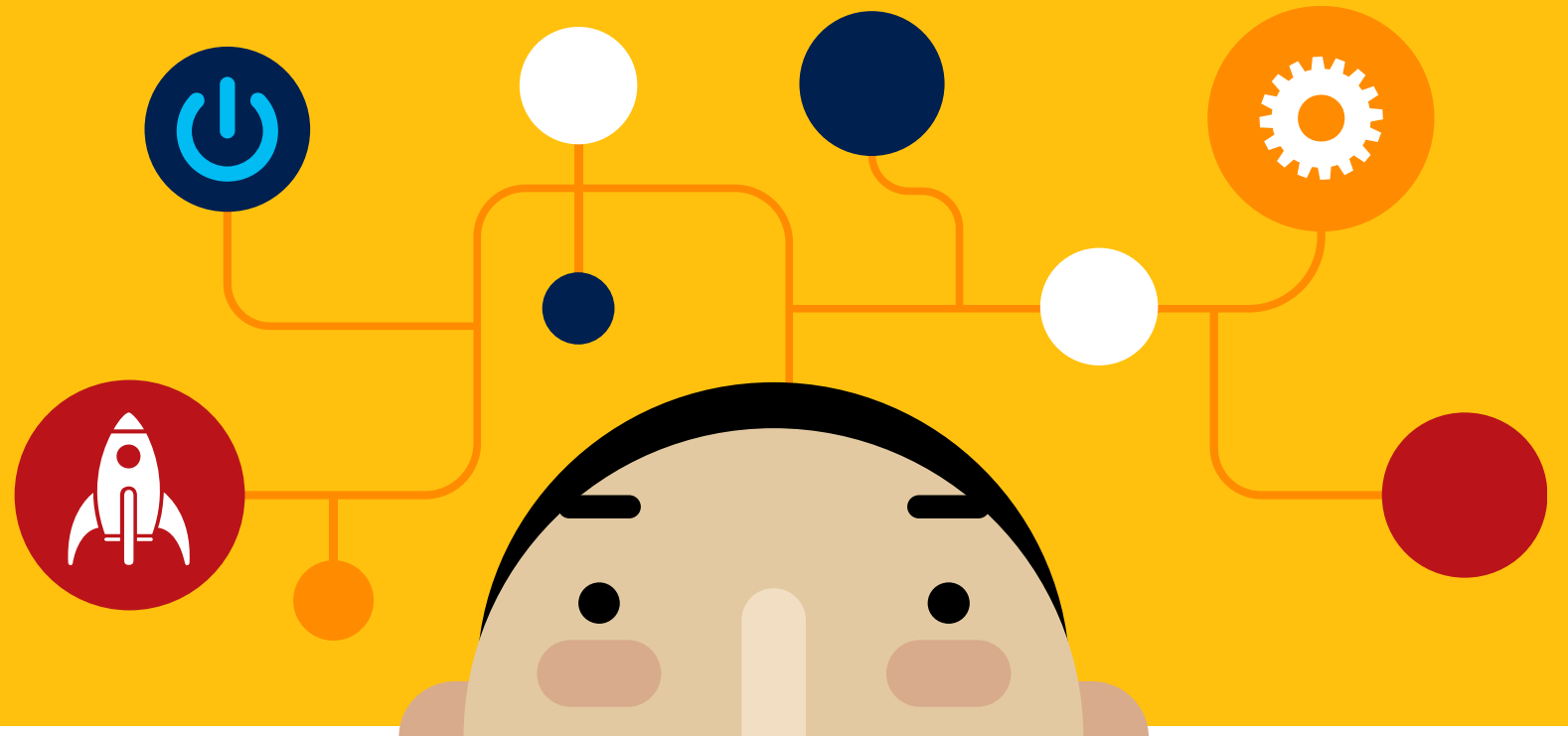
# TypeScript



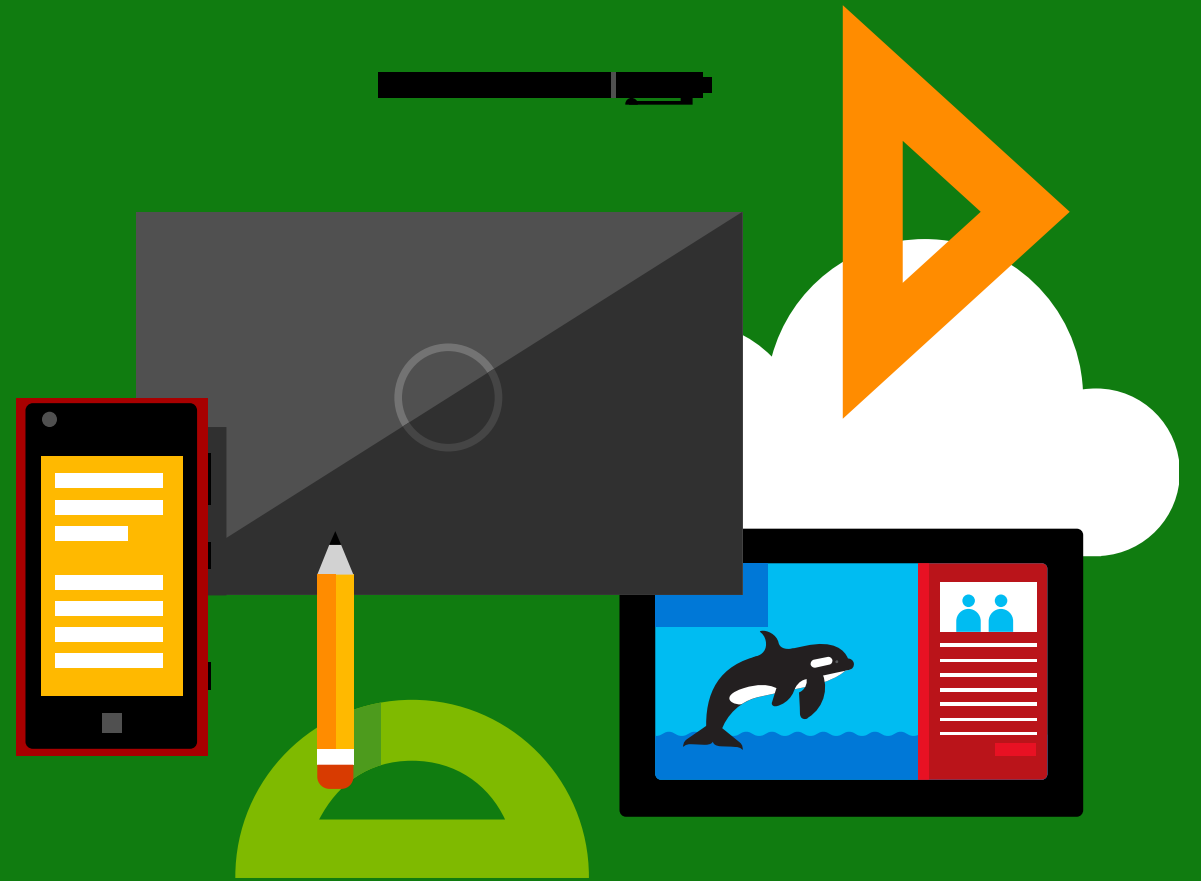


# Laboratório

- Abra as instruções do arquivo Lab00\_Ambiente



# Visual Studio Code



Como obter o Visual Studio Code?

<https://code.visualstudio.com/>

# Node.js



# Node.js

- Ambiente de execução de JavaScript no lado servidor
- Construída sobre a Google Chrome's V8 Javascript Engine
- Extremamente leve e eficiente

<https://nodejs.org>



# Node.js

- Recursos:
  - Documentação da API
    - <https://nodejs.org/api/>
  - Suporte ao ECMAScript 2015 (ES6)
    - <https://nodejs.org/en/docs/es6/>
    - <http://node.green/>
  - Ambiente de execução com TypeScript
    - <https://github.com/TypeStrong/ts-node>
  - Arquivos de definição de tipos para TypeScript
    - <http://definitelytyped.org/>
  - Execução automática de alterações no código
    - <https://nodemon.io/>

# Node.js – Características

- Utiliza um modelo de programação orientada a eventos
- A execução de operações segue o modelo não-bloqueante, isto é, são operações assíncronas
- Utiliza uma única thread de execução de código
- Provê uma grande coleção de módulos com funcionalidades adicionais
  - Utiliza o gerenciador de pacote NPM

# NPM





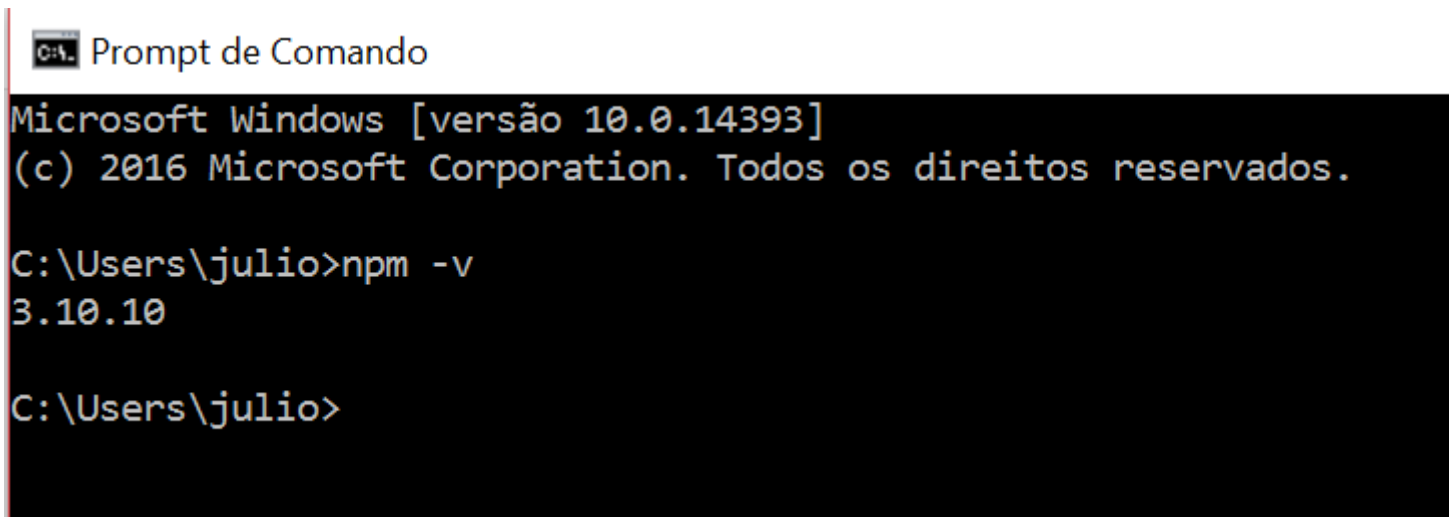
# NPM

- NPM é um acrônimo para *Node.js Package Manager*
- NPM proves um repositório público, uma especificação para a construção de pacotes, e um executável de linha de comando para gerenciar pacotes
- Node.js já instala uma versão do executável *npm*, mas ele é realmente um programa em separado com versões e atualizações em separado

<https://www.npmjs.com/>

# NPM

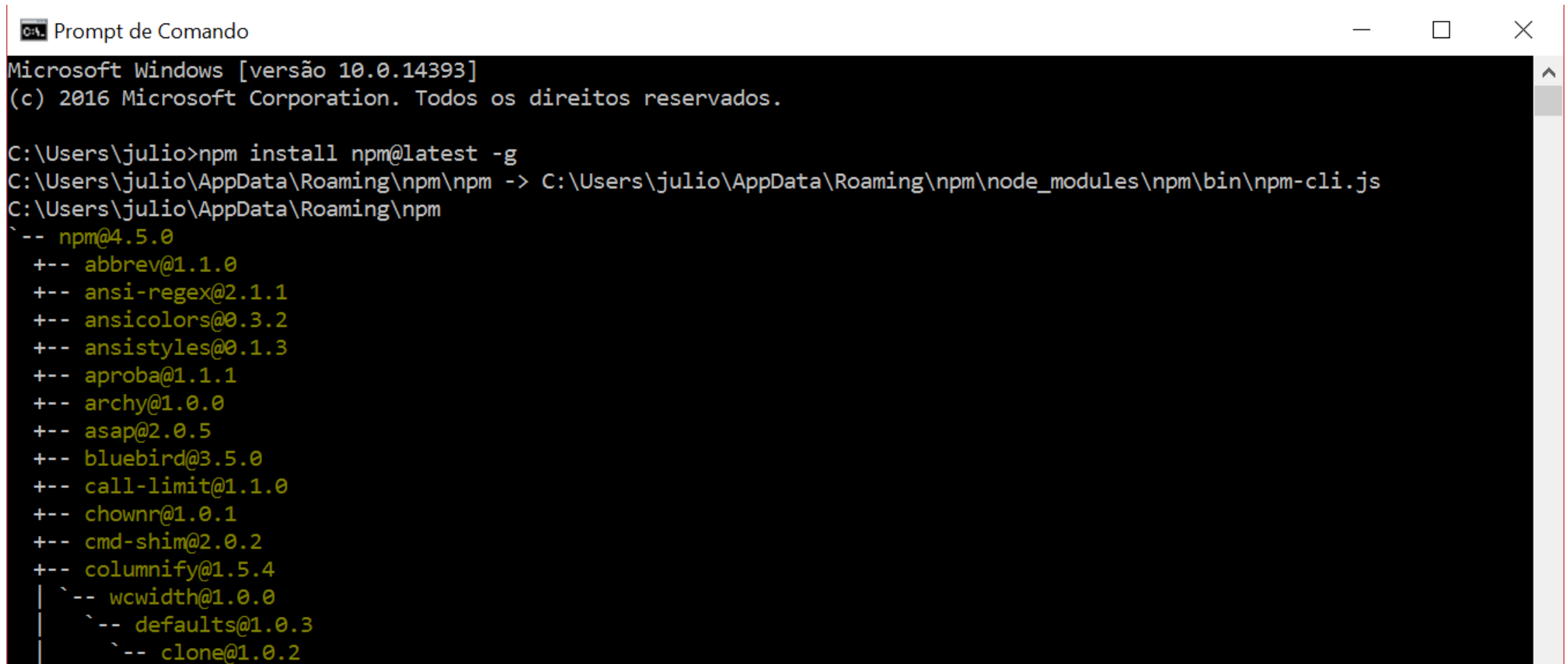
- Para verificar instalação e versão:
  - Na linha do comando usar *npm -v*



```
C:\> Prompt de Comando  
Microsoft Windows [versão 10.0.14393]  
(c) 2016 Microsoft Corporation. Todos os direitos reservados.  
  
C:\Users\julio>npm -v  
3.10.10  
  
C:\Users\julio>
```

# NPM

- Para atualizar versão:
  - Na linha do comando usar *npm install npm@latest -g*



```

C:\> Prompt de Comando

Microsoft Windows [versão 10.0.14393]
(c) 2016 Microsoft Corporation. Todos os direitos reservados.

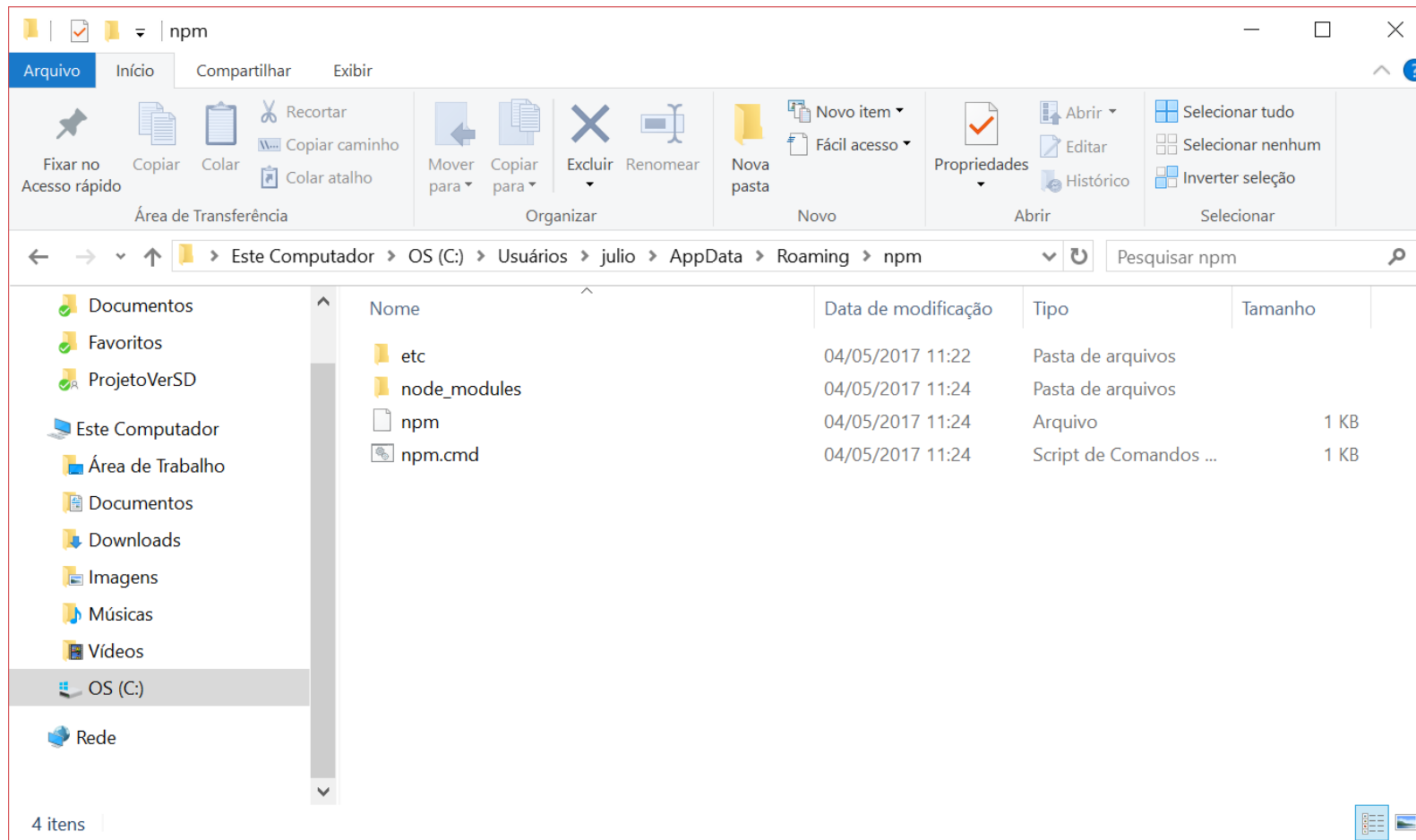
C:\Users\julio>npm install npm@latest -g
C:\Users\julio\AppData\Roaming\npm\npm -> C:\Users\julio\AppData\Roaming\npm\node_modules\npm\bin\npm-cli.js
C:\Users\julio\AppData\Roaming\npm
`-- npm@4.5.0
   +-- abbrev@1.1.0
   +-- ansi-regex@2.1.1
   +-- ansicolors@0.3.2
   +-- ansistyles@0.1.3
   +-- aproba@1.1.1
   +-- archy@1.0.0
   +-- asap@2.0.5
   +-- bluebird@3.5.0
   +-- call-limit@1.1.0
   +-- chownr@1.0.1
   +-- cmd-shim@2.0.2
   +-- columnify@1.5.4
   | `-- wcwidth@1.0.0
   |   `-- defaults@1.0.3
   |     `-- clone@1.0.2
```

# NPM – Pacotes Locais vs Globais

- Pacotes podem ser instalados localmente ou globalmente
  - Pacotes locais são armazenados localmente a um projeto, no subdiretório *node\_modules*
  - Pacotes globais são armazenados globalmente em um diretório de sistema
- Tipicamente, pacotes locais são bibliotecas de código utilizadas pelo projeto
- Tipicamente, pacotes globais são executáveis utilizados para realizar alguma operações sobre um projeto
  - Por exemplo, geradores de códigos
- Pacotes locais estão disponíveis somente ao projeto que os referencia, e pacotes globais estão disponíveis para qualquer projeto dentro do sistema

# NPM – Pacotes Locais vs Globais

- Pacotes globais estão armazenados (no Windows) em:
  - C:\Users\<nome do usuário>\AppData\Roaming\npm



# NPM – Arquivo `package.json`

- Todos os projetos (que também são pacotes) precisam ser configurados para trabalhar com o NPM
- O comando **`npm init`** é utilizado para configurar um projeto
  - Após sucessivas perguntas, um arquivo **`package.json`** será criado
- O arquivo *`package.json`* possui metadados sobre o projeto, tais como as dependências
- Quando se instala um novo pacote via NPM em um projeto, a dependência é adicionada ao arquivo *`package.json`*

# NPM – Arquivo package.json

```
{
  "name": "my_package",
  "description": "",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/ashleygwilliams/my_package.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ashleygwilliams/my_package/issues"
  },
  "homepage": "https://github.com/ashleygwilliams/my_package"
}
```

# NPM – Instalação e Remoção de Pacotes

- O executável *npm* é utilizado para gerenciar pacotes
- O primeiro argumento é o comando a ser executado
- Pacotes podem ser instalados com o comando **install**, e desinstalado com o comando **uninstall**
- Existem muitos outros comandos disponíveis
- Observação:
  - Se ocorrer erros de permissão ao instalar globalmente acesse <https://docs.npmjs.com/getting-started/fixing-npm-permissions>

```
$ npm install -g grunt-cli
```

```
$ npm install grunt
```

```
$ npm uninstall -g grunt-cli
```

```
$ npm uninstall grunt
```

- A opção **-g** define o escopo global, sem esta opção os pacotes são instalados localmente