



Quem se prepara, não para.

Arquitetura de Aplicações Web

5º período

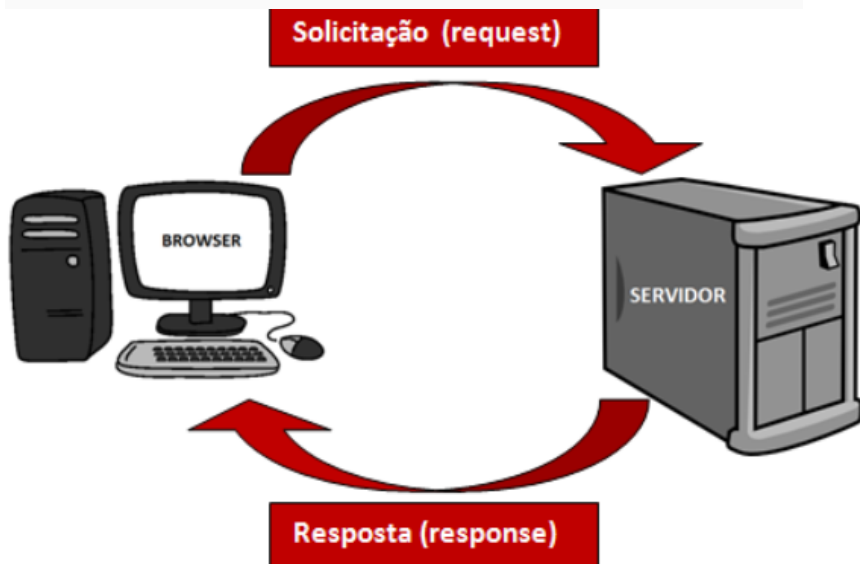
Professora: Michelle Hanne

Como Funciona as Aplicações Web

Requisição

```
GET /index.html HTTP/1.1  
Host: www.twitter.com
```

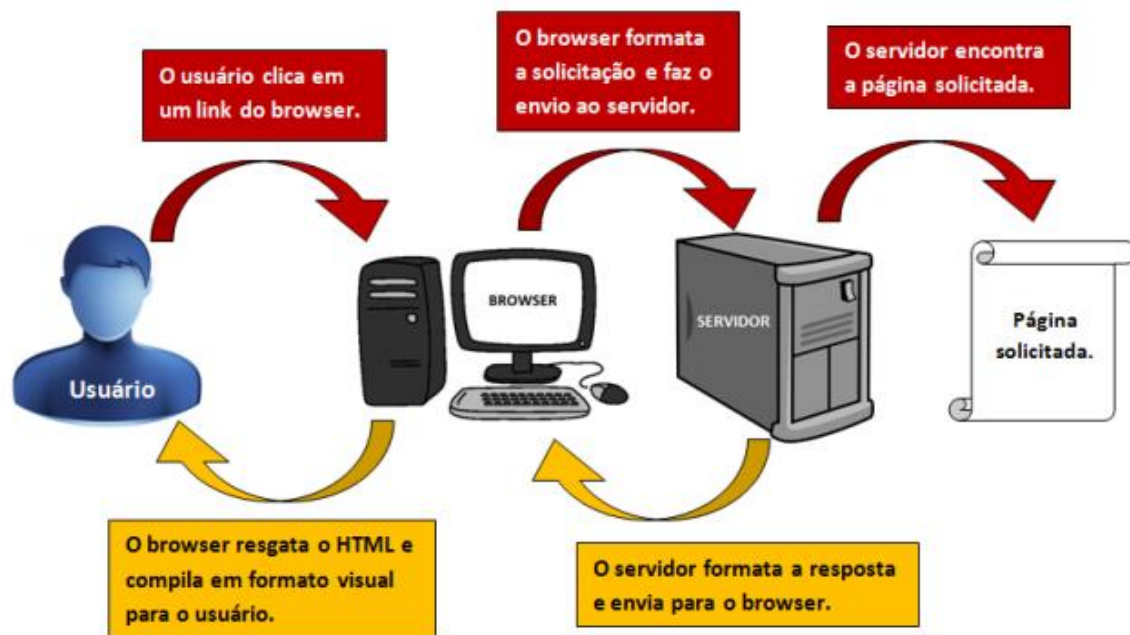
A **função do servidor web** é receber uma solicitação (requisição) e devolver (resposta) algo para o cliente.



Resposta

```
HTTP/1.1 200 OK  
Date: Mon, 23 May 2005 22:38:34 GMT  
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)  
Content-Type: text/html; charset=UTF-8  
Content-Length: 131  
<!DOCTYPE html>  
<html>  
<head>  
<title>Twitter</title>  
</head>  
<body>  
    Olá mundo, este é um tweet.  
</body>  
</html>
```

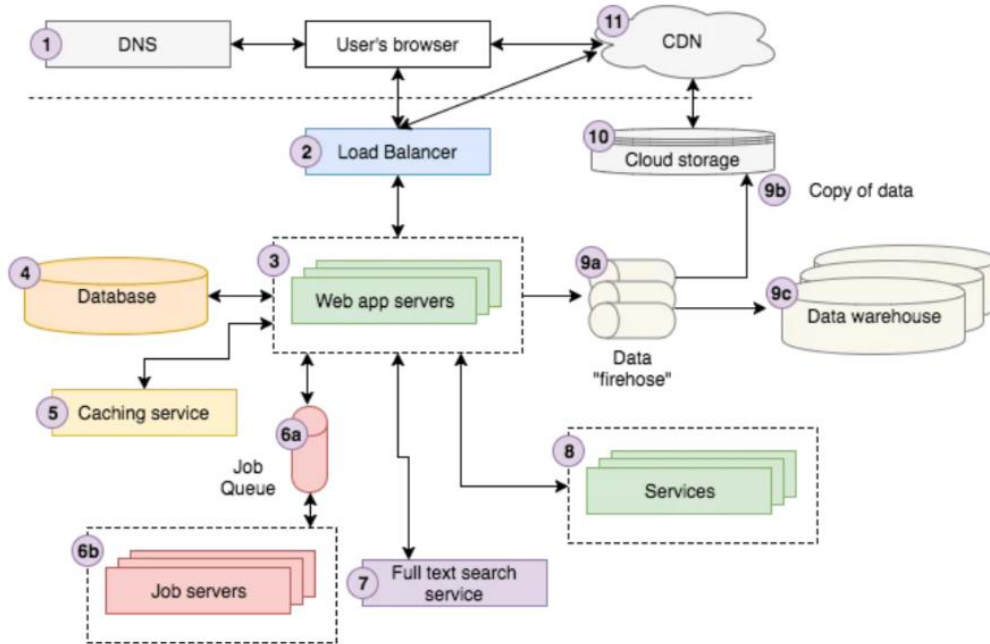
Como Funciona as Aplicações Web



Protocolo HTTP

- Além do servidor web, podemos ter um **back-end**.
- **Back-end** é uma aplicação escrita por nós, executada pelo **servidor web**.
- Cada requisição é recebida pelo servidor web, que a delega ao back-end.
- Também podemos ter um SGBD

Como Funciona as Aplicações Web



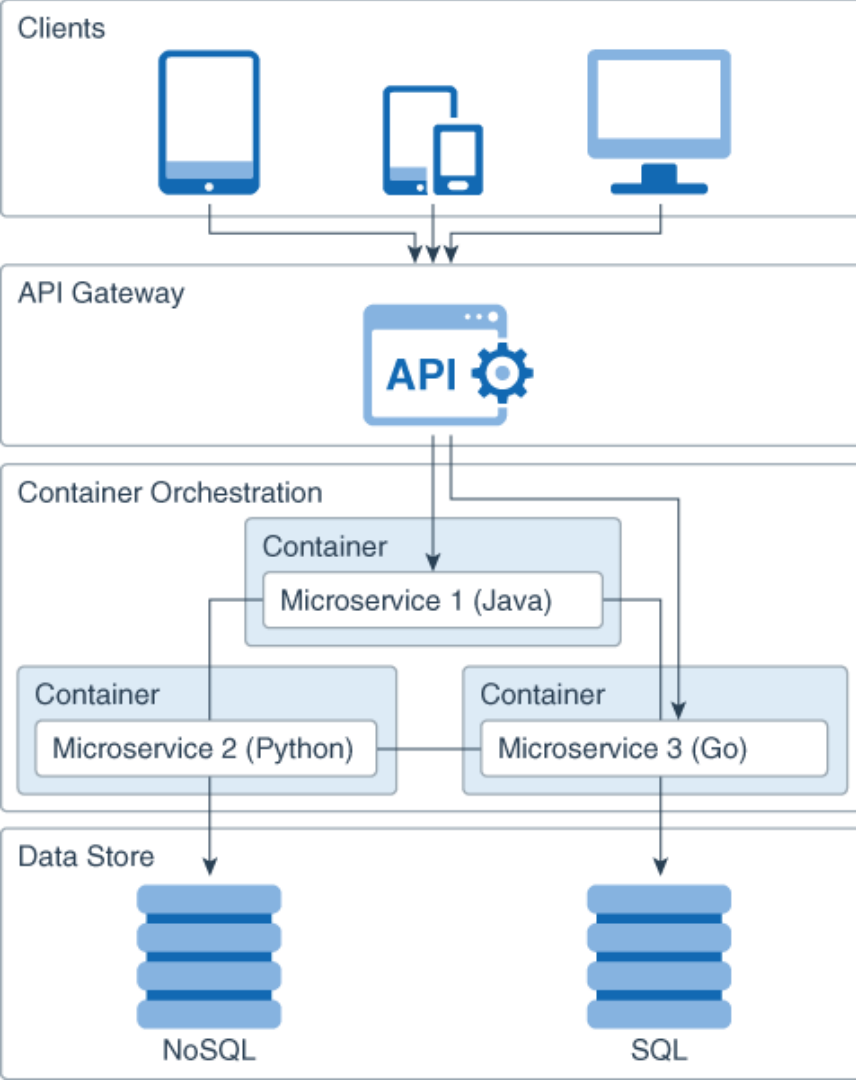
CDN (Content Delivery Network - Rede de Entrega de Conteúdo) é um grupo de servidores geograficamente distribuídos que aceleram a entrega do conteúdo da Web. Armazenam conteúdo em cache, como páginas da Web, imagens e vídeos em servidores proxy próximos à sua localização física.

Arquitetura de Microserviços

Microserviços são aplicações desmembradas em serviços independentes. Para isso, os serviços se comunicam entre si usando APIs.

Em uma arquitetura de microserviço, **cada função define um serviço e é criada e implementada de modo independente**. Essa característica implica no fato de que cada serviço, de forma individual, pode funcionar ou apresentar falha sem comprometer os demais.

A tendência se tornou popular nos últimos anos, à medida que as empresas buscam se tornar mais ágeis e avançar em direção a um DevOps e testes contínuos.

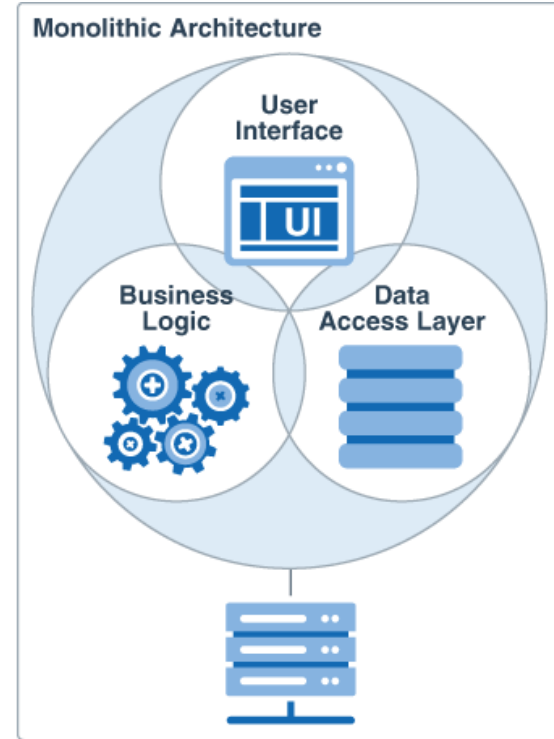
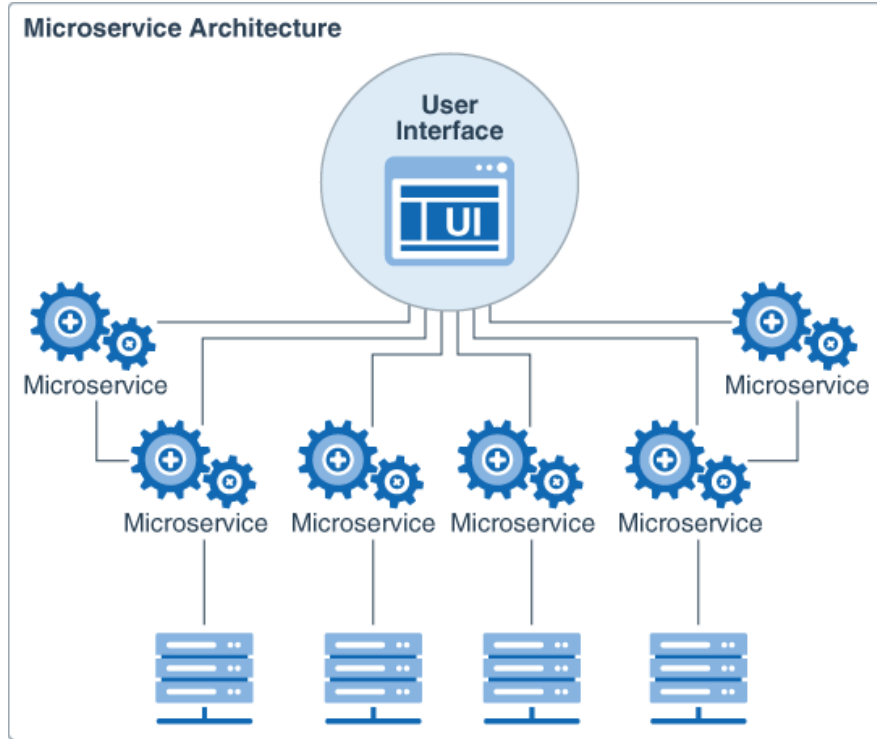


Arquitetura de Microsserviços

Os microsserviços seguem o modelo share-nothing e são executados como **processos sem estado**. Essa abordagem facilita a escala e a manutenção do aplicativo.

- **A camada API** é o ponto de entrada de todas as solicitações do cliente para um microsserviço. A comunicação ocorre através de protocolos: HTTP, gRPC e TCP/UDP.
- **A camada lógica** se concentra em uma única tarefa de negócios, minimizando as dependências dos outros microsserviços.
- **A camada de armazenamento** de dados fornece um mecanismo de persistência, como um mecanismo de armazenamento de banco de dados
- **Normalmente, cada microsserviço é executado em um contêiner que fornece um ambiente de tempo de execução leve.**

Arquitetura Monolítica vs Microserviços



Arquitetura Monolítica vs Microserviços

Característica	Arquitetura de microserviços	Arquitetura Monolítica
Desenho da unidade	O aplicativo consiste em serviços acoplados vagamente. Cada serviço suporta uma única tarefa de negócios.	Todo o aplicativo é projetado, desenvolvido e implantado como uma única unidade.
Reutilização da funcionalidade	Microserviços definem APIs que expõem sua funcionalidade a qualquer cliente. Os clientes podem até ser outras aplicações.	A oportunidade de reutilizar a funcionalidade entre aplicativos é limitada.
Comunicação no âmbito do pedido	Para se comunicar entre si, os microserviços de um aplicativo usam o modelo de comunicação solicitação-resposta. A implementação típica usa chamadas de API REST com base no protocolo HTTP.	Os procedimentos internos (chamadas de função) facilitam a comunicação entre os componentes do aplicativo. Não é necessário limitar o número de chamadas de procedimento interno.
Flexibilidade tecnológica	Cada microservice pode ser desenvolvido usando uma linguagem de programação e estrutura que melhor se adapte ao problema que o microservice é projetado para resolver.	Geralmente, todo o aplicativo é escrito em uma única linguagem de programação.
Gerenciamento de dados	Descentralizado: Cada microserviço pode usar seu próprio banco de dados.	Centralizado: todo o aplicativo usa um ou mais bancos de dados.
Implantação	Cada microservice é implantado de forma independente, sem afetar os outros microservices no aplicativo.	Qualquer alteração, por menor que seja, requer a reimplantação e reinicialização de todo o aplicativo.
Capacidade de Manutenção	Os microserviços são simples, focados e independentes. Portanto, o aplicativo é mais fácil de manter.	À medida que o escopo do aplicativo aumenta, a manutenção do código se torna mais complexa.
Resiliência	A funcionalidade do aplicativo é distribuída em vários serviços. Se um microservice falhar, a funcionalidade oferecida pelos outros microservices continuará disponível.	Uma falha em qualquer componente pode afetar a disponibilidade de todo o aplicativo.
Escalabilidade	Cada microserviço pode ser dimensionado independentemente dos outros serviços.	Todo o aplicativo deve ser dimensionado, mesmo quando o requisito de negócios é dimensionar apenas determinadas partes do aplicativo.

Exemplos de Microsserviços

- Muitos gigantes da web – incluindo Amazon, Netflix, Twitter, PayPal, Spotify e The Guardian – adotaram com sucesso a arquitetura de microsserviço.
- Barra de busca em um site, marketing place ou e-commerce
- Recomendações de produtos relacionados ao que está buscando em um e-commerce
- Adicionar produtos no carrinho
- Realização de checkout

Netflix é um dos melhores exemplos de implementação de arquitetura de microsserviços. Em 2009, a Netflix mudou de uma arquitetura monolítica para microsserviços devido à crescente demanda por seus serviços. Porém, como não existiam microsserviços na época, os engenheiros da Netflix criaram uma tecnologia de código aberto.

Exemplos de Microsserviços

Uber: começou com uma arquitetura monolítica. Era mais simples para os fundadores da empresa quando forneciam aos clientes apenas o serviço UberBLACK. Mas, como a inicialização cresceu rapidamente, os desenvolvedores decidiram mudar para microsserviços para usar várias linguagens e estruturas. **Agora, o Uber tem mais de 1.300 microsserviços com foco na melhoria da escalabilidade do aplicativo.**

Spotify: com mais de 172 milhões de usuários premium (em 2021), os fundadores do Spotify decidiram construir um sistema com componentes escalonáveis de forma independente para tornar a sincronização mais fácil. **Para o Spotify, o principal benefício dos microsserviços é a capacidade de prevenir falhas massivas. Mesmo que vários serviços falhem simultaneamente, os usuários não serão afetados.**



- Não é um servidor web
- Ryan Dahl, seu criador, teve o seguinte raciocínio:
 - Gosto de JavaScript e gostaria de poder usar a linguagem fora dos navegadores
 - Hmm, o pessoal da Google fez um ótimo serviço ao criar a máquina virtual que executa JavaScript no Google Chrome: a V8
 - Já sei: vou separar essa V8 do navegador e criar um ambiente para que programas JS possam acessar o sistema de arquivos y outras cositas más





É uma plataforma para se desenvolver aplicações usando JavaScript fora do navegador. Características:

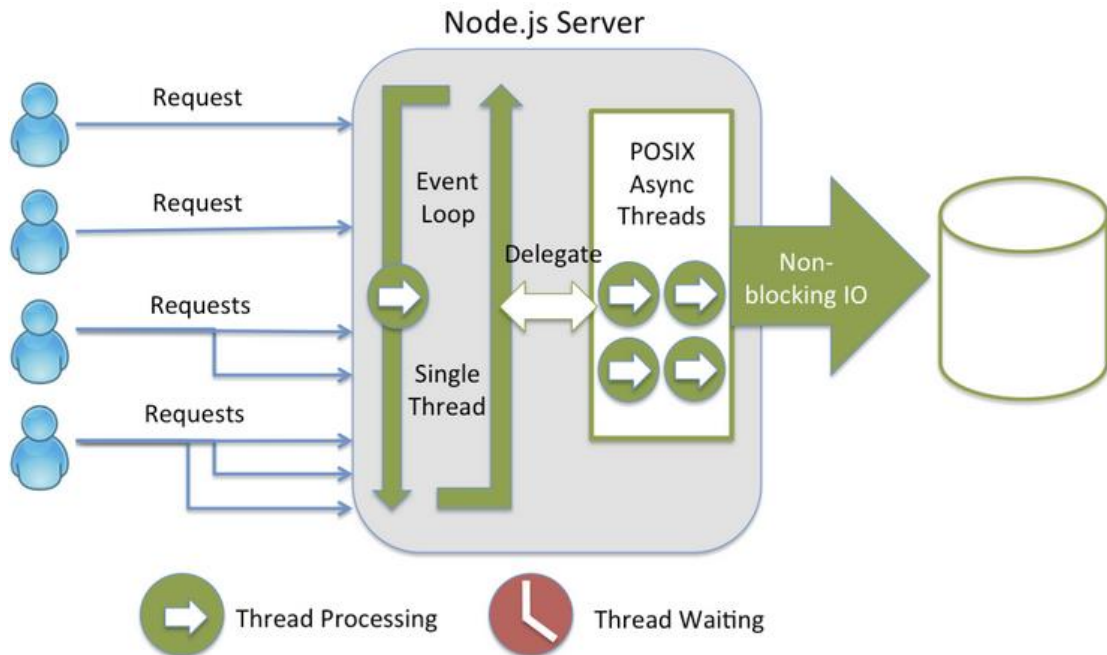
- Escrito em C/C++ e JavaScript
- Executa JavaScript de forma leve (pouca memória) e rápida
- Arquitetura de thread única e orientada a eventos
- E/S não-blocante (non-blocking I/O)
- Não é uma bala de prata!!
- É uma ferramenta que serve bem para aplicações DIRTy (*data-intensive real-time applications*)



- O Node.js é um ***runtime environment*** de código aberto executado na **engine V8 do Chrome usando JavaScript**, com alto potencial de escalabilidade sem estar atrelado a plataformas ou dispositivos.
- Tecnologia usada para desenvolver aplicativos *server-side (back-end)*.

- Leveza e flexibilidade fazem do Node.JS uma tecnologia indicada para a implementação de serviços e componentes de arquiteturas como a de **microsserviços e serverless**.
- Além disso, conta com suporte das principais empresas de produtos e serviços Cloud do mercado, como a **AWS, Google Cloud e Microsoft Azure**.
- **Indicações:**
 - Aplicações em Tempo Real (ex. chat)
 - Ambientes Escaláveis (grande numero de conexões concorrentes)
 - Camada de Entrada do Servidor : O Node.js faz pouco processamento de dados e apenas passa a requisição para frente, se comunicando com serviços de *backend*.
 - API com NoSQL.

Node.js



- cada requisição é processada de uma só vez (Event Loop), que delega para o **Async Thread** o processo de pull de threads (ex. 4 threads alocadas)

<https://www.youtube.com/watch?v=KtDwdoxQL4A>

Preparando o Ambiente

Instalação do Github desktop e do VS studio ou Atom

- <https://desktop.github.com/>
- <https://code.visualstudio.com/docs/?dv=win>
- <https://atom.io/>

Instalando o Node JS e o NPM

<https://nodejs.org/en/>

Testar se foi instalado corretamente

```

C:\Users\michelle_pc>node -v
v12.16.1

C:\Users\michelle_pc>npm -v
6.13.4

C:\Users\michelle_pc>
```

Hello World

Hello World em Node.js

- Instale o Node.js
- Crie um arquivo, hello.js, contendo:

```
console.log('woot woot');
```

- Execute o arquivo no terminal:

```
$ node hello.js
```

Mas onde está “servidor web” nisso?

- A arquitetura do Node.js (*event-driven + non-blocking I/O*), somados às facilidades desenvolvidas na plataforma tornam a criação de um servidor Web muito concisa e simples

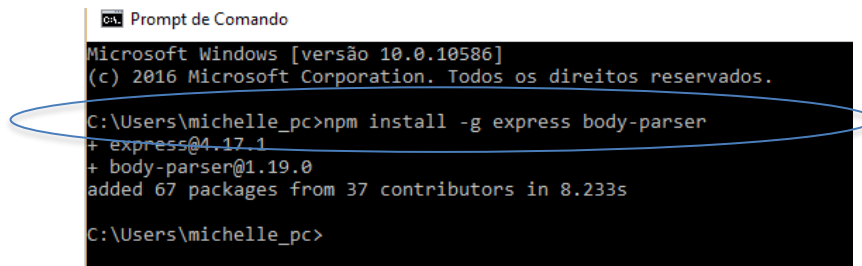
Instalando o Express.js

O Express permite criar aplicações web simples e outras aplicações.

Para instalar acesse o terminal e digite:

npm install -g express body-parser

O comando **npm install -g** instala o Express globalmente no seu sistema, acrescentar dependências com o comando **body-parser**.



```
cmd Prompt de Comando
Microsoft Windows [versão 10.0.10586]
(c) 2016 Microsoft Corporation. Todos os direitos reservados.

C:\Users\michelle_pc>npm install -g express body-parser
+ express@4.17.1
+ body-parser@1.19.0
added 67 packages from 37 contributors in 8.233s

C:\Users\michelle_pc>
```

Instalando o React Native

Instalação do React Native

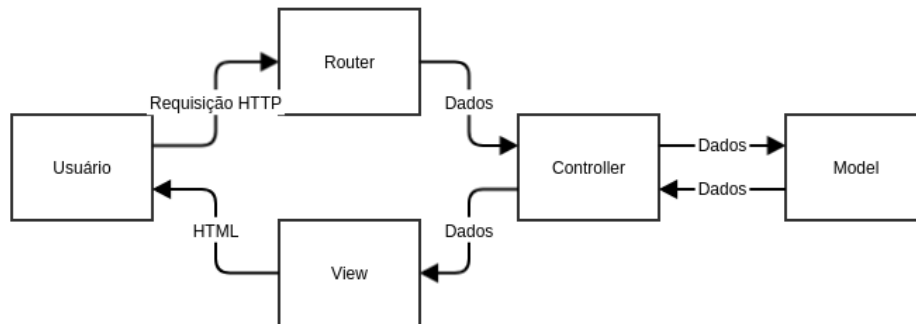
```
>npm install -g react-native-cli
```

Testar se está instalado

```
>react-native -h
```

MVC em Node.JS

- O padrão MVC é simples na utilização, mas sua implementação no código pode ser um tanto complexa.
- Segue um exemplo simples e básico que utiliza do conceito MVC em Node.JS.



MVC Vantagens

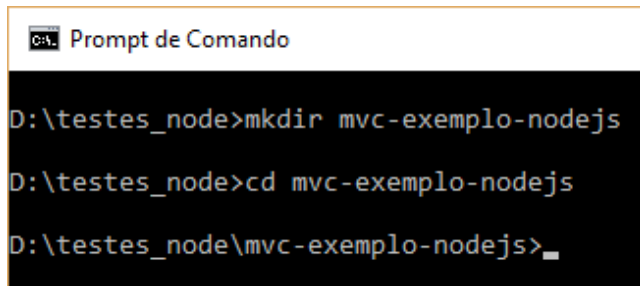
- **Reusabilidade de código:** uma funcionalidade já implementada em uma parte do sistema, pode ser novamente utilizada em outra parte, sem ser necessário o seu re-desenvolvimento.
 - **Exemplo:** em um sistema toda vez que um usuário altera sua senha, ele recebe um e-mail informando a alteração. Neste mesmo sistema quando ocorre uma venda, também é enviado um e-mail informando este usuário.
- **Desenvolvimento rápido:** neste modelo é possível que desenvolvedores e designers trabalhem simultaneamente na aplicação. Cada um trabalharia em uma camada diferente da aplicação, tornando o desenvolvimento produtivo.
- **Fácil manutenção:** como o código é dividido em partes distintas, se torna fácil adicionar novas funcionalidades e alterar características antigas. O código também fica mais fácil de ser compreendido por outros desenvolvedores.

Exemplo - MVC em Node.JS

- A aplicação para exemplificação do MVC em Node.JS, consiste em uma tela onde serão listadas as últimas notícias fictícias.
- Será utilizado um arquivo JSON onde estarão salvas as notícias que serão exibidas.
- O código da aplicação está disponível no [GitHub: https://github.com/andersonirias/mvc-exemplo-nodejs](https://github.com/andersonirias/mvc-exemplo-nodejs).

Exemplo - MVC em Node.JS

- **Passo 1:** Criar e acessar a pasta do Projeto (**mvc-exemplo-nodejs**)



```

C:\> Prompt de Comando

D:\testes_node>mkdir mvc-exemplo-nodejs

D:\testes_node>cd mvc-exemplo-nodejs

D:\testes_node\mvc-exemplo-nodejs>_

```

- **Passo 2:** Criar a estrutura de Arquivos e pastas da raiz do Projeto. **Pastas:** data e src. **Arquivos:** app.js e package.json

Exemplo - MVC em Node.JS

```
D:\testes_node\mvc-exemplo-nodejs>mkdir data
D:\testes_node\mvc-exemplo-nodejs>mkdir src
D:\testes_node\mvc-exemplo-nodejs>type nul >app.js
D:\testes_node\mvc-exemplo-nodejs>npm init -y
Wrote to D:\testes_node\mvc-exemplo-nodejs\package.json:

{
  "name": "mvc-exemplo-nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

D:\testes_node\mvc-exemplo-nodejs>
```

Exemplo - MVC em Node.JS

Passo 3: Acrescentar no arquivo **package.json** os metadados/pacotes necessários para o projeto

```

C:\> Prompt de Comando

D:\testes_node\mvc-exemplo-nodejs>npm i express consign ejs fs --save

> ejs@3.0.2 postinstall D:\testes_node\mvc-exemplo-nodejs\node_modules\ejs
> node --harmony ./postinstall.js

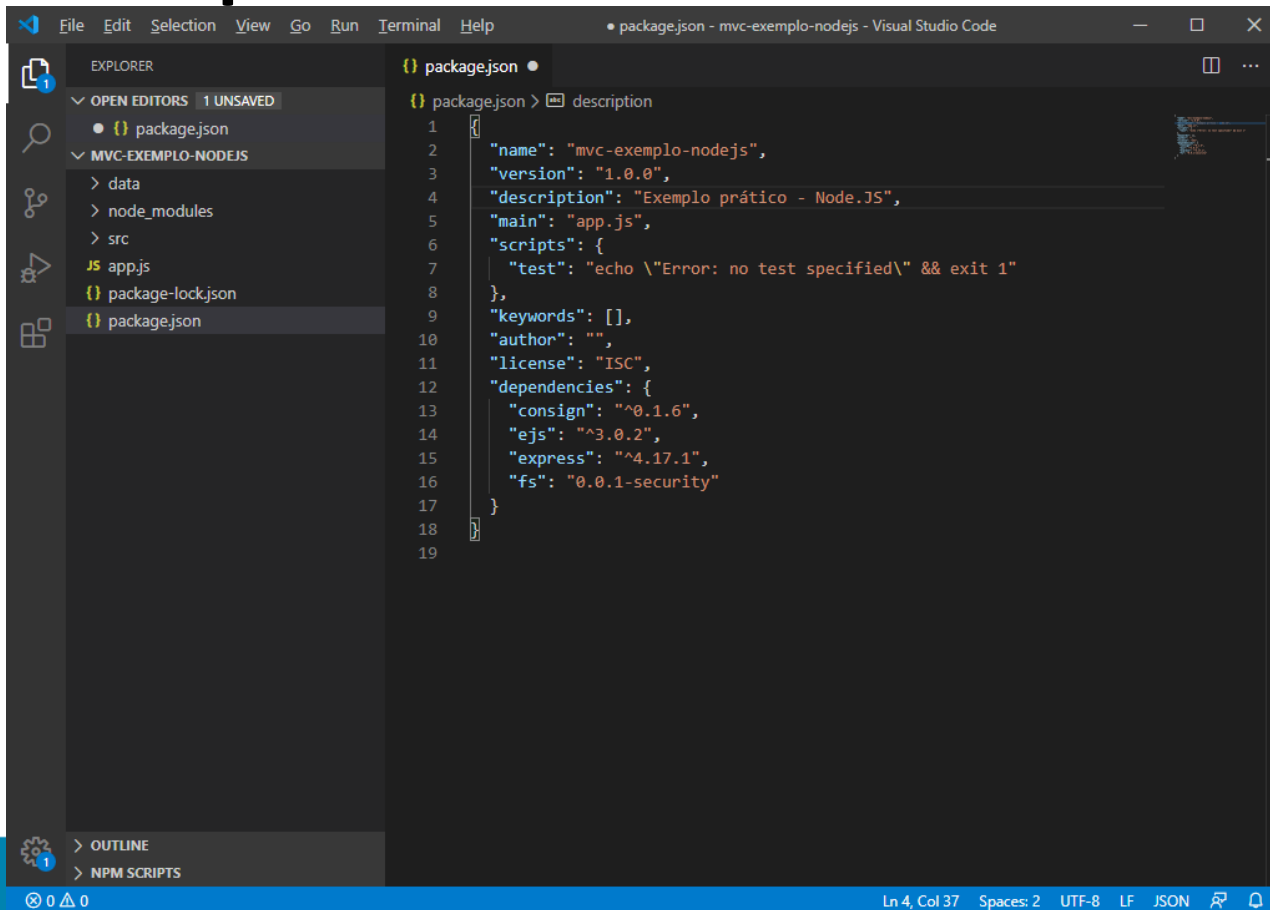
Thank you for installing EJS: built with the Jake JavaScript build tool (https://jakejs.com/)

npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN mvc-exemplo-nodejs@1.0.0 No description
npm WARN mvc-exemplo-nodejs@1.0.0 No repository field.

+ consign@0.1.6
+ ejs@3.0.2
+ fs@0.0.1-security
+ express@4.17.1
added 53 packages from 39 contributors and audited 129 packages in 7.997s
found 0 vulnerabilities

D:\testes_node\mvc-exemplo-nodejs>
```

Exemplo - MVC em Node.JS



The screenshot shows the Visual Studio Code interface with the `package.json` file open. The Explorer on the left shows the project structure with folders `data`, `node_modules`, and `src`, and files `app.js`, `package-lock.json`, and `package.json`. The `package.json` file contains the following content:

```
{
  "name": "mvc-exemplo-nodejs",
  "version": "1.0.0",
  "description": "Exemplo prático - Node.JS",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "consign": "^0.1.6",
    "ejs": "^3.0.2",
    "express": "^4.17.1",
    "fs": "0.0.1-security"
  }
}
```

Passo 4: Abrir o Visual Studio Code:

Conferir se o arquivo `package.json`, possui os metadados relevantes para o projeto.

MVC em Node.JS

Passo 5: O arquivo **app.js** é o arquivo principal do projeto. Primeiramente incluímos as dependências do projeto. Depois configuramos o módulo **ejs**, que é para trabalharmos com as views. Após isso configuramos o módulo **consign**, que faz o carregamento automático dos scripts do nosso projeto. E por fim configuramos o servidor web do framework **express** para escutar requisições **http** na porta 3000.

```
JS app.js x
1  var express = require('express');
2  var consign = require('consign');
3
4  var app = express();
5  app.set('view engine', 'ejs');
6  app.set('views', './src/views');
7
8  consign()
9    .include('src/routes')
10   .then('src/models')
11   .then('src/controllers')
12   .into(app);
13
14  app.listen(3000, function(){
15    console.log('APP rodando na porta 3000');
16  });
17
```

Exemplo - MVC em Node.JS

```
var express = require('express');
var consign = require('consign');

var app = express();
app.set('view engine', 'ejs');
app.set('views', './src/views');

consign()
  .include('src/routes')
  .then('src/models')
  .then('src/controllers')
  .into(app);

app.listen(3000, function(){
  console.log('APP rodando na porta 3000');
});
```

Abrir no **Visual Studio Code**

Copiar e colar o conteúdo do arquivo **app.js**.

Exemplo - MVC em Node.JS

Passo 6: Acessar a pasta **data** e criar o arquivo **noticias.json**

cmd Prompt de Comando

```
D:\testes_node\mvc-exemplo-nodejs>cd data
D:\testes_node\mvc-exemplo-nodejs\data>type nul > noticias.json
D:\testes_node\mvc-exemplo-nodejs\data>_
```

Passo 7: Abrir no **Visual Studio Code** o arquivo **noticias.json**, copiar e colar o conteúdo.

```
{
  "noticias": [
    {
      "titulo": "Covid-19 29-03",
      "noticia": "O Ministério da Saúde confirmou 4.256 infecções e 136 mortes por coronavírus (Covid-19) no Brasil. Foram 353 novos casos nas últimas 24 horas, com uma taxa de letalidade de 3,2%."
    },
    {
      "titulo": "Coronavírus: fim do isolamento social não impede recessão econômica",
      "noticia": "Os presidentes do Brasil, Jair Bolsonaro, e dos Estados Unidos, Donald Trump, recentemente defenderam o fim do isolamento generalizado das populações e a retomada da vida normal para evitar uma crise maior."
    }
  ]
}
```

Exemplo - MVC em Node.JS

Passo 8: Criar o arquivo `src/routes/news.js`, responsável por realizar o roteamento da aplicação. Nele falamos que ao se acessar a raiz da aplicação, com o método **GET** “<http://localhost:3000/>”. Será chamado o **controller news**.

Siga os passos abaixo:

```
C:\> Prompt de Comando

D:\testes_node\mvc-exemplo-nodejs>cd src

D:\testes_node\mvc-exemplo-nodejs\src>mkdir routes

D:\testes_node\mvc-exemplo-nodejs\src>cd routes

D:\testes_node\mvc-exemplo-nodejs\src\routes>type nul > news.js

D:\testes_node\mvc-exemplo-nodejs\src\routes>
```

OBS: para voltar ao diretório anterior o comando é **cd..**

Exemplo - MVC em Node.JS

Abrir o **Visual Studio Code**, editar o arquivo **src/routes/news.js**, copiar e colar o conteúdo.

```
module.exports = function(application){  
  application.get('/', function(req, res){  
    application.src.controllers.news.index(application, req, res);  
  });  
}
```

Exemplo - MVC em Node.JS

Passo 9: O arquivo **src/controllers/news.js** é o **controller** da aplicação. Primeiramente estamos criando um objeto do model news. Depois estamos utilizando a função **getLastNews** para recuperar as últimas 5 notícias do arquivo **noticias.json**. E por fim passamos as notícias para a view **news/index**. **Siga os passos:**

```

C:\> Prompt de Comando

D:\testes_node\mvc-exemplo-nodejs>cd src
D:\testes_node\mvc-exemplo-nodejs\src>mkdir controllers
D:\testes_node\mvc-exemplo-nodejs\src>cd controllers
D:\testes_node\mvc-exemplo-nodejs\src\controllers>type nul > news.js
D:\testes_node\mvc-exemplo-nodejs\src\controllers>_
```

Exemplo - MVC em Node.JS


Abrir o **Visual Studio Code**, editar o arquivo `src/controllers/news.js`, copiar e colar o conteúdo.

```
module.exports.index = function(application, req, res) {  
  var newsModel = new application.src.models.news();  
  
  newsModel.getLastNews(function(err, result) {  
    res.render("news/index", {news : result});  
  });  
}
```

Exemplo - MVC em Node.JS

Passo 10: O arquivo `src/models/news.js` é o model da aplicação. Primeiramente incluímos o módulo `fs` que auxiliará na leitura do arquivo JSON. Depois criamos a função `news` que vai funcionar como uma classe `news`. Após isto temos a função **`getLastNews`**, que realiza a leitura das notícias contidas no arquivo `noticias.json` e recupera somente as 5 ultimas, conforme a regra de negócio fictícia que foi estabelecida.

Siga os passos:

 Prompt de Comando

```
D:\testes_node\mvc-exemplo-nodejs>cd src
D:\testes_node\mvc-exemplo-nodejs\src>mkdir models
D:\testes_node\mvc-exemplo-nodejs\src>cd models
D:\testes_node\mvc-exemplo-nodejs\src\models>type nul > news.js
D:\testes_node\mvc-exemplo-nodejs\src\models>_
```

Exemplo - MVC em Node.JS

Abrir o **Visual Studio Code**, editar o arquivo **src/models/news.js**, copiar e colar o conteúdo.

```
var fs = require('fs');

function news() {}

news.prototype.getLastNews = function(callback) {
  fs.readFile('./data/noticias.json', 'utf8', function(err, result) {
    var data = [];

    if (!err) {
      var obj = JSON.parse(result);


      if (obj.noticias.length > 4) {
        var i = 4;
      } else {
        var i = (obj.noticias.length - 1);
      }

      obj.noticias.forEach(function(noticia) {
        if (i >= 0) {
          data[i] = noticia;
          i--;
        }
      });
    }
    callback(err, data);
  });
};

module.exports = function(){
  return news;
}
```

Exemplo - MVC em Node.JS

Passo 11: O arquivo `src/views/news/index.ejs` é a view da aplicação. Ela é um arquivo **HTML** simples, onde pegamos as notícias enviadas pelo controller e realizamos uma iteração sobre elas, para exibir todas na tela inicial.

 Prompt de Comando

```
D:\testes_node\mvc-exemplo-nodejs>cd src
D:\testes_node\mvc-exemplo-nodejs\src>mkdir views
D:\testes_node\mvc-exemplo-nodejs\src>cd views
D:\testes_node\mvc-exemplo-nodejs\src\views>mkdir news
D:\testes_node\mvc-exemplo-nodejs\src\views>cd news
D:\testes_node\mvc-exemplo-nodejs\src\views\news>type nul > index.ejs
D:\testes_node\mvc-exemplo-nodejs\src\views\news>
```


Exemplo - MVC em Node.JS

Abrir o **Visual Studio Code**, editar o arquivo **src/views/news/index.ejs**, copiar e colar o conteúdo.

```
<!doctype html>
<html lang="pt">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>MVC Exemplo Node.JS</title>
    <link href="https://getbootstrap.com/docs/4.0/dist/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <br/><br/><br/>
      <h2>MVC Exemplo Node.JS</h2>
      <br/>
      <% if (news.length > 0) { %>
        <div class="row">
          <% for(var i = 0; i < news.length; i++) { %>
            <div class="col-md-12 jumbotron">
              <h4><%= news[i].titulo %></h4>
              <p><%= news[i].noticia %></p>
            </div>
          <% } %>
        </div>
      <% } %>
    </div>
  </body>
</html>
```

Exemplo - MVC em Node.JS

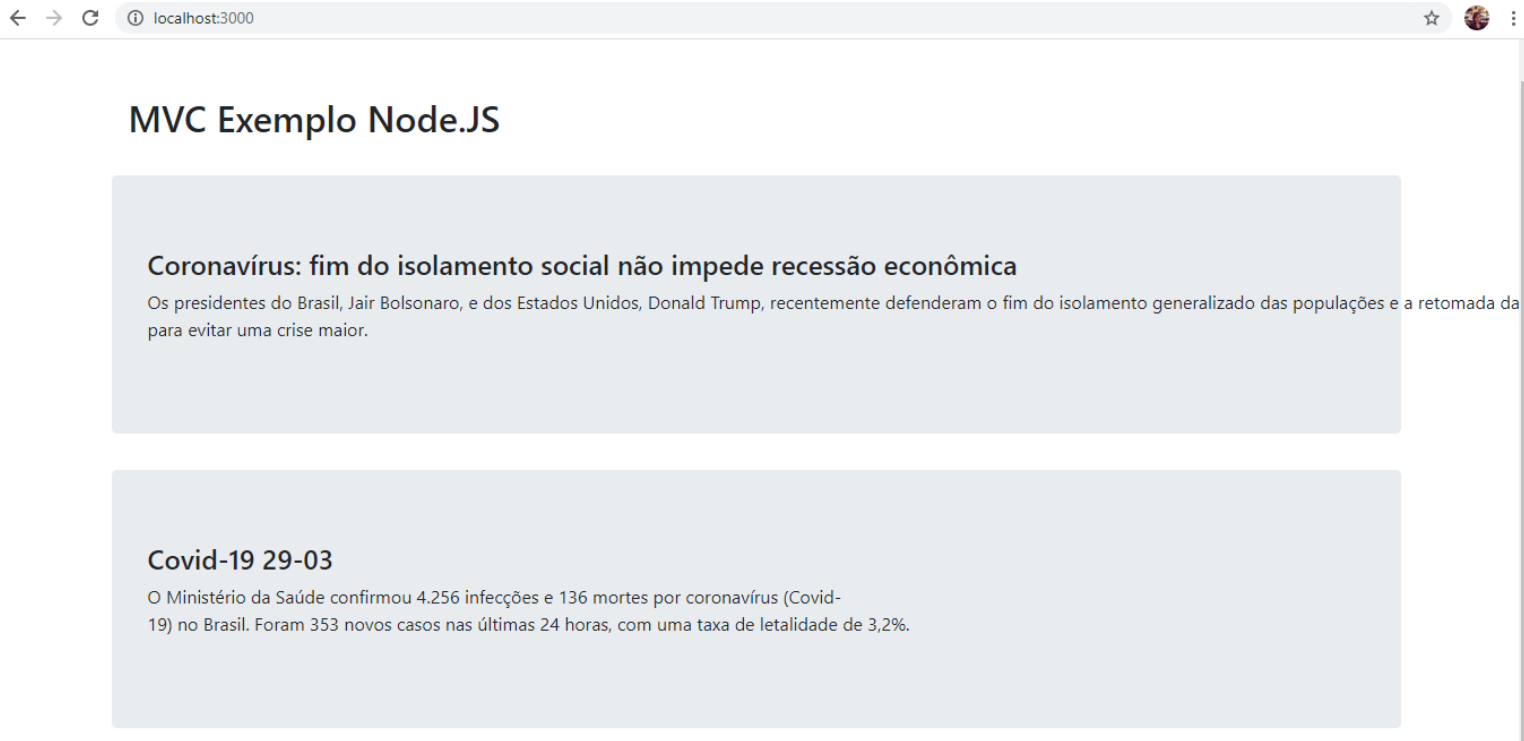
Passo 12: Testando a aplicação. Siga os passos:

 Prompt de Comando - node app.js

```
D:\testes_node\mvc-exemplo-nodejs>node app.js
consign v0.1.6 Initialized in D:\testes_node\mvc-exemplo-nodejs
+ .\src\routes\news.js
+ .\src\models\news.js
+ .\src\controllers\news.js
APP rodando na porta 3000
```

Acesse o navegador para testar: localhost:3000

Exemplo - MVC em Node.JS



Referências

<https://docs.oracle.com/pt-br/solutions/learn-architect-microservice/index.html#GUID-1A9ECC2B-F7E6-430F-8EDA-911712467953>

Node.js in Action, First Edition,
Mike Cantelon, Marc Harter, T.J. Holowaychuk, Nathan Rajlich