# Fundamentos da Programação *Web*

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

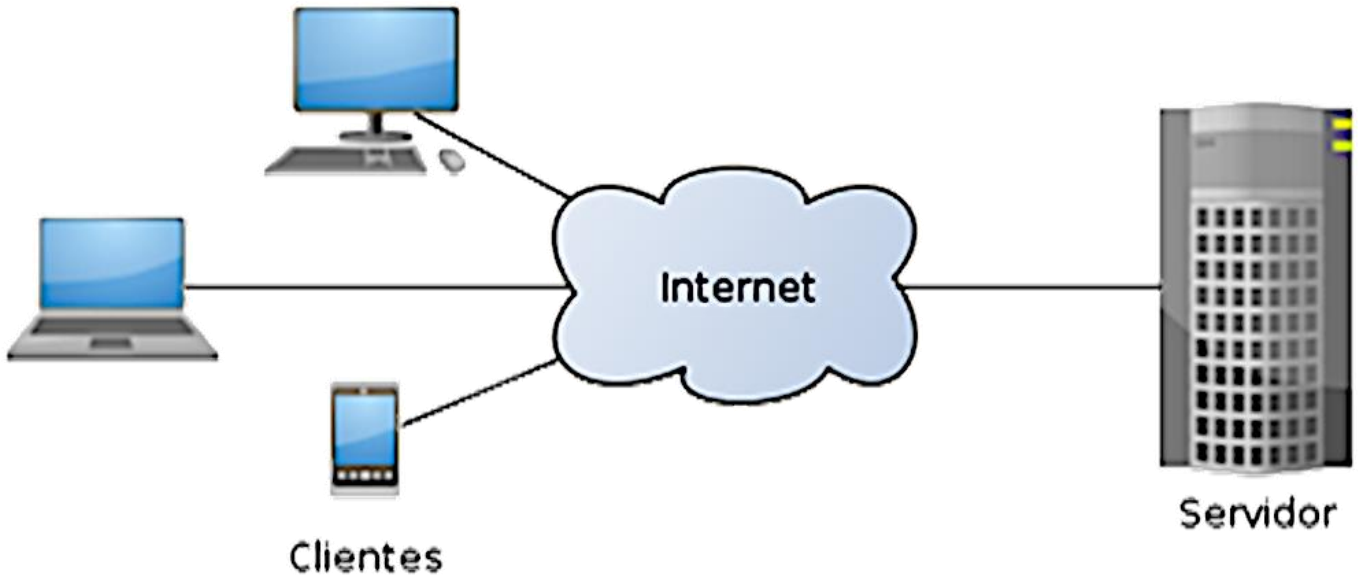
Herculano De Biasi
herculano.debiasi@unoesc.edu.br

# Tópicos

- Cliente-servidor
- *Backend/Frontend*
- IDEs vs Editores
- *Stacks*
- MVC
- Arquiteturas *web*

# CLIENTE-SERVIDOR

- O modelo Cliente-Servidor é um modelo de computação distribuída, desenvolvido na Xerox PARC durante os anos 1970 que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, chamados de clientes
  - Atualmente é considerado um tipo específico de arquitetura multicamada (inspirado nas camadas no modelo OSI)
  - A *world wide web* inteira é baseada na arquitetura cliente-servidor




UNOESC

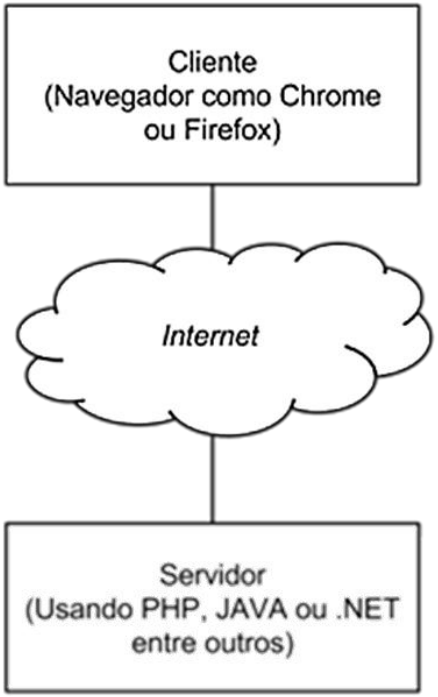
# CLIENTE-SERVIDOR

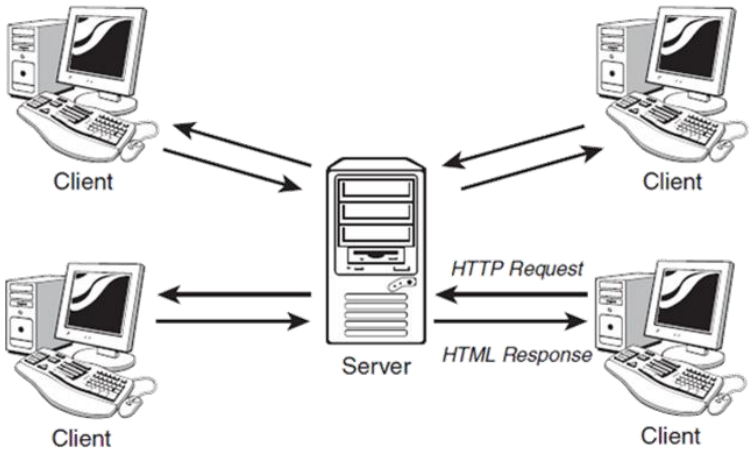
- Servidores
  - Esperam por um pedido de um cliente
  - Atendem os pedidos e, em seguida, respondem aos clientes com os dados solicitados
  - Fornecem recursos de rede
  - Normalmente interagem somente indiretamente com os usuários finais através da interface de usuário do cliente

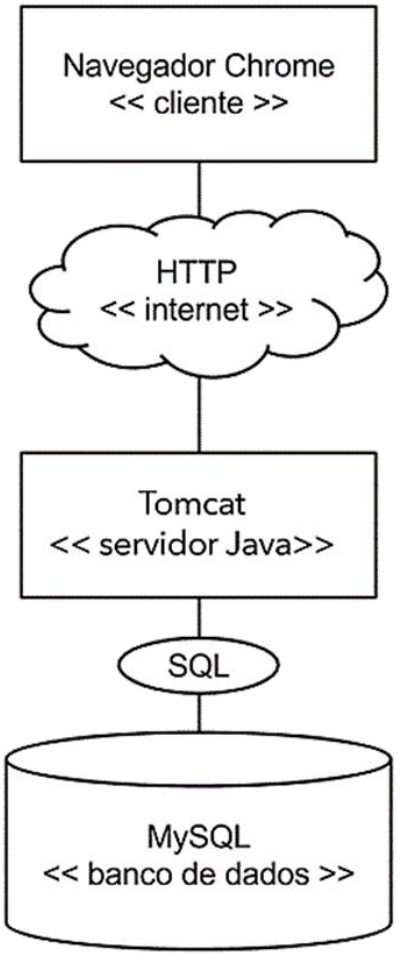
- Clientes
  - Iniciam pedidos para servidores
  - Esperam por respostas e as recebem
  - Normalmente interagem diretamente com os servidores através de seu software de aplicação específico, que lhes possibilita a comunicação com o servidor

# CLIENTE-SERVIDOR

- Desenvolvimento *web*

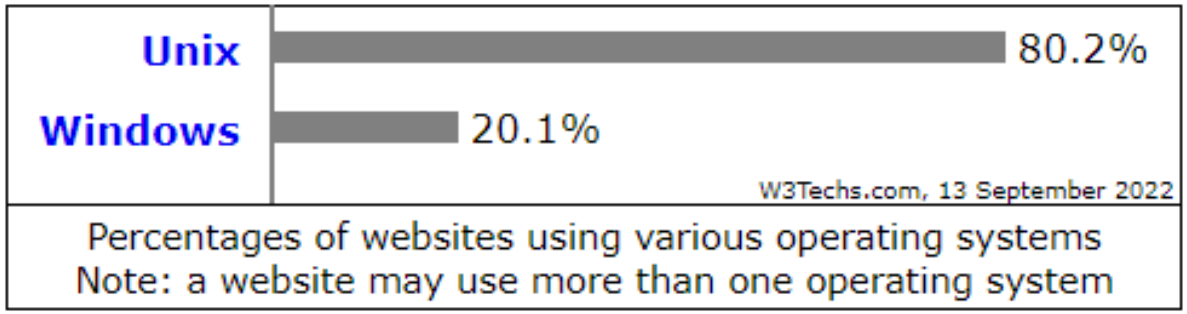


# CLIENTE-SERVIDOR

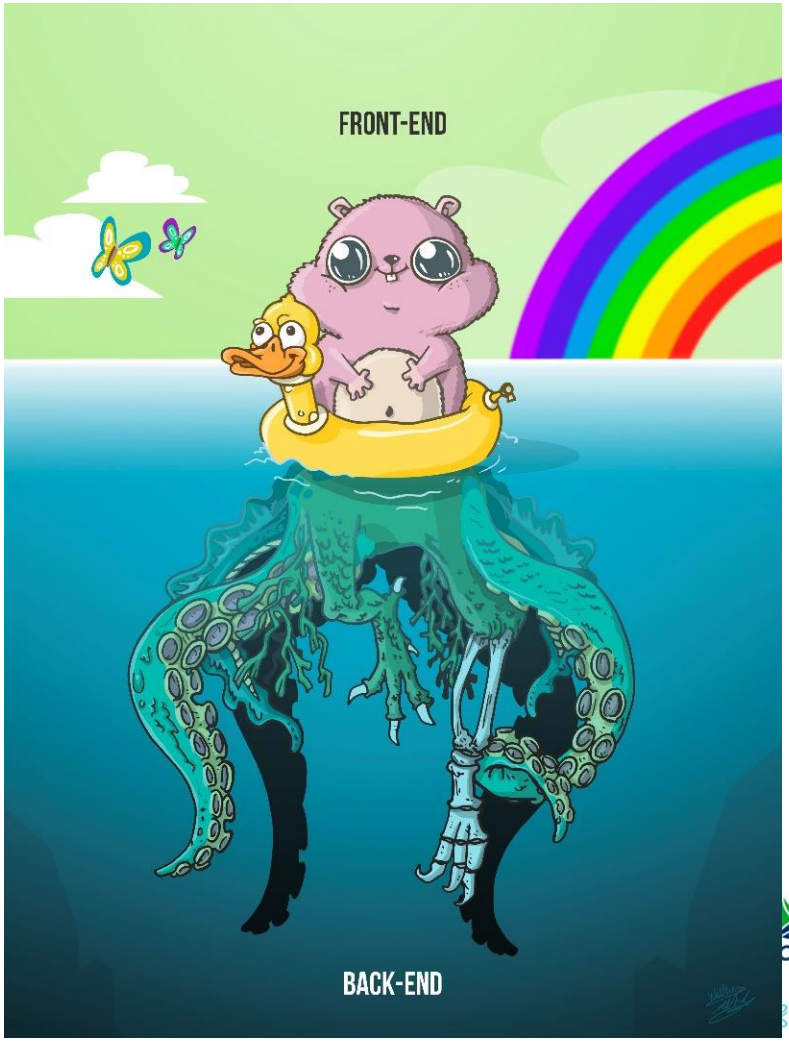
- Plataformas (SO dos *websites*)




Percentages of websites using various operating systems
Note: a website may use more than one operating system

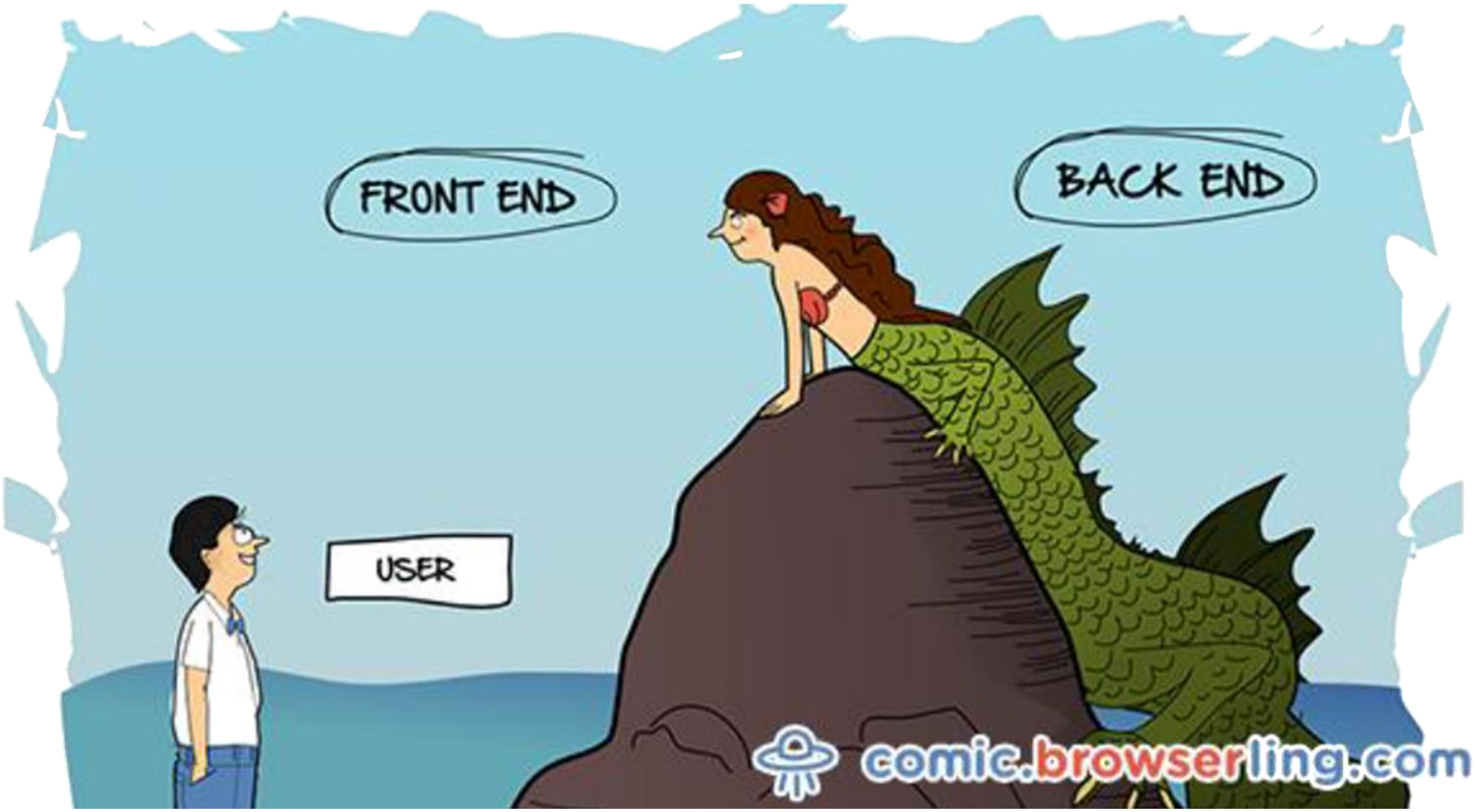
# BACKEND/FRONTEND

- ☺



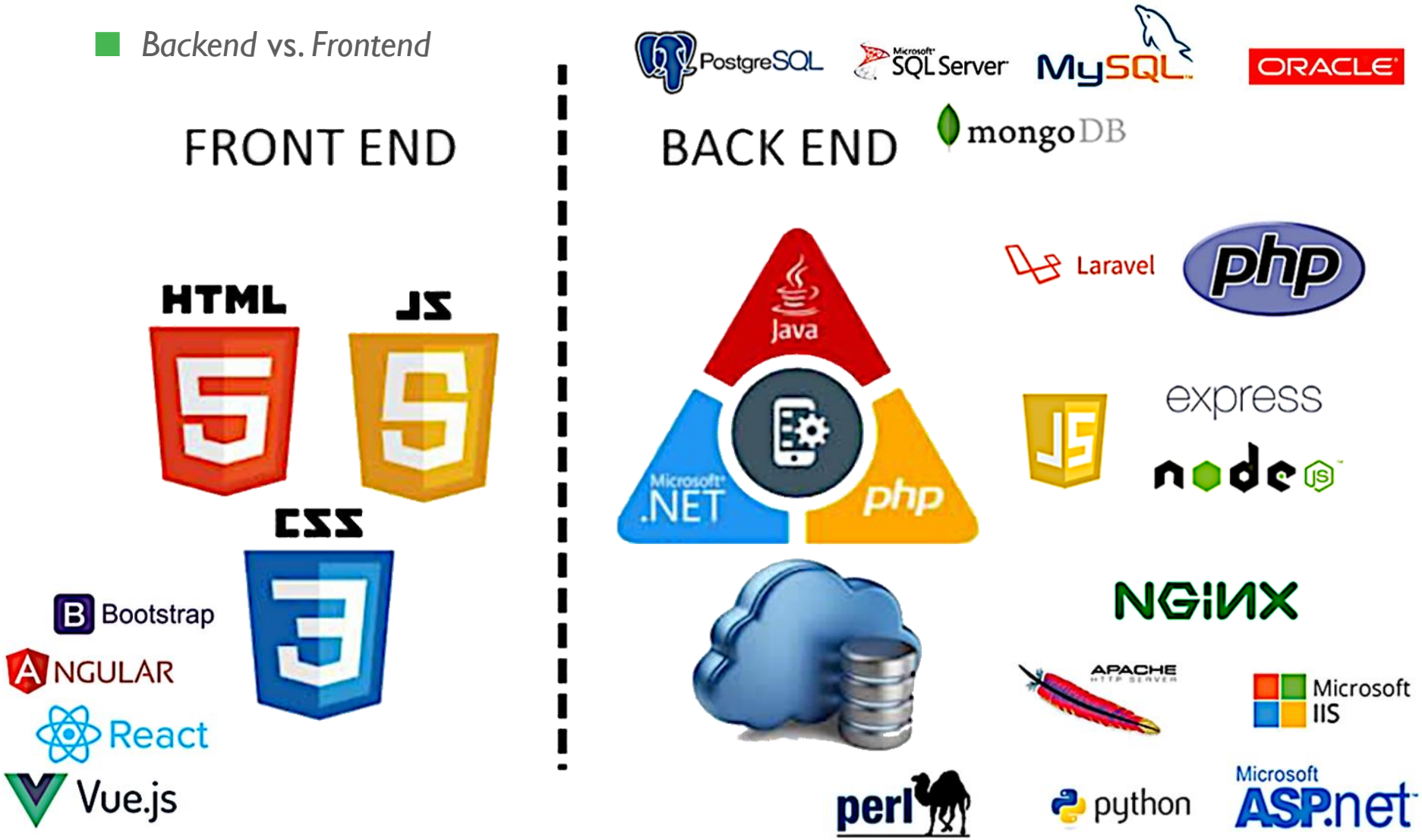
# BACKEND/FRONTEND

- ☺



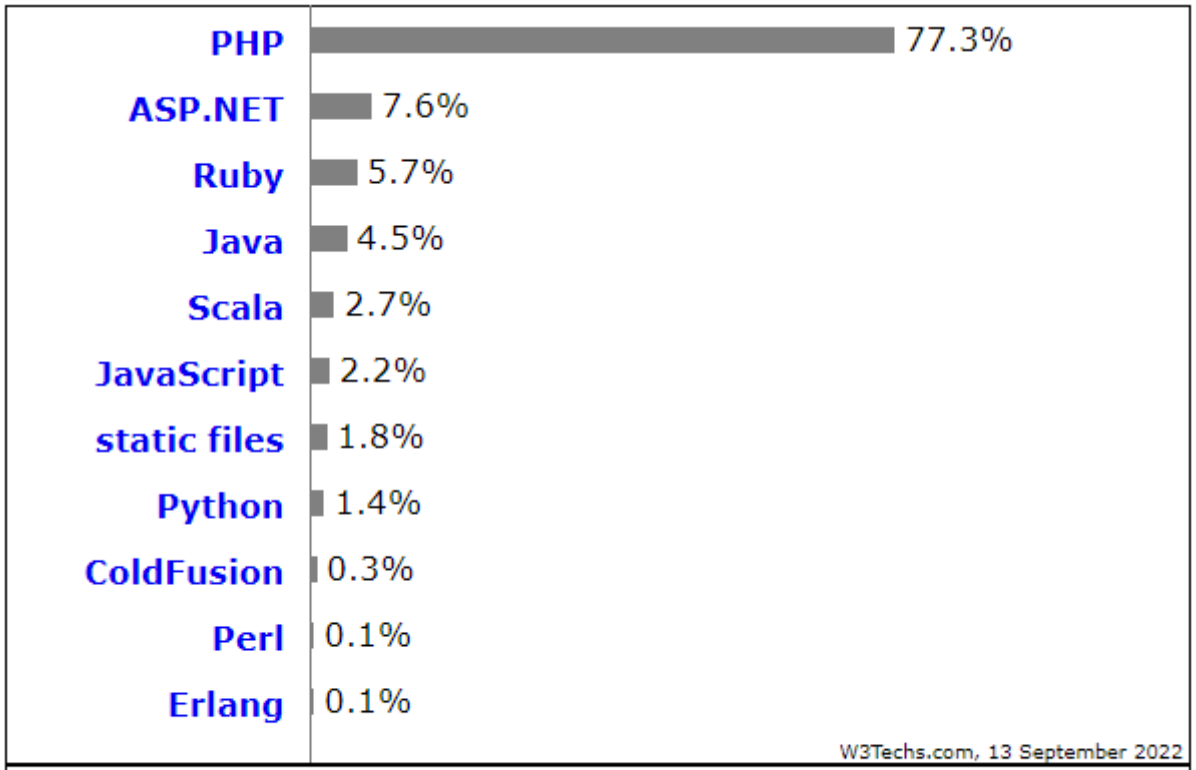
# BACKEND/FRONTEND

- *Backend vs. Frontend*



# BACKEND/FRONTEND

- *Backend*
  - Roda no servidor
  - Não é visto pelo usuário
  - Sistema
  - Regras de negócio
  - Banco de dados
  - Segurança

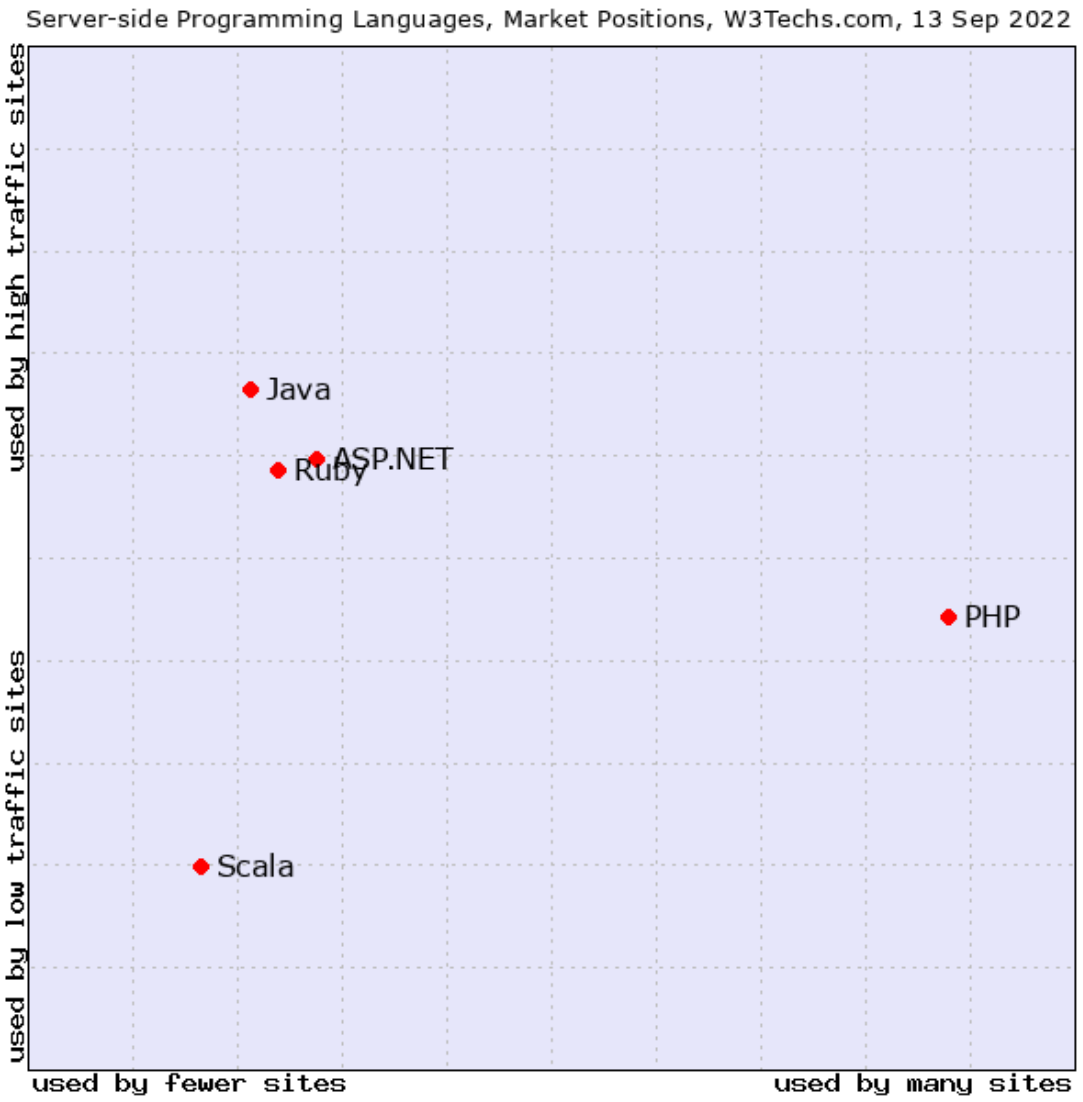



Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

UNOESC

# BACKEND/FRONTEND

- *Backend*
  - Roda no servidor
  - Não é visto pelo usuário
  - Sistema
  - Regras de negócio
  - Banco de dados
  - Segurança

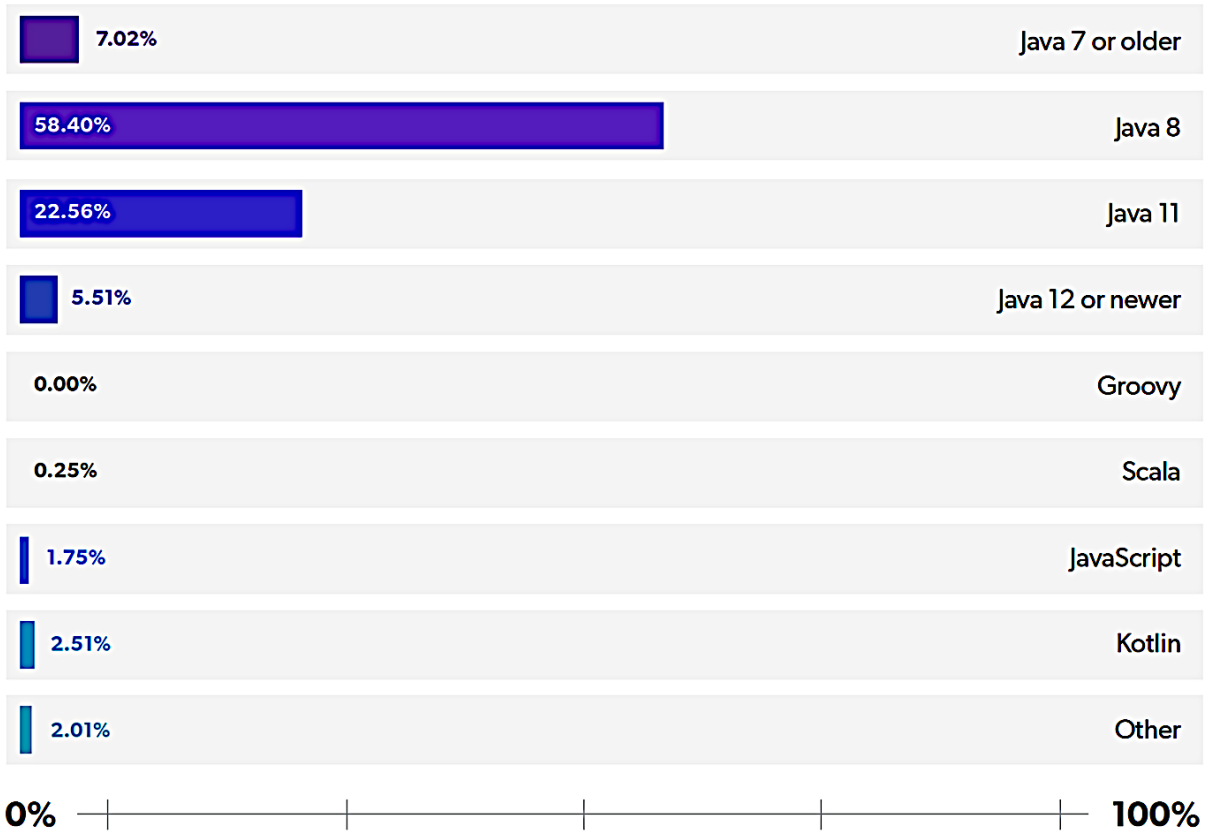



UNOESC

# Backend/Frontend

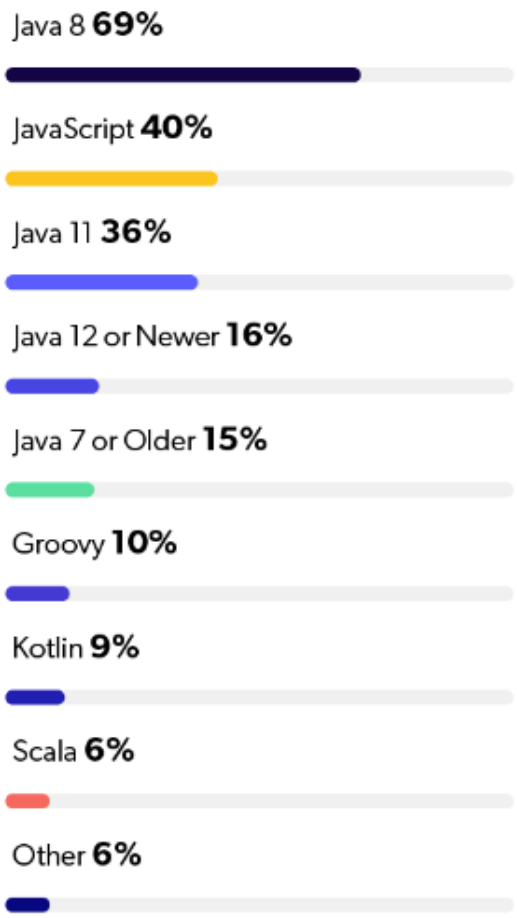
## Java

What Java programming language are you using in your main application?




2020

2021



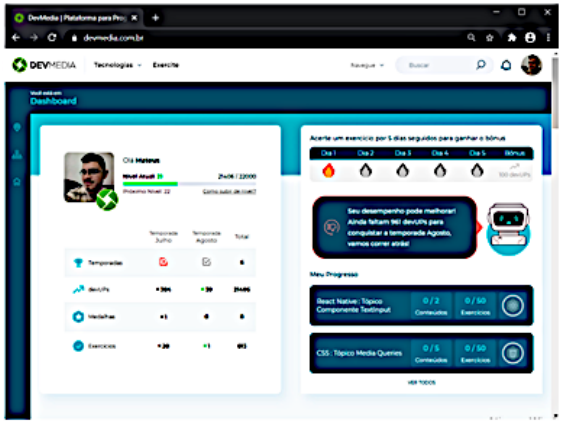
# BACKEND/FRONTEND

- *Frontend*
  - É o que o usuário vê e interage
  - *Interface* gráfica
  - UX – *User Experience*
  - Roda no navegador
  - Telas no navegador
  - *App mobile*


Camada **Back-end**

Foto do usuário
Nome do usuário
Quantidade de devUPs...

O Back-end fornece os dados para o **Front-end**



Camada **Front-end**

O **Angular**, o **React** ou o **Vue.js** exibe os dados na tela do usuário de forma amigável

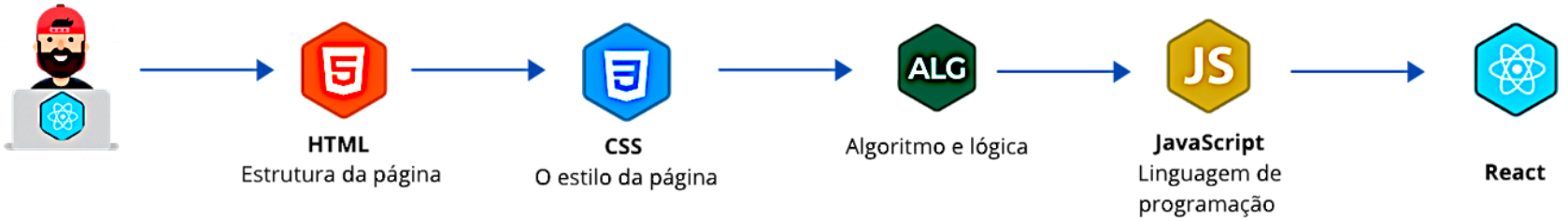
UNOESC

# BACKEND/FRONTEND

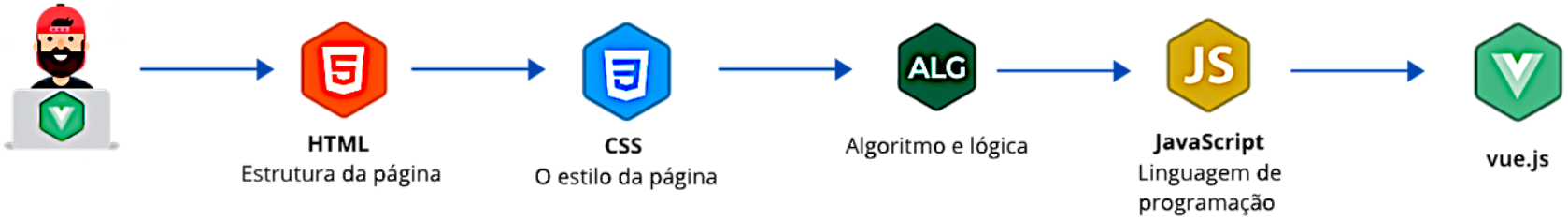
- *Frameworks* para desenvolvimento *web*

Programador Front-end **Angular**


Programador Front-end **React**

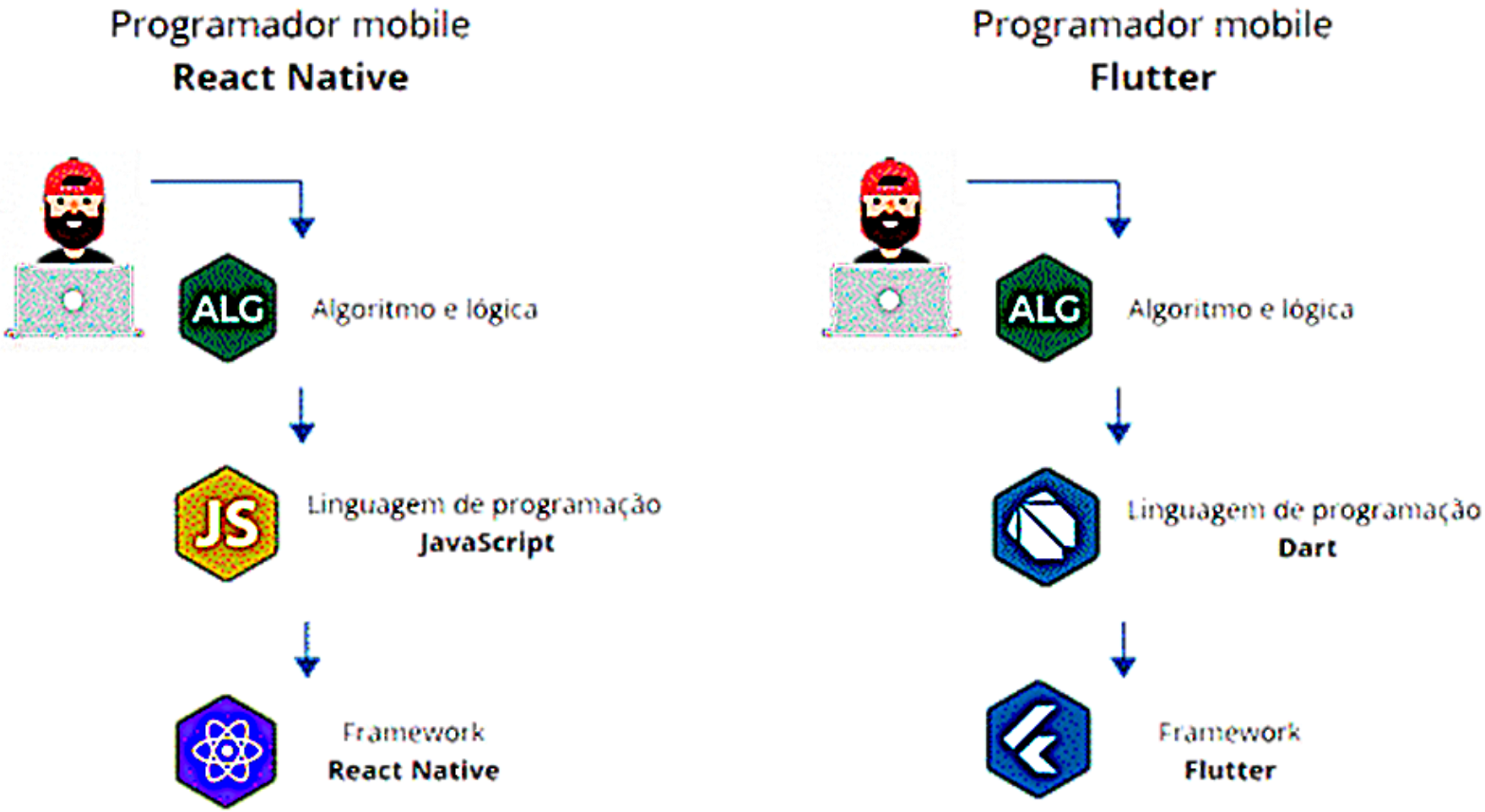



Programador Front-end **Vue.js**



# BACKEND/FRONTEND

- *Frameworks* para desenvolvimento *mobile*



# BACKEND/FRONTEND

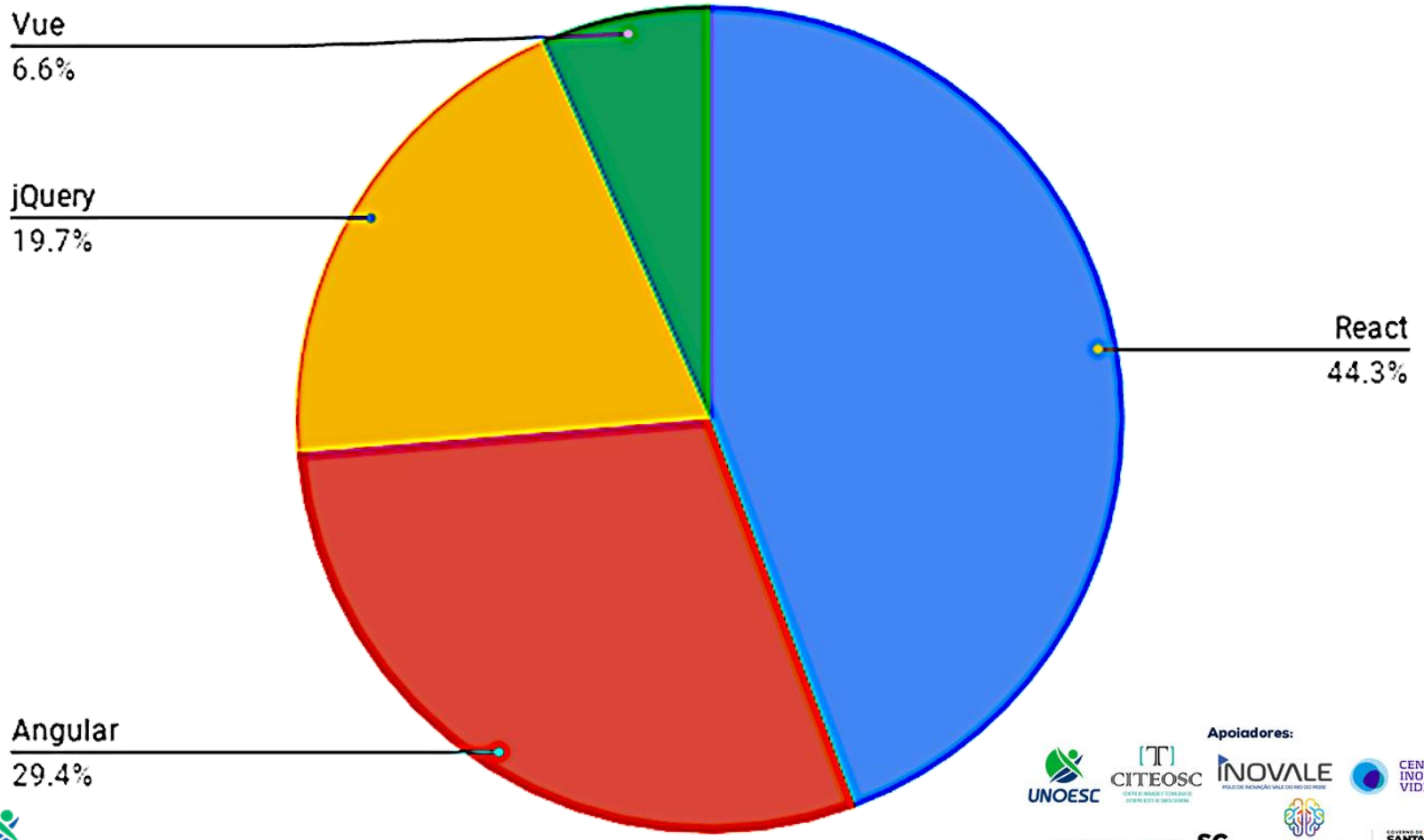
- *Frameworks* CSS mais populares (2020), segundo GitHub, Slant.co, StackShare e Stackoverflow

| Parameters | Bootstrap | Bulma | Foundation | MaterializeCSS | Ulkit |
|---|---|---|---|---|---|
| Release date | Aug 19, 2011 | Jan 24,2016 | Sep 2011 | Nov 4,2014 | Jul 19, 2013 |
| Licence | MIT | MIT | MIT | MIT | MIT |
| Git star, forks, contributors | 141k 68.7k 1115 | 39.6k 3.4k 302 | 28.6k 5.8k 982 | 37.5k 4.8k 250 | 15.7k 2.2k 28 |
| Hacker News, Reddit, StackOverflow | 3.3k 9k 98k | - 1.2k 581 | - 1.2k 803 | - 374 3.2k | 139 21 8.2k |
| Current version | 4.4.1 | 0.8.2 | 6.6.3 | 1.0.0 | 3.4.3 |
| Size | Minified CSS: ~61.7 kB Gzip: ~15.8 kB | Minified CSS: ~209.5 kB Gzip: ~25.9 kB | Minified CSS: ~139.2 kB Gzip: ~34 kB | Minified CSS: ~175.2 kB Gzip: ~41.5 kB | Minified CSS: ~131.2 kB Gzip: ~41.1 kB |
| No. Of Sites | ~20,737,671 | ~30,987 | ~441,292 | ~111,481 | ~311,897 |
| Language & Pre-Processor | HTML5, CSS3, JS, Sass | HTML5, CSS3, Sass | HTML5, CSS3, JS, Sass | HTML5, CSS3, JS, Sass | HTML5, CSS3, JS, Sass, {less} |
| Grid | Mobile-first Flexbox grid, 12 column system | Power of Flexbox grid with 12 Column in a simple interface | XY 12- Column Grid,Floated (flexbox in latest version) | Standard 12 column fluid responsive grid system | Fully responsive, fluid and nestable grid layout. |
| Description | Most popular HTML, CSS, and JS framework | CSS framework based on Flexbox and built with Sass | Most advanced responsive front-end framework | CSS framework based on Material Design | A lightweight and modular front-end framework |
| Core concept | RWD and mobile first | RWD,mobile first, Modern free | RWD,mobile first, semantic | RWD and based on Google Material Design | RWD and mobile first |
| Unique Features | Card component, Jumbotron, Responsive navbar | - | Iconbar, Flex video, Pricing table | Scroll fire, Toast, Parallax, | Article, Cover, HTML editor |
| Browser Support (L) - Latest | L L L L L 10+ v5.0+ | L L L L L 10+ | L L L L L 9+ v4.4+ | 35+ 9+ 31+ L L 11+ | L 9.1+ L L L 11+ |
| Javascript skill require | + − | − | + | + | + − |
| Learning Curve | Mild | Mild | Steep | Mild | Moderate |
| Documentation | Excellent | Good | Good | Good | Good |

UNOESC

# BACKEND/FRONTEND

- *Frameworks* JavaScript (2019)



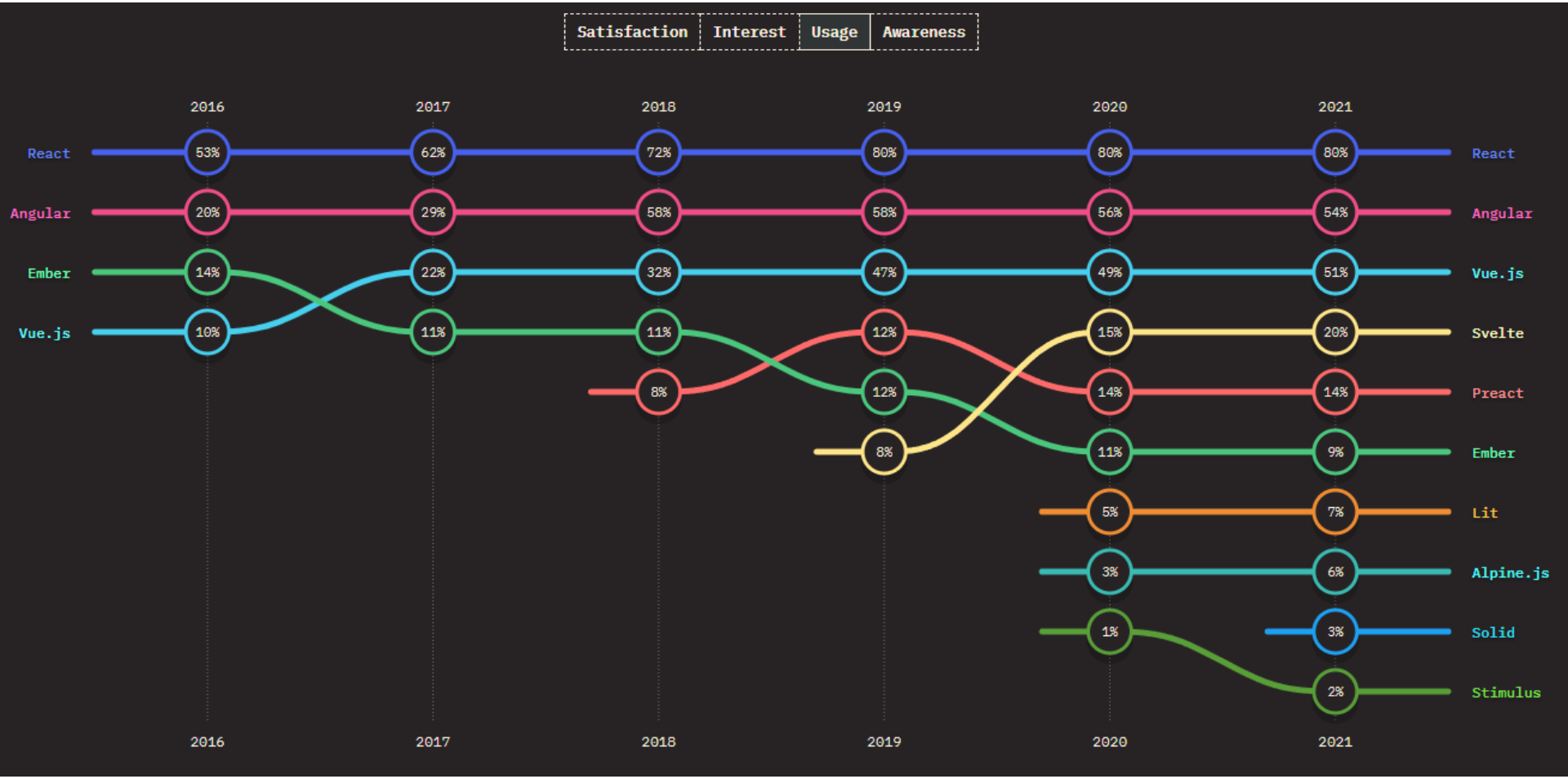

UNOESC

Apoiadores:

UNOESC CITEOSC INOVALE CENTRO DE INOVAÇÃO VIDEIRA DEV-TI SC TEC fapesc SANTA CATARINA

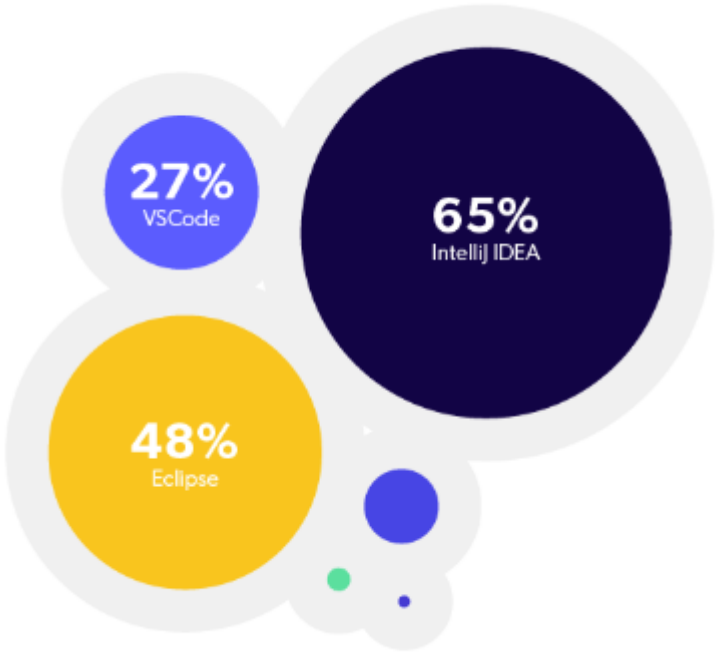
# Backend/Frontend

- *Frameworks* JavaScript *frontend* (2021)



# IDEs vs. Editores

- IDEs (2021)




IntelliJ IDEA **65%**

Eclipse **48%**

VSCode **27%**

NetBeans **13%**

Browser-Based IDE **4%**

Other **2%**

Apoiadores:

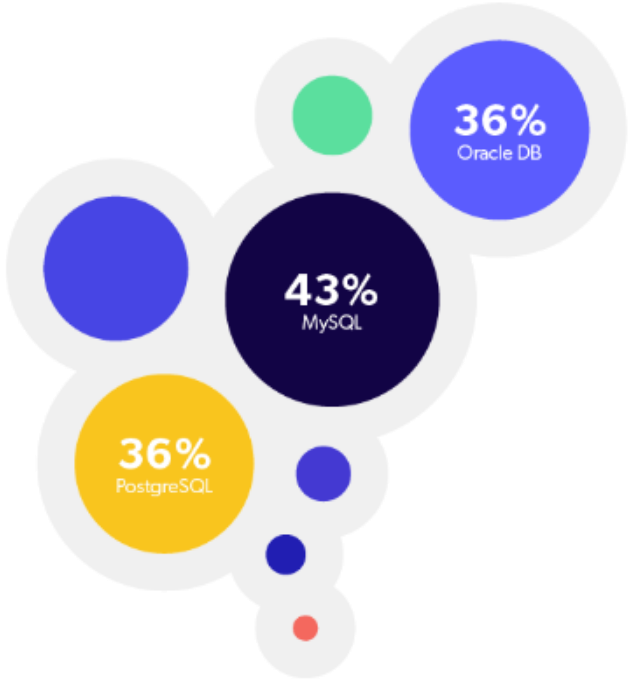
UNOESC

UNOESC CITEOSC INOVALE CENTRO DE INOVAÇÃO VIDEIRA

DEV-TI SC TEC fapesc GOVERNO DE SANTA CATARINA

# Banco de Dados

- IDEs (2021)




MySQL **43%**

Oracle DB **36%**

PostgreSQL **36%**

MongoDB **29%**

Other SQL **16%**

Other NoSQL **11%**

Cassandra **8%**

Neo4j **5%**

Apoiadores:

UNOESC

CITEOSC

INOVALE

CENTRO DE INOVAÇÃO VIDEIRA

DEV-TI

SC TEC

fapesc

SANTA CATARINA

UNOESC

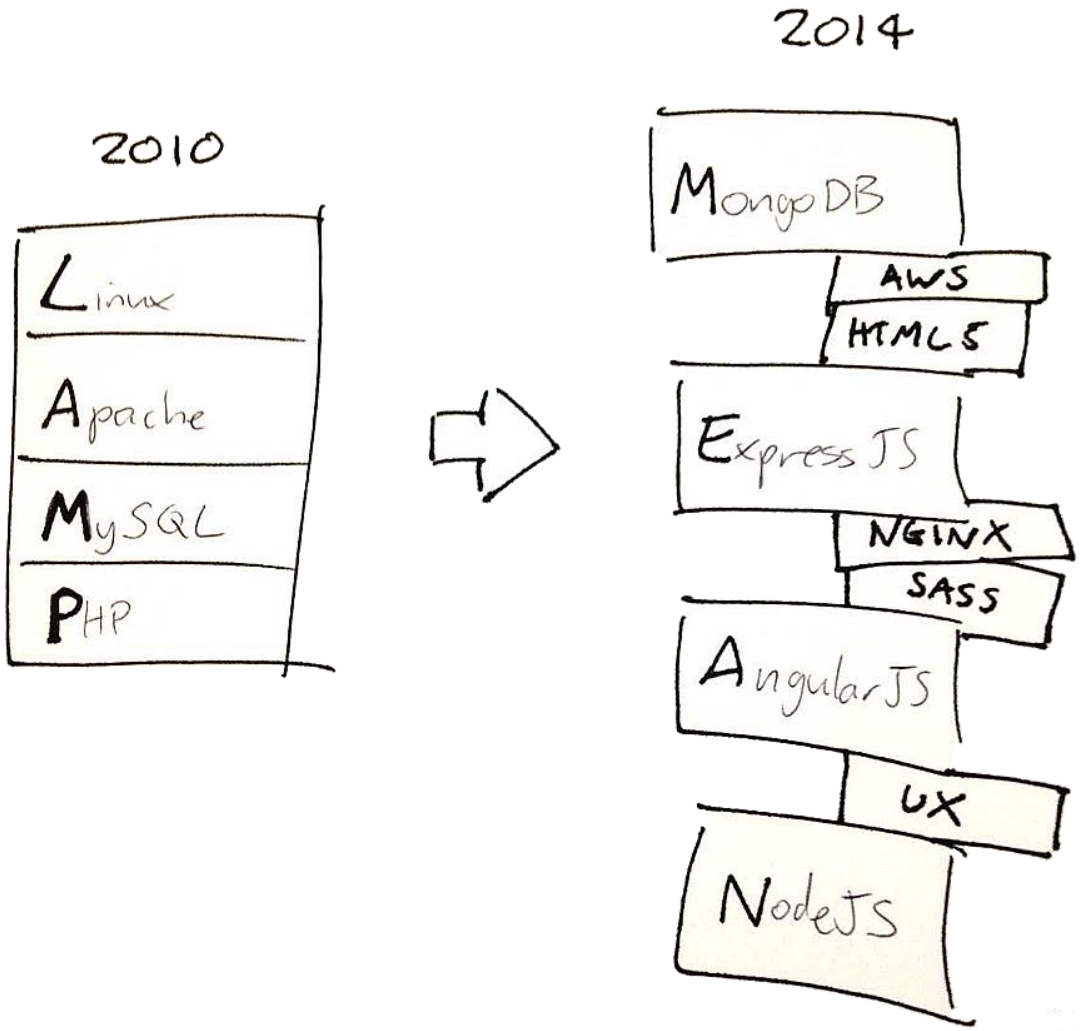
# STACKS

- Uma *stack* é um conjunto de componentes ou tecnologias de software necessários para criar uma plataforma completa com todas as ferramentas necessárias para atingir um determinado objetivo em um domínio específico

- No caso de desenvolvimento de aplicações *web*, a *stack* inclui o sistema operacional, o servidor *web*, o sistema gerenciador de banco de dados e a linguagem de programação

# STACKS

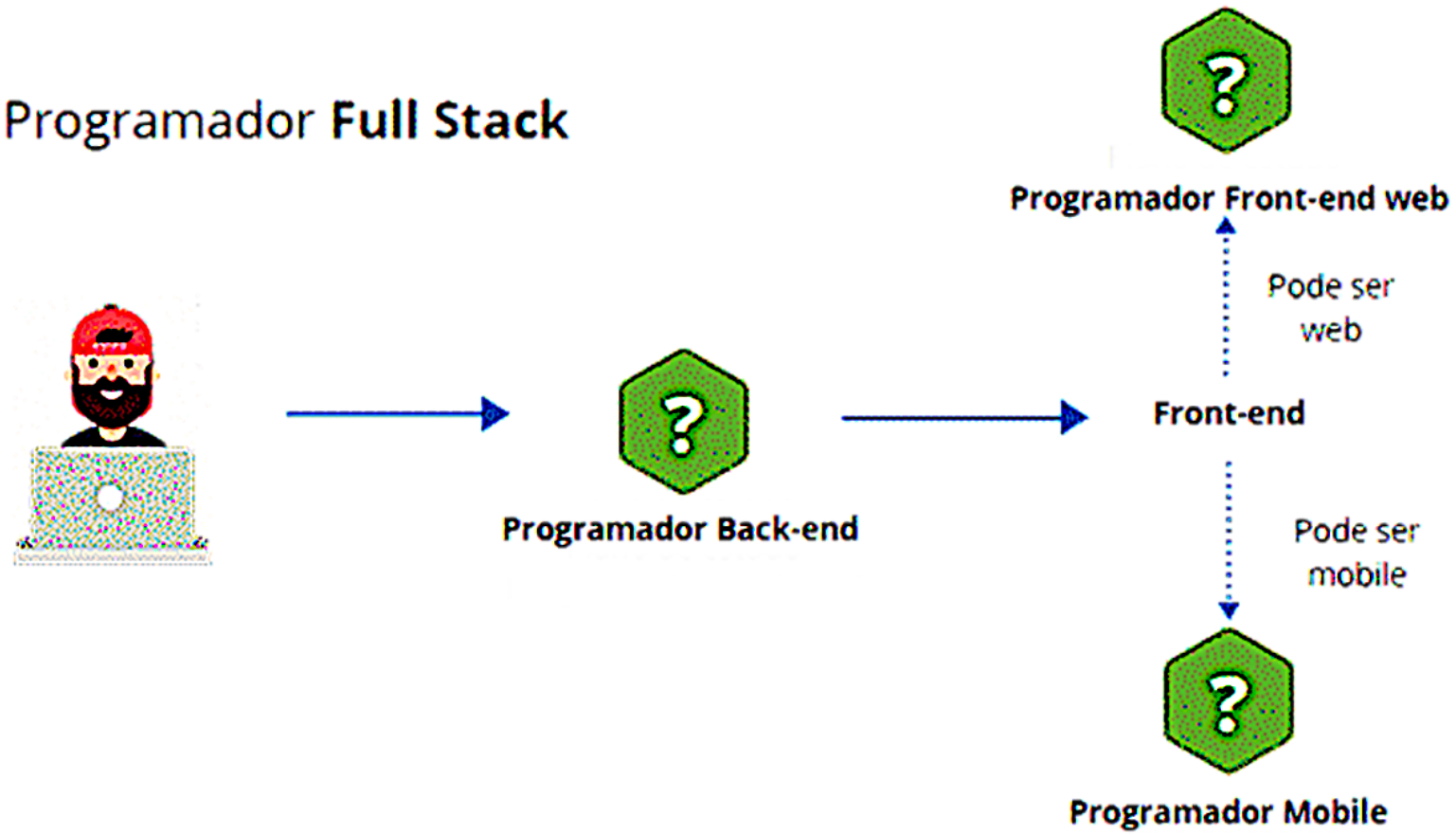
- Exemplos
  - LAMP
    - Linux
    - Apache
    - MySQL/MariaDB
    - PHP/Python/Perl
  - MEAN
    - MongoDB
    - Express.js
    - AngularJS
    - Node.js




UNOESC

# STACKS

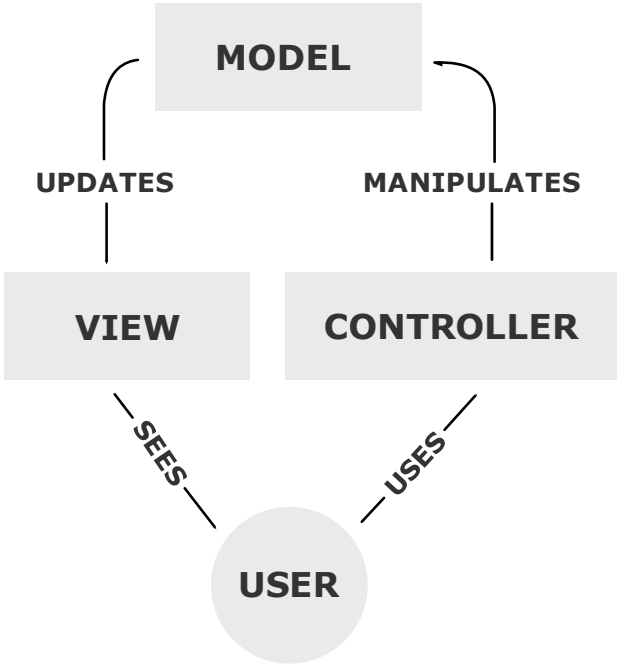
- Desenvolvedor *full stack* Profissional capaz de trabalhar com toda a *stack* de desenvolvimento (*frontend* e *backend*), além do banco de dados, servidor e *mobile*



# MVC

- MVC (*Model-View-Controller*): Modelo-Visão-Controlador
  - Padrão de projeto e de arquitetura de software que separa a representação da informação da interação do usuário
  - Descrito pela primeira vez em 1979 por Trygve Reenskaug, que trabalhava no Smalltalk na Xerox PARC

- Componentes
  - Modelo: Dados da aplicação, regras de negócios, lógica e funções
  - Visão: Qualquer saída de representação dos dados, como uma tabela ou diagrama
  - Controlador: Faz a mediação da entrada de dados, convertendo-a em comandos para o modelo ou visão




UNOESC

# MVC

- Em aplicações web
  - Modelo: Objetos de domínio que tratam da lógica de negócios, persistência, etc
  - Visão: Documento HTML
  - Controlador: Recebe as requisições `GET` ou `POST` e as redireciona para processamento

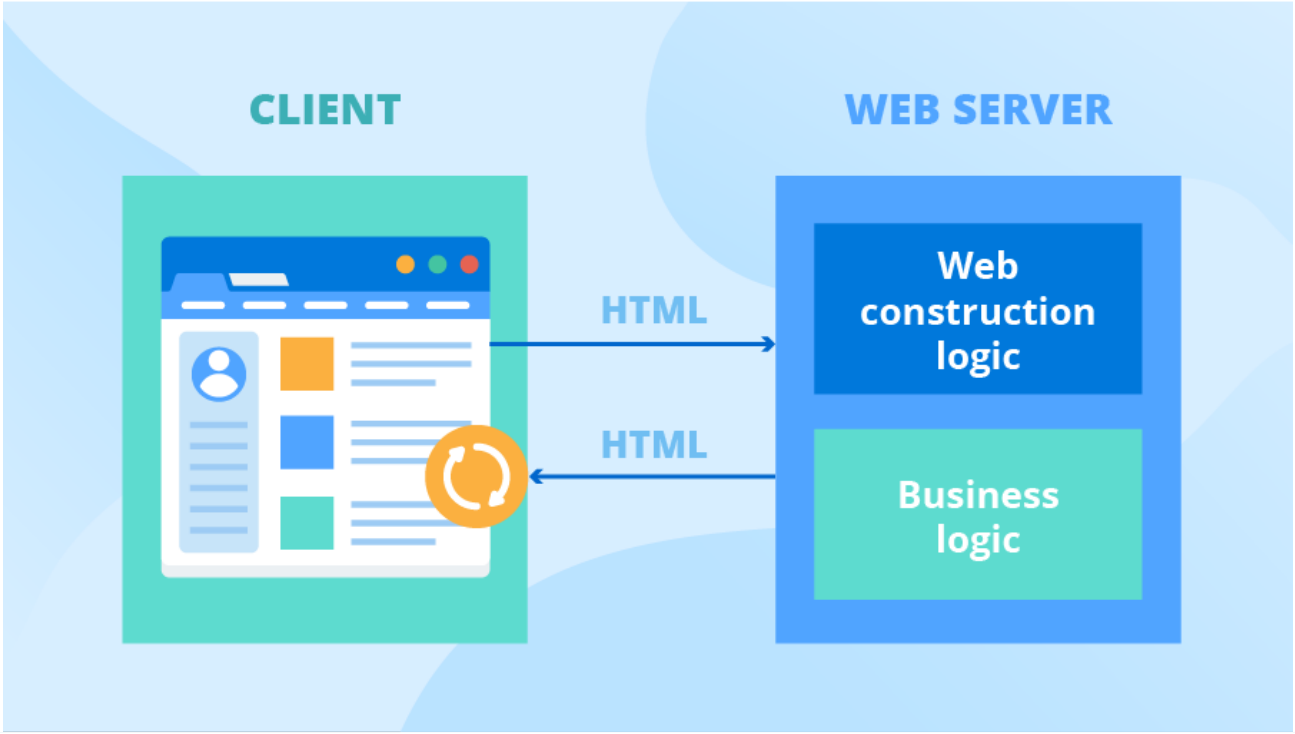
- Extensões
  - HMVC: *Hierarchical Model-View-Controller*
  - MVA: *Model-View-Adapter*
  - MVP: *Model-View-Presenter*
  - MVVM: *Model-View-Viewmodel*
  - MVW: *Model-View-Whatever*

# ARQUITETURAS *WEB*

- Tipos de arquiteturas de aplicações *web*
  - Arquitetura tipo 1: *Server-Side* (HTML)
  - Arquitetura tipo 2: AJAX (*widget web app*)
  - Arquitetura tipo 3: SPA (*Single-Page Web Application*)
  - *Single Page Application* (SPA) vs. *Multiple Page Application* (MPA)

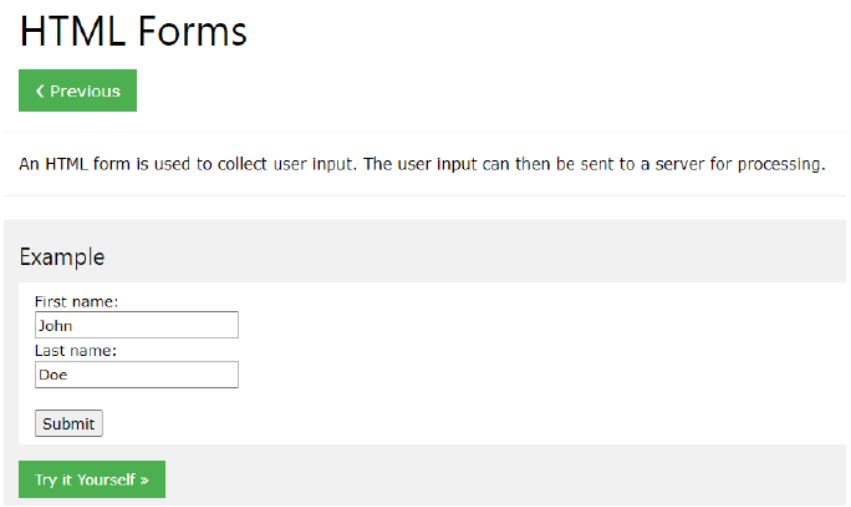
# Arquiteturas *Web*

- Arquitetura tipo 1: *Server-Side* (HTML)
  - Arquitetura em duas camadas (*two-tiered*): Um servidor contém a lógica da aplicação, tanto das regras de negócios quanto da construção das páginas
  - A interação é feita através de um navegador renderizando páginas HTML
  - Para ver uma atualização o cliente precisa atualizar completamente a página
  - Servidor cria as páginas HTML e as envia para o navegador



# ARQUITETURAS *WEB*

- Arquitetura tipo 1: *Server-Side* (HTML)
  - Utilização de formulários e troca de informações entre o cliente e servidor através das requisições GET e POST
  - Tecnologias como PHP e ASP muito utilizadas



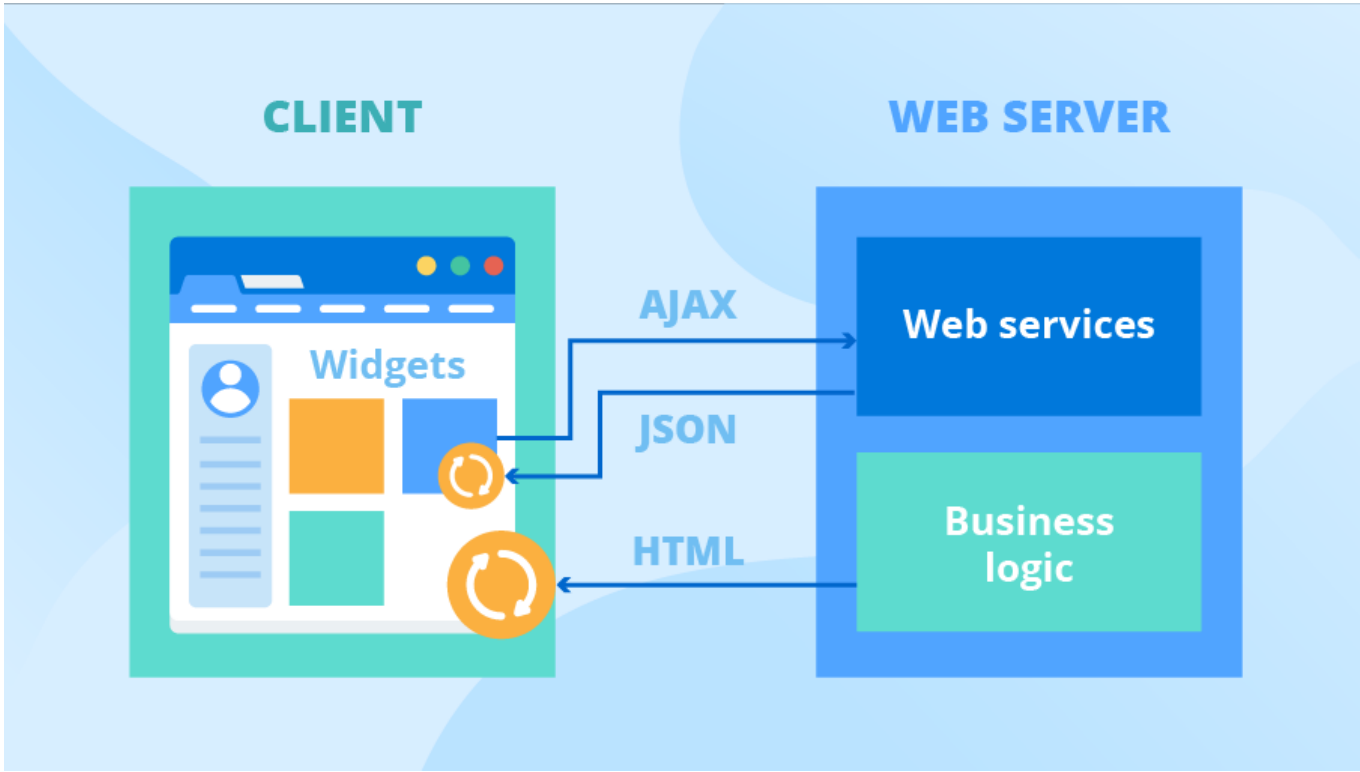

```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called
"/action_page.php".</p>

</body>
</html>
```

UNOESC

Apoiadores:

# ARQUITETURAS *WEB*

- Arquitetura tipo 2: AJAX (*widget web app*)
  - A lógica de construção das páginas é substituída por *web services* e cada página possui entidades separadas chamadas de *widgets*
  - Consultas AJAX aos *web services*, permitem a atualização do cliente sem recarregar as páginas completamente, tornando-as mais dinâmicas e amigáveis



# ARQUITETURAS *WEB*

- Arquitetura tipo 2: AJAX (*widget web app*)
  - Utilização de JavaScript e DOM
  - Requisições assíncronas
  - Classe `XMLHttpRequest` e método `onreadysatechange()`

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

# ARQUITETURAS *WEB*

- Arquitetura tipo 3: SPA (*Single-Page Web Application*)
  - A página é transferida ao cliente uma única vez, sendo que as atualizações posteriores são feitas através de uma camada JavaScript que se comunica com *web services* no lado servidor, obtendo atualizações em tempo real
  - Torna fácil a transformação do *web app* em um *app mobile* híbrido