



PROGRAMAÇÃO
DO ZERO com
**Renato
Gava**

Bem-vindos!

- Este curso vai te dar o conteúdo necessário para que você **se transforme** em um programador.
- O que vocês vão fazer depois que aprenderem a programar?
- Um pouco sobre mim

Orientações sobre o Curso

- Como vai ser o curso? Vamos desenvolver um sistema juntos do zero
- Curso dividido em 8 Módulos
- Aulas toda terça e quinta, não vai perder!
- Todas as aulas serão gravadas
- E se cair minha internet? E se cair a sua internet?
- Não entendeu de primeira? Não tem problema – decora.
- Prestar atenção na aula – desliga o celular!
- Faça os Exercícios
- Tire suas Dúvidas durante a aula e no grupo do Telegram
- Certificado de Conclusão, se cadastrar: <https://forms.gle/RcRQvtYZLeDtsMsB6>

Módulos do Curso

1. Introdução a Programação
2. Lógica de Programação e Algoritmos
3. Front-end Web
4. API
5. Programação Orientada a Objetos
6. Banco de Dados
7. Controle de Código Fonte
8. Deployment

AULA 1

Introdução a Programação

Você vai aprender na aula de hoje

- O que é Programação
- O Mercado de Trabalho para Programadores
- Como funciona um computador
- Hardware e Software
- Linguagem de Programação
- IDE

20:45 Intervalo de 15 minutos

- Instalando o Visual Studio e Baixando a Estrutura de Aulas

O que é Programação

- Programação é a forma como nos comunicamos com os computadores.
- Programar é o processo de **construção de um programa** de computador.
- O programa é escrito em uma **linguagem de programação**.
- Programador é o profissional de **constrói programas** de computador utilizando uma linguagem de programação.

Mercado de Trabalho

Como ganhar dinheiro e trilhar uma carreira de sucesso?

- Acumule Experiência no campo de batalha
- Aprenda sob demanda
- Se especialize em uma tecnologia e tire Certificações Oficiais
- Aprenda Inglês!



🔍 programador

📍 Brasil

Pesquisar



Vagas ▾

Data do anúncio ▾

Nível de experiência ▾

Empresa ▾

Tipo de vaga ▾

Presencial/remoto ▾

Todos os filtros 1

Programador em: Brasil

1.693 resultados

Configurar alerta



Analista Programador Android Sr.

BRQ Digital Solutions

Brasil (Remoto)



2 conexões trabalham aqui

Promovida • 19 candidaturas • Candidate-se facilmente



355 - Analista Programador .NET (sênior)

Prime IT Solutions

Brasil (Remoto)



Seu perfil corresponde a esta vaga

Promovida • Candidate-se facilmente



Programador de sistemas

Newcon Software S/A

Tupã, São Paulo, Brasil (Presencial)

Promovida



Analista Programador IOS PL (Flutter)

Tata Consultancy Services

São Paulo, Brasil (Remoto)

Analista Programador Android Sr.



BRQ Digital Solutions • Brasil (Remoto) há 2 semanas • 19 candidatos



Tempo integral • Pleno-sênior



1.001-5.000 funcionários • Serviços e consultoria de TI



2 conexões



Veja as tendências de contratação recentes na BRQ Digital Solutions. [Experimente o Premium grátis](#)

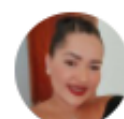


Recrutando agora

Candidate-se agora

Salvar

Anunciada por



Ana Julia Santos 2°

Tech Recruiter Jr. na BRQ Digital Solutions

PREMIUM
[Envie InMail](#)

Analista Programador Android Sr.

Mercado de Trabalho

Como ganhar dinheiro e trilhar uma carreira de sucesso?

CLT – famoso carteira assinada

- **Estagiário:** mais erra do q acerta R\$ 1.200,00
- **Júnior:** precisa de muita ajuda R\$ 2.000,00
- **Pleno:** dá conta do recado R\$ 3.000,00 a R\$ 5.000,00
- **Sênior:** muita experiência, estrutura, ensina e ajuda. R\$ 13.000,00 a R\$ 15.000,00
- **Fullstack:** conhece de front-end, back-end, banco de dados e deployment. R\$ 13.000,00 a R\$ 15.000,00

Mercado de Trabalho

Como ganhar dinheiro e trilhar uma carreira de sucesso?

- PJ – Abre uma empresa e presta serviço. Cobra por hora
- Freelancer – projetos e trabalhos pontuais. Cobra por job
- Empreendedor – criar seus próprios produtos: vender, alugar, modelo freemium ou free com ads.
- Mais de um ao mesmo tempo

DÚVIDAS?

Como funciona um Computador

- Como funciona um Computador
- Dividido em 2 Partes: Hardware e Software
- Hardware: parte física (mouse, teclado, monitor etc)
- Software: parte lógica (sistema operacional, **programas** etc)

Exemplos de Hardware

- Teclado
- Mouse
- Monitor
- HD ou SSD
- Processador
- Memória



Exemplos de Software

Sistemas Operacionais

- Windows
- Android
- iOS



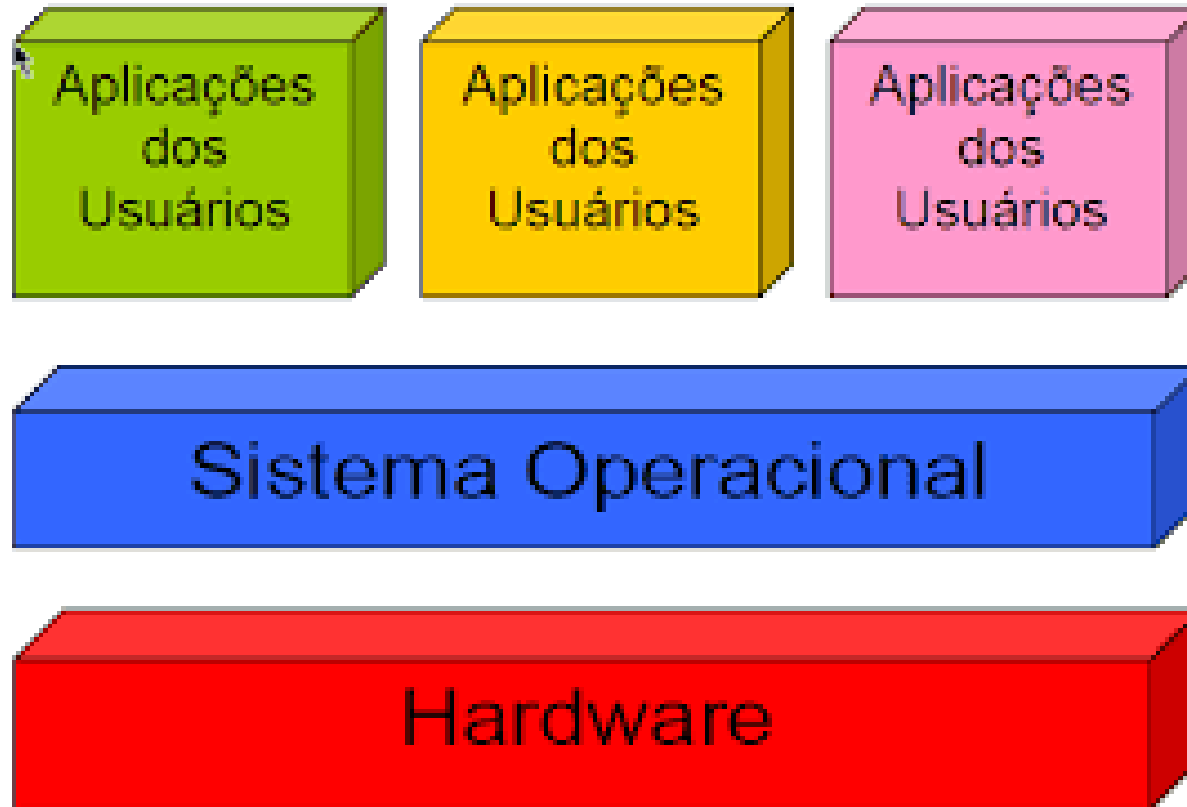
Programas

- WhatsApp
- iFood
- Instagram



Hardware e Software Juntos

Programas



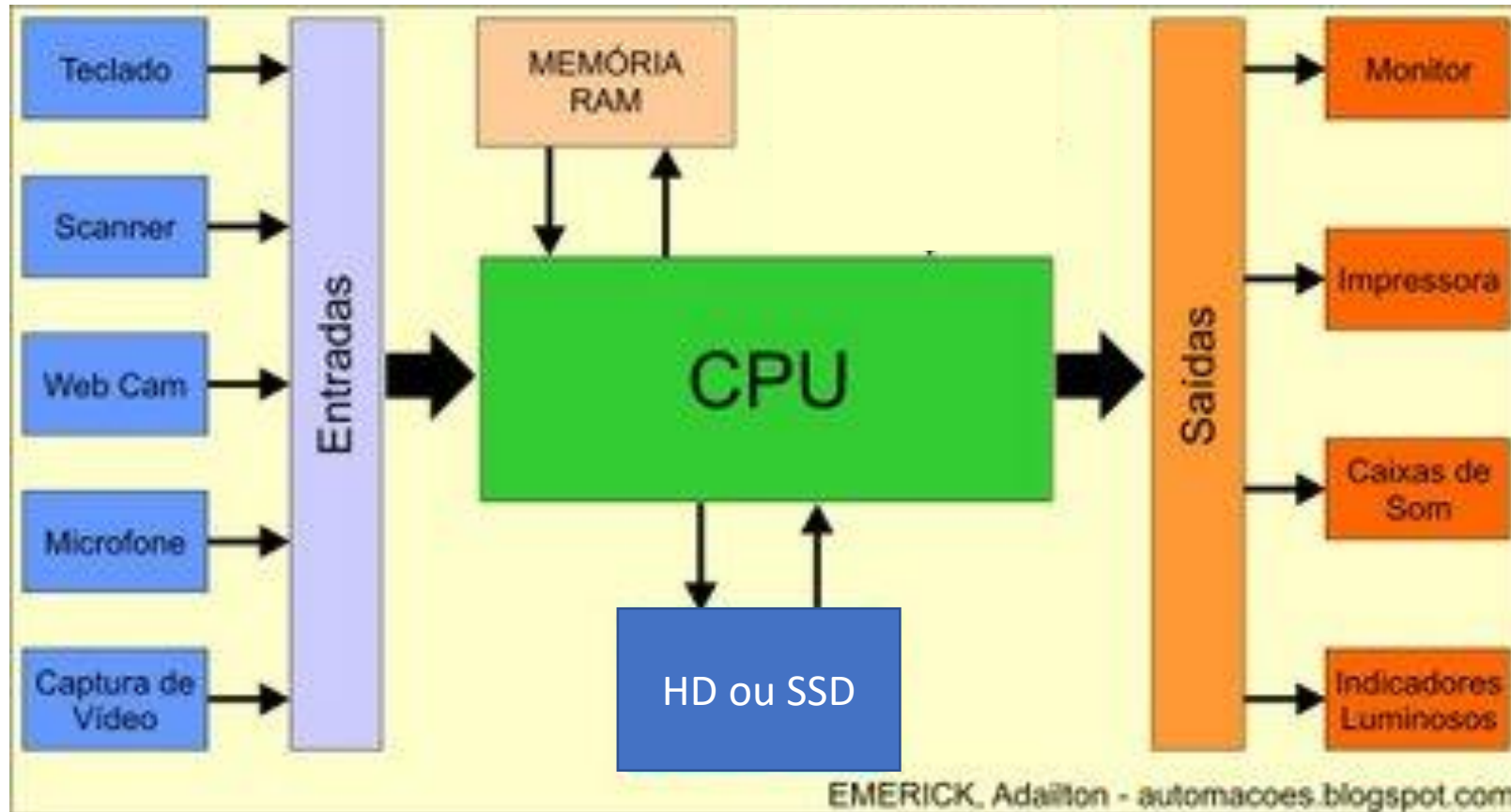
Hardware e Software Juntos



Como funciona o Hardware

- Entradas
- Processamento
- Armazenamento
- Saídas

Como funciona o Hardware

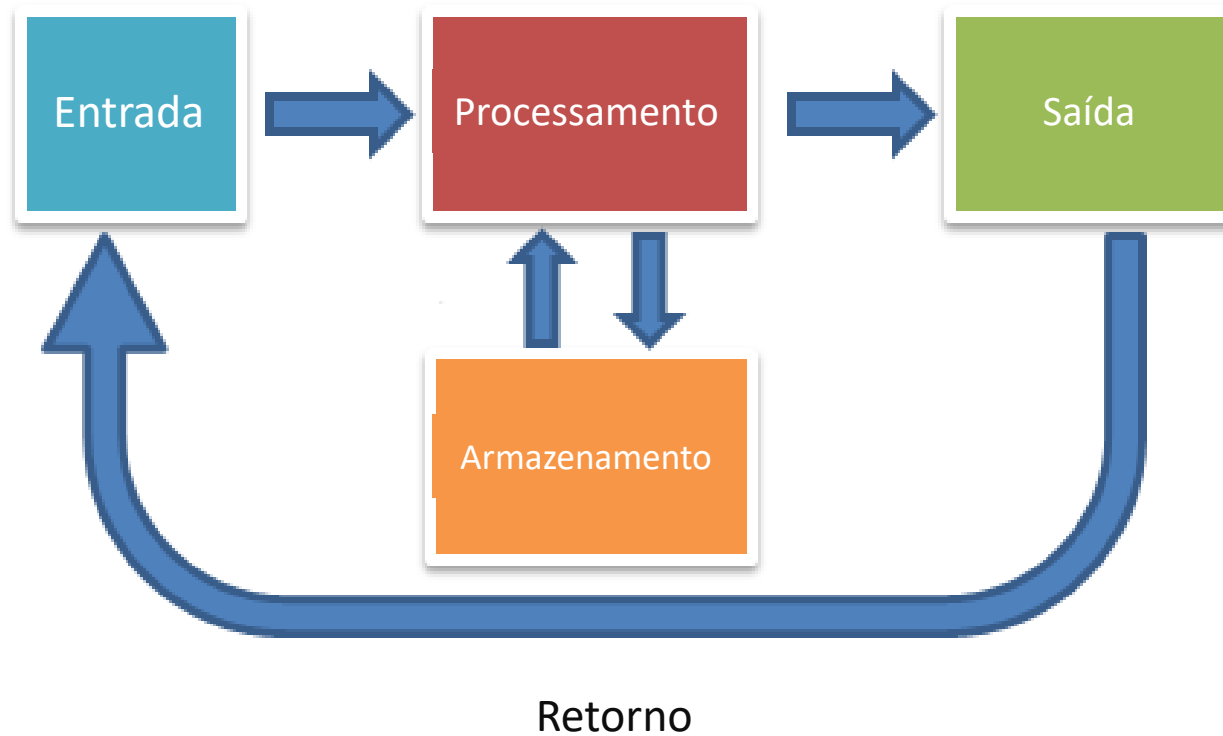


Como funciona o Software

Da mesma maneira que o Hardware:

- Entradas
- Processamento
- Armazenamento
- Saídas

Como funciona o Software



DÚVIDAS?

Como fazer um Programa?

Você vai precisar de:

- Uma Linguagem de Programação
- Uma IDE

O que é Linguagem de Programação

- É um conjunto de regras **léxicas** (ortografia) e **sintáticas** (gramática)
- para se escrever programas.

Linguagem de Programação - Parte Léxica

- Diz respeito à **ortografia**:

Em Português:

- cachorro
- caxorro

Em Linguagem de programação:

- main
- maim.

Linguagem de Programação - Parte Sintática

- Diz respeito às **sentenças**:

Em Português:

Correto: O cachorro está com fome

Errado: Está com cachorro fome o

Em Linguagem de programação:

Correto: $x = x + 2 + y;$

Errado: $= x + 2 + y;$

Principais Linguagens de Programação

Entre as 10 mais usadas no mundo:

- **JavaScript**
- Java
- **C#**
- Python
- PHP
- C++

IDE

IDE: Integrated Development Environment ou Ambiente de Desenvolvimento Integrado.

Ferramenta para a construção de programas.

Iremos trabalhar com o Microsoft Visual Studio Community 2019

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Tarefas

1 - Instalar o Microsoft Visual Studio Community 2019

2 - Baixar a estrutura de exercícios

Acessar: <https://renatogava.com.br/aulas/>

VISÃO GERAL DO VISUAL STUDIO

Abrir arquivos

Gerenciador de Soluções

Trocar tema, fontes e cores

AULA 2

Lógica de Programação e Algoritmos – parte 1

Recapitulando a aula anterior

- Como funciona um computador
- O que é programação e programador
- O mercado de trabalho
- O que é linguagem de programação
- O que é IDE
- Instalando o Visual Studio e baixando o material das aulas

Você vai aprender na aula de hoje

- O que é Lógica de Programação e Algoritmos
- Estruturas Sequenciais
- Operações Matemáticas
- Variáveis
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

- Exercícios – parte 2

Lógica de Programação (ou Algoritmo)

- Lógica de Programação é uma sequência de instruções para resolver um problema.

Problema: lavar roupa suja

Sequência de Instruções:

- 1) Colocar a roupa em um recipiente
- 2) Colocar um pouco de sabão e amaciante
- 3) Encher de água
- 4) Mexer tudo até dissolver todo o sabão
- 5) Deixar de molho por vinte minutos
- 6) Esfregar a roupa
- 7) Enxaguar
- 8) Torcer

Automação

- Consiste em utilizar computadores para executar o procedimento desejado de forma automática.

Automação

Sequência de Instruções:

- 1) Colocar a roupa em um recipiente
- 2) Colocar um pouco de sabão e amaciante
- 3) Encher de água
- 4) Mexer tudo até dissolver todo o sabão
- 5) Deixar de molho por vinte minutos
- 6) Esfregar a roupa
- 7) Enxaguar
- 8) Torcer

Para que serve um computador

- Programas são **algoritmos** executados pelo computador.
- O computador é uma máquina que **automatiza** a execução de
- **algoritmos**.

DÚVIDAS?

Estruturas Sequenciais

Os algoritmos são uma sequência de instruções executadas **de cima pra baixo**.

Errado:

```
soma = x + y
```

```
x = 10
```

```
y = 20
```

Correto:

```
x = 10
```

```
y = 20
```

```
soma = x + y
```


Estruturas Sequenciais em JavaScript

Sempre finalizar uma instrução com ponto-e-vírgula

```
var x = 10;
```

```
var y = 20;
```

```
var soma = x + y;
```

Operações Matemáticas

Somar: +

Subtrair: -

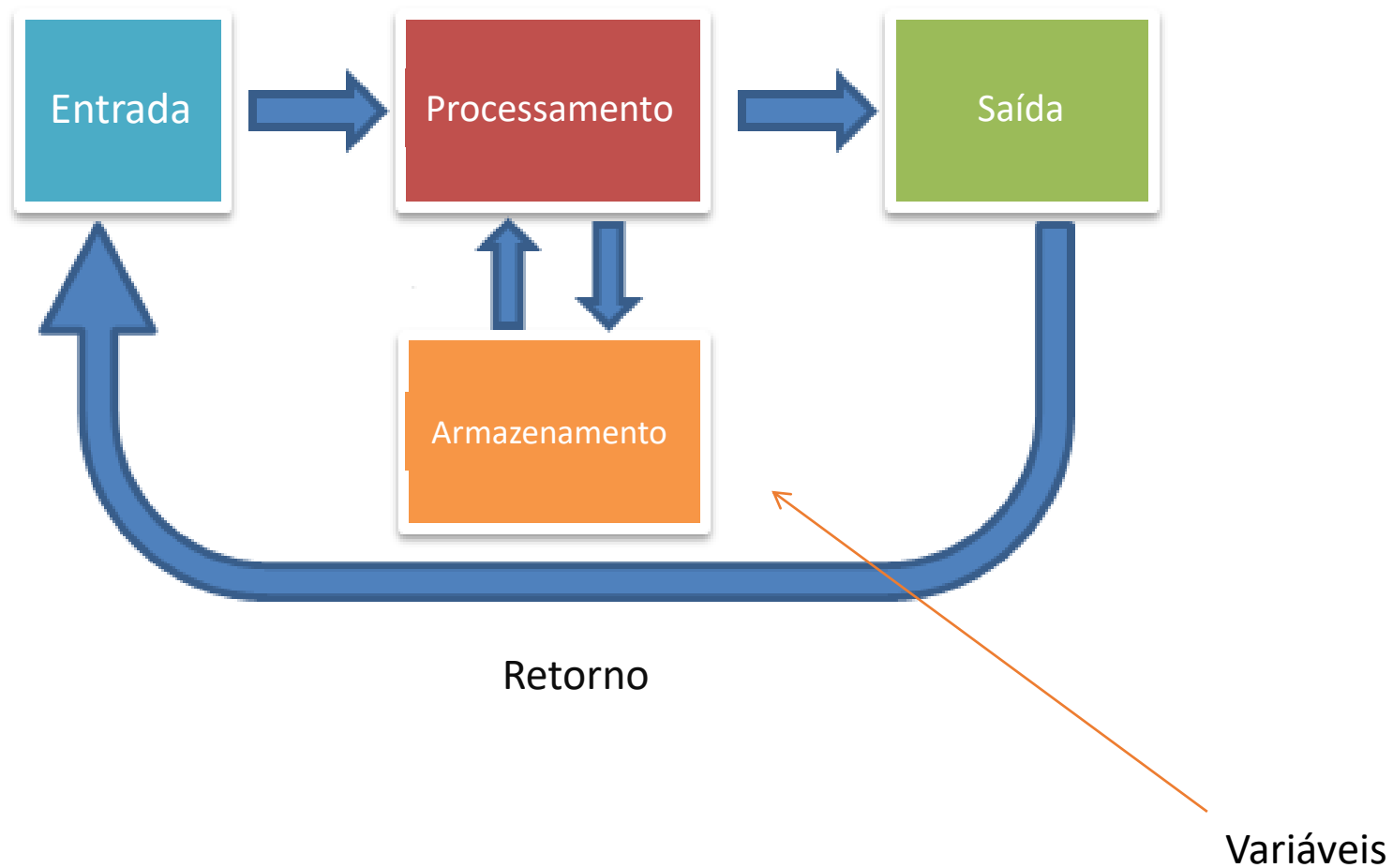
Multiplicar: *

Dividir: /

Variáveis

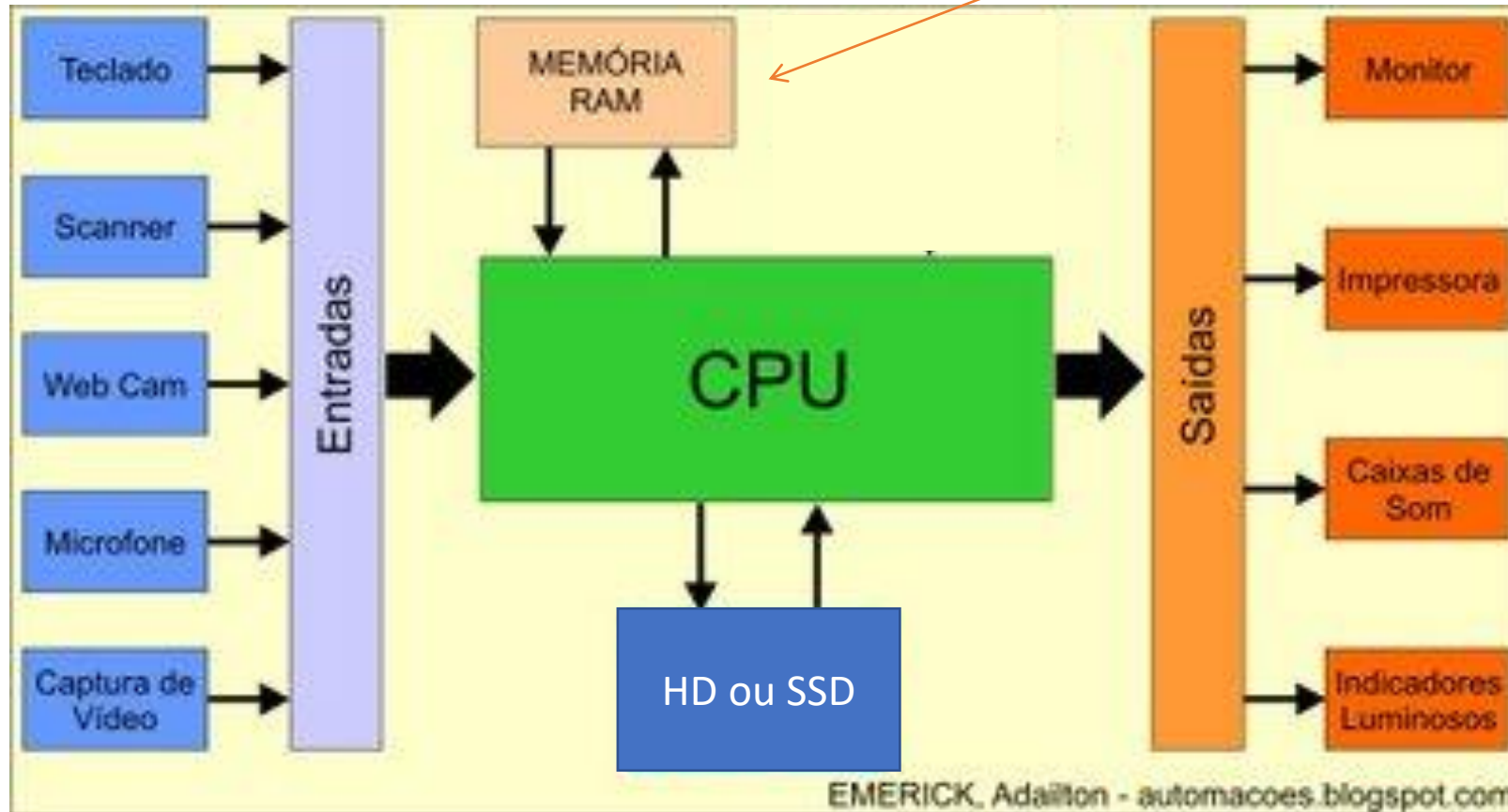
Uma variável é uma porção de memória (RAM) utilizada para armazenar dados durante a execução dos programas.

Variáveis no Software



Variáveis no Hardware

Variáveis



Variáveis

Nomes de variáveis:

- Não pode ter espaço em branco
- Não usar acentos ou ç
- Sugestão: use o padrão CamelCase

Exemplo de variável em JavaScript:

```
var salarioDoFuncionario = 2000;
```

Variáveis

Uma variável é composta por **nome**, **tipo** e **valor**.

Exemplo:

rua = Rua das Arandeiras

idade = 28

Tipos Básicos de Variáveis

Texto

Exemplo: Renato

Data

Exemplo: 12/02/2022

Número Inteiro

Exemplo: 23

Lógico

Verdadeiro ou Falso

Decimal

Exemplo: 23,09

Tipos Básicos em JavaScript

Texto

```
var nome = "Renato";
```

Data

```
var nascimento = new Date(1984,1,28);
```

Número Inteiro

```
var idade = 23;
```

Lógico

```
var asAulasComecaram = true;
```

Decimal

```
var valor = 23.09;
```

As 3 operações básicas da programação



1 - Entrada de Dados

- O usuário utiliza um dispositivo de entrada (teclado) para informar dados para o programa.
- Exemplo: usuário digita seu e-mail e senha na tela de login do iFood.

2 - Processamento de Dados

- O programa lê os dados informados pelo usuário e processa.
- Exemplo: o iFood processa seus dados, verificando se seu e-mail e senha estão corretos.



3 - Saída

- O programa devolve o retorno para o dispositivo de saída (monitor)
- Exemplo: depois que o iFood validou seus dados, ele concede acesso ao aplicativo ou exibe a mensagem “senha inválida”.

Dispositivo de SAÍDA



Exercícios

- 1 - Faça um programa onde o usuário clica em um botão e ele exibe uma mensagem “Olá Mundo”.
- 2 - Faça um programa onde o usuário informa seu nome e clica em um botão. O programa exibe uma mensagem “Olá Fulano”, onde Fulano é o nome informado.

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 3 - Faça um programa onde o usuário informa dois números, o programa realiza a soma e exibe o resultado.
- 4 - Faça um programa onde o usuário informa dois números, o programa realiza a média e exibe o resultado.

Exercícios para Casa

- 5 - Faça um programa para ler a largura, comprimento e valor do metro quadrado de um terreno retangular com duas casas decimais. Em seguida, o programa deve mostrar o valor da área do terreno e o preço total do terreno, ambos com duas casas decimais.
- 6 - Faça um programa para calcular o troco no processo de pagamento de um produto de uma mercearia. O programa deve ler o preço unitário do produto, a quantidade de unidades compradas deste produto, e o valor em dinheiro dado pelo cliente. Seu programa deve mostrar o valor do troco a ser devolvido ao cliente.
- 7 - Faça um programa para ler o nome de um funcionário, o valor que ele recebe por hora e a quantidade de horas trabalhadas por ele no mês. Ao final, mostrar o valor do pagamento do funcionário com uma mensagem explicativa.
- 8 - Faça um programa para ler a distância total percorrida por um carro (em km), e o total de combustível gasto por este carro ao percorrer essa distância (em litros). Seu programa deve mostrar o consumo médio do carro (km por litro).

AULA 3

Lógica de Programação e Algoritmos – parte 2

Recapitulando a aula anterior

- O que é Lógica de Programação e Algoritmos
- Estruturas Sequenciais
- Operações Matemáticas
- Variáveis

Correção dos exercícios

- Validar os exercícios da aula anterior

Você vai aprender na aula de hoje

- Expressões Comparativas
- Expressões Lógicas
- Estruturas Condicionais
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

- Exercícios – parte 2

Expressões Comparativas

Expressão retorna resultado **verdadeiro** ou **falso**

- 5 é maior que 10? resultado é falso
- 4 é maior que 2? resultado é verdadeiro

Operadores Comparativos

> maior

< menor

>= maior ou igual

<= menor ou igual

= igual

<> diferente

Operadores Comparativos

Suponha que a variável **x** seja igual a **5**:

x > 0 resultado verdadeiro

x = 3 resultado falso

x <> 3 resultado verdadeiro

x <= 5 resultado verdadeiro

x < 5 resultado falso

Operadores Comparativos em JavaScript

Suponha que a variável **x** seja igual a **5**:

x > 0 resultado verdadeiro

x == 3 resultado falso

x != 3 resultado falso

x <= 5 resultado verdadeiro

x < 5 resultado falso

Operadores Lógicos

E: todas as condições devem ser verdadeiras

OU: apenas uma das condições precisa ser verdadeira

NÃO: verdadeiro se a condição for falsa

Operadores Lógicos em JavaScript

&&: todas as condições devem ser verdadeiras

||: apenas uma das condições precisa ser verdadeira

!: verdadeiro se a condição for falsa

Operador “E”

Você pode obter sua habilitação de motorista se:

For aprovado no exame psicotécnico

E

For aprovado no exame técnico

E

For aprovado no exame prático

Todas as condições devem ser verdadeiras

Operador “E”

Suponha que a variável **x** seja igual a **5**:

$x > 0$ **e** $x < 10$ resultado verdadeiro

$x > 6$ **e** $x < 10$ resultado falso

Operador “OU”

Você pode estacionar na vaga especial se:

For idoso

OU

For deficiente

OU

For gestante

Pelo menos uma condição deve ser verdadeira

Operador “OU”

Suponha que a variável **x** seja igual a **5**:

$x > 0$ **ou** $x < 10$ resultado verdadeiro

$x > 6$ **ou** $x < 10$ resultado verdadeiro

$x > 6$ **ou** $x < 3$ resultado falso

Operador “NÃO”

Você tem direito a receber uma bolsa de estudos:

NÃO

Possuir renda maior que R\$ 3.000,00

O operador NÃO **inverte** a condição

Operador “NÃO”

Suponha que a variável **x** seja igual a **5**:

x = 5 resultado verdadeiro

NÃO x = 5 resultado falso

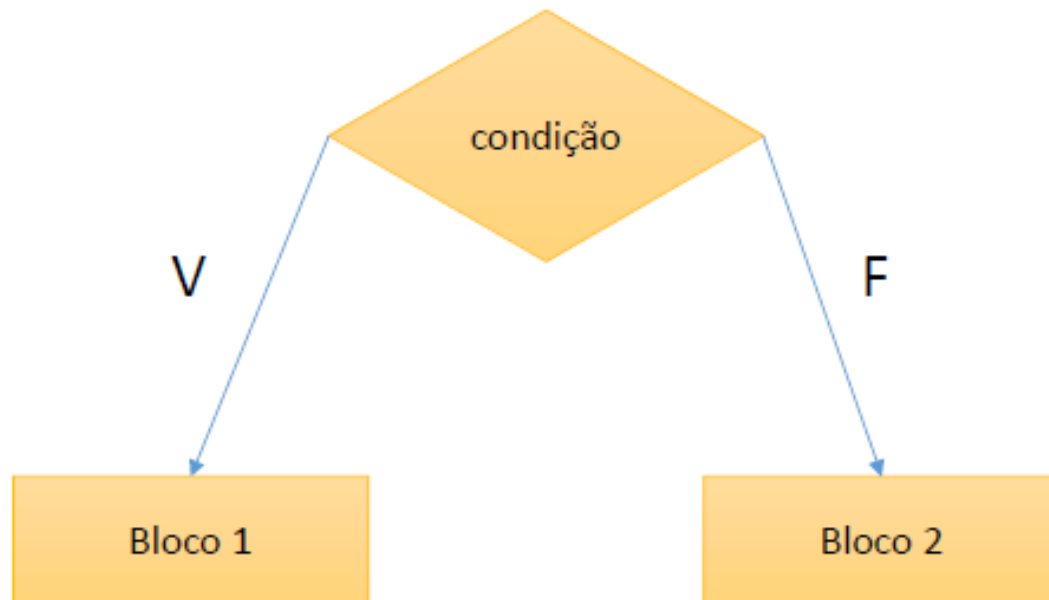
x = 6 resultado falso

NÃO x = 6 resultado verdadeiro

DÚVIDAS?

Estrutura Condicional

- É uma estrutura que permite definir que um certo bloco de comandos somente será executado dependendo de uma condição.



Estrutura Condicional

- SE, ENTÃO

SE *condição* **ENTÃO**

comando 1

comando 2

FIM

OBS: repare na indentação

Estrutura Condicional

- SE, ENTÃO e SENÃO

SE *condição* **ENTÃO**

comando 1

comando 2

FIM

SENÃO

comando 3

comando 4

FIM

Estrutura Condicional

SE condição 1 **ENTÃO**

comando 1

comando 2

FIM

SENÃO SE condição 2 **ENTÃO**

comando 3

comando 4

FIM

SENÃO

comando 5

comando 6

FIM

Operadores em JavaScript

- SE **if**
- SENÃO **else**
- ENTÃO {
- FIM }
- SENÃO SE **else if**
- E **&&**
- OU **||**
- NÃO **!**

Estrutura Condicional em JavaScript

```
if (condição1 && condição2) {  
    comando 1;  
    comando 2;  
}  
else if (condição3) {  
    comando 3;  
    comando 4;  
}  
else {  
    comando 5;  
    comando 6;  
}
```


Exercícios

- 1 – Faça um programa que exiba a mensagem “Boa noite” se a hora digitada for maior ou igual a 18.
- 2 – Faça um programa que exibe a mensagem “Bom dia” se a hora atual for menor que 12. A mensagem “Boa tarde” se a hora atual for maior ou igual a 12 e menor que 18. A mensagem “Boa noite” se a hora atual for maior ou igual a 18.

para obter a hora atual: **`new Date().getHours()`**

- 3 - Fazer um programa para ler três números inteiros. Em seguida, mostrar qual o menor dentre os três números lidos.

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 4 - Programa para ler duas notas que o aluno obteve, uma no primeiro semestre e outra no segundo. Em seguida, mostrar a nota final que o aluno obteve no ano juntamente com um texto explicativo. Caso a nota final do aluno seja inferior a 6, mostrar a mensagem "REPROVADO".
- 5 - Uma operadora de telefonia cobra R\$ 50,00 por um plano que dá direito a 100 minutos de ligação. Cada minuto que exceder a franquia de 100 minutos custa R\$ 2,00. Fazer um programa para ler a quantidade de minutos que uma pessoa consumiu e mostrar o valor a ser pago.

Exercícios

6 - Fazer um programa para ler a quantidade de glicose no sangue de uma pessoa e depois mostrar na tela a classificação desta glicose de acordo com a tabela de referência abaixo:

- **Normal:** Até 100 mg/dl
- **Elevado:** Maior que 100 até 140 mg/dl
- **Diabetes:** Maior de 140 mg/dl

AULA 4

Lógica de Programação e Algoritmos – parte 3

Recapitulando a aula anterior

- Expressões Comparativas
- Expressões Lógicas
- Estruturas Condicionais

Correção dos exercícios

- Validar os exercícios da aula anterior

Você vai aprender na aula de hoje

- Estruturas Repetitivas
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

- Exercícios – parte 2

Estrutura “enquanto”

- Repete um bloco de comandos **enquanto** uma condição for verdadeira.
- Quando usar? Quando não se sabe previamente quantas repetições precisarão ser feitas.

Exemplo de “Enquanto”

- Repete um bloco de comandos **enquanto** uma condição for verdadeira.
- Quando usar? Quando não se sabe previamente quantas repetições precisarão ser feitas.

Exemplo de “Enquanto”

- Repete um bloco de comandos **enquanto** uma condição for verdadeira.
- Quando usar? Quando não se sabe previamente quantas repetições precisarão ser feitas.

Estrutura Enquanto

- ENQUANTO

ENQUANTO condição **FAÇA**

comando 1

comando 2

FIM

OBS: repare na indentação

Estrutura Enquanto no JavaScript

- ENQUANTO

```
while (condição) {  
    comando 1;  
    comando 2;  
}
```

OBS: repare na indentação

Exercícios

1 - Fazer um programa para ler a quantidade vezes que ele irá dizer “Olá mundo”. Exibir a mensagem “Olá mundo” de acordo com a quantidade de vezes digitada, utilizando “Enquanto”.

Estrutura “para”

- Repete um bloco de comandos **x** uma condição for verdadeira.
- Quando usar? Quando se sabe exatamente quantas repetições precisarão ser feitas.

Estrutura Para

- PARA

PARA *variável* de *valor_inicial* até *valor_final* incrementa *variável* **FAÇA**

comando 1

comando 2

FIM

A variável inicia com o *valor_inicial*

Se o valor da variável não exceder o *valor_final*, executa o bloco de comandos.

Assim que executar o bloco, incrementa 1 na variável

Estrutura “Para” em JavaScript

```
for (var i = 1; i <= 15; i++) {  
    comando 1;  
    comando 2;  
}
```

Estrutura “Para” regressiva em JavaScript

```
for (var i = 15; i >= 1; i--) {  
    comando 1;  
    comando 2;  
}
```

Exercícios

- 2 - Fazer um programa para ler a quantidade vezes que ele irá dizer “Olá mundo”. Exibir a mensagem “Olá mundo” de acordo com a quantidade de vezes digitada, utilizando “Para”.
- 3 - Fazer um programa para ler a quantidade vezes que ele irá dizer “Olá mundo”. Exibir a mensagem “Olá mundo” de acordo com a quantidade de vezes digitada, utilizando “Para” de forma regressiva.
- 4 - Fazer um programa que lê um número e exibe a tabuada desse número.

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

5 - Faça um programa que lê dois números. A seguir calcule e exiba a soma entre os números ímpares entre eles.

6 - Leia um valor inteiro X. Em seguida mostre os ímpares de 1 até X, inclusive o X, se for o caso.

Exercícios para Casa

7 - Leia um valor inteiro X. Em seguida exiba somente os números primos de 1 até X, inclusive o X, se for o caso.

8 - Leia um valor inteiro X. Em seguida some os números pares de 1 até X, inclusive o X, se for o caso. Exiba o valor da soma na tela.

AULA ADICIONAL

Arrays e Vetores

Você vai aprender na aula de hoje

- Entendendo Coleções de Dados
- O que é um Array?
- O que é um Vetor
- Operações com Arrays
- Exercícios

Entendendo Coleções de Dados

Em programação, frequentemente precisamos trabalhar com coleções de dados, como uma lista de números, nomes de pessoas ou qualquer tipo de informação.

Arrays e vetores são estruturas de dados que nos permitem armazenar e manipular essas coleções de forma eficiente.

O que é um Array?

Um array é uma estrutura de dados que armazena um conjunto de elementos, como números, texto ou qualquer outro tipo de dado, em uma única variável.

Cada elemento em um array é acessado através de um índice, que é um número que representa a posição do elemento no array.

Os arrays são zero indexados, o que significa que o primeiro elemento tem índice 0, o segundo tem índice 1 e assim por diante.

Vetores

- Um vetor é uma coleção de dados **indexada** e **homogênea**.

0	Roberto
1	Adriana
2	José
3	Lucas
4	Felipe
5	Eduardo

O que é um Vetor?

O termo "vetor" é frequentemente usado como sinônimo de "array", especialmente em contextos de programação.

Portanto, quando as pessoas falam de "vetores" na programação, geralmente estão se referindo a arrays.

Exemplo de Vetor

- **Indexado:** os elementos são acessados por meio de índices
- **Homogêneo:** todos os elementos são do mesmo tipo

0	Roberto
1	Adriana
2	José
3	Lucas
4	Felipe
5	Eduardo

Exemplo de Vetor em JavaScript

```
var clientes = ["Roberto", "Adriana", "José", "Lucas", "Felipe",  
"Eduardo"];
```

```
alert(clientes[1]); //Adriana
```

0	Roberto
1	Adriana
2	José
3	Lucas
4	Felipe
5	Eduardo


Operações com Arrays/Vetores

Você pode realizar várias operações com arrays/vetores, como adicionar elementos, remover elementos, encontrar o tamanho do array e percorrê-lo para realizar ações em cada elemento.

Aqui estão alguns exemplos em JavaScript:

1. Adicionar um elemento ao final do array:


javascript

 Copy code

```
numeros.push(6); // Adiciona o número 6 ao final do array
```

1. Remover um elemento do final do array:


javascript

 Copy code

```
numeros.pop(); // Remove o último elemento do array
```

1. Encontrar o tamanho do array:

javascript

 Copy code

```
let tamanho = numeros.length; // Isso retornará 5, o tamanho do array
```

Exercícios

- 1 - Faça um programa que leia o índice do array de clientes abaixo e exiba o nome de um respectivo cliente.
- 2 - Faça um programa que exiba o nome de todos os clientes do array abaixo.

0	Roberto
1	Adriana
2	José
3	Lucas
4	Felipe
5	Eduardo

Exercícios

3 - Faça um programa que leia a quantidade de números que você irá informar. Em seguida informe cada número e armazene-os em um array. Por fim, exiba o maior, o menor e a média dos números fornecidos.

4 - Faça um programa que leia a quantidade de números que você irá informar. Em seguida informe cada número e armazene-os em um array. Por fim, exiba somente os números negativos digitados.

5 - Faça um programa que leia a quantidade de pessoas que você irá informar. Em seguida informe o nome e a idade de cada pessoa, e armazene o nome em um vetor e a idade em outro vetor. Por fim, exiba o nome da pessoa mais velha.

AULA 5

Front-end Web – parte 1

Recapitulando a aula anterior

- Estruturas Repetitivas
- Exercícios

Você vai aprender na aula de hoje

- Como funciona a internet e a Web. Referência: <https://developer.mozilla.org/>
- O que é Front-end e Back-end
- O que é HTML, CSS e JavaScript
- HTML

20:45 Intervalo de 15 minutos

- Exercícios

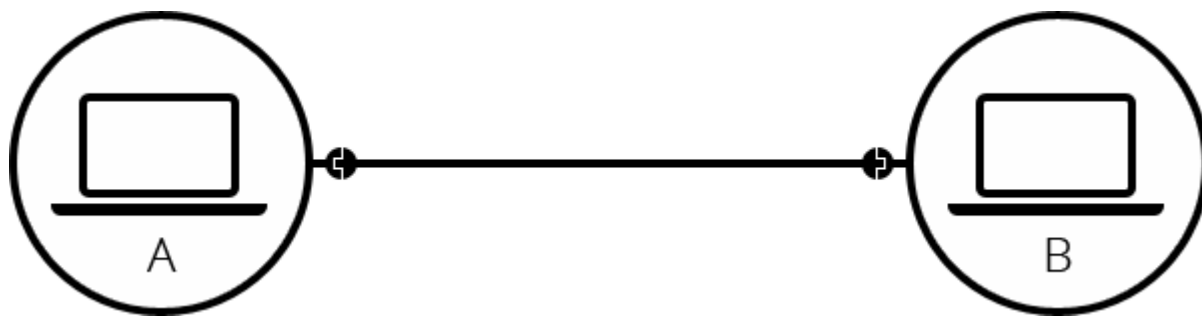
Como funciona a internet e a Web?

- A **Internet** é a espinha dorsal da **Web**, a infraestrutura técnica que faz a Web possível.
- Mas basicamente, a Internet é uma gigantesca **rede de computadores**.

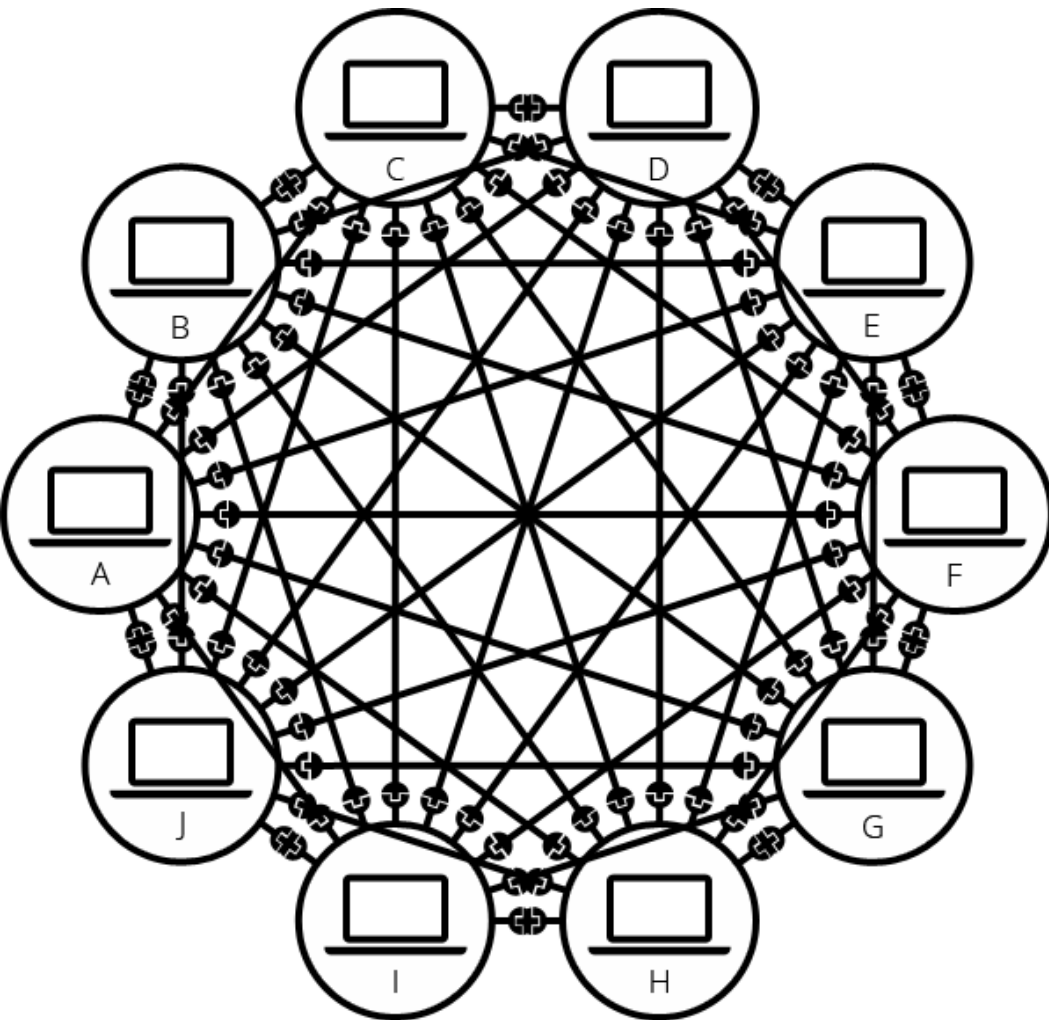
Como funciona a internet?

Uma rede simples

- Quando dois computadores precisam se comunicar, você precisa conectá-los, por cabo de redes, bluetooth ou wi-fi.

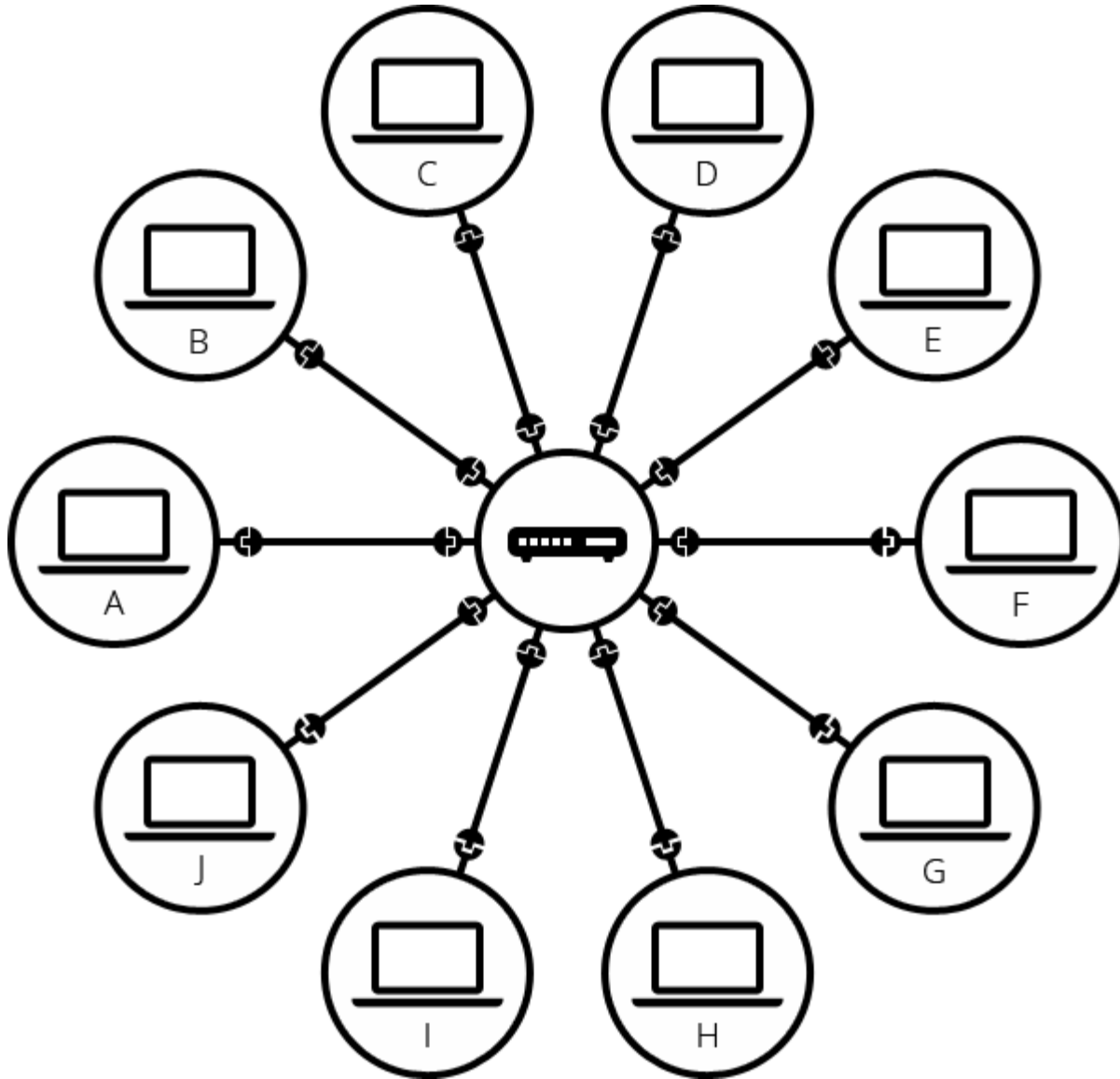


A internet



Uma rede não é limitada a dois computadores. Você pode conectar quantos computadores desejar. Mas isto se torna complicado.

Se você está tentando conectar, digamos, dez computadores, você irá precisar de 45 cabos, com 9 conexões por computador.

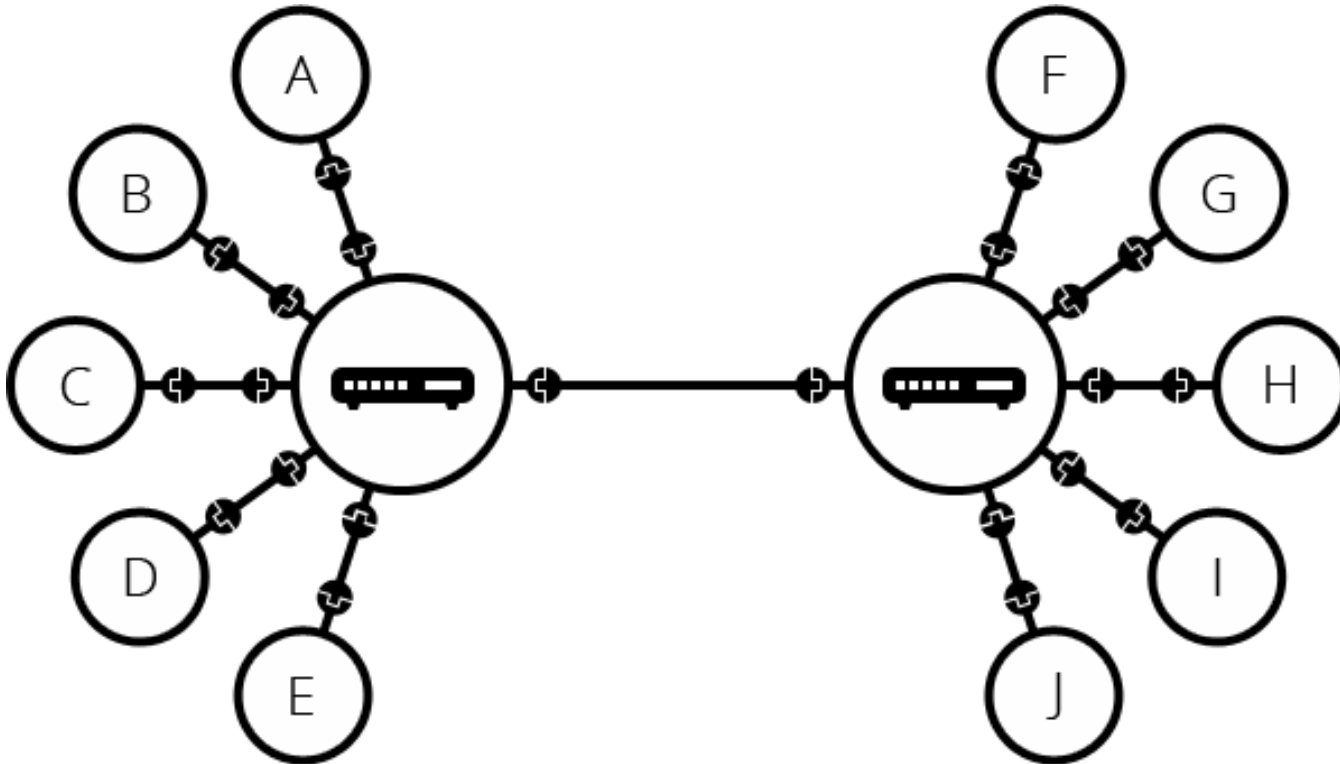


Para resolver este problema, cada computador na rede está conectado a um pequeno computador especial chamado de **roteador**.

O roteador tem um único trabalho: garantir que uma mensagem enviada por um determinado computador chegue ao computador destinatário.

Mas como conectar centenas, milhares, bilhões de computadores?

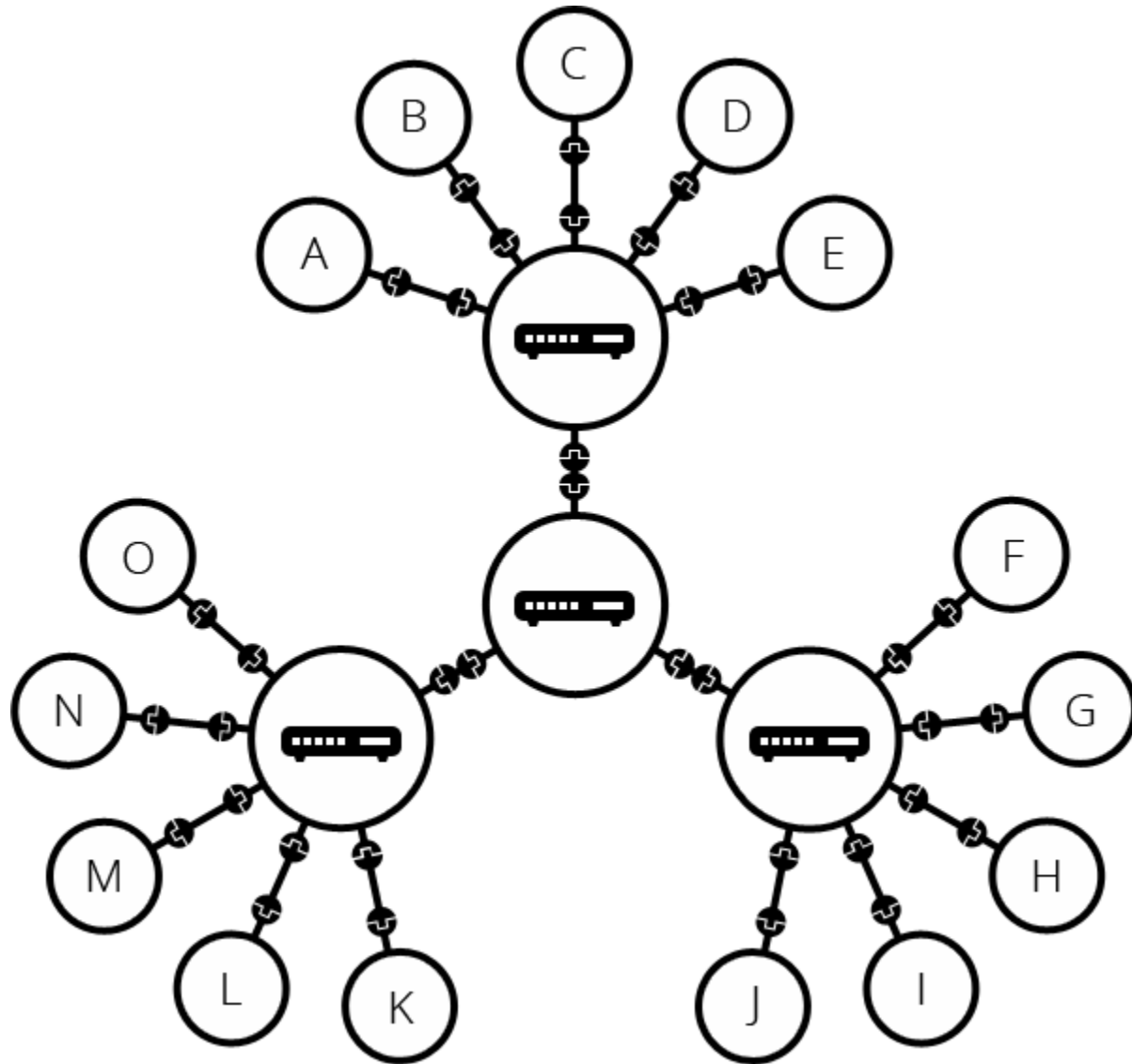
Conectando **um roteador em outro**.



A internet

Conectando roteadores a roteadores nós podemos escalar nossa rede infinitamente.

Esta rede é muito parecida com o que chamamos de **Internet**.

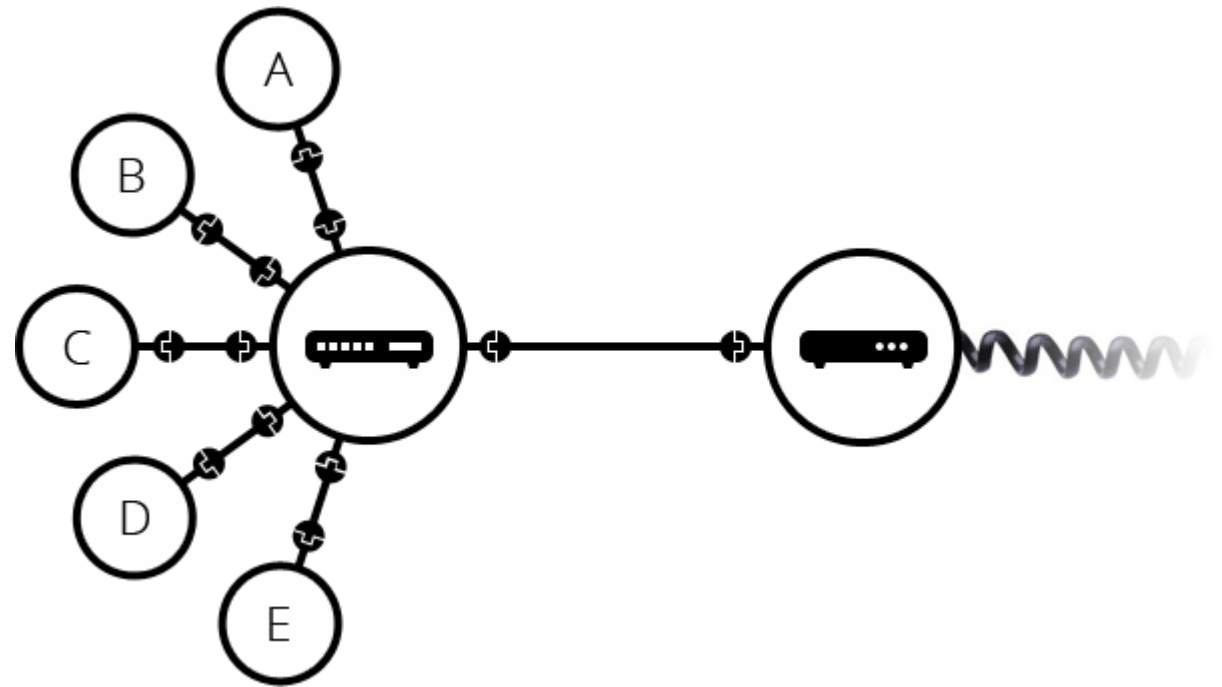


A internet

É inviável ligarmos cabos entre nossas casas e o resto do mundo, então como nos podemos lidar com isso?

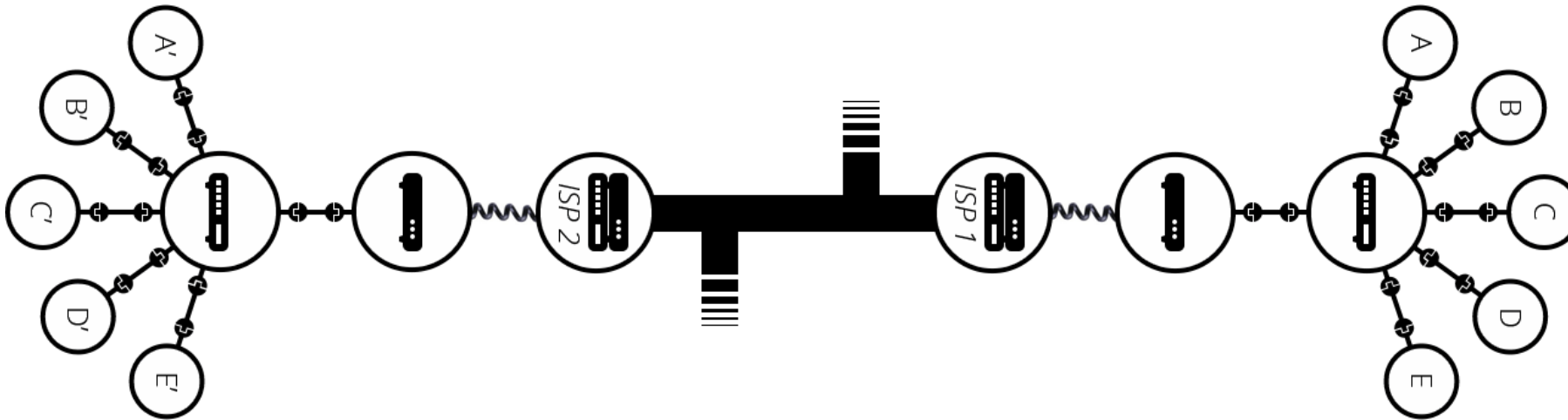
Já existem cabos ligados a sua casa, como por exemplo, cabos de eletricidade e telefone.

Para conectar nossa rede a rede telefônica, precisamos de um equipamento especial chamado **modem**.



A internet

O próximo passo é enviar mensagens da nossa rede para a rede que nós desejamos alcançar. Para fazer isto, vamos precisar conectar nossa rede a um **Provedor de Serviço de Internet** (ISP, em inglês)



A internet

Se você quer enviar uma mensagem para um computador, você precisa especificar qual é este computador.

Qualquer computador conectado à uma rede possui um único endereço de identificação, chamado de **Endereço IP** (onde IP, do inglês Internet Protocol, significa Protocolo de Internet).

Este é um endereço composto por uma série de 4 números separados por pontos, por exemplo: 192.168.2.10.

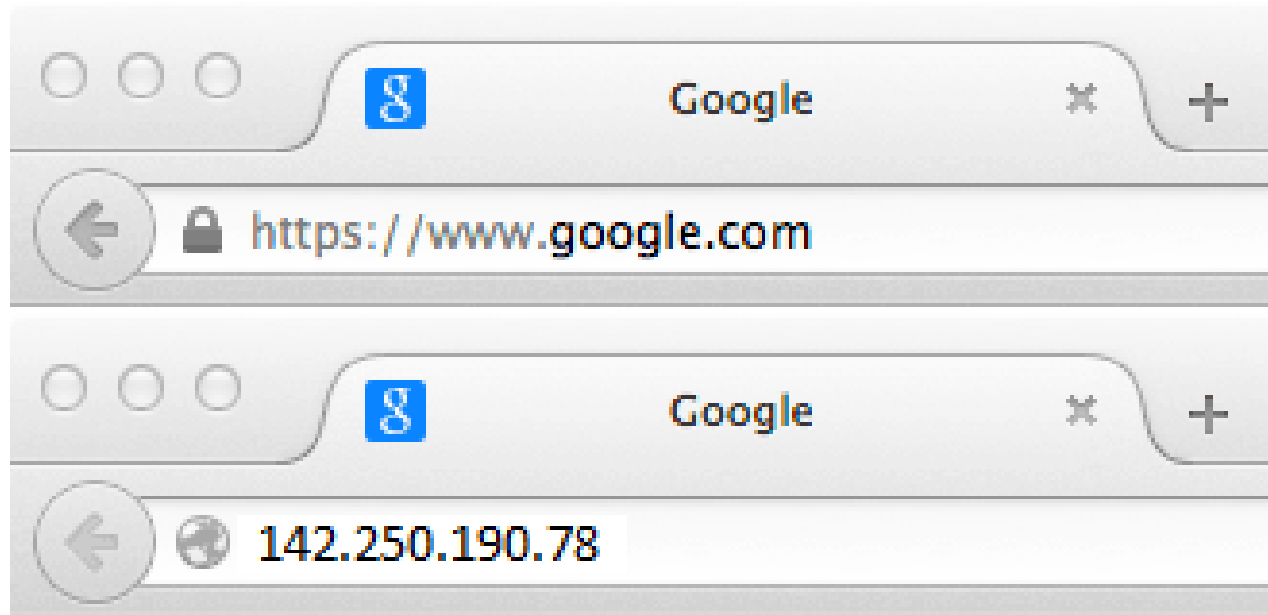
A internet

Isto é perfeito para computadores, mas nós seres humanos temos dificuldades para lembrar estes endereços.

Para tornar as coisas mais fáceis, nós podemos dar apelidos aos endereços IP que nós humanos podemos compreender, chamados nome de **domínio**. Quando navegamos na Web com nossos navegadores, normalmente utilizamos os nomes de **domínios** para chegar a um website.

A Internet

Por exemplo, google.com é um nome de domínio usado para apelidar o endereço 142.250.190.78.



A internet e a Web

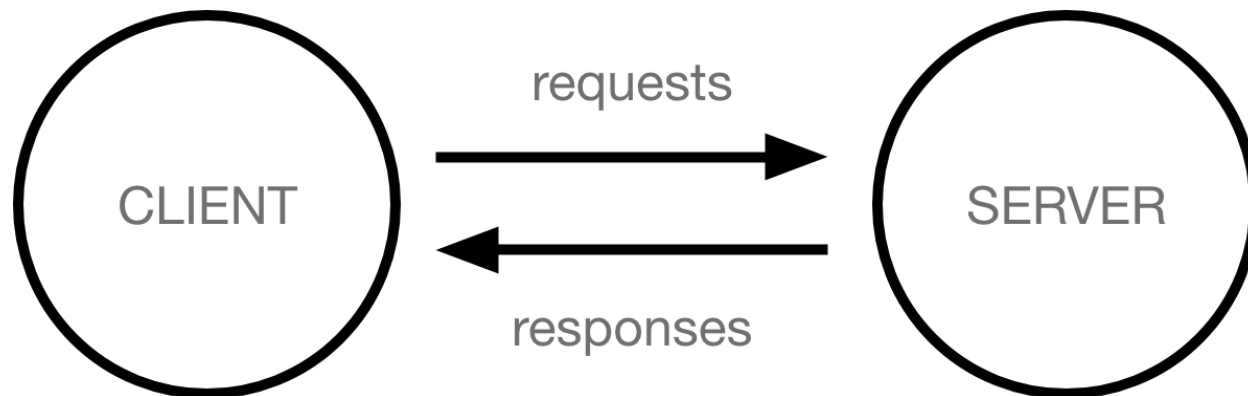
Internet e a Web **não são** a mesma coisa.

A **Internet** é uma **infraestrutura** técnica que permite conectar bilhões de computadores. Entre estes computadores, alguns computadores (chamados de servidores Web) podem enviar mensagens inteligíveis para navegadores Web.

A **Web** é um serviço construído em cima dessa infraestrutura. Web é sinônimo de **World Wide Web** ou **WWW**.

Clientes e Servidores

Computadores conectados à Web são chamados **clientes** e **servidores**. Um diagrama simplificado de como eles interagem pode ter essa aparência:



Clientes e Servidores

Clientes são os dispositivos conectados à internet dos usuários da web (por exemplo, seu computador conectado ao seu Wi-Fi ou seu telefone conectado à sua rede móvel) e programas de acesso à Web disponíveis nesses dispositivos (geralmente um navegador como Firefox ou Chrome).

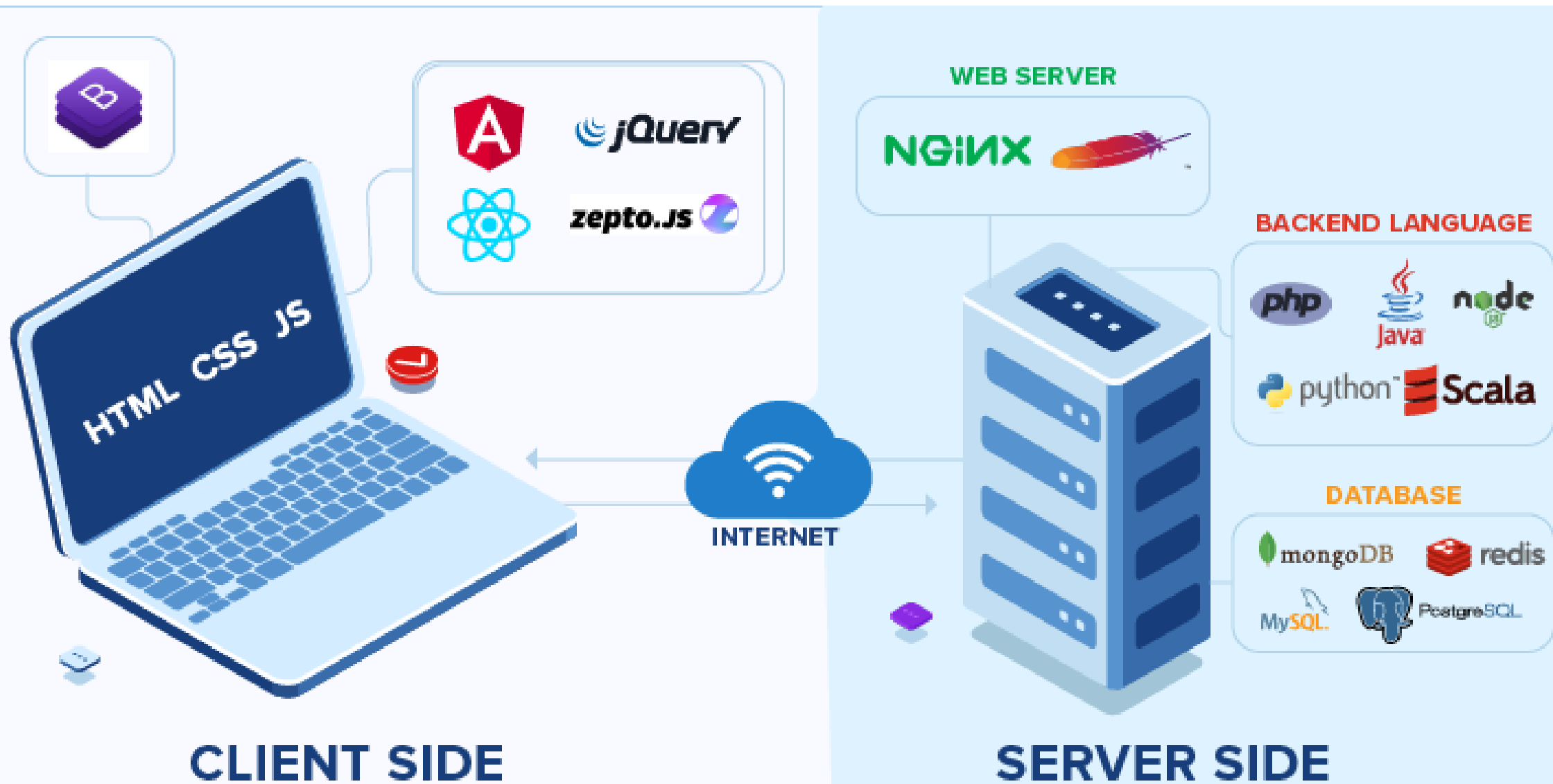
Servidores são computadores que armazenam páginas, sites ou aplicativos.

O que é Front-end Web

- Podemos classificar como a parte visual de um site, que roda do lado do **cliente**, aquilo que conseguimos interagir.
- Quem trabalha com Front-end é responsável por desenvolver por meio de código uma interface gráfica, normalmente com as tecnologias base da Web (HTML, CSS e JavaScript).

O que é Back-end

- Back-end, como o próprio nome sugere, vem da ideia do que tem por trás de uma aplicação. É o que roda no **servidor**.
- Ele trabalha fazendo a ponte entre os dados que vem do navegador e o banco de dados.



Front-end Web

Os sites são feitos principalmente de **HTML**, **CSS** e **JavaScript**.

HTML: provê a estrutura básica dos sites, que será melhorada e modificada por outras tecnologias como o CSS e o JavaScript.

CSS: é usado para controlar a apresentação, formatação e layout.

JavaScript: é usado para controlar o comportamento dos elementos.

O que é HTML

- **Hyper Text Modeling Language:** Linguagem de Modelagem de Hipertexto.
- A base de todas as páginas da web.
- É a linguagem que os navegadores entendem.
- Usar como referência: <https://www.w3schools.com/>

HTML

Coleção de tags indentadas:

```
<tag>coisas dentro</tag>
```

```
<tag1>
```

```
  <tag2>
```

```
    teste
```

```
  </tag2>
```

```
</tag1>
```

Tags Fundamentais

`<!DOCTYPE html>` diz ao navegador que o arquivo é html

`<html>` tag principal de html

`<head>` tag de cabeçalho

`<title>` tag de título

`<body>` tag de corpo

Tags Meta

```
<meta charset="utf-8">
```

- Especifica o codificação dos caracteres do documento HTML.
- UTF-8 é a codificação de caracteres mais comum.

Tags Meta

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0">
```

- **width=device-width**: diz que a largura da página deverá seguir a largura da tela do dispositivo (que irá variar de acordo com o dispositivo).
- **initial-scale=1.0**: diz qual é o nível inicial de zoom que a página irá ter quando é carregada pela primeira vez pelo navegador.

Tags Link

```
<link rel="stylesheet" href="style-login.css" />
```

```
<link rel="icon" type="image/x-icon" href="./favicon.ico">
```

- Define a relação entre o document HTML e recursos externos, como estilos e ícones.

Tags Complementares

<h1>, <h2>, <h3> tags de título

<p> parágrafo

 pula linha

<a> link

<label> legenda para um determinado elemento

 negrito

 imagem

 lista de itens

<hr> linha

<div> contêiner genérico

, lista de itens

Tags de Formulários

`<form>` formulário

`<input type="text">` caixa de texto

`<input type="email">` caixa de texto para email

`<input type="number">` caixa de número

`<input type="password">` caixa de senha

`<input type="checkbox">` caixa de marcação

`<input type="radio">` caixa de opção

`<button>` botão

`<select></select>` caixa de seleção

Exercícios

- 1 - Monte uma página que tenha a seguinte estrutura: Título Principal, parágrafo, subtítulo e parágrafo
- 2 - Monte uma página que tenha a seguinte estrutura: Título Principal, caixa de texto Nome, caixa de texto Sobrenome e Botão.
- 3 - Monte uma página que tenha a seguinte estrutura: Título Principal, parágrafo, formulário, caixa de seleção, caixa de texto e caixa de opção.
- 4 - Monte uma página que tenha a seguinte estrutura: Título Principal, parágrafo, imagem e lista de itens

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 5 - Login: desenhe uma tela de login que possua título, imagem, caixas de texto que leiam e-mail e senha do usuário, um botão para fazer login, um link para cadastrar-se e um link para esqueceu senha.
- 6 - Cadastro: desenhe uma tela de cadastro de clientes que possua título, imagem, caixas de texto que leiam os seguintes dados: nome, sobrenome, telefone, e-mail, gênero e senha, com um botão para fazer cadastro e um link para a pessoa fazer login, caso já tenha conta.
- 7 - Esqueceu Senha: desenhe uma tela de cadastro de clientes que possua título, imagem, uma caixa de texto que leia o e-mail, um botão e um link para fazer login.

AULA 6

Front-end Web – parte 2

Recapitulando a aula anterior

- Como funciona a internet e a Web
- O que é Front-end e Back-end
- O que é HTML, CSS e JavaScript
- HTML

Você vai aprender na aula de hoje

- CSS
- Exercícios

20:45 Intervalo de 15 minutos

- Exercícios

CSS

- Cascading Style Sheets: Folhas de Estilo em Cascata
- É um mecanismo para editar a aparência a um documento web e seus elementos HTML.

Principais atributos do CSS

color: define a cor do texto

background: define a cor do fundo

font-size e font-family: define o tamanho e a fonte do texto

border: define a borda de um elemento

margin: define o espaçamento entre um elemento e outro da borda pra fora

width: largura

height: altura

Classe

O atributo **class** atribui uma classe a um elemento HTML

Exemplo: `<p class="paragrafo">Parágrafo</p>`

```
<style>
.paragrafo {
  color: red
}
</style>
```

Todos os elementos que possuem a classe “paragrafo” serão afetados.

Id do Elemento

No CSS você pode identificar um elemento HTML pelo seu **id**, utilizando #:

Exemplo: `<p id="paragrafo-1">Parágrafo</p>`

```
<style>
#paragrafo-1 {
  color: red
}
</style>
```

Apenas o elemento com id paragrafo-1 será afetado

Tipo do Elemento

No CSS você pode identificar um elemento HTML pelo seu tipo.

Exemplo: `<p>Parágrafo</p>`

```
<style>
p {
  color: red
}
</style>
```

Todos os elementos `p` serão afetados.

Como utilizar o CSS

- Você pode utilizar utilizar o CSS de duas formas:
 - 1 - Utilizando a tag `<style></style>` dentro do documento html
 - 2 - Criando um arquivo .css e conectando-o à página com uma tag `<link>`

Exercícios

- 1 - Copie o arquivo 00-todos-elementos.html e aplique estilos a ele, utilizando a tag `<style>`
- 2 - Aplique estilos utilizando a classe, o id e o tipo de elemento HTML.
- 3 - Copie novamente o arquivo 00-todos-elementos.html e aplique a folha de estilos padrão do sistema `style.css`

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

4 - Aplique estilo na página de Login

5 - Aplique estilo na página de Cadastro

PARA CASA

6 - Aplique estilo na página de Esqueceu Senha

AULA 7

Front-end Web – parte 3

Recapitulando a aula anterior

- CSS

Você vai aprender na aula de hoje

- Eventos JavaScript
- Funções JavaScript
- Exercícios

20:45 Intervalo de 15 minutos

- Exercícios

JavaScript

JavaScript (frequentemente abreviado como JS) é atualmente a principal linguagem para programação em navegadores web.

Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web.

Enquanto o **HTML** é usado para armazenar o conteúdo e a formatação de uma página web e o **CSS** define a formatação e a aparência, o **JavaScript** é usado para adicionar interatividade e comportamento aos elementos da página.

JavaScript

Pode ser usado de duas maneiras:

- 1 – tag `<script></script>` no documento HTML
- 2 – em um arquivo `.js` e posteriormente linkado

Eventos

Eventos são ações ou ocorrências que acontecem no sistema que estamos desenvolvendo, no qual este te alerta sobre essas ações para que você possa responder de alguma forma, se desejado.

Por exemplo, se o usuário clica em um botão numa pagina web, você pode querer responder a esta ação mostrando na tela uma caixa de informações.

Eventos

Principais eventos:

onclick: usuário clicou no elemento

onfocus: o foco do cursor entrou no elemento

onblur: o foco do cursor saiu do elemento

onchange: o usuário selecionou uma opção do elemento

onsubmit: o usuário submeteu um formulário

Funções

Funções são blocos de construção fundamentais em JavaScript. Uma função é um procedimento de JavaScript - um conjunto de instruções que executa uma tarefa ou calcula um valor.

```
function raizQuadrada(numero) {  
    return numero * numero;  
}
```

Chamando uma Função

A definição de uma função não a executa. Definir a função é simplesmente nomear a função e especificar o que fazer quando a função é chamada.

Chamar a função executa realmente as ações especificadas com os parâmetros indicados. Por exemplo, se você definir a função square, você pode chamá-la do seguinte modo:

```
var valor = raizQuadrada(5);  
alert(valor);
```

Exercícios

- 1 - Faça um programa que o clique do botão chame uma função que retorna a mensagem: “Olá mundo, eu sou uma função”.
- 2 - Faça um programa que leia um número, chame função para calcular a raiz quadrada do número e retorne o valor calculado.

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 3 - Defina o comportamento dos elementos da página de Login: ao carregar a tela, focar na primeira caixa de texto; ao clicar no botão, verificar se todos os campos estão preenchidos e exibir uma mensagem de erro caso não estejam;
- 4 - Defina o comportamento dos elementos da página de Cadastro: ao carregar a tela, focar na primeira caixa de texto; ao clicar no botão, verificar se todos os campos estão preenchidos e exibir uma mensagem de erro caso não estejam;
- 5 - Defina o comportamento dos elementos da página de Esqueceu Senha: ao carregar a tela, focar na primeira caixa de texto; ao clicar no botão, verificar se todos os campos estão preenchidos e exibir uma mensagem de erro caso não estejam;

AULA 8

API – parte 1

Recapitulando a aula anterior

- Eventos JavaScript
- Funções JavaScript

Você vai aprender na aula de hoje

- API
- REST
- EndPoints

20:45 Intervalo de 15 minutos

- Exercícios

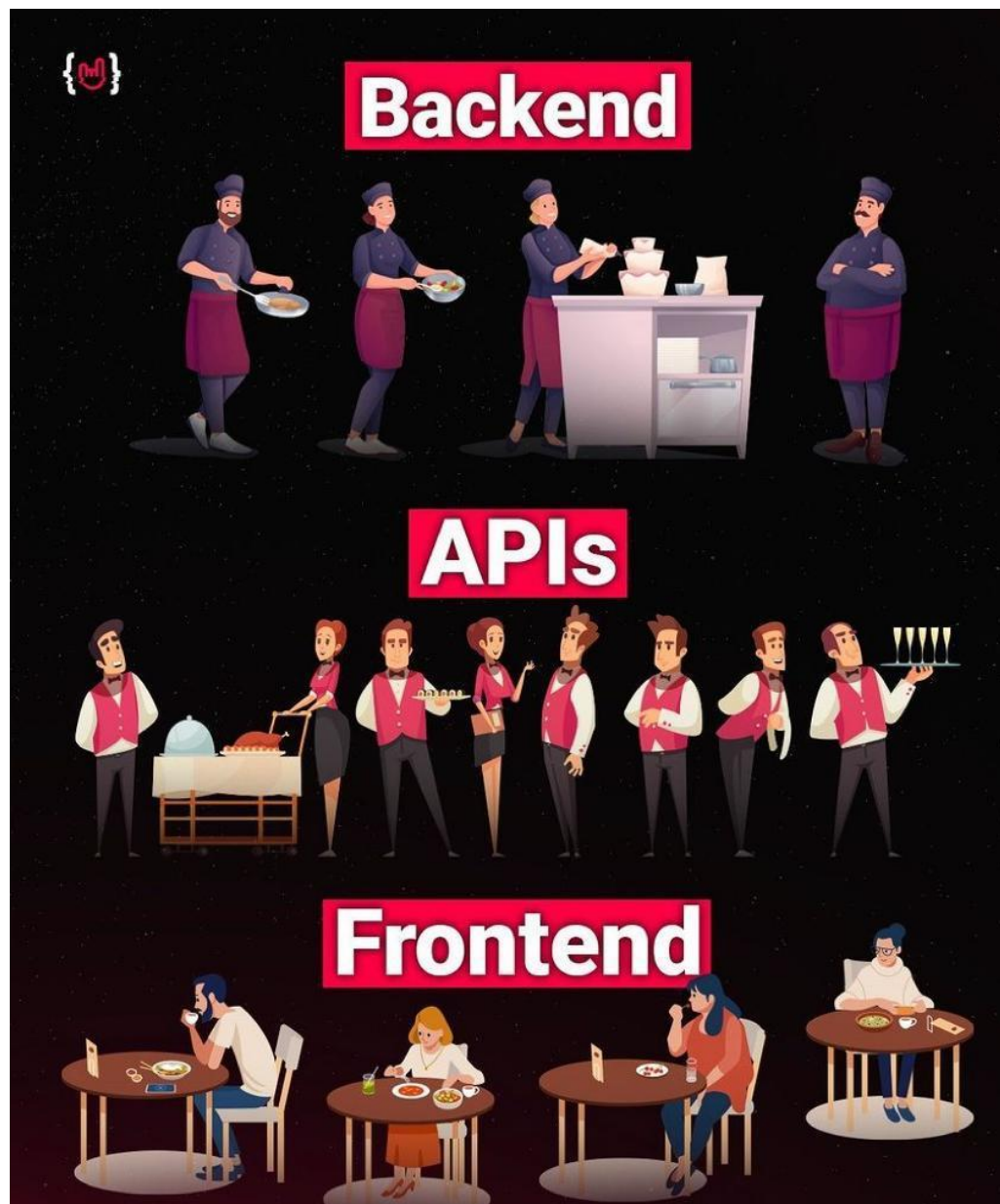
O que é API?

- **Application Programming Interface**, ou, em português, Interface de Programação de Aplicativos.
- APIs servem para conectar sistemas, softwares e aplicativos.
- Garantir que dois softwares diferentes possam se comunicar entre si.
- O intuito de uma API é trocar dados entre sistemas diferentes.

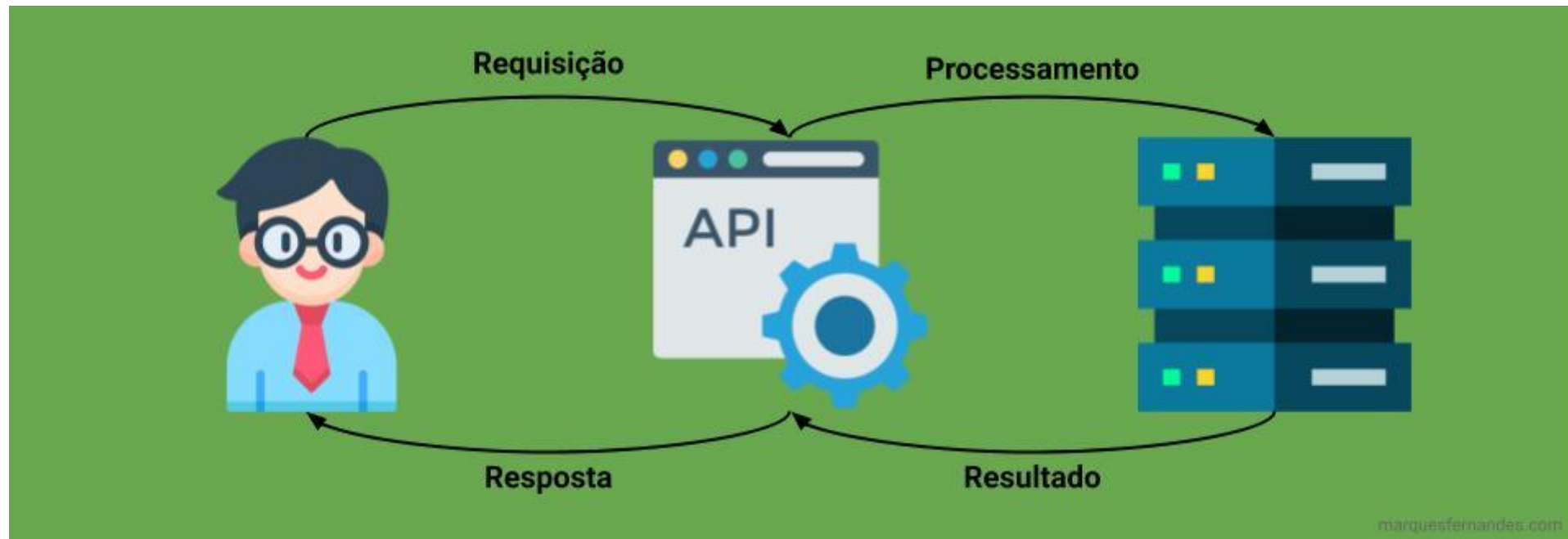
APIs Famosas

- WhatsApp: [https://www.whatsapp.com/business/api?lang=pt br](https://www.whatsapp.com/business/api?lang=pt_br)
- iFood: <https://developer.ifood.com.br/en-US/>

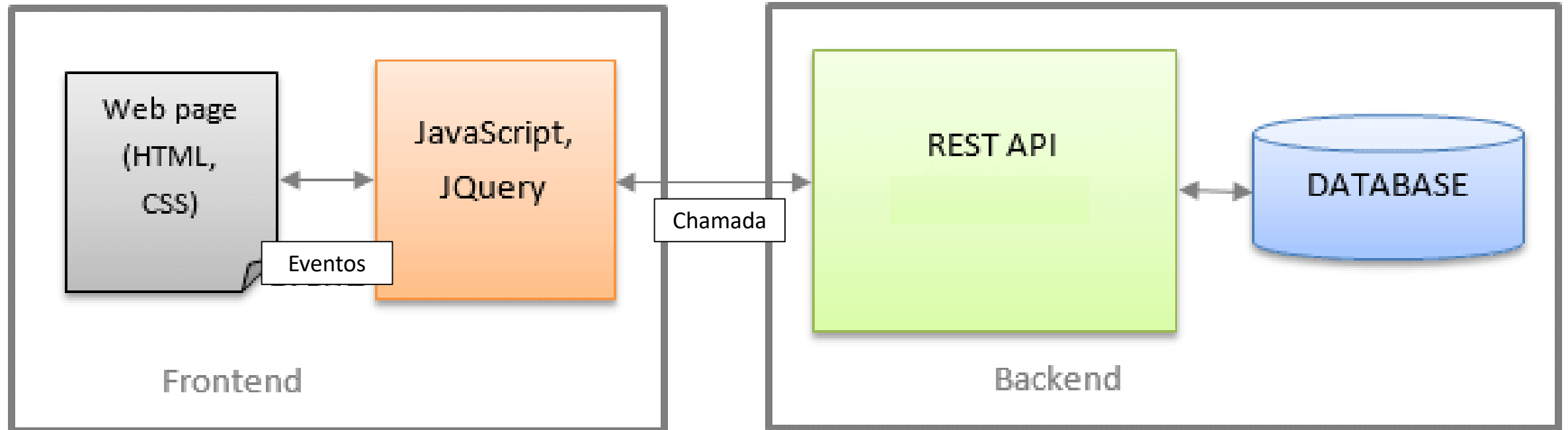
Como Funciona uma API



Como Funciona uma API



Como Funciona uma API

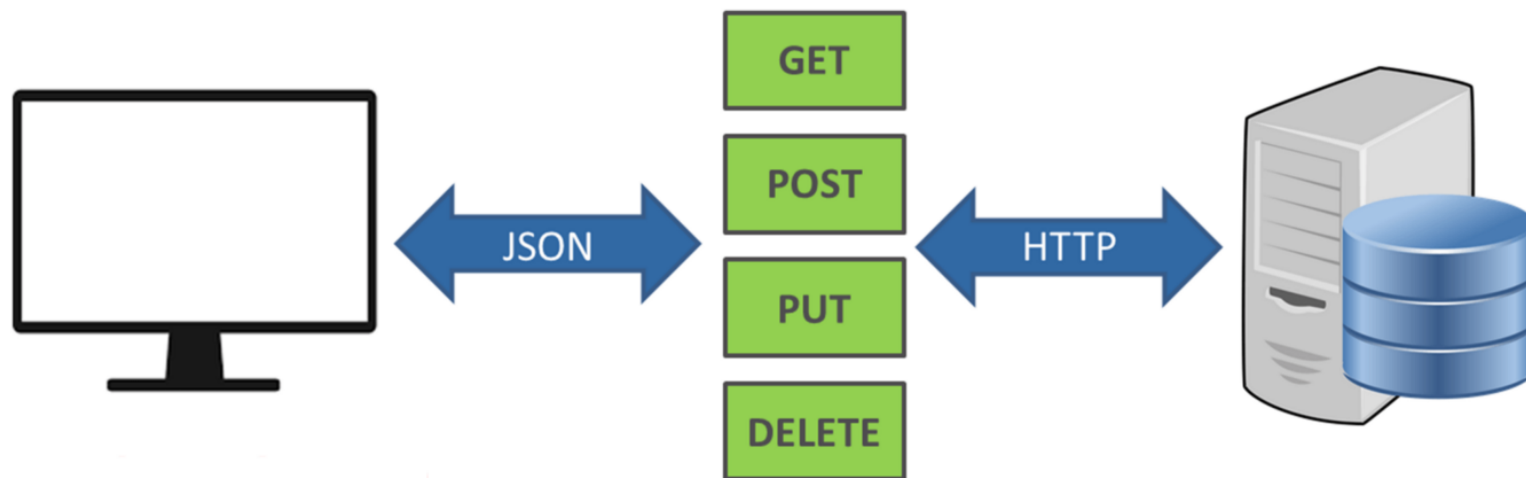


REST API

- REST é acrônimo para Representation State Transfer - Transferência de Estado Representacional.
- O nome completo é RESTFul API.
- Um REST API é a interface dessa transferência de dados. Isso ajuda a separar as aplicações back-end e front-end.

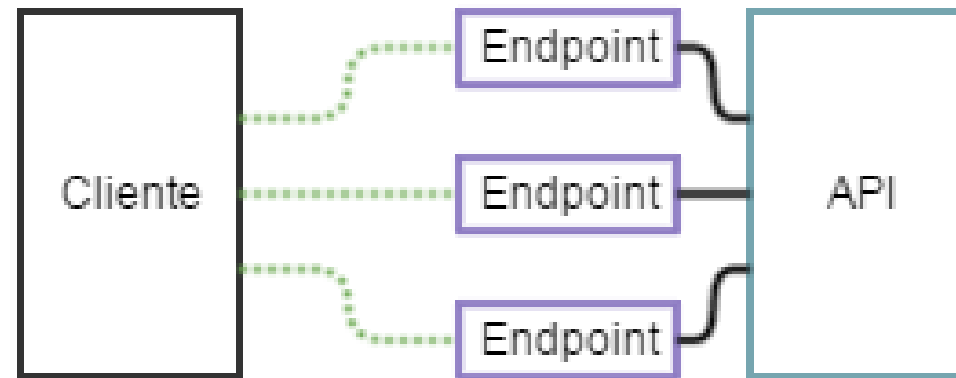
Operações REST

- As mais importantes são:
- **POST** – Enviar dados
- **GET** – Obter dados
- **PUT** – Atualizar dados
- **DELETE** – Apagar dados



EndPoints

- Uma API é composta por uma coleção de EndPoints.
- Um EndPoint é uma interface exposta pela API.



Exemplos

- Login
- Esqueceu Senha
- Cadastrar Usuário
- Editar Usuário
- Apagar Usuário

▼ Programação do Zero

POST api/usuario/login

POST api/usuario/esqueceusenha

POST api/usuario/cadastrar

PUT api/usuario/editar

DEL api/usuario/apagar

Criando nossa primeira API

Configurar seu novo projeto

API Web do ASP.NET Core

C#

Linux

macOS

Windows

Nuvem

Serviço

Web

Nome do projeto

ProgramacaoZero.Api

Local

C:\GitProjects\

...

Nome da solução ⓘ

ProgramacaoZero

☐ Colocar a solução e o projeto no mesmo diretório

POSTMAN

- Melhor Ferramenta para testar APIs
- Baixar: <https://www.postman.com/downloads/>

Exercícios

- Criar novo projeto de API no Visual Studio
- Baixar e instalar o Postman

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- Fazer os exercícios da Aula 2 em forma de API

Exercícios para Casa

- Finalizar todos os exercícios da Aula 2 em forma de API

AULA 9

API – parte 2

Recapitulando a aula anterior

- API
- REST
- EndPoint

Você vai aprender na aula de hoje

- Json
- Chamadas API a partir do Frontend
- CORS
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

- Exercícios – parte 2

O que é JSON

- JSON: é um acrônimo de **JavaScript Object Notation**. É um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas.



Exemplo de JSON

```
{
  "carro": {
    "marca": "Honda",
    "modelo": "Fit",
    "placa": "DXG-2343",
    "cor": "Preto"
  }
}
```

Habilitando CORS

- Cross-Origin Resource Sharing ou CORS é um mecanismo que permite que recursos restritos em uma página da web sejam recuperados por outro domínio fora do domínio ao qual pertence o recurso que será recuperado.

```
app.UseCors(x => x
    .SetIsOriginAllowed(origin => true)
    .AllowAnyMethod()
    .AllowAnyHeader()
    .AllowCredentials());
```

Exercícios

- 1 – Montar JSON Carro
- 2 – Acessar JsonViewer: <http://jsonviewer.stack.hu/>

Exercícios

- 3 – Montar as telas da aula 2 e chamar os endPoints da API:
 - 1 - Olá mundo
 - 2 - Olá mundo personalizado
 - 3 - Somar

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 4 – Montar os endPoints para as operações: login, cadastro e esqueceu senha

Exercícios para Casa

- 5 – Montar as telas de login, cadastro e esqueceu senha, chamando os endPoints da API

AULA 10

API – parte 3

Recapitulando a aula anterior

- Json
- CORS
- Chamadas API a partir do Frontend

Você vai aprender na aula de hoje

- Chamadas API com JSON a partir do Front-end:
 - Login
 - Cadastro
 - Esqueceu Senha

Exercícios

- Fazer as telas abaixo chamarem a API:
 - Login
 - Cadastro
 - Esqueceu Senha

AULA 11

Programação Orientada a Objetos – parte 1

Recapitulando a aula anterior

- Chamadas API do Front-end

Você vai aprender na aula de hoje

- O que é POO
- Classes e Objetos
- Atributos e Métodos
- Encapsulamento, Herança e Polimorfismo
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

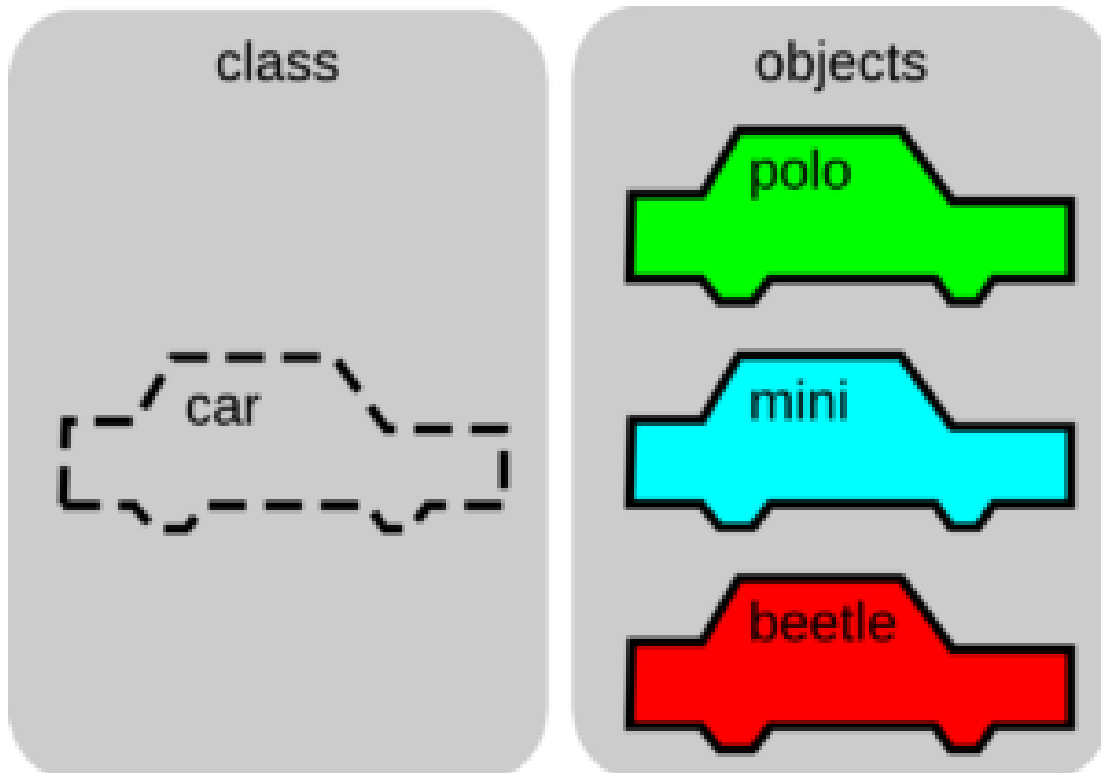
- Exercícios – parte 2

O que é POO

- POO – Programação Orientada a Objetos
- Uma forma de programar que tenta se aproximar da vida real.
- Um “Objeto” é a representação de algo tangível, como um Carro por exemplo.
- Esse novo paradigma se baseia principalmente em dois conceitos chave: **classes** e **objetos**.

Classes e Objetos

- Uma **classe** é um conjunto de características e comportamentos que definem o conjunto de objetos pertencentes à essa classe, como um **molde**.



Atributos e Métodos

- Imagine que você comprou um carro na loja.
- Esse carro possui **características**, como: cor, marca, modelo e placa.
- E também possui **comportamentos**, como: acelerar, frear, buzinar, dar seta.
- Podemos dizer que seu carro novo é um **Objeto**, onde suas características são seus **atributos** e seus comportamentos são ações ou **métodos**.
- Seu carro é um objeto seu mas na loja onde você o comprou existiam vários outros carros muito similares, com as mesmas características e comportamentos.

Classes e Objetos

- O seu carro novo será um **objeto**, uma **instância** da classe Carro.
- Exemplo de Classe: Marca, Modelo, Cor, Placa
- Exemplo de Objeto: Honda, Fit, Preto, CXE-2377

Construtor

- Determina que ações devem ser executadas quando um objeto é criado ou instanciado.

Exemplo de Classe em C#

```
1 referência
public class Carro
{
    1 referência
    public string Placa { get; set; }

    1 referência
    public string Modelo { get; set; }

    1 referência
    public string Marca { get; set; }

    1 referência
    public string Cor { get; set; }

    1 referência
    public void Acelerar()
    {
        //aqui vai a funcionalidade acelerar
    }

    1 referência
    public void Frear()
    {
        //aqui vai a funcionalidade frear
    }
}
```


Exemplo de Objeto em C#

```
var meuCarro = new Carro();  
  
meuCarro.Cor = "Preto";  
meuCarro.Marca = "Honda";  
meuCarro.Modelo = "Fit";  
meuCarro.Placa = "CXE-2377";
```

Exercícios

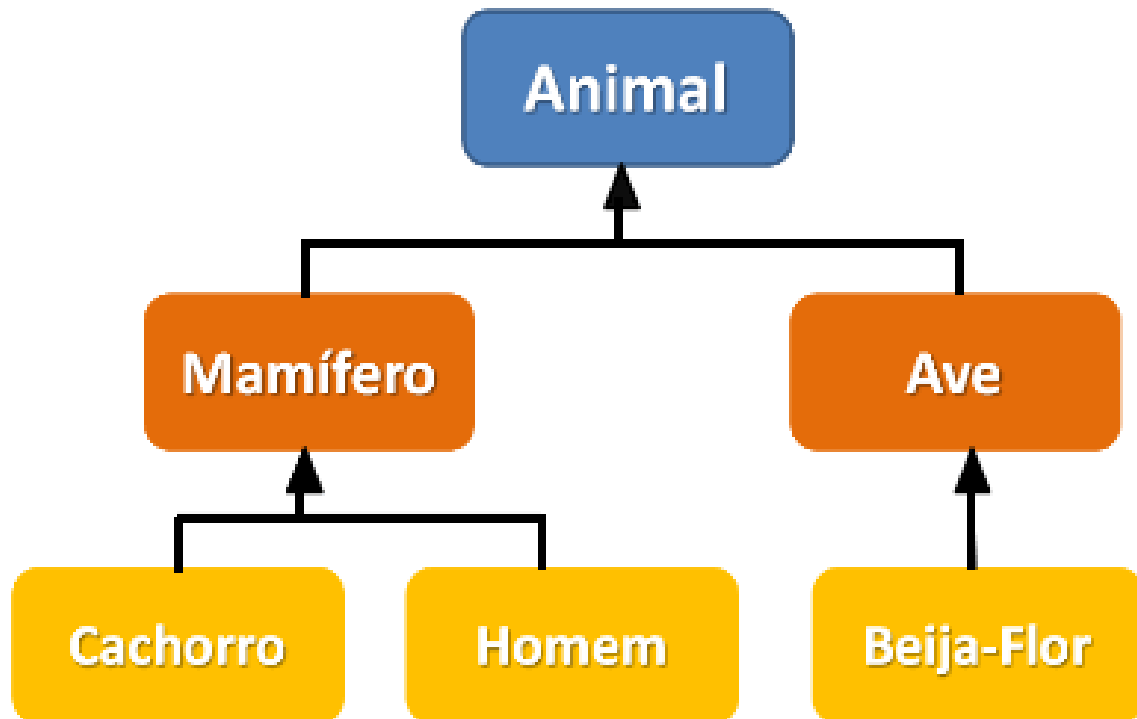
- 1 - Criar uma classe chamada Veiculo, com os seguintes atributos: Cor, Marca, Modelo, Placa, Tanque de Combustível e os seguintes métodos: Acelerar e Frear.
- 2 - Crie o construtor da classe Veiculo, e atribua o valor 40 para o atributo TanqueCombustivel.

Principais características da POO

- As 3 Principais características são: Herança, Encapsulamento e Polimorfismo

Herança

- Quando dizemos que uma classe Cachorro é um **tipo de** classe Mamífero, dizemos que a classe Cachorro **herda** as características da classe Mamífero e que a classe Mamífero é **mãe** da classe Cachorro, estabelecendo então uma relação de **herança** entre elas.



Exercícios

- 5 – Crie uma classe chamada Carro que herde de Veiculo
- 6 – Crie uma classe chamada Moto que herde de Veiculo
- 7 – Criar classe BaseResult, e fazer as classes LoginResult, CadastroResult e EsqueceuSenhaResult herdarem dela

BREAK 15 MINUTOS

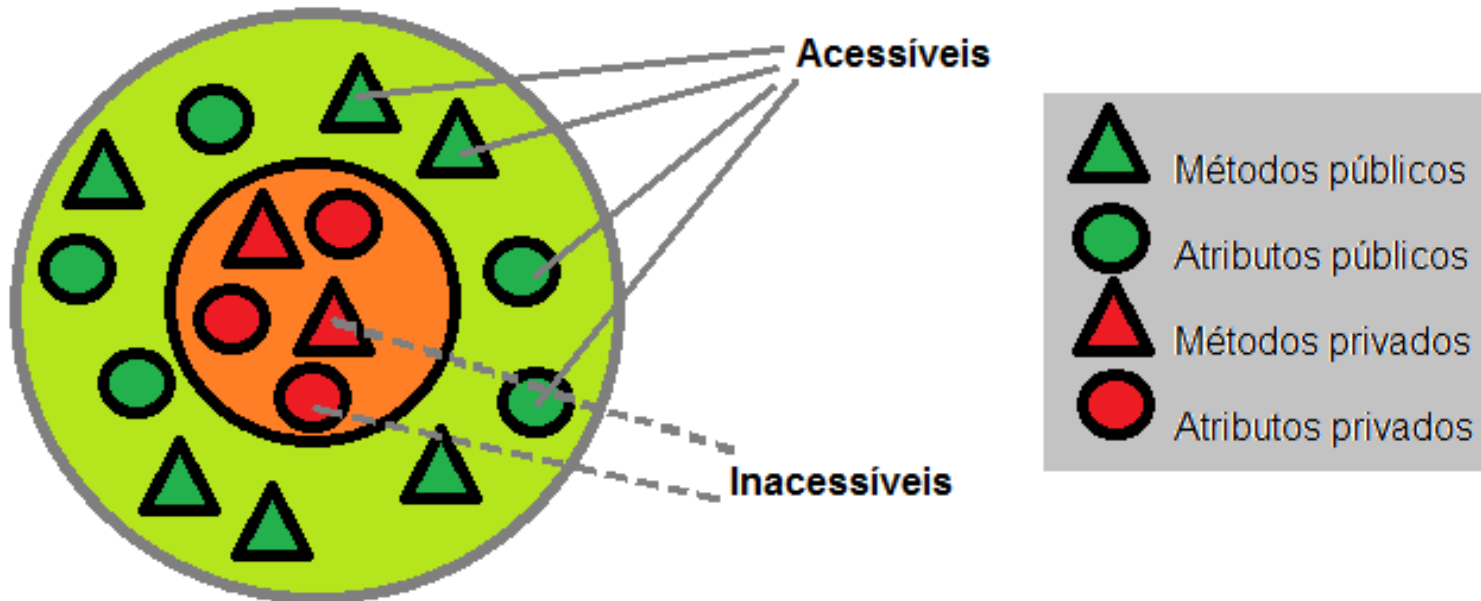
https://www.youtube.com/watch?v=u_BcMXgws6Y

Encapsulamento

- Ainda usando a analogia do carro, sabemos que ele possui atributos e métodos, ou seja, características e comportamentos.
- Os métodos do carro, como acelerar, podem usar atributos e outros métodos do carro como o tanque de gasolina e o mecanismo de injeção de combustível, respectivamente, uma vez que acelerar gasta combustível.
- No entanto, se alguns desses atributos ou métodos forem facilmente visíveis, isso pode dar liberdade para que alterações sejam feitas, resultando em efeitos colaterais imprevisíveis.

Encapsulamento

- Na POO, um atributo ou método que não é visível de fora do próprio objeto é chamado de "privado" e quando é visível, é chamado de "público".

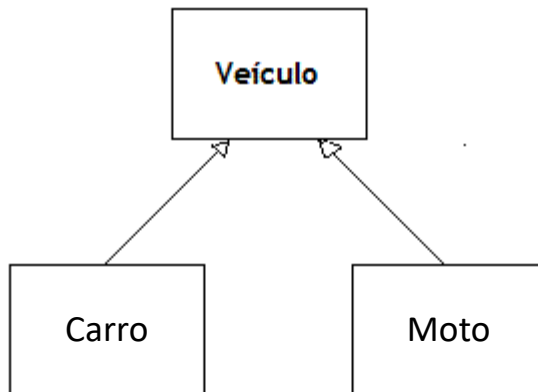


Exercícios

- 3 – Crie um método privado InjetarCombustível() na classe Veiculo. Subtraia 1 do atributo TanqueCombustível, e faça o método Acelerar() chamar o método InjetarCombustível()

Polimorfismo

- Polimorfismo é quando dois objetos, de duas classes diferentes, têm um mesmo método que é implementado de formas diferentes, ou seja, um método que possui várias formas, mas que possuem o mesmo efeito.
- Exemplo:



Exercícios

- 10 – Permita que os métodos Acelerar() e Frear() possam ser sobrescritos.
- 11 – Sobrescreva os métodos nas classes Carro e Moto

AULA 12

Programação Orientada a Objetos – parte 2

Recapitulando a aula anterior

- O que é POO
- Classes e Objetos
- Atributos e Métodos
- Encapsulamento, Herança e Polimorfismo

Você vai aprender na aula de hoje

- Divisão de Responsabilidades
- Controllers, Services e Repositories
- Models e Entities
- Exercícios parte 1

20:45 Intervalo de 15 minutos

- Exercícios parte 2

Divisão de Responsabilidades

- Divisão de responsabilidades(separation of concerns) é um princípio fundamental na engenharia de software.
- É uma forma de **organizar pensamentos** e modularizar seu software.
- Visa facilitar o entendimento para outros programadores e para você mesmo no futuro.

Dividindo as responsabilidades da Api

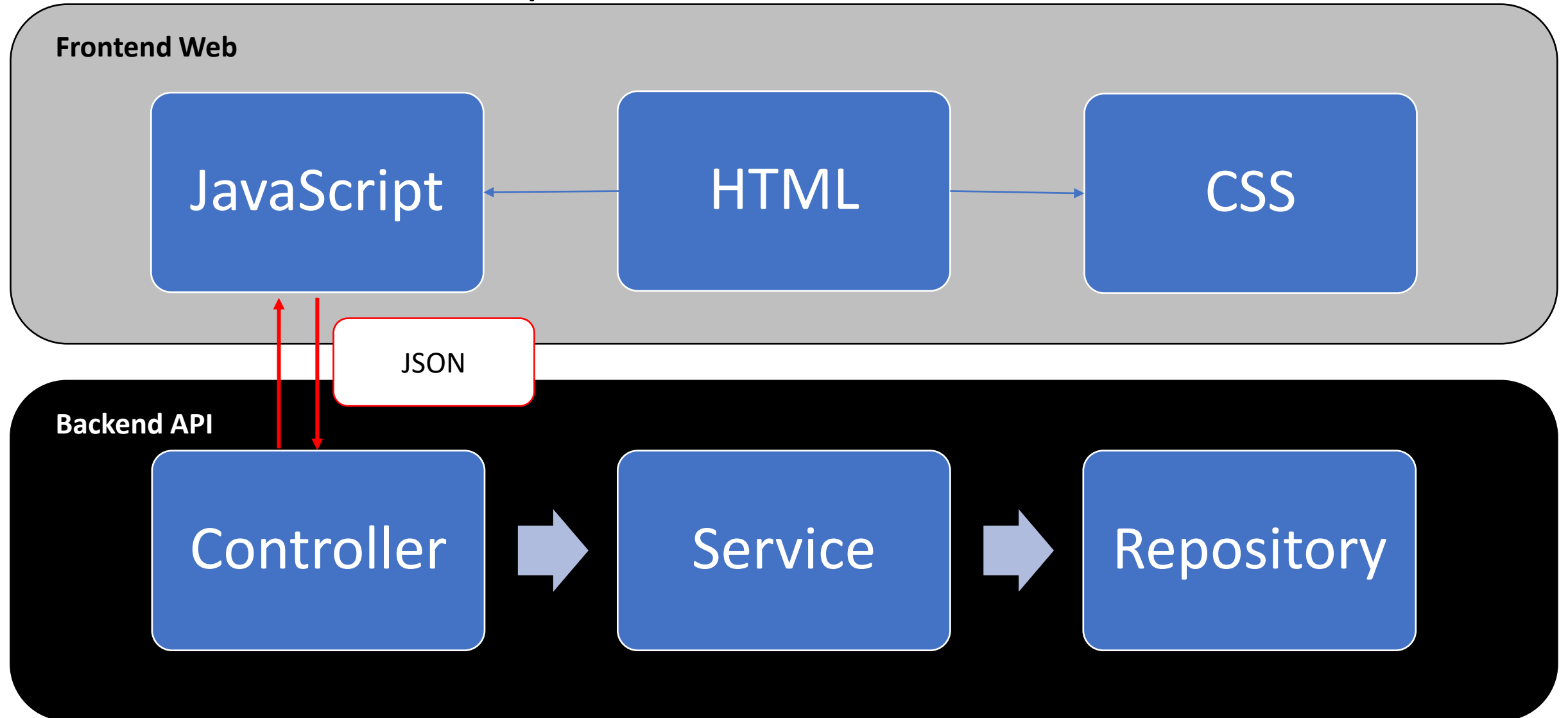
- **Controller:** Responsável por receber e enviar dados do Frontend, garantir que estão corretos, e interagir com os Serviços;
- **Service:** Responsável por conter as regras de negócio do sistema;
- **Repository:** Responsável por Ler, Inserir, Atualizar e Apagar dados;



Divisão de Responsabilidades

- **Models:** objetos recebidos e enviados pela Controller, e podem ser de um dos dois tipos: Request ou Response
- **Entities:** objetos que irão mapear o banco de dados e conter os dados do sistema

Divisão de Responsabilidades



Exercícios

- 1 – Criar a classe `UsuarioService`, com 3 métodos: `Login()`, `Cadastrar()` e `EsqueceuSenha()`
- 2 – Criar a classe `UsuarioRepository`, com 2 métodos: `Inserir()` e `ObterPorEmail()`
- 3 – Criar a classe `Usuario`, com os atributos: `Nome`, `Sobrenome`, `Email`, `Telefone`, `Gênero` e `Senha`

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 4 – Desenvolver lógica nos métodos Login, Cadastro e Esqueceu Senha da classe UsuarioService

Pra Casa:

- Instalar o MySql: <https://dev.mysql.com/downloads/mysql/>

AULA 13

Banco de Dados – parte 1

Recapitulando a aula anterior

- Divisão de Responsabilidades
- Controllers, Services e Repositories
- Models e Entities

Você vai aprender na aula de hoje

- O que é Banco de Dados
- Tabelas, Colunas, Chaves e Linhas
- Linguagem SQL
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

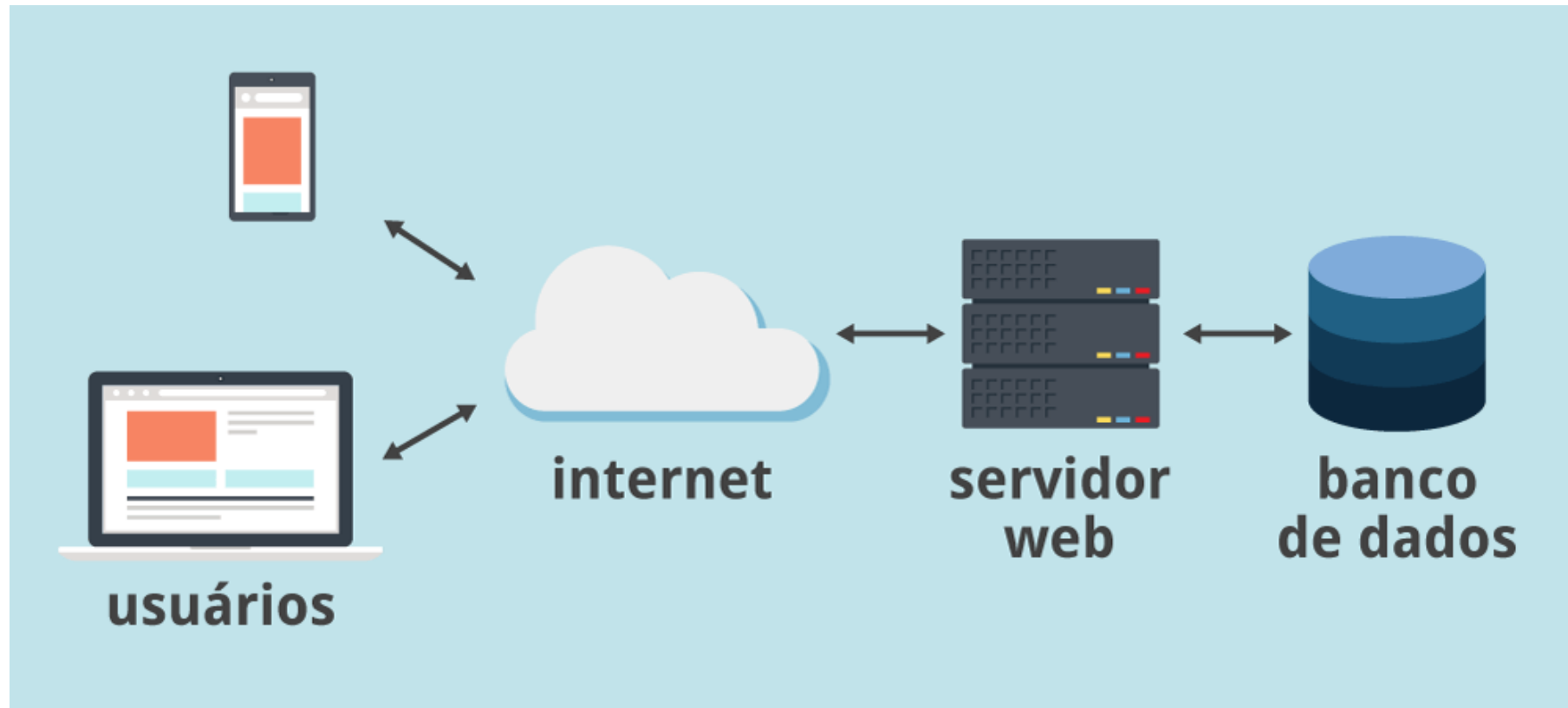
- Exercícios – parte 2

O que é banco de dados

- Bancos de dados ou bases de dados são conjuntos de arquivos relacionados entre si com registros sobre pessoas, lugares ou coisas.

O que é banco de dados

- Bancos de dados ou bases de dados são conjuntos de arquivos relacionados entre si com registros sobre pessoas, lugares ou coisas.



MySQL

- É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo.
- Link para download: <https://dev.mysql.com/downloads/>

Banco de Dados Relacional

- Sistema de Gerenciamento de Banco de Dados Relacionais (SGBDR).
- É um software que controla o armazenamento, recuperação, exclusão e integridade dos dados em **tabelas** que são relacionadas entre si.

Tabelas e Colunas

- Tabelas são organizadas em colunas, e cada coluna armazena um tipo de dados (inteiro, números reais, strings de caracteres, data, etc.)

Linhas

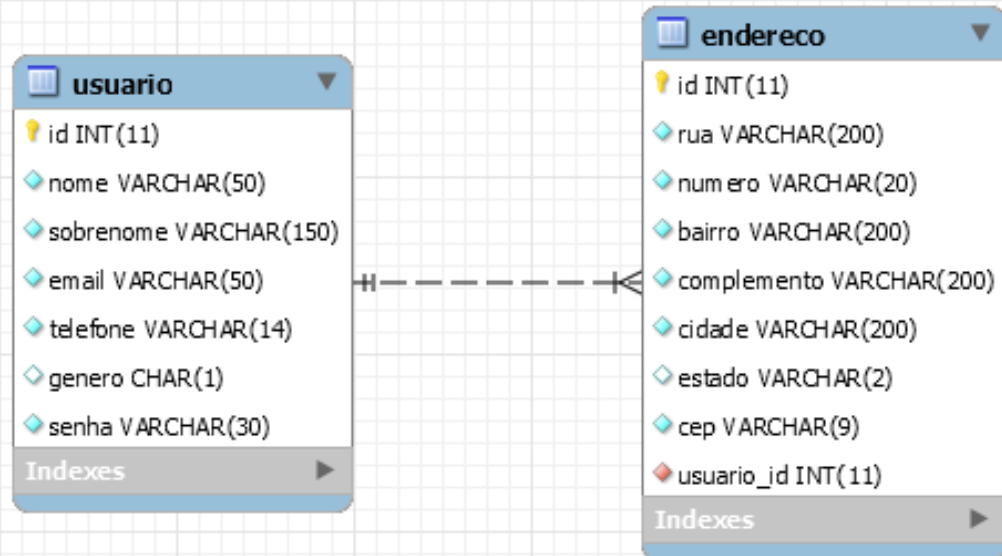
- Os dados de uma simples “instância” de uma tabela são armazenados como uma **linha**.
- Exemplo:
 - Tabela: Cliente
 - Colunas: Nome, Sobrenome e Telefone.
 - Linha 1: Renato, Gava, 11 94342-3234
 - Linha 2: Gustavo, Pereira, 44 23335-23434

Chaves

- Tabelas tipicamente possuem **chaves**. Uma coluna que identifica unicamente uma linha na tabela é chamada de **chave primária**.
- No caso da tabela cliente, a chave seria coluna Id. Exemplo:
 - Colunas: Id, Nome, Sobrenome, Telefone
 - Linha 1: 1, Renato, Gava, 11 94342-3234
 - Linha 2: 2, Gustavo, Pereira, 44 23335-23434

Chaves

- Tabelas também possuem **chaves estrangeiras**, que servem para relacionar tabelas.
- Exemplo: um usuário pode possuir um ou mais endereços.



CRUD

- CRUD - Create, Read, Update e Delete – que significa as operações de Inserção, Leitura, Atualização e Exclusão de dados.
- A forma mais fácil de manipular um banco de dados relacional é submeter declarações escritas na **linguagem SQL** a ele.

Linguagem SQL

- Structured Query Language, ou Linguagem de Consulta Estruturada ou SQL, é a linguagem de pesquisa declarativa padrão para banco de dados relacional.

Linguagem SQL

- **Linguagem de Definição de Dados** ou DDL (Data Definition Language): A DDL permite ao usuário permite a manipulação de tabelas e elementos associados, tipo chave primária e chaves estrangeira, índices, etc. Os principais comandos são CREATE, DROP, ALTER(em algumas situações).
- **Linguagem de Manipulação de Dados** ou DML (Data Manipulation Language): Utilizada para Selecionar(SELECT), Inserir(INSERT), Atualizar(UPDATE) e Apagar(DELETE).

Exercícios

- 1 - Criar o banco de dados programacao_do_zero.
- 2 - Criar a tabela Usuário com as seguintes colunas: Id, Nome, Sobrenome, Telefone, E-mail, Gênero e Senha.
- 3 - Criar a tabela Endereço com as seguintes colunas: Id, Rua, Número, Bairro, CEP, Complemento, Cidade e Estado.
- 4 - Relacionar as tabelas Usuário e Endereço.

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- 5 – Inserir usuário
- 6 – Selecionar usuário (todos e somente 1)
- 7 – Alterar usuário
- 8 – Deletar usuário
- 9 – Inserir Endereços para o usuário 1
- 10 – Selecionar os endereços do usuário 1

AULA 14

Banco de Dados – parte 2

Recapitulando a aula anterior

- O que é Banco de Dados
- Tabelas, Colunas, Chaves e Linhas
- Linguagem SQL

Você vai aprender na aula de hoje

- ADO.NET
- Connection String
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

- Exercícios – parte 2

ADO.NET

- O ADO.Net é um mecanismo de acesso a dados que contem todos os recursos necessários para acesso e manipulação de dados.

Connection String

- É uma cadeia de caracteres de conexão, ou **string de conexão**, é uma cadeia de caracteres que especifica informações sobre uma fonte de dados e os meios de conexão com ela.

- Exemplo:

server=SEU-SERVIDOR; **database**=SEU-BANCO; **uid**=SEU-USUARIO;
pwd=SUA-SENHA;

ADO.NET

- **MySqlConnection:** cria, abre e fecha a conexão com o banco de dados
- **MySqlCommand:** executa os comandos no banco de dados

Exercícios

- 1 - Instalar o pacote MySql.Data
- 2 - Configurar a connection string no appsettings.json
- 3 - Fazer os métodos: Inserir() e ObterPorEmail()

AULA 15

Controle de Acesso – parte 1

Recapitulando a aula anterior

- ADO.NET
- Connection String

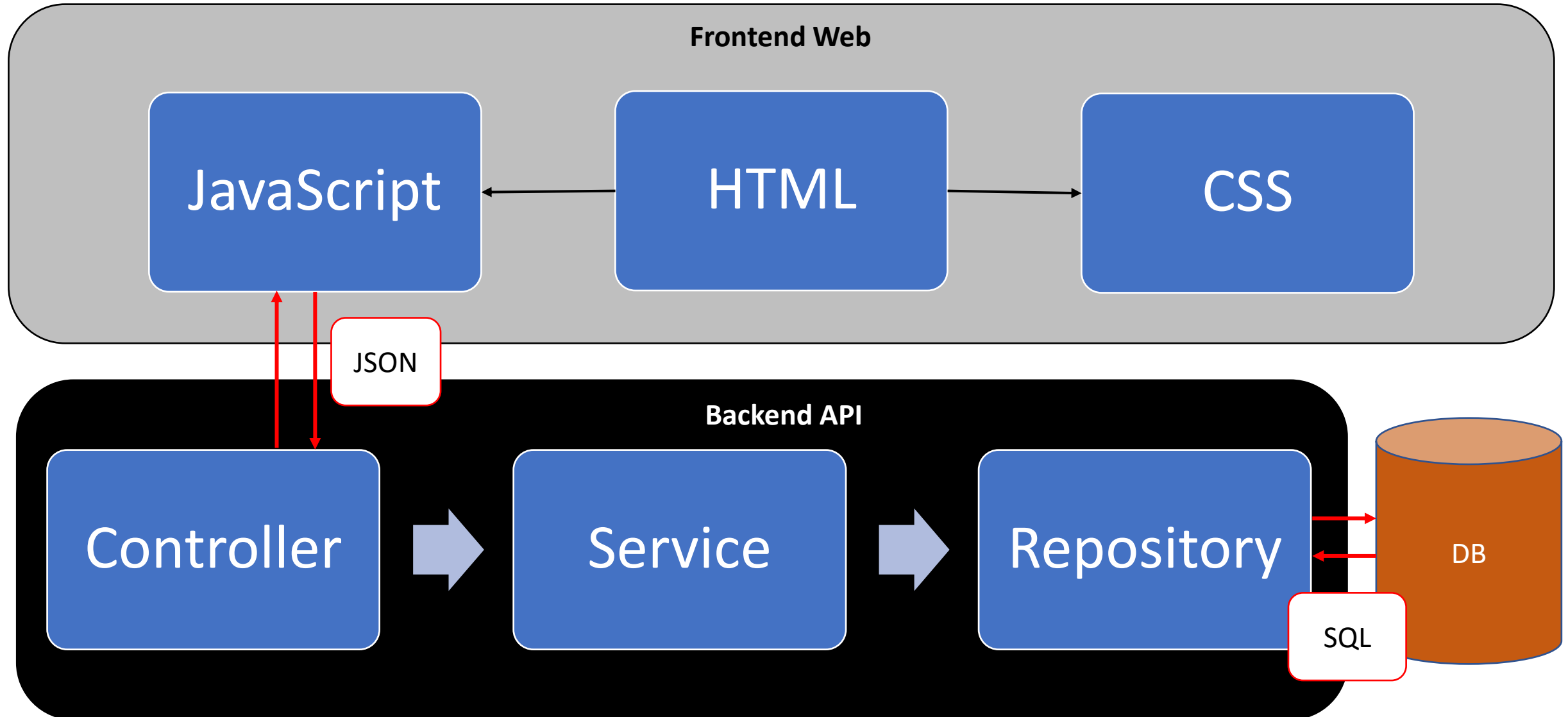
Você vai aprender na aula de hoje

- Visão Geral do Sistema
- Controle de Acesso
- Exercícios – parte 1

20:45 Intervalo de 15 minutos

- Exercícios – parte 2

Visão Geral do Sistema



Exercícios

- Adicionar campo usuarioGuid na tabela usuário
- Retornar usuarioGuid no cadastro e no login

BREAK 15 MINUTOS

https://www.youtube.com/watch?v=u_BcMXgws6Y

Exercícios

- Criar página Home
- Armazenar o usuarioGuid no localStorage no Cadastro e no Login

AULA 16

Parte Final – Disparo de E-mail, Home, Login e Logout

Recapitulando a aula anterior

- Adicionar campo `usuarioGuid` na tabela usuário
- Retornar `usuarioGuid` no cadastro e no login
- Criar página Home
- Armazenar o `usuarioGuid` no `localStorage` no Cadastro e no Login

Você vai aprender na aula de hoje

- Criar endPoint obterUsuario (controller, service e repository)
- Chamada para o endPoint a partir da Home (mensagem de bem-vindo)
- Logout
- Disparo de e-mail esqueceu senha

Conclusão

- Preencha o formulário: <https://forms.gle/F3aanSL8KMCFof5y7>
- Emissão do Certificado
- Suporte via Telegram até o final do ano

Próximos Passos

- Monte seu perfil no LinkedIn, adicione o curso, pesquise por vagas e faça contato
- Ingresse em curso superior (tecnólogo, EAD) para poder conseguir estágio
- Siga desenvolvendo novos recursos no projeto, como: cadastro de clientes, produtos, pedidos etc
- Busque outros cursos e siga estudando