Aula 07

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

Síncrono e Assíncrono

Síncrono e Assíncrono

Os processamentos síncrono e assíncrono são modelos de processamento utilizados na programação, que estão diretamente ligados ao fluxo de execução da aplicação, eles determinam como a nossa aplicação será executada.

Síncrono e Assíncrono

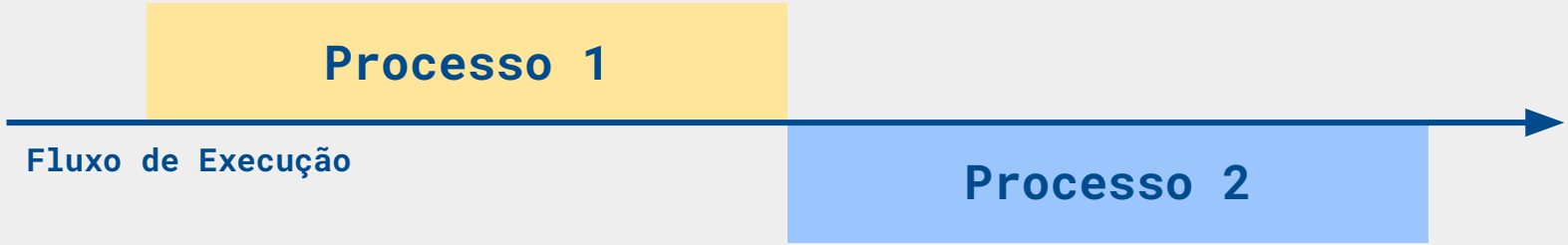
- Processamento Síncrono

No processamento síncrono cada operação, cada processo, cada parte do código, precisa ter a sua execução finalizada para poder executar a próxima operação, o próximo processo, a próxima parte do código.

Ou seja, as operações acontecem em uma sequência.

Síncrono e Assíncrono

- Processamento Síncrono



Síncrono e Assíncrono

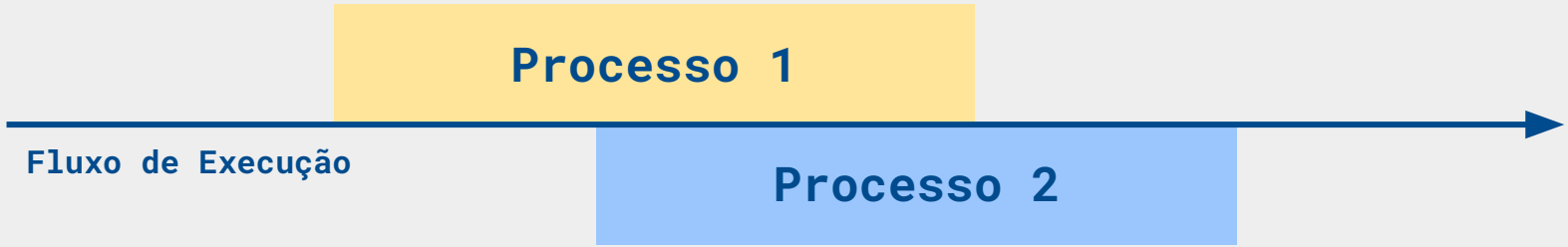
- Processamento Assíncrono

Já no processamento assíncrono cada operação, cada processo, cada parte do código, **NÃO** precisa ter a sua execução finalizada para poder executar a próxima operação, o próximo processo, a próxima parte do código.

Ou seja, as operações acontecem em forma alternada.

Síncrono e Assíncrono

- Processamento Assíncrono



Síncrono e Assíncrono

- Processamento Assíncrono

Nesse contexto, nós fazemos uso do processamento assíncrono em APIs para permitir que as rotas sejam executadas por vários usuários, várias vezes, ao mesmo tempo, sem a necessidade de esperar a primeira execução finalizar para começar a próxima.

Síncrono e Assíncrono

- C#: Processamento Assíncrono

No C#, assim como na maior parte das linguagens de programação, fazemos uso da palavra-chave async para definir que os métodos sejam executados em processamento assíncrono.

Síncrono e Assíncrono

- C#: Processamento Assíncrono

```
public async void MetodoAssincrono()
{
    // Bloco de código do método
}
```

Síncrono e Assíncrono

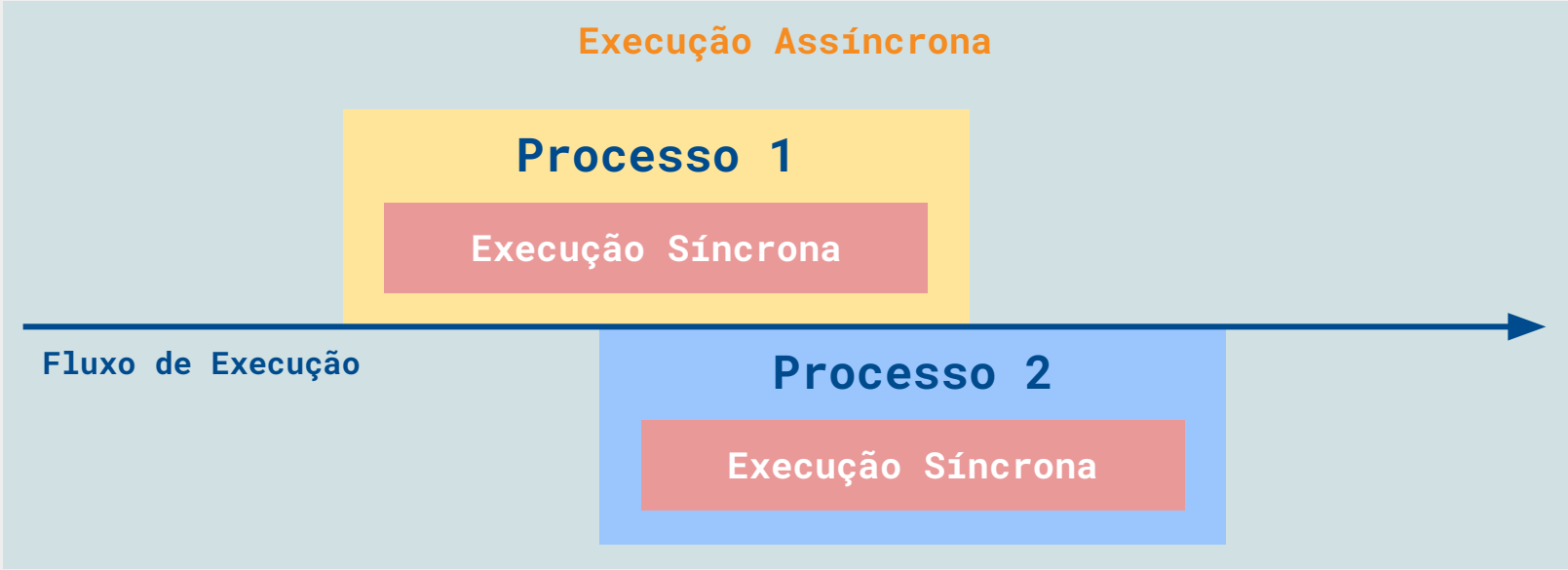
- Processamento Assíncrono

Importante lembrar que, dentro do método assíncrono, a execução ainda acontece de forma síncrona.

A definição do método como assíncrono, apenas define que aquele método pode ser executado várias vezes ao mesmo tempo, e independente do restante do processo, mas internamente, ele ainda é executado de forma síncrona.

Síncrono e Assíncrono

- Processamento Assíncrono



Síncrono e Assíncrono

- C#: Processamento Assíncrono

Mas quando trabalhamos com métodos com retorno, somente a palavra-chave async não é suficiente, pois será necessário que o processamento finalize, para só então retornar o valor do método.

Para esses casos, precisamos também de um carinha chamado Task<TResult>, que representa um processo assíncrono que retorna um valor.

Síncrono e Assíncrono

- C#: Processamento Assíncrono

O Task<TResult> funciona meio que como uma “promessa”, informando para a nossa aplicação que, quando o método assíncrono finalizar o processo, ele vai retornar o valor determinado por ele.

Para determinar o valor de retorno dessa “promessa”, basta informar o tipo de retorno no lugar do TResult.

Síncrono e Assíncrono

- C#: Processamento Assíncrono

Se eu preciso retornar a promessa de uma string, eu uso:

```
Task<string>
```

Se eu preciso retornar a promessa de um objeto Pessoa, eu uso:

```
Task<Pessoa>
```

Síncrono e Assíncrono

- C#: Processamento Assíncrono

```
public async Task<string> MetodoAssincrono()
{
    // Bloco de código do método
}
```

Síncrono e Assíncrono

- C#: Processamento Assíncrono

```
public async Task<Pessoa> MetodoAssincrono()
{
    // Bloco de código do método
}
```

Síncrono e Assíncrono

- C#: Processamento Assíncrono

Quando eu tenho um método assíncrono, que faz a chamada de outro método assíncrono, eu tenho a possibilidade de forçar a espera da finalização de um processo, para dar continuidade ao resto do processo, fazendo uso da palavra-chave await.

Síncrono e Assíncrono

- C#: Processamento Assíncrono

```
public async Task<string> MetodoAssincrono()
{
    // Processo do método
    string texto = await TextoAssincrono();
    // Processo do método
}

public async Task<string> TextoAssincrono()
{
    // Bloco de código do método
}
```

Exercício

Exercício

gg.gg/BackEnd07

github.com/hmaltaurodev/slides