Flutter Avançado - Aula 4

Exercícios 10 a 12

Fernando Duwe

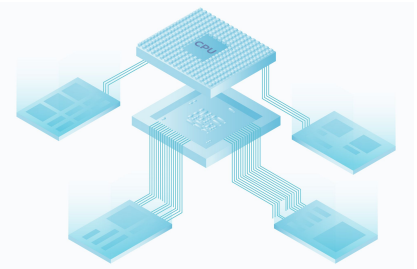
v 1.0

Dicas de hoje

- Flutter Flare - O Login mais bonito do mundo! Ver mais

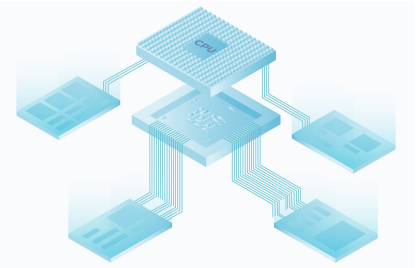
Ver mais

- Uma thread é unidade de um processo, um pequeno conjunto de instruções criadas para serem executadas independentes de do processo pai
- Dart é *single thread*, ou seja, nossa aplicação é executada utilizando somente uma Thread
- Quando utilizamos funções assíncronas, estamos trabalhando na mesma thread. Caso nossa função seja uma função demorada, poderá transparecer ao usuário essa demora



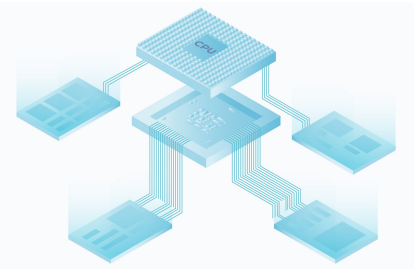
Ver mais

- Para resolver problemas mais complexos ou pesados, podemos trabalhar com múltiplas threads em Dart
- Ao criar uma nova thread, ela não compartilhará o mesmo espaço em memória com outras threads, ficando isolada, por isso nós a chamamos de isolates
- Para criar novos isolates, podemos utilizar duas formas diferentes: compute() e Isolate.spawn()



- O compute é o método nativo do Flutter para criação de threads
- Seu funcionamento é muito simples, ao ser chamado, criará uma thread para a função passada por parâmetro
- Abaixo, o parâmetro Q é o parâmetro passado para a função
- R é o tipo de retorno da função

```
compute<Q, R>(isolates.ComputeCallback<Q, R> callback);
```

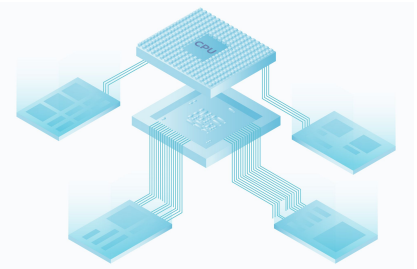


Ver mais

Exemplo:

```
double countTo(RangeValues prRange) {
     double vrI = 0;

     for (vrI = prRange.start; vrI <= prRange.end; vrI++)
          print(vrI);

     return vrI;
}


compute<RangeValues, double>(countTo, RangeValues(100, 500));
```

- O Isolate.spawn é o método padrão do Dart para a criação de threads
- Pode ser utilizado da mesma forma que o compute é utilizado
- Sua principal diferença em relação ao compute, é que podemos controlar melhor seu funcionamento, se necessário
- Com o spawn, é possível também trocar mensagens entre duas ou mais threads



Ver mais

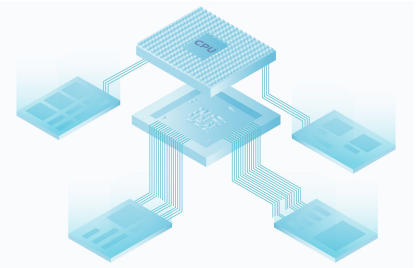
Exemplo básico usando spawn:

```
double countTo(RangeValues prRange) {
     double vrI = 0;

     for (vrI = prRange.start; vrI <= prRange.end; vrI++)
          print(vrI);

     return vrI;
}


Isolate.spawn<RangeValues>(this.countTo, RangeValues(100, 200));
```

- Podemos também, com o spawn controlar seu fluxo, pausando e retornando o seu funcionamento
- Para isso, usamos os comandos a seguir

Capturando a instância do Isolate

```
Isolate vrI = await Isolate.spawn<RangeValues>(widget.countTo, RangeValues(1, 500));
```

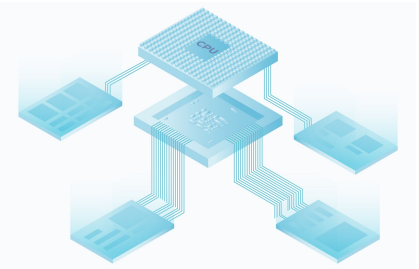


Matando o processo:

```
vrI.kill();
```

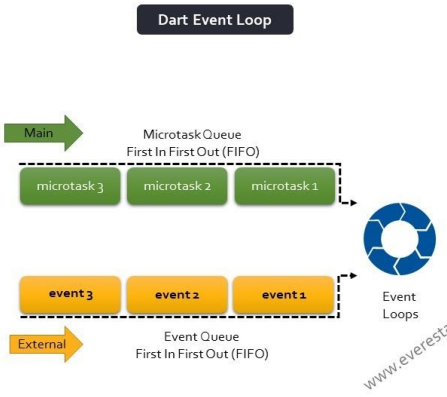
Pausando o processo:

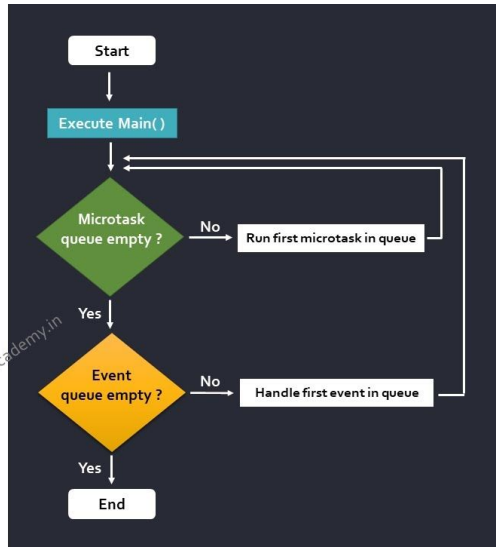
```
Capability vrCap = vrI.pause(vrI!.pauseCapability);
```

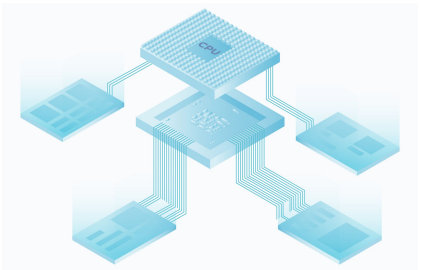


Resumindo o processo:

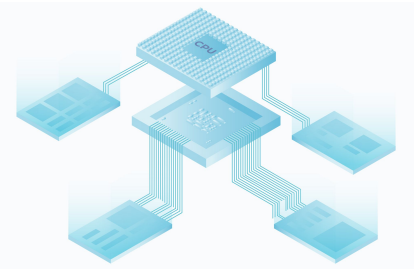
```
vrI.resume(vrCap);
```







- Com spawn, podemos também fazer com que duas ou mais threads possam conversar entre si
- Para isso, utilizamos dois objetos ReceivePort e SendPort
- Eles funcionarão como streams, transportando uma as mensagens para dentro e para fora do spawn
- Ambos implementam streams, na verdade



Isolate.spawn - cont

Ver mais

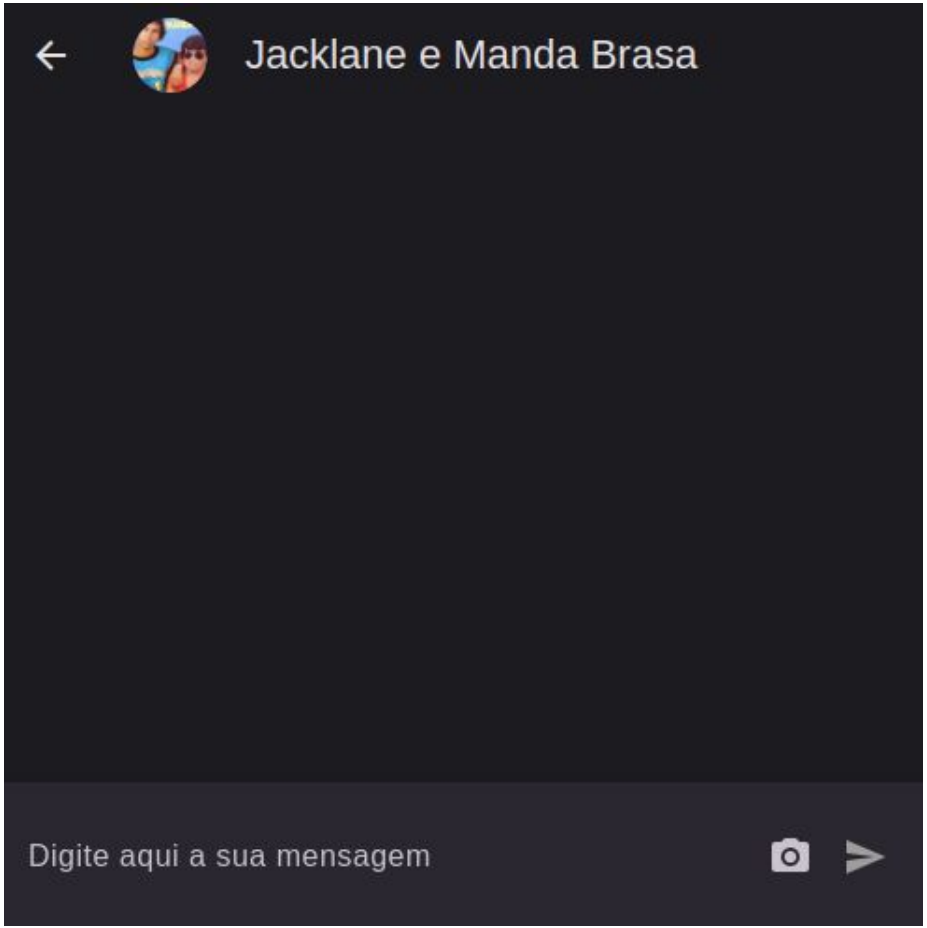
```
Future<void> createIsolate() async {
      // Porta por onde eu escutarei os retornos da thread
      ReceivePort vrReceivePort = ReceivePort();

      // Crio o Isolate
      Isolate.spawn<SendPort>(widget.sumTo, vrReceivePort.sendPort);

      // Fico no aguardo, pois o método dentro da thread precisa me devolver uma porta para que eu possa enviar   //
mensagens
      SendPort vrSendPort = await vrReceivePort.first;

      // Crio outra porta para escutar o retorno da mensagem que eu enviar
      ReceivePort vrReceiveResponsePort = ReceivePort();

      // Mando uma mensagem e a porta pela qual a thread deve me responder
      vrSendPort.send([
            1.0,
            10000.0,
            vrReceiveResponsePort.sendPort
      ]);

      // Esperando a resposta chegar
      final vrResposta = await vrReceiveResponsePort.first;

      print(vrResposta);
}
```

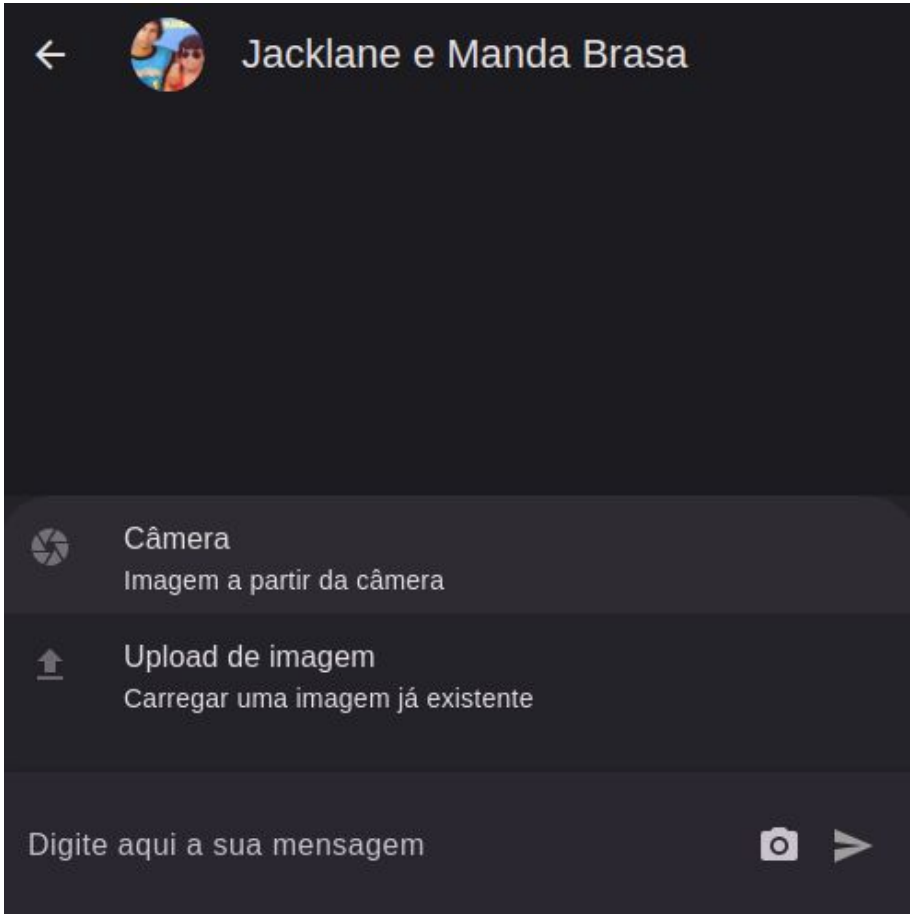
Isolate.spawn - cont

Ver mais

```
Future<void> sumTo(SendPort prSend) async {
      // Porta por onde eu escutarei as respostas de fora da Thread
      ReceivePort vrReceivePort = ReceivePort();

      // Envio de volta a porta pela qual eu escutarei os retornos
      prSend.send(vrReceivePort.sendPort);

      // Fica ouvindo as mensagens enviadas para dentro da thread
      await for (var vrMessage in vrReceivePort) {
            if (vrMessage is List) {
                  print("Mensagem recebida");

                  final vrInitialValue = vrMessage[0];
                  final vrFinalValue = vrMessage[1];

                  double vrTotal = 0;

                  for(double vrI = vrInitialValue; vrI <= vrFinalValue; vrI++)
                        vrTotal += vrI;

                  final SendPort vrPortToSend = vrMessage[2];

                  vrPortToSend.send(vrTotal);
            }
      }
}
```

- Vamos criar agora o upload de imagem
- Para isso, utilizaremos o plugin image_picker
- O Firebase Storage também precisa ser configurado dentro do projeto
- Para que o código de upload funcione na web, alguns ajustes especiais precisam ser feitos
- Neste exercício, vamos subir a imagem para o storage e gravar uma mensagem somente

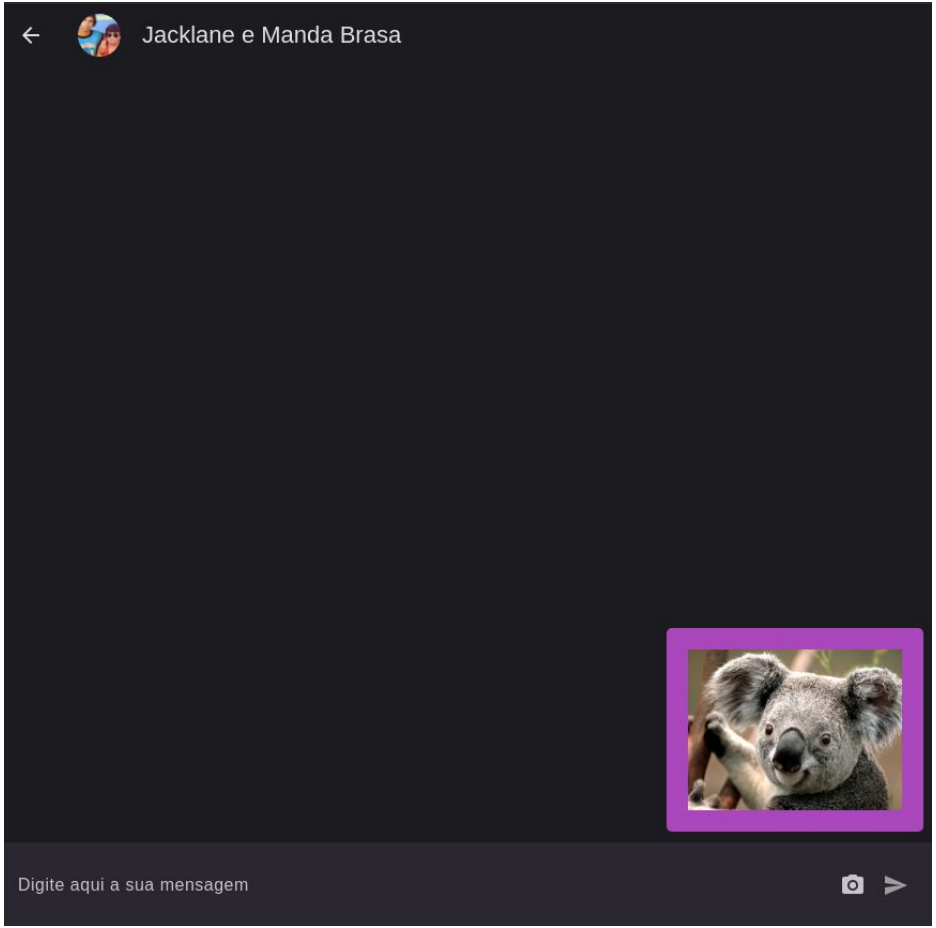

Jacklane e Manda Brasa
Digite aqui a sua mensagem

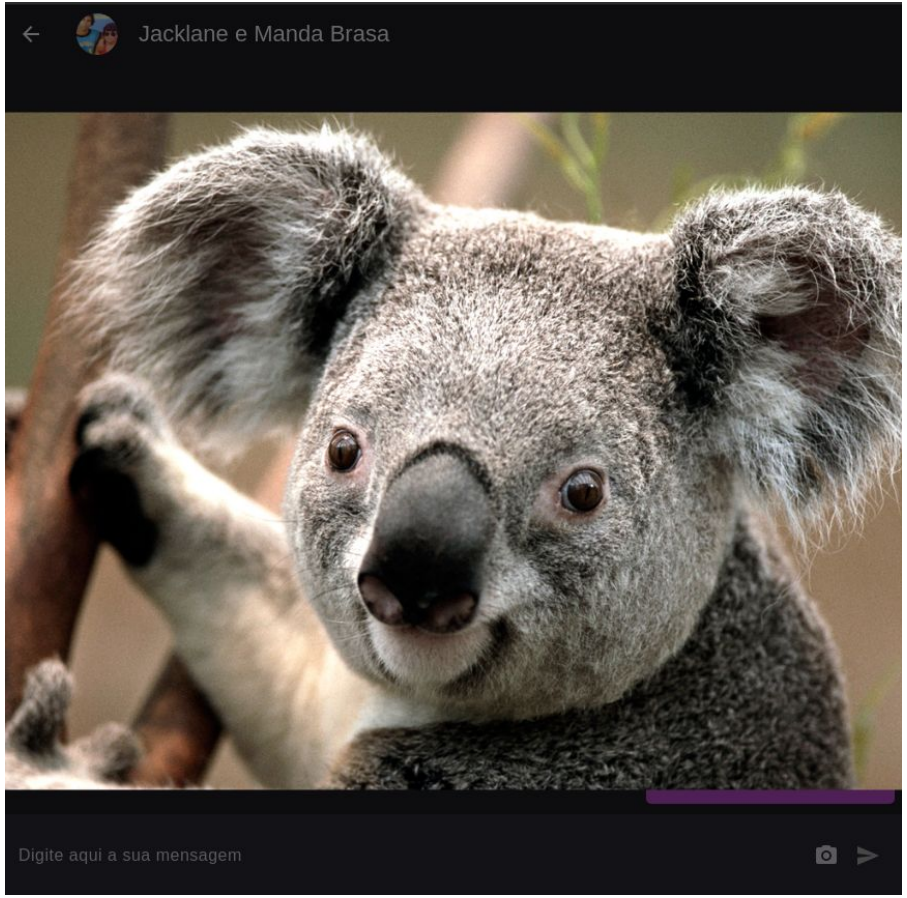

Jacklane e Manda Brasa
Câmera
Imagem a partir da câmera
Upload de imagem
Carregar uma imagem já existente
Digite aqui a sua mensagem

- Vamos agora alterar o objeto de mensagem para que exiba as mensagens com imagens
- Para isso, deve ser verificado o tipo da mensagem
- Se for do tipo texto, exibimos como hoje é feito
- Se for do tipo image, exibimos a imagem, usando a URL salva em params
- Ao clicar sobre a mensagem, abrimos a imagem em tamanho real em um popup ou uma tela separada

Exercício 11 - cont

Ver mais





- Aproveitando que estamos trabalhando com upload de imagens, vamos alterar também a página de perfil do usuário
- Crie um menu Popup, com as opções para ver a foto, subir uma nova foto a partir da galeria ou a partir da câmera
- Ao salvar, gravar também o novo caminho
- O upload da imagem pode ser feito ao clicar no botão

Exercício 12 - cont

Ver mais