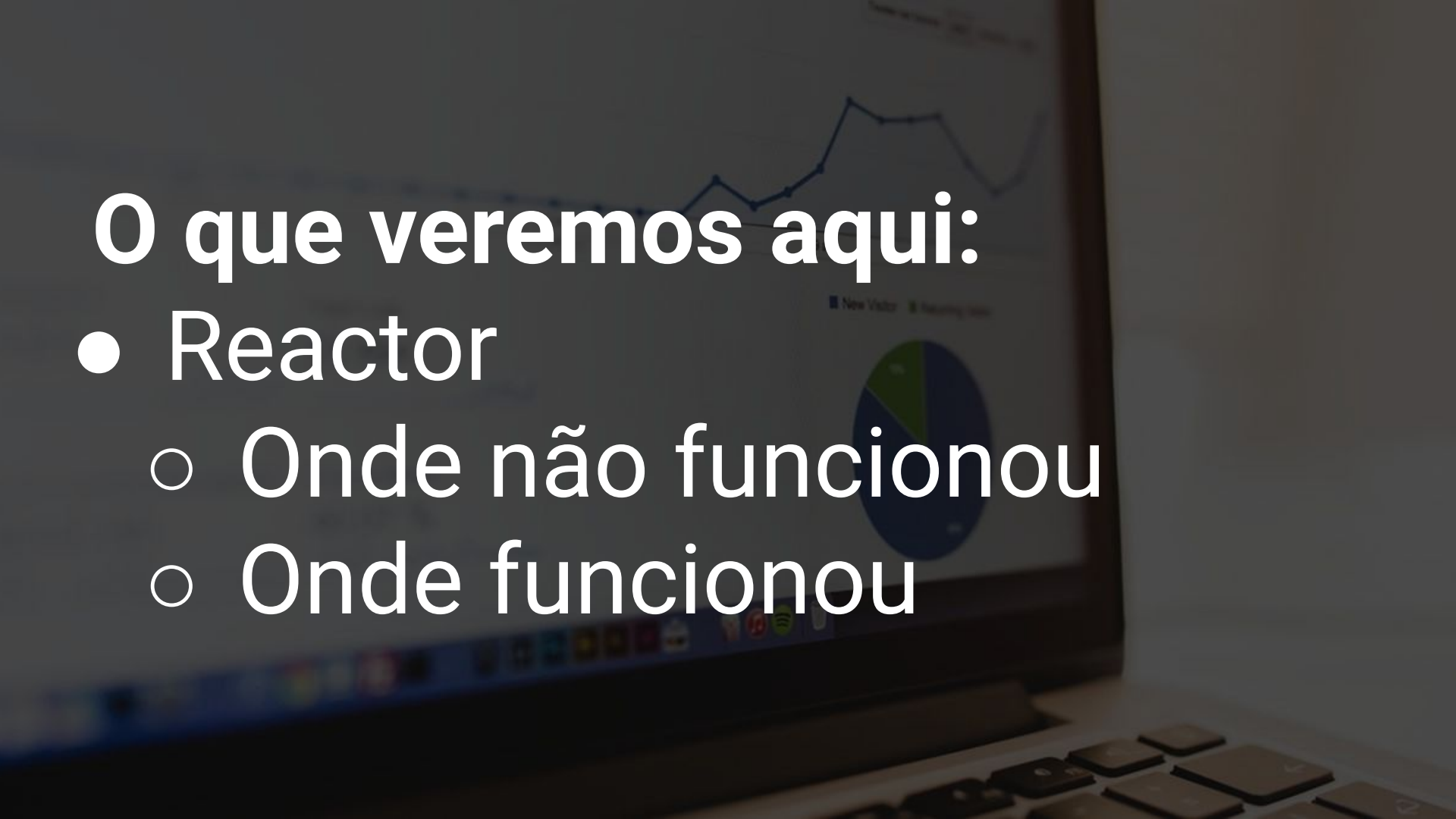
Java non-blocking code - Como não usar

Camila Balboni

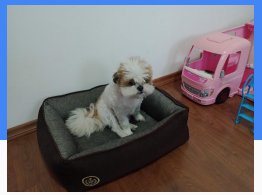


O que veremos aqui:

- Reactor
  - Onde não funcionou
  - Onde funcionou

Oi, eu sou a Camila

Programando desde quando Javascript não era linguagem de programação



Empresas

- Dell, 2021
- Fiap, 2022
- HCL, 2018
- Cognizant, 2010
- CPM Braxis, 2009
- Philips, 2007

Formação

- Jornalismo, 2003
- Engenharia Software (Pós), 2014
- Engenharia da Computação, 2021

Certificações

- SCJP (Java 5), 2009

Hobbies

- Crossfit
- Plantas
- Livros

Non Blocking code

```
var chocolateRequest = chocolateMapper.mapOrderchocolateTochocolate(orderchocolate);
var chocolateMono = candychocolateApiClientService.createchocolate(chocolateRequest);


return Mono.zip(chocolateMono, candyMono)
        .flatMap(objects -> {
            var chocolate = objects.getT1();
            var candy = objects.getT2();
(...)
```

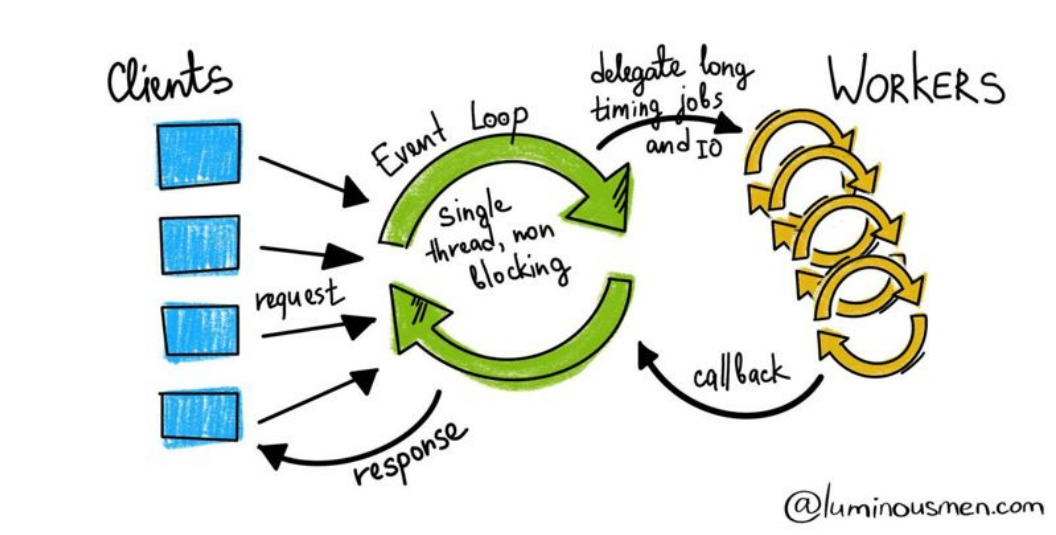
Project reactor

```
<dependency>
<groupId>io.projectreactor</groupId>
<artifactId>reactor-core</artifactId>
<version>3.4.16</version>
</dependency>
```

https://www.baeldung.com/reactor-core

https://www.udemy.com/course/reactive-programming-in-modern-java-using-project-reactor/?kw=java+reactor&src=sac

Uncle Bob - Arquitetura limpa, cap.6 (Programação funcional)



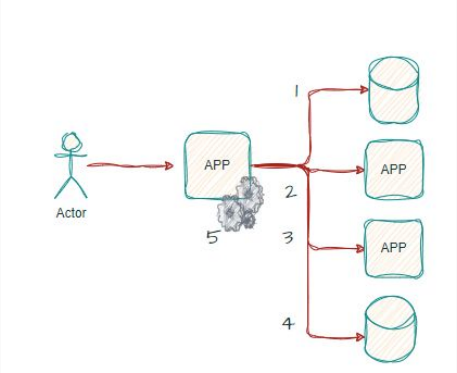
https://luminousmen.com/post/asynchronous-programming-blocking-and-non-blocking

O problema

Um endpoint precisa entregar dados de clientes, esses dados são compostos por Nomes, Endereços, Telefones e Emails;

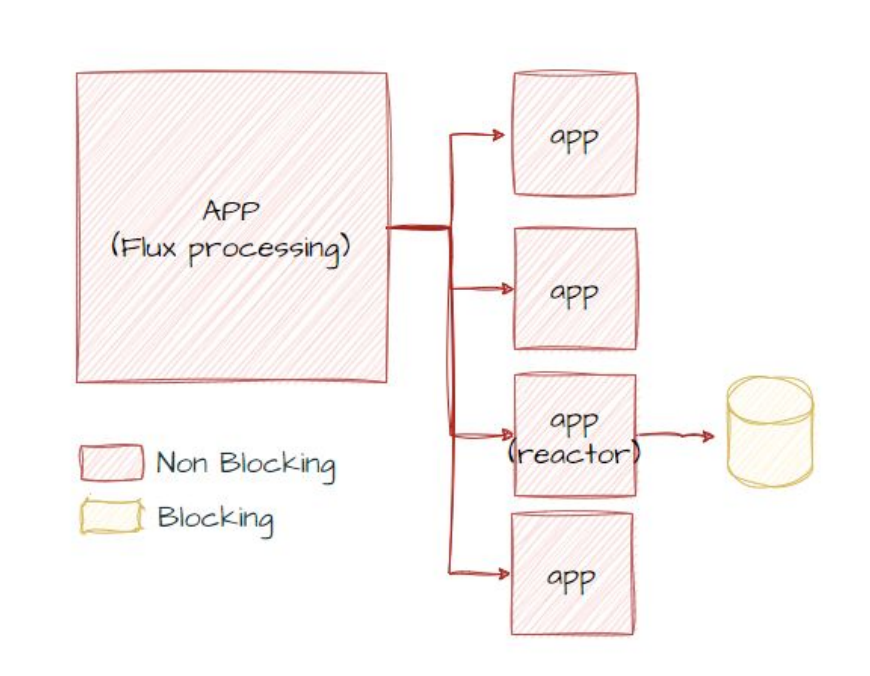
A lista principal vem do produto que o cliente comprou, esse produtos tem vários clientes diferentes (o que emitiu a nota, o que está com o produto em mãos e pode ser até a loja que vendeu)

Problema adicional: È preciso decidir a ordem dos endereços seguindo regras de negócio.



A solução

Paralelizar !



APP do Banco

Controller

```
    @ApiOperation(value = "Get Chocolate of a given Candy Bar.",
        authorizations = {@Authorization(value = "@Oauth2", scopes =
{})})
  @GetMapping(value = "/Chocolates", params = "CandyBar")
  @ApiResponses(value = {
        @ApiResponse(code = 401, message = "Full authentication is
required to get Chocolate"),
        @ApiResponse(code = 403, message = "Client not allowed to
get Chocolate"),
        @ApiResponse(code = 404, message = "Chocolate not found")
  })
  @PreAuthorize("hasAccess('" + BLABLABLA + "') ")
  public Mono<Page<Chocolate>> getChocolateByCandyBar(
        @ApiParam(name = "CandyBar", value = "Candy Bar", required =
true) @RequestParam("CandyBar") String CandyBar) {
      return ChocolateCandy.getChocolateByCandyBar(CandyBar);
  }
```

Service

```
  public Mono<Page<Chocolate>> findChocolatesByCandyKey(String
CandyNumber, String camAddressId, Long CandyBar, Pageable pageable) {
      var ChocolatesIds =
partyAddressChocolateRepository.findChocolatesIdByCandyNumberAndCandyBarA
ndCamAddressId(CandyNumber, camAddressId,
            CandyBar, pageable);

      return getChocolates(ChocolatesIds, pageable);
  }

private Mono<Page<Chocolate>> getChocolates(Page<Long> ChocolatesIds,
Pageable pageable) {
      return Flux.fromIterable(ChocolatesIds.getContent())
            .parallel()
            .runOn(Schedulers.boundedElastic())
            .flatMap(serviceService::getChocolateByIdIncludeAddresses)
            .sequential()
            .collect(Collectors.toList())
            .flatMap(Chocolates -> {
              Page<Chocolate> ChocolatesPage = new PageImpl<>(Chocolates,
                    pageable, ChocolatesIds.getTotalElements());

              return Mono.just(ChocolatesPage);
            });
  }
```

APP principal

```
public Mono<PagedRestResponse<ChocolatePartyAddress>>
        findChocolatesByChocolate(Long ChocolateId, Pageable
page) {
    return partyAddressApiWebClient
            .get()
            .uri(uriBuilder ->
uriBuilder.path(basePartyAddressChocolateUriV1)
                    .queryParam("ChocolateId", ChocolateId)
                    .queryParam("CandyId", CandyId)
                    .queryParam("partyAddressId", partyAddressId)
                    .queryParam("bar", bar)
                    .queryParam("page", page.getPageNumber())
                    .queryParam("size", page.getPageSize())
                    .build())
            .retrieve()
            .bodyToMono(new ParameterizedTypeReference<>() {
            });
}
```

```
private Mono<chocolateCandyBarResponseMin> validateChocolate(chocolateCandyBarMin
chocolateCandyBarDTO) {
    if (Arrays.stream(aeonCandies).noneMatch(purposeEnum ->
chocolateCandyBarDTO.getPurposeId().equals(purposeEnum.getId()))) {
        return Mono.error("ERROR");
    }
    return Mono.zip(chocolateApiClientService.getchocolateById(chocolateCandyBarDTO.getchocolateId()),

CandyBarAggregateClientService.getCandyBarByCandyBarId(chocolateCandyBarDTO.getCandyBarId()),

CandyBarApiClientService.findchocolateCandyBarBychocolateAndCandyBarAndPurpose(chocolateCandyBar
DTO.getchocolateId(),
                Optional.of(chocolateCandyBarDTO.getPurposeId()),
Optional.of(chocolateCandyBarDTO.getCandyBarId()),
                Optional.empty(), Pageable.ofSize(1)))
            .flatMap(objects -> {
                var chocolate = objects.getT1();
                var CandyBar = objects.getT2();
                var existingchocolateCandyBar = objects.getT3();

                if (CANDY_PARTY_OK.getId().equals(chocolateCandyBarDTO.getCandyId())) {
                    (...)
                }
                if (!CollectionUtils.isEmpty(existingchocolateCandyBar.getContent())) {
                    (...)
                }
                if (validateUcid(chocolate, CandyBar)) {
                    (...)
                }
                if (SourceSiteEnum.CANDY.getId().equals(chocolate.getCandyId())) {
                    (...)
                }
                return Mono.empty();
            });
}
```

Resultado

40 horas de sessão aberta no banco de dados !!!

Tirar Reactor do DB

Controller

```
    @ApiOperation(value = "Get Chocolate of a given Candy Bar.",
        authorizations = {@Authorization(value = "@Oauth2", scopes =
{})})
   @GetMapping(value = "/Chocolates", params = "CandyBar")
   @ApiResponses(value = {
        @ApiResponse(code = 401, message = "Full authentication is
required to get Chocolate"),
        @ApiResponse(code = 403, message = "Client not allowed to
get Chocolate"),
        @ApiResponse(code = 404, message = "Chocolate not found")
   })
   @PreAuthorize("hasAccess('" + BLABLABLA + "') ")
   public Page<Chocolate> getChocolateByCandyBar(
        @ApiParam(name = "CandyBar", value = "Candy Bar", required =
true) @RequestParam("CandyBar") String CandyBar) {
      return ChocolateCandy.getChocolateByCandyBar(CandyBar);
   }
```

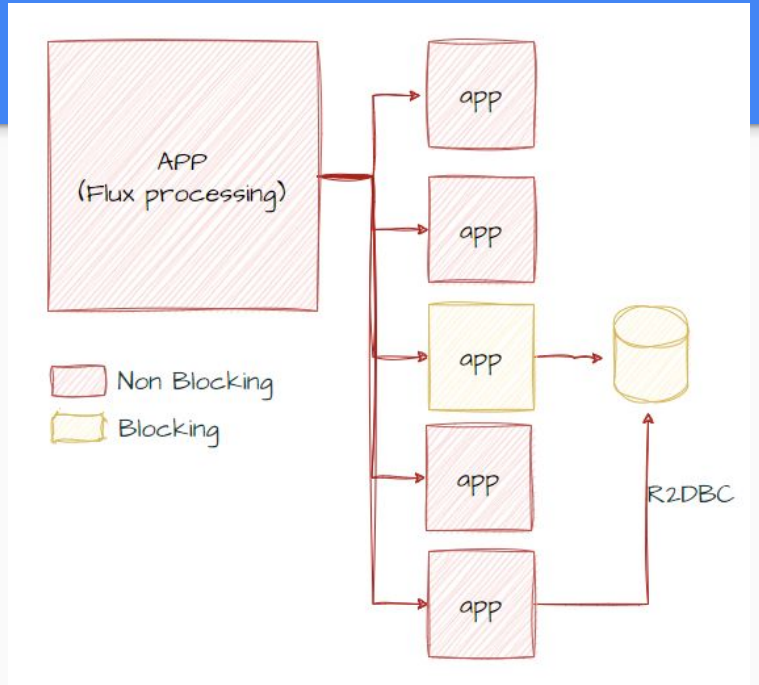
Service

```
   public Page<Chocolate> findChocolatesByCandyKey(String CandyNumber, String
camAddressId, Long CandyBar, Pageable pageable) {
      var ChocolatesIds =
partyAddressChocolateRepository.findChocolatesIdByCandyNumberAndCandyBarA
ndCamAddressId(CandyNumber, camAddressId,
            CandyBar, pageable);

      return getChocolates(ChocolatesIds, pageable);
   }

 @SneakyThrows
   private Page<Chocolate> getChocolate(Page<Long> ChocolateIds, Pageable
pageable) {
      var Chocolate = new ArrayList<Chocolate>();
      var futures = new ArrayList<CompletableFuture<Chocolate>>();
      ChocolateIds.getContent().forEach(ChocolateId ->

futures.add(clientService.getChocolateByIdIncludeCandyesAsync(ChocolateId)
                  .thenApply(Chocolate -> {
                     Chocolate.add(Chocolate);
                     return Chocolate;
                  })));
      CompletableFuture.allOf(futures.toArray(CompletableFuture[]::new)).join();
      return new PageImpl<>(Chocolate,
            pageable, ChocolateIds.getTotalElements());
   }
```

R2DBC

implementation("com.oracle.database.r2dbc:oracle-r2dbc:1.1.1")

```
public interface ReactiveQueryRepository<T extends AuditEntity, ID>  {

    Flux<T> findAll(Criteria criteria, Class<T> clazz, Pageable pageable);

    Flux<T> findAll(Criteria criteria, Class<T> clazz, boolean includeAudit, Pageable
pageable);

    Mono<T> findBy(Criteria criteria, Class<T> clazz, boolean includeAudit);

    Mono<Long> count(Criteria criteria, Class<T> clazz);

    Mono<T> findById(ID id, Class<T> clazz, boolean includeAudit);

}
```



Obrigada