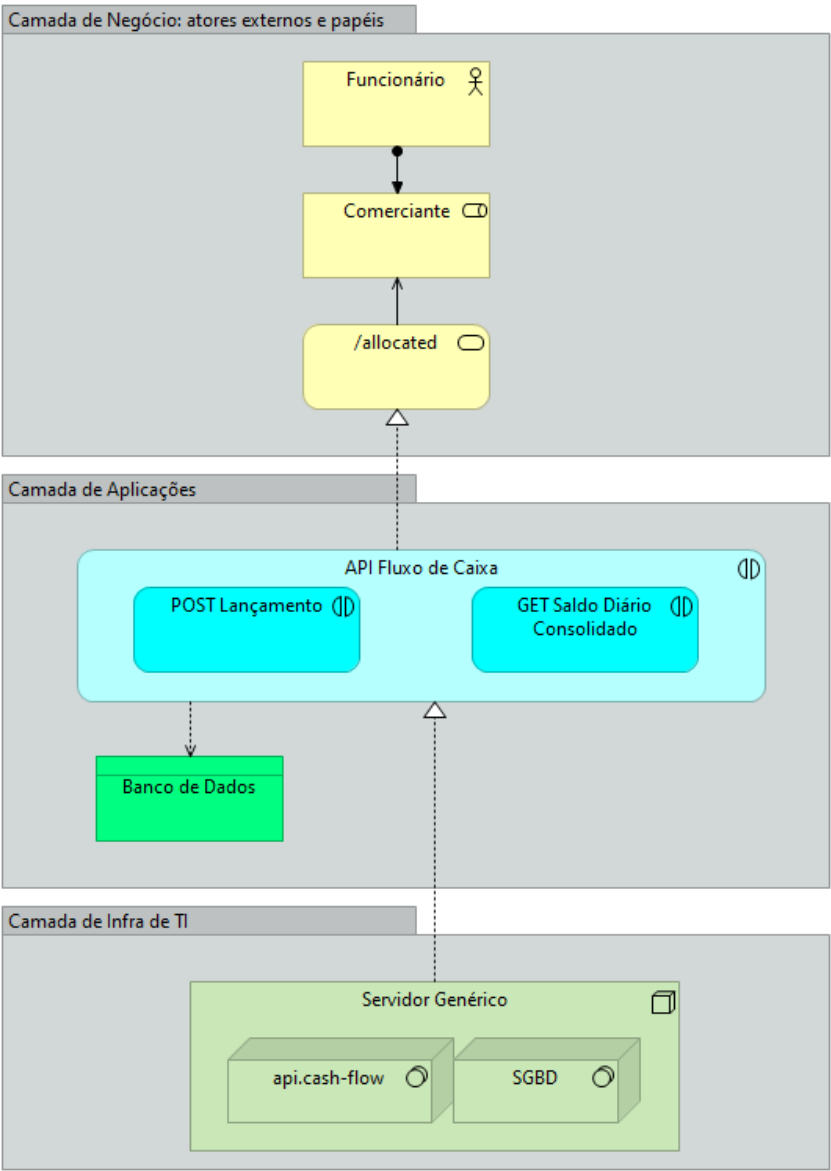


# Mttechne Challenge

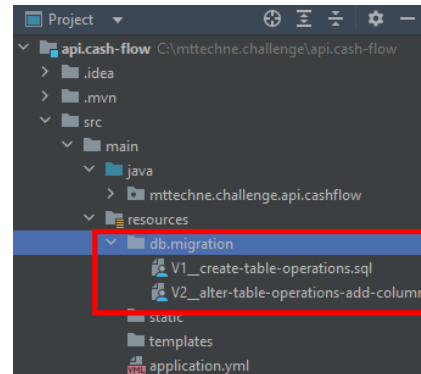
## Desenho da Solução



## Considerações e Controle de Versão da Base de Dados

1 – Formatação de queries e outras configurações de log foram deixadas propositalmente para um melhor entendimento da solução.

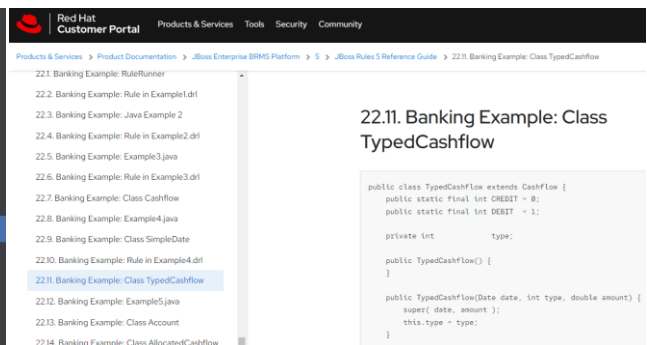
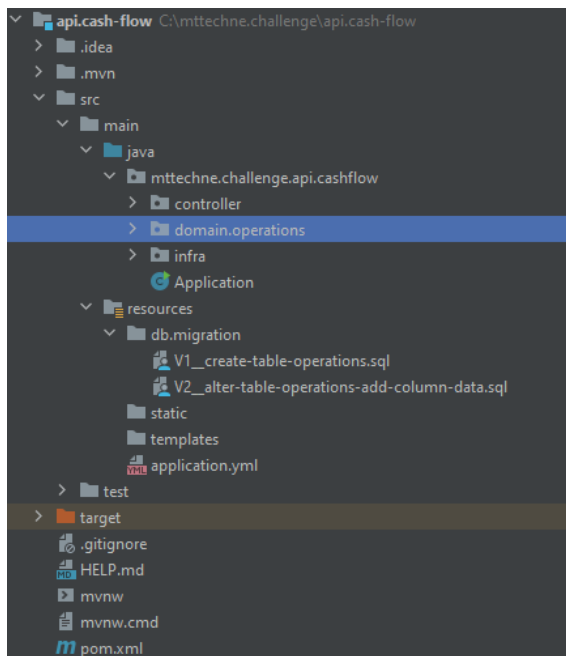
2 – No desafio criei a conjuntura onde a solução considera uma base relacional legada para exemplificar o uso do controle de versão das alterações e evoluções feitas no Banco de Dados aumentando a confiabilidade das implantações. Além de fornecer um mecanismo base para refatoração em microserviços. Isto é, não é uma solução em microserviços, mas uma estrutura alicerce neste sentido.



Tecnologia utilizada: Flyway (<https://flywaydb.org/>)

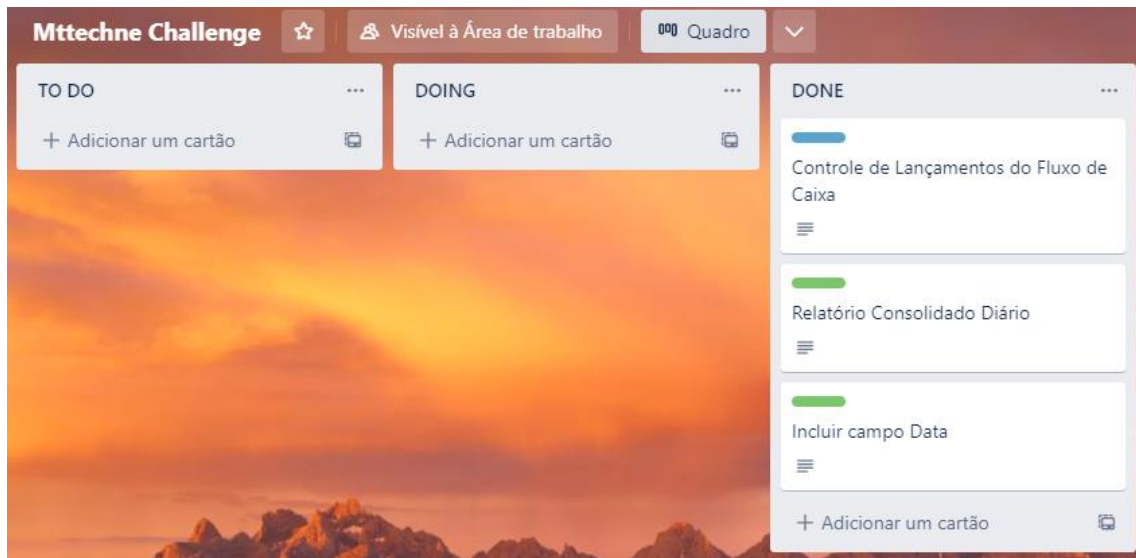
Referência para o domínio usado:

[https://access.redhat.com/documentation/en-us/jboss\\_enterprise\\_brms\\_platform/5/html/jboss\\_rules\\_5\\_reference\\_guide/banking\\_tutorial\\_class\\_allocatedcashflow](https://access.redhat.com/documentation/en-us/jboss_enterprise_brms_platform/5/html/jboss_rules_5_reference_guide/banking_tutorial_class_allocatedcashflow)



## Controle dos Requisitos:

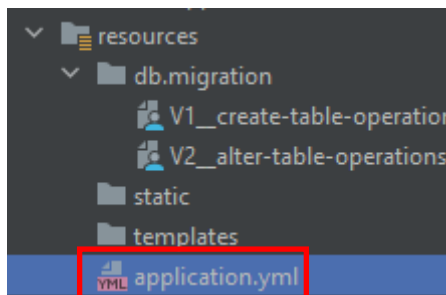
Para o controle dos requisitos foi utilizado a ferramenta <https://trello.com/>



<https://trello.com/invite/b/hFTIQvJk/ATTI810cf8f7cd34d0e6b71e6179f60a15aeC658EF53/mttechne-challenge>

## YAML ([12 Factor App](#))

Configuração mais legível com YAML, uma vez que não contém prefixos repetidos. Além de legibilidade reduz a repetição, o uso de YAML facilita o armazenamento de variáveis de configuração de ambiente, conforme recomenda o [12 Factor App](#), uma metodologia bastante conhecida e utilizada que define 12 boas práticas para criar uma aplicação moderna, escalável e de manutenção simples.



```
1 spring:
2   datasource:
3     driver-class-name: com.mysql.cj.jdbc.Driver
4     url: jdbc:mysql://localhost:3306/cashflow_api
5     username: root
6     password: root
7   jpa:
8     properties:
9       hibernate:
10        format_sql: true
11        dialect: org.hibernate.dialect.MySQL8Dialect
12    show-sql: true
```

## API RESTful

REST (Representational State Transfer). RESTful é um **estilo de arquitetura**, ou seja, uma série de restrições que devem ser seguidas no processo de criação de um serviço web. Para tornar a API RESTful, ela deve aderir a graus de maturidade.

<https://restfulapi.net/rest-architectural-constraints/>

201 é o código que representa que o registro foi criado. Regras:

- 1) Devolve no corpo da resposta os dados do novo registro criado
- 2) Devolve também um cabeçalho http (Location)

GET `http://localhost:8080/allocated`


200 OK 12.2 ms 433 B

Preview Headers Cookies Timeline

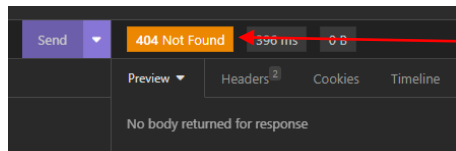
```
1 {
2   "account": 123,
3   "amount": 132.98,
4   "typed": "CREDIT",
5   "operationdate": "2023-03-14T20:53:22"
6 }
7 {
8   "account": 123,
9   "amount": 148.98,
10  "typed": "CREDIT",
11  "operationdate": "2023-03-14T20:53:29"
12 }
13 {
14   "account": 123,
15   "amount": 148.98,
16   "typed": "DEBIT",
17   "operationdate": "2023-03-14T20:53:45"
18 }
19 {
20   "account": 123,
21   "amount": 148.98,
22   "typed": "DEBIT",
23   "operationdate": "2023-03-14T20:53:46"
24 }
25 {
26   "account": 123,
27   "amount": 148.98,
28   "typed": "DEBIT",
29   "operationdate": "2023-03-14T20:53:49"
30 }
31 }
32 }
```

```
@GetMapping
@Transactional
public ResponseEntity<List<ConsolidatedDailyBalance>> dailyBalance()
{
    List<ConsolidatedDailyBalance> list = repository.findAll().stream().map(ConsolidatedDailyBalance::new).toList();
    return ResponseEntity.ok(list);
}
```

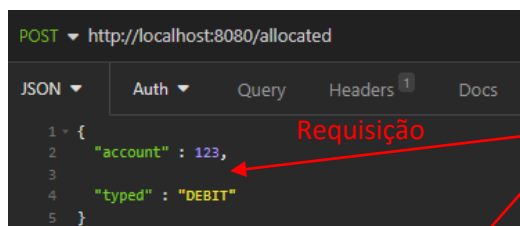
## Tratamento de Erros

GET  http://localhost:8080/allocated/6999

ID não existe

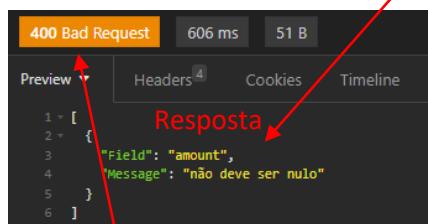


```
@RestControllerAdvice
public class ErrorHandler
{
    @ExceptionHandler(EntityNotFoundException.class)
    public ResponseEntity Error404()
    {
        return ResponseEntity.notFound().build();
    }
}
```



Campo obrigatório  
(amount)

Requisição



Resposta

```
@ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity Error400(MethodArgumentNotValidException ex)
{
    List<FieldError> error = ex.getFieldErrors();
    return ResponseEntity.badRequest().body(error.stream().map(ValidationErrorData::new).toList());
}

1 usage
private record ValidationErrorData(String Field, String Message)
{
    1 usage
    public ValidationErrorData(FieldError error)
    {
        this(error.getField(), error.getDefaultMessage());
    }
}
```

Dica para ajudar com os códigos http:

<https://http.cat/>

<https://http.dog/>