CHALLENGE 03

O projeto consiste no desenvolvimento de uma estrutura para um e-commerce, utilizando as tecnologias e conhecimentos aprendidos no curso.

Entrega

O desafio será em grupo e o código deverá ser disponibilizado no GitHub.

Entrega

Prazo para envio do e-mail com link do repositório: 09/02/2024 (sexta-feira), até às 18h.

Prazo para conclusão do Desafio: 09/02/2024 (sexta-feira), até as 18h

Apresentações: 14/02/2024 a 16/02/2024

Grupo de 3 pessoas.

Envio do e-mail

Crie um e-mail com o seguinte assunto:

- Challenge 3 - [nome da Equipe]

Exemplo de texto para o corpo do e-mail:

> Olá instrutores,
>
> Estamos enviando o link do desafio 3, como solicitado.
>
> Link do repositório: [link do repositório no GitHub]
>
> Equipe:
>
> -
>
> Nome dos membros:
>
> -

Envie o e-mail para os seguintes destinatários:

Para:

- franciele.ciostek@compasso.com.br

- diego.bonetti@compasso.com.br
- yago.lopes@compasso.com.br
- eduardo.borges@compasso.com.br
- gladson.ramos@compasso.com.br
- jean.lima@compasso.com.br

Instruções

1. Utilizar o Git e GitHub para versionamento.
2. Implemente o CRUD completo para cada domínio (produto), ou seja, as operações de criação, leitura, atualização e exclusão de dados.
3. Documente sua aplicação para facilitar o uso e a manutenção do código.
4. Configure um banco de dados da sua preferência para armazenar os dados do projeto.
5. Realizar Tratamento de Exceções.

Git e GitHub

No repositório do projeto, criado no GitHub.

- Os membros da equipe
- Deve criar uma branch para cada funcionalidade
- Deve manter o histórico de commits visível
- Deve ter uma branch "main"
- Deve ter pelo menos 3 commits em cada branch
- Deve ter pelo menos 1 pull/merge request de cada branch criada para a branch main

Requisitos

- **Tecnologias**: Java 11. Usar JDK11
- **Documentação**: O código deve ser documentado de forma clara e concisa.
- **Diagrama UML**: Criação do Diagrama UML que represente o site e-commerce do grupo.
- **Banco de dados**: O sistema deve utilizar um banco de dados MySQL ou outro que seja de seu conhecimento, configurado de forma segura e eficiente.
- **Boas práticas**: O código deve ser desenvolvido seguindo as boas práticas de desenvolvimento de software, para garantir a qualidade e a manutenibilidade.
- **Versionamento**: o código fonte deve estar no repositório do GitHub.

O que entregar?

Deve ser entregue a aplicação completa.

O que será considerado como completo:

- Implementação:
    - Cada funcionalidade possui 5 (cinco) opções no console.
    - A entrega deverá conter todos as opções do menu.
    - Inclui a arquitetura, boas práticas, configuração do banco de dados (application).
    - Tratamento de Erro
- Documentação:
    - Diagrama UML com as funcionalidades da implementação.

Funcionalidades

A atividade consiste no desenvolvimento de uma estrutura para um e-commerce com a funcionalidade de:

1. Produto: A funcionalidade Produto permite que os usuários criem, leiam, atualizem e excluam produtos.

Regras de Negócio - Geral:

- Todos os campos data, devem seguir o padrão ISO 8601 (exemplo: 2023-07-20T12:00:00Z).
- Todos os campos data, devem ser definidos automaticamente.

Produto

A funcionalidade Produto permite que os usuários criem, leiam, atualizem e excluam produtos. Deve ser criado os métodos que retornarão os dados conforme regra abaixo:

Operações:

Opção	Descrição
1	Lista produto: deve retornar todos os produtos.
2	Buscar produto: Retorna as informações de um produto específico.
3	Cadastrar pedido: Cria um novo produto.
4	Atualizar produto: atualiza as informações de um pedido existente.
5	Excluir produto: Excluir um produto existente.

Regras de negócio:

- O nome do produto deve ser único.
- A descrição do produto deve ter no mínimo 10 caracteres.
- O valor do produto deve ser um número positivo.

Em caso de listagem de todos os produtos:


```
{
   "id": 1,
   "name": "Product name",
   "description": "Product description",
   "value": 10.5
},
{
   "id": 2,
   "name": "Product name",
   "description": "Product description",
   "value": 10.5
},
{
   "id": 3,
   "name": "Product name",
   "description": "Product description",
   "value": 10.5
}
```


Após cadastro, consulta ou atualização de um produto retornar no console da seguinte forma:


```
{
   “id”: 1,
   "name": "Product name",
   "description": "Product description",
   "value": 10.5
}
```

Em caso de exclusão de produto

- **O produto foi excluído com sucesso.**

Fluxo de Erro

Erro de Validação (deverá ser exibido uma mensagem simulando retorno de uma api- pode ser no console)


```
{
  "code": 400,
  "status": "Bad Request",
  "message": "O campo 'nome' é obrigatório.",
  "details": [
    {
      "field": "nome",
      "message": "O campo 'nome' é obrigatório."
    }
  ]
}
```


Erro de Negócio


```
{
  "code": 400,
  "status": "Bad Request",
  "message": "O pedido já foi cancelado.",
  "details": []
}
```


Erro Inesperado


```
{
  "code": 500,
  "status": "Internal Server Error",
  "message": "Ocorreu um erro inesperado.",
```

```
  "details": []
}
```

Regras de Negócio para a resposta de exceção:

- A resposta de exceção deve conter as seguintes informações:
  - code : Código de simulação (200,400,500 - status HTTP)
  - message : Mensagem de erro
- O campo details é opcional e pode ser usado para fornecer informações adicionais sobre o erro.

As tabelas devem ter as seguintes colunas:

Tabela products

- id : chave primária, auto-incremento
- name : nome do produto
- value : preço/valor do produto
- description : descrição do produto