|  |  |
| --- | --- |
|  | **JAVA** |
| **ALUNO:** | **TURMA:** |
| **PROFESSOR:** Benefrancis do Nascimento | |
| **CHECKPOINT III** | **VALE 10 PONTOS** |

A Holding Benezinho Edifício comercial entrou no ramo de logística, está fornecendo essa atividade para várias empresas.

Contratamos você como Engenheiro de Software para ajudar a construir um sistema capaz de gerenciar o estoque, bem como o carrinho de vendas de produtos para diversas empresas que contratam o serviço de logística do nosso grupo empresarial.

Na sprint atual, você foi incumbido de fazer:

1. O Mapeamento Objeto Relacional das primeiras classes envolvidas neste projeto de software;
2. A criação automatizada das tabelas no banco de dados Oracle;
3. A persistência de todos os dados, e;
4. A criação de dois métodos capazes de realizar consultas as Vendas persistidas previamente:
   1. Consulta todos os vendas findAll;
   2. Consulta venda pela chave primária findById.

Veja o Diagrama de Classes abaixo (as classes já foram criadas):

|  |
| --- |
|  |

**Você deverá:**

**Faça o fork do projeto do github :**

https://github.com/Benefrancis/ benezinho-mao-de-vaca.git  
  
Caso o github esteja indisponível, você deverá pegar o projeto no diretório compartilhado.

1. **(0,5 Ponto)** acessar o arquivo persistence.xml e alterar as configurações da **persistence-unit** para que seja possível conectar-se ao banco de dados Oracle da FIAP com o seu usuário e senha (manter o seu usuário e senha ativo é sua responsabilidade). Não utilize o usuário e senha de outro aluno. Caso tenha problema para autenticar, comunique o professor.
2. **(1,5 Pontos)** adicionar corretamente as anotações JPA na classe **Carrinho**.

Lembre-se que:

1. Existe relacionamento **Muitos para Muitos** entre **Carrinho** e Produto no atributo produtos.
2. **(1 Ponto)** adicionar corretamente as anotações JPA na classe **Cliente**.

Lembre-se que:

1. Existe relacionamento **Muitos para Um** entre **Cliente** e **Pessoa** a no atributo pessoa.
2. **(2 Pontos)** adicionar corretamente as anotações JPA na classe **Produto**.

Lembre-se que:

1. Existe relacionamento **Muitos para Um** entre Produto e Fornecedor no atributo fornecedor.
2. A classe Seguro possui duas classes herdeiras:
   1. ProdutoPerecivel;
   2. ProdutoNaoPerecivel.
3. **(0,5 Ponto)** adicionar corretamente as anotações JPA na classe **Pessoa**.

Lembre-se que:

1. A classe Seguro possui duas classes herdeiras:
   1. Pessoa Física;
   2. Pessoa Jurídica.
2. **(1 Ponto)** adicionar corretamente as anotações JPA na classe **PessoaFisica**.

Lembre-se que:

* 1. Deverá adicionar uma ***constraint*** para que não seja possível ter mais de uma Pessoa Física com o mesmo número de **CPF** na tabela do banco de dados;
  2. Existe relacionamento **Muitos para Muitos** entre **PessoaFisica** e **PessoaFisica** no atributo filhos.

1. **(1 Ponto)** adicionar corretamente as anotações JPA na classe **PessoaJuridica**.

Lembre-se que:

* 1. Deverá adicionar uma ***constraint*** para que não seja possível ter mais de uma Pessoa Jurídica com o mesmo número de **CNPJ** na tabela do banco de dados;
  2. Existe relacionamento **Muitos para Muitos** entre **PessoaJuridica** e **Pessoa** no atributo socios.

1. **(1,5 Pontos)** adicionar corretamente as anotações JPA na classe **Venda**.

Lembre-se que:

* 1. Existe relacionamento **Muitos para Muitos** entre **Venda** e **Carrinho** no atributo responsáveis.

1. **(0,5 Ponto)** criar um método capaz de consultar uma **Venda** pelo seu identificador na correspondente tabela no banco de dados;
2. **(0,5 Ponto)** criar um método capaz de consultar todos as Vendas na correspondente tabela no banco de dados;

A avaliação é individual.

Boa avaliação.