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

A Holding Benezinho Edifício comercial expandiu muito o seu portifólio de sistemas gerenciados, para prover maior segurança ao acesso dos diversos sistemas da Holding, os arquitetos de software propuseram o desenvolvendo um sistema para gerenciamento de permissões dos sistemas empresariais da Holding Benezinho.

Contratamos você como Engenheiro de Software para ajudar a construir esse sistema.

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 aos Usuários persistidos previamente:
   1. Consulta todos os usuários findAll;
   2. Consulta usuário pela chave primária findById.

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

|  |
| --- |
|  |

**Importe o projeto do github :**

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

**Você deverá:**

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 **Profile**.

Lembre-se que:

1. Deverá adicionar uma ***constraint*** para que não seja possível ter mais de um Profile com o mesmo **nome** na tabela do banco de dados;
2. Existe relacionamento **Muitos para Muitos** entre **Profile** e **Role** no atributo roles.
3. **(1 Ponto)** adicionar corretamente as anotações JPA na classe **Role**.

Lembre-se que:

1. Deverá adicionar duas ***constraint*** para que não seja possível ter mais de uma Role com o mesmo **nome** na tabela do banco de dados;
2. Existe relacionamento **Muitos para Um** entre **Role** e **Sistema** a no atributo sistema.
3. **(2 Pontos)** adicionar corretamente as anotações JPA na classe **User**.

Lembre-se que:

1. Deverá adicionar uma *constraint* para que não seja possível ter mais de um **User** com o mesmo endereço de **email** na tabela do banco de dados;
2. Existe relacionamento **Muitos para Um** entre User e Pessoa no atributo pessoa.
3. Existe relacionamento Muitos para Muitos entre **User** e **Profile** no atributo profiles.
4. **(0,5 Ponto)** adicionar corretamente as anotações JPA na classe **Pessoa**.

Lembre-se que:

1. A classe Seguro possui três 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 **Sistema**.

Lembre-se que:

* 1. Deverá adicionar uma ***constraint*** para que não seja possível ter mais de um Sistema com a mesma **Sigla** na tabela do banco de dados;
  2. Existe relacionamento **Muitos para Muitos** entre **Sistema** e **Pessoa** no atributo responsáveis.

1. **(0,5 Ponto)** criar um método capaz de consultar um **Usuario** pelo seu identificador na correspondente tabela no banco de dados;
2. **(0,5 Ponto)** criar um método capaz de consultar todos os **Usuarios** na correspondente tabela no banco de dados;

A avaliação é individual e sem consulta.

Boa avaliação.