Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

20 - ATIVIDADE AVALIATIVA

1. Banco de Dados

1.1. Inicie o ***WampServer.***

1.2. Acesse o **MySQL Workbench** e execute o *script* abaixo:

```
CREATE DATABASE PC2_AVALIACAO_1B;

USE PC2_AVALIACAO_1B;

CREATE TABLE TBL_PROFESSOR(

ID_PROFESSOR BIGINT PRIMARY KEY AUTO_INCREMENT,

TX_NOME VARCHAR(50) NOT NULL,

NR_MATRICULA BIGINT NOT NULL

);

CREATE TABLE TBL_DISCIPLINA(

ID_DISCIPLINA BIGINT PRIMARY KEY AUTO_INCREMENT,

TX_NOME VARCHAR(50) NOT NULL,

NR_AULAS_SEMANAIS DOUBLE NOT NULL,

DT_CRIACAO DATETIME NOT NULL
);

CREATE TABLE TBL_REL_DISCIPLINA_PROFESSOR(

ID_DISCIPLINA BIGINT NOT NULL,

ID_PROFESSOR BIGINT NOT NULL,

FOREIGN KEY (ID_DISCIPLINA) REFERENCES TBL_DISCIPLINA (ID_DISCIPLINA),

FOREIGN KEY (ID_PROFESSOR) REFERENCES TBL_PROFESSOR (ID_PROFESSOR),

PRIMARY KEY (ID_DISCIPLINA, ID_PROFESSOR)

);

INSERT INTO TBL_PROFESSOR (TX_NOME, NR_MATRICULA) VALUES ('Lucia Satiko', 12367);

INSERT INTO TBL_PROFESSOR (TX_NOME, NR_MATRICULA) VALUES ('Valter Costa Junior', 12345);

INSERT INTO TBL_PROFESSOR (TX_NOME, NR_MATRICULA) VALUES ('Eliane Marion', 12356);

INSERT INTO TBL_PROFESSOR (TX_NOME, NR_MATRICULA) VALUES ('Luiz Souza', 12378);

INSERT INTO TBL_PROFESSOR (TX_NOME, NR_MATRICULA) VALUES ('Marcos Bellotti', 12390);
```


© copyright ETEC Profº Horácio Augusto da Silveira

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

2. Preparação do Ambiente

2.1. Integre o *Lombok* ao Eclipse.

2.2. Abra o *Eclipse* e defina a *workspace* em **C:/PCII-Avaliacao.**

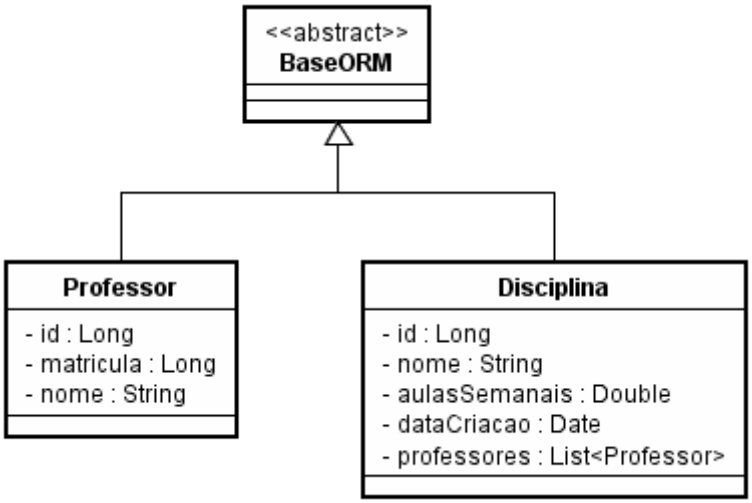
2.3. Configure o *Maven* para apontar para o arquivo ***settings.xml*** de **D:\maven\.m2**.

2.4. Copie o projeto **pc2-aval1bim** da pasta **[rede]****Professor****PCII****projetos** para **D:\pc2\avaliacao**.

2.5. Importe o projeto *Maven* **pc2-aval1bim** para o *Eclipse*.

3. Modelagem

3.1. Crie as classes abaixo dentro do pacote **model**.



3.2. As classes devem herdar de `BaseORM` e conter os mapeamentos *JPA*.

3.3. O atributo `professores` deverá ser anotado com `@ManyToMany`.

```
@ManyToMany
@JoinTable(name = "TBL_REL_DISCIPLINA_PROFESSOR",
           joinColumns = { @JoinColumn(name = "ID_DISCIPLINA") },
           inverseJoinColumns = { @JoinColumn(name = "ID_PROFESSOR") })
private List<Professor> professores;
```

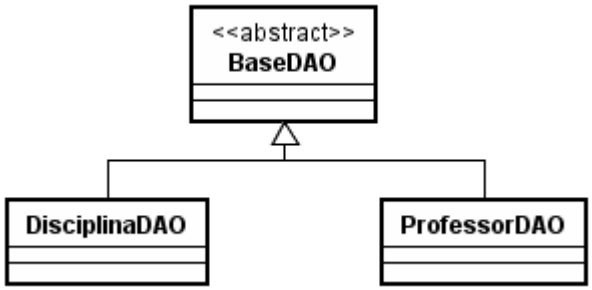

© copyright ETEC Profº Horácio Augusto da Silveira

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

3.4. A classe *Disciplina* terá o atributo `dataCriacao`, obrigatório na base de dados. Como é um campo que NÃO será preenchido em tela, resolva o problema com a anotação `@PrePersist`.

4. Data Access Object

4.1. Crie as classes abaixo dentro do pacote **dao**.

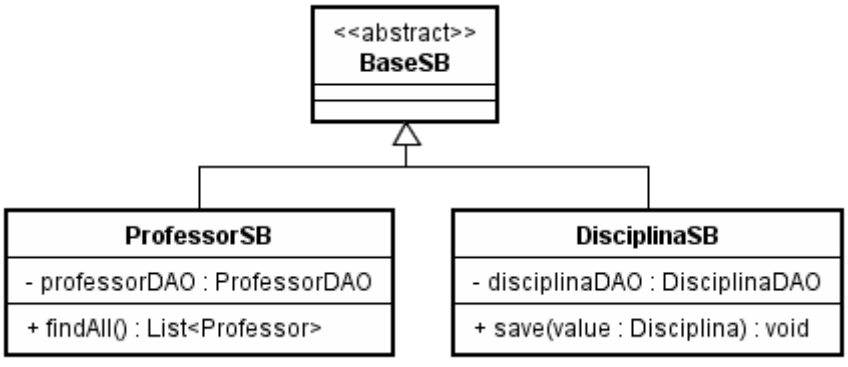


4.2. As classes devem herdar de `BaseDAO`.

4.3. Pressione CTRL + SHIFT + F para identar o código e salve a classe (CTRL + S).

5. Business Object

5.1. Crie as classes abaixo dentro do pacote **business**.



5.2. As classes devem herdar de `BaseSB`.

5.3. Pressione CTRL + SHIFT + F para identar o código e salve a classe (CTRL + S).

HIBERNATE

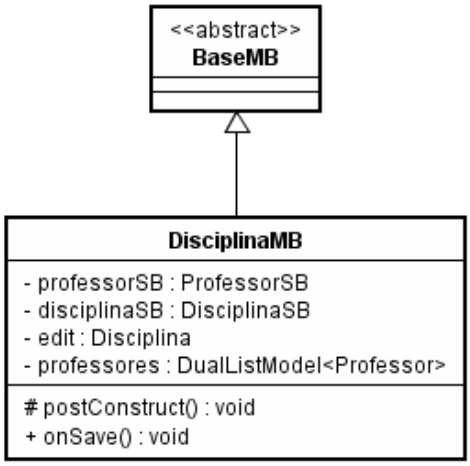
Centro Estadual de Educação Tecnológica Paula Souza - CEETEPS

© copyright ETEC Profº Horácio Augusto da Silveira

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

6. ManagedBean

6.1. Crie as classes abaixo dentro do pacote **view**.



6.2. A classe deve herdar de `BaseMB` e anotada conforme exemplo abaixo:

```
@Getter
@Setter
@Controller
@Scope("view")
public class AlunoMB extends BaseMB {


}
```

6.3. Pressione CTRL + SHIFT + F para identar o código e salve a classe (CTRL + S).

6.4. O método `postConstruct()` deverá armazenar os professores da base de dados no atributo `source` do `DualListModel`.

6.5. O método `onSave()` salvará a disciplina com seus professores e exibir a mensagem `"Registro inserido com sucesso."` através do método `showInsertMessage()`.

6.6. Verifique se existem mais de dois professores na listagem `target` do `DualListModel`. Caso verdadeiro, deverá ser exibida a mensagem de erro `"É permitido apenas dois professores por disciplina."`.

© copyright ETEC Profº Horácio Augusto da Silveira

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

7. JavaServer Faces

7.1. Abra a página **disciplina.xhtml** através da combinação Ctrl + Shift + R para localização.

7.2. Localize a linha **23** para ligar o `ManagedBean` com as páginas, definindo no atributo `value` a ligação do atributo `nome` do objeto `edit` da classe `DisciplinaMB`.

7.3. Repita a operação com a linha **30**.

7.4. Localize a linha **34** para configurar o carregamento do objeto `professores` no componente `<p:pickList>`. Defina, também, a variável *prof*:

```
<p:pickList value="#{disciplinaMB.professores}" var="prof"
        effect="blind" label="Professor(es)" style="background-color:#fff;height: 3px;"
        itemValue="" itemLabel=""
        converter="" required="true">
        <f:facet name="sourceCaption">Professor(es)</f:facet>
        <f:facet name="targetCaption">Professor(es) Selecionados</f:facet>
</p:pickList>
```

7.5. Na propriedade `itemValue` defina o valor a ser enviado ao *ManagedBean* que será o objeto *prof* que foi definido no atributo `var`. Já na propriedade `itemLabel` será exibido na tela os nomes dos professores:

```
<p:pickList value="#{disciplinaMB.professores}" var="prof"
        effect="blind" label="Professor(es)" style="background-color:#fff;height: 3px;"
        itemValue="#{prof}" itemLabel="#{prof.nome}"
        converter="" required="true">
        <f:facet name="sourceCaption">Professor(es)</f:facet>
        <f:facet name="targetCaption">Professor(es) Selecionados</f:facet>
</p:pickList>
```

7.6. O componente `<p:pickList>` precisa de um *conversor* para funcionar adequadamente. Foi desenvolvido um *conversor* chamado `entityConverter` que será usado na propriedade `converter`.

7.7. Faça a chamada do botão **Salvar** ao método `onSave()`.

7.8. Abra a classe **`AvaliacaoApplication`** através da combinação Ctrl + Shift + R para localização.

7.9. Para executar o código, clique com o botão direito do mouse na classe e selecione as opções ***Run As*** ➔ ***Java Application***.

7.10. Observe a inicialização do *framework* Spring e, ao final da inicialização, será exibida a informação de porta e contexto da aplicação.

7.11. Acesse o navegador de sua preferência com o endereço **localhost**, **porta** e **contexto** exibidos.

© copyright ETEC Profº Horácio Augusto da Silveira

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

8. Github

8.1. Acesse o github (**https://github.com/**) e entre na sua conta.

8.2. Crie um repositório chamado **pc2-aval1bim**.

8.3. Acesse a pasta do projeto (via **Prompt de Comando**).

8.4. Transforme o diretório **pc2-aval1bim** em um repositório do Git.

```
git init
```

8.5. Verifique a situação dos arquivos no repositório Git.

```
git status
```

8.6. Faça com que os arquivos sejam rastreados pelo Git.

```
git add .
```

8.7. Verifique a situação dos arquivos no repositório Git novamente.

8.8. Execute o **commit** para gravar as mudanças no repositório com a mensagem **"Avaliação 1. Bim de PCII"**.

8.9. Verifique a situação do arquivo no repositório Git novamente.

8.10. Execute o **push** para incluir seu código no repositório remoto.

8.11. Repita a operação para o **github** da outra pessoa.

© copyright ETEC Profº Horácio Augusto da Silveira