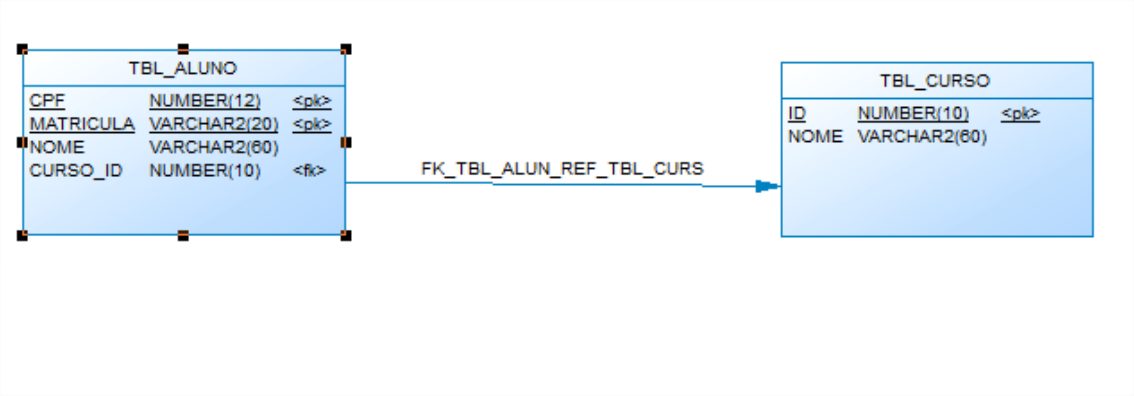
Tática Solutions SOLUÇÕES EM TI	**Avaliação – Analista de Sistemas Java**

Observações:

- Marcar o tempo gasto para conclusão das atividades;
- Não deixe os códigos com warnings;
- Comente seus métodos e classes;
- Utilize JPA;
- Utilize o framework web de preferência;
- Utilize maven;
- Utilize Oracle e PL/SQL;
- Siga o DER abaixo como modelo.
- Entregue apenas scripts e artefatos java, não é necessário lib's.

Boa prova!



1. **Crie um procedimento (PL/SQL) para inserir/atualizar dados na tabela TB_CURSO. Caso o código do curso não seja passado como parâmetro, o mesmo deve ser gerado automaticamente, retorne uma mensagem informando se o curso foi inserido/atualizado ou se ocorreu algum erro.**
2. **Crie um procedimento (PL/SQL) que receba como parâmetro o nome do curso e faça uma consulta, utilizando "like", a consulta deve ser inserida em cursor, o qual deve ser um parâmetro de saída.**
3. **Crie uma página web, que liste os alunos em um datatable (utilize lazy loading) e permita o usuário excluir/editar/inserir um registro, além disso, o usuário pode filtrar registros (utilize somente JPA).**
4. **Execute o exercício a seguir com base no arquivo em anexo (a-bb-rt-vrd-001-mx80.txt)**

   Encontre no arquivo o seguinte comando "show interfaces brief | no-more", a partir do mesmo determine as interfaces físicas, interfaces físicas são identificadas no seguinte texto "Physical interface: xe-0/0/0, Enabled, Physical link is Up", onde "xe-0/0/0" é a interface física. Com base na interface física, determine as interfaces lógicas da interface física (uma interface física pode ter N interfaces lógicas), as

interfaces lógicas são determinadas a partir do seguinte texto “Logical interface ge-1/0/4.452”, onde 452 é o nome da interface lógica.

Com base na descrição acima, crie um script que preencha as seguintes classes:

```
public class PhysicalInterface {

String description;
List<LogicalInterface> logicalInterfaces;

}

public class LogicalInterface {

      String description;
}
```

Imprima no console, cada interface física e suas interfaces lógicas conforme exemplo abaixo:

```
ge-1/0/4 - 10
ge-1/0/4 - 12
ge-1/0/4 – 452
xe-0/0/1 - 0
```

5. Execute o seguinte exercício:

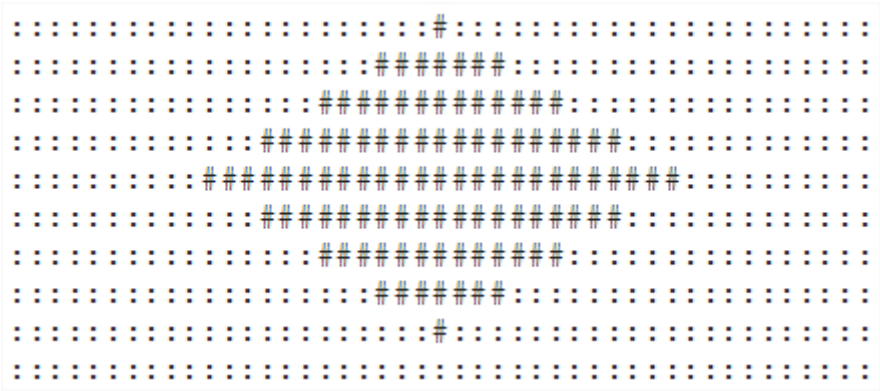
O algoritmo de Dijkstra, cujo nome se origina de seu inventor, o cientista da computação Edsger Dijkstra, soluciona o problema do caminho mais curto num grafo dirigido ou não dirigido com arestas de peso não negativo, em tempo computacional O([m+n]log n) onde m é o número de arestas e n é o número de vértices. O algoritmo que serve para resolver o mesmo problema em um grafo com pesos negativos é o algoritmo de Bellman-Ford, que possui maior tempo de execução que o Dijkstra.

O algoritmo de Dijkstra assemelha-se ao BFS, mas é um algoritmo guloso, ou seja, toma a decisão que parece ótima no momento. Para a teoria dos grafos uma "estratégia gulosa" é conveniente já que sendo P um menor caminho entre 2 vértices U e V, todo sub-caminho de P é um menor caminho entre 2 vértices pertencentes ao caminho P, desta forma construímos os melhores caminhos dos vértices alcançáveis pelo vértice inicial determinando todos os melhores caminhos intermediários. Nota: diz-se 'um menor caminho' pois caso existam 2 'menores caminhos' apenas um será descoberto.

O algoritmo considera um conjunto S de menores caminhos, iniciado com um vértice inicial I. A cada passo do algoritmo busca-se nas adjacências dos vértices pertencentes a S aquele vértice com menor distância relativa a I e adiciona-o a S e então repetindo os passos até que todos os vértices alcançáveis por I estejam em S. Arestas que ligam vértices já pertencentes a S são desconsideradas.

Com base no texto acima, escreva um programa que leia o número de nós, informe as ligações entre cada nó e o peso, ao final o programa deve receber o nó de origem e o nó de destino, e informar o menor caminho entre os nós. (Este código pode ser desenvolvido em Java, Javascript ou PLSQL)

6. Gere a figura abaixo formada pelos caracteres ": e #".



Regras:

- **Gere apenas um caractere de cada vez.**
- **Não use array ou vetores.**
- **Gerar no console.**
- **Utilizar estrutura de "loops" ou algoritmos recursivos**