INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

SUL-RIO-GRANDENSE - CAMPUS PASSO FUNDO

EMI – Programação III - 2022

Terceiro Trabalho individual

Prof. Telmo Júnior

Considere o estudo de caso "**Counter Strike Go – Single Player**" para o desenvolvimento de uma API na linguagem de programação Javascript/TypeScript. Cada Jogador possui um endereço, lista de artefatos e patentes. Além disso, possui uma lista de Compras. Cada compra possui uma lista de Itens. Cada Item referencia um Artefato e tem a informação da quantidade. Um artefato pode ser uma Arma ou uma Munição.

Arma é de um tipo (Branca ou Fogo) e tem uma lista de munições compatíveis. Cada Munição possui um determinado Calibre (C03, C05 ou C08). Uma Partida referência um Jogador e tem um lista de Round. Cada Round possui um Modo (TERRORISTA E CONTRATERRORISTA). Round possui uma lista de Objetivos. Objetivo tem uma lista de Locais. Mapa possui uma lista de Locais. Um Resultado referência um Objetivo e um Round, além da informação do Status (SIM ou Não)

No diagrama de classes ilustrado na Figura 1 temos as classes e relacionamentos para atender os pré-requisitos descritos anteriormente. Sendo assim, você deverá aplicar seus conhecimentos, para codificar um programa com os seguintes recursos/funcionalidades:

Observação: A entrega desse trabalho deverá ser realizada exclusivamente pelo Moodle (fontes, link do repositório e video). Não serão aceitos trabalhos enviados por e-mail.

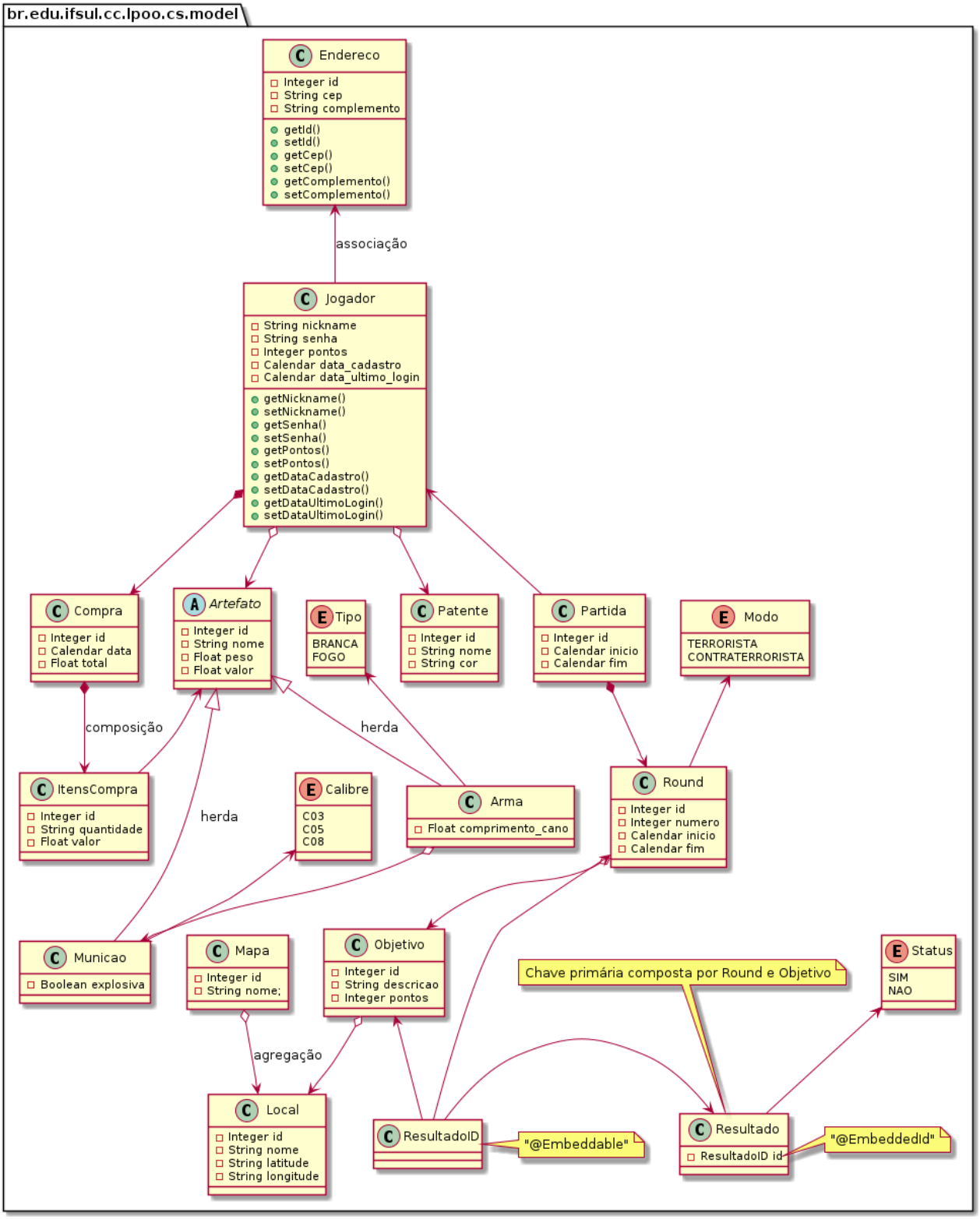

Figura 1 – Diagrama de Classes

1) Executar os procedimentos disponibilizados no Moodle da parte 1 até a 11, para a criação das tabelas, models, decorators, controllers, routes e test. O projeto deverá ser versionado no github.com em um repositório público. (2,0 ponto)

```
              List of relations
 Schema |        Name
--------+---------------------+
 public | tb_arma_municao
 public | tb_artefato
 public | tb_compra
 public | tb_endereco
 public | tb_jogador
 public | tb_jogador_artefato
 public | tb_jogador_patente
 public | tb_objetivo
 public | tb_partida
 public | tb_patente
 public | tb_resultado
 public | tb_round
 public | tb_itenscompra
 public | tb_mapa
 public | tb_local
 public | tb_mapa_local
 public | tb_objetivo_local
```

2) Criar o arquivo **Persistencia_Trabalho.test.ts** no diretório *src*/app/__test__ e implementar uma sequencia de testes de persistência envolvendo no mínimo duas classes/tabelas que tenham ligação entre si, conforme o Quadro 1. Os Procedimentos para a execução das operações CRUD devem ser os seguintes: (Peso: 2,0)

- Passo 1: selecionar todos os registros da tabela, bem como, as respectivas os dados da segunda tabela e demais tabelas.
- Passo 2: caso encontre, imprimir na tela todas as informações de todas as tabelas;

- Passo 3: após a impressão remover os dados das tabelas.
- Passo 4: caso não encontre no passo 2, inserir novos registros nas duas tabelas.

3) Defesa do trabalho: Apresentar o trabalho de forma presencial para o Professor, demonstrado a execução das funcionalidades CRUD previstas nesse documento. (2,0 ponto)

Distribuição

Aluno	**Classe 1**	**Classe 2**	**Classe 3 /Enum**
Luiz Eduardo e Gabriela	Partida	Round	Modo
Estevão, Augusto, Gustavo, Iuri e Luana	Mapa	Local	
Vainer, Gabrielly, Gabriel, Ivan, João Victor.	Compra	ItensCompra	Artefato
Kátya, Luisa, Marcelo, Paulo e, Renan Igor	Jogador	Patente	
Renan Gatto, Raduan, Samuel e Solano.	Objetivo	Local	

Quadro 1 – Distribuição das Classes por Aluno