Trabalho Prático

Fase 3 – Design da Solução

Docentes:

Leonor Melo

Inês Domingues

Turma Prática: P2

Trabalho realizado por:

Diogo Rafael Abrantes Oliveira – 2021146037 - a2021146037@isec.pt

Tomás Alexandre Dias Laranjeira – 2021135060 - a 2021135060@isec.pt

Gabriel Alexandre Nascimento Duarte – 2021114516 - a 2021114516@isec.pt

Índice

Índice de Figuras

Descrição Pormenorizada do Caso de Uso

Título: Inscrever na Competição.

Descrição: Um atleta inscreve-se na competição.

Sistema: Sistema da Taça do Mundo de Trampolins – STMT.

Ator Primário: Atleta.

Objetivo: STMT registar a inscrição do atleta.

Pós-Condição: É guardada no STMT a inscrição do atleta.

Casos de uso relacionados: Inscrever.

Fluxo de Eventos:

1. O STMT apresenta uma mensagem de boas-vindas.
2. Atleta seleciona a opção “Inscrição Atleta”.
3. Atleta introduz os dados pessoais.
4. Sistema mostra as disciplinas e treinadores disponíveis para seleção e atletas selecionam as respetivas opções.
5. Atleta conclui a tarefa de inscrição e mostra um resumo da inscrição.

Cenários alternativos:

3. a) STMT verifica que atleta já se encontra inscrito.

1. STMT indica que atleta com os dados introduzidos já se encontra inscrito.

b) Os dados introduzidos não são válidos

1. STMT indica que dados introduzidos não são válidos e volta para 3.

4. a) O treinador não se encontra inscrito.

1. Atleta pretende voltar a verificar os treinadores inscritos, continua e 4.

2. Utilizador seleciona a opção cancelar. Vai para 5.a)

b) A disciplina não se encontra disponível.

1. Atleta deve selecionar uma das disciplinas disponíveis, continua em 4.

2. Utilizador seleciona a opção cancelar. Vai para 5.a)

5. Atleta pretende não guardar a inscrição.

1. Atleta seleciona opção cancelar e é questionado se pretende mesmo cancelar:

a) Caso o utilizador confirme que não pretende guardar a inscrição, não devem ser guardados quaisquer dados e termina.

b) Caso o utilizador indique que não pretende cancelar, volta para 4.

Modelo do domínio

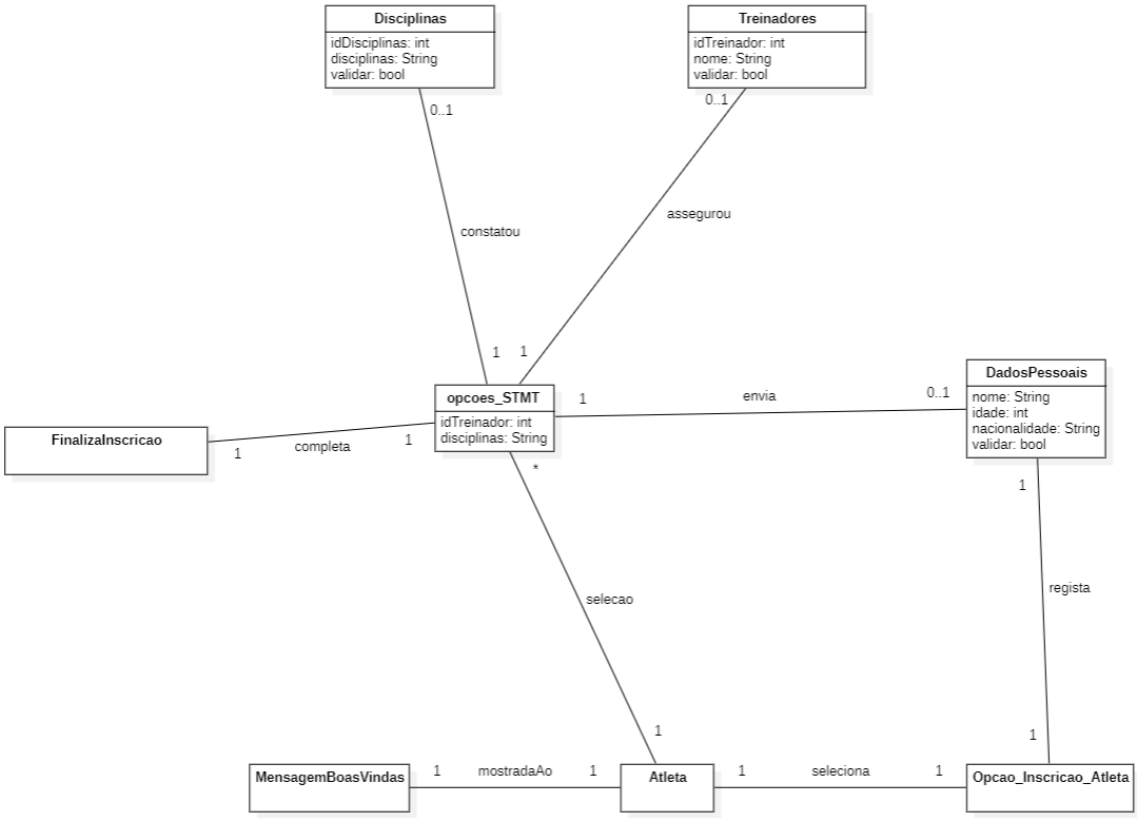

Figura 1 - Modelo do Domínio

Diagrama de Sequência do Sistema

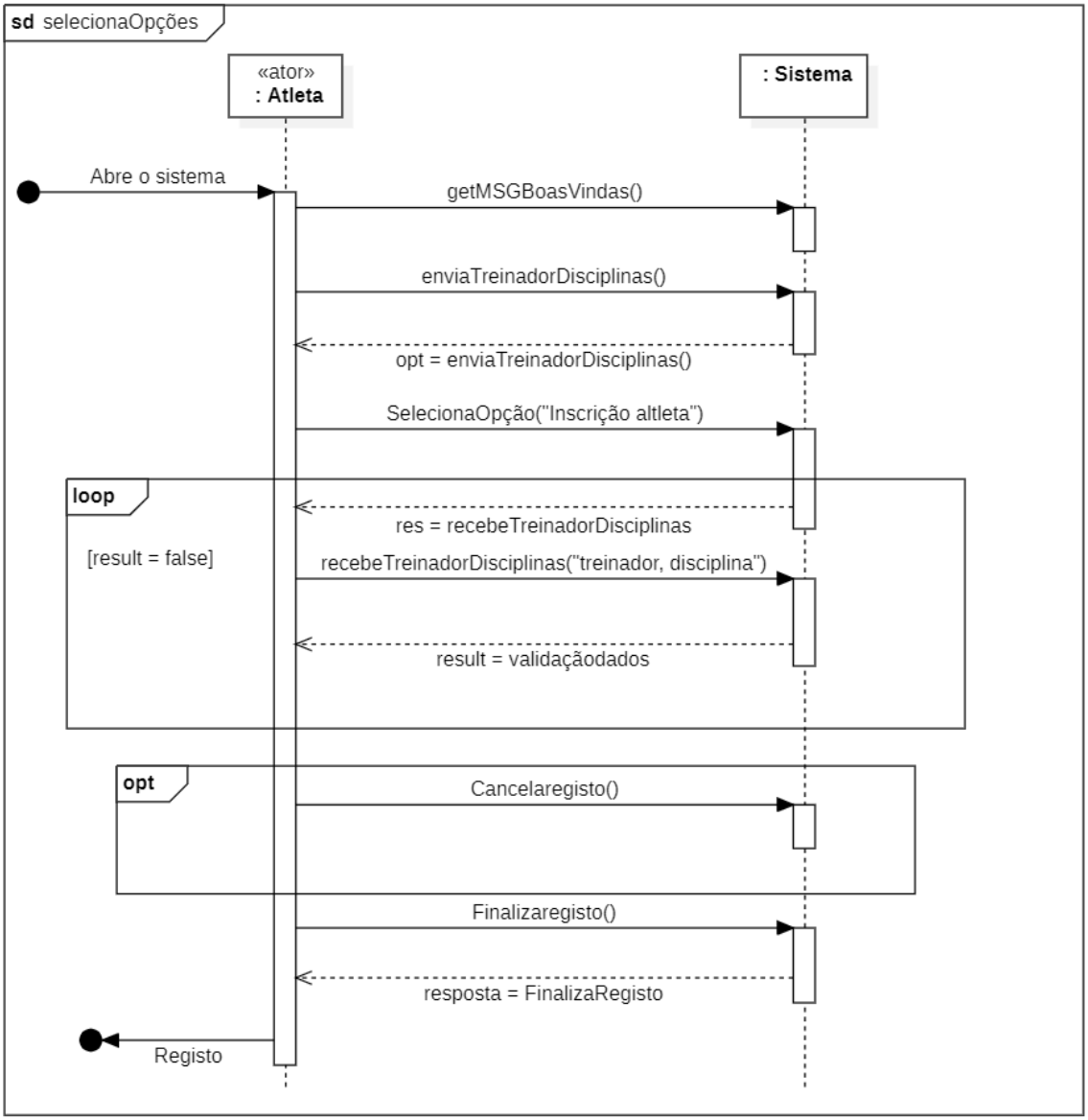

Figura 2 - Diagrama de Sequência do Sistema

Este Digrama de Sequência representa o ator **Atleta**, a efetuar a Inscrição no **Sistema**, inicialmente o **Atleta** recebe uma mensagem de Boas Vindas do Sistema, de seguida o Atleta recebe as informações dos Treinadores e das Disciplinas vindas do Sistema, seguidamente o **Atleta**, escolhe a opção de se inscrever, depois o resultado das Inscrição, passa por um "*loop"*, enquanto o resultado não foi positivo, se este for positivo os dados validados passam para o "*result"*, após a verificação o **Sistema** finaliza o registo, invocando o *"FinalizarRegisto()"* e o **Atleta** recebe uma resposta para verificar que a sua

inscrição foi bem sucedida, no entanto, em qualquer momento da Inscrição o **Atleta** pode cancelar o seu registo, invocando o *"Cancelaregisto()"*.

Glossário

- **Disciplinas** – contido no primeiro modelo, representa as várias variâncias da competição, como por exemplo, trampolim individual, trampolim sincronizado, duplo mini- trampolim e tumbling.
- **idDisiciplinas** – contido no primeiro modelo (figura 1), tem como função ser o número indentificador de cada disciplinas disponíveis na competição.
- **idTreinador** – contido no primeiro modelo (figura 1), tem como função ser o número indentificador de cada treinador disponível na competição.
- **MensagemBoasVindas** – contido no primeiro modelo (figura 1), é uma classe que representa uma Mensagem de Boas Vindas apresentada ao Atleta, no início do caso de uso.
- **enviaTreinadorDisciplinas()** – contido no segundo modelo (figura 2), é um método que o Atleta pede ao Sistema para mostrar as os Treinadores e as Disciplinas disponíveis para a inscrição.
- **recebeTreinadorDisciplinas()** - contido no segundo modelo (figura 2), é uma variável que guarda o as escolhas do Atleta no momento da inscrição.
- **result = validaçãodados** - contigo no segundo modelo (figura 2), é uma variável que esta a ser verificada no *"loop"*, com o propósito de validar se os dados do formulário estão bem preenchidos.
- **enviaNrInscrição()** - contido no terceiro modelo (figura 3), é um método para enviar o número de inscrição ao formulário.
- **nrInscrição** - contido no terceiro modelo (figura 3), é uma classe dependente da Inscrição.
- **inicaAtletaUI()** - contido no terceiro modelo (figura 3), é um método para iniciar a Interface, que o Atleta vai interagir.
- **GeraNumInscrição()** - contido no terceiro modelo (figura 3), é um método para gerar o número de inscrição de cada Atleta, depois de se inscreverem.
- **mostraEstadoInscricao()** - contido no quarto modelo (figura 3),é um método para mostrar o utilizador o estado do Formulário.
- **verificaDadosTreinador()** - contido no quarto modelo (figura 4), é um método booleano para verificar se os dados dos treinadores estão corretos.
- **verificaDadosDisciplina()** - contido no quarto modelo (figura 4), é um método booleano para verificar se os dados das disciplinas inseridos estão corretos.

Mensagens

Diagrama de classes

EnviaTreinadorDisciplinas("treinador, disciplina")

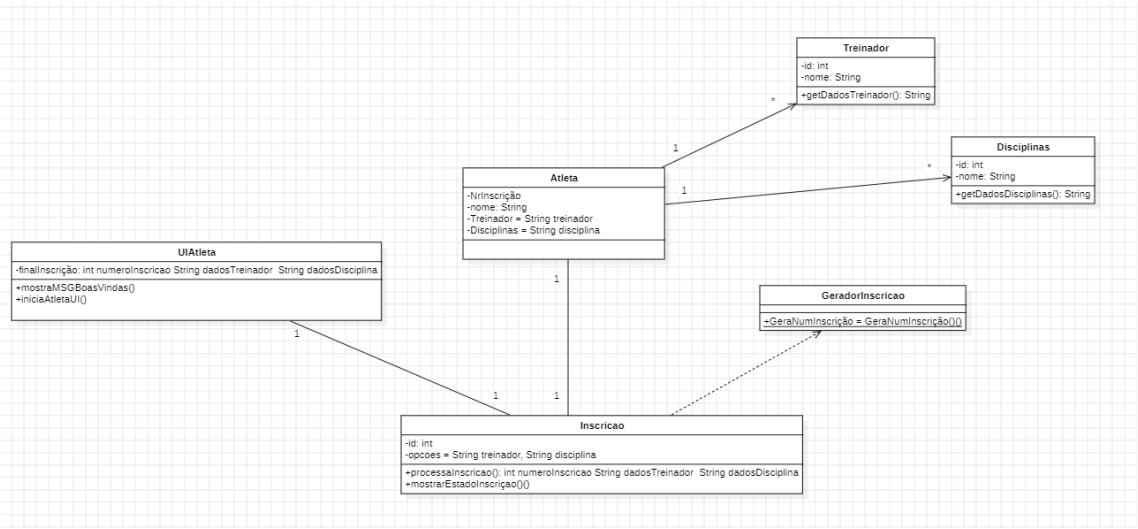

Figura 3 - Diagrama de classes *Envia*TreinadorDisciplinas("treinador, disciplina")

O diagrama de classes EnviaTreinadorDisciplinas do método EnviaTreinadorDisciplinas() contém seis classes:

A classe UIAtleta é invocada inicialmente com o método iniciaAtletaUI(), tendo um método próprio para mostrar uma mensagem de boas-vindas e invoca o método ProcessaInscrição() da Classe Inscrição que retorna para a classe UIAtleta as informações sobre os treinadores e disciplinas disponíveis.

A classe Inscrição invoca a função estática GeraNumInscrição() da classe GeradorInscrição (existindo aqui uma relação de dependência), e envia esse respetivo número como parâmetro na invocação do construtor da classe Atleta.

A classe atleta é responsável por invocar as funções que obtêm os dados dos Treinadores e Disciplinas através da invocação dos respetivos métodos(getDadosTreinador() e getDadosDisciplina) de forma a propagar a informação para as classes anteriores como resposta. Existe uma relação de Associação apenas no sentido classe Atleta para as classes Treinador e Disciplinas, isto é, apenas a classe Atleta conhece os métodos da classe Treinador e classe Disciplina.

recebeTreinadorDisciplinas("treinador, disciplina")

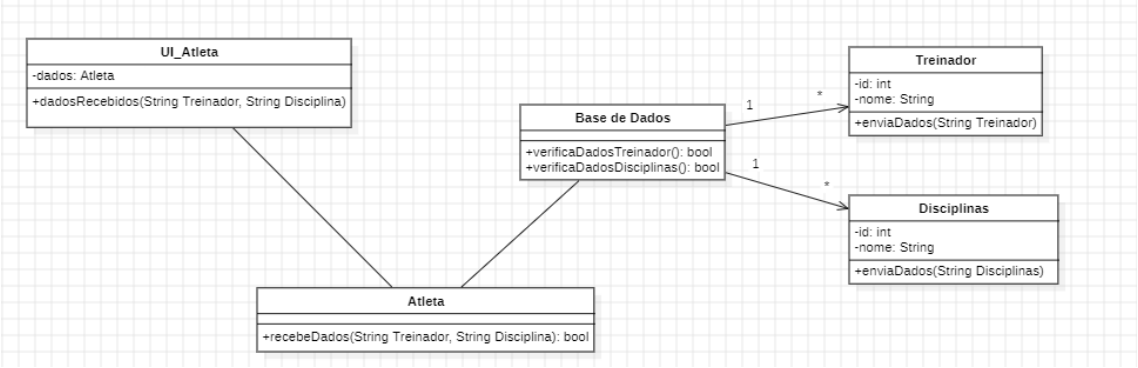

Figura 4 - Diagrama de classes recebeTreinadorDisciplinas("treinador, disciplina")

Este é o diagrama de classes recebeTreinadorDisciplinas do método recebeTreinadorDisciplinas("treinador, disciplina"). Ele é composto por uma classe **UI_Atleta** que é inicialmente invocada pelo método dadosRecebidos(String Treinador, String Disciplina)**,** tem um atributo dados do tipo Atleta.

A classe **Atleta** é invocada pelo método dadosRecebidos(in String Treinador, in String Disciplina).

A classe **Base de Dados** é composta por dois métodos, verificaDadostreinador() e verificaDadosDisciplina(), estes servem para verificar se os ambos os dados são admissiveis.

Por fim as classes **Treinador** e **Disciplinas** são compostas por dois atributos, atributo id do tipo int e atributo nome do tipo String, têm também um método EnviaDados(String Treinador) e EnviaDados(String Disciplinas), respetivamente.

Existe uma relação de Associação apenas no sentido classe **Base de Dados** para as classes **Treinador** e **Disciplinas**, isto é, apenas a classe Base de Dados conhece os métodos da classe Treinador e classe Disciplina.

Diagrama de sequência

EnviaTreinadorDisciplinas("treinador, disciplina")

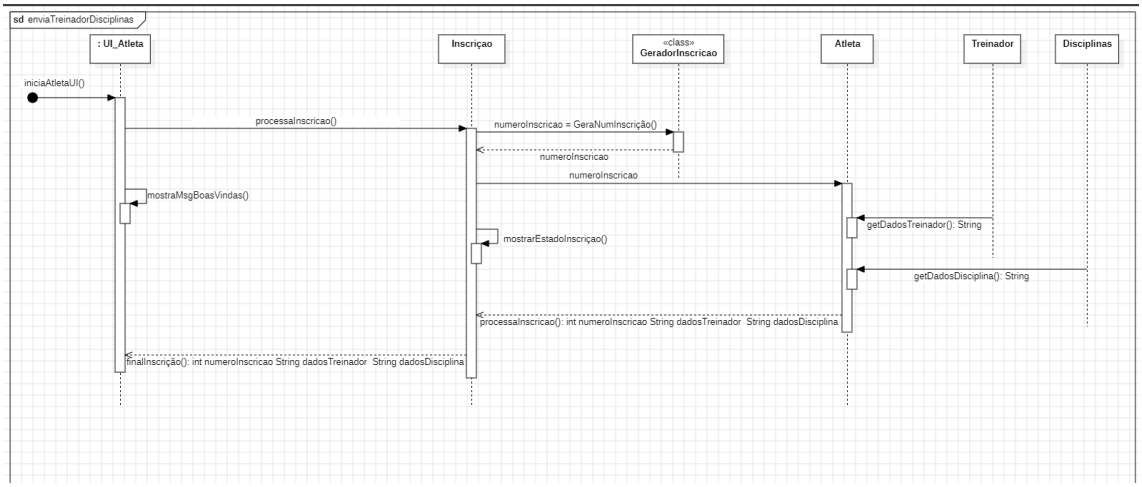

Figura 5 - Diagrama de Sequência EnviaTreinadorDisciplinas("treinador, disciplina")

recebeTreinadorDisciplinas(“treinador, disciplinas”)

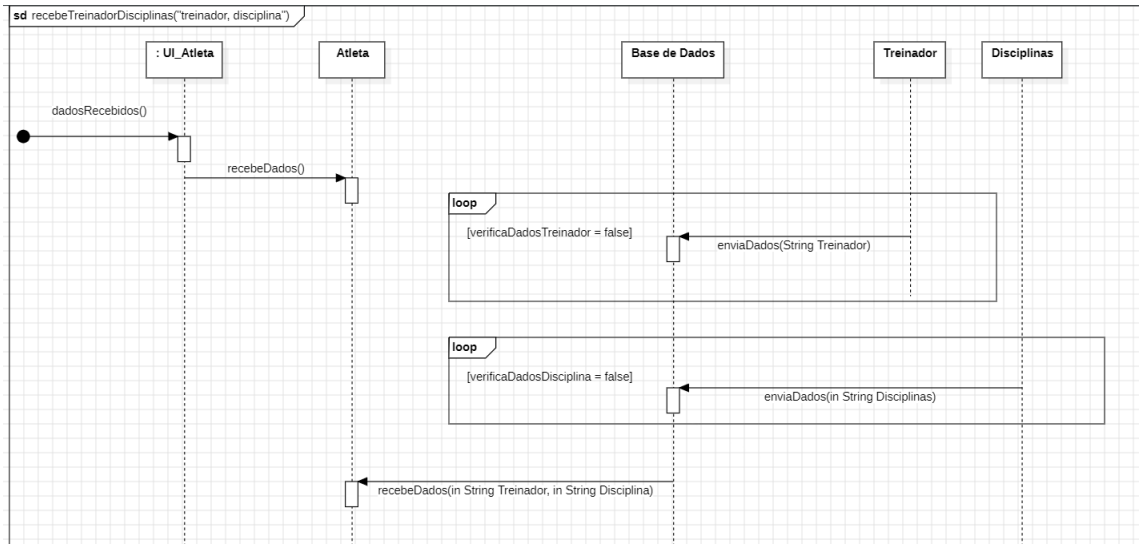

Figura 6 - Diagrama de Sequência recebeTreinadorDisciplinas(“treinador, disciplinas”)