Métodos Formais em Engenharia de Software

Mestrado Integrado em Engenharia Informática e Computação

PerfectGym

Relatório Final

Janeiro 2019

4MIEIC02

Afonso Ramos	|	up201506239
Cláudia Rodrigues	|	up201508262

Índice

Descrição Informal do Sistema e Lista de Requisitos **2**

Descrição Informal do Sistema 2

Lista de requisitos 2

Modelo UML **3**

Use Case Model 3

Descrição dos casos de uso 4

Class Model 8

Classes 8

Classes de Teste 8

Breve descrição das classes 9

Modelo Formal VDM++ **10**

Verificação do Modelo **10**

Exemplo de verificação de um domínio 10

Exemplo de verificação de uma invariante 11

Geração de código **12**

Conclusões **13**

Resultados obtidos 13

Possíveis melhoramentos 13

Contribuição 13

Referências **14**

1. Descrição Informal do Sistema e Lista de Requisitos

1.1. Descrição Informal do Sistema

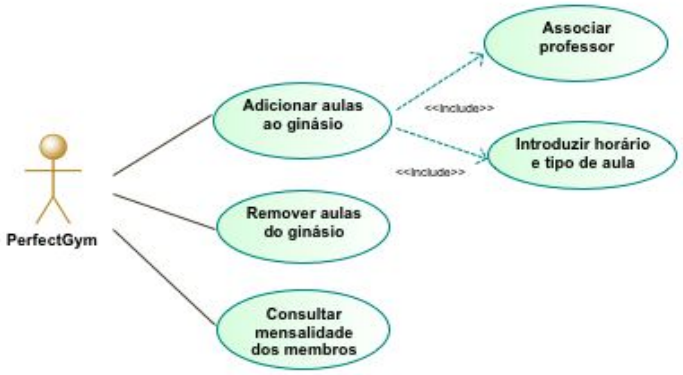
PerfectGym é uma aplicação de *fitness* destinada à organização de um ginásio, com perfil para os utilizadores que o frequentam, e para os professores. Desta forma, é possível que os membros se registem de forma a consultar o seu plano de treino individual, fornecido por um professor, ou inscrever em aulas coletivas. Estas encontram-se disponíveis de acordo com o mapa de aulas do ginásio. Por fim, é também adicionada uma perspetiva de gestão, nomeadamente a consulta das mensalidades dos membros.

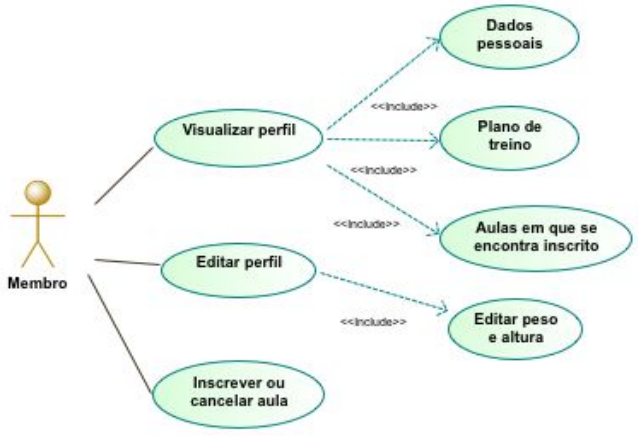
1.2. Lista de requisitos

ID	Prioridade	Descrição
R01	Obrigatória	Um utilizador pode criar conta, sendo atribuído um número de sócio único.
R02	Obrigatória	Um utilizador registado (Membro ou Treinador) pode fazer login e logout.
R03	Obrigatória	Um membro pode aceder e editar a sua informação pessoal, consultar o seu plano de treino e as aulas em que se encontra inscrito.
R04	Obrigatória	Um membro pode inscrever-se numa aula coletiva se não estiver sobrelotada ou cancelar a sua inscrição.
R05	Obrigatória	Um treinador pode associar um plano de treino a um membro, composto por uma sequência de exercícios.
R06	Obrigatória	Criar exercícios.
R07	Obrigatória	Adicionar ou remover exercícios de um plano de treino.
R08	Obrigatória	Consultar o mapa de aulas do ginásio.
R09	Obrigatória	Adicionar aulas ao ginásio se não existir sobreposição de horários ou remover aulas existentes.
R10	Opcional	Um utilizador pode criar conta, sendo referenciado por um membro já registado.
R11	Opcional	Consultar os horários de uma determinada aula, ou as aulas dadas por um determinado professor.
R12	Opcional	Consulta da mensalidade dos membros.

2. Modelo UML

2.1. Use Case Model





Descrição dos casos de uso

Cenário	**Registo de um utilizador**
Descrição	Um utilizador cria conta no sistema.
Pré-condições	1. Não existe nenhum utilizador registado com o mesmo número de sócio. 2. O email tem entre 5 e 50 caracteres. 3. O nome e password têm entre 1 e 20 caracteres. 4. O género é <*Masculine*> ou <*Feminine*>. No caso de utilizador registado ser um membro: 5. O peso e altura são valores positivos.
Pós-condições	1. O utilizador encontra-se registado no sistema com a informação introduzida.
Passos	O utilizador introduz as suas informações pessoais.
Exceções	(unspecified)

Cenário	**Iniciar sessão**
Descrição	Um utilizador inicia sessão.
Pré-condições	1. Não existe nenhum utilizador com sessão iniciada. 2. A password introduzida tem entre 1 e 20 caracteres.
Pós-condições	1. Se o número de sócio se encontrar registado no sistema e a password for correta, o utilizador tem sessão iniciada no sistema.
Passos	O utilizador introduz o seu número de sócio e password.
Exceções	(unspecified)

Cenário	**Terminar sessão**
Descrição	Um utilizador termina sessão.
Pré-condições	1. Existe um utilizador com sessão iniciada.
Pós-condições	1. Não existe nenhum utilizador com sessão iniciada.
Passos	(unspecified)
Exceções	(unspecified)

Cenário	**Visualizar perfil/ Plano de treino/ Aulas em que está inscrito**
Descrição	Um Membro visualiza a sua informação pessoal, consulta o seu plano de treino, ou as aulas em que se encontra inscrito.
Pré-condições	1. O membro tem sessão iniciada.
Pós-condições	(unspecified)
Passos	(unspecified)
Exceções	(unspecified)

Cenário	**Editar perfil**
Descrição	Um Membro edita a sua informação pessoal (peso ou altura).
Pré-condições	1. O membro tem sessão iniciada. 2. O peso ou altura são valores positivos.
Pós-condições	1. O peso ou altura é alterado para o valor introduzido.
Passos	(unspecified)
Exceções	(unspecified)

Cenário	**Inscrever membro em aula**
Descrição	Um Membro inscreve-se em uma aula.
Pré-condições	1. O membro tem sessão iniciada. 2. Não se encontrava já inscrito na aula. 3. A aula não se encontra sobrelotada.
Pós-condições	1. O membro encontra-se na lista de participantes da aula. 2. A aula tem menos um lugar disponível.
Passos	(unspecified)
Exceções	(unspecified)

Cenário	**Remover inscrição de membro em aula**
Descrição	Um Membro cancelar a sua inscrição em uma aula.
Pré-condições	1. O membro tem sessão iniciada. 2. Estava inscrito na aula.

Pós-condições	1. O membro não se encontra na lista de participantes. 2. A aula tem mais um lugar disponível.
Passos	(unspecified)
Exceções	(unspecified)

Cenário	**Criar um plano de treino para um membro**
Descrição	Um professor associa um plano de treino a um membro
Pré-condições	1. O professor tem sessão iniciada. 2. O membro encontra-se registado no sistema.
Pós-condições	1. O plano de treino do utilizador é alterado.
Passos	1. Criar exercícios com descrição, tipo, peso e repetições. 2. Associar ou remover exercícios do plano de treino, de forma a criar uma sequência de exercícios.
Exceções	(unspecified)

Cenário	**Consultar mapa de aulas**
Descrição	Um utilizador consulta as aulas existentes no ginásio, e as suas informações, nomeadamente o horário.
Pré-condições	(unspecified)
Pós-condições	(unspecified)
Passos	1. Se pretendido, é possível introduzir um tipo de aula, professor ou dia da semana, de forma a filtrar os resultados.
Exceções	(unspecified)

Cenário	**Adicionar aulas ao ginásio**
Descrição	É adicionada uma nova aula ao ginásio.
Pré-condições	1. A aula não se encontra já registada. 2. Não existe nenhuma aula no ginásio com o mesmo nome. 3. O professor associado á aula encontra-se registado.

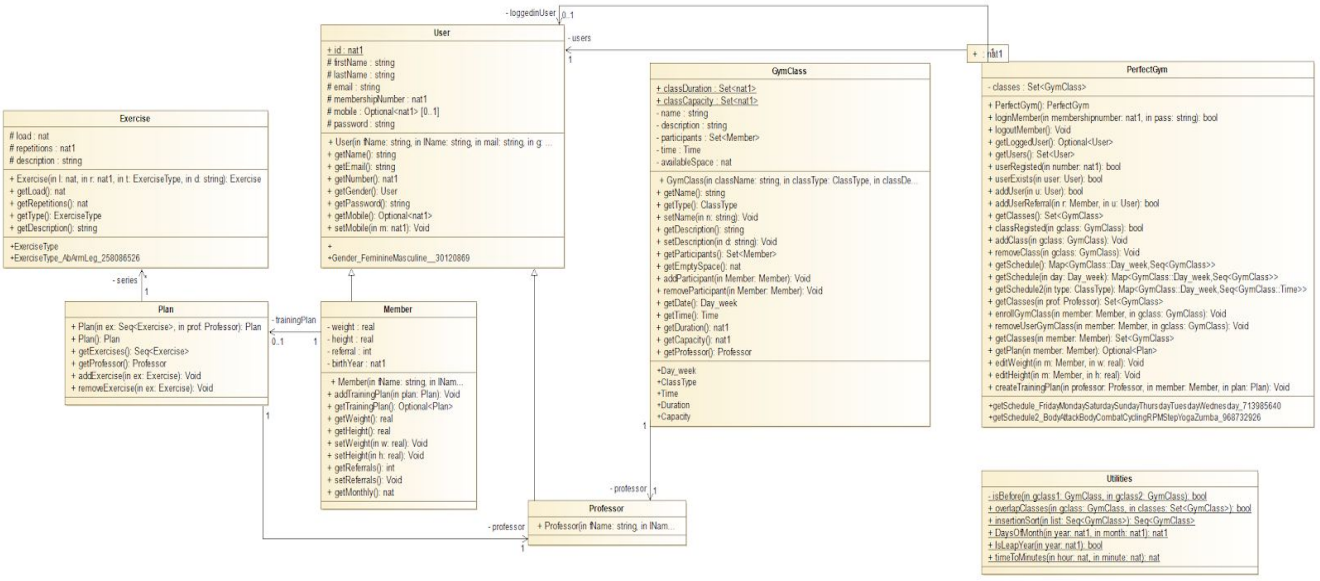
	4. Não existe sobreposição de horário com nenhuma aula já existente.
Pós-condições	1. A aula encontra-se registada no ginásio.
Passos	1. Criar uma aula, especificando o seu tipo, horário, capacidade, entre outros atributos. 2. Associar um professor à aula.
Exceções	(unspecified)

Cenário	**Remover aula do ginásio**
Descrição	É removida uma aula do ginásio.
Pré-condições	1. A aula encontra-se registada.
Pós-condições	1. A aula não se encontra registada.
Passos	(unspecified)
Exceções	(unspecified)

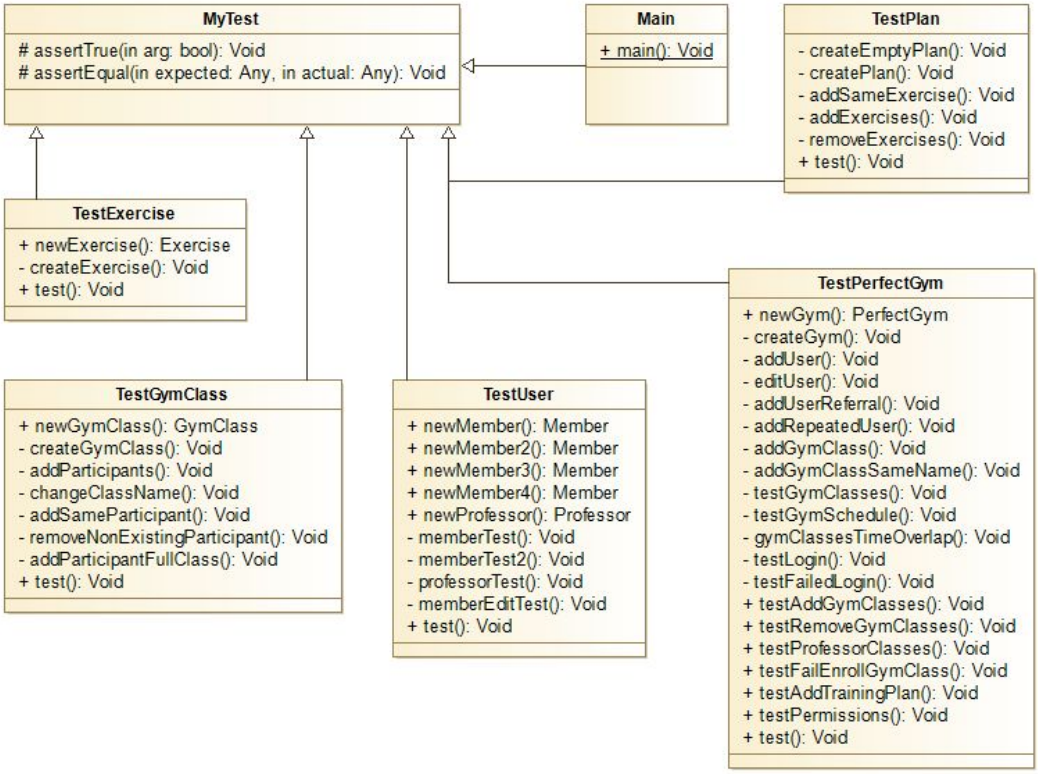
Cenário	**Registo de um utilizador referenciado**
Descrição	Um utilizador cria conta no sistema referenciado por outro membro.
Pré-condições	Cenário idêntico ao registo de um utilizador, acrescentando: 1. O utilizador que referencia o novo membro encontra-se registado no sistema.
Pós-condições	Cenário idêntico ao registo de um utilizador, acrescentando: 1. O utilizador que referencia o novo membro tem mais uma referência.
Passos	(unspecified)
Exceções	(unspecified)

2.2. Class Model

Classes



Classes de Teste



Breve descrição das classes

Classe	Descrição
User	Classe genérica que representa um utilizador.
Member	Representa um membro registado no ginásio. Contém informação pessoal do mesmo, e referência para o seu plano de treino.
Professor	Representa um professor do ginásio. É responsável por aulas de grupo e planos de treino dos membros.
Exercise	Exercício de um plano de treino, com informação de peso, repetições, tipo e descrição.
Plan	Plano de treino constituído por uma sequência de exercícios.
GymClass	Aula de grupo no ginásio. Inclui o seu horário, tipo de aula, professor, capacidade e participantes.
PerfectGym	Ginásio que contêm um conjunto de utilizadores (membros e professores) e aulas de grupo. Contêm a maioria das funções que permitem interagir com o programa.
Utilities	Funções auxiliares como a verificação de sobreposição de horários de aulas, ou ordenação das mesmas.
MyTest	Superclass para as classes de teste. Define assertEquals e assertTrue.
Main	Classe que invoca todas as classes de teste definidas.
TestExercise	Teste da classe Exercise.
TestGymClass	Teste da classe GymClass.
TestPerfectGym	Teste da classe PerfectGym, que engloba a maioria dos cenários de casos de uso.
TestPlan	Teste da classe Plan.
TestUser	Teste da classe User, Member e Professor.

3. Modelo Formal VDM++

O modelo VDM++ bem como a sua coverage poderão ser encontrados no anexo **vdmpp+coverage.pdf**. São incluídos os ficheiros que correspondem à validação do modelo.

4. Verificação do Modelo

4.1. Exemplo de verificação de um domínio

Uma *proof obligations* gerada pelo Overture é:

Número	Nome	Tipo
44	PerfectGym`getUser(nat1)	legal map application

O código sob análise é o seguinte:

```
1.  -- Get user according to membership number
2.  public getUser: nat1 ==> User
3.  getUser(number) == (
4.     return users(number);
5.  )
6.  pre userRegistered(number);
```

A *proof obligation* gerada pelo Overture é a seguinte:

```
1.  (forall number:nat1 & (userRegistered(number) => (number in set (dom users))))
```

Esta operação retorna o utilizador registado com um determinado número de sócio. Esta prova torna-se trivial, pois a pré-condição da função assegura que o número de sócio existe entre os utilizadores, assegurando que a aplicação do mapa é legal.

```
1.  -- Checks if there is a user with a given membership number
2.  public pure userRegistered: nat1 ==> bool
3.  userRegistered(number) == (
4.      return number in set dom users;
5.  );
```

4.2. Exemplo de verificação de uma invariante

Uma *proof obligations* gerada pelo Overture é:

Número	Nome	Tipo
18	Member`setHeight(real)	state invariant holds

O código sob análise é o seguinte:

```
1.  -- Set member height
2.  public setHeight: real ==> ()
3.  setHeight(h) == height:= h
4.  pre h > 0
5.  post height = h;
```

Invariante:

```
1.  inv weight > 0 and height > 0;
```

A *proof obligation* gerada pelo Overture é a seguinte:

```
1.  (forall h:real & ((h > 0) => (((weight > 0) and (height > 0)) => ((weight > 0)
    and (h > 0)))))
```

A invariante em análise define valores válidos para o peso e altura, de forma a evitar a alteração dos mesmos para valores negativos. Depois da atribuição, torna-se necessário provar que a pós condição do bloco implica a invariante.

```
1.  height = h => weight > 0 and height > 0;
```

Como a pré-condição restringe a altura a valores positivos, h > 0.

```
2.  height > 0 => weight > 0 and height > 0;
```

Como o peso não foi alterado, o seu valor é válido, weight > 0. Assim, a implicação é verdadeira e a invariante mantêm-se.

5. Geração de código

Para gerar o código Java, foi utilizada a opção '*Code Generation*' do Overture. Foi criada uma interface de linha de comandos que interage com o código gerado, e permite utilizar facilmente o programa.

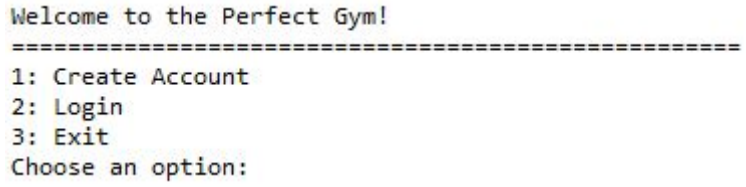


Imagem 1 - Menu inicial.

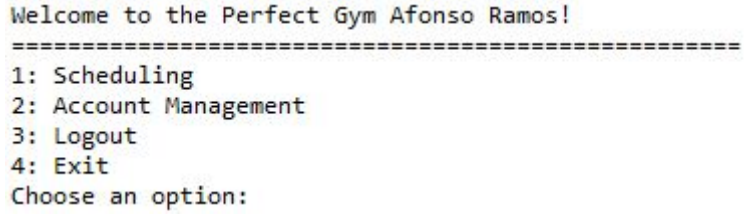


Imagem 2 - Membro faz login na sua conta.

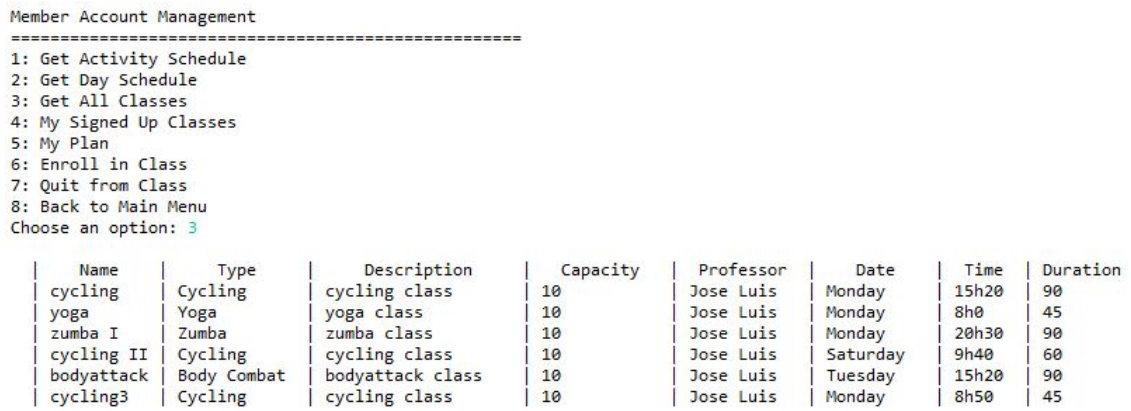


Imagem 3 - Opções de Scheduling.

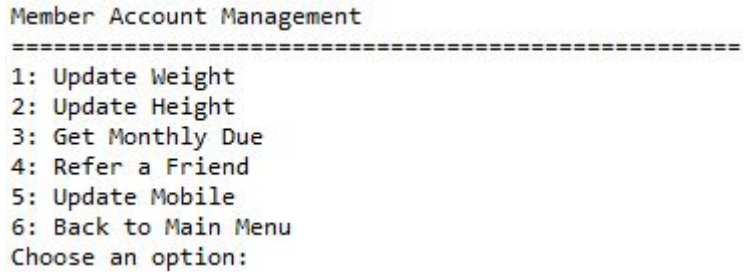


Imagem 4 - Opções de Account Management.

6. Conclusões

6.1. Resultados obtidos

Os resultados pretendidos foram obtidos, e o resultado final é bastante satisfatório. A solução implementada encontra-se corretamente modelada, e corresponde à especificação pedida. Foram implementadas todas as características principais de um ginásio, do ponto de vista de um utilizador, professor, ou administrador, tornando a solução final bastante complexa.

6.2. Possíveis melhoramentos

A equipa cumpriu os objetivos especificados da modelação em VDM++. A interface poderia ser melhorada para uma interface gráfica, no lugar de uma linha de comandos.

6.3. Contribuição

Este projeto demorou cerca de 50 horas a concluir. Os membros do grupo trabalharam de forma colaborativa em todas as componentes do trabalho.

7. Referências

1. Apontamentos da unidade curricular
2. Overture tool website - http://overturetool.org
3. Perfect Gym website - https://www.perfectgym.com