Modelo Formal de um Centro de Exposições em VDM++

Mestrado Integrado em Engenharia Informática e Computação

Métodos Formais em Engenharia de Software

7 de janeiro de 2019

Anabela Costa e Silva, up201506034
Beatriz Souto de Sá Baldaia, up201505633

Descrição Informal do Sistema e Lista de Requisitos

Descrição Informal do Sistema

O nosso projecto é modelar um sistema de gestão de um centro de eventos, como por exemplo o "FIL - Centro de exposições de Lisboa".
Neste sistema é possível o administrador adicionar, remover e alterar, instalações e serviços e aceder a lista e detalhes de todos os utilizadores, eventos, instalações e serviços, ou seja, controlar e gerir o centro.
Os utilizadores poderão criar eventos, comprando bilhetes para eventos, aceder à sua informação pessoal e à de eventos, instalações e serviços.
Para facilitar a pesquisa e uso, fornecemos a capacidade de filtrar eventos e instalações por datas e disponibilidade.

Requisitos Funcionais

ID	Prioridade	Descrição
RF1	Obrigatório	Um utilizador é capaz de se registar e autenticar.
RF2	Opcional	Um utilizador autenticado consegue ver informação pessoal (o seu nome, eventos a que está associado e dinheiro gasto em compra de bilhetes para eventos)
RF3	Opcional	Um utilizador autenticado se for o administrador consegue listar o nome de todos os utilizadores existentes e ver os detalhes dos mesmos.
RF4	Obrigatório	Um utilizador autenticado é capaz de agendar um evento.
RF5	Obrigatório	Um utilizador autenticado é capaz de ver os nomes de todos eventos, públicos e privados para o qual foi convidado.
RF6	Opcional	Um utilizador autenticado é capaz de ver os nomes de todos eventos, públicos e privados para o qual foi convidado, disponíveis, isto é, que ainda têm bilhetes à venda.
RF7	Opcional	Um utilizador autenticado é capaz de ver os nomes de todos eventos, públicos e privados para o qual foi convidado, disponíveis, isto é, que ainda têm bilhetes à venda, entre duas datas especificadas por ele.
RF8	Obrigatório	Um utilizador autenticado é capaz de ver os detalhes de um evento, públicos ou privados para o qual foi convidado.
RF9	Opcional	Um utilizador autenticado é capaz de comprar um bilhete para um evento, públicos ou privados para o qual foi convidado, com bilhetes ainda disponíveis.
RF10	Obrigatório	Um utilizador autenticado se for o administrador, é capaz de consultar o lucro que o centro de exposições fez num determinado ano em aluguer de espaços e serviços para eventos.
RF11	Opcional	Um utilizador autenticado se for o administrador ou o organizador do evento, é capaz de ver o dinheiro que foi gasto no evento com o aluguer do espaço e dos serviços.

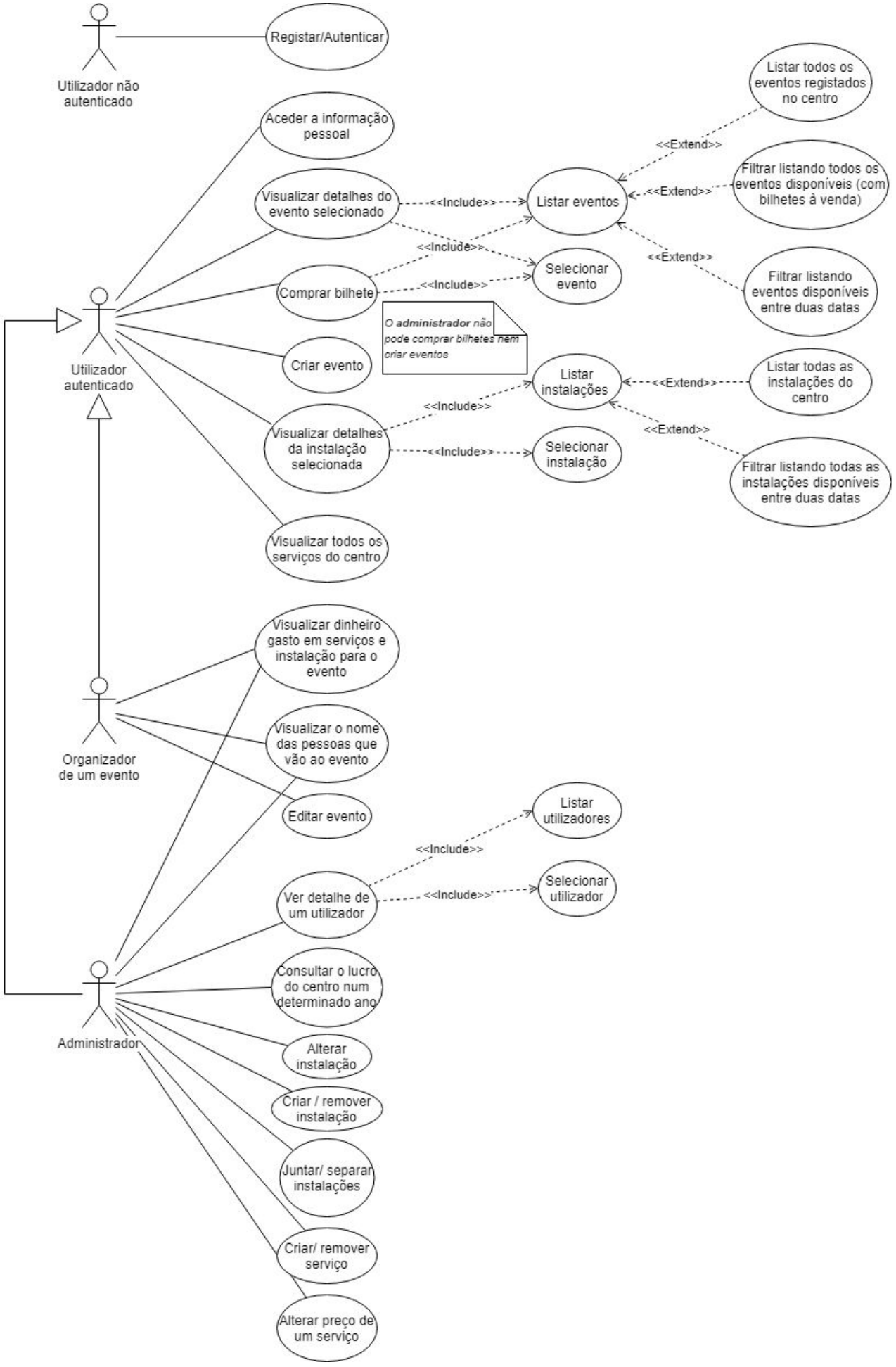
RF12	Opcional	Um utilizador autenticado se for o administrador ou o organizador do evento, é capaz de ver os utilizadores que vão ao evento (organizador, membros do staff e participantes).
RF13	Opcional	Um utilizador se for o organizador do evento, é capaz de editá-lo : mudar o nome, privacidade, número total de bilhetes, preço dos bilhetes, datas, tipo do evento e instalação; adicionar e remover serviços; remover utilizadores (participante ou funcionário); adicionar funcionários; convidar utilizadores (se o evento for privado).
RF14	Obrigatório	Um utilizador autenticado é capaz de listar todas as instalações do centro.
RF15	Opcional	Um utilizador autenticado é capaz de listar todas as instalações livre/disponíveis entre duas datas especificadas por ele.
RF16	Obrigatório	Um utilizador autenticado é capaz de aceder à informação detalhada (custo, dimensões, composição e condições) das instalações disponíveis.
RF17	Opcional	Um utilizador autenticado se for o administrador é capaz de editar as instalações: mudar as medidas, preço da renda e condições/propriedades da instalação.
RF18	Opcional	Um utilizador autenticado se for o administrador é capaz de criar uma nova instalação.
RF19	Opcional	Um utilizador autenticado se for o administrador é capaz de remover uma instalação já existente.
RF20	Opcional	Um utilizador autenticado se for o administrador é capaz de juntar instalações, por exemplo, uma sala já existente passar a fazer parte de um pavilhão.
RF21	Opcional	Um utilizador autenticado se for o administrador é capaz de separar instalações, por exemplo, tirar uma sala de um pavilhão.
RF22	Opcional	Um utilizador autenticado é capaz de aceder à informação (custo e tipo) do conjunto de serviços que o centro tem disponível.
RF23	Opcional	Um utilizador autenticado se for o administrador é capaz de criar um serviço.
RF24	Opcional	Um utilizador autenticado se for o administrador é capaz de remover um serviço existente.
RF25	Opcional	Um utilizador autenticado se for o administrador é capaz de alterar o preço de um serviço.

Restrições Relevantes

ID do Requisito	Descrição da(s) restrição(ões)
RF1	Ao criar uma conta, o nome não dado não pode já existir no sistema.
RF4	O utilizador que vai criar um evento não pode ser o administrador; a instalação onde decorrerá o evento não pode ser a mesma que a de outro evento que ocorre na mesma altura; se um parte de uma instalação (sala ou entrada) já estiver a ser alugada, para o evento a criar não se poderá alugar essa instalação num todo.
RF5, RF6, RF7	Se um evento for privado e o utilizador não foi convidado, não for o administrador, não for funcionário no evento nem organizador do mesmo, não consegue visualizar este evento na listagem de eventos.
RF9	O administrador, organizador do evento e os funcionários do mesmo, não conseguem comprar bilhete para o evento; para os eventos privados, apenas os utilizadores convidados podem comprar bilhete.
RF12	Ao mudar a instalação onde decorrerá o evento, não pode ser selecionada uma instalação que já esteja a ser usada no período em que o evento decorrerá.
RF19	Apenas podem ser adicionadas salas e entradas a pavilhões e entradas a auditório; se uma sala ou entrada já fizer parte de uma instalação, não pode ser agregadas a uma outra instalação.

Modelo UML

Modelo de casos de utilização



Seguem-se as descrições dos cenários de caso de utilização de maior importância:

Cenário	**Registar/ autenticar**
Descrição	Cenário normal de entrada no sistema, quer por registo (criação de uma conta) quer por autenticação (entrada numa conta já existente).
Pré-condição	1- Os nomes dos utilizadores são únicos.
Pós-condição	1- O nome e palavra-passe introduzidos têm que estar registados no sistema (lista de utilizadores).
Passos	1- Introduzir o nome de utilizador 2- Introduzir a palavra-passe
Exceções	1- O nome introduzido não se encontra no sistema, em caso de autenticação 2- Em caso de autenticação, a palavra-passe não ser a correspondente ao nome dado 3- Em caso de registo, o nome introduzido já se encontrar no sistema

Cenário	**Aceder à informação pessoal**
Descrição	Cenário normal em que um utilizador acede à informação sobre si: nome, eventos a que está associado e dinheiro gasto com a compra de bilhetes.
Pré-condição	1- O nome do utilizador autenticado corresponde ao nome do utilizador cuja informação pessoal será mostrada.
Pós-condição	1- O nome apresentado é o mesmo que o do utilizador autenticado 2- Os eventos a que está associado são os para os quais o utilizador comprou bilhete, é organizador ou funcionário 3- O dinheiro gasto é o somatório do preço dos bilhetes já comprados para eventos
Passos	1- Selecionar a opção de ver perfil de utilizador
Exceções	-

Cenário	**Visualizar detalhes do evento selecionado**
Descrição	Cenário normal em que um utilizador autenticado acede à informação detalhada de um evento.
Pré-condição	1- O evento selecionado é público. Se for privado, o utilizador autenticado terá de ser o administrador ou o organizador ou funcionário ou convidado do evento.
Pós-condição	1- O nome do evento é o mesmo que o nome selecionado 2- O número de bilhetes vendidos corresponde ao número de participantes do evento 3- As características apresentadas(datas, instalação, tipo de evento, etc.) correspondem às do evento selecionado
Passos	1- Listar eventos 2- Selecionar um evento específico dos da lista (nome do

	evento)
Exceções	1- O evento selecionado não se encontrar na lista de eventos apresentada ao utilizador

Cenário	**Comprar bilhete**
Descrição	Cenário normal em que o utilizador autenticado compra um bilhete para um evento
Pré-condição	1- O evento é público. Se o evento for privado, o utilizador terá de ser um convidado para o evento 2- O evento ainda tem bilhetes disponíveis 3- O utilizador não pode já estar associado ao evento (já ter comprado bilhete ou ser funcionário ou o organizador) 4- O utilizador não é o administrador
Pós-condição	1- O nome do utilizador autenticado passa a estar na lista de participantes do evento 2- O evento passa a estar na lista de eventos do utilizador
Passos	1- Listar eventos 2- Selecionar um evento específico dos da lista (nome do evento)
Exceções	1- O evento selecionado não se encontrar na lista de eventos apresentada ao utilizador 2- O evento já não ter mais bilhetes disponíveis para venda 3- O evento ser privado e o utilizador não ter sido convidado para o evento 4- O utilizador ser administrador, funcionário ou organizador do evento

Cenário	**Criar evento**
Descrição	Cenário normal em que o utilizador autenticado cria um evento.
Pré-condição	1- O nome do nome evento não pode já existir no sistema (lista de eventos) 2- A data de início tem de ser igual ou inferior à de fim 3- O tipo de evento tem de ser um dos possíveis no sistema (conferência, feira, festa, musicale “team building”) 3- A instalação, onde decorrerá o evento, tem de fazer parte do sistema (lista de instalações) 4- A instalação selecionada não pode estar já a ser usada por outro evento, nem nenhuma subdivisão (sala ou entrada) dessa instalação pode estar a ser usada para outro evento
Pós-condição	1- O evento criado passa a fazer parte dos eventos do sistema (lista de eventos) 2- O organizador do evento é o utilizador autenticado que o criou 3- O evento passa a estar na lista de eventos do utilizador
Passos	1- Introduzir o nome do evento 2- Introduzir o número total de bilhetes a vender

	3- Introduzir o preço dos bilhetes 4- Introduzir o dia, mês e ano das datas de início e fim 5- Introduzir o tipo de evento 6- Selecionar uma instalação
Exceções	1- O tipo de evento escolhido não existir no sistema 2- A instalação escolhida não fazer parte do sistema 3- O nome do evento já existir no sistema 4- O utilizador ser o administrador

Cenário	**Visualizar detalhes da instalação selecionada**
Descrição	Cenário normal em que o utilizador autenticado acede à informação detalhada de uma instalação
Pré-condição	1- O nome da instalação selecionada tem de fazer parte do sistema
Pós-condição	1- O nome da instalação detalhada é o mesmo que o introduzido 2- As características (medidas, condições e composição) da instalação são as mesma que a da instalação selecionada
Passos	1- Listas instalações 2- Introduzir o nome de uma instalação
Exceções	1- A instalação escolhida não pertencer ao sistema

Cenário	**Visualizar todos os serviços do centro**
Descrição	Cenário normal em que o utilizador autenticado acede à informação de todos os serviços que o centro disponibiliza/vende
Pré-condição	-
Pós-condição	1- Os serviços mostrados fazem todos parte do sistema (lista de serviços
Passos	1- Aceder à área relativa a serviços
Exceções	-

Cenário	**Visualizar dinheiro gasto em serviços e instalação para o evento**
Descrição	Cenário normal em que o utilizador autenticado vê as despesas do evento em serviços e aluguer do espaço.
Pré-condição	1- O evento pertence ao sistema 2- O utilizador autenticado é o organizador do evento ou o administrador
Pós-condição	1- O dinheiro gasto em serviços é igual ao somatório dos produtos de número de dias do evento vezes o preço de cada serviço comprado para o evento 2- O dinheiro gasto com o aluguer do espaço é igual ao produto número de dias do evento vezes preço da instalação

Passos	1- Listar eventos 2- Selecionar um evento
Exceções	1- O utilizador não ser nem o administrador nem o organizador do evento 2- O evento não existir no sistema

Cenário	**Visualizar o nome das pessoas que vão ao evento**
Descrição	Cenário normal em que o utilizador autenticado vê os nomes dos utilizadores que vão ao evento (organizador, funcionários e participantes)
Pré-condição	1- O evento pertence ao sistema 2- O utilizador autenticado é o organizador do evento ou o administrador
Pós-condição	1- O organizador é organizador do evento 2- Os participantes são todos os que pertencem a lista de participantes do evento 3- Os funcionários são todos os que pertencem a lista de funcionários do evento
Passos	1- Listar eventos 2- Selecionar um evento
Exceções	1- O utilizador não ser nem o administrador nem o organizador do evento 2- O evento não existir no sistema

Cenário	**Editar evento**
Descrição	Cenário normal em que o utilizador autenticado edita o evento (mudar nome, custo dos bilhetes, datas, instalação, funcionários, convidados, etc.)
Pré-condição	1- O utilizador autenticado é o organizador do evento 2- A nova data de começo tem de ser igual ou anterior à de fim 3- A nova instalação não poderá ser igual à de outro evento que decorra na mesma altura que este 4- O novo serviço a adicionar terá de fazer parte do sistema 5- O serviço a remover do evento terá de fazer parte do evento e do sistema 6- O utilizador a remover do evento terá de fazer parte da lista de participantes ou de funcionários do evento 7- O utilizador a adicionar ao evento terá de fazer parte do sistema e não pode já estar associado ao evento 8- Só se pode convidar um utilizador para o evento se este for privado 9- O utilizador a convidar para o evento tem de fazer parte do sistema e não pode já estar associado ao evento
Pós-condição	1- Os novos valores atribuídos têm que estar atualizados no evento 2- O novo serviço adicionado tem que agora fazer parte da lista de serviços do evento 3- O serviço removido do evento agora já não se encontra na lista de serviço do evento 4- O utilizador removido se era um participante deixa de estar na lista de participantes do evento e na sua lista

	de eventos deixa de estar este evento. Se o utilizador era um funcionário, o utilizador deixa de estar na lista de funcionários do evento e na sua lista de eventos deixa de estar este evento. 5- O utilizador adicionado ao evento se era um participante passa a estar na lista de participantes do evento e na sua lista de eventos passa a estar este evento. Se o utilizador era um funcionário, o utilizador passa a estar na lista de funcionários do evento e na sua lista de eventos passa a estar este evento.
Passos	1- Listar eventos 2- Selecionar um evento 3- Selecionar que parâmetro alterar
Exceções	1- O utilizador não ser o organizador do evento 2- O evento não existir no sistema

Cenário	**Ver detalhes de um utilizador**
Descrição	Cenário normal em que o utilizador autenticado acede à informação pessoal de um utilizador.
Pré-condição	1- O utilizador tem de ser o administrador 2- O nome do utilizador do qual se pretende obter informação, tem de pertencer ao sistema
Pós-condição	1- O nome apresentado é o mesmo que o utilizador autenticado selecionou 2- Os eventos a que está associado são os para os quais o utilizador comprou bilhete, é organizador ou funcionário 3- O dinheiro gasto é o somatório do preço dos bilhetes já comprados para eventos
Passos	1- Listar utilizadores 2- Introduzir o nome do utilizador
Exceções	1- O utilizador selecionado não existir no sistema 2- O utilizador autenticado não ser o administrador

Cenário	**Consultar o lucro do centro num determinado ano**
Descrição	Cenário normal em que o utilizador autenticado consulta quanto dinheiro o centro lucrou num dado ano em aluguer de espaço e serviços para a realização de eventos
Pré-condição	1- O utilizador tem de ser o administrador
Pós-condição	1- O lucro do dado ano é equivalente ao somatório do dinheiro que os eventos, que decorreram nesse ano, gastaram em serviços e aluguer de instalações.
Passos	1- Aceder à área de eventos
Exceções	1- O utilizador autenticado não ser o administrador

Cenário	**Alterar instalação**
Descrição	Cenário normal em que o utilizador autenticado altera as características (medidas, condições e preço) de uma instalação.

Pré-condição	1- O utilizador autenticado é o administrador 2- A instalação selecionada existe no sistema
Pós-condição	1- As medidas da instalação são agora iguais às novas medidas dadas 2- As condições (ter ou não luz natural, telemóveis, paredes amovíveis, etc.) da instalação são agora iguais às novas condições dadas 3- O preço de renda da instalação é agora igual ao novo valor dado
Passos	1- Listar instalações 2- Selecionar uma instalação
Exceções	1- O utilizador autenticado não ser o administrador 2- A instalação selecionada não existir no sistema

Cenário	**Criar/ remover instalação**
Descrição	Cenário normal em que o utilizador autenticado cria ou remove uma instalação.
Pré-condição	1- O utilizador autenticado é o administrador 2- Em caso de eliminar instalação, a instalação terá de se encontrar no sistema 3- Em caso de criação, o nome da nova instalação não poderá já fazer parte do sistema
Pós-condição	1- Em caso de eliminação de instalação, a instalação selecionada deixa de se encontrar no sistema 2- Em caso de criação, a instalação criada para a encontrar-se no sistema 3- Se a instalação a ser removida conter outras instalações do sistema (salas e/ou entradas) então estas também serão apagadas do sistema
Passos	1- Em caso de criação, introduzir o nome, medidas, condições e valor de renda da nova instalação 2- Em caso de eliminação, introduzir o nome da instalação a remover do sistema
Exceções	1- O utilizador autenticado não ser o administrador 2- A instalação selecionada não existir no sistema 3- O nome da nova instalação ser igual ao de outra já existente

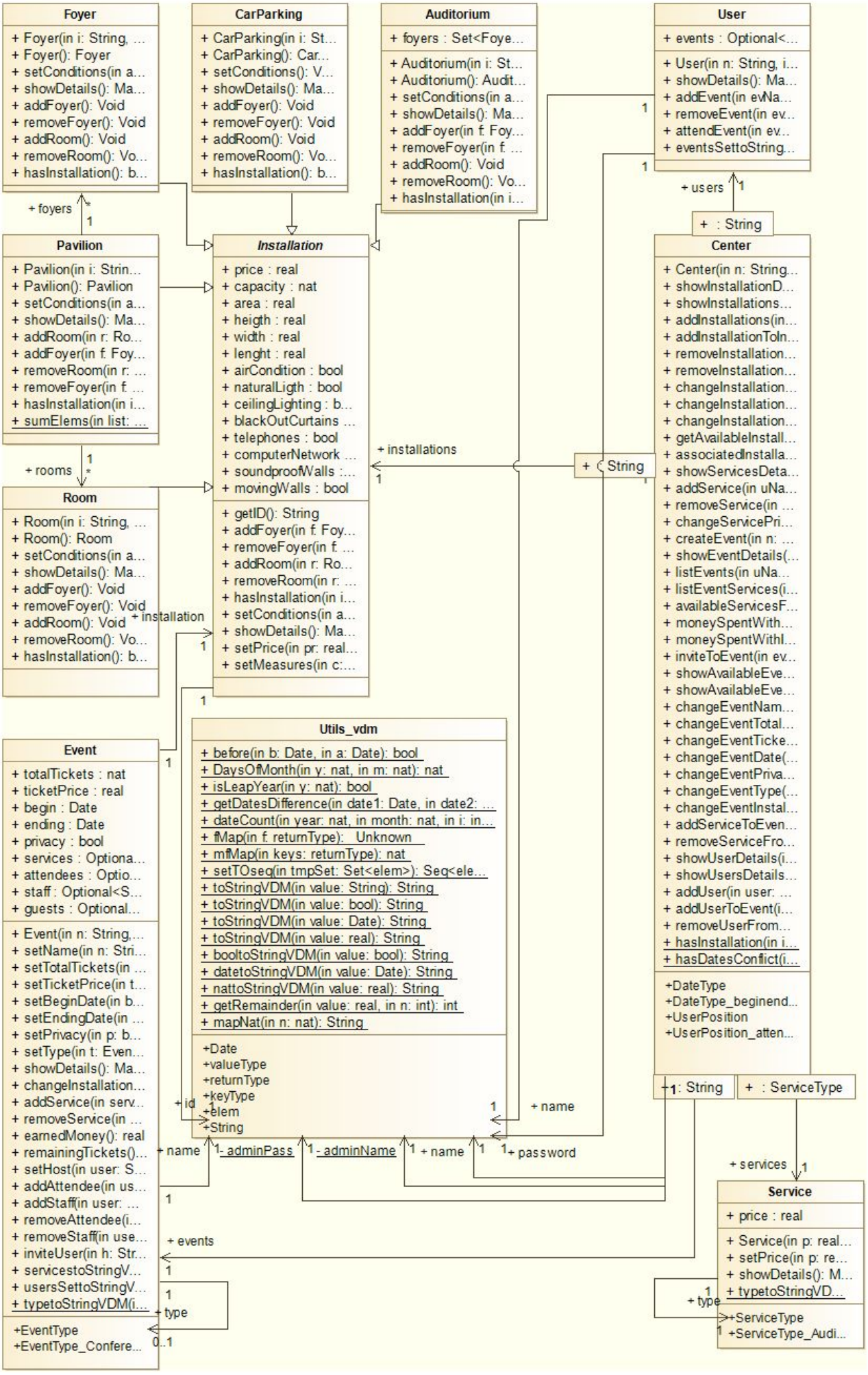
Cenário	**Juntar/ separar instalações**
Descrição	Cenário normal em que o utilizador autenticado agrega ou separa duas instalações
Pré-condição	1- O utilizador autenticado tem de ser o administrador 2- As instalações selecionadas têm de fazer parte do sistema 3- Se for adicionada/removida uma instalação x de uma instalação y, então x só pode ser uma sala ou uma entrada e y só pode ser um pavilhão ou um auditório, sendo que se y for um auditório x não pode ser uma sala
Pós-condição	1- Se uma instalação x for adicionada a uma instalação y, então x encontrar-se á na lista de salas ou entradas da instalação y

Passos	1- Introduzir o nome de utilizador 2- Introduzir a palavra-passe
Exceções	1- O utilizador autenticado não ser o administrador 2- As instalações selecionadas não pertencerem ao sistema 3- A regra de que só podem ser adicionadas/removidas entradas a auditórios e pavilhões e adicionadas/removidas salas a pavilhões, ser violada

Cenário	**Criar/ remover serviço**
Descrição	Cenário normal em que o utilizador autenticado cria ou remove um serviço.
Pré-condição	1- O utilizador autenticado é o administrador 2- O serviço a criar é de um tipo que existe no sistema 3- O serviço a remover existe no sistema
Pós-condição	1- O serviço removido deixa de se encontrar no sistema 2- O serviço adicionada encontra-se no sistema
Passos	1- Introduzir o nome do serviço a remover 2- Introduzir o nome do tipo de serviço e preço do serviço a criar
Exceções	1- O utilizador autenticado não ser o administrador 2- O serviço selecionado a remover não existir no sistema

Cenário	**Alterar o preço de um serviço**
Descrição	Cenário normal em que o utilizador autenticado altera o preço de um serviço.
Pré-condição	1- O utilizador autenticado é o administrador 2- O serviço selecionada pertence ao sistema
Pós-condição	1- O preço do serviço é igual ao novo valor dado
Passos	1- Introduzir o nome do serviço 2- Introduzir o novo valor a atribuir ao serviço
Exceções	1- O utilizador autenticado não ser o administrador 2- O serviço selecionado não existir no sistema

Modelo de classes



Classe	Descrição
Center	Classe principal. Define as variáveis e operações disponíveis ao utilizador. É o centro que contém instalações, eventos, serviços e utilizadores.
Installation	Classe mãe que representa uma instalação.
Pavilion	Subclasse da classe Installation. Representa um pavilhão, que pode ter salas e entradas
Auditorium	Subclasse da classe Installation. Representa um auditório, que pode ter entradas
Room	Subclasse da classe Installation. Representa uma sala.
Foyer	Subclasse da classe Installation. Representa uma entrada.
CarParking	Subclasse da classe Installation. Representa uma parque de estacionamento.
User	Classe que representa um utilizador do nosso sistema/centro.
Event	Classe que representa um evento, que decorrerá numa das instalações do nosso sistema/centro.
Service	Classe que representa um serviço que o nosso sistema/centro vende aos organizadores de eventos.
Utils_vdm	Classe que define tipo e funções úteis para auxiliar operações com datas, strings e maps.

Modelo Formal VDM++

O nosso modelo formal VDM++ encontra-se no pdf que enviamos em anexo já com as tabelas com a cobertura de cada classe implementada. Para certas classes há operações que têm cobertura a 0% pois como são subclasses têm de implementar as operações da classe mãe, no entanto, tendo em conta o contexto do nosso trabalho, algumas subclasses não devem executar essas operações. Por exemplo, apenas os pavilhões e auditórios podem ter entrada, por isso a operação addFoyer() deve falhar para as classes Room, Foyer e CarParking.

Validação do Modelo

Usamos as classes Test e CenterTest para validar o modelo, nelas invocamos as operações com parâmetros exemplificativos para os casos de uso.

```
class Test_vdm

operations
protected assertTrue: bool ==> ()
assertTrue(arg) == return pre arg;
protected assertEqual: ? * ? ==> ()
assertEqual(expected, actual) == if expected <> actual then ( IO`print("Value ("); IO`print(actual);
IO`print(") different from expected ("); IO`print(expected); IO`println(")\n")
) post expected = actual

end Test_vdm
```

```
class CenterTest  is subclass of Test_vdm
operations

 /***** USE CASE SCENARIOS ******/

 --Center Init
       public createCenter: Utils_vdm`String * Installation ==> Center
       createCenter(name, inst) == (
              dcl res: Center := new Center(name, inst);
              res.addUser(new User("admin", "admin1234"));
              return res
       )
       post "admin" in set dom RESULT.users and card dom RESULT.installations = 1;

--Changing installations
-- Here we add installations to the center, aggregate installations to other installations,
-- remove installations from other installations and from center and
-- change installations' attributes
--RF16, RF17, RF18, RF19, RF20, RF21
       public testInstallations: Center ==> ()
       testInstallations(center) == (
              dcl room1: Installation := new Room("Room1", 15, 10, 7, 20, 20, false, true, true, true,
false, false, true, false);
              dcl room2: Installation := new Room("Room2", 15, 10, 7, 20, 20, false, true, true, true,
false, false, true, false);
              dcl pavilion1: Installation := new Pavilion("Pavilion1", 150, 50, 10, 50, 70, false,
false, true, false, false);
              dcl pavilion2: Installation := new Pavilion("Pavilion2", 150, 50, 10, 50, 70, false,
false, true, false, false);
              dcl foyer1: Installation := new Foyer("Foyer1", 10, 15, 4, 6, 6, false, true, false,
true);
              dcl foyer2: Installation := new Foyer("Foyer2", 10, 15, 4, 6, 6, false, true, false,
true);
              dcl foyer3: Installation := new Foyer("Foyer3", 10, 15, 4, 6, 6, false, true, false,
true);
              dcl parking1: Installation := new CarParking("Car Parking1", 30, 50, 7, 50, 50);
              dcl auditorium1: Installation := new Auditorium("Auditorium1", 50, 120, 10, 40, 40,
true, false, true, false, true, false);
              --Add installations
              center.addInstallations("admin", room1);
              center.addInstallations("admin", pavilion1);
              center.addInstallations("admin", pavilion2);
              assertTrue({room1, pavilion1} subset rng center.installations);

              center.addInstallations("admin", auditorium1);
              center.addInstallationToInstallation("admin", "Auditorium1", foyer1);
              assertTrue(center.associatedInstallations(auditorium1, foyer1));
              assertTrue(not center.hasInstallation(auditorium1, room1));
              assertTrue(not center.hasInstallation(room1, auditorium1));
              center.addInstallationToInstallation("admin", "Pavilion1", room1);
              center.addInstallationToInstallation("admin", "Pavilion1", room2);
              assertTrue(center.associatedInstallations(pavilion1, pavilion1));
              assertTrue(center.associatedInstallations(pavilion1, room1));
              center.addInstallationToInstallation("admin", "Pavilion1",  foyer2);
              center.addInstallationToInstallation("admin", "Pavilion2",  foyer3);
              assertEqual(narrow_(center.installations("Pavilion1"), Pavilion).rooms, {room1,room2});
              assertEqual(narrow_(center.installations("Pavilion1"), Pavilion).foyers, {foyer2});
```

```
            assertTrue(foyer2 in set rng center.installations);
            --Remove installations
            center.removeInstallationFromInstallation("admin", "Auditorium1", foyer1);
            assertTrue(not auditorium1.hasInstallation(foyer1));
            center.removeInstallationFromInstallation("admin", "Pavilion2", foyer3);
            assertTrue(not pavilion2.hasInstallation(foyer3));
            center.removeInstallationFromInstallation("admin", "Pavilion1", room1);
            assertEqual(narrow_(center.installations("Pavilion1"), Pavilion).rooms, {room2});
            assertTrue(room1 in set rng center.installations);

            center.addInstallationToInstallation("admin", "Pavilion1", foyer1);
            center.removeInstallation("admin", "Foyer2");
            center.removeInstallation("admin", "Pavilion1");
            center.removeInstallation("admin", "Foyer3");
            center.removeInstallation("admin", "Pavilion2");
            assertTrue(rng center.installations inter {pavilion1, foyer2} = {});
            --Change installation attributes
            center.addInstallations("admin", pavilion1);
            center.addInstallations("admin", foyer2);
            center.addInstallations("admin", parking1);
            center.changeInstallationMeasures("admin", "Room1", 20, 10, 10, 25);
            center.changeInstallationMeasures("admin", "Pavilion1", 20, 10, 10, 25);
            center.changeInstallationMeasures("admin", "Foyer2", 20, 10, 10, 25);
            center.changeInstallationMeasures("admin", "Car Parking1", 20, 10, 10, 25);
            center.changeInstallationMeasures("admin", "Auditorium1", 20, 10, 10, 25);
            assertTrue(room1.capacity = 20 and room1.heigth = 10 and room1.width = 10 and
room1.lenght = 25);
            center.changeInstallationRent("admin", "Room1", 32);
            center.changeInstallationRent("admin", "Pavilion1", 32);
            center.changeInstallationRent("admin", "Foyer2", 32);
            center.changeInstallationRent("admin", "Car Parking1", 32);
            center.changeInstallationRent("admin", "Auditorium1", 32);
            assertEqual(room1.price, 32);
            center.changeInstallationConditions("admin", "Room1", true, true, true, true, true,
true, true, true);
            center.changeInstallationConditions("admin", "Pavilion1", true, true, true, true, true,
true, true, true);
            center.changeInstallationConditions("admin", "Foyer2", true, true, true, true, true,
true, true, true);
            center.changeInstallationConditions("admin", "Car Parking1", true, true, true, true,
true, true, true, true);
            center.changeInstallationConditions("admin", "Auditorium1", true, true, true, true,
true, true, true, true);
            assertTrue(room1.airCondition and room1.naturalLigth and room1.ceilingLighting and
room1.blackOutCurtains and
                    room1.telephones and room1.computerNetwork and room1.soundproofWalls and
room1.movingWalls);
            assertTrue((parking1.airCondition or parking1.naturalLigth or parking1.ceilingLighting
or parking1.blackOutCurtains and
                    parking1.telephones or parking1.computerNetwork or parking1.soundproofWalls or
parking1.movingWalls) = false);

            assertEqual(card center.showInstallationsDetails(), card dom center.installations);
            center.addInstallations("admin", foyer1);

    )
    pre card dom center.installations = 1
    post card dom center.installations = 6;
```

```
--Simple action to add and remove a service from the center
--RF22, RF23, RF24
	public testServices: Center ==> ()
	testServices(center) == (
		dcl service1: Service := new Service(10, <AudioVisual>);

		center.addService("admin", service1);
		assertEqual(card dom center.services, 1);
		center.removeService("admin", <AudioVisual>);
		assertEqual(card dom center.services, 0);
		center.addService("admin", service1);
		assertEqual(card center.showServicesDetails(), 1)
	)
	pre card dom center.services = 0
	post card dom center.services = 1;

--Simple action of adding users to our center
--RF1
	public testUsers: Center ==> ()
	testUsers(center) == (
		dcl user1: User := new User("User1", "1234");
		dcl user2: User := new User("User2", "1234");
		dcl user3: User := new User("User3", "1234");
		dcl user4: User := new User("User4", "1234");

		center.addUser(user1);center.addUser(user2);
		center.addUser(user3);center.addUser(user4);
		assertEqual(card dom center.users, 5);

	)
	pre card dom center.users = 1
	post card dom center.users = 5;

--RF2, RF5, RF8, RF9, RF14, RF15
	public testEdge: Center ==> ()
	testEdge(center) == (
	(
		dcl pavilion3: Installation := new Pavilion("Pavilion3", 150, 50, 10, 50, 70, false,
false, true, false, false);
		dcl foyer4: Installation := new Foyer("Foyer4", 10, 15, 4, 6, 6, false, true, false,
true);
		center.addInstallations("admin", pavilion3);
		center.addInstallationToInstallation("admin", "Pavilion3", foyer4);
	);
	(

		dcl event2: Event := center.createEvent("Event2", 3, 10, mk_Utils_vdm`Date(2018, 12, 4),
mk_Utils_vdm`Date(2018, 12, 4), false, <Conference>, center.installations("Pavilion3"), "User1");
		dcl event3: Event := center.createEvent("Event3", 3, 10, mk_Utils_vdm`Date(2018, 12, 5),
mk_Utils_vdm`Date(2018, 12, 5), true, <Conference>, center.installations("Foyer4"), "User1");
		dcl event4: Event := center.createEvent("Event4", 3, 10, mk_Utils_vdm`Date(2018, 12, 6),
mk_Utils_vdm`Date(2018, 12, 6), true, <Conference>, center.installations("Room1"), "User1");
		dcl event5: Event := center.createEvent("Event5", 3, 10, mk_Utils_vdm`Date(2018, 12, 7),
mk_Utils_vdm`Date(2018, 12, 7), true, <Conference>, center.installations("Room1"), "User1");
		dcl service1: Service := new Service(10, <IT>);
		dcl d1: Utils_vdm`Date := mk_Utils_vdm`Date(2020, 2, 4);
		dcl d2: Utils_vdm`Date := mk_Utils_vdm`Date(2100, 2, 4);
```

```
            assertTrue("Pavilion3" not in set dom
center.getAvailableInstallations(mk_Utils_vdm`Date(2018, 12, 4), mk_Utils_vdm`Date(2018, 12, 4)));
            assertTrue("Foyer4" not in set dom
center.getAvailableInstallations(mk_Utils_vdm`Date(2018, 12, 4), mk_Utils_vdm`Date(2018, 12, 4)));
            assertTrue("Pavilion3" not in set dom
center.getAvailableInstallations(mk_Utils_vdm`Date(2018, 12, 5), mk_Utils_vdm`Date(2018, 12, 5)));
            assertTrue("Foyer4" not in set dom
center.getAvailableInstallations(mk_Utils_vdm`Date(2018, 12,5), mk_Utils_vdm`Date(2018, 12, 5)));


            center.addUserToEvent("Event3", "User2", <staff>);
            assertTrue("User2" in set event3.staff);
            center.addUserToEvent("Event3", "User3", <staff>);
            assertTrue("User3" in set event3.staff);
            center.addUserToEvent("Event3", "User2", <host>);
            assertEqual(event3.host, "User2");
            assertTrue(not "User2" in set event3.staff);
            center.removeUserFromEvent("Event3", "User3", <staff>);
            assertTrue(not "User3" in set event3.staff);

            center.addService("admin", service1);
            center.changeEventPrivacy("Event2", "User1", true);
            assertEqual(event2.privacy, true);
            assertEqual(event2.guests, event2.attendees);
            center.addServiceToEvent("Event2", "User1", <AudioVisual>);
            center.addServiceToEvent("Event2", "User1", <IT>);
            assertTrue(len event2.services = 2);

            assertTrue(center.listEventServices("Event2") union
center.availableServicesForEvent("Event2") = rng center.services);
            center.removeService("admin", <IT>);
            assertTrue(len event2.services = 1);

            center.addService("admin", service1);
            assertTrue(center.listEventServices("Event2") union
center.availableServicesForEvent("Event2") = rng center.services);
            assertTrue(center.listEvents("admin") = rng center.events);
            assertTrue(center.showAvailableEventsBetweenDates("admin",mk_Utils_vdm`Date(2018, 12,
7), mk_Utils_vdm`Date(2018, 12, 7)) = {event5});
            center.inviteToEvent("Event5", "User1", "User2");
            center.addUserToEvent("Event5", "User2", <attendee>);
            center.changeEventTotalTickets("Event5", "User1", 1);
            assertTrue(card event5.attendees = event5.totalTickets);
            assertTrue(center.showAvailableEventsBetweenDates("admin",mk_Utils_vdm`Date(2018, 12,
7), mk_Utils_vdm`Date(2018, 12, 7)) = {});
            assertTrue(center.showAvailableEvents("admin") = rng center.events \ {event5});

            assertEqual(center.showEventDetails("admin", "Event3")("Name"), "Event3");
            assertEqual(center.showEventDetails("User2", "Event3")("Name"), "Event3");

            assertTrue(center.events("Event3").earnedMoney() = 0);
            assertTrue(center.events("Event3").remainingTickets() = 3);

            assertTrue(Event`typetoStringVDM(<TeamBuilding>) ="Team Building");
            assertTrue(Event`typetoStringVDM(<TradeFair>) ="Trade Fair");
            assertTrue(Event`typetoStringVDM(<Musical>) ="Musical");
```

```
            assertTrue(Service`typetoStringVDM(<Catering>) ="Catering");
            assertTrue(Service`typetoStringVDM(<IT>) ="IT");
            assertTrue(Service`typetoStringVDM(<Cleaning>) ="Cleaning");
            assertTrue(Service`typetoStringVDM(<Security>) ="Security");
            assertTrue(Service`typetoStringVDM(<Decoration>) ="Decoration");

            assertEqual(Utils_vdm`getRemainder(1.33,0),33);
            assertEqual(Utils_vdm`mapNat(4),"4");
            assertEqual(Utils_vdm`mapNat(7),"7");
            assertEqual(Utils_vdm`mapNat(9),"9");

            assertTrue(Pavilion`sumElems([1, 0.1]) = 1.1);

            assertTrue(not center.users("admin").attendEvent("Event3"));

            assertTrue(center.showUserDetails("User1","User1")("Name") = "User1");

            center.changeEventDate("Event5", "User1", <ending>, mk_Utils_vdm`Date(2019, 12, 5));
            assertEqual(center.events("Event5").ending, mk_Utils_vdm`Date(2019, 12, 5));
            center.changeEventDate("Event5", "User1", <begin>, mk_Utils_vdm`Date(2019, 12, 2));
            assertEqual(center.events("Event5").begin, mk_Utils_vdm`Date(2019, 12, 2));

            )
      )

      pre card dom center.events = 1
      post card dom center.events = 5;

--Create an event and change its attributes (event details, services and installation)
--RF4, RF6, RF7, RF13
      public testEvent: Center ==> ()
      testEvent(center) == (
            dcl event1: Event := center.createEvent("Event1", 3, 10, mk_Utils_vdm`Date(2018, 12, 1),
mk_Utils_vdm`Date(2018, 12, 3), true, <Conference>, center.installations("Room1"), "User1");

            --Check event and its availability to the users
            assertTrue("Room1" not in set dom
center.getAvailableInstallations(mk_Utils_vdm`Date(2018, 12, 1), mk_Utils_vdm`Date(2018, 12, 3)));
            assertEqual(center.showAvailableEvents("User2"), {});
            center.inviteToEvent("Event1", "User1", "User2");
            assertEqual(center.showAvailableEvents("User2"), {event1});
            assertEqual(center.showAvailableEventsBetweenDates("User3", mk_Utils_vdm`Date(2018, 12,
1), mk_Utils_vdm`Date(2018, 12, 3)), {});
            --Add and remove services from event
            center.addServiceToEvent("Event1", "User1", <AudioVisual>);
            assertTrue(len event1.services = 1);
            center.removeServiceFromEvent("Event1", "User1", <AudioVisual>);
            assertTrue(len event1.services = 0);
            center.addServiceToEvent("Event1", "User1", <AudioVisual>);
            center.changeEventInstallation("Event1", "User1", "Foyer1");
            assertEqual(event1.installation.id, "Foyer1");
            --Change event attributes
            center.changeEventName("Event1", "User1", "New Event Name");
            assertEqual(event1.name, "New Event Name");
            center.changeEventTotalTickets("New Event Name", "User1", 6);
            assertEqual(event1.totalTickets, 6);
            center.changeEventTicketPrice("New Event Name", "User1", 5);
            assertEqual(event1.ticketPrice, 5);
```

```
		center.changeEventDate("New Event Name", "User1", <begin>, mk_Utils_vdm`Date(2018, 12,
2));
		assertEqual(event1.begin, mk_Utils_vdm`Date(2018, 12, 2));
		center.changeEventDate("New Event Name", "User1", <ending>, mk_Utils_vdm`Date(2018, 12,
5));
		assertEqual(event1.ending, mk_Utils_vdm`Date(2018, 12, 5));
		center.changeEventPrivacy("New Event Name", "User1", false);
		assertEqual(event1.privacy, false);
		assertEqual(center.showAvailableEventsBetweenDates("User3", mk_Utils_vdm`Date(2018, 12,
1), mk_Utils_vdm`Date(2018, 12, 3)), {event1});
		center.changeEventType("New Event Name", "User1", <Party>);
)
	pre card dom center.events = 0
	post card dom center.events = 1;

--By changing differente external factors, like services' price,
-- installation's rent and number of attendees,
-- we will analise the money earned and lost with the event
--RF2, RF3, RF9, RF11, RF12, RF25
	public testEventProfit: Center ==> ()
	testEventProfit(center) == (
		dcl event1: Event := center.events("New Event Name");
		--Add users to the event and change their position in it (host, attendee or staff)
		center.addUserToEvent("New Event Name", "User2", <attendee>);
		assertTrue("User2" in set event1.attendees);
		center.addUserToEvent("New Event Name", "User3", <host>);
		assertEqual(event1.host, "User3");
		center.addUserToEvent("New Event Name", "User1", <staff>);
		assertTrue("User1" in set event1.staff);
		center.addUserToEvent("New Event Name", "User4", <attendee>);
		assertEqual(event1.attendees, {"User2", "User4"});
		center.addUserToEvent("New Event Name", "User4", <host>);
		center.addUserToEvent("New Event Name", "User3", <attendee>);
		assertEqual(event1.host, "User4");
		assertEqual(event1.attendees, {"User2", "User3"});
		center.removeUserFromEvent("New Event Name", "User3", <attendee>);
		assertEqual(event1.attendees, {"User2"});
		center.addUserToEvent("New Event Name", "User3", <attendee>);
		assertEqual(event1.attendees, {"User2", "User3"});
		assertEqual(center.showEventDetails("admin", "New Event Name")("Sold tickets"), "2, 0");
		let usersDetails = center.showUsersDetails("admin") in (
			for all userDetails in set usersDetails do (
				if(userDetails("Name") in set event1.attendees) then
assertEqual(userDetails("Money spent"), "5, 0")
			)
		);
		--Check money spent with the event
		assertEqual(center.moneySpentWithServices("User4", "New Event Name"), 40);--since
service (<AudioVisual>) price is 10 and event days are 4
		assertEqual(center.moneySpentWithServices("admin", "New Event Name"), 40);--since
service (<AudioVisual>) price is 10 and event days are 4
		assertEqual(center.moneySpentWithInstallation("User4", "New Event Name"), 40);--since
installation ("Foyer1") rent is 10 and event days are 4
		assertEqual(center.moneySpentWithInstallation("admin", "New Event Name"), 40);--since
installation ("Foyer1") rent is 10 and event days are 4
		center.changeServicePrice("admin", <AudioVisual>, 30);
		assertEqual(center.services(<AudioVisual>).price, 30);
		assertEqual(center.moneySpentWithServices("User4", "New Event Name"), 120)--since
```

```
service (<AudioVisual>) price is 10 and event days are 4

	)
	pre card dom center.events = 1;
 /***** TEST CASES WITH VALID INPUTS ******/

	public static main: () ==> ()
	main() ==
 (
		dcl centerTest: CenterTest := new CenterTest();
		dcl center: Center := centerTest.createCenter("Super Center", new Foyer("Foyer1", 10,
15, 4, 6, 6, false, true, false, true));
		centerTest.testInstallations(center);
		centerTest.testServices(center);
		centerTest.testUsers(center);
		centerTest.testEvent(center);
		centerTest.testEventProfit(center);
		centerTest.testEdge(center);
	);
	  /***** TEST CASES WITH INVALID INPUTS (EXECUTE ONE AT A TIME) ******/
	public static testBuyTicketToPrivateWhileNotInvited: () ==> ()
	testBuyTicketToPrivateWhileNotInvited() == (
		dcl center: Center := new CenterTest().createCenter("Super Center", new Foyer("Foyer1",
10, 15, 4, 6, 6, false, true, false, true));
		dcl user1: User := new User("User1", "1234");
		dcl user2: User := new User("User2", "1234");
		(
		dcl e : Event := new Event("Event1", 3, 10, mk_Utils_vdm`Date(2018, 12, 4),
mk_Utils_vdm`Date(2018, 12, 4), true, <Conference>, center.installations("Foyer1"), "User1");
		e.addAttendee("User2");
		)
	);
	public static testRoomToTwoPavilions: () ==> ()
	testRoomToTwoPavilions() == (
		dcl center: Center := new CenterTest().createCenter("Super Center", new Foyer("Foyer1",
10, 15, 4, 6, 6, false, true, false, true));
		dcl room1: Installation := new Room("Room1", 15, 10, 7, 20, 20, false, true, true, true,
false, false, true, false);
		dcl pavilion1: Installation := new Pavilion("Pavilion1", 150, 50, 10, 50, 70, false,
false, true, false, false);
		dcl pavilion2: Installation := new Pavilion("Pavilion2", 150, 50, 10, 50, 70, false,
false, true, false, false);
		--Add installations
		center.addInstallations("admin", pavilion1);
		center.addInstallations("admin", pavilion2);
		center.addInstallationToInstallation("admin", "Pavilion1", room1);
		center.addInstallationToInstallation("admin", "Pavilion2", room1);
	);
	public static testCreateRoomWithoutParameters: () ==> ()
	testCreateRoomWithoutParameters() == (
		dcl r : Room := new Room(); -- pre-condition fail
		skip;
	);
	public static testAddServicesWithoutAdmin: () ==> ()
	testAddServicesWithoutAdmin() == (
		dcl center: Center := new CenterTest().createCenter("Super Center", new Foyer("Foyer1",
10, 15, 4, 6, 6, false, true, false, true));
		center.addService("User1", new Service(10, <IT>)); -- pre-condition fail
```

```
	);
	public static testEventWithWrongDates: () ==> ()
	testEventWithWrongDates() == (
		dcl event: Event := new Event("WRONG", 10, 9.99, mk_Utils_vdm`Date(2019,1,1),
mk_Utils_vdm`Date(2018,1,1), false, <Party>, new Foyer("Foyer1", 10, 15, 4, 6, 6, false, true, false,
true), "ME");
		skip; -- pre-condition fail
	);
end CenterTest
```

Verificação do Modelo

Exemplo de verificação do domínio

Umas das *"proof obligations"* geradas pelo Overture é:

No.	PO Name	Type
30	Center`removeInstallation(Utils_vdm`String,Utils_vdm`String)	legal map application

O código em análise (com a aplicação do mapeamento relevante a sublinhado) pertence à operação removeInstallation(uName, instID), da classe Center, tal como se pode ver abaixo:

```
--Remove installation from installations map
	public removeInstallation: Utils_vdm`String * Utils_vdm`String ==> ()
	removeInstallation(uName, instID) == (
		dcl instToRemove: set of Utils_vdm`String := {instID};
		for all inst in set rng installations \ {installations(instID)} do (
			if(hasInstallation(inst, installations(instID))) then (instToRemove :=
instToRemove union {inst.id});
			if(inst.hasInstallation(installations(instID))) then
(removeInstallationFromInstallation(uName, inst.id, installations(instID)))
		);
		installations := instToRemove <-: installations
	)
	pre uName in set dom users and uName = adminName and
		instID in set dom installations
	post instID not in set dom installations and
	not exists inst in set rng installations & hasInstallation(installations~(instID), inst);
```

"Proof Obligation" mostra o seguinte:

```
(forall uName:Utils_vdm`String, instID:Utils_vdm`String & (((uName in set (dom users)) and ((uName =
adminName) and (instID in set (dom installations)))) => (instID in set (dom installations))))
```

Neste caso, a prova é trivial porque a condição **'instID in set dom installations'** presente na pré-condição garante que o map "installations" é apenas acedido dentro do seu domínio.

Exemplo de verificação do invariante

Outra "proof obligation" gerada pelo Overture é:

No.	PO Name	Type
5	Installation`setMeasures(nat, real, real, real)	State invariant holds

O código em análise (com a relevante mudança de estado a sublinhado) pertence à operação setMeasures(c, h, w, l), da classe Installation, tal como se pode ver abaixo:

```
-- Changes installation measures
      public setMeasures: nat * real * real * real ==> ()
      setMeasures(c, h, w, l) == (
        atomic(
            capacity := c;
            heigth := h;
            width := w;
            lenght := l;
            area := w * l;
         )
       )
      pre c > 0 and h > 0 and w > 0 and l > 0
      post capacity = c and heigth = h and width = w and lenght = l;
```

A relevante invariante em análise é:

```
inv area = width * lenght
```

Após a execução do bloco 'atomic' nós temos (tecnicamente, isto é a pós-condição do bloco):

```
capacity = c and heigth = h and width = w and lenght = l and area = w * l
```

Assim, temos de provar que isto implica que a invariante se verifica, ou seja, que a seguinte condição é assegurada:

```
width = w and lenght = l and area = w * l => area = width * lenght
```

Como para a garantia da invariante a capacidade e a altura não influenciam, ignoramos a parte 'capacity = c and heigth = h'.

Por uma relação de equivalência, o primeiro membro da expressão acima pode ser convertido para:

```
area = width * lenght
```

O que nos leva a area = width * lenght => area = width * lenght, o que é obviamente verdade.

Geração de Código

Usando a funcionalidade "Code Generation" do Overture Tools geramos o código Java para ser usado com a nossa interface de consola. As classes em VDM++ foram convertidas para classes java que ficaram guardadas em generated/java/src/ExhibitionCenter, e os types definidos em classes VDM++ foram convertidos para classes com terminação "Quote" (por exemplo, para o valor <AudioVisual> do type ServiceType, foi criada uma classe AudioVisualQuote.java), guardadas em generated/java/src/ExhibitionCenter/quotes.
Encontramos alguns problemas, pois algumas funcionalidades permitidas em VDM++ ainda não têm um gerador para Java, assim tivemos de alterar o nosso modelo em VDM++.
Outro problema encontrado foi que parte do código gerado dava erro pois, apesar da verificação do tipo da variável "res" em "is_bool(res)", na função fMap() da classe Utils_vdm, estar a funcionar, a variável não estava a ser reconhecida como booleano em "if(res)". Portanto, tivemos novamente de alterar o modelo VDM++ de forma a que o código em Java reconhecesse o tipo da variável em questão. Apenas tivemos de mudar para "if(res = true)".

Conclusões

O modelo completado cobre todos os requisitos pedidos, expandindo-se para outros opcionais. Assim, o resultado foi um modelo que representa um centro de eventos, capaz de gerir as suas instalações, serviços e utilizadores , permitindo que as instalações e serviços sejam usados na realização de eventos. Desta forma, o centro demonstra ser, não só uma plataforma informativa (obter informação relativa ao espaço, utilizadores e eventos), mas também uma plataforma para agendar eventos, alterá-los e comprar bilhetes para os mesmos.
Ao longo do desenvolvimento do nosso trabalho tentamos aproveitar ao máximo do potencial de VDM++, usando *polymorphic functions*, *curried functions*, expressões lambda, set, seq, map, record, isofclass(), narrow_(), forall, let, if-then-else, cases, atomic() e expressões is.

A nossa interface é realizada na consola, assim é um possível melhoramento deste trabalho seria a sua conversão para uma interface gráfica mais apelativa. Outro melhoramento possível é a implementação das verificações de pré e pós-condições em Java já implementadas em VDM++.

O trabalho foi dividido equitativamente pelos membros do grupo.

Referências

1. VDM-10 Language Manual, Peter Gorm Larsen et al, Overture Technical Report Series No. TR-001, March 2014
2. Overture tool web site, http://overturetool.org
3. "Algorithm for Determining the Number of Days in a Month", 2014 Dispersion Design, http://www.dispersiondesign.com/articles/time/number_of_days_in_a_month
4. "Algorithm For Determining If a Year is a Leap Year", 2014 Dispersion Design, http://www.dispersiondesign.com/articles/time/determining_leap_years