|  |  |
| --- | --- |
| EENG | **Escola de Engenharia**  Departamento de Engenharia Informática  Mestrado Integrado em Engenharia Informática  Desenvolvimento de Sistemas de Software |

Relatório de Desenvolvimento de Sistemas de Software

2016/2017

**Partilha de despesas num apartamento**

**(Fase 1)**

|  |  |  |
| --- | --- | --- |
| C:\Users\João\Desktop\14619865_1249060378471816_236619821_n.jpg | Manel | Jons |
| João Martins  (A68646) | Manuel Castro  (A71646) | João Pereira  (A75273) |

# Introdução

Todos os anos, milhares de alunos que entram para a Universidade acabam por ir estudar para um estabelecimento de ensino superior afastado da sua morada, tornando-se muitas vezes necessário alugar um apartamento próximo da Universidade onde entraram. Para poupar dinheiro, é comum que estes novos alunos partilhem apartamentos, dividindo as despesas entre eles. Existem muitas outras situações em que faz sentido dividir as despesas de um apartamento pelos seus moradores. Independentemente da situação, se não houver acordos e transparência na partilha de despesas de um apartamento, bem como registos daquilo que cada morador deve pagar, podem facilmente surgir tensões e desentendimentos entre moradores. Neste contexto, torna-se útil ter uma aplicação de suporte à partilha de despesas, que permita o registo de despesas e a gestão do seu pagamento por parte de moradores registados.

É neste enquadramento que surge o trabalho prático de DSS, que visa aplicar os conhecimentos adquiridos nas aulas ao desenvolvimento de uma aplicação de suporte à partilha de despesas num apartamento. Em termos de fases, o trabalho divide-se em:

* **Fase 1:** Análise de requisitos, modelação do domínio da partilha de despesas num apartamento, modelação de *use cases* do sistema de *software* a desenvolver e especificação de *use cases*;
* **Fase 2:** Modelação estrutural, comportamental e implementação do sistema de *software*;

Este relatório diz respeito à 1ª dessas fases.

## Objetivos desta fase

A 1ª fase deste trabalho prático teve como objetivos:

* Analisar os requisitos da aplicação a desenvolver, para se capturar as necessidades e expectativas dos potenciais utilizadores;
* Produzir um modelo de domínio da partilha de despesas num apartamento, de forma a se representar as entidades e relacionamentos relevantes no contexto da partilha de despesas num apartamento;
* Desenvolver um modelo de *use cases*, identificando quem vai interagir com o sistema (atores), que funcionalidade deverá ser suportada e o que é que cada ator deverá poder fazer no sistema.

# Análise de requisitos

De forma a se garantir o desenvolvimento do sistema certo, procedeu-se à análise de requisitos com base em:

* conversas com moradores que dividem despesas num apartamento (i.e.: potenciais utilizadores da aplicação), principalmente colegas de curso que partilham um apartamento;
* discussões com os docentes de DSS;
* soluções já existentes para a partilha de despesas num apartamento – analisou-se com especial atenção a aplicação *Splitwise* (<https://www.splitwise.com>);

Com base nestes procedimentos, o grupo chegou aos seguintes requisitos que precisam de ser cumpridos:

* Qualquer pessoa que partilha o apartamento pode criar uma conta de utilizador indicando o seu nome, e-mail e palavra-passe;
* Um morador registado no sistema deverá poder autenticar-se no mesmo;
* Um morador autenticado no sistema deverá poder terminar a sua sessão;
* Depois de autenticado, um utilizador poderá modificar o e-mail associado à sua conta;
* Depois de autenticado, um utilizador poderá modificar a sua password;
* Cada morador registado no sistema deverá ter a si associada uma conta corrente que regista os movimentos de dinheiro referentes a esse utilizador:
  + pagamentos feitos,
  + dinheiro que lhe foi pago por outros utilizadores
  + dinheiro em dívida a outros moradores
* Estando autenticado, um morador deverá poder consultar a sua conta corrente;
* Qualquer morador autenticado deverá poder registar despesas no sistema. Ao registar uma despesa, um morador deverá indicar uma descrição para a despesa, o valor total da mesma, a data em que a despesa foi contraída, os moradores registados que deverão participar no pagamento, o valor a pagar por cada um deles (i.e.: o plano de divisão da despesa) e o tipo de despesa (recorrente ou pontual). Se a despesa for recorrente, o morador que a registar deverá também indicar a periodicidade da despesa, ou seja, o intervalo de tempo que a despesa leva a repetir-se. Deverá também existir a opção de definir uma data limite para o pagamento da despesa registada;
* Qualquer morador autenticado deverá poder registar um pagamento que fez a outro utilizador, para pagar a totalidade ou parte de uma dívida que tem para com ele;
* Estando autenticado, um morador deverá poder registar o pagamento da totalidade ou parte de uma despesa que esteja a partilhar. Se o morador registar um pagamento de montante superior ao que lhe foi atribuído no plano de divisão da despesa, o sistema deverá calcular o valor que os restantes moradores envolvidos na partilha da despesa ficarão a dever ao morador que pagou mais do que devia;
* Quando um morador cria uma conta, se já existirem despesas registadas no sistema, a decisão de quais e que parte das despesas existentes deve ser paga pelo novo morador fica ao critério dos restantes moradores que partilham as despesas do apartamento;
* Qualquer morador registado deverá poder sair da partilha de despesas do apartamento, através da eliminação da sua conta. Se, ao abandonar a partilha de despesas, o morador ainda tiver despesas ou dívidas por pagar, o registo do pagamento das despesas deverá ficar à responsabilidade dos restantes moradores, ao passo que o registo do pagamento das dívidas deverá caber ao(s) respetivo(s) credor(es);

# Modelo de domínio

No desenvolvimento do modelo de domínio da partilha de despesas num apartamento, seguiu-se a abordagem habitual, i.e.: começou-se por identificar as entidades que fazem parte do domínio em questão, os relacionamentos entre essas entidades, as multiplicidades das entidades envolvidas em cada relacionamento e, finalmente, representou-se esses aspetos através de um diagrama classes da UML. Desta abordagem, resultou o modelo de domínio apresentado na Figura 1:

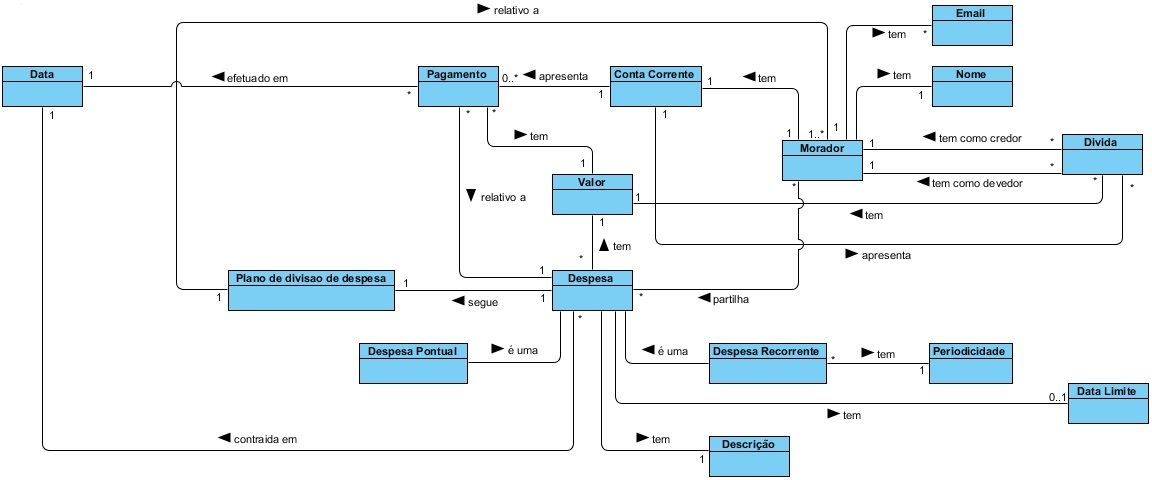


Figura 1-Modelo de domínio da partilha de despesas num apartamento

# Modelo e especificação de *use cases*

## 4.1 Modelo de *uses cases*

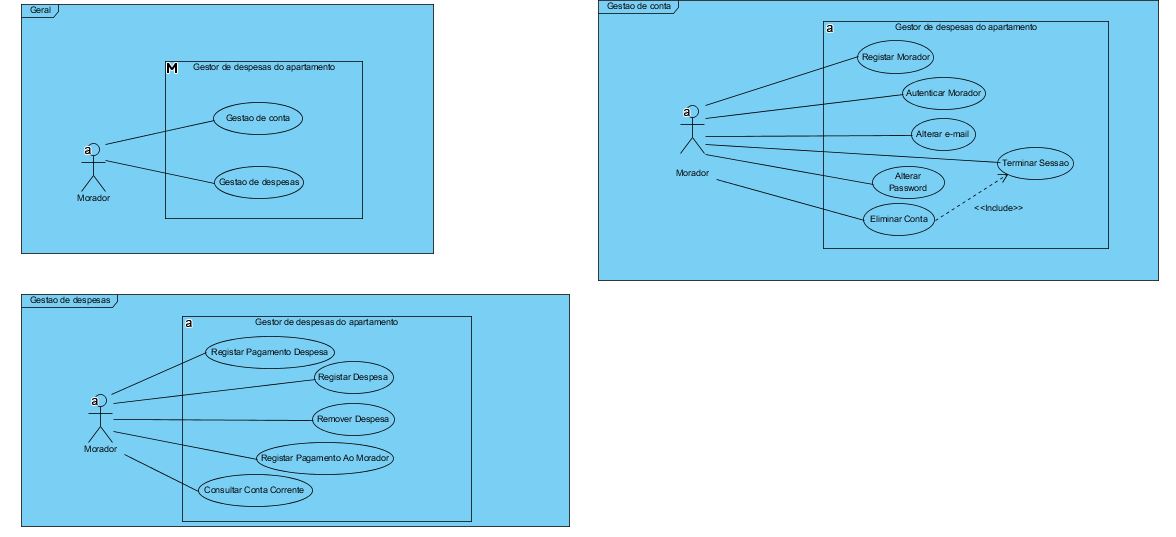


Figura 2-Modelo de use cases do gestor de despesas de um apartamento

## 4.2 Especificação de *use cases*

|  |  |  |
| --- | --- | --- |
| **Use Case:** Registar Morador | | |
| **Descrição:** Morador regista-se no sistema | | |
| **Pré-condição:** Nenhum morador autenticado | | |
| **Pós-condição:** Morador fica registado | | |
|  | **Ator** | **Sistema** |
| **Comportamento**  **Normal** | 1. Indica nome de utilizador, e-mail e palavra-passe |  |
|  | 2. Regista novo morador e notifica restantes moradores registados acerca do registo |
| **Comportamento Alternativo**  **(passo 2)**  **[Primeiro morador a registar-se]** |  | 2.1. Regista novo morador |
| **Exceção 1**  **(passo 2)**  **[Nome utilizador já existe]** |  | 2.1. Informa que já existe um morador com o nome selecionado |
|  | 2.2 Cancela registo |
| **Exceção 2**  **(passo 2)**  **[E-mail já existe]** |  | 2.1. Informa que já existe um morador com o e-mail selecionado |
|  | 2.2 Cancela registo |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Autenticar Morador | | |
| **Descrição:** Morador autentica-se no sistema | | |
| **Pré-condição:** Nenhum morador autenticado | | |
| **Pós-condição:** Morador fica autenticado e com sessão iniciada | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** | 1. Indica nome de morador e palavra-passe |  |
|  | 2. Valida acesso e regista início de sessão |
| **Exceção**  **(passo 2)**  **[nome ou palavra-passe inválida]** |  | 2.1. Indica que pelo menos uma das credenciais introduzidas é inválida |
|  | 2.2. Cancela a operação |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Alterar e-mail | | |
| **Descrição:** Morador autenticado altera o e-mail a ele associado | | |
| **Pré-condição:** Utilizador autenticado | | |
| **Pós-condição:** Morador fica com um e-mail novo associado a ele | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** | 1. Fornece novo e-mail a associar à conta |  |
|  | 2. Regista novo e-mail associado à conta |
| **Exceção 1**  **(passo 2)**  **[O e-mail fornecido já está registado no sistema]** |  | 2.1. Notifica o morador que o e-mail dado já se encontra em uso |
|  | 2.2. Cancela alteração de e-mail |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Alterar Password | | |
| **Descrição:** Morador autenticado altera a password da sua conta | | |
| **Pré-condição:** Utilizador autenticado | | |
| **Pós-condição:** Conta do Morador fica com uma password de conta diferente | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** | 1. Fornece nova password |  |
|  | 2. Regista nova password |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Terminar Sessão | | |
| **Descrição:** Morador autenticado termina a sua sessão | | |
| **Pré-condição:** Existe um morador autenticado | | |
| **Pós-condição:** A sessão aberta é terminada | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** |  | 1. Termina a sessão do morador autenticado |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Eliminar Conta | | |
| **Descrição:** Eliminar uma conta do sistema | | |
| **Pré-condição:** Existe um morador autenticado | | |
| **Pós-condição:** A conta do morador inicialmente autenticado é removida do sistema | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** | 1. Digita palavra-passe |  |
|  | 2. Valida palavra-passe |
|  | 3. **<<include>>** Terminar Sessão |
|  | 4. Elimina conta do morador inicialmente autenticado e notifica os restantes moradores registados |
| **Comportamento Alternativo (passo 4)**  **[O actor é o único morador registado no sistema]** |  | 4.1. Elimina conta do morador inicialmente autenticado |
| **Exceção 1**  **(passo 2)**  **[Palavra-passe inválida]** |  | 2.1. Informa que a password introduzida é inválida |
|  | 2.2. Cancela eliminação de conta |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Registar Despesa | | |
| **Descrição:** Morador regista uma despesa | | |
| **Pré-condição:** Morador autenticado | | |
| **Pós-condição:** Uma nova despesa fica registada no sistema | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** | 1. Fornece descrição da despesa |  |
|  | 1. Regista descrição da despesa |
|  | 1. Pergunta o tipo da despesa |
| 1. Indica que a despesa é pontual |  |
|  | 1. Regista que a despesa é pontual |
| 1. Fornece valor da despesa |  |
|  | 1. Regista valor da despesa |
| 1. Indica data da despesa |  |
|  | 1. Regista data da despesa |
| 1. Fornece data limite |  |
|  | 1. Regista data limite |
| 1. Identifica que moradores dividem a despesa |  |
| 1. Indica quanto é que cada morador deve pagar |  |
|  | 1. Regista como é dividida a despesa |
|  | 1. Atualiza conta corrente dos moradores envolvidos |
| **Comportamento Alternativo 1**  **(passo 4)**  **[a despesa é recorrente]** | 4.1. Indica que a despesa é recorrente |  |
| 4.2. Indica periodicidade da despesa (i.e. com que frequência é que a despesa deve ser paga) |  |
|  | 4.3. Regista periodicidade da despesa |
|  | 4.4. Regista que a despesa é recorrente |
|  | 4.5. Regressa a 6 |
| **Comportamento Alternativo 2**  **(passo 10)**  **[a despesa não tem data limite]** |  | 10.1. Regressa a 13 |
| **Exceção**  **(passo 14)**  **[soma dos valores a pagar difere do valor total da despesa]** |  | 14.1. Informa que a soma dos valores indicados para cada morador pagar difere do valor total da despesa |
|  | 14.2. Cancela registo da despesa |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Registar Pagamento Despesa | | |
| **Descrição:** Morador regista no sistema um pagamento de uma despesa | | |
| **Pré-condição:** Morador autenticado, com pelo menos uma despesa por pagar | | |
| **Pós-condição:** Pagamento fica registado e a conta corrente do morador é atualizada | | |
|  | **Ator** | **Sistema** |
| **Comportamento**  **Normal** |  | 1. Apresenta despesas do morador autenticado |
| 2. Seleciona despesa |  |
| 3. Indica data do pagamento |  |
|  | 4. Regista data do pagamento |
| 5. Indica valor pago não superior ao valor que lhe faltava pagar |  |
|  | 6. Regista pagamento |
|  | 7. Atualiza a conta corrente do morador autenticado |
| **Comportamento Alternativo 1**  **(passo 5)**  **[pagamento de valor superior ao que faltava pagar]** | 5.1. Indica valor pago superior ao que lhe faltava pagar |  |
|  | 5.2. Regista pagamento |
|  | 5.3. Calcula montante que os restantes moradores devem ao ator |
|  | 5.4. Atualiza conta corrente do morador autenticado e as contas correntes dos moradores que ficaram em dívida |
| **Excepção 1**  **(passo 5)**  **[montante do pagamento excede valor total em falta]** |  | 5.1 O registo do pagamento é cancelado |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Registar Pagamento Ao Morador | | |
| **Descrição:** Um morador regista um pagamento de uma dívida | | |
| **Pré-condição:** Morador autenticado e com pelo menos uma dívida por pagar | | |
| **Pós-condição:** O pagamento da dívida fica registado e as contas correntes dos moradores envolvidos são atualizadas | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** |  | 1. Apresenta moradores a quem o ator deve dinheiro |
| 2. Seleciona morador a quem pretende efetuar o pagamento |  |
| 3. Indica montante a pagar |  |
|  | 4. Regista o pagamento do montante indicado |
|  | 5. Atualiza conta corrente dos moradores envolvidos |
| **Exceção**  **(passo 4)**  **[Montante superior ao devido]** |  | 4.1. Indica que o montante a pagar não pode ser superior ao devido |
|  | 4.2. Cancela operação |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Consultar Conta Corrente | | |
| **Descrição:** Morador observa o conteúdo da sua conta corrente | | |
| **Pré-condição:** O utilizador é um morador autenticado | | |
| **Pós-condição:** Morador consulta a sua conta corrente | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** |  | 1. Apresenta o conteúdo da conta corrente do morador autenticado |
| **Comportamento alternativo 1**  **(passo 1)**  **[conta corrente vazia]** |  | 1.1. Informa que a conta corrente não tem quaisquer registos |

|  |  |  |
| --- | --- | --- |
| **Use Case:** Remover despesa | | |
| **Descrição:** Elimina uma despesa registada no sistema | | |
| **Pré-condição:** Morador autenticado | | |
| **Pós-condição:** A despesa selecionada é removida do sistema | | |
|  | **Actor** | **Sistema** |
| **Comportamento**  **Normal** |  | 1. Apresenta despesas registadas |
| 2. Seleciona a despesa a remover |  |
|  | 3. Atualiza as contas correntes dos moradores que pagaram parte ou totalidade da despesa, retirando os pagamentos referentes à despesa |
|  | 4. Remove despesa do sistema |

# Conclusões

Nesta fase do trabalho, foi concebido um modelo de domínio, um modelo de *use cases* e as especificações dos *use cases* que nos guiarão durante o processo de desenvolvimento da nossa aplicação. Esta fase revelou-se mais trabalhosa do que esperávamos devido a várias dificuldades que encontrámos durante a modelação do domínio, identificação de *use cases* e especificação de *use cases*. Apesar de tudo, conseguimos arranjar uma solução adequada ao problema em causa, solução esta que vai ser melhorada e implementada na próxima fase do projeto. A nosso ver, a maior limitação da nossa solução é o facto de não contemplar a possibilidade de editar as despesas depois de criadas. No entanto, consideramos ter um bom modelo, que se aproxima de soluções já postas em prática no mercado de software.