

Universidade do Minho  
Mestrado Integrado em Engenharia Informática

# Sistema de suporte à partilha de despesas num apartamento

## Desenvolvimento de Sistemas de Software

2016/2017

Ana Isabel Castro a55522	Joana Miguel a57127	Lúcia Abreu a71634

Braga, 30 de Dezembro de 2016

## Índice

1.	Introdução .....	2
2.	Identificação e análise de requisitos .....	4
3.	Modelo de Domínio .....	6
4.	Análise dos <i>Use Cases</i> .....	11
4.1.	Diagrama de <i>Use Cases</i> .....	11
4.2.	Especificação dos <i>Use Cases</i> .....	12
5.	Desenho da Interface.....	26
5.1.	Mockups .....	26
5.2.	Máquinas de estado.....	29
5.3.	Experiência de utilização.....	32
6.	Diagramas de sequência de sistema e de subsistemas dos <i>Use Cases</i> .....	33
7.	Diagrama de packages e Diagrama de classes .....	50
8.	Diagrama de sequência para métodos importantes .....	52
9.	Implementação da camada de persistência ORM (base de dados).....	56
9.1.	Decisões importantes.....	56
9.2.	Modelo Entidade Relacionamento da base de dados .....	56
9.3.	Diagrama de classes (com DAO) .....	56
10.	Interface .....	59
10.1.	MVC (Model-View-Controller).....	59
10.2.	Mockups vs Interface .....	59
11.	Diagrama de Instalação .....	63
12.	Conclusão .....	64

## 1. Introdução

Os apartamentos nas grandes cidades são caros, o que se torna incomportável para a grande maioria das pessoas. Uma alternativa cada vez mais popular é a partilha de apartamentos, entre estudantes e não só! O mundo está a mudar e a tornar-se cada vez mais global e, portanto, existem cada vez mais pessoas, já com uma vida profissional, a aderir a este conceito. A ideia de dividir as despesas e partilhar recursos permite, não só, diminuir os gastos como traz outras vantagens, tais como, compartilhar as experiências e momentos do dia-a-dia.

As férias em grupos de amigos e família e, até mesmo desconhecidos que rapidamente se tornam companheiros, é também uma alternativa que tem ganho mais adeptos. Também nestas circunstâncias opta-se muitas vezes por partilhar apartamento e dividir despesas.

Contudo, a gestão das despesas num apartamento partilhado pode tornar-se uma autêntica “dor de cabeça”. *Quem pagou o quê? Quem pagou a quem? Quem deve a quem?* Rapidamente pode tornar-se um pesadelo!

Tendo em conta a tendência de cada vez mais pessoas estarem a partilhar apartamentos, tornou-se óbvia a necessidade de gerir as várias despesas entre os vários moradores.

A aplicação SplitUM pretende dar resposta a esta problemática e desenvolver um sistema de suporte à partilha de despesas num apartamento. A ideia consiste em disponibilizar um sistema fácil e intuitivo para cada morador gerir as despesas associadas à sua estadia no apartamento. A splitUM permite acabar com as preocupações e confusões normais quando se partilha casa com outras pessoas: *agora quantos mais, melhor!*

O desenvolvimento do sistema será suportado por duas etapas principais.

Os objectivos da primeira etapa consistiram:

- Identificação e análise de requisitos
- Identificação das entidades
- Modelo de domínio
- Análise dos *Use Cases*: Requisitos funcionais
- Especificação dos *Use case*

Os objectivos da segunda etapa:

- Refinar e validar o trabalho realizado na primeira etapa
- Modelar a solução do problema através dos diagramas de sequência de sistema e de subsistema
- Definir os métodos mais importantes com diagramas de subsistemas de métodos
- Construir o diagrama de packages e o diagrama de classes
- Implementação da solução modelada numa linguagem orientada aos objectos – Java.

- Desenhar os mockups, o diagrama de estados e implementar a interface gráfica
- Implementar a camada de persistência
- Diagrama de instalação

Para o desenvolvimento do projecto optou-se por seguir uma metodologia RUP, pois esta permite que através de sucessivas iterações se vá refinando a solução pretendida.

## 2. Identificação e análise de requisitos

A identificação de requisitos é uma actividade decisiva no desenvolvimento de *software*. Reside num processo iterativo e necessita de constante validação.

O objectivo da identificação e levantamento de requisitos consiste em compreender e explorar as necessidades dos utilizadores. Se a aplicação não responder adequadamente ao que os utilizadores necessitam, ou seja, se não disponibilizar os requisitos essenciais demonstra-se insuficiente acabando por ser descartada.

Portanto, esta etapa é fundamental e crucial para o sucesso da aplicação.

No sentido de fazer o levantamento de requisitos para a aplicação SplitUM foram realizados os seguintes processos:

- **Entrevistas a pessoas que partilham apartamentos**

Pessoas que partilham apartamentos constituem uma excelente fonte de informação para levantamento de requisitos para a SplitUM. Este grupo de pessoas comprehende e permite colecionar as necessidades reais de quem divide um apartamento.

Nestas entrevistas/conversas foi possível identificar os principais problemas associados à divisão de despesas.

- **Pesquisas na internet**

Foram efectuadas várias pesquisas na *internet* para encontrar possíveis informações, sobretudo em fóruns de discussão.

- **Brainstorming e introspeção**

Foram realizadas várias reuniões de *brainstorming* entre os elementos que desenvolveram a SplitUM. Desta forma foram geradas e exploradas várias ideias com base nas experiências dos elementos do grupo/equipa. As reuniões de *brainstorming* mostraram ser essenciais e muito importantes no levantamento de requisitos.

Assim, foram identificados os seguintes requisitos:

- A partir do momento em que uma pessoa se regista torna-se um morador e tem uma conta associada na aplicação
- O apartamento pode ter vários moradores
- Os moradores associados ao apartamento podem fazer a gestão das despesas

- Um apartamento tem despesas associadas
- Qualquer morador pode adicionar uma despesa
- Uma despesa pode ser do tipo: recorrente ou extraordinária (despesa ocasional)
- Uma despesa recorrente tem uma periodicidade (diária, semanal, mensal, anual, etc)
- Uma despesa recorrente pode ser fixa (valor periódico fixo) ou variável (valor periódico variável)
- O morador que adiciona a despesa especifica os detalhes associados à despesa:
  - Outros moradores associados à despesa
  - Tipo de despesa (fixa ou variável)
  - Qual o tipo de divisão de pagamento (divisão por percentagem, divisão por valor)
- Cada morador pode fazer alterações aos detalhes da despesa
- Um morador que esteja ausente um período de tempo pode indicar a ausência durante esse intervalo ficando dissociado das despesas que permitem ausência
- Cada morador pode consultar o balanço/saldo da sua conta corrente
- Cada morador tem um balanço com cada um dos restantes moradores
- Cada morador pode consultar o balanço/saldo global do apartamento e verificar o estado das dívidas
- Cada morador pode depositar quantias na sua conta corrente que serão utilizadas para o pagamento de despesas
- A gestão de despesas deve ser global e o saldo global do apartamento deve ser a soma dos balanços/saldos de cada morador
- Mesmo que o morador não tenha despesas a si associadas mas existam divididas no apartamento, o valor do morador é utilizado para pagar essas dívidas. Neste caso, os morados que estão associados à despesa em causa mas não têm saldo suficientes, ficam a dever ao morador/apartamento.

### 3. Modelo de Domínio

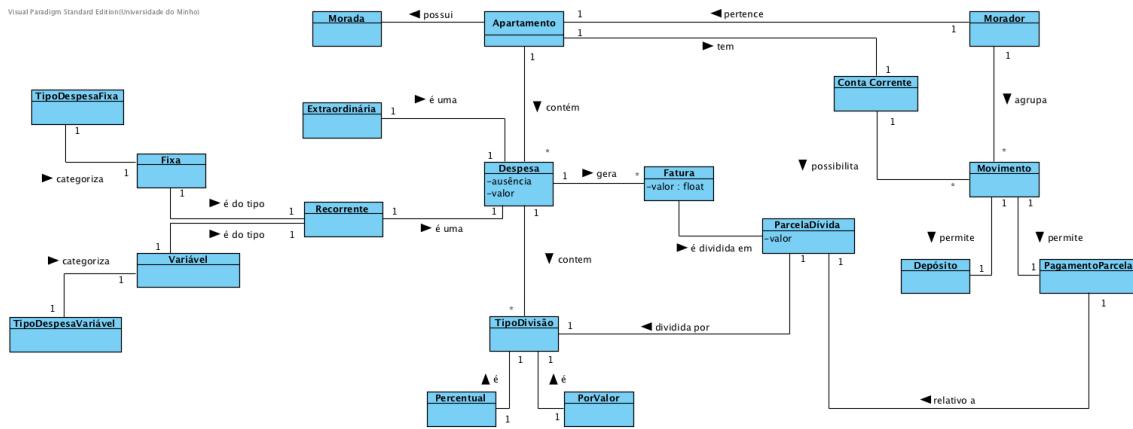
O modelo de domínio representa o problema a desenvolver. Captura, como o próprio nome indica, o domínio do problema. Ou seja, captura as entidades e o relacionamento entre elas. Constitui a base para a análise de requisitos e ajuda a raciocinar sobre o problema.

A ideia principal da SplitUM é providenciar uma aplicação que permita gerir as despesas de um apartamento. Com base nos requisitos levantados e das várias ideias que foram surgindo decidiu-se implementar uma aplicação nativa, off-line e dedicada para cada apartamento. A ideia central reside em gerir o apartamento através dos vários depósitos que os moradores efectuam e utilizá-los para fazer o pagamento de dívidas do apartamento.

Após a análise dos requisitos anteriormente apresentados foram identificadas as seguintes entidades:

- Morador
- Apartamento
- Morada
- Despesa
  - Recorrente
    - Fixa
      - TipoDespesaFixa
    - Variável
      - TipoDespesaVariavel
  - Extraordinária
- Fatura
- TipoDivisão
  - Percentual
  - PorValor
- ParcelaDívida
- ContaCorrente
- Movimento
  - Depósito
  - PagamentoParcela

O modelo de domínio foi representado por um diagrama de classes. Na **Figura 1** apresenta-se o modelo de domínio obtido.



**Figura 1 – Modelo de domínio**

Uma importante entidade da aplicação splitUM é o Morador. O Morador representa o utilizador registado que pertence ao Apartamento. Portanto, o Apartamento tem vários Morador(es).

O Apartamento possui uma Morada e contém uma (ou mais) Despesa(s) que resulta(m) dos gastos dos Moradores que partilham o Apartamento. Uma Despesa pode ser do tipo Recorrente ou do tipo Extraordinária.

Uma despesa Extraordinária diz respeito a gastos ocasionais que ocorrem excepcionalmente ou a imprevistos. Por outro lado, uma despesa Recorrente é uma despesa regular que se sucede periodicamente, pode ser Fixa ou Variável. Categorizaram-se as despesas Fixas em TipoDespesaFixa e as despesas variáveis em TipoDespesaVariável.

Uma despesa tem associado um determinado valor e pode (ou não) permitir ausência, isto é, um Morador pode definir que durante um intervalo de tempo não pretende ter associado aquela despesa.

Cada Despesa gera várias Faturas. Cada Fatura é dividida em várias ParcelasDívida resultante do TipoDivisão (percentual ou PorValor). Um Morador agrupa várias ParcelasDívida que correspondem às suas respectivas dívidas.

O Apartamento tem associado uma ContaCorrente que possibilita Movimentos de cada um dos Moradores. Os movimentos podem ser Depósito ou PagamentoParcela. O PagamentoParcela é relativo ao pagamento de uma ParcelaDivida.

## Decisões mais importantes

- **Existe apenas um Apartamento**

Esta decisão foi tomada para simplificar o sistema a desenvolver.

Caso, o morador tenha várias residências então deverá instalar a aplicação SplitUM em todas as moradias.

Exemplo: instalar a aplicação na morada permanente, na morada em tempos de aulas ou até mesmo em moradas em intervalos periódicos (como férias)

- **Uma despesa pode permitir Ausência**

Um Morador pode definir que durante um intervalo de tempo não pretende ter associado aquela despesa.

Apenas algumas Despesas deverão permitir esta opção, como a água e a luz. A renda do apartamento é um exemplo de uma despesa que não deverá permitir esta opção.

Exemplo: Um estudante que partilha as despesas de um apartamento e que todos os fins de semana regresse a casa, pode dissociar as despesas relativas ao período de tempo que não se encontra no apartamento, neste caso, ao fim de semana.

Exemplo: Um Morador que se ausente para passar 2 semanas de férias não quer ter associado as despesas referentes a esses dias.

- **Uma despesa pode ser extraordinária ou recorrente (Fixa ou Variável)**

Surgiu a necessidade de dividir as despesas em extraordinárias e recorrentes.

As despesas extraordinárias surgem ocasionalmente e as despesas recorrentes surgem periodicamente.

Ainda nas despesas recorrentes definiu-se que estas podem ser fixas ou variáveis.

Exemplo:

Um exemplo de despesa extraordinária poderá ser o pagamento de algum eletrodoméstico que se tenha partido inesperadamente.

Um exemplo de despesa recorrente fixa é a renda da casa. A renda da casa apresenta um valor constante a pagar mensalmente.

Um exemplo de despesa recorrente variável é a conta da água, luz ou gás.

Este tipo de despesas tem que ser paga recorrentemente (por exemplo, por mês) mas o valor associado a cada pagamento pode variar conforme o que foi gasto desse recurso.

- **Uma despesa gera faturas**

Uma despesa dá origem a uma ou mais faturas. Caso a despesa seja extraordinária, terá uma única fatura. No caso de ser recorrente, não existir várias faturas, cada uma, relativa à periodicidade associada à despesa.

**Exemplo:** Numa despesa extraordinária de um eletrodoméstico partido, haverá uma única fatura.

**Exemplo:** Para a despesa recorrente renda da casa, com periodicidade mensal, todos os meses é gerada uma nova fatura.

- **Uma fatura é dividida em várias Parcelas de dívida**

Uma fatura deve ser dividida em várias Parcelas de dívida que ficam agrupadas e associadas aos vários Moradores.

**Exemplo:** Se uma fatura da *renda da casa* for dividida de igual forma por todos os Moradores então, cada um, terá associado a respectiva Parcela de Dívida.

- **Uma despesa apresenta um tipo de divisão percentual ou por valor**

Para tornar o sistema mais interessante, flexível e possibilitar aos utilizadores definir como é dividida a despesa definiu-se que cada despesa pode ser dividida de diferentes formas, conforme o desejado.

**Exemplo:** A despesa renda da casa poderá ser dividida de forma percentual por todos. Ou, se um dos moradores tiver um quarto com maiores dimensões/condições poderá ter a si associado, uma fatia percentual maior do pagamento da renda da casa quando comparado com os restantes moradores.

**Exemplo:** A divisão da despesa poderá ser feita por valor. Em vez de se utilizar percentagens, define-se a quantidade numérica que cada morador tem que pagar de uma despesa. Por exemplo, numa compra de alimentação do tipo *takeway* cada um poderá pagar exatamente o respetivo valor do seu pedido.

- **Um Morador tem uma conta corrente**

Apesar de a ideia central ser gerir as contas do apartamento, definiu-se que cada Morador tem, em termos conceptuais, uma conta corrente com os respetivos movimentos associados.

- **A conta corrente possibilita movimentos (depósitos e pagamentos de parcelas)**

A conta corrente tem a si associados dois tipos de movimentos.

Os moradores podem fazer depósitos que permitem fazer pagamentos de parcelas.

Desta forma, permite-se registar os valores que entram para pagar as despesas do apartamento e, efetivamente, fazer os pagamentos.

**Exemplo:** Um morador faz um depósito de 100 euros associado à sua conta corrente. Este valor fica disponível no sistema apartamento para efetuar pagamentos de parcelas de dívidas que necessitam de ser liquidadas.

## 4. Análise dos *Use Cases*

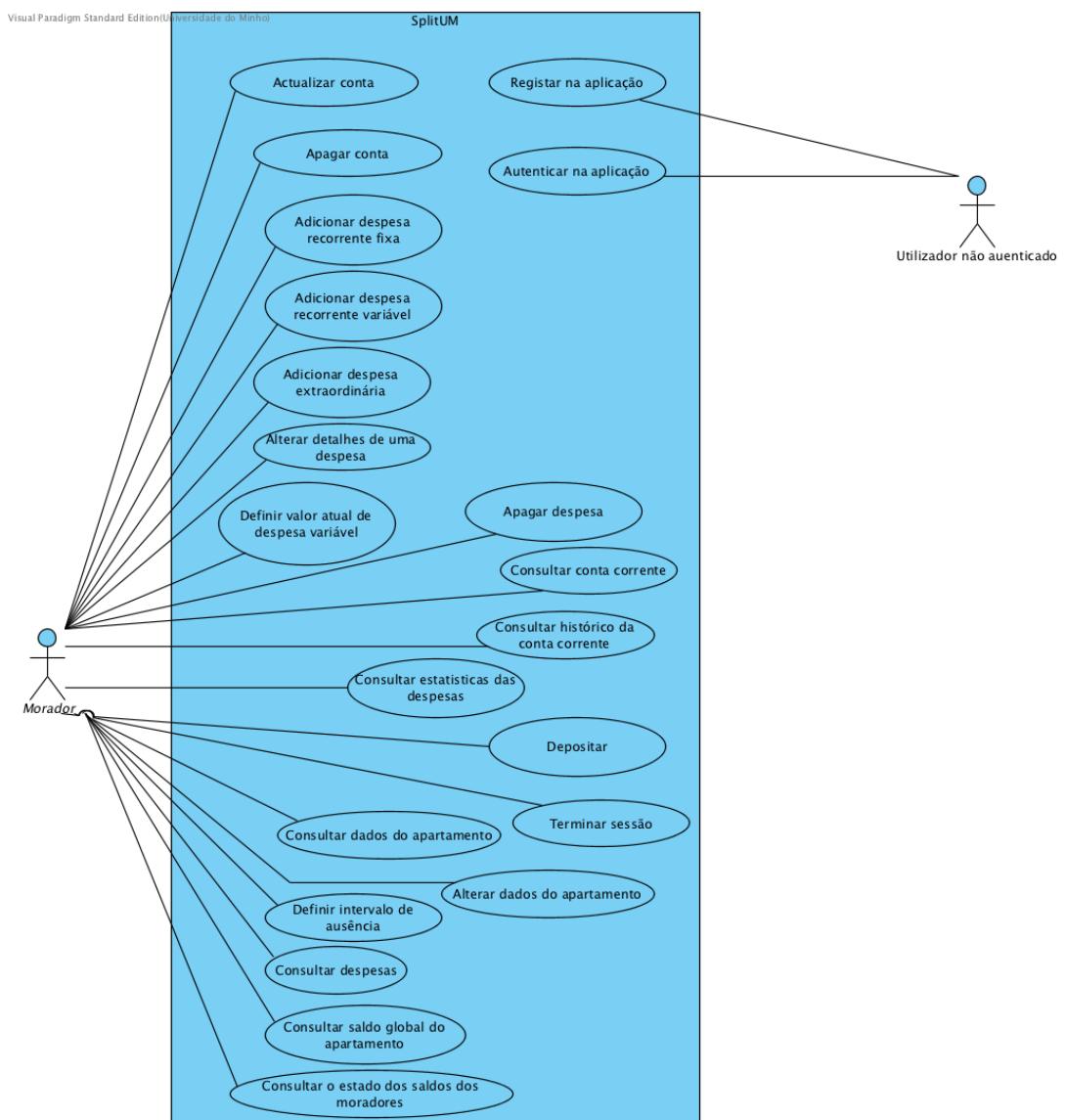
### 4.1. Diagrama de *Use Cases*

Os *Use Cases* são uma metodologia utilizada para analisar, identificar, esclarecer e organizar os requisitos do sistema. Um *Use Case* é constituído pelas interacções entre os actores e o sistema num ambiente em particular, com um objectivo específico.

Na aplicação splitUM foram identificados dois tipos de actores que interagem com o sistema:

- **Utilizador não autenticado:** apenas poderá registar-se ou autenticar-se na aplicação, tornando-se num Morador
- **Morador:** apresenta várias interacções com o sistema

Na **Figura 2** apresenta-se o diagrama de *use cases*.



**Figura 2 – Diagrama de *use cases***

Assim, os use cases identificados e que se pretendem implementar são:

- **UC1:** Registar na aplicação
- **UC2:** Autenticar na aplicação
- **UC3:** Atualizar conta
- **UC4:** Apagar conta
- **UC5:** Adicionar despesa recorrente fixa
- **UC6:** Adicionar despesa recorrente variável
- **UC7:** Adicionar despesa extraordinária
- **UC8:** Alterar detalhes de uma despesa (exemplo: Recorrente fixa)
- **UC9:** Definir valor atual da despesa recorrente variável
- **UC10:** Apagar/desativar despesa
- **UC11:** Consultar conta corrente do morador
- **UC12:** Consultar histórico/movimentos da conta corrente do morador
- **UC13:** Consultar estatísticas das despesas de um morador
- **UC14:** Depositar quantia na conta
- **UC15:** Terminar sessão
- **UC16:** Definir tempo intervalo de ausência
- **UC17:** Consultar detalhes do apartamento
- **UC18:** Alterar detalhes do apartamento
- **UC19:** Consultar despesas
- **UC20:** Consultar saldo global do apartamento
- **UC21:** Consultar o estado do saldo dos moradores

#### **4.2. Especificação dos *Use Cases***

Tendo em conta o diagrama de *Use Cases* apresentado anteriormente foram realizadas as especificações destes *Use Cases*.

A especificação de um *Use Case* consiste na sequência de interacções entre o sistema e o utilizador. Deve conter todas as actividades do sistema que tenham significado para os utilizadores.

Na elaboração da especificação dos *Use Cases* foi utilizado uma notação tabular utilizando o *template* disponibilizado pelo *Visual Paradigm*.

Nas seguintes **Tabelas** seguem a especificações dos *Use Cases*.

**Tabela 1** – Especificação do UC1 - Registar na aplicação

<b>Use Case</b>	Registrar na aplicação																
<b>Brief Description</b>	Permite que um utilizador não autenticado se registe na aplicação																
<b>PreConditions</b>																	
<b>Pos-Conditions</b>	O utilizador fica registado como morador na aplicação																
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Insere dados de registo</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Valida os dados de registo</td></tr> <tr> <td><b>3</b></td><td></td><td>Regista utilizador</td></tr> <tr> <td><b>4</b></td><td></td><td>Informa sucesso de registo</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Insere dados de registo		<b>2</b>		Valida os dados de registo	<b>3</b>		Regista utilizador	<b>4</b>		Informa sucesso de registo	
	<b>Actor input</b>	<b>System Response</b>															
<b>1</b>	Insere dados de registo																
<b>2</b>		Valida os dados de registo															
<b>3</b>		Regista utilizador															
<b>4</b>		Informa sucesso de registo															
<b>Excepção 1 (Passo 2) [Dados de registo inválidos]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td></td><td>Informa sobre dados inválidos</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa sobre dados inválidos										
	<b>Actor input</b>	<b>System Response</b>															
<b>1</b>		Informa sobre dados inválidos															

**Tabela 2** – Especificação do UC2 – Autenticar na aplicação

<b>Use Case</b>	Autenticar na aplicação																
<b>Brief Description</b>	Permite que um utilizador não autenticado se autentique como morador																
<b>PreConditions</b>	O utilizador não está autenticado																
<b>Pos-Conditions</b>	O utilizador fica autenticado como morador																
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Insere dados de autenticação</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Valida os dados de autenticação</td></tr> <tr> <td><b>3</b></td><td></td><td>Autentica morador</td></tr> <tr> <td><b>4</b></td><td></td><td>Informa sucesso de autenticação</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Insere dados de autenticação		<b>2</b>		Valida os dados de autenticação	<b>3</b>		Autentica morador	<b>4</b>		Informa sucesso de autenticação	
	<b>Actor input</b>	<b>System Response</b>															
<b>1</b>	Insere dados de autenticação																
<b>2</b>		Valida os dados de autenticação															
<b>3</b>		Autentica morador															
<b>4</b>		Informa sucesso de autenticação															
<b>Excepção 1 (Passo 2) [Dados de autenticação inválidos]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td></td><td>Informa sobre dados de autenticação inválidos</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa sobre dados de autenticação inválidos										
	<b>Actor input</b>	<b>System Response</b>															
<b>1</b>		Informa sobre dados de autenticação inválidos															

**Tabela 3 – Especificação do UC3 – Actualizar conta**

<b>Use Case</b>	Actualizar conta																					
<b>Brief Description</b>	Permite actualizar os dados da conta de morador																					
<b>PreConditions</b>	O morador encontra-se autenticado																					
<b>Pos-Conditions</b>	Os dados da conta foram actualizados com sucesso																					
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td>Escolhe actualizar os dados da sua conta</td> <td></td> </tr> <tr> <td><b>2</b></td> <td></td> <td>Apresenta as informações da conta do morador</td> </tr> <tr> <td><b>3</b></td> <td>Insere dados actualizados</td> <td></td> </tr> <tr> <td><b>4</b></td> <td></td> <td>Valida dados</td> </tr> <tr> <td></td> <td></td> <td>Atualiza conta</td> </tr> <tr> <td><b>5</b></td> <td></td> <td>Informa sucesso na actualização dos dados na conta</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Escolhe actualizar os dados da sua conta		<b>2</b>		Apresenta as informações da conta do morador	<b>3</b>	Insere dados actualizados		<b>4</b>		Valida dados			Atualiza conta	<b>5</b>		Informa sucesso na actualização dos dados na conta
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>	Escolhe actualizar os dados da sua conta																					
<b>2</b>		Apresenta as informações da conta do morador																				
<b>3</b>	Insere dados actualizados																					
<b>4</b>		Valida dados																				
		Atualiza conta																				
<b>5</b>		Informa sucesso na actualização dos dados na conta																				
<b>Excepção 1 (Passo 4) [Dados de actualização inválidos]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td></td> <td>Informa sobre dados inválidos</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa sobre dados inválidos															
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>		Informa sobre dados inválidos																				

**Tabela 4 – Especificação do UC4 – Apagar conta**

<b>Use Case</b>	Apagar conta																					
<b>Brief Description</b>	Permite apagar a conta de morador																					
<b>PreConditions</b>	O morador encontra-se autenticado																					
<b>Pos-Conditions</b>	A conta de morador é eliminada e é terminada a sessão																					
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td>Escolhe apagar a sua conta</td> <td></td> </tr> <tr> <td><b>2</b></td> <td></td> <td>Pergunta se o morador tem a certeza</td> </tr> <tr> <td><b>3</b></td> <td>Confirma a eliminação</td> <td></td> </tr> <tr> <td><b>4</b></td> <td></td> <td>Verifica se existem dívidas por saldar</td> </tr> <tr> <td><b>5</b></td> <td></td> <td>Elimina conta</td> </tr> <tr> <td><b>6</b></td> <td></td> <td>Termina sessão</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Escolhe apagar a sua conta		<b>2</b>		Pergunta se o morador tem a certeza	<b>3</b>	Confirma a eliminação		<b>4</b>		Verifica se existem dívidas por saldar	<b>5</b>		Elimina conta	<b>6</b>		Termina sessão
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>	Escolhe apagar a sua conta																					
<b>2</b>		Pergunta se o morador tem a certeza																				
<b>3</b>	Confirma a eliminação																					
<b>4</b>		Verifica se existem dívidas por saldar																				
<b>5</b>		Elimina conta																				
<b>6</b>		Termina sessão																				
<b>Excepção 1 (Passo 3) [Decide cancelar a eliminação]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td>Cancela a eliminação</td> <td></td> </tr> <tr> <td><b>2</b></td> <td></td> <td>Cancela a operação</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Cancela a eliminação		<b>2</b>		Cancela a operação												
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>	Cancela a eliminação																					
<b>2</b>		Cancela a operação																				
<b>Excepção 2 (Passo 5) [Dívidas por saldar]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td></td> <td>Informa que é impossível eliminar contas devido a dívidas</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa que é impossível eliminar contas devido a dívidas															
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>		Informa que é impossível eliminar contas devido a dívidas																				

**Tabela 5 – Especificação do UC5 – Adicionar despesa recorrente fixa**

<b>Use Case</b>	Adicionar despesa recorrente fixa	
<b>Brief Description</b>	Permite a um morador adicionar uma despesa recorrente fixa	
<b>PreConditions</b>	O morador encontra-se autenticado	
<b>Pos-Conditions</b>	A despesa recorrente fixa foi adicionada e associada aos moradores seleccionados	
<b>Flow of Events</b>	<b>Actor input</b>	<b>System Response</b>
	1 Morador escolhe adicionar uma despesa recorrente fixa	
	2	Pede dados da despesa recorrente fixa
	3 Fornece a descrição da despesa, o valor da despesa, a forma de divisão do pagamento, a periodicidade, se permite ausência e seleciona os moradores associados à despesa	
	4	Valida os dados da despesa
	5	Adiciona a despesa
	6	Informa sobre sucesso de adição
<b>Excepção 1 (Passo 4) [Dados inválidos]</b>	<b>Actor input</b>	<b>System Response</b>
	1	Informa sobre dados inválidos

**Tabela 6** – Especificação do UC6 – Adicionar despesa recorrente variável

<b>Use Case</b>	Adicionar despesa recorrente variável	
<b>Brief Description</b>	Permite a um morador adicionar uma despesa recorrente variável	
<b>PreConditions</b>	O morador encontra-se autenticado	
<b>Pos-Conditions</b>	A despesa recorrente variável foi adicionada e associada aos moradores seleccionados	
<b>Flow of Events</b>	<b>Actor input</b>	<b>System Response</b>
	1 Morador escolhe adicionar uma despesa recorrente variável	
	2	Pede dados da despesa recorrente variável
	3 Fornece a descrição da despesa, o valor da despesa, a forma de divisão do pagamento, a periodicidade, se permite ausência e seleciona os moradores associados à despesa	
	4	Valida os dados da despesa
	5	Adiciona a despesa
	6	Informa sobre sucesso de adição
<b>Excepção 1 (Passo 4) [Dados inválidos]</b>	<b>Actor input</b>	<b>System Response</b>
	1	Informa sobre dados inválidos

**Tabela 7 – Especificação do UC7 – Adicionar despesa extraordinária**

<b>Use Case</b>	Adicionar despesa extraordinária	
<b>Brief Description</b>	Permite a um morador adicionar uma despesa extraordinária	
<b>PreConditions</b>	O morador encontra-se autenticado	
<b>Pos-Conditions</b>	A despesa extraordinária foi adicionada e associada aos moradores seleccionados	
<b>Flow of Events</b>	<b>Actor input</b>	<b>System Response</b>
	1 Morador escolhe adicionar uma despesa extraordinária	
	2	Pede dados da despesa extraordinária
	3 Fornece a descrição da despesa, o valor da despesa, a forma de divisão do pagamento e seleciona os moradores associados à despesa	
	4	Valida os dados da despesa
	5	Adiciona a despesa
	6	Informa sobre sucesso de adição
<b>Excepção 1 (Passo 4) [Dados inválidos]</b>	<b>Actor input</b>	<b>System Response</b>
	1	Informa sobre dados inválidos

**Tabela 8 – Especificação do UC8 – Adicionar detalhes de uma despesa recorrente fixa**

<b>Use Case</b>	Alterar detalhes de uma despesa	
<b>Brief Description</b>	Permite alterar detalhes de uma despesa recorrente fixa	
<b>PreConditions</b>	O morador encontra-se autenticado e a despesa existe	
<b>Pos-Conditions</b>	Os detalhes da despesa ficaram alterados	
<b>Flow of Events</b>	<b>Actor input</b>	<b>System Response</b>
	1 Seleciona a despesa recorrente fixa	
	2	Apresenta detalhes da despesa
	3 Altera os detalhes da despesa	
	4	Valida os dados alterados
	5	Actualiza os dados da despesa
	6	Informa sobre sucesso
<b>Excepção 1 (Passo 4) [Dados inválidos]</b>	<b>Actor input</b>	<b>System Response</b>
	1	Informa que os dados são inválidos

**Nota:** utilizou-se como exemplo adicionar detalhes de uma despesa recorrente fixa. Para as restantes despesas o processo é idêntico.

**Tabela 9 – Especificação do UC9 – Definir valor atual da despesa recorrente variável**

<b>Use Case</b>	Definir valor atual da despesa variável	
<b>Brief Description</b>	Permite que o morador atualize o valor de uma despesa recorrente do tipo variável	
<b>PreConditions</b>	O morador encontra-se autenticado e a despesa existe	
<b>Pos-Conditions</b>	É definido o novo valor da despesa variável	
<b>Flow of Events</b>	<b>Actor input</b>	<b>System Response</b>
	1 Seleciona a despesa recorrente variável	
	2	Apresenta dados da despesa recorrente variável
	3 Insere o novo valor da despesa	
	4	Valida os dados inseridos
	5	Atualiza valor da despesa
<b>Excepção 1 (Passo 4) [Dados inválidos]</b>	<b>Actor input</b>	<b>System Response</b>
	1	Informa que os dados inseridos são inválidos

**Tabela 10 – Especificação do UC10 – Apagar/desativar despesa**

<b>Use Case</b>	Apagar despesa (Desativar)	
<b>Brief Description</b>	Permite ao morador apagar/desativar despesa	
<b>PreConditions</b>	O morador encontra-se autenticado e a despesa existe	
<b>Pos-Conditions</b>	A despesa foi apagada	
<b>Flow of Events</b>	<b>Actor input</b>	<b>System Response</b>
	1 Seleciona a despesa a despesa a apagar	
	2	Desativa a despesa
	3	Informa sobre o sucesso

**Tabela 11** – Especificação do UC11 – Consultar conta corrente do morador

<b>Use Case</b>	Consultar conta corrente													
<b>Brief Description</b>	Permite consultar a conta corrente do morador													
<b>PreConditions</b>	O morador encontra-se autenticado													
<b>Pos-Conditions</b>	O morador consulta a sua conta corrente													
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Consultar a sua conta corrente</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Calcula o balanço da conta corrente do morador</td></tr> <tr> <td><b>3</b></td><td></td><td>Apresenta o balanço da conta corrente e o balanço com cada um dos moradores do apartamento</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Consultar a sua conta corrente		<b>2</b>		Calcula o balanço da conta corrente do morador	<b>3</b>		Apresenta o balanço da conta corrente e o balanço com cada um dos moradores do apartamento	
	<b>Actor input</b>	<b>System Response</b>												
<b>1</b>	Consultar a sua conta corrente													
<b>2</b>		Calcula o balanço da conta corrente do morador												
<b>3</b>		Apresenta o balanço da conta corrente e o balanço com cada um dos moradores do apartamento												

**Tabela 12** – Especificação do UC12 – Consultar histórico/movimentos da conta corrente do morador

<b>Use Case</b>	Consultar histórico da conta corrente													
<b>Brief Description</b>	Permite ao morador consultar o histórico da sua conta corrente													
<b>PreConditions</b>	O morador encontra-se autenticado													
<b>Pos-Conditions</b>	O morador consulta o histórico da conta corrente													
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Escolhe ver o histórico da sua conta corrente</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Recolhe os movimentos da conta corrente do morador</td></tr> <tr> <td><b>3</b></td><td></td><td>Apresenta os movimentos da conta corrente ordenados cronologicamente</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Escolhe ver o histórico da sua conta corrente		<b>2</b>		Recolhe os movimentos da conta corrente do morador	<b>3</b>		Apresenta os movimentos da conta corrente ordenados cronologicamente	
	<b>Actor input</b>	<b>System Response</b>												
<b>1</b>	Escolhe ver o histórico da sua conta corrente													
<b>2</b>		Recolhe os movimentos da conta corrente do morador												
<b>3</b>		Apresenta os movimentos da conta corrente ordenados cronologicamente												

**Tabela 13** – Especificação do UC13 – Consultar estatísticas das despesas

<b>Use Case</b>	Consultar estatísticas das despesas																					
<b>Brief Description</b>	Permite que o morador possa consultar estatísticas associadas às despesas.																					
<b>PreConditions</b>	O morador encontra-se autenticado																					
<b>Pos-Conditions</b>	O morador consegue consultar as estatísticas																					
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Escolhe consultar as estatísticas das despesas</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Pede o intervalo de tempo</td></tr> <tr> <td><b>3</b></td><td>Define intervalo de tempo</td><td></td></tr> <tr> <td><b>4</b></td><td></td><td>Valida intervalo de tempo</td></tr> <tr> <td><b>5</b></td><td></td><td>Calcula as estatísticas referentes às despesas do intervalo definido</td></tr> <tr> <td><b>6</b></td><td></td><td>Apresenta estatísticas</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Escolhe consultar as estatísticas das despesas		<b>2</b>		Pede o intervalo de tempo	<b>3</b>	Define intervalo de tempo		<b>4</b>		Valida intervalo de tempo	<b>5</b>		Calcula as estatísticas referentes às despesas do intervalo definido	<b>6</b>		Apresenta estatísticas
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>	Escolhe consultar as estatísticas das despesas																					
<b>2</b>		Pede o intervalo de tempo																				
<b>3</b>	Define intervalo de tempo																					
<b>4</b>		Valida intervalo de tempo																				
<b>5</b>		Calcula as estatísticas referentes às despesas do intervalo definido																				
<b>6</b>		Apresenta estatísticas																				
<b>Excepção 1 (Passo 4) [Intervalo de tempo inválido]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td></td><td>Informa que o intervalo de tempo definido não é válido</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa que o intervalo de tempo definido não é válido															
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>		Informa que o intervalo de tempo definido não é válido																				
<b>Alternativa 1 (Passo 6) [Sem estatísticas para mostrar]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td></td><td>Informa que não existem estatísticas</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa que não existem estatísticas															
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>		Informa que não existem estatísticas																				

**Tabela 14 – Especificação do UC14 – Depositar quantia na conta**

<b>Use Case</b>	Depositar quantia na conta																					
<b>Brief Description</b>	Permite que o morador possa depositar uma quantia na sua conta corrente																					
<b>PreConditions</b>	O morador encontra-se autenticado																					
<b>Pos-Conditions</b>	A conta corrente do morador é incrementada com a quantia depositada																					
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Escolhe depositar uma quantia na sua conta corrente</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Pede o valor da quantia</td></tr> <tr> <td><b>3</b></td><td>Insere o valor da quantia a depositar</td><td></td></tr> <tr> <td><b>4</b></td><td></td><td>Valida o valor</td></tr> <tr> <td><b>5</b></td><td></td><td>Deposita o valor da quantia na conta corrente do morador</td></tr> <tr> <td><b>6</b></td><td></td><td>Informa sucesso</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Escolhe depositar uma quantia na sua conta corrente		<b>2</b>		Pede o valor da quantia	<b>3</b>	Insere o valor da quantia a depositar		<b>4</b>		Valida o valor	<b>5</b>		Deposita o valor da quantia na conta corrente do morador	<b>6</b>		Informa sucesso
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>	Escolhe depositar uma quantia na sua conta corrente																					
<b>2</b>		Pede o valor da quantia																				
<b>3</b>	Insere o valor da quantia a depositar																					
<b>4</b>		Valida o valor																				
<b>5</b>		Deposita o valor da quantia na conta corrente do morador																				
<b>6</b>		Informa sucesso																				
<b>Excepção 1 (Passo 4) [Valor da quantia inválido]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td></td><td>Informa que o valor da quantia inserido não é válido</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa que o valor da quantia inserido não é válido															
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>		Informa que o valor da quantia inserido não é válido																				

**Tabela 15 – Especificação do UC15 – Terminar sessão**

<b>Use Case</b>	Terminar sessão												
<b>Brief Description</b>	Permite que um morador/utilizador autenticado termine sessão na aplicação												
<b>PreConditions</b>	O utilizador está autenticado												
<b>Pos-Conditions</b>	O utilizador deixa de estar autenticado na aplicação												
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Escolhe terminar sessão</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Termina a sessão</td></tr> <tr> <td><b>4</b></td><td></td><td>Informa sucesso do término da sessão</td></tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Escolhe terminar sessão		<b>2</b>		Termina a sessão	<b>4</b>		Informa sucesso do término da sessão
	<b>Actor input</b>	<b>System Response</b>											
<b>1</b>	Escolhe terminar sessão												
<b>2</b>		Termina a sessão											
<b>4</b>		Informa sucesso do término da sessão											

**Tabela 16** – Especificação do UC16 – Definir tempo intervalo de ausência

<b>Use Case</b>	Definir tempo intervalo de tempo em que vai o morador vai estar ausente																					
<b>Brief Description</b>	Permite que o morador possa definir um intervalo de tempo em que vai estar ausente																					
<b>PreConditions</b>	O morador encontra-se autenticado																					
<b>Pos-Conditions</b>	O intervalo de tempo de ausência foi registado																					
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td>Morador escolhe definir um intervalo de tempo de ausência</td> <td></td> </tr> <tr> <td><b>2</b></td> <td></td> <td>Pede intervalo de tempo</td> </tr> <tr> <td><b>3</b></td> <td>Fornece intervalo de tempo</td> <td></td> </tr> <tr> <td><b>4</b></td> <td></td> <td>Valida intervalo de tempo</td> </tr> <tr> <td><b>5</b></td> <td></td> <td>Regista o intervalo de tempo de ausência no morador</td> </tr> <tr> <td><b>6</b></td> <td></td> <td>Informa sucesso</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Morador escolhe definir um intervalo de tempo de ausência		<b>2</b>		Pede intervalo de tempo	<b>3</b>	Fornece intervalo de tempo		<b>4</b>		Valida intervalo de tempo	<b>5</b>		Regista o intervalo de tempo de ausência no morador	<b>6</b>		Informa sucesso
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>	Morador escolhe definir um intervalo de tempo de ausência																					
<b>2</b>		Pede intervalo de tempo																				
<b>3</b>	Fornece intervalo de tempo																					
<b>4</b>		Valida intervalo de tempo																				
<b>5</b>		Regista o intervalo de tempo de ausência no morador																				
<b>6</b>		Informa sucesso																				
<b>Excepção 1 (Passo 4) [Intervalo de tempo inválido]</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td></td> <td>Informa que o intervalo de tempo inserido é inválido</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>		Informa que o intervalo de tempo inserido é inválido															
	<b>Actor input</b>	<b>System Response</b>																				
<b>1</b>		Informa que o intervalo de tempo inserido é inválido																				

**Tabela 17** – Especificação do UC17 – Consultar detalhes de apartamento

<b>Use Case</b>	Consultar dados do apartamento									
<b>Brief Description</b>	Permite que o morador possa consultar os dados do apartamento									
<b>PreConditions</b>	O morador encontra-se autenticado									
<b>Pos-Conditions</b>	O morador consulta os dados do apartamento									
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th><b>Actor input</b></th> <th><b>System Response</b></th> </tr> </thead> <tbody> <tr> <td><b>1</b></td> <td>Morador escolhe consultar os dados do apartamento</td> <td></td> </tr> <tr> <td><b>2</b></td> <td></td> <td>Apresenta os dados do apartamento</td> </tr> </tbody> </table>		<b>Actor input</b>	<b>System Response</b>	<b>1</b>	Morador escolhe consultar os dados do apartamento		<b>2</b>		Apresenta os dados do apartamento
	<b>Actor input</b>	<b>System Response</b>								
<b>1</b>	Morador escolhe consultar os dados do apartamento									
<b>2</b>		Apresenta os dados do apartamento								

**Tabela 18** – Especificação do UC18 – Alterar detalhes de apartamento

<b>Use Case</b>	Alterar dados do apartamento																						
<b>Brief Description</b>	Permite que o morador possa modificar dados do apartamento																						
<b>PreConditions</b>	O morador encontra-se autenticado																						
<b>Pos-Conditions</b>	Os dados do apartamento são modificados																						
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th>Actor input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Morador escolhe modificar os dados do apartamento</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Pede novos dados</td></tr> <tr> <td><b>3</b></td><td>Insere os novos dados</td><td></td></tr> <tr> <td><b>4</b></td><td></td><td>Valida os dados inseridos</td></tr> <tr> <td><b>5</b></td><td></td><td>Atualiza os dados inseridos do apartamento</td></tr> <tr> <td><b>6</b></td><td></td><td>Informa que os dados foram atualizados com sucesso</td></tr> </tbody> </table>			Actor input	System Response	<b>1</b>	Morador escolhe modificar os dados do apartamento		<b>2</b>		Pede novos dados	<b>3</b>	Insere os novos dados		<b>4</b>		Valida os dados inseridos	<b>5</b>		Atualiza os dados inseridos do apartamento	<b>6</b>		Informa que os dados foram atualizados com sucesso
	Actor input	System Response																					
<b>1</b>	Morador escolhe modificar os dados do apartamento																						
<b>2</b>		Pede novos dados																					
<b>3</b>	Insere os novos dados																						
<b>4</b>		Valida os dados inseridos																					
<b>5</b>		Atualiza os dados inseridos do apartamento																					
<b>6</b>		Informa que os dados foram atualizados com sucesso																					
<b>Excepção 1 (Passo 4) [Dados inválidos]</b>	<table border="1"> <thead> <tr> <th></th> <th>Actor input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td></td><td>Informa que os dados inseridos são inválidos</td></tr> </tbody> </table>			Actor input	System Response	<b>1</b>		Informa que os dados inseridos são inválidos															
	Actor input	System Response																					
<b>1</b>		Informa que os dados inseridos são inválidos																					

**Tabela 19** – Especificação do UC19 – Consultar despesas

<b>Use Case</b>	Consultar despesas										
<b>Brief Description</b>	Permite a um morador consultar todas as despesas										
<b>PreConditions</b>	O morador encontra-se autenticado										
<b>Pos-Conditions</b>	O morador consulta as despesas existentes										
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th>Actor input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Morador escolhe consultar mas despesas</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Apresenta as despesas</td></tr> </tbody> </table>			Actor input	System Response	<b>1</b>	Morador escolhe consultar mas despesas		<b>2</b>		Apresenta as despesas
	Actor input	System Response									
<b>1</b>	Morador escolhe consultar mas despesas										
<b>2</b>		Apresenta as despesas									

**Tabela 20** – Especificação do UC20 – Consultar saldo global do apartamento

<b>Use Case</b>	Consultar saldo global do apartamento													
<b>Brief Description</b>	Permite que o morador possa consultar o saldo global do apartamento													
<b>PreConditions</b>	O morador encontra-se autenticado													
<b>Pos-Conditions</b>	O morador consulta o saldo global do apartamento													
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th>Actor input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Morador escolhe consultar saldo global do apartamento</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Calcula o saldo global do apartamento</td></tr> <tr> <td><b>3</b></td><td></td><td>Apresenta o saldo global do apartamento</td></tr> </tbody> </table>		Actor input	System Response	<b>1</b>	Morador escolhe consultar saldo global do apartamento		<b>2</b>		Calcula o saldo global do apartamento	<b>3</b>		Apresenta o saldo global do apartamento	
	Actor input	System Response												
<b>1</b>	Morador escolhe consultar saldo global do apartamento													
<b>2</b>		Calcula o saldo global do apartamento												
<b>3</b>		Apresenta o saldo global do apartamento												

**Tabela 21** – Especificação do UC21 – Consultar o estado do saldo dos moradores

<b>Use Case</b>	Consultar o estado do saldo dos moradores													
<b>Brief Description</b>	Permite que o morador possa consultar o estado do saldo dos restantes moradores													
<b>PreConditions</b>	O morador encontra-se autenticado													
<b>Pos-Conditions</b>	O morador consulta o estado dos saldos dos restantes moradores													
<b>Flow of Events</b>	<table border="1"> <thead> <tr> <th></th> <th>Actor input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td><b>1</b></td><td>Morador escolhe consultar o estado do saldo dos restantes moradores</td><td></td></tr> <tr> <td><b>2</b></td><td></td><td>Calcula o estado do saldo dos restantes moradores</td></tr> <tr> <td><b>3</b></td><td></td><td>Apresenta o estado do saldo dos outros moradores</td></tr> </tbody> </table>		Actor input	System Response	<b>1</b>	Morador escolhe consultar o estado do saldo dos restantes moradores		<b>2</b>		Calcula o estado do saldo dos restantes moradores	<b>3</b>		Apresenta o estado do saldo dos outros moradores	
	Actor input	System Response												
<b>1</b>	Morador escolhe consultar o estado do saldo dos restantes moradores													
<b>2</b>		Calcula o estado do saldo dos restantes moradores												
<b>3</b>		Apresenta o estado do saldo dos outros moradores												

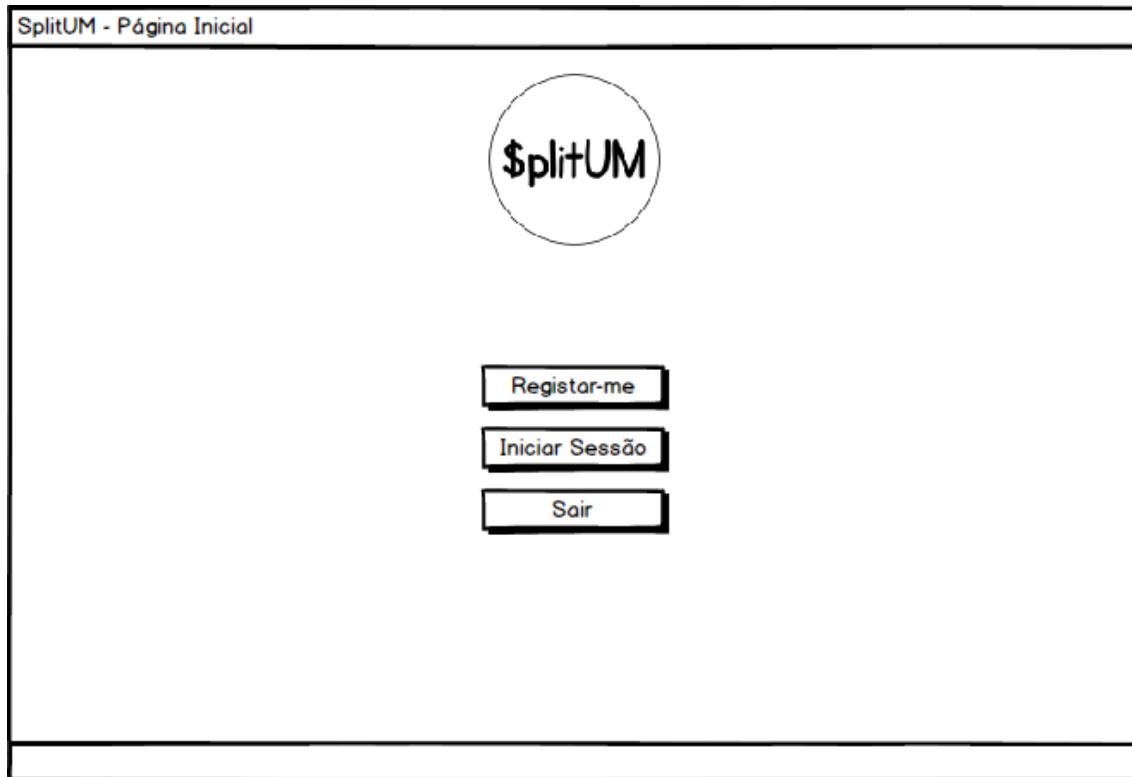
## 5. Desenho da Interface

Com vista à obtenção de uma interface , para uma utilização fácil e mais interativa da aplicação por parte dos moradores que partilham despesas de um apartamento, baseada nos requisitos dos moradores, realizou-se um esboço da futura interface (Mockups), de seguida especificou-se a máquina de estados correspondente à interface, e por fim justificou-se as opções tomadas relativamente aos tipo de interface que se deseja construir.

### 5.1. Mockups

Um mockup é um modelo em escala ou de tamanho real de um projeto ou dispositivo, usado para ensino, demonstração, avaliação de design, promoção e outros propósitos. Em engenharia de software a aplicação mais comum dos mockups é para a criação de interfaces para o utilizador, de modo a mostrar ao utilizador que a interface será semelhante com mockups, sem com isto ter de construir o software em si ou a funcionalidade subjacente.

De seguida apresentam-se alguns exemplos dos mockups da interface mais relevantes.



**Figura 3 – Mockup do Menu Inicial**

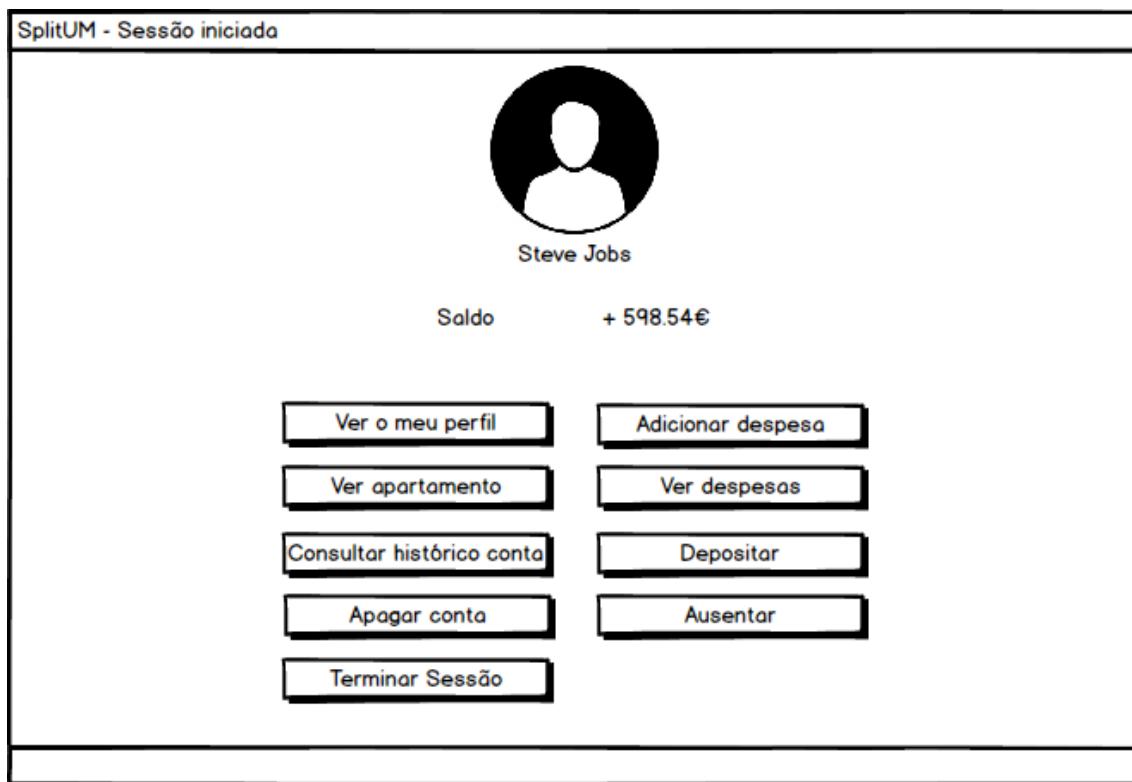


Figura 4 – Mockup do Menu Morador autenticado

SplitUM - O apartamento

« Voltar

Steve Jobs

Morada      Código Postal

Quinta das Tílias N°53 - 3ºEsq      4925-874

Editar Morada

Moradores

Moradores	Saldo	Dívidas
Ricardo Silva	20€	--
Ana Maria	-40€	Steve Jobs (20€), Ricardo (20€)
José Manuel	0€	--
Steve Jobs	9€	--

Saldo Total: 300€

Figura 5 – Mockup do Menu ver apartamento

SplitUM - Histórico da conta corrente

[Voltar](#)



Steve Jobs

Histórico da conta corrente

Nome	Data	Movimento
Renda	2014/04/04	+ 25€
Água	2014/03/28	- 37€
		-12€

Saldo

Figura 6 – Mockup do Menu consultar histórico da conta

SplitUM - Adicionar despesa



Steve Jobs

Descrição da despesa

Valor da despesa

Tipo da despesa

Recorrente

Extraordinária

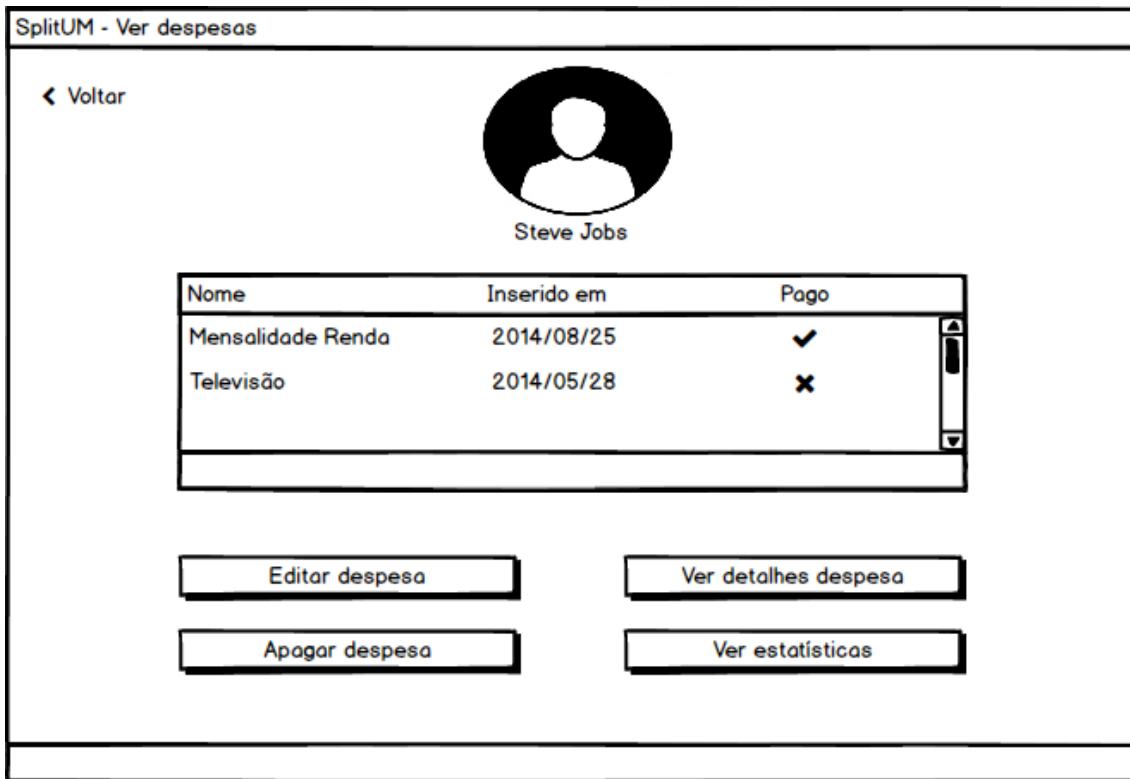
Divisão da despesa

Por valor

Percentual

[Continuar](#) [Cancelar](#)

Figura 7 – Mockup do Menu adicionar despesa



**Figura 8 – Mockup do Menu ver despesas**

## 5.2. Máquinas de estado

As máquinas de estado permitem modelar o comportamento de um dado objecto/sistema de forma global, isto é, modela-se todos os estados possíveis que o objecto/sistema atravessa em resposta aos eventos que podem ocorrer. Neste caso utilizou-se as máquinas de estado para modelar o comportamento da interface com o utilizador.

De forma a modelar o comportamento da interface com o utilizador, especificou-se as máquinas de estado:

- Login/Registrar
- Morador Autenticado
- Adicionar/Editar Despesa
- Ver Despesa

A máquina de estado da interface foi subdividida em 4 máquinas de estado de modo a facilitar a sua compreensão.

## Máquina Estado - Login/Register

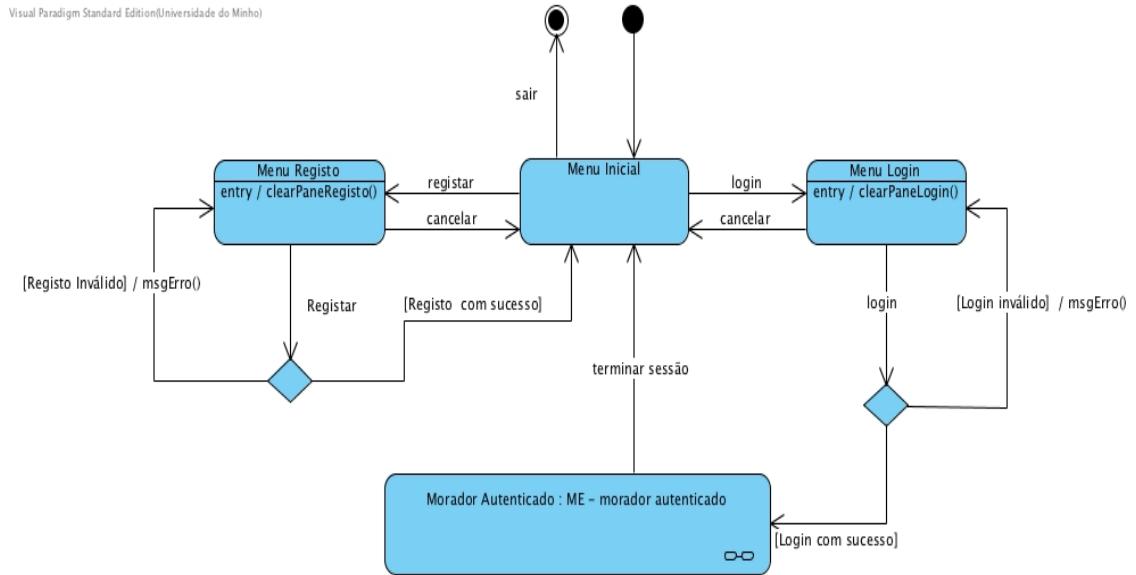


Figura 9 – Maquina de Estado Login/Register

## Máquina Estado - Morador Autênticado

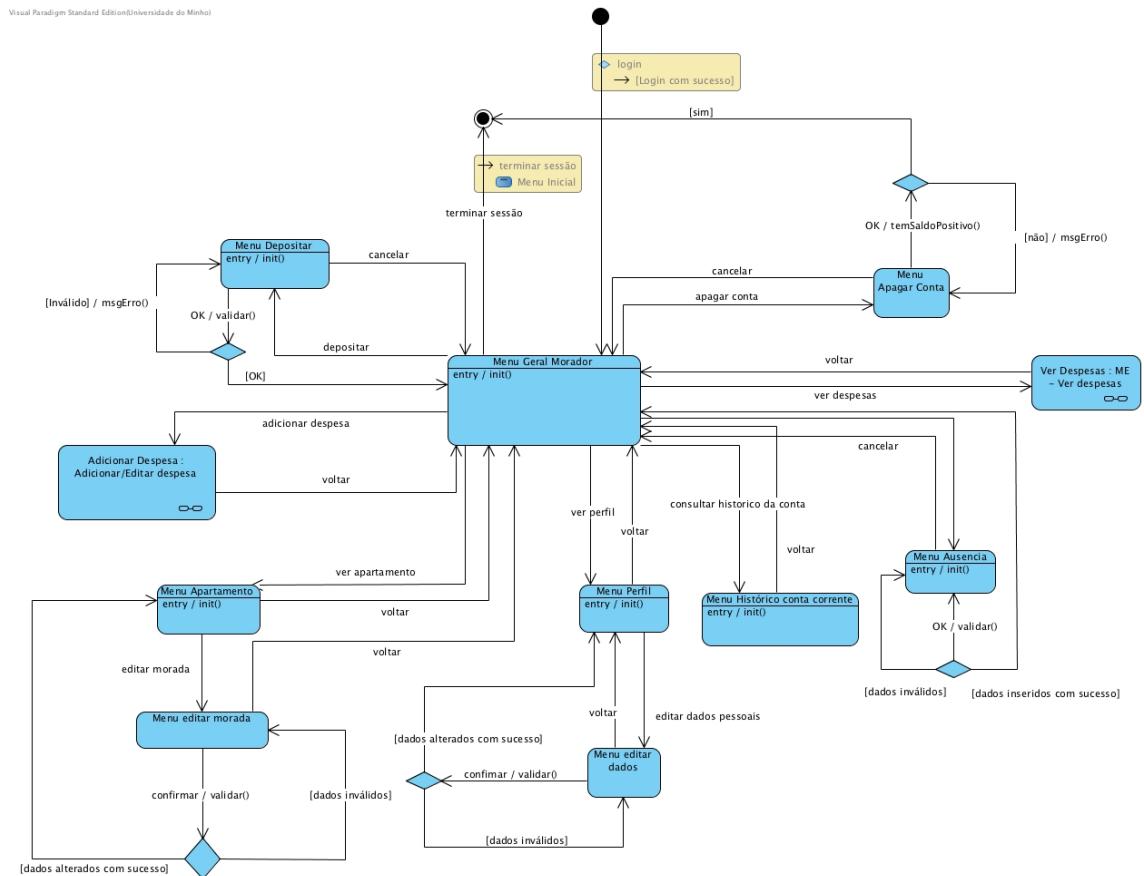
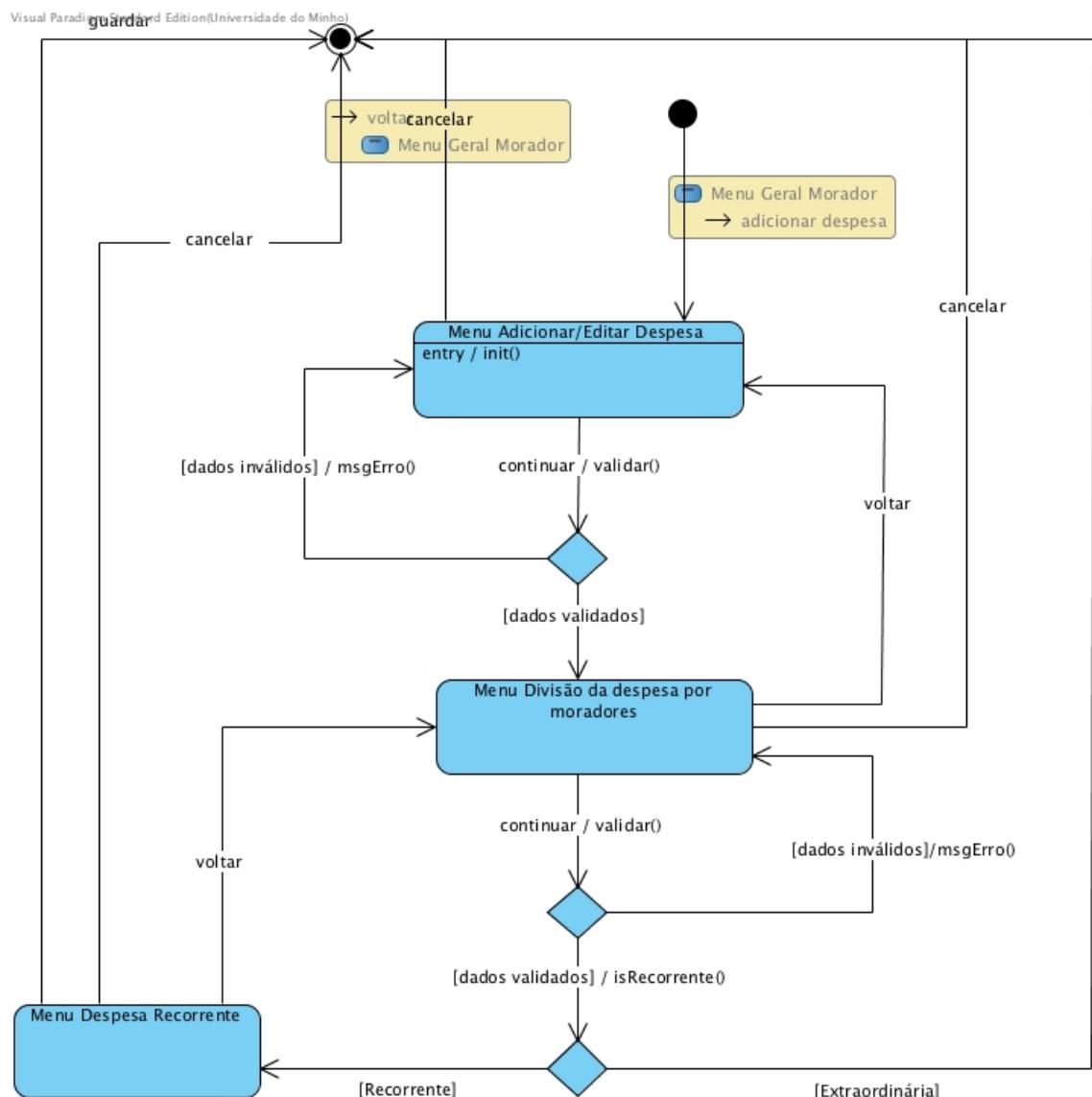


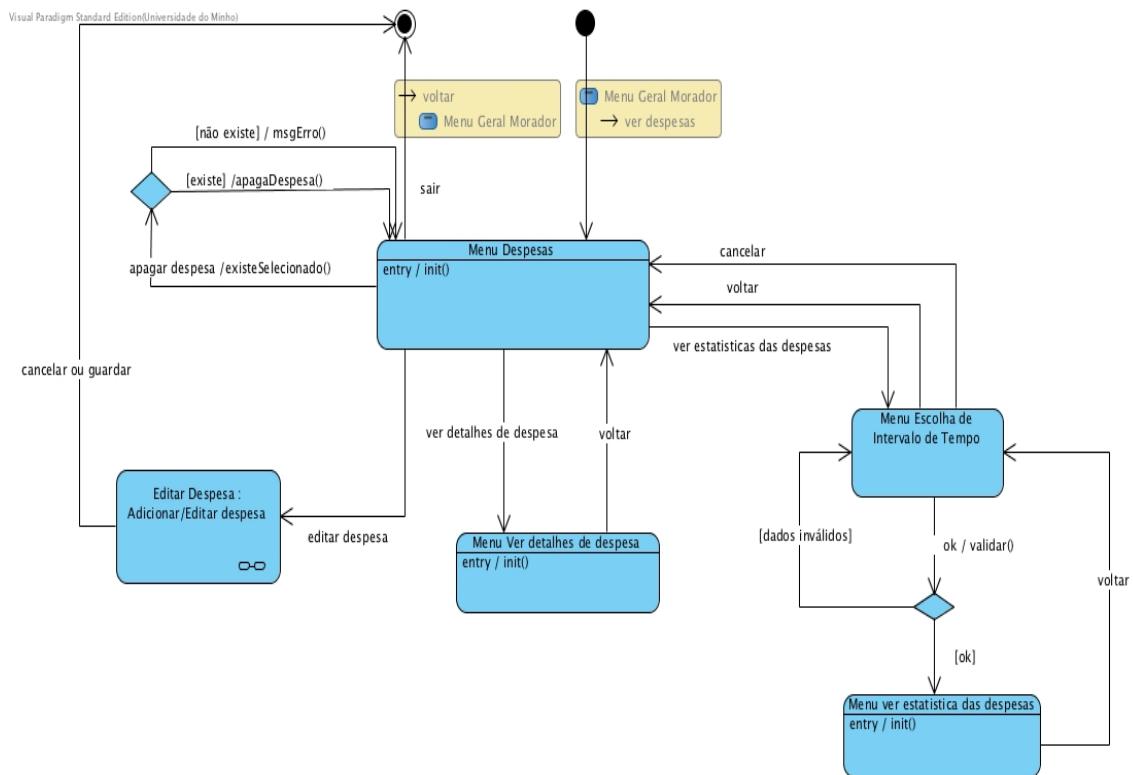
Figura 10 – Maquina de Estado – Morador Autênticado

## Máquina Estado – Adicionar/Editar Despesa



**Figura 11 – Maquina de Estado – Adicionar/Editar despesa**

## Máquina Estado - Ver Despesa



**Figura 12 – Maquina de Estado – ver despesa**

### 5.3. Experiência de utilização

Um dos objectivos que se visa alcançar com a futura interface é que esta permita ao utilizador ter uma experiência de utilização agradável de fácil aprendizagem. Portanto, optou-se por ter vários menus com pouca informação, de modo a não agregar demasiada informação em só alguns menus.

Teve-se também o cuidado em possibilitar que o utilizador consiga voltar sempre ao menu anterior, seja pela botão “Voltar” no topo da janela do lado esquerdo, seja por um botão “Cancelar” dedicado, quando este faz mais sentido para o caso concreto.

O tipo de letra usado também foi escolhido de modo a ser de fácil visualização para o morador, no neste caso, tamanho de letra 16. Também o tamanho das linhas das tabelas foram alargados para facilitar a sua leitura e manter o aspetto leve e simplificado da aplicação.

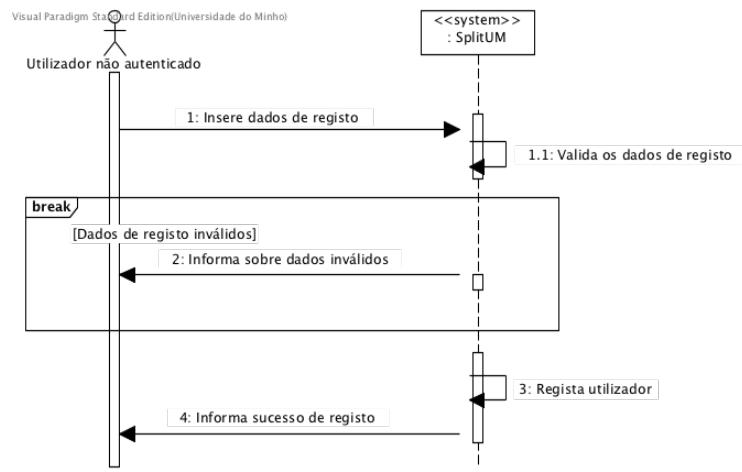
Por último, tentou-se desenhar uma interface “minimalista”, onde temos pouco informação em cada menu, em vez da alternativa onde se usa pouco menus, mas com uma alta densidade de informação.

## 6. Diagramas de sequência de sistema e de subsistemas dos *Use Cases*

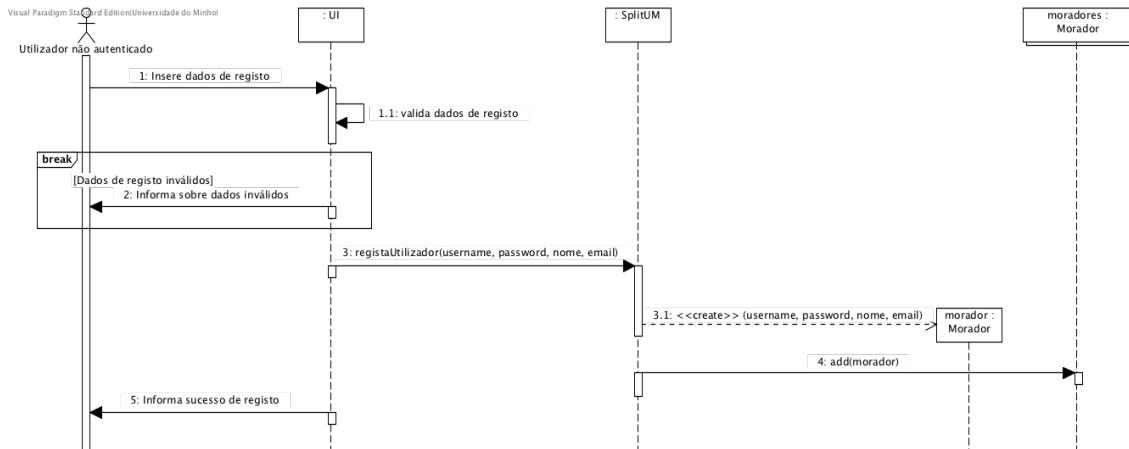
A próxima etapa do desenvolvimento da aplicação SplitUM consiste em passar para a análise e concepção da solução a desenvolver para, posteriormente, implementar. Definir a solução deve consistir na sua arquitectura e na lógica de funcionamento.

Numa primeira etapa desenvolveram-se os diagramas de sequência de sistema para cada use case. Desta forma, define-se e especifica-se cada use case.

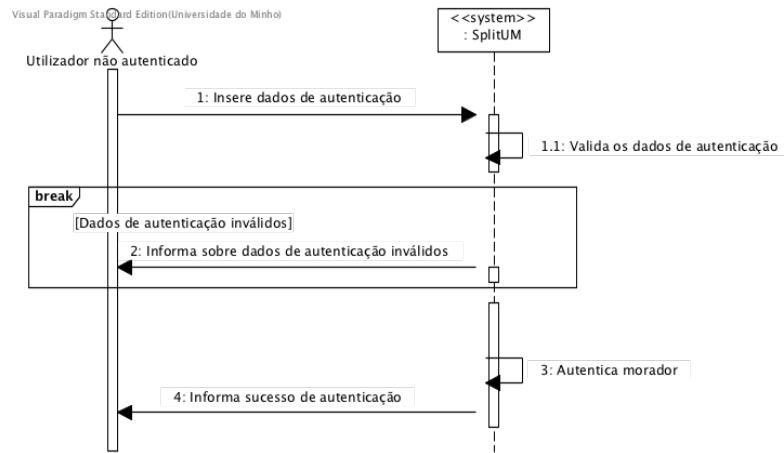
O sistema representado pelo SplitUM apresenta-se como uma “black box” que necessita de ser dividida em partições do sistema, ou subsistemas. Apresentam-se de seguida os diagramas de sequência sistema e de subsistemas para cada use cases.



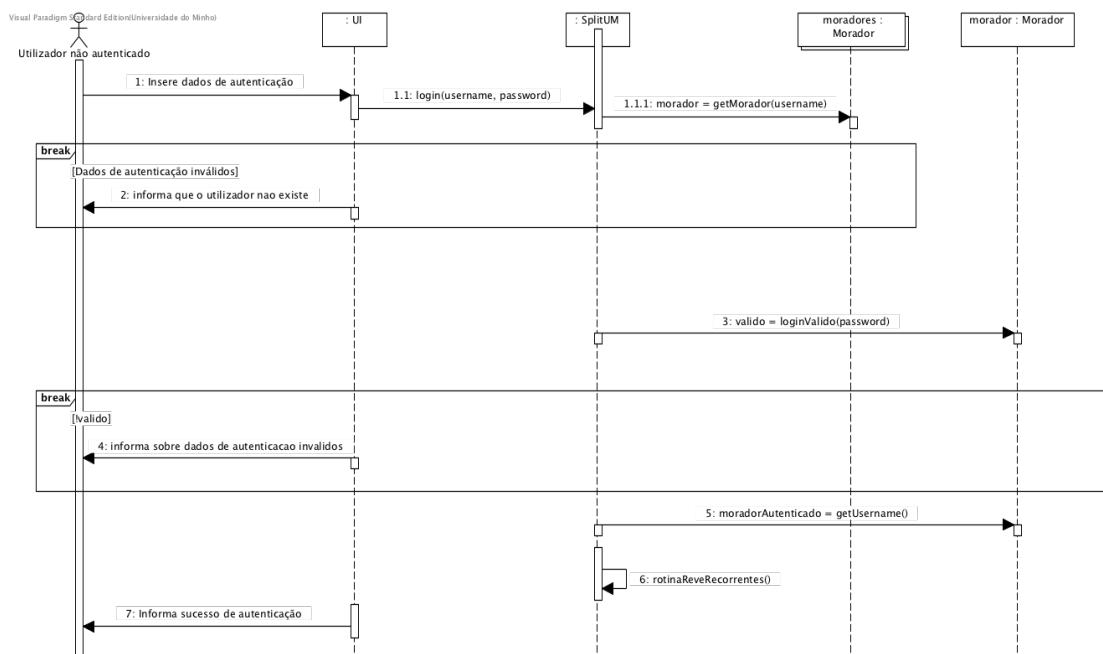
**Figura 13 – Diagrama de sequência de sistema do UC1 – Registar na aplicação**



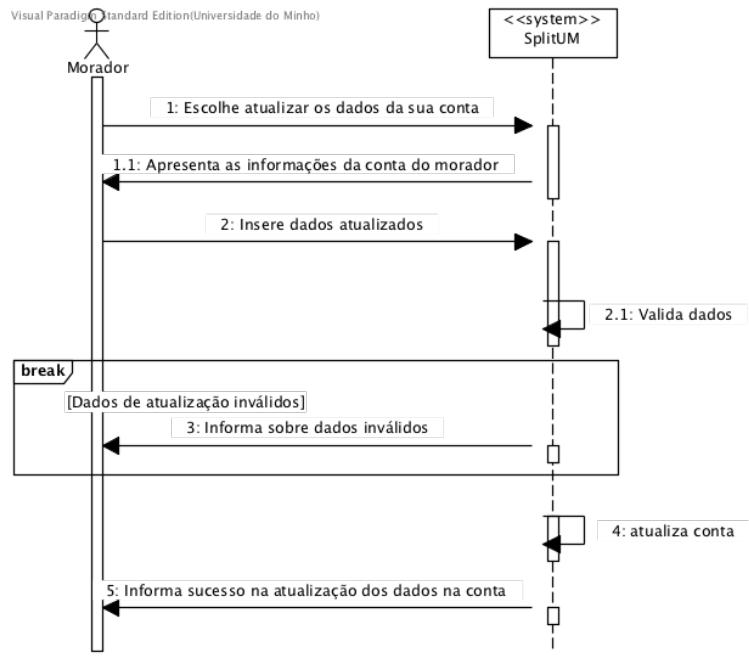
**Figura 14 – Diagrama de sequência de subsistemas do UC1 – Registar na aplicação**



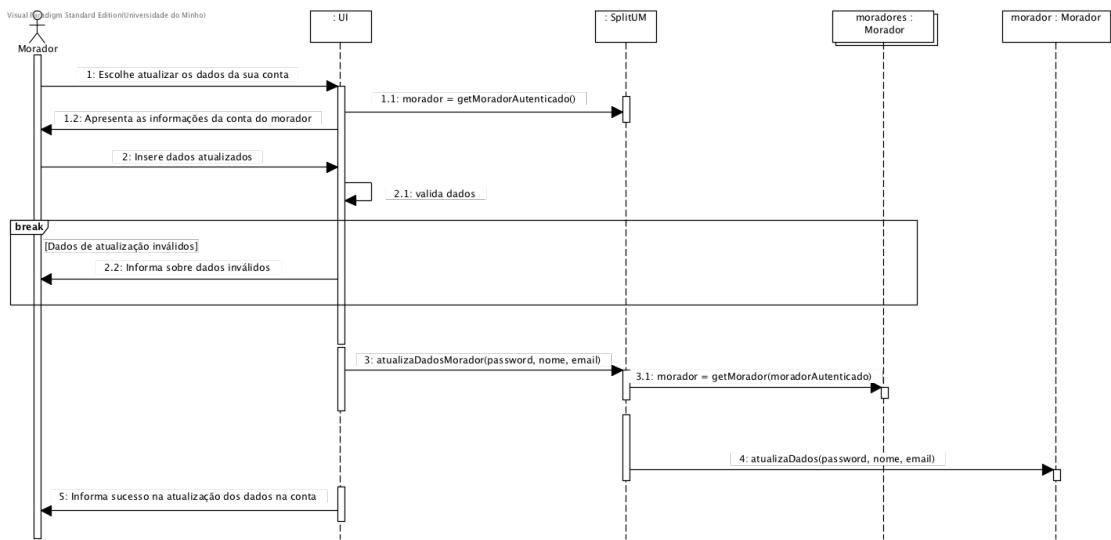
**Figura 15 – Diagrama de sequência de sistema do UC2 – Autenticar na aplicação**



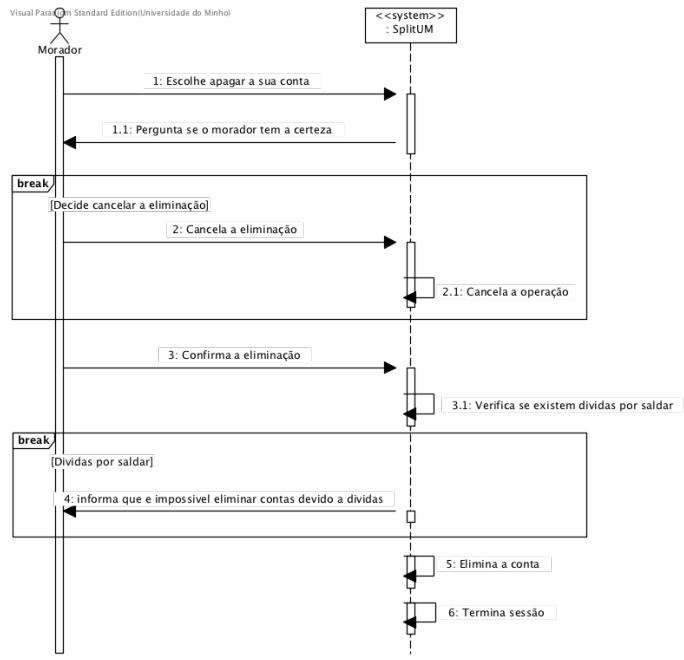
**Figura 16 – Diagrama de sequência de subsistemas do UC2 – Autenticar na aplicação**



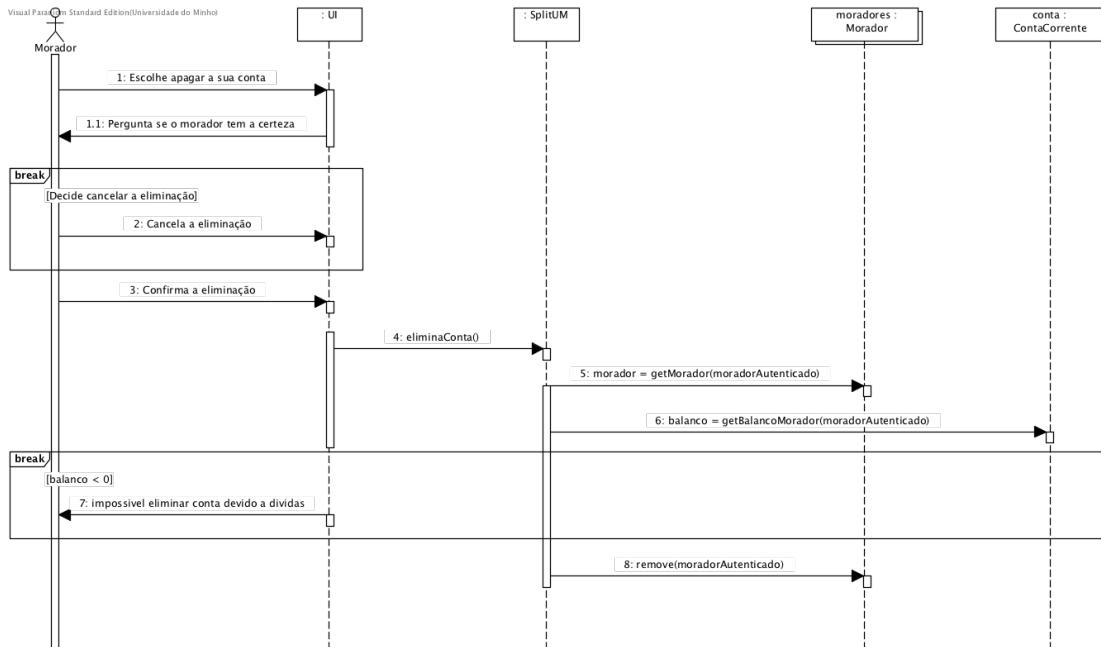
**Figura 17 – Diagrama de sequência de sistema do UC3 – Atualizar conta**



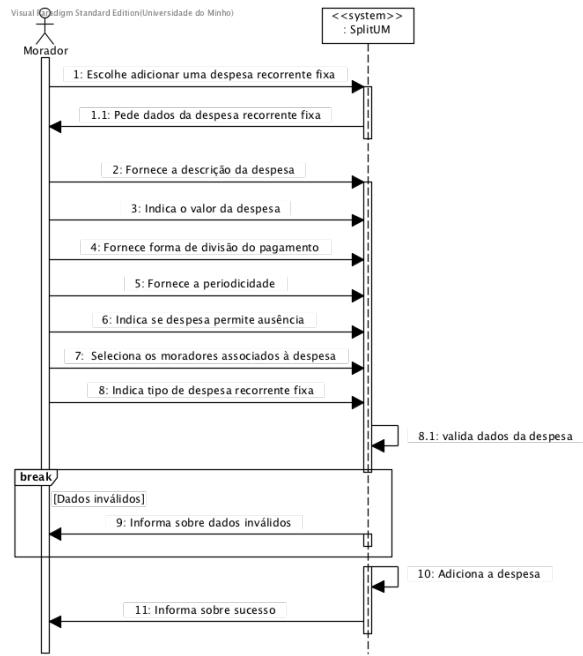
**Figura 18 – Diagrama de sequência de subsistemas do UC3 – Atualizar conta**



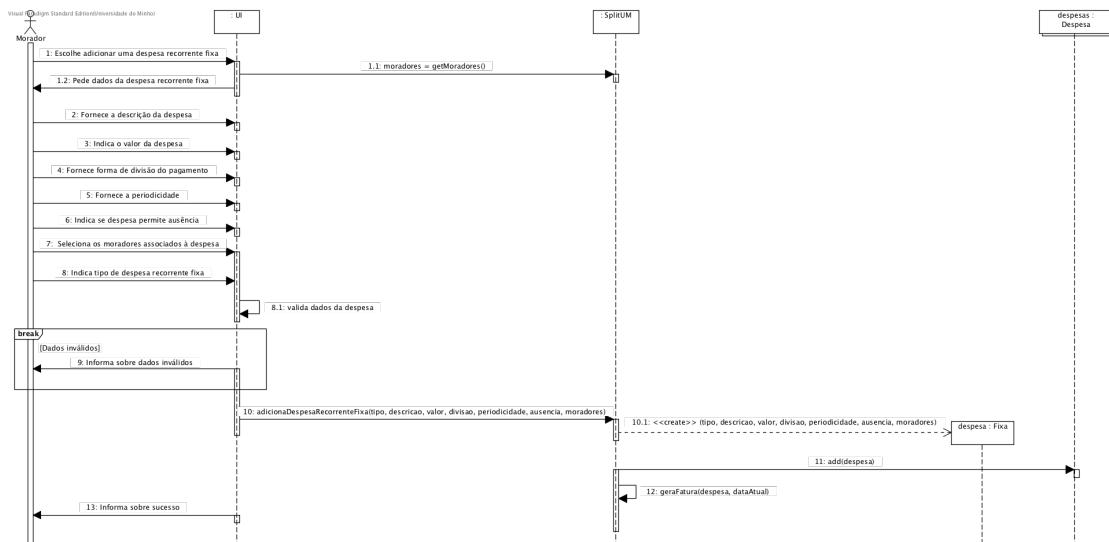
**Figura 19 – Diagrama de sequência de sistema do UC4 – Apagar conta**



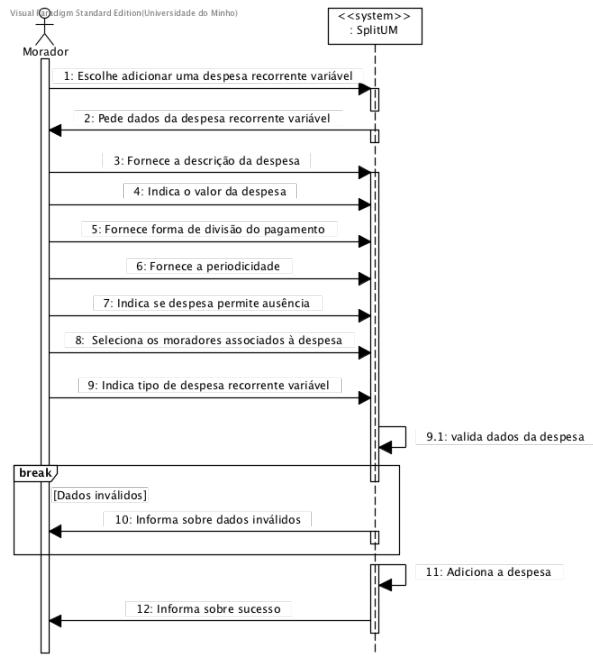
**Figura 20 – Diagrama de sequência de subsistemas do UC4 – Apagar conta**



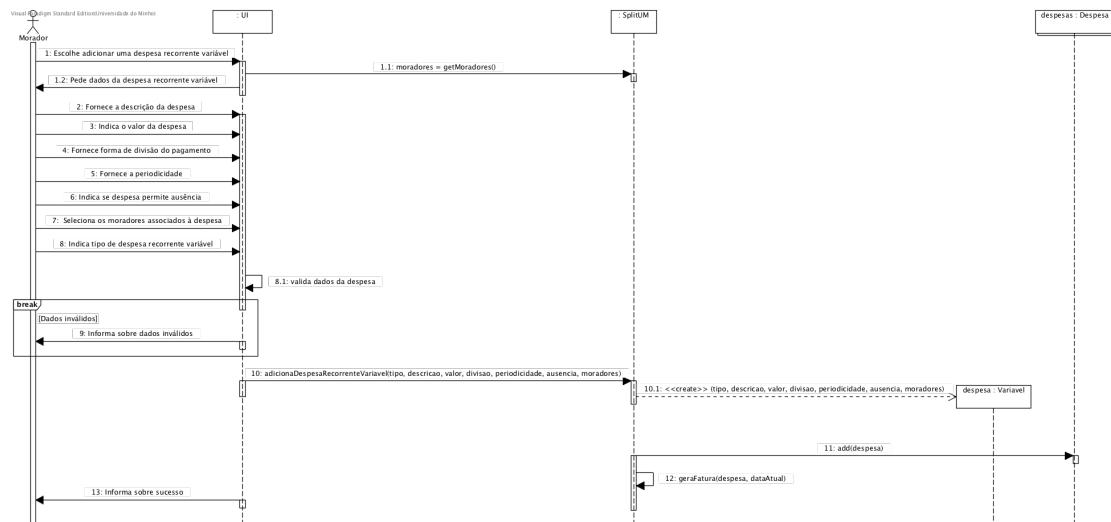
**Figura 21 – Diagrama de sequência de sistema do UC5 – Adicionar despesa recorrente fixa**



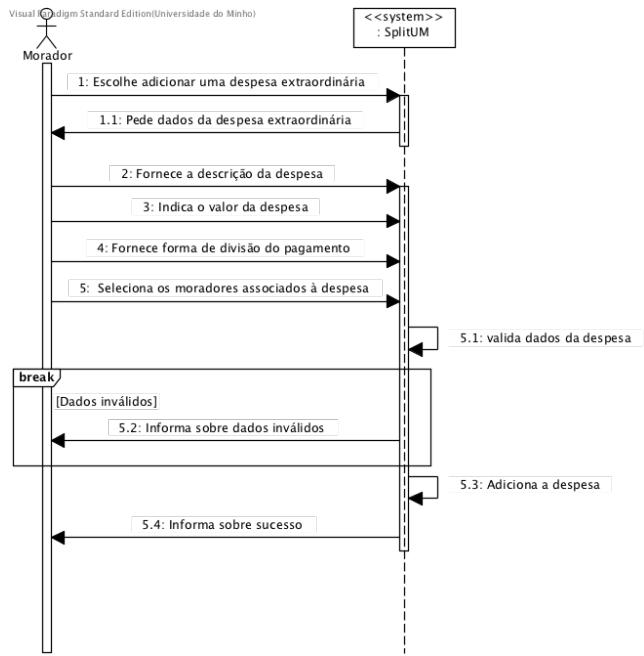
**Figura 22 – Diagrama de sequência de subsistemas do UC5 – Adicionar despesa recorrente fixa**



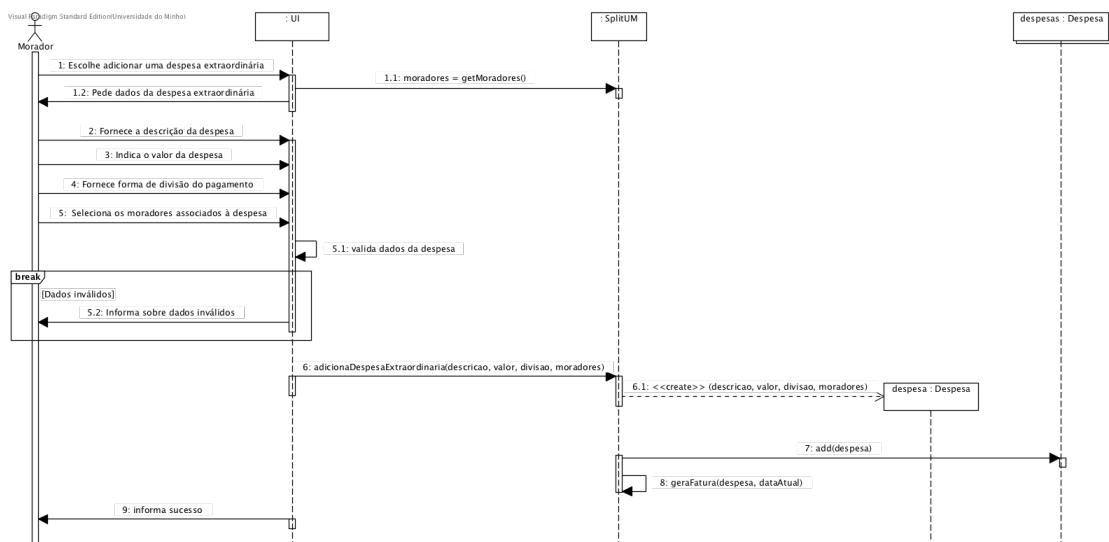
**Figura 23 – Diagrama de sequência de sistema do UC6 – Adicionar despesa recorrente variável**



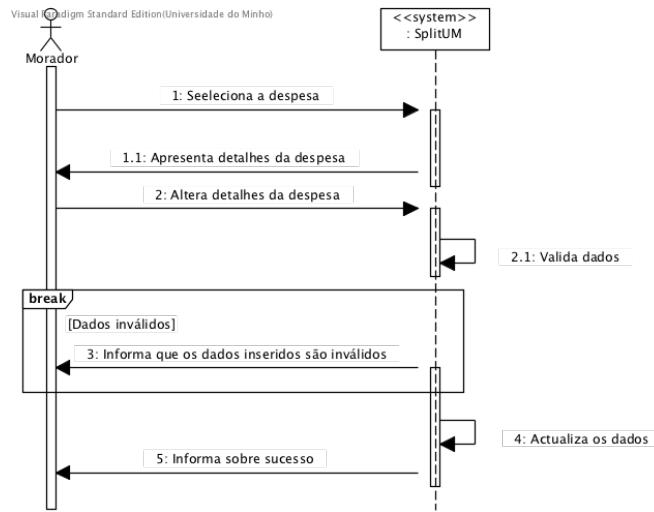
**Figura 24 – Diagrama de sequência de subsistemas do UC6 – Adicionar despesa recorrente variável**



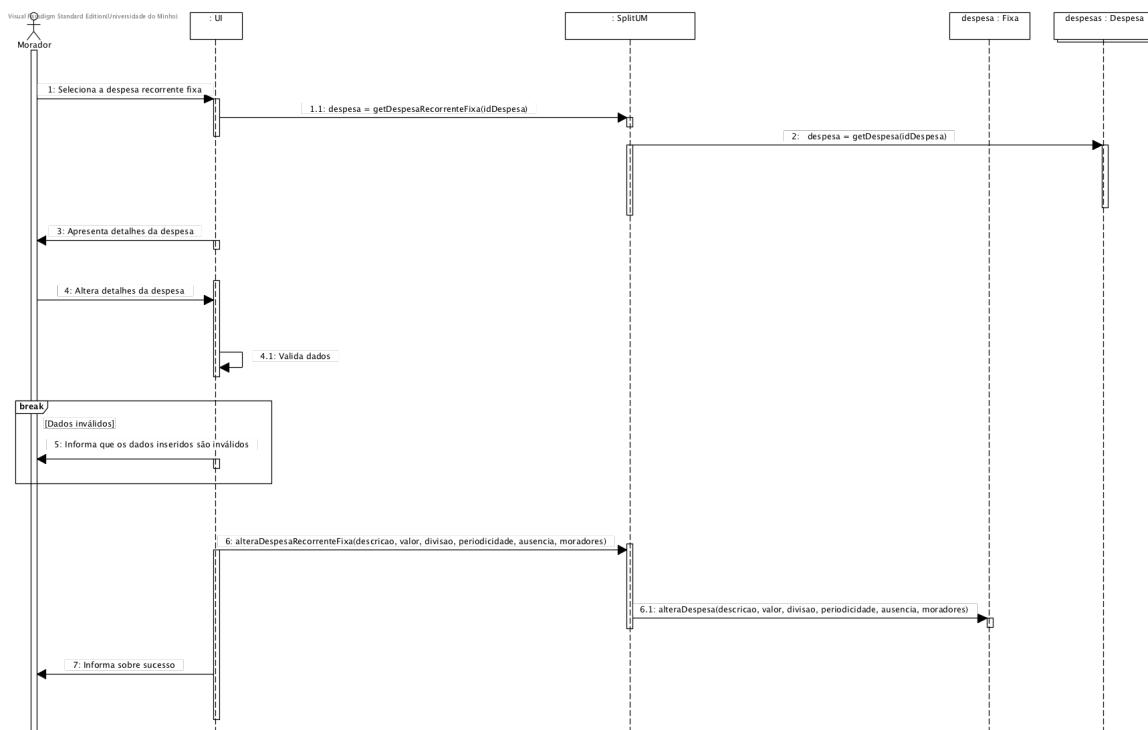
**Figura 25 – Diagrama de sequência de sistema do UC7 – Adicionar despesa extraordinária**



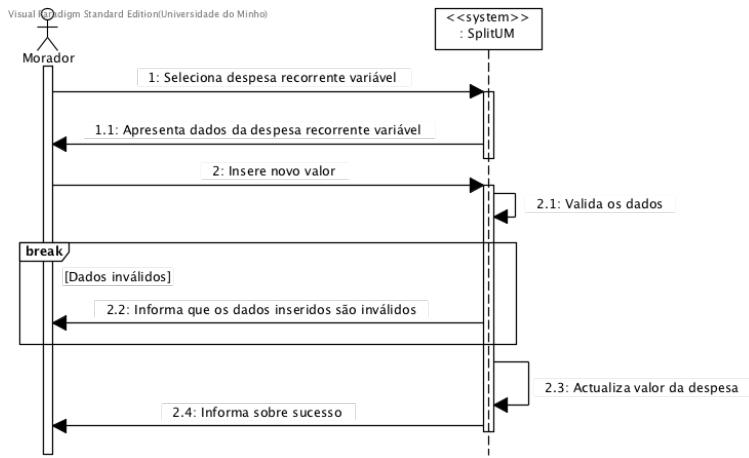
**Figura 26 – Diagrama de sequência de subsistemas do UC7 – Adicionar despesa extraordinária**



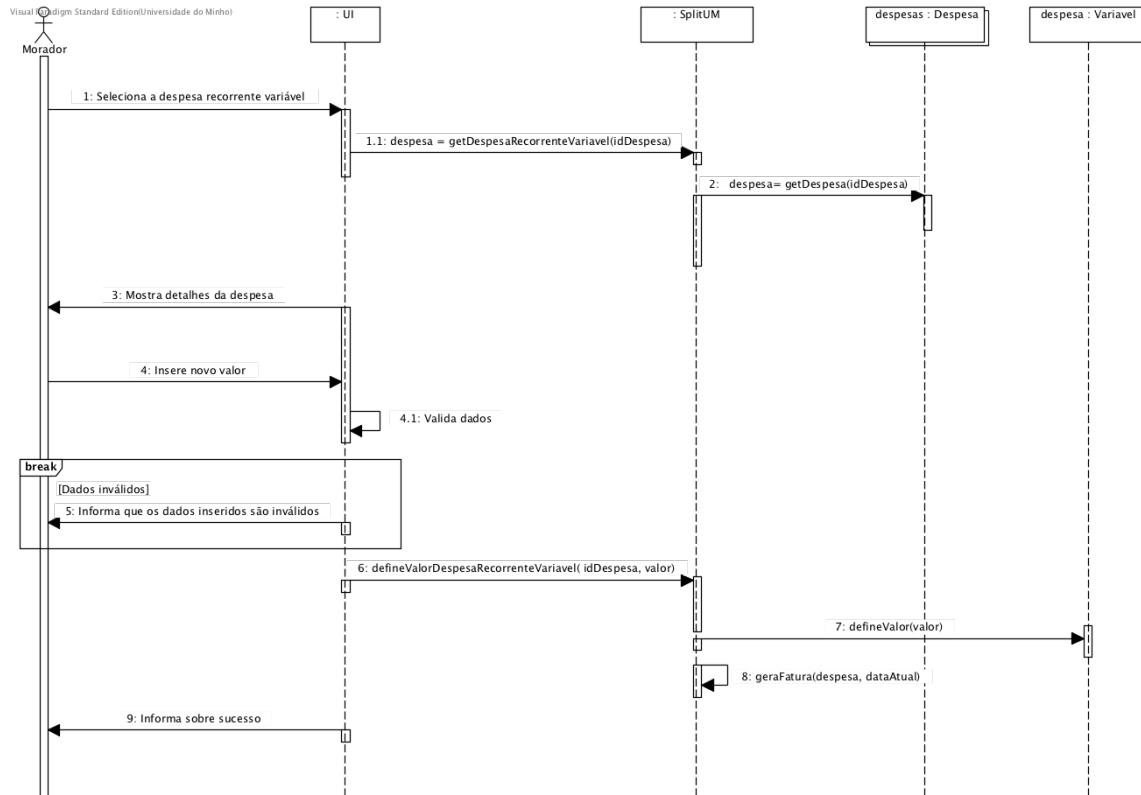
**Figura 27 – Diagrama de sequência de sistema do UC8 – Adicionar detalhes de uma despesa recorrente fixa**



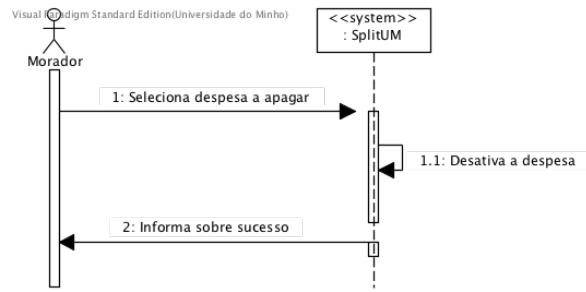
**Figura 28 – Diagrama de sequência de subsistemas do UC8 – Adicionar detalhes de uma despesa recorrente fixa**



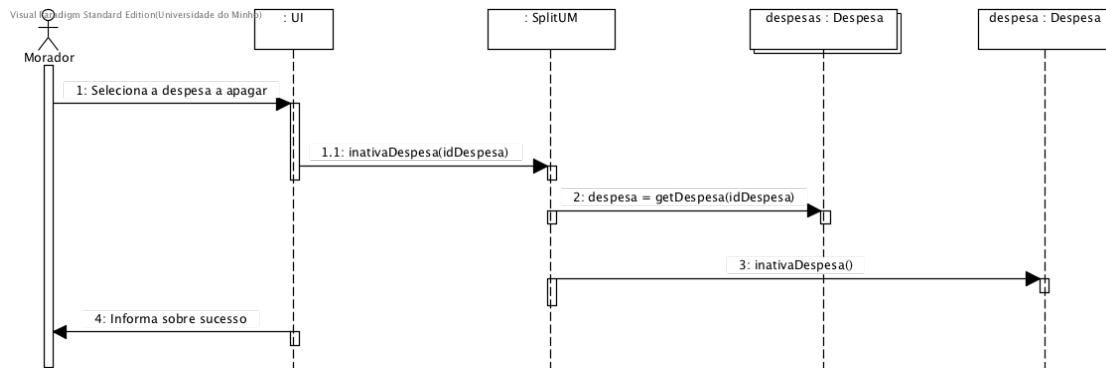
**Figura 29 – Diagrama de sequência de sistema do UC9 – Definir valor atual da despesa recorrente variável**



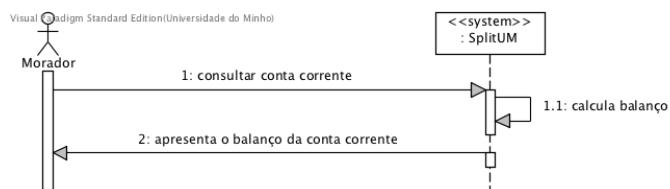
**Figura 30 – Diagrama de sequência de subsistemas do UC9 – Definir valor atual da despesa recorrente variável**



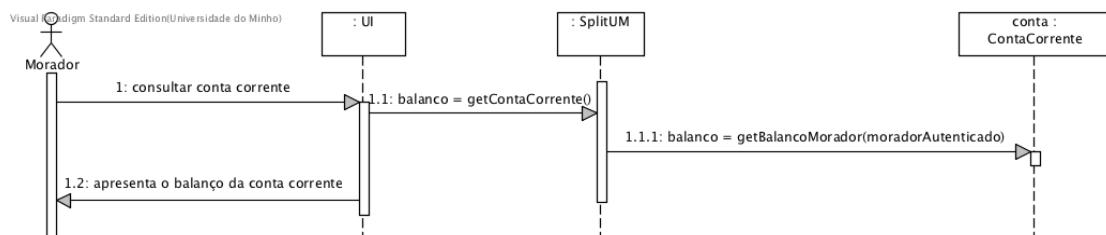
**Figura 31 – Diagrama de sequência de sistema do UC10 – Apagar/desativar despesa**



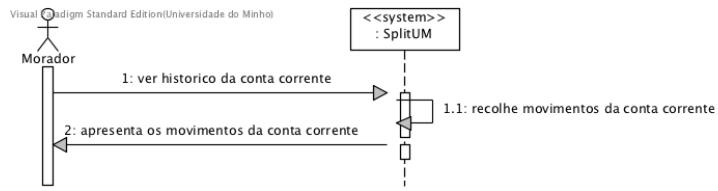
**Figura 32 – Diagrama de sequência de subsistemas do UC10 – Apagar/desativar despesa**



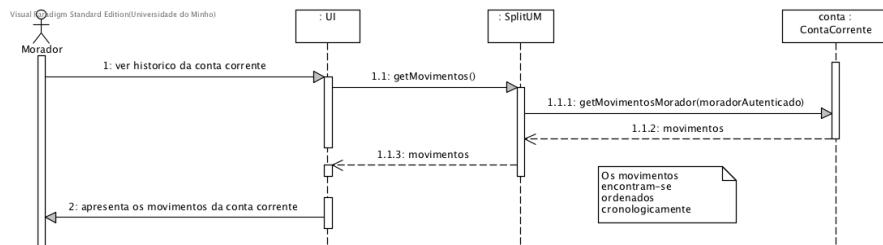
**Figura 33 – Diagrama de sequência de sistema do UC11 – Consultar conta corrente do morador**



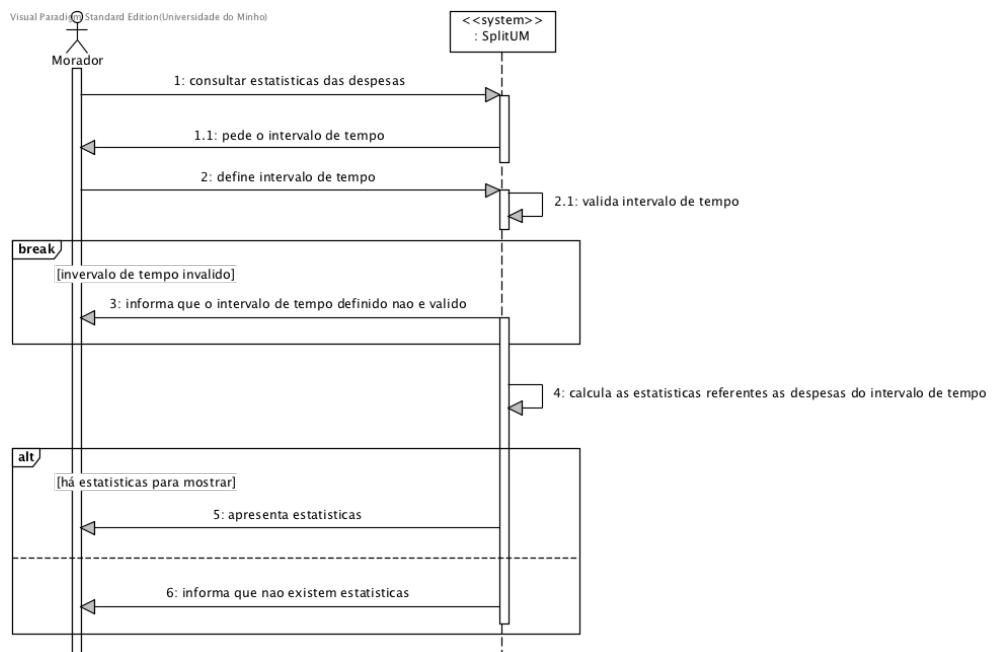
**Figura 34 – Diagrama de sequência de subsistemas do UC11 - Consultar conta corrente do morador**



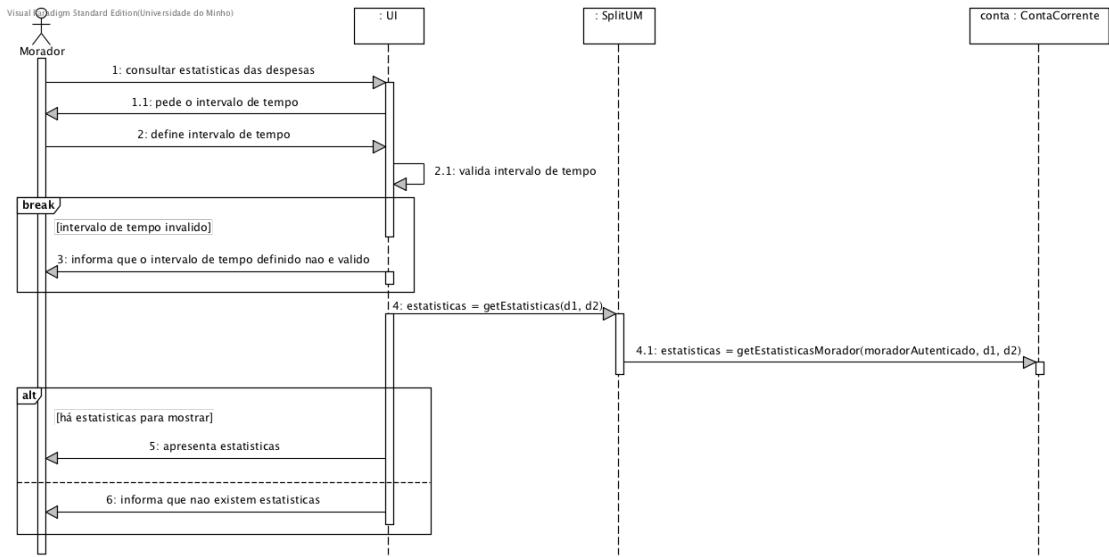
**Figura 35 – Diagrama de sequência de sistema do UC12 – Consultar histórico/movimentos da conta corrente do morador**



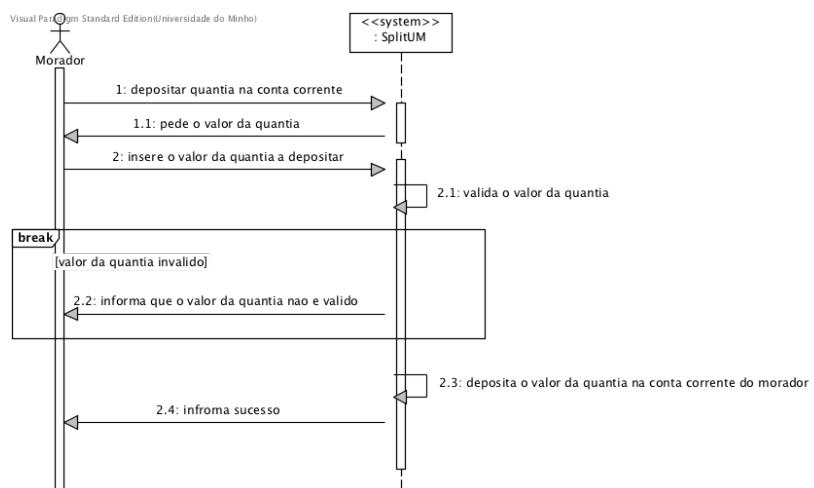
**Figura 36 – Diagrama de sequência de subsistemas do UC12 - Consultar histórico/movimentos da conta corrente do morador**



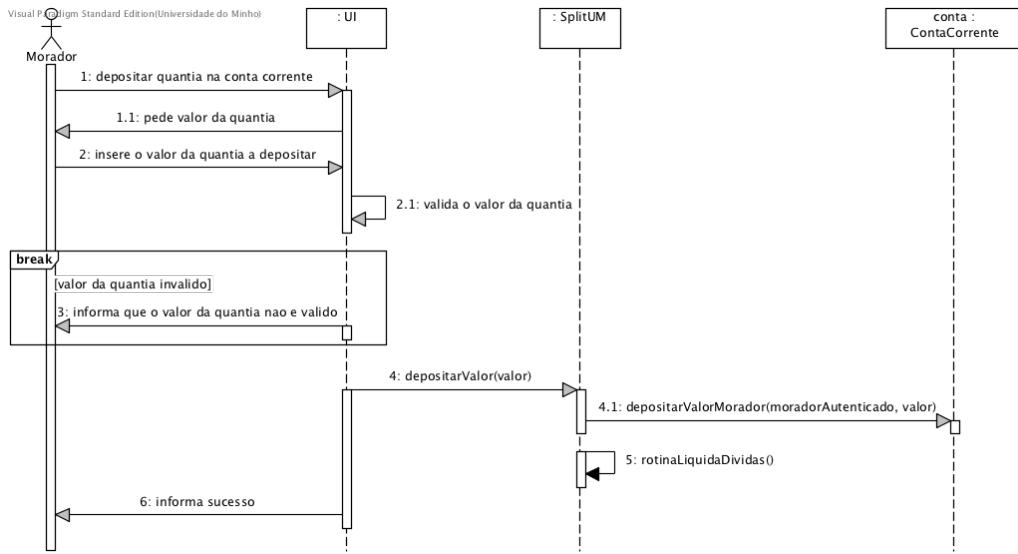
**Figura 37 – Diagrama de sequência de sistema do UC13 – Consultar estatísticas das despesas**



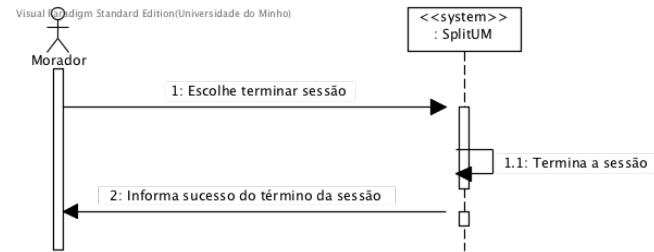
**Figura 38 – Diagrama de sequência de subsistemas do UC13 - Consultar estatísticas das despesas**



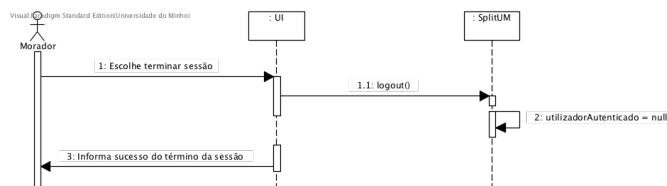
**Figura 39 – Diagrama de sequência de sistema do UC14 – Depositar quantia na conta**



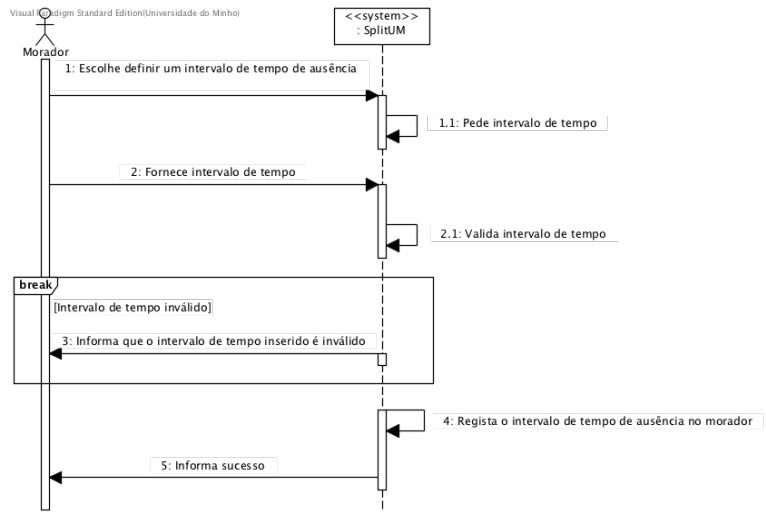
**Figura 40 – Diagrama de sequência de subsistemas do UC14 – Depositar quantia na conta**



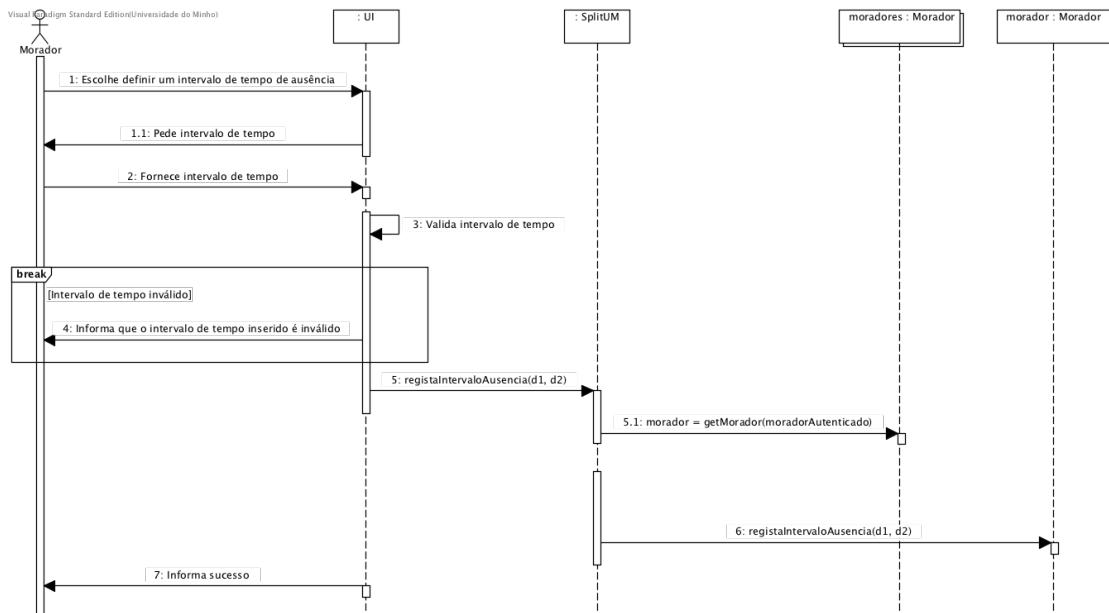
**Figura 41 – Diagrama de sequência de sistema do UC15 – Terminar sessão**



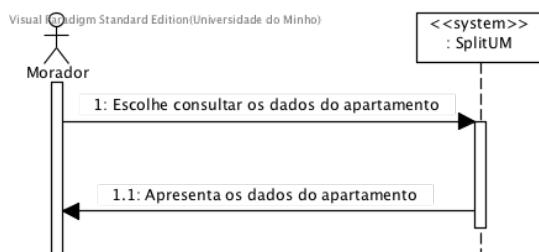
**Figura 42 – Diagrama de sequência de subsistemas do UC15 – Terminar sessão**



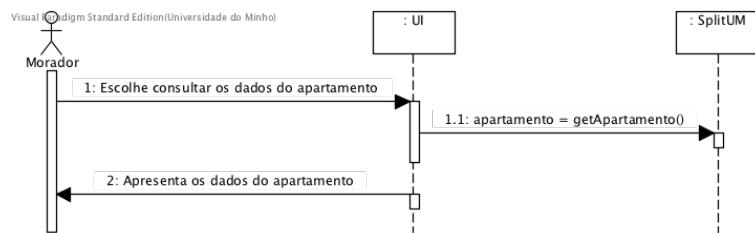
**Figura 43 – Diagrama de sequência de sistema do UC16 – Definir tempo intervalo de ausência**



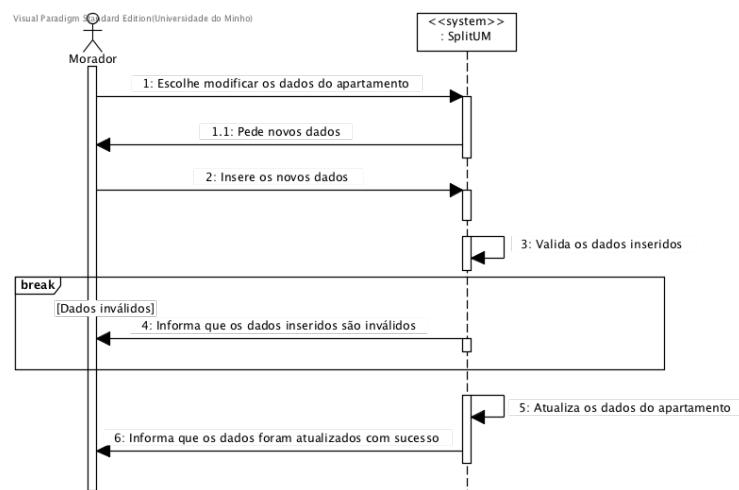
**Figura 44 – Diagrama de sequência de subsistemas do UC16 – Definir tempo intervalo de ausência**



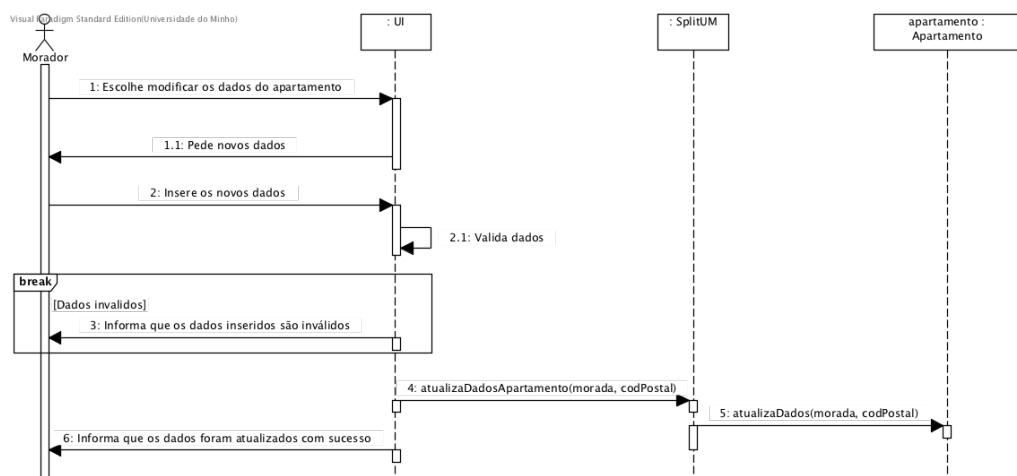
**Figura 45 – Diagrama de sequência de sistema do UC17 – Consultar detalhes de apartamento**



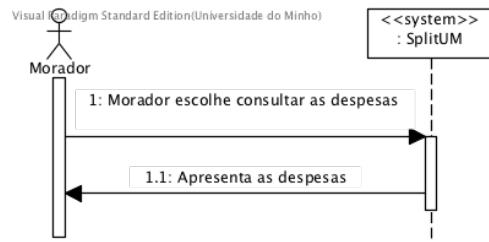
**Figura 46 – Diagrama de sequência de subsistemas do UC17 – Consultar detalhes de apartamento**



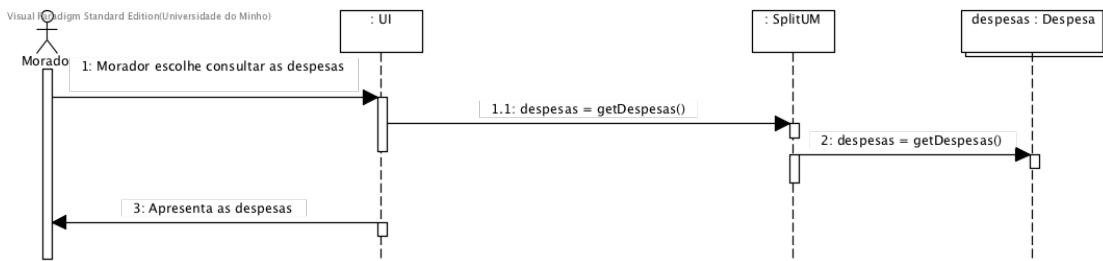
**Figura 47 – Diagrama de sequência de sistema do UC18 – Alterar detalhes de apartamento**



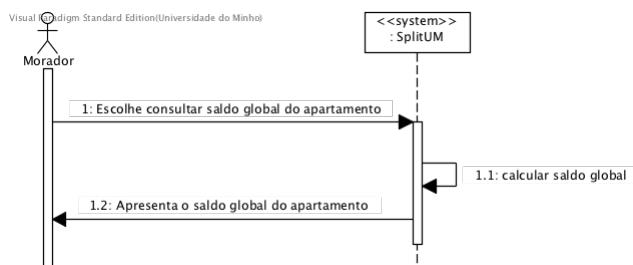
**Figura 48 – Diagrama de sequência de subsistemas do UC18 – Alterar detalhes de apartamento**



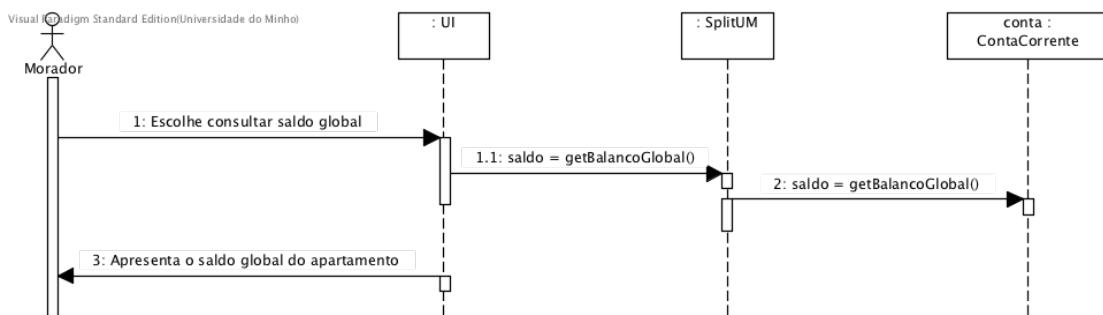
**Figura 49 – Diagrama de sequência de sistema do UC19 – Consultar despesas**



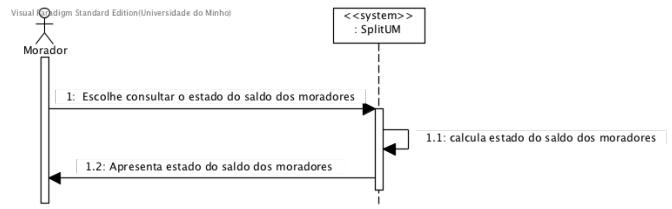
**Figura 50 – Diagrama de sequência de subsistemas do UC19 - Consultar despesas**



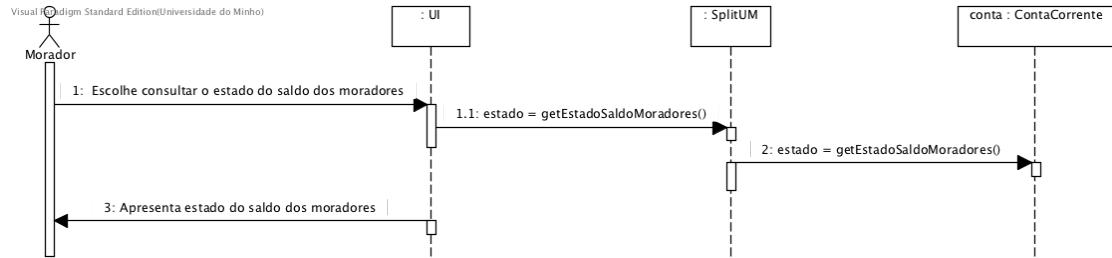
**Figura 51 – Diagrama de sequência de sistema do UC20 – Consultar saldo global do apartamento**



**Figura 52 – Diagrama de sequência de subsistemas do UC20 - Consultar saldo global do apartamento**



**Figura 53 – Diagrama de sequência de sistema do UC21 – Consultar o estado do saldo dos moradores**



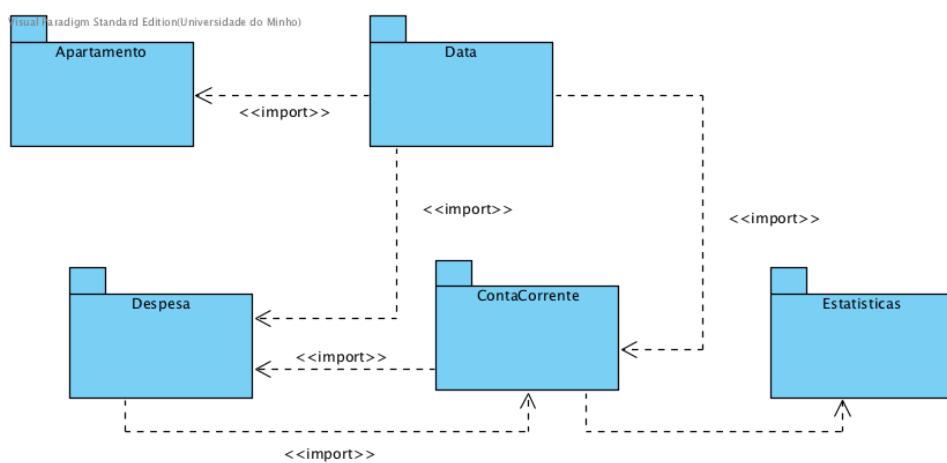
**Figura 54 – Diagrama de sequência de subsistemas do UC21 - Consultar o estado do saldo dos moradores**

## 7. Diagrama de packages e Diagrama de classes

Um package é uma agregação lógica de conceitos onde se pode ter um conjunto de entidades, de alguma forma, relacionadas. O diagrama de packages é um diagrama estrutural que permite estruturar e ilustrar a forma como os elementos do sistema se organizam e as respetivas dependências.

Na **Figura 55** encontra-se ilustrado o diagrama de packages, com as relações entre packages.

O diagrama de packages foi obtido através da análise dos diagramas de sequência de sistemas, de subsistemas e de métodos através de várias iterações até se conseguir identificar todos os packages existentes.

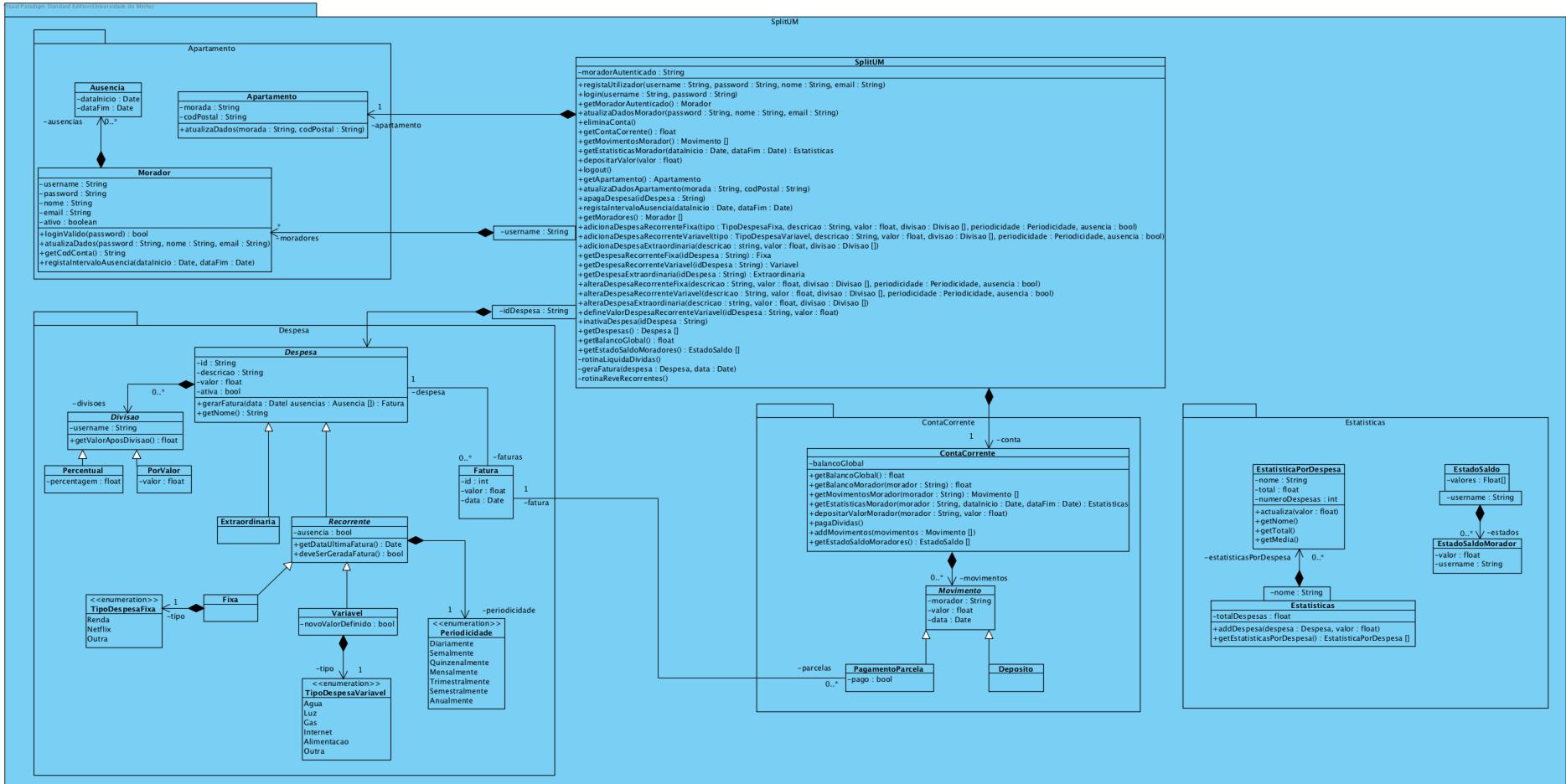


**Figura 55 – Diagrama de packages da aplicação SplitUM**

O diagrama de classes permite representar as entidades computacionais do sistema e determina a relação entre elas. O diagrama de classes é o diagrama adequado para a fase de conceção arquitetural e identifica todos os componentes estruturais do sistemas, ilustrando de forma clara as classes e as relações existentes.

Na **Figura 56** encontra-se o diagrama de classes, com as respetivos atributos e métodos, obtido a partir da análise dos diagramas de sequências de sistemas e de subsistema e do diagrama de packages.

O diagrama de classes apresentado corresponde a uma versão do diagrama de classes construído, ainda com associações qualificadas. Este diagrama é importante para definir de forma conceptual as classes e as suas relações sem preocupações de persistência.



**Figura 56** – Diagrama de classes da aplicação SplitUM

## 8. Diagrama de sequência para métodos importantes

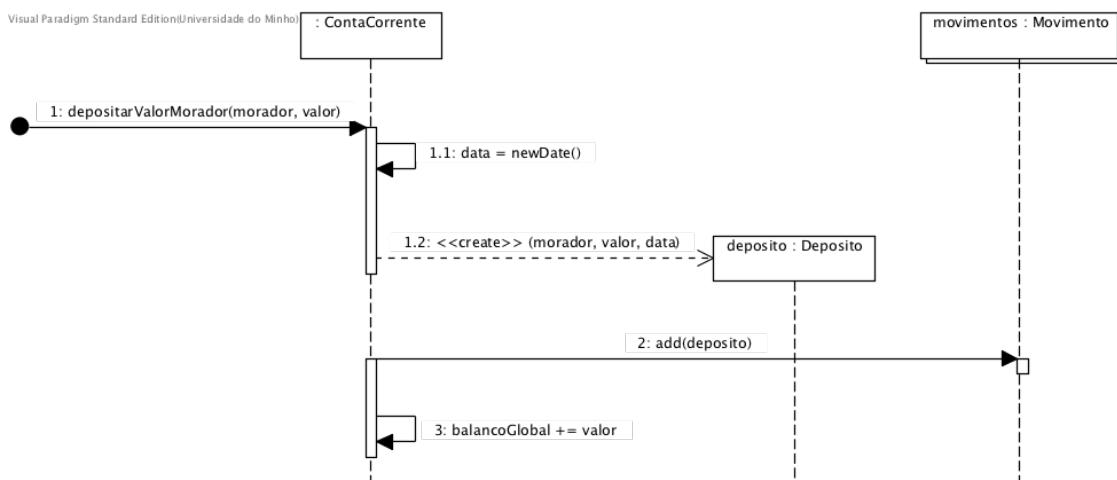
No sentido de especificar métodos que possam parecer mais elaborados ou que necessitam de um maior nível de especificação elaboraram-se os respectivos diagramas de sequência.

Consideraram-se 6 métodos importantes dos quais se segue a especificação.

- **Método depositarValor(morador, valor)**

O método depositarValor possibilita registar no sistema o valor que um determinado morador depositou. O registo fica guardado numa lista de Movimentos (dado que um Depósito é um Movimento).

Na **Figura 57** encontra-se o diagrama de sequência para o método depositarValor.



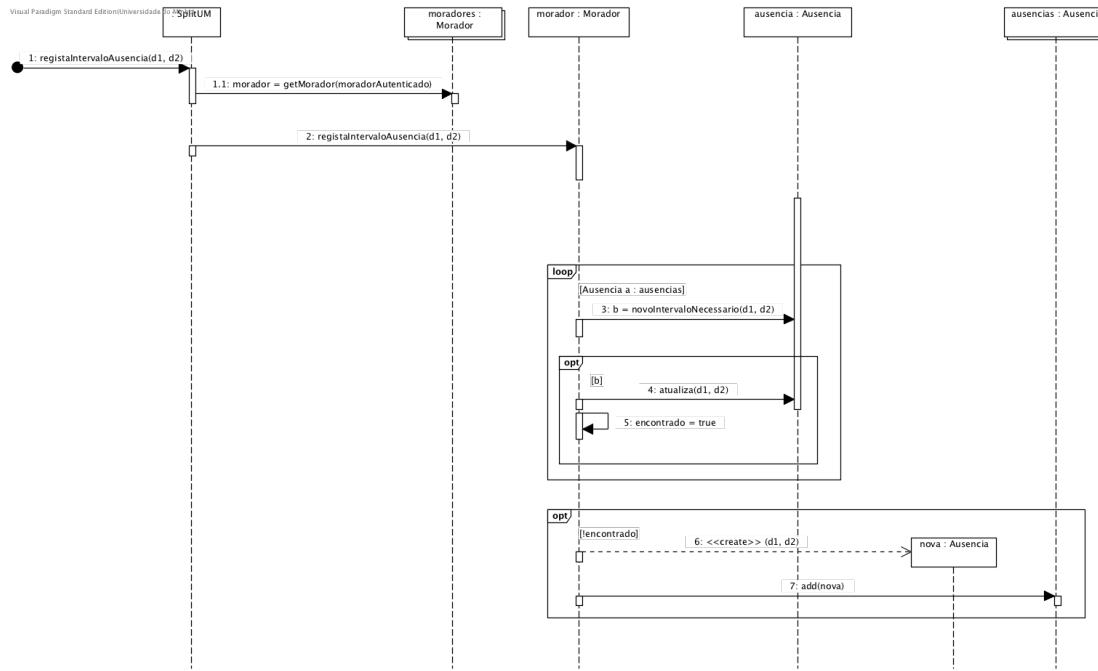
**Figura 57 – Diagrama de sequência do método depositarValor**

- **Método registaIntervaloAusencia(d1, d2)**

O método registaIntervaloAusencia permite a um morador indicar que durante um intervalo de tempo entre data de inicio (d1) e data fim (d2) estará ausente do apartamento. Isto permite que as despesas, que possibilitam ausência, possam ser desativadas para aquele morador durante aquele intervalo de tempo.

O morador tem uma lista dos intervalos de ausência à qual deverá ser adicionado o novo intervalo de ausência. No entanto, este novo intervalo de ausência só deverá ser adicionado caso não exista na lista de ausências um intervalo que o englobe.

Na **Figura 58** encontra-se o diagrama de sequência para o método registaIntervaloAusencia.



**Figura 58 – Diagrama de sequência do método `registaIntervaloAusencia`**

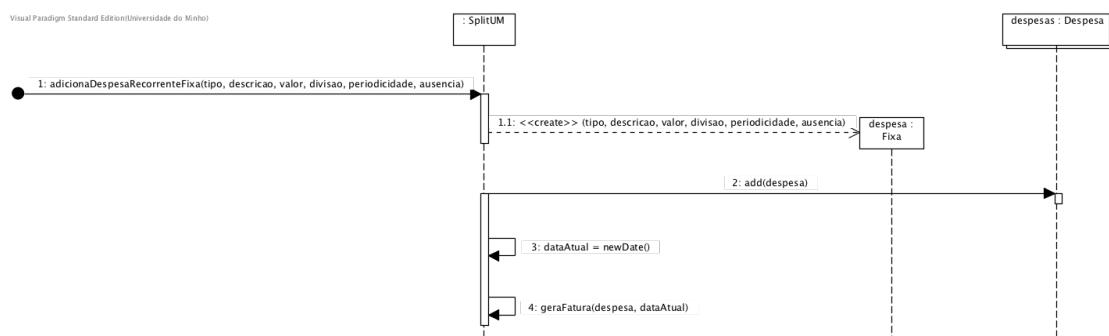
- **Método**

**adicionaDespesaRecorrenteFixa**(tipo, descrição, valor, divisão, periodicidade, ausência)

O método `adicionaDespesaRecorrenteFixa` permite adicionar uma despesa recorrente fixa no sistema. Para tal, é necessário passar os respetivos atributos.

A despesa é adicionada à lista de despesas.

Na **Figura 59** encontra-se o diagrama de sequência para o método `adicionaDespesaRecorrenteFixa`.



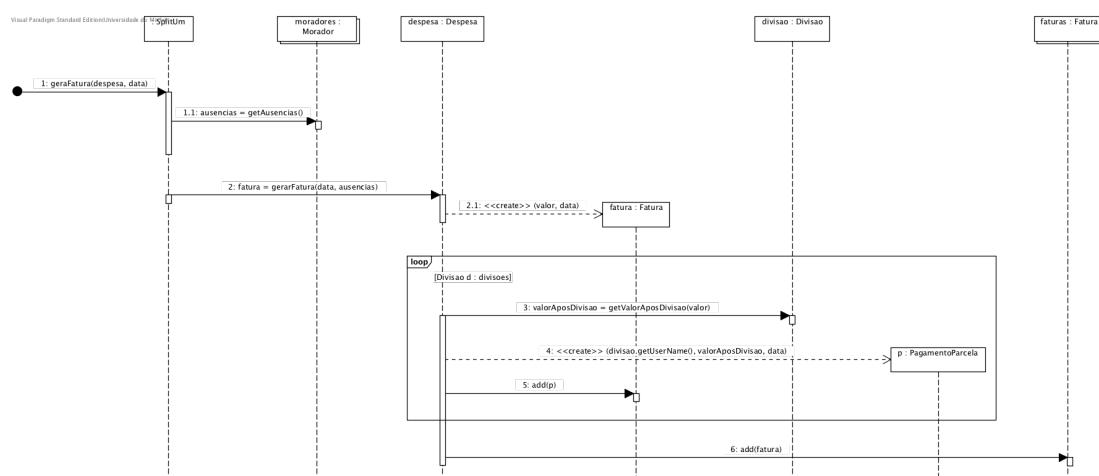
**Figura 59 – Diagrama de sequência do método `adicionaDespesaRecorrenteFixa`**

**Nota:** Existem várias tipos de despesa (Recorrente fixa, Recorrente variável, Extradordinária). Para todas, o processo de adicionar ao sistema é idêntico e, portanto, o método `adicionaDespesaRecorrenteFixa` foi utilizado como exemplo. Entre as despesas apenas variam os respetivos atributos específicos de cada uma.

- **Método geraFatura(despesa, data)**

O método `geraFatura` permite gerar uma fatura a partir de uma despesa associando a data da fatura. Para tal, é necessário verificar se, para a despesa associada à fatura, existem ausências de moradores que não devem entrar para o pagamento da fatura, total ou parcialmente (dependendo do tempo de ausência). É criada a fatura e adicionada à lista de faturas existente, ficando associada à despesa. Ainda é necessário fazer a divisão do valor total da fatura pelos vários moradores associados à despesa. Para tal, para cada morador é criado um objeto `PagamentoParcela` que corresponde à parcela de dívida resultante da sua fatia a pagar daquela fatura.

Na **Figura 60** encontra-se o diagrama de sequência para o método `geraFatura`.



**Figura 60 – Diagrama de sequência do método geraFatura**

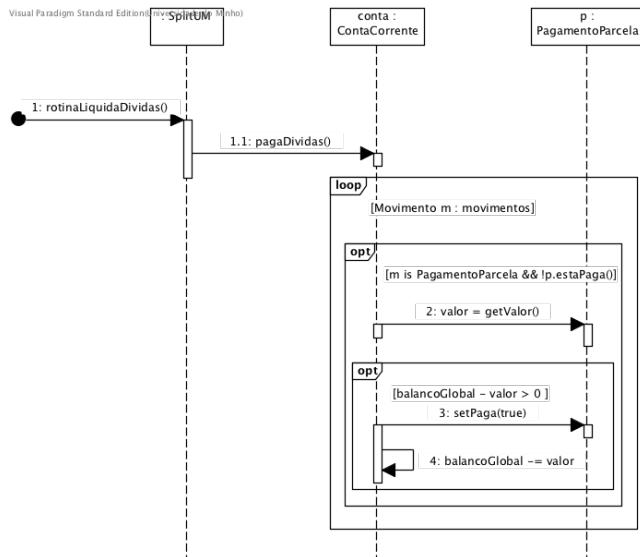
- **Método rotinaLiquidaDividas()**

Esta rotina permite verificar se as parcelas de dívidas existentes podem ser pagas com os valores disponíveis no sistema.

O método `rotinaLiquidaDividas` é chamado em duas situações:

- **sempre que algum morador deposita um valor:** Se um morador fizer um depósito é verificado se a quantia depositada permite pagar alguma parcela de dívida.
- **sempre que seja gerada uma nova fatura:** Se uma fatura é gerada é verificado se existe valor suficiente para a pagar.

Na **Figura 61** encontra-se o diagrama de sequência para o método `rotinaLiquidaDividas`.



**Figura 61 – Diagrama de sequência do método rotinaLiquidaDividas**

- **Método rotinaReveRecorrentes()**

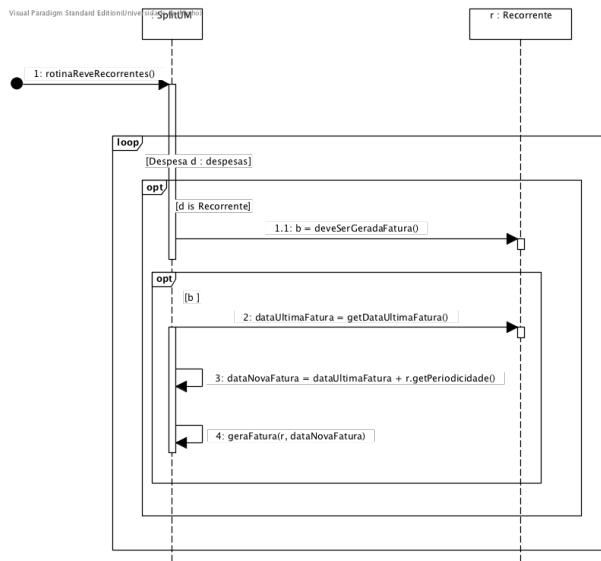
Esta rotina permite verificar se já é o momento de gerar as faturas recorrentes.

Este método é chamado sempre que um utilizador faz login.

As faturas das despesas recorrentes têm de ser geradas pelo sistema periodicamente (de acordo com a periodicidade selecionada).

Portanto, sempre que um morador entra na aplicação é verificado se existem faturas associadas a despesas recorrentes que devem ser geradas.

Na **Figura 62** encontra-se o diagrama de sequência para o método `rotinaReveRecorrentes`.



**Figura 62 – Diagrama de sequência do método rotinaReveRecorrentes**

## 9. Implementação da camada de persistência ORM (base de dados)

### 9.1. Decisões importantes

O mapeamento dos objetos relevantes em tabelas da base de dados foi relativamente direto. Apenas em dois casos, foram agrupados objetos de classes diferentes na mesma tabela, por partilharam muita informação em comum (partilham a mesma super-classe).

A tabela “Despesa” representa informação de três classes concretas: Despesa Extraordinária, Despesa Recorrente Fixa e Despesa Recorrente Variável. A despesa extraordinária tem os seguintes atributos da tabela a nulo: “periodicidade”, “ausência”, “tipo”, “fixa” e “novoValorDefinido”. Já as duas despesas recorrentes usam todos os atributos da tabela “Despesa”, sendo a diferença principal o valor do atributo booleano “fixa” ser verdadeiro para despesas recorrentes fixas, e falso em caso de despesas recorrentes variáveis.

A tabela “Movimento” também agrupa as duas classes “PagamentoParcela” e “Depósito”, já que ambas são sub-classes da classe “Movimento”. A diferença entre elas é que os atributos “paga” e “idFatura” não estão definidos (são nulos) para os depósitos.

### 9.2. Modelo Entidade Relacionamento da base de dados

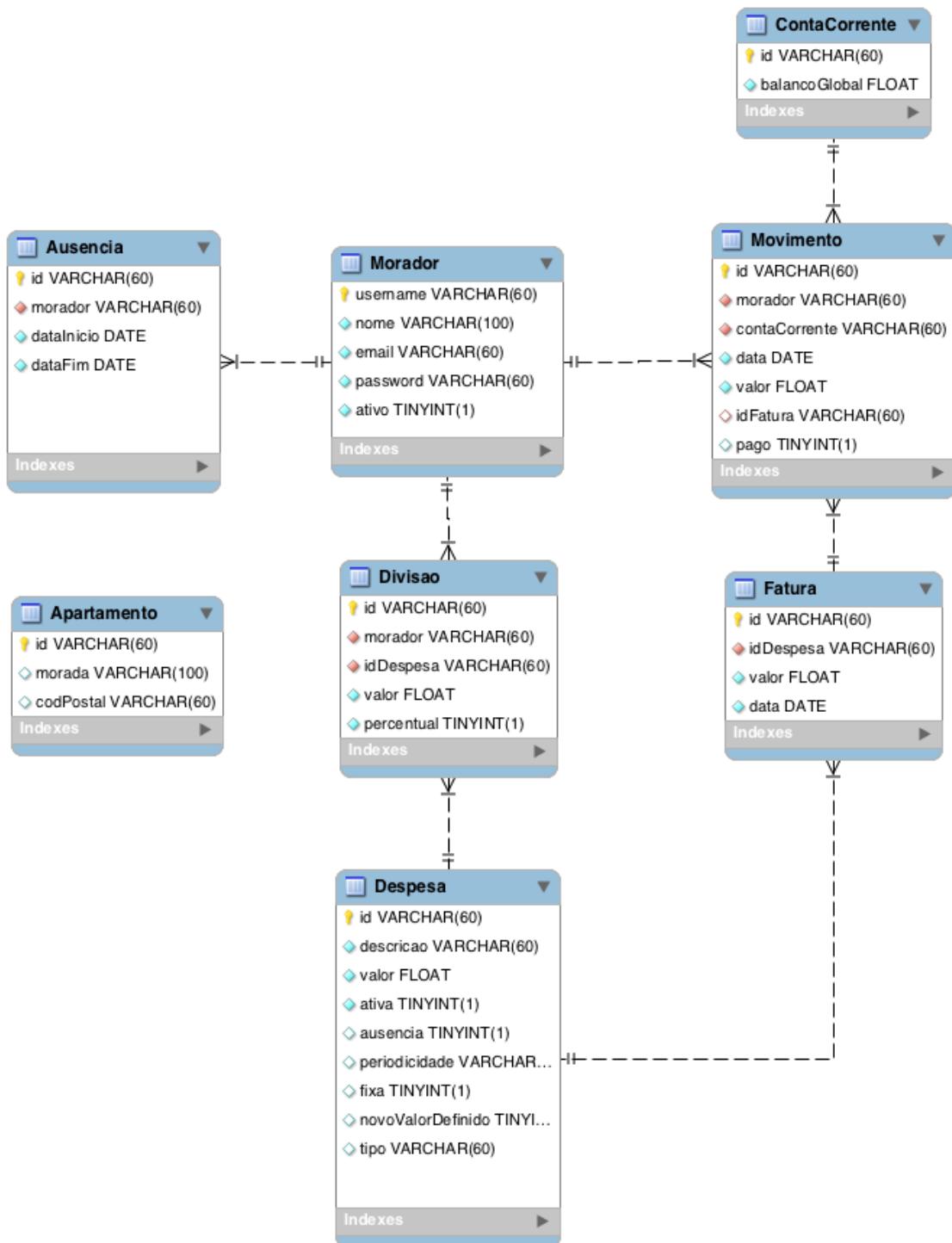
O Modelo de Entidade Relacionamento (Modelo ER) é um modelo de dados que permite descrever o domínio de negócio ou seus requisitos de processo de forma abstrata. Na **Figura 63**, pode-se visualizar o Modelo ER construído para a criação da base de dados, que irá guardar os dados necessários à aplicação SplitUM.

### 9.3. Diagrama de classes (com DAO)

Os DAOs (Data Access Object) simplificam o acesso aos dados, ao permitir carregar em memória, e de forma transparente, instâncias de objetos persistidos na base de dados. Isto é conseguido pelo facto de todas as classes DAOs implementarem a interface "java.util.Map" do Java.

Este facto faz com que a integração de DAOs com a camada de negócio seja extremamente simples, porque os métodos funcionam exatamente da mesma maneira. Apenas basta substituir a variáveis que era mapas, para serem instâncias de DAOs.

Na **Figura 64** apresenta-se o diagrama de classes redefinido com os DAOs.



**Figura 63** - Modelo da Entidade Relacionamento da base de dados.

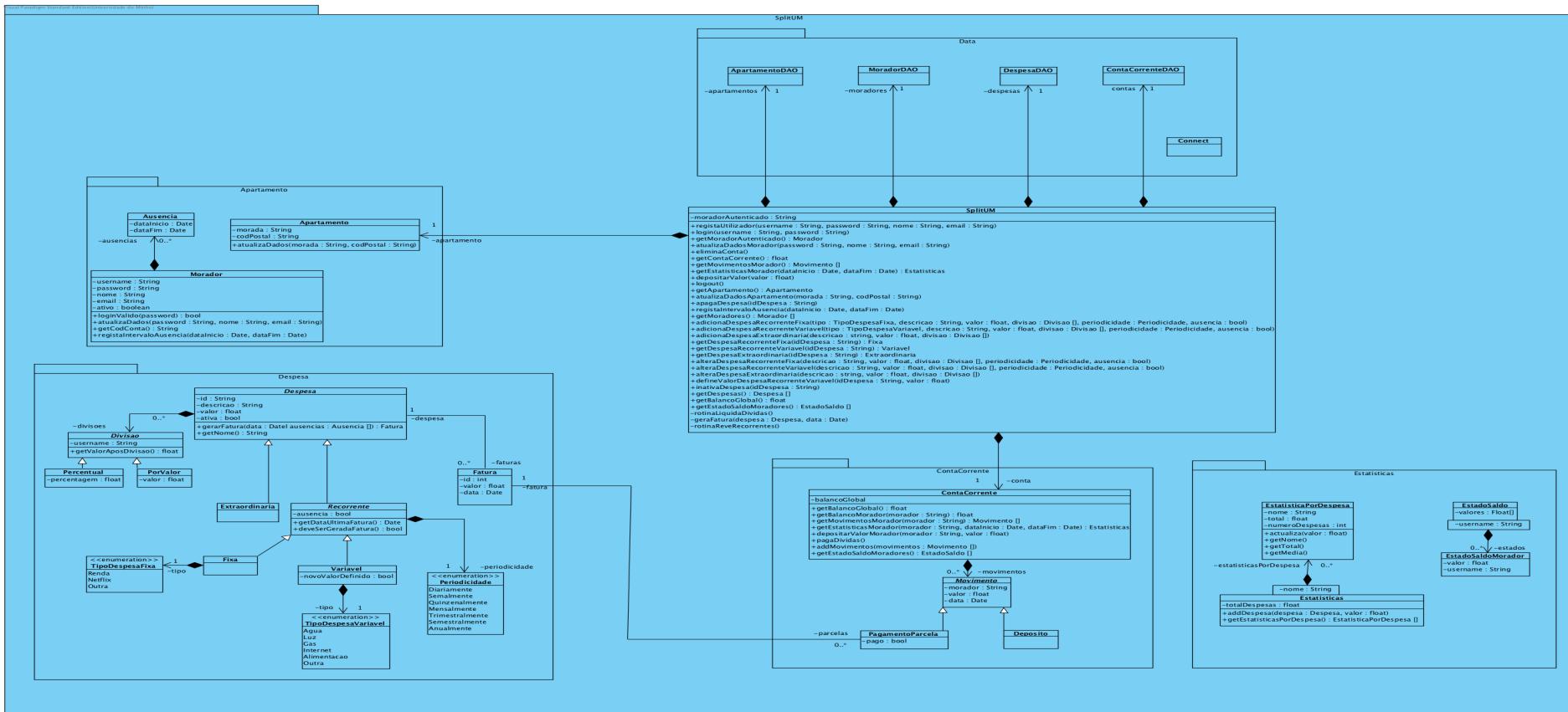


Figura 64 – Diagrama de Classes com os DAOs

## 10. Interface

Após a fase Desenho da Interface, passou-se, finalmente, a construção e implementação da interface, utilizando em certas situações o padrão MVC.

### 10.1. MVC (Model-View-Controller)

O MVC (Model-View-Controller) é um padrão de engenharia de software que define uma clara separação entre a interação com o utilizador (controller), onde estão os dados (Model) e como apresentar esses mesmos dados ao utilizador (View).

No nosso caso, estando o trabalho implementado em Java e usando interface gráfica em Swing, o Controller são os eventos gerados pelos componentes java swing como botões ou check-boxes, o Model é a camada lógica de negócio (facade SplitUM) e a View é todos as componentes gráficas da interface que apresentam dados do Model ao utilizador.

Por exemplo, o botão Login (Controller) da interface escuta por eventos em que o rato do utilizador pressiona nele mesmo e chama um método login(...) do SplitUM (Model), que se tiver sucesso, notifica os interessados da View, neste caso o painel do topo da aplicação para atualizar o nome do morador autenticado. Para tal, o painel (JPanel) tem que implementar a interface Observer do java e implementar o método update(...) para receber atualização do Model. Do lado do SplitUM, esta classe extends a classe Observable e sempre que há alterações ao dados relevantes, chama o método setChanged() e o notifyObservers() para notificar todos os observers da View.

### 10.2. Mockups vs Interface

Nesta secção mostra-se alguns exemplos interface construída para os mesmos exemplos dos mockups apresentados na secção 5.1. Poderá se verificar que os Mockups se encontram bastante semelhantes com a interface implementada.

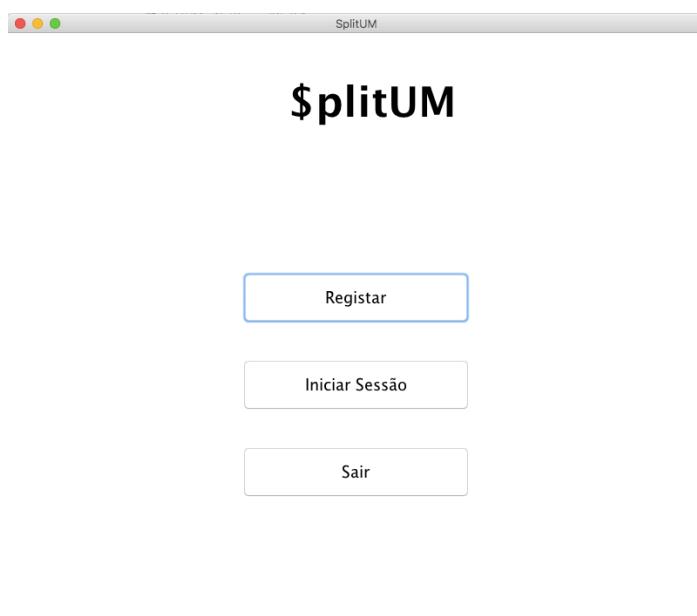
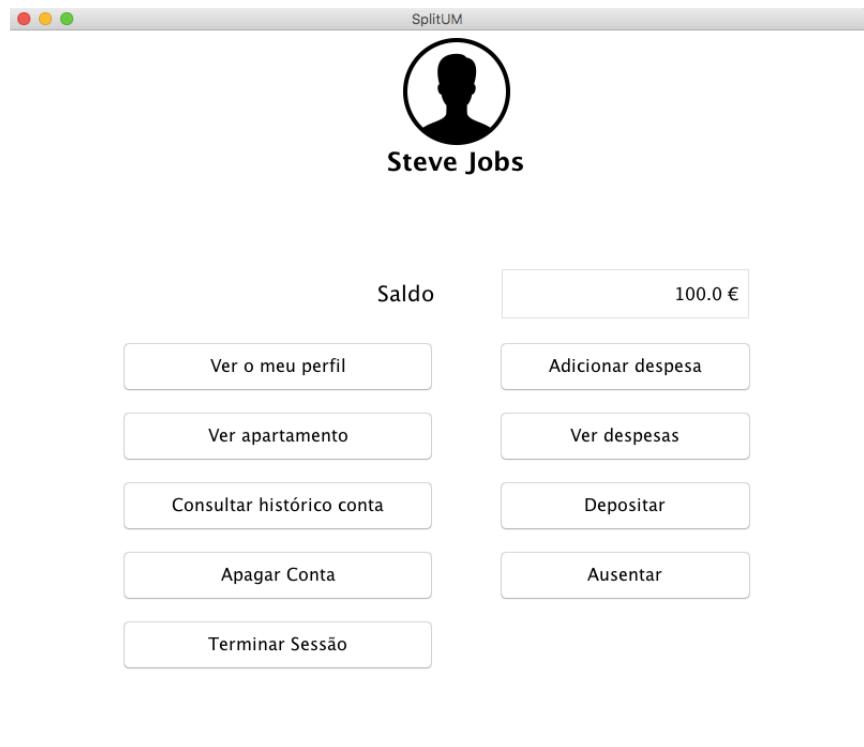


Figura 65 - Menu Inicial



Nome	Saldo	Dividas
Ana Maria	+100.0 €	--
José Manuel	-100.0 €	--
Ricardo Silva	-100.0 €	Ana Maria (100.0€)
Steve Jobs	0.0 €	--

**Morada**  
Quinta das Tílias N°53 – 3ºEsq

**Código Postal**  
4925-874

**Moradores**

Nome	Saldo	Dividas
Ana Maria	+100.0 €	--
José Manuel	-100.0 €	--
Ricardo Silva	-100.0 €	Ana Maria (100.0€)
Steve Jobs	0.0 €	--

**Saldo Apartamento: 100.0 €**

**Figura 66 - Menu Morador Autenticado.**

**Figura 67 - Menu Apartamento**



**Figura 68 - Menu Histórico da Conta Corrente**

The screenshot shows the SplitUM application interface for adding a new expense. At the top, there is a navigation bar with three colored window control buttons (red, yellow, green) and the text "SplitUM". Below the navigation bar is a circular profile picture placeholder with the text "Steve Jobs" underneath it.

The main form consists of several input fields and selection buttons:

- A text input field labeled "Descrição da despesa" (Description of expense) with an empty input box.
- A text input field labeled "Valor da despesa" (Expense value) with an empty input box.
- A radio button group labeled "Tipo da despesa" (Type of expense) with two options: "Recorrente" (Recurrent) and "Extraordinária" (Extraordinary). The "Recorrente" option is selected.
- A radio button group labeled "Divisão da Despesa" (Expense distribution) with two options: "Por Valor" (By Value) and "Percentual". The "Por Valor" option is selected.
- Two buttons at the bottom: "Continuar" (Continue) on the left and "Cancelar" (Cancel) on the right.

**Figura 69 - Menu Adicionar Despesa**

The screenshot shows a window titled "SplitUM" with a close button in the top-left corner. In the top-right corner is a circular profile picture of a person. Below the profile picture, the name "Steve Jobs" is displayed. On the left side of the window, there is a back arrow labeled "< Voltar". The main content area contains a table with four rows of expense data:

Nome	Inserido Em	Pago
Renda do Apartamento	2016-12-30	x
Luz	2016-12-30	✓
Televisão	2016-12-30	x

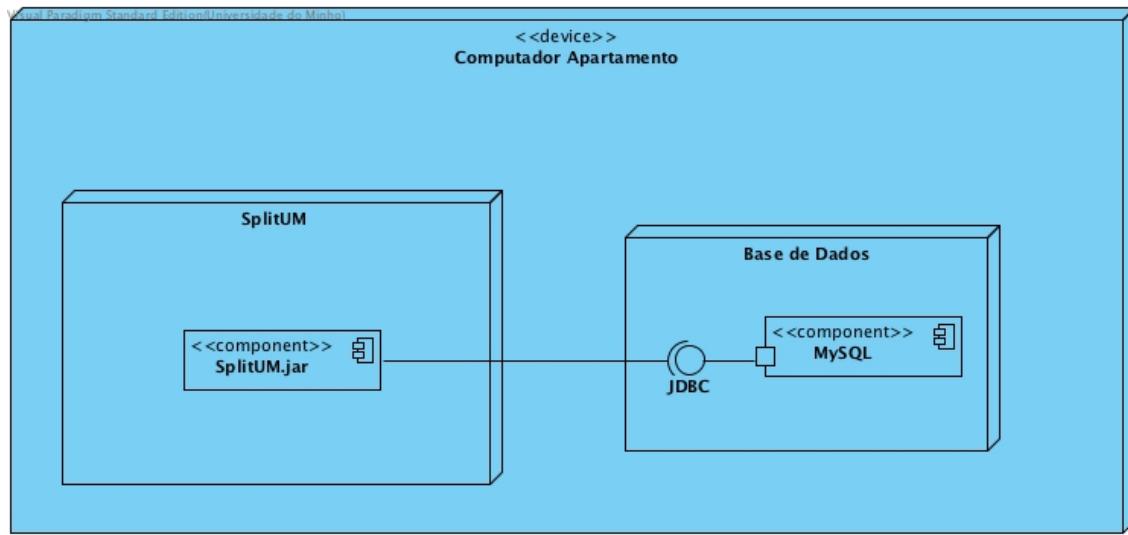
Below the table are four buttons arranged in a 2x2 grid:

- Editar Despesa (Edit Expense)
- Ver Detalhes Despesa (View Expense Details)
- Apagar Despesa (Delete Expense)
- Ver Estatísticas (View Statistics)

**Figura 70** - Menu Ver Despesas

## 11. Diagrama de Instalação

Os Diagramas de Instalação especificam a arquitetura física do sistema. Na **Figura 71** pode-se visualizar o diagrama de instalação da aplicação SplitUM.



**Figura 71** - Diagrama de Instalação da aplicação SplitUM.

## 12. Conclusão

A SplitUM pretende resolver um problema que é cada vez mais comum e que necessita rapidamente de uma resposta adequada. As alternativas atualmente existentes mostram-se muito complicadas ou demasiado simples, não respondendo às necessidades dos utilizadores.

Pretende-se apresentar uma solução que facilite a vida das pessoas que dividem apartamentos e possam gerir as despesas partilhadas de forma rápida, fácil e despreocupada!

Neste relatório foram apresentadas as etapas do desenvolvimento deste sistema que permite partilha de despesas num apartamento.

A primeira etapa consistiu no levantamento de requisitos através de pesquisa e junto de pessoas que partilham ou já partilham apartamentos. Estas pessoas mostraram ser uma fonte importante de informação pois são potenciais utilizadores reais da aplicação SplitUM. O levantamento de requisitos mostrou ser uma tarefa árdua e complicada, uma vez que é difícil extrair as funcionalidades exatas que os utilizadores mais precisam. É fundamental perceber o que os utilizadores pretendem e necessitam. Sem isso, a aplicação torna-se inútil. Nesta etapa foi onde se encontrou mais dificuldades, a identificação dos requisitos não é imediata e estes necessitam de constante validação. Assim, nas etapas seguintes houve um ajuste dos requisitos para que estes fossem de encontro ao esperado pelos utilizadores.

Na segunda etapa, a partir dos requisitos identificados foi construído o modelo de domínio. Esta construção foi feita de forma iterativa em que foram refinadas as ligações entre as entidades foi ajustado o domínio da aplicação.

Na etapa seguinte, a partir dos requisitos funcionais e do modelo de domínio foi elaborado o diagrama de *Use Cases* especificando, essencialmente, as interações do ator morador com o sistema.

De seguida, a partir da especificação dos *use cases*, para cada *use case*, definiram-se os respectivos diagramas de sequência de sistemas e de subsistemas. Esta abordagem permitiu modelar a solução e de seguida elaborar o diagrama de *packages* e o diagrama de classes. Finalizada a parte de modelação foi implementada a solução e o respectivo código.

Uma vez que a metodologia adoptada para o projeto foi a do RUP, além da constante validação dos requisitos também as restantes etapas sofreram, de forma continua e iterativa, de validação e refinamento de forma iterativa, até ao final do projeto.

Para implementar a camada de dados, modelou-se as tabelas da base de dados de acordo com as classes relevantes do diagrama de classes feito anteriormente e de seguida implementou-se as respetivas classes DAOs da camada de dados, para a camada de negócios aceder de forma transparente à base de dados.

A implementação de uma interface gráfica permite aos utilizadores da SplitUM uma experiência de usabilidade mais fácil e intuitiva, do que a execução da aplicação

na linha de comandos. Desta forma a aplicação terá como possíveis clientes uma maior população alvo. Antes da implementação da interface procedeu-se ao desenho de esboços da mesma (Mockups) e a implementação da máquinas de estados da interface, que auxiliou na verificação de redundâncias da futura interface.

Como se trata de uma aplicação desenvolvida sobre o padrão MVC, se for necessário adaptar a interface a outras plataformas ou tecnologias, uma vez que esta adaptação apenas afeta uma das três camadas (*View* - apresentação), será bastante facilitada.

A aplicação tem um modelo construído com pontos de extensão futuros. Caso se queira adicionar a funcionalidade de gerir múltiplos apartamentos, este desenvolvimento já se encontra suportado. Também a noção de múltiplas contas correntes é existente e pode ser aproveitada na eventualidade de existir essa funcionalidade.

Também o facto da persistência estar abstraída da camada de negócio, permite que se troque a forma de persistência sem que a camada de negócio sofra o impacto dessa alteração.