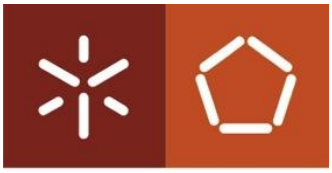

Universidade do Minho

Desenvolvimento de Sistemas de Software

Relatório do Projeto Prático

"Gestão de Despesas num Apartamento"

Ano Letivo 2016/2017

A74207 - Bruno Rafael Lamas Corredoura Dantas

A75655 - Daniel Camelo Rodrigues

A74219 - Hugo Alves Carvalho

A74260 - Luís Miguel da Cunha Lima

Introdução

O presente relatório é elaborado no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software, do 3º ano do Mestrado Integrado em Engenharia Informática, com vista à implementação de um sistema de suporte à gestão de despesas num apartamento.

A aplicação deverá suportar o registo e consulta das despesas do apartamento por parte dos moradores e do senhorio, assim como a gestão do pagamento, mantendo uma conta corrente para cada um destes.

Uma vez que todos os membros do nosso grupo residem num apartamento durante o período letivo, foi realizada uma recolha de informação de modo a compreender melhor o problema. Assim, analisando como são geridas as despesas nos seus apartamentos e, ao implementar o nosso modelo de domínio, conseguimos identificar requisitos fundamentais para o sistema.

Numa parte inicial deste relatório, correspondente à primeira fase do projeto, para além da apresentação do modelo de domínio, estão também inseridos os diagramas de *Use Case*, assim como as suas respetivas especificações. Com a identificação destes *Use Case*s, conseguimos definir o comportamento do sistema, isto é, a comunicação entre o sistema e os seus atores. O conjunto de todos estes *Use Cases* define a funcionalidade do sistema que mais tarde será implementado.

Em seguida, na segunda fase do projeto, serão mostrados os diagramas das máquinas de estado, os diagramas de sequência, que inicialmente foram desenvolvidos sem subsistemas e em seguida com a existência de diferentes subsistemas, o diagrama de classes, os diagramas de *package* com subsistemas, os diagramas de atividade de cada *Use Case* e o diagrama de instalação.

Por último, será explicado como foi desenvolvida a aplicação bem como a sua interface gráfica.

1 - Modelo de Domínio

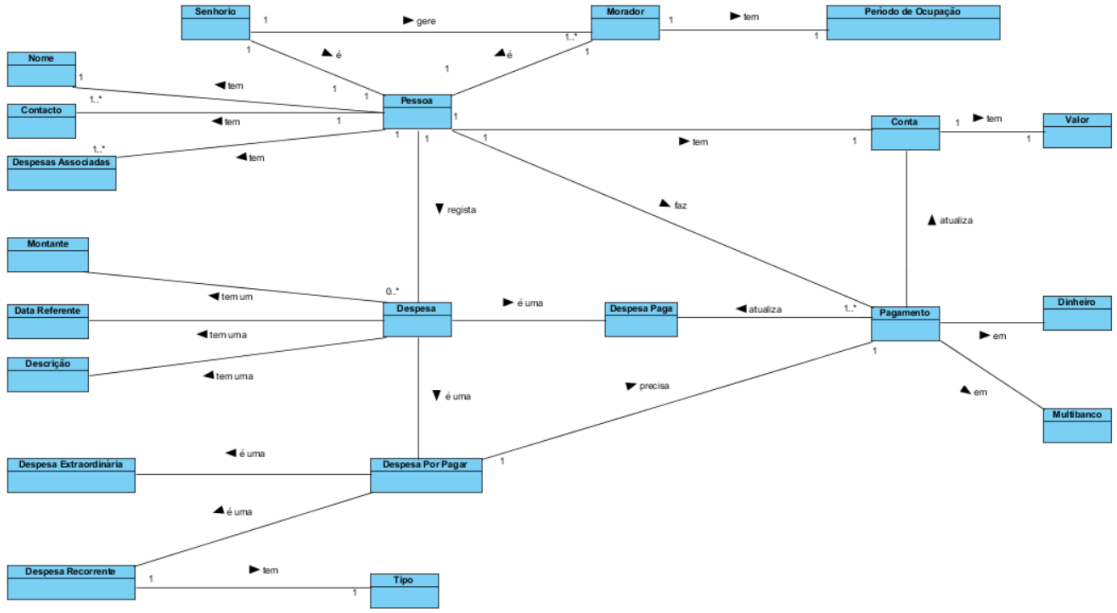

Figura 1 - Diagrama de Modelo de Domínio

No nosso modelo de domínio, consideramos a existência de dois atores fundamentais, o senhorio e o morador. Uma vez que estes têm comportamentos semelhantes, interpretamos os dois como sendo uma pessoa, com a diferença que o morador tem que ter associado um período de ocupação do apartamento, que posteriormente irá influenciar a distribuição das despesas nos seus montantes individuais. Cada pessoa tem que ter um nome e um ou mais contactos, de modo a poderem ser identificados. Estes atores têm também uma lista com as despesas que lhes estão associadas, e que irão fazer parte do cálculo do montante a individual a pagar.

Como no nosso sistema consideramos que tanto o senhorio como o morador podem registar, consultar e pagar despesas, cada um destes atores terá também que ter uma conta associada, que conterá um valor. Este valor é negativo quando o ator em questão tiver contas em atraso para pagar, positivo quando alguém tiver que dar dinheiro a esse ator (por exemplo no caso em que um morador tenha pago uma despesa, e só depois é que os restantes moradores lhe pagam a sua parte individual), e está nulo quando as despesas estão todas em dia.

Estes atores podem também registar as despesas que já foram pagas e as que estão por pagar. Cada uma destas terá que ter um montante, uma data referente à despesa e uma descrição. No caso das despesas por pagar, podemos considerar dois casos: despesas extraordinárias, como por exemplo quando um morador estraga algum objeto no apartamento

e é necessário repará-lo, e despesas recorrentes, como por exemplo as contas do gás, da água, da internet, da luz e do condomínio. Para estas despesas recorrentes, está associado um tipo, permitindo assim, a qualquer momento, considerar uma nova despesa como despesa recorrente, tornando assim o sistema mais flexível.

Todas as despesas que estão por pagar precisam de um pagamento que poderá ser efetuado em dinheiro ou através de multibanco. Ao efetuar o pagamento, a despesa passa a ser considerada uma despesa paga, e a conta associada ao ator, é atualizada.

2- Diagramas de Use Case

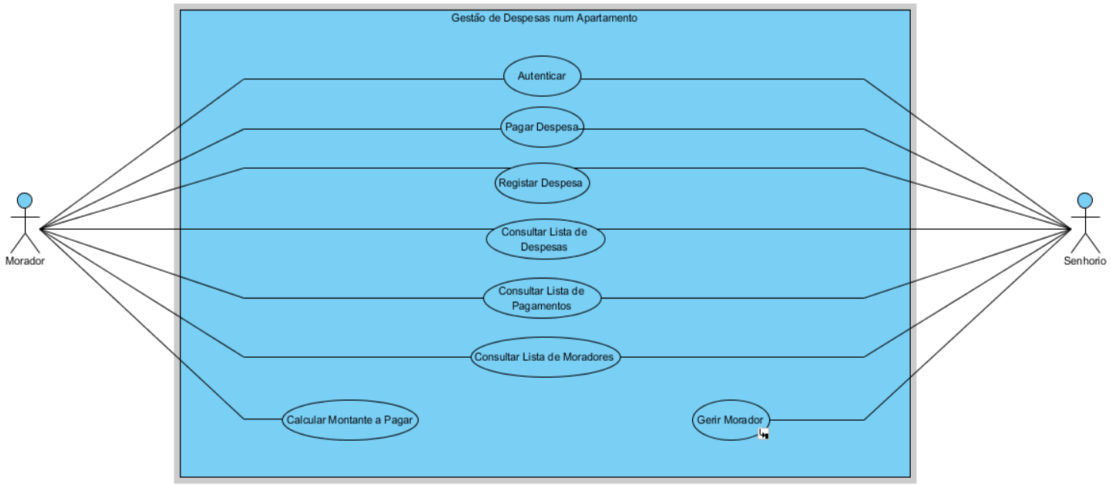

Figura 2 - Diagrama de Use Case - Gestão de Despesas num Apartamento

Na nossa implementação dos Diagramas de Use Case, decidimos optar por um diagrama simples, com Use Cases capazes de cobrir as funcionalidades fundamentais do programa. De modo a tornar o diagrama mais flexível a possíveis erros/alterações noutras fases do projeto, tentamos generalizar o máximo possível os Use Case apresentados.

Com base nestas razões, foram considerados, no diagrama principal (Figura 2), os seguintes Use Cases:

Morador: Calcular Montante a Pagar;

Senhorio: Gerir Morador;

Morador e Senhorio: Autenticar, Registar Despesa, Pagar Despesa, Consultar Lista de Despesas, Consultar Lista de Pagamentos e Consultar Lista de Moradores.

Para facilitar a interpretação deste diagrama principal, foi criado um subdiagrama para o Use Case Gerir Morador, uma vez que este engloba outros Use Cases fundamentais destinados ao Senhorio. Assim, este subdiagrama (Figura 3) que contém apenas o Senhorio como ator, apresenta os seguintes Use Cases:

Senhorio: Registar Morador, Remover Morador, Alterar Dados Morador, Consultar Dados Morador.

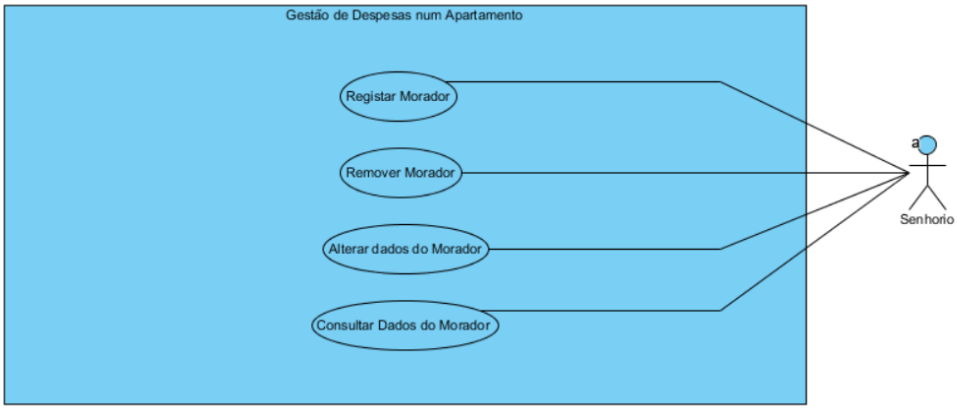

Figura 3 - Diagrama de Use Case - Gestão de Despesas num Apartamento [Gerir Morador]

3 - Especificação dos Use Cases

3.1 – Use Case Autenticar

Use Case: Autenticar
Descrição: Um utilizador tenta aceder ao sistema de despesas.
Pré-condição: -
Pós-condição: O utilizador tem acesso ao sistema.

	Actor	Sistema
Comportamento normal:		1. Pede ao utilizador o seu e-mail e a sua senha.
	2. Introduz a informação pedida.	
		3. Valida os dados.
		4. Permitir o acesso ao sistema.
Exceção (Passo 3) [Os dados introduzidos estão incorretos.]		1.1. Informar o utilizador que os dados estão incorretos.

Figura 4 - Especificação do Use Case Autenticar

Este Use Case é um dos mais importantes uma vez que sem ele, o utilizador não tem acesso às funcionalidades da aplicação. O facto de tanto Senhorio como Morador tenha que estar autenticado para aceder à aplicação, serve para manter a segurança e a privacidade dos dados do utilizador.

3.2 – Use Case Registar Despesa

Use Case: Registar despesa.		
Descrição: O morador ou o senhorio adiciona uma despesa.		
Pré-condição: Estar autenticado.		
Pós-condição: É adicionada uma despesa á lista de despesas por pagar ao/s morado/res.		
	Actor	Sistema
Comportamento normal:		1. Apresentar opções do tipo de despesa. (Extraordinária, recorrente)
	2. Escolher opção extraordinária.	
		3. Processar informação, pedir descrição da despesa.
	4. Inserir descrição e informação relevante da despesa.	
		5. Adicionar despesa e ajustar o valor a pagar para cada morador.
Comportamento alternativo: [despesa recorrente] (Passo 2)	2.1. Escolhe opção recorrente.	
		2.2. Apresenta opções de despesa recorrente.
	2.3. Escolher opção.	
		2.4. Continuar passo 5

Figura 5 - Especificação do Use Case Registar Despesa

Este Use Case é fundamental para a aplicação, uma vez que esta está estritamente relacionada com a existência de despesas.

Serve para que, posteriormente, estas despesas sejam apresentadas aos utilizadores e para que estes possam interagir com estas.

3.3 – Use Case Consultar Lista de Despesas

Use Case: Consultar lista de despesas
Descrição: Apresentar lista de despesas por pagar.
Pré-condição: Estar autenticado.
Pós-condição: O morador ou o senhorio tem acesso a lista de despesas.

	Actor	Sistema
Comportamento normal:		1. Apresentar opções de organização da lista.
	2. Escolher opção de ordem.	
		3. Apresentar lista ordenada segundo a opção.
Exceção: **(Passo 1)** **[Não há despesas a apresentar.]**		1.1. Apresentar mensagem a informar que não há despesas.

Figura 6 - Especificação do Use Case Consultar Lista de Despesas

Este Use Case facilita a visualização das várias despesas existentes no apartamento, que estão por pagar. Estão serão apresentadas com a sua data, montante, descrição e tipo (no caso de ser uma despesa recorrente).

3.4 – Use Case Pagar Despesa

Use Case: Pagar despesa.
Descrição: O morador paga uma despesa.
Pré-condição: O morador tem de estar autenticado.
Pós-condição: A despesa é paga e o morador recebe um recibo.

	Actor	Sistema
Comportamento normal:	1. <<Include>>Consultar lista de despesas.	
	2. Escolhe a despesa que quer pagar.	
		3. Apresenta opções de pagamento.
	4. Escolhe opção de pagamento.	
		5. Confirma pagamento
Exceção: **(Passo 1)** **[Não há despesas a pagar]**		1.1. Informar que não há despesas a pagar.

Figura 7 - Especificação do Use Case Pagar Despesa

O Use Case serve para que os utilizadores possam pagar as suas despesas, e para que esta operação fique registada tanto na sua conta corrente, como também na lista de pagamentos.

3.5 – Use Case Consultar Lista de Pagamentos

Use Case: Consultar lista de pagamentos.
Descrição: Apresentar lista de despesas pagas.
Pré-condição: Estar autenticado.
Pós-condição: O morador ou o senhorio tem acesso a lista de pagamentos.

	Actor	Sistema
Comportamento normal:		1. Apresentar opções de organização da lista.
	2. Escolher opção de ordem.	
		3. Apresentar lista ordenada segundo a opção.
Exceção: **(Passo 1)** **[Não há pagamentos a apresentar.]**		1.1. Apresentar mensagem a informar que não há pagamentos efetuados com sucesso.

Figura 8 - Especificação Use Case Consultar Lista de Pagamentos

Este Use Case tal como o Use Case de Consultar Lista de Despesas, serve para ter acesso as despesas já pagas pelos moradores, sendo uma prova do seu pagamento e é útil para confirmar que as várias despesas foram pagas, dado que o não pagamento das despesas recorrentes no seu prazo traz consequência para os moradores.

3.6 – Use Case Calcular Montante a Pagar

Use Case: Calcular montante a pagar.		
Descrição: Calcula o montante a pagar de uma ou várias despesas.		
Pré-condição: Estar autenticado.		
Pós-condição: Obter o valor do montante a pagar.		
	Actor	Sistema
Comportamento normal:	1. <<Include>> Consultar lista de despesas.	
	2. Escolher despesas que pretende juntar ao valor do montante.	
		3. Calcular o valor do montante e apresentar ao morador.

Figura 9 - Especificação do Use Case Calcular Montante a Pagar

Este Use Case é fundamental, uma vez que surge da possibilidade de existirem diversas formas de dividir as despesas. É assim possível apresentar o valor da despesa, como também para dividir essa despesa pelos vários utilizadores que estão associados a esta.

3.7 – Use Case Consultar Lista de Moradores

Use Case: Consultar lista de moradores.
Descrição: Apresenta a lista de moradores do apartamento.
Pré-condição: Estar autenticado.
Pós-condição: O senhorio tem acesso a lista de moradores.

	Actor	Sistema
Comportamento normal:		1. Apresentar opções de organização da lista.
	2. Escolher opção de ordem.	
		3. Apresentar lista ordenada segundo a opção.

Figura 10 - Especificação do Use Case Consultar Lista de Moradores

Este Use Case serve aos mesmo propósitos que os anteriores Use Case de consulta, por exemplo como consulta de valor em dívida, entre outros.

3.8 – Use Case Registar Morador

Use Case: Regista morador.
Descrição: É adicionado um novo morador ao apartamento.
Pré-condição: Estar autenticado como senhorio.
Pós-condição: O novo morador é adicionado com sucesso a lista de moradores.

	Actor	Sistema
Comportamento normal:	1. Inserir nome e e-mail.	
		2. Verificar a unicidade do nome e e-mail.
		3. Pedir mais informações sobre o morador.
	4. Inserir informações.	
		5. Registar dados e inserir o morador na lista de moradores.
Exceção: (Passo 2) [já existe um morador com o mesmo nome]		2.1. Apresenta mensagem a indicar o conflito de identificadores.

Figura 11 - Especificação do Use Case Registar Morador

Este use case está relacionado com o autenticar, permitindo acrescentar novo utilizadores ao sistema.

3.9 – Use Case Remover Morador

Use Case: Remover morador.		
Descrição: Um dos moradores da lista é removido.		
Pré-condição: Estar autenticado como senhorio.		
Pós-condição: O morador escolhido é removido com sucesso.		
	Actor	Sistema
Comportamento normal:	1. <<Include>> Consultar lista de moradores.	
	2. Escolher o morador.	
		3. Verificar se o morador tem dívidas.
		4. Remove o morador da lista.
Comportamento alternativo: [O morador possui dívidas] (Passo 4)		4.1 Avisar que o morador ainda possui dívidas.

Figura 12 - Especificação do Use Case Remover Morador

Uma vez há a possibilidade de um morador deixar o apartamento, este deverá ser removido do sistema, tal como representado neste Use Case.

3.10 – Use Case Alterar Dados do Morador

Use Case: Alterar dados do morador.		
Descrição: Alterar os dados de um morador.		
Pré-condição: Estar autenticado como senhorio.		
Pós-condição: Os dados do morador foram alterados com sucesso.		
	Actor	Sistema
Comportamento normal:	1. <<Include>> Consultar lista de moradores.	
	2. Escolher morador.	
		3. Apresentar dados de morador.
	4. Alterar dados.	
		5. Verificar alterações.
		6. Guardar alterações.

Figura 13 - Especificação do Use Case Alterar Dados do Morador

Este Use Case é útil caso o morador queira mudar algum dos seus dados, como por exemplo a sua senha de autenticação.

3.11 – Use Case Consultar Dados do Morador

Use Case: Consultar dados morador.		
Descrição: Os dados de um morador são apresentados.		
Pré-condição: Estar autenticado como senhorio.		
Pós-condição: O senhorio acesso aos dados do morador.		
	Actor	Sistema
Comportamento normal:	1.<<Include>>Consultar lista de moradores.	
	2. Escolher morador.	
		3. Apresentar dados de morador.

Figura 14 - Especificação do Use Case Consultar Dados Morador

Este Use Case é útil para obter informações referentes ao utilizador, principalmente o valor da sua conta corrente, e a percentagem que tem de pagar de cada despesa recorrente.

4 – Diagramas de Máquinas de Estado

Neste capitulo apresentamos os diagramas de maquinas de estado, diagramas estes que modelam todos os estados possíveis que o objeto/sistema atravessa em resposta aos eventos que podem ocorrer.

Optamos por dividir em dois para representar o comportamento do morador e outro do senhorio como podemos ver nas figuras abaixo, de modo a ficar mais organizado e simplificar a leitura e interpretação.

4.1 Máquinas de estado – morador

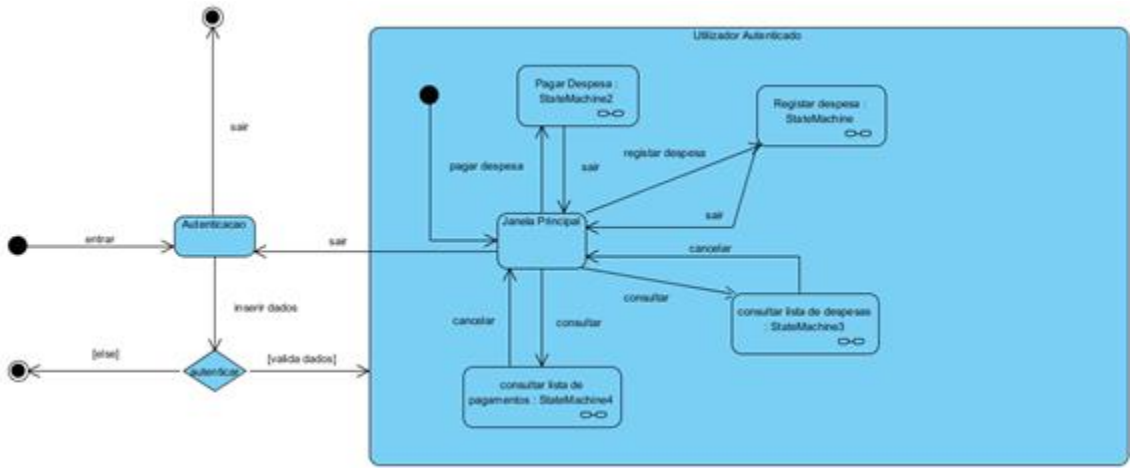

Figura 15 - Diagrama de Máquina de Estado (Morador)

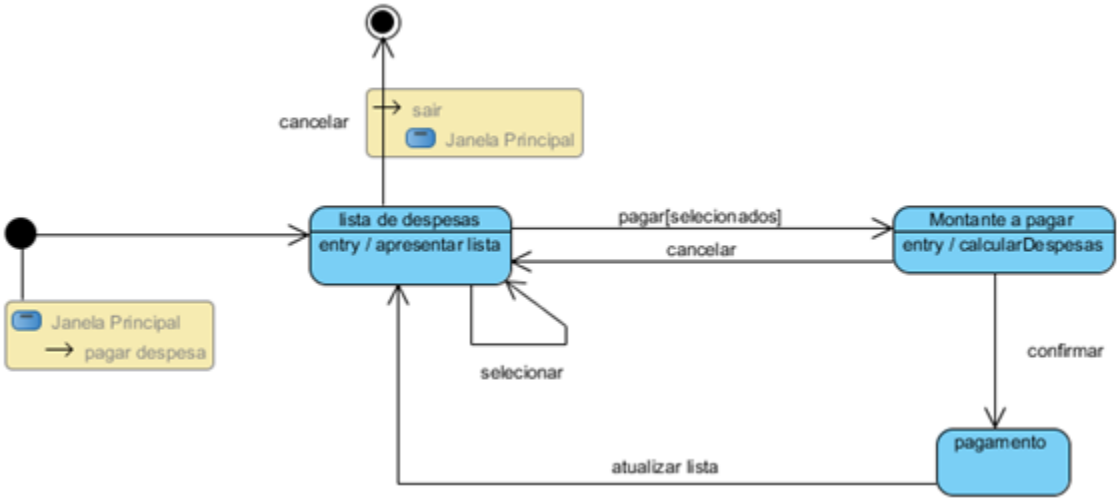

Figura 16 - Diagrama de Máquina de Estado (Morador – Pagar Despesa)

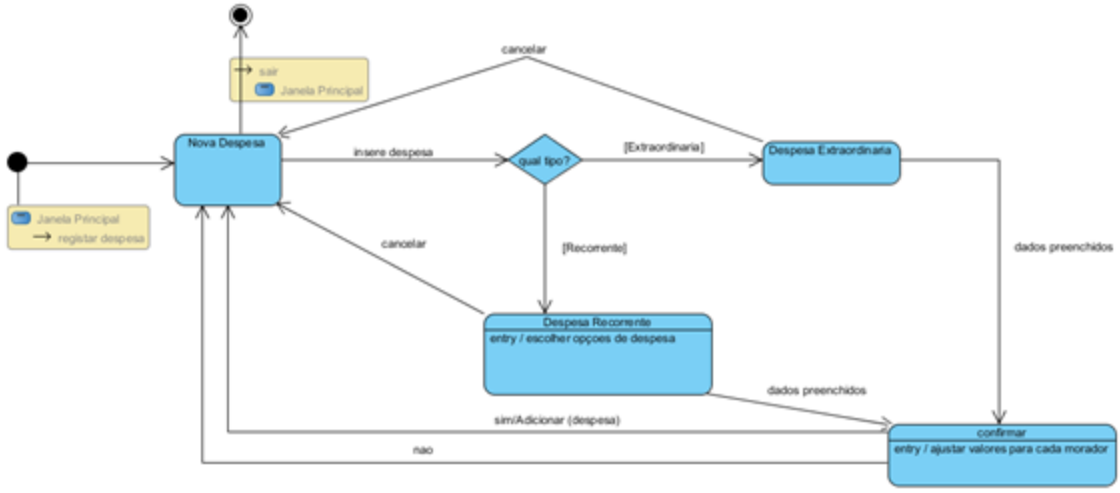

Figura 17 - Diagrama de Máquina de Estado (Morador – Registar Despesa)

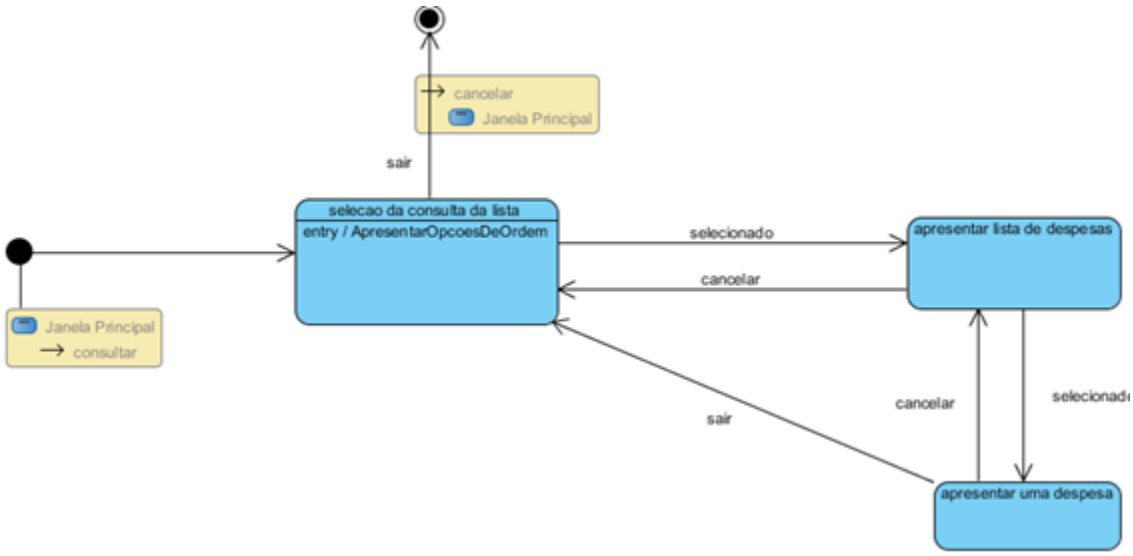

Figura 18 - Diagrama de Máquina de Estado (Morador – Consultar Lista de Despesas)

Os subdiagramas consultar despesas, consultar pagamentos e registar despesas são iguais aos apresentados em cima para o morador.

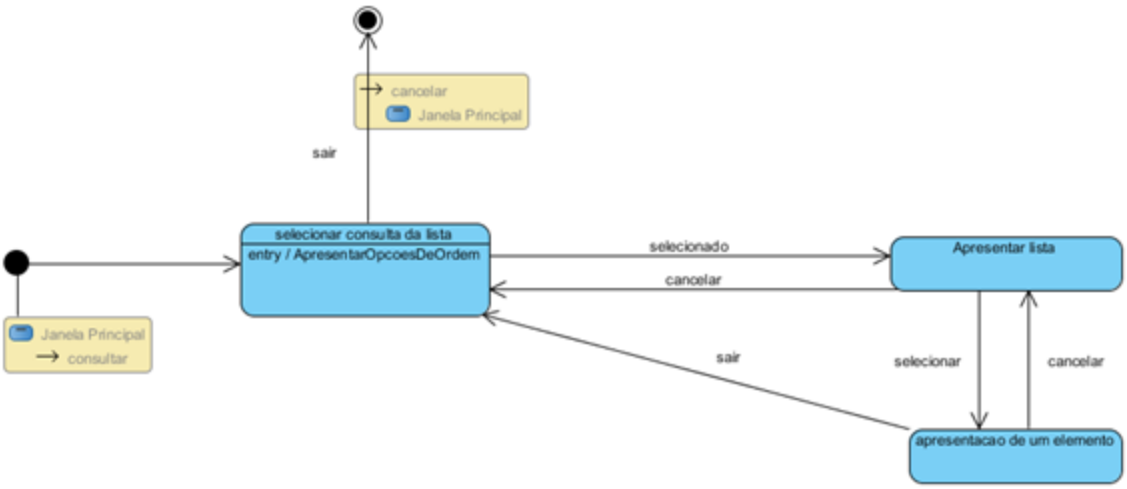

Figura 19 - Diagrama de Máquina de Estado (Morador – Consular lista Pagamentos)

4.2 Máquinas de estado – senhorio

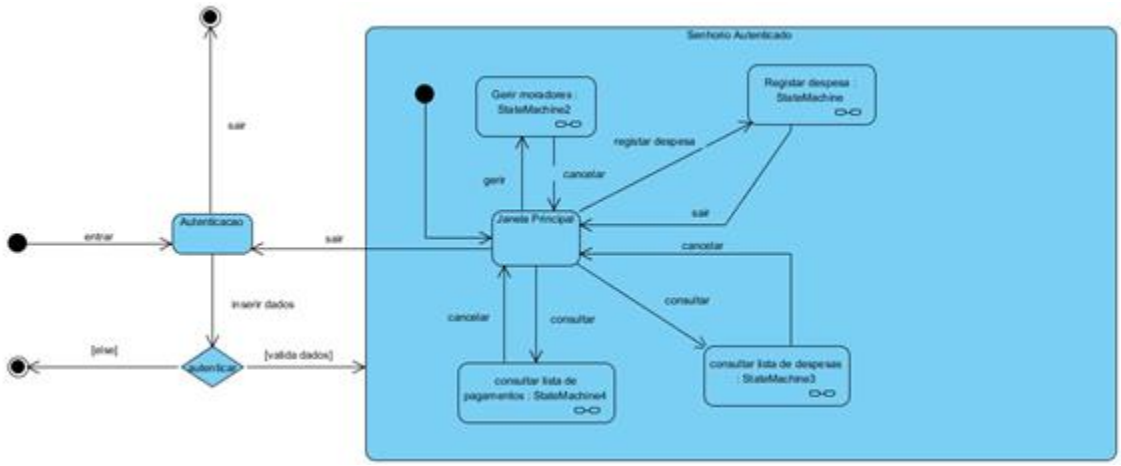

Figura 20 - Diagrama de Máquina de Estado (Senhorio)

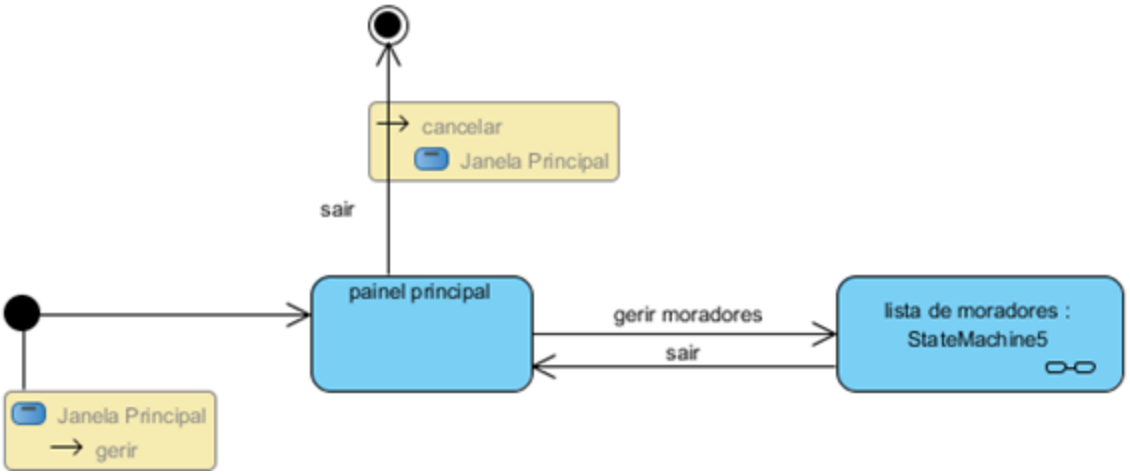

Figura 21 - Diagrama de Máquina de Estado (Senhorio - Gerir Moradores)

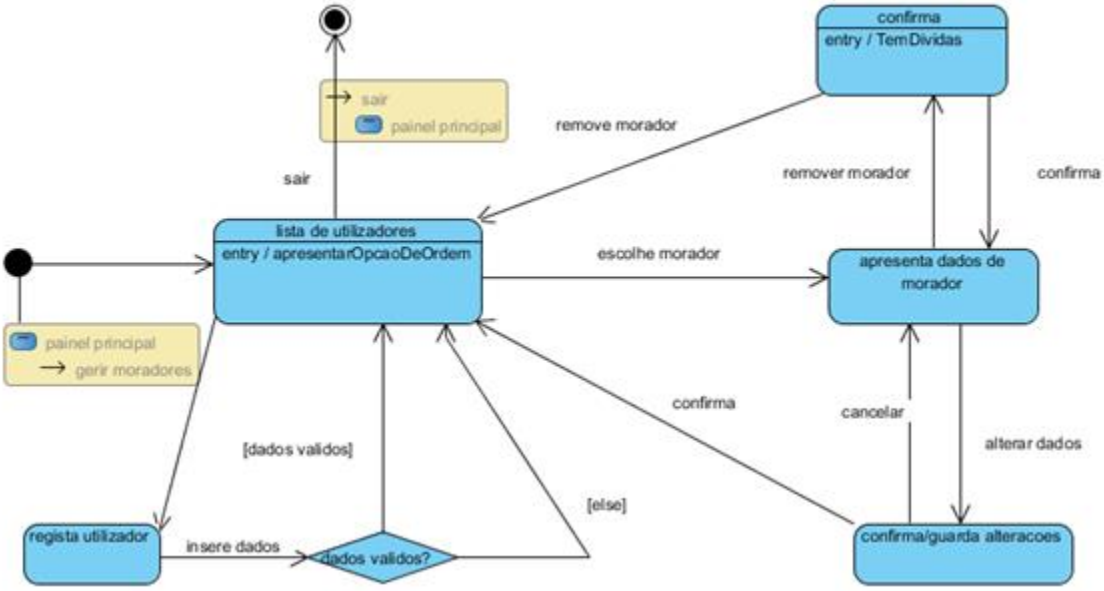

Figura 22 - Diagrama de Máquina de Estado (Senhorio - Gerir Moradores - subsistema)

5 - Diagramas de Sequência

Neste capítulo demonstramos as análises dos Use Cases partindo da organização tabular do capitulo 3 até à especificação em subsistemas, tentando representar a informação de uma forma simples e lógica. Estes diagramas permitem representar as sequencias dos processos que ocorrem nos vários Use Case bem como as interações entre objetos através das mensagens trocadas entre si. Este é um estudo importante, pois permite ter uma ideia melhor do desenvolvimento do problema de uma forma mais temporal.

5.1 – Diagramas Ator/Sistema

5.1.1 – Autenticar

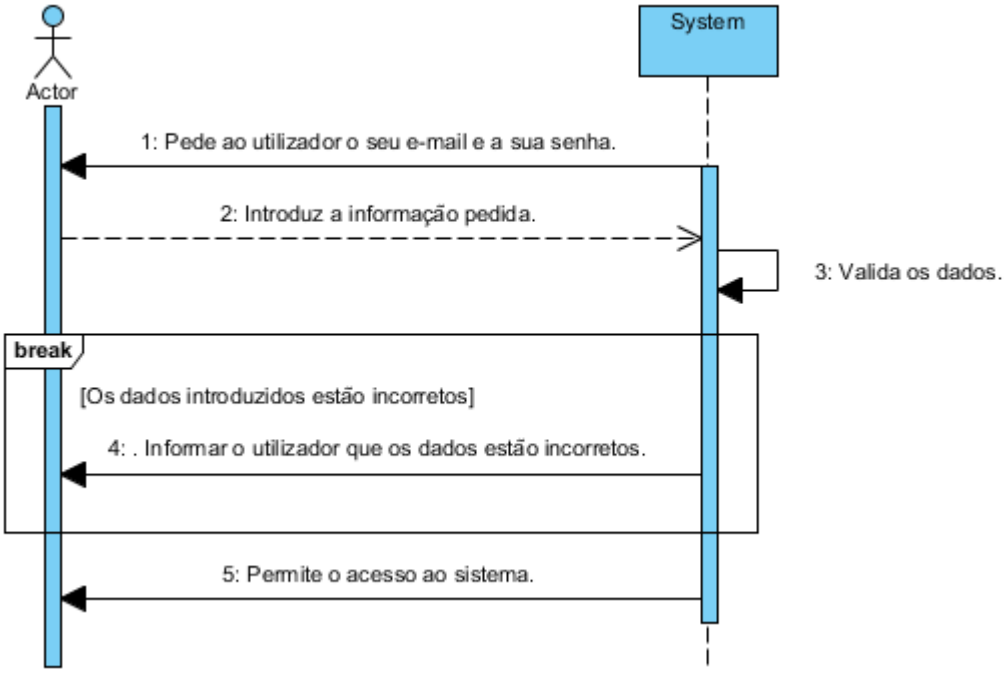

Figura 23 - Diagrama de Sequência com Actor/Sistema para UC Autenticar

5.1.2 – Registar Despesa

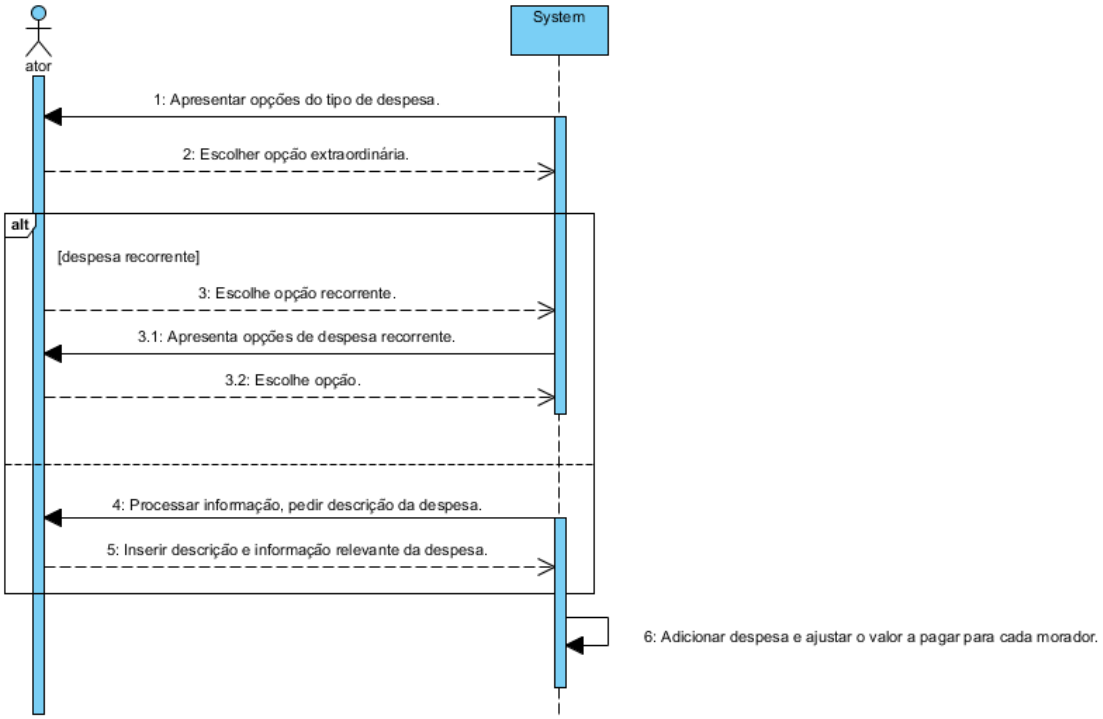

Figura 24 - Diagrama de Sequência com Actor/Sistema para UC Registar Despesa

5.1.3 – Pagar Despesa

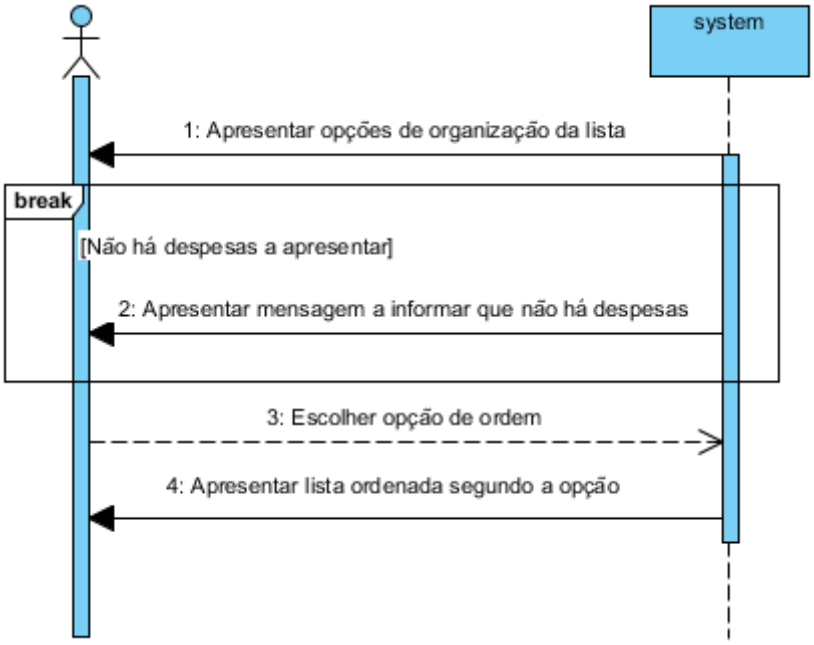

Figura 25 - Diagrama de Sequência com Actor/Sistema para UC Pagar Despesa

5.1.4 – Consultar Lista de Despesas

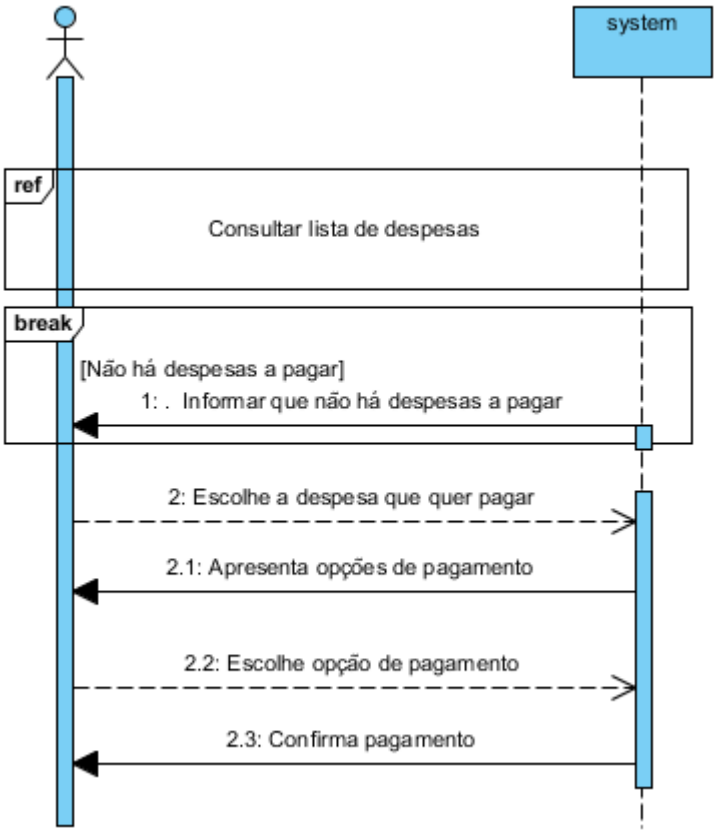

Figura 26 - Diagrama de Sequência com Actor/Sistema para UC Consultar Lista de Despesas

5.1.5 – Consultar Lista de Pagamentos

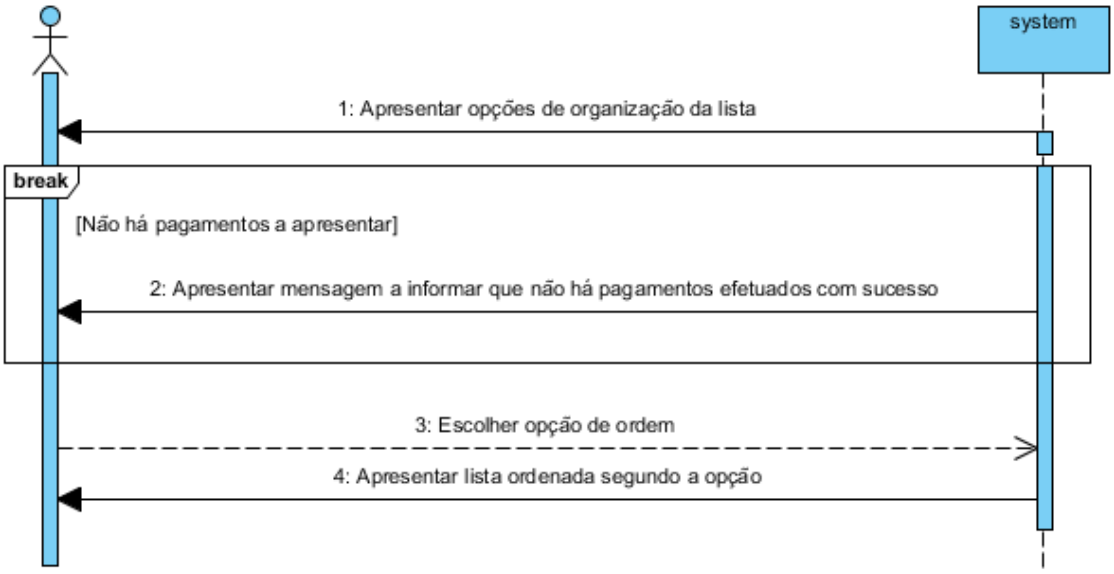

Figura 27 - Diagrama de Sequência com Actor/Sistema para UC Consultar Lista de Pagamentos

5.1.6 – Calcular Montante a Pagar

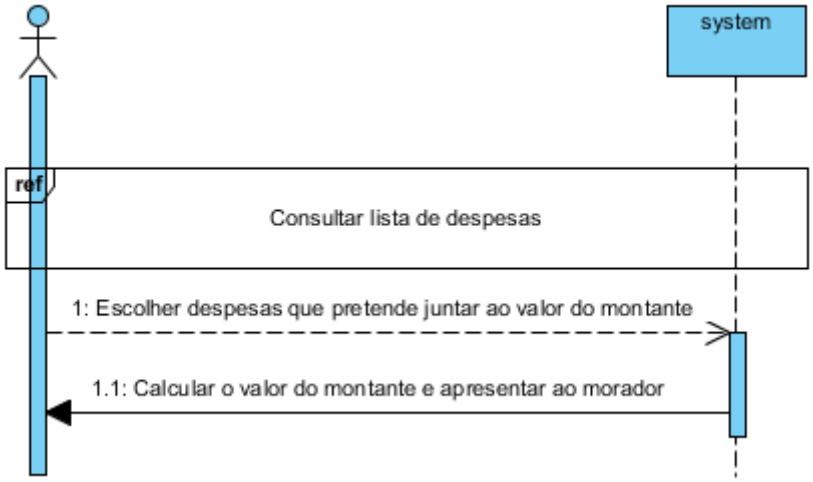

Figura 28 - Diagrama de Sequência com Actor/Sistema para UC Calcular Montante a Pagar

5.1.7 – Consultar Lista de Moradores

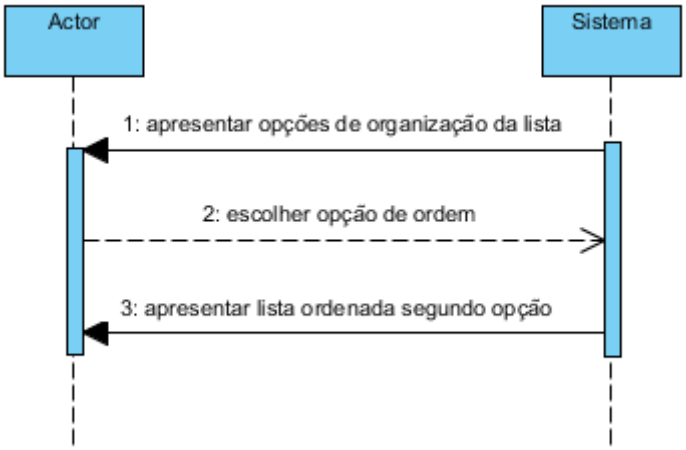

Figura 29 - Diagrama de Sequência com Actor/Sistema para UC Consultar Lista de Moradores

5.1.8 – Registar Morador

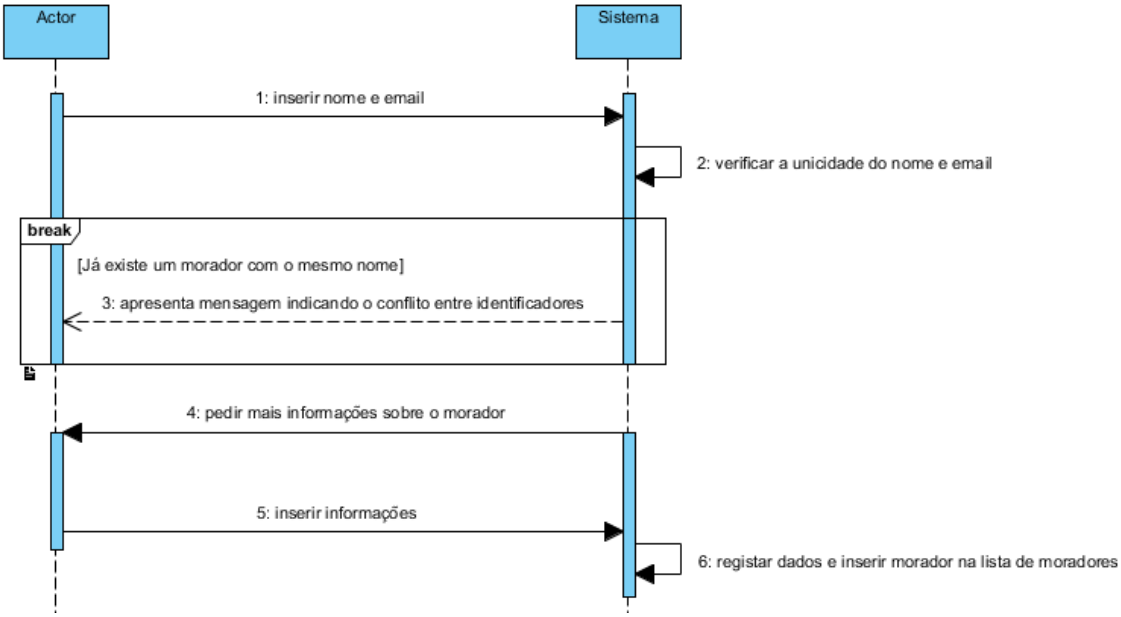

Figura 30 - Diagrama de Sequência com Actor/Sistema para UC Registar Morador

5.1.9 – Remover Morador

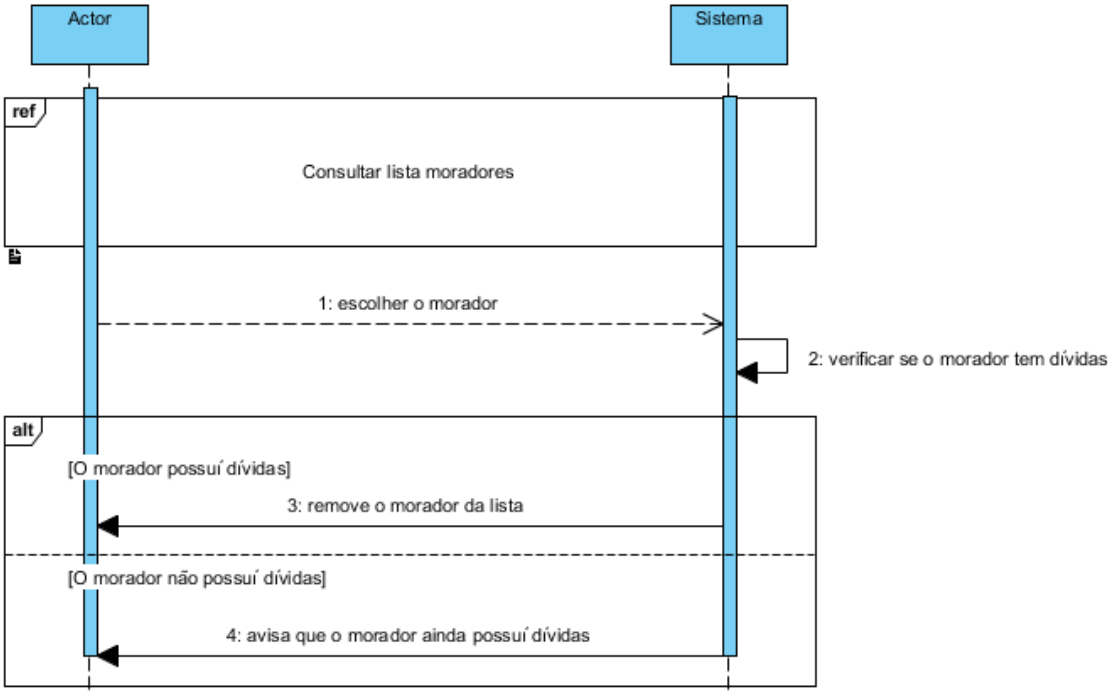

Figura 31 - Diagrama de Sequência com Actor/Sistema para UC Remover Morador

5.1.10 – Alterar Dados do Morador

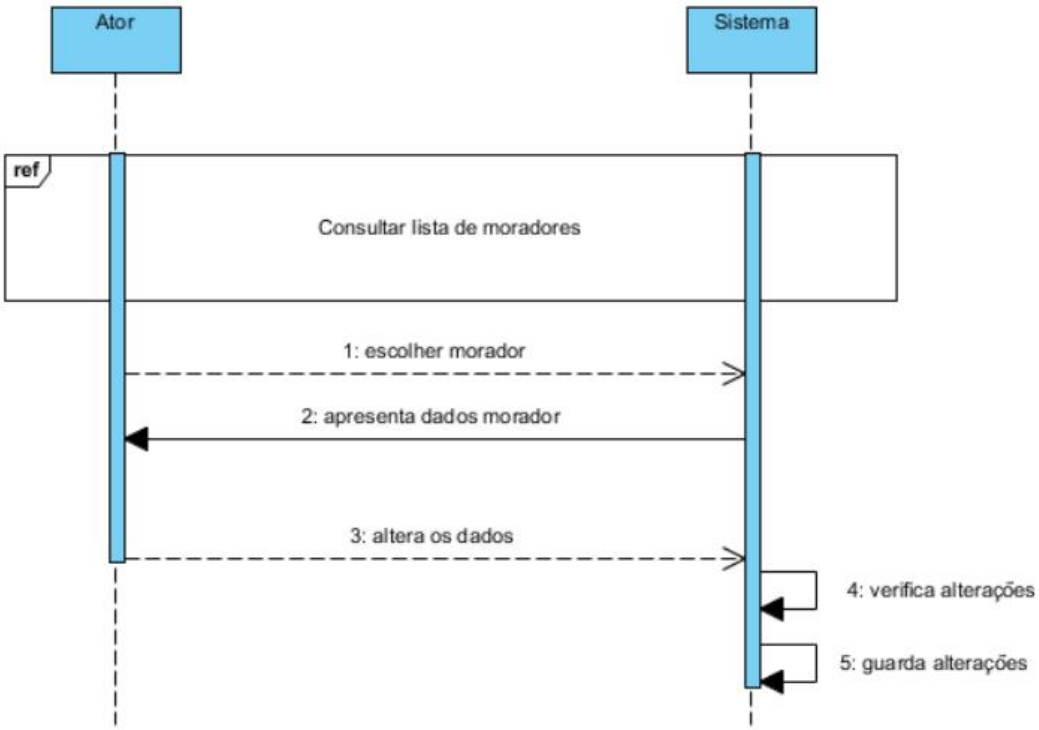

Figura 32 - Diagrama de Sequência com Actor/Sistema para UC Alterar Dados do Morador

5.1.11 – Consultar Dados do Morador

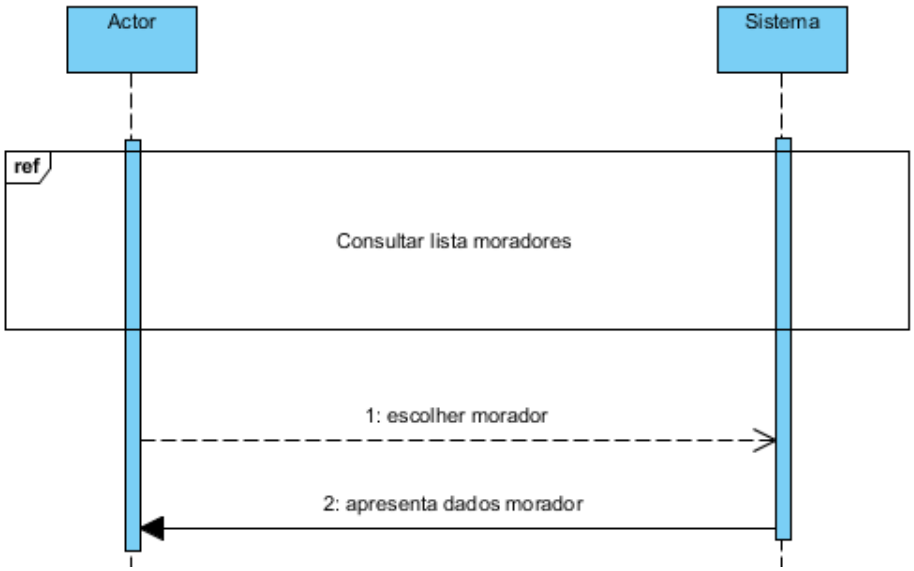

Figura 33 - Diagrama de Sequência com Actor/Sistema para UC Consultar Dados do Morador

5.2 – Diagramas com subsistemas

5.2.1 – Autenticar

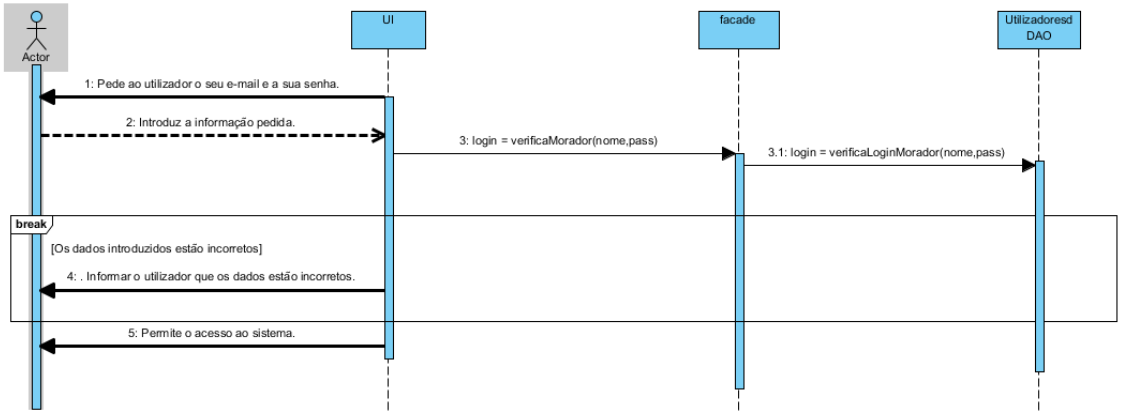

Figura 34 - Diagrama de Sequência com Subsistemas Autenticar

5.2.2 – Registar Despesa

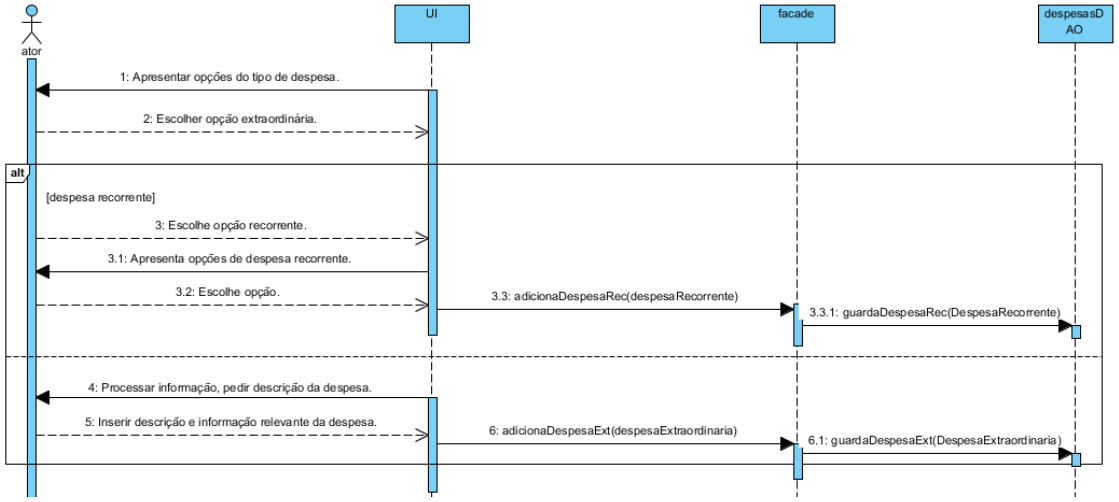

Figura 35 - Diagrama de Sequência com Subsistemas Registar Despesa

5.2.3 – Pagar Despesa

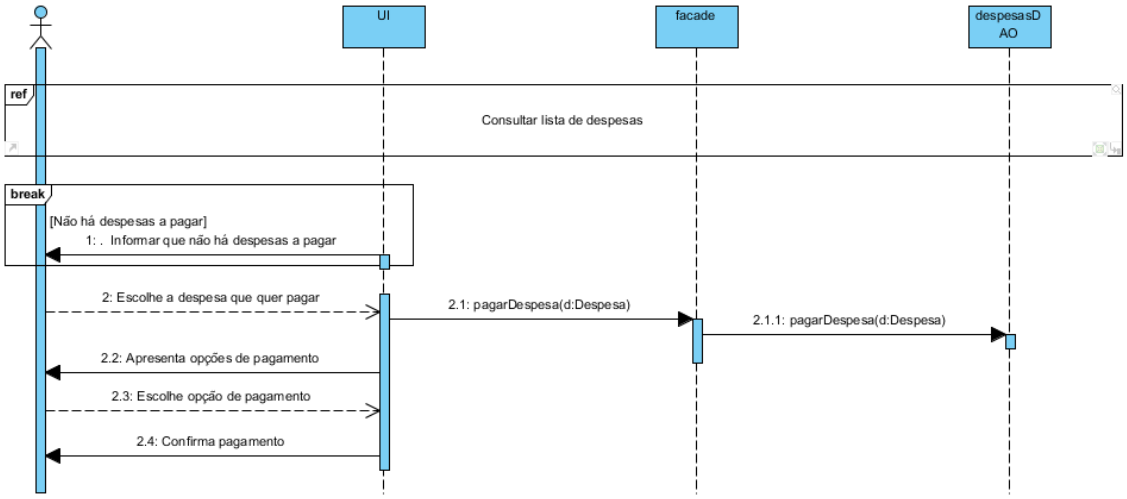

Figura 36 - Diagrama de Sequência com Subsistemas Pagar Despesa

5.2.4 – Consultar Lista de Despesas

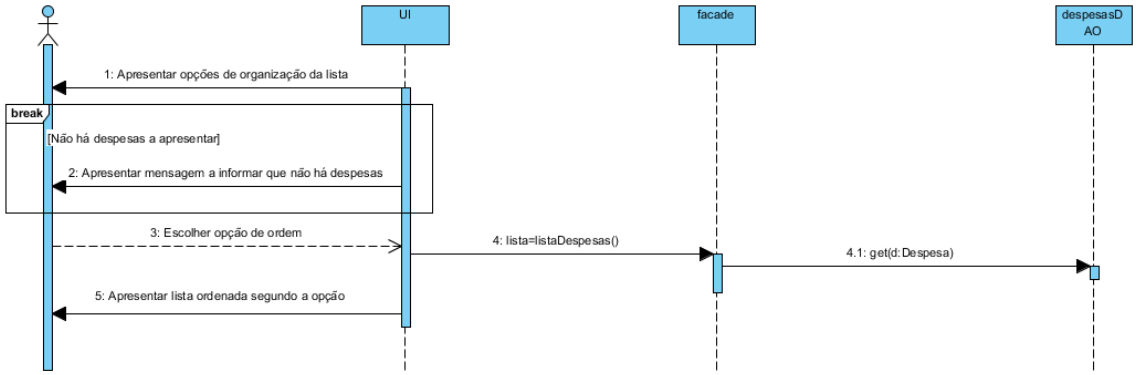

Figura 37 - Diagrama de Sequência com Subsistemas Consultar Lista de Despesas

5.2.5 – Consultar Lista de Pagamentos

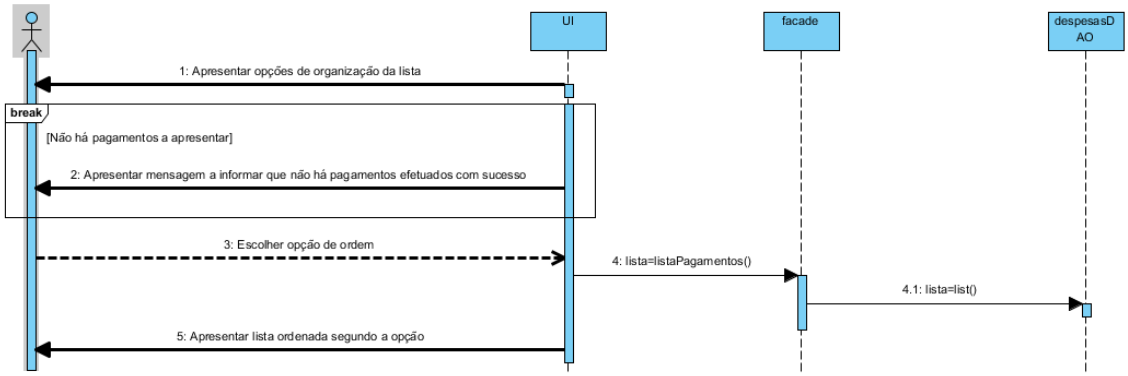

Figura 38 - Diagrama de Sequência com Subsistemas Consultar Lista de Pagamentos

5.2.6 – Calcular Montante a Pagar

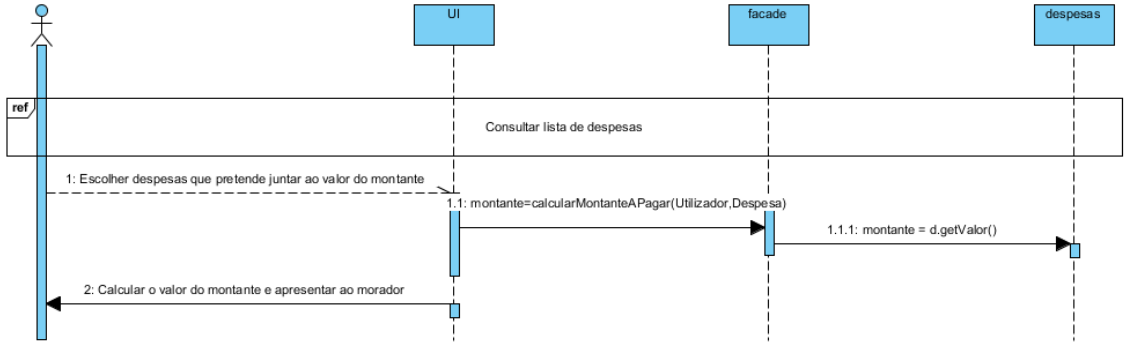

Figura 39 -Diagrama de Sequência com Subsistemas Calcular Montante a Pagar

5.2.7 – Consultar Lista de Moradores

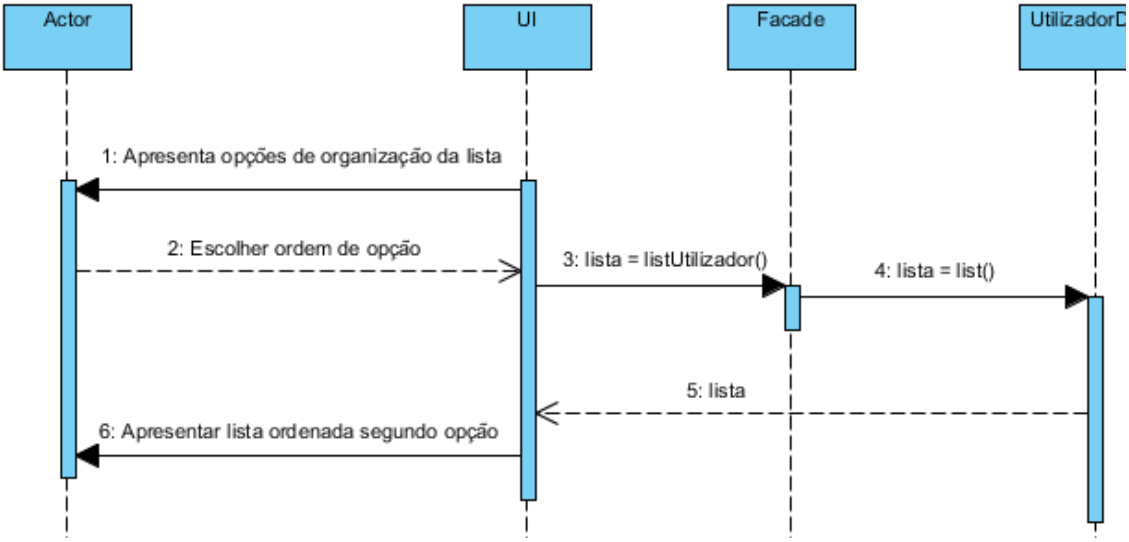

Figura 40 - Diagrama de Sequência com Subsistemas para UC Consultar Lista de Moradores

5.2.8 – Registar Morador

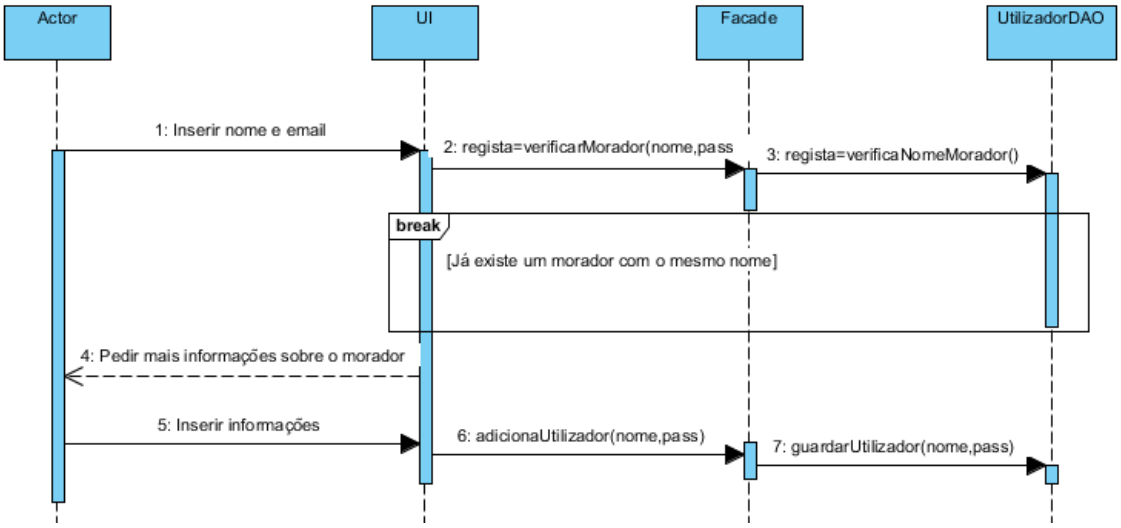

Figura 41 - Diagrama de Sequência com Subsistemas para UC Registar Morador

5.2.9 – Remover Morador

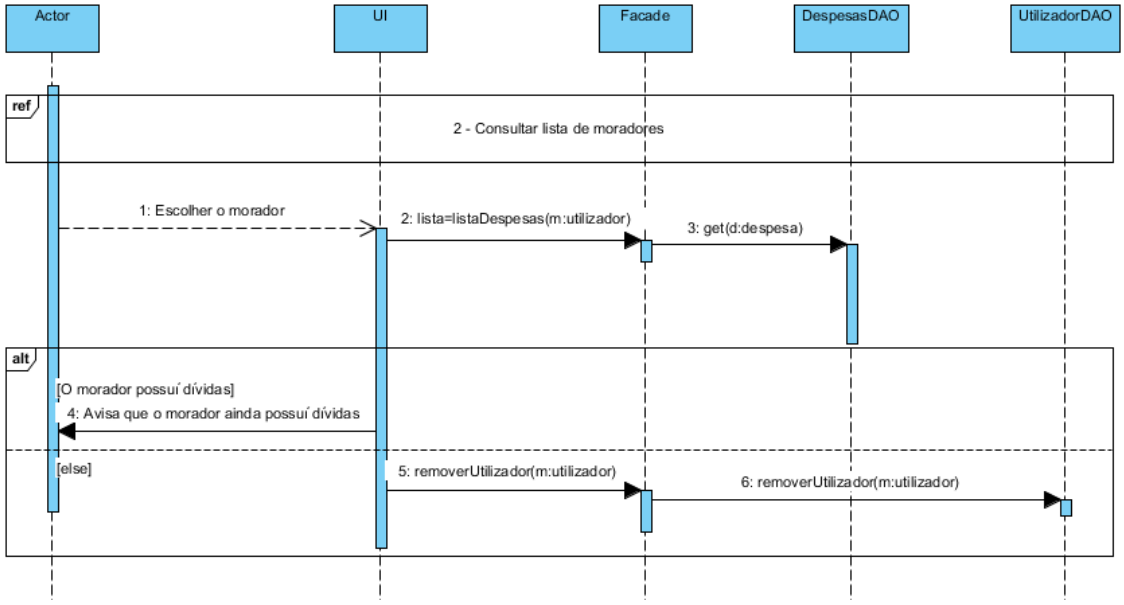

Figura 42 - Diagrama de Sequência com Subsistemas para UC Remover Morador

5.2.10 – Alterar Dados do Morador

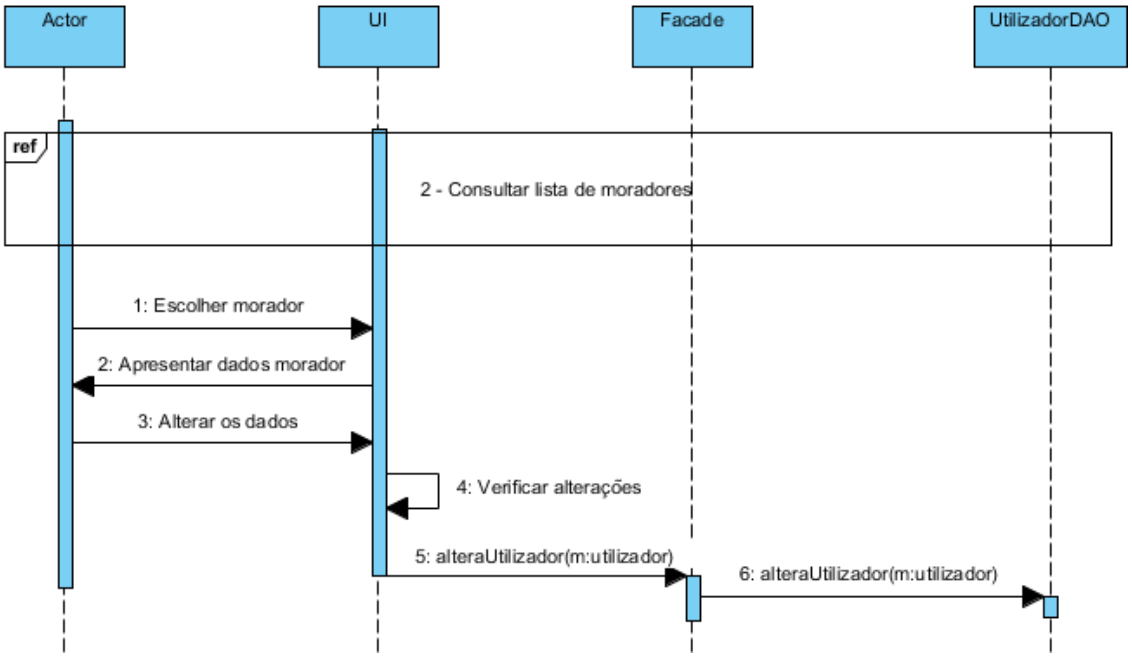

Figura 43 - Diagrama de Sequência com Subsistemas para UC Alterar Dados do Morador

5.2.11 – Consultar Dados do Morador

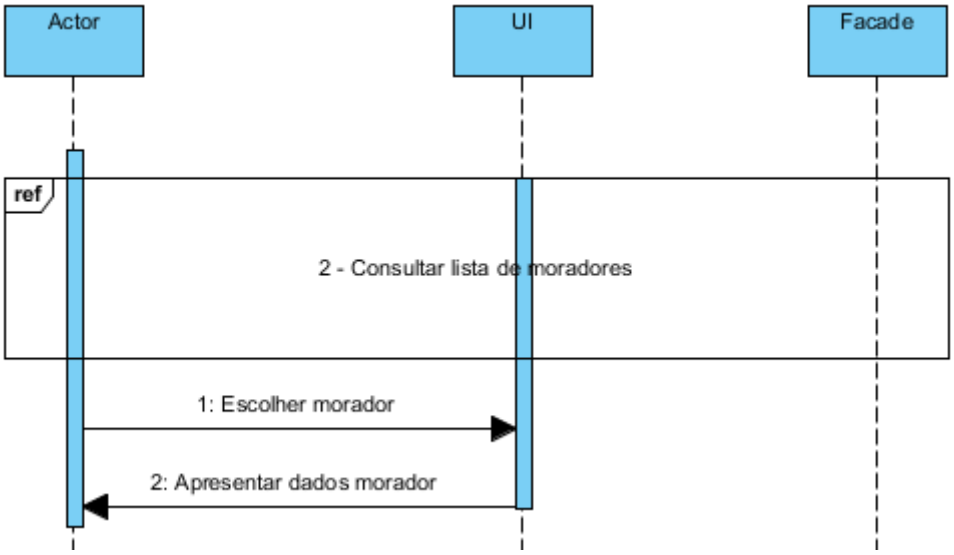

Figura 44 - Diagrama de Sequência com Subsistemas para UC Consultar Dados do Morador

6 – Diagramas de Package com subsistemas

De seguida, são apresentados os diagramas de *Package* que elaboramos para o nosso projeto.

Estes diagramas servem para representar os diferentes packages do projeto, bem como as relações existentes entre eles.

6.1 – Autenticar

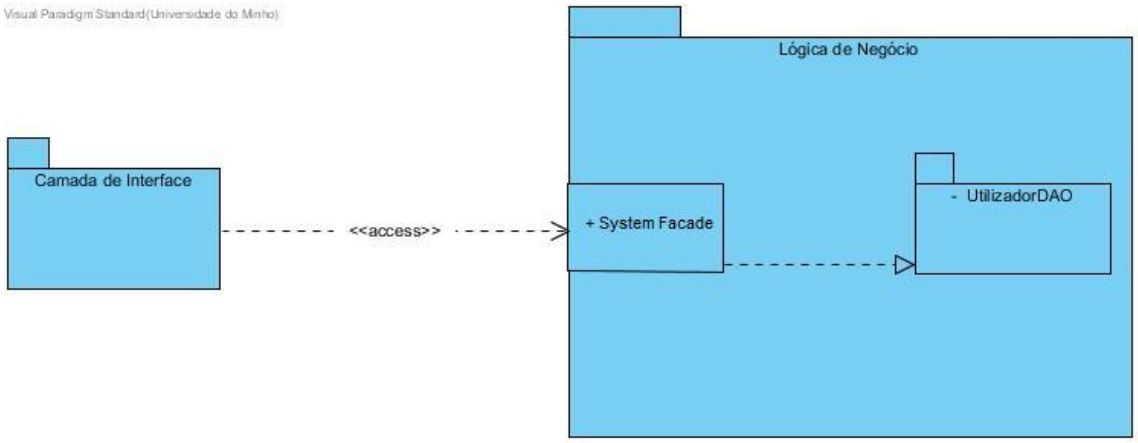

Figura 45 - Diagrama de Package Autenticar

6.2 – Registar Despesa

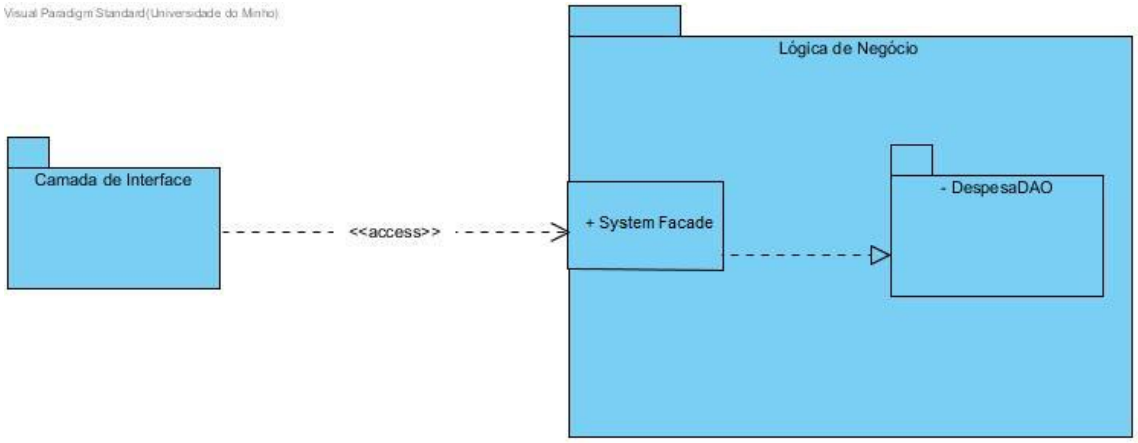

Figura 46 - Diagrama de Package Registar Despesa

6.3 – Consultar Lista de Despesas

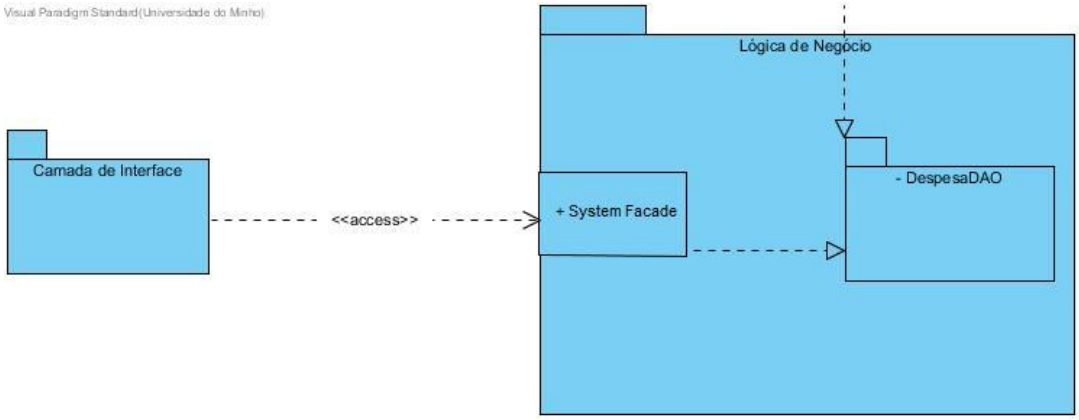

Figura 47 - Diagrama de Package Consultar Lista de Despesas

6.4 – Pagar Despesa

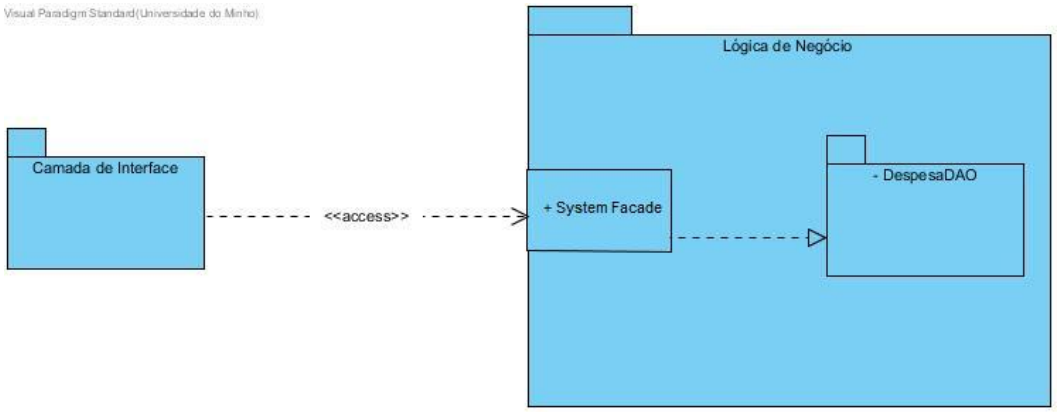

Figura 48 - Diagrama de Package Pagar Despesa

6.5 – Consultar Lista de Pagamentos

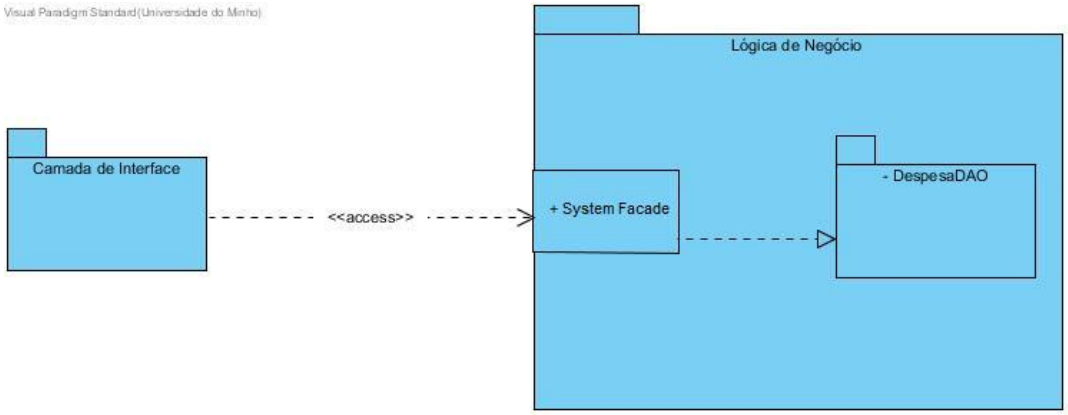

Figura 49 - Diagrama de Package Consultar Lista de Pagamentos

6.6 – Calcular Montante a Pagar

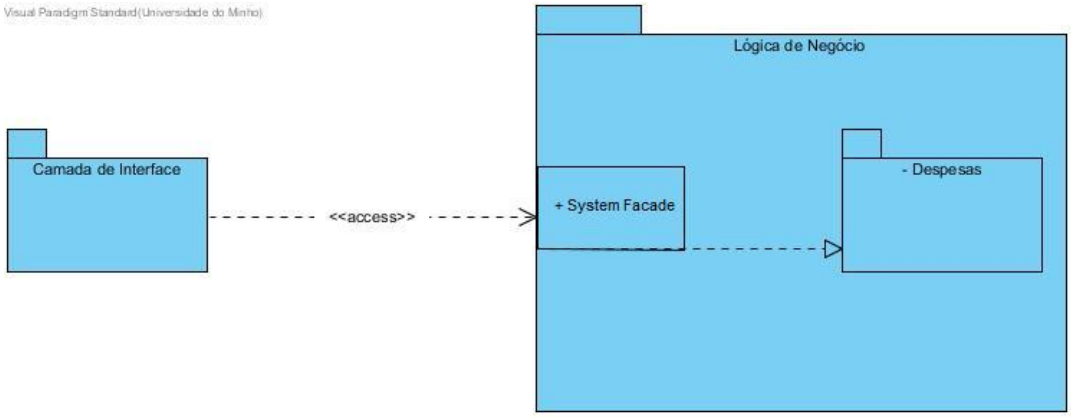

Figura 50 - Diagrama de Package Calcular Montante a Pagar

6.7 – Consultar Lista de Moradores

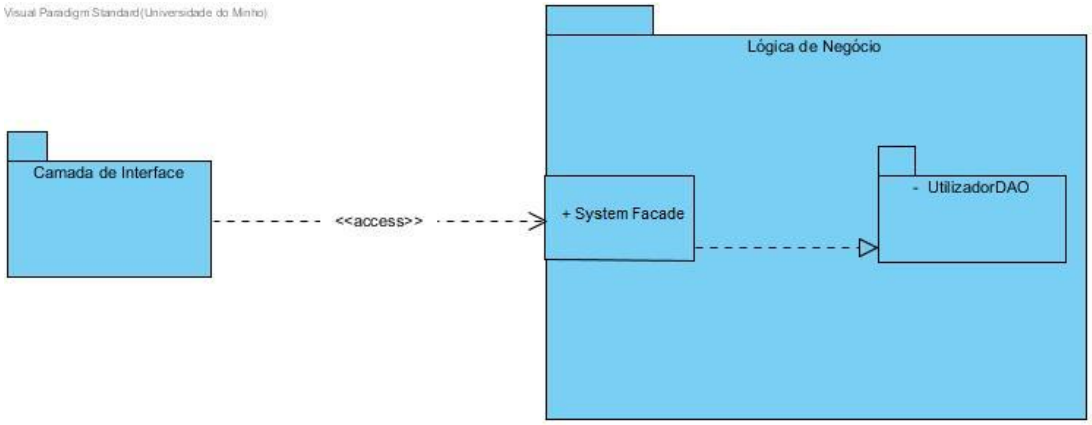

Figura 51 - Diagrama de Package Consultar Lista de Moradores

6.8 – Registar Morador

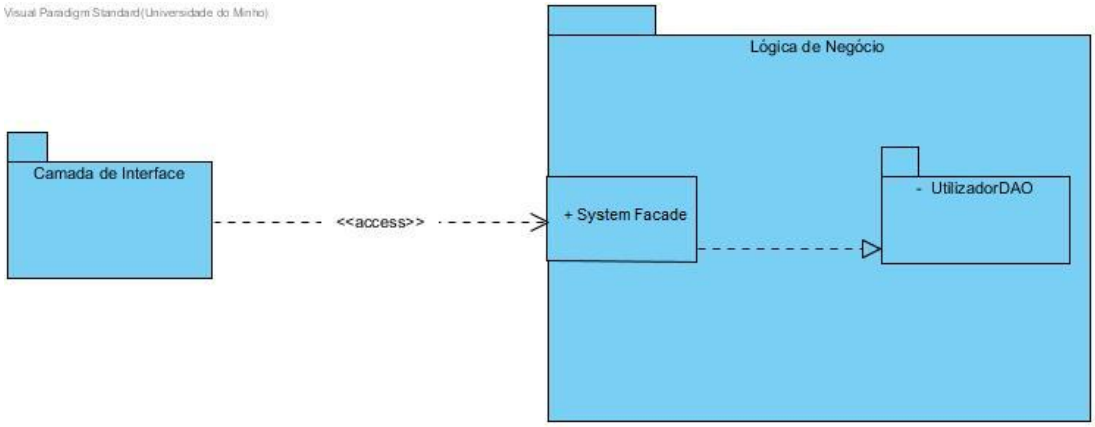

Figura 52 - Diagrama de Package Registar Morador

6.9 – Remover Morador

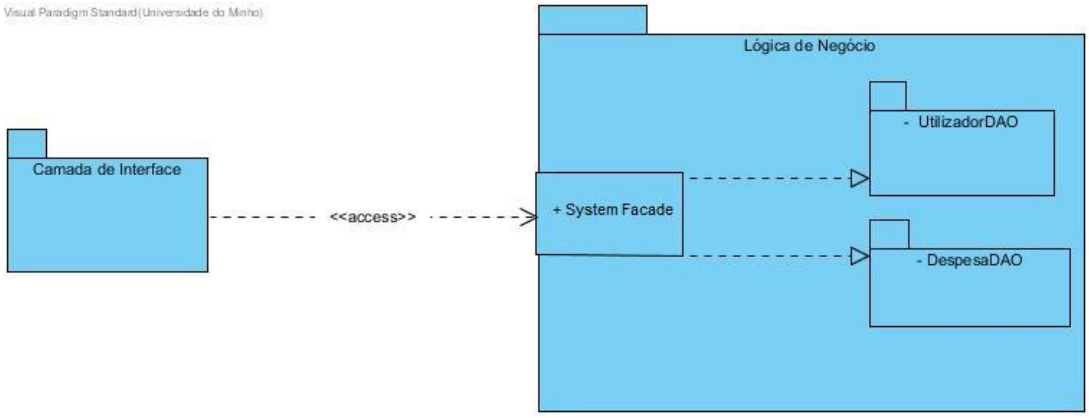

Figura 53 - Diagrama de Package Remover Morador

6.10 – Alterar Dados do Morador

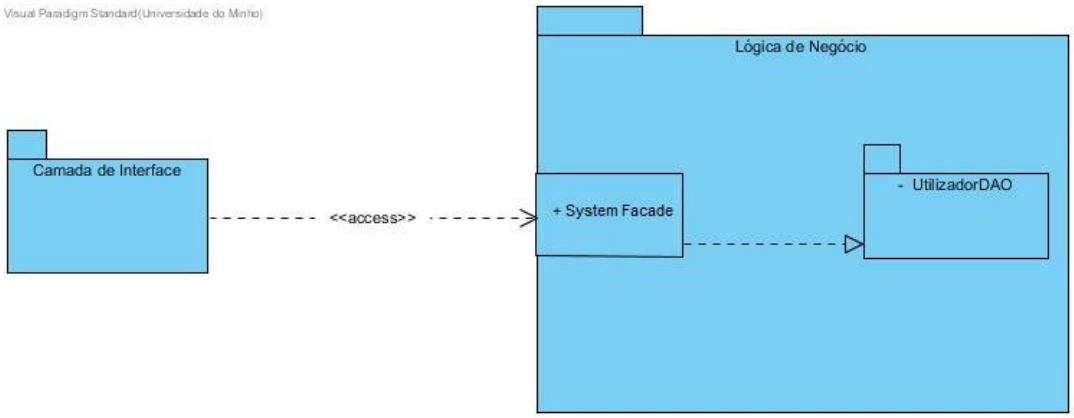

Figura 54 - Diagrama de Package – Alterar Dados Morador

6.11 – Consultar Dados do Morador

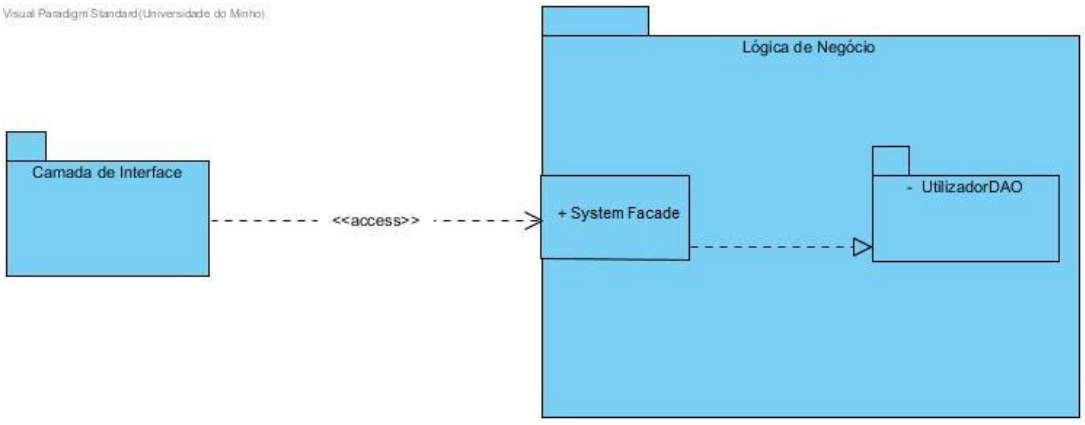

Figura 55 - Diagrama de Package Consultar Dados do Morador

7 – Diagrama de Classes

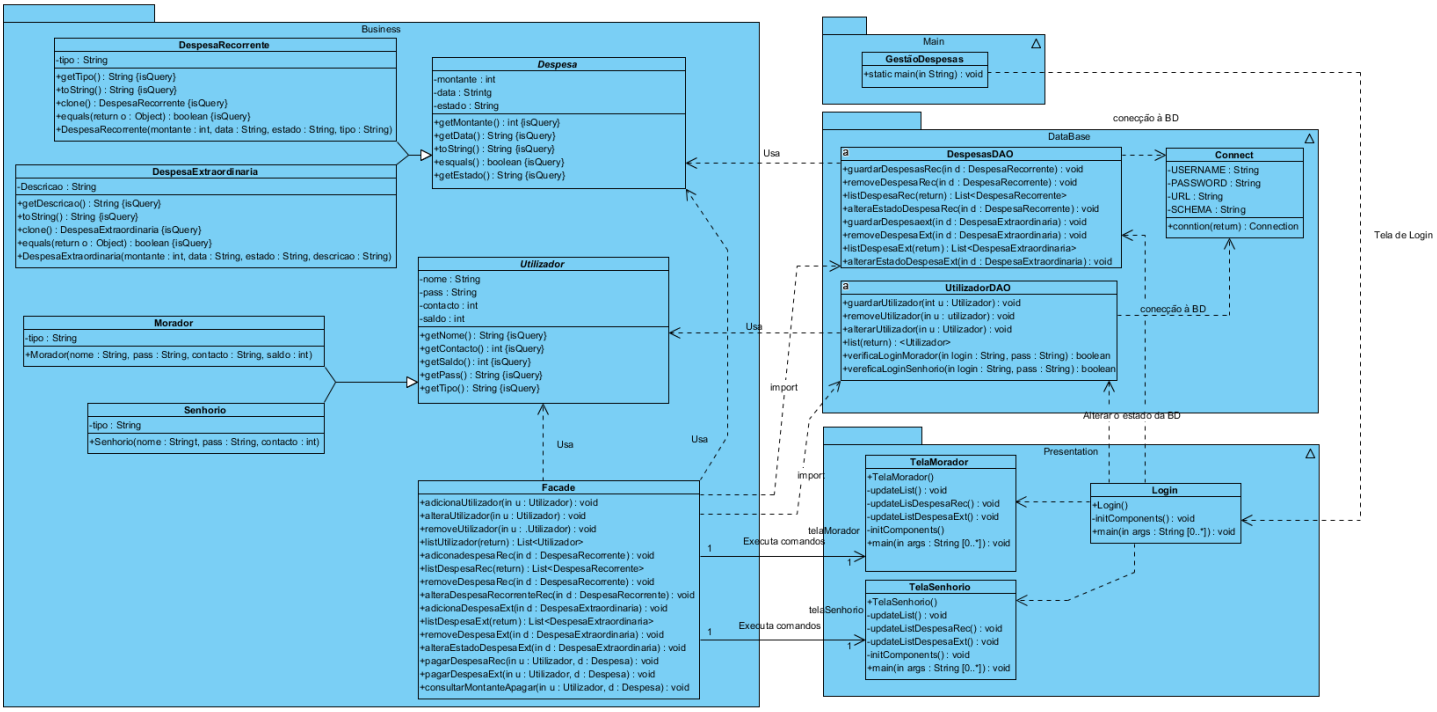

Figura 56 - Diagrama de Classes

No diagrama de classes optamos por dividir os vários módulos do programa em pastas separadas como se pode ver na imagem, de modo a organizar o código, facilitando a sua leitura/interpretação.

No *packag*e Business temos as duas classes essenciais do trabalho e por onde começamos a fazer o código, nomeadamente as entidades Despesa e Utilizador, e a classe Facade que permite a ligação entre o módulo da apresentação e da Base de Dados, sendo responsável pela execução dos *Use Cases*.

No *package* DataBase temos a classe Connect que permite a ligação entre o programa e a Base de Dados, permitindo que os métodos das classes DespesasDAO e UtilizadorDAO manipulem a Base de Dados.

No *package* Main temos a classe de execução do programa que inicia a tela de login para os utilizadores poderem autenticar-se e usufruir dos serviços. Para os dois tipos de utilizador foram criadas duas classes de tela/janela, uma vez que o senhorio tem mais liberdade para alterar o estado do programa, enquanto que o utilizador é limitado a alterar o estado dos seus dados e de algumas despesas.

8 – Diagrama de Instalação

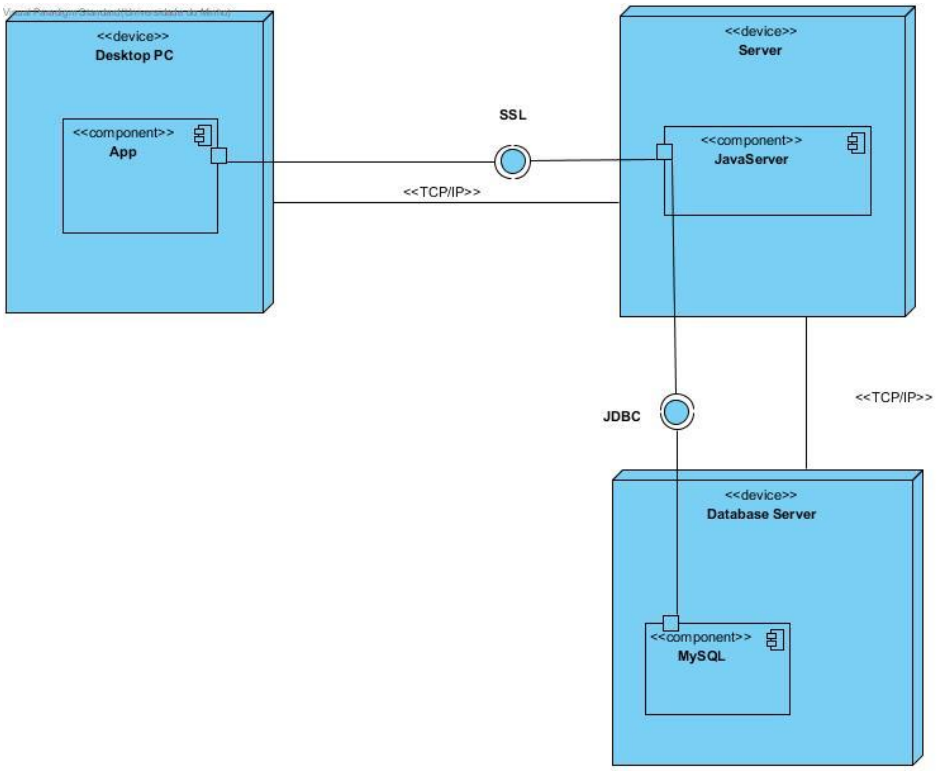

Figura 57 - Diagrama de Instalação

9 – Diagramas de Atividade

Neste capítulo estão os diagramas de atividades para cada Use Case correspondente.

9.1 – Autenticar

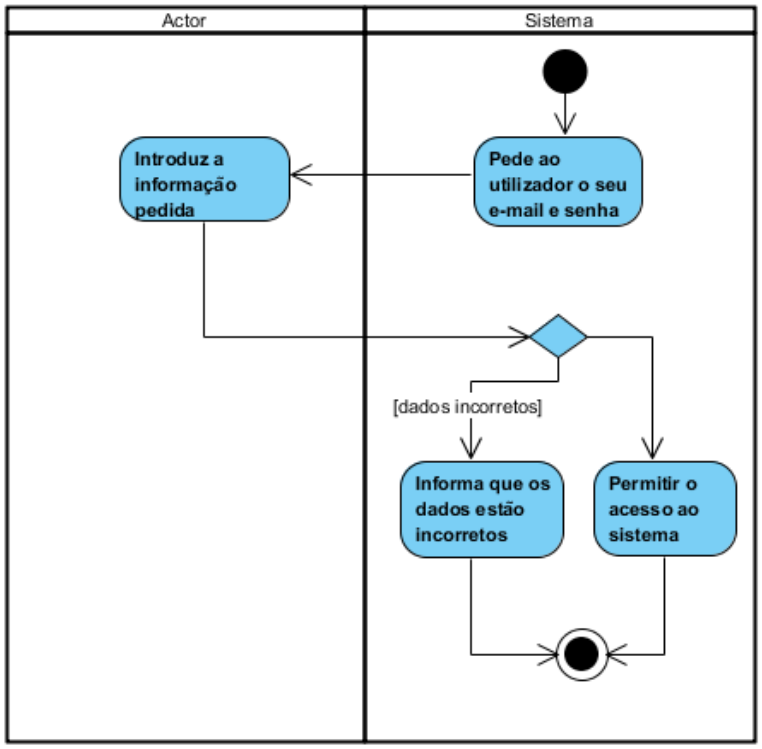

Figura 58 - Diagrama de Atividade para UC Autenticar

9.2 – Registar Despesa

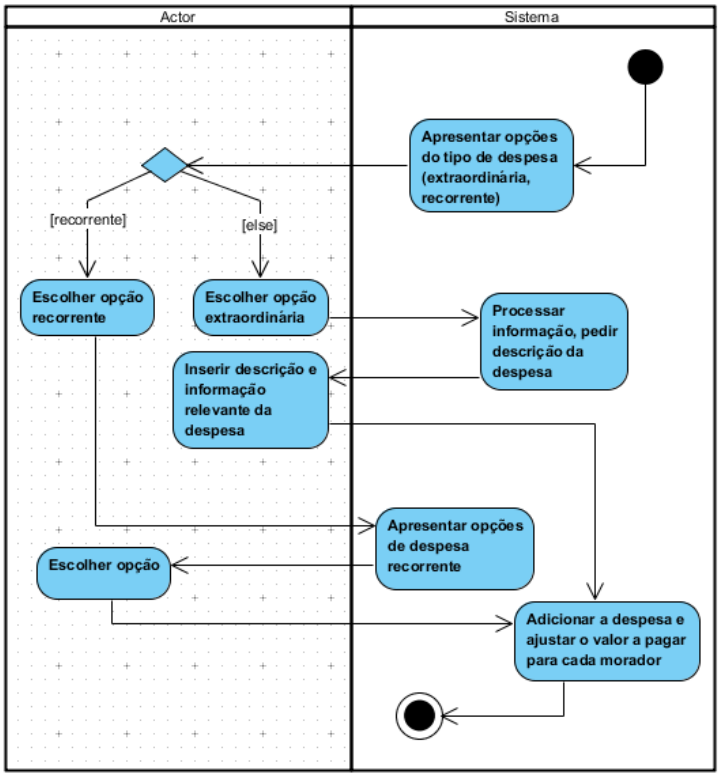

Figura 59 - Diagrama de Atividade para UC Registar Despesa

9.3 – Consultar Lista de Despesas

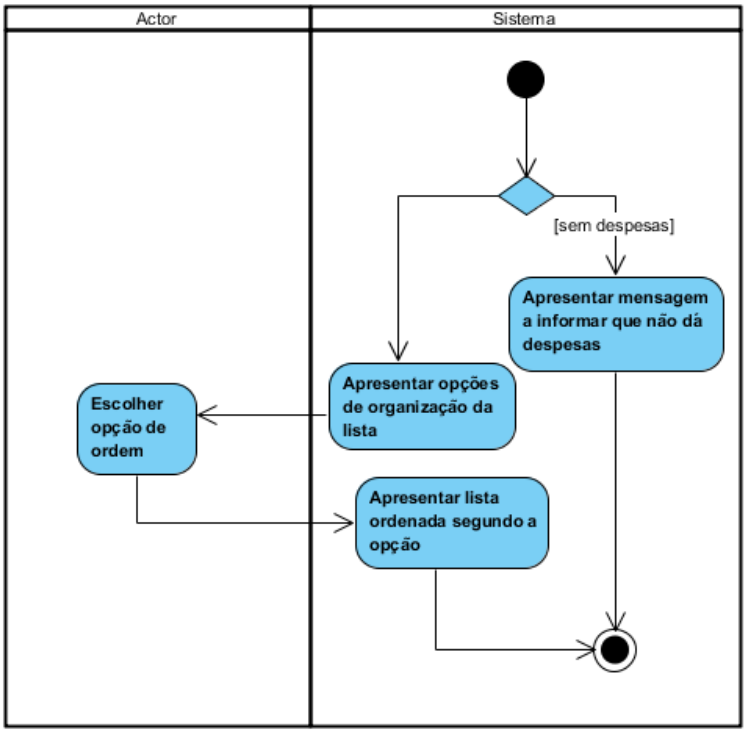

Figura 60 - Diagrama de Atividade para UC Consultar Lista de Despesas

9.4 – Pagar Despesa

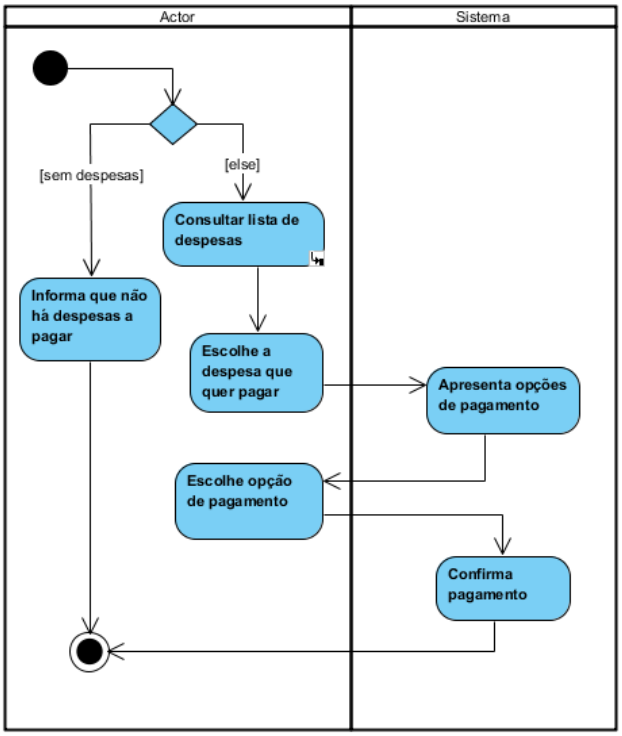

Figura 61 -Diagrama de Atividade para UC Pagar Despesa

9.5 – Consultar Lista de Pagamentos

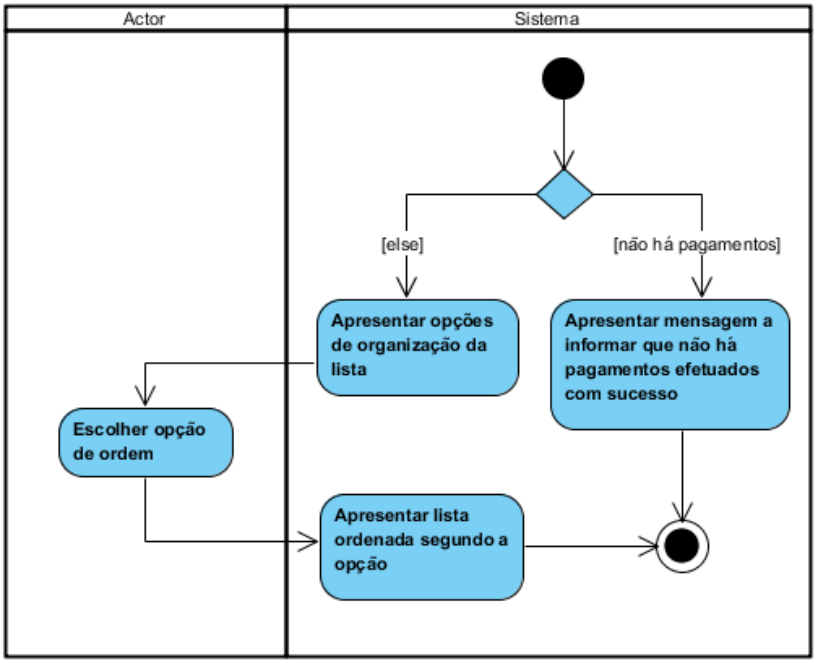

Figura 62 - Diagrama de Atividade para UC Consultar Lista de Pagamentos

9.6 – Calcular Montante a Pagar

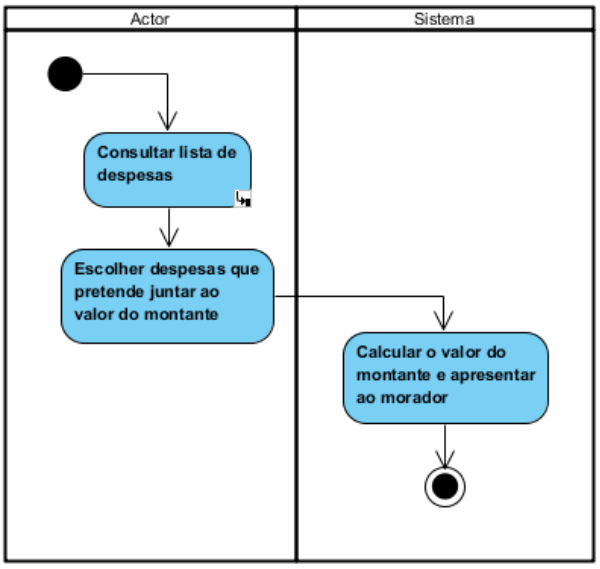

Figura 63 - Diagrama de Atividade para UC Calcular Montante a Pagar

9.7 – Consultar Lista de Moradores

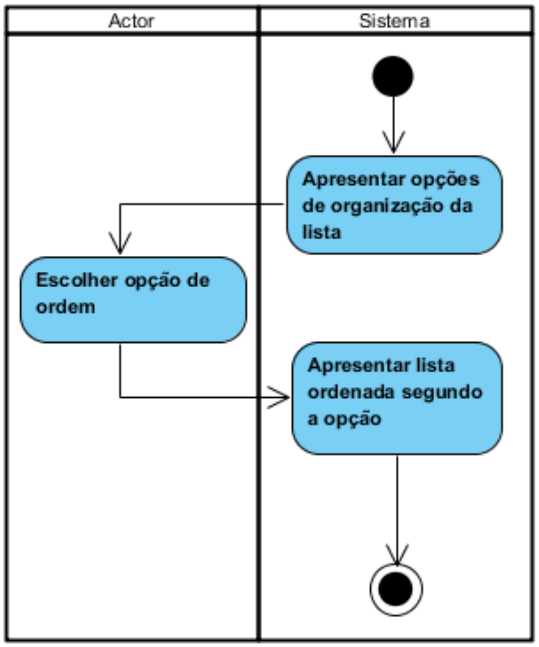

Figura 64 - Diagrama de Atividade para UC Consultar Lista de Moradores

9.8 – Registar Morador

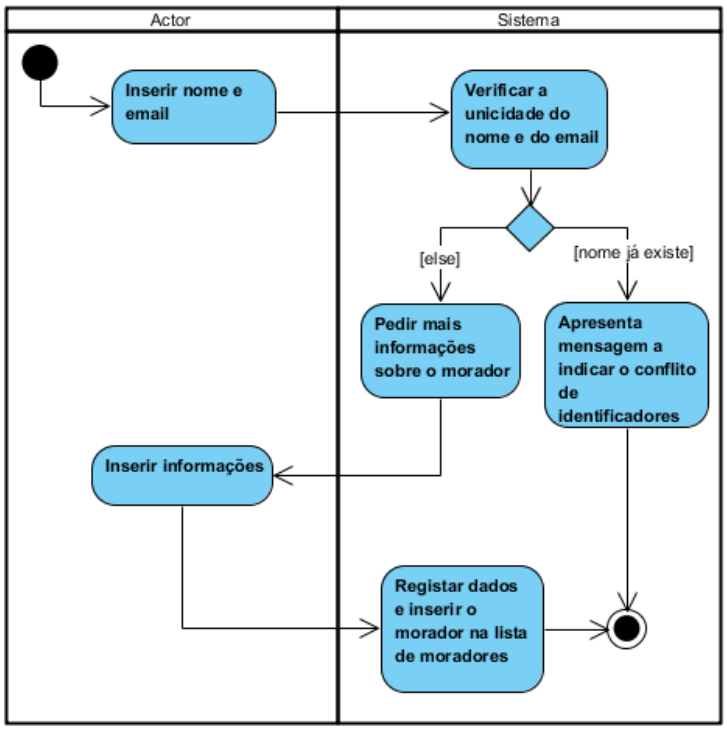

Figura 65 - Diagrama de Atividade para UC Registar Morador

9.9 – Remover Morador

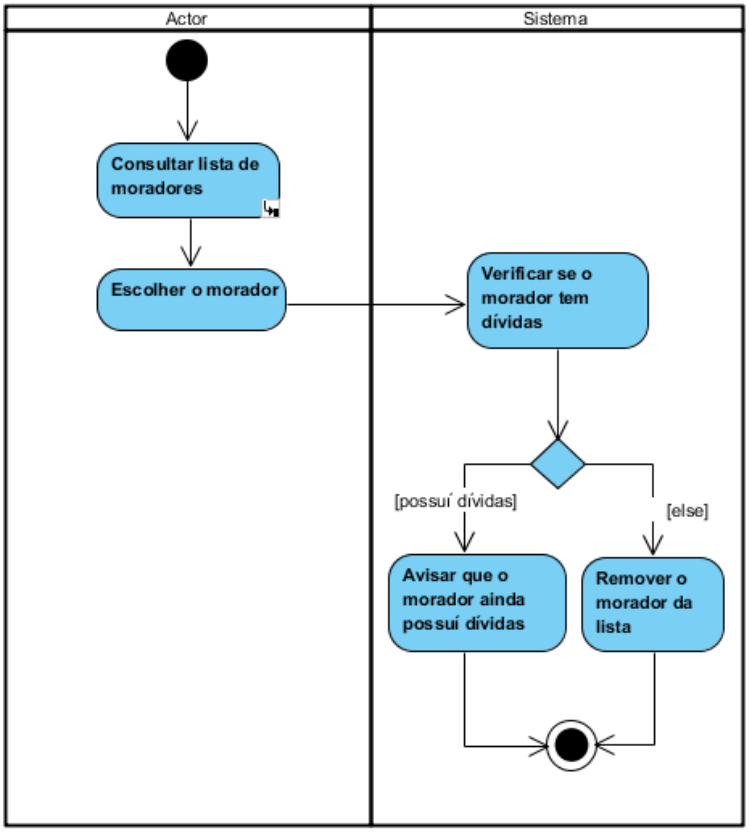

Figura 66 - Diagrama de Atividade para UC Remover Morador

9.10 – Alterar Dados do Morador

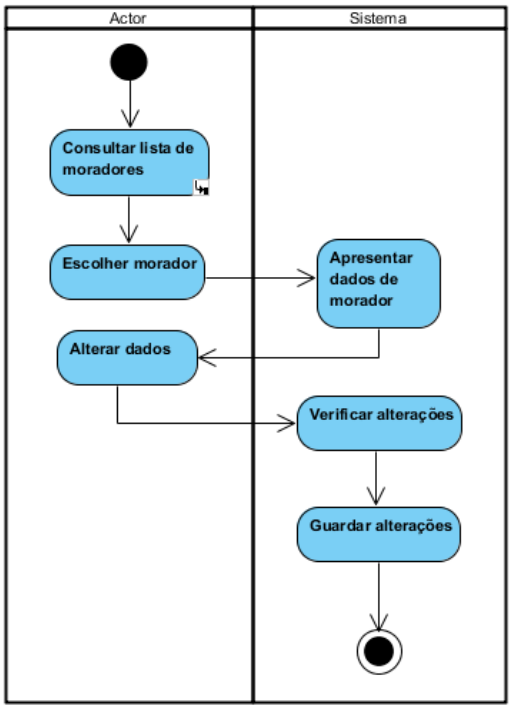

Figura 67 - Diagrama de Atividade para UC Alterar Dados do Morador

9.11 – Consultar Dados do Morador

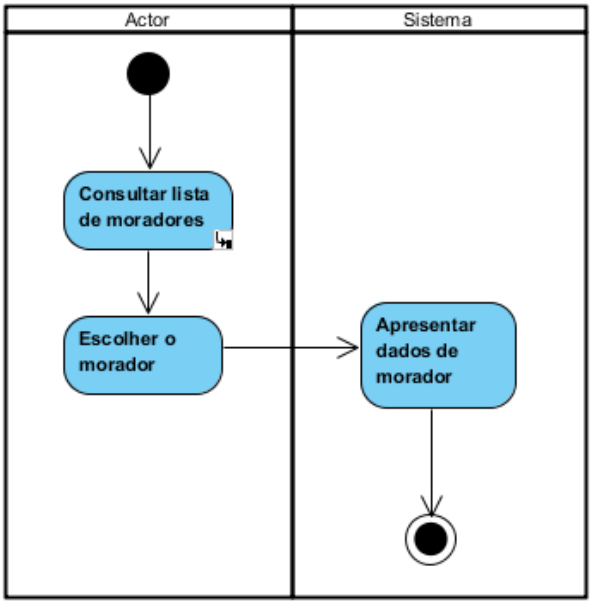

Figura 68 - Diagrama de Atividade para UC Consultar Dados do Morador

9 – Programa (Aplicação)

9.1 – Camada de negócio

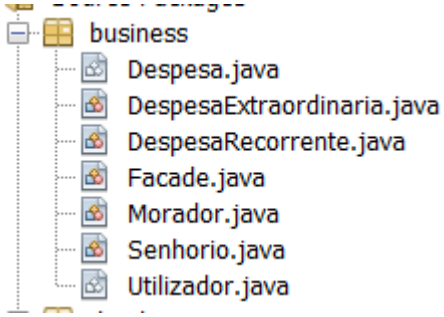

Figura 69 - Camada de negócio

Para o desenvolvimento da aplicação, consideramos a existência de duas classes abstratas: **Despesa** e **Utilizador**. A classe Despesa tem duas subclasses: **DespesaExtraordinária** e **DespesaRecorrente**. Por sua vez, a classe Utilizador tem as subclasses **Morador** e **Senhorio**.

Foi também criado um **Facade** para podermos "ligar" a camada de negócio com a camada dos dados.

9.2 – JDBC e DAO's

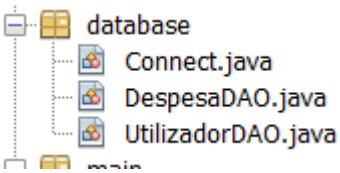

Figura 70 - Camada de Dados

Para a camada de dados foram criadas as classes **Connect**, **DespesaDAO** e **UtilizadorDAO**.

Em seguida mostramos um esboço das tabelas que foram criadas na nossa base de dados e que está implementada no projeto:

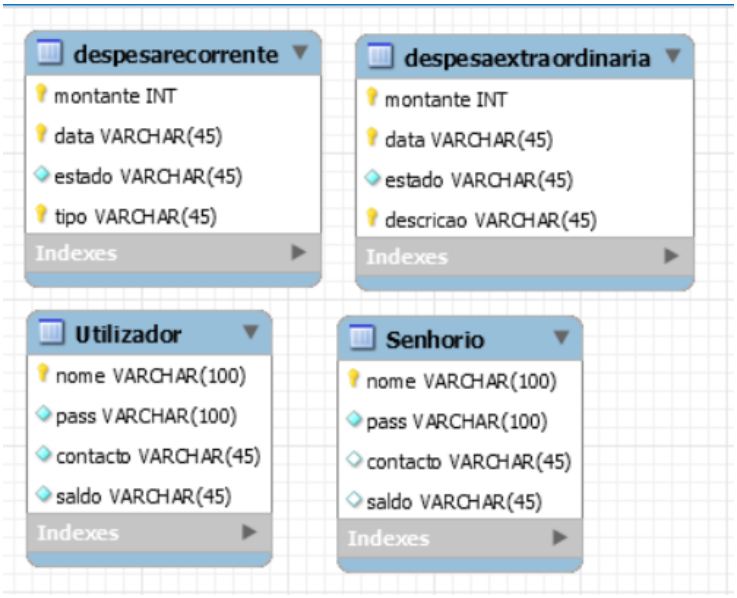

Figura 71 - Tabelas da Base de Dados

9.3 – Interface

9.3.1 – Login

Foi criada uma tela de login onde os moradores do apartamento e o senhorio tem que preencher para se poderem autenticar.

Foi considerado por defeito que os dados para entrar como senhorio são **login**: admin e **password**: admin.

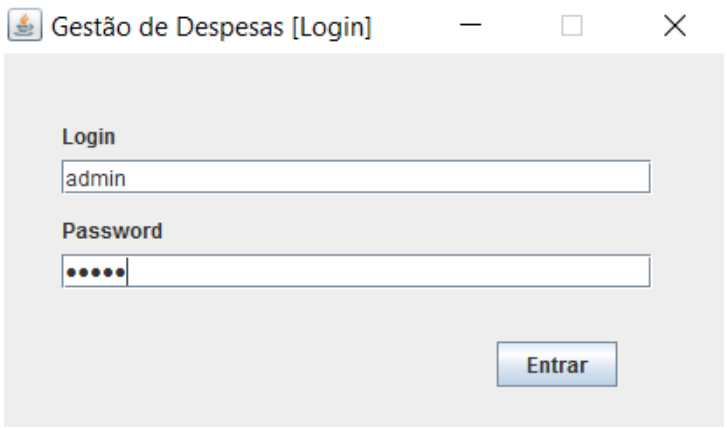

Figura 72 - Tela de Login

Caso os dados preenchidos sejam incorretos, a tela emite uma mensagem a dizer isso mesmo, tal como podemos ver na figura em baixo.

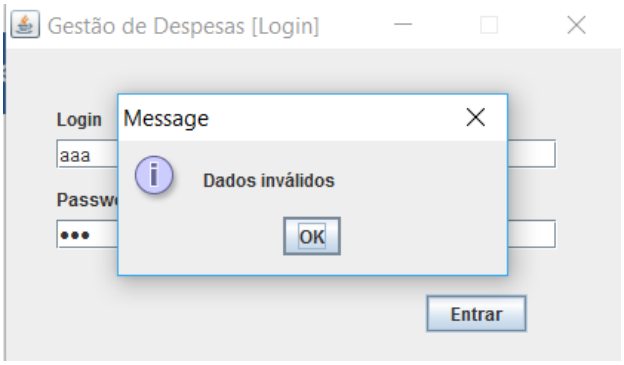

Figura 73 - Dados inválidos no login

9.3.2 – Tela do Senhorio

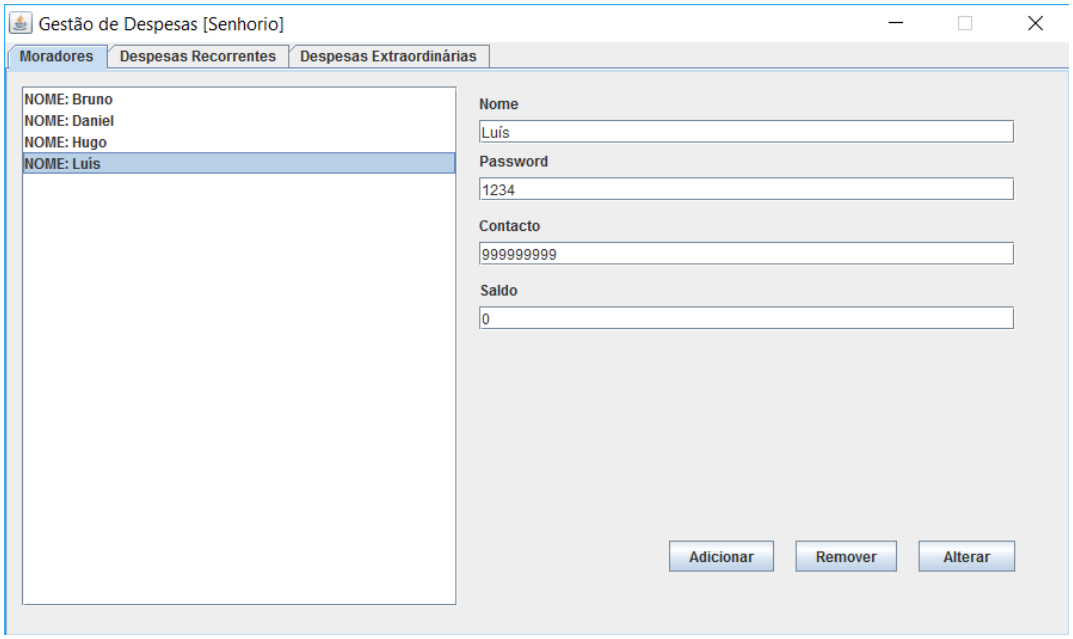

Figura 74 - Tela Senhorio (Moradores)

O senhorio pode adicionar e remover um morador do sistema, bem como alterar os seus dados, como podemos verificar na figura em cima. Este tem sempre acesso a todos os dados de cada morador uma vez que é ele quem gere o apartamento.

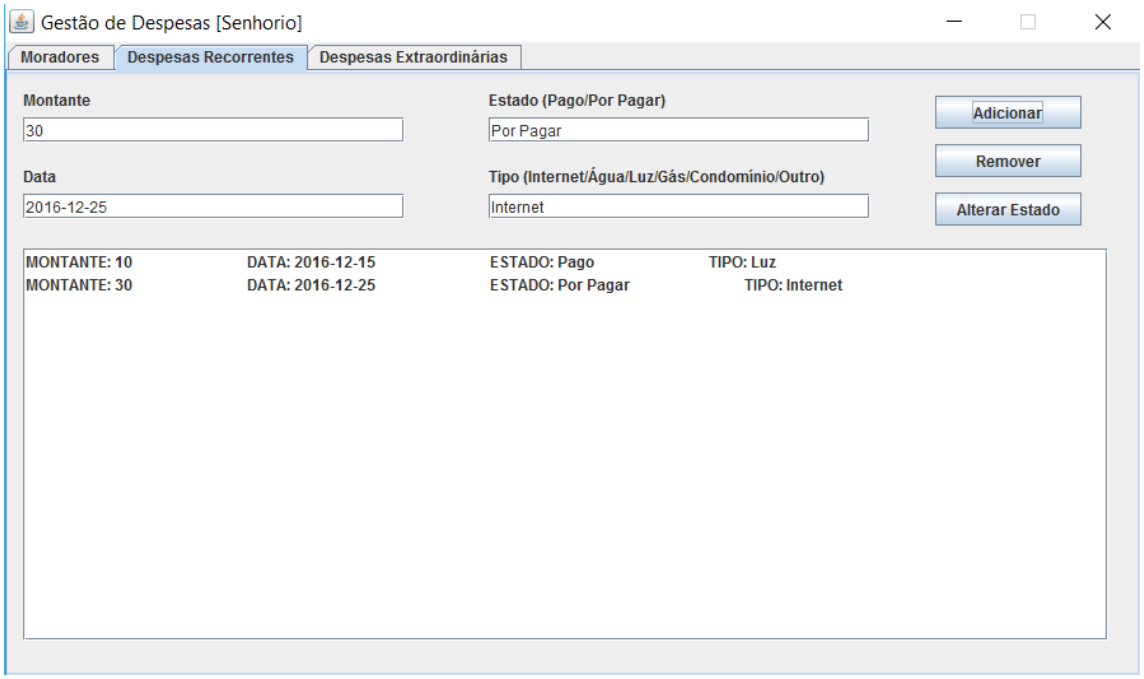

Figura 75 - Tela Senhorio (Despesas Recorrentes)

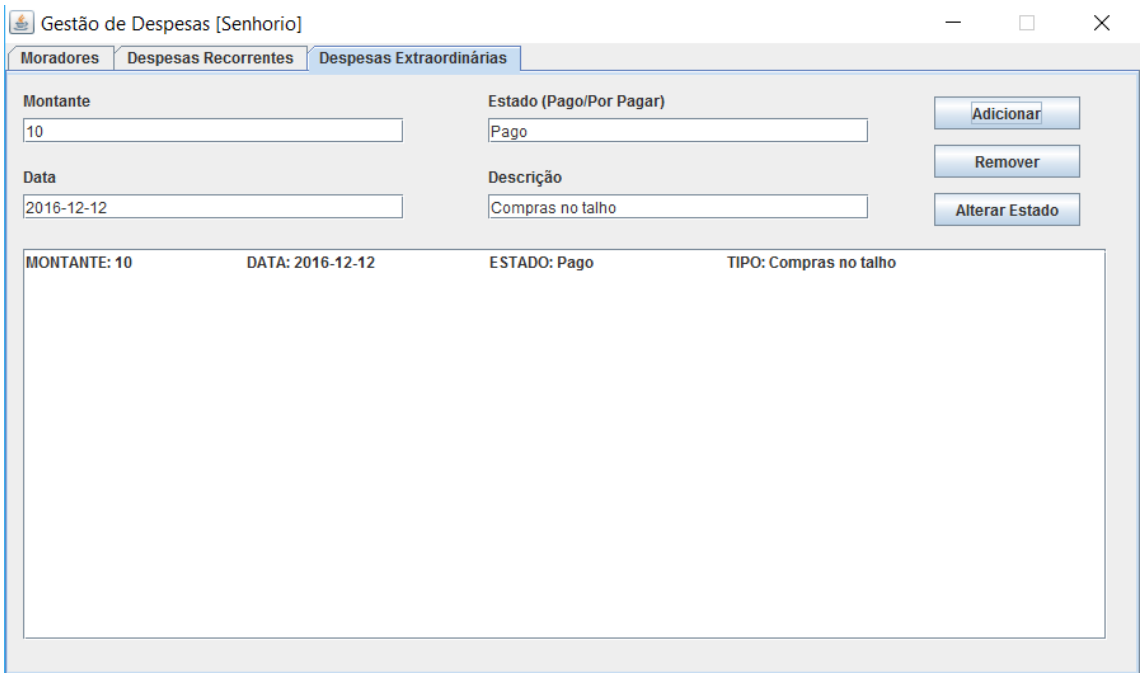

Figura 76 - Tela Senhorio (Despesas Extraordinárias)

Em ambos os tipos de despesa, o senhorio pode adicionar e remover uma despesa, bem como alterar o estado desta.

9.3.3 – Tela do Morador

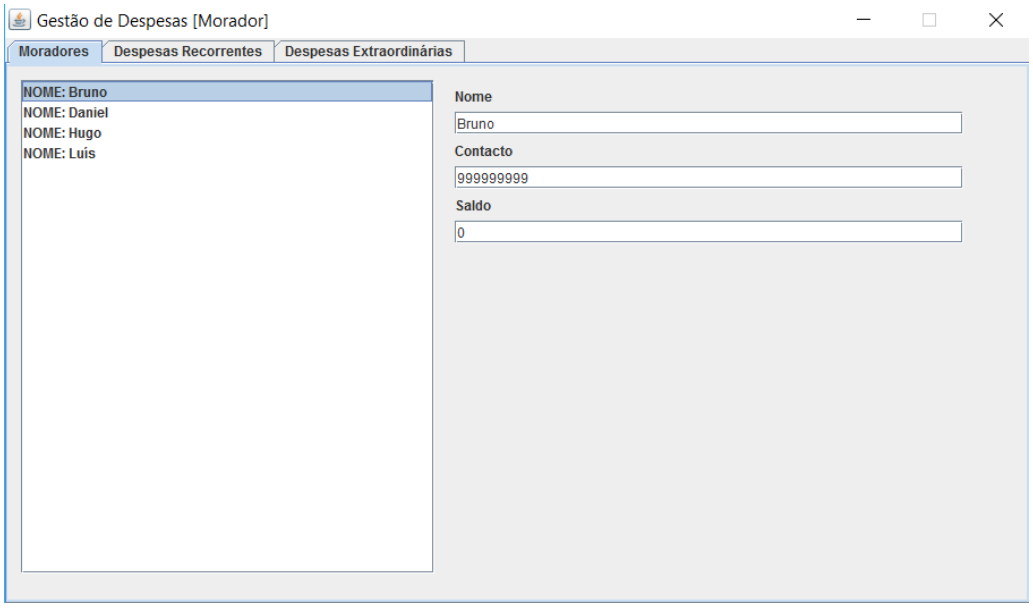

Figura 77 - Tela Morador (Moradores)

Ao contrário do senhorio, o morador apenas pode consultar os dados nome, contacto e saldo de cada um dos moradores. Assim, este não tem privilégios de poder adicionar, remover ou alterar um morador, e também não pode consultar a sua password.

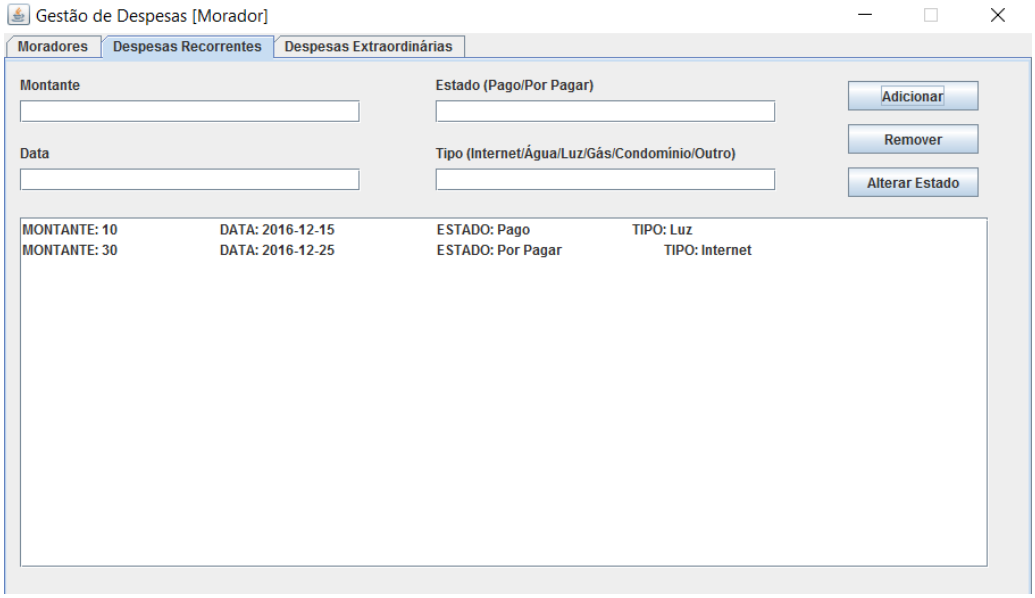

Figura 78 - Tela Morador (Despesas Recorrentes)

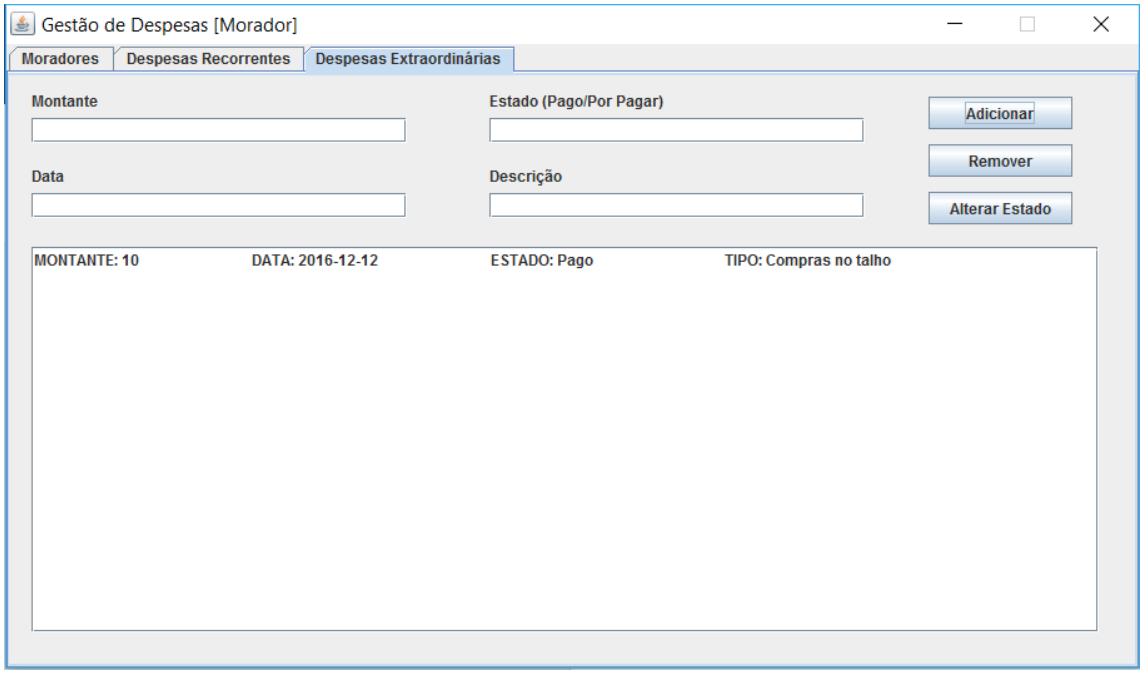

Figura 79 - Tela Morador (Despesas Extraordinárias)

O morador tem os mesmos privilégios que o senhorio em relação às despesas, isto é, pode adicionar, remover e alterar o estado de uma despesa recorrente ou extraordinária.

Conclusão

Nume primeira fase do projeto prático, foi elaborado um modelo de domínio com as várias entidades relevantes e importantes para o sistema, um modelo de *Use Case* que representa o comportamento entre os atores e o sistema, as várias funcionalidades para a gestão das despesas, bem como as suas respetivas especificações.

Na primeira fase de observação e estudo do objetivo do projeto, construímos um modelo muito simples e bastante incompleto que não cumpria todos os requisitos pedidos. Numa posterior e melhor análise do projeto a desenvolver, e após identificarmos os requisitos fundamentais para o sistema, chegamos a um melhor e mais completo sistema que serve de base à partilha de despesas. Sistema este que garante o registo de diversos tipos de despesas, a gestão do seu pagamento, assim como a sua repartição pelos vários moradores registados.

Após construirmos o modelo de domínio, os *Use Cases* e as suas respetivas especificações, passamos para a modelação do comportamento dos diferentes utilizadores (neste caso Senhorio e Morador), elaborando assim os diagramas de máquinas de estado. Estes diagramas foram úteis para podermos modelar o comportamento do sistema como um todo, especialmente na ligação da interface com o utilizador.

De seguida, passamos à elaboração dos diagramas que sequência para cada *Use Case*, que nos permitiu descrever o conjunto de interações do utilizador com o sistema para poder realizar o *Use Case* pretendido. Nestes diagramas, passamos por duas fases: numa primeira implementação, construímos o diagrama apenas com comunicação ator/sistema e de seguida construímos uma implementação com a existência de diferentes subsistemas.

Após a implementação dos diagramas de sequência com subsistemas, construímos os diagramas de package correspondentes a cada diagrama de sequência, o que nos permitiu representar os diferentes *packages* do sistema, bem como as suas relações.

Seguidamente, foi elaborado um diagrama de classes, que nos permitiu identificar as diferentes classes necessárias para a implementação do sistema, bem como os seus métodos e associações.

Implementamos também os diagramas de atividade correspondentes a cada *Use Case,* que foram úteis para especificar os diferentes comportamentos. Foi também elaborado um diagrama de instalação do sistema.

Por fim, utilizando o Java Swing do IDE Netbeans, construímos uma simples implementação do sistema com base nos diagramas construídos anteriormente. Nesta etapa do trabalho, não concluímos a implementação de todos os *Use Cases*, implementando apenas alguns dos que foram apresentados.