Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Desenvolvimento de Sistemas de Software

Ano Letivo de 2016/2017

Trabalho Prático: "Bill Splitter"

A27748 - Gustavo José Afonso Andrez

A74634 - Rogério Gomes Lopes Moreira

A67664 - Samuel Gonçalves Ferreira

30 de dezembro 2016

Docentes:

Doutor José Francisco Creissac Freitas Campos

Doutor António Manuel Nestor Ribeiro

Índice

1. Relatório

1.1. Introdução

Este trabalho consiste no planeamento de um sistema de suporte à gestão dos custos associados a um conjunto de pessoas que vivem na mesma habitação. Assim, a aplicação deverá ser capaz de registar despesas e respetivos pagamentos, associar moradores e indicar os valores que cada pessoa tem a despender ou receber.

Para o desenvolvimento deste trabalho fez-se recurso do programa Visual Paradigm tendo sido desenvolvidos:

- Diagrama de Classes;
- Modelo de Domínio;
- Diagrama de Use Case;
- Máquinas de estado
- Diagrama de Package
- Diagramas de sequência
- Diagrama de Objetos
- Diagrama de Instalação
- Diagrama de Componentes

Foi feito um esboço da interface (mockups) para o utilizador no programa PowerPoint.

Recorrendo ao programa MySQL foi desenvolvida a base de dados de suporte ao programa a desenvolver. A interface gráfica do usuário (GUI) foi implementada utilizando GUI Builder do NetBeans IDE. As funcionalidades foram implementadas em linguagem java recorrendo ao programa Netbeans.

1.2. Considerações iniciais

Neste ponto do trabalho apresentam-se as funcionalidades que, após discussão, foram selecionadas para fazer parte da aplicação “Bill Splitter”

Como ponto de partida tivemos em consideração que o sistema tem que ser simples de usar uma vez que o público-alvo, maioritariamente estudantes, só irá usar a ferramenta se esta for simples, mas mais eficaz que os tradicionais métodos de dividir despesas. Por isso tivemos em conta a funcionalidade, a conveniência e a simplicidade da aplicação a desenvolver. Como anteriormente referido, definimos que o público-alvo da aplicação “Bill Splitter” serão estudantes universitários.

Destacamos em seguida algumas das funcionalidades mais importantes definidas:

- a app permite o registo de utilizadores(morador);
- o morador tem associado uma data de entrada na casa e uma conta corrente;
- a app permite o lançamento de despesas (recorrentes ou não);
- a despesa pode ser dividida por vários moradores ou não, com iguais ou diferentes pesos;
- o pagamento das despesas é lançado, sendo associado quem a liquidou;
- quando uma despesa é paga, as contas correntes dos moradores são atualizadas;

1.3. Análise de Requisitos

Foram definidos então os seguintes requisitos:

- Criação de casas
- Criação de contas de utilizador/morador
- Suportar o registo das despesas de moradores registados
- Cada morador tem uma conta corrente
- Cada morador capaz de gerir as despesas não pagas (alterar, apagar, pagar)
- A distribuição das despesas pelos moradores deverá ser feita tendo em consideração quem, em cada momento, estava a ocupar o apartamento.
- Diferenciação entre despesas recorrentes ou extraordinárias.
- Diferentes formas de divisão da despesa (montantes exatos, percentagem ou equitativa)
- Capacidade de ajustar as contas sem necessidade de pagamentos
- Capacidade de inserir ou eliminar moradores
- Capacidade de os utilizadores comentarem as despesas

1.3.1 Breve descrição do programa

Para melhor descrever o funcionamento do programa iremos considerar que três universitários (moradores A, B e C) vão morar juntos para uma habitação temporária e que irão utilizar a aplicação Bill Spliter para gestão das suas despesas. Nenhum dos moradores (A, B ou C) pode estar já associado a uma casa na aplicação (não pode estar registado no programa).

Assim o morador A entra na aplicação e seleciona “Registar casa”. Neste menu irá preencher os seus dados de utilizador e os dados da casa. Depois do registo, a casa (X) é criada com um morador (A). Nesta altura o morador A já tem acesso às funcionalidades da aplicação depois de fazer login no menu de abertura. Em seguida, o morador A irá adicionar um outro morador à casa através do botão “adicionar morador”. Assim, o morador A adiciona o morador B, inserindo o seu nome, email e password. Depois de adicionado, o morador B fica com acesso à aplicação e, antes de tudo, deverá fazer login e selecionar o menu “Editar conta” de modo a alterar a password que o morador A lhe atribuiu. Em seguida, o morador C deverá ser adicionado à casa X pelo morador A ou B proceder da forma já descrita para alterar os dados de acesso.

Os moradores apercebem-se que na casa não existem copos e decidem que é necessária adquirir 12 copos. Decidem que é o morador B que irá fazer a compra. Depois de fazer a compras dos copos, que custaram 12 euros, o morador B lança a despesa no Bill Splitter. Assim, autentica-se no programa e seleciona a opção “Adicionar despesa” preenche os campos e seleciona a opção “Equi” de maneira a que a despesa seja igualmente dividida pelos 3 moradores. Depois de lançar a despesa o morador B seleciona a despesa no menu “Despesas por liquidar” e “Paga” a despesa. Nesta altura irá aparecer na tabela de compromissos o valor 4 no campo do morador A e C, e no saldo 8. Assim, fica a saber que os moradores A e C lhe devem 4 euros cada um e que tem 8 euros a haver. Nos moradores A e C irá constar um saldo de “-4” e no campo do utilizador B irá constar o mesmo valor, pois é com ele a dívida.

O morador A decide então saldar a dívida com o morador B, pagando-lhe 4 euros. Em seguida acede ao Bill Spliter e seleciona o campo do utilizador B e confirma que efetuou o pagamento. Nesta altura o morador A fica com os valores a zero e quando o utilizador B se aceder ao programa, irá ver um saldo de 4 e no campo do morador A um zero.

1.4. Modelos

1.4.1 Modelo de Domínio

Abaixo apresentamos o Modelo de Domínio da aplicação elaborado. Neste diagrama são claramente identificadas as entidades necessárias.

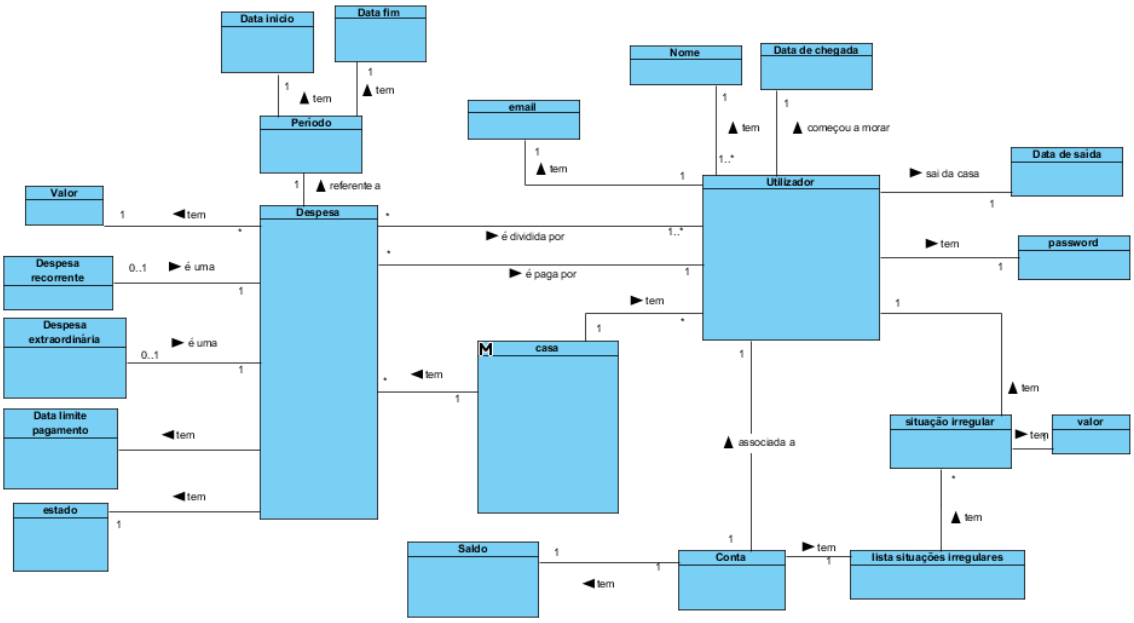

Figura – Modelo de domínio

Neste modelo existem três entidades centrais:

- Casa - é identificado por um Id, e tem associado uma morada, localidade, código postal, telefone, lista de despesas, uma lista de moradores e campo para observações.
- Utilizador - é identificado por um nome, um email, uma password, uma data de entrada e uma data de saída. As datas servem para fazer um melhor controlo sobre as despesas que aquele utilizador tem que pagar, por exemplo se um utilizador não entrar no inicio do mês é justo que não tenha que pagar a conta, por exemplo, da água total. O sistema pode assim, baseando-se na data de entrada e saída dos moradores calcular de forma mais justa as percentagens a pagar por cada utilizador.
- Despesa - pode ser recorrente ou extraordinária e tem um estado que indica se já foi liquidada (uma despesa pode ser paga por um morador antes dos outros lhe pagarem). Uma despesa na aplicação “Bill Splitter” é criada por um morador para repartir pelos outros

moradores, ou seja, há um que paga verdadeiramente a despesa e os restantes pagam a esse morador. Quando um morador cria uma despesa é criada uma situação irregular nas contas dos restantes que depois tem que ser saldada por estes, seja pagando (transferência de uma quantia) seja com outras despesas (a aplicação calcula automaticamente o saldo de cada utilizador baseando-se nos valores a receber e a pagar a outros). Para além disto a despesa tem um período associado que, como já foi referido, permite saber quais os moradores que estavam naquele momento a habitar.

1.4.2 Diagrama de Classes

Abaixo apresentamos o Modelo de Classes da aplicação elaborado. Neste diagrama são claramente identificadas as classes usadas.

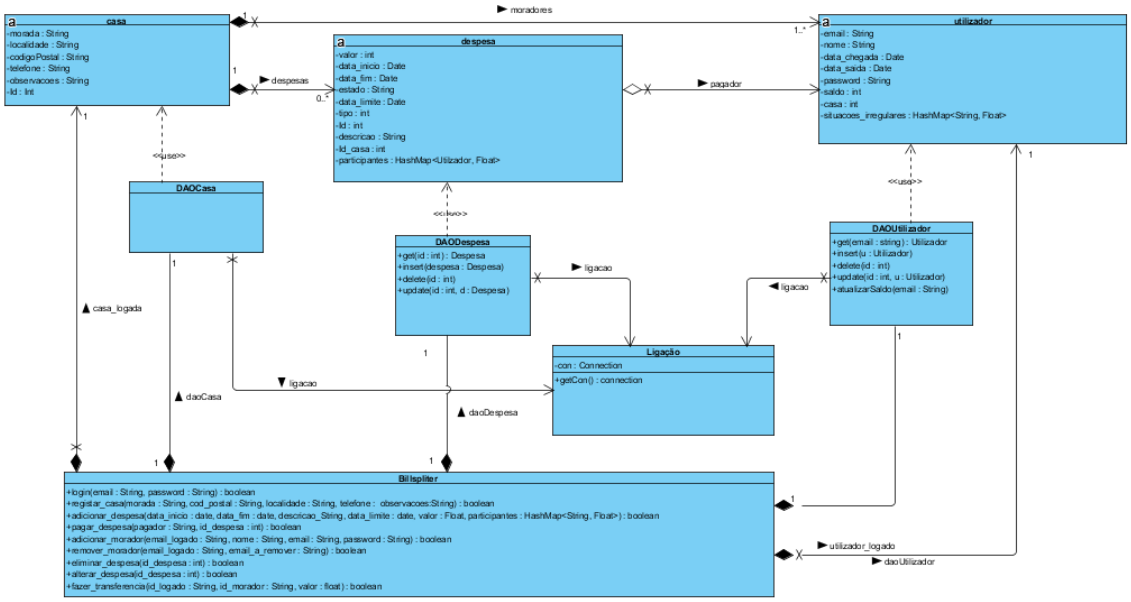

Figura – Diagrama de classes

1.4.3 Diagrama de Use Case

Apresentamos em seguida o Diagrama Use Case da aplicação no qual podem ser identificadas as funcionalidades da aplicação.

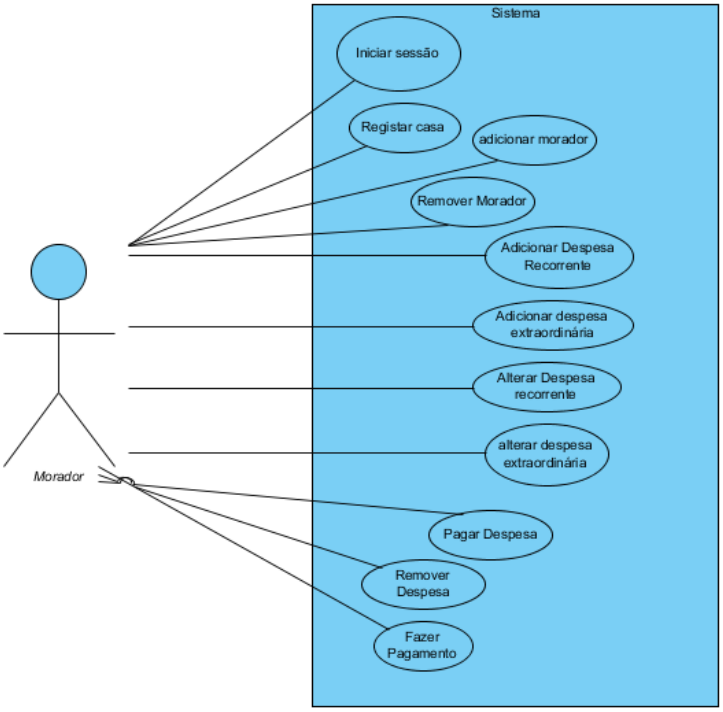

Figura 2 – Diagrama de use case

O diagrama dos Use Cases conta apenas com um ator que é o Morador. Optamos por não incluir nenhum tipo de ator com mais privilégios que o ator “comum” já que tendo em conta o público-alvo e a situação onde vai ser usado, na nossa opinião, não se adequa ter um utilizador com papel de administrador. Na aplicação “Bill Splitter” todos os utilizadores podem adicionar ou remover um morador, sendo que os restantes são avisados quando uma dessas situações acontece. Temos por isso apenas um ator, o “Morador”. Este ator pode adicionar, remover ou alterar despesas, pode pagar despesas, consultar o saldo e fazer a gestão dos utilizadores da aplicação.

1.4.4 Especificação de Use Cases

Em seguida apresentam-se as descrições dos dois Use Cases mais centrais:

- “Adicionar despesa”

Super Use Case	
Author	
Date	
Brief Description	Morador adiciona uma despesa extraordinaria ao sistema
Preconditions	Morador está autenticado no sistema
Post-conditions	É lançada uma despesa extraordinaria ao sistema

Flow of Events	1		Apresenta o formulario
	2	fornece os dados do formulario	
	3		Valida os dados
	4		adiciona despesa ao sistema
	5		informa utilizador que a despesa foi adicionada com sucesso

Exceção 1 [valores % não válidos] (passo3)		**Actor Input**	**System Response**
	1		indica que soma das % diferente de 100

Exceção 1 [valores € não válidos] (passo3)		**Actor Input**	**System Response**
	1		indica que soma dos parciais diferente do valor da despesa

Exceção 1 [datas não válidas] (passo3)		**Actor Input**	**System Response**
	1		indica que data fim < data início

Figura – Use case – adicionar despesa

- “Pagar despesa”.

Super Use Case	
Author	
Date	
Brief Description	morador salda uma despesa
Preconditions	Utilizador está autenticado
Post-conditions	A despesa passa para paga e as contas correntes dos restantes utilizadores sao alteradas

Flow of Events

	Actor Input	System Response
1	Seleciona a despesa que pretende pagar	
2		Avisa que depois de pagar a despesa esta nao pode ser alterada nem eliminada. Pergunta se o utilizador quer mesmo pagar a despesa.
3	Utilizador confirma	
4		Retira ao saldo de cada utilizador envolvido na despesa o montante associado ao utilizador em questao. Adiciona ao saldo do utilizador que pagou a soma do que os outros lhe devem.
5		Despesa é marcada como PAGA
6		informa que o pagamento foi efetuado com sucesso

Excecao 1 [Utilizador cancela pagamento] (Passo 3)

	Actor Input	System Response
1	Utilizador cancela	
2		Volta ao menu inicial

Figura – Use case – pagar despesa

1.4.5 Máquinas de estado

O seguinte esquema é útil na análise do estado dos diversos menus a aplicação bem como as funcionalidades que permite:

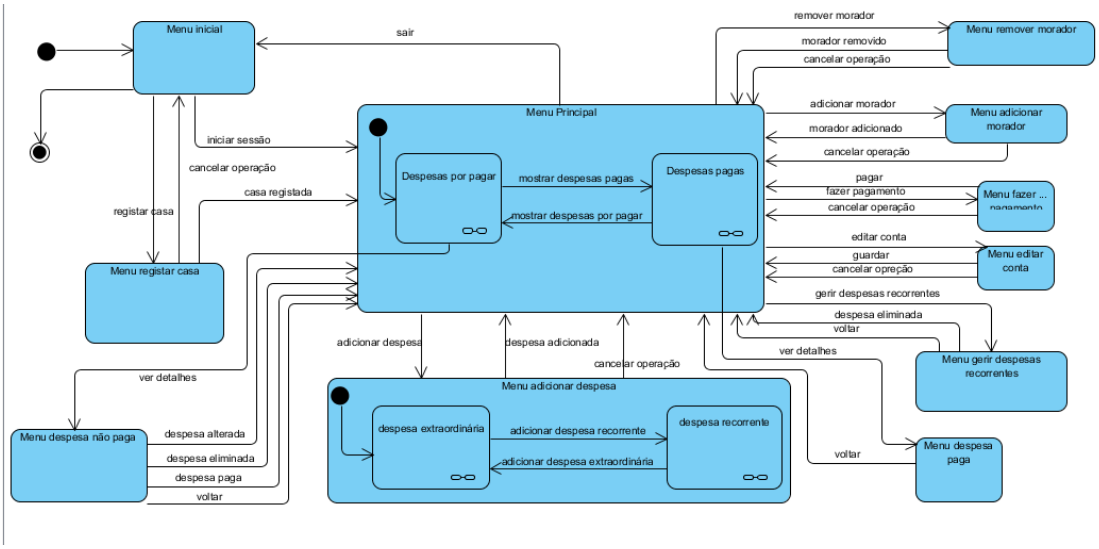

Figura – Máquina de estado - Interface

O seguinte esquema mostra a os estados pelos quais passa uma despesa extraordinária. Vemos que, depois de paga, apenas é possível eliminá-la se se eliminar a casa à qual está associada.

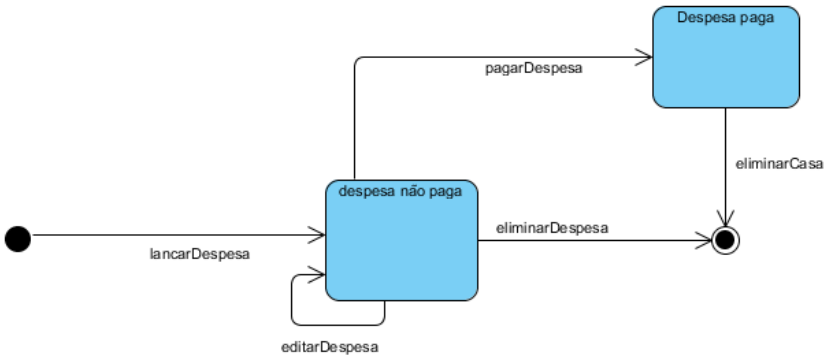

Figura – Máquina de estado – Despesa extraordinária

1.4.6 Diagrama de sequência

Em seguida apresentam-se dois diagramas de sequência de operações centrais da aplicação desenvolvida – adicionar despesa extraordinária e efetuar um pagamento de um valor entre dois moradores.

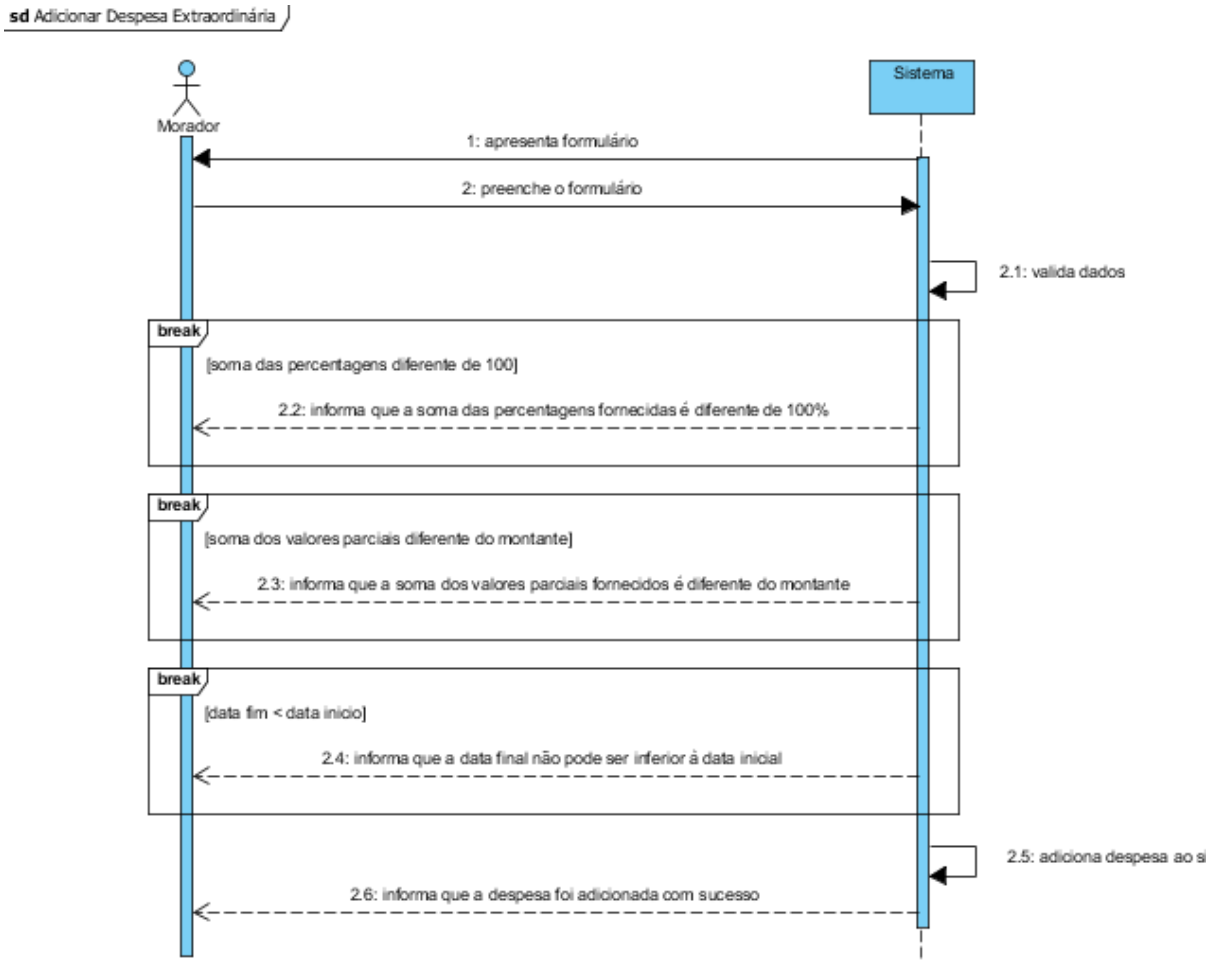

Figura – Diagrama de sequência – adicionar despesa extraordinária

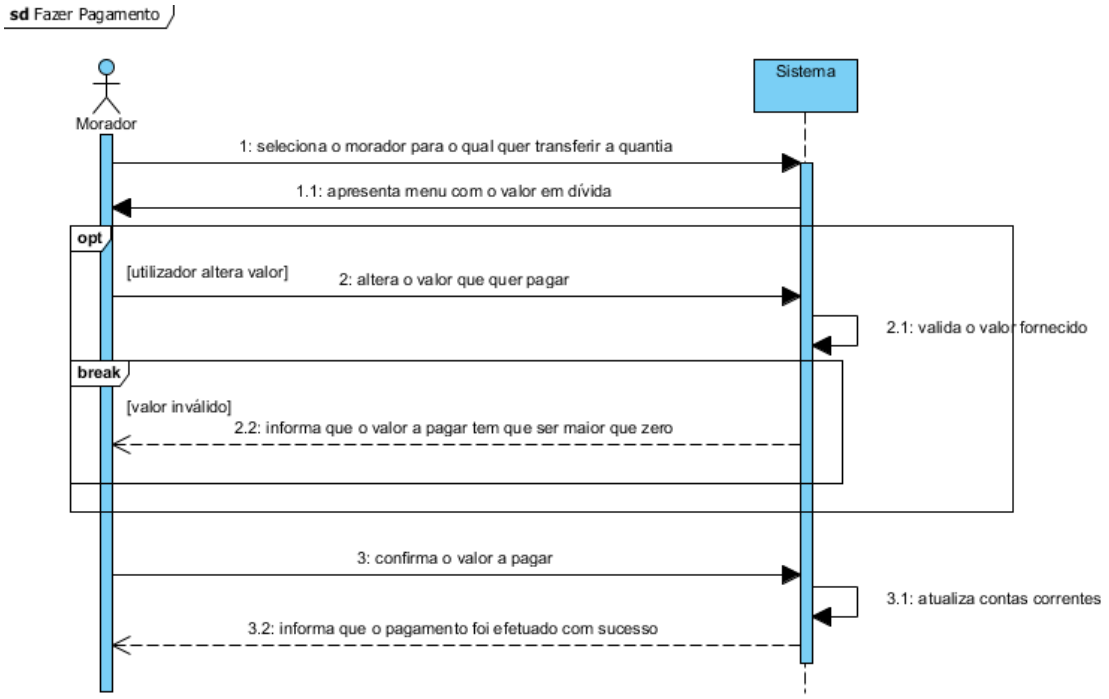

Figura – Diagrama de sequência – fazer pagamento

1.4.7 Diagramas de package

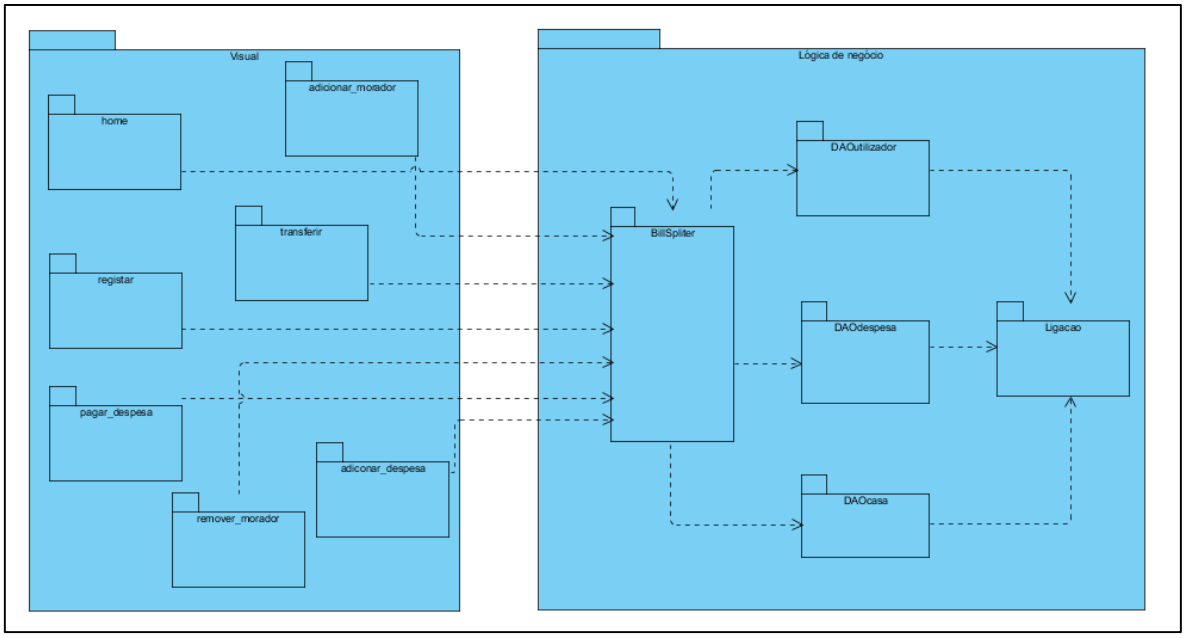

Figura – Diagrama de package

1.4.8 Diagrama de objetos

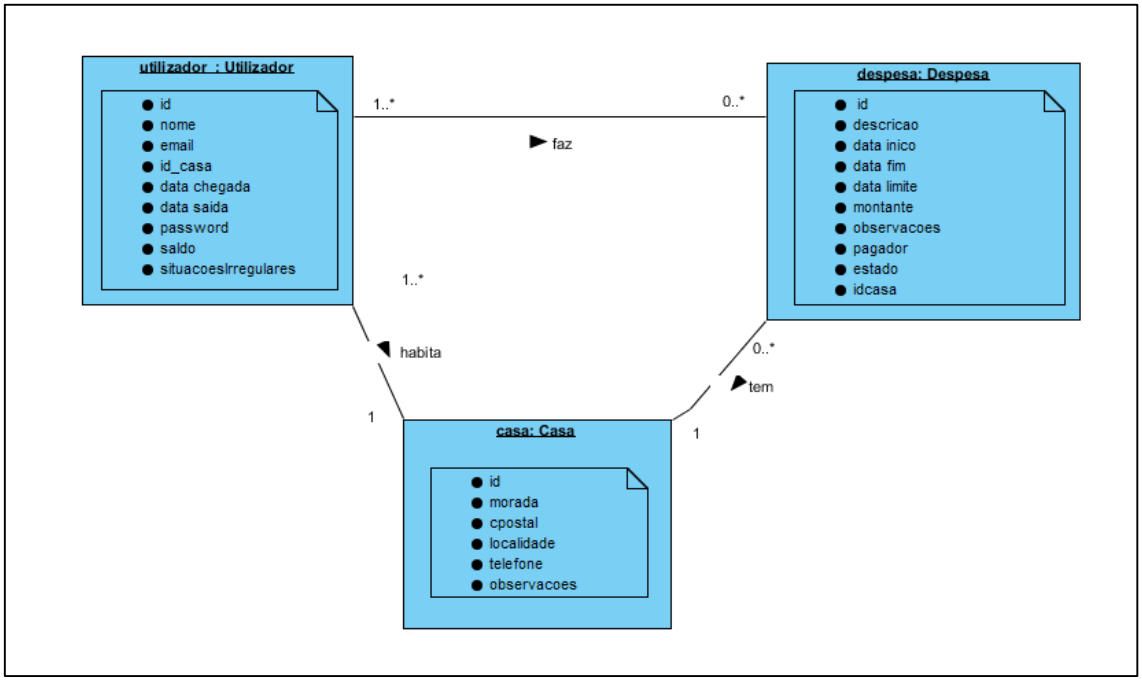

Figura – Diagrama de objetos

1.4.9 Diagrama de componentes

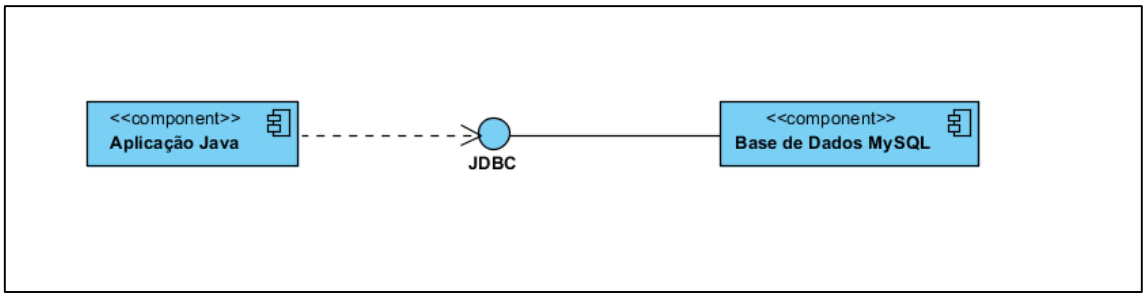

Figura – Diagrama de componentes

1.4.10 Mockups

Antes do desenvolvimento do layout da aplicação foram construídos mockups em PowerPoint. Em seguida mostramos alguns dos Menus projetados.

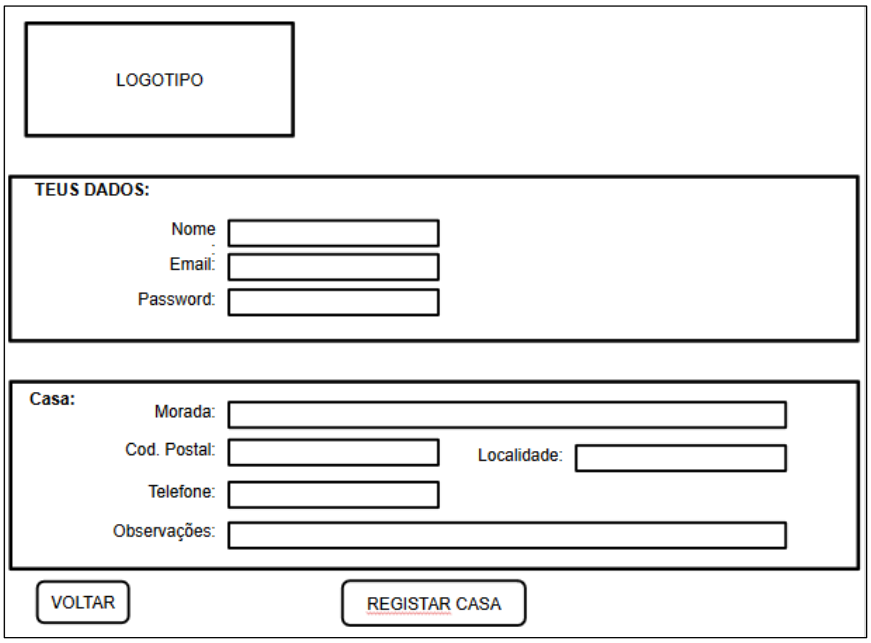

Figura – Mockup - Menu "Registar Casa"

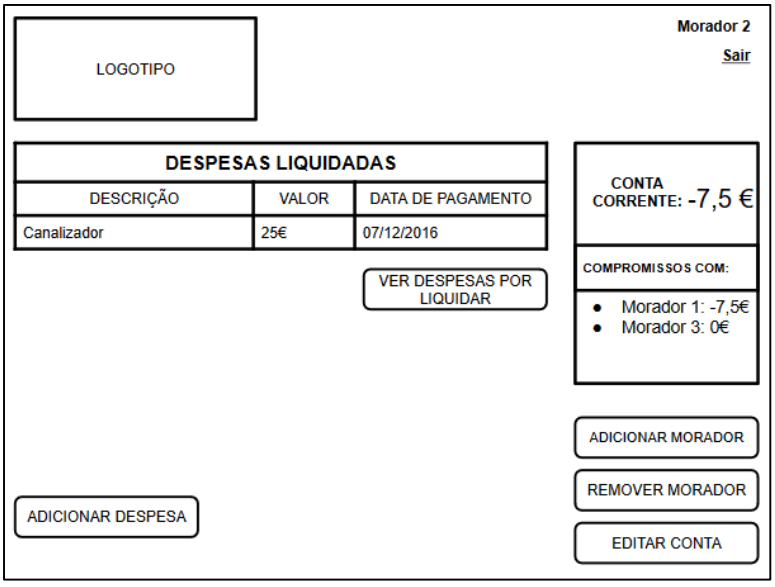

Figura – Mockup - Menu "Despesas Liquidadas"

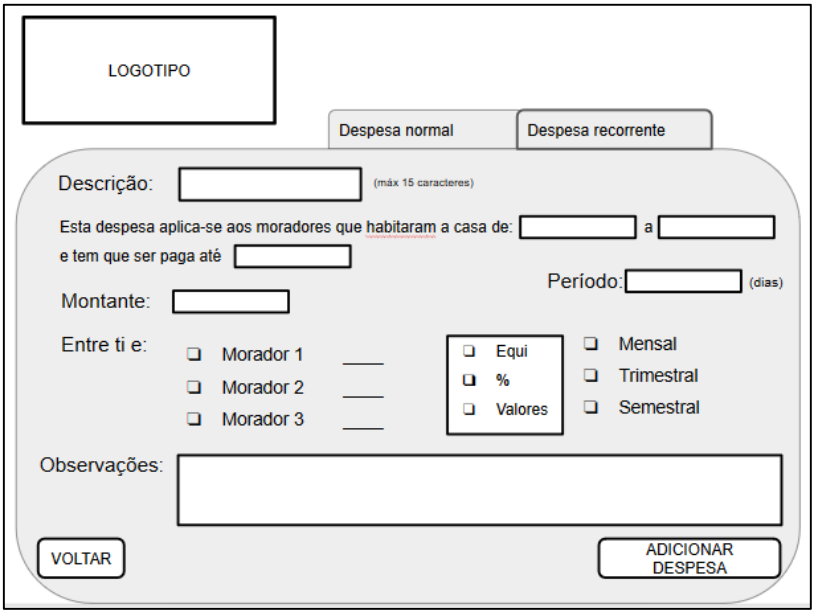

Figura – Mockup - Menu "Adicionar despesa recorrente"

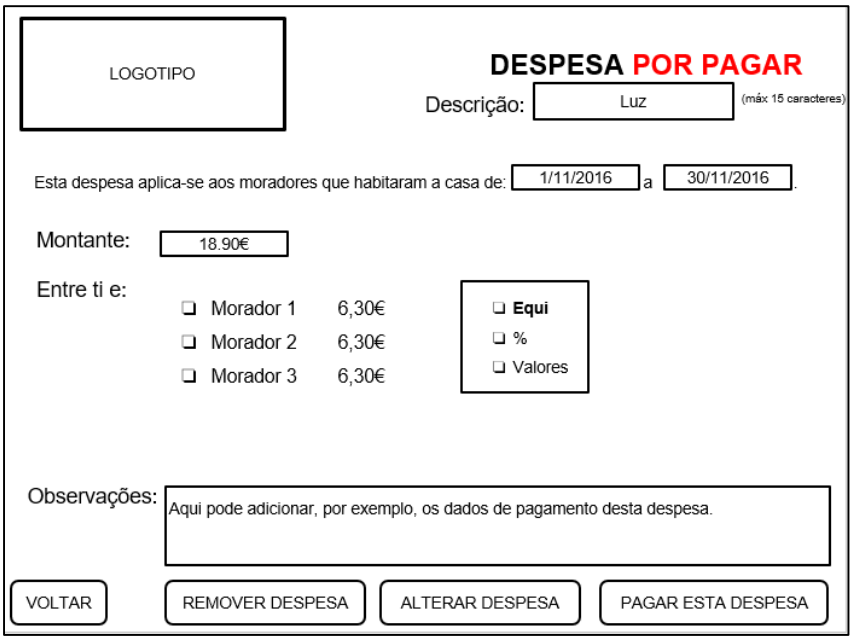

Figura – Mockup - Menu "Detalhes de Despesa não paga"

1.5. Implementação da aplicação

A aplicação desenvolvida não tem implementada todas as funcionalidades planeadas, no entanto, estão funcionais os procedimentos mais relevantes, a saber:

- Criação de contas de utilizador/morador
- Criação de casas
- Suportar o registo das despesas de moradores registados
- Cada morador tem uma conta corrente
- Cada morador capaz de gerir as despesas não pagas (pagar)
- Diferenciação entre despesas recorrentes ou extraordinárias
- Aplicada uma formas de divisão da despesa (equitativa)
- Capacidade de ajustar as contas sem necessidade de pagamentos
- Capacidade de os utilizadores comentarem as despesas

Ficaram ainda por implementar funções de validação de dados que resultariam num programa robusto e fiável. Também a estruturação do programa não foi a pretendida.

1.6. Conclusão

De uma maneira geral podemos dizer que neste trabalho estão relatados claramente os requisitos, funcionalidades e comportamento da aplicação “Bill Splitter”.

Como conclusão salientamos os seguintes aspetos:

- após discussão entre os elementos do grupo, foram definidos os requisitos para que a aplicação a desenvolver simples e de fácil interpretação;

- foram realizados vários diagramas necessários para uma adequada implementação da aplicação;

- foram realizados mockups iniciais da aplicação;

- a aplicação desenvolvida reflete os mockups criados;

- fez-se um esforço de modo a privilegiar a simplicidade e facilidade de uso da aplicação;

- consideramos que a organização e estrutura pensada para a aplicação são adequadas;

- foi implementada a maioria das funcionalidades pensadas;

- consideramos que o código gerado não traduz fielmente a representação dos diagramas.

Pelo referido, no geral, consideramos que os objetivos do trabalho foram cumpridos.

2. Anexos

Super Use Case	
Author	
Date	
Brief Description	Morador autentica-se no sistema para ter acesso às despesas
Preconditions	O utilizador está registado e não está autenticado
Post-conditions	O utilizador fica autenticado no sistema

Flow of Events

	Actor Input	System Response
1		apresenta formulário de login
2	introduz email e password	
3		Valida os dados de autenticação
4		processa o inicio de sessão
5		notifica que o início de sessão foi efetuado com sucesso

Exceção 1 [cancelar operação] (passo 2)

	Actor Input	System Response
1	indica que quer cancelar o início de sessão	
2		termina o programa

Exceção 2 [dados inválidos] (passo 4)

	Actor Input	System Response
1		indica a password e/ou o email estão errados

Figura – Diagrama Use Case – Iniciar sessão

Super Use Case	
Author	
Date	
Brief Description	Utilizador cria no sistema uma conta e uma casa
Preconditions	email fornecido pelo utilizador é valido
Post-conditions	Fica registada uma nova casa no sistema e adicionado um utilizador ao sistema

Flow of Events

	Actor Input	System Response
1		apresenta formulário
2	fornece informações solicitadas	
3		valida dados
4		adiciona casa e morador ao sistema
5		apresenta menu principal

Exceção 1 [email já registado] (passo 3)

	Actor Input	System Response
1		3.1 indica que já existe uma conta de morador registada com o email fornecido

Exceção 2 [campos inválidos] (passo3)

	Actor Input	System Response
1		3.1 indica que existe erros tipográficos/campo inválido/campo não preenchido (p.ex. código postal não existente)

Figura – Diagrama Use Case – Registar Casa

Super Use Case	
Author	
Date	
Brief Description	Morador (morador 1) cria conta no sistema para outro morador (morador 2) que lhe vai permitir aceder às despesas e estar incluído na divisão das mesmas
Preconditions	Morador 1 está registado e com sessão iniciada; email fornecido é válido
Post-conditions	é adicionado um morador novo (morador 2) à casa

Flow of Events

	Actor Input	System Response
1		Apresenta formulario
2	fornece nome, email e password.	
3		valida dados
4		adiciona morador 2 ao sistema, associado à mesma casa do morador 1
5		apresenta menu principal

Exceção 1 [Email já registado] (passo 3)

	Actor Input	System Response
1		3.1 Notifica o morador 1 que já existe uma conta com o email fornecido

Exceção 1 [utilizador cancela registo de morador] (passo 2)

	Actor Input	System Response
1	indica que pretende cancelar	
2		apresenta menu principal

Figura – Diagrama Use Case – Adicionar morador

Super Use Case	
Author	
Date	
Brief Description	Remover um morador da casa
Preconditions	Utilizador (morador 1) está autenticado e é diferente do morador (morador 2) a remover
Post-conditions	O morador 2 é removido da casa

Flow of Events

	Actor Input	System Response
1	indica que quer eliminar conta do morador 2	
2		verifica se o morador 2 tem situações irregulares a zero
3		apaga a conta do sistema

Exceção 1 [situações irregulares não nulas] (passo 2)

	Actor Input	System Response
1		informa o utilizador da impossibilidade de apagar a conta do morador 2

Figura – Diagrama Use Case – Remover morador

Super Use Case	
Author	
Date	
Brief Description	Morador adiciona uma despesa extraordinaria ao sistema
Preconditions	Morador está autenticado no sistema
Post-conditions	É lançada uma despesa extraordinaria ao sistema

Flow of Events

	Actor Input	System Response
1		Apresenta o formulario
2	fornece os dados do formulario	
3		Valida os dados
4		adiciona despesa ao sistema
5		informa utilizador que a despesa foi adicionada com sucesso

Exceção 1 [valores % não válidos] (passo3)

	Actor Input	System Response
1		indica que soma das % diferente de 100

Exceção 1 [valores € não válidos] (passo3)

	Actor Input	System Response
1		indica que soma dos parciais diferente do valor da despesa

Exceção 1 [datas não válidas] (passo3)

	Actor Input	System Response
1		indica que data fim < data início

Figura – Diagrama Use Case – Adicionar despesa extraordinária ao sistema

Super Use Case	
Author	
Date	
Brief Description	O utilizador remove uma despesa do sistema.
Preconditions	Utilizador está registado e autenticado. Despesa ainda nao foi paga
Post-conditions	Despesa e removida do sistema

Flow of Events

	Actor Input	System Response
1		Apresenta a informação da despesa
2	Carrega no botao para remover despesa	
3		remove a despesa do sistema

Figura – Diagrama Use Case – Remover despesa

Super Use Case	
Author	
Date	
Brief Description	Morador altera uma despesa recorrente.
Preconditions	Utilizador está autenticado
Post-conditions	A despesa fica alterada com os novos dados.

Flow of Events

	Actor Input	System Response
1		Apresenta o formulario
2	Altera os dados que pretende alterar	
3		Valida os dados
4		Os dados sao validos, a despesa e alterada no sistema. Indica ao utilizador que as alteracoes foram efetuadas com sucesso.
5		Atualiza o sistema. Caso o montante ou a divisao das despesas tenha sido alterado entao sao calculadas as novas contas correntes dos morados envolvidos na despesa.

Exceção 1 [valores % nao validos] (passo 3)

	Actor Input	System Response
1		Indica que a soma das % e diferente de 100

Exceção 2 [valores € nao validos] (Passo 3)

	Actor Input	System Response
1		Indica que a soma dos parciais e diferente do valor total da despesa.

Exceção 3 [datas nao validas] (Passo 3)

	Actor Input	System Response
1		Indica que a data fim < data inicio

Figura – Diagrama Use Case – Alterar despesa recorrente

Date	
Brief Description	Transferencia de dinheiro entre utilizadores
Preconditions	Moradores registados na mesma casa e utilizador autenticado
Post-conditions	é feita uma troca de saldo de um morador para outro

Flow of Events

	Actor Input	System Response
1	Seleciona o utilizador para o qual pretende transferir a quantia	
2		Sistema apresenta menu com o valor em divida
3	Utilizador confirma ou altera o valor a pagar.	
4		Sistema verifica.
5		Contas correntes sao atualizadas

Exceção 1 [Montante nao valido] (Passo 4)

	Actor Input	System Response
1		Sistema informa que o montante inserido nao e valido.

Figura – Diagrama Use Case – Fazer pagamento

Super Use Case	
Author	
Date	
Brief Description	Utilizador pretende alterar uma despesa ja inserida no sistema.
Preconditions	Utilizador autenticado e despesa nao paga.
Post-conditions	A despesa fica com os novos dados

Flow of Events	**Actor Input**	**System Response**
1		Apresenta formulario
2	Altera os campos que pretender	
2	Altera os campos que pretender	
3		Valida os dados
4		Altera no sistema os dados correspondentes aquela despesa
5		informa que a despesa foi alterada com sucesso

Exceção 1 [valores % não válidos] (passo3)	**Actor Input**	**System Response**
1		indica que soma das % diferente de 100

Exceção 1 [valores € não válidos] (passo3)	**Actor Input**	**System Response**
1		indica que soma dos parciais diferente do valor da despesa

Exceção 1 [datas não válidas] (passo3)	**Actor Input**	**System Response**
1		indica que data fim data início

Figura – Diagrama Use Case – Alterar despesa extraordinária

Super Use Case	
Author	
Date	
Brief Description	morador salda uma despesa
Preconditions	Utilizador está autenticado
Post-conditions	A despesa passa para paga e as contas correntes dos restantes utilizadores sao alteradas

Flow of Events	1	Seleciona a despesa que pretende pagar	
	2		Avisa que depois de pagar a despesa esta nao pode ser alterada nem eliminada. Pergunta se o utilizador quer mesmo pagar a despesa.
	3	Utilizador confirma	
	4		Retira ao saldo de cada utilizador envolvido na despesa o montante associado ao utilizador em questao. Adiciona ao saldo do utilizador que pagou a soma do que os outros lhe devem.
	5		Despesa é marcada como PAGA
	6		informa que o pagamento foi efetuado com sucesso

Excecao 1 [Utilizador cancela pagamento] (Passo 3)		**Actor Input**	**System Response**
	1	Utilizador cancela	
	2		Volta ao menu inicial

Figura – Diagrama Use Case – Pagar despesa

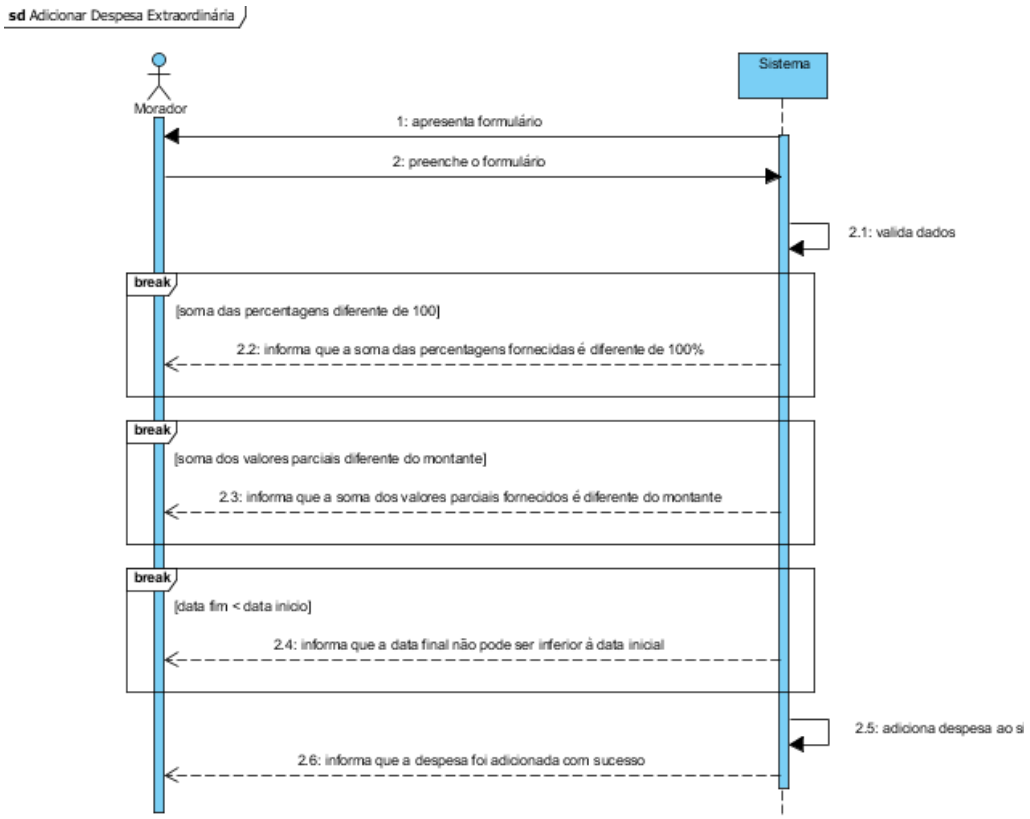

Figura – Diagrama de sequência – adicionar despesa extraordinária

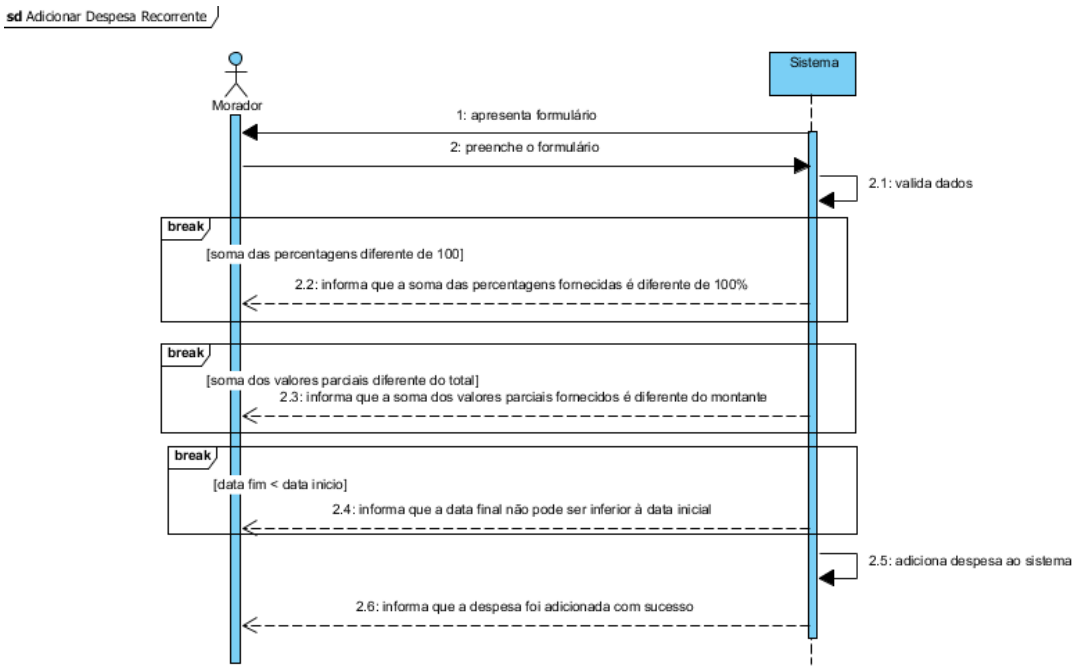

Figura – Diagrama de sequência – adicionar despesa recorrente

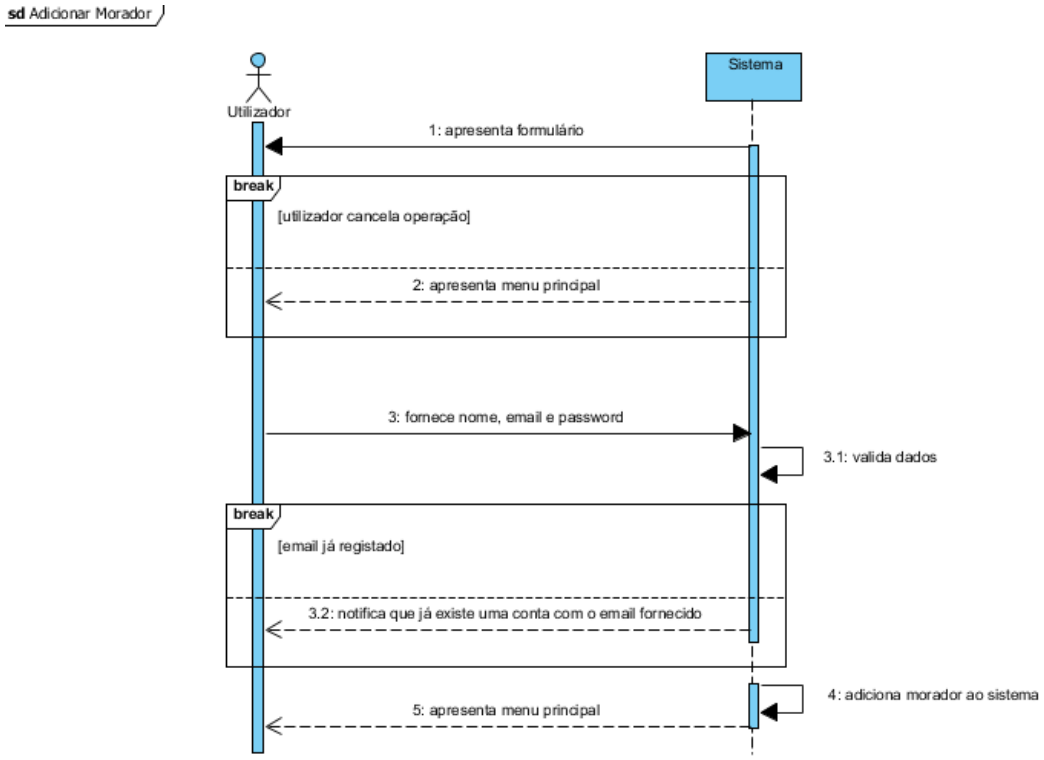

Figura – Diagrama de sequência – adicionar morador

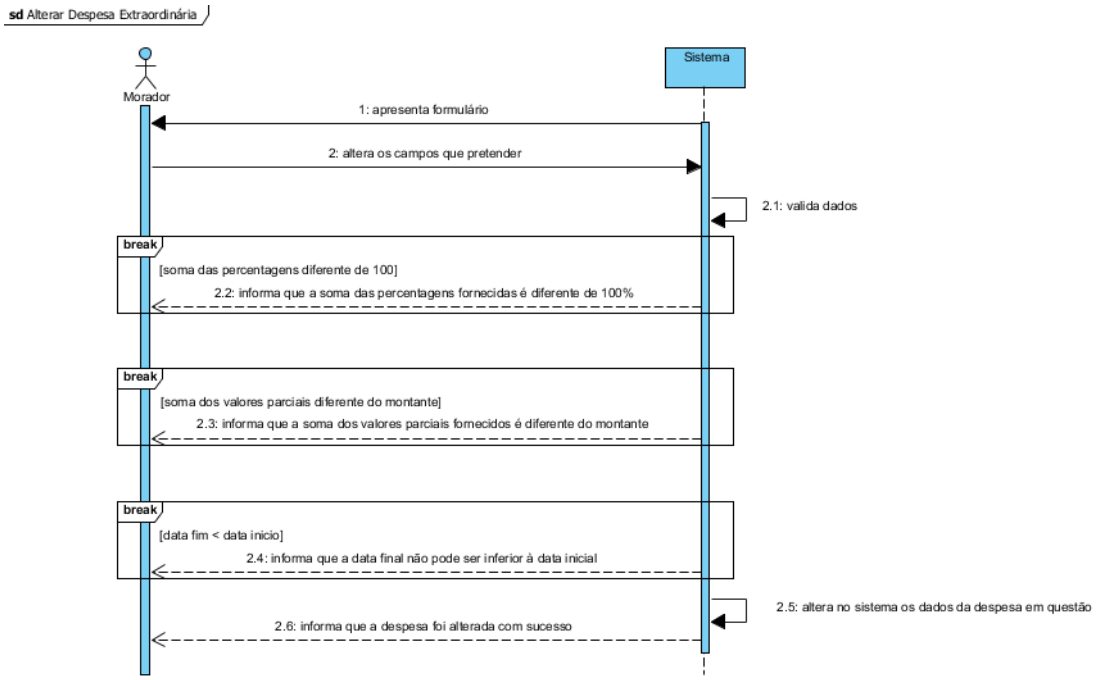

Figura – Diagrama de sequência – alterar despesa extraordinária

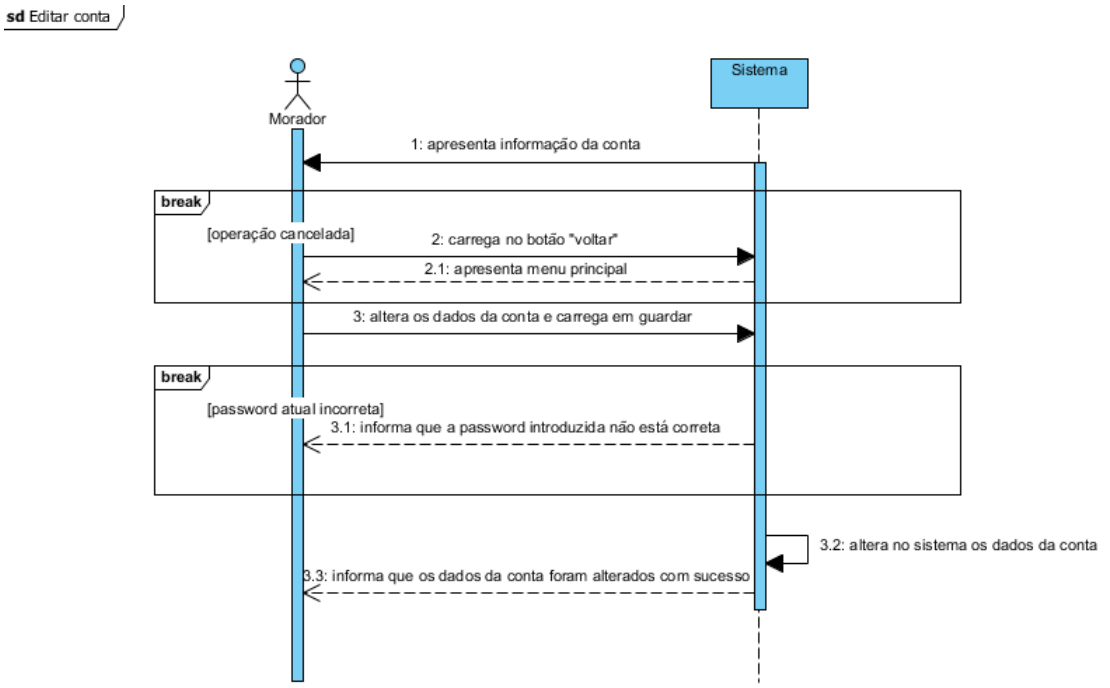

Figura – Diagrama de sequência – editar conta

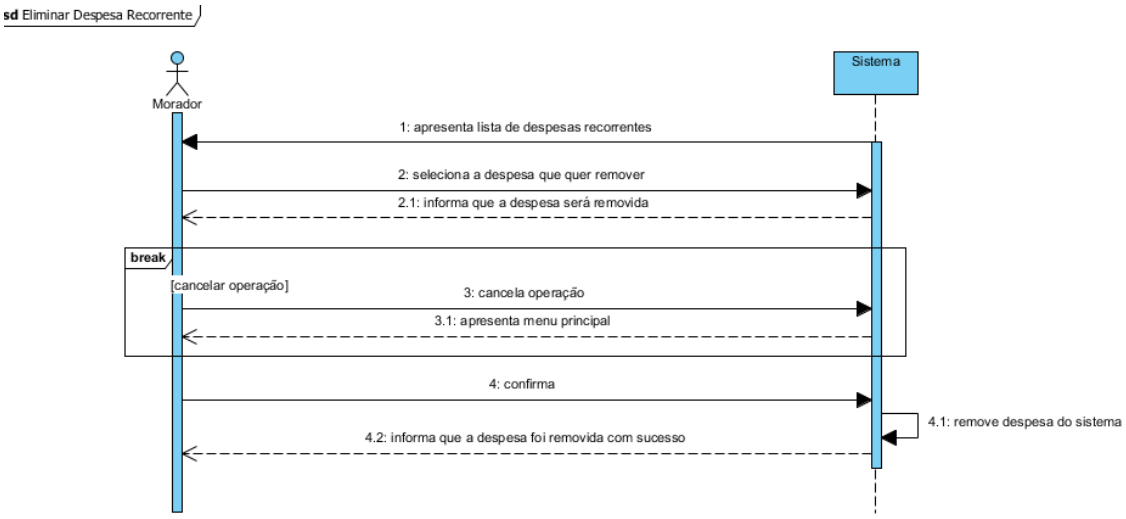

Figura – Diagrama de sequência – eliminar despesa recorrente

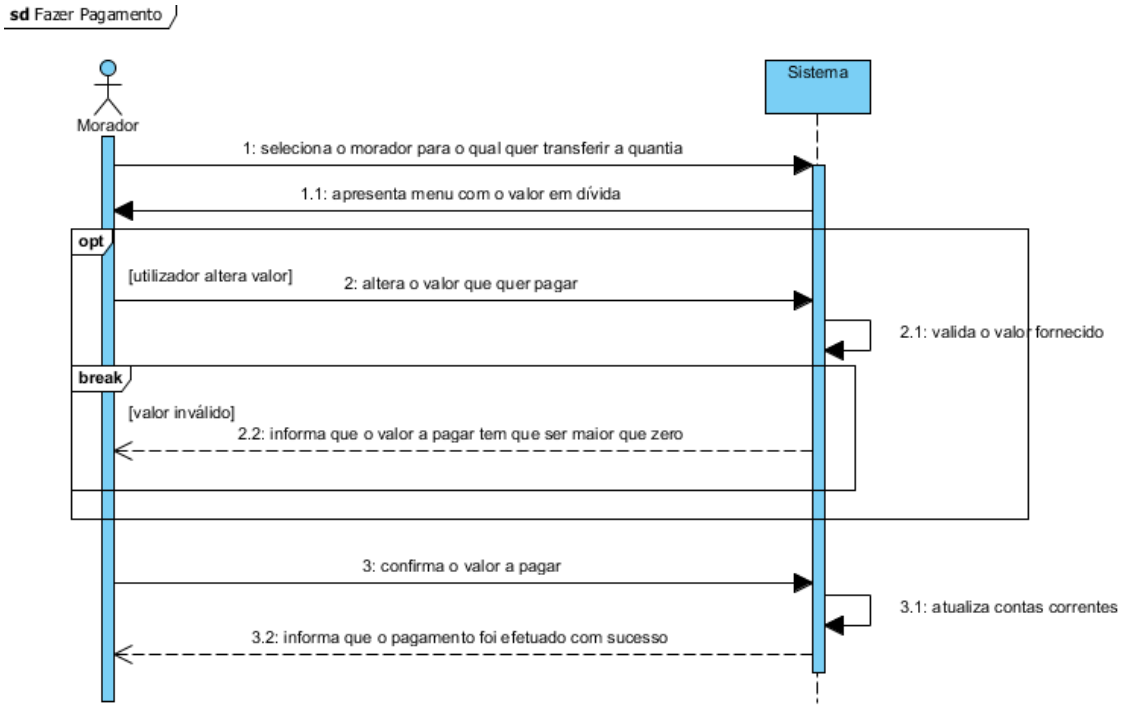

Figura – Diagrama de sequência – fazer pagamento

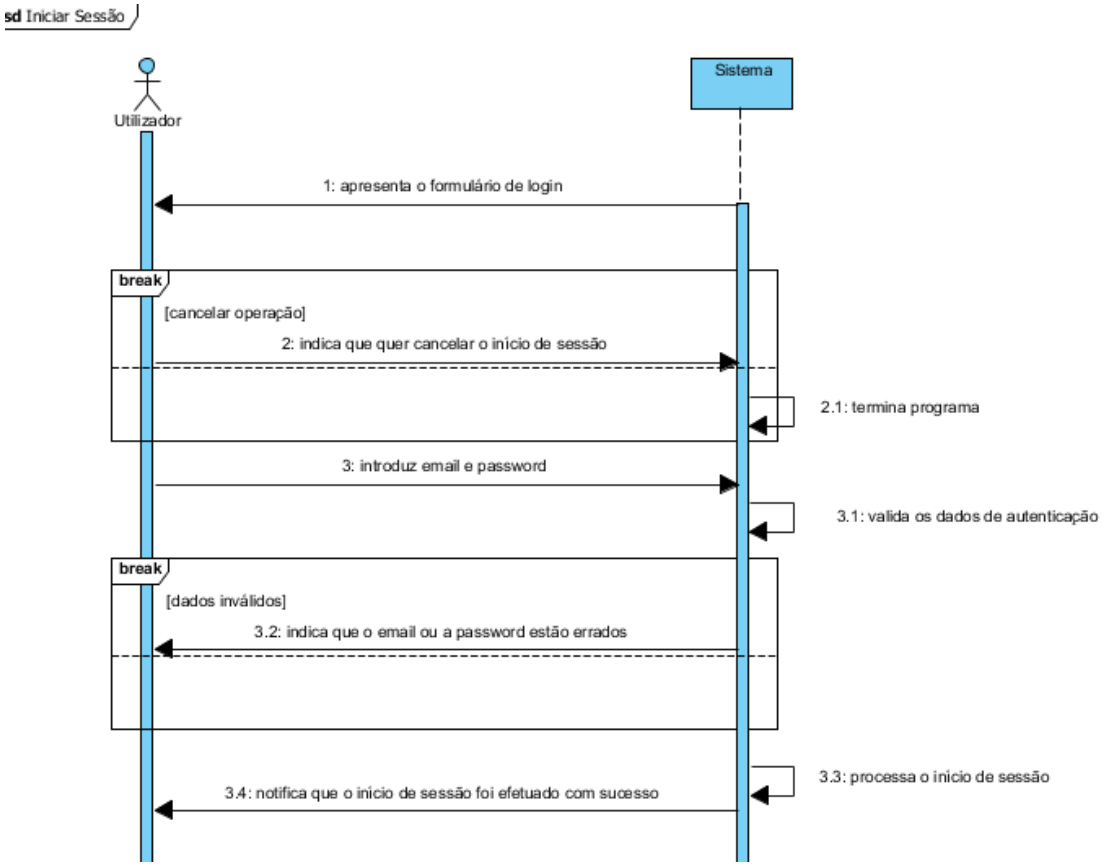

Figura – Diagrama de sequência – iniciar sessão

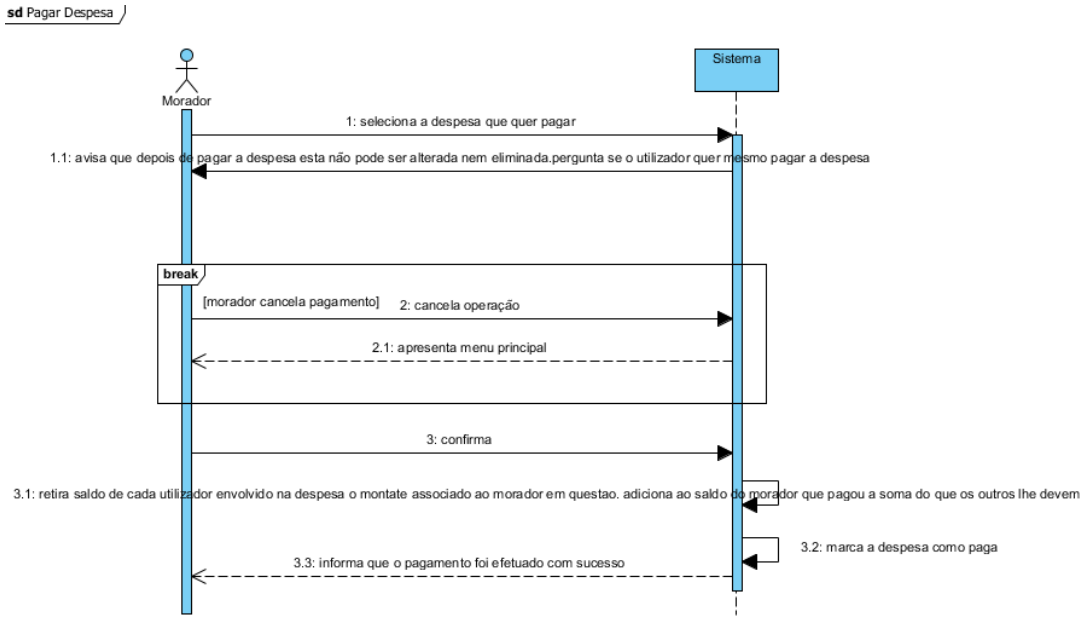

Figura – Diagrama de sequência – pagar despesa

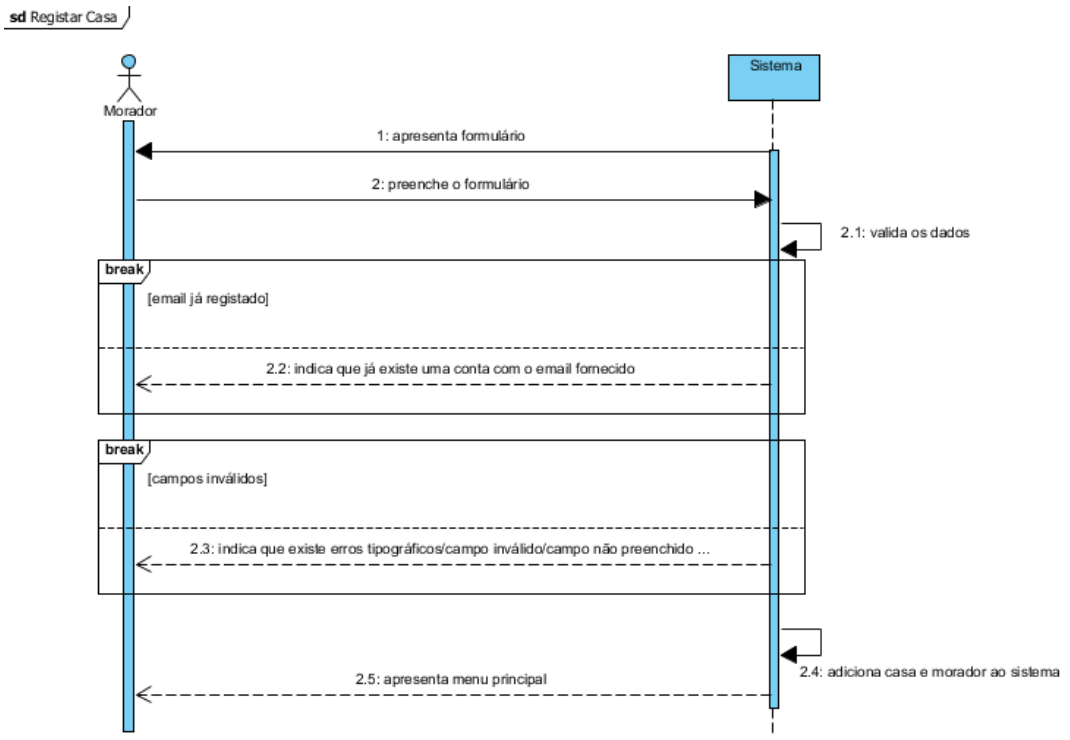

Figura – Diagrama de sequência – Registar casa

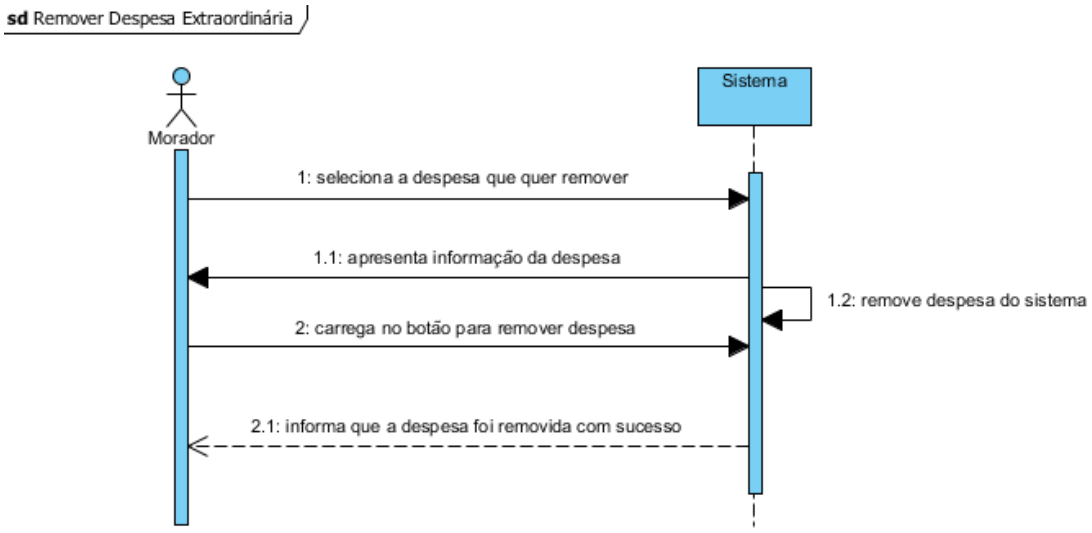

Figura – Diagrama de sequência – Remover despesa extraordinária

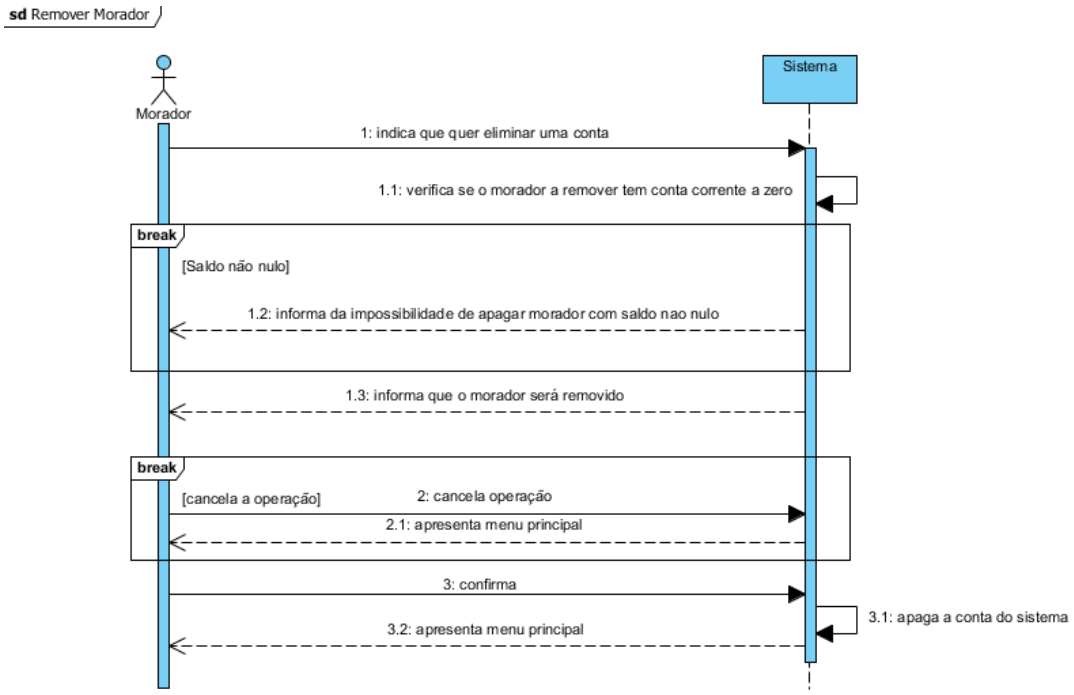

Figura – Diagrama de sequência – Remover morador