



Universidade do Minho
Departamento de Informática

Unidade Curricular de Desenvolvimento de Sistemas Software

Trabalho Prático

Aplicação de Gestão de Despesas

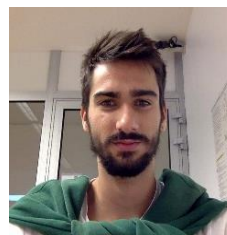
Ano Letivo 2016/2017



Marcos Luís
A70676



Mário Ferreira
A70441



Nelson Parente
A71625



Ruben Santos
A70644

Braga, Dezembro de 2016

Resumo

Este documento descreve todo o processo de estudo e elaboração de uma aplicação no âmbito da Unidade Curricular de Desenvolvimento de Sistemas Software para o ano letivo 2016/2017, tendo como tema incidente o registo e gestão de despesas de uma habitação dividida por vários inquilinos.

Índice

Introdução	5
Objetivos	5
Estrutura do Relatório	5
Funcionalidades	5
1. Modelo de Domínio	6
1.1 Morador.....	7
1.2 Pagamento.....	9
1.3 Despesa	11
2. Diagrama de Use Cases.....	16
3. Proposta inicial da Interface.....	26
3.1 Login.....	26
3.2 Menu Principal (Utilizador Normal)	27
3.4 Consultar Despesas	28
3.5 Consultar Despesa.....	29
3.7 Registrar Despesa.....	30
3.8 Detalhes de despesa paga.....	30
4. Diagramas de Sequência	31
5. Diagramas de Máquinas de Estado	38
6. Diagramas de Atividade.....	40
7. Diagramas de Classe	42
8. Diagramas de Package	44
9. Diagramas de Sequência com Subsistemas.....	45
10. Interface	50
13. Conclusões e Trabalho Futuro	58

Índice de Figuras

Figura 1. Morador - Versão 1.....	7
Figura 2. Morador – Versão 2.....	7
Figura 3. Morador - Versão 3.....	8
Figura 5. Pagamento – Versão 1	9
Figura 6. Pagamento – Versão 2	9
Figura 7. Pagamento – Versão 3	10
Figura 8. Despesa – Versão 1.....	11
Figura 9. Despesa – Versão 2.....	12
Figura 10. Despesa – Versão 3.....	13
Figura 11. Modelo de Domínio – 1ºFase	14
Figura 12. Modelo de Domínio – 2ªFase	15
Figura 13. Diagrama Use Cases.....	16
Figura 14. Use Case – Registrar Despesa.....	17
Figura 15. Use Case – Efetuar Pagamento.....	18
Figura 16. Use Case – Consultar Despesa	18
Figura 17. Use Case – Consultar Lista Despesas.....	19
Figura 18. Use Case – Editar Dados Morador	20
Figura 19. Use Case – Login.....	21
Figura 20. Use Case – Logout.....	22
Figura 21. Use Case – Registrar Morador	23
Figura 22. Use Case – Remover Morador.....	24
Figura 23. Use Case – Filtrar Despesas	25
Figura 24. Use Case – Sign Up	25
Figura 25. Use Case – Liquidar Divida.....	26
Figura 26. Ecrã de Login.....	27
Figura 27. Ecrã do Menu Principal (Utilizador Normal)	27
Figura 28. Ecrã do Menu Principal (Administrador)	28
Figura 29. Ecrã de Consultar Despesas.....	28
Figura 30. Ecrã dos detalhes de uma despesa	29
Figura 31. Ecrã de registo de pagamento	29
Figura 32. Ecrã de registo de despesa	30
Figura 33. Ecrã de detalhes de despesa paga	30
Figura 34. Diagrama de Sequência –Consultar Despesa.....	31
Figura 35. Diagrama de Sequência –Consultar Lista Despesa.....	31
Figura 36. Diagrama de Sequência –Editar Dados Morador	32

Figura 37. Diagrama de Sequência – Efetuar Pagamento.....	33
Figura 38. Diagrama de Sequência –Filtrar Despesa	34
Figura 39. Diagrama de Sequência –Liquidar Divida.....	34
Figura 40. Diagrama de Sequência – Login.....	35
Figura 41. Diagrama de Sequência – Logout.....	35
Figura 42. Diagrama de Sequência – Registrar Morador	36
Figura 43. Diagrama de Sequência – Remover Morador.....	36
Figura 44. Diagrama de Sequência –Sign Up	37
Figura 45. Máquina de Estado - Administrador.....	38
Figura 46. Máquina de Estado - Morador	39
Figura 47. Diagrama de Atividade – Apresentar Lista Despesas.....	40
Figura 48. Diagrama de Atividade – Efetuar Pagamento.....	40
Figura 49. Diagrama de Atividade – Registrar Despesa	41
Figura 50. Diagrama de Atividade – Remover Morador	41
Figura 51. Diagrama de classes.....	42
Figura 52. Diagrama de classes DAO.....	43
Figura 53. Diagrama de Package	44
Figura 54. Consultar Despesa	45
Figura 55. Consultar Lista Despesas.....	46
Figura 56. Editar Dados Morador	46
Figura 57. Login.....	47
Figura 58. Logout.....	48
Figura 59. Registrar Morador	49
Figura 60. Login.....	50
Figura 61. Menu Principal	50
Figura 62. Registrar Despesa.....	51
Figura 63. Registrar Pagamento	51
Figura 64. Consulta Despesa.....	52
Figura 65. Consultar Despesas.....	52
Figura 66. Filtrar Despesas	53
Figura 67. Perfil	53
Figura 68. Registrar Morador	54
Figura 69. Consultar Dividas	54
Figura 70. Pagar Divida	55
Figura 71. Sign Up.....	55
Figura 72. Modelo Lógico da Aplicação de Gestão de Despesas	57

Introdução

Um espaço de habitação, hoje em dia, é um “bem” que a maioria da população mundial usufrui. Em Portugal, segundo o estudo oficial de “Estatísticas da Construção e Habitação” realizado em 2012 pelo Instituto Nacional de Estatística, cada alojamento tem uma média de 1.8/2 inquilinos, ou seja, uma partilha de recursos e despesas torna-se evidente. Um estudo mais recente realizado em 2015 pela PORDATA, Base de Dados de Portugal, revela que existem aproximadamente cerca de 6 milhões de habitações em território nacional.

O uso e preservação de um espaço comum neste caso uma habitação acarreta várias responsabilidades aos seus inquilinos, portanto uma gestão de tarefas nomeadamente despesas torna-se algo necessário. Para facilitar este trabalho o desenvolvimento de uma aplicação que tenha como principais funções registar, gerir e arquivar despesas revela grande interesse para a facilitação da realização desta necessidade.

Sendo esta o relatório final do trabalho prático algumas alterações foram feitas aos diagramas e modelos da primeira fase devido a novas decisões e alterações feitas durante o desenvolvimento da aplicação.

Objetivos

Tendo em conta a contextualização acima descrita, sabemos que é fundamental a gestão das várias responsabilidades que um alojamento tem, tendo assim como principal objetivo deste trabalho pratico a criação de uma aplicação que auxiliará a gestão de um apartamento, utilizando as técnicas e modos quer de estudo quer de implementação lecionados na Unidade Curricular de Desenvolvimento de Sistemas Software.

O principal objetivo desta aplicação será permitir aos utilizadores registar, gerir e consultar as despesas, conseguindo ter assim um espaço comum onde possam partilhar todas as responsabilidades referentes ao imóvel do qual usufruem e partilham.

Estrutura do Relatório

Este relatório descreve detalhadamente, o processo de estudo e conceção do trabalho prático, que se encontra de seguida apresentado através de diferentes tipos de modelos, gráficos, planos e posteriormente desenvolvimento de uma aplicação de *software*.

Funcionalidades

A aplicação necessita de implementar algumas funcionalidades básicas e necessárias para servir o propósito para o qual foi criada em primeiro lugar. Será necessário inicialmente qualquer utilizador que se torne morador registar se na aplicação e futuramente fazer o devido login para isso é preciso haver uma classe denominada Morador que serve esse propósito. Cada entidade que já conste no sistema poderá proceder então ao registo de uma nova despesa preenchendo diversos campos sobre a mesma, poderá consultar o histórico do apartamento onde se encontra a habitar de momento desde do mês que se tornou inquilino. A aplicação permitirá também aos utilizadores pagar alguma despesa do modo que melhor servir as suas necessidades. Será visível uma diferenciação entre despesas correntes tais como luz e agua entre outras e despesas extraordinárias que resumidamente são despesas que apenas existem ocasionalmente tal como por exemplo o arranjo de um bem comum.

Cada despesa terá os seus campos tais como o valor, a data limite, posteriormente também terá por exemplo a data em que o respetivo pagamento foi efetuado e o morador que o fez. Por final o morador conseguirá imediatamente após fazer o login verificar quanto lhe falta pagar de momento ou seja o que tem em dívida como também quanto já “investiu” em despesas no apartamento onde se encontra a morar de momento.

1. Modelo de Domínio

O modelo de domínio é um esquema que captura todas as entidades do sistema, denominadas classes e todos os relacionamentos entre estas. Para conseguir cumprir todos os parâmetros acima descritos começou-se por estudar o caso concluindo que temos como classes principais:

- Morador
- Despesa
- Pagamento

Estas classes são fundamentais, sendo o core do *software* que se irá desenvolver.

1.1 Morador

Começamos por nos focar no morador que para além de informações pessoais tais como o nome e o correio eletrónico terá o seu histórico de despesas. Houve a necessidade de haver também a criação de um período de Ocupação para conseguir futuramente associar quais as despesas que pertencem ao seu período, evitando assim situações em que existem contas para pagar pendentes, mas que não são referentes ao período de tempo de ocupação do utilizador em questão. Como pudemos constar na ilustração todas as relações são de 1 para 1. O histórico de despesas denominado “Lista de Despesas será associado à Lista de Despesas do apartamento sendo este um subconjunto desse. A primeira versão então naturalmente resultou da descrição acima feita.

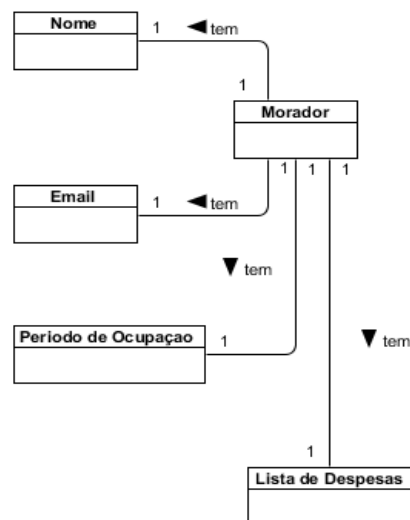


Figura 1. Morador - Versão 1

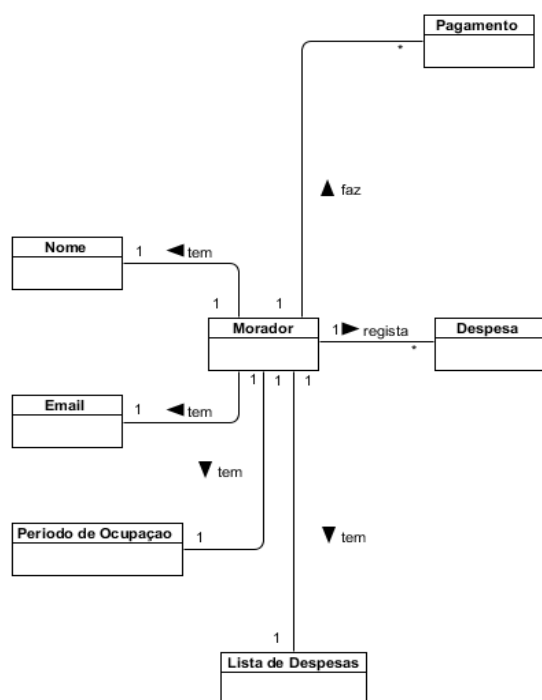


Figura 2. Morador – Versão 2

Posteriormente chegou-se à conclusão que o morador também terá associado diretamente a ele mesmo a despesa podendo regista-la e o pagamento das mesmas, havendo uma atualização resultando numa nova modulação do domínio podendo ser visualizado no exemplo ilustrativo a seguir. As duas relações são de 1 para * no sentido do Morador quer para o Pagamento quer para a Despesa, sendo assim ilustrado que ele pode fazer vários registos e pode proceder a vários pagamentos.

Por fim, houve a identificação por parte dos elementos do grupo de trabalho que iria ser útil ao utilizador ter acesso imediato ao valor total que já pagou desde da data de início da sua estadia até à data atual, tal como em contrapartida saber a dívida atual, por outras palavras quanto lhe faltava pagar de despesas que já tenham sido lançadas até ao momento, dá se assim origem à última versão referente ao morador e as suas relações, nesta ultima versão podemos constar com uma classe de nome Balanço de Contas que se ramifica nas duas situações acima abordadas sendo elas o Histórico e a Dívida.

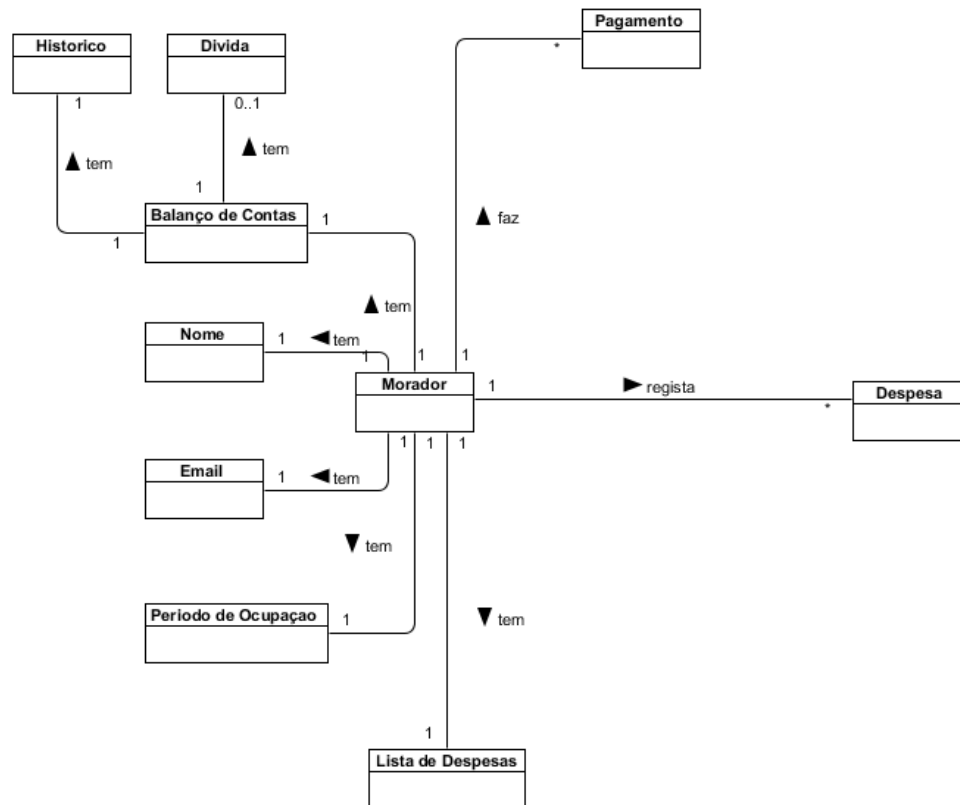


Figura 3. Morador - Versão 3

1.2 Pagamento

Esta classe consta como uma das centrais da aplicação devido a ser o principal motivo da mesma, tudo se desenrola há volta do pagamento pois é com esta ação que sabemos em que estado estamos, ou seja, por outras palavras sabe-se o que já está resolvido e pode ser arquivado, com isto quer se dizer o que já foi pago, e também sabemos obviamente o que ainda se encontra pendente. O pagamento poderá ser efetuado de 3 maneiras diferentes sendo estas Multibanco, Cartão de Credito e dinheiro, tendo assim o utilizador oportunidade de pagar determinada despesa da maneira que mais lhe desejar. Cada pagamento terá uma data associada sendo relativa ao dia em que o mesmo foi realizado, este é um fator importante pois com esta data pudemos saber se houve o cumprimento em termos de prazo de pagamento por parte do utilizador em questão.

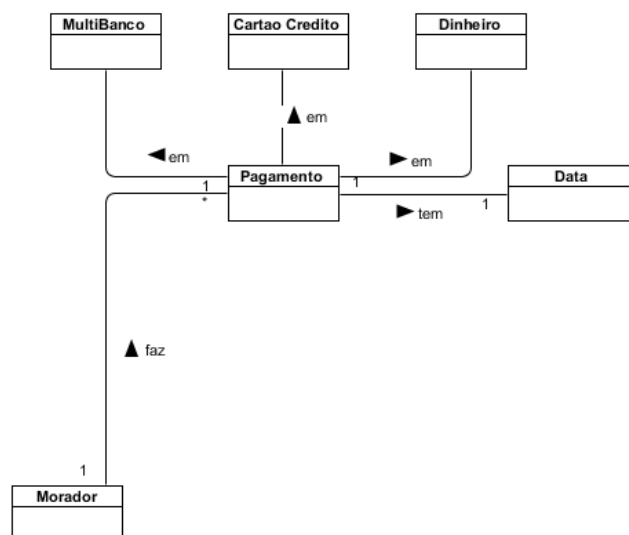


Figura 4. Pagamento – Versão 1

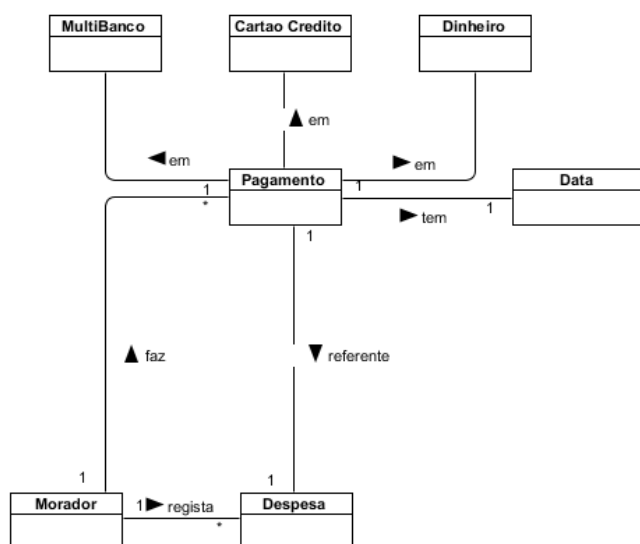


Figura 5. Pagamento – Versão 2

Obviamente que após a criação da classe Despesa que será descrita posteriormente foi também necessário estabelecer uma relação entre essa classe e a corrente classe Pagamento originando assim a uma evolução do modelo de domínio como se pode constatar de seguida.

Por fim, chegou-se à conclusão que era necessário para a parte do balanceamento de contas do utilizador funcionar em pleno existir um valor associado ao pagamento que foi efetuado, portanto houve a adição do valor ao pagamento havendo uma relação de 1 para 1 entre eles. Também nesta fase houve a atualização e completação de algumas relações anteriormente ainda em discussão tais como as formas de pagamento que chegou se a conclusão que por exemplo um pagamento pode ou não ser efetuado por multibanco sendo uma relação de 1 para 0..1. De seguida pode-se visualizar a ultima versão referente ao Pagamento, confirmando os objetivos a cima referidos.

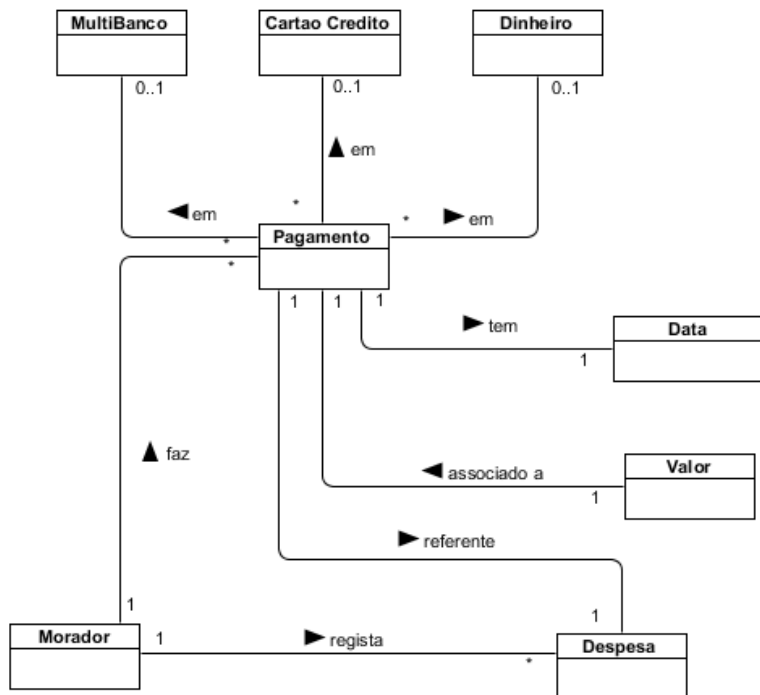


Figura 6. Pagamento – Versão 3

1.3 Despesa

Sendo a despesa o foco da relação entre as duas classes anteriormente descritas, Morador e Pagamento, decidiu-se torná-la a terceira e última classe central da aplicação. Basicamente serve de ponte de comunicação que existe entre as outras duas classes centrais. Foi necessário partir então de uma classe principal Despesa que foi necessário ramificar em duas subclasses distinguidas pelo nome dado sendo uma delas a Despesa Corrente e a outra Despesa Extraordinária. Também houve a decisão de qualquer despesa que fosse inserida na aplicação por parte de algum utilizador será automaticamente arquivada numa Lista de Despesas que servirá para posterior consulta caso algum utilizador o deseje fazer. Com a exceção da relação entre Despesa e Lista de Despesas que obviamente tem cardinalidade de 1 para *, todas as outras são relações de 1 para 1 como se pode verificar. Como já citado anteriormente cada Despesa Recorrente pode ser eletricidade, gás, água e tem também uma data a si associada como data limite para ser paga, já a Despesa Extraordinária não tem necessariamente uma data, mas também tem um leque de opções tais como Reparação, Utensilio, Material e Outro. Em termos de relações são todas 1 para 1 pois são todas subclasses ou variáveis da classe originária, Despesa. Podemos de seguida ver a parte do modelo de domínio que foi originada depois do estudo completo da classe Despesa

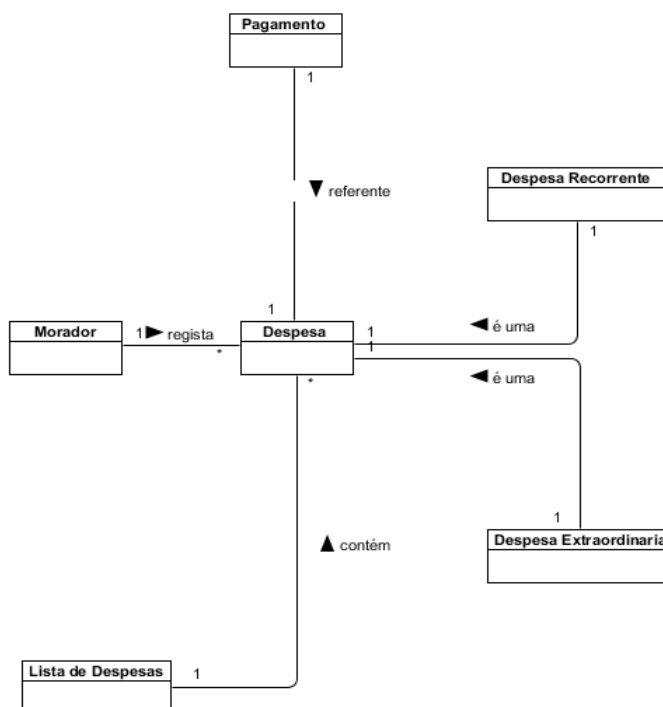


Figura 7. Despesa – Versão 1

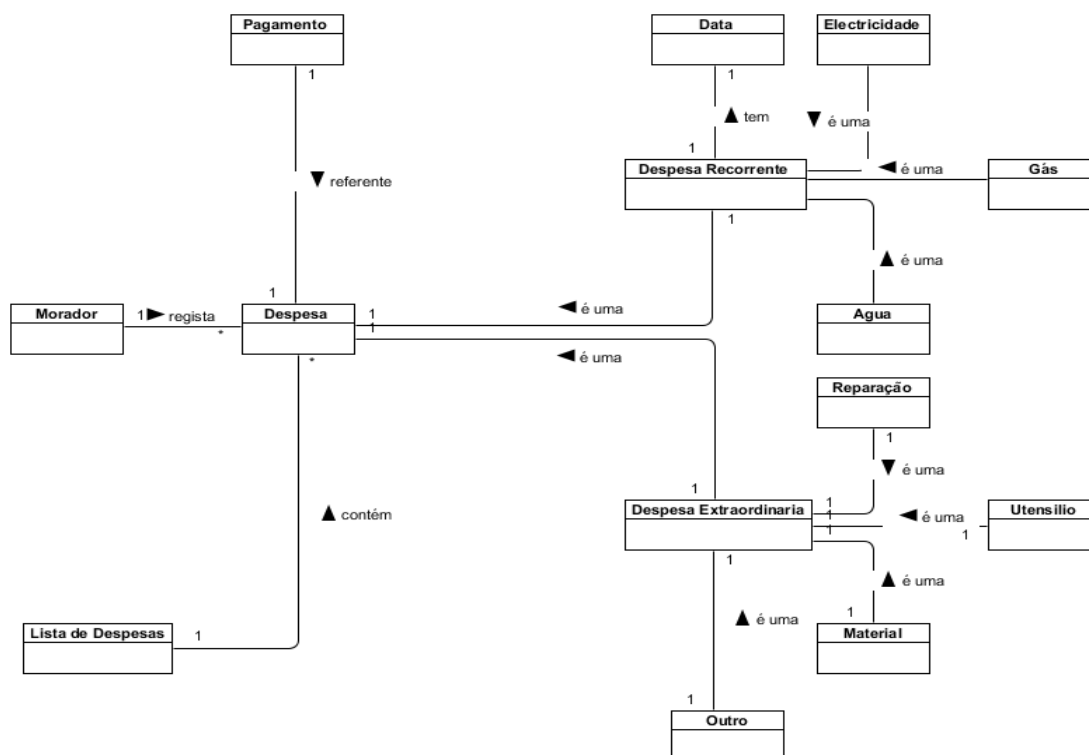


Figura 8. Despesa – Versão 2

Mais uma vez houve a necessidade de alterar devido à inclusão do valor esta classe também terá uma relação com a despesa por razões bastante óbvias sendo estas que cada despesa tem um valor associado a si mesma dando assim origem a um relacionamento de 1 para 1. Também nesta fase de estudo chegou-se ao consenso que a Despesa Recorrente deveria também ter a opção outras duas situações que encontra mos persuasivas para tal implementação foram, caso o apartamento em causa não seja de algum dos moradores havendo assim o pagamento de uma renda e também caso os inquilinos cheguem ao acordo de haver uma empregada de limpeza por exemplo de duas em duas semanas, passa a ser também uma situação recorrente.

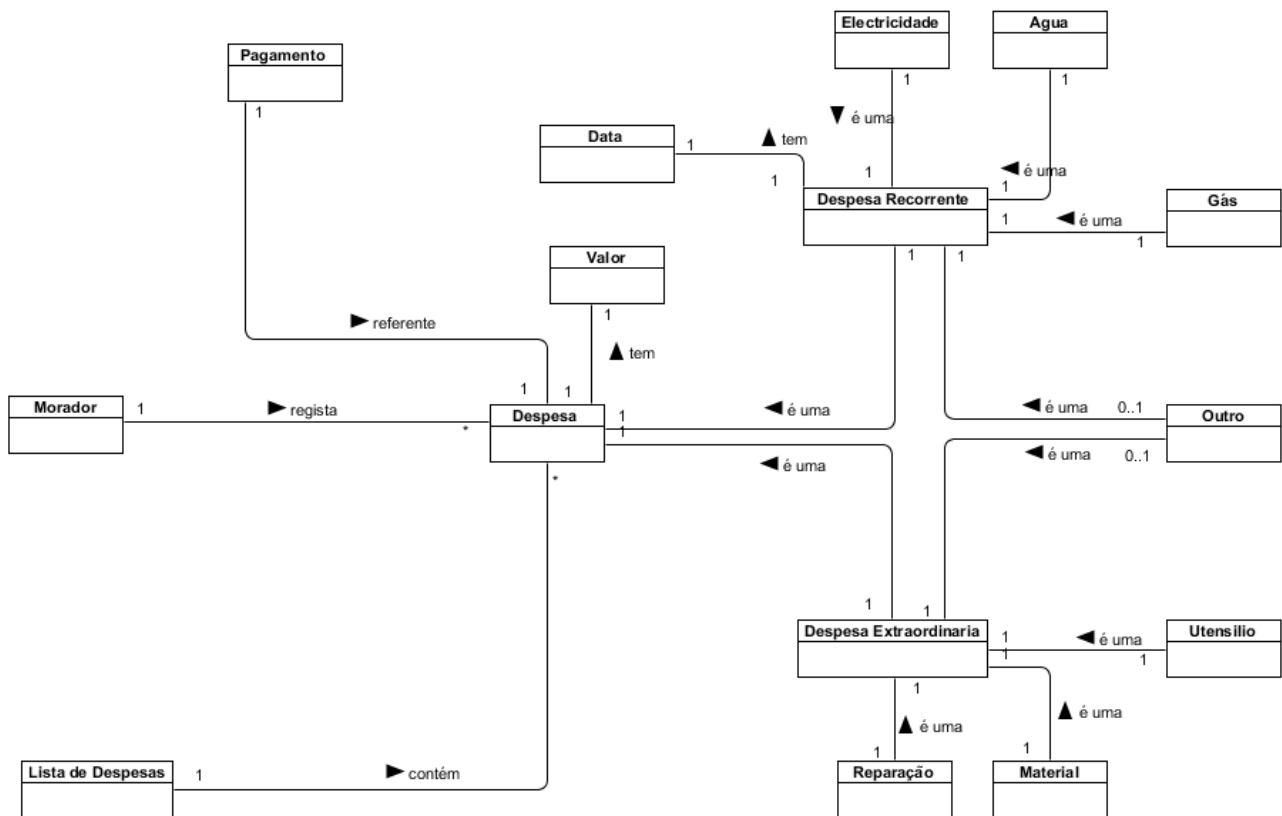


Figura 9. Despesa – Versão 3

Por fim, para finalizar este primeiro estudo encontra-se o modelo de domínio que ilustra as funcionalidades a cima descritas, e juntando todos os esquemas acima apresentados. Como observações complementares é de notar que entidades como Data e Lista de despesas servem múltiplos propósitos e por isso partilhadas por classes superiores, mas com fins totalmente diferentes.

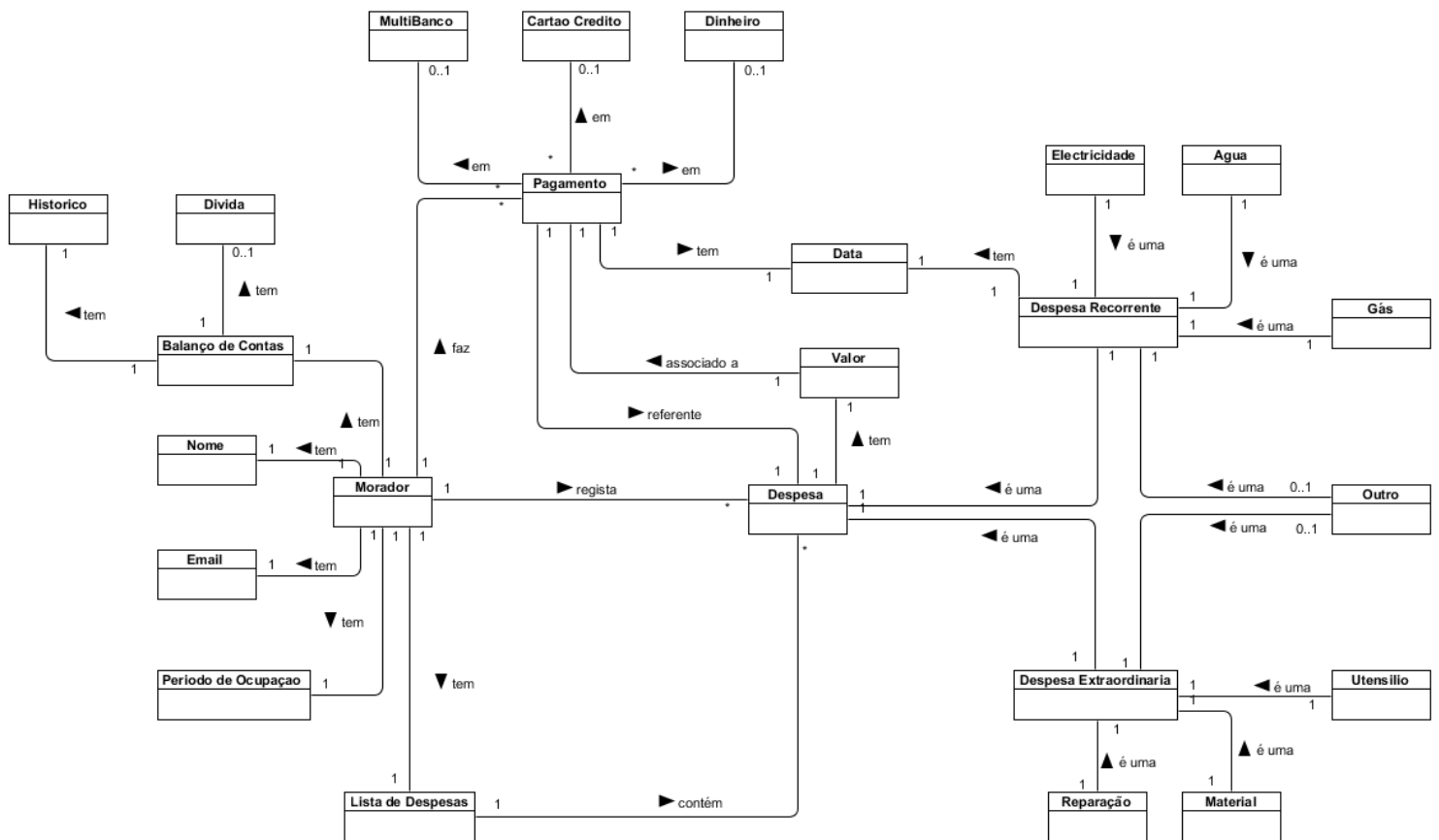


Figura 10. Modelo de Domínio – 1ºFase

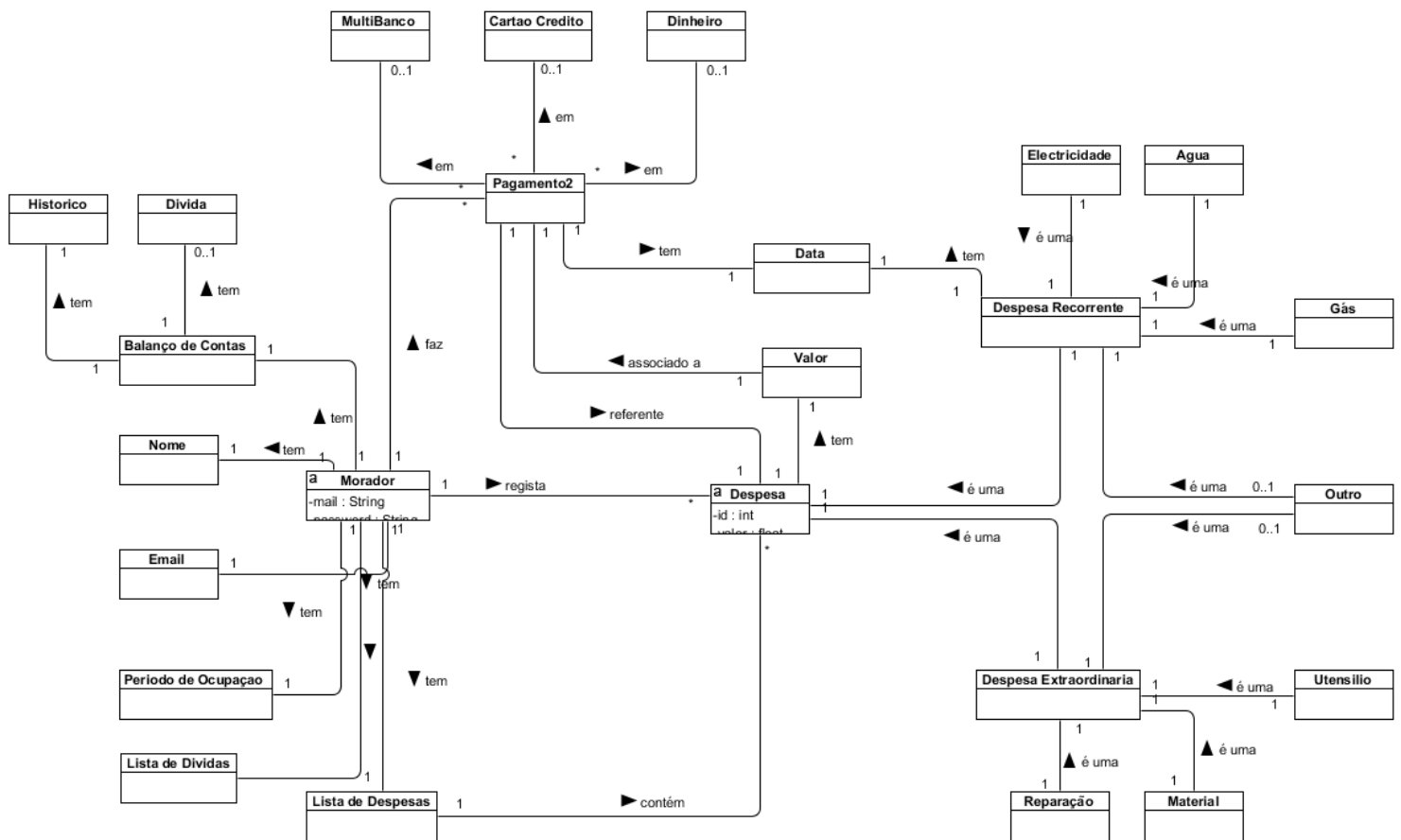


Figura 11. Modelo de Domínio – 2ªFase

2. Diagrama de Use Cases

Para modelar um sistema, o aspeto mais importante a capturar é o seu comportamento ao longo da execução deste, complementando o seu comportamento estático. Um diagrama capaz de representar este comportamento dinâmico é o diagrama de *use cases*.

Nestes diagramas, existem agentes (internos ou externos) conhecidos como atores do sistema. Assim, estes diagramas consistem em atores, ações e relações entre estas. Cada *use case* representa uma funcionalidade particular do sistema.

Num sistema de partilha de despesas, foi determinado que seriam necessários apenas dois atores, que interagem com este, o morador e o administrador.

O morador é o agente que se regista, entra/sai na plataforma e é-lhe possibilitado registar despesas, efetuar pagamentos e consultar a lista de despesas.

Por outro lado, o administrador possui todas estas permissões e pode ainda editar dados de um morador ou mesmo removê-lo do sistema.

Assim, o diagrama de *use cases* para este problema é o seguinte:

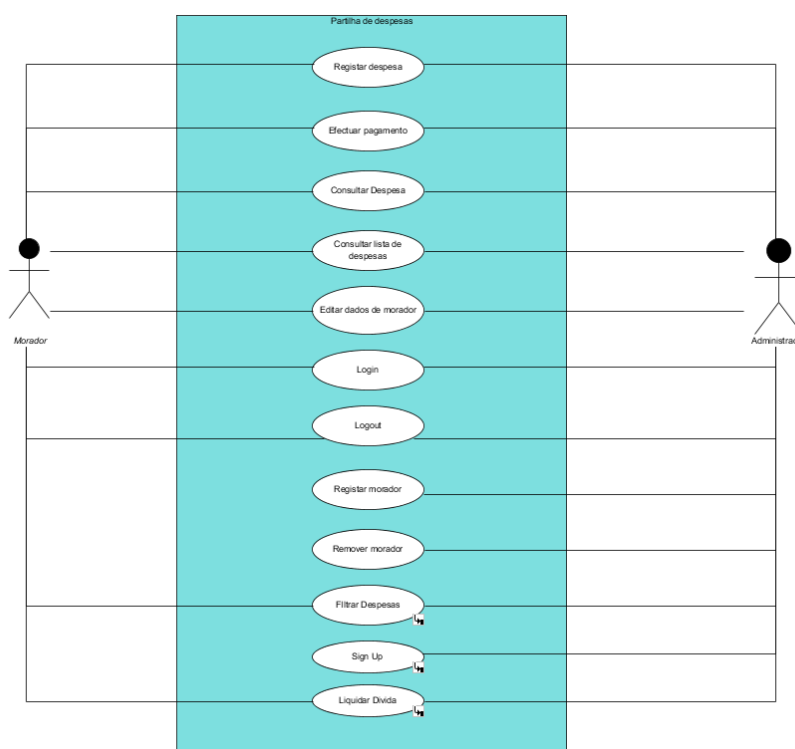


Figura 12. Diagrama Use Cases

Registrar Despesa

Esta ação permite aos moradores registrar uma despesa na lista de despesas atual. Uma vez que esteja autenticado, menu “Despesas”, se clicar em “Registrar despesa” e os dados forem corretamente introduzidos o sistema vai adicionar esta despesa à lista. Se os dados forem introduzidos de forma incorreta, o sistema pede ao utilizador para este inserir novamente os dados. Esta ação termina com a lista de despesas atualizada e ainda no menu “Despesas”. O utilizador pode voltar a registar uma despesa se assim entender.

Brief Description	Utilizador regista uma despesa	
Preconditions	Utilizador no menu "Despesas"	
Post-conditions	Utilizador volta ao menu "Despesas"	
Flow of Events		Actor Input
	1	Clica em "Registrar despesa"
	2	
	3	Introduz dados
	4	
	5	
		System Response
		Pede dados
		Regista despesa
		Volta ao menu "Despesas" atualizado
Exceção 1 (passo 3) [Dados Incorretos]		Actor Input
	1	
	2	
		System Response
		Informa que dados são invalidos
		Regressa ao passo 2
Alternativa 1 (passo 3) [Retroceder]		Actor Input
	1	Clica em "Retroceder"
	2	
		System Response
		Regressa ao passo 5

Figura 13. Use Case – Registrar Despesa

Efectuar Pagamento

Esta ação permite aos utilizadores efetuar o pagamento de uma despesa. O sistema irá distinguir entre despesas já pagas e não pagas através do *layout* destas. O utilizador informa o sistema da despesa referente ao pagamento, o sistema informa o utilizador do montante a pagar, seguidamente o Actor escolhe o tipo de pagamento e paga, após o sistema verificar se o pagamento foi concluído com sucesso informa o utilizador e adiciona a despesa à lista de despesas. Esta ação influencia o balanço de contas de todos os moradores. Quem efetuar o pagamento terá este montante adicionado ao seu balanço e será adicionado o mesmo valor, negativo, aos restantes moradores. Assim, numa divisão dividida por três pessoas, se a primeira despesa de 20€, por exemplo, for paga por um morador, este terá um balanço positivo de 20€ e os restantes dois terão um balanço negativo de 20€, cada um. Isto permite a cada morador controlar os seus gastos e quando precisa ser a sua vez a pagar.

Brief Description	Utilizador regista o pagamento de uma despesa	
Preconditions	Utilizador no menu "Consulta de Despesa"	
Post-conditions	Utilizador volta ao estado inicial	
Flow of Events		Actor Input
	1	Clica em "Registar pagamento"
	2	
	3	Introduz dados
	4	Escolhe opção de pagamento total ou parcelado
	5	
Alternativa 1 (passo 2) [Retroceder]		System Response
	1	Clica em "Retroceder"
Exceção 1 (passo 3) [Dados inválidos]		System Response
	1	Infoma que os dados estão incorretos
	2	Regressa ao passo 3

Figura 14. Use Case – Efetuar Pagamento

Consultar Despesa

Esta ação permite ao utilizador observar os detalhes de uma despesa em específico. Para tal, basta clicar em “Consultar despesa”, quando seleccionar uma despesa no menu “Despesas”. O sistema imprime os dados desta e é possível retroceder assim que for necessário.

Brief Description	Utilizador consulta despesa específica	
Preconditions	Utilizador no menu "Lista de despesas"	
Post-conditions	Utilizador no estado inicial	
Flow of Events		Actor Input
	1	Escolhe despesa
	2	Clica em "Consultar"
	3	
	4	Retrocede
	5	
		System Response
		O sistema imprime os dados da despesa
		Sistema volta ao estado inicial

Figura 15. Use Case – Consultar Despesa

Consultar Lista de Despesas

Esta ação permite aos utilizadores consultar a lista de despesas, uma vez que acede ao menu “Depesas”. O sistema irá distinguir entre despesas já pagas e não pagas através do *layout* destas. O utilizador toma conhecimento da lista de despesas e pode voltar atrás para o menu principal, se desejar. Neste menu é possível efetuar registos de pagamentos, registar uma despesa ou consultar os detalhes de uma despesa.

Brief Description	Utilizador consulta as despesas		
Preconditions	Utilizador no menu principal		
Post-conditions	Utilizador volta ao estado inicial		
Flow of Events		Actor Input	System Response
	1	Clica em "Despesas"	
	2		Imprime Lista de Despesas
	3	Clica em "Retroceder"	
	4		Volta ao estado inicial

Figura 16. Use Case – Consultar Lista Despesas

Editar dados do Morador

Esta ação permite ao administrador ou utilizador alterar os dados do seu perfil. Assim, uma vez no menu “Editar Dados Morador”, existem 3 possíveis alternativas, “Mudar email”, “Mudar Nome” ou “Mudar Password”, após a escolha de uma destas 3 o sistema pede ao utilizador os dados referentes à escolha, depois de o utilizador os inserir são verificados pelo sistema, caso algo não esteja correto o sistema reporta ao utilizador o erro dando a possibilidade de os voltar a inserir ou cancelar a ação. Após a alteração de um dos campos do utilizador este pode voltar a alterar um dos seus dados ou sair do menu.

Brief Description	Alterar dados do morador	
Preconditions	Morador previamente registado	
Post-conditions	Utilizador no menu principal	
Flow of Events		Actor Input
	1	Clica em "Editar dados"
	2	
	3	Escolher campo a editar
	4	
	5	
		System Response
		Apresenta opções de edição
		Informa alterações efectuadas
		Volta ao menu principal
Alternativa 1 (passo 3) [Alterar nome do morador]		Actor Input
	1	Inserção no novo nome
	2	
	3	Confirmar alteração
	4	
		System Response
		Verificar existência do nome inserido
		Informa alteração bem sucedida
Alternativa 2 (passo 3) [Alterar email]		Actor Input
	1	Inserção do novo email
	2	
	3	Confirmar alteração
	4	
		System Response
		Validar o email inserido
		Informa alteração bem sucedida
Alternativa 3 (passo 3) [Alterar password]		Actor Input
	1	Inserção da nova password
	2	
	3	Confirmar alteração
	4	
		System Response
		Verificar validade da password ou se é igual à actual
		Informa alteração bem sucedida
Excepção 2 (passo 3.2) [Password inválida]		Actor Input
	1	
	2	
		System Response
		Informa que a password inserida não cumpre os requisitos ou é igual à actual
		Regressa ao passo 3.1

Figura 17. Use Case – Editar Dados Morador

Login

Esta ação permite ao administrador ou utilizador entrar no sistema. Assim, uma vez no menu “Login”, o sistema pede os dados, estes são fornecidos pelo utilizador, após o sistema verificar os dados, informa que a ação foi concluída com sucesso. Em caso de o e-mail fornecido já não se encontrar no sistema, o utilizador é informado e volta a poder inserir novos dados (ou cancelar esta ação).

Brief Description	Utilizador autentica-se no sistema	
Preconditions	Morador previamente registado	
Post-conditions	Actor autenticado	
Flow of Events		Actor Input
		System Response
	1	Clica em "Login"
	2	Pede a introdução do username e password
	3	Introduz o username e password
	4	Valida os dados introduzidos
Alternativa 1 (passo 3) [Dados incorrectos]	5	Informa que o utilizador foi autenticado com sucesso
		Actor Input
		System Response
	1	Informa que os dados introduzidos no passo 2 estão incorrectos
	2	Regressa ao passo 2

Figura 18. Use Case – Login

Logout

Esta ação permite ao administrador ou utilizador sair do sistema. Assim, uma vez no menu “Logout”, o sistema fecha a sessão do dado utilizador e informa que a ação foi concluída com sucesso.

Brief Description	Utilizador termina sessão	
Preconditions	Utilizador previamente autenticado	
Post-conditions	Logout efectuado	
Flow of Events		Actor Input
	1	Pede ao sistema para efectuar logout
	2	
	3	Confirma logout
	4	
	5	
		System Response
		Pede ao actor confirmação da acção
		Efectua logout
		Informa que o actor já não se encontra autenticado no sistema
Alternativa 1 (passo 3) [Cancela logout]		Actor Input
	1	Cancela logout
	2	
		System Response
		Informa que o actor continua autenticado no sistema

Figura 19. Use Case – Logout

Registrar Morador

Esta ação permite apenas ao administrador registrar um morador no sistema. Assim, uma vez no menu “Registrar morador”, o sistema pede os dados, estes são fornecidos pelo administrador e o sistema faz o registo e informa que a ação foi concluída com sucesso. O administrador pode em qualquer altura cancelar o registo. Em caso de o e-mail fornecido para registrar já se encontrar no sistema, o administrador é informado disto e volta a poder inserir novos dados (ou cancelar esta ação).

Brief Description	Registrar novo morador no apartamento do administrador	
Preconditions	Administrador autenticado e no menu principal	
Post-conditions	Novo morador registado	
Flow of Events		Actor Input
		System Response
	1	Pede ao sistema para registar um novo morador
	2	Pede ao actor os dados do novo morador
	3	Inserir dados
	4	Validação dos dados inseridos
Alternativa 1 (passo 3) [Cancelar novo registo]	5	Faz o registo do novo morador
	6	Informa que o morador foi registado com sucesso
Excepção 1 (passo 4) [Dados inválidos]		Actor Input
		System Response
	1	Informa que os dados estão inválidos
	2	Volta para o passo 3

Figura 20. Use Case – Registrar Morador

Remover Morador

Esta ação permite apenas ao administrador remover um morador do sistema. Assim, uma vez no menu “Remover morador”, o sistema pede os dados, estes são fornecidos pelo administrador e o sistema faz o registo e informa que a ação foi concluída com sucesso. Em caso de o e-mail fornecido para remover não se encontrar no sistema, o administrador é informado disto e volta a poder inserir novos dados (ou cancelar esta ação).

Brief Description	Remoção de um morador	
Preconditions	Administrador autenticado e estar no menu principal	
Post-conditions	Morador removido e volta ao menu principal	
Flow of Events		Actor Input
	1	Pede ao sistema para remover o morador
	2	
	3	Insere o email
	4	
	5	
	6	
		System Response
		Pede ao actor o email do morador a remover
		Verifica a validade do email e se o morador tem despesas por pagar
		Remove morador
		Informa que morador foi removido
Alternativa 1 (passo 3) [Cancelar remoção do morador]		Actor Input
	1	Cancela remoção do morador
	2	
		System Response
		Informa que a remoção do morador foi cancelada
Excepção 1 (passo 4) [Email não registado]		Actor Input
	1	
	2	
		System Response
		Informa que email inserido não está registado
		Regressa a 3
Excepção 2 (passo 4) [Morador com despesas pendentes]		Actor Input
	1	
	2	
		System Response
		Informa que o morador a remover tem despesas pendentes
		Cancela remoção de morador

Figura 21. Use Case – Remover Morador

Filtrar Despesas

Esta ação permite ao utilizador visualizar as suas despesas filtradas de acordo com parâmetros específicos tais como, despesas anteriores ou posteriores a uma data, estado, tipo ou descrição. Depois de escolhidos os parâmetros e confirmar o sistema retorna ao menu “Consultar Despesas” em que apresenta agora as despesas segundo o que foi definido anteriormente.

Brief Description	Consultar despesas segundo filtros		
Preconditions	Utilizador no "Menu Despesa"		
Post-conditions	Volta ao estado inicial		
Flow of Events		Actor Input	System Response
	1	Carrega no botão "Filtrar"	
	2		Pede para introduzir as opções de filtro
	3	Introduz opções de filtro	
	4		Volta ao menu de despesa e apresenta despesas que se encaixam nos parâmetros

Figura 22. Use Case – Filtrar Despesas

Sign Up

Esta ação permite a criação de um novo administrador e consequentemente um novo apartamento. São pedidos os mesmos dados de um registo de um novo morador e ainda o pin do novo apartamento. O sistema faz o registo e caso seja bem sucedido o utilizador é redirecionado para o menu inicial. Caso os dados sejam inválidos o utilizador é informado e pode escolher inserir novos dados ou voltar ao menu inicial.

Brief Description	Registar novo administrador		
Preconditions	Utilizador no Menu Inicial		
Post-conditions	Utilizador no Menu Inicial		
Flow of Events		Actor Input	System Response
	1	Carrega no botão "Sign Up"	
	2		Pede dados relativos ao administrador
	3	Inserir dados	
	4		Regista novo administrador
	5		Volta ao Menu Inicial
Exceção 1 (passo 3) [Dados inválidos]		Actor Input	System Response
	1		Informa que dados estão inválidos
	2		Regressa ao passo 2
Alternativa 1 (passo 3) [Retroceder]		Actor Input	System Response
	1	Carrega em "Retroceder"	
	2		Volta ao passo 5

Figura 23. Use Case – Sign Up

Liquidar Dívida

Esta ação permite a um utilizador consultar e liquidar dívidas que tenha para com outro morador. Ao carregar no botão “Consultar Dívidas” do menu principal o utilizador é redirecionado para um menu onde é apresentada uma lista com os moradores a quem deve e o valor. Escolhendo a dívida e carregando no botão “Efetuar Pagamento” é aberto um menu de pagamento em que o utilizador indica o valor da dívida que pretende liquidar. Confirmando, o sistema volta para o menu de consulta de dívidas.

Brief Description	Liquidar divida para com outro morador		
Preconditions	Estar no menu "Consultar Dividas"		
Post-conditions	Divida parcialmente ou totalmente liquidada e volta ao estado inicial		
Flow of Events		Actor Input	System Response
	1	Escolhe a despesa a liquidar e carrega em "Efectuar Pagamento"	
	2		Pede dados
	3	Insere dados	
	4		Valida dados inseridos
	5		Volta ao menu "Consultar Dividas"

Figura 24. Use Case – Liquidar Dívida

3. Proposta inicial da Interface

Nesta primeira fase foi feito também um esboço do aspeto geral e das funcionalidades da interface, que irá ser implementada no futuro utilizando tecnologias orientadas a objetos, nomeadamente Java Swing.

Para a formulação desta proposta de interface foram obviamente seguidos os modelos realizados anteriormente, Domínio e Use Cases.

3.1 Login

No ecrã de login não há distinção entre um utilizador normal e o administrador. Após a introdução das credenciais, dependendo dos privilégios o utilizador é redirecionado para a página devida.

Caso não possua uma conta na aplicação, o morador tem neste ponto a hipótese de completar um registo.

Figura 25. Ecrã de Login

3.2 Menu Principal (Utilizador Normal)

Trata-se do ecrã seguinte ao Login, em que o utilizador tem um conjunto de informações imediatamente disponibilizadas, tais como a total dívida de todas as despesas que ainda não pagou, o total pago em despesas desde que reside no apartamento e uma lista com todas as despesas em que está envolvido. Nesta lista, as despesas marcadas a vermelha identificam o pagamento em falta e as marcadas a verde indicam que a liquidação da despesa já foi efetuada.

Neste menu, pode ainda consultar a lista total de despesas do apartamento utilizando o botão “Consultar Despesas”, sendo depois redirecionado para outro ecrã.

Por fim, existe também a opção de editar os seus dados pessoais em “Editar Perfil”.

Figura 26. Ecrã do Menu Principal (Utilizador Normal)

3.3 Menu Principal (Administrador)

Este menu é praticamente igual ao menu de um utilizador normal, sendo que agora incorpora as operações disponíveis para um administrador, “Registar Morador” e “Remover Morador”.

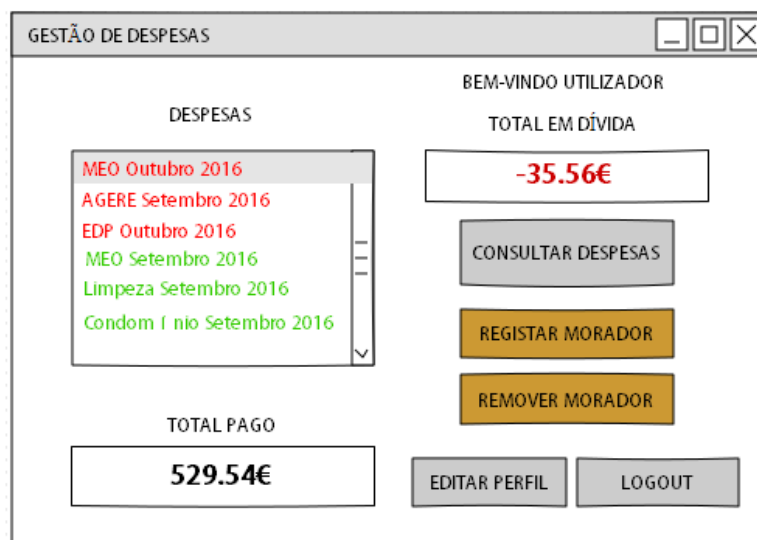


Figura 27. Ecrã do Menu Principal (Administrador)

3.4 Consultar Despesas

Este menu é acedido através do menu principal e são mostradas numa lista todas as despesas do apartamento. Neste ponto, o utilizador pode registar uma nova despesa em “Registar Despesa” ou aceder aos detalhes de uma despesa carregando sobre uma das que está presente na lista.

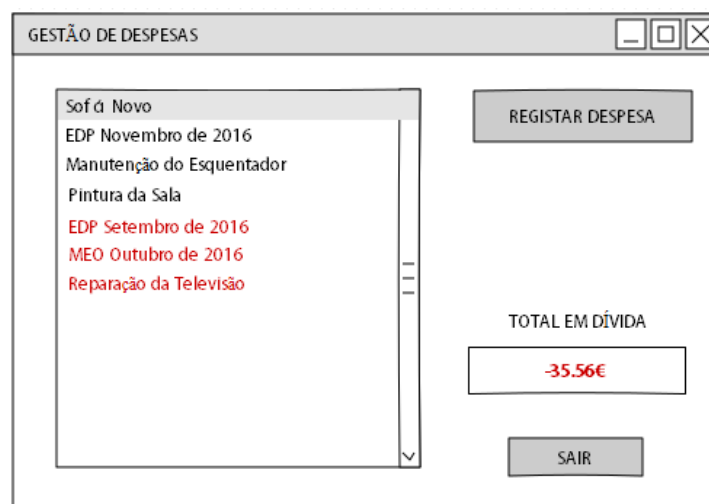


Figura 28. Ecrã de Consultar Despesas

3.5 Consultar Despesa

Quando um utilizador pretende aceder aos detalhes de uma despesa é apresentado este ecrã com um conjunto de informações associadas, descrição da despesa, valor, moradores a quem a despesa se refere, data limite de pagamento, e informação de pagamento.

Se um utilizador pretender registar o pagamento dessa despesa acede a um novo ecrã criado para o efeito em “Registar Pagamento”.

The screenshot shows a window titled 'GESTÃO DE DESPESAS'. Inside, the title 'Reparação de Interruptor' is at the top. Below it is a text box containing '[DESCRIÇÃO] Interruptor da sala não funcionava. Reparação efectuada.' To the left, there are three sections: 'Valor' with '35 €', 'Moradores abrangidos pela despesa' with 'Gonçalo Guedes [50%]' and 'Né Ison Semedo [50%]', and 'Data Limite' with '25/11/2016'. To the right, there is a 'Referência de Pagamento' section with '[ENTIDADE] 10311' and '[REFERÊNCIA] 1 23 313 839'. At the bottom right, there are two buttons: 'REGISTAR PAGAMENTO' and 'SAIR'.

Figura 29. Ecrã dos detalhes de uma despesa

3.6 Registar Pagamento

Um utilizador que queira registar um pagamento de uma despesa tem de preencher os detalhes presentes neste ecrã. É necessário especificar o método de pagamento e a data em que foi efetuado. Tem também a opção de escrever qualquer observação que ache pertinente.

The screenshot shows a window titled 'GESTÃO DE DESPESAS'. Inside, the title 'Reparação de Interruptor' is at the top. Below it is a section for 'MÉTODO DE PAGAMENTO' with three radio buttons: 'Multibanco', 'Cartão de Crédito', and 'Dinheiro'. To the right is a large text box labeled 'OBSERVAÇÕES'. Below the payment method section is a text box labeled 'INSIRA DATA DE PAGAMENTO'. At the bottom right, there are two buttons: 'CONFIRMAR PAGAMENTO' and 'SAIR'.

Figura 30. Ecrã de registo de pagamento

3.7 Registar Despesa

Quando um utilizador pretende registar uma nova despesa é-lhe apresentado um ecrã em que terá de inserir as informações pedidas referentes a essa despesa.

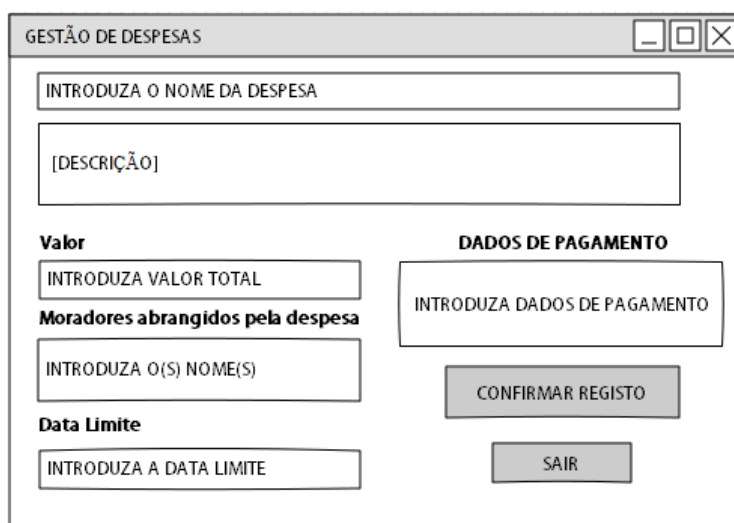


Figura 31. Ecrã de registo de despesa

3.8 Detalhes de despesa paga

A um utilizador que pretenda consultar os detalhes de uma despesa que já foi paga é apresentado o seguinte ecrã:

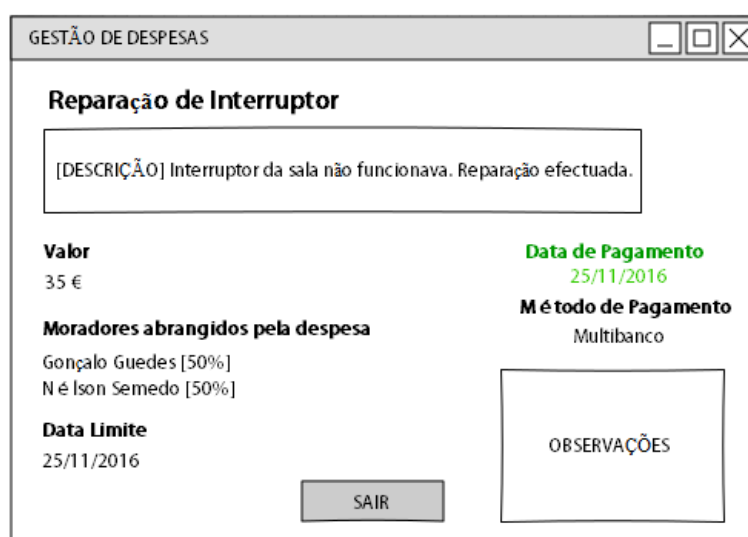


Figura 32. Ecrã de detalhes de despesa paga

4. Diagramas de Sequência

Os diagramas de sequência representam cronologicamente as interações entre o ator e o sistema nos diversos use cases. É uma forma simples, mas explícita que ajuda a perceber “o que faz” cada use case.

Consultar Despesa

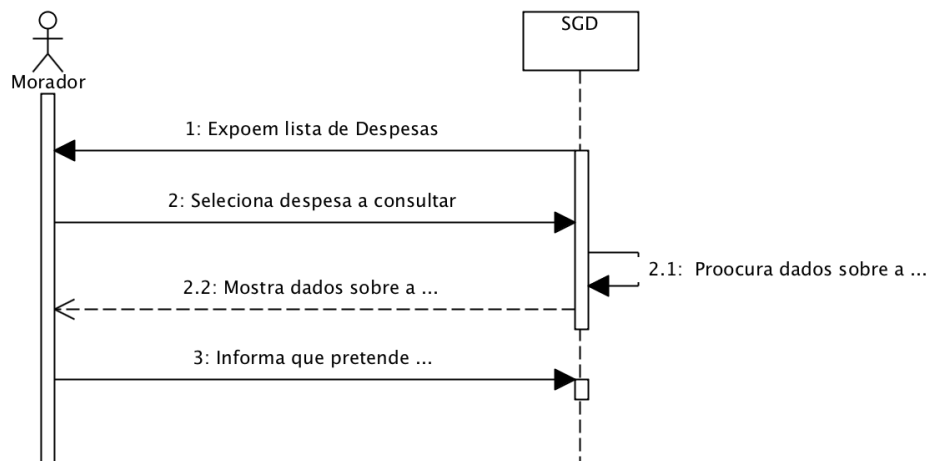


Figura 33. Diagrama de Sequência –Consultar Despesa

Consultar Lista Despesas

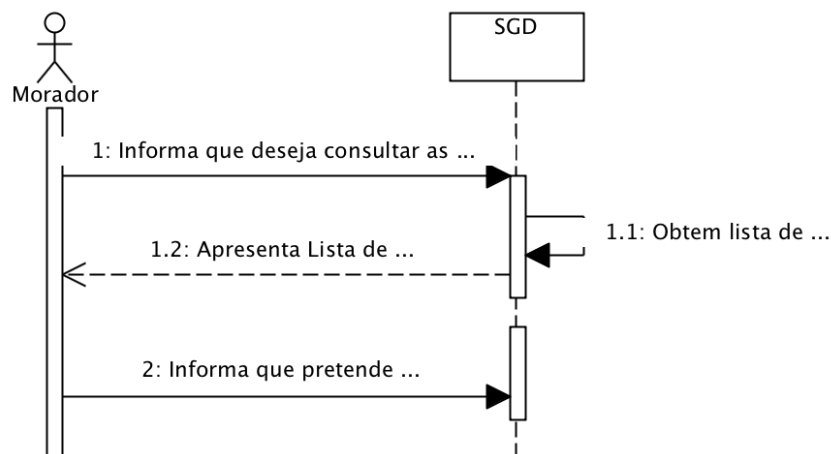


Figura 34. Diagrama de Sequência –Consultar Lista Despesa

Editar Dados Morador

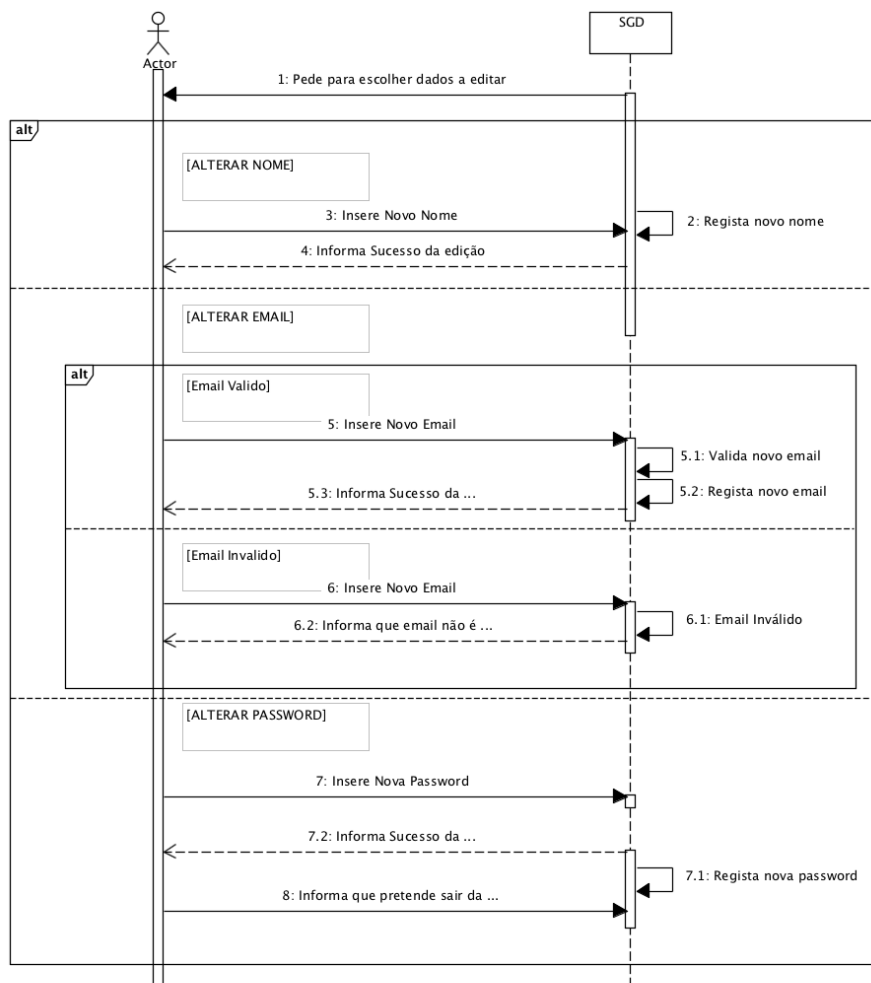


Figura 35. Diagrama de Sequência –Editar Dados Morador

Efetuar Pagamento

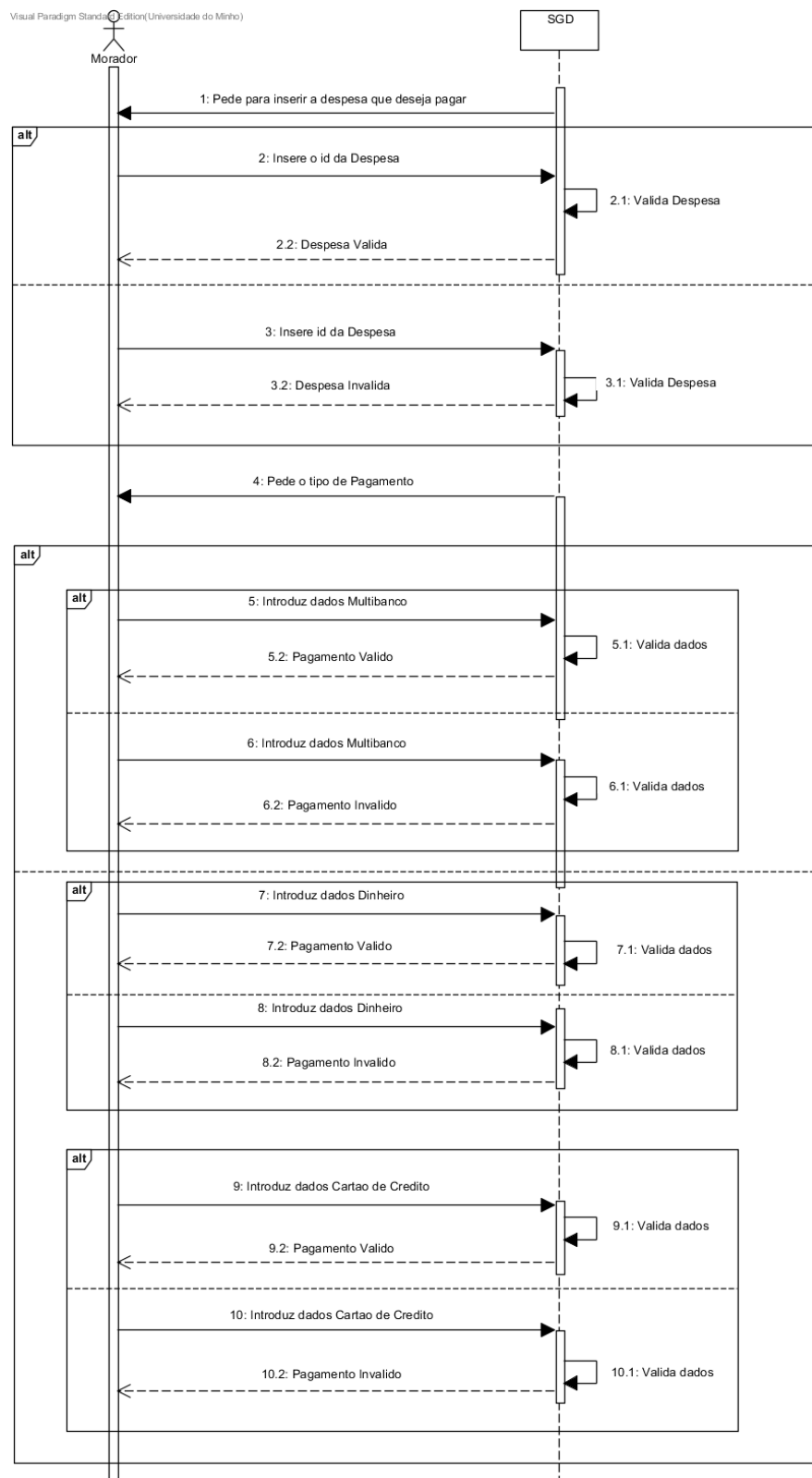


Figura 36. Diagrama de Sequência – Efetuar Pagamento

Filtrar Despesas

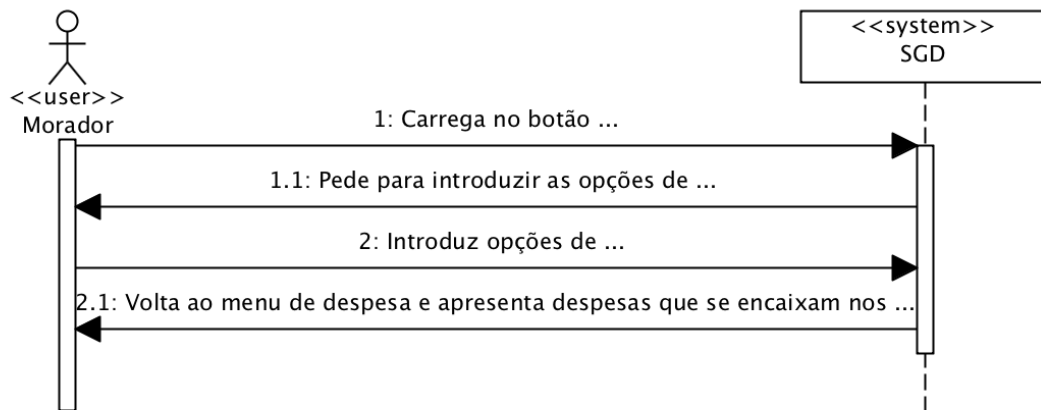


Figura 37. Diagrama de Sequência –Filtrar Despesa

Liquidar Dívida

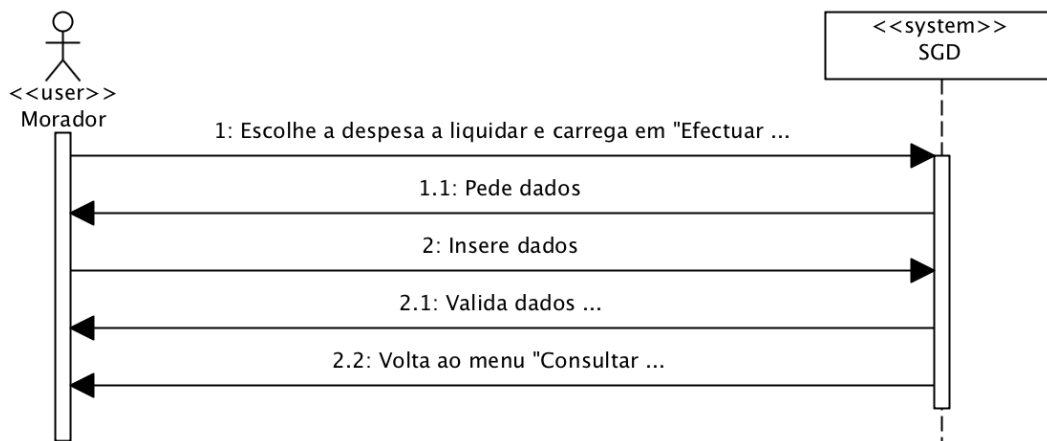


Figura 38. Diagrama de Sequência –Liquidar Dívida

Login

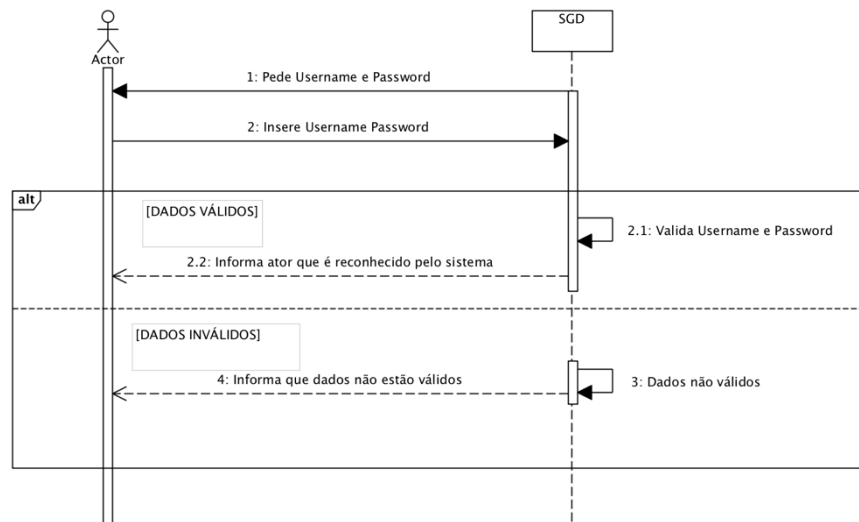


Figura 39. Diagrama de Sequência – Login

Logout

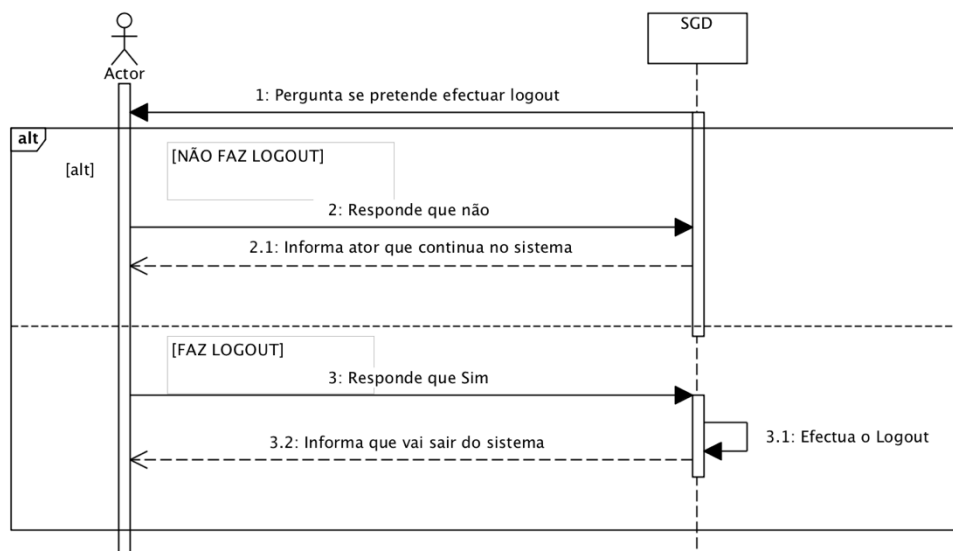


Figura 40. Diagrama de Sequência – Logout

Registrar Morador

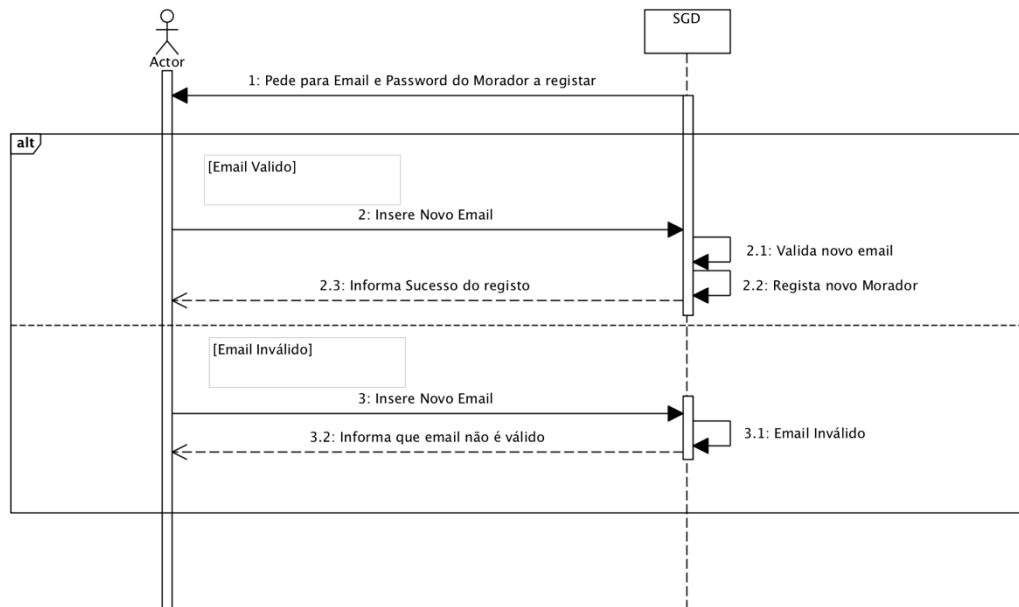


Figura 41. Diagrama de Sequência – Registrar Morador

Remover Morador

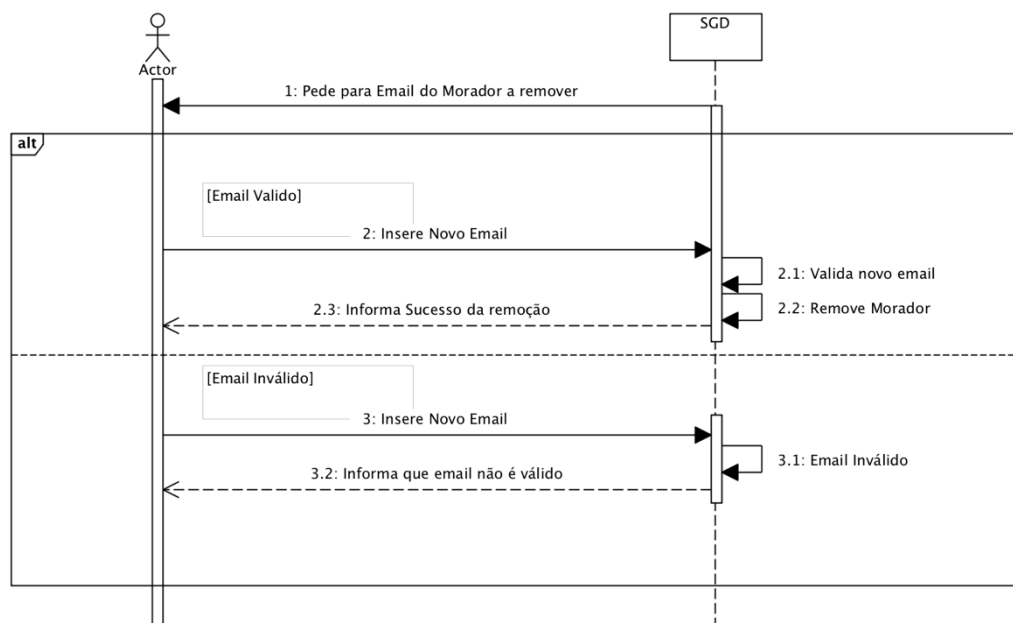


Figura 42. Diagrama de Sequência – Remover Morador

Sign Up

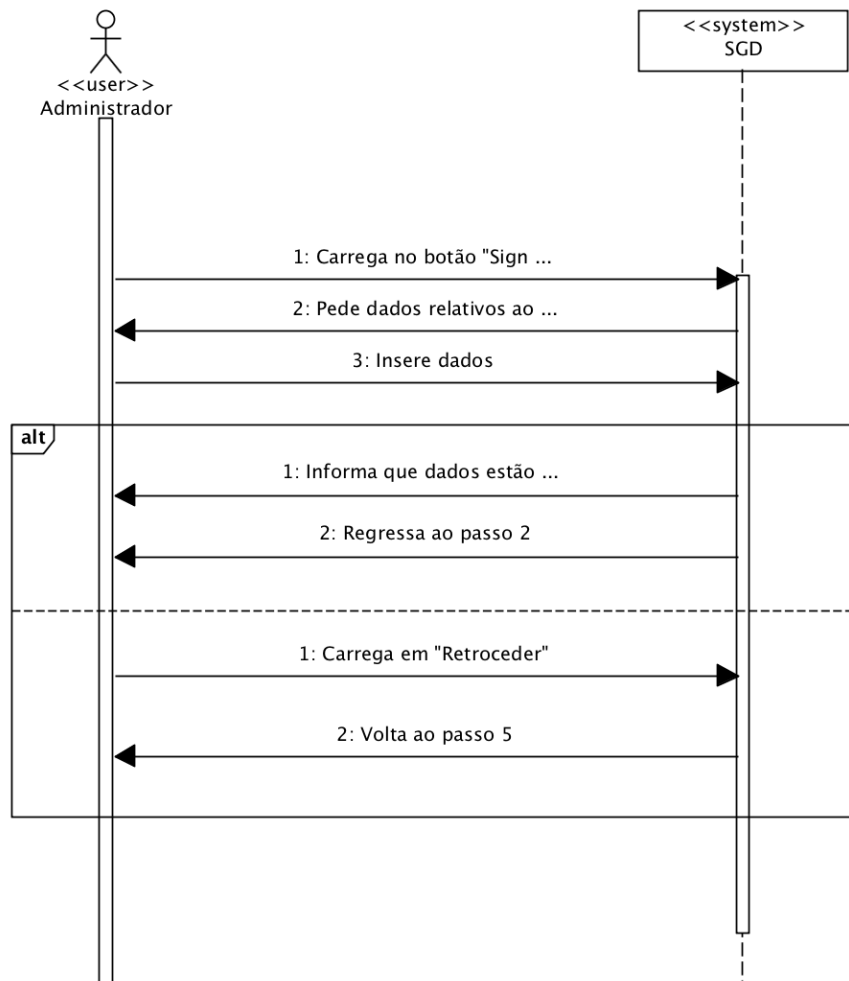


Figura 43. Diagrama de Sequência –Sign Up

5. Diagramas de Máquinas de Estado

As máquinas de estado representam os diferentes estados nos quais cada ator presente no sistema se encontra ao longo da utilização da aplicação.

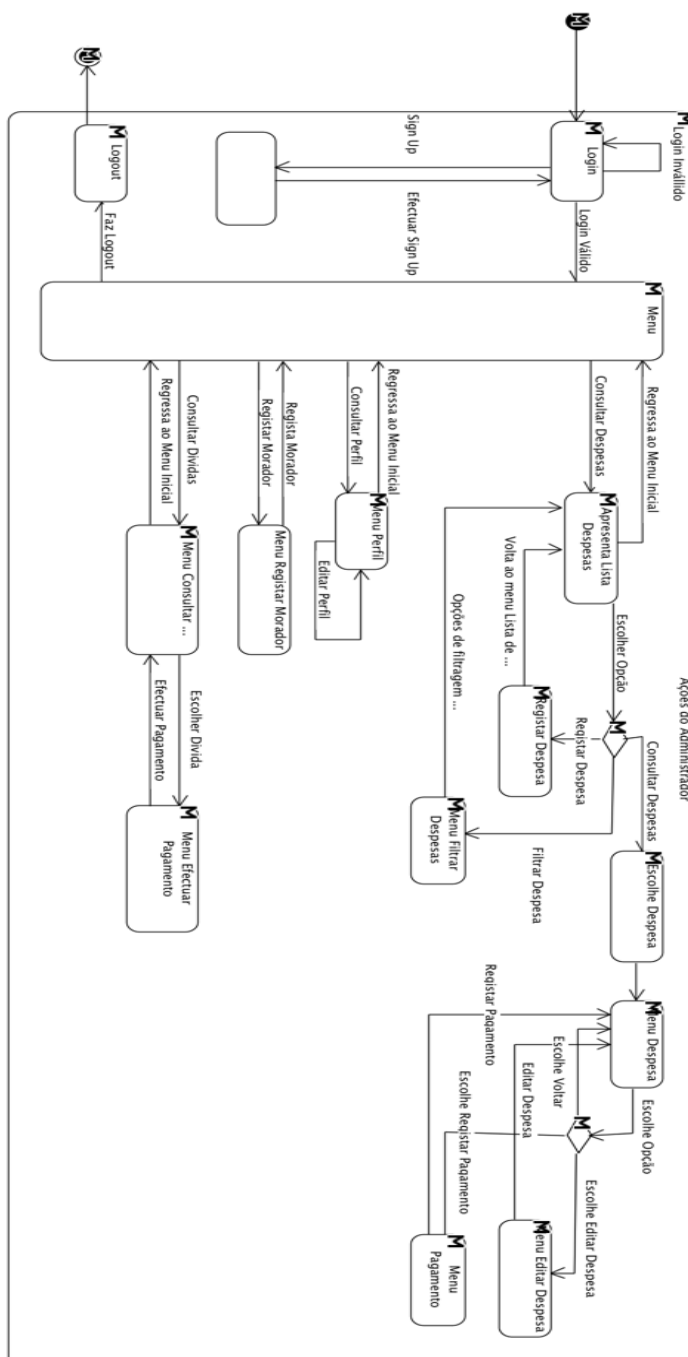


Figura 44. Máquina de Estado - Administrador

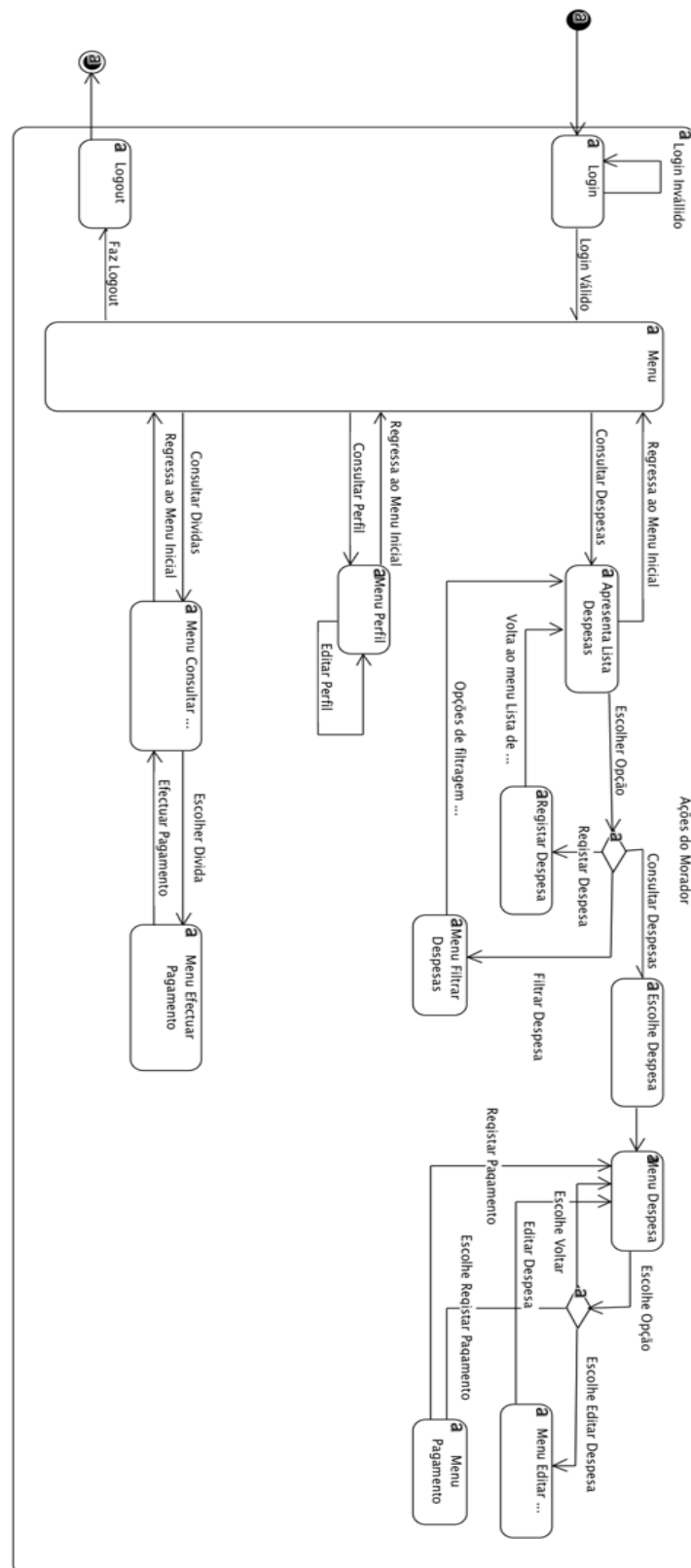


Figura 45. Máquina de Estado - Morador

6. Diagramas de Atividade

Os diagramas de atividade representam o fluxo de informações dentro de um objeto do sistema, tem como objetivo apresentar o fluxo do método, as suas exceções e alternativas. De modo que são uma boa forma de compreender as entidades envolvidas no decorrer de um método e as ações por estas executadas.

Em seguida apresentamos alguns dos diagramas relativos ao nosso sistema.

Apresentar Lista de Despesas

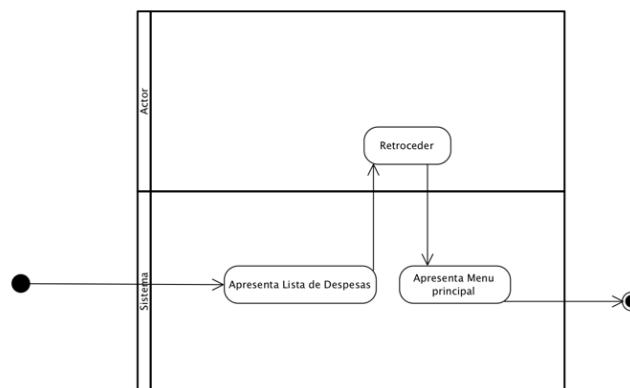


Figura 46. Diagrama de Atividade – Apresentar Lista Despesas

Efetuar Pagamento

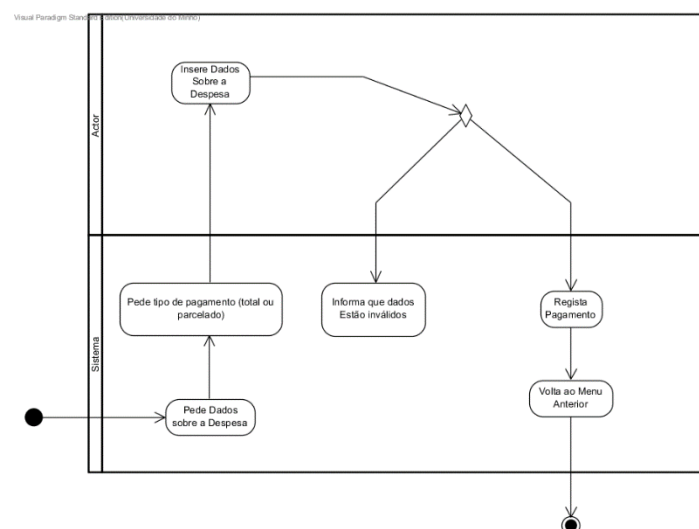


Figura 47. Diagrama de Atividade – Efetuar Pagamento

Registrar Despesa

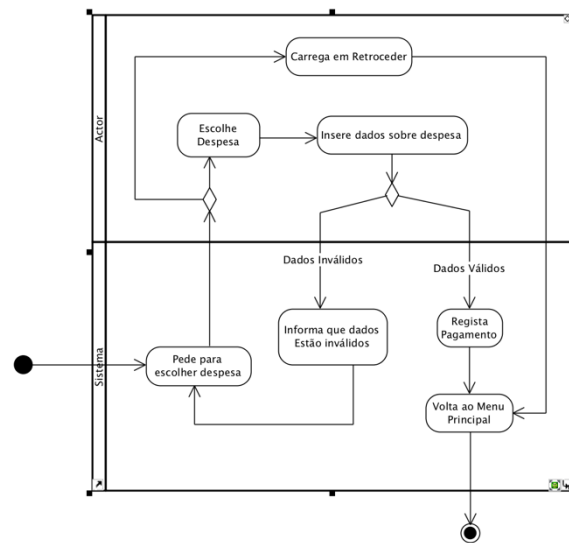


Figura 48. Diagrama de Atividade – Registrar Despesa

Remover Morador

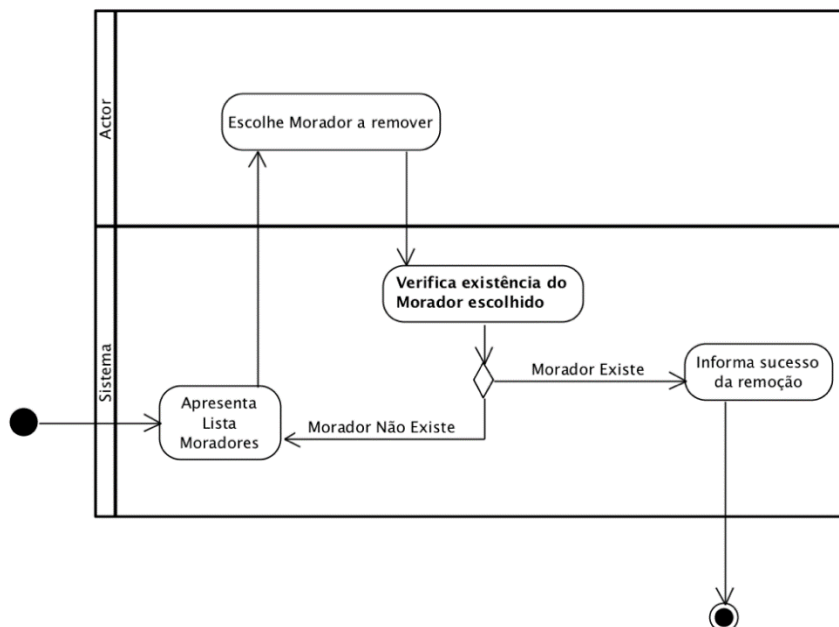


Figura 49. Diagrama de Atividade – Remover Morador

7. Diagramas de Classe

Recorrendo ao conceito de classes e relacionamentos o diagrama de classes representa a estrutura do sistema, identificando-se os objetos relevantes do sistema em estudo. Os objetos são algo que queremos registar e que tem uma identificação, um estado, e um comportamento.

As classes são um conjunto de objetos que partilham o mesmo meio de identificação, propriedades de estado, comportamento, relações e semântica.

Em qualquer sistema existem objetos que se relacionam entre si, sendo que existem vários tipos de relações, associação (agregação e composição), generalização e relação de dependência.

Este tipo de diagrama é importante pois define todas as classes que o sistema necessita e é a base para a construção dos diagramas de sequência e estado.

Para a sua construção deste diagrama partiu-se da base do modelo de domínio em que se identifica as classes e com um estudo detalhado das relações entre as mesmas foi possível chegar ao digrama apresentado de seguida, onde são representados todos os atributos e métodos.

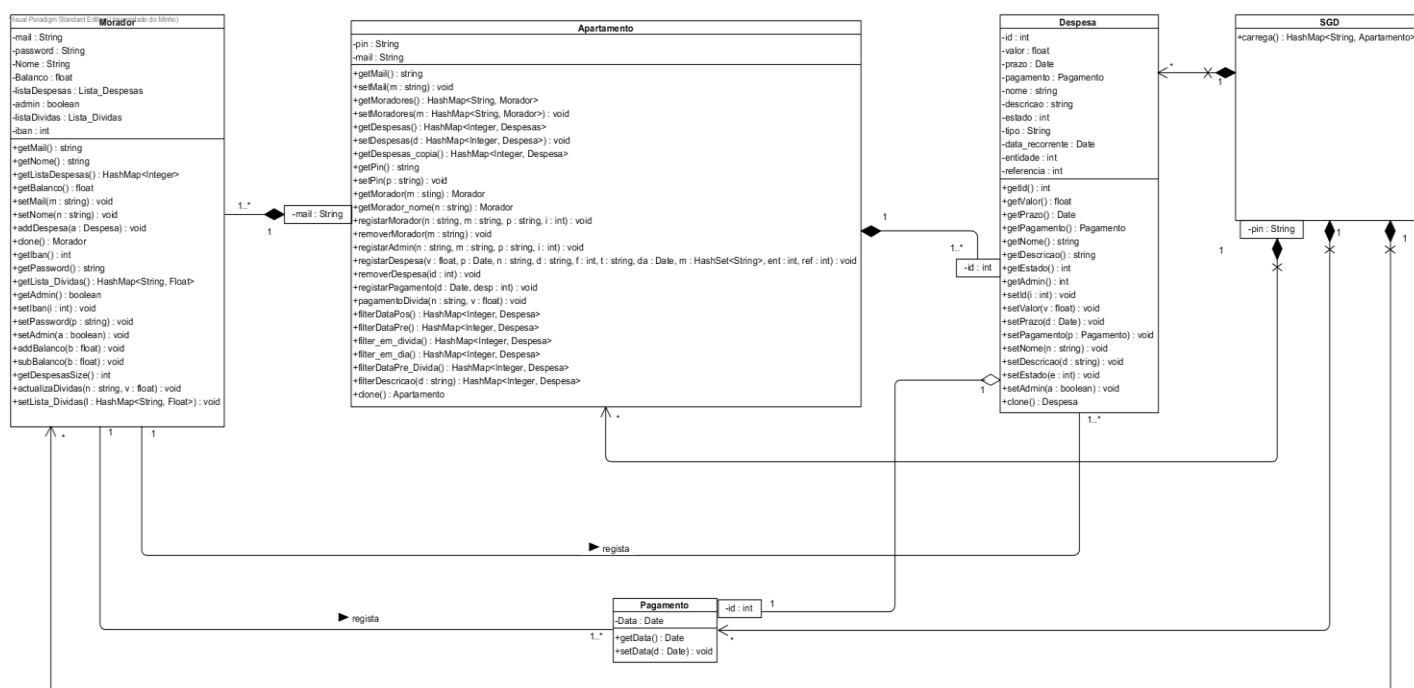


Figura 50. Diagrama de classes

Elaborada a primeira versão do diagrama de classes foi feita uma segunda onde se substitui os Maps que se relacionam com a facade por Data Access Objects) DAO's.

Os DAO's têm todos implementação semelhante por se resumirem a inserções, actualizações, remoções e consultas.

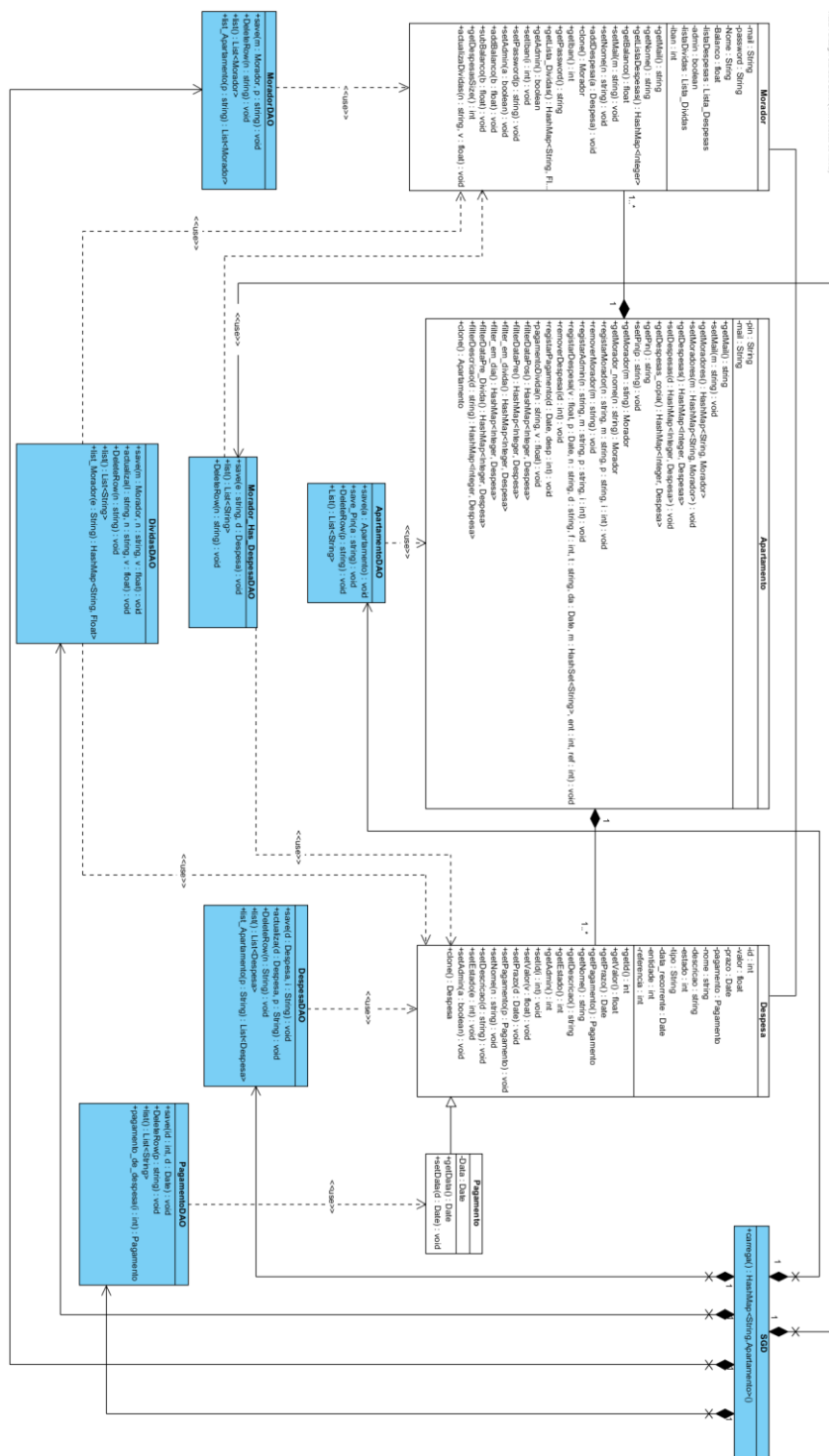


Figura 51. Diagrama de classes DAO

8. Diagramas de Package

Neste momento já temos implementadas a organização das classes no código, deste modo podemos apresentá-las segundo um diagrama de package que ilustra a maneira como estas estão dispostas e interagem entre si.

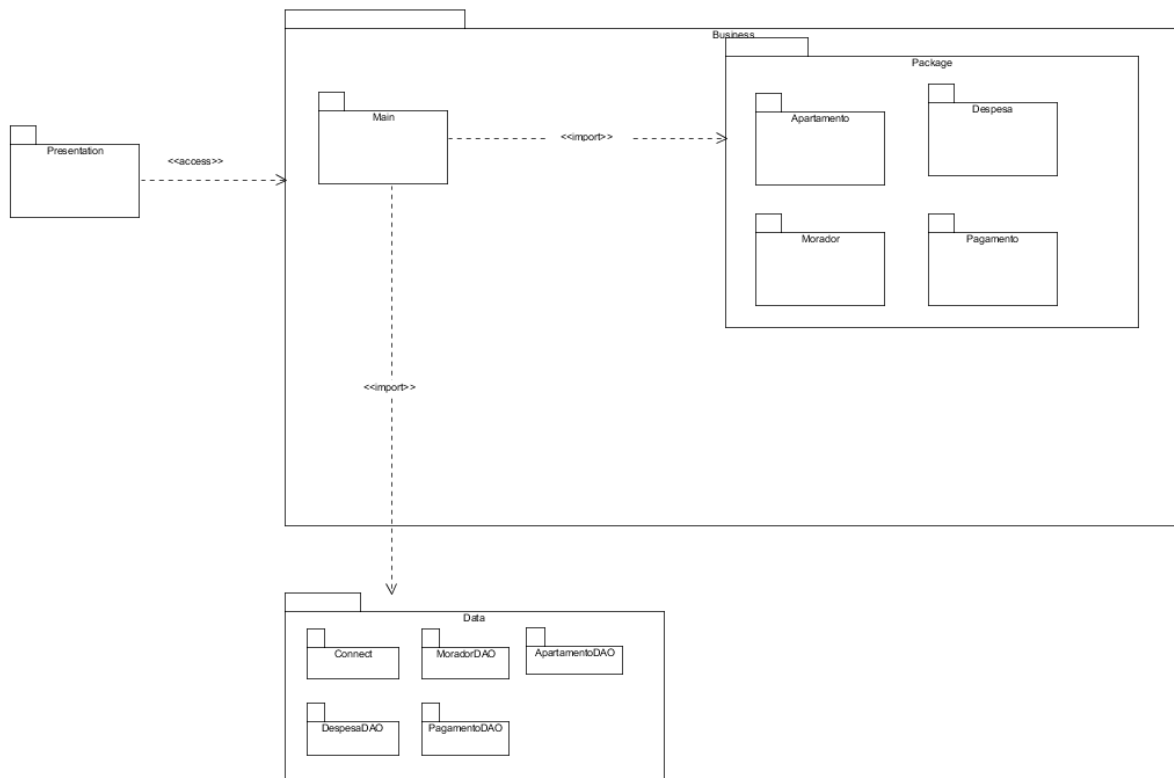


Figura 52. Diagrama de Package

9. Diagramas de Sequência com Subsistemas

Consultar Despesa

Para visualizar os detalhes de uma despesa específica, é necessário consultar todas as despesas e depois de escolhida, através de filtragem ou não, conseguir entrar nessa mesma despesa. Para isto, no que diz respeito a código, o ator atravessa as classes Main, para obter o apartamento a que pertence, Apartamento, para obter todas as despesas relativas a esse apartamento e Despesa, para obter a despesa desejada.

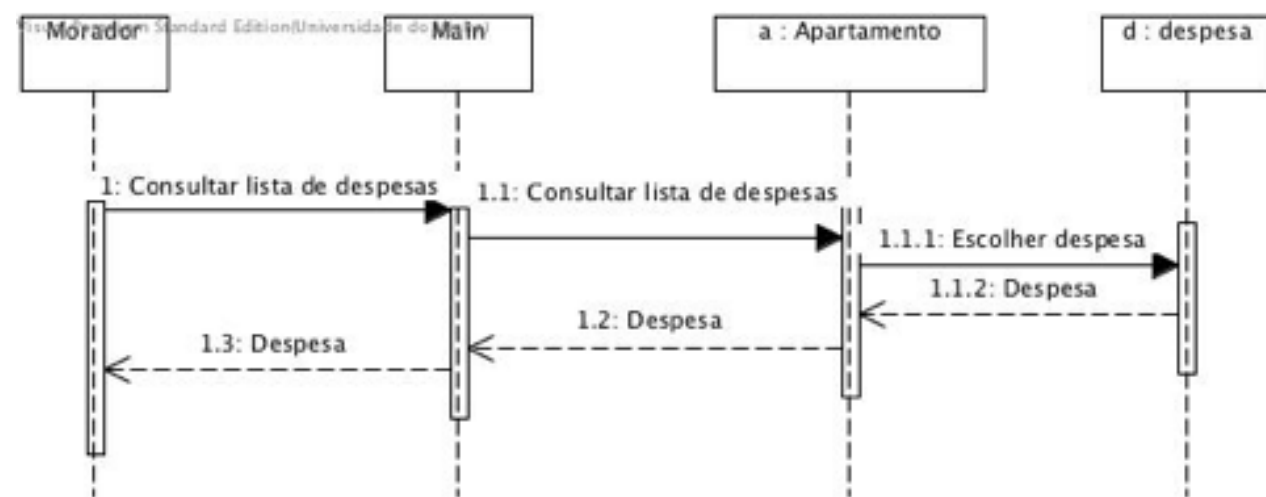


Figura 53. Consultar Despesa

Consultar Lista Despesas

Para consultar a lista de todas as despesas presentes, como está implícito no diagrama acima, é necessário apenas consultar a lista de todas as despesas, no menu principal. Para tal, o ator atravessa a classe Main que o relaciona com o seu apartamento e a classe Apartamento que possui a lista que este deseja.

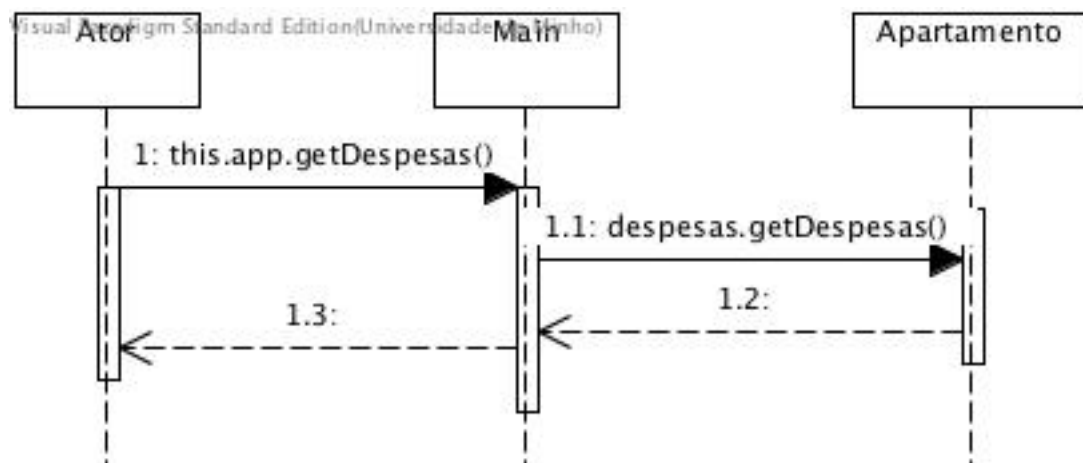


Figura 54. Consultar Lista Despesas

Editar Dados Morador

Para editar os dados, uma vez presente no menu principal, basta seleccionar a opção “Editar dados”. No que diz respeito á arquitetura deste método, é preciso obter todos os moradores do apartamento e relacionar com o que está logado. Uma vez que essa associação está feita, altera-se os dados desse mesmo morador, da maneira que ele desejar ou cancela. Para isso, atravessa portanto as classes Main, Apartamento e Morador.

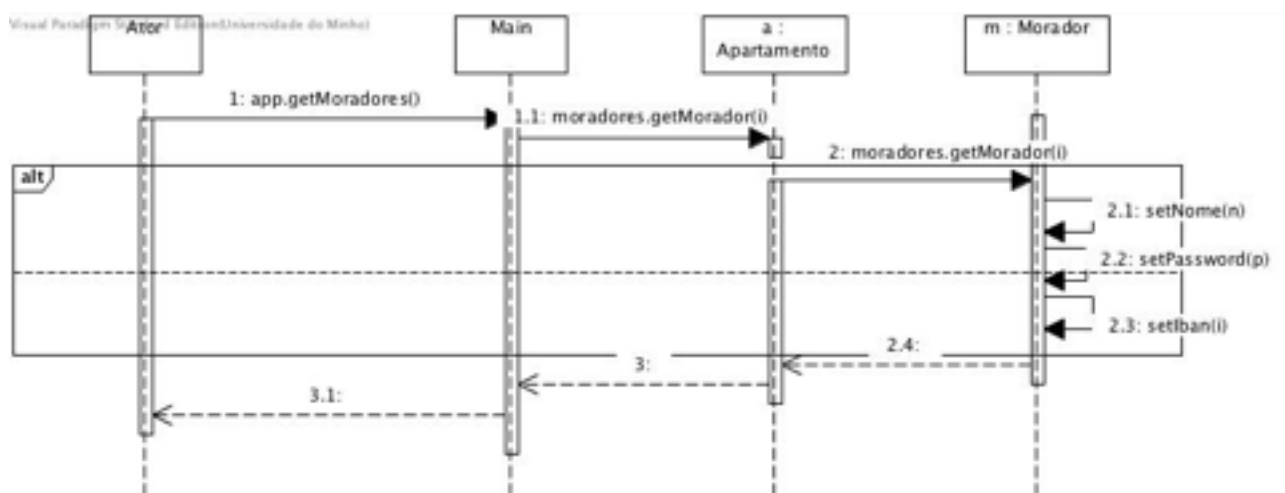


Figura 55. Editar Dados Morador

Login

Para iniciar sessão, o ator está a comunicar com a classe Main e a obter todos os moradores. De seguida, apenas obtém o seu objeto Morador, que é o que lhe interessa, e vai confirmar se os seus dados (mail, password e pin) estão corretos. Só inicia sessão se todos estes se confirmarem.

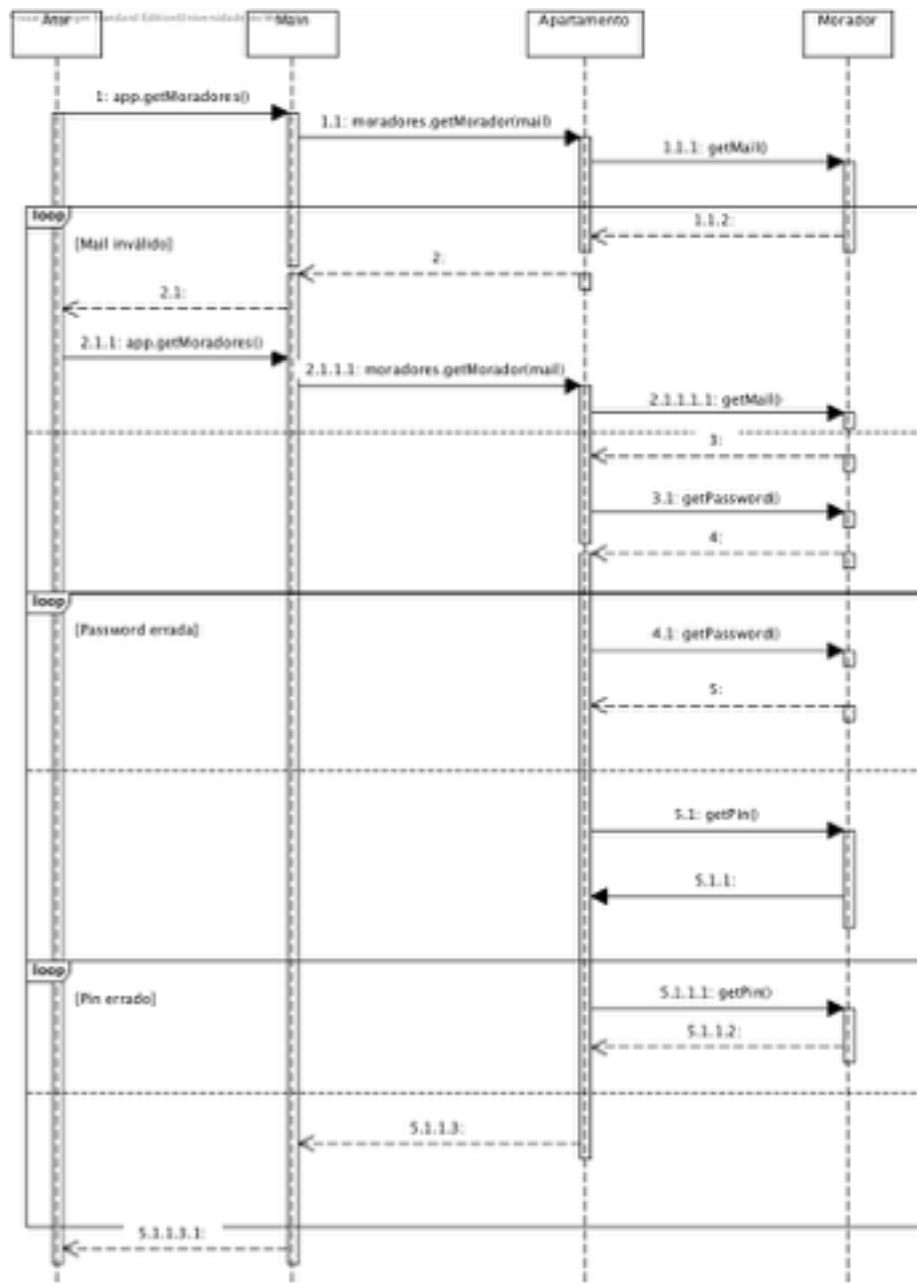


Figura 56. Login

Logout

Para fechar sessão, o ator apenas precisa de confirmar que deseja sair e deixa de ter sessão iniciada.

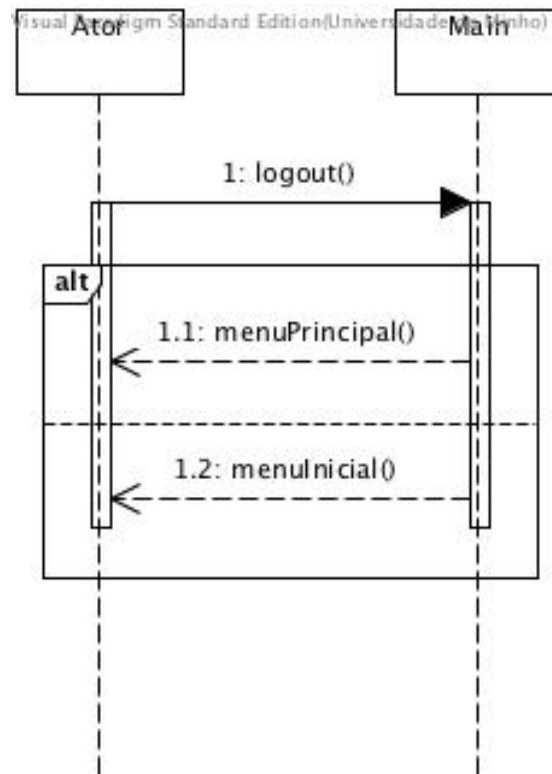


Figura 57. Logout

Registrar Morador

Para registar o morador, método que só o admin tem permissão, uma vez que confirma que a instância “admin” tem o valor *true*, atravessa as classes Main e Apartamento, e é no Apartamento que ele regista o morador de forma a estar associado a este. Se os dados forem válidos, ele cria um novo objeto Morador e associa-o ao HashMap de morador presente no Apartamento.

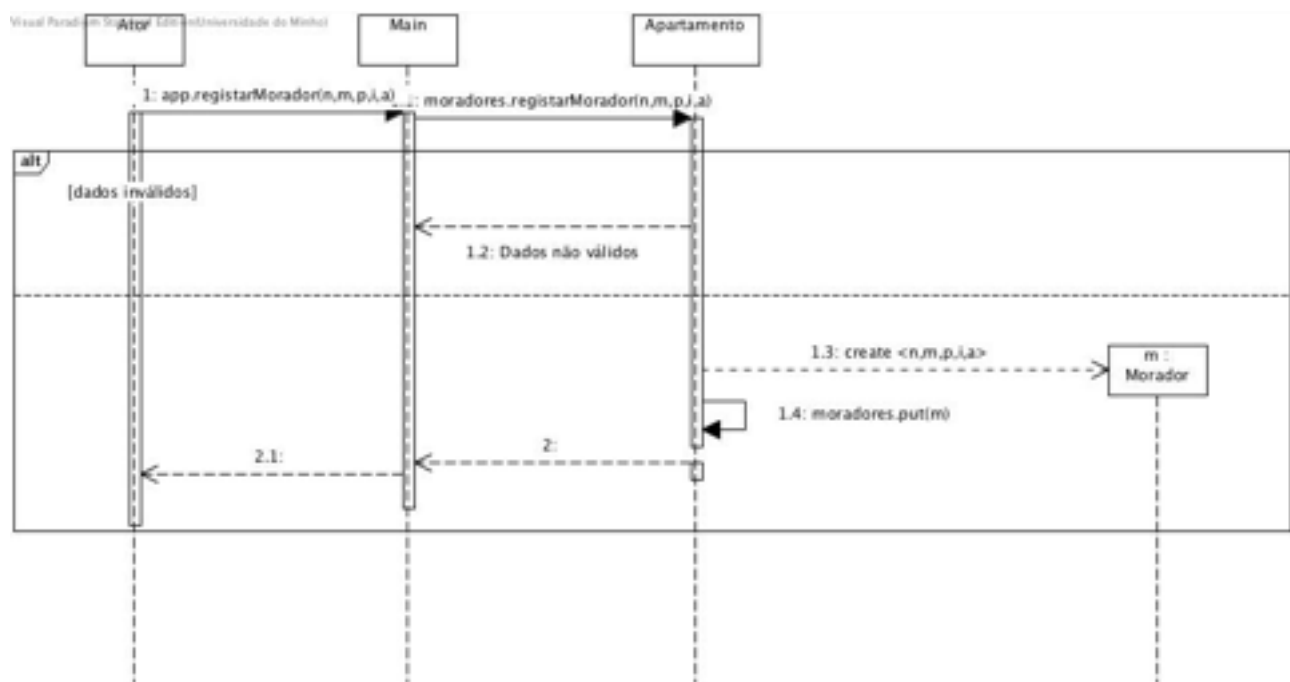
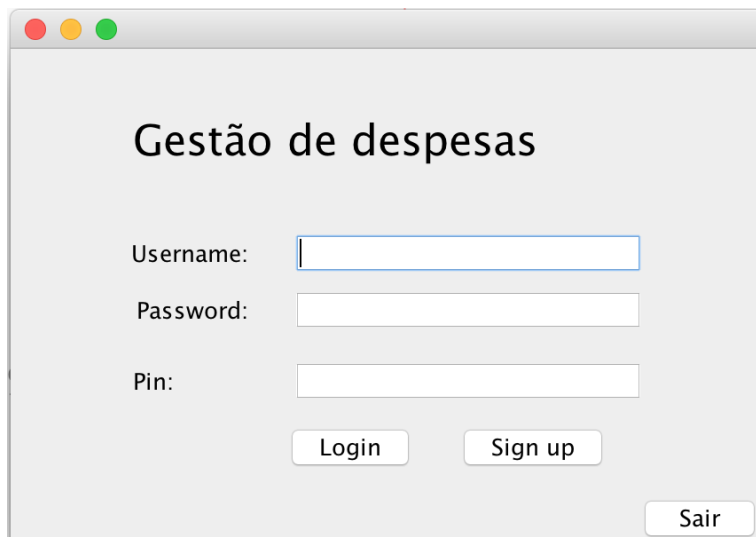


Figura 58. Registrar Morador

10. Interface

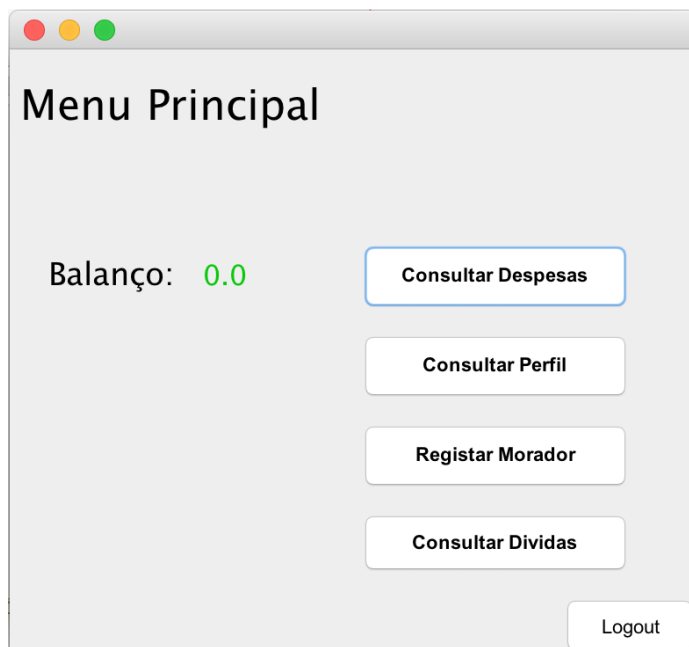
De modo a fazer uma aproximação ao utilizador final desenvolvemos a interface do sistema, para isto foi utilizada a ferramenta do java SWING tal como proposto nas aulas práticas.

A interface procura cobrir todas as funcionalidades propostas na modelação anterior de uma forma agradável e intuitiva para utilizador.



The image shows a Java Swing window titled "Gestão de despesas". It has a light gray background and a standard macOS-style title bar with red, yellow, and green buttons. The window contains three text input fields labeled "Username:", "Password:", and "Pin:". Below the "Pin:" field are two buttons: "Login" and "Sign up". In the bottom right corner, there is a button labeled "Sair".

Figura 59. Login



The image shows a Java Swing window titled "Menu Principal". It has a light gray background and a standard macOS-style title bar with red, yellow, and green buttons. The window displays "Balanço: 0.0" in green text. To the right of the balance are four buttons stacked vertically: "Consultar Despesas", "Consultar Perfil", "Registrar Morador", and "Consultar Dividas". In the bottom right corner, there is a button labeled "Logout".

Figura 60. Menu Principal

Registrar Despesa

Nome: MEO - Dezembro

Descrição: Fatura de dezembro da MEO (telefone, telemovel, tv)

Valor: 40

Data Limite: 17/12/2016

Tipo: Normal

Data Recorrência: 17/01/2017

Dados de pagamento: Entidade: 12312 Referência: 129129391

Confirmar Voltar

dd/mm/aaaa

Figura 61. Registrar Despesa

Pagamento

Moradores: creissac@mail.com

Data:

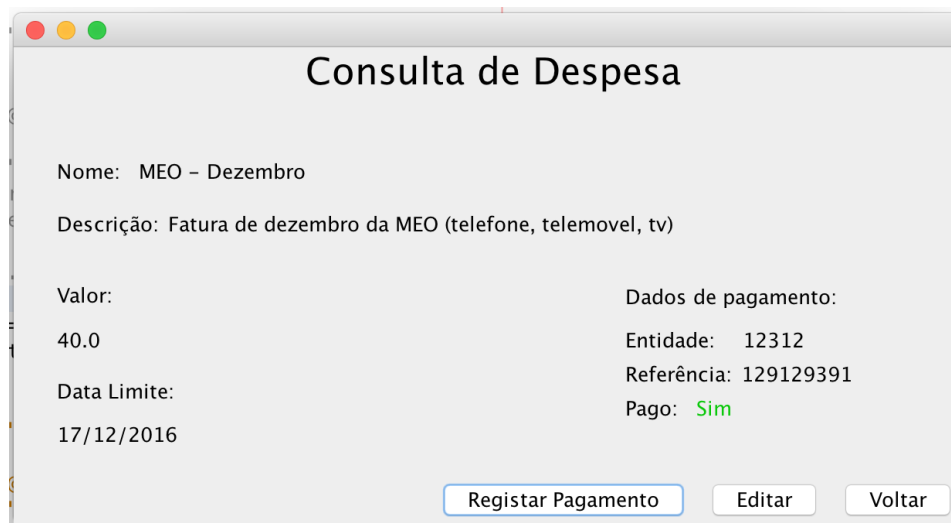
Limpar

Pagamento Individual

Pagamento Colectivo

Voltar

Figura 62. Registrar Pagamento



Consulta de Despesa

Nome: MEO - Dezembro

Descrição: Fatura de dezembro da MEO (telefone, telemovel, tv)

Valor: 40.0

Data Limite: 17/12/2016

Dados de pagamento:

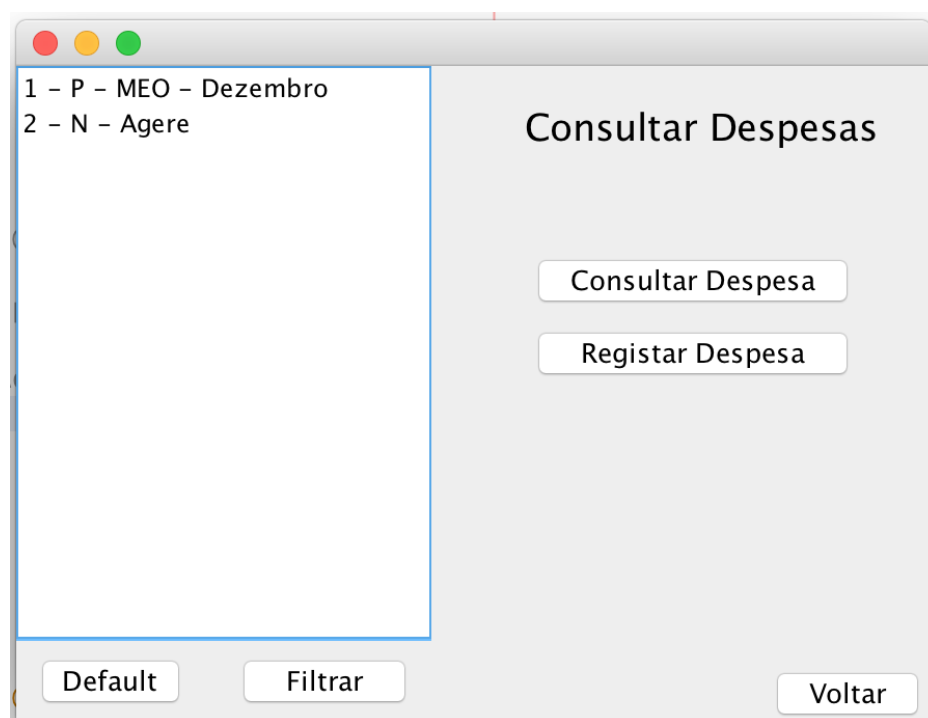
Entidade: 12312

Referência: 129129391

Pago: Sim

Registrar Pagamento Editar Voltar

Figura 63. Consulta Despesa



Consultar Despesas

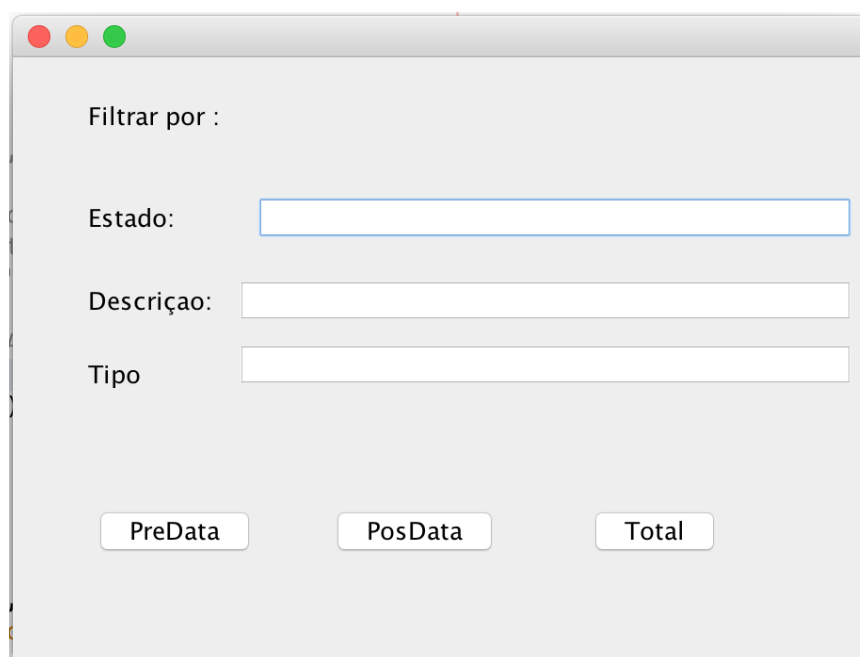
1 - P - MEO - Dezembro
2 - N - Agere

Consultar Despesa

Registrar Despesa

Default Filtrar Voltar

Figura 64. Consultar Despesas



Filtrar por :

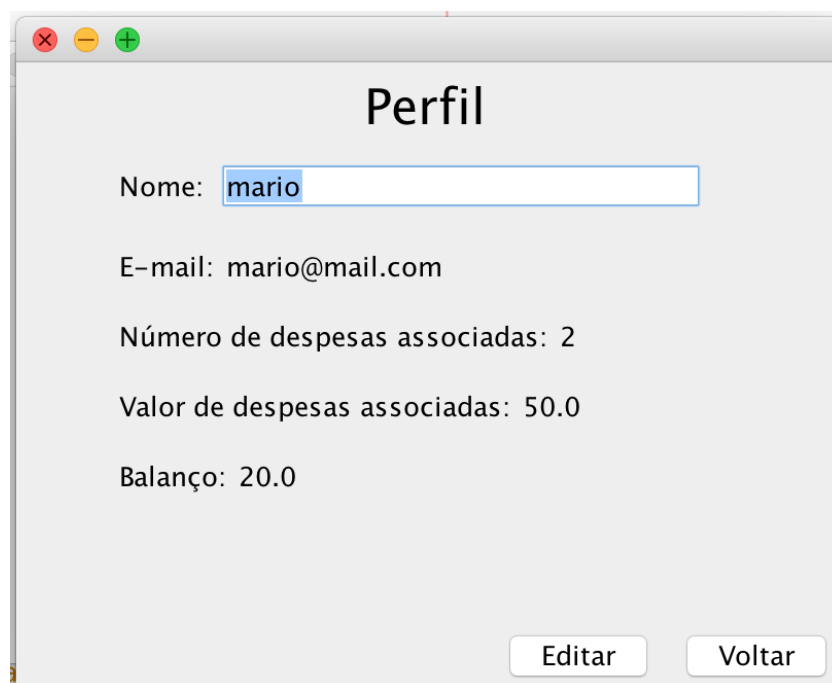
Estado:

Descrição:

Tipo:

PreData PosData Total

Figura 65. Filtrar Despesas



Perfil

Nome:

E-mail: mario@mail.com

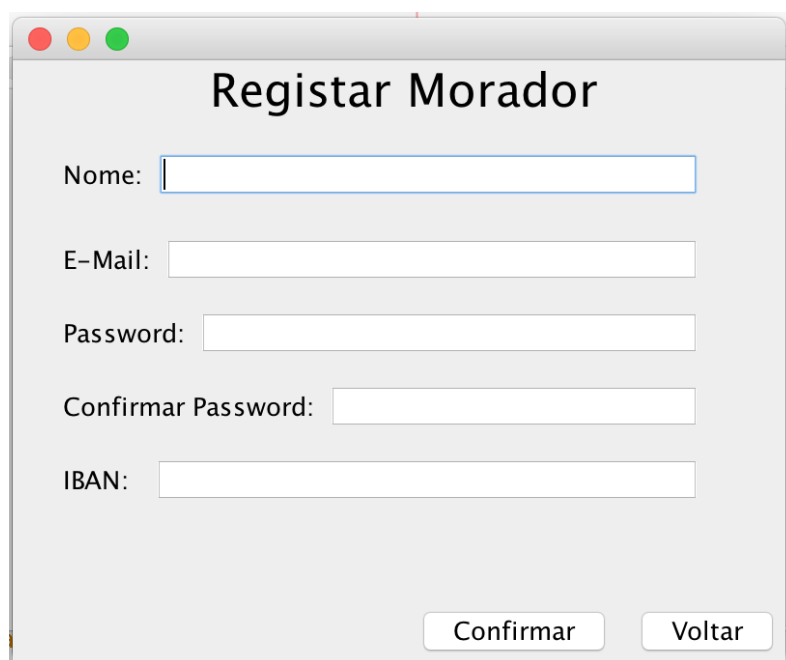
Número de despesas associadas: 2

Valor de despesas associadas: 50.0

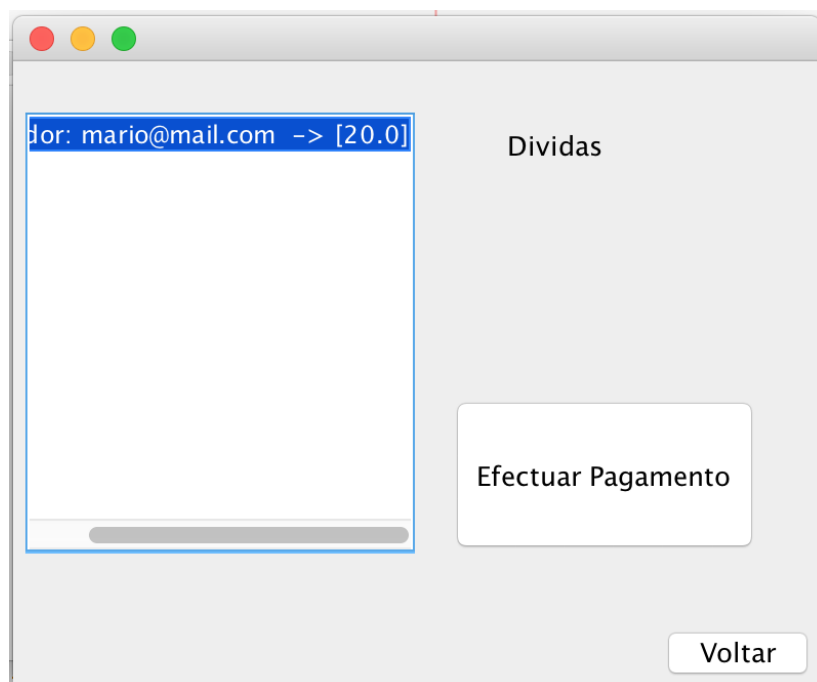
Balanço: 20.0

Editar Voltar

Figura 66. Perfil

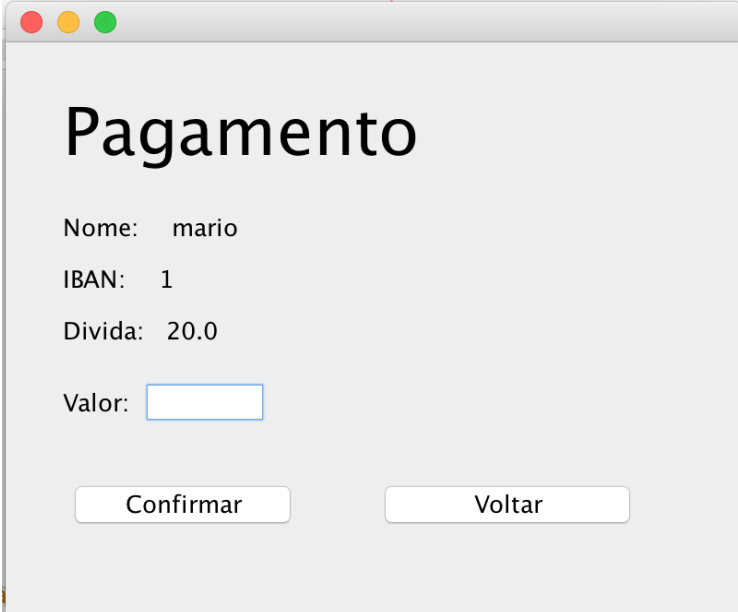


A screenshot of a web application window titled "Registrar Morador". The window has a light gray background and standard macOS window controls (red, yellow, green buttons) at the top left. The form contains five input fields: "Nome:", "E-Mail:", "Password:", "Confirmar Password:", and "IBAN:". At the bottom right, there are two buttons: "Confirmar" and "Voltar".

Figura 67. Registrar Morador

A screenshot of a web application window titled "Consultar Dividas". The window has a light gray background and standard macOS window controls at the top left. On the left side, there is a text input field containing "Morador: mario@mail.com" and a dropdown menu showing "-> [20.0]". Below this is a large empty rectangular area. On the right side, the title "Dividas" is displayed. Below it is a button labeled "Efectuar Pagamento". At the bottom right, there is a button labeled "Voltar".

Figura 68. Consultar Dividas



Pagamento

Nome: mario

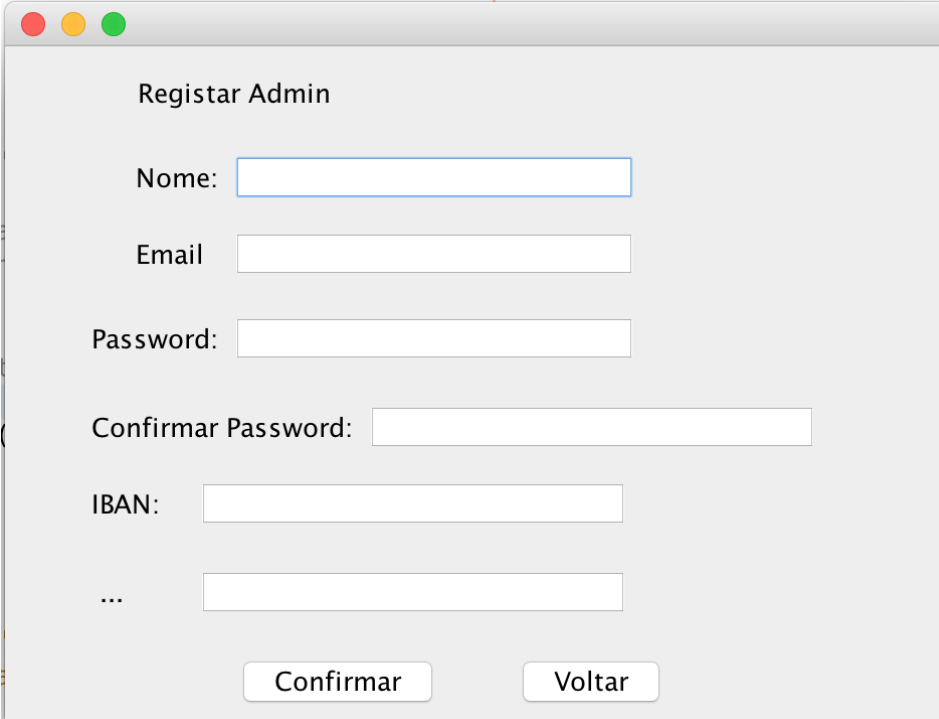
IBAN: 1

Divida: 20.0

Valor:

Confirmar Voltar

Figura 69. Pagar Divida



Registrar Admin

Nome:

Email

Password:

Confirmar Password:

IBAN:

...

Confirmar Voltar

Figura 70. Sign Up

11. Implementação

Relativamente à camada business seguiu-se à implementação descritos em cima, todas as uses cases estão implementadas, foram criadas em termos de código as classes:

- Despesa
- Morador
- Pagamento
- Apartamento
- Main

Houve uma preocupação em termos de código de o simplificar e por isso mesmo por exemplo o utilizador anda pode efetuar pagamento que uma despesa de diferentes maneiras, mas ao registar a despesa mete os dados necessários para qualquer tipo de pagamento. Outra implementação que foi feita de forma a simplificar foi na classe Morador haver apenas uma flag que nos diz se é administrador ou não sendo depois utilizada para saber se existe permissão ou não para certas tarefas.

Teve-se especial atenção em manter o encapsulamento e a proteger as estruturas utilizadas como exemplo criamos a classe apartamento que tem como estrutura:

- private String pin;
- private HashMap<String, Morador> moradores;
- private HashMap<Integer, Despesa> despesas;

Ou seja, ao ser utilizada como variável na main não há indicação nenhuma da estratégia que está a ser utilizada não podendo ser assim tao facilmente explorada alguma técnica contra o funcionamento do programa. Existe a variável pin que foi uma ideia que surgiu durante o desenvolvimento de código que a aplicação poderia suportar vários apartamentos ao mesmo tempo e como seria alguém que não iria influenciar de grande maneira todo o desenvolvimento anterior optou se por adotá-la , assim sendo, cada pin é criado quando um utilizador se regista como administrador e tem que ser único em toda a base dados sendo depois utilizado como uma credencial para fazer login em determinado apartamento em que seja morador.

12. Base de Dados

Para o uso corrente da aplicação em desenvolvimento houve a necessidade de implementar um sistema que guardasse todos os dados que seriam introduzidos e executados nesta própria. Para isso recorreu-se à ferramenta MySQL Workbench para modelar a base de dados.

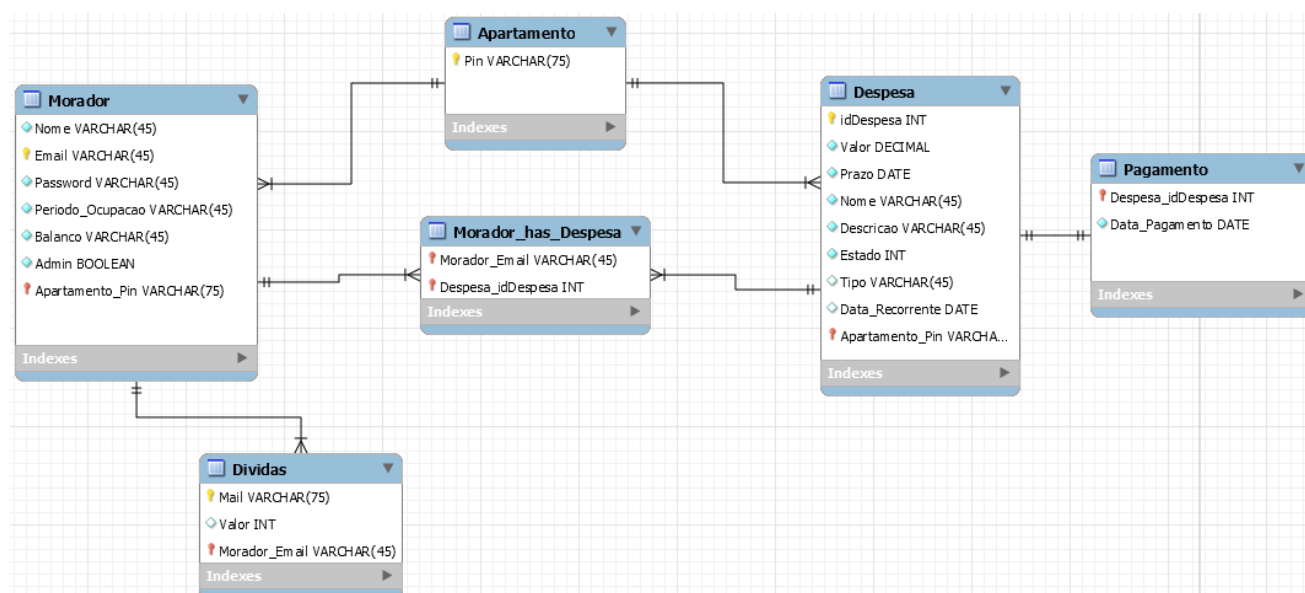


Figura 71. Modelo Lógico da Aplicação de Gestão de Despesas

Como podemos observar temos o apartamento em que cada um tem o seu pin único que é a sua chave primária, cada apartamento tem de 1..N moradores e de 0..N despesas havendo assim a propagação natural para duas tabelas distintas com relação de 1 para N. Cada morador tem de 0..N despesas e cada despesa pode envolver de 1..N moradores havendo assim uma relação de N..N, existindo assim obrigatoriamente uma tabela no meio denominada Morador_has_Despesa com chaves estrangeiras que relacionam as duas. Cada despesa tem apenas um pagamento correspondente sendo uma relação de 1 para 1. O morador tem uma lista de dívidas para cada um dos outros moradores do mesmo apartamento havendo assim uma sexta tabela de nome Dividas que corresponde a esta lista.

Foram de seguida implementadas as classes DAO na aplicação sendo a camada de Data, cada uma destas classes tem como principais métodos o guardar, remover, listar entre outros.

As classes da camada de Business que comunica com a camada de Data são a Main que procede ao carregamento dos dados que já se encontram na base de dados quando esta é iniciada e a classe Apartamento que sempre que existe o registo de algum objeto que corresponde a uma destas tabelas é inserido e guardado.

13. Conclusões e Trabalho Futuro

Este trabalho permitiu analisar um sistema real de gestão de despesas num apartamento entre os diversos moradores e simulá-lo em ambiente informático.

O sistema de gestão de despesas acaba por ser um tema interessante devido ao facto de todos os elementos do grupo partilharem apartamentos com outros colegas de casa estudantes.

Sendo que a adoção deste sistema é bastante simples, os serviços oferecidos pelo sistema são bastante úteis no contexto real da estadia de um aluno universitário, dado que por diversas vezes existem conflitos no pagamento das dividas entre colegas de casa devido a mal-entendidos que não existiriam com o uso deste sistema.

A aplicação não acarreta um elevado volume de dados sendo que foi feita num contexto académico, podendo mais tarde ser escalada para o contexto maior.

Após a análise e modelação do sistema descrito no enunciado tornou-se possível fazer uma descrição suficiente pormenorizada do sistema a ser implementado. O recurso a UML foi extremamente fulcral no desenvolver deste trabalho para a descrição e elaboração de todas as suas funcionalidades através de diagramas que tornam a sua exposição muito mais pormenorizada e a sua compreensão mais fácil e direta.

Depois da análise de requisitos e modelação completa do sistema, foi aplicado todo esse trabalho na implementação da Aplicação. Foi desenvolvida na linguagem Java segundo o Paradigma Orientado a Objetos, respeitando todos os seus princípios nomeadamente o encapsulamento de dados, foi também usada a linguagem SQL na parte da Base de Dados do sistema. Nesta fase a metodologia aplicada na elaboração dos diagramas e modelos, proporcionaram uma enorme facilidade no processo de desenvolvimento da aplicação devido a estarem todas as funcionalidades e objetivos previamente definidos e organizados.

De forma geral os objetivos traçados tanto pelo grupo foram atingidos com sucesso. Todas as funcionalidades requeridas pelo enunciado foram implementadas e coberta toda a documentação necessária para suportar as decisões implementadas.

Na perspetiva da utilização podemos dizer que esta foi desenvolvida tendo em conta a experiência do utilizador garantindo uma usabilidade intuitiva e agradável. A ferramenta utilizada para desenvolver a parte de UI foi *JavaSwing* dado ter sido a que na opinião dos elementos do grupo seria a que melhor se adaptava às necessidades de implementação da interface do utilizador.

Apesar de todos os objetivos terem sido atingidos com sucesso durante o percurso de desenvolvimento deparamos-nos com diversas dificuldades nomeadamente a implementação da base de dados em JDBC, dado ser um tema que não foi abordado nas aulas, esta dificuldade foi superada através da palestra organizada sobre o tema pelo CESIUM.

Em suma, os objetivos foram atingidos como esperávamos e as nossas capacidades nas diversas componentes abordadas foram melhoradas ao longo da elaboração, tendo servido obviamente como um ótimo elemento de estudo.