

Universidade do Minho – 2016/2017

Mestrado Integrado em Engenharia Informática

## Desenvolvimento de Sistemas de Software



Grupo 6:

Bruno Pereira, 75135

Diogo Silva , 76407

Maria Ana de Brito, 73580

## Introdução

Este projecto visa explorar a iniciativa UML a partir do desenvolvimento de um sistema de software seguindo a metodologia adequada no mercado de trabalho.

Deste modo, como meio de modelação e planeamento de software abordamos várias formas de modelar, sendo elas os Use Cases, Diagramas de Sequência, Diagramas de Sequência de Subsistemas, Diagrama de Use Cases, Máquinas de Estado, Diagramas de Packages e Diagrama de Instalação.

Para criar a interface gráfica fez-se mockups das várias janelas, refinando-as na sua implementação, através de Java Swing.

## O Sistema de Software: Gestão de Apartamento

A primeira fase deste projecto consistiu no desenvolvimento de um modelo de domínio: a sua finalidade é apresentar um esboço inicial do que futuramente seria o software implementado. De forma a modelar o comportamento do programa, em seguida foi esboçado o modelo de Use Cases que já apresentou dois atores, o inquilino (utilizador normal) e o administrador. Para simular a interação do utilizador com o programa, de seguida, foram especificados Use Cases discriminando o tipo de actor correspondente.

Assim, foi possível obter uma visão inicial do que será necessário desenvolver na interface gráfica. Por fim, apresentamos os Diagramas de Sequência iniciais respetivos.

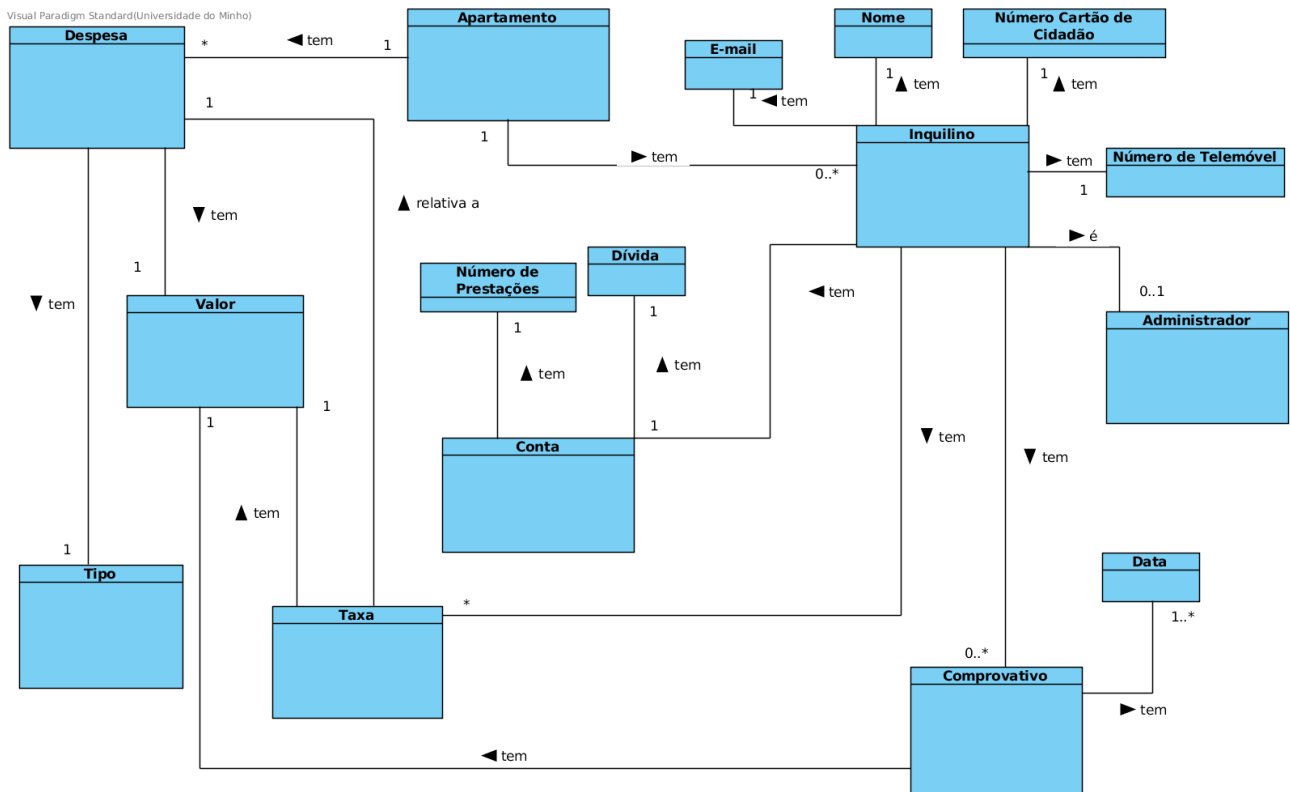
A segunda e fase final do projecto consistiu na implementação do software e na criação de novos diagramas auxiliares à compreensão do seu funcionamento.

Começamos por desenvolver os Diagramas de Sequência com Subsistemas a partir dos diagramas especificados anteriormente. Em seguida, focando-nos finalmente no código do programa em si, desenvolvemos o diagrama de classes que seria reformulado futuramente para suportar Data Access Objects. Não descoramos a formulação de mockups da interface gráfica iniciais, visto que permitiram antever pontos que se verificariam necessários aquando a implementação do software.

Após a formulação de um código inicial seguindo o diagrama de classes, partimos para a interface gráfica e, assim, fomos aperfeiçoando o nosso programa.

Por fim, acompanhado do desenvolvimento do Diagrama de Instalação, especificamos o Diagrama de Pacotes e terminamos com alguns exemplos das Máquinas de Estado da interface gráfica que consideramos relevantes do Gestão de Apartamento.

# Modelo de Domínio

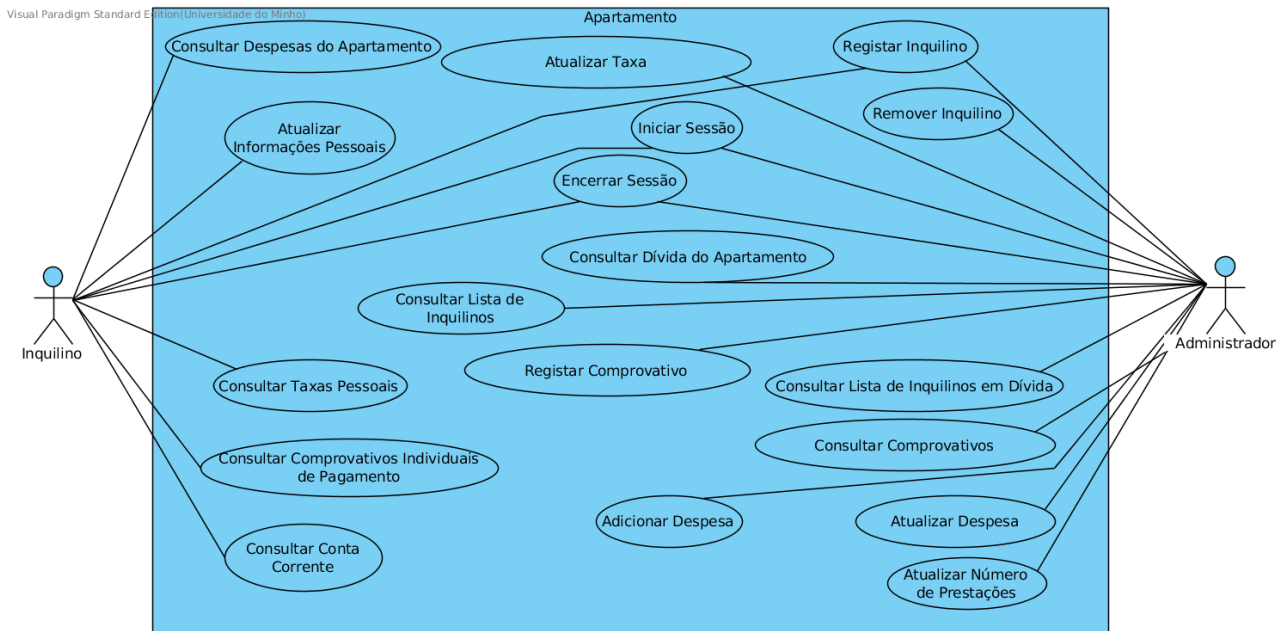


Após o grupo repensar nas decisões tomadas anteriormente, foram feitas alterações ao modelo de domínio original.

Assim, decidimos assumir que a despesa recorrente é um caso particular de uma despesa genérica, assim como o conjunto de despesas extraordinárias. Logo, é o administrador que define quais são as despesas do apartamento. Por conseguinte, decidimos não discriminar os vários tipos de despesas no modelo, pois tornava-se supérfluo.

Além disso o tipo de pagamento tornou-se mais simples, apenas fazemos menção do número de prestações que o inquilino tem de pagar (se for um, então tem de pagar a totalidade da dívida de uma só vez, caso seja superior, tem de pagar as várias prestações da dívida, correspondentes ao número indicado).

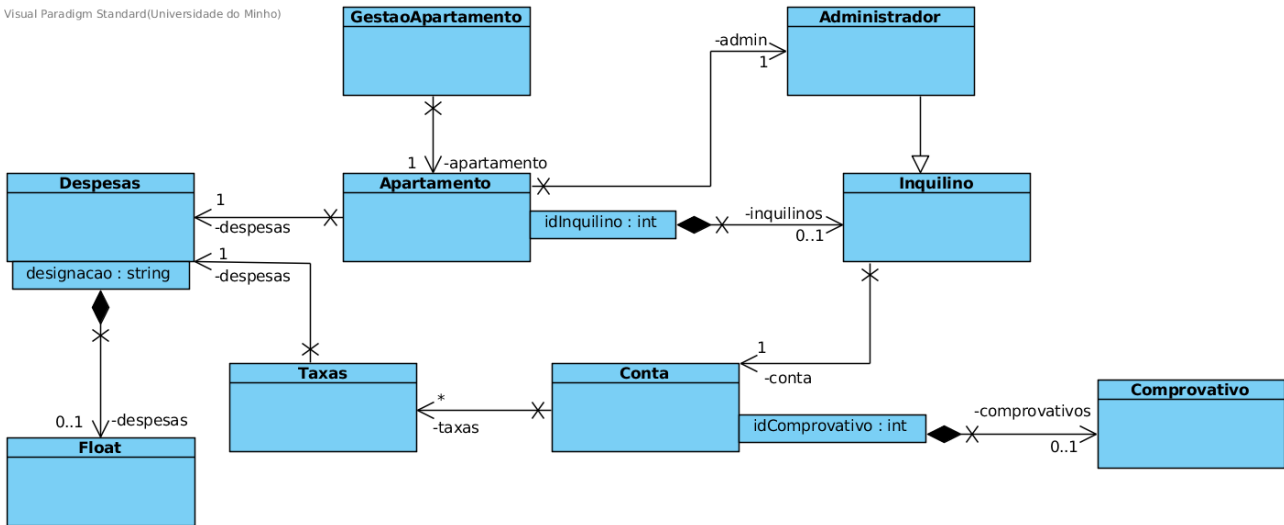
# Diagrama de Use Cases



Comparativamente ao diagrama de Use Cases entregue na fase anterior, foram efetuadas algumas mudanças, nomeadamente, ao inquilino foram retirados alguns Use Cases, que, por sua vez, foram entregues ao administrador (como, por exemplo, alterar a taxa extraordinária).

Denote-se que as mudanças não foram drásticas e surgiram do desenvolvimento natural e iterativo do projeto.

Visual Paradigm Standard(Universidade do Minho)

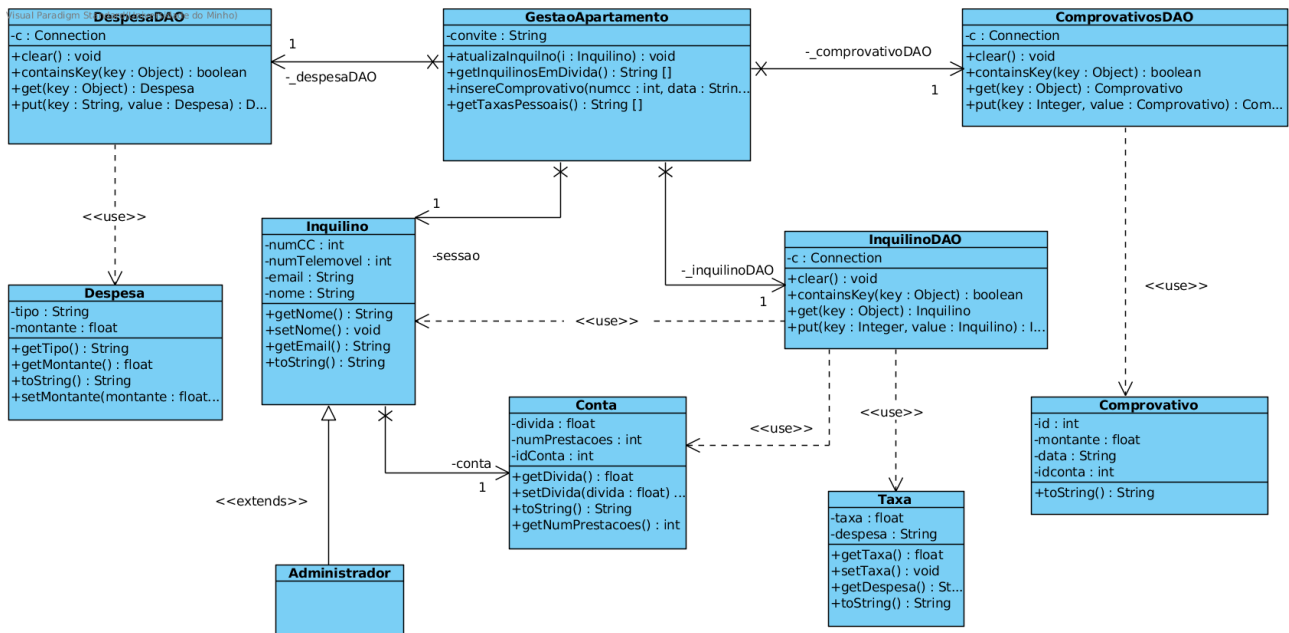


O diagrama de classe foi concebido através da análise dos Use Cases especificados e dos requisitos definidos para o sistema.

Assim, o sistema apenas possui um apartamento, que, por sua vez, contém despesas, um administrador (neste diagrama inicial) e vários inquilinos. As despesas são armazenadas num Map, em que a chave é o tipo da despesa (recorrente, ginásio, piscina, ...) e o valor é o montante da despesa. O inquilino, que, por sua vez, pode ser um administrador, contém uma conta que possui um Map de comprovativos e várias taxas que são relativas às despesas.

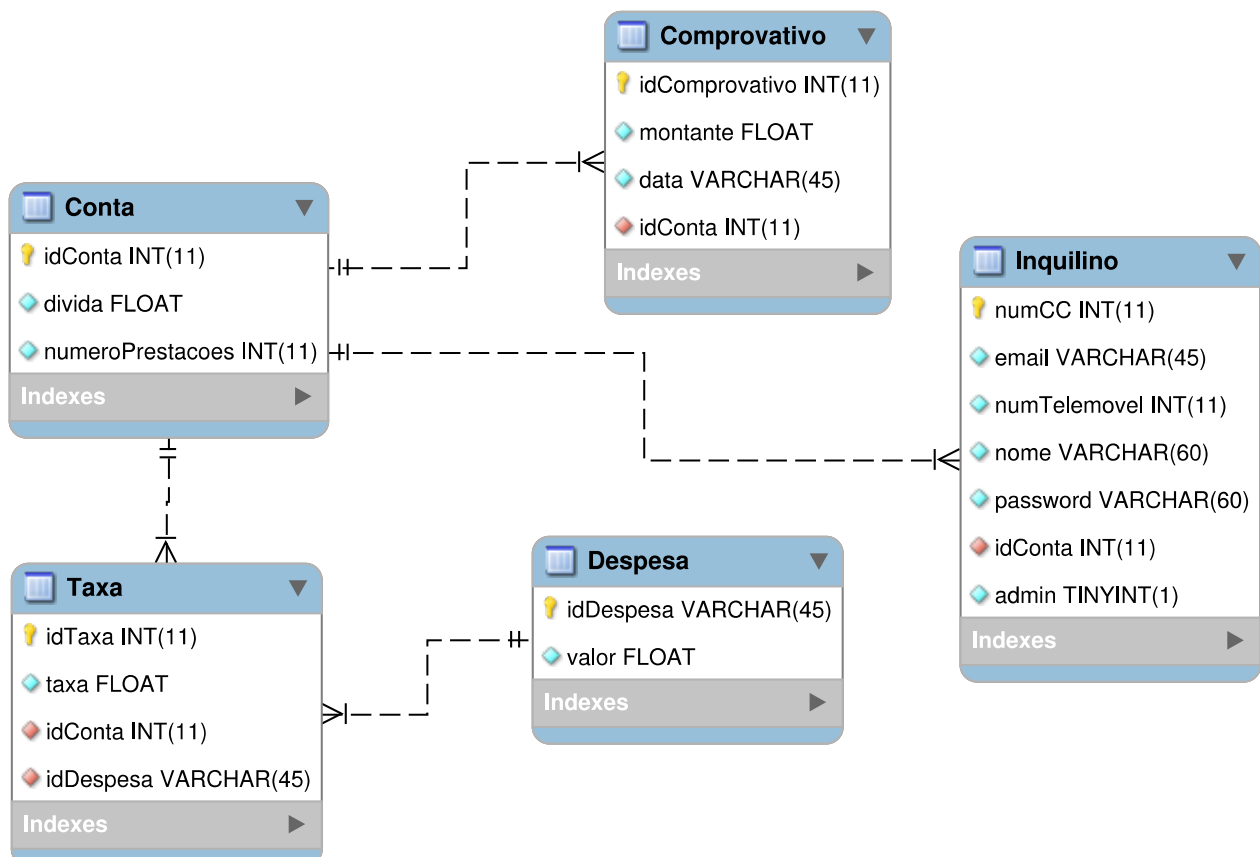
A partir deste diagrama de classes inicial, sem a introdução de DAOs, foi concebido o diagrama de classes com recurso aos Data Access Objects, que é o diagrama final e, efetivamente, implementado fisicamente.

## Diagrama de Classes com DAO's



A maior transformação do diagrama de classes anterior para o o atual foi a implementação de DAO's (Data Access Objects). Os DAO's são classes que permitem a conexão a uma base de dados e servem maioritariamente para colocar, ou obter a informação depositada numa base de dados. Assim sendo, de modo a substituir os Map's utilizados no diagrama anterior, o grupo criou três classes do tipo DAO que acedem à base de dados "apartamento" e que, juntas, conseguem aceder a todos os dados da base de dados. Todos os DAO's implementados possuem uma variável Connection que permite a conexão à base de dados.

A base de dados "apartamento" tem o seguinte modelo lógico:



A classe Apartamento, no diagrama anterior, tinha uma variável Despesas que possuía um Map, que mapeava o tipo da despesa (String) para o seu valor (float). No novo diagrama, o GestaoApartamento possui um DespesaDAO que permite o acesso a todos os dados referentes às despesas colocados na base de dados (tais como o idDespesa (nome da despesa), e o valor (da despesa)). Este DAO usa a classe Despesa para, por exemplo, retornar um objeto Despesa ao GestaoApartamento quando esta lhe é pedida. O grupo achou necessário adicionar um DAO neste caso, visto que poderiam ser muitas despesas e revelou-se necessário armazenar estas na base de dados.

Além das despesas, a classe GestaoApartamento possui um ComprovativoDAO que permite aceder a todos os dados referentes aos comprovativos dos inquilinos. Tal como nas despesas, o ComprovativoDAO usa a classe Comprovativo para poder devolver o objeto Comprovativo quando pedido. A razão para transformar o Map de inteiros (id do comprovativo) para o objeto Comprovativo, do diagrama anterior, para um DAO surgiu da possibilidade do crescimento exponencial dos comprovativos ao longo do tempo.

Finalmente, a classe Apartamento do diagrama anterior possuía um Map<Integer, Inquilino> que mapeava os números de cartão de cidadão com a classe Inquilino, que foi atualizado para um InquilinoDAO. No entanto, o InquilinoDAO, não acede só aos dados referentes ao Inquilino. O DAO também acede à tabela que contém a conta do inquilino e às tabelas que contêm as taxas do inquilino. O grupo decidiu que não existiria razão para criar um DAO para a classe Conta, pois, sendo o relacionamento 1 para 1, o número de tabelas de inquilinos na base de dados será igual ao número de tabelas de contas desse inquilino. Deste modo, consegue aceder ao inquilino e à sua conta através da chave estrangeira contida na tabela inquilino. Relativamente a um possível TaxaDAO, como, muito dificilmente, um inquilino terá o número de taxas igual ao número de despesas (será quase sempre inferior), não há necessidade de o implementar. O InquilinoDAO consegue, assim, aceder, através de chaves estrangeiras, ao inquilino, à sua conta, e às suas taxas (InquilinoDAO usa Inquilino, Conta, e Taxa).

Concluindo, a última alteração reside no facto do Apartamento já não possuir um Administrador, mas sim, o Administrador extends Inquilino, o que passa a permitir vários administradores.

## Mockups da Interface Gráfica

se CC ou outro campo  
invalidado, dar msg  
de erro e voltar  
a janela c/ os  
campos vazios

Mockups das Interfaces

**Iniciar sessão**

Nº CC

Pass

**Registrar Utilizador**

Email

Nº CC

Nome

Pass

Admin? ☒

**Pass**

Interfaces

**Registrar Apartamento**

Rua

Nº Porta

Localidade

Cod Postal

**Remover inquilino**

Nº CC

Nº CC

**Registrar Comprovativo**

Nº CC

Valor Pago

**Alterar tipo pagamento**

Nº CC

**Tipo pagamento**

**Mudar Password**

Nº CC

Pass antiga

Pass nova

**Atualizar dados**

**Sim**

**Não**

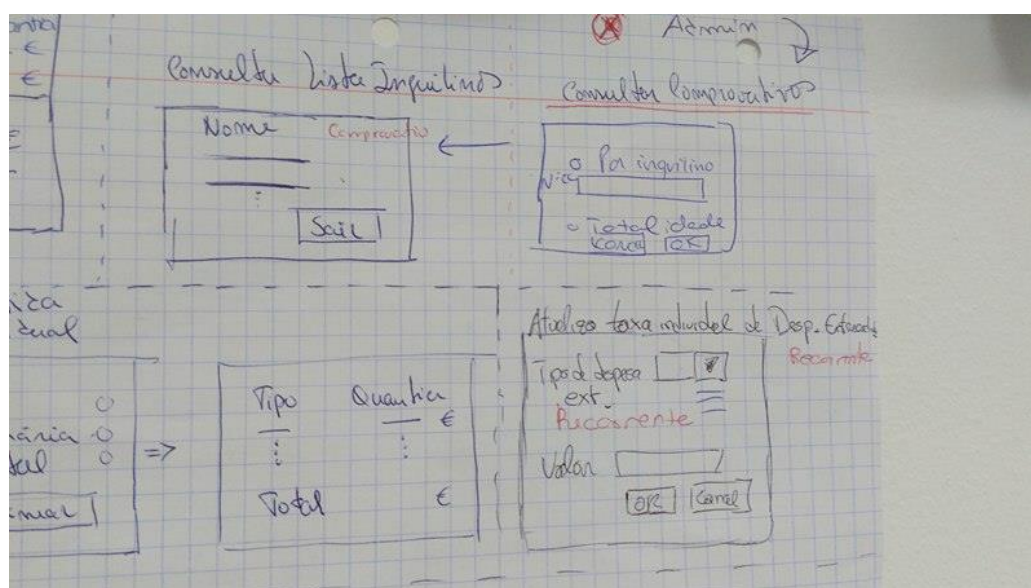
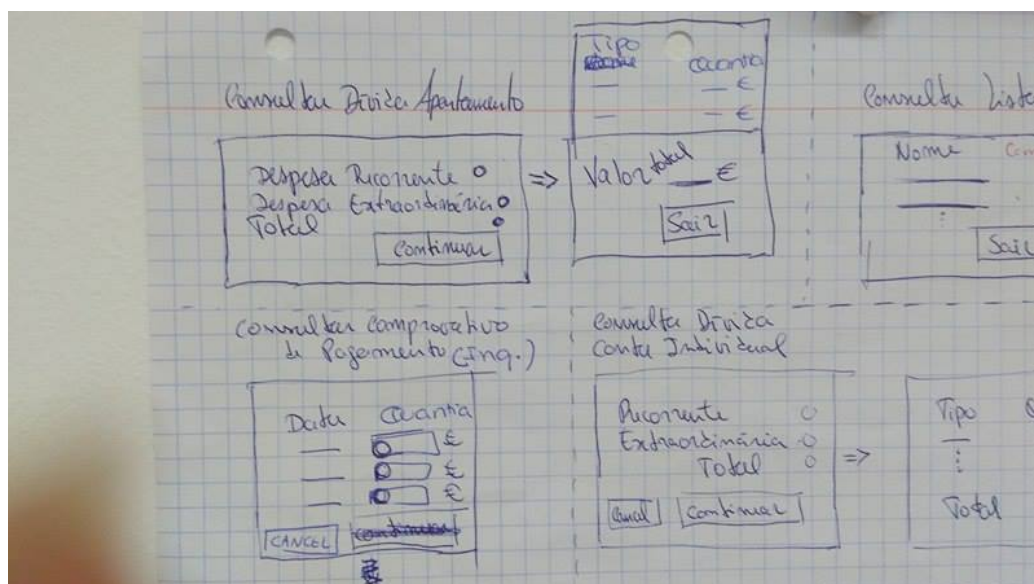
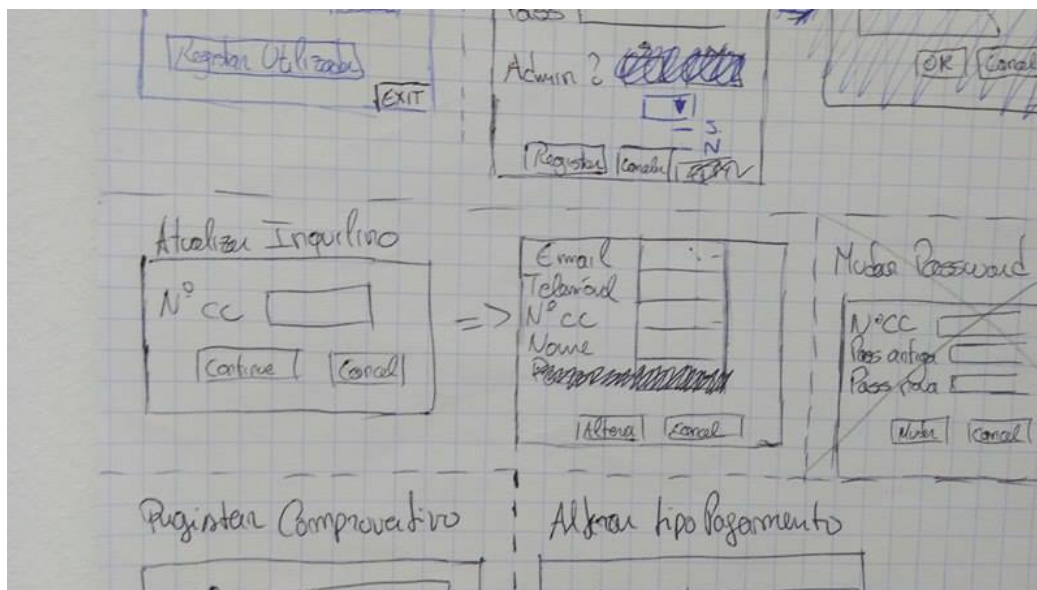
**Depende do tipo de pagamento:**

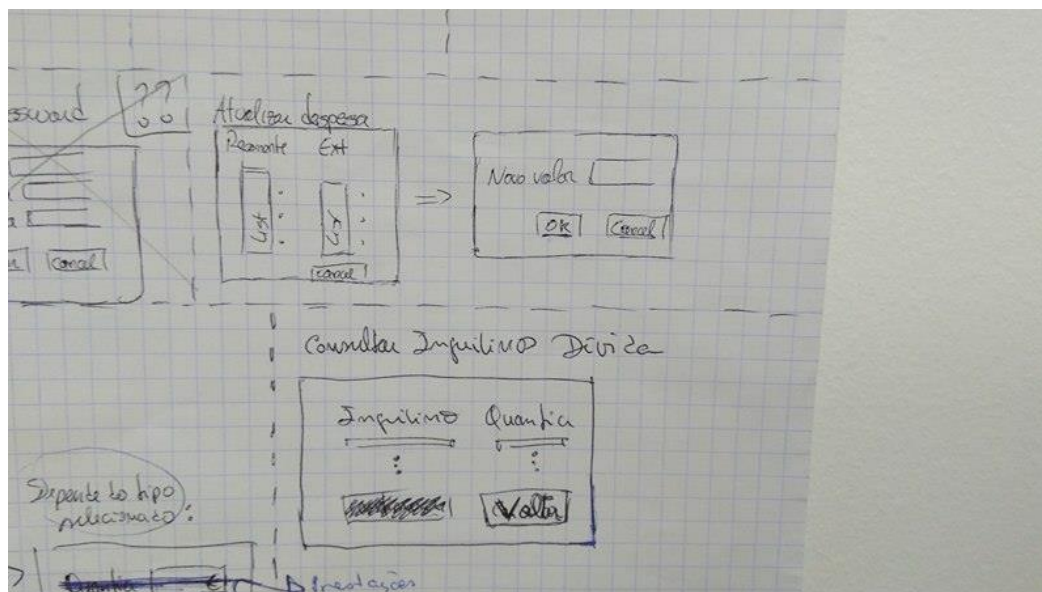
**Cancelar**

**Sim**

**Não**







Denote-se que a interface final sofreu algumas alterações consoante se foi desenvolvendo o código e refletindo sobre algumas decisões tomadas.

# Diagramas de Use Cases

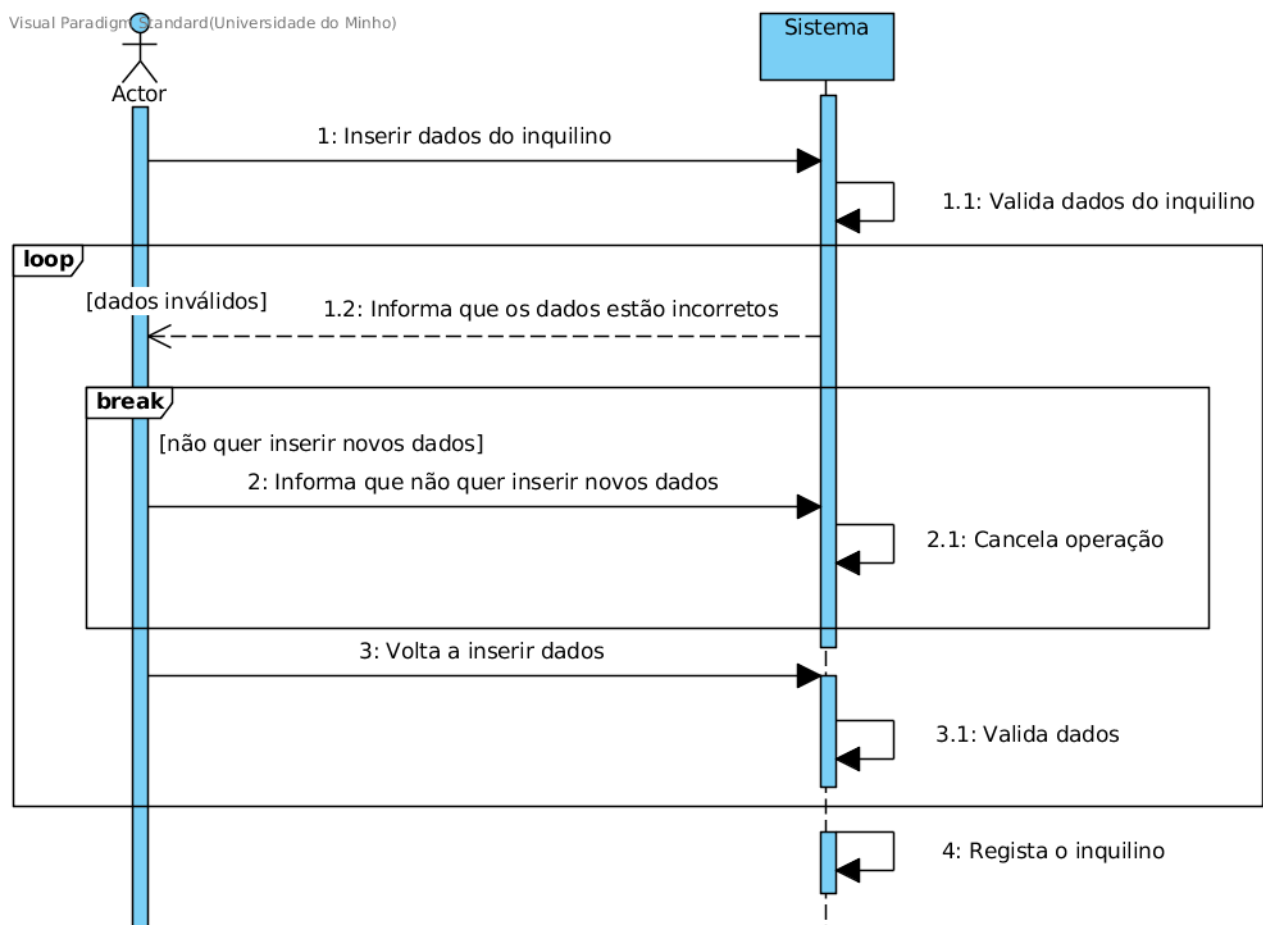
## Registar Utilizador (efetuado por Inquilino e Administrador)

### - Especificação do Use Case

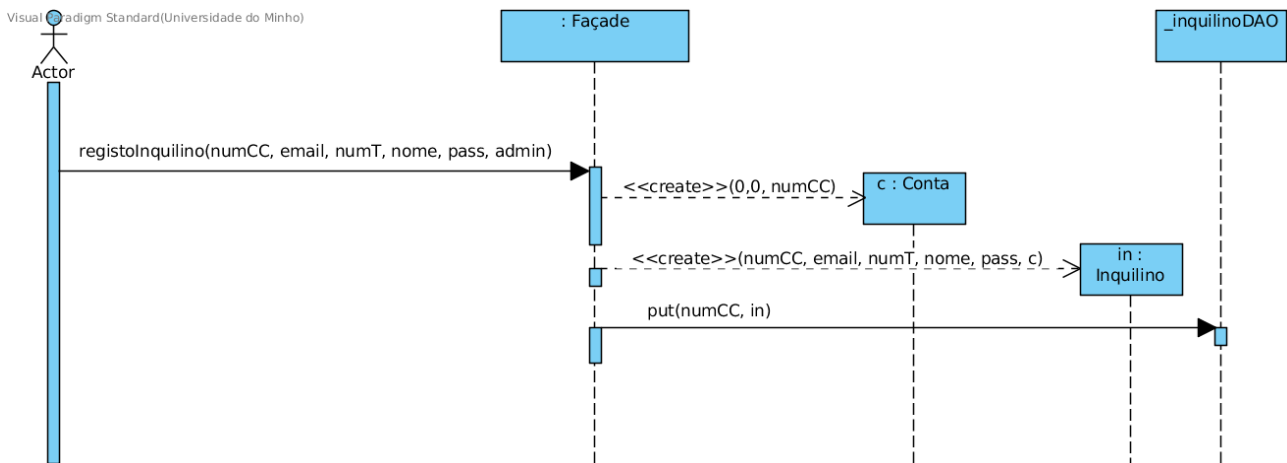
Use Case:	Registar Inquilino	
Descrição:	Utilizador regista um novo inquilino	
Pré-Condição:	Utilizador está autenticado	
Pós-Condição:	Inquilino fica registado no sistema	
	Ator	Sistema
Comportamento Normal	1. Inserir dados do novo inquilino	
		2. Valida dados do inquilino
		3. Regista o inquilino
Comportamento Alternativo [dados inseridos incorretos] (passo 2)		2.1 Informa que dados estão incorretos
	2.2 Volta a inserir os dados	
		Volta ao passo 2
Exceção [não quer inserir novos dados] (passo 1)	1.1 Informa que não quer inserir novos dados	
		1.2 Cancela opção

O utilizador deve inserir todos os dados pedidos, tem ainda a hipótese de inserir o convite de administrador caso assim o pretenda. Se os dados inseridos forem incorretos é lhe informado que estes são inválidos e sugerido que os volte a inserir. Se o utilizador quiser cancelar a operação pode o fazer em qualquer momento.

### - Diagrama de Sequência



## - Diagrama de Sequência com Subsistemas



O actor, neste caso já estamos no contexto da interface gráfica, após a validação dos dados invoca um método do façade, que se encarrega de criar um objecto Conta e um objecto Inquilino. A finalização deste método consiste em executar um put dos dados necessários no objecto \_inquilinoDAO (variável de instância no façade).

## - Registar Utilizador no contexto da interface gráfica

### Janela Inicial



### Menu de Registar Utilizador

Registar Utilizador	
Número de Cartão de Cidadão	17090123
Nome	João Ribeiro
E-mail	j.ribeiro95@gmail.com
Número de Telemóvel	91099123
Password	joaoribeiro91
Convite de Administrador	
Cancelar	Confirmar

## - Registrar Utilizador aquando a inserção incorreta de dados

The screenshot shows a window titled "Registrar Utilizador". It contains several input fields with the following data: "Número de Cartão de Cidadão" (17090123), "Nome" (João Ribeiro), "E-mail" (j.ribeiro95@gmail.com), "Número de Telemóvel" (91teste), "Password" (joaoribeiro91), and "Convite de Administrador" (empty). At the bottom are "Cancelar" and "Confirmar" buttons.

The screenshot shows the same "Registrar Utilizador" window, but the "Número de Cartão de Cidadão" field is empty. A red error message "Utilizador inválido!" is displayed next to the field. The other fields remain the same as in the previous screenshot.

## Adicionar Comprovativo

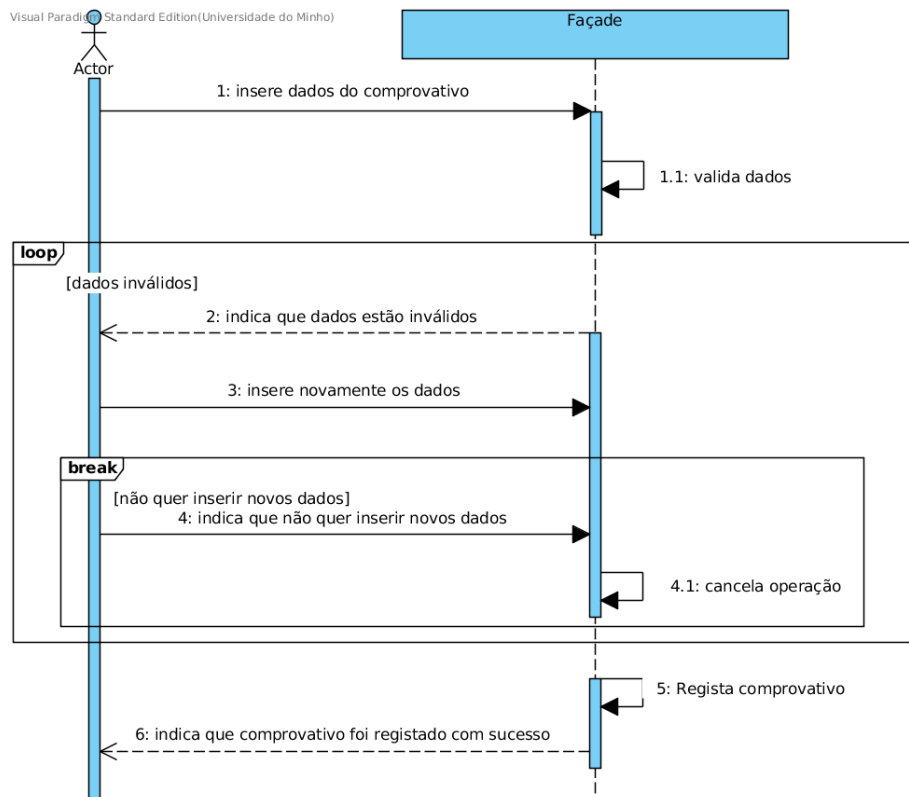
Um dos Use Cases específicos do administrador é o Registrar/Adicionar Comprovativo, uma vez que ele é o único que tem permissão para o fazer. Assim, este Use Case torna-se exclusivo ao administrador e só inquilinos com tais credenciais é que podem executá-lo.

## - Especificação de Use Case

Use Case:	Regista Comprovativo	
Descrição:	Administrador regista comprovativo de pagamento	
Pré-Condição:	Administrador está autenticado	
Pós-Condição:	Pagamento e comprovativo ficam registados	
	Ator	Sistema
Comportamento Normal		1. Mostra lista de inquilinos
	2. Seleciona inquilino	
	3. Indica data do comprovativo	
		4. Valida data do comprovativo
		5. Regista comprovativo de pagamento
Comportamento Alternativo [data do comprovativo incorreta] (passo 4)		4.1 Informa que data está incorreta
		Volta ao passo 3
Exceção [não insere nova data] (passo 3)	3.1 Indica que não quer inserir nova data	
		3.2 Cancela operação

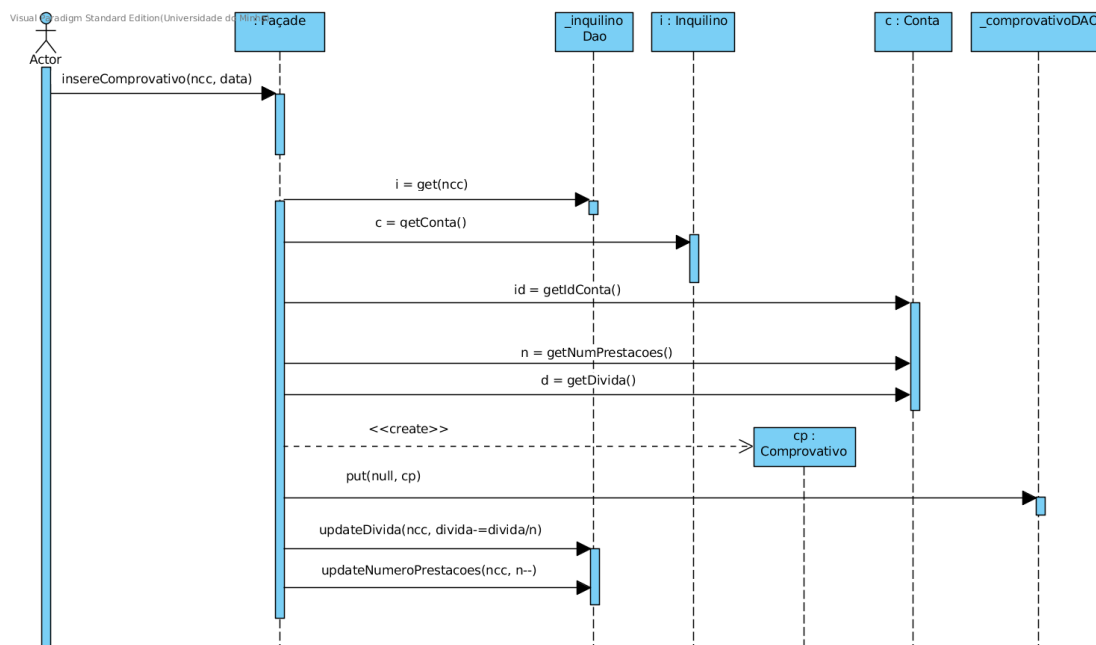
A especificação deste Use Case mostra que além do comportamento normal em que a informação inserida é inserida corretamente e, assim, validada e registada, também há a possibilidade de inserir dados inválidos. Neste caso, é dada a possibilidade de inserir novamente os dados ou, se preferir, cancelar a operação.

## - Diagrama de Sequência



No diagrama de sequência que foi baseado na sua especificação, conseguimos ver a sua tradução fiel na forma de um esquema, mais apelativo ao cliente. Assim, consegue-se perceber o modo de funcionamento desta funcionalidade do sistema.

## - Diagrama de Sequência de Subsistemas



A partir deste novo diagrama obtemos uma informação mais precisa do modo de operação do Use Case, incluindo os métodos necessários para a sua realização. Denote-se que já se fez a introdução dos Data Access Objects (DAO), que fazem a recolha e manipulação de dados na base de dados. Também temos a classe Façade que, por sua vez, comunica com os DAOs, isto é, serve de ponto intermédio entre a interface e os dados.

## - Interface

The screenshot shows a window titled "Menu de Administrador". It contains two columns of radio button options. The first column includes "Atualizar despesas", "Atualizar taxas", "Atualizar número de prestações", "Adicionar despesa", and "Remover inquilino". The second column includes "Consultar inquilinos em dívida", "Consultar dívida total do apartamento", "Consultar lista de inquilinos", "Adicionar Comprovativos" (which is selected), and "Consultar comprovativos". At the bottom, there are three buttons: "Logout", "Menu de Inquilino", and "OK".

The screenshot shows a window titled "Adicionar Comprovativo". It features a dropdown menu labeled "Escolha o inquilino:" with the selected item "1 - Afonso - 145.23€". Below this, there is a section "Insira a data:" with three input fields for "Dia" (containing "12"), "Mês" (containing "12"), and "Ano" (containing "2016"). At the bottom, there are three buttons: "Retroceder", "Comprovativo registado!", and "Confirmar".

The screenshot shows a window titled "Inquilino em Dívida". It contains a list of tenants and their debts: "3 - O Primeiro - 324.075€", "2 - Henriques - 123.93€", and "1 - Afonso - 72.615€". At the bottom, there is a "Retroceder" button.

Quando um inquilino com credenciais de administrador inicia sessão é conduzido ao menu de administrador que mostra opções que são exclusivas ao mesmo. Assim, ao selecionar a opção “Adicionar Comprovativos” é-lhe mostrada uma nova janela em que pode selecionar o inquilino em questão e registar a data do pagamento. Se esta estiver correta, o comprovativo é registado no sistema, caso contrário é-lhe indicado que os dados são inválidos.

Este último caso não leva a que o administrador volte à janela anterior (menu de administrador), pois seria uma opção pouco eficiente, uma vez que se se enganar ou quiser introduzir vários comprovativos de outros inquilinos consegue poupar muitos “clicks”. Retornando, apenas, para a janela anterior quando selecionar o botão “Retroceder”.

Por fim, se, após registar o comprovativo com sucesso, for verificar a dívida do inquilino em questão (Afonso), consegue verificar que, de facto, o valor decresceu de acordo com o número de prestações do mesmo.

## Atualizar Taxa

### - Especificação de Use Cases

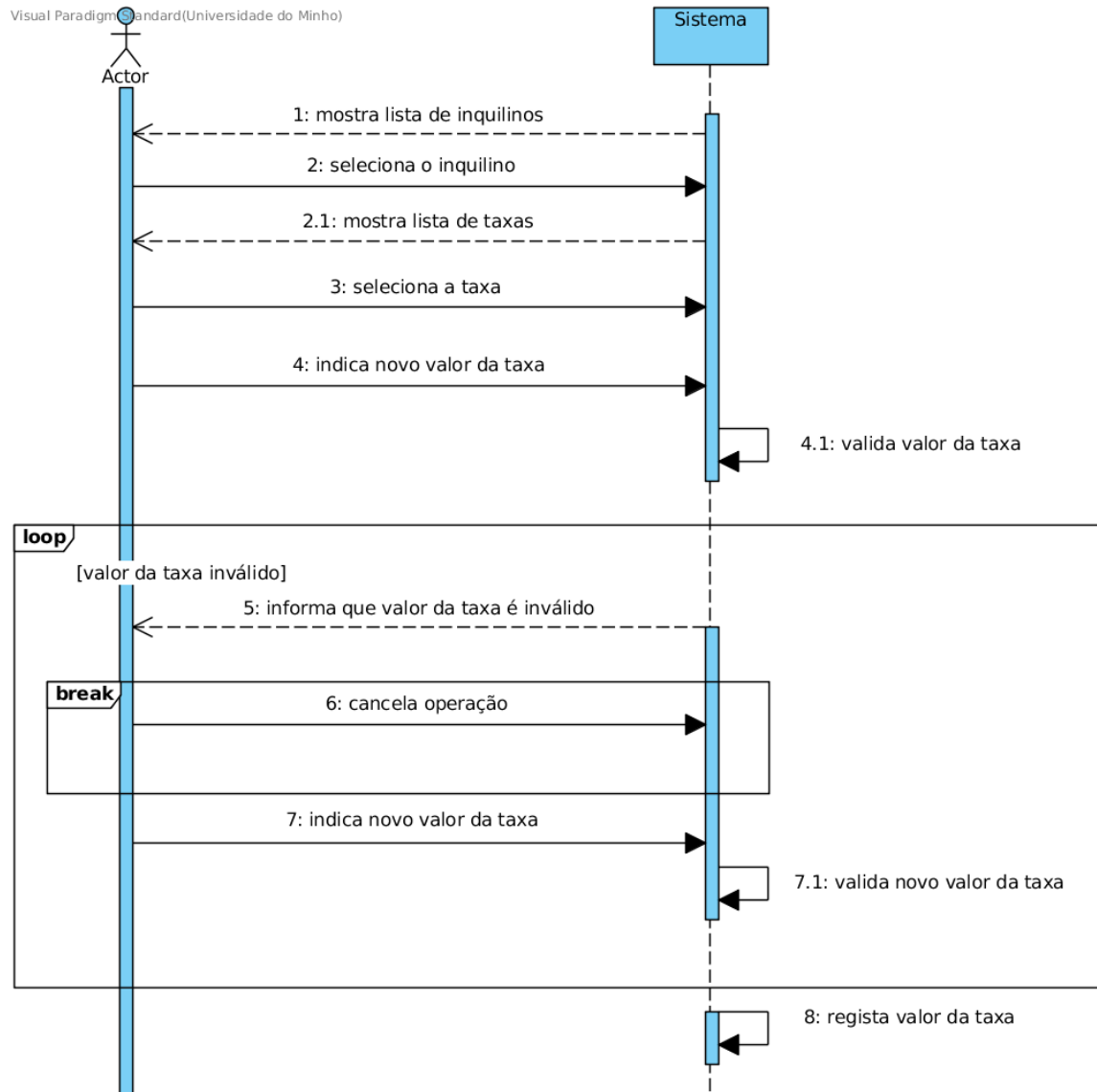
Use Case:	Atualizar Taxa	
Descrição:	Administrador atualiza taxa	
Pré-Condição:	Administrador está autenticado	
Pós-Condição:	Taxa foi atualizada	
	Ator	Sistema
Comportamento Normal		1. Indica tipos de despesas
	2. Seleciona tipo de despesa pretendida	
	3. Indica novo valor da taxa da despesa escolhida	
		4. Valida o valor da taxa
		5. Regista valor da taxa
Comportamento Alternativo [valor da taxa inválida] (passo 4)	4.2 Volta a inserir novo valor da taxa	4.1 Indica que valor da taxa é inválida
		Volta ao passo 4
Exceção 2 [não insere novos dados] (passo 4.2)	4.2.1 Indica que não quer inserir novos dados	
		4.2.2 Cancela operação

Este Use Case representa a atualização de uma taxa. O sistema indica os tipos de despesa, um dos quais vai ser selecionado pelo ator, que depois irá indicar o novo valor de taxa. Caso o ator não queira inserir dados, a operação é cancelada (exceção). Se ele inserir dados, estes têm de ser validados. Caso estejam corretos, a despesa é validada. Caso contrário, o ator tem de inserir novamente a despesa (comportamento alternativo).

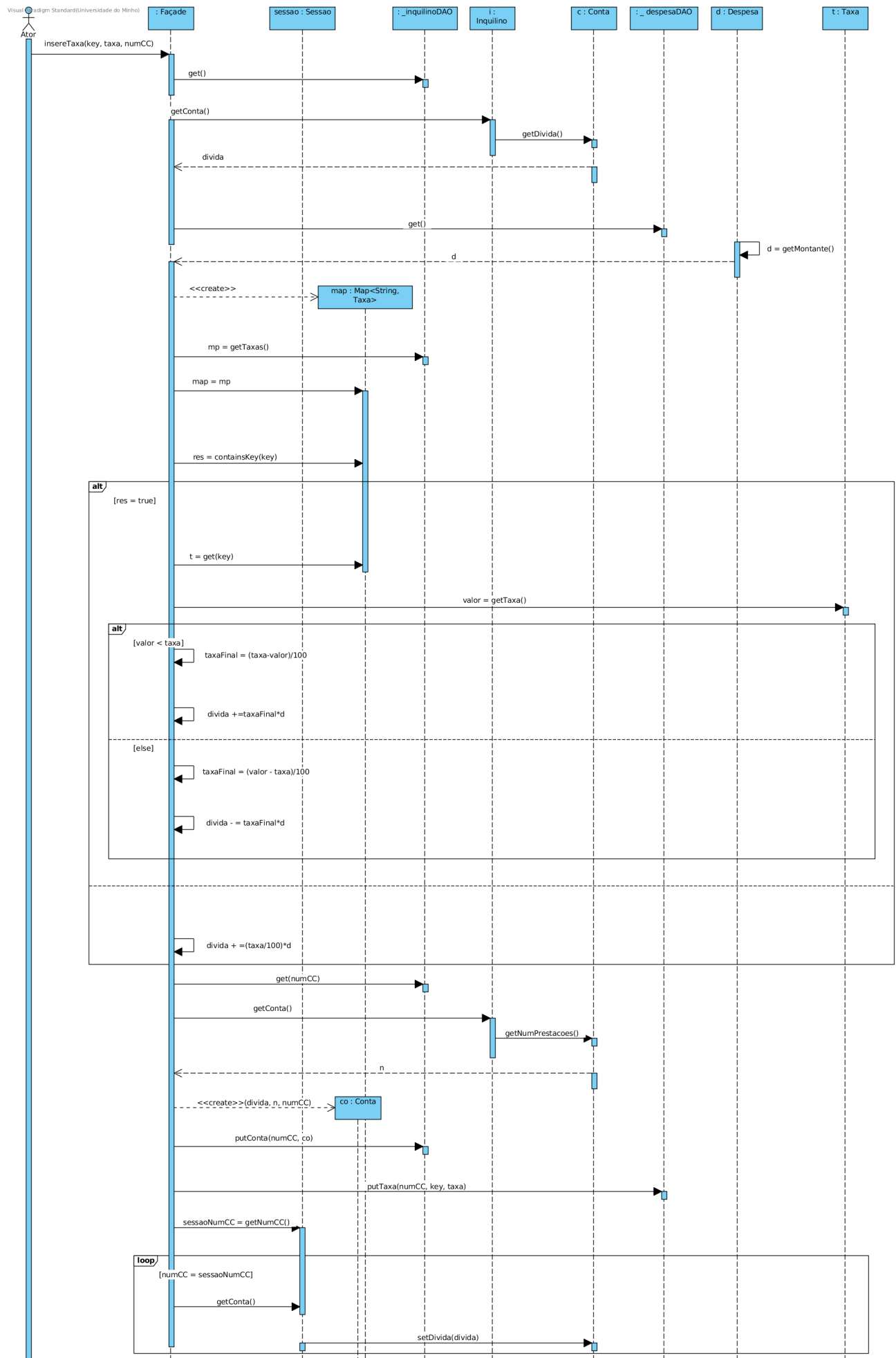


## - Diagrama de Sequência

O diagrama de sequência desta operação representa, em forma de diagrama, a especificação do Use Case, sendo a única diferença digna de referência a utilização de um ciclo enquanto ele não inserir os dados corretos.



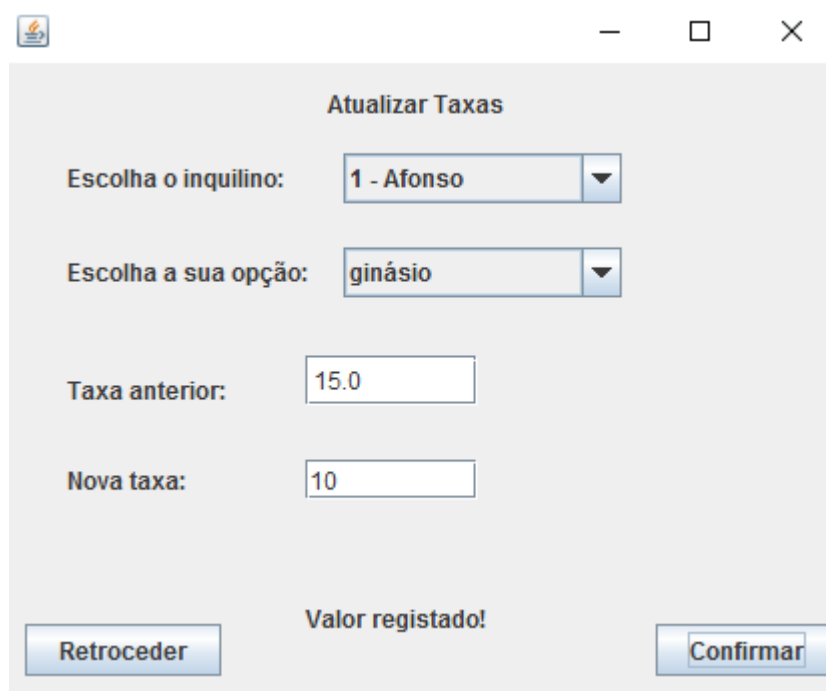
## - Diagrama de SubSistemas



A estratégia para esta operação consistiu em ir à base de dados e atualizar o valor que lá estava. Caso não existisse nenhuma taxa, seria necessário criá-la com todos os parâmetros passados no método `insereTaxa` (sendo eles a despesa, a taxa, o número de cartão de cidadão). Primeiramente, vão buscar-se três coisas à base de dados (através dos DAO's). A dívida atual, o valor da despesa (relativa à taxa que se quer alterar), e um mapeamento de todas as taxas do inquilino. A partir daí, verificamos se existe alguma taxa referente a essa despesa. Se existir, atualiza-se o valor. Se não existir, cria-se uma nova com esse valor. Aquando a alteração da taxa, o valor da dívida também tem de ser atualizado.

Finalmente, inserem-se os novos valores relativos à conta e à taxa alterada na base de dados. Se o administrador estiver a alterar a sua própria taxa, é necessária atualizar também esse valor de dívida na `GestaoApartamento`, pois esta possui um objeto inquilino que representa o inquilino com a sessão iniciada no sistema.

## - Interface gráfica



**Atualizar Taxas**

Escolha o inquilino: 1 - Afonso ▼

Escolha a sua opção: ginásio ▼

Taxa anterior: 15.0

Nova taxa: 10

Valor registado!

Retroceder Confirmar

A interface apresenta duas escolhas: uma para o inquilino e outra para a despesa. Quando o administrador selecionar as duas, são desencadeadas duas ações. A primeira mostra o valor antigo da taxa. A segunda permite editar o novo valor da taxa. Clicando no botão confirmar, regista a nova taxa (se esta for válida). Caso esta seja inválida, o sistema avisa que o valor é inválido. A qualquer momento, o administrador pode desistir da operação.

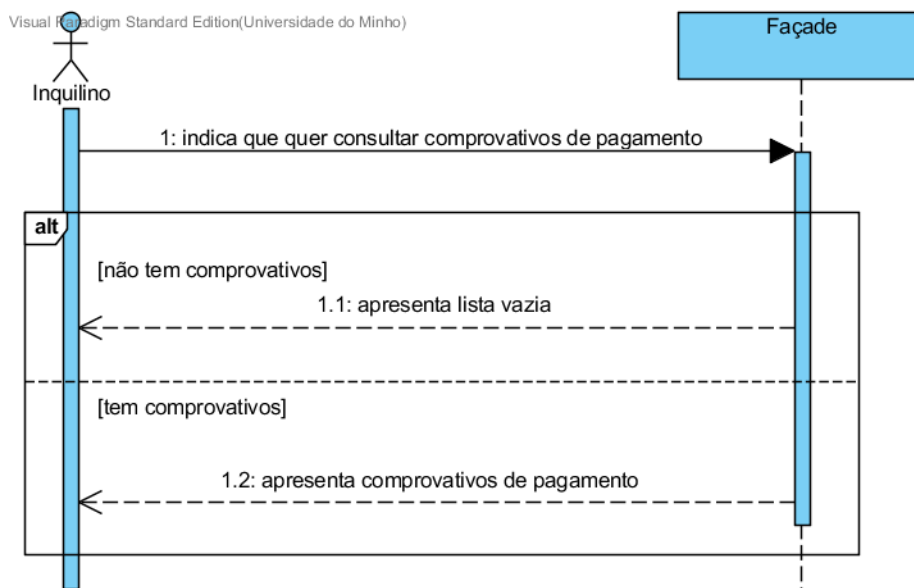
# Consultar Comprovativos de Pagamento (Inquilino)

## - Especificação do Use Case

Use Case:	Consultar Comprovativos Individuais de Pagamento	
Descrição:	Inquilino consulta lista de comprovativos individuais de pagamento	
Pré-Condição:	Inquilino está autenticado	
Pós-Condição:		
	Ator	Sistema
Comportamento Normal		1. Mostra comprovativos de pagamento
Exceção [inquilino não tem comprovativos] (passo 1)		1.1 Mostra lista vazia

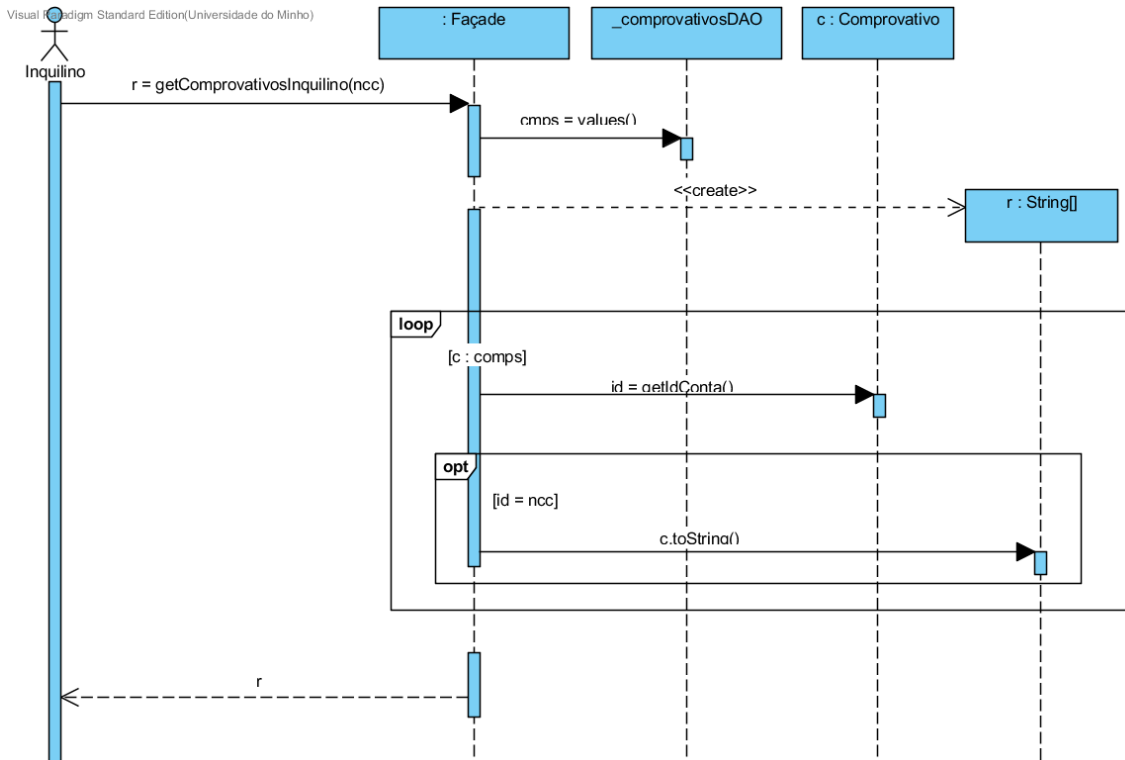
O sistema indica os comprovativos de pagamento do inquilino (comportamento normal), ou, caso não os haja, indica que não há comprovativos de pagamento.

## - Diagrama de sequência



Tal como referido no use case, o inquilino indica que quer consultar os seus comprovativos de pagamento. A resposta do sistema irá variar dependendo se ele tem comprovativos ou não.

## - Diagrama de subsistemas

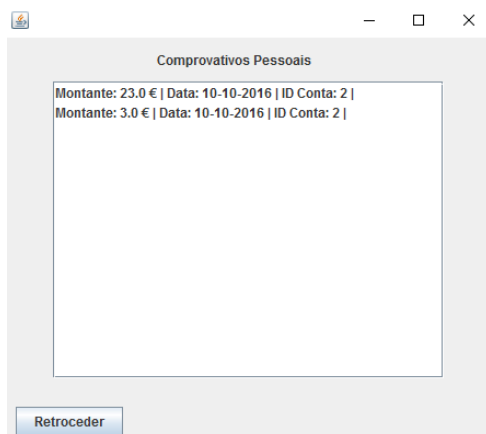


A estratégia para este diagrama de subsistemas consistiu em retornar um array de Strings, representativas de comprovativos. A razão de se retornar um array de strings reside na facilidade de trabalhar com esta estrutura na interface gráfica.

Primeiramente, cria-se um array com o tamanho de todos os comprovativos (pois o tamanho do array nunca será superior a todos os comprovativos). De seguida, vão-se buscar todos os comprovativos à base de dados (através do DAO) e, aqueles que forem do inquilino, adicionam-se ao array.

Por fim, retorna-se o array com todos os comprovativos do inquilino.

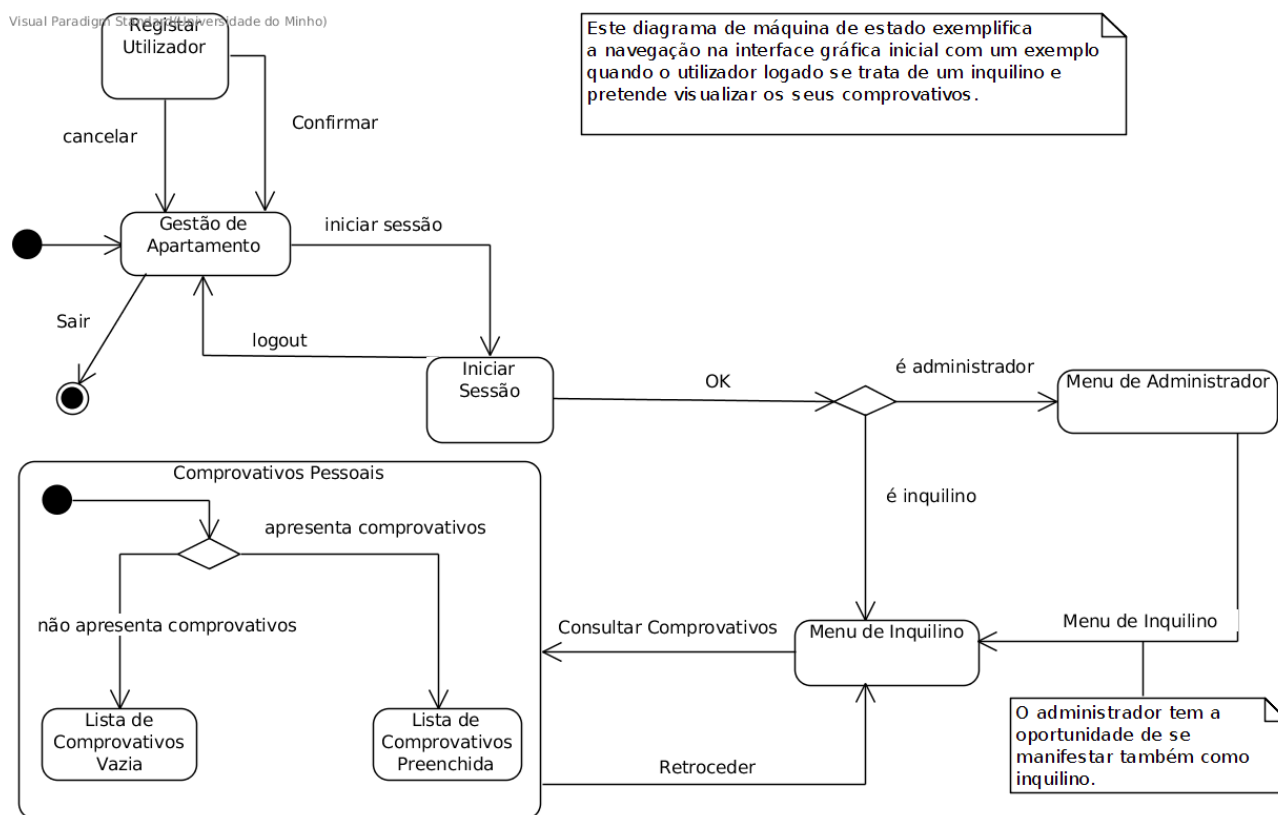
## - Interface gráfica



Escolhida a opção, ele mostra os comprovativos do inquilino, ou, se não existirem, não mostra nada.

# Diagramas de Máquinas de Estado

## - Representação da navegação do menu inicial.

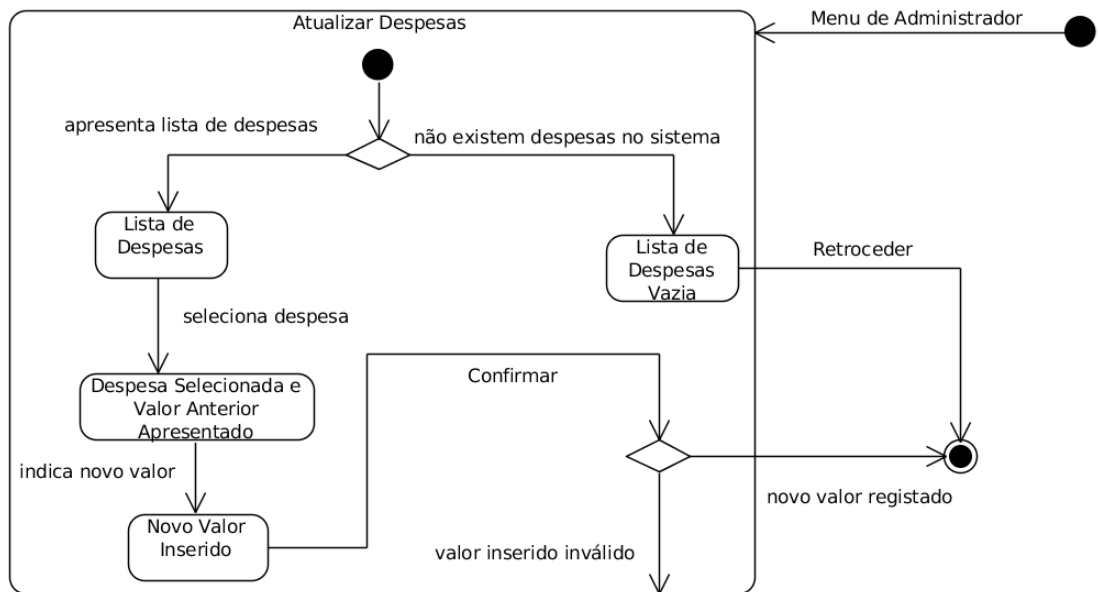


O estado inicial da interface reside na frame Gestão de Apartamento. O utilizador pode se registar ou iniciar sessão. Após o menu de registo com sucesso, o inquilino tem de iniciar sessão com as suas credenciais para poder avançar para o Menu de Inquilino ou Menu de Administrador se for o caso. Case se trate de um Administrador, pode a qualquer momento mudar para o Menu de Inquilino se assim pretender. Denote-se que o inquilino (utilizador normal) **nunca** tem acesso ao Menu de Administrador.

No diagrama de máquina de estado acima representado, está descrito o exemplo de um menu de inquilino: Comprovativos Pessoais. Aqui uma de duas coisas podem ocorrer: o inquilino não tem comprovativos e uma lista vazia é apresentada ou é lhe descrito o seu histórico de comprovativos. A qualquer momento o inquilino pode retroceder para o menu anterior.

## - Atualizar Despesas (Administrador)

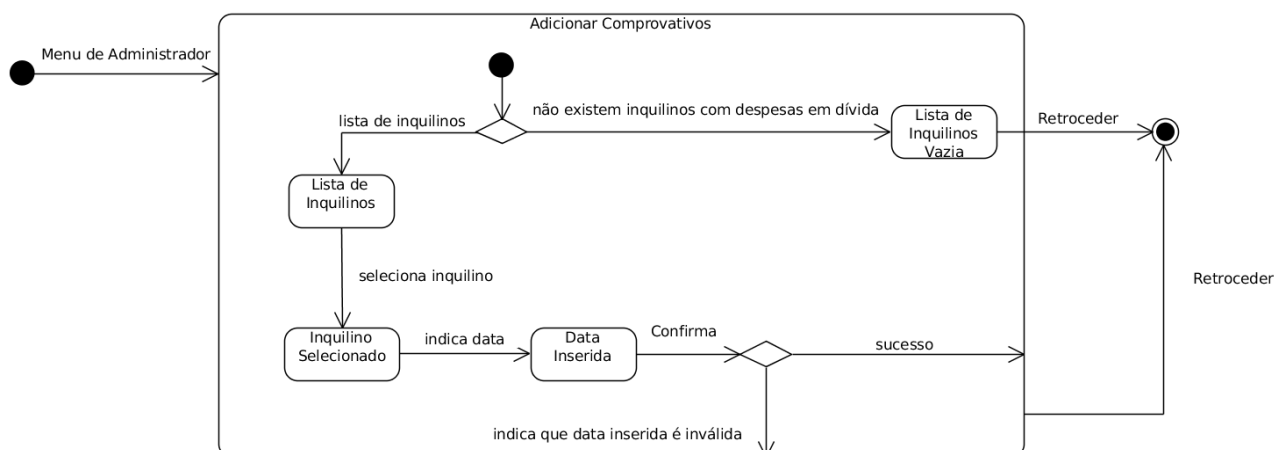
Visual Paradigm Standard (Universidade do Minho)  
Deve estar previamente logado como administrador.



O administrador tem a oportunidade de atualizar despesas já existentes. Ao entrar no Menu Atualizar Despesas uma de duas coisas pode ocorrer: se não existirem despesas então o administrador só pode retroceder ou caso existam este deve selecionar a despesa, de seguida, deve indicar o novo valor e por fim confirmar. Se o valor estiver correcto então o valor é registado caso contrário é lhe informado que o valor inserido é inválido e este é convidado a refazer a operação. A qualquer momento o Administrador pode retroceder para o menu anterior.

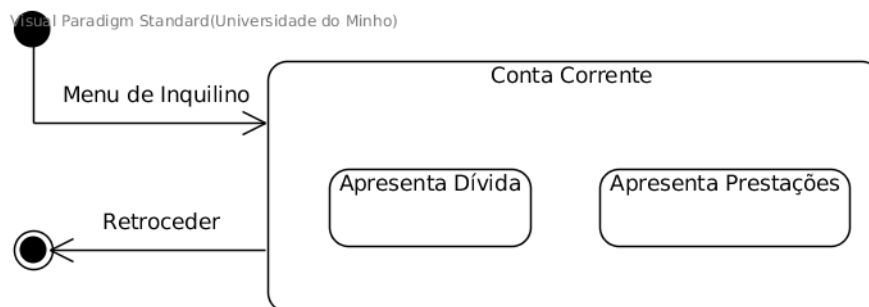
## - Adicionar Comprovativos (Administrador)

Visual Paradigm Standard (Universidade do Minho)  
Deve estar previamente logado como administrador.



O administrador tem a oportunidade de adicionar comprovativos que consiste em comprovar o pagamento de um determinado inquilino. Ao entrar no menu Adicionar Comprovativos, uma de duas coisas pode ocorrer: se não existirem inquilinos com pagamentos em dívida então o administrador apenas pode retroceder para o menu anterior ou caso existam, o administrador deve seleccionar o inquilino pretendido, de seguida, deve inserir a data e confirmar a operação. Se a data inserida estiver incorreta então é-lhe informado que não pode efetuar a operação. O administrador a qualquer momento pode retroceder para o menu anterior.

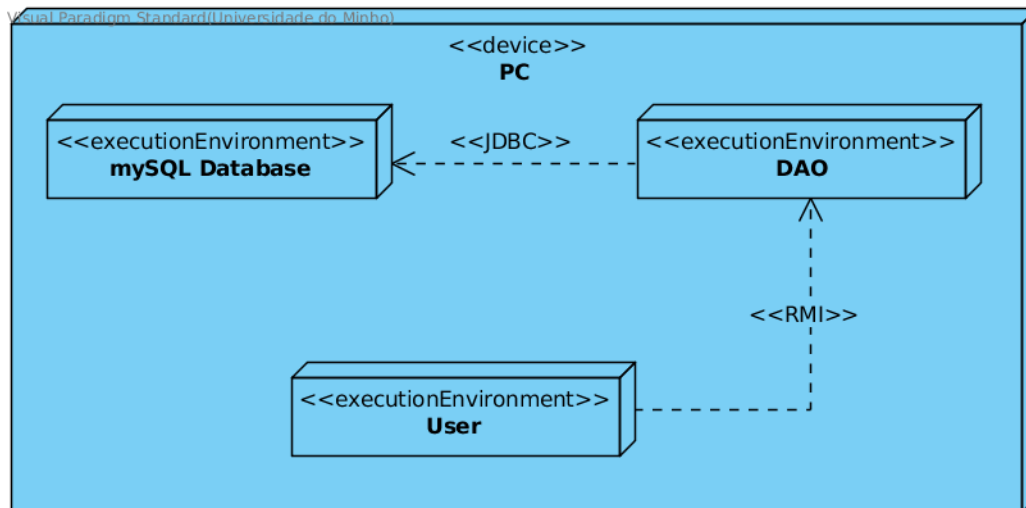
#### - Consultar Conta Corrente (Inquilino e Administrador)



Tanto o inquilino como o administrador podem visualizar a sua conta corrente. Isto é, o valor em dívida. Assim, caso haja alguma dívida para ser saldada, ao entrar no menu Conta Corrente, o utilizador pode visualizar que o valor em dívida e o valor das prestações é 0, ou então visualizar o valor da dívida e o número de prestações restantes.

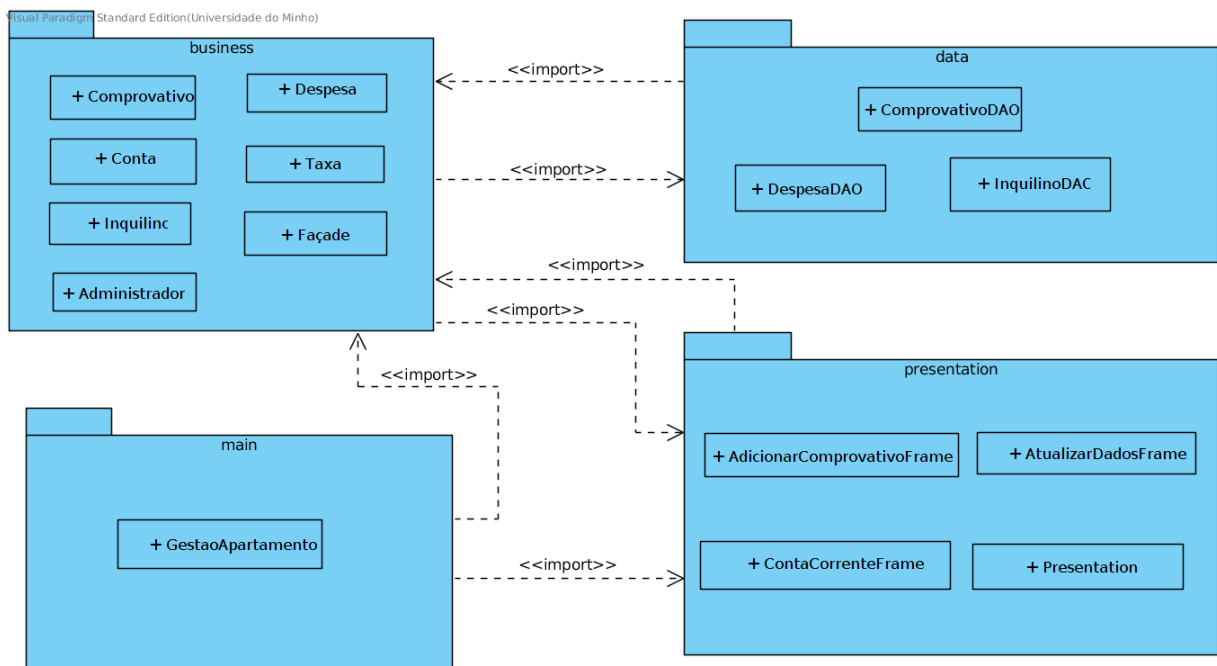


## - Diagrama de Instalação



No mesmo ambiente de execução, o utilizador através da façade invoca métodos aos DAO's. Por sua vez, os DAO's através de uma interface JDBC armazenam e obtêm dados através de uma base de dados SQL. O hardware correspondente é um PC.

## Diagramas de Packages



Agrupamos as classes em quatro pacotes, de acordo com a sua função e conteúdo, sendo eles: business, data, main e presentation. Dentro do package business, temos as classes relativas à lógica de negócio, Comprovativo, Despesa, Conta, Inquilino, Taxa e Administrador, bem como a classe Façade que acede aos DAOs. Das classes que fazem parte deste package, apenas a Façade importa classes dos outros packages definidos, sendo elas: todas as classes do pacote data e as classes de Exceção, que estão no pacote presentation.

De seguida, temos o package data, do qual fazem parte as classes relativas ao Data Access Objects (DAOs): ComprovativoDAO, DespesaDAO e InquilinoDAO. São estas que acedem aos dados armazenados na base de dados. Deste modo, têm necessidade de fazer import de algumas das classes do package business para criar novas instâncias de objetos.

O pacote presentation tem como responsabilidade agregar todas as classes relativas à interface do sistema. Deste modo, encontramos aqui todas as classes Jframes criadas, que acedem à Façade do package business.

Por fim, a classe main possui a classe GestaoApartamento, que, por sua vez, inclui a função main do programa.

## Conclusão

Nesta segunda fase do projeto, procuramos melhorar o que tínhamos definido na fase anterior, pois fomos aperfeiçoando os diagramas desenvolvidos à medida que o desenvolvimento do projeto avançava. Com o auxílio dos vários diagramas concebidos foi-nos possível desenvolver um software funcional e, de acordo, com os requisitos por nós exigidos, uma vez que o nível de pormenor aumentava em cada tipo de diagrama.

Com este planeamento, o grupo conseguiu executar a implementação física do software, seguindo os passos anteriormente desenvolvidos.

O grupo tirou partido do facto deste método ser iterativo, pois permitiu alterar certas ideias iniciais que estavam erradas e não contribuíam para o bom funcionamento do sistema.

Em suma, obtivemos um bom resultado, pois conseguimos criar um programa completamente funcional que respeita o planeamento efetuado. Com uma boa gestão do tempo, o grupo implementou todas as funcionalidades descritas nos Use Cases definidos, bem como a sua totalidade.

Relativamente a pontos fortes do programa, ele agrega diversos inquilinos (e administradores), assim como aceita uma grande variedade de despesas e taxas. O nosso programa é, assim, flexível para qualquer eventualidade, assim como persistente, visto que se conecta a uma base de dados. Em relação à interface gráfica, ela é fácil e simples de utilizar, promovendo a boa utilização por parte dos utilizadores. Denote-se que se visou diminuir a carga de trabalho do utilizador, visto que a interface apresenta os dados redundantes, em vez de ser o próprio a inseri-los. Em termos de consistência de dados, temos as verificações necessárias para a garantir.

Algumas das limitações são o facto deste software apenas se aplicar a um apartamento, o que seria muito fácil de alterar. No entanto, o grupo apenas visou o seu contexto, tendo em conta um único apartamento.