

Gestão de Condomínio

RELATÓRIO DE PROJETO FINAL DA LICENCIATURA EM ENGENHARIA INFORMÁTICA

AUTOR

Luís Manuel Magalhães de Sousa

ORIENTADOR(ES)

António Alberto dos Santos Pinto

ANO

2014

Agradecimentos

O desenvolvimento deste trabalho não ficaria completo sem agradecer a todos os que me ajudaram a tornar possível a realização do mesmo. Em primeiro lugar, quero agradecer ao meu orientador Professor Doutor António Pinto, pela sua disponibilidade, paciência, simpatia, ajuda e apoio na concretização deste projeto, sem a sua preciosa ajuda não seria possível. Os meus agradecimentos finais vão para a minha família, pelos incentivos ao longo da realização deste trabalho e, sobretudo, pela paciência e em especial à minha avó que antes de nos deixar me tinha dito que iria conseguir apoiando me sempre incondicionalmente.

Resumo

Este projeto foi desenvolvido no âmbito da disciplina de Projeto Final da Escola Superior de Tecnologia e Gestão de Felgueiras tendo como finalidade o desenvolvimento de uma aplicação para a gestão de condomínios que visa-se os objetivos propostos pelo Prof. Doutor António Pinto. Já existe algum software deste tipo mas é muito vocacionado para empresas de gestão de condomínios e não para condomínios que se querem gerir a eles próprios.

Pretende-se uma aplicação de fácil utilização, de interface apresentável e multi-plataforma. A capacidade de execução multi-plataforma, o principal requisito do cliente, levou a que a aplicação fosse desenvolvida na linguagem de programação Java com uma base de dados embebida. A base de dados a adotar não deveria implicar a instalação de software adicional no computador do cliente e deveria ser um simples ficheiro. Com os requisitos que foram propostos a aplicação para além de ter sido desenvolvida em Java com interface gráfica JavaFX, utilizou-se para o armazenamento dos dados sqlite e para a criação de relatórios foi usada a biblioteca JasperReports.

Dentro das várias funcionalidades a aplicação permite a inserção de condomínios, frações e rubricas. Permite também preparar um proposta de orçamento ou mesmo um orçamento definitivo em que este foi aprovado por todas as partes envolvidas. Ainda tem como funcionalidade gerar avisos de débito, gerar recibos, associar uma mapa de despesas a um determinado orçamento e por fim gerar relatórios finais com toda a informação relevante sobre um determinado orçamento.

Índice

1 – Introdução.....	1
2 – Ferramentas Utilizadas.....	3
2.2 – Netbeans 8.....	3
2.2.2 - SQLite Database Browser.....	6
2.2.3 - JavaFX Scene Builder 2.0.....	7
2.2.4 - iReport Designer.....	8
2.3 – Sumário.....	10
3 – Descrição do trabalho.....	11
3.1 – Funcionalidades.....	11
3.2 – Cálculo de mensalidades.....	12
3.3 – Modelo Conceptual.....	15
3.4 – Modelo Relacional.....	16
3.5 – Diagrama de Classes.....	17
4 – Avaliação do trabalho.....	18
4.1 – Operação com a base de dados.....	19
4.2 – Execução multi-plataforma.....	22
.....	25
4.3 – Conclusões.....	28
5 – Conclusões.....	29
5.1 – Trabalho futuro e dificuldades.....	30
7 – Referências Bibliográficas.....	31
8 – Anexos.....	32
Documento de requisitos.....	32
1 – Introdução.....	32
1.1 – Objetivo.....	32
1.2 – Âmbito.....	32
1.3 – Acrónimos.....	32
1.4 – Referências.....	32
1.5 – Visão geral do documento.....	33
2 – Descrição geral.....	33
2.1 – Contexto do produto.....	33
2.2 – Funções do produto.....	33
2.3 – Características de utilizador.....	34
2.4 – Restrições.....	35
2.5 – Pressupostos e dependências.....	35
3 – Requisitos específicos.....	35
3.1 – Requisitos de interfaces externos.....	35
3.1.1 – Interfaces com os utilizadores.....	35
3.2 – Requisitos funcionais.....	37
3.3 – Requisitos de informação.....	65
3.3.1- Diagrama de classes.....	65
3.3.2- Diagrama de classes – Condomínio.....	66
3.3.2- Diagrama de classes – Fração.....	67
3.3.2- Diagrama de classes – Rubrica.....	68
3.3.2- Diagrama de classes – Orçamento.....	69
3.3.2- Diagrama de classes – Aviso de Débito.....	70
3.3.2- Diagrama de classes – Recibos.....	71
3.3.2- Diagrama de classes – Despesas.....	72
3.3.2- Diagrama de classes – Ver documentos.....	73
3.4 – Requisitos não-funcionais.....	74
3.4.1. Disponibilidade.....	74
3.4.2. Eficiência.....	74
3.4.3. Manutenção.....	74

3.4.5. Segurança.....	74
3.4.6. Usabilidade.....	74

Índice de figuras

Figura 1: Interface principal do NetBeans.....	3
Figura 2: Inteligência do NetBeans.....	4
Figura 3: Visualização do projeto no NetBeans.....	4
Figura 4: Debugger do Netbeans.....	5
Figura 5: Base de Dados e Serviços no NetBeans.....	6
Figura 6: Interface principal do SQLite Database Browser.....	7
Figura 7: Interface principal do JavaFX Scene Builder 2.0.....	8
Figura 8: Interface principal do iReport 5.5.0.....	9
Figura 9: Ciclo de vida de um relatório na framework JasperReports.....	10
Figura 10: Modelo conceptual da Base de dados.....	15
Figura 11: Modelo Relacional da Base de Dados.....	16
Figura 12: Teste inserção de uma fração.....	19
Figura 13: Verificação de uma inserção na BD.....	20
Figura 14: Edição de uma fração.....	20
Figura 15: Verificação da edição dos campos na BD.....	21
Figura 16: Remoção de uma fração.....	21
Figura 17: Verificação da remoção na BD.....	22
Figura 18: Adição de um Orçamento em Windows 8.1 PRO.....	23
Figura 19: Adição de um Orçamento em Ubuntu 14.04 LTS.....	23
Figura 20: Primeiro passo na definição de um orçamento em Windows 8.1 Pro.....	24
Figura 21: Primeiro passo na definição de um orçamento em Ubuntu 14.04 LTS.....	24
Figura 22: Segundo passo na definição de um orçamento em Windows 8.1 Pro.....	25
Figura 23: Segundo passo na definição de um orçamento em Ubuntu 14.04 LTS.....	25
Figura 24: Terceiro passo na definição de um orçamento em Windows 8.1 Pro.....	26
Figura 25: Terceiro passo na definição de um orçamento em Ubuntu 14.04 LTS.....	26
Figura 26: Quarto passo na definição de um orçamento em Windows 8.1 Pro.....	27
Figura 27: Quarto passo na definição de um orçamento em Ubuntu 14.04 LTS.....	27
Figura 28: Esboço Interface principal da aplicação.....	35
Figura 29: Esboço relativo à gestão de frações.....	36
Figura 30: Esboço relativo à visualização dos recibos gerados.....	37
Figura 31: Diagrama casos de uso da aplicação.....	37
Figura 32: Diagrama de sequência - Criar Nova Base de Dados.....	38
Figura 33: Diagrama de sequência – Abrir Condomínio.....	39
Figura 34: Diagrama de sequência – Guardar Condomínio.....	40
Figura 35: Diagrama de sequência – Inserir Fração.....	41
Figura 36: Diagrama de sequência – Editar Fração.....	42
Figura 37: Diagrama de sequência – Remover Fração.....	43
Figura 38: Diagrama de sequência – Listar Fração.....	44
Figura 39: Diagrama de sequência – Inserir Rubrica.....	45
Figura 40: Diagrama de sequência – Remover Rubrica.....	46
Figura 41: Diagrama de sequência – Listar Rubrica.....	47
Figura 42: Diagrama de sequência – Inserir Orçamento.....	48
Figura 43: Diagrama de sequência – Editar Orçamento.....	49
Figura 44: Diagrama de sequência – Remover Orçamento.....	50
Figura 45: Diagrama de sequência – Listar Orçamento.....	51
Figura 46: Diagrama de sequência – Gerar avisos de débito.....	52
Figura 47: Diagrama de sequência – Ver avisos de débito.....	53
Figura 48: Diagrama de sequência – Filtrar avisos de débito.....	54
Figura 49: Diagrama de sequência – Enviar avisos de débito por email.....	55
Figura 50: Diagrama de sequência – Gerar recibos.....	56
Figura 51: Diagrama de sequência – Ver recibos.....	57
Figura 52: Diagrama de sequência – Filtrar recibos.....	58
Figura 53: Diagrama de sequência – Enviar recibos por email.....	59
Figura 54: Diagrama de sequência – Inserir Despesa.....	60
Figura 55: Diagrama de sequência – Editar Despesa.....	61

Figura 56: Diagrama de sequência – Remover Despesa.....	62
Figura 57: Diagrama de sequência – Listar Despesa.....	63
Figura 58: Diagrama de sequência – Ver Documentos.....	64
Figura 59: Diagrama de classes - Condomínio.....	66
Figura 60: Diagrama de classes - Fração.....	67
Figura 61: Diagrama de classes - Rubrica.....	68
Figura 62: Diagrama de classes - Orçamento.....	69
Figura 63: Diagrama de classes - Aviso de Débito.....	70
Figura 64: Diagrama de classes - Recibos.....	71
Figura 65: Diagrama de classes - Despesas.....	72
Figura 66: Diagrama de classes - Ver Documentos Finais.....	73

Índice de tabelas

Tabela 1: A pernilagem de uma fração.....	12
Tabela 2: O montante por Rubrica.....	13
Tabela 3: As rubricas de uma fração.....	13
Tabela 4: Algoritmo do cálculo mensalidades.....	14

Acrónimos

IDE- Integrated development environment, ambiente de desenvolvimento integrado.

SQL- Structured Query Language, linguagem de consulta estruturada para base de dados relacional, em que esta é inspirada em bases de álgebra relacional.

API- Application Programming Interface, (interface de programação para aplicações) que disponibiliza um conjunto de classes e métodos que permitem ser usados por programadores sem saber os detalhes da sua implementação mas sim sabendo só o que fazem.

UML- Unified Modeling Language, linguagem de modelação unificada.

Definições

query- consulta na base de dados.

open-source- código fonte aberto.

javafx- tecnologia Java para o desenvolvimento de aplicações desktop com interface gráfica avançada.

swing- tecnologia Java para o desenvolvimento de aplicações desktop com interface gráfica simples.

background – algo informaticamente a correr em modo silencioso e invisível para o utilizador.

1 – Introdução

Este trabalho consistiu no desenvolvimento de uma aplicação para a gestão de condomínios que fosse ao encontro das necessidades do Prof. Doutor António Pinto. Existe algum software deste tipo mas é muito vocacionado para empresas de gestão de condomínios e não para condomínios que se querem gerir a eles próprios.

Adicionalmente, pedia-se uma aplicação de fácil utilização, de interface apresentável e multi-plataforma. A capacidade de execução multi-plataforma, o principal requisito do cliente, levou a que a aplicação fosse desenvolvida na linguagem de programação Java com uma base de dados embebida. A base de dados a adotar não deveria implicar a instalação de software adicional no computador do cliente e deveria ser um simples ficheiro. Adicionalmente, a aplicação deveria permitir pelo menos as seguintes funcionalidades:

1. Inserir, editar, remover e listar frações, rubricas e um condomínio.
2. Utilizar-se uma base de dados para cada condomínio.
3. Calcular as mensalidades para cada fração para um determinado orçamento.
4. Gerar avisos de débito para as frações pagantes, gerando o respetivo aviso em PDF e permitir o seu envio por email.
5. Gerar recibo para os vários débitos, gerando o respetivo PDF e permitir o seu envio por email.
6. Criar relatórios finais para cada orçamento.

Adicionalmente, deverão ser possibilitadas as seguintes funcionalidades:

1. Criar e definir todas as metas necessárias para o preenchimento de um Orçamento, calculando no final as mensalidades a pagar por cada fração. A aplicação permite definir se o orçamento em causa é definitivo ou é apenas uma proposta a apresentar a todos os condóminos que posteriormente poderá passar a definitivo caso seja aceite por todos os intervenientes.
2. Gerar recibos em que um recibo pode ser relativo a um ou vários débitos para uma e só uma fração. Posteriormente a aplicação permite ver todos os recibos gerados e também enviá-los por email.

O Capítulo 2 descreve as principais ferramentas utilizadas para o desenvolvimento do projeto. O Capítulo 3 descreve todo o trabalho elaborado, as principais funcionalidades da aplicação, descreve como foi desenvolvida a funcionalidade de cálculo das mensalidades, apresenta o diagrama conceptual da base de dados, mas também o diagrama relacional da mesma. O Capítulo 4 apresenta

os testes e a metodologia adotada. O Capítulo 5 conclui o presente relatório, indicando as principais dificuldades e apresentado possíveis melhoramentos futuros. Em anexo, pode-se encontrar o documento de requisitos que descreve os requisitos funcionais e não funcionais da aplicação, o diagrama de caso de uso, os respectivos diagramas de sequência para cada caso de uso e os vários diagramas de classe.

2 – Ferramentas Utilizadas

Ao longo do desenvolvimento do projeto foram utilizadas 4 ferramentas: 1) o Netbeans IDE; 2) o SQLite Database Browser; 3) o JavaFX Scene Builder; 4) o iReport Designer. Estas aplicações serão descritas nas secções seguintes, incluindo os objetivos de cada uma, as suas principais funcionalidades e em que situações foram utilizadas no decorrer do projeto.

2.2 – Netbeans 8

O NetBeans IDE é uma ferramenta que permite desenvolver aplicações em Java, móveis e Web, aplicações com XML, HTML, Groovy, Javadoc, JavaScript JSP e CSS sendo que oferece também ferramentas para programadores de PHP e C/C++. Por ser gratuito e open-source é utilizado por uma grande parte de programadores de todo o mundo e foram essas as principais razões para a utilização do mesmo. Mas também foi utilizado porque a linguagem utilizada no desenvolvimento desta aplicação foi Java e o NetBeans em termos da componente Java antecipa -se em primeira mão a todas as novidades que vão surgindo dentro do tema.

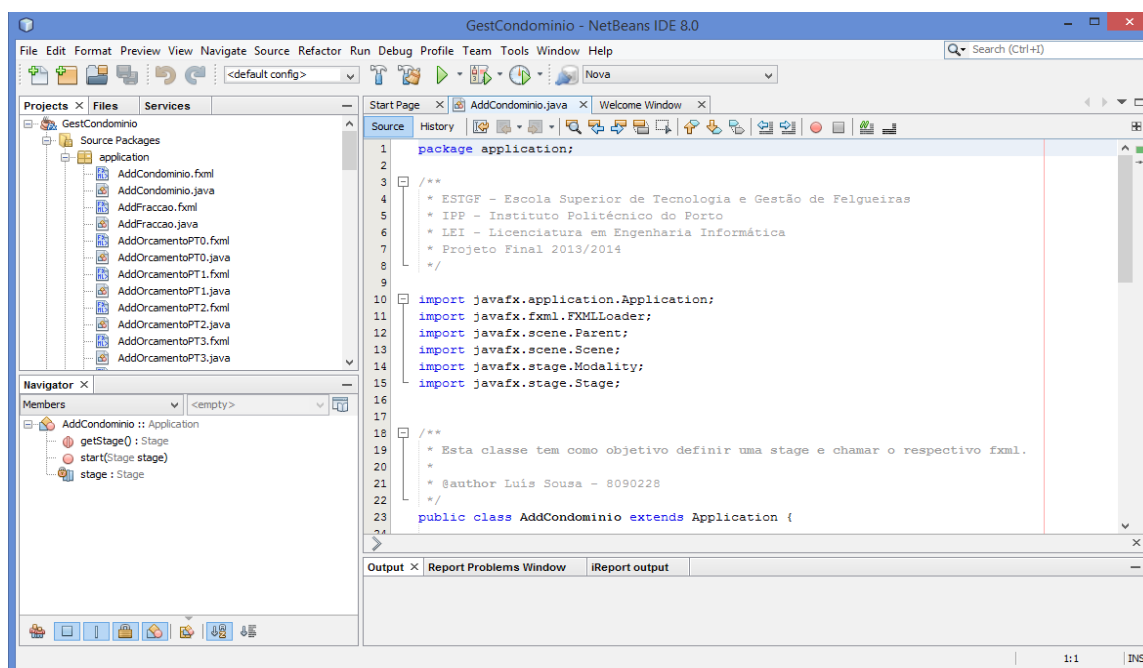


Figura 1: Interface principal do NetBeans

A Figura 1 representa a interface principal do NetBeans IDE 8.0 que permite assim de uma forma global mostrar as capacidades do mesmo.

Edição de Códigos Rápida e Inteligente

Destaca -se pela forma como interage com o utilizador em que apresenta sugestões conforme o utilizador vai inserindo código, recua linhas se necessário, associa palavras e realça código semanticamente e também sintaticamente. Em Java ajuda a fazer alguns métodos, sendo possível através de apenas alguns clicks, adicionar métodos construtores de uma classe e os métodos de acesso gets e sets. É uma ferramenta extensível, apesar do suporte por defeito para várias linguagens de programação podemos ainda adicionar mais linguagens.[4]

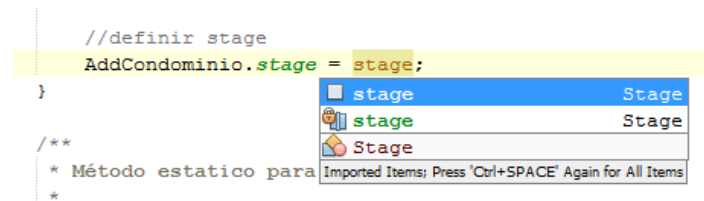


Figura 2: Inteligência do NetBeans

A Figura 2 mostra como o NetBeans influencia na altura de inserir código, em que através de teclas de atalho apresenta sugestões ao utilizador de código que poderá estar a precisar naquele momento.

Gestão de Projetos Fácil e Eficiente

O NetBeans IDE oferece diferentes vistas de dados, várias janelas do projeto e ferramentas para configurar aplicações e geri-las. Permite compreender toda a estrutura da aplicação em causa, pois o código estará organizado para melhor compreensão do utilizador.

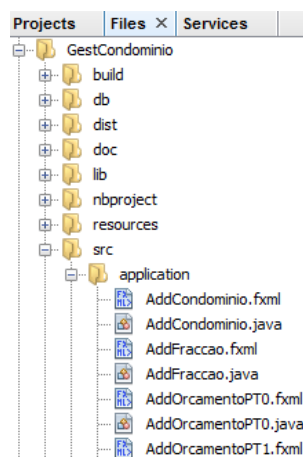


Figura 3: Visualização do projeto no NetBeans

A Figura 3 permite compreender como os ficheiros da aplicação são organizados.

Códigos sem Erros

O debugger(depurador) do NetBeans permite colocar breakpoints (pontos de interrupção) no código-fonte, inspecionar um determinado atributo, ver o estado de uma variável, as threads em execução e o seu estado, tirar fotos e acompanhar a execução de toda a aplicação para podermos verificar onde o código está com problemas. Permite ainda anexar o debugger a um processo que já esteja execução. O NetBeans inclui ainda um debugger visual para tirar fotos da interface e explorar visualmente aplicações em JavaFX ou em Swing.

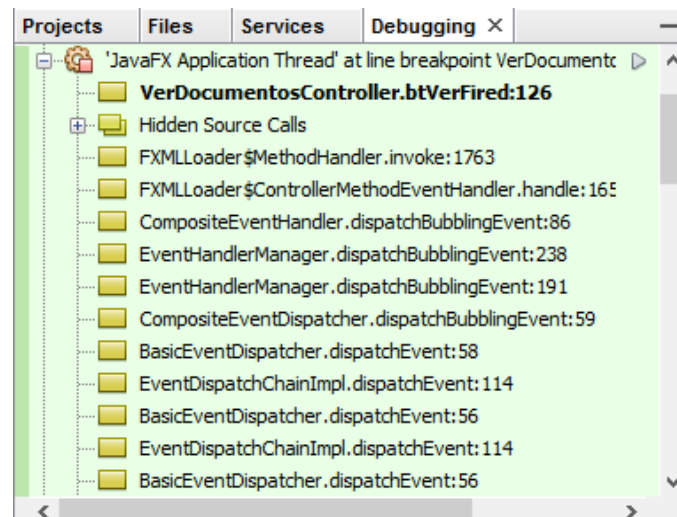


Figura 4: Debugger do Netbeans

A Figura 4 mostra o que é possível inspecionar com o NetBeans em modo de debugger. Neste caso verifica-se que threads estão em execução.

Suporte Multi-plataforma

É possível instalar o NetBeans IDE em todos os sistemas operativos que suportem Java, desde sistemas Windows, Linux a Mac OS porque até o próprio NetBeans é escrito em Java. O NetBeans além de oferecer suporte para utilizadores de C/C++, PHP e Java tem editores e ferramentas para XML, HTML, Groovy, Javadoc, JavaScript entre outras.

Databases e Serviços

A janela Serviços permite aceder a muitos recursos auxiliares, tais como base de dados, servidores e serviços web, permitindo parar e iniciar base de dados e servidores diretamente no NetBeans e assim listar, adicionar, remover e modificar diretamente os dados.

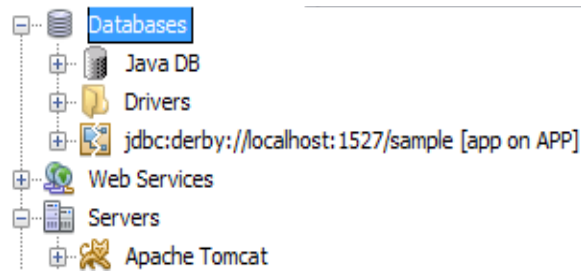


Figura 5: Base de Dados e Serviços no NetBeans

A Figura 5 mostra como o NetBeans agrupa as funcionalidade das Base de Dados, servidores e até de Web Services.

Plugins

O NetBeans oferece variedade de plugins. Tem uma funcionalidade, o Plugin Manager que permite adicionar, remover ou atualizar plugins. Estão disponíveis plugins para todos os tipos de desenvolvimento, desde HTML5, PHP , C/C++ ou até mesmo Java que é o que interessa neste caso. Permite ainda aos utilizadores desenvolver novos plugins porque o NetBeans é extensível e facilita o trabalho do programador.

De forma a complementar a nossa informação sobre plugins do netbeans podemos consultar o livro referenciado na bibliografia.

2.2.2 - SQLite Database Browser

Base de dados SQLite Browser é uma ferramenta visual utilizada para criar e editar ficheiros de base de dados compatíveis com o SQLite. Sua interface simples destina-se a ser utilizada por programadores que pretendam criar base de dados SQL sem a necessidade de aprender comandos SQL complicados permitindo assim a qualquer utilizador sem conhecimentos avançados utilizar a ferramenta sem complicações, sendo essa uma das principais razões da sua utilização no desenvolvimento desta aplicação. A ferramenta contém as seguintes características[5]:

- Inserir, listar, editar e remover tabelas e ver toda a estrutura da base de dados..
- Criar, definir e remover índices.
- Procurar, editar, adicionar e excluir dados.
- Fazer pesquisas nas tabelas.
- Importar e exportar os dados para ficheiros de texto.
- Importar e exportar tabelas para ficheiros do tipo CSV.

- Criar definir e remover vistas.
- Executar comandos SQL.

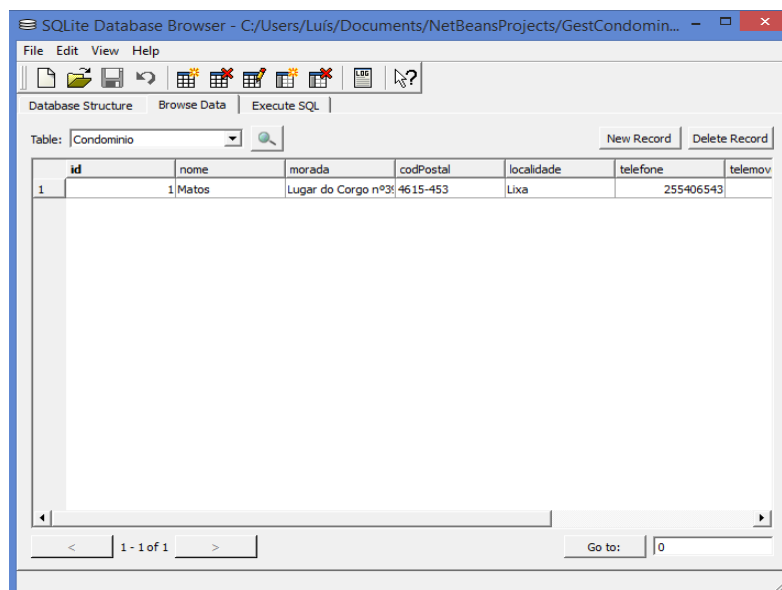


Figura 6: Interface principal do SQLite Database Browser

A Figura 6 representa a interface principal do SQLite Database Browser que permite assim de uma forma global mostrar as capacidades do mesmo.

2.2.3 - JavaFX Scene Builder 2.0

JavaFX Scene Builder é uma ferramenta de layout que permite aos utilizadores criarem interfaces de aplicações com a tecnologia JavaFX, sem codificação. Permite arrastar e soltar os componentes para uma área de trabalho, modificar as suas propriedades, aplicar folhas de estilo (css) , e o código FXML para o layout é gerado automaticamente em segundo plano. O resultado é um ficheiro FXML que pode ser combinado com um projeto Java, servindo de “ponte” entre a interface apresentada ao utilizador e a lógica da aplicação[6].

FXML é uma linguagem de marcação baseada em XML que permite definir a interface de uma aplicação separada da lógica de programação. Fornece a possibilidade de pré-visualizar a interface que está a ser desenvolvida, permite ainda aplicar folhas de estilo para sofisticar a aparência, inserir animações e efeitos. Contém assim condições para a criação de interfaces de certo modo avançadas[6].

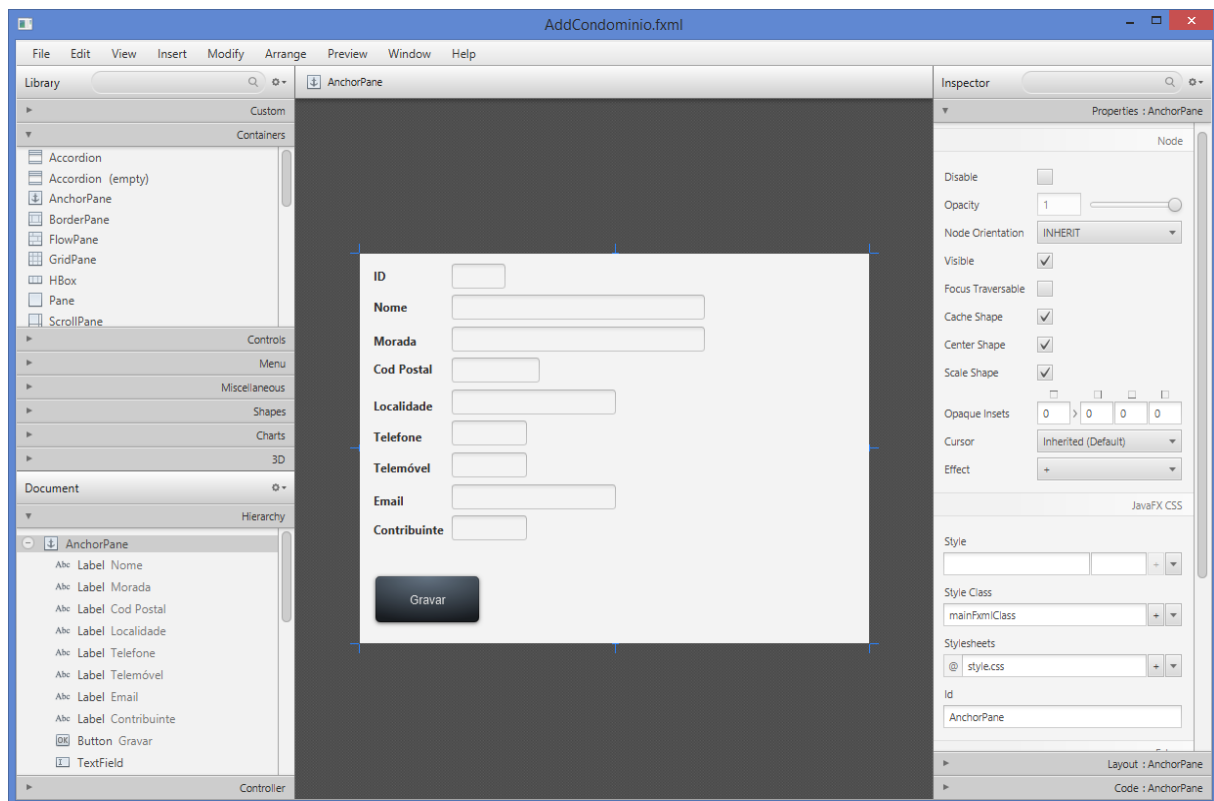


Figura 7: Interface principal do JavaFX Scene Builder 2.0

A Figura 7 mostra como é disponibilizada ao utilizador a interface principal do JavaFX Scene Builder 2.0 e de uma forma global mostra as capacidades do mesmo.

2.2.4 - iReport Designer

iReport Designer é um ambiente para criação de relatórios. Utiliza a framework JasperReports (consultar [2]) sendo que iReport Designer permite criar relatórios e a framework JasperReports permite executá-los e gerar a saída numa aplicação Java. Os relatórios podem ser projetados a partir do zero ou a partir de um dos muitos modelos já pré-feitos prontos para serem usados. iReport Designer ajuda durante todas as fases do desenvolvimento dos relatórios: design, compilação, execução de relatório e de exportação ou de visualização de documentos, sendo que estes pontos foram importantíssimos para a utilização do mesmo no desenvolvimento da aplicação me causa.

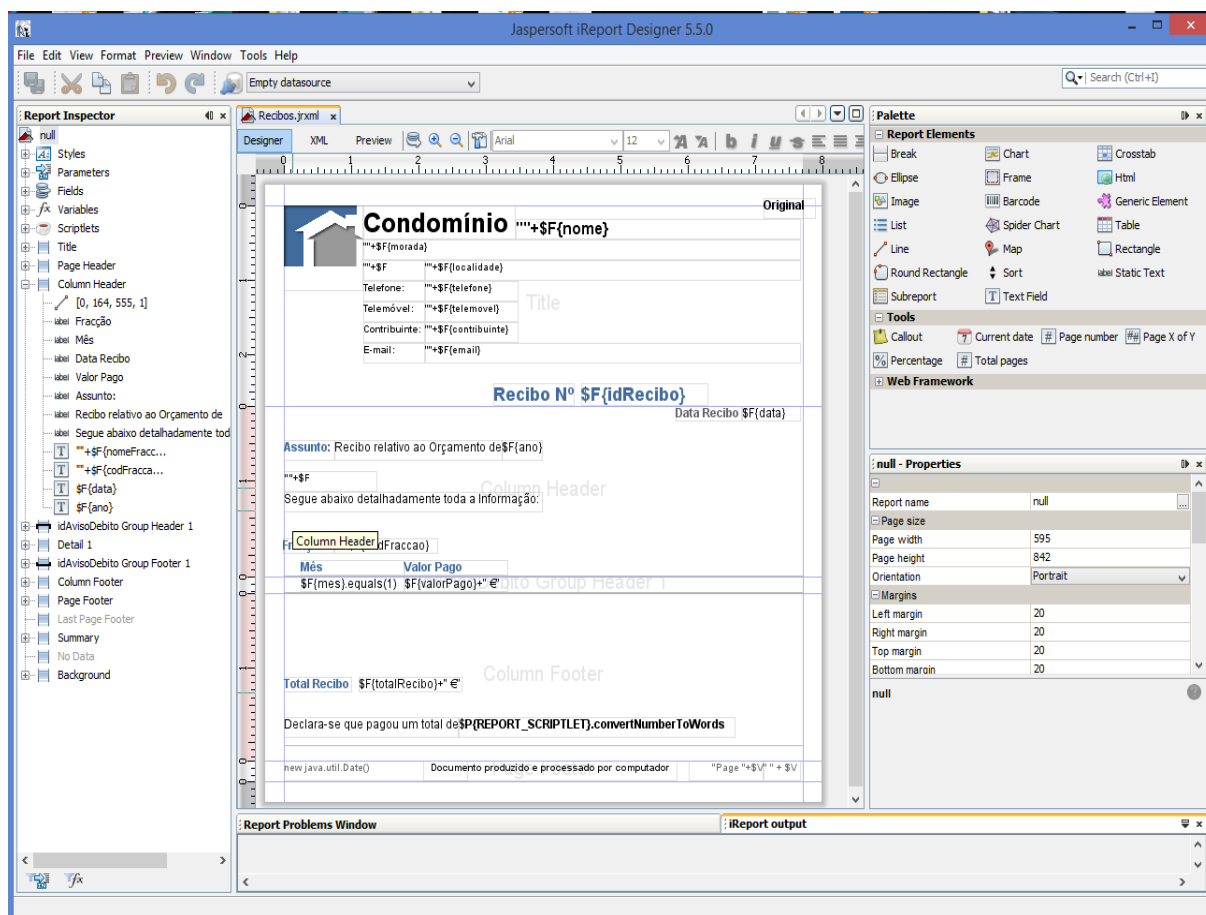


Figura 8: Interface principal do iReport 5.5.0

A Figura 8 representa a interface principal do iReport Designer, mostrando o acesso às suas principais funcionalidades e à configuração e moldagem visual de um relatório.

Aquando da criação de um relatório com iReport Designer estamos a criar um ficheiro JRXML, que é um documento XML que contém a definição do layout do relatório. Antes de executar um relatório, o JRXML deve ser compilado e transformado num objeto binário, o ficheiro JASPER. Esta compilação é feito por razões de desempenho. O ficheiro Jasper é necessário na aplicação desenvolvida para executar os demais relatórios [1]. A Figura 9 é a representação do que foi dito.

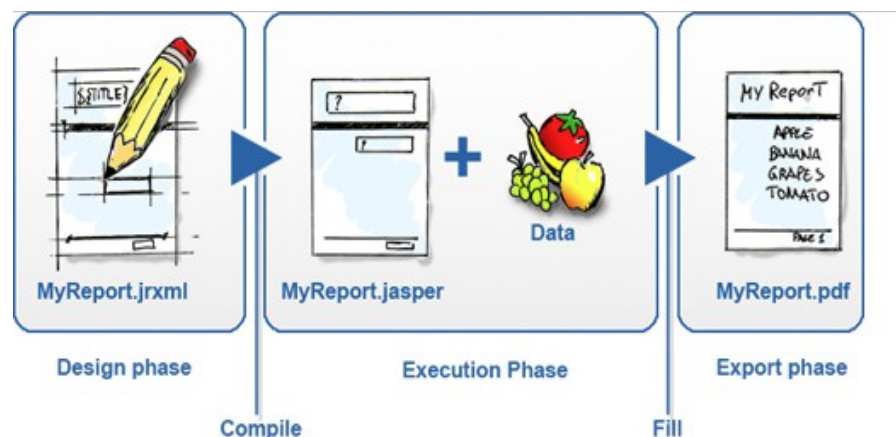


Figura 9: Ciclo de vida de um relatório na framework JasperReports

Fonte ¹

2.3 – Sumário

Todas as ferramentas contribuíram à sua maneira e conforme a sua utilidade, de uma forma positiva para o desenvolvimento da aplicação. Sendo o principal destaque para o NetBeans que realmente sem margem para dúvidas é uma ferramenta acima de tudo funcional, que permite a qualquer programador usufruir de um ambiente cómodo, bem organizado, de fácil compreensão, utilizado para várias linguagens de programação, em que neste caso foi utilizado na vertente da linguagem Java.

O SQLite Database Browser foi também um programa essencial na verificação e visualização dos dados que estavam a ser inseridos na base de dados SQL, permitindo verificar se os métodos desenvolvidos estavam a ter os resultados esperados conforme a lógica de programação, permitindo também editar, apagar, inserir e remover diretamente na base de dados, sendo que por vezes foi muito útil também nesse capítulo e também para testar as queries antes de as utilizar na camada de programação.

Tanto o SQLite Database Browser como o NetBeans foram os mais determinantes, mas o JavaFX Scene Builder e o iReportDesigner também foram determinantes não só em programação, pois não era essa a principal funcionalidade, mas em termos de designer foram cruciais. O JavaFX Scene Builder foi importante para todo o desenho da parte gráfica da aplicação em si e o iReport designer para manipular e “desenhar” os relatórios necessários.

¹ <http://community.jaspersoft.com/wiki/ireport-designer-getting-started>

3 – Descrição do trabalho

Nesta parte do relatório descreve-se as principais funcionalidades implementadas e os respetivos diagramas da base de dados tanto o conceptual como o relacional. Para um mais completo esclarecimento do funcionamento da aplicação, ler o Documento de requisitos . Também será devidamente explicado como funciona o cálculo das mensalidades de cada fração e apresentado o algoritmo desenvolvido.

3.1 – Funcionalidades

Esta aplicação possui várias funcionalidades das quais podemos destacar:

- **Criação de múltiplas base de dados.** Esta foi uma das funcionalidades pedidas e que foi concretizada. Foi elaborada com o pressuposto de permitir que se crie uma base de dados para cada condomínio a gerir.
- **Abrir sempre a última base de dados.** Permite que quando se inicia a aplicação seja aberto sempre o último condomínio utilizado, permitindo assim ao utilizador acabar o trabalho que ficou por fazer.
- **Criar e editar um Condomínio.** Para cada base de dados está associado um Condomínio, em que cada condomínio depois de inserido também pode ser editado.
- **Inserir, editar, listar e remover frações.** Um condomínio pode ter de uma a várias frações, sendo assim como era esperado a aplicação permite inserir, editar, listar e remover frações.
- **Gestão de rubricas.** Em cada Orçamento, as frações têm rubricas associadas e por consequentemente o orçamento fica ligado a essas rubricas, por isso a aplicação permite inserir e eliminar rubricas.
- **Definir um Orçamento ou proposta e calcular as mensalidades.** Esta é uma funcionalidade crucial pois na altura de inserção de um orçamento definimos um estado, esse estado pode tomar o valor de definitivo ou proposta. Se for definitivo quer dizer que vai ser aquele orçamento utilizado para gerir o condomínio durante um ano, sendo que foi acordado por todas as partes envolvidas. Se for do tipo proposta como o próprio nome indica ainda não está decidida a sua utilização, devendo ser apresentado por exemplo numa reunião de condomínio para se definir a sua utilização ou não. Permite ainda associar rubricas a um orçamento, editar e remover caso seja necessário, permite especificar que frações pagam naquele orçamento, que rubricas estão associadas a cada fração, inserir, editar e remover observações associadas a uma determinada fração e por fim calcular as respetivas mensalidades para cada fração através da seguinte **fórmula**:

$(\text{valor rubrica} \div \text{soma permilagens que pagam a rubrica}) \times \text{permilagem da fracção}$
Fórmula 1: Cálculo mensalidade de uma fracção

- **Gerar avisos de débito.** A aplicação permite gerar avisos de débito, para uma ou várias fracções, para um ou mais meses relativamente a um Orçamento.
- **Listar, filtrar, imprimir ou enviar por email os avisos de débito.** Depois dos avisos de débitos gerados para um determinado orçamento a aplicação permite ao utilizador criar um documento PDF com os dados do aviso de débito gerado, permitindo assim imprimir o PDF ou enviá-lo por email para a fracção correspondente. Permite ainda listar todos os avisos de débito numa só interface ou filtrar por fracção, auxiliando o utilizador na hora de enviar por email.
- **Listar, filtrar, imprimir ou enviar por email recibos.** Depois dos avisos de débitos podemos gerar recibos para os débitos que foram gerados em relação a um determinado orçamento. A aplicação cria automaticamente um documento PDF com os dados de um recibo, sendo que um recibo poderá estar ligado a um ou vários débitos de uma só fracção. Permite ainda que esse PDF relativamente ao recibo seja impresso para entregar à devida pessoa ou enviado por email. Existe também a possibilidade de listar todos os recibos já gerados numa só interface ou filtrar recibos por fracção ajudando o utilizador na hora de pesquisar por um recibo específico.
- **Apresentar documentos finais** para um determinado orçamento. Gera relatórios com as contas finais, pagamentos realizados para cada fracção, despesas efetuadas pelo condomínio, saldo atual para cada fracção, entre outros documentos.

3.2 – Cálculo de mensalidades

Exemplo teórico prático em que um condomínio tem a seguinte distribuição para um determinado orçamento :

Fracção	Permilagem
A	400
B	600

Tabela 1: A permilagem de uma fracção

A Tabela Tabela 1 associa uma permilagem para cada fracção.

Rubrica	Valor
Água	1000 €
Luz	500 €
Total 1:	1500 €
Fundo comum de Reserva 10%	$1500 * 0.10 = 150 \text{ €}$
Total:	1650 €

Tabela 2: O montante por Rubrica

A Tabela Tabela 2 associa para cada rubrica o valor a pagar.

O passo seguinte passa por definir para cada fração que rubricas pagam como se pode verificar na Tabela 3.

Fração	Rubrica(s)
A	Água , Fundo comum de reserva
B	Água, Luz , Fundo comum de reserva

Tabela 3: As rubricas de uma fração

A fração A só paga, para este exemplo de orçamento, a água, enquanto que a fração B paga a água e a luz. A rubrica fundo comum de reserva é obrigatoriamente paga por todas as frações pagantes. Agora passa-se aos cálculos das mensalidades para as frações pagantes A e B, recorrendo à **Fórmula 1: Cálculo mensalidade de uma fração** para o cálculo das mensalidades.

Cálculo para a fração **A**, rubrica **água**: $(1000 / 1000) * 400 = 400 \text{ €}$ anuais, ou seja $400 / 12 \approx 33.33 \text{ €}$ mensais para esta rubrica. Cálculo para a fração **A** rubrica **fundo comum de reserva 10%**: $(150 / 1000) * 400 = 60 \text{ €}$ anuais, ou seja $60 / 12 = 5 \text{ €}$ mensais para esta rubrica. O **total das rubricas** para a **fração A**, ou seja o valor total a pagar, é obtido pela soma das rubricas, o que dá neste exemplo $400 + 60 = 460 \text{ €}$ anuais, ou seja $33.33 + 5 = 38.33 \text{ €}$ mensais que é igual a $460 / 12$.

Cálculo para a fração **B** rubrica **água**: $(1000 / 1000) * 600 = 600 \text{ €}$ anuais, ou seja $600 / 12 = 50 \text{ €}$ mensais. Cálculo para a fração **B** rubrica **Luz**: $(500 / 600) * 600 = 500 \text{ €}$ anuais, $500 / 12 \approx 41.67 \text{ €}$ mensais. Cálculo para a fração **B** rubrica **fundo comum de reserva 10%**: $(150 / 1000) * 600 = 90 \text{ €}$ anuais, $90 / 12 = 7.50 \text{ €}$ mensais para esta rubrica. Agora soma-se o **total das rubricas** para a **fração B**: $600 + 500 + 90 = 1190 \text{ €}$ anuais, $50 + 41.67 + 7.50 = 99.17$ que é igual $1190 / 12$.

Somando os 1190 € anuais da fração B com os 460 € anuais da fração A, obtém-se os 1650 €, sendo este o total a pagar em rubricas para este orçamento.

Algoritmo desenvolvido

```
public void calcAndSaveResults(int idO) throws ClassNotFoundException, SQLException {
    int ano = dao.getOrcamento(idO).getAno();
    ArrayList<Fracao> fraccoesPagantes = new ArrayList<Fracao>();
    ArrayList<Fracao> fraccoesRubrica = new ArrayList<Fracao>();
    fraccoesPagantes = getAllFraccoesPagantes(idO);
    String codPagante;
    String rubrica;
    float resultado = 0;
    float valorRubrica;
    float somaPermilagemRubrica = 0; // soma das permilagens de uma rubrica
    ArrayList<RubricaOrcamento> rubricas = new ArrayList<RubricaOrcamento>();

    for (Fracao f2 : fraccoesPagantes) {
        System.out.println("Fracao a tratar" + f2.getCod());
        resultado = 0;
        System.out.println("Resultado Inicial" + resultado);
        codPagante = f2.getCod(); // fracao que vai ser usada para os calculos
        rubricas = this.getRubricas(codPagante, idO); // rubricas para a fracao
        for (RubricaOrcamento rub : rubricas) { // percorrer rubricas da fracao pagante
            rubrica = rub.getRubrica(); // rubrica a tratar
            System.out.println("Rubrica a tratar" + rubrica);
            valorRubrica = rub.getValor();
            System.out.println("Valor rubrica" + valorRubrica);
            somaPermilagemRubrica = 0;
            System.out.println("Soma Permilagem Inicial" + somaPermilagemRubrica);
            fraccoesRubrica = this.getFraccoes(rubrica, idO); // fraccoes k tem aquela rubrica
            for (Fracao fracao1 : fraccoesRubrica) { // percorrer fraccoes com aquela rubrica
                somaPermilagemRubrica = somaPermilagemRubrica + fracao1.getPermilagem();
                System.out.println("Soma Permilagem" + somaPermilagemRubrica);
            }
            System.out.println("Que resultado chegou aqui" + resultado);
            resultado = resultado + ((valorRubrica / somaPermilagemRubrica) * f2.getPermilagem());
            System.out.println("Resultado na rubrica " + rub.getRubrica() + resultado);
        }
        System.out.println("O resultado da fracao" + f2.getCod() + "é:" + resultado);
        fracaoQuotasOrcamento = new FracaoQuotasOrcamento();

        fracaoQuotasOrcamento.setFracao(f2.getCod());
        fracaoQuotasOrcamento.setNome(f2.getNome());
        fracaoQuotasOrcamento.setPerm(f2.getPermilagem());
        fracaoQuotasOrcamento.setMensal(round(resultado / 12, 2));
        fracaoQuotasOrcamento.setAnual(round(resultado, 2));
        fracaoQuotasOrcamento.setAno(ano);
        fracaoQuotasOrcamento.setIdOrcamento(idO);
        fraccoesQuotasOrcamento.add(fracaoQuotasOrcamento);
        dao.adicionar(fracaoQuotasOrcamento);
    }
}
```

Tabela 4: Algoritmo do cálculo mensalidades

O algoritmo ilustrado na Tabela 4 percorre todas as frações pagantes daquele orçamento. Depois verifica que rubricas estão associadas a cada fração pagante. Depois para cada rubrica verifica que frações têm aquela rubrica e depois aplica a Fórmula 1: Cálculo mensalidade de uma fração e assim sucessivamente até percorrer todas as rubricas de cada fração. Depois passa-se para

a fração pagante seguinte e repete -se todo o processo.

3.3 – Modelo Conceptual

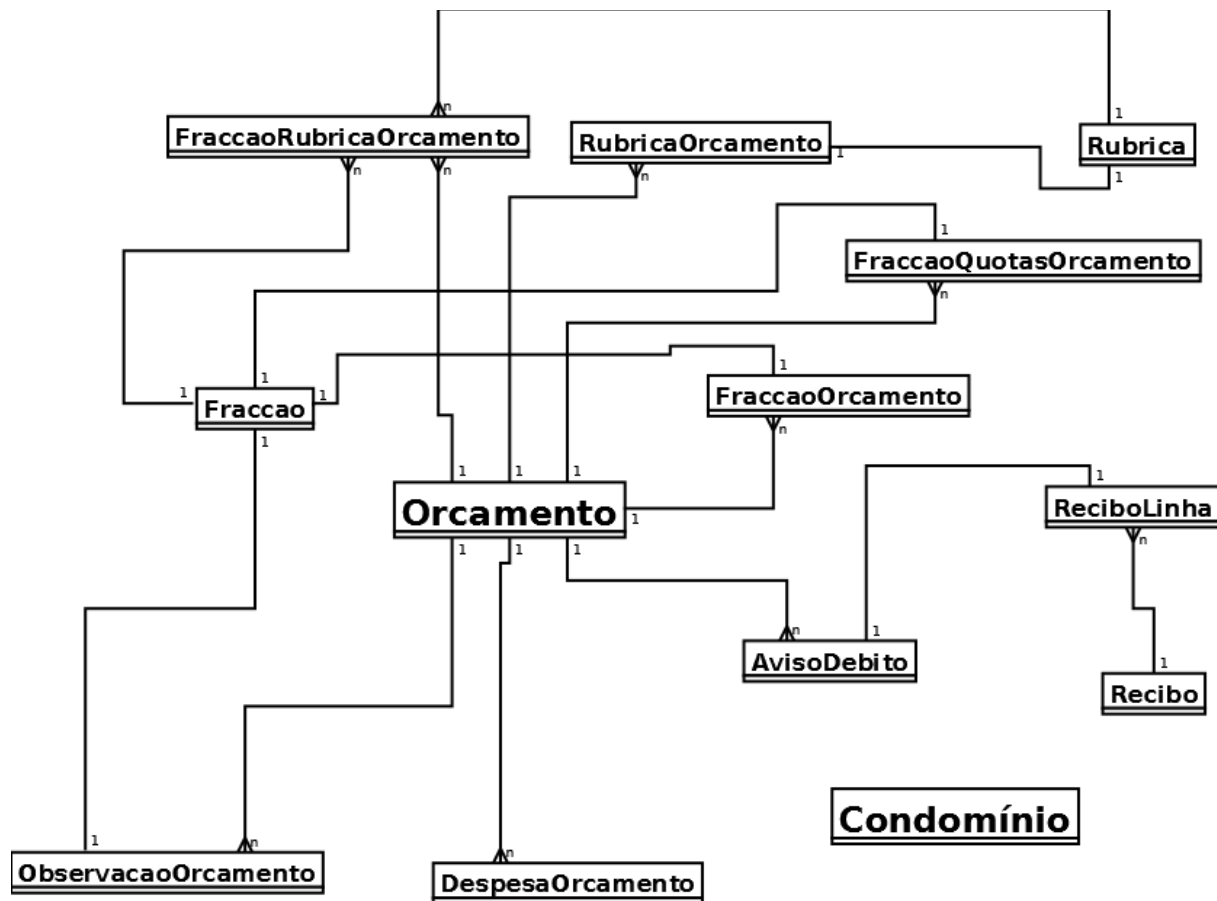


Figura 10: Modelo conceptual da Base de dados

Na Figura 10 temos o modelo conceptual da base de dados da aplicação, representado com alto nível de abstração, pois a modelação no modelo conceptual é feita de forma mais natural independentemente da BD. A relação condomínio não tem ligação com outras relações, porque um dos pedidos feitos foi que para cada condomínio fosse criada uma nova base de dados, logo todos os dados inseridos na base de dados incidem sempre sobre o mesmo condomínio, não implicando à identificação do condomínio nas outras relações.

3.4 – Modelo Relacional

Depois de ter sido feito o modelo conceptual, foi feito o modelo relacional da base de dados.

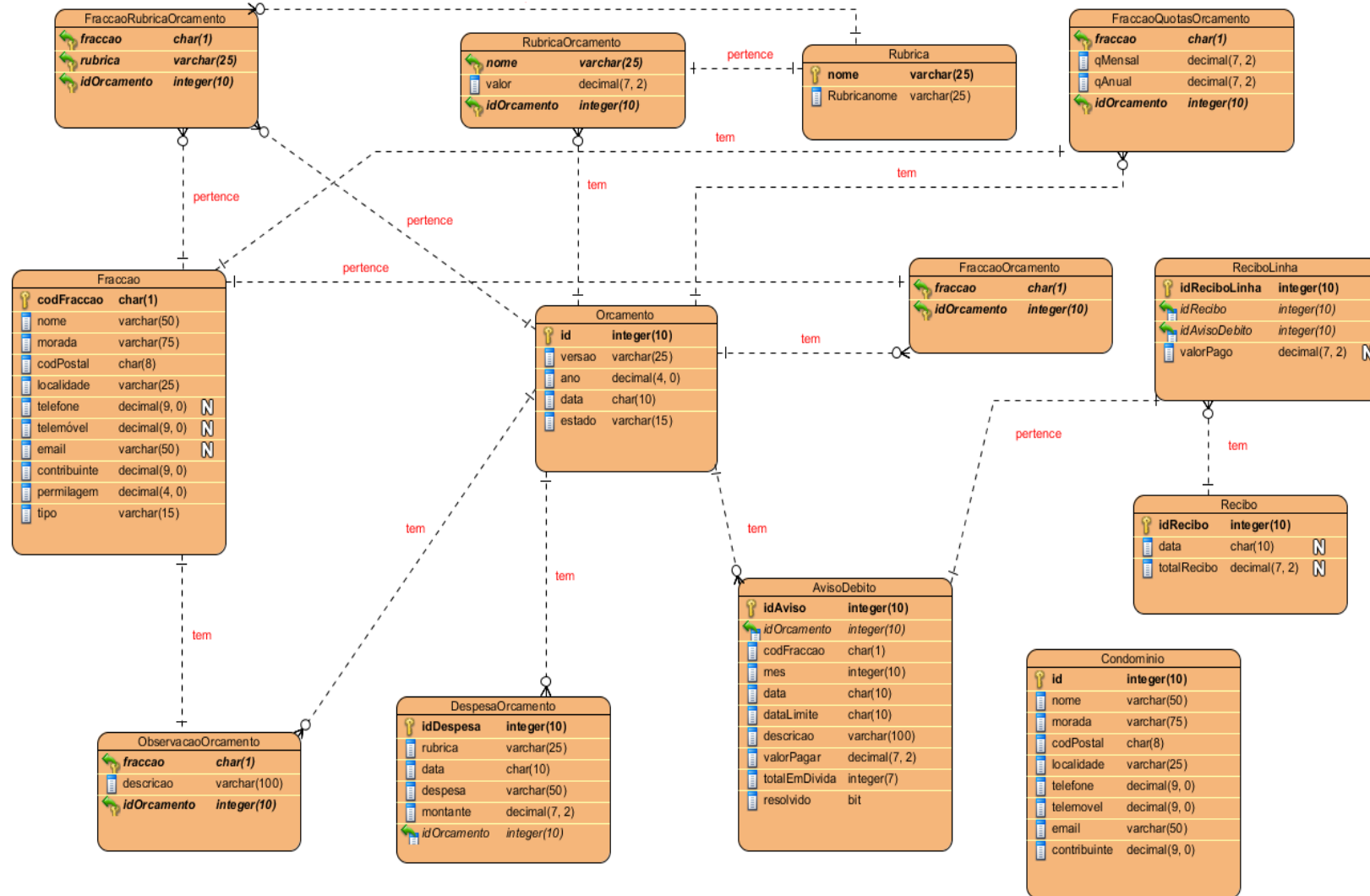


Figura 11: Modelo Relacional da Base de Dados

A Figura 11 ilustra o modelo relacional da base de dados da aplicação. Todas as relações existentes são apresentadas e descritas com o seu grau de cardinalidade.

3.5 – Diagrama de Classes

O diagrama ilustra a estrutura de classes do projeto e as iterações entres elas. Este diagrama procura focar apenas na estrutura de classes utilizada pelo projeto e não entra em muitos detalhes sobre atributos e operações (métodos) das mesmas. Consultar o final do documento, no capítulo 8 – Anexos, na parte 3.3.1- Diagrama de classes pois contém vários diagramas de classes da aplicação divididos por partes para melhor compreensão.

4 – Avaliação do trabalho

Nesta parte do documento é descrita a metodologia usada no desenvolvimento do trabalho e explicam-se os diferentes tipos de testes que foram efetuados à aplicação, desde testes sobre operações à base de dados a testes multi-plataforma para verificar se a aplicação consegue manter a estabilidade e compatibilidade de funcionalidades em várias plataformas, um requisito essencial na elaboração desta aplicação. Para os testes foram utilizados 2 sistemas operativos: Windows 8.1 e Ubuntu 14.04 LTS ambos de 64 bits, afim de verificar o comportamento da aplicação. Não foi adotada nenhuma metodologia específica, sendo que o projeto foi dividido em 4 fases com atividades distintas.

Na **primeira fase**, fase de **inicialização**, foi feito o levantamento de requisitos funcionais para definir as funcionalidades que o sistema deve suportar, em que temos um conjunto de entradas, que geram comportamento e saídas. Depois foi feito o levantamento de requisitos não-funcionais que são requisitos mais ligados ao uso da aplicação em termos de segurança, desempenho, sendo requisitos que surgem conforme a necessidade do utilizador. Depois seguiu-se a elaboração dos diagramas UML necessários, nomeadamente de classes e base de dados.

Na **segunda fase**, fase de **preparação**, foi escolhida a base de dados que melhor se adaptava ao desenvolvimento da aplicação. Depois de feitos todos os diagramas UML e de reuniões sobre os requisitos funcionais e não funcionais e restrições da aplicação foi pensada e definida a melhor maneira de organizar os dados e em que tipo de base de dados.

Na **terceira fase**, fase de **desenvolvimento**, como o próprio nome indica, foi quando se começou a programar todas as funcionalidades da aplicação. Em que durante uma semana desenvolvia-se alguma ou algumas funcionalidades da aplicação, depois havia uma reunião com o professor doutor António Pinto sobre a ou as funcionalidades que foram desenvolvida(s), debatendo o que estava bem e o que se poderia melhorar e definindo novos progressos até a reunião semanal seguinte, e assim sucessivamente até à fase final.

Na **quarta fase**, fase de **conclusão**, desenvolveram-se os testes finais sobre a aplicação, corrigindo eventuais falhas não detetadas até ao momento e principalmente se a aplicação desenvolvida funciona em vários sistemas operativos. Foram realizados alguns testes em torno da aplicação, principalmente à base de dados. A fase de testes foi dividida em duas partes. Na **primeira parte** foi verificado se os dados estavam a ser bem introduzidos na base de dados, se estavam a ser guardados de forma correta e conforme os expectável, sendo esta a primeira parte de testes utilizada. Na **segunda parte** foram realizados testes para ver se a aplicação funcionaria corretamente em diferentes sistemas operativos, sendo esta uma parte importantíssima na elaboração do trabalho.

4.1 – Operação com a base de dados

Nesta primeira fase de testes foi verificado se os dados estavam a ser bem introduzidos na base de dados, se estavam a ser guardados de forma correta e conforme os expectável, sem fugas de informação, respeitando também a integridade dos dados e sua lógica.

A Figura 12 apresenta um teste feito baseando -se na inserção de uma fração. Uma fração tem como atributos o código da mesma que só pode conter letras identificadoras de A a Z, tem o nome do titular da fração, junto com a sua morada, o código postal e a sua localidade, poderá ter um telefone e ou um telemóvel associado, um endereço de email, o seu número de contribuinte e ainda em relação à fração, terá a permissão da mesma e o tipo de fração ou seja Habitação ou Comércio.

The screenshot shows a web application titled "Gestão das Frações do Condomínio". On the left, there is a table with two columns: "CodFracao" and "Nome". The table contains four rows: A (Luis Sousa), B (Luisa Costa), C (Luisa), and D (Manuel). The row for "D" is highlighted with a red border. To the right of the table is a form for inserting a new fraction. The form fields are: "Código Fracao*" (D), "Nome*" (Manuel), "Morada*" (Lugar Lixa), "Cod Postal*" (4615-453), "Localidade*" (Lixa), "Telefone**" (255345678), "Telemóvel**" (914534218), "Email*" (manuel@gmail.com), "Contribuinte*" (123456789), "Permissão*" (200), and "Tipo*" (Habituação). A red box highlights the form fields. Below the form, there is a message: "* Campos obrigatórios" and "** É obrigatório pelo menos o preenchimento de um dos campos". At the bottom right, there are three buttons: "Inserir", "Editar", and "Remover". The "Inserir" button is highlighted with a red box. In the bottom left, there is a dialog box titled "Mensagem" with a blue information icon and the text "Fracção inserida com sucesso!!!". The dialog box has an "Ok" button.

CodFracao	Nome
A	Luis Sousa
B	Luisa Costa
C	Luisa
D	Manuel

Código Fracao* D

Nome* Manuel

Morada* Lugar Lixa

Cod Postal* 4615-453

Localidade* Lixa

Telefone** 255345678

Telemóvel** 914534218

Email* manuel@gmail.com

Contribuinte* 123456789

Permissão* 200

Tipo* Habitação

* Campos obrigatórios
** É obrigatório pelo menos o preenchimento de um dos campos

Inserir

Editar

Remover

Mensagem

Fracção inserida com sucesso!!!

Ok

Figura 12: Teste inserção de uma fração

A aplicação afirma que foi inserida com sucesso.

SQLite Database Browser - C:/Users/Luís/Documents/NetBeansProjects/GestCondominio/db/nova.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: Fraccao

	codFracao	nome	morada	codPostal	localidade	telefone	telemovel	email	contribuinte	permiagem	tipo
1	A	Luis Sousa	Serra	4615-453	Lixa	255496758	913436780	luis_sousa16@hotmail.com	123456789	300	Habitação
2	B	Luisa Costa	Lugar Corgo nº39	4115-234	Felgueiras	223456778	964325432	lmsf.c.p@gmail.com	123456756	200	Comércio
3	C	Luisa	serra	4625-234	Lixa	255342345	923454567	luis_sousa16@hotmail.com	124367656	500	Habitação
4	D	Manuel	Lugar Lixa	4615-453	Lixa	255345678	914534218	manuel@gmail.com	123456789	200	Habitação

New Record Delete Record

Figura 13: Verificação de uma inserção na BD

Como se verifica na Figura 13 os dados foram inseridos corretamente na base de dados, sem nenhuma anomalia nem perda de informação, em que foi guardada toda a informação conveniente sobre a fração D e o seu titular.

A Figura 14 evidencia a edição do campo morada e localidade da fração D, fração que foi introduzida anteriormente e verificar-se à posteriormente se as alterações são efetuadas na base de dados em conformidade com as alterações feitas pelo utilizador na camada de aplicação.

Gestão das Frações do Condomínio

CodFracao	Nome
A	Luis Sousa
B	Luisa Costa
C	Luisa
D	Manuel

Código Fracao* D

Nome* Manuel

Morada* Margaride nº39

Cod Postal* 4615-453

Localidade* Felgueiras

Telefone** 255345678

Telemóvel** 914534218

Email* manuel@gmail.com

Contribuinte* 123456789

Permiagem* 200.0

Tipo* Habitação

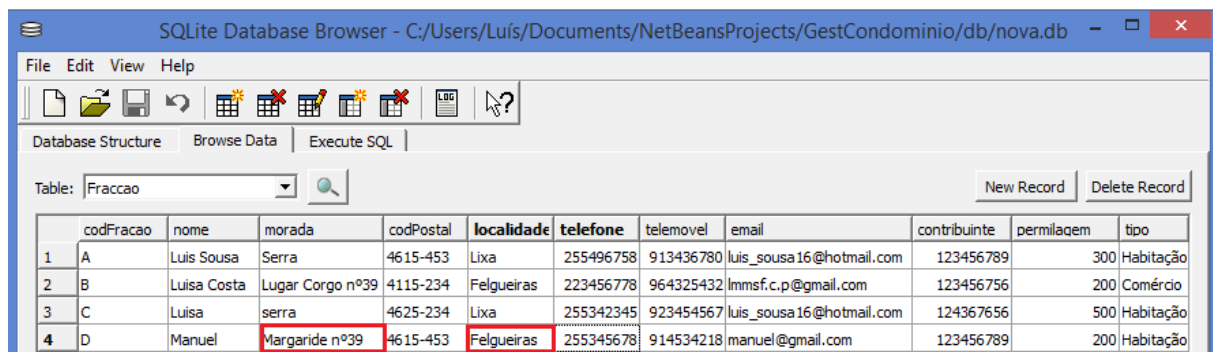
* Campos obrigatórios
** É obrigatório pelo menos o preenchimento de um dos campos

Mensagem: Fracção editada com sucesso!!!

Inserir Editar Remover

Figura 14: Edição de uma fração

A aplicação afirma que foi editada com sucesso e a afirmação pode ser comprovada, pois a Figura 15 assim o indica.



SQLite Database Browser - C:/Users/Luís/Documents/NetBeansProjects/GestCondominio/db/nova.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: Fracao

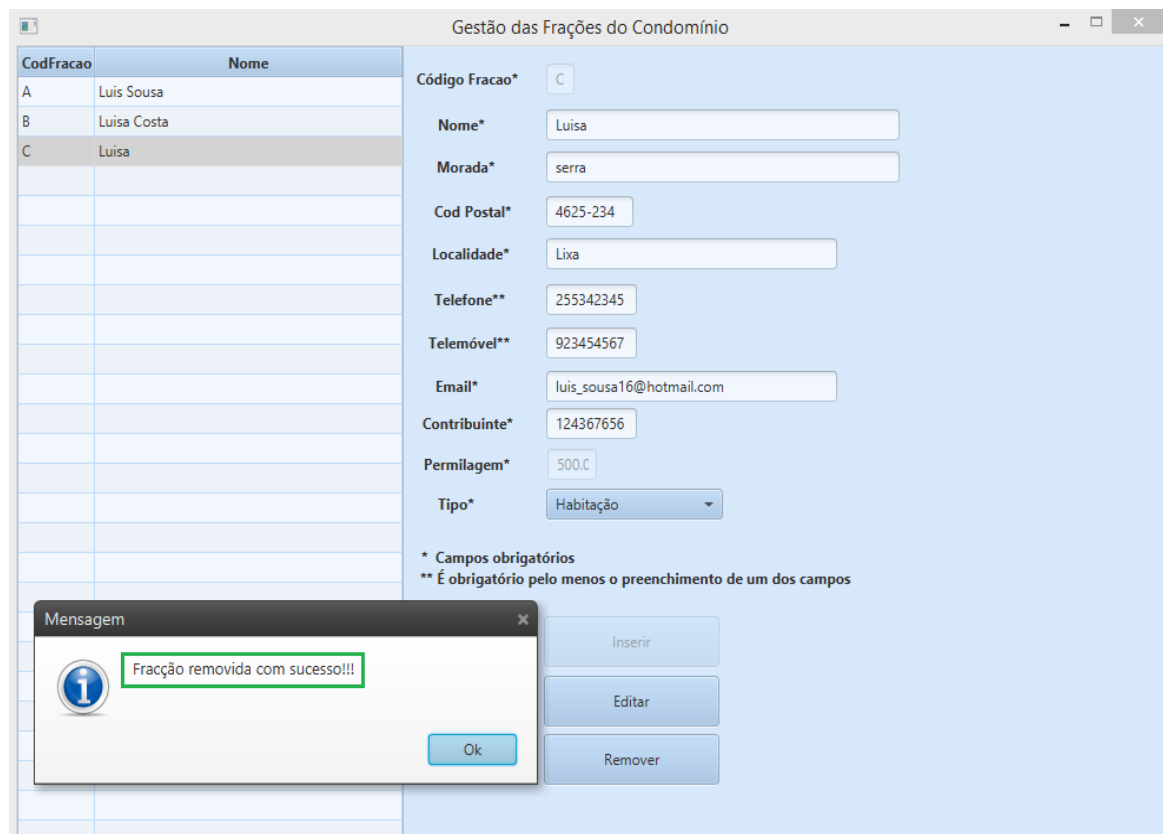
	codFracao	nome	morada	codPostal	localidade	telefone	telemovel	email	contribuinte	permilagem	tipo
1	A	Luis Sousa	Serra	4615-453	Lixa	255496758	913436780	luis_sousa16@hotmail.com	123456789	300	Habitação
2	B	Luisa Costa	Lugar Corgo nº39	4115-234	Felgueiras	223456778	964325432	lmsf.c.p@gmail.com	123456756	200	Comércio
3	C	Luisa	serra	4625-234	Lixa	255342345	923454567	luis_sousa16@hotmail.com	124367656	500	Habitação
4	D	Manuel	Margaride nº39	4615-453	Felgueiras	255345678	914534218	manuel@gmail.com	123456789	200	Habitação

New Record Delete Record

Figura 15: Verificação da edição dos campos na BD

Verifica-se que os dados foram editados com sucesso na base de dados, conforme as alterações que foram pedidas pelo utilizador.

Agora removeu-se a fração que tinha sido inserida no início do teste à base de dados, a fração com o código D. Como se pode verificar pela Figura 16 a fração D foi removida com sucesso.



Gestão das Frações do Condomínio

CodFracao	Nome
A	Luis Sousa
B	Luisa Costa
C	Luisa

Código Fracao* C

Nome* Luisa

Morada* serra

Cod Postal* 4625-234

Localidade* Lixa

Telefone** 255342345

Telemóvel** 923454567

Email* luis_sousa16@hotmail.com

Contribuinte* 124367656

Permilagem* 500.0

Tipo* Habitação

* Campos obrigatórios
** É obrigatório pelo menos o preenchimento de um dos campos

Inserir Editar Remover

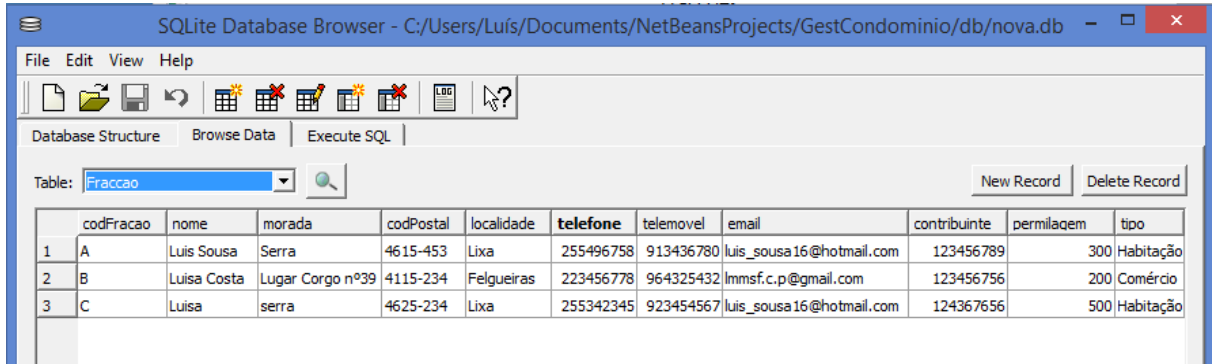
Mensagem

Fracção removida com sucesso!!!

Ok

Figura 16: Remoção de uma fração

De seguida verificou-se na base de dados e realmente a fração D foi removida como se pode verificar na Figura 17.



	codFracao	nome	morada	codPostal	localidade	telefone	telemovel	email	contribuinte	permiagem	tipo
1	A	Luis Sousa	Serra	4615-453	Lixa	255496758	913436780	luis_sousa16@hotmail.com	123456789	300	Habitação
2	B	Luisa Costa	Lugar Corgo nº39	4115-234	Felgueiras	223456778	964325432	lmsf.c.p@gmail.com	123456756	200	Comércio
3	C	Luisa	serra	4625-234	Lixa	255342345	923454567	luis_sousa16@hotmail.com	124367656	500	Habitação

Figura 17: Verificação da remoção na BD

A fração D já não se encontra na base de dados.

4.2 – Execução multi-plataforma

Foram feitos testes à funcionalidade principal, funcionalidade essa que permite definir e gerir todos os dados relativos a um orçamento. Os seguintes testes foram feitos em duas plataformas distintas a fim de verificar se funcionavam dentro da normalidade tanto em Windows 8.1 Pro de 64 bits como no Ubuntu 14.04 LTS de 64 bits. Pois a aplicação foi desenvolvida em ambiente Windows mas tem de funcionar obrigatoriamente em plataforma Linux, pois foi essa uma das muitas razões para a elaboração deste projeto.

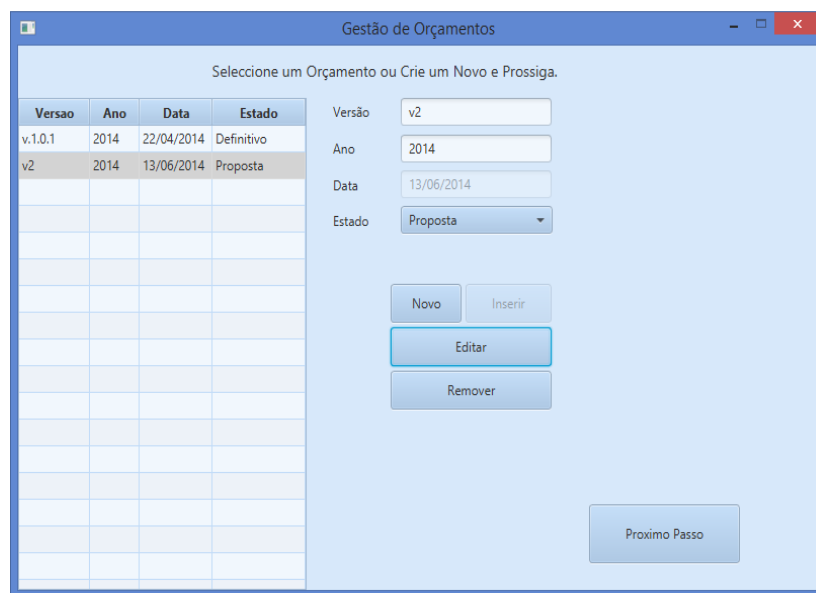


Figura 18: Adição de um Orçamento em Windows 8.1 PRO

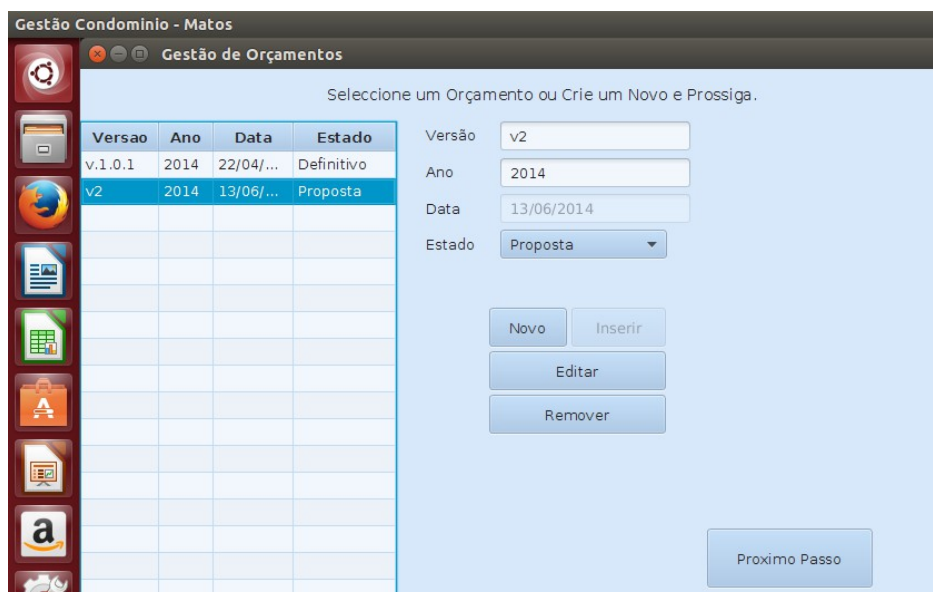


Figura 19: Adição de um Orçamento em Ubuntu 14.04 LTS

A Figura 18 mostra que todo o processo correu normalmente, isto para a plataforma Windows. Enquanto que a Figura 19 evidencia o mesmo para a plataforma Linux. A interface visualizada corresponde à criação de um orçamento, sendo que depois deste passo ainda existem mais quatro, em que o passo final apresenta as mensalidades para cada fração.

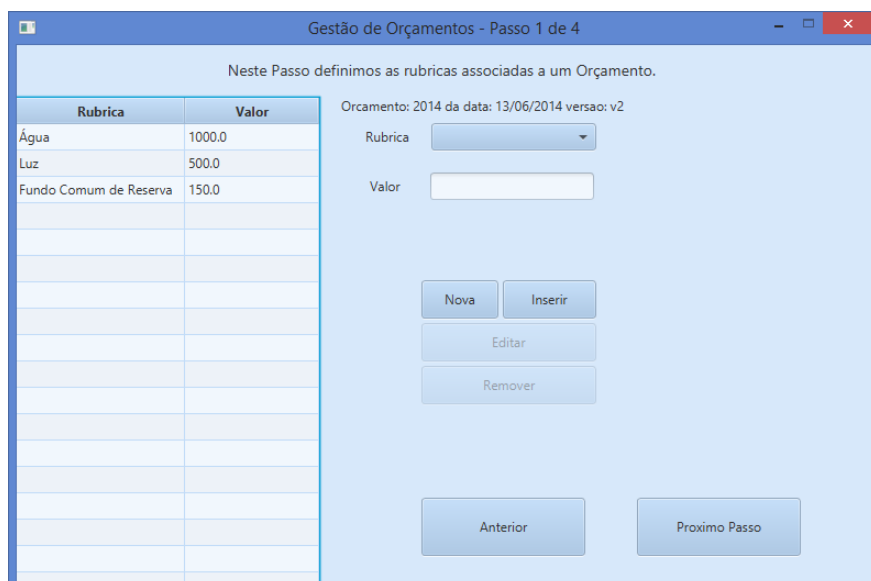


Figura 20: Primeiro passo na definição de um orçamento em Windows 8.1 Pro

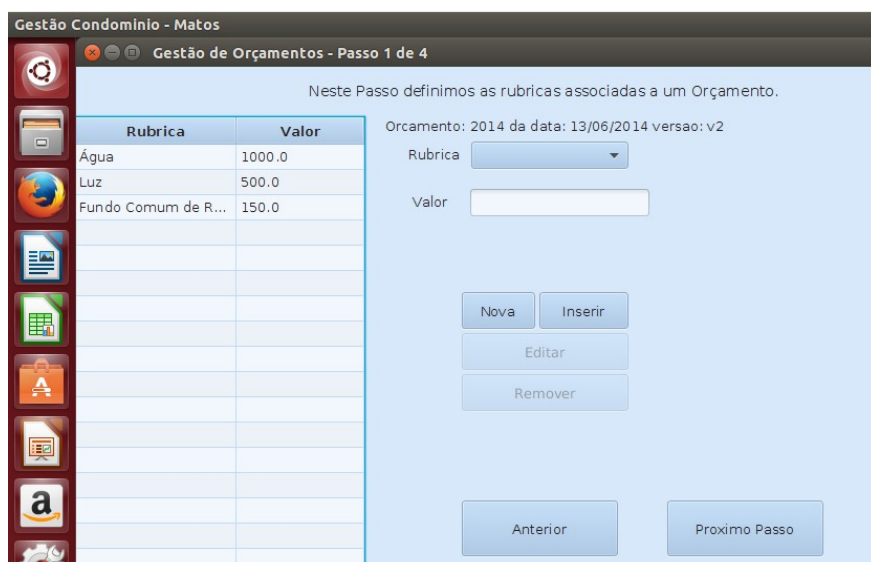


Figura 21: Primeiro passo na definição de um orçamento em Ubuntu 14.04 LTS

A Figura 20 mostra que todo o processo correu normalmente, isto para a plataforma Windows. Enquanto que a Figura 21 evidencia o mesmo para a plataforma Linux. A interface visualizada corresponde ao primeiro passo para a definição e gestão de um orçamento sendo que permite associar rubricas e o seu valor a um determinado orçamento.

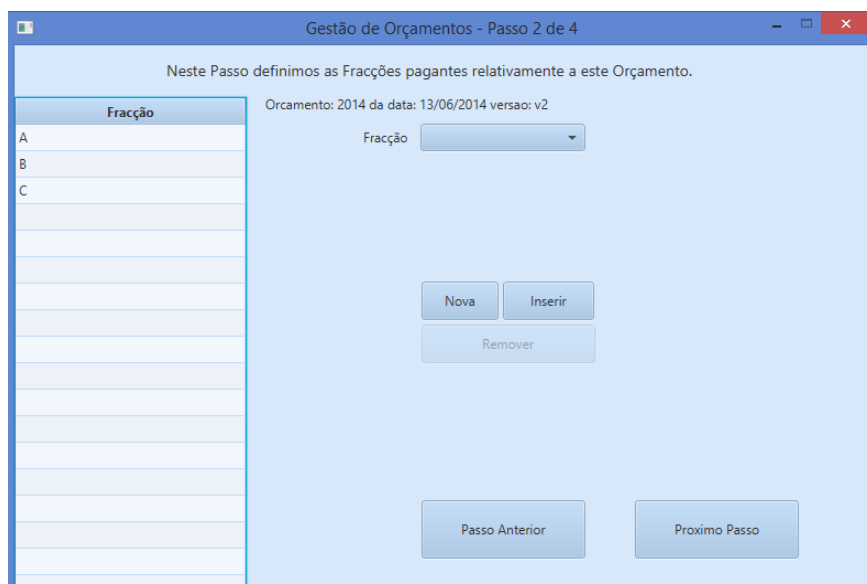


Figura 22: Segundo passo na definição de um orçamento em Windows 8.1 Pro

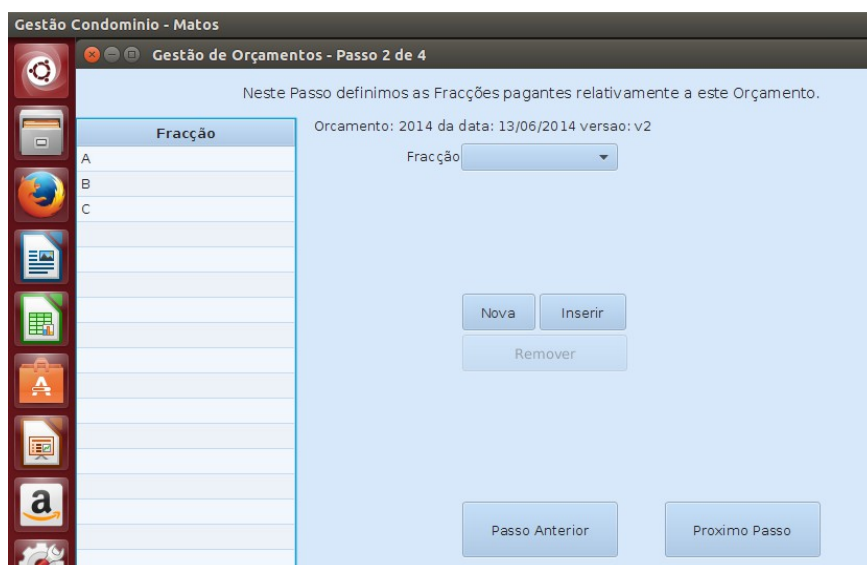


Figura 23: Segundo passo na definição de um orçamento em Ubuntu 14.04 LTS

A Figura 22 mostra que todo o processo correu normalmente, isto para a plataforma Windows. Enquanto que a Figura 23 evidencia o mesmo para a plataforma Linux. A interface visualizada corresponde ao segundo passo para a definição e gestão de um orçamento sendo que este permite definir as frações pagantes para um determinado orçamento.

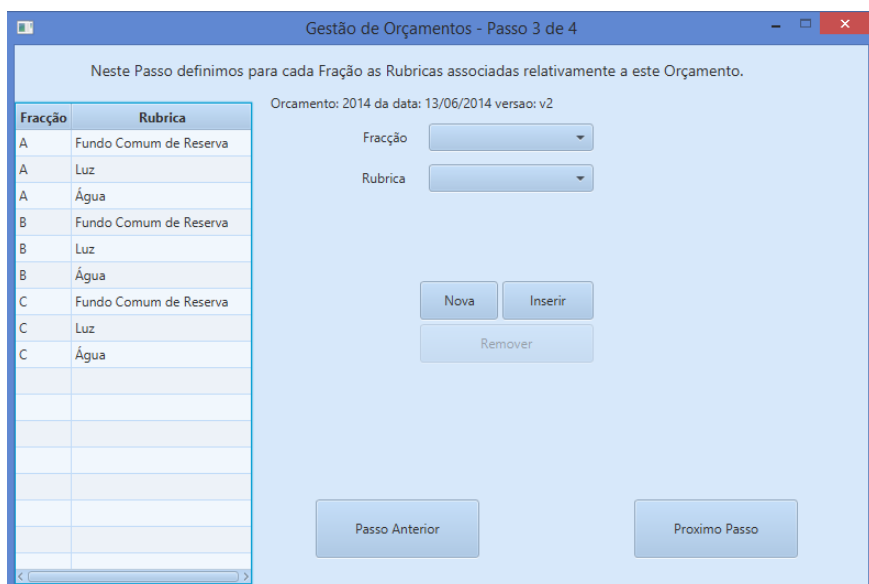


Figura 24: Terceiro passo na definição de um orçamento em Windows 8.1 Pro

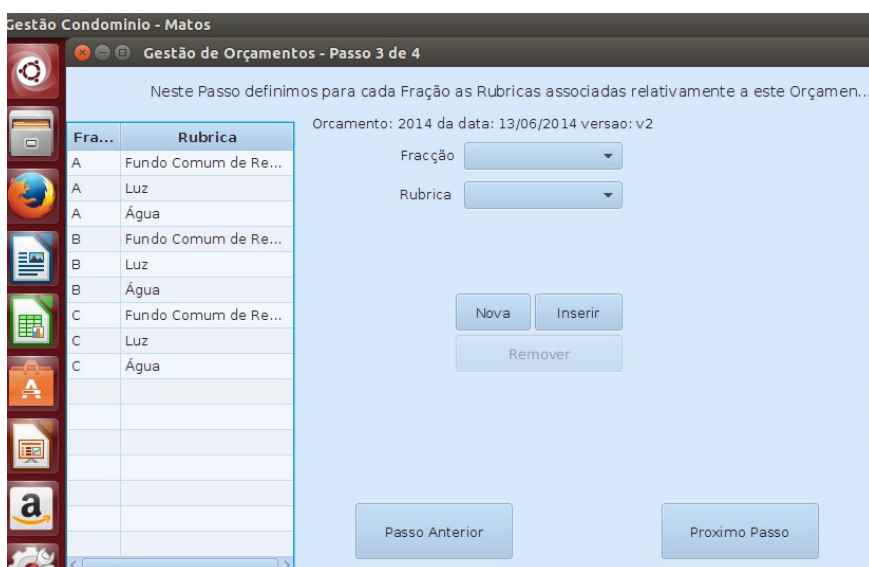


Figura 25: Terceiro passo na definição de um orçamento em Ubuntu 14.04 LTS

A Figura 24 mostra que todo o processo correu normalmente, isto para a plataforma Windows. Enquanto que a Figura 25 evidência o mesmo para a plataforma Linux. A interface visualizada corresponde ao terceiro passo para a definição e gestão de um orçamento sendo que este permite definir para cada fração pagante as rubricas que lhe estão associadas.

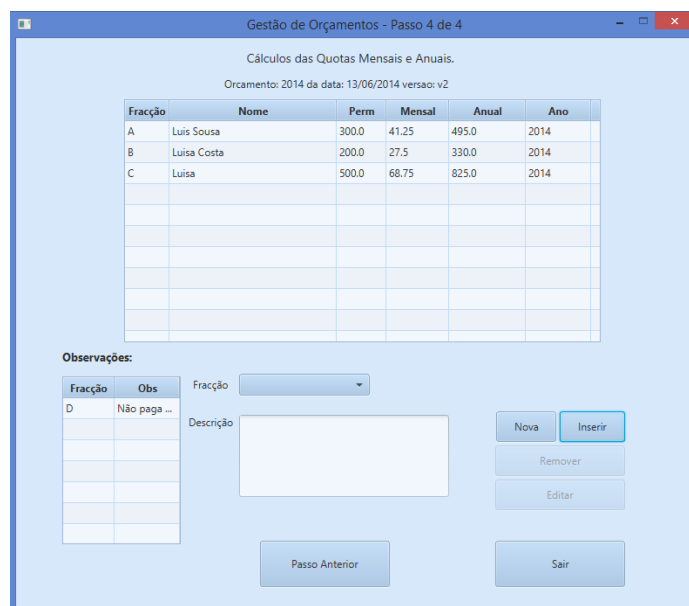


Figura 26: Quarto passo na definição de um orçamento em Windows 8.1 Pro

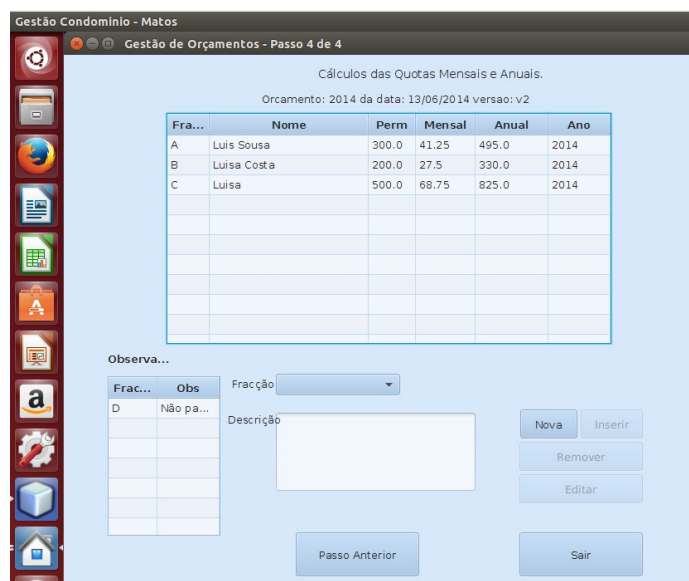


Figura 27: Quarto passo na definição de um orçamento em Ubuntu 14.04 LTS

A Figura 26 mostra que todo o processo correu normalmente, isto para a plataforma Windows. Enquanto que a Figura 27 evidência o mesmo para a plataforma Linux. A interface visualizada corresponde ao quarto e último passo para a definição e gestão de um orçamento sendo que este mostra as mensalidades a pagar para cada fração e permite ainda inserir observações para uma ou várias frações.

4.3 – Conclusões

A aplicação permite toda a gestão em torno de um orçamento. Outro requisito era o de que a aplicação funcionasse em diversos sistemas operativos. A aplicação executou a funcionalidade principal, em duas plataformas, Windows 8.1 x64 e Ubuntu 14.04 x64 sem problemas e dentro da normalidade. Já em termos visuais foram encontradas algumas diferenças como o tipo de letra e a aparência da aplicação, nada que impossibilitasse a aplicação de correr da melhor maneira possível.

5 – Conclusões

A aplicação desenvolvida cumpriu os requisitos mínimos propostos, sendo que foram obtidos vários resultados relevantes. Primeiramente começou-se com o requisito de ter **suporte a várias base de dados**, onde cada base de dados estava destinada a armazenar os dados de um e só um condomínio, sendo que a aplicação permite ainda **abrir qualquer condomínio** usado em **tempo de execução**. Estas foram as primeiras funcionalidades a serem desenvolvidas.

Permite **inserir condomínios** em que cada condomínio tem os seus próprios campos, guardando na base de dados toda a informação necessária para identificar o condomínio que está a ser gerido.

Depois da inserção de um condomínio, seguiu-se a **inserção de uma ou várias frações**, ficando guardada toda a informação de uma fração na base de dados em que cada fração tem associada a si um código de fração, o nome do condómino daquela fração e os restantes atributos que eram relevantes para a identificação de uma fração. Além da inserção foi desenvolvido também a listagem, edição e remoção das mesmas.

De seguida foi desenvolvida a funcionalidade de **inserção e remoção de rubricas**. A inserção de rubricas era importante para posteriormente quando fosse criado um orçamento se pudesse associar rubricas a pagar por parte de determinadas frações.

Depois seguiu-se para a **criação de um orçamento**. A aplicação permite definir um orçamento, uma versão para esse orçamento e ainda definir o estado desse orçamento, sendo o estado do tipo definitivo em que já foi aprovado por todas as partes intervenientes (condóminos), ou do tipo proposta em que, como o nome indica, ainda não foi aprovado por todas as partes e ainda não pode ser utilizado na gestão do condomínio num determinado ano. Na definição de um orçamento a aplicação possibilita também a **definição das rubricas** que entram nas contas para aquele orçamento e o montante a pagar pelos condóminos por aquela rubrica, que **frações é que pagam** (num determinado ano a fração A pode ser uma fração pagante, mas no ano a seguir pode não entrar nas contas do condomínio). Neste mesmo contexto permite ainda associar **para cada fração que rubricas é que pagam**, pois, por exemplo, a fração A pode pagar a rubrica água e luz e fundo comum de reserva, já a fração B pode pagar apenas a rubrica água e fundo comum de reserva, sendo o fundo comum de reserva obrigatório para todas as frações pagantes. A aplicação **calcula as mensalidades** a pagar para cada fração.

Permite também **gerar aviso de débitos** para um determinado orçamento, em que cada aviso tem a informação da fração, do mês em débito e quanto deve, sendo que dentro desta funcionalidade a aplicação possibilita gerar o devido PDF do aviso de débito, permitindo visualizá-lo, imprimi-lo ou até enviá-lo por email.

Em seguida, foi desenvolvida a parte de **pagamentos dos débitos**, em que cada **recibo** pode ser relativo a um ou vários avisos de débito de uma fração e se possibilita também imprimir recibos ou o seu envio por email.

Tem ainda como funcionalidade a definição de um **mapa de despesas** por orçamento em que a aplicação permite inserir, editar, remover e listar as despesas para um orçamento. Por fim temos acesso a todos os **documentos finais** com os **relatórios de contas** para um determinado orçamento.

5.1 – Trabalho futuro e dificuldades

Em termos futuros, poder-se-á melhorar a parte do envio dos PDF's por email, permitindo que o email de quem envia não tenha de ser necessariamente gmail. A aplicação só ficou configurada para o envio de emails a partir de emails provenientes de contas gmail. Poderá ainda se fazer algumas filtrações e pesquisas nas diferentes interfaces da aplicação permitindo assim um melhor acesso a informação específica.

Uma das dificuldades no desenvolvimento desta aplicação foi perceber como funcionava um condomínio, pois existiam termos técnicos que eram precisos perceber ao detalhe para depois serem aplicados no desenvolvimento da aplicação da melhor maneira, sendo que não havia um grande à vontade neste tema.

Uma dificuldade que foi sentida foi na parte do envio de email em que a janela ficava sem resposta durante o envio dos vários emails ao mesmo tempo. A *thread* JavaFX, enquanto não concluísse o envio de todos os emails, não deixava fazer mais nada no programa. Resolveu-se o problema com recurso a técnicas de concorrência em JavaFX, nomeadamente criando-se uma *task* (*thread*), para que enquanto esta *task* enviava emails em *background*, a *thread* JavaFX ficava liberta para controlar a janela e permitir o seu uso[3].

7 – Referências Bibliográficas

- [1] Giulio Toffoli.(data desconhecida).iReport Designer Getting Started.[ONLINE].Disponível em:http://community.jaspersoft.com/wiki/ireport-designer-getting-started#User_Interface
- [2] Jaspersoft Corporation.(2011).THEJASPERREPORTSULTIMATEGUIDE.[ONLINE].Disponível em:<http://jasperreports.sourceforge.net/JasperReports-Ultimate-Guide-3.pdf>
- [3] Irina Fedortsova.(2012, Junho).Concurrency in JavaFX.[ONLINE].Disponível em: <http://docs.oracle.com/javafx/2/threads/jfxpub-threads.htm>
- [4] Oracle.(2013).NetBeans IDE - The Smarter and Faster Way to Code.[ONLINE].Disponível em: <https://netbeans.org/features/index.html>
- [5] PedroPinto.(Abril,2011).Sqlite database browser – Um gestor gráfico para o sqlite. [ONLINE].Disponível em:<http://pplware.sapo.pt/windows/software/sqlite-database-browser-%E2%80%93-um-gestor-grafico-para-o-sqlite/>
- [6] Oracle.JavaFX Scene Builder - A Visual Layout Tool for JavaFX Applications.[ONLINE].Disponível em:<http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html>
- [7] Carla Pereira.(2011-12).RECOMENDAÇÕES PARA ELABORAÇÃO DO DOCUMENTO DE ESPECIFICAÇÃO DE REQUISITOS

8 – Anexos

Documento de requisitos

1 – Introdução

Neste documento pode ser encontrado os requisitos funcionais e não funcionais que a aplicação deverá ter, os respetivos diagramas de casos de usos, alguns diagramas de sequência e de classe. Este documento de requisitos foi elaborado com base na estrutura da referência [7]

1.1 – Objetivo

O objetivo deste documento visa especificar os requisitos de sistema relacionados com uma aplicação de gestão de condomínio, aplicação desenvolvida no âmbito da disciplina de Projeto final, do curso de Engenharia Informática na Escola Superior de Tecnologia e Gestão de Felgueiras no ano letivo dois mil e treze – dois mil e quatorze. Este documento é direcionado para quem tem a função de implementar a aplicação ou para quem quiser ter uma vista global sobre o funcionamento da mesma.

1.2 – Âmbito

Esta aplicação será desenvolvida para ajudar a maximizar a organização de condomínios, permitindo gerir as contas de uma forma eficaz e tecnológica em vez da maneira tradicional que é a utilização de papeis para tudo, pois da maneira tradicional por vezes fica difícil manter este tipo de sistema de uma forma eficaz. Tecnologicamente fica tudo mais compacto, pois os dados são todos armazenados numa BD.

1.3 – Acrónimos

BD – base de dados

1.4 – Referências

IEEE. IEEE Std 830-1998 IEEE Recommended Practice para Especificações de Requisitos de Software. IEEE Computer Society, 1998.

1.5 – Visão geral do documento

No capítulo 1, será apresentado uma pequena introdução ao documento, explicando o porquê deste documento, qual a sua finalidade. No capítulo 2 , a secção de Descrição geral , como o nome indica dá uma perspetiva global das funcionalidades da aplicação, o contexto, as restrições.

No capítulo3 será apresentado os requisitos funcionais, seguido do diagrama de caso de uso e dos respetivos diagramas de sequência para cada caso de uso, os vários diagramas de classes e por fim a listagem dos requisitos não funcionais .

2 – Descrição geral

A aplicação em causa, aplicação para gestão de condomínios, deve permitir gerir da melhor forma todo um condomínio e as suas necessidades guardando toda a informação numa base de dados.

2.1 – Contexto do produto

A aplicação é auto-contida, não interliga com outras aplicações, sendo então uma aplicação independente. A aplicação tem de tema toda a parte que engloba um condomínio, desde frações , rubricas , avisos de débitos, recibos, orçamentos e despesas.

2.2 – Funções do produto

De esta aplicação podemos destacar:

- **Criação de múltiplas base de dados.** Esta foi uma das funcionalidades pedidas e que deve ser concretizada com sucesso. Terá de permitir que se crie uma base de dados para cada condomínio a gerir.

- Abrir sempre a última base de dados.** Permitir que quando se inicia a aplicação seja aberto sempre o último condomínio utilizado, permitindo assim ao utilizador acabar o trabalho que estava a ser elaborado.

- **Criar e editar um Condomínio.** Para cada base de dados está associado um Condomínio, em que cada condomínio depois de inserido também poderá ser editado.

-Inserir, editar listar e remover frações. Um condomínio pode ter de uma a várias frações, permitindo assim inserir editar listar e remover frações.

-Gestão de rubricas. Em cada Orçamento, as frações têm rubricas associadas por isso a aplicação deve ter a possibilidade de inserir e eliminar rubricas.

-Definir um Orçamento ou proposta e calcular as mensalidades. Esta é uma funcionalidade crucial pois na altura de inserção de um orçamento deverá ser definido um estado, esse estado pode tomar o valor de definitivo ou proposta. Se for definitivo quer dizer que vai ser aquele orçamento utilizado para a gestão e que foi acordado por todas as partes envolvidas, se for do tipo proposta como o próprio nome indica ainda não está decidida a sua utilização , devendo ser apresentado numa reunião de condomínio aos condomínios para definirem a sua utilização ou não.

Possibilitar a associação de rubricas a um orçamento/proposta, editar e remover caso seja necessário, especificar que frações pagam naquele orçamento, que rubricas estão associadas a cada fração, inserir editar e remover observações associadas a uma determinada fração e por fim calcular as respetivas mensalidades para cada fração.

- **Gerar avisos de débito** para uma ou várias frações, para um ou mais meses relativamente a um orçamento.

- **Listar, filtrar, imprimir ou enviar por email os avisos de débito.** Depois dos avisos de débitos gerados para um determinado orçamento a aplicação deverá permitir ao utilizador criar um documento PDF com os dados do aviso de débito gerado , possibilitando assim a impressão do PDF ou enviá-lo por email para a respetiva fração.

- **Listar, filtrar, imprimir ou enviar por email recibos.** Depois dos avisos de débitos , gerir recibos para os débitos que foram gerados em relação a um determinado orçamento é uma funcionalidade importante. A aplicação criará automaticamente um documento PDF com os dados de um recibo , sendo que um recibo poderá estar ligado a um ou vários débitos de uma só fração. Possibilitar ainda que esse PDF seja impresso para entregar ao titular da fração ou seja enviado por email.

- **Apresentar documentos finais** para um determinado orçamento. Gerar relatórios com as contas finais, pagamentos realizados por cada fração, despesas efetuadas pelo condomínio, saldo atual para cada fração entre outros documentos que forem precisos.

2.3 – Características de utilizador

Para a utilização desta aplicação o utilizador não tem ser nenhum grande entendido na matéria. Deve é ter conhecimento de como utilizar cada funcionalidade da melhor maneira, sendo que para que as coisa sejam facilitadas ao utilizador final. O desenvolvedor da aplicação deve organizar e estruturar a aplicação de forma a que as interfaces sejam os mais apelativas, funcionais e simples não criando

dúvidas de utilização por parte de quem as irá usar. Não terá de ter muita experiência nem avançados conhecimentos técnicos.

2.4 – Restrições

- a) a base de dados da aplicação terá de ser uma base de dados embebida, em que não é preciso instalar software adicional para termos a base de dados funcional, ficando assim os dados guardados em um único ficheiro facilitando o backup da mesma.
- b) cada base de dados só deve ser utilizada para a gestão de um e só um condomínio.
- c) um recibo poderá ser relativo a vários débitos da mesma fração.

2.5 – Pressupostos e dependências

Um outro item que pode limitar o desenvolvimento da aplicação é a linguagem de programação a adotar. A aplicação tem de ser multi-plataforma e a linguagem de desenvolvimento terá de ser obrigatoriamente Java, pois esse é o principal objetivo requerido.

3 – Requisitos específicos

3.1 – Requisitos de interfaces externos

3.1.1 – Interfaces com os utilizadores

De seguida serão apresentadas alguns esboços de interfaces para a aplicação em causa.

Esboço da interface principal

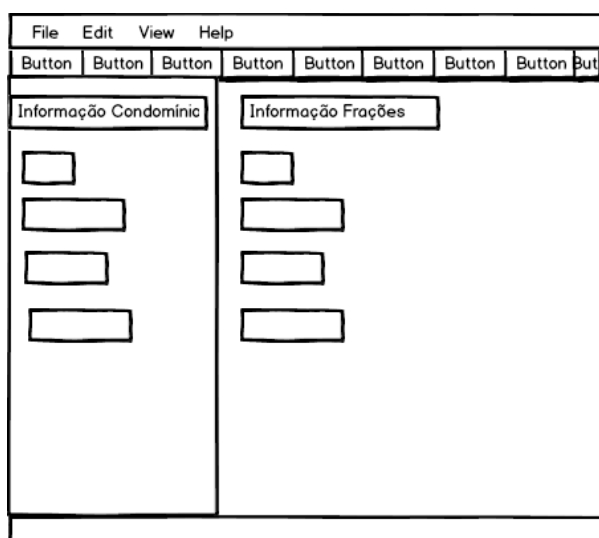


Figura 28: Esboço Interface principal da aplicação

A Figura 28 demonstra um esboço da interface principal da aplicação que deverá ter no topo um menu bar para o utilizador aceder a algumas funcionalidades da aplicação. Depois deverá ter uma barra com botões para o utilizador aceder às funcionalidades que irão ser mais utilizadas e depois no corpo principal deverá ter informações sobre o condomínio que está a ser utilizado, isto à esquerda. À direita deverá ter informações sobre as frações e incluindo por exemplo o saldo corrente para cada fração.

Frações	
Lista com as frações	Nome <input type="text"/>
	Morada <input type="text"/>
	Restantes campos
	⋮
	<input type="button" value="Inserir"/> <input type="button" value="Editar"/> <input type="button" value="Remover"/>

Figura 29: Esboço relativo à gestão de frações

A figura Figura 29 mostra um pequeno esboço base do que deverá ser a interface de gestão de frações, permitindo do lado esquerdo obter a lista de frações que já foram inseridas e visualizar a que está seleccionada no lado direito nas respectivas caixas de texto e no fundo do lado direito os botões inserir, editar e remover, cada um com a sua respetiva funcionalidade.

Na figura Figura 30 é apresentado um, esboço do que deverá ser a interface de visualização dos recibos gerados , permitindo do lado esquerdo escolher um recibo da lista de recibos que já foram gerados e visualizar o relatório do recibo que está selecionado no lado direito e no fundo do lado direito os botões enviar email e imprimir, cada um com a sua respetiva funcionalidade.

3.2 – Requisitos funcionais

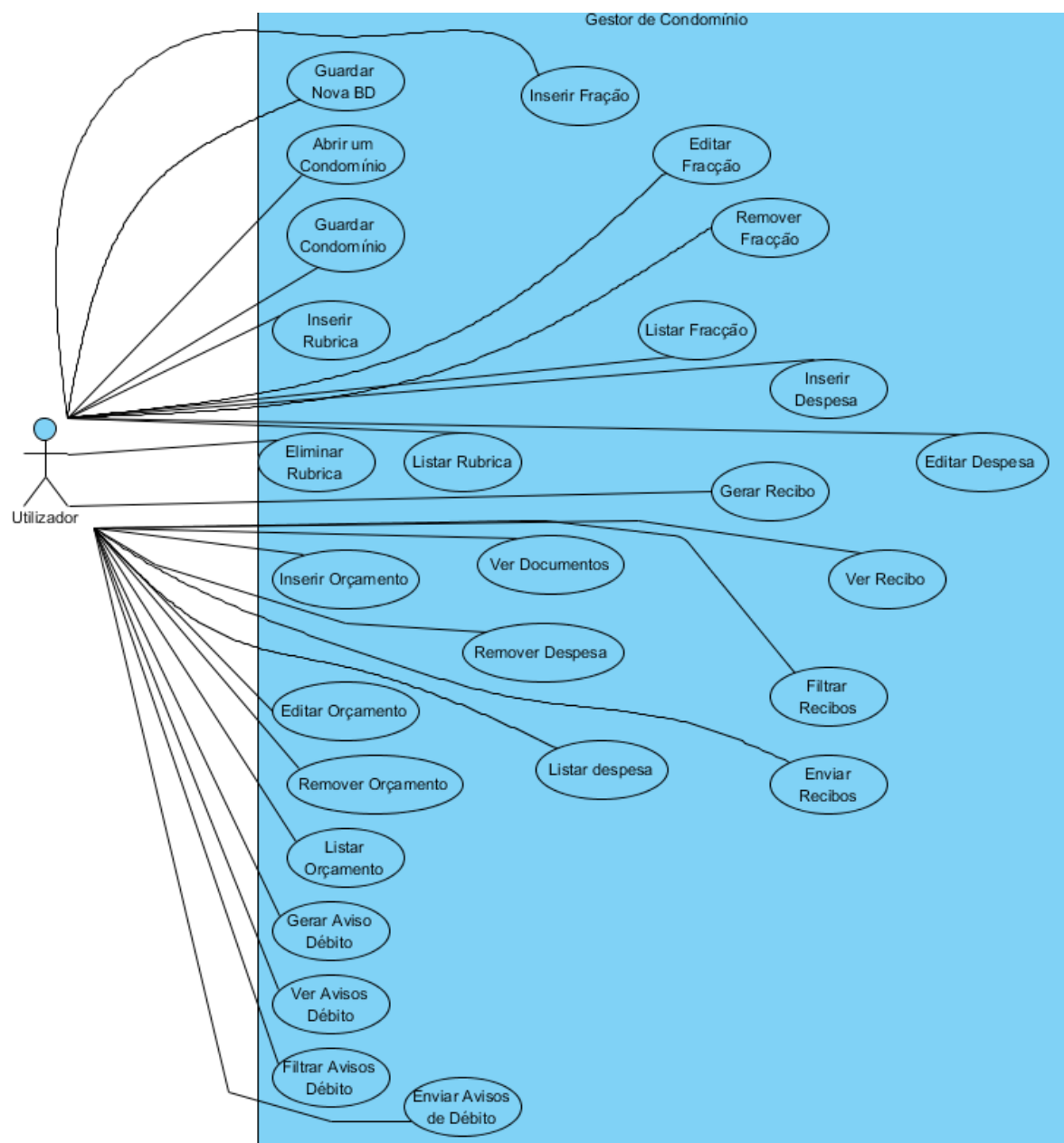


Figura 31: Diagrama casos de uso da aplicação

A Figura 31 representa o diagrama de casos de uso da aplicação sendo que os diagramas de caso de uso são uma boa ferramenta para o levantamento de requisitos funcionais da aplicação a desenvolver.

Caso de Uso 1 – Criar Nova Base de Dados

identificador: UC1

nome: Criar nova base de Dados.

descrição sumária: Este caso de uso refere-se à criação de uma nova base de dados e é representado na Figura 32 o respetivo diagrama de sequência com as mensagens passadas entre objetos e a sequência de processos.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) clicar no menu condomínio.
- 2) clicar em Nova BD.
- 3) escolher o nome da base de dados e clicar em guardar.
- 4) nova base de dados criada.

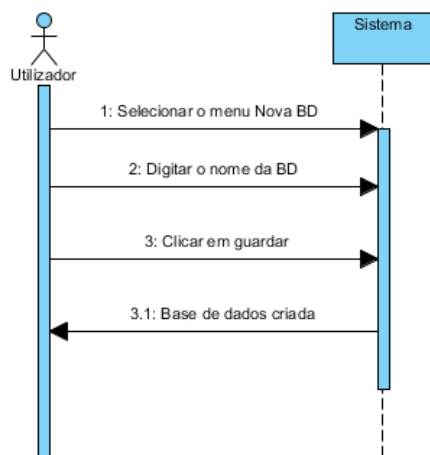


Figura 32: Diagrama de sequência - Criar Nova Base de Dados

Caso de Uso 2 – Abrir um condomínio

identificador: UC2

nome: Abrir um condomínio

descrição sumária: Caso de uso relativo à abertura de um condomínio.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) clicar no Abrir Condomínio
- 2) seleccionar condomínio
- 3) clicar em Abrir
- 4) condomínio aberto

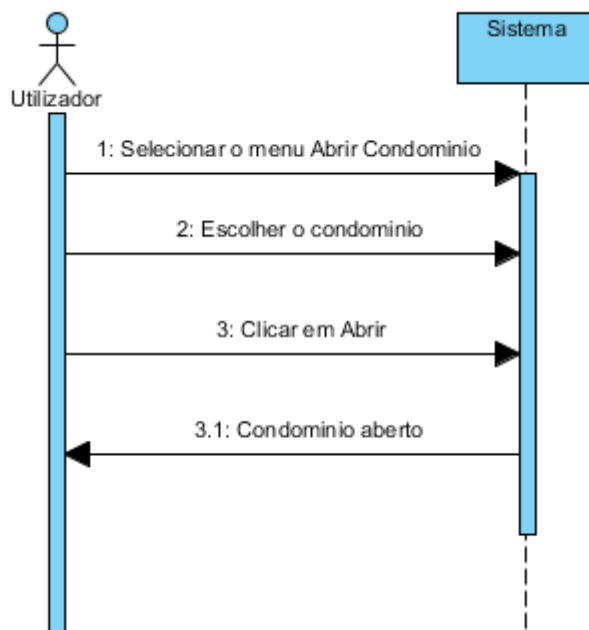


Figura 33: Diagrama de sequência – Abrir Condomínio

Caso de Uso 3 – Guardar condomínio

identificador: UC3

nome: Guardar um condomínio

descrição sumária: Caso de uso relativo à inserção e edição de um condomínio.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) preencher os campos necessários para o registo do condomínio
- 2) clicar em guardar condomínio

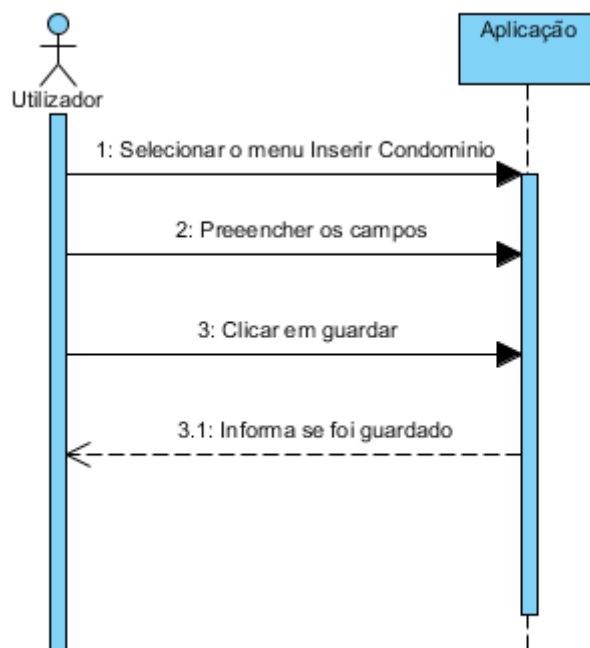


Figura 34: Diagrama de sequência – Guardar Condomínio

Caso de Uso 4 – Inserir Fração

identificador: UC4

nome: Inserir Fração

descrição sumária: Caso de uso relativo à inserção de uma fração na base de dados.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) preencher os campos necessários para o registo da fração
- 2) clicar no botão Inserir

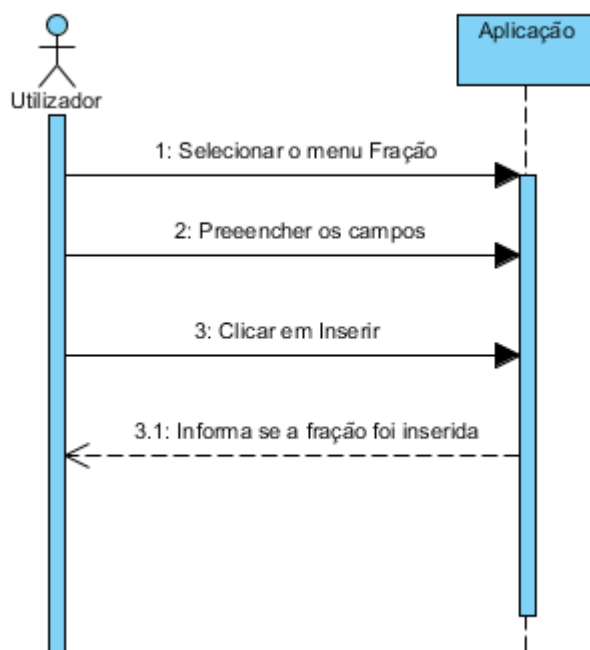


Figura 35: Diagrama de sequência – Inserir Fração

Caso de Uso 5 – Editar Fração

identificador: UC5

nome: Editar Fração

descrição sumária: Caso de uso relativo à edição de uma fração na base de dados.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) editar os campos necessários para a fração
- 2) clicar no botão Editar

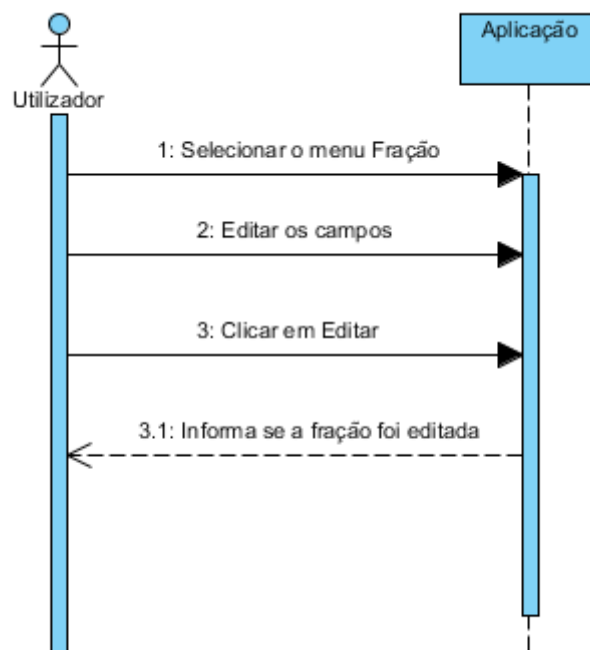


Figura 36: Diagrama de sequência – Editar Fração

Caso de Uso 6 – Remover Fração

identificador: UC6

nome: Remover Fração

descrição sumária: Caso de uso relativo à remoção de uma fração na base de dados.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar a fração
- 2) clicar no botão Remover

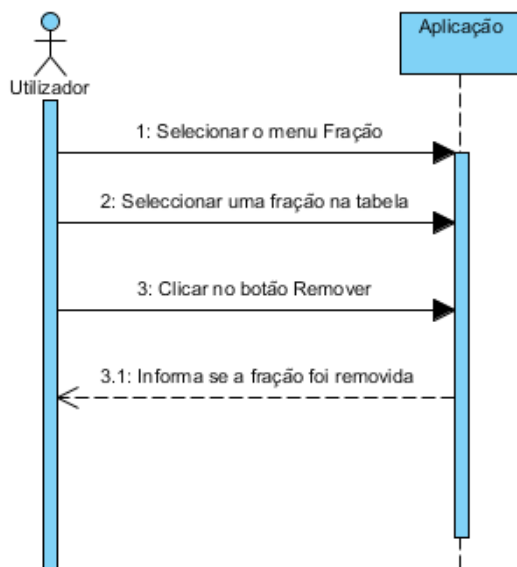


Figura 37: Diagrama de sequência – Remover Fração

Caso de Uso 7 – Listar Fração

identificador: UC7

nome: Listar Fração

descrição sumária: Caso de uso relativo à listagem de uma fração ou várias frações.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

1) seleccionar a fração na tabela

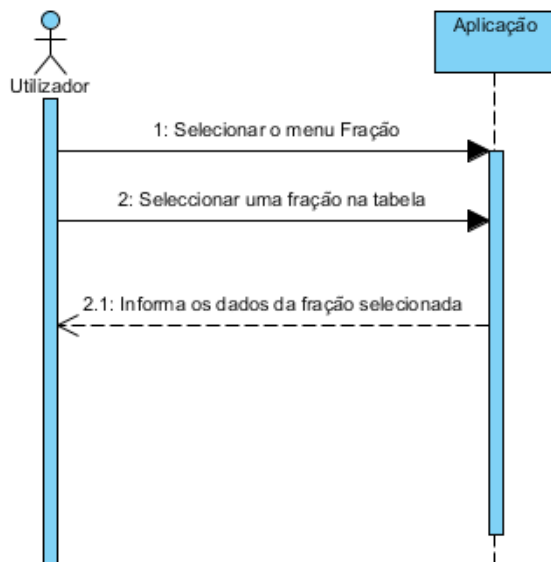


Figura 38: Diagrama de sequência – Listar Fração

Caso de Uso 8 – Inserir Rubrica

identificador: UC8

nome: Inserir Rubrica

descrição sumária: Caso de uso relativo à inserção de uma rubrica.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) preencher os campos necessários para o registo
- 2) clicar no botão Inserir

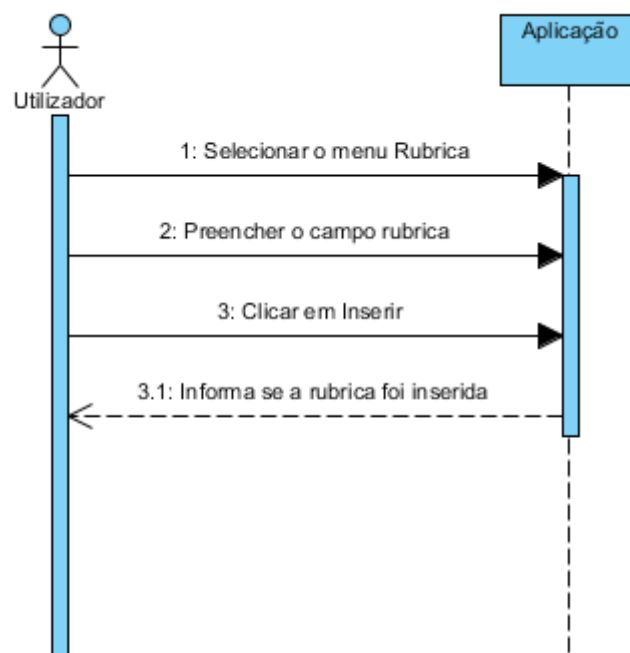


Figura 39: Diagrama de sequência – Inserir Rubrica

Caso de Uso 9 – Remover Rubrica

identificador: UC9

nome: Remover Rubrica

descrição sumária: Caso de uso relativo à remoção de uma rubrica.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

1) seleccionar a rubrica na tabela

2) clicar no botão Eliminar

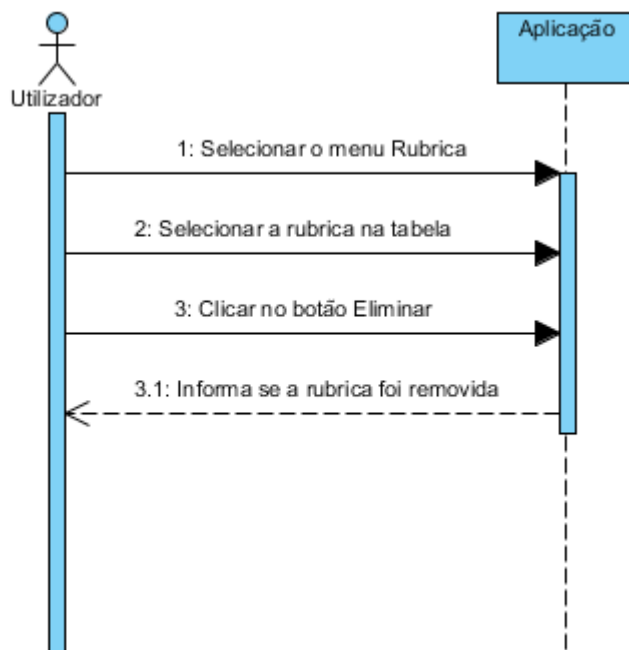


Figura 40: Diagrama de sequência – Remover Rubrica

Caso de Uso 10 – Listar Rubrica

identificador: UC10

nome: Listar Rubrica

descrição sumária: Caso de uso relativo à listagem das rubricas.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

1) seleccionar a rubrica na tabela

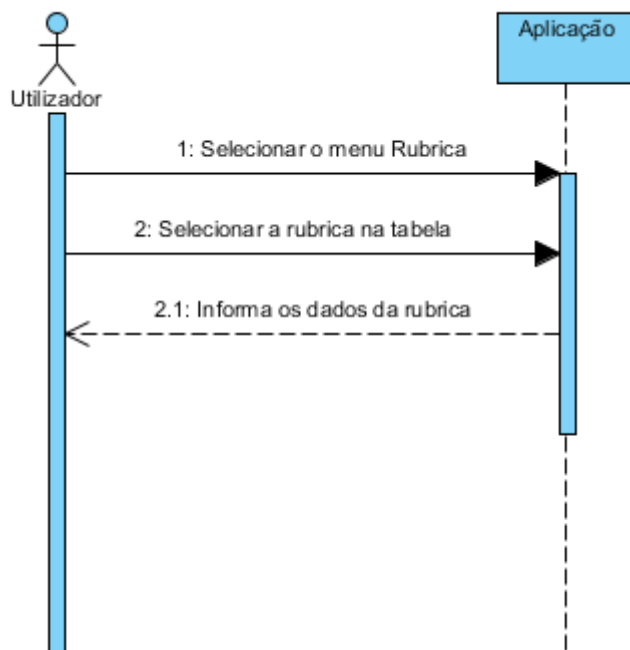


Figura 41: Diagrama de sequência – Listar Rubrica

Caso de Uso 11– Inserir Orçamento

identificador: UC11

nome: Inserir Orçamento

descrição sumária: Caso de uso relativo à inserção de um orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) preencher os campos necessários para o registo
- 2) clicar no botão Inserir

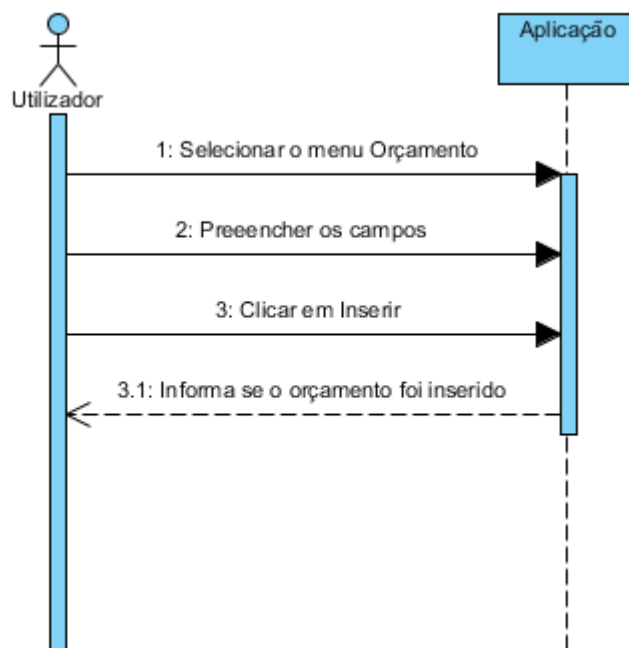


Figura 42: Diagrama de sequência – Inserir Orçamento

Caso de Uso 12– Editar Orçamento

identificador: UC12

nome: Editar Orçamento

descrição sumária: Caso de uso relativo à edição de um orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) preencher os campos necessários para a edição
- 2) clicar no botão Editar

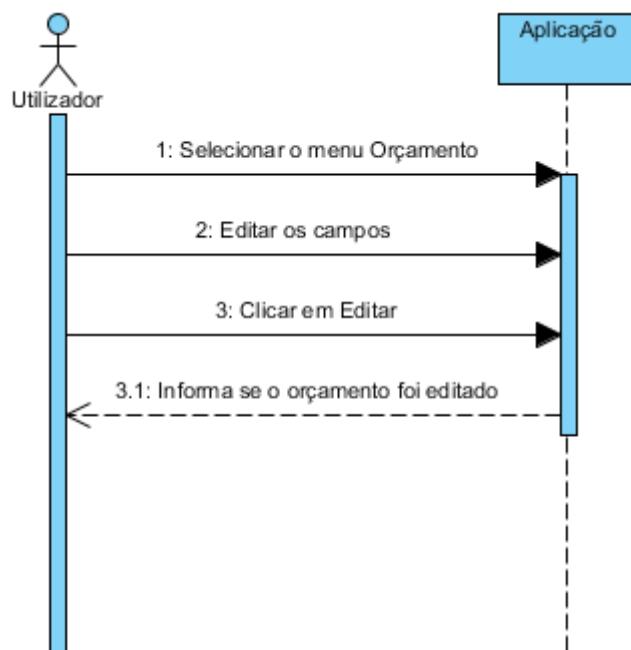


Figura 43: Diagrama de sequência – Editar Orçamento

Caso de Uso 13– Remover Orçamento

identificador: UC13

nome: Remover Orçamento

descrição sumária: Caso de uso relativo à remoção de um orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento a remover na tabela
- 2) clicar no botão Remover

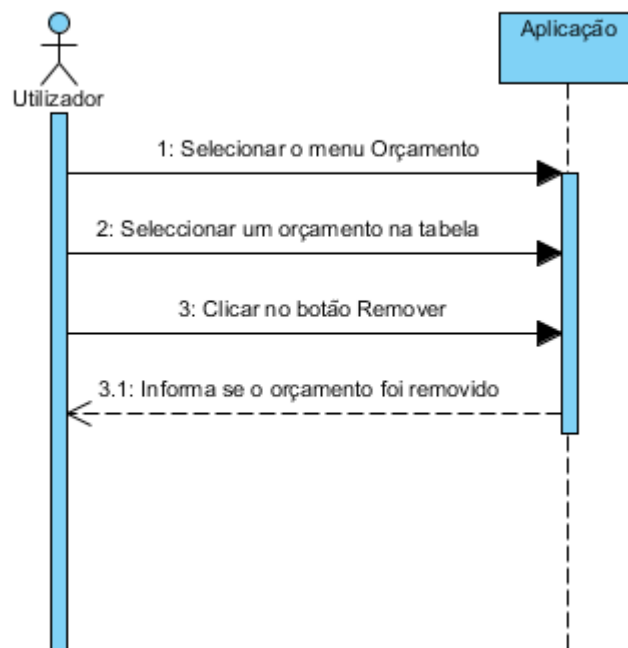


Figura 44: Diagrama de sequência – Remover Orçamento

Caso de Uso 14– Listar Orçamento

identificador: UC14

nome: Listar Orçamentos

descrição sumária: Caso de uso relativo à listagem de um ou vários orçamentos.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

1) seleccionar o orçamento a listar na tabela

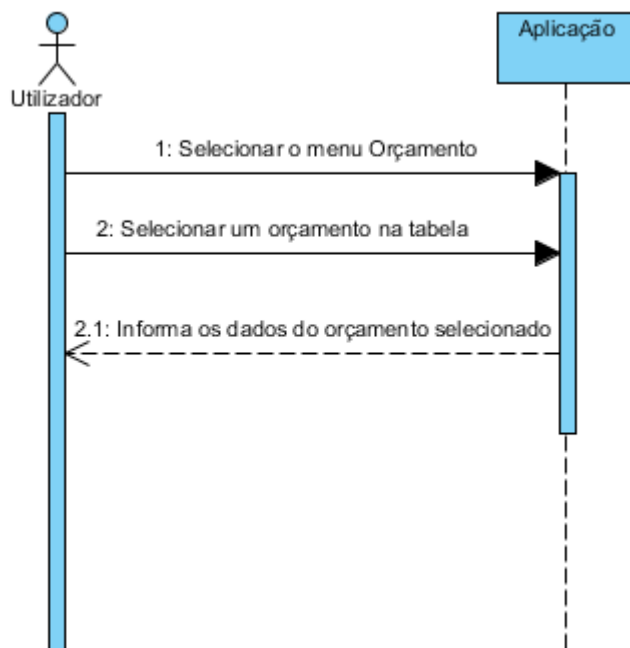


Figura 45: Diagrama de sequência – Listar Orçamento

Caso de Uso 15– Gerar avisos de débito

identificador: UC15

nome: Gerar avisos de débito

descrição sumária: Caso de uso relativo à geração de aviso de débitos relativamente a um Orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) seleccionar as frações , os meses e especificar o dia limite pagamento
- 3) clicar em Gerar Avisos

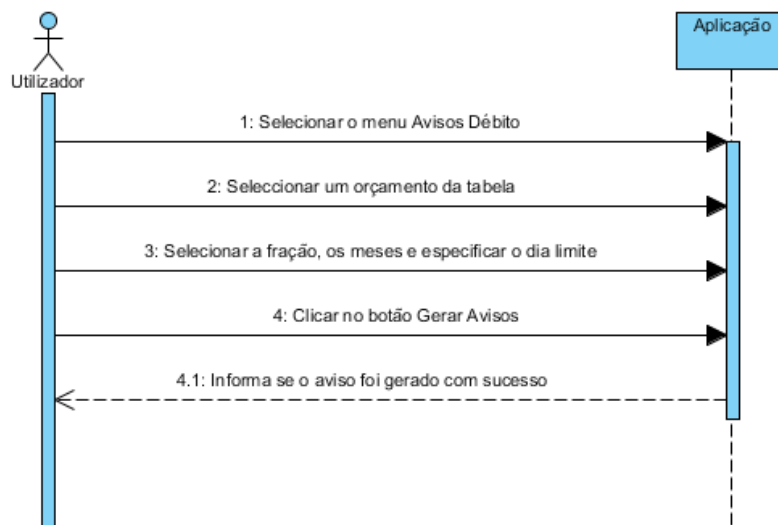


Figura 46: Diagrama de sequência – Gerar avisos de débito

Caso de Uso 16– Ver avisos de débito

identificador: UC16

nome: Ver avisos de débito

descrição sumária: Caso de uso relativo à pré-visualização dos avisos de débito gerados relativamente a um Orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) seleccionar o aviso de débito a pré-visualizar

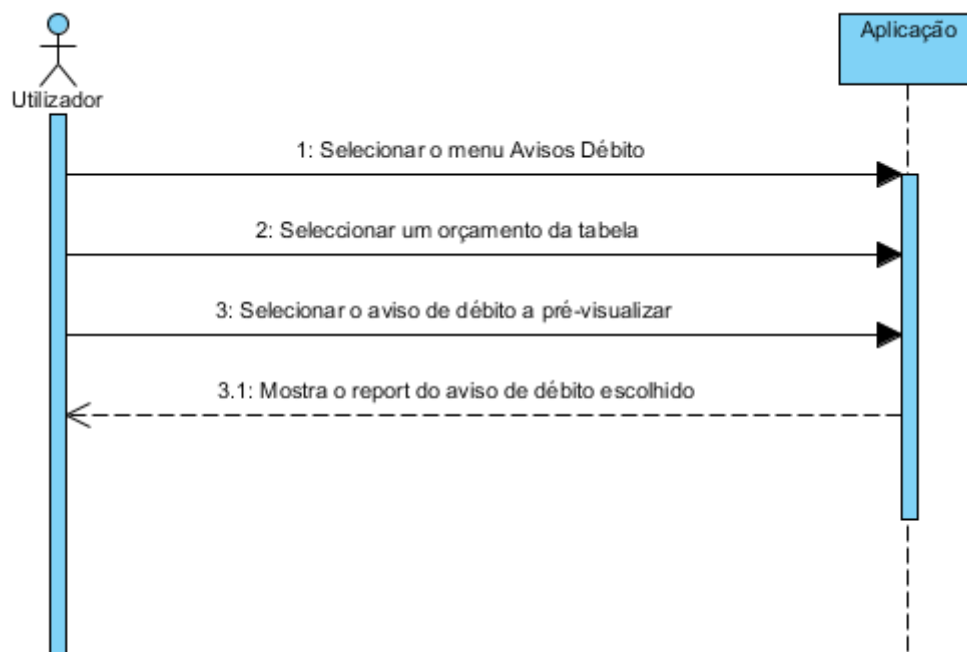


Figura 47: Diagrama de sequência – Ver avisos de débito

Caso de Uso 17– Filtrar avisos de débito

identificador: UC17

nome: Filtrar Avisos de débito

descrição sumária: Caso de uso relativo à pesquisa de um aviso de débito, por fração.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) introduzir o código da fração a filtrar
- 3) clicar no botão Filtrar

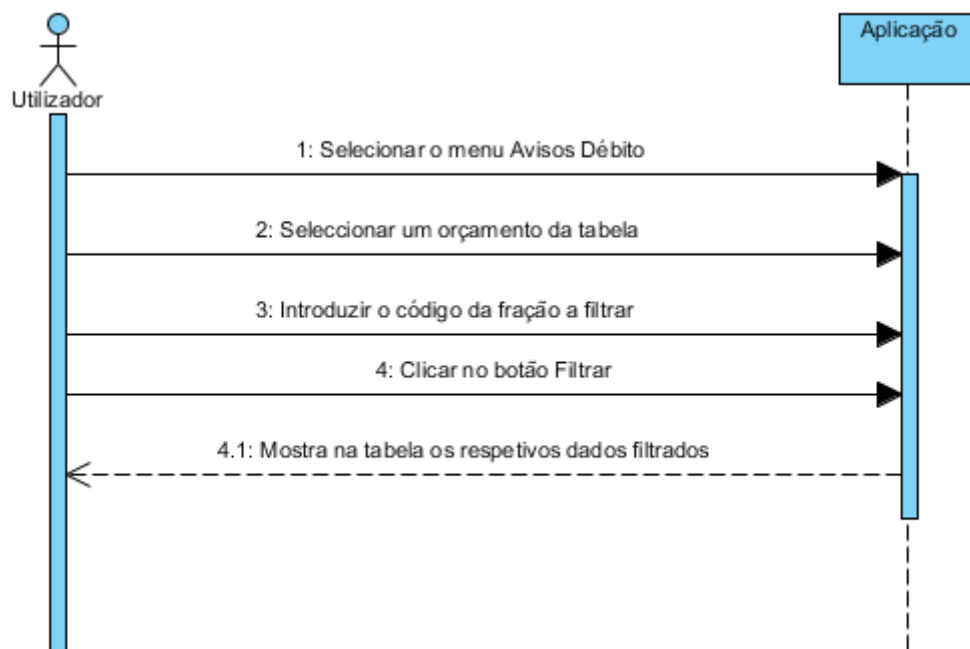


Figura 48: Diagrama de sequência – Filtrar avisos de débito

Caso de Uso 18– Enviar avisos de débito

identificador: UC18

nome: Enviar avisos de débito

descrição sumária: Caso de uso relativo ao envio de avisos de débito por email.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) seleccionar um ou vários avisos de débito
- 3) clicar no botão Enviar Email
- 4) introduzir o email e a password do condomínio
- 5) clicar no botão Login

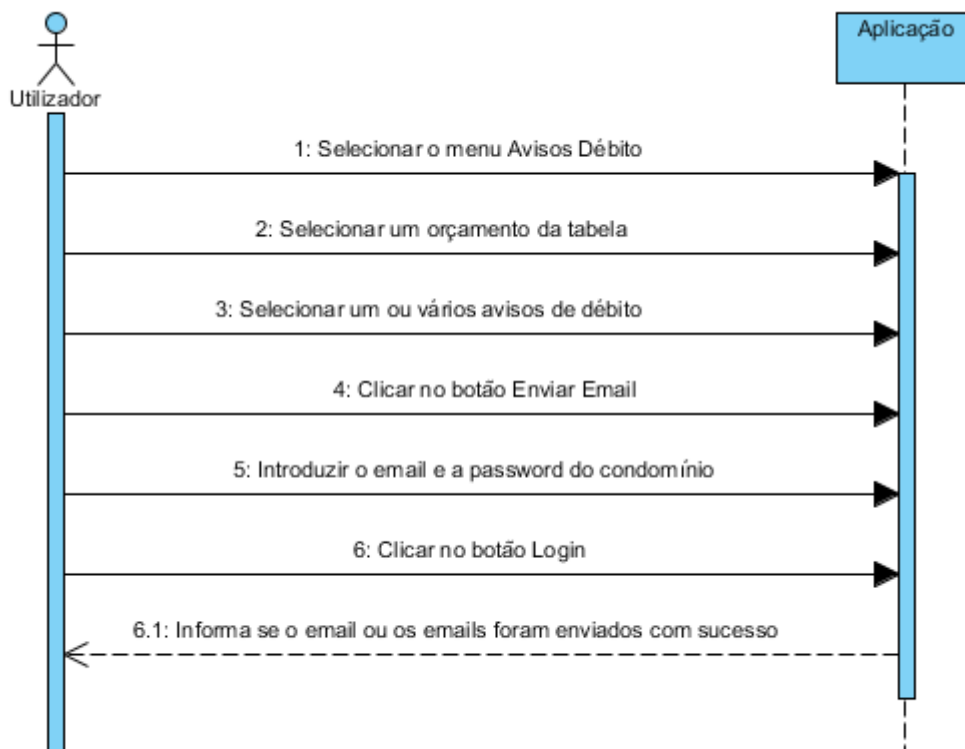


Figura 49: Diagrama de sequência – Enviar avisos de débito por email

Caso de Uso 19– Gerar Recibo

identificador: UC19

nome: Gerar Recibos

descrição sumária: Caso de uso relativo à geração de recibos relativamente a um Orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) seleccionar um ou vários avisos de débito da mesma fração não resolvidos
- 3) clicar no botão Gerar Recibo
- 4) introduzir os valores a pagar
- 5) clicar em gerar

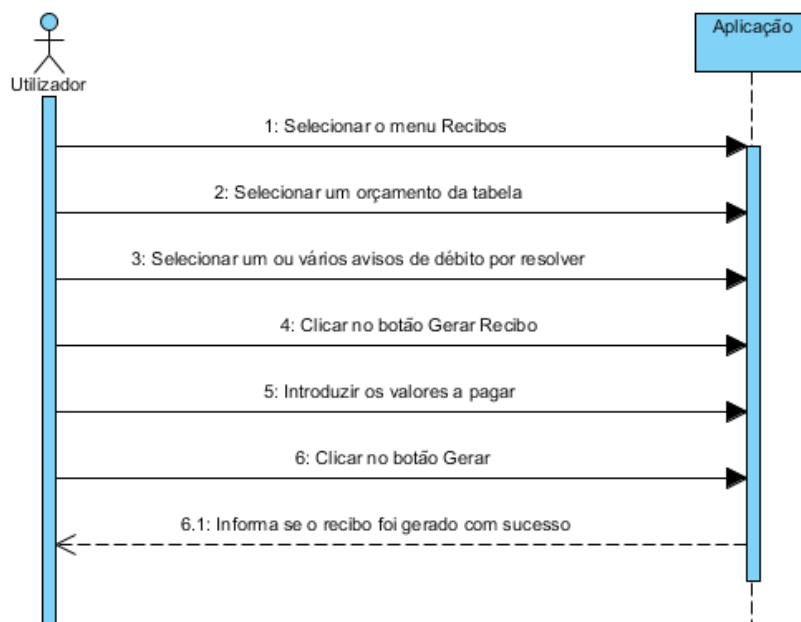


Figura 50: Diagrama de sequência – Gerar recibos

Caso de Uso 20– Ver Recibo

identificador: UC20

nome: Ver recibos

descrição sumária: Caso de uso relativo à pré-visualização dos recibos já gerados relativamente a um Orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) seleccionar o recibo a pré-visualizar

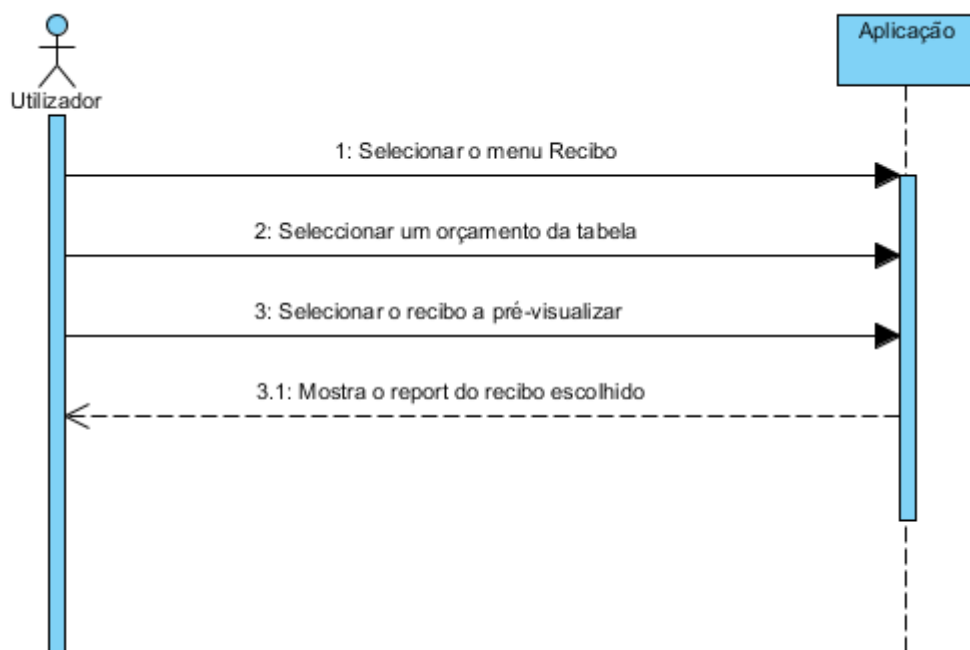


Figura 51: Diagrama de sequência – Ver recibos

Caso de Uso 21– Filtrar recibo

identificador: UC21

nome: Filtrar Recibo

descrição sumária: Caso de uso relativo à pesquisa de um recibo através do código da fração.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) introduzir o código da fração a filtrar
- 3) clicar no botão Filtrar

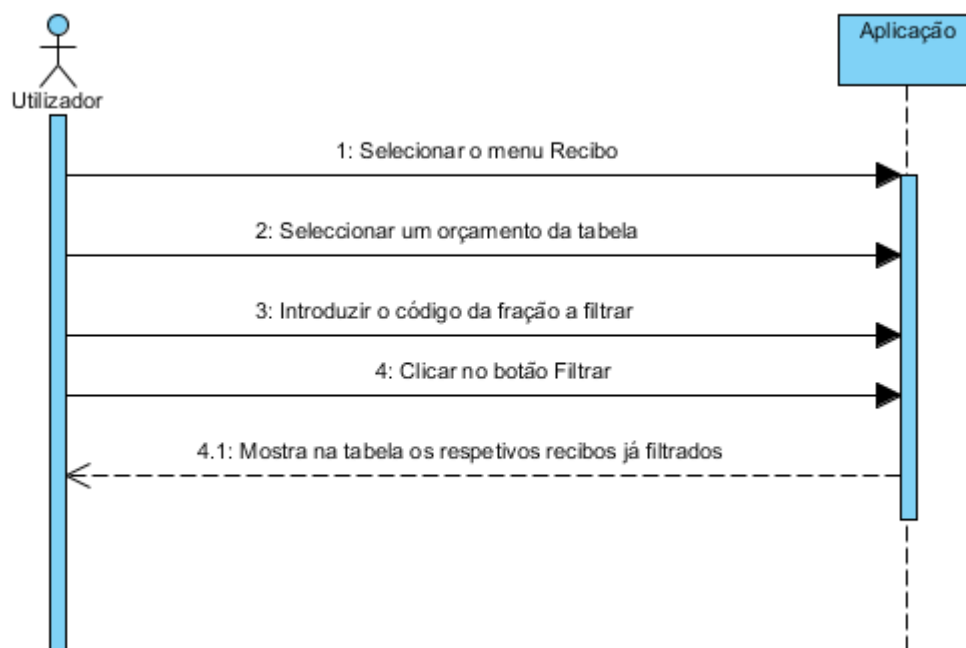


Figura 52: Diagrama de sequência – Filtrar recibos

Caso de Uso 22– Enviar recibo

identificador: UC22

nome: Enviar recibo

descrição sumária: Caso de uso relativo ao envio de recibos por email.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o orçamento na tabela
- 2) seleccionar um ou vários recibos
- 3) clicar no botão Enviar Email
- 4) introduzir o email e a password do condomínio
- 5) clicar no botão Login

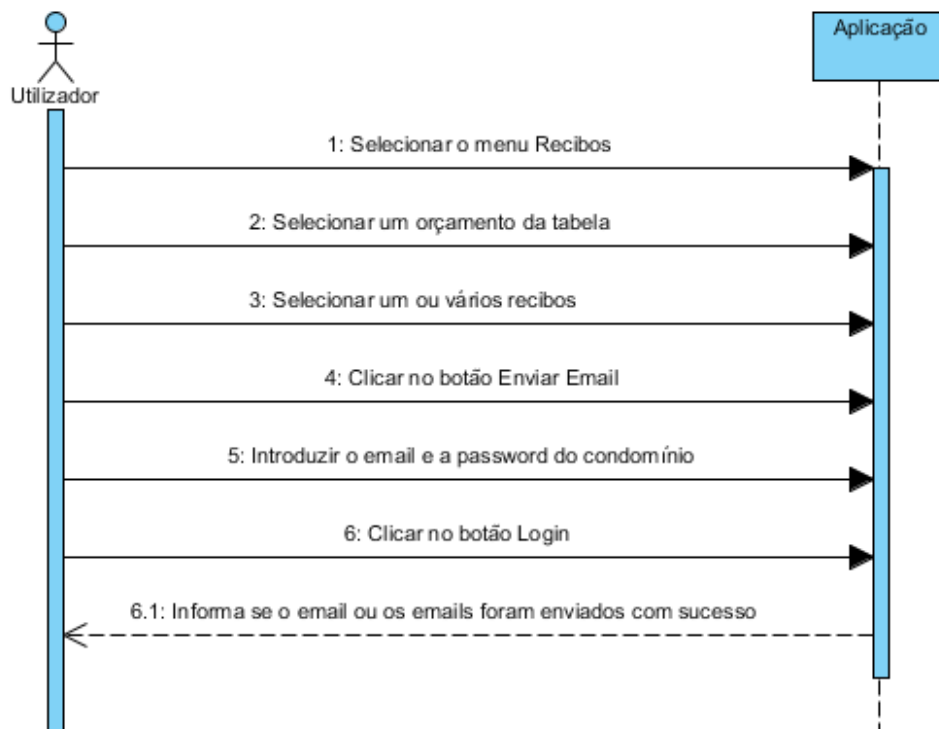


Figura 53: Diagrama de sequência – Enviar recibos por email

Caso de Uso 23– Inserir Despesa

identificador: UC23

nome: Inserir Despesa

descrição sumária: Caso de uso relativo à inserção de despesas para um determinado orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o menu Despesas
- 2) seleccionar um orçamento
- 3) preencher os campos necessários
- 4) clicar no botão Inserir

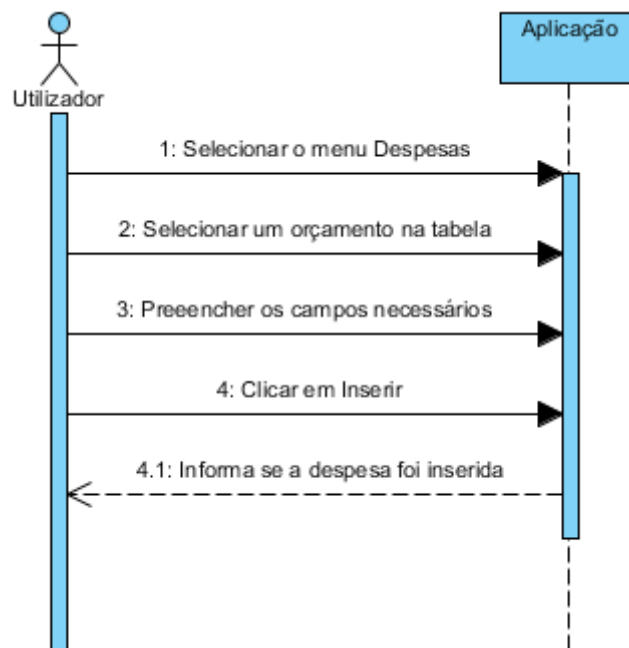


Figura 54: Diagrama de sequência – Inserir Despesa

Caso de Uso 24– Editar Despesa

identificador: UC24

nome: Editar Despesa

descrição sumária: Caso de uso relativo à edição de despesas para um determinado orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o menu Despesas
- 2) seleccionar um orçamento
- 3) editar os campos que pretender
- 4) clicar no botão Editar

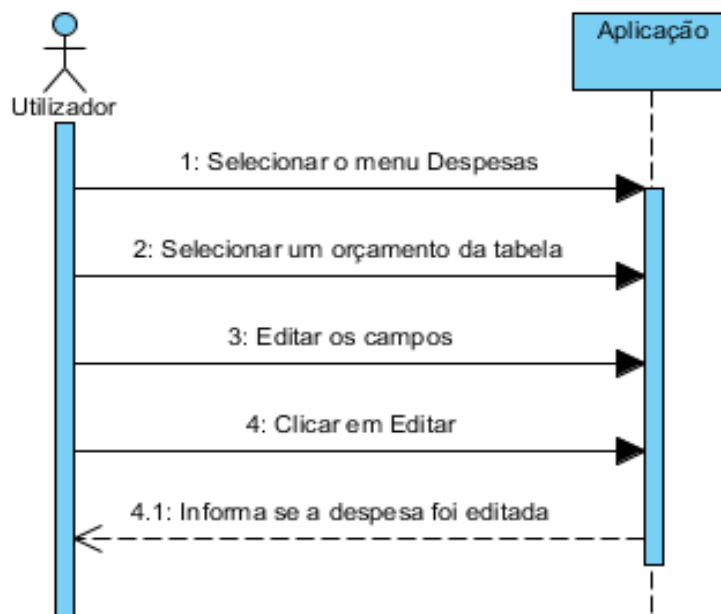


Figura 55: Diagrama de sequência – Editar Despesa

Caso de Uso 25– Remover Despesa

identificador: UC25

nome: Remover Despesa

descrição sumária: Caso de uso relativo à remoção de despesas para um determinado orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o menu Despesas
- 2) seleccionar um orçamento
- 3) seleccionar a despesa a remover
- 4) clicar no botão Remover

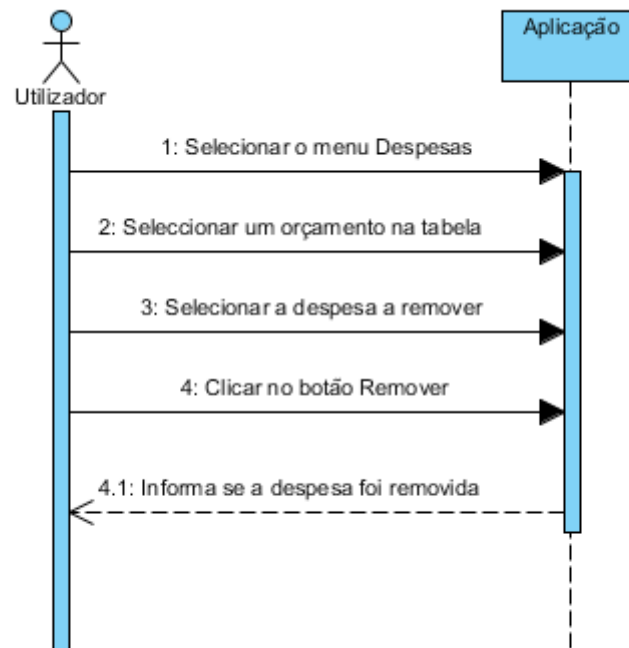


Figura 56: Diagrama de sequência – Remover Despesa

Caso de Uso 26– Listar Despesa

identificador: UC26

nome: Listar Despesa

descrição sumária: Caso de uso relativo à listagem de despesas para um determinado orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

1) seleccionar o menu Despesas

2) seleccionar um orçamento

3) seleccionar uma despesa

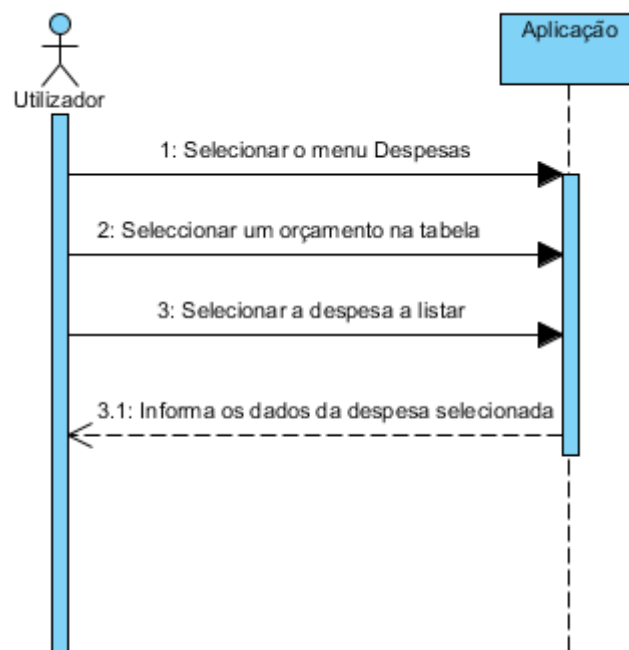


Figura 57: Diagrama de sequência – Listar Despesa

Caso de Uso 27– Ver Documentos

identificador: UC27

nome: Ver Documentos

descrição sumária: Caso de uso relativo à visualização de documentos para um determinado orçamento.

ator(es): o utilizador da aplicação é o único interveniente neste caso de uso, sendo o próprio a iniciar a interação.

prioridade: média

sequência de funcionamento / fluxo de eventos:

- 1) seleccionar o menu Documentos
- 2) seleccionar um documento e um orçamento
- 3) clicar no botão Ver

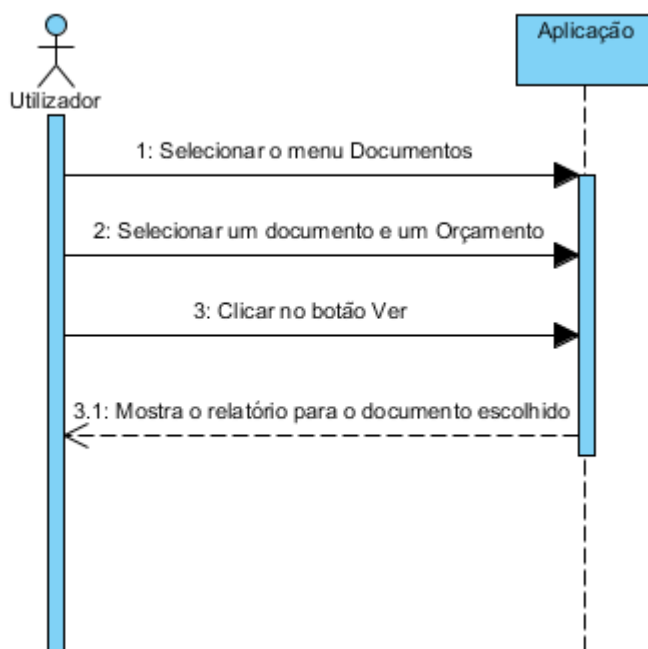


Figura 58: Diagrama de sequência – Ver Documentos

3.3 – Requisitos de informação

3.3.1- Diagrama de classes

Para melhor compreensão a representação das classes da aplicação, foram divididas em oito partes.

- 1- Diagrama de classes que representa as classes envolvidas na fase de inserção, edição e visualização de um **condomínio**.
- 2- Diagrama de classes que representa as classes envolvidas na fase de inserção, edição, remoção e visualização de uma **fração**.
- 3- Diagrama de classes que representa as classes envolvidas na fase de inserção e remoção de uma **rubrica**.
- 4- Diagrama de classes que representa as classes envolvidas na fase de inserção, edição, remoção e visualização de uma **orçamento**.
- 5- Diagrama de classes que representa as classes envolvidas na fase geração, pré-visualização, filtragem e envio por email de **aviso de débitos**.
- 5- Diagrama de classes que representa as classes envolvidas na fase geração, pré-visualização, filtragem e envio por email de **recibos**.
- 7- Diagrama de classes que representa as classes envolvidas na fase de inserção, edição, remoção e visualização de **despesas**.
- 8- Diagrama de classes que representa as classes envolvidas na fase de pré-visualização dos **documentos finais** relativamente a um orçamento.

3.3.2- Diagrama de classes – Condomínio

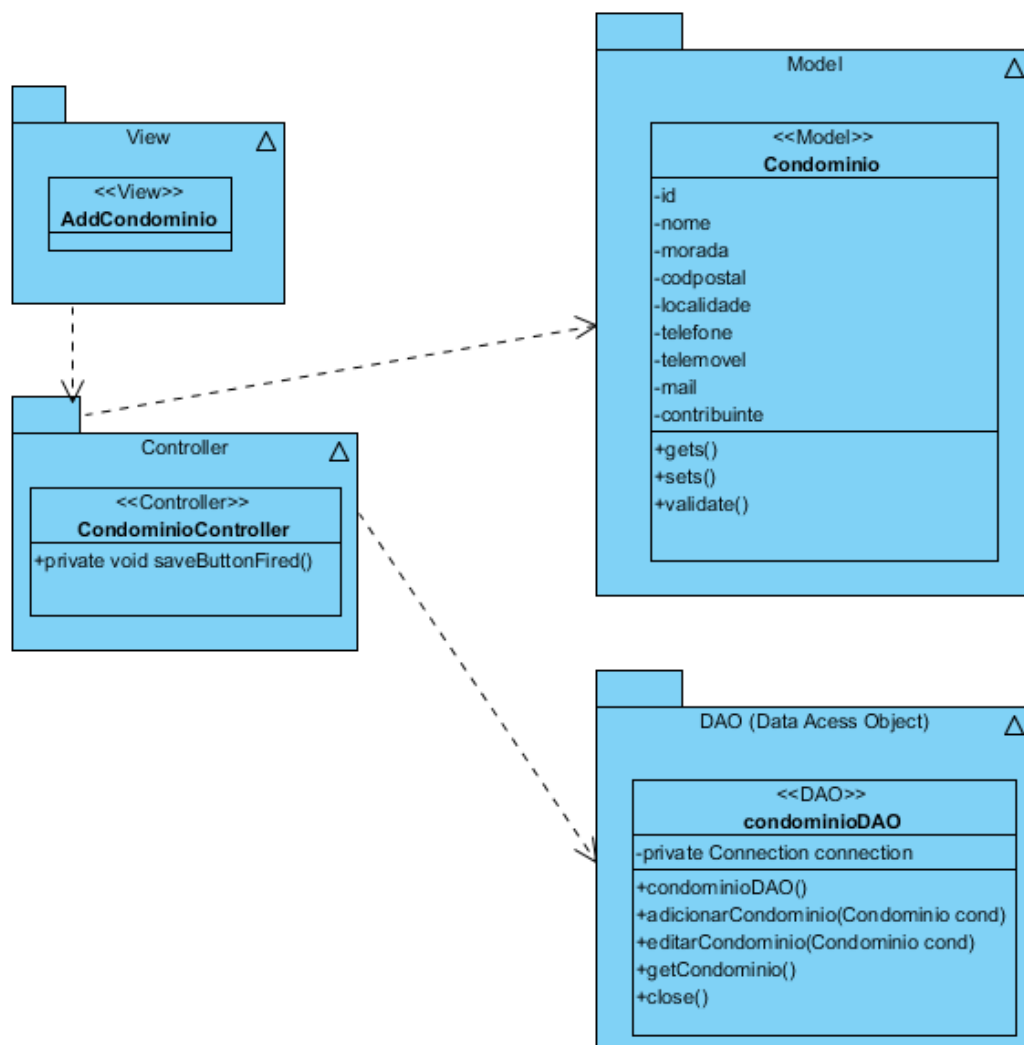


Figura 59: Diagrama de classes - Condomínio

3.3.2- Diagrama de classes – Fração

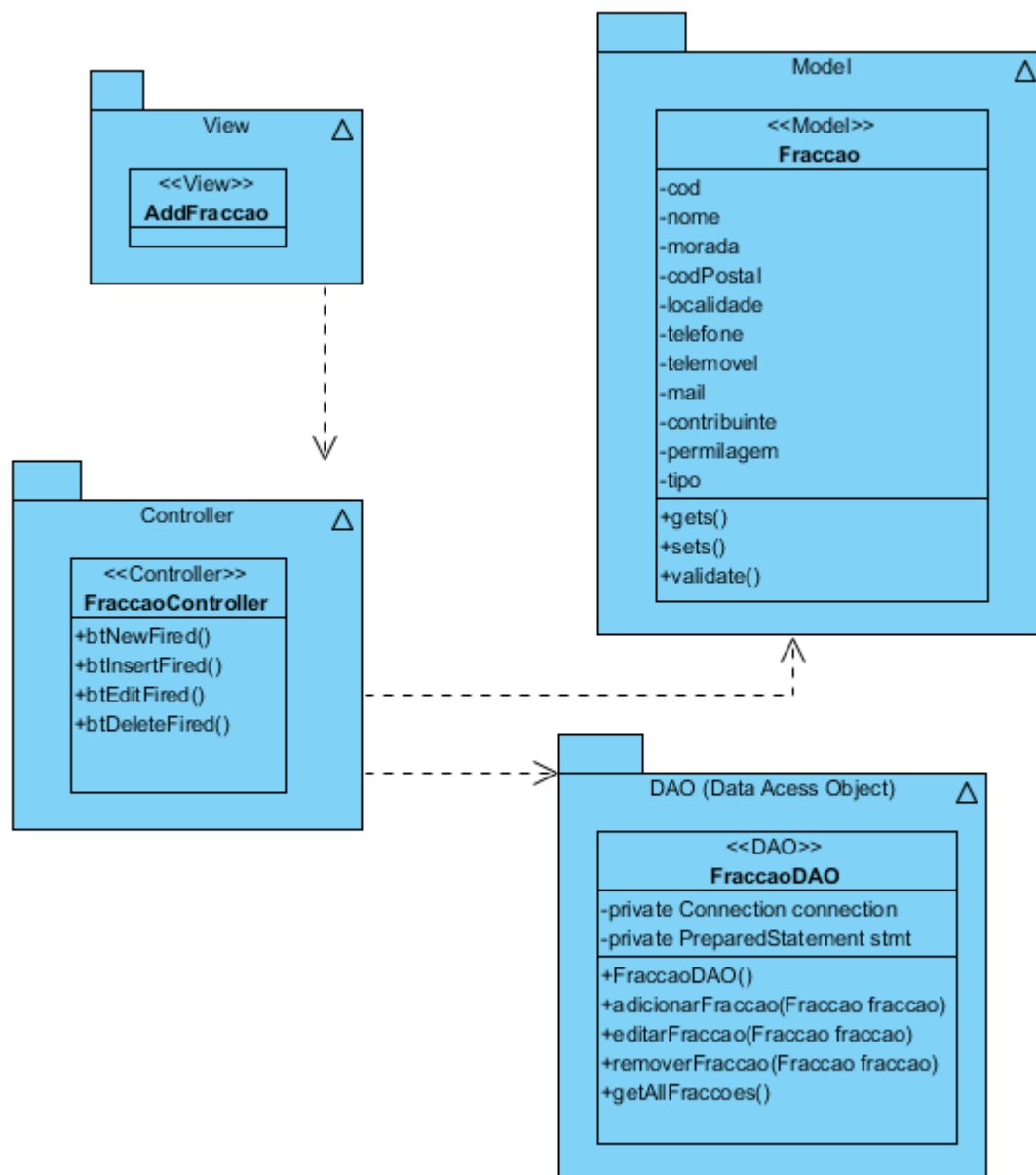


Figura 60: Diagrama de classes - Fração

3.3.2- Diagrama de classes – Rubrica

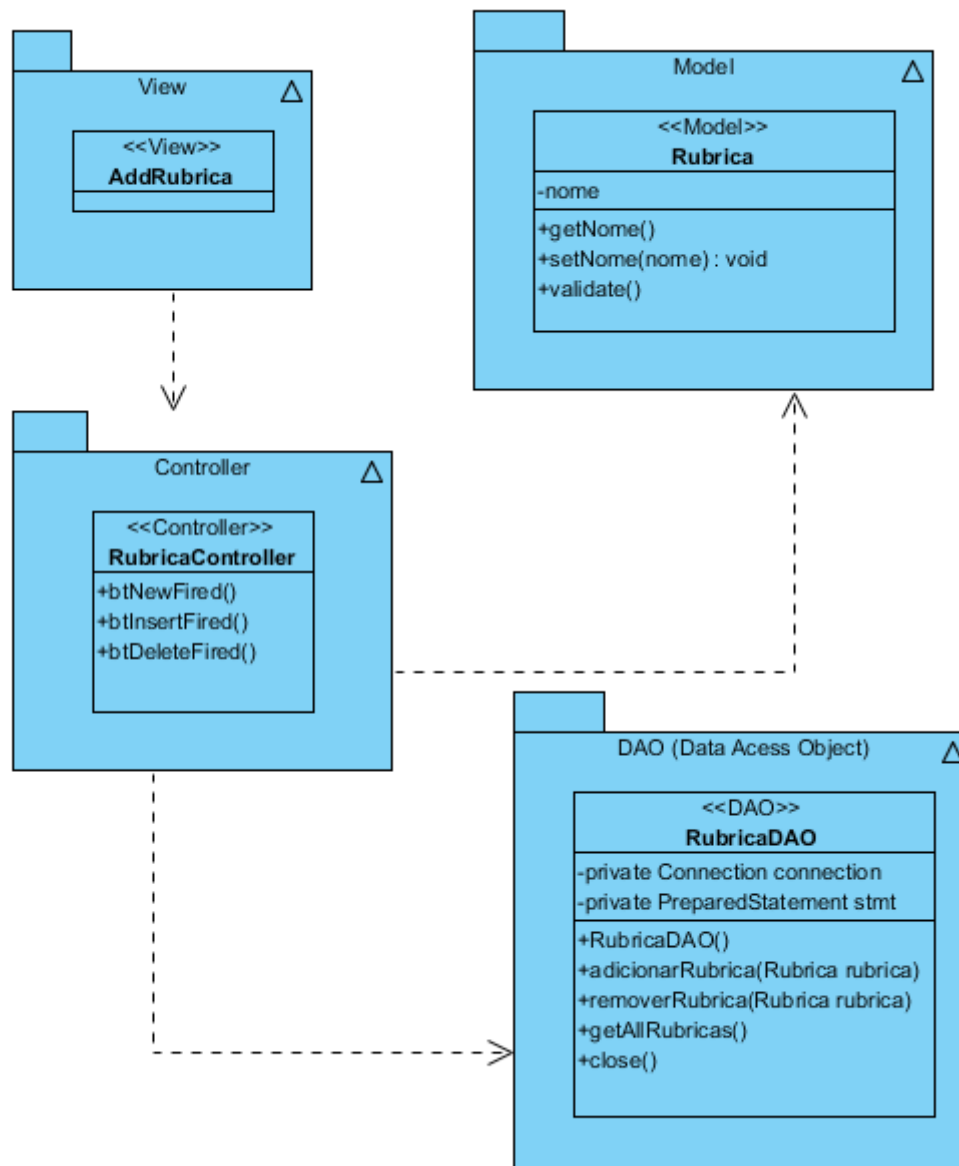


Figura 61: Diagrama de classes - Rubrica

3.3.2- Diagrama de classes – Orçamento

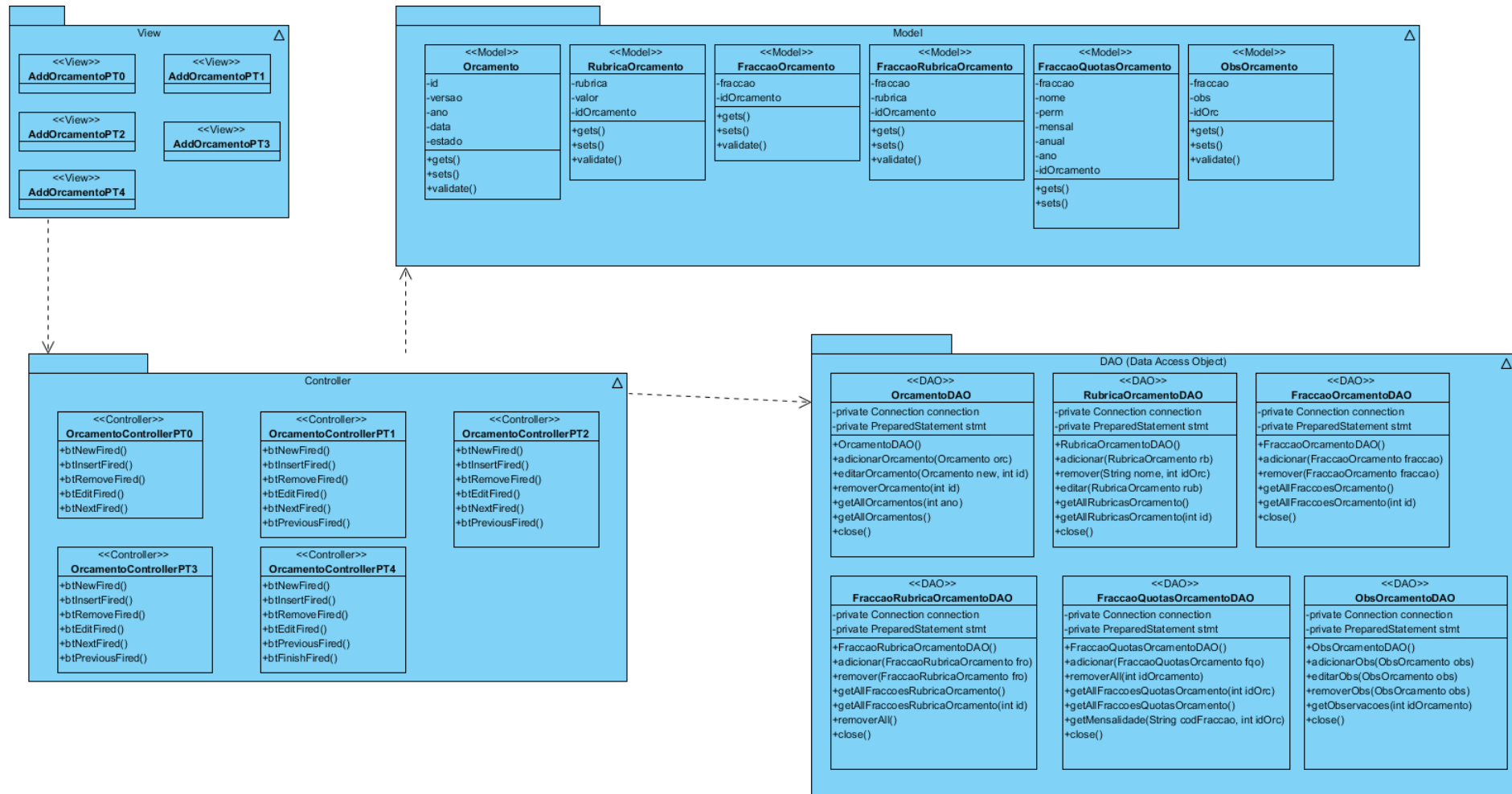


Figura 62: Diagrama de classes - Orçamento

3.3.2- Diagrama de classes – Aviso de Débito

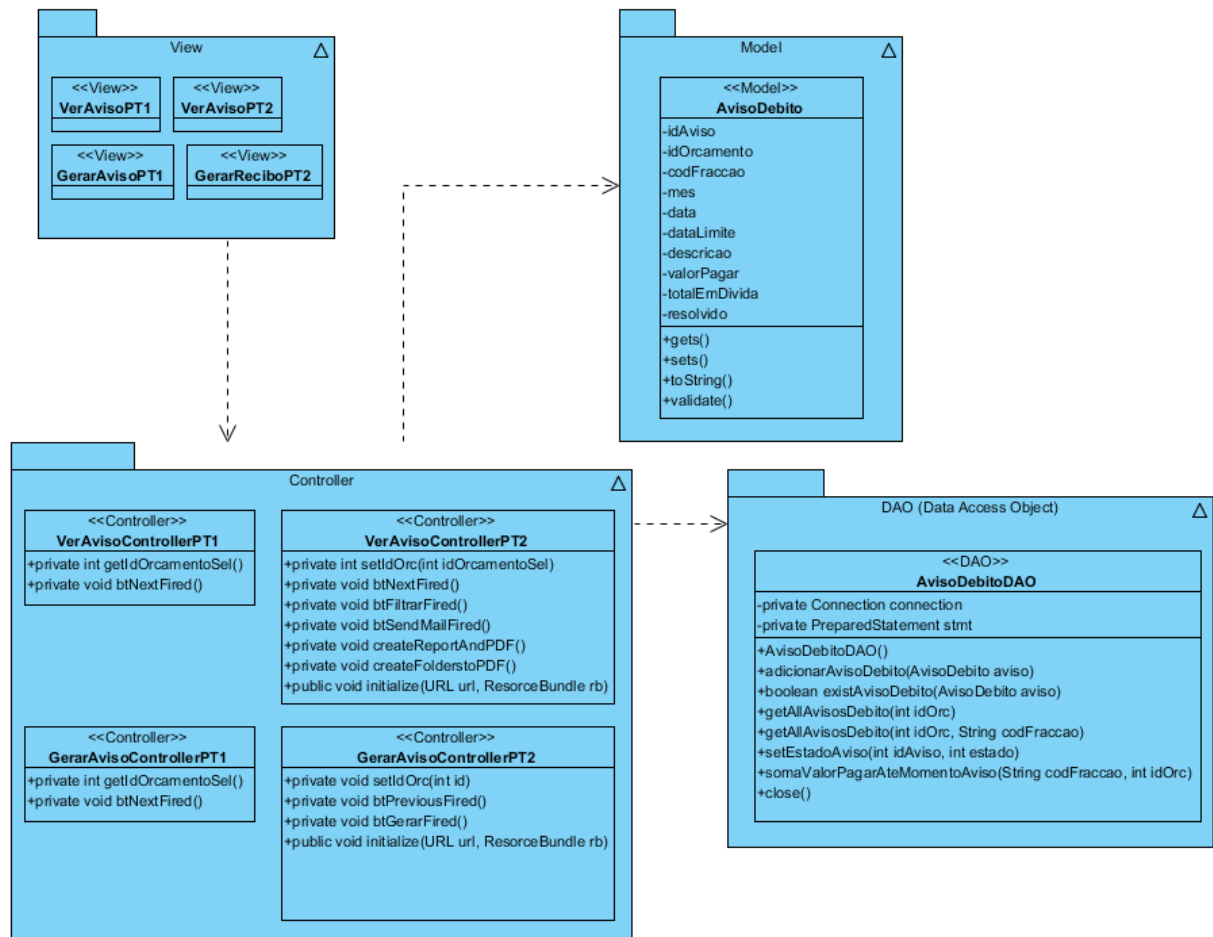


Figura 63: Diagrama de classes - Aviso de Débito

3.3.2- Diagrama de classes – Recibos

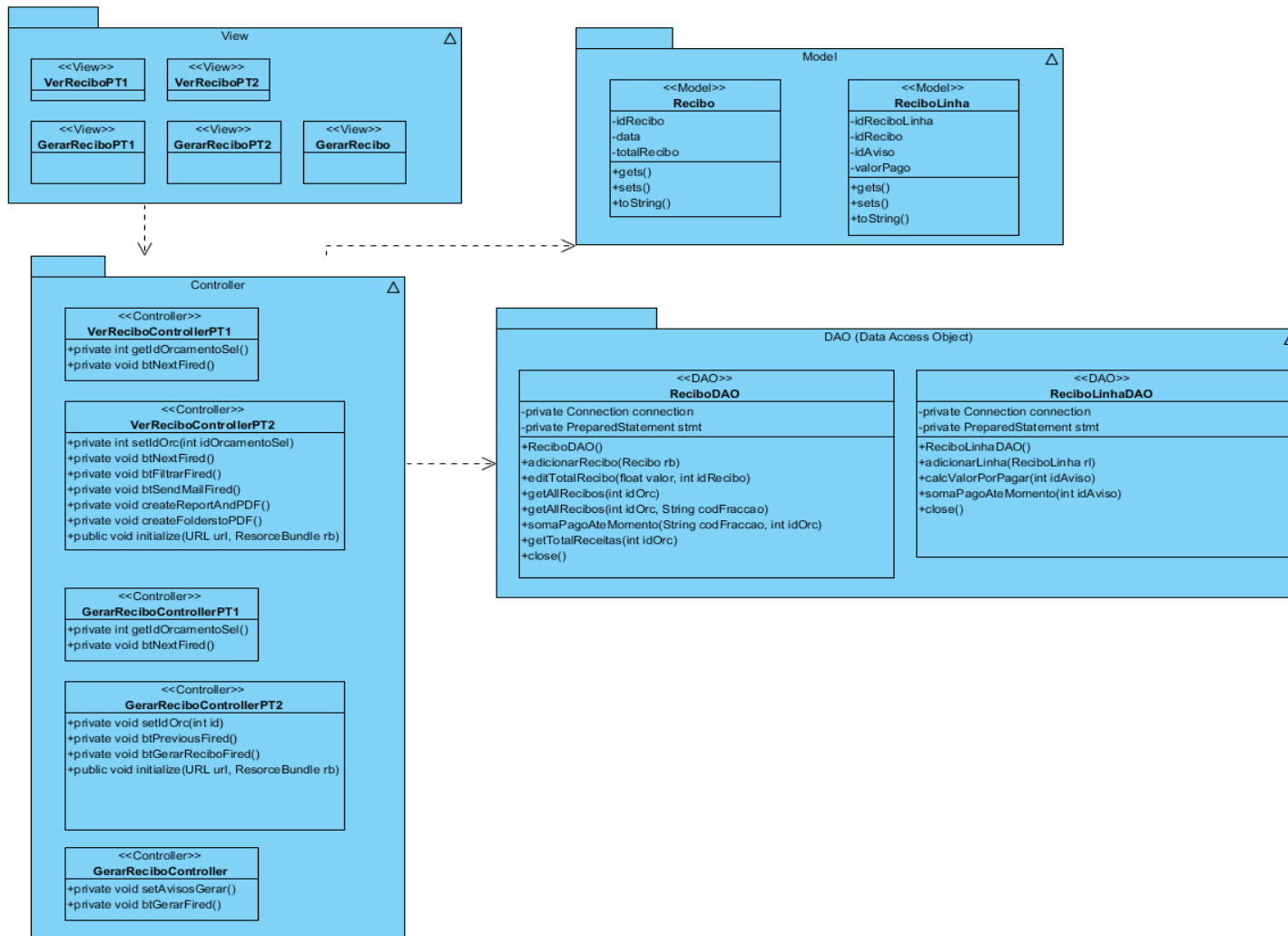


Figura 64: Diagrama de classes - Recibos

3.3.2- Diagrama de classes – Despesas

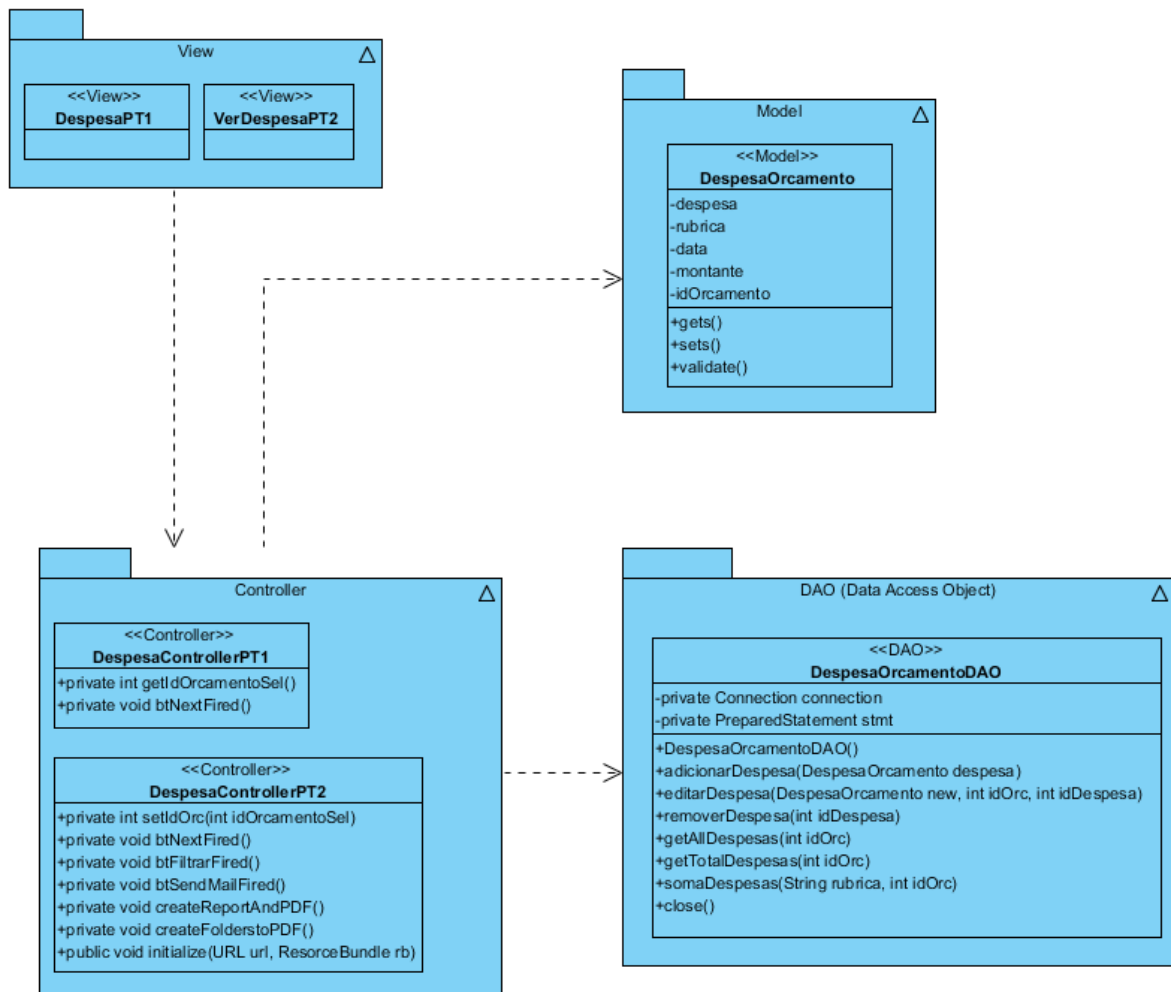


Figura 65: Diagrama de classes - Despesas

3.3.2- Diagrama de classes – Ver documentos

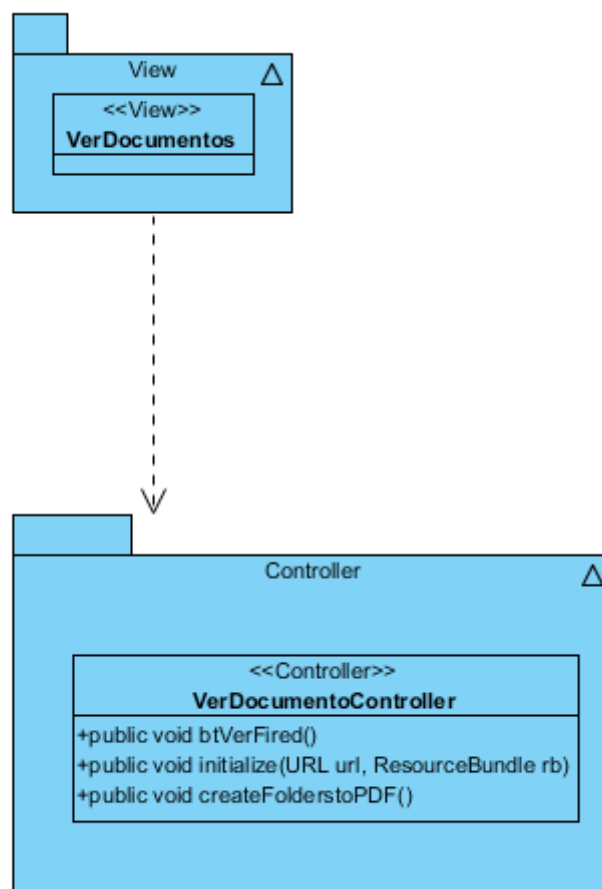


Figura 66: Diagrama de classes - Ver Documentos Finais

3.4 – Requisitos não-funcionais

Estes requisitos não estão diretamente relacionados com as funcionalidades, mas são igualmente importantes para o sistema. Descrevem o sistema do ponto de vista da sua disponibilidade, segurança, eficiência, entre outros, igualmente importantíssimos na qualidade do produto desenvolvido, definindo assim os objetivos que um sistema deverá cumprir. De seguida são apresentados esses requisitos.

3.4.1. Disponibilidade

Deverá ser uma aplicação resistente a falhas em que de maneira alguma possa ser um fator que impossibilite o funcionamento da mesma, de modo a que este esteja sempre disponível.

3.4.2. Eficiência

O tempo de execução das operações deve ser o mínimo possível, obtendo-se assim uma boa eficiência dando uma melhor experiência ao utilizador.

3.4.3. Manutenção

A aplicação deve ter um javadoc bem estruturado e comentado, ajudando à compreensão do código em manutenções futuras. O código deve estar bem organizado e pensado de maneira a possibilitar alterações no futuro com alguma facilidade e até mesmo a inserção de mais funcionalidades.

3.4.5. Segurança

Devem se fazer as validações necessárias para manter a integridade dos dados, não permitindo apagar nem editar dados importantes e que vão comprometer a regra de negócio da aplicação.

3.4.6. Usabilidade

aplicação deverá ser desenvolvida na linguagem Java.