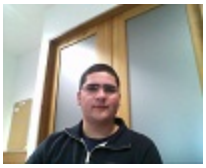
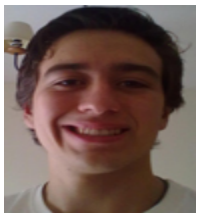
Desenvolvimento de sistemas de software

Aplicação: *"Gestão Habitat"*

José Pereira(67680)



Pedro Cunha(67677)



Rui Oliveira (67661)

Tomás Ferreira (67701)

Índice

Introdução

É pretendida a realização de uma aplicação para a instituição sem fim lucrativos Habitat. Esta aplicação deve obedecer aos requisitos presentes neste relatorio, que foram discutidos numa reunião com o representante da Habitat.

O desenvolvimento da aplicação irá obedecer á forma de desenvolvimento aprendida na unidade curricular de DSS, que irá incluir: analise de requisitos, modelo de domínio, modelo de *use cases*, proposta de interface para aplicação com os seus diagramas de estado e a sua posterior implementação. De seguida irá ser explicado os diagramas de sequência que foram criados: de sistema, de sistema com sub-sistemas e implementação. Depois irá-se falar sobre o diagrama de package e logo a seguir das diferentes etapas dos diagramas de Classes a especificaçãoe a implementação.

Para finalizar utilizando um diagrama de class e outro de sequência irá ser explicado o funcionamento da interface da aplicação.

Resumo

O presente relatorio pretende ilustrar o trabalho realizado no desenvolvimento da aplicação para a instituição Habitat. Para além do já incluído no relatório da primeira etapa, este relatório terá presente os diagramas de Sequência de Sistema para cada use case. Em seguida encontrar-se-á o diagrama de Package. Depois disso teremos os diagramas de Sequência, diagramas de Classe (de especificação, de implementação com Maps e de implementação com DAO). Finalmente apresentaremos a documentação do código que produzimos.

Analise de Requisitos

A habitat é constituído por 4 órgãos:

- Direção
- Comissão de fundos
- Comissão de famílias
- Comissão de construção

A **direção** é o orgão máximo da habitat, e todas as inscrições têm que ter a aprovação da direção

A **comissão de fundos** é responsável por gerir os donativos, criar eventos com o objetivo de angariar fundos e voluntários.

A **comissão de famílias** é responsável por gerir as inscrições das famílias, tratar do acompanhamento com a família após a execução da obra.

A **comissão de contrução** é responsável por gerir as obras, criando um plano para a cada uma e acompanhado-a.

Logica de inscrição

1. **Família**: inscreve-se no programa.
2. **CF**: marca uma data para uma reunião com a família, onde é entregue um questionário detalhado sobre a constituição da família e os seus rendimentos.
3. **Família**: responde ao questionário e entrega.
4. **Direção**: aprecia o processo e aceita ou não.
5. **CF**: Faz primeira apreciação do questionário da família
6. **Direção**: Aprova ou não o projeto
7. **CC**: elabora o projeto preliminar e orçamento.
8. **Família**: aceita ou não o orçamento.
9. **CC** / CF: são mobilizados voluntários e angariação de fundos e donativos (materiais).
10. **CC**: realiza plano de construção.
11. **CC**: inicia obra.
12. **CC**: de semana em semana é feito o balanço das tarefas que já foram feitas para perceber se está atrasado.
13. **CC**: fim da obra
14. **Família**: muda-se para nova casa
15. **Família**: passa a pagar uma prestação mensal variável e sem juros, até pagar a casa.
16. **CF** (enquanto a família paga a casa): vigia o estado da casa.

Requisitos

1. Geral
    a. Deve ser possível gerir de forma fácil cada uma das comissões, suportando toda a lógica de inscrição, por cada uma das partes.
    b. Cada Comissão edita e mantém os dados respetivos, contudo pode consultar os dados das restantes comissões.
    c. Ter acesso toda a informação; voluntários, projetos, doadores, famílias;
    d. Só funcionários é que podem usar a aplicação e administradores para coordenar a aplicação;

2. Sobre Voluntários/doações:
    a. Existem dois tipos de voluntários: individuais e equipas organizadas.
    b. A equipa tem uma designação.
    c. Existem doações em dinheiro e/ou espécie (materiais, equipamentos ou serviços, como por exemplo horas de trabalho de trabalhadores de uma empresa de construção, ou horas de equipamento cedido) e são feitas por empresas ou indivíduos.
    d. Existem doações de serviços de caracter genérico, ou seja que não são diretamente aplicáveis a uma obra, sendo aplicadas à Habitat no geral.
    e. Existem eventos de angariações de fundos e são geridos pela comissão de fundos, caso as doações sejam provenientes de um evento deve ser possível saber qual o evento.
    f. Deve ser possível saber a origem e destino do donativo (dinheiro ou espécie), sabendo as tarefas onde foi utilizado.
    g. Relacionar projetos com doadores para saber quem deu para quê;

3. Sobre Construção:
    a. Deve ser possível abrir e fechar o registo de uma obra, guardando a duração e custo estimados e finais.
    b. A obra tem data de fim de garantia a partir do qual as reparações deixam de ser obrigatórias por parte da habitat.
    c. Deve ser possível guardar um registo do planeamento e orçamento da obra (antes da construção)
    d. As obras são constituídas por um conjunto de tarefas para as quais são efetuados registos de horas de voluntariado e recursos gastos,
    e. Deve ser possível definir a data de abertura e conclusão da tarefa.
    f. Deve ser possível comparar o planeamento com a execução da obra.
    g. Deve ser possível aceder ao inventário do projecto, nº horas de voluntariado e dinheiro gasto e onde o dinheiro que foi gasto, saber em que obra as doações foram usadas;
    h. Deve ser possível identificar um projeto como urgente.
    i. Deve poder haver mais que uma obra simultaneamente.

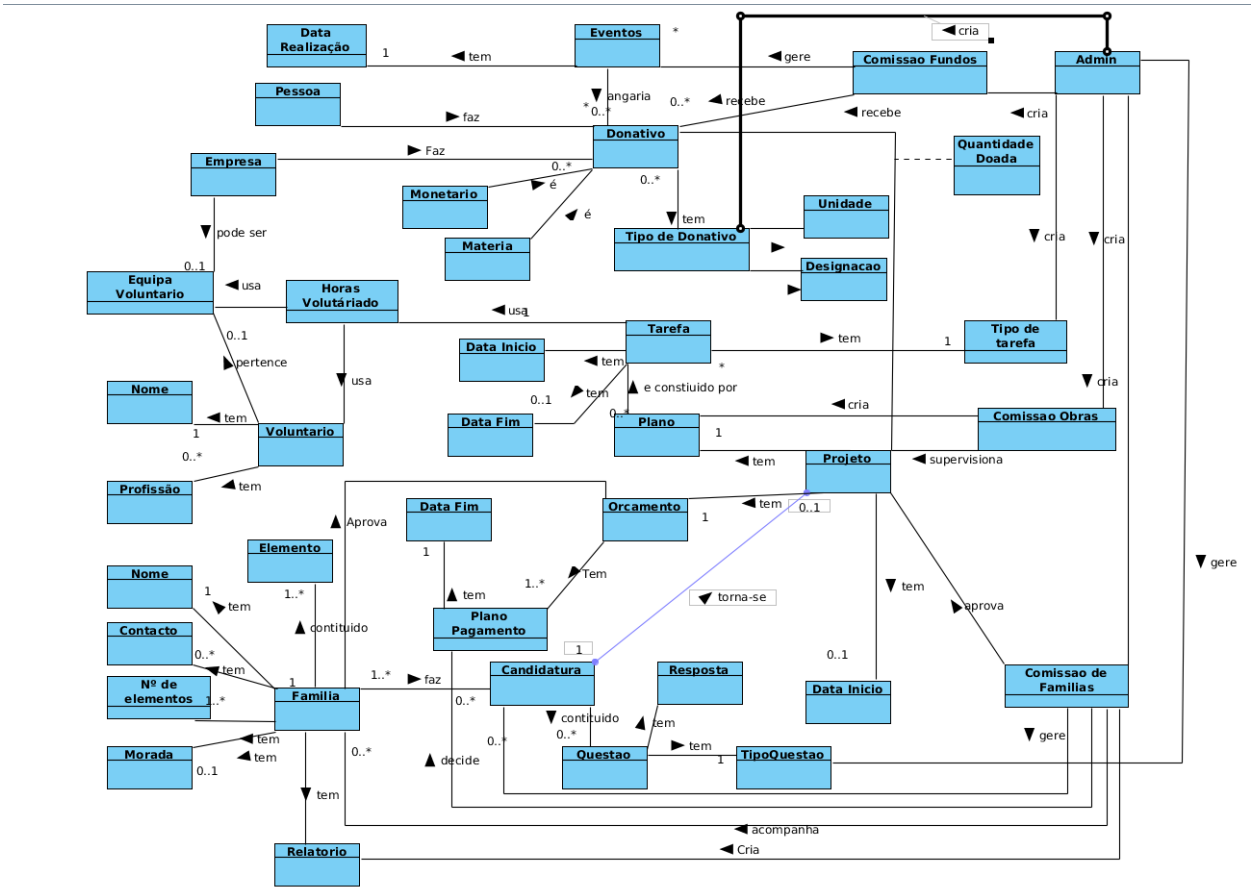
j. A ficha da obra pode ser reaberta a qualquer momento de forma a registrar reparações nas casas (novas tarefas);

4. Sobre Família
    a. Uma família só tem uma candidatura que pode ser reaberta.
    b. Direção decide aprovação de uma candidatura,
    c. Uma processo de candidatura pode ser catalogado como: aprovado, não aprovado, não aceite pela família, em construção e concluído.
    d. São realizados relatórios de acompanhamento apos a entrega da habitação.
    e. É necessário conhecer o histórico de prestações e poder actualizá-las

5. Outros
    a. Empresas podem escolher se doam para Habitat ou para uma família específica;
    b. Terreno pode ser doado ou da família ajudada;
    c. Material é divisível, uma doação pode servir para vários projetos;

Modelo de Domínio

Esta foi a primeira etapa de desenvolvimento do projeto e tem como objetivo uma primeira interpretação do problema.



Alguns exemplos de leitura deste diagrama:

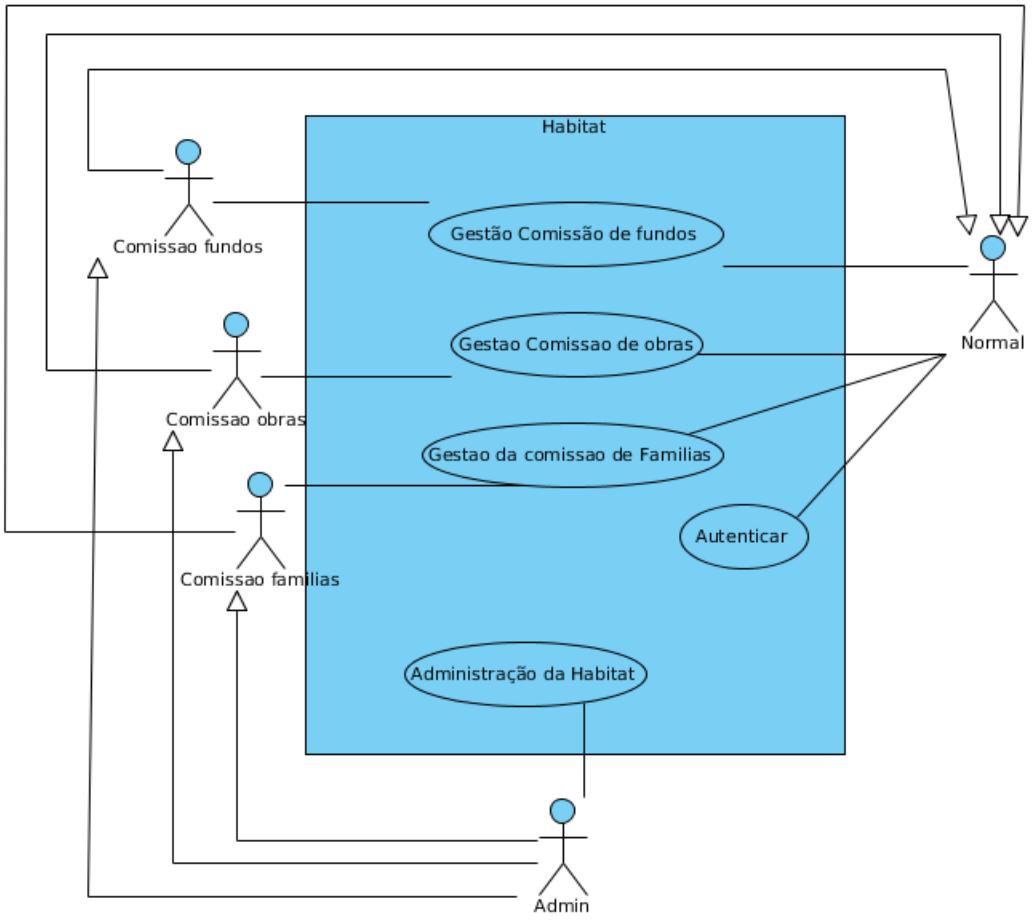
- Uma Família faz Inscrição
- Uma Inscrição pode tornar-se num projeto.
- Um Projeto tem um Plano
- Um Plano é constituído por várias Tarefas.
- Uma Tarefa Usa Horas de Voluntariado.

Modelo de Use cases

Esta foi a segunda etapa de desenvolvimento do projeto, tendo com objetivo conhecer exatamente o que cada utilizador deverá ser capaz de fazer na aplicação.
Foram criados 4 perfis de utilização, são eles:

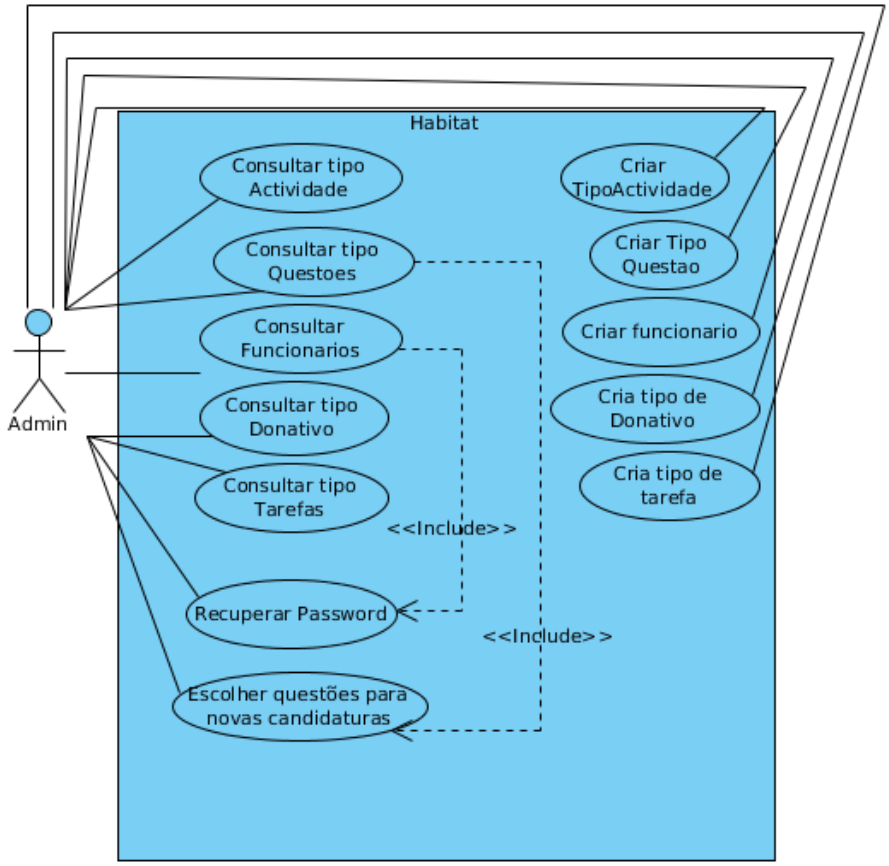
- administrador,
- comissão de angariação de fundos,
- comissão de Obras
- comissão das Famílias.

Todos os 4 perfis podem consultar a informação e autenticar-se. Notando-se que a única ação que um utilizador pode fazer sem estar autenticado é autenticar-se.
O Diagrama de Use case foi dividido em 4 por motivos de organização, nas seguintes secções irão se falar de cada um.



Administração da Habitat

O administrador da aplicação é responsável por gerir os funcionários, e algum tipo de informação que foi tipificada tal como o Tipo de Actividade, Tipo de Questao, Tipo de Donativo e Tipo de tarefa. Poderá portanto consultar Toda a informação e Criar nova.
Tal como podemos constar todos os Criar e Consultar são idênticos entre si.
Apenas foram especificados os Use cases Consultar Funcionários e Criar Funcionários. Todos os outros são semanticamente equivalentes



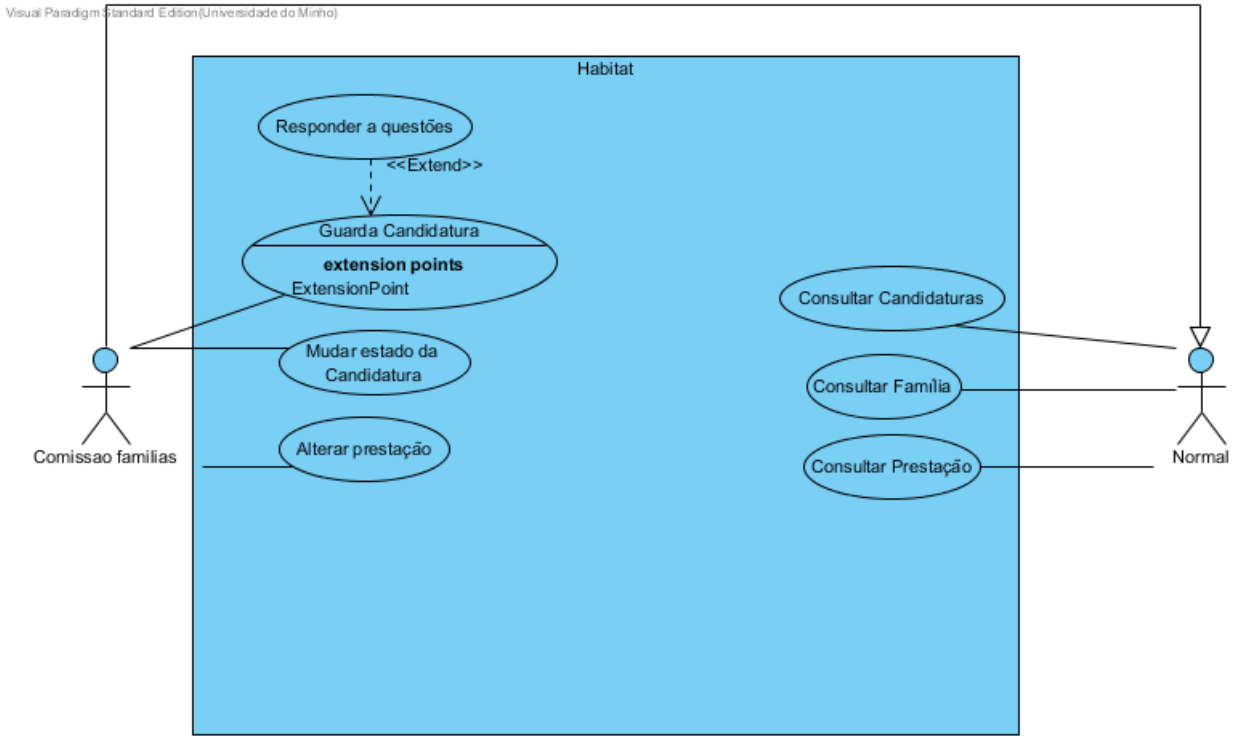
A especificação de “Recuperar Password” é a seguinte, ou seja para se recuperar a password de um utilizador primeiro este terá que ser procurado.

		Actor Input	System Response
Flow of Events	1	<<include>> consultar funcionario	
	2	Defenir nova password	
	3		Password e guardada

As restantes especificações podem ser encontradas em anexo

Comissão de Famílias

O funcionário da comissão de famílias é responsável por guardar as candidaturas das famílias na base de dados, alterar o estado das candidaturas e alterar a prestação paga pelas famílias. Pode também consultar a informação sobre as candidaturas, famílias e prestações.
Apenas foram especificados os Use Cases do Consultar Candidaturas pois todos os outros Use Cases de consulta são semanticamente equivalentes.



A especificações de por exemplo Consultar Candidatura:

Brief Description	O utilizador consulta os dados da candidatura de uma família
Preconditions	Estar autenticado
Post-conditions	
Flow of Events	

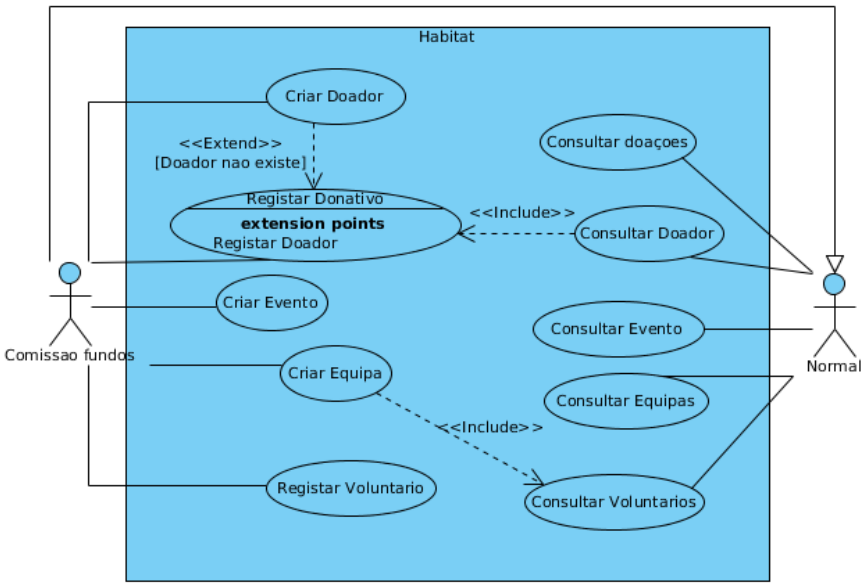
	Actor Input	System Response
1		Apresenta lista das candidaturas
2	Seleciona a candidatura a apresentar	
3		Retorna os dados da candidatura

As restantes especificações podem ser encontradas em anexo.

Comissão de fundos

O funcionário da comissão de fundos é responsável por guardar a informação sobre os doadores, donativos, eventos, equipas e voluntários. Toda esta informação pode ser consultada por todos os elementos da habitat.
Apenas foram especificados os Use Cases do Consultar doações e voluntários pois todosos outros Use Cases de consulta são semanticamente equivalentes.

dasasdsa



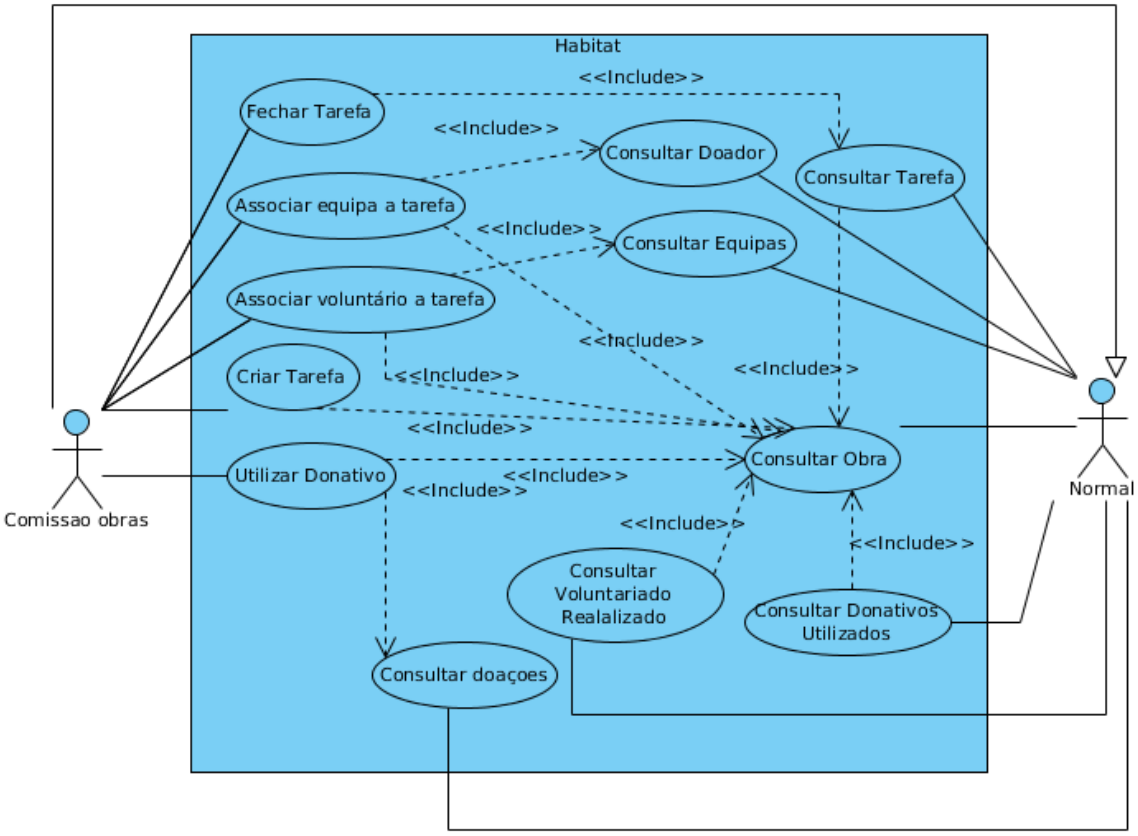
Especificação de registar Donativo

Preconditions	Estar autenticado
Post-conditions	
Flow of Events	<table><tr><th></th><th>Actor Input</th><th>System Response</th></tr><tr><td>1</td><td><<include>> consultar doador (registar doador)</td><td></td></tr><tr><td>2</td><td>Insere quantia monetaria a ser doada</td><td></td></tr><tr><td>3</td><td></td><td>Regista doação</td></tr></table>
Alternativo1 donativo não é monetario. Passo 2	<table><tr><th></th><th>Actor Input</th><th>System Response</th></tr><tr><td>1</td><td>insere material e quantidade</td><td></td></tr><tr><td>2</td><td></td><td>volta a passo 3</td></tr></table>

As restantes especificações podem ser encontradas em anexo.

Comissão de obras

A comissão de obras na aplicação é responsável por gerir as tarefas executadas e criar registar o trabalho das equipas e dos Voluntários.
A maior parte das ações que esta comissão faz precisa do useCase Consultar Obra, pois são sempre ações sobre uma determinada obra.
É de se notar que os use case de Associar Equipa e Associar voluntario a tarefa e Utilizar donativo fazem recurso a use cases da comissão de Fundos respectivamente consultar equipa, consultar Doador e consultar Doações.



A especificações de por exemplo associar voluntario:

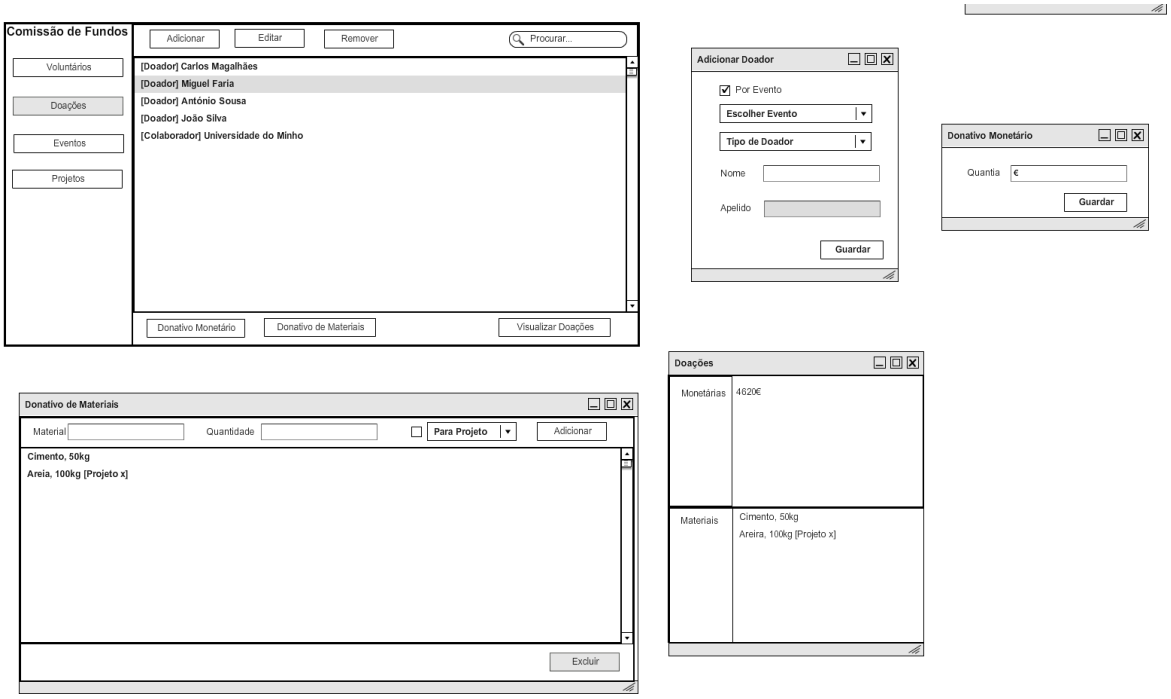
Preconditions	Utilizador está autenticado		
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	<<include>> consultar obra	
	2	Seleciona tipo tarefa	
	3	<<include>> consultar voluntario	
	4		Associa voluntário à tarefa

As restantes especificações podem ser encontradas em anexo.

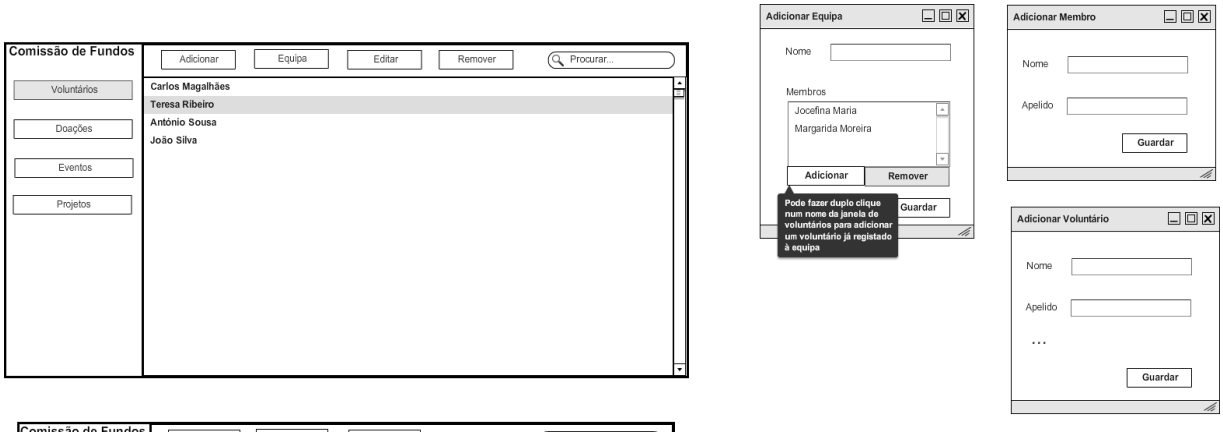
Proposta de Interface

Neste capitulo do relatorio podem-se encontrar alguns exemplos da proposta de interface ao utilizador (mais Imagens em anexo ao relatório).
A aplicação terá uma janela de login, depois abrirá uma janela comum a todas as comissões onde terá uma tab para cada comissão onde estarão as funções especificas de cada comissão e as consultas que qualquer utilizador pode fazer.
Qualquer perfil irá poder navegar pelas tabs, mas as funcionalidades á qual não tiver acesso estaram bloqueadas (não irá poder clicar).

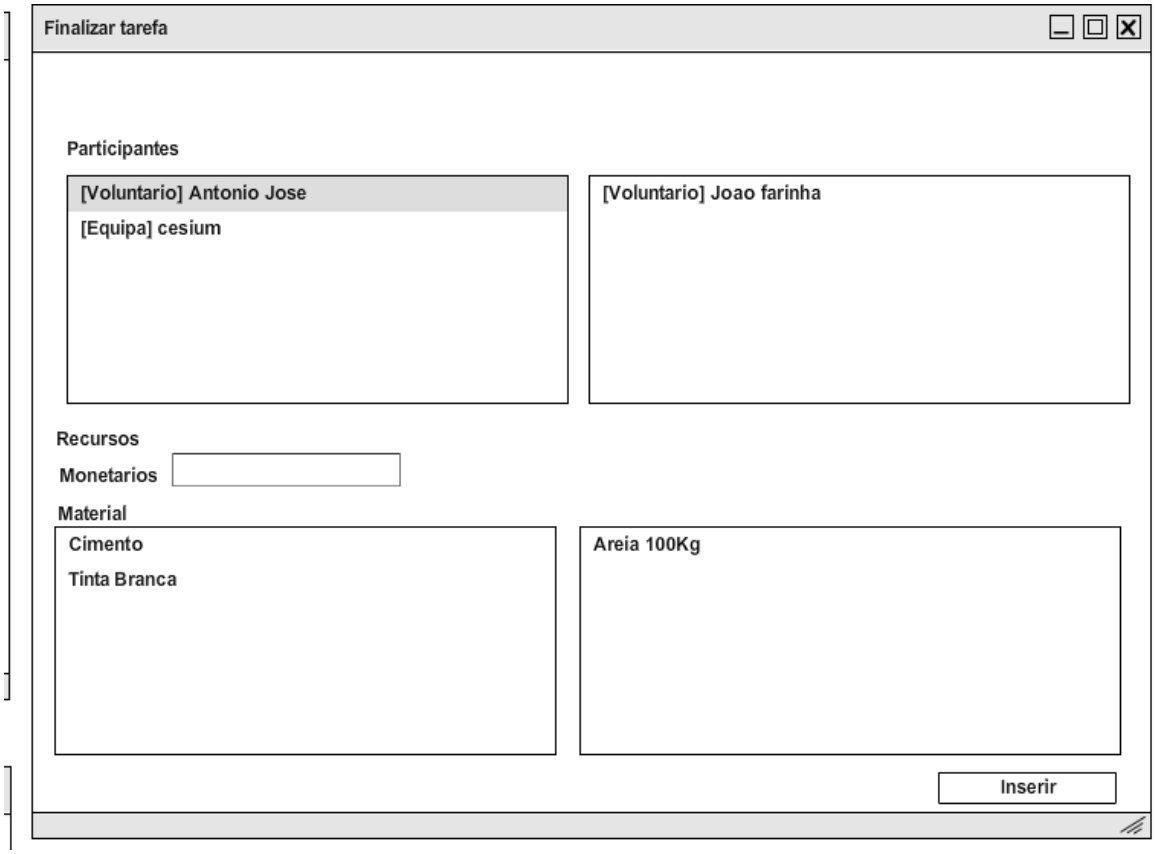
Panel da comissão de fundos (adicionar doações).



Panel da comissão de fundos (adicionar voluntarios/equipa).



Comissão de obras finalizar tarefa

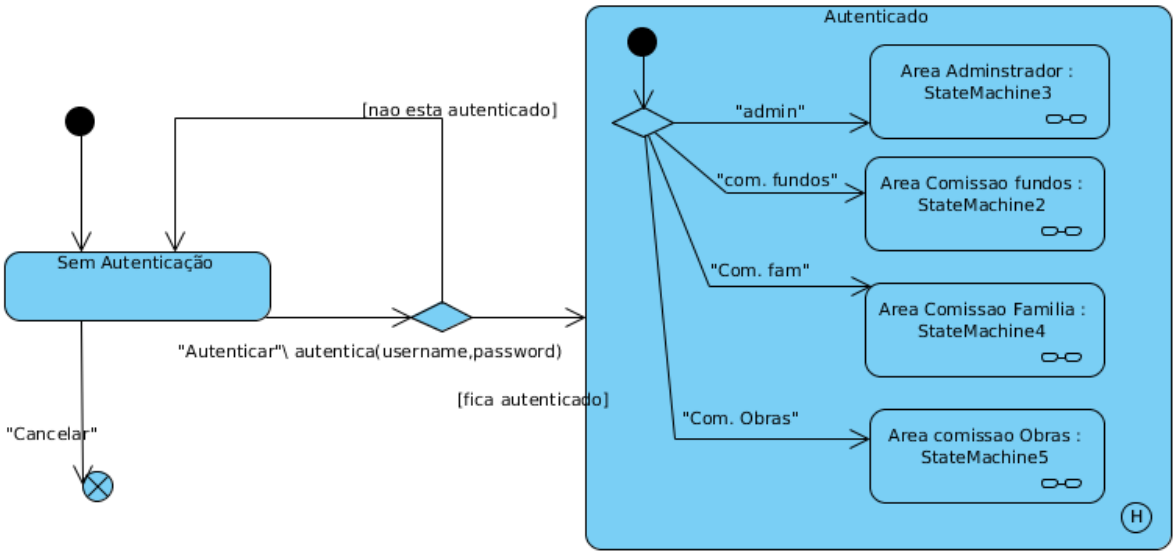


P

painel da Comissão de famílias

Diagrama de estado da interface

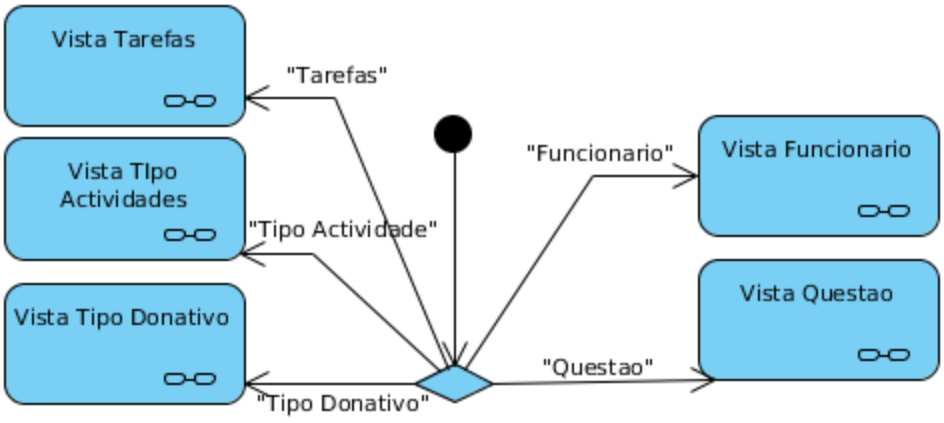
Esta secção tem por fim explicar utilizando diagramas de estado o comportamento da aplicação.
A aplicação começa por não estar autenticada, depois caso se autentique entra na janela principal onde irá ter uma tab para cada comissão e uma para o administrador



Em cada um das tabs tem as vistas que o utilizador tem disponível.

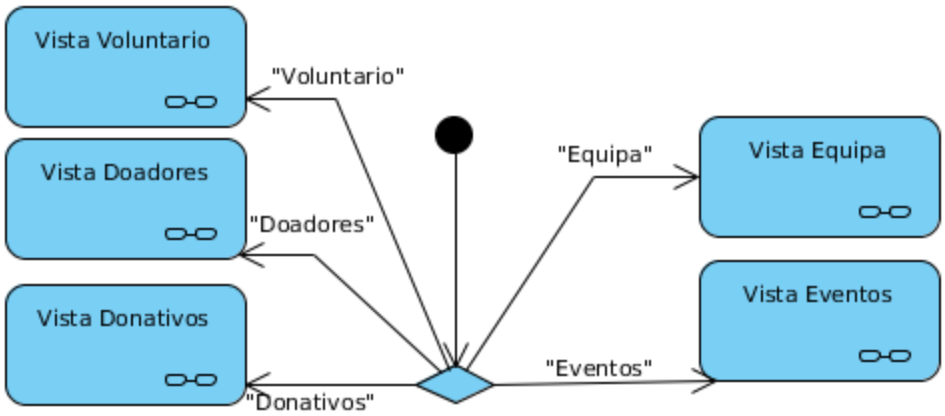
Área Administrador

Dentro da área do administrador temos aceso a 5 vistas Tarefas, Tipo de Actividade, Tipo Donativo, Funcionarios, Questoes.



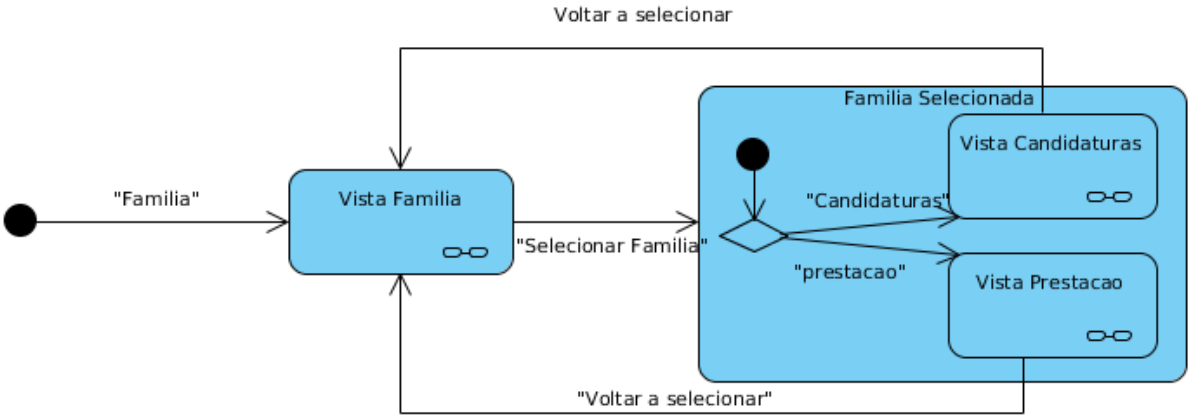
Área Comissão de fundos

Dentro da área da comissão de fundos temos aceso a 5 vistas: Voluntarios, Doadores, Donativos, Equipa e Eventos



Área Comissão de Familia

Dentro da área da comissão de família temos aceso a 3 vistas: família, Candidaturas e Prestações. Com a diferenças das anteriores que que a vista das Candidaturas e da prestação de uma família que tenha sido selecionada na Vista da família



.

Depois de ter selecionado um família o utilizador voltar a selecionar outra.

Área Comissão de Obras

Dentro da área da comissão de obras temos aceso a 4 vistas: obras, Tarefas, Donativos. e voluntariado. Tal como o anterior a vista das Tarefas, dos donativos e do Voluntariado pertencem a uma Obra, logo apenas podem ser acedidos depois de se selecionar uma obra.

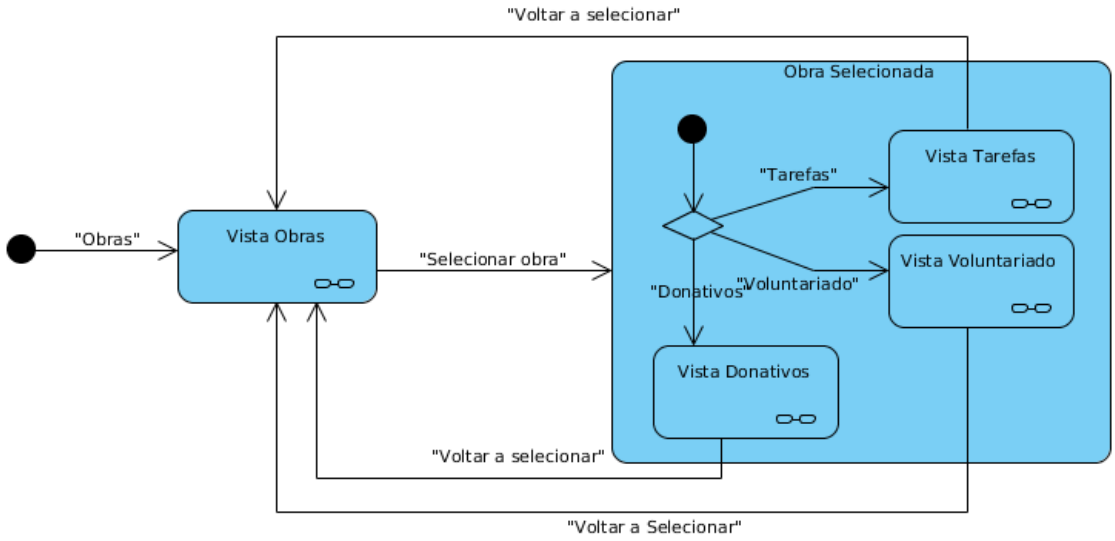


Diagrama de Vista

Cada vista terá um comportamento muito semelhante. O exemplo apresentado mais abaixo é a Tarefa do Administrador.

Quando se entra na vista pode-se procurar por um string, adicionar um novo elemento, ou selecionar um elemento. Quando seleciona-se um novo elemento imediatamente aparecem alguns detalhes sobre este. A partir do elemento selecionado podemos editar, eliminar e ver detalhes .

Implementação da Interface

Esta secção pretende explicar como foi implementada a interface, que tem a particularidade de estar generalizado ao nível de cada vista, e toda a interface estar centralizada. Irá-se também comparar a proposta de interface com a implementação da interface .

A tecnologia utilizada para a implementação da Interface foi o Java Swing, pois é simples de se usar e existem boas ferramentas para desenhar a interface, neste caso foi utilizado o Netbeans.

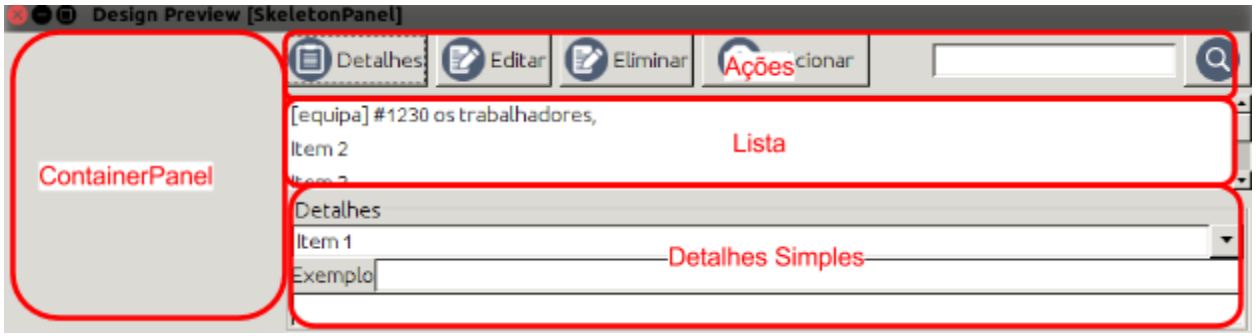
Comportamento genérico de cada Vista

Como foi explicado na capitulo anterior as cada uma das vistas, tem um comportamento muito parecido, mostrar todos os elementos numa lista e depois tem a capacidade de se editar, eliminar e criar novos elemento. E como no total da aplicação existem 17 vista, foi criar uma *UIDimension* que generaliza as vistas, passando-lhe o comportamento espessifico, tal como as janelas que irão abrir quando se clicar em cada umas das ações.

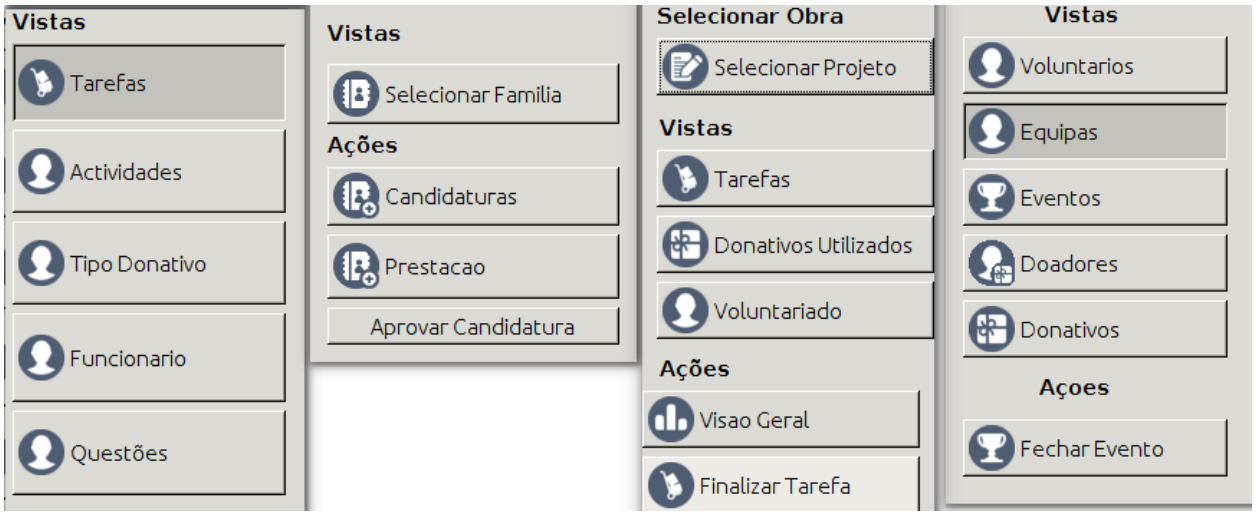
Nestas janelas é que estará o o código a afetivamente Editar ou adicionar o elemento, ou seja o UIDimesion é só um class de gerir a interface, não acede diretamente ao dados.

Esqueleto da interface

Para que a aplicação fosse equivalente nas diferentes áreas da aplicação foi criado o SkeletonPanel. O ContainerPanel é o painel onde se coloca a toolBar de cada área.
A lista do SkeletonPanel é carregada de forma genérica partir de um UIDimension que é previamente carregado no skeleton, depois deste estar carregado as ações são delegadas também para o UIDimesions (Editar, Adicionar, Eliminar e Detalhes)



ToolBar que são colocadas no ContainerPanel do Skeleton



As toolbars são JPanels pois desta forma consegue-se ter mais flexibilidade naquilo que se constrói.

Centralização da aplicação.

De forma a que qualquer parte da aplicação tenha acesso a outra parte da aplicação, existe a class AppState, nesta class são guardados os 4 Skeletons e os 17 UIDimensions. E facilmente pode existir interação.

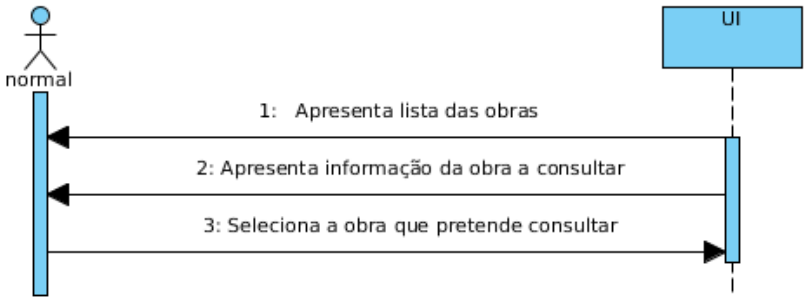
Diagramas de Sequência

Neste capítulo encontram-se os diagramas de sequência criados no desenvolvimento da aplicação. O processo de desenvolvimento foi o seguinte: Diagramas de Sequência de Sistema, Diagramas de Sequência com Subsistema e finalmente diagramas de implementação.

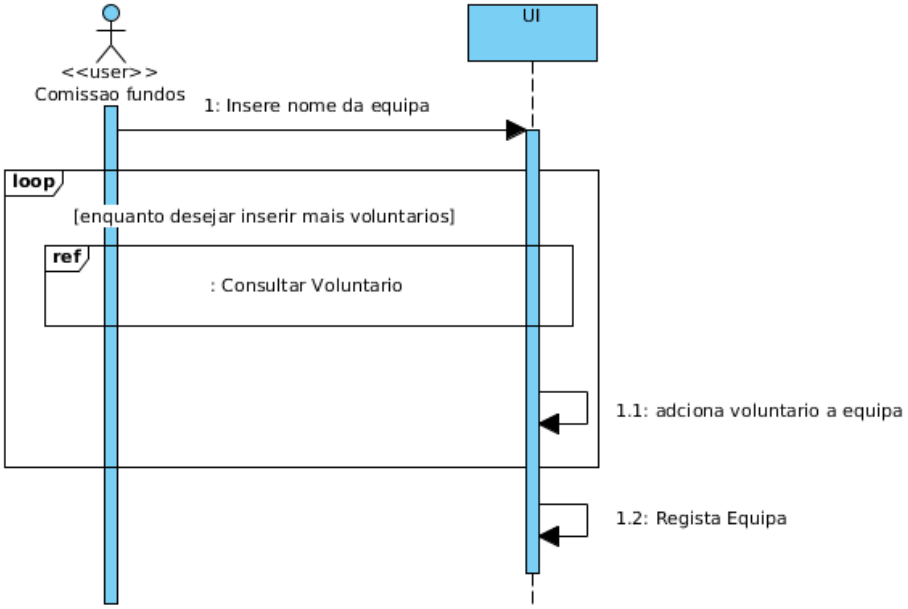
Diagramas de Sequência de Sistema (DSS)

Estes diagramas foram criados utilizando a conversão do VisualParadigm e posteriormente refinados manualmente adicionando os breaks, alts, loops e self-messages.

DSS Consultar de Obra

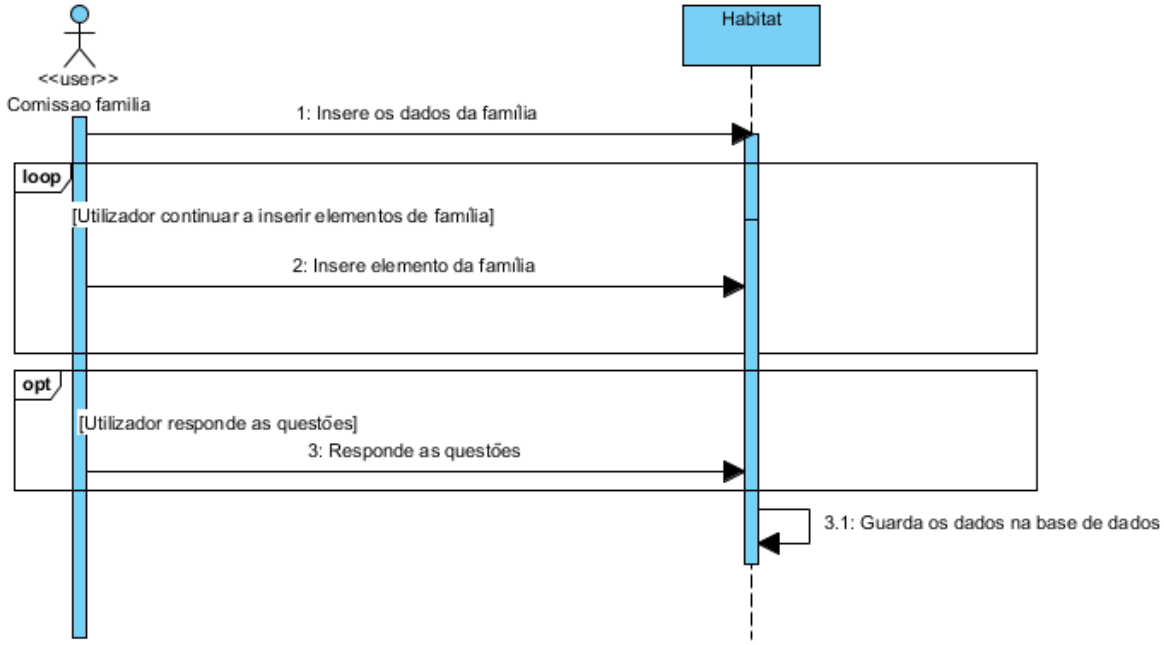


DSS criar equipa



Neste caso vai-se inserindo os voluntários à equipa e no fim é que se regista a equipa.

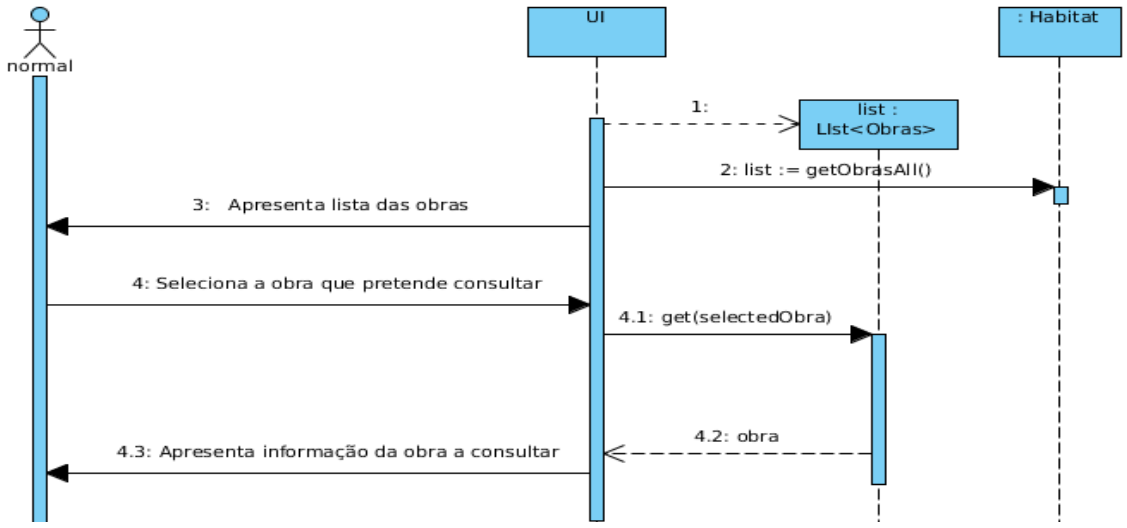
DSS Guardar Candidatura



Diagramas de Sequência com subsistemas

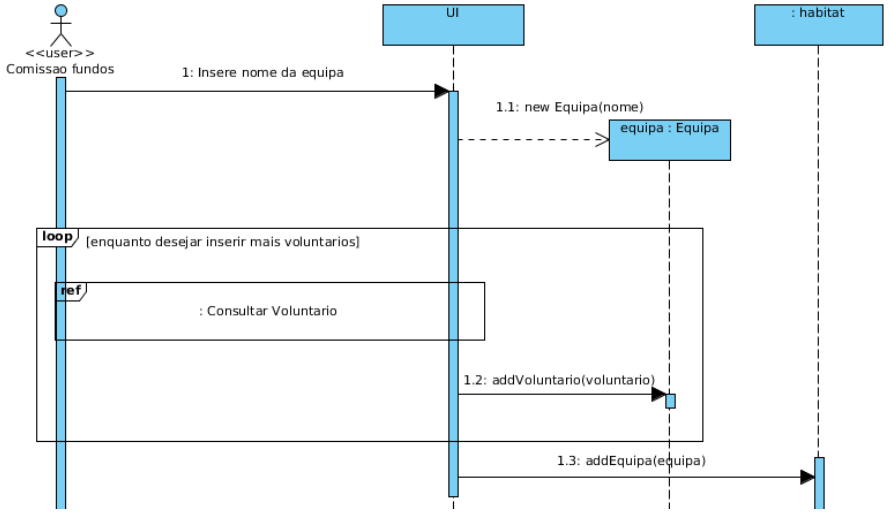
O passo seguinte foi continuar a refinir e criar as class's que são necessárias em cada use case. utilizando o facade da Habitat

DSS consultar de Obra



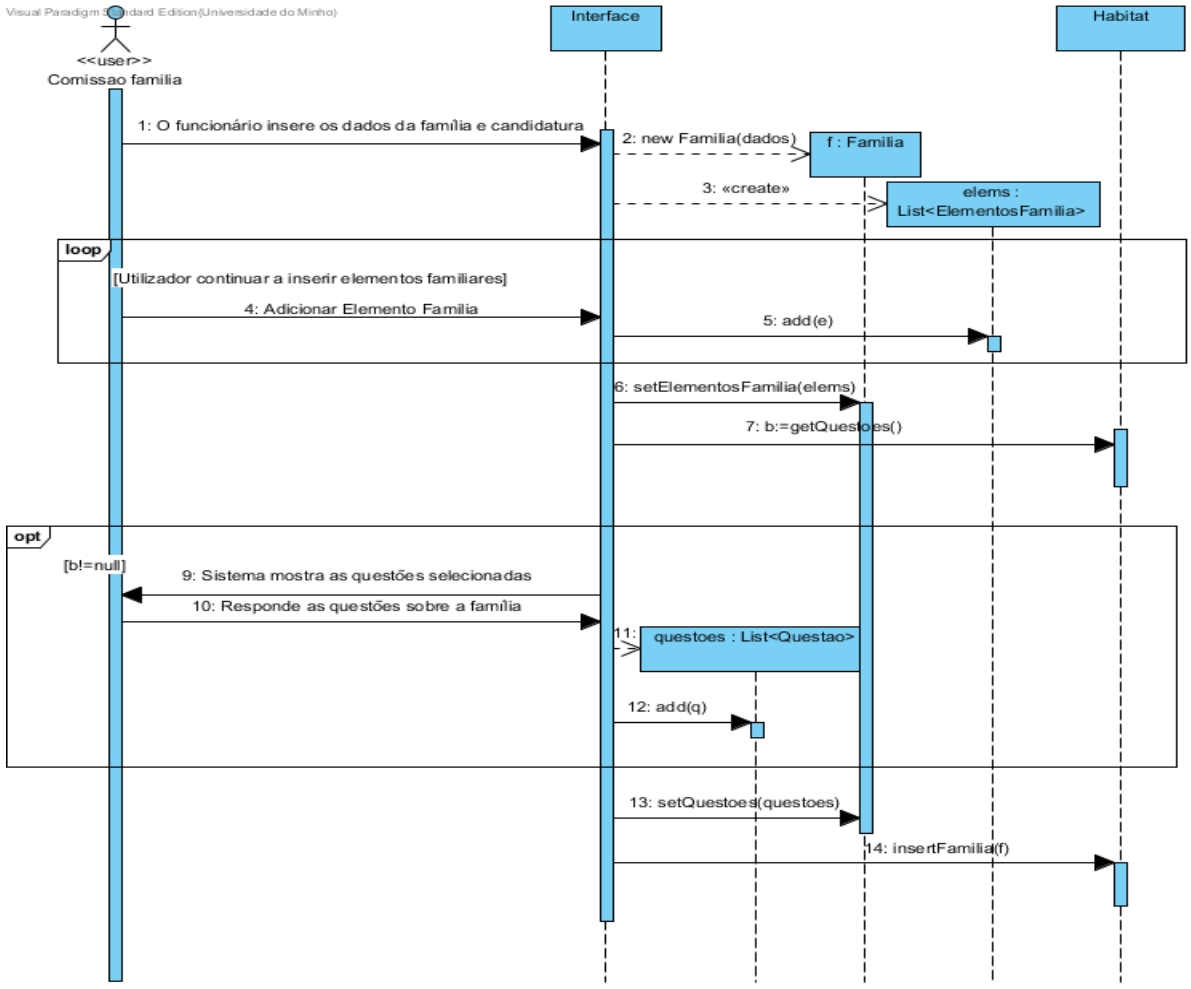
Quando se seleciona uma obra essa obra é retornada. Outros Diagramas de sequência que utilizam o "Consultar Obra" iram usar o o objeto obra.

Criar Equipa



Após a criação da equipa é que se adiciona a equipa à habitat

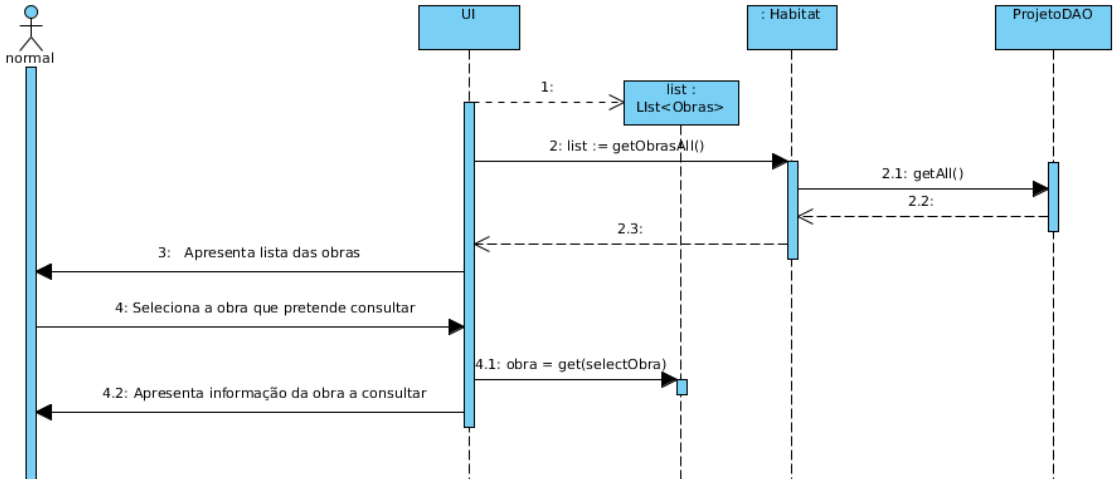
Guardar candidatura



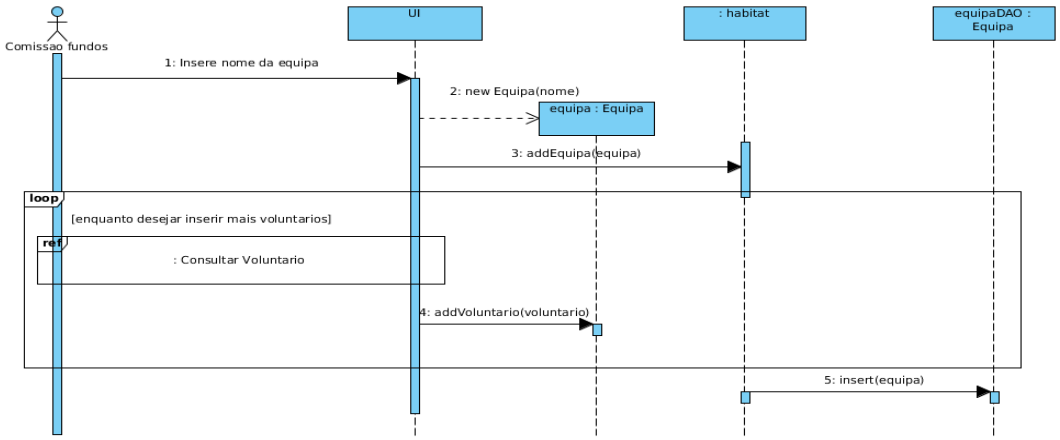
Diagramas de Sequência de Implementação

O ultimo passo do refinamento dos Diagramas de sequência foi adicionar a utilização dos DAOS

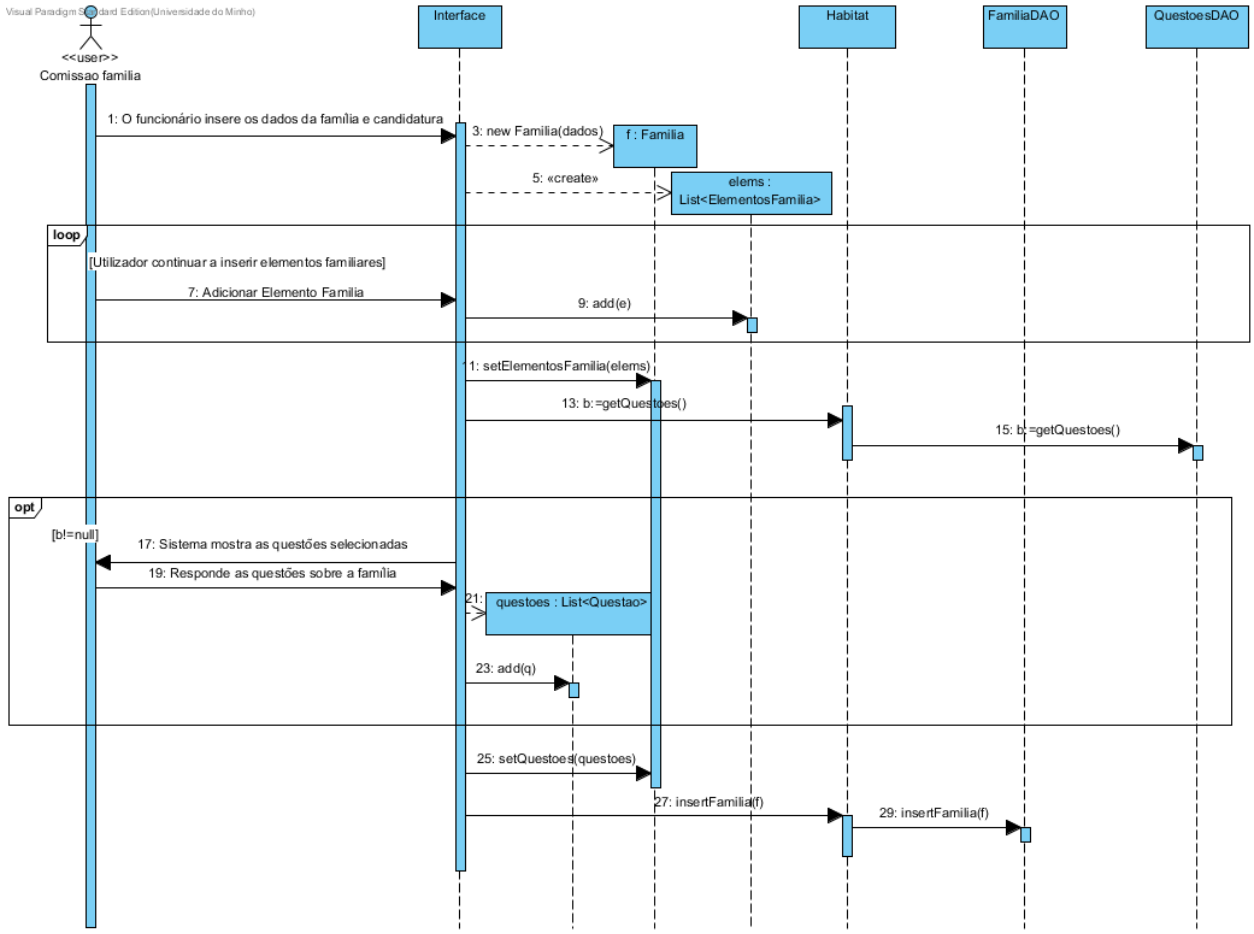
DSS Consultar de Obra



Criar Equipa



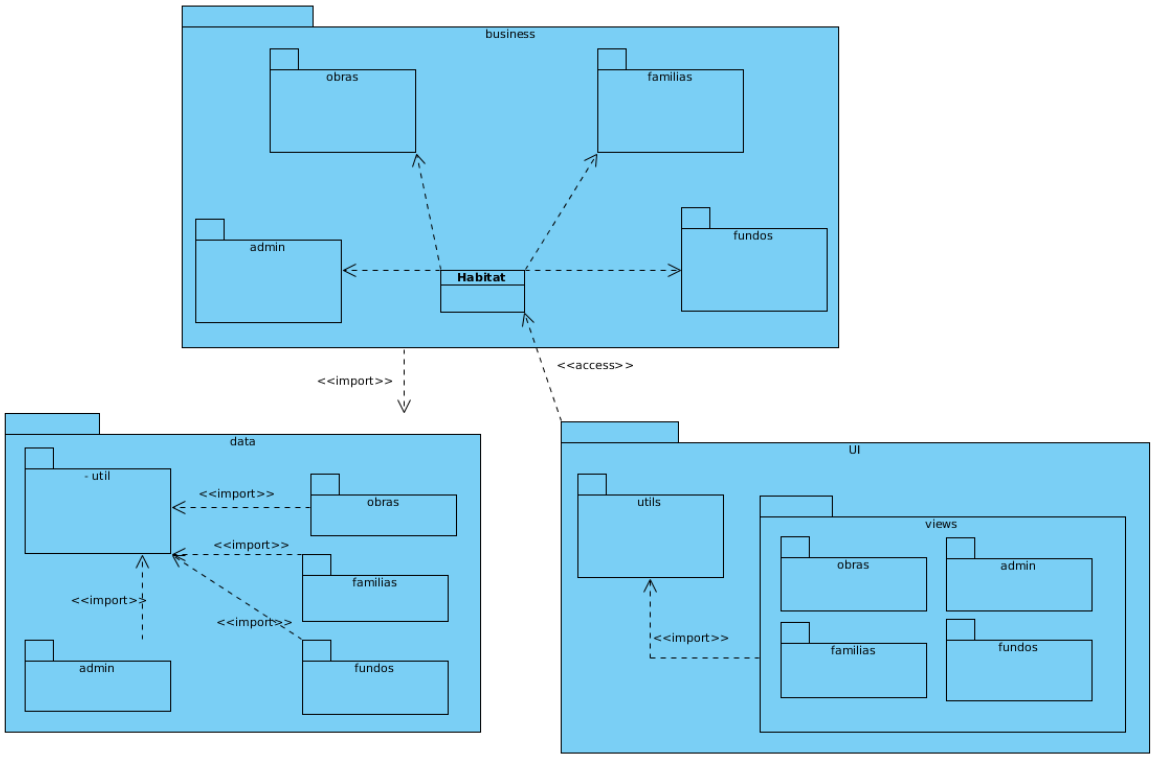
Guardar candidatura



Neste diagrama a interface vai guardando os dados inseridos num objeto da classe família e só no fim do utilizador responder as questões, que são carregadas da base de dados, é que e é feito a inserção na base de dados

Diagrama de Package do sistema

A aplicação foi separada em 3 grandes packages business, data e UI. em business está toda a camada de negocio, o facade deste package é a class Habitat. O package data é o package de acesso aos dados, aqui estão todos as class de DAO. O package UI tem toda a Interface do utilizador. Os dois últimos packages tem um package chamado Utils onde estão as generalizações que foram contruídas para a aplicação.



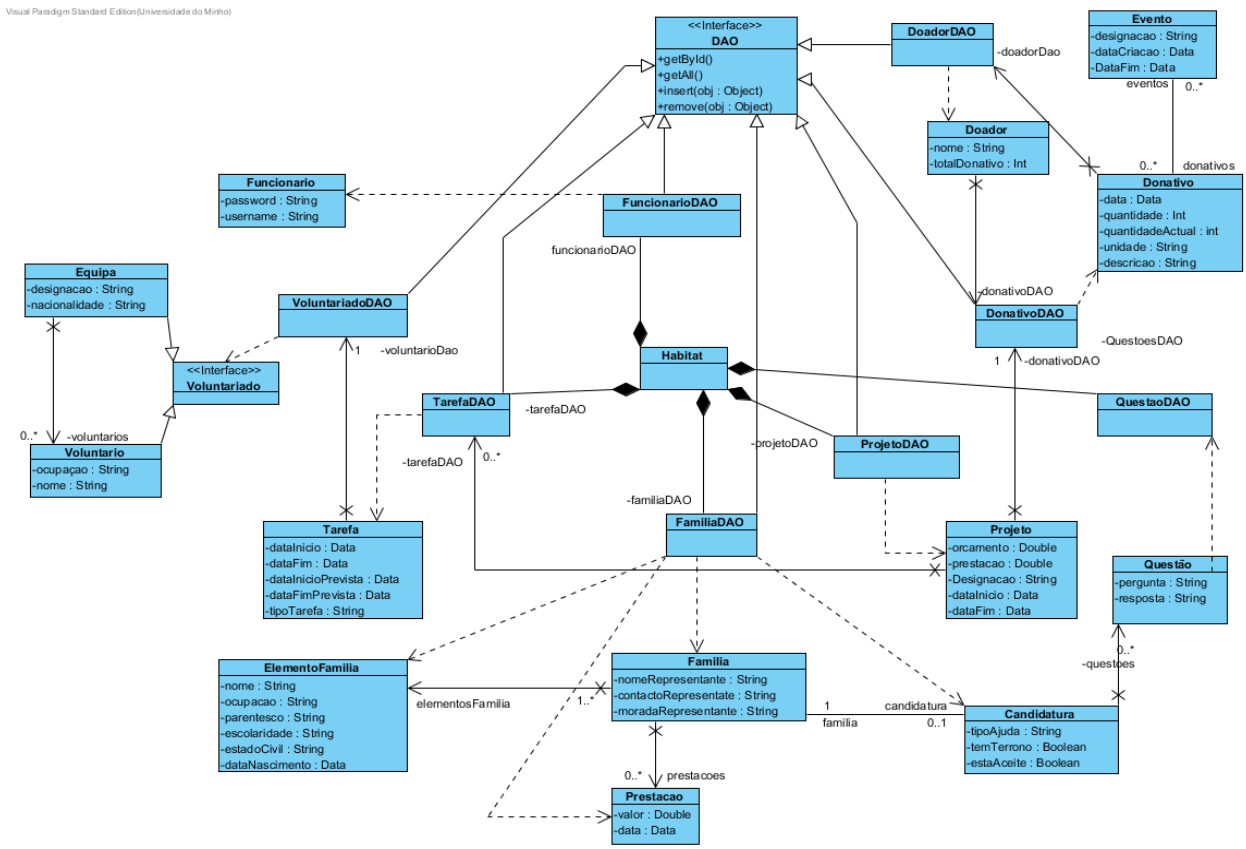
Diagramas de Classe

Neste Capítulo estão presentes os diagramas de classe que suportaram o desenvolvimento da aplicação.

Foram 3 diagramas ao longo do desenvolvimento Conceptual, Especificação e implementação.

Especificação

A primeira aproximação ao desenvolvimento do projeto já foi realizada usando DAOs, devido a impossibilidade de uma estrutura do tipo Map ser guardada numa base de dados.



O diagrama de especificação contém a ideia inicial das classes necessárias para o funcionamento do programa, o diagrama contém DAOs que são as classes do package data que fazem o acesso a base de dados para guardar, atualizar ou consultar informação, a classe Habitat que é o facade para implementar esse DAOs, enquanto as outras classes servem para manter e fazer operações sobre os dados em memória.

Implentação

DAO e GenericDAO

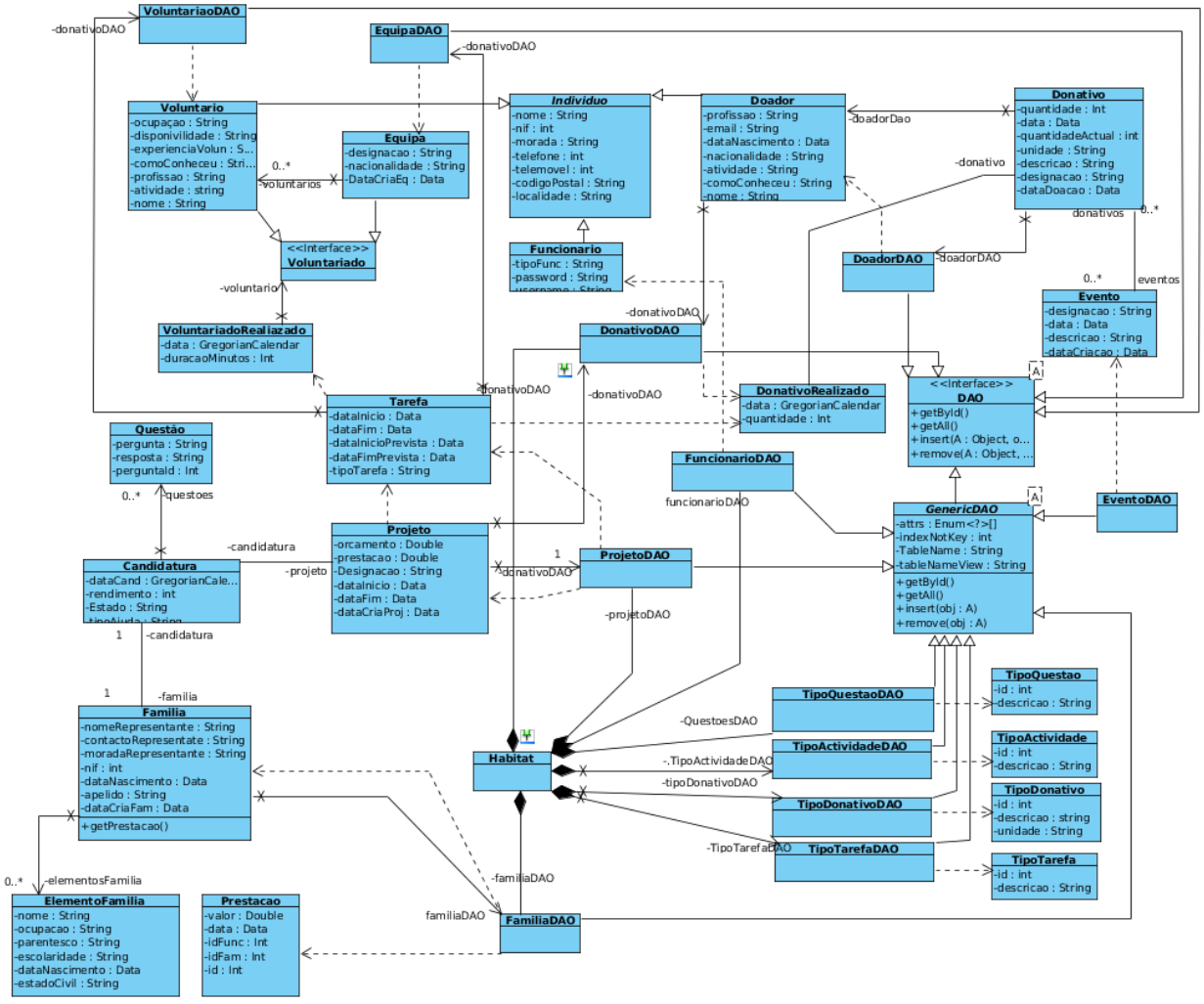
De forma a facilitar o a cesso à base de dados, foram criadas class's genéricas de acesso à base de dados a class DAO fornece o primeiro nivel de abstração e permite executar querys passando texto, deixando de lado problemas de conexão.
O segundo nivel de abstração cria automaticamente o Insert, Update, delete, GetAll, GetById e um GetBy generico que permite pesquisar por um objeto exemplo.
Todas as class's de DAO esta no packege Data.

Class que utilizam os DAOS

Os DAOS estão no packege data e são utilizados pelas class do business. A Habitat tem acesso aos DAOS: TipoTarefaDAO, TipoActividadeDAO, TipoDonativoDAO, TipoTarefaDAO, FamiliaDAO, FuncionarioDAO, ProjetoDAO, DonativoDAO e DoadorDAO. Ou seja a class Habitat permite gerir toda a informação dos DAOs.

Para que toda a informação não seja carregada ao criar os Objetos, como é o caso do projeto e das Tarefas. A class Projeto tem um ProjetoDAO para aceder ás tarefas apenas quando se faz o getTarefas. Esta situação repetesse nas relaçoes entre Tarefa Voluntariado, Familia Prestaçao, entre outros.

Comparação da implementação com os diagramas
Nesta secção apresentamos uma comparação entre o código desenvolvido e o comportamento modelado.



Generalização da Interface

Tal como introduzido no capitulo da Implementação da interface, o elevado número de vistas diferentes conbinado com um comportamento muito similar entre as vistas levou-nos a uma pequena generalização.

Existem 4 nomes Fundamentais: AppState, UIDimension,UIDimension.JDetails e SkeletonPane.

AppState - Corresponde ao atual estado da aplicação, e deve poder ser acedido de qualquer parte da interface. Esta Class tem o facade Habitat para que este possa ser usado em qualquer sitio.

UIDimension - presenta um vista, e tem ações como carregar uma lista consultar selecionado, pedir para mostrar janela de detalhes, janela de adicionar ou remover.

UIDimension.JDetails - interface que permite as janelas ter comportamento generico nesseçario para o UIDimension saber trabalhar com elas.

SkeletonPane - Janela que contem cada um das área de trabalho.

Diagrama de classes da interface

Por simplificação não esta representado tudo aqui, mas pode-se verificar que o AppState vai ter 4 Skeleton, um para cada área. relacionados com cada skeleton tem uma Toolbar.

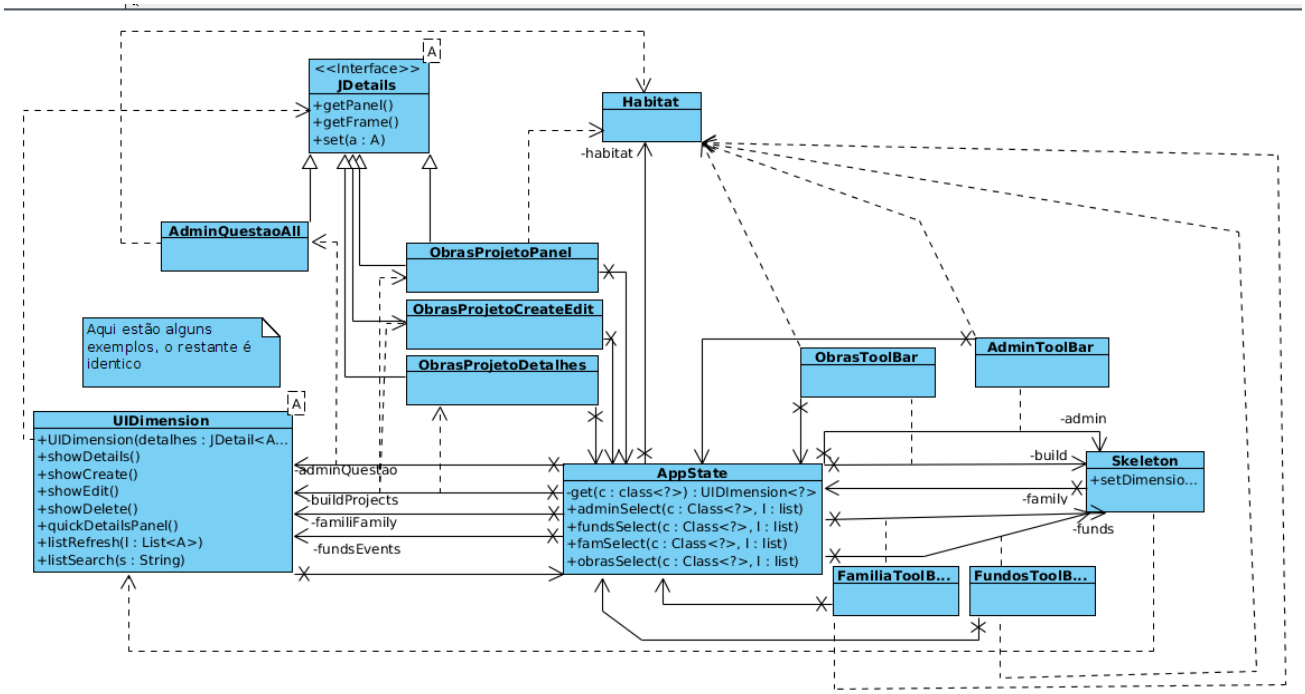
O AppState tem tantas UIDimension quantas vistas no total 17, no exemplo apenas tem 4.

e para cada relação com o UIDimension o AppState tem que dizer quais são as janelas que vão abrir (ObrasProjetoPanel, ObrasProjetoCreateEdit, ObrasProjetoDetalhes, entre muitos outros).

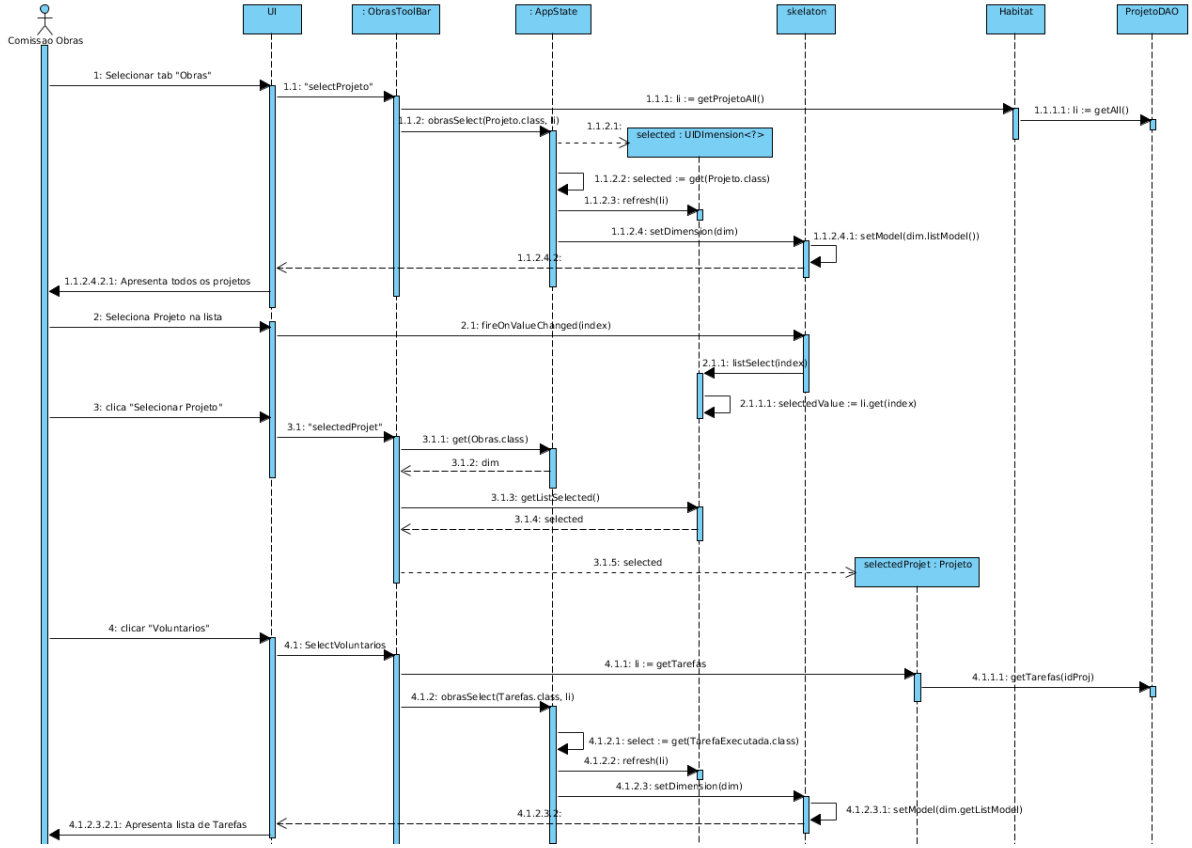
Mas para que UIDimesion saiba manipular estas janelas, elas tem que implementar o JDetials.

Desta forma Adicionar novas funcionalidades na interface, e

Diagrama de classes da interface



Excerto de Diagrama de sequência da Interface (Associar voluntario a tarefa)



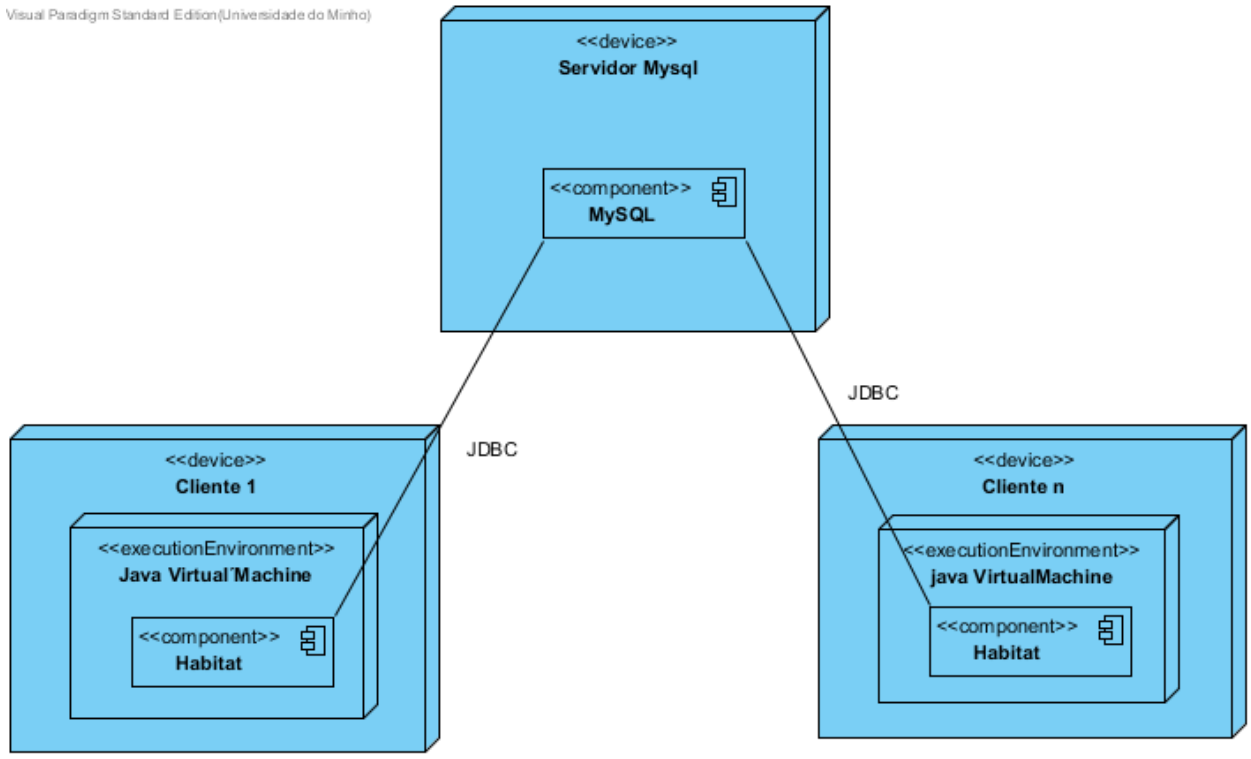
Depois de se ter feito o diagrama de sequência da interface foi feita uma pequena correção à estrutura. O lacuna era que mesmo depois de se fazer o load da UIDimension no skeleton, esta quando recebia algum evento de click tinha que ir ao appState perguntar quem é que ele próprio tinha selecionado. Agora passou a ter um setDimension, e o skeleton guarda a UIDimension que esta carregada. Este erro seria difícil de se detetar sem uma visão global do problema,

Software utilizado

O software utilizado foi VisualParadigm, Netbeans, java7, git, mysql, MockFlow.
No VisualParadigm criamos todos os diagramas aqui descritos e os que estão em anexo, nos use cases tiramos proveito da utilização de sub-diagramas uma forma que o visual paradigm oferece para ligar diferentes diagramas entre si.
Foi utilizado o netbeans para a implementação da aplicação, foi escolhido após comparar a forma de criar java Swing do Eclipse com a do Netbeans. Apesar que fomos obrigados pelo Netbeans a utilizar o Java7, pois o java8 tinha incompatibilidades com as Swing.
Foi também utilizado o Git através do bitbucket.com para a partilha do projeto.
Para utilizador a base de dados utilizou-se mysql e a conexão a base dados através do java.
E o software para o desenho da proposta de interface foi o Mockflow uma extensão para o Chorme.

Diagrama de Instalação

A aplicação habitat está prevista para correr em 4 computadores, e a base de dados irá estar numa o outra máquina.



Como se pode ver no diagrama, a aplicação vai correr nos computadores utilizando o java Virtual Machine. Existirá um outro computador que irá funcionar como servidor e estará a correr a base de dados MySQL. Para que a aplicação tenha acesso ao servidor é necessário que tenha a drive instalado (o JBDC).

Conclusões

Analise de requisitos.
A analise de requisitos é das partes mais importantes do desenvolvimento, este e um dos maiores pilares mais importantes para a criação de software especialmente quando as datas são apertadas e não a espaço para voltar a fazer tudo de novo. No Caso desta aplicação não existiu grande variação entre o que foi definido no inicio e o que foi conseguido.

Proposta de interface vs Implementação
No final a Interface acabou por ficar um pouco diferente da proposta original, isto é devido a algumas alterações que foram sendo feitas para facilitar a sua implementação, estas alterações foram feitas por várias razões: porque a interface original era apenas um esboço, então, simplificações que foram feitas pois eram mais acessíveis de programar e mudanças nos requisitos ou base de dados.

Diagramas de estado
Os diagramas de estado da interface foram cruciais para ajudar a perceber os padrões do problema, e tentar assim generalizar a interface.
Mas como demasiada generalização pode ser perigosa, tem que se achar um equilíbrio entre generalização e a flexibilidade à mudança. para isso deixou-se que cada comissão pudesse ter uma toolbar quem manipula directamente o AppState e o mesmo se passa nas janelas das comissões.

Refinamento de Use Cases
Tal como diz o processo de desenvolvimento começamos por definir os Usecases e a partir daí foi sempre sucessivos refinamentos pela seguinte ordem: especificações, Diagramas de sistema, diagramas com sub-sistemas e finalmente diagramas de implementação. Com os dois últimos diagramas construímos o diagrama de classes

Problemas com os DAOs
Apesar de termos conseguido generalizar um pouco os DAOs com o GenericDAO. Estes são sempre Complicados e propícios a erros. Um dos problemas que tivemos que resolver era no caso de editar uma família, e depois editar os elementos dessa família, ou seja um update a uma família pode originar inserts ou removes, dependendo do caso. O que é complicado de gerir.

Apreciação final
De forma resumida sentimo-nos satisfeitos com o trabalho ainda existe muita coisa que pode ser melhorada, mas está ele encontra-se funcional. Seguir um plano de desenvolvimento ajudou pois foi mais fácil de prever as dificuldades e ultrapassar-las.

Trabalhos futuros

Para melhorar a qualidade do trabalho que apresentamos podemos adicionar ou alterar certas partes do mesmo, como por exemplo:

- melhorar o aspecto visual da listagem da informação de forma que fique mais simples para o utilizador compreender;
- melhorar o tratamento de exceções;
- adicionar a comissão de famílias opções para guardar o acompanhamento as famílias;
- adicionar mensagens de aviso ao utilizador quando uma ação lhe é negada sem explicação;
- Melhorar os dados que os detalhes fornecem, como por exemplo os donativos que o doador ou em que projetos participaram os voluntários.