

Universidade do Minho
LEI 3ºAno 1ºSemestre
Desenvolvimento de Sistemas de Software
GestHabitat
2ª Etapa

Daniel Caldas a67691
José Cortez a67716
Marcelo Gonçalves a67736
Ricardo Silva a67728

10 de Janeiro de 2015



Conteúdo

1 Contexto do problema e análise de requisitos	4
1.1 Habitat	4
1.1.1 Estrutura organizacional da Habitat	4
1.1.2 Funcionamento - Processo de candidatura de uma família	5
1.1.3 Candidatura aceite	5
1.2 A Aplicação	6
2 Modificações	7
2.1 Modelo de Domínio	7
2.2 Use cases	8
2.2.1 Modificados	8
2.2.2 Novos use cases	12
3 Diagramas de Seguência de Sistemas	14
3.1 Autenticação	14
3.2 Doações	14
3.3 Famílias	22
3.4 Projetos	26
3.5 Voluntários	40
4 Diagramas de Classes	46
4.1 Um passo em frente na modelização	46
4.2 Diagrama de Classes V1	47
4.3 Diagrama de Classes V2	48
4.4 Diagrama de Classes V3	49
5 Diagrama de packages	50
5.1 Separação lógica	50
5.1.1 Diagrama	51
5.2 Package - Doacoes	52
5.3 Package - Familias	53
5.4 Package - Projetos	54
5.5 Package - Recursos Humanos	55
5.6 Package - Data Access	56
6 Diagramas de Seq. de Subsistemas	57
6.1 Autenticação	57
6.2 Doações	58
6.3 Famílias	70
6.4 Projetos	75
6.5 Recursos Humanos	90
7 Diagramas Seq. Implementação	99
7.1 Autenticação	99
7.2 Consultar candidatura	100
7.3 Procura Representante	101
7.4 Insere Projeto	102
7.5 Total Doador por um Doador	103

8 Máquinas de estado	105
9 Modelo controlo de diálogo	109
10 Aplicação GestHabitat	111
10.1 Detalhes de implementação	111
10.2 GUI final	112
10.3 Trabalho efetuado	114
10.3.1 Ferramentas da interface	114
10.3.2 Entrega	119
11 Conclusão	120
11.1 Comentário	120
11.2 Aprendizagem	120
12 Anexos	121
12.1 Documentação javadoc	121

1 Contexto do problema e análise de requisitos

1.1 Habitat

ONG Americana internacional criada em 1976. Construiu cerca de 800 000 casas. Ajudou cerca de 4 000 000 de pessoas. ONG principal na área da construção, é a primeira organização a ser chamada para reconstrução de habitações a quando de catástrofes naturais. Atividade em Portugal só teve início em 1996. Desde então já (re)construíram cerca de 63 casas. Em Portugal a Habitat trabalha exclusivamente com famílias. Ainda não trabalha com outras instituições mas está de momento a estudar a possibilidade.

1.1.1 Estrutura organizacional da Habitat

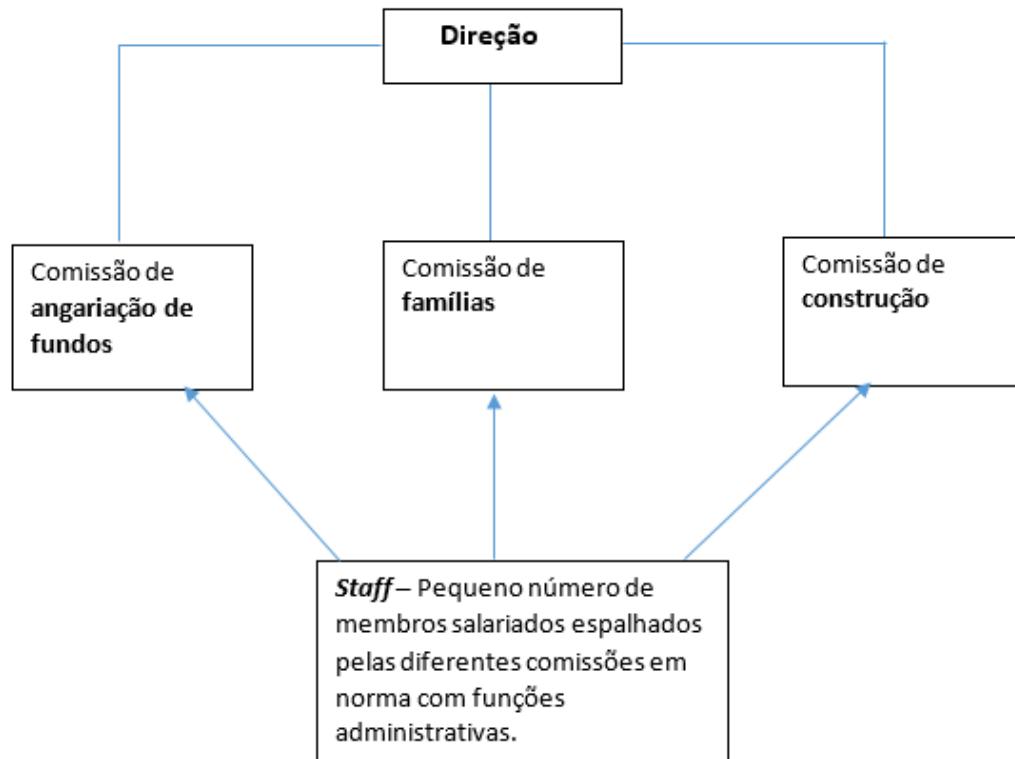


Figura 1: Acima da linha a tracejado os elementos são voluntários.

Todos os outros não mencionados são voluntários.

1.1 Habit@t CONTEXTO DO PROBLEMA E ANÁLISE DE REQUISITOS

1.1.2 Funcionamento - Processo de candidatura de uma família

É preenchida uma **ficha** pela família. As casas são pagas pelas famílias e os membros do agregado têm de participar na construção da casa. É feita uma análise dos dados e uma entrevista à família. Essa análise/triagem dos dados é feita pela **comissão de famílias**, a mesma comissão constrói um **relatório** da análise feita para ser passado à **direção**. A direção analisa o relatório bem como toda a parte da construção da habitação e a possibilidade financeira da obra. A direção toma a decisão final.

1.1.3 Candidatura aceite

É feito um orçamento para apresentar à família esta decide se pretende ou não continuar. Caso resposta positiva a construção avança, mas apenas quando o projeto em curso (se existir) terminar (um projeto de cada vez!). Antes de se começar o projeto é feita a **angariação de fundos** (dinheiro, voluntários, material...).

Durante a execução da obra é feito um acompanhamento (**pelo menos**) semanal de **custos, horas de voluntariado, fases de construção**. Muito **importante** para verificar se projeto está a decorrer como planeado. O **projeto** é planificado (em tarefas).

Na fase final é feito um apuramento dos **custos reais** e apresentada a conta à família. É calculada em função dos rendimentos da família uma mensalidade para abater o custo da casa. Essa prestação **pode variar** ao longo do período de pagamento, isto é, subir ou descer consoante as possibilidades da família (prestação moldável).

1.2 A Aplicação

1.2 A Aplicação

"Queremos uma aplicação que nos permita crescer, que permita optimizar a utilização dos nossos recursos. Neste momento toda a informação da instituição se encontra muito dispersa. Queremos ter o poder do cruzamento de dados internos. Queremos obter mais fundos. O objetivo é economizar tempo e poder servir cada vez mais e mais famílias." - João Cruz

A plataforma deverá permitir agregar toda a informação da Habitat.
As quatro principais funcionalidades espectáveis da plataforma:

- **Registo e gestão de voluntários;**
- **Registo e gestão de doadores e donativos;**
- **Registo e gestão de famílias;**
- **Registo e gestão de projetos;**

Algumas especificidades:

- Registar processo de candidatura e seleção de famílias;
- Registar todo o processo de construção (tarefas);
- Registar doações (associadas ou não a uma certa obra);
- Possibilidade de associar voluntários a um dado projeto;
- Consultar empresas que doaram mais recentemente;

As entidades que interagem com a plataforma são apenas os funcionários (STAFF) da habitat e administração (perfis de utilizador)

2 Modificações

2.1 Modelo de Domínio

Nesta etapa final começamos por fazer o refinamento do nosso modelo de domínio, criando uma nova versão do mesmo, sendo este uma aproximação mais realista ao diagrama de classes que é implementado.

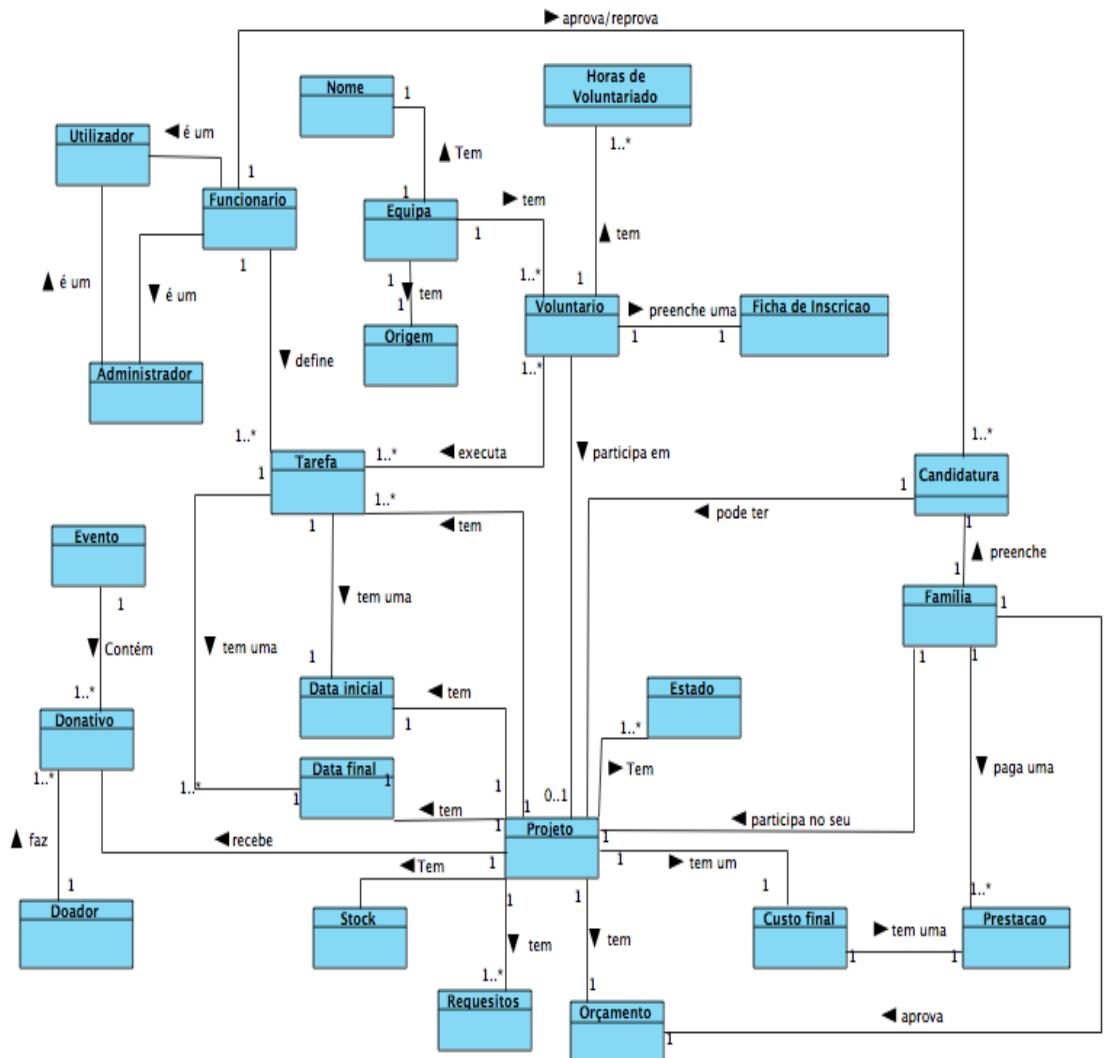


Figura 2: Na imagem podemos observar potencias classes da aplicação e as interações entre elas.

2.2 Use cases

Existiu também a necessidade de criar novos *use cases* e de modificar alguns que implementamos na primeira fase do projeto. De se guida vamos fazer uma exposição dos use cases modificados.

2.2.1 Modificados

Use cases que modificamos:

- Adiconar Tarefa foi alterado porque deixamos de fazer a associação de material gasto.
- Editar projecto, regista prestação, regista reparação, registar custo final, registrar orçamento, foram alterados porque existiam duas mensagens consecutivas do actor para o sistema.
- Registar donativo, foi eliminada a distinção do tipo de donativo, não faz diferença se é material, serviço ou monetário.
- Remover tarefa, foi removida a exceção relacionada com a confirmação de eliminação por parte do actor.
- Regista prestação, registar uma prestação correspondente ao valor que uma família irá pagar;
- Regista reparação, quando é feita um reparação numa habitação construída pela Habitat;
- Regista custo final, a quando término de uma obra é registado o seu custo final;
- Regista orçamento, registar o valor do orçamento de uma empreitada;

Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Tarefa fica registada	
Flow of Events	Actor Input	System Response
	1 indica nome ou id projeto	verifica se projeto esta ativo
	2	
	3 adiciona nova tarefa	regista tarefa
	4	guarda projeto
	5	
Excepção 1 [projeto não existe] (passo 2)	Actor Input	System Response
	1	indica que projeto não existe

Figura 3: Use case **Registrar Tarefa**

Super Use Case	Editar Projecto	
Author		
Date	10/Nov/2014 22:27:08	
Brief Description		
Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Atualiza Projecto	
Flow of Events	Actor Input	System Response
	1 indica nome ou id projeto	verifica se existe projeto
	2	apresenta lista de projectos
	3	
	4 seleciona projeto	apresenta detalhes projeto
	5	
	6 modifica projeto	regista alterações
	7	guarda projeto
	8	
Excepção 1 [Projecto não existe] (passo 2)	Actor Input	System Response
	1	projecto não existe

Figura 4: Use case **Editar Projeto**

Preconditions	Utilizador Autenticado como Staff comissão de angariação de fundos	
Post-conditions	Donativo Registrado	
Flow of Events	Actor Input	System Response
	1 Informa dados do doador	
	2	Verifica doador
	3	Questiona por dados do donativo.
	4 Fornece dados do donativo	
	5	Verifica tipo do donativo.
	6	Regista donativo.
Exceção 1 (Passo 2) (Doador não existe)	Actor Input	System Response
	1	Informa que o doador não existe

Figura 5: Use case **Regista Donativo**

Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Remove tarefa	
Flow of Events	Actor Input	System Response
	1 indica nome ou id projecto	verifica existencia projeto
	2	apresenta lista de tarefas
	3	
	4 indica tarefa	
	5	verifica existencia tarefa
	6	remove tarefa
Excepção 1 [Projecto não existe] (passo 2)	Actor Input	System Response
	1	indica que projecto não existe
Excepção 2 [tarefa não existe] (passo 6)	Actor Input	System Response
	1	indica que tarefa não existe

Figura 6: Use case Remove Tarefa

Super Use Case	Registrar Custo Final	
Author		
Date	10/Nov/2014 22:43:50	
Brief Description		
Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto Custo final fica registado	Actor Input
Post-conditions		
Flow of Events		System Response
1 indica nome ou id projecto		
2	verifica existencia projeto	
3	apresenta lista projectos	
4 seleciona projecto		
5	apresenta detalhes projecto	
6 indica custo final		
Excepção [projecto não existe] (passo 2)	Actor Input	System Response
	1	indica projecto não existe

Figura 7: Use case Regista custo final

Super Use Case	Registrar Orçamento	
Author		
Date	10/Nov/2014 22:38:41	
Brief Description		
Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto Orçamento fica registado	Actor Input
Post-conditions		
Flow of Events		System Response
1 indica nome ou id projecto		
2	verifica existencia projeto	
3	apresenta lista projectos	
4 seleciona projecto		
5	indica detalhes projecto	
6 indica orçamento		
Excepção 1 [projecto não existe] (passo 2)	Actor Input	System Response
	1	indica que projecto não existe

Figura 8: Use case Regista orçamento

Super Use Case	Regista Prestação	
Author		
Date	10/Nov/2014 22:52:45	
Brief Description		
Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Prestação fica registada	
Flow of Events	Actor Input	System Response
	1 indica nome ou id projecto	verifica existencia projeto
	2	apresenta lista projectos
	3	
	4 seleciona projecto	apresenta detalhes projeto
	5	
	6 indica prestação	regista prestação
	7	guarda projeto
	8	
Excepção 1 [projecto não existe] (passo 2)	Actor Input	System Response
	1	indica que projecto não existe

Figura 9: Use case Regista Prestação

Super Use Case	Registrar Reparação	
Author		
Date	13/Nov/2014 21:25:38	
Brief Description		
Preconditions	Utilizador Registado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Pedido reparação registado	
Flow of Events	Actor Input	System Response
	1 indica nome ou id do projecto	
	2	verifica se projecto existe
	3	apresenta lista de projectos
	4 seleciona projecto	apresenta detalhes projeto
	5	
	6 indica detalhes da reparação	regista pedido de reparação
	7	guarda projeto
	8	
Excepção [projecto não existe] (passo 2)	Actor Input	System Response
	1	projecto não existe

Figura 10: Use case Regista Reparação

2.2.2 Novos use cases

De seguida expomos os novos *use cases* que implementamos:

Foram criados tres novos *use cases*:

- Associa material a tarefa;
- Associa candidatura a projecto;
- Consultar projecto, por necesseidade após nova análise do projecto, como tal tambem tiveram de ser criados os seus respectivos Use Cases.

Preconditions	Utilizador Registrado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Associa material a tarefa	
Flow of Events	Actor Input	System Response
	1 indica nome ou id do projecto	verifica se projecto existe
	2	apresenta lista de projectos
	3	
	4 seleciona projecto	apresenta tarefas
	5	
	6 seleciona tarefa	apresenta detalhes da tarefa
	7	
	8 indica material gasto	associa material a tarefa
Excepção 1 [projecto não existe] (passo 2)	Actor Input	System Response
	1	indica que projecto não existe

Figura 11: Use case Associa Material a Tarefa

Preconditions	Utilizador Registrado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Associa projecto a candidatura	
Flow of Events	Actor Input	System Response
	1 indica nome ou id do projecto	verifica se projecto existe
	2	
	3 indica nome ou id da candidatura	verifica se candidatura existe
	4	apresenta lista de projectos
	5	
	6 seleciona projecto	apresenta lista de candidaturas
	7	
	8 seleciona candidatura	associa candidatura a projecto
Excepção 1 [projecto não existe] (passo 2)	Actor Input	System Response
	1	indica que projecto não existe
Excepção 2 [candidatura não existe] (passo 4)	Actor Input	System Response
	1	indica que candidatura não existe

Figura 12: Use case Consultar Projeto

Preconditions	Utilizador Registrado como Staff Comissão de Construção e Projecto Aberto	
Post-conditions	Projecto consultado	
Flow of Events	Actor Input	System Response
	1 indica nome ou id do projecto	
	2	verifica se projecto existe
	3	apresenta lista de projectos
	4 seleciona projecto	apresenta detalhes do projecto
	5	
Excepção 1 [projecto não existe] (passo 2)	Actor Input	System Response
	1	indica que projecto não existe

Figura 13: Use case Remover Tarefa

3 Diagramas de Seguência de Sistemas

Nesta secção vamos expor todos os diagramas de sequência de uses cases que foram gerados diretamente a partir dos *use cases*, estes diagramas são **diagramas de sequência de sistemas**. Iremos poder observar de uma forma generalizada todo o conjunto de operações relevantes da aplicação.

3.1 Autenticação

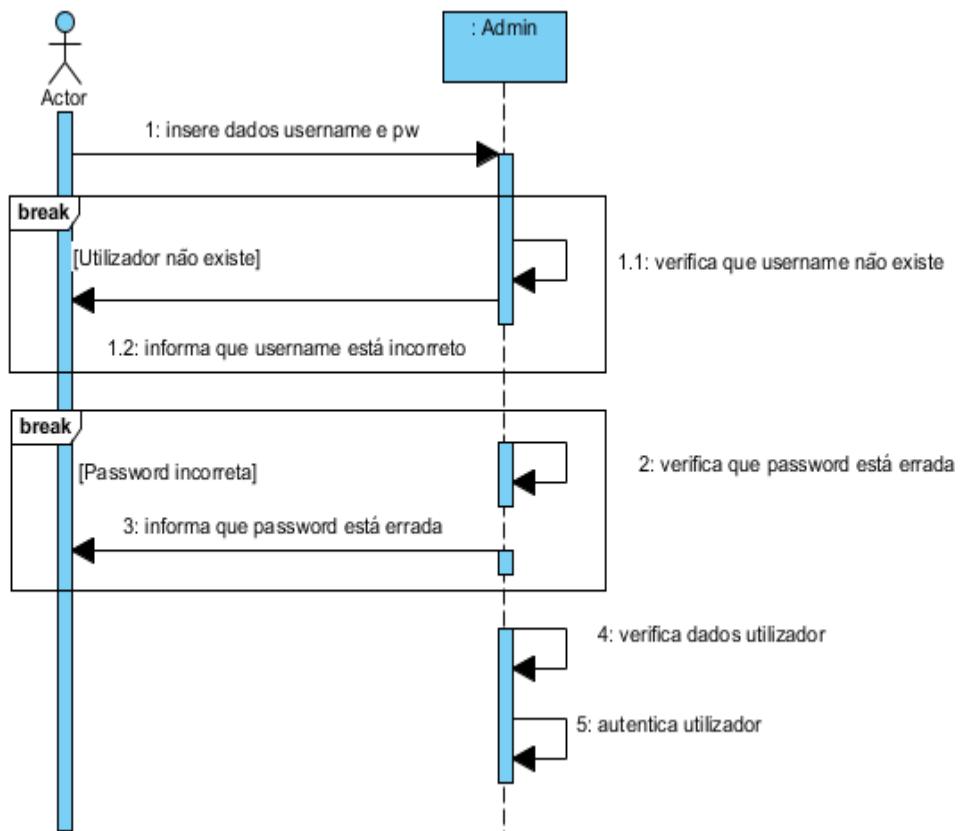
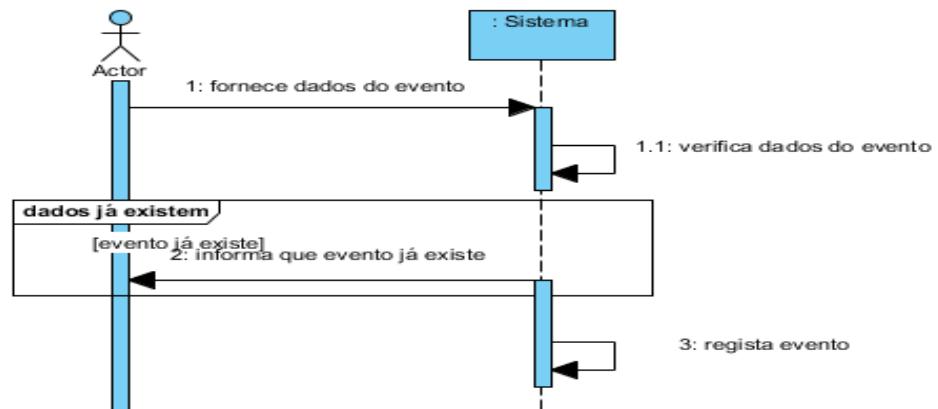
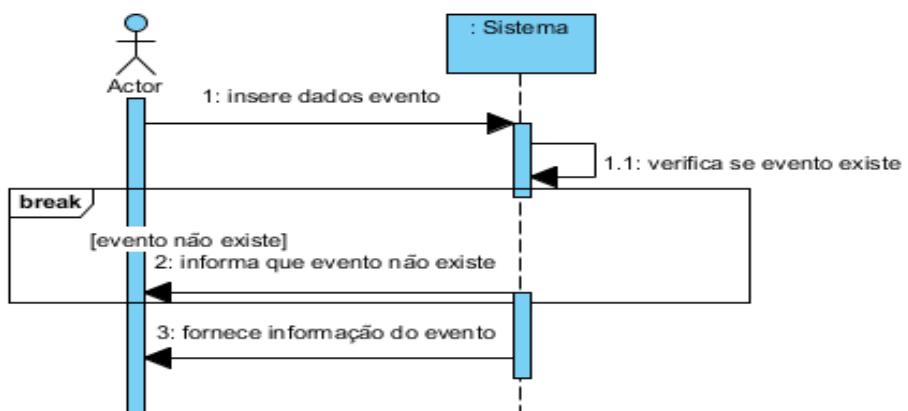


Figura 14: Diagrama de sequência Autenticação

3.2 Doações

Figura 15: Diagrama de sequência **Registrar Evento**Figura 16: Diagrama de sequência **Consultar Evento**

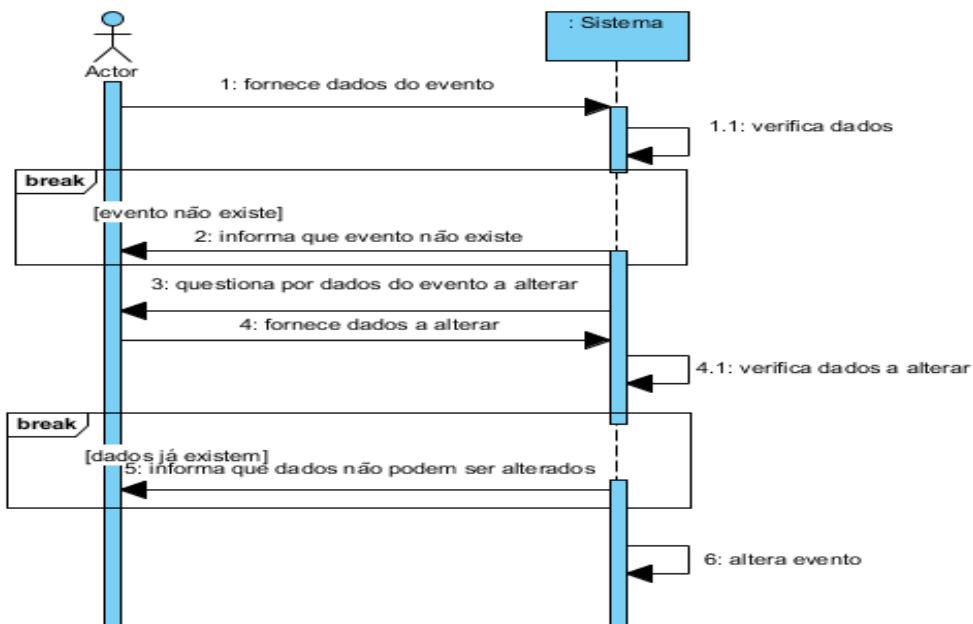


Figura 17: Diagrama de sequência Editar Evento

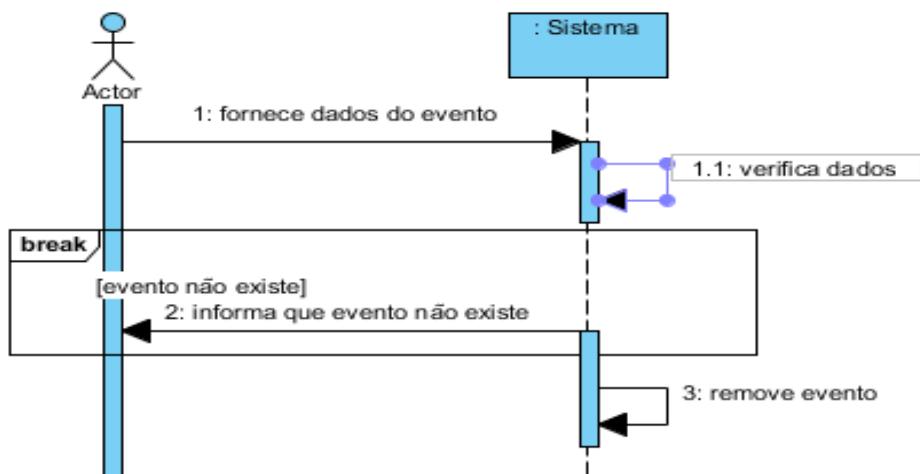


Figura 18: Diagrama de sequência Remove Evento

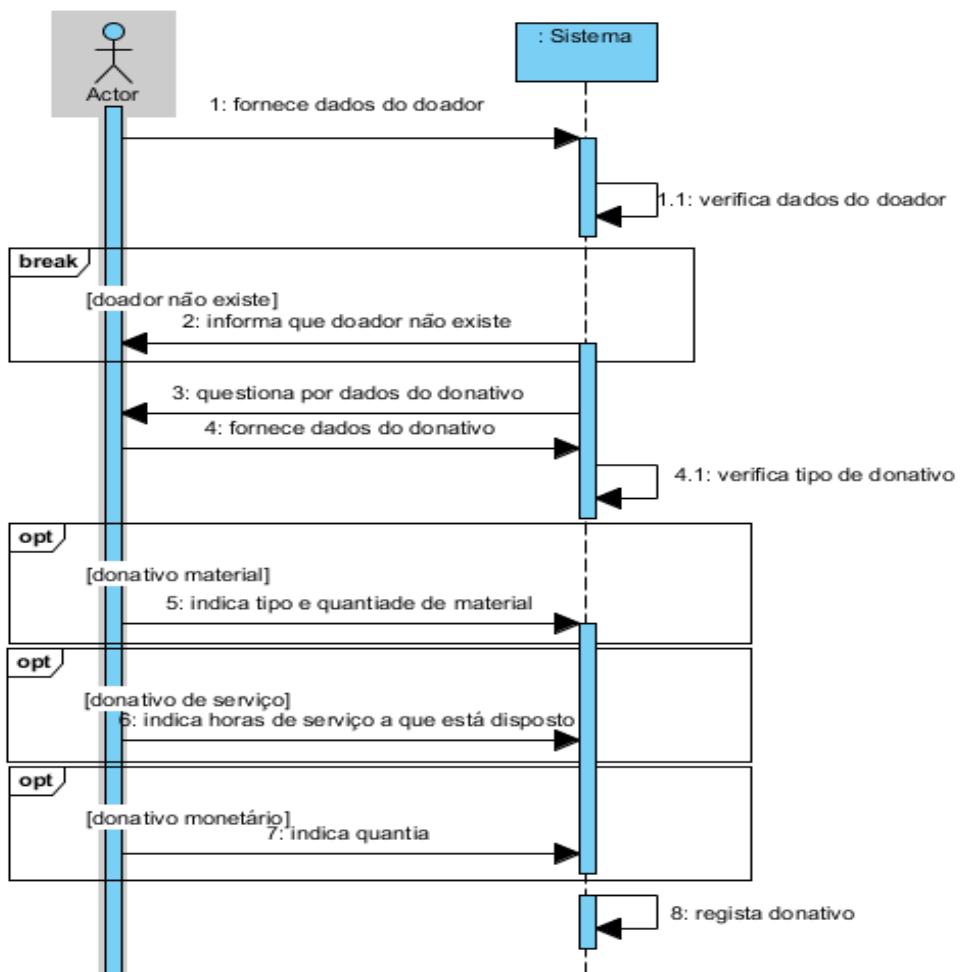


Figura 19: Diagrama de sequência Registar Donativo

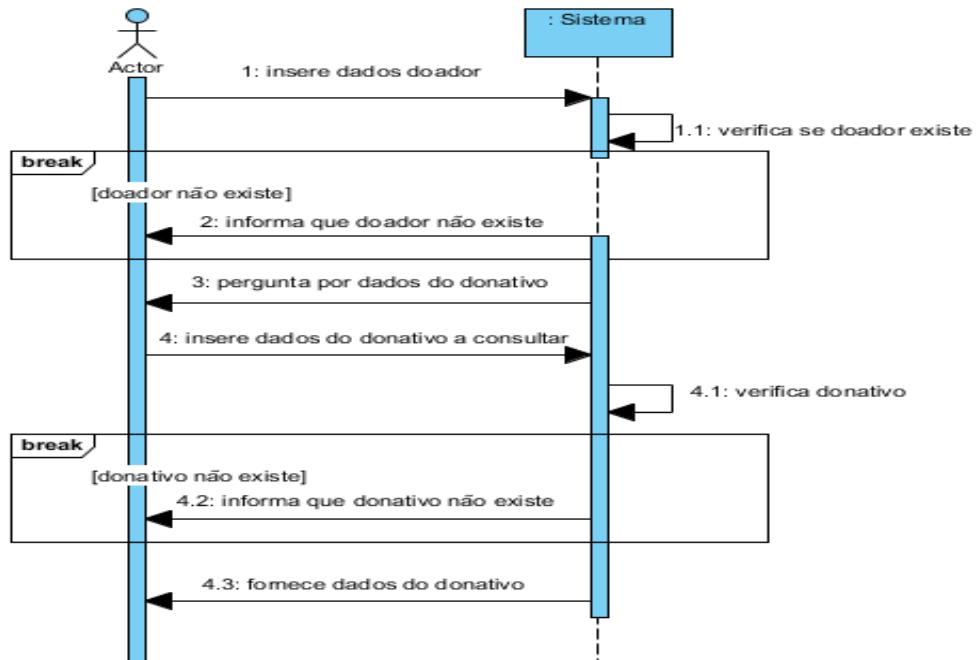


Figura 20: Diagrama de sequência Consultar Donativo

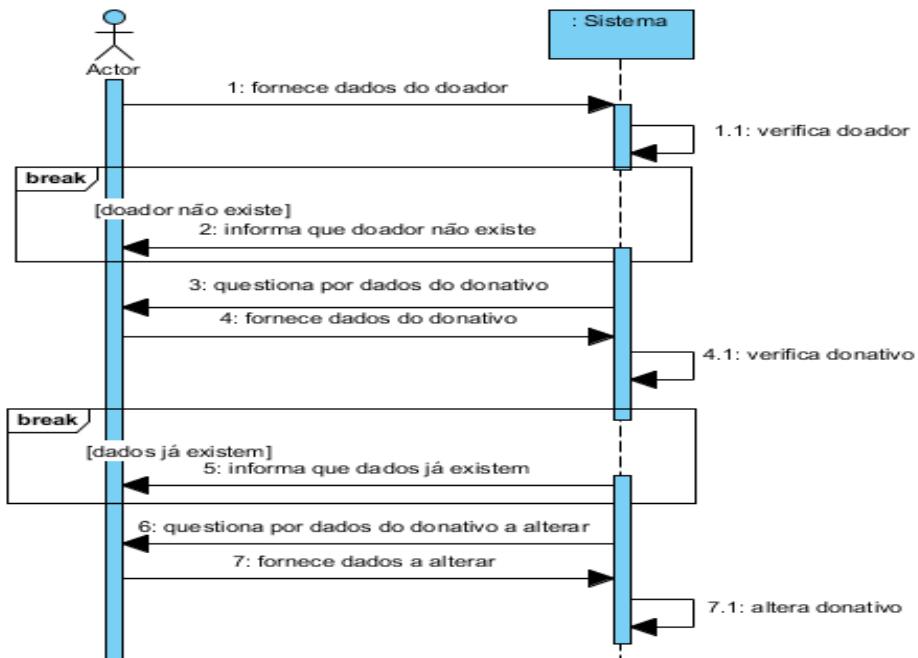


Figura 21: Diagrama de sequência Editar Donativo

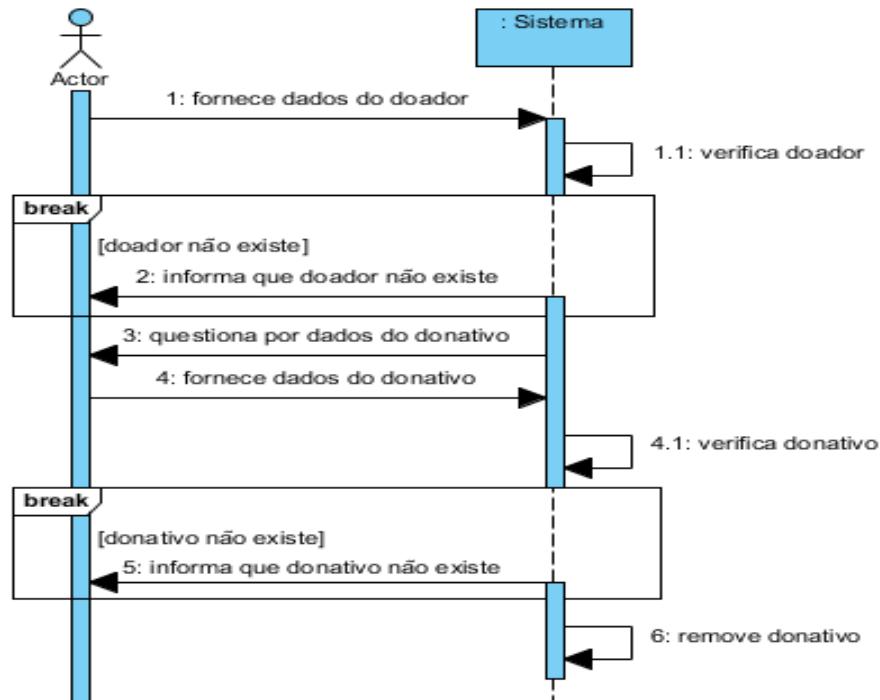


Figura 22: Diagrama de sequência Remover Donativo

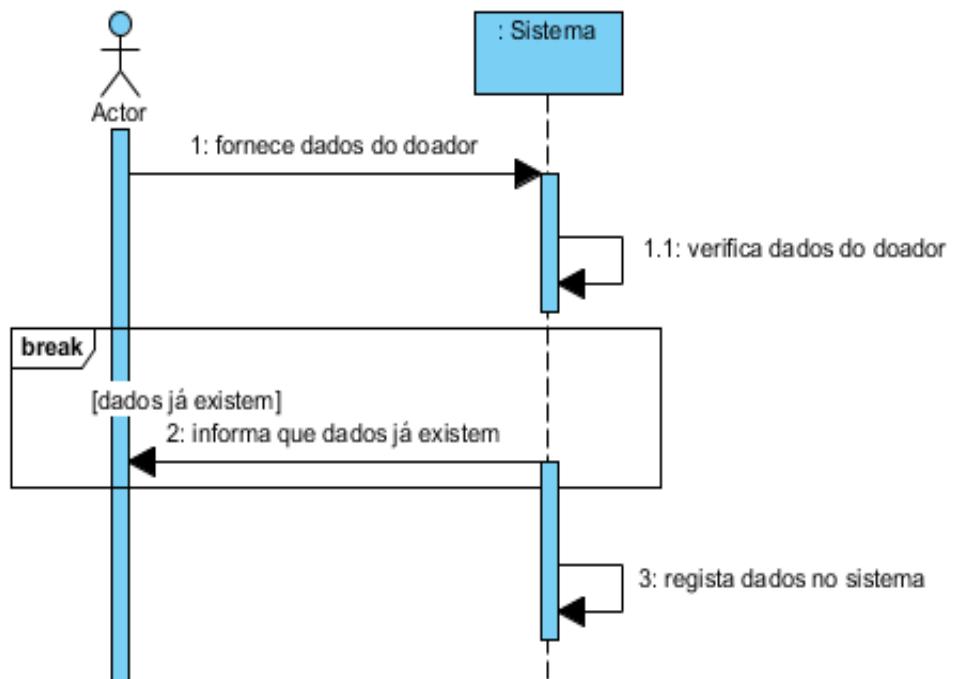
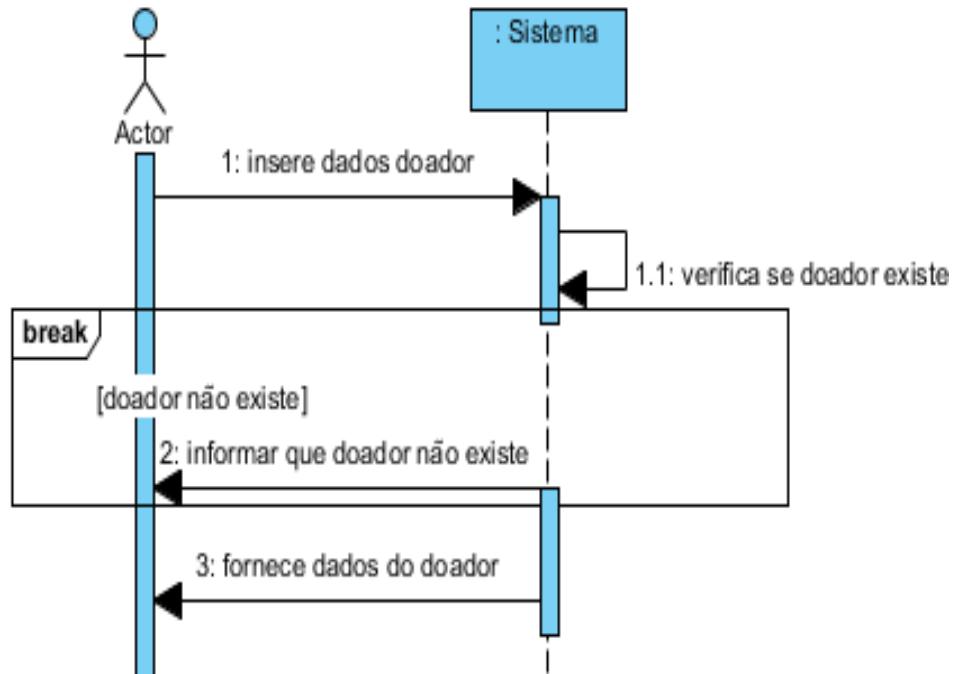
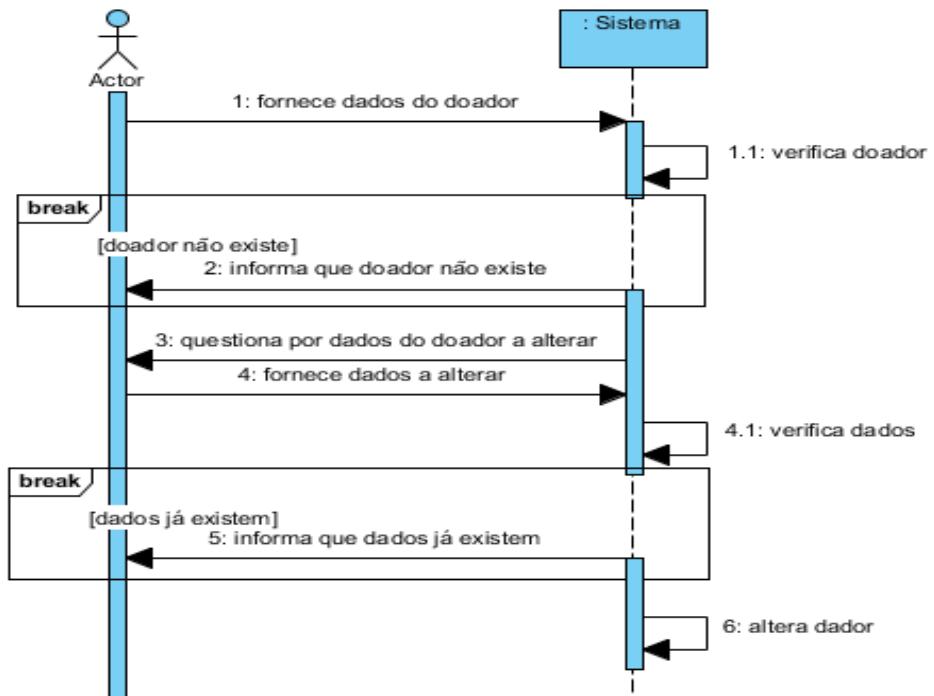
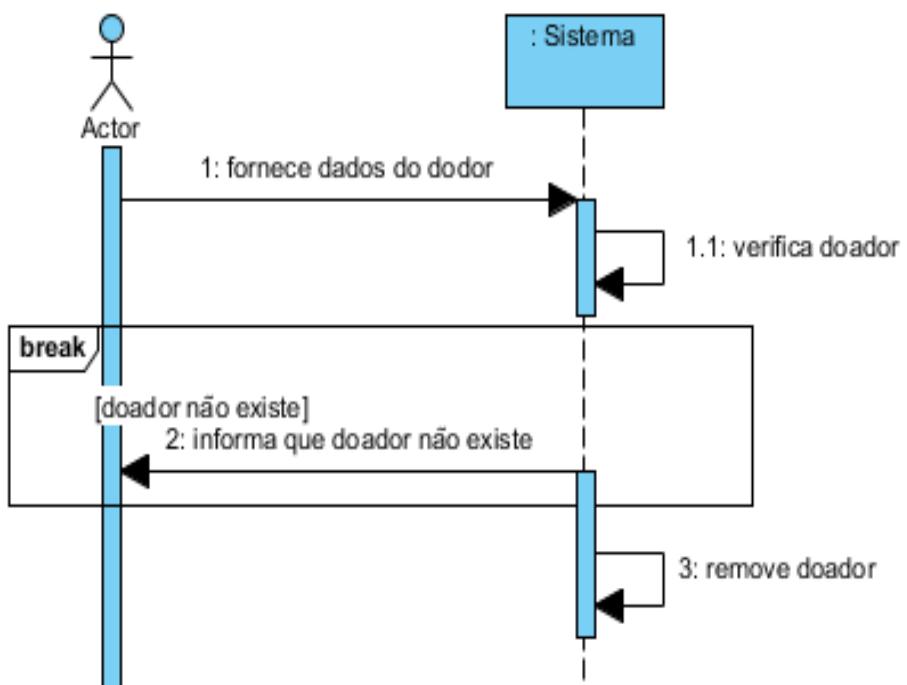


Figura 23: Diagrama de sequência Registar Doador

Figura 24: Diagrama de sequência **Consultar Doador**Figura 25: Diagrama de sequência **Edita Doador**

Figura 26: Diagrama de sequência **Remover Doador**

3.3 Famílias

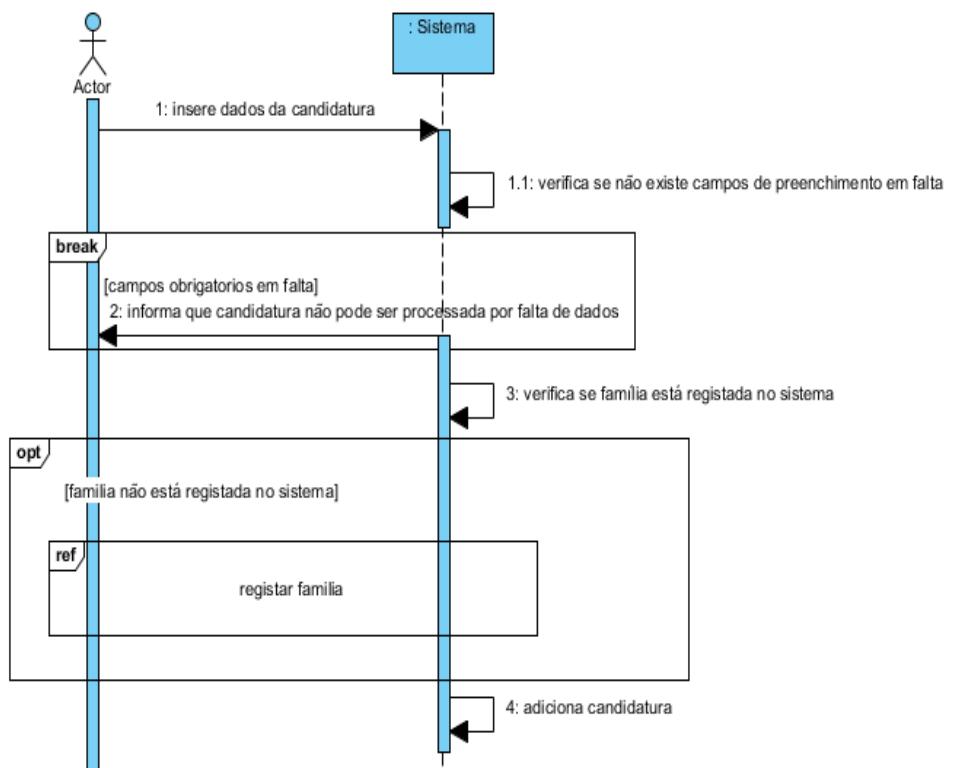
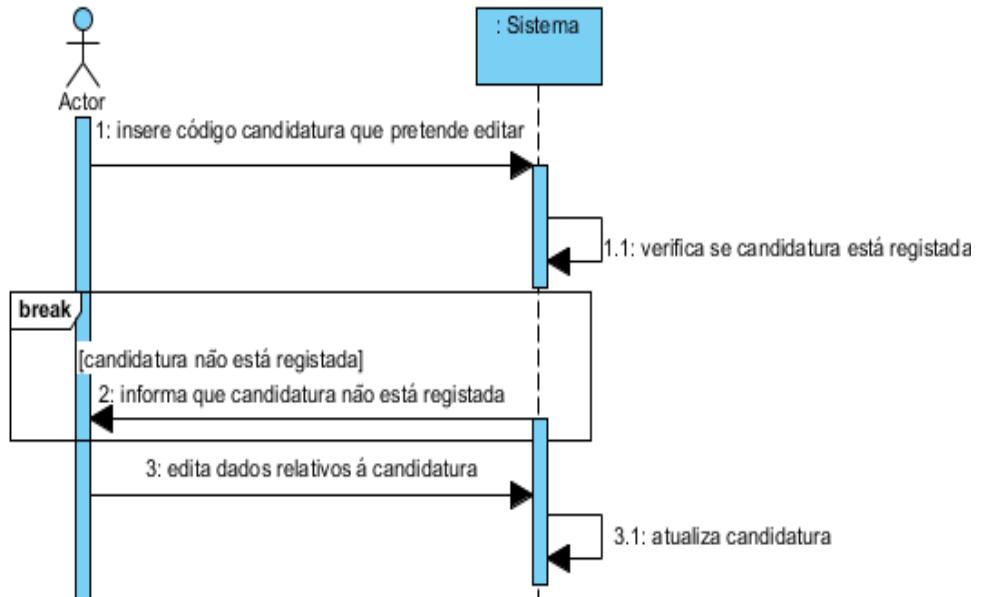
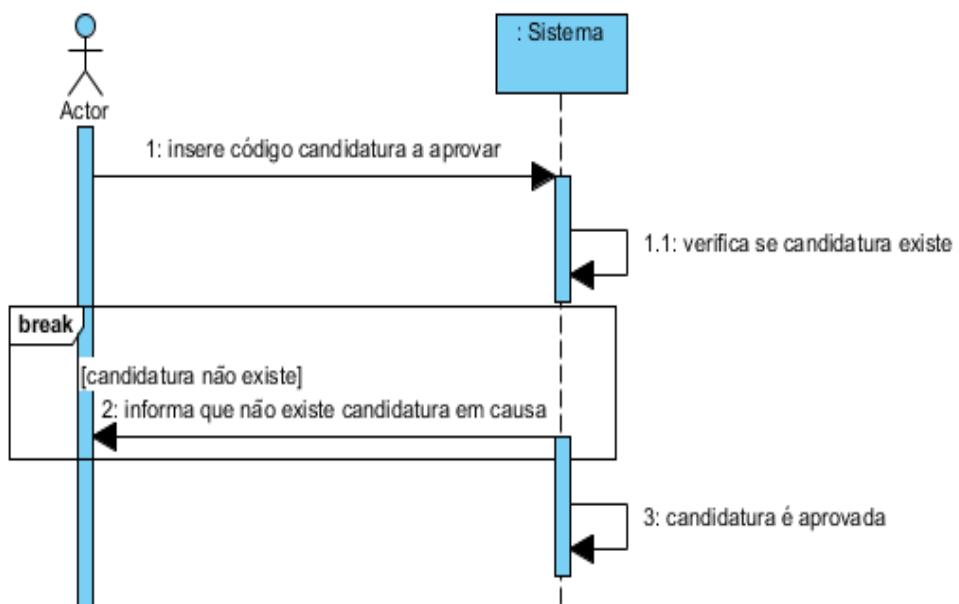
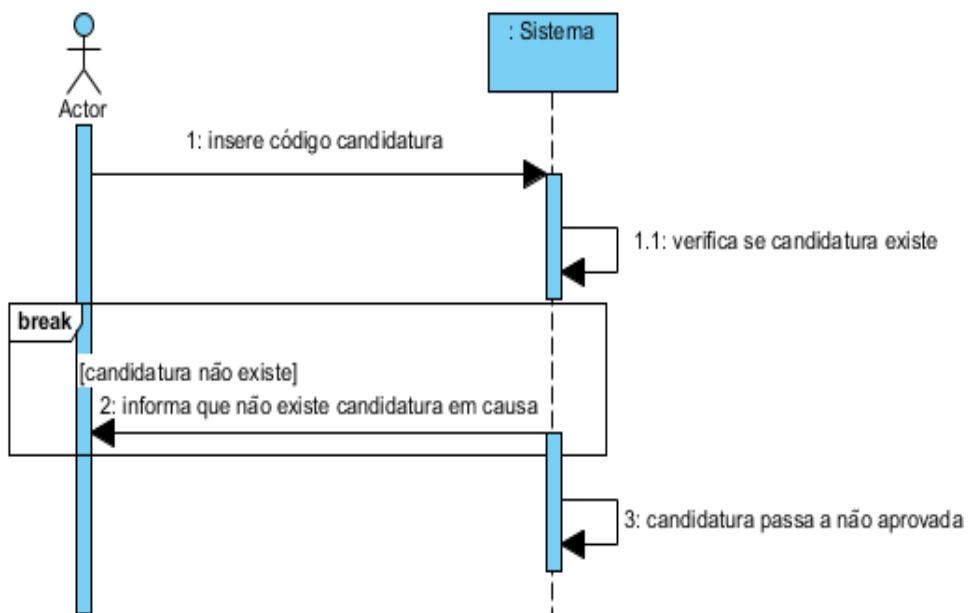
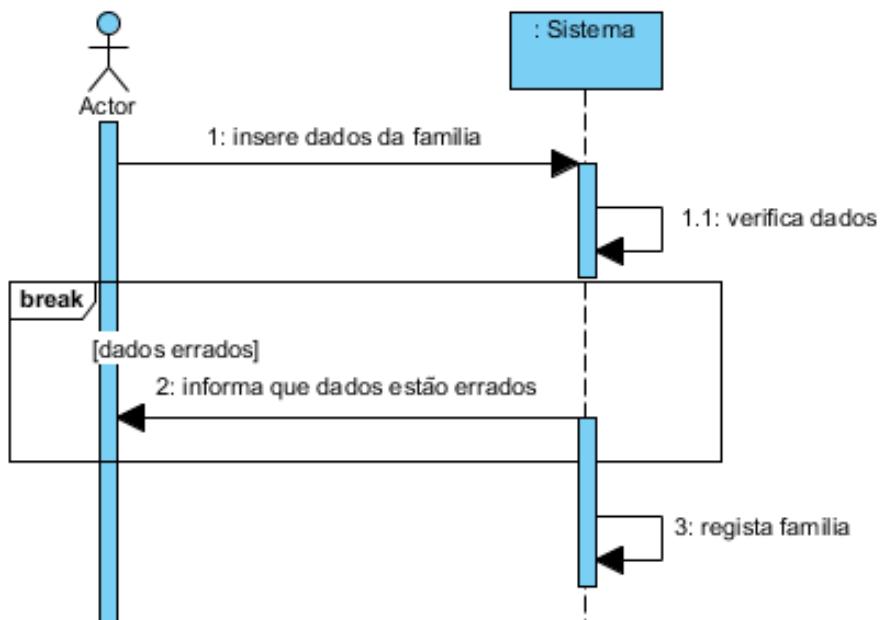
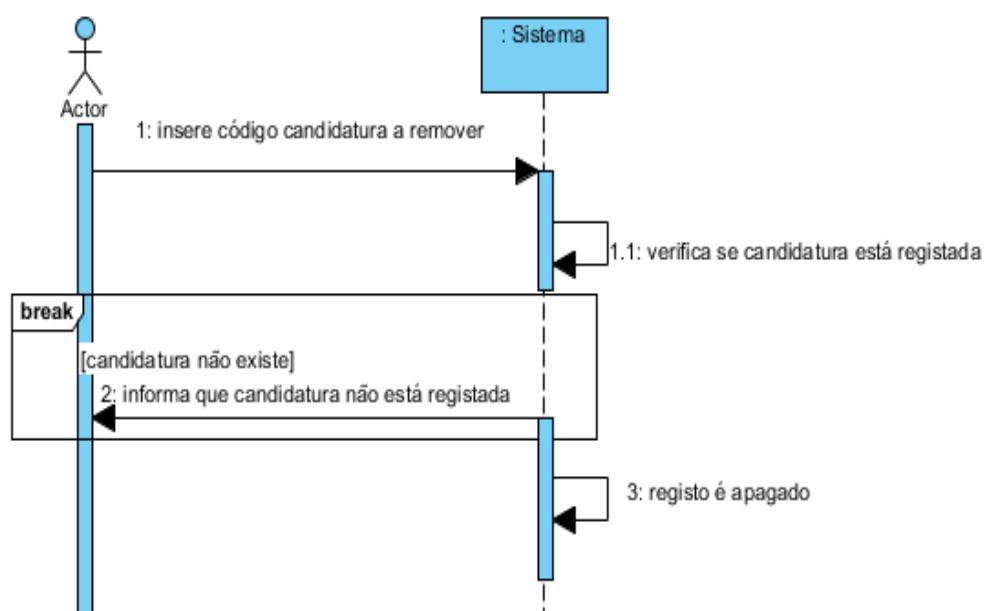


Figura 27: Diagrama de sequência **Adicionar Candidatura**

Figura 28: Diagrama de sequência **Edita Candidatura**Figura 29: Diagrama de sequência **Aprovar Candidatura**

Figura 30: Diagrama de sequência **Rejeitar Candidatura**Figura 31: Diagrama de sequência **Registrar Família**

Figura 32: Diagrama de sequência **Remover Candidatura**

3.4 Projetos

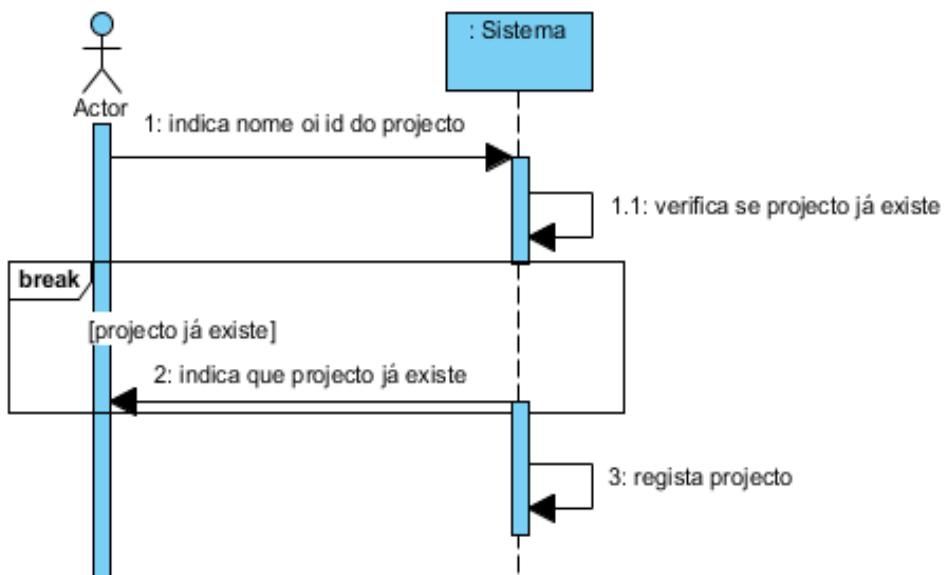
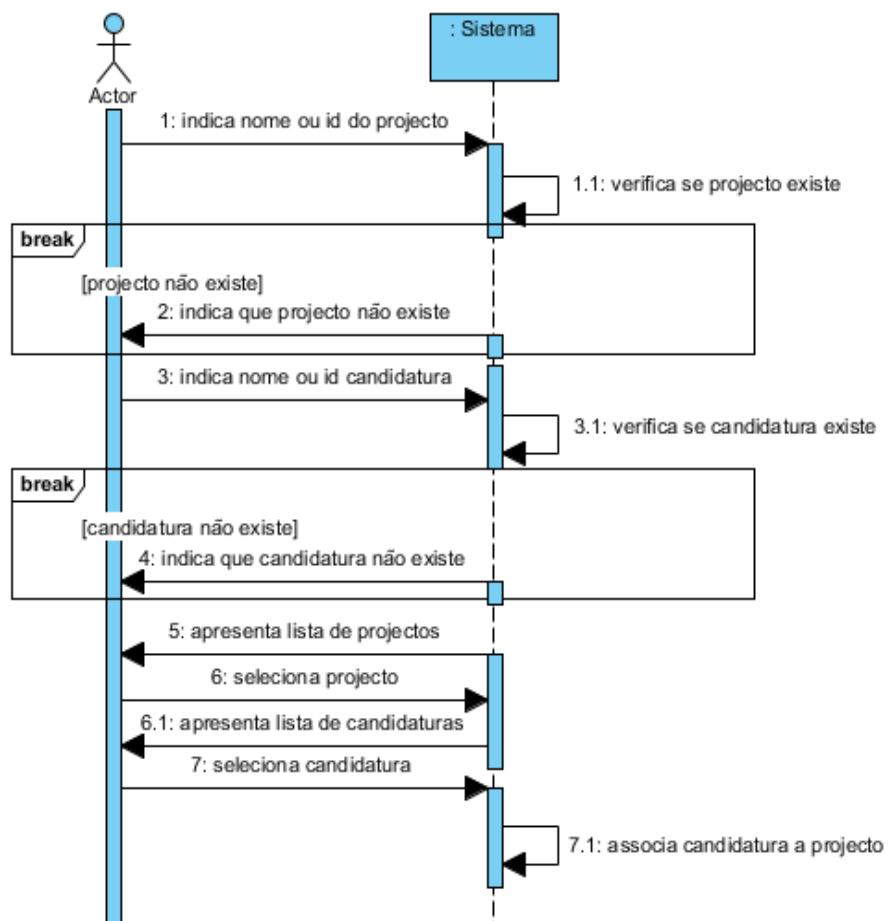
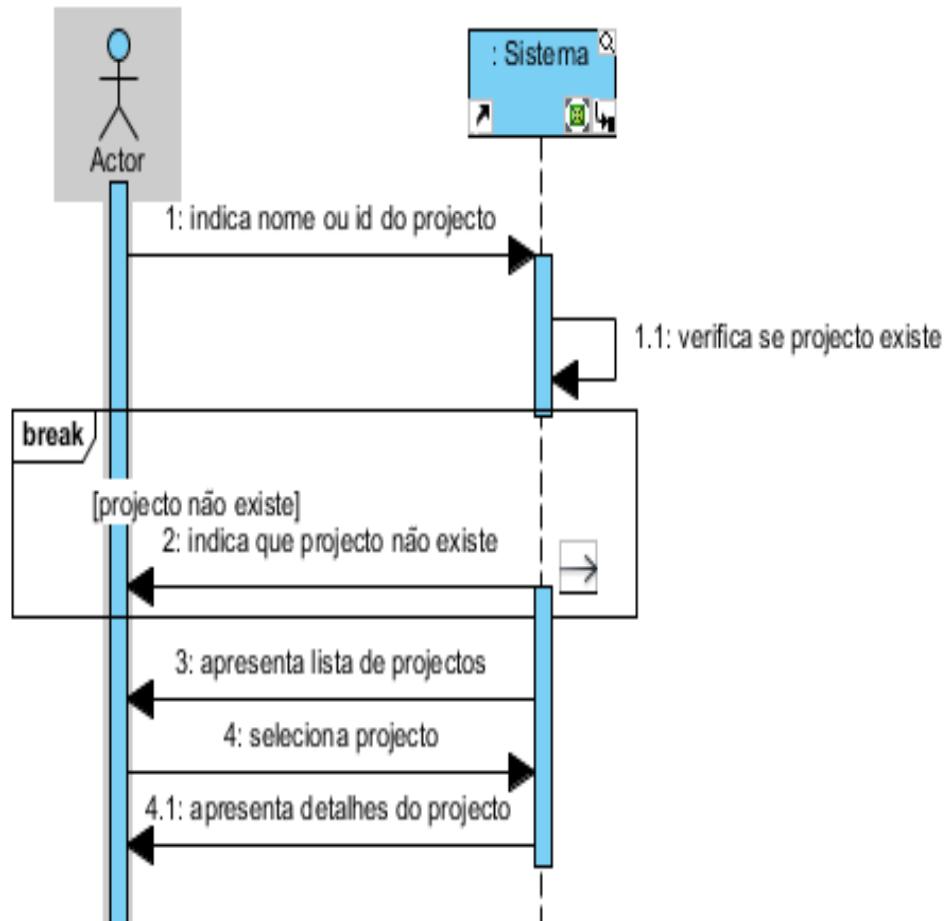


Figura 33: Diagrama de sequência Adicionar Projeto

Figura 34: Diagrama de sequência **Associa Candidatura a Projeto**

Figura 35: Diagrama de sequência **Consulta Projeto**

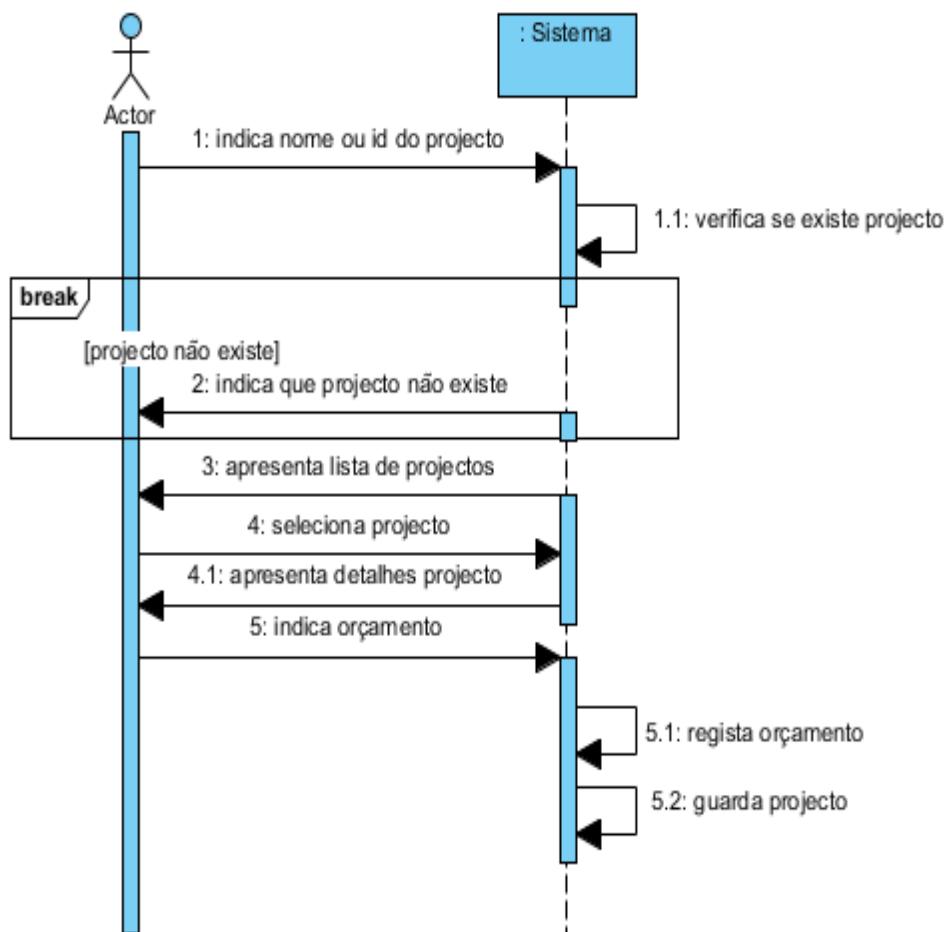
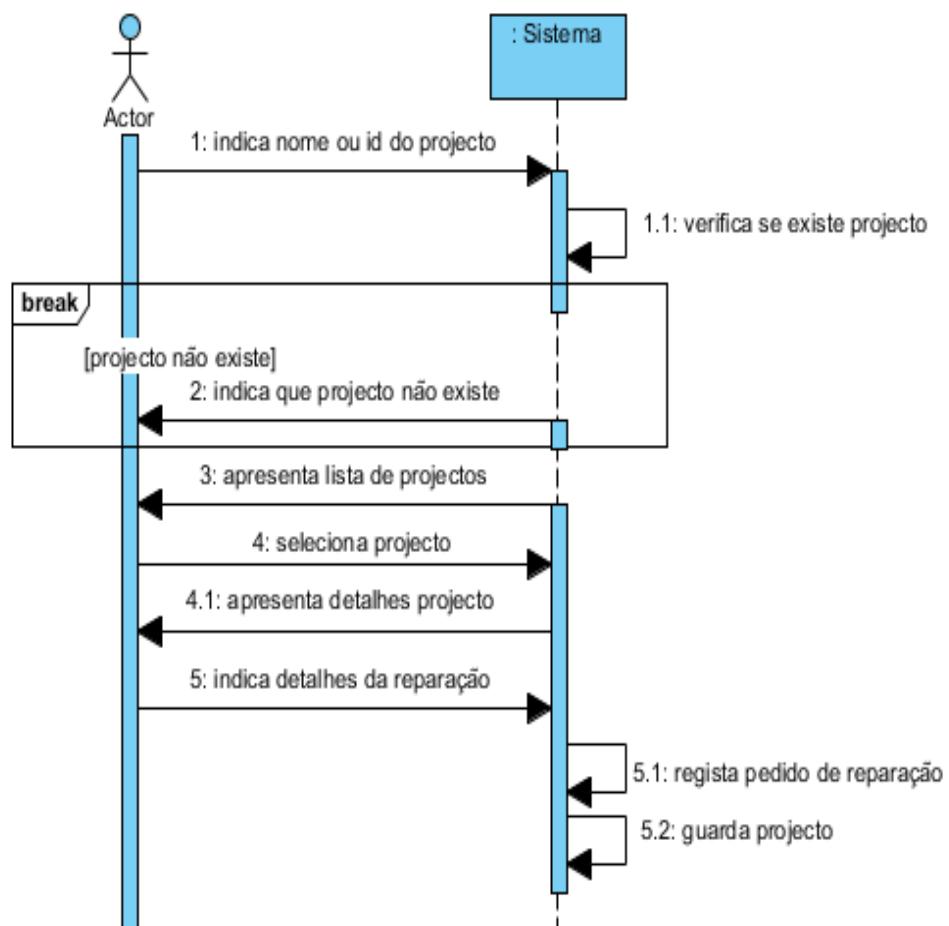
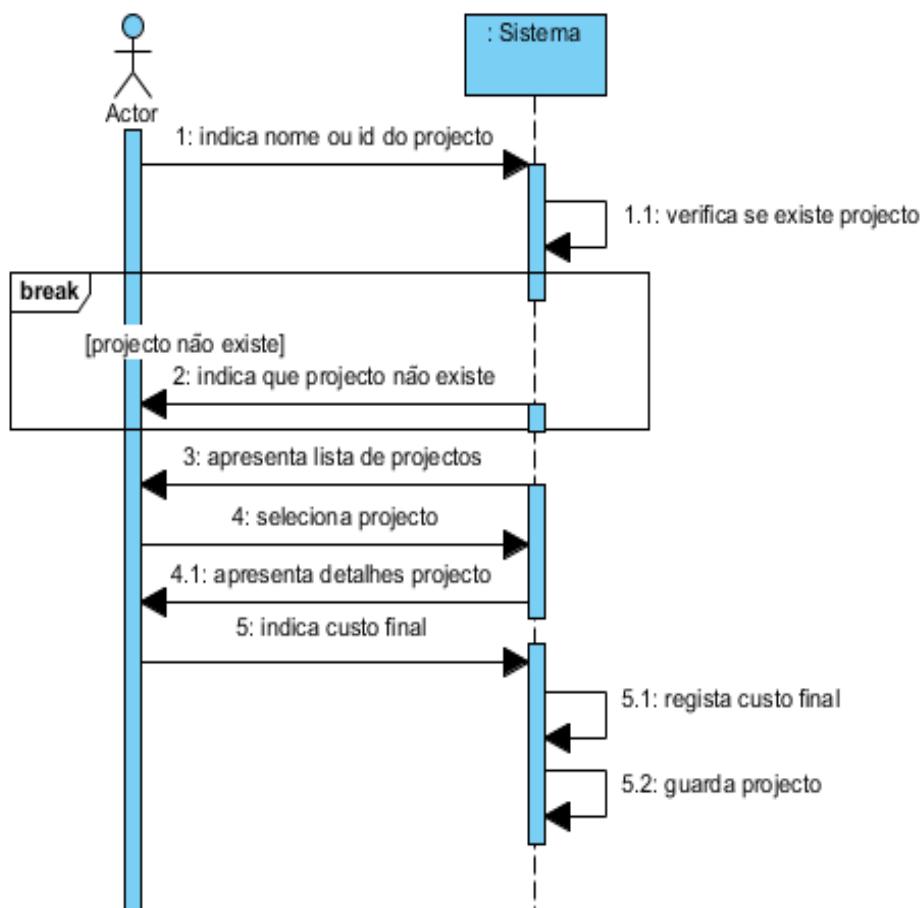
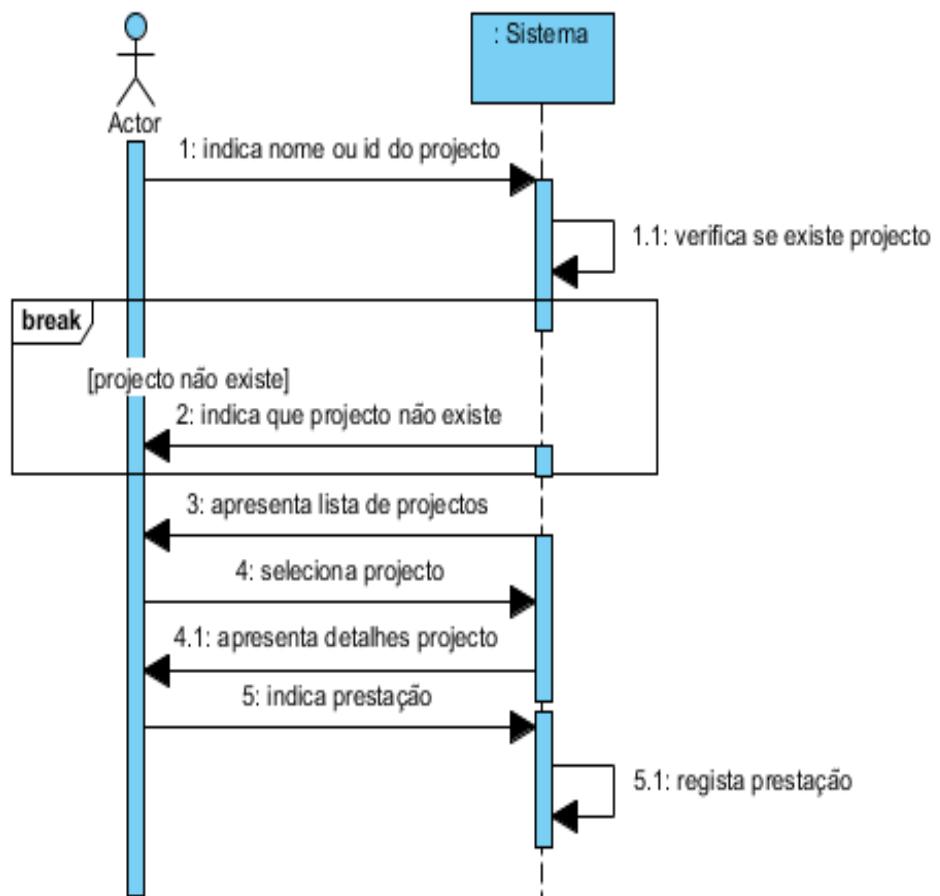


Figura 36: Diagrama de sequência Regista Orçamento

Figura 37: Diagrama de sequência **Regista Reparação**

Figura 38: Diagrama de sequência **Regista Custo Final**

Figura 39: Diagrama de sequência **Regista Prestação**

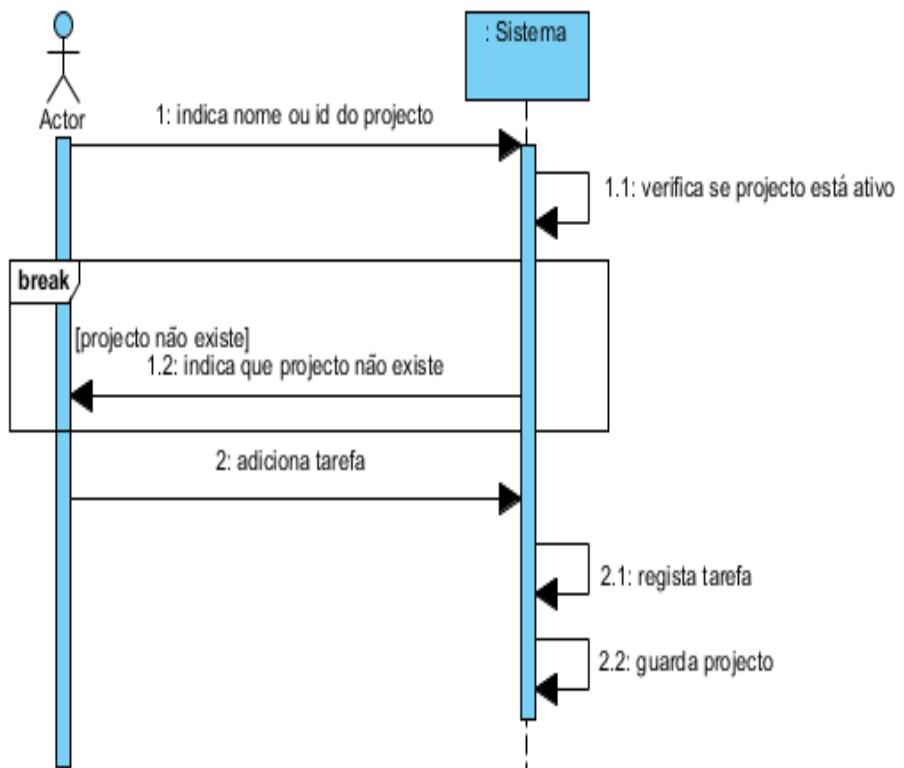


Figura 40: Diagrama de sequência Adicionar Tarefa

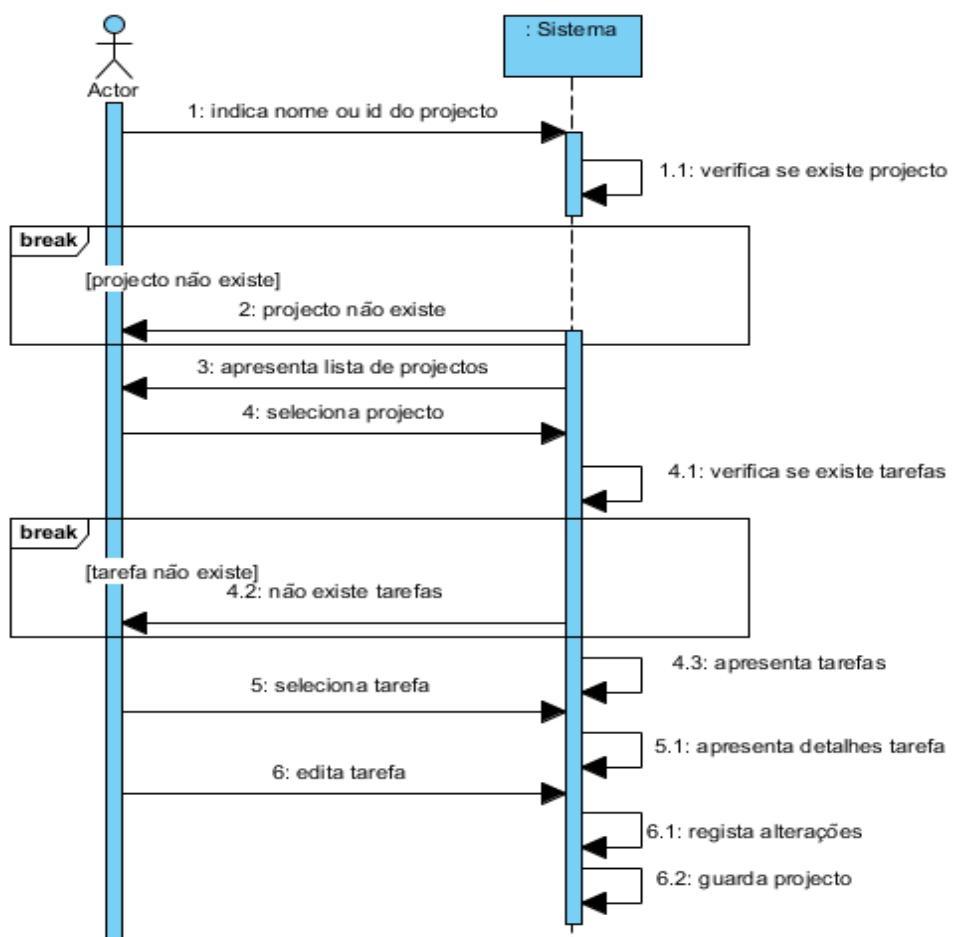


Figura 41: Diagrama de sequência Editar Tarefa

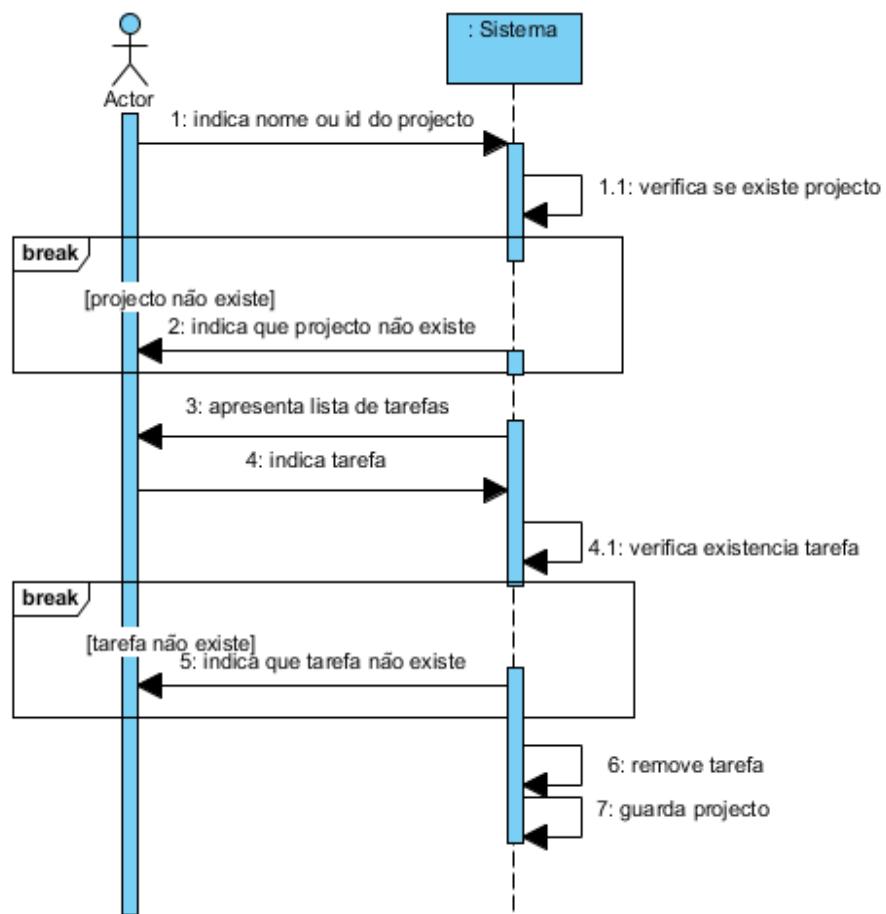
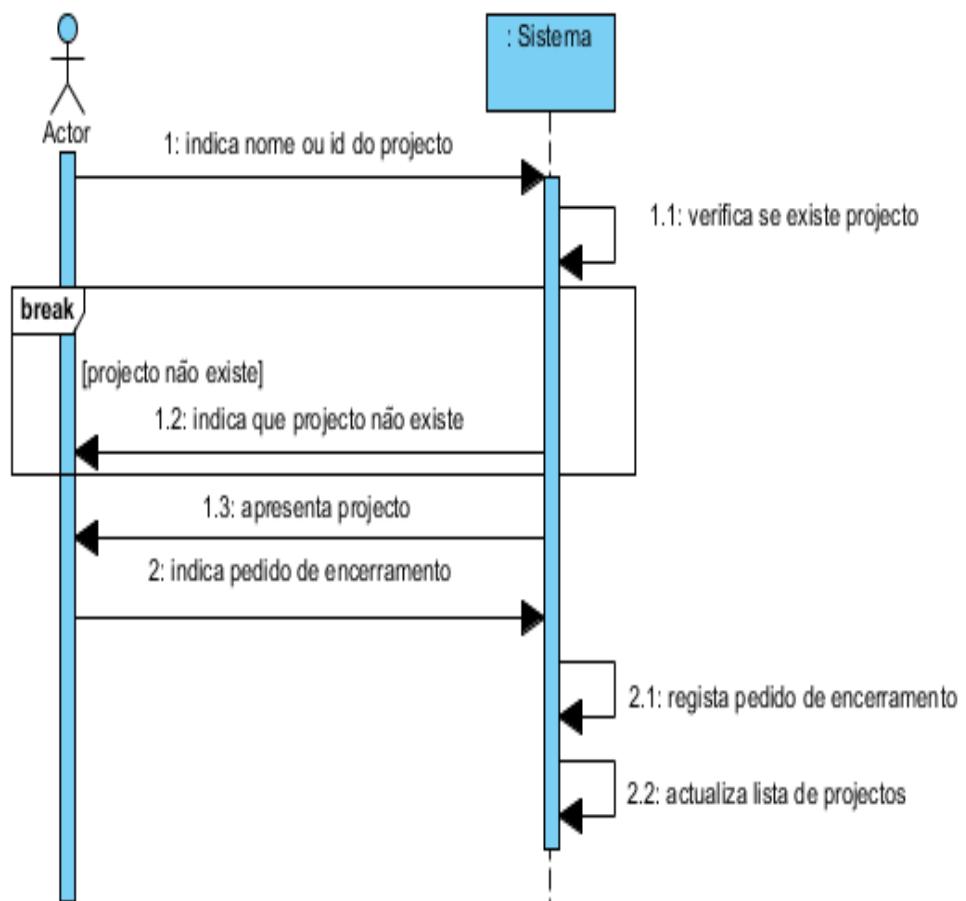
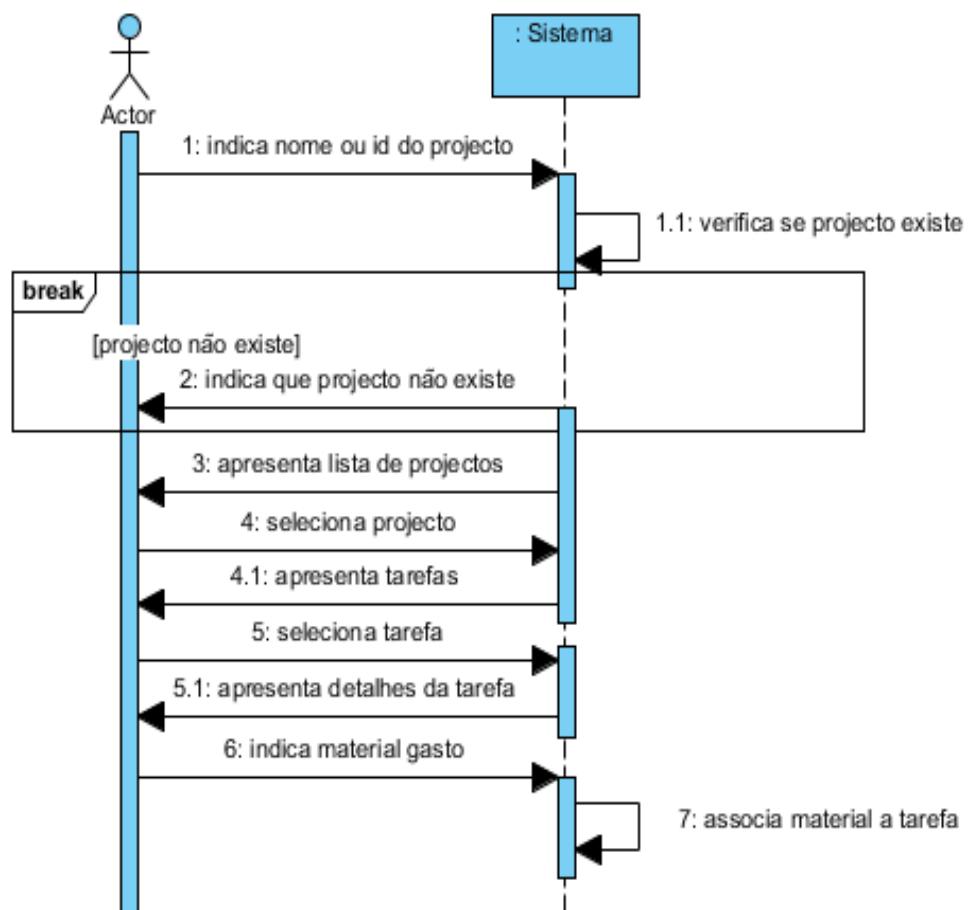
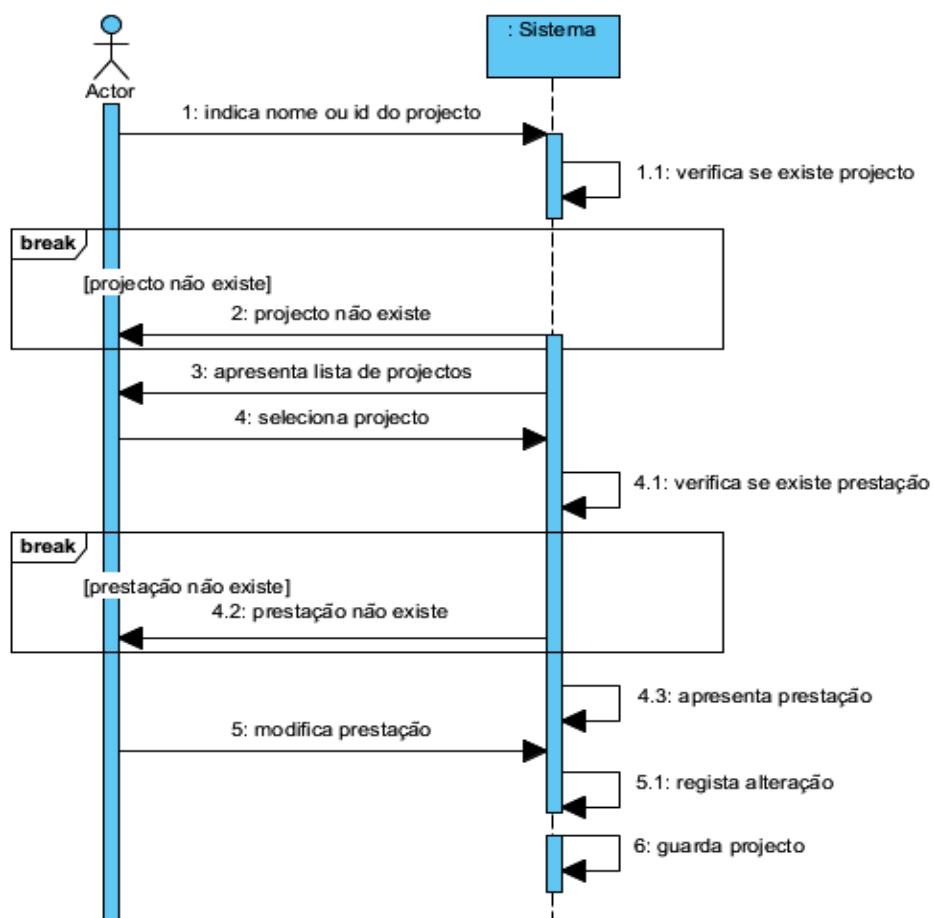


Figura 42: Diagrama de sequência Remover Tarefa

Figura 43: Diagrama de sequência **Encerrar Projeto**

Figura 44: Diagrama de sequência **Associar Material a Tarefa**

Figura 45: Diagrama de sequência **Editar Prestação**

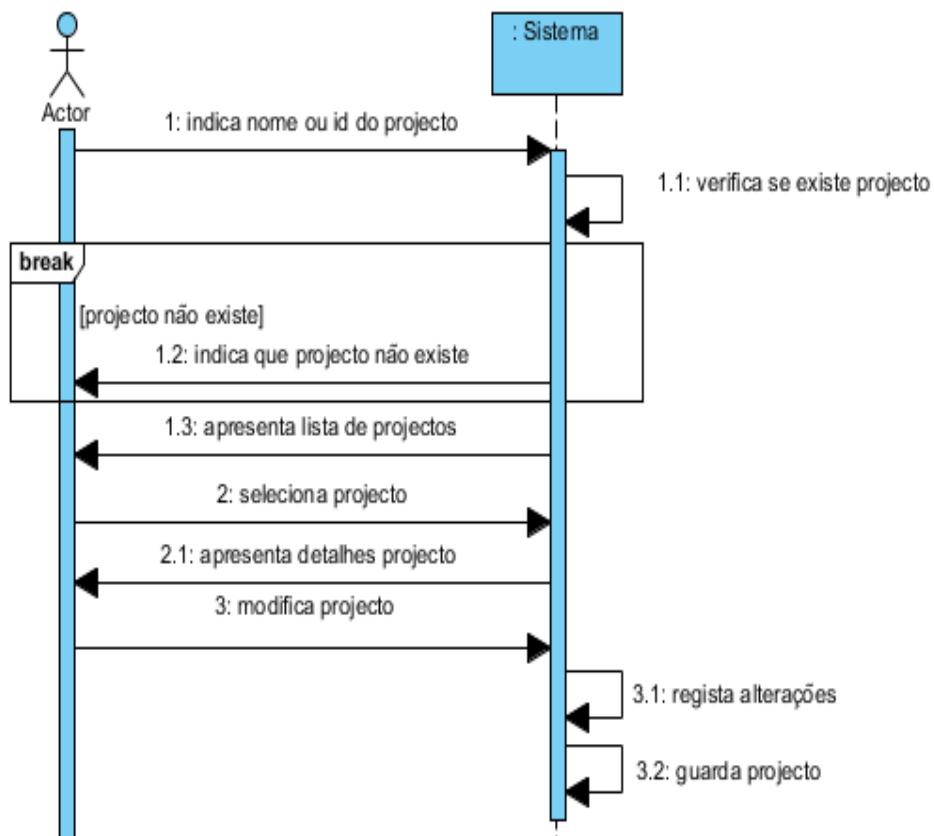


Figura 46: Diagrama de sequência Editar Projeto

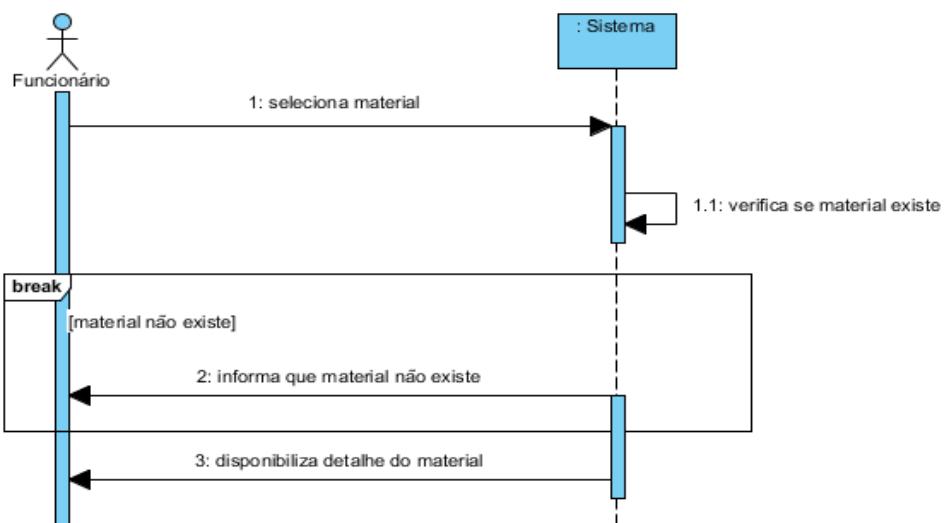


Figura 47: Diagrama de sequência Consultar Stock

3.5 Voluntários

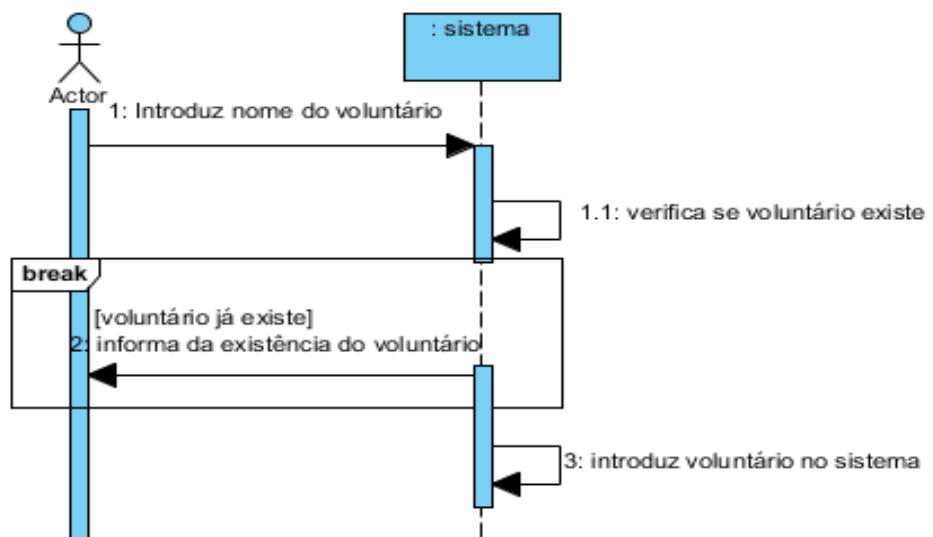


Figura 48: Diagrama de sequênciа Registar Voluntário

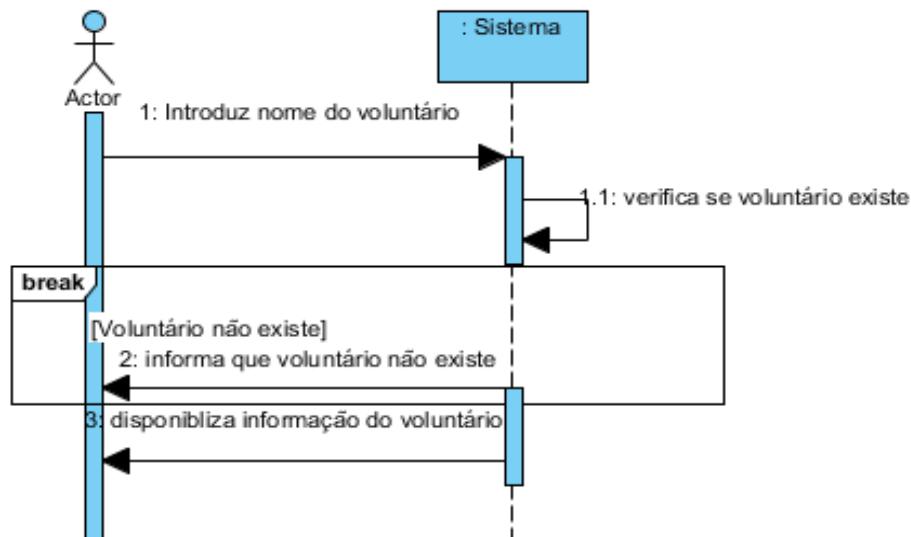


Figura 49: Diagrama de sequência Consultar Voluntário

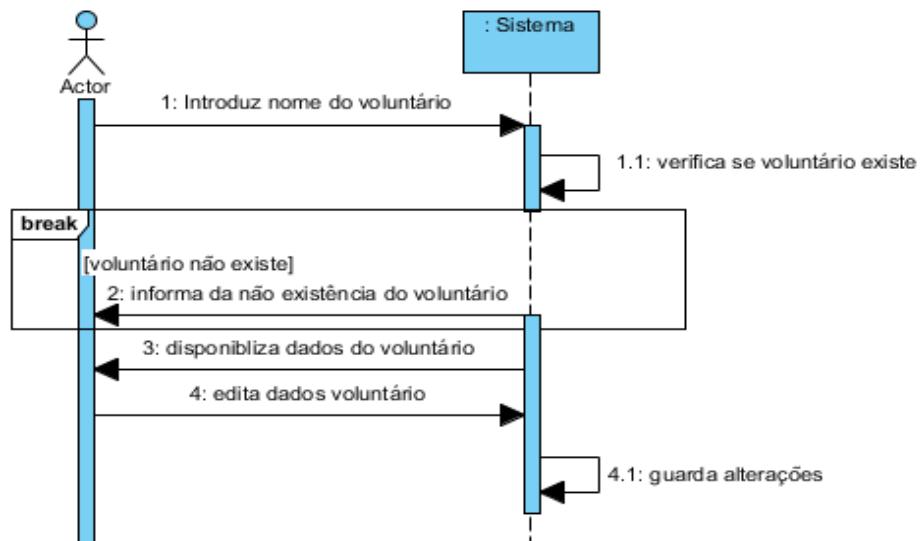


Figura 50: Diagrama de sequência Editar Voluntário

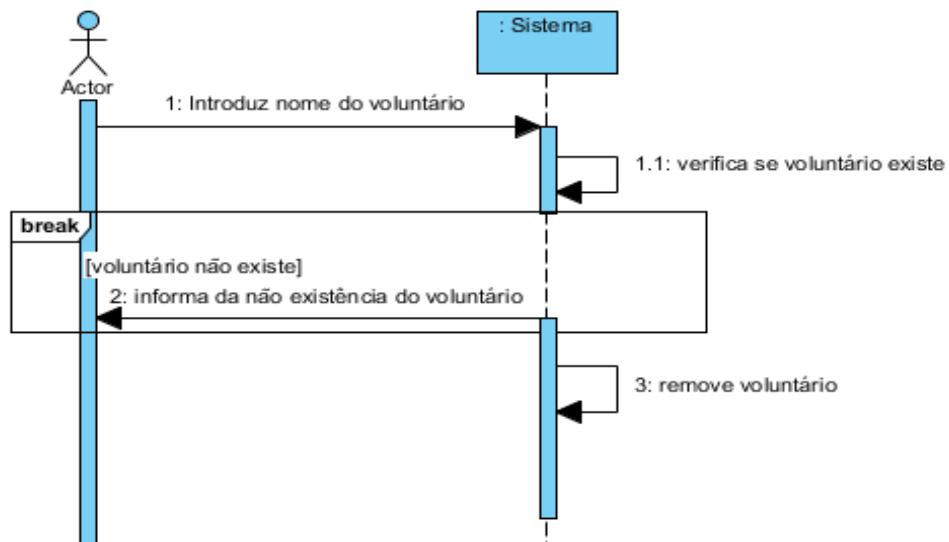


Figura 51: Diagrama de sequência Remover Voluntário

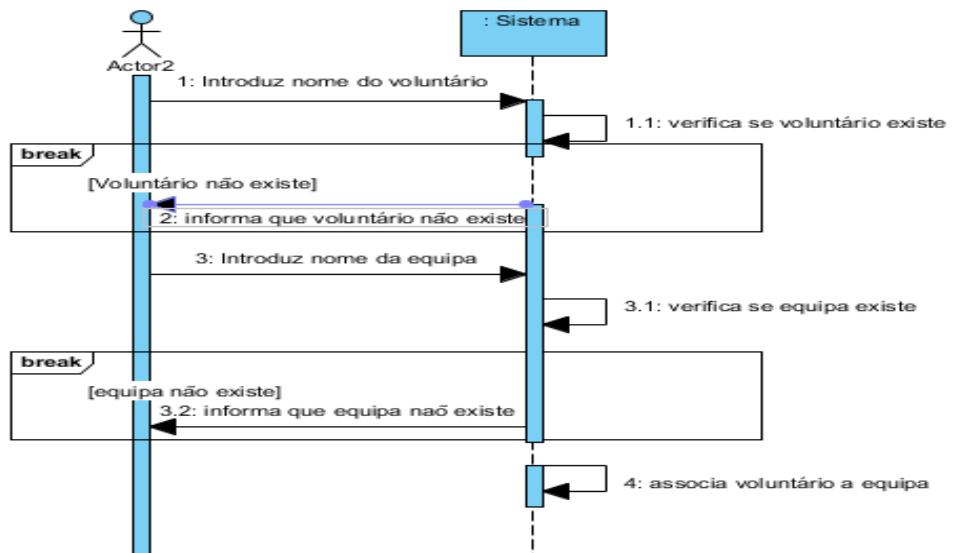


Figura 52: Diagrama de sequência Associa Voluntário a Equipa

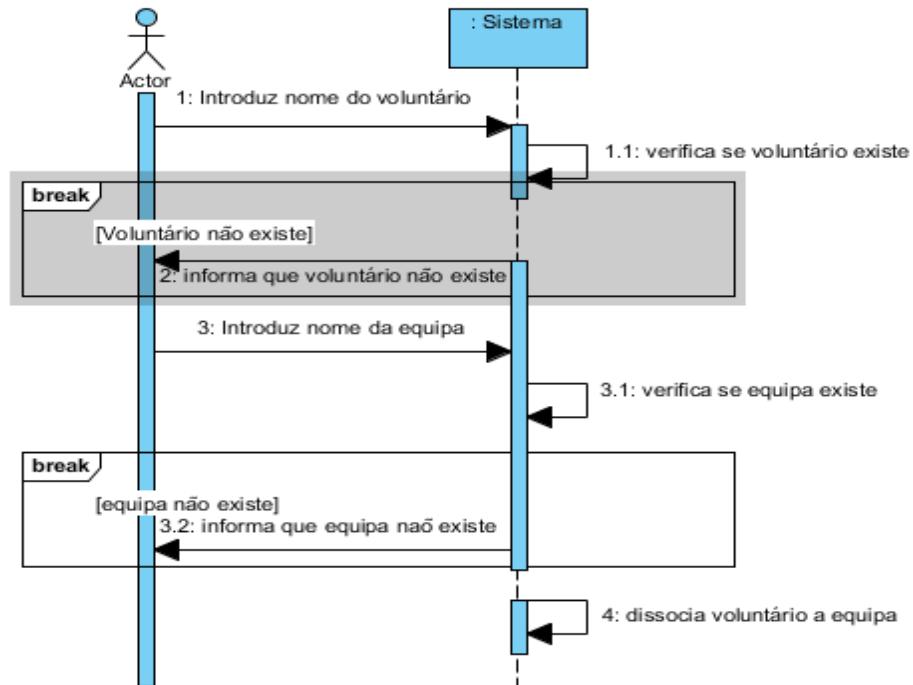


Figura 53: Diagrama de sequência Dissocia Voluntário a Equipa

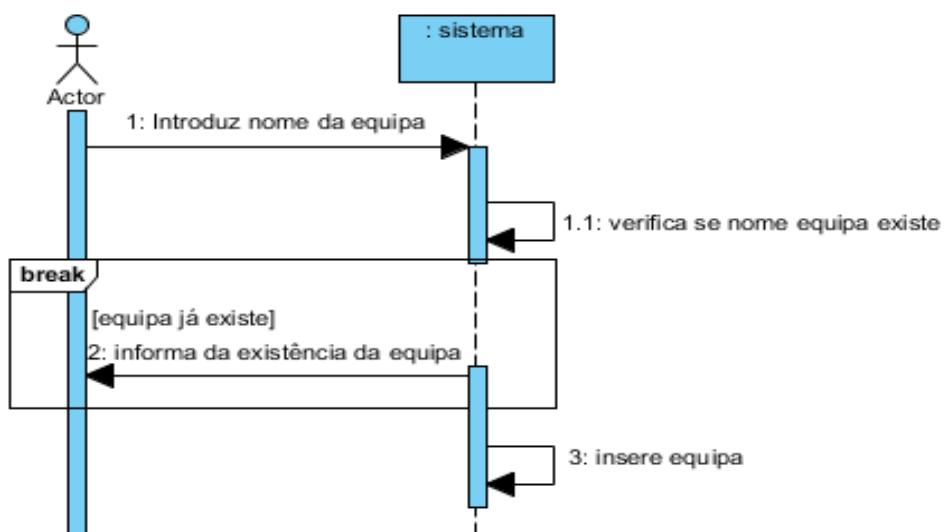


Figura 54: Diagrama de sequência Criar a Equipa

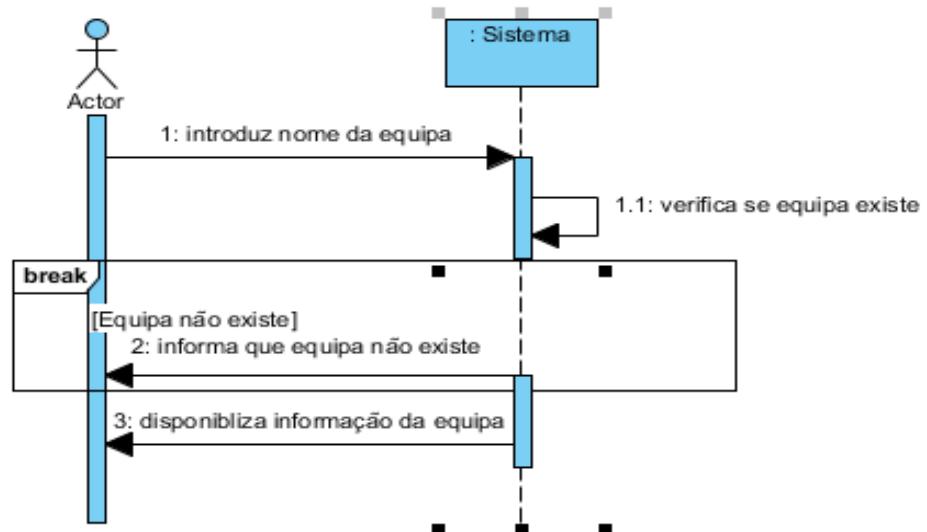


Figura 55: Diagrama de sequência Consultar a Equipa

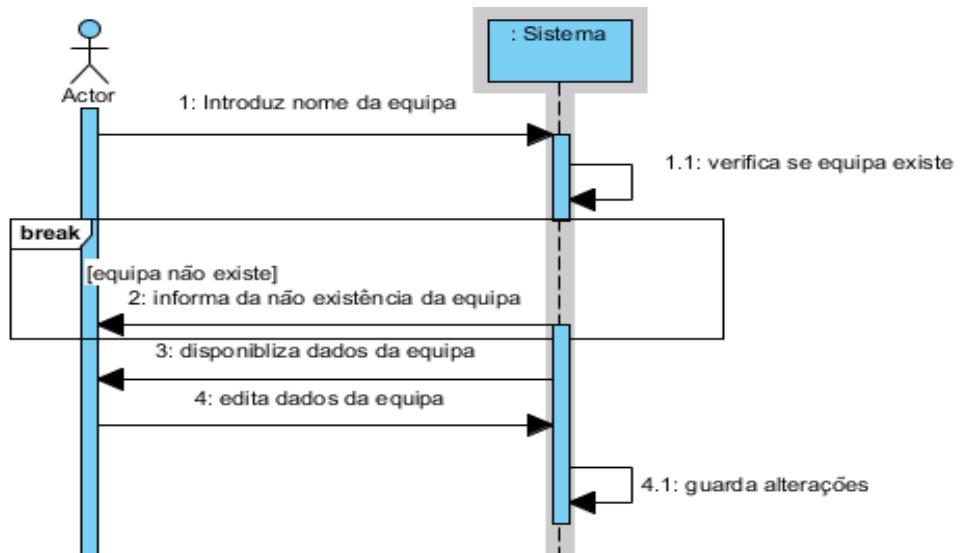
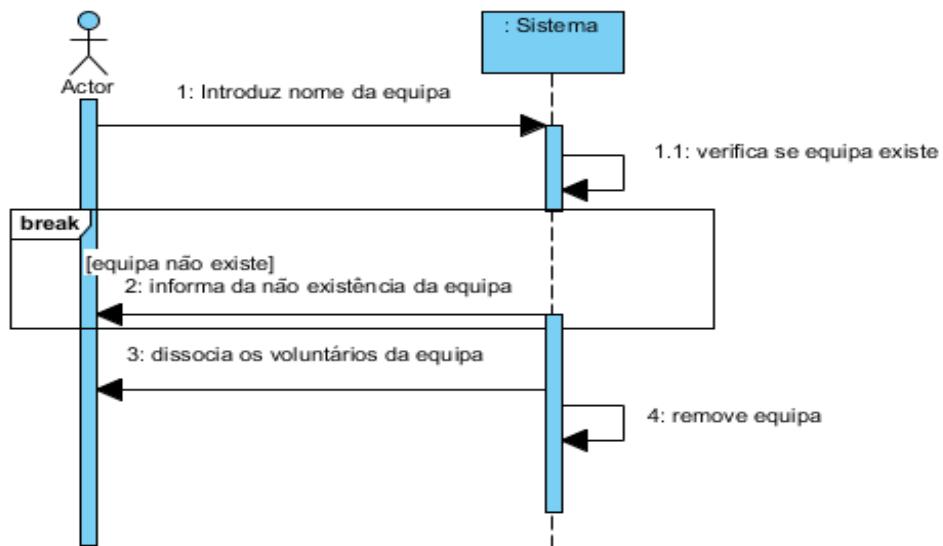


Figura 56: Diagrama de sequência Editar a Equipa

Figura 57: Diagrama de sequência **Remover a Equipa**

4 Diagramas de Classes

4.1 Um passo em frente na modelização

Nesta secção vamos apresentar todo o processo de criação do **diagrama de classes** passando pelas diferentes versões do mesmo, até que atingimos a versão que realmente foi implementada.

A partir do **modelo de domínio** onde identificamos as principais entidades (potenciais classes) do nosso problema vamos dar um passo em frente na modelização e implementar diagramas de classes.

4.2 Diagrama de Classes V1

Nesta primeira versão do diagrama foi essencialmente feita uma **filtragem** do modelo de domínio de onde retirámos as classes. Podemos já observar ligações de agregação das classes e as mesmas *recheadas* com as variáveis de instância.

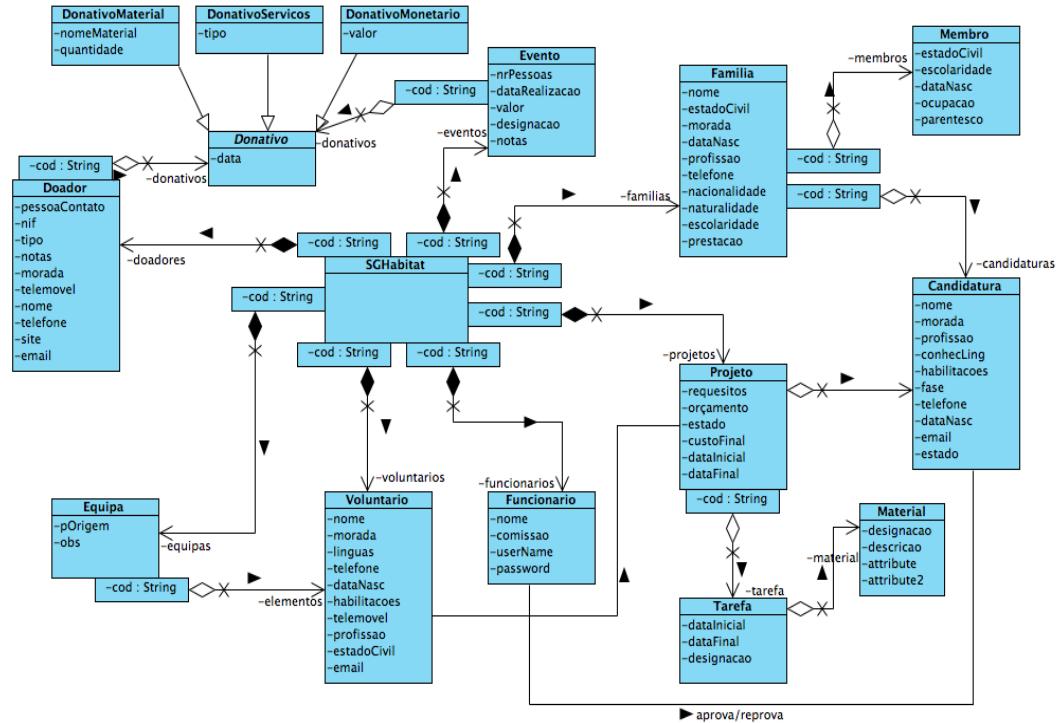


Figura 58: Diagrama de classes V1

4.3 Diagrama de Classes V2

Numa segunda versão fizemos o refinamento de cada classe e estudamos com mais detalhe as relações entre as mesmas, alterando conforme necessário a maneira como os *Mapas* são implementados.

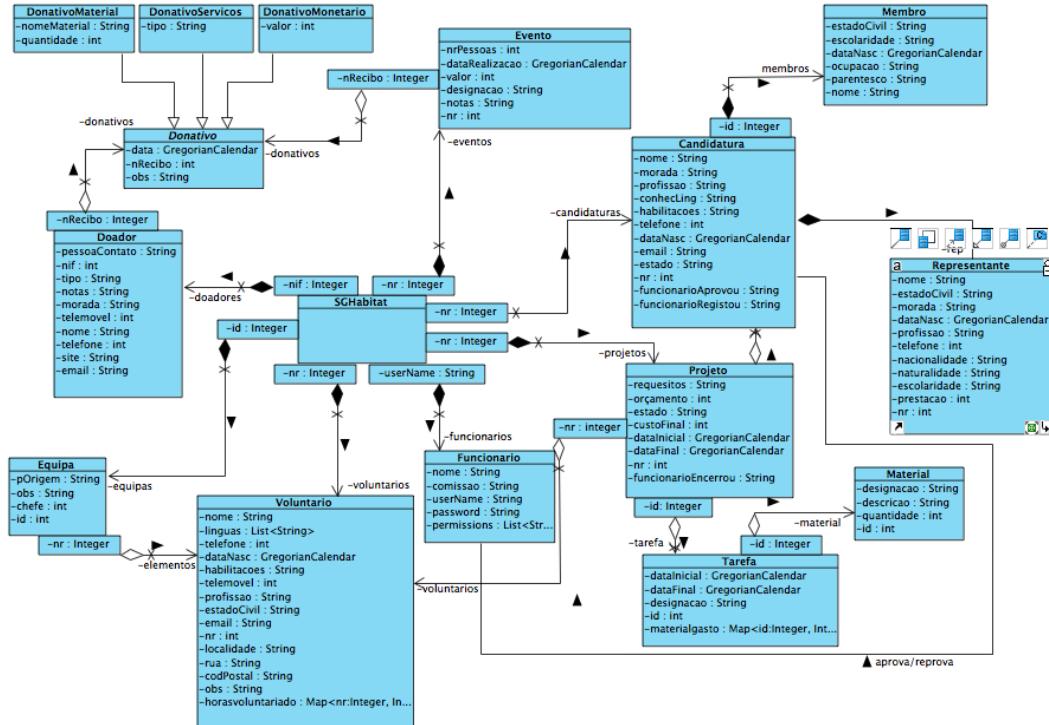


Figura 59: Diagrama de classes V2

4.4 Diagrama de Classes V3

Esta é a versão final do diagrama de classes onde substituimos os *Mapas* que se relacionavam com o facade por **DAO's** (*Data Access Objects*), acrescentamos também mais detalhe às classes inserindo os métodos que cada uma disponibiliza.

Os **DAO's** têm todos implementação semelhante por se resumirem a **inserções**, **atualizações**, **remoções** e **consultas** por isso o grupo decidiu não incluir nos diagramas os métodos implementados, também para melhor legibilidade do diagrama final.

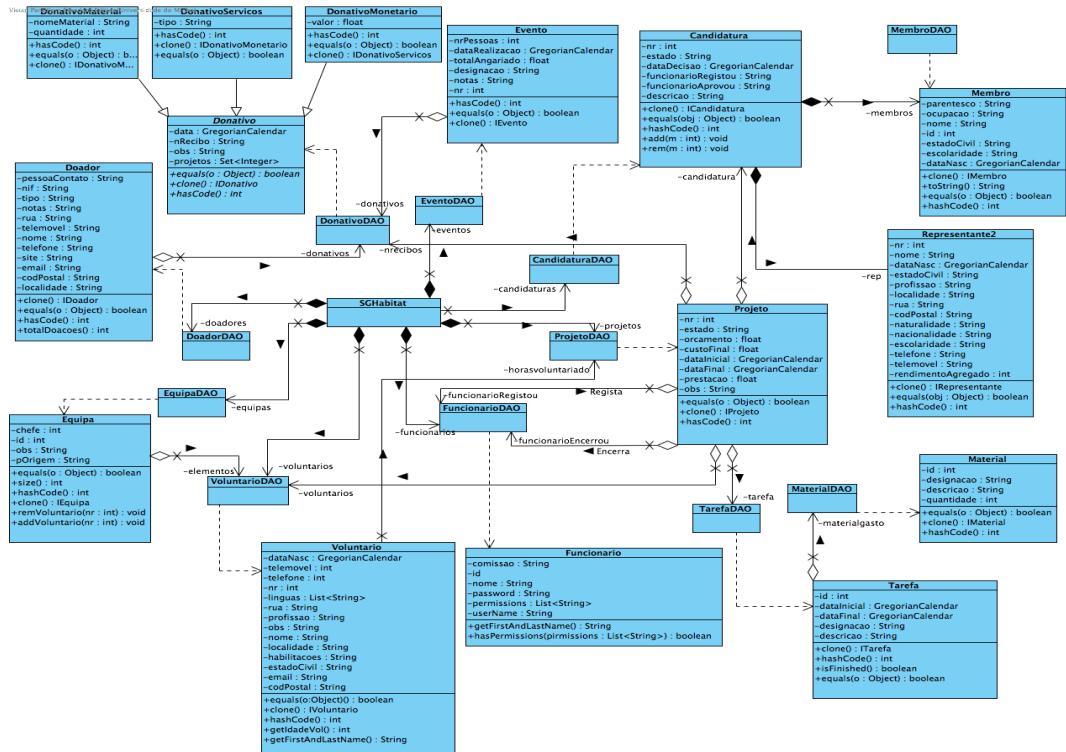


Figura 60: Diagrama de classes V3

5 Diagrama de packages

Agora que conhecemos as classes do nosso projeto e estudamos o histórico da sua implementação, vamos demonstrar como foram organizadas essas classes e com que fundamento.

5.1 Separação lógica

Quando o grupo de trabalho pensou sobre como iríamos distribuir as diversas classes por packages de uma forma lógica e estruturada, reparámos com base na análise de requisitos até então que as classes se incluíam nas seguintes principais áreas:

- **Doações:** Donativos, doadores e eventos, tudo está relacionado com a comissão de angariação de fundos.
- **Famílias:** Candidaturas e famílias, onde são geridos registos de candidaturas e de membros dos agregados familiares.
- **Projetos:** Classes relativas a obras como o projeto, material e tarefa do projeto.
- **Recursos Humanos:** Onde são geridos os voluntários e funcionários da Habitat.

Esta separação permitiu-nos uma melhor organização e distribuição de trabalho, é também uma boa forma de separar logicamente as classes, pelo que os quatro packages da camada de negócio são esses mesmos, como podemos ver na seguinte figura.

5.1.1 Diagrama

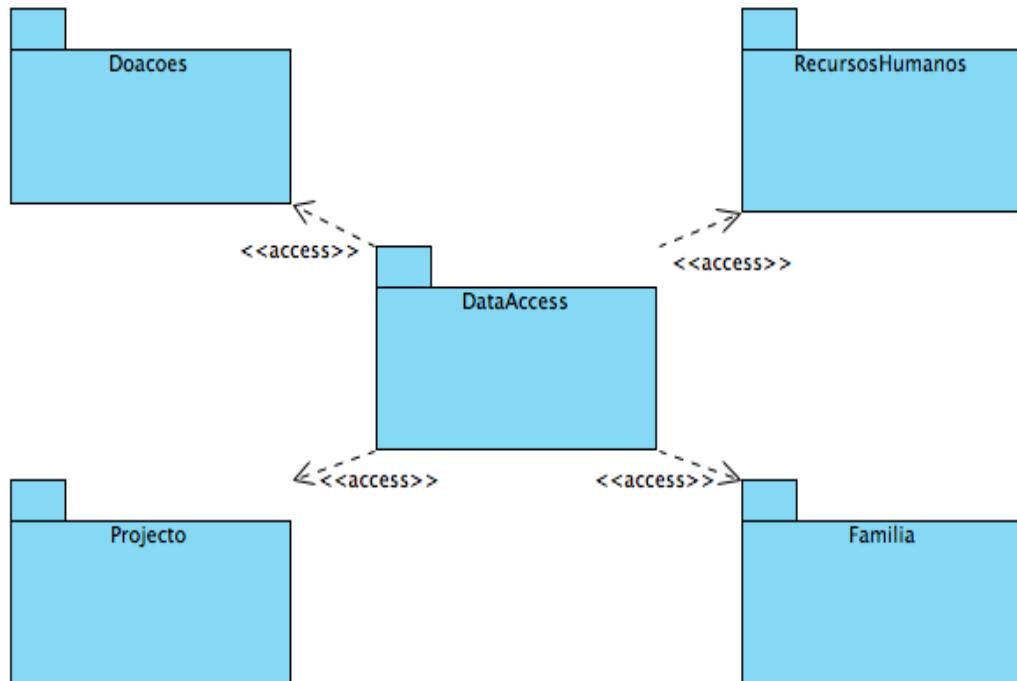


Figura 61: Diagrama de packages genérico.

Como podemos ver o quinto package é o package **DataAccess** onde implemámos os DAO's e o *facade SGHabitat*.

De seguida apresentamos com mais detalhe cada um dos **subsistemas** mencionados.

5.2 Package - Doacoes

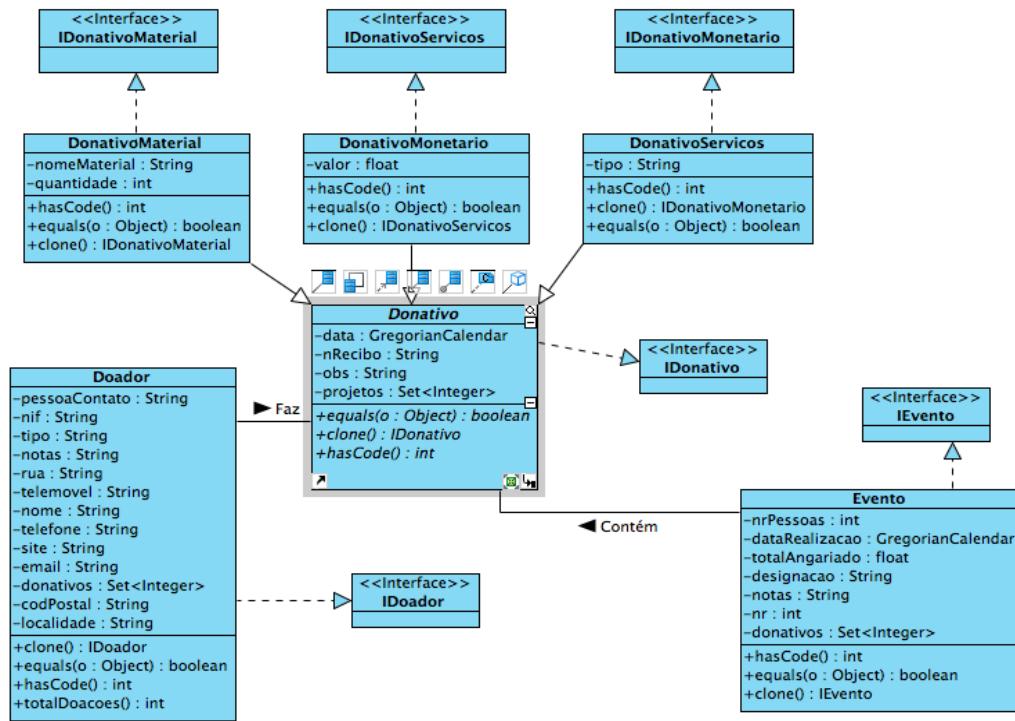


Figura 62: Subsistema Doacoes.

5.3 Package - Familias

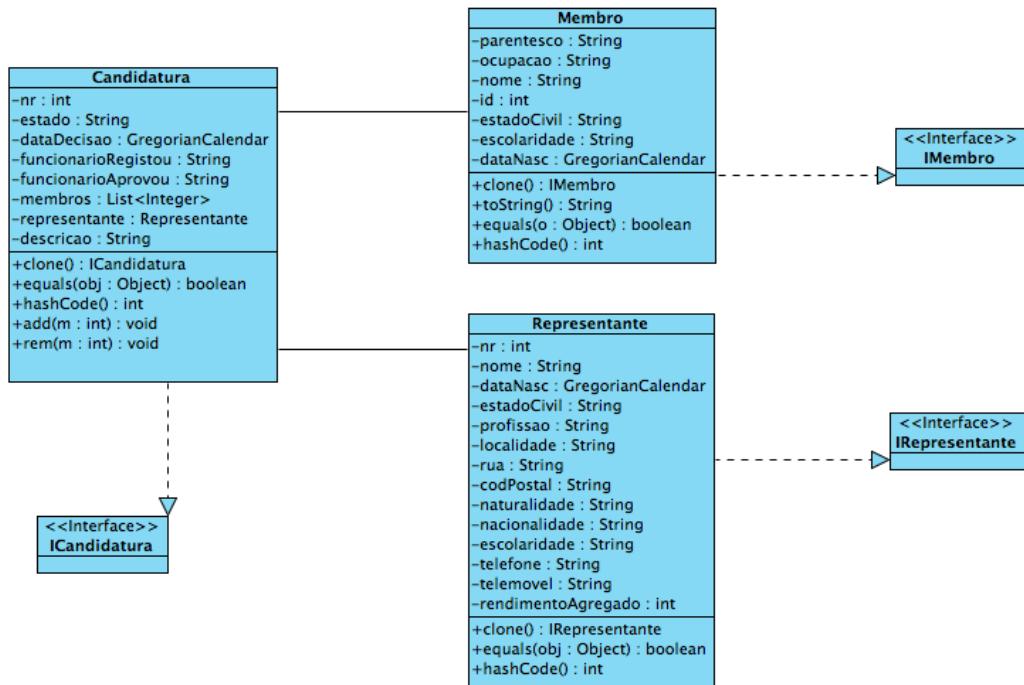
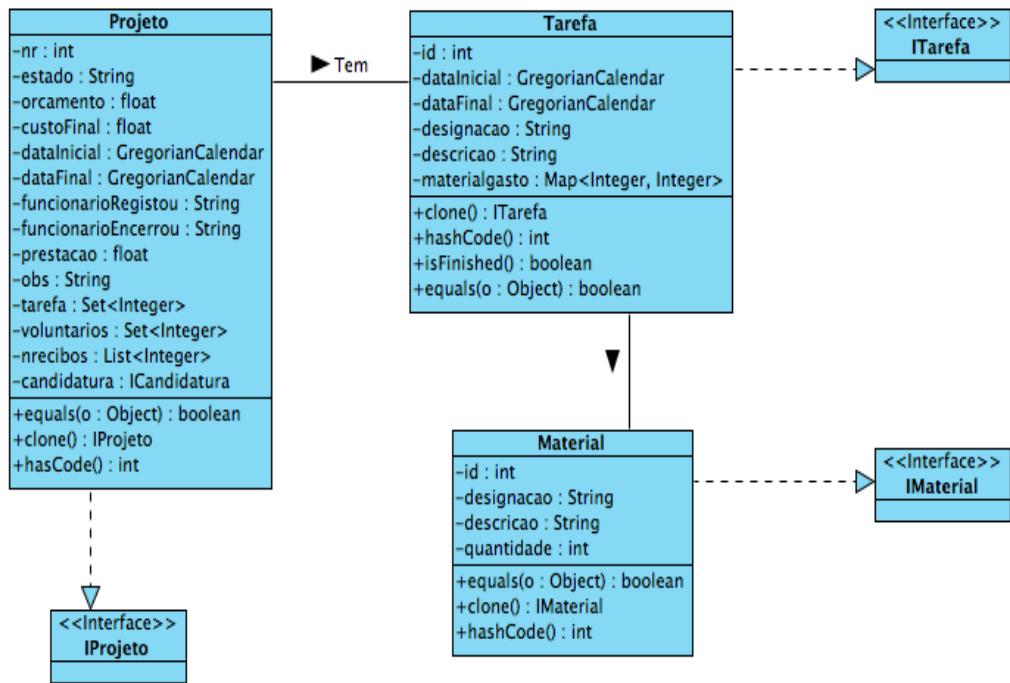


Figura 63: Subsistema **Familias**.

5.4 Package - Projetos

Figura 64: Subsistema **Projetos**.

5.5 Package - Recursos Humanos

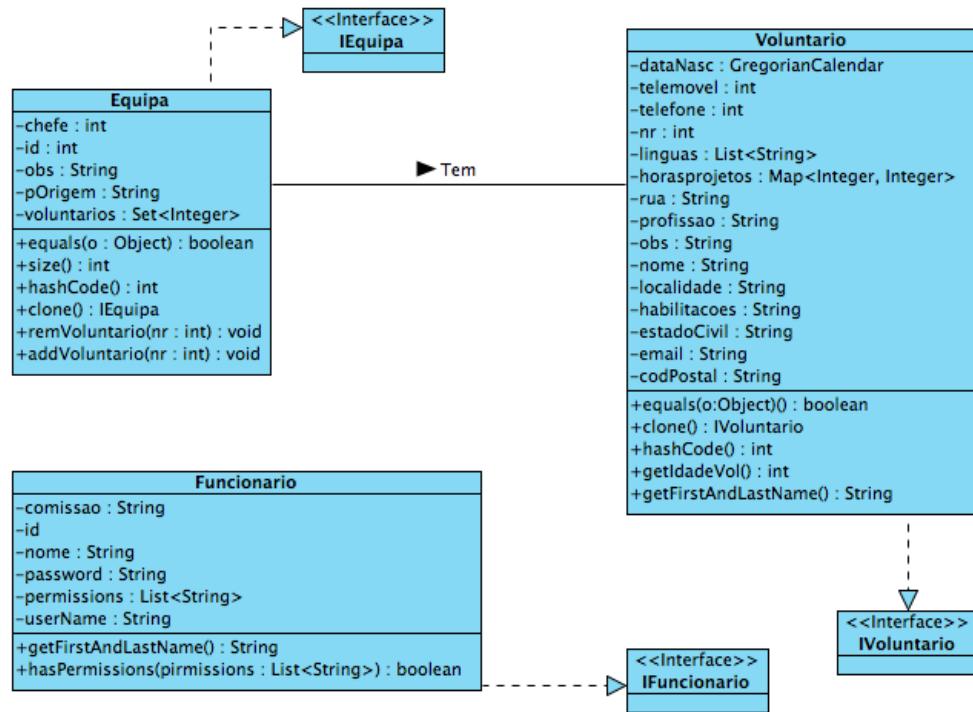


Figura 65: Subsistema Recursos Humanos.

5.6 Package - Data Access

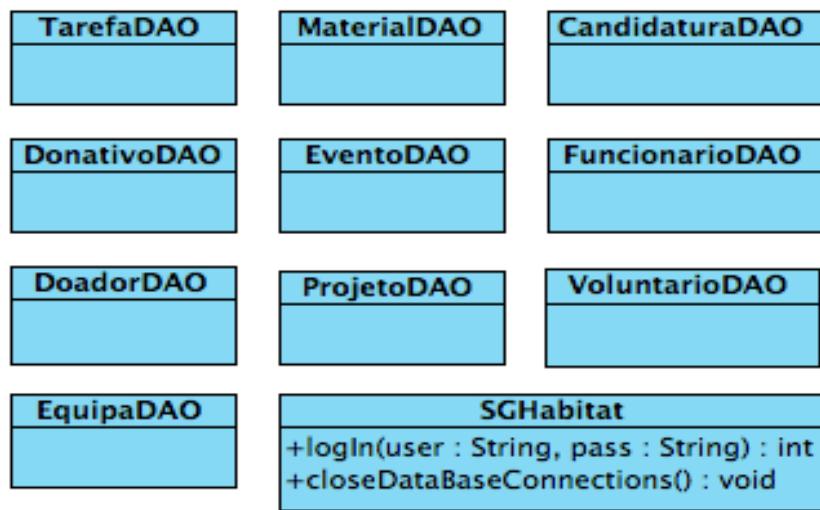


Figura 66: Subsistema **Data Access**.

6 Diagramas de Seq. de Subsistemas

Após identificarmos os **subsistemas** através do **diagrama de packages** vamos para cada *use case* definir o seu diagrama de subsistemas, *desdobrando* os diagramas de sistemas nos vários subsistemas, onde iremos obter mais detalhe sobre as interações da futura implementação.

Portanto seguem-se os diagramas de subsistemas pela mesma ordem que apresentamos os diagramas de sistemas.

6.1 Autenticação

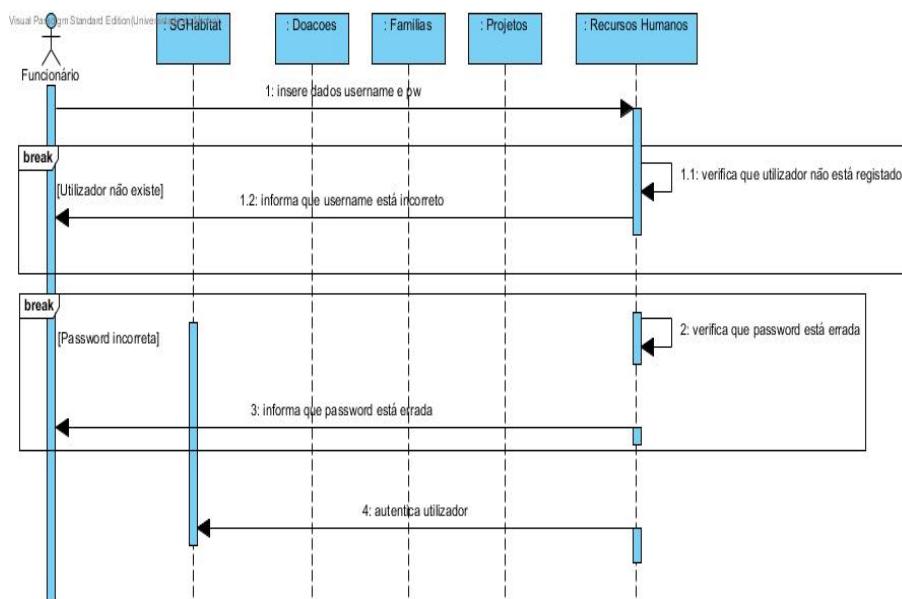


Figura 67: Diagrama de sequência de subsistemas de **Autenticação**.

6.2 Doações

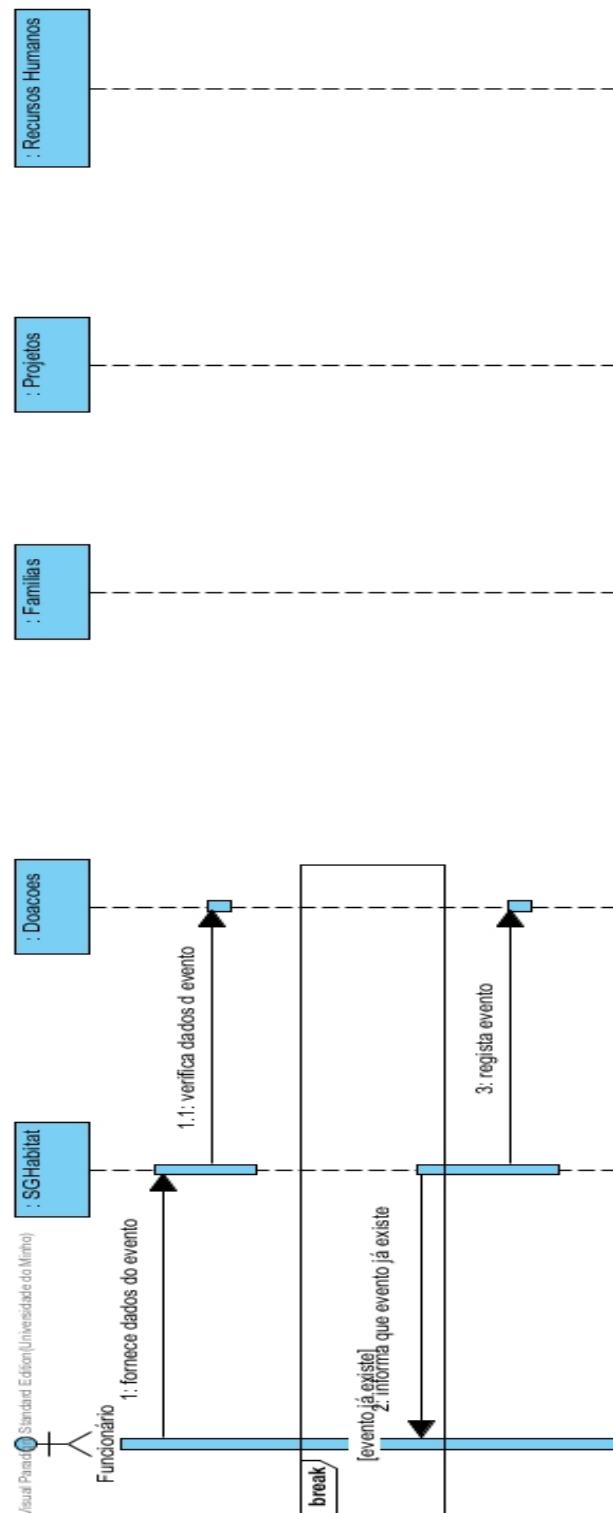


Figura 68: Diagrama de sequência de subsistemas de Registar Evento.

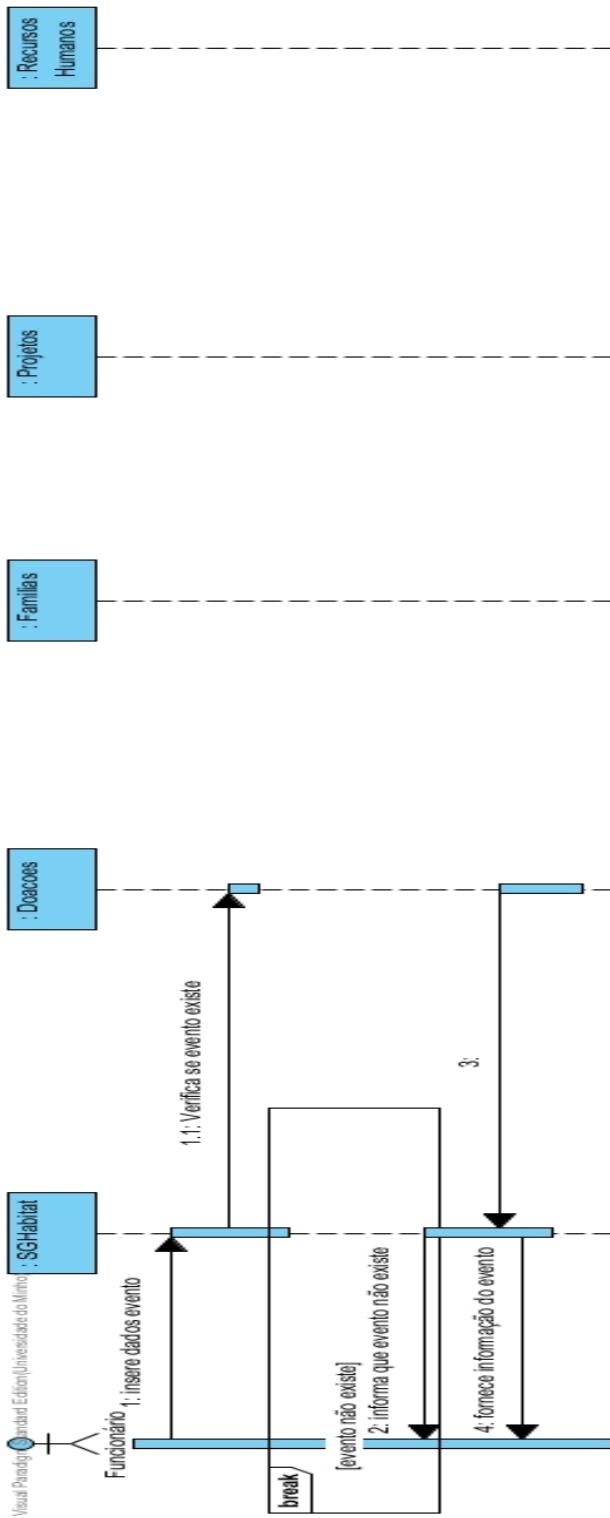


Figura 69: Diagrama de sequência de subsistemas de Consulta Evento.

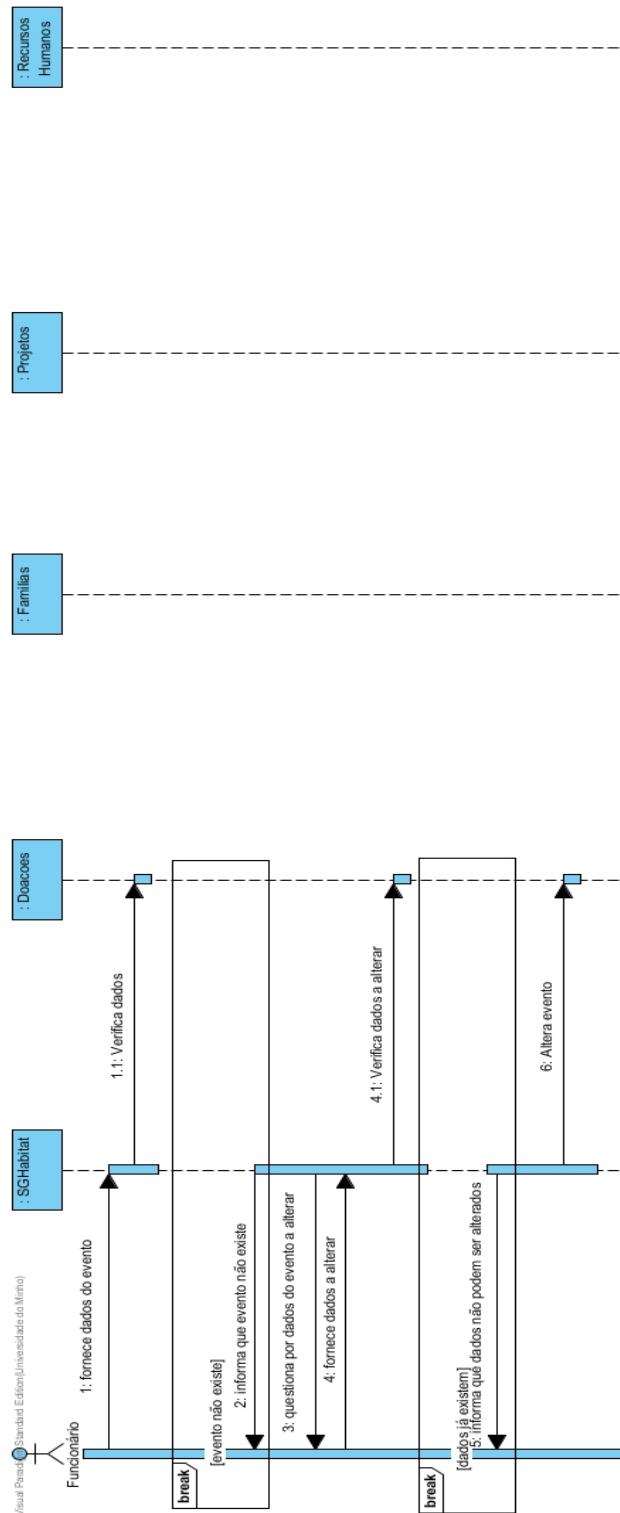


Figura 70: Diagrama de sequência de subsistemas de Edita Evento.

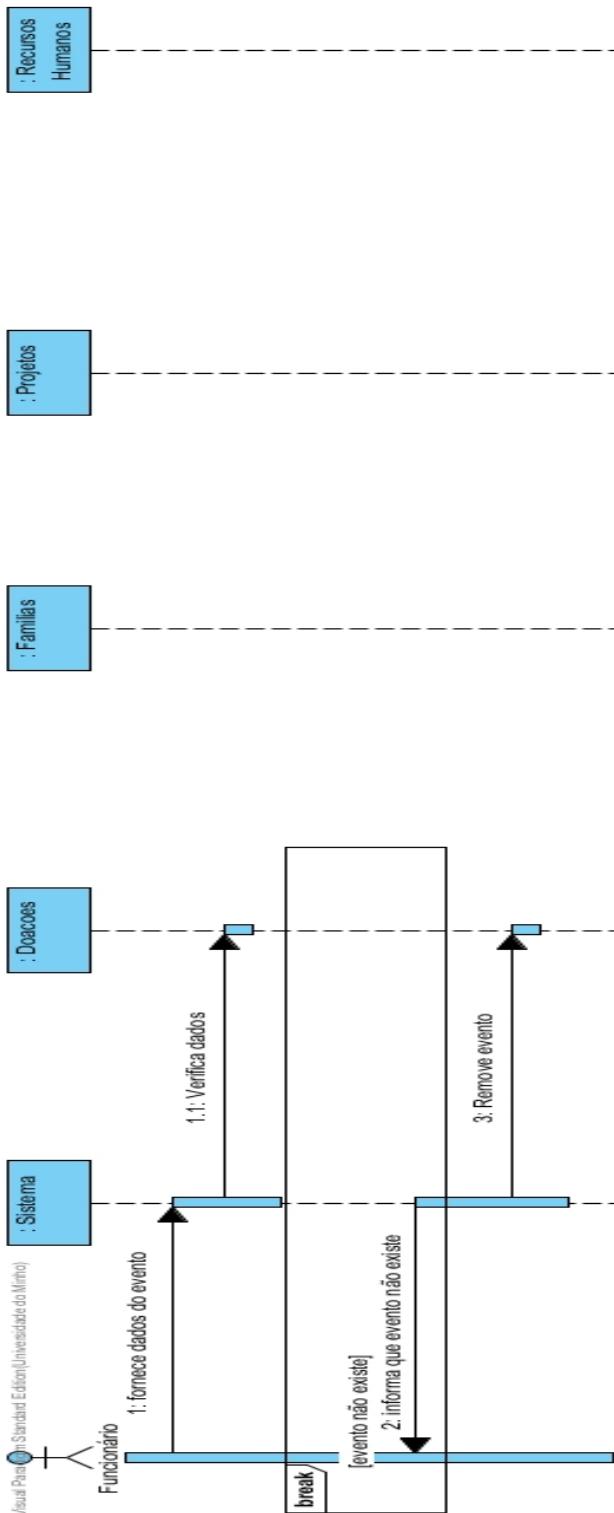


Figura 71: Diagrama de sequência de subsistemas de Remover Evento.

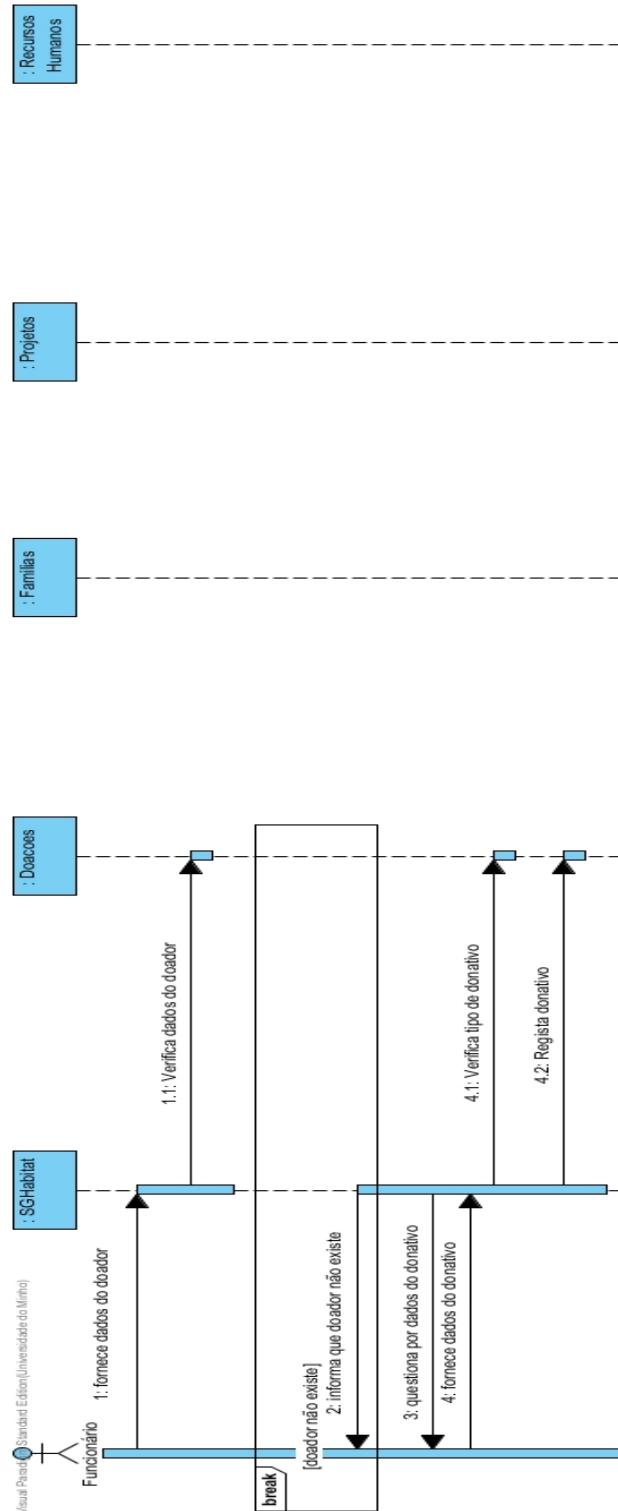


Figura 72: Diagrama de seqüência de subsistemas de Regista Donativo.

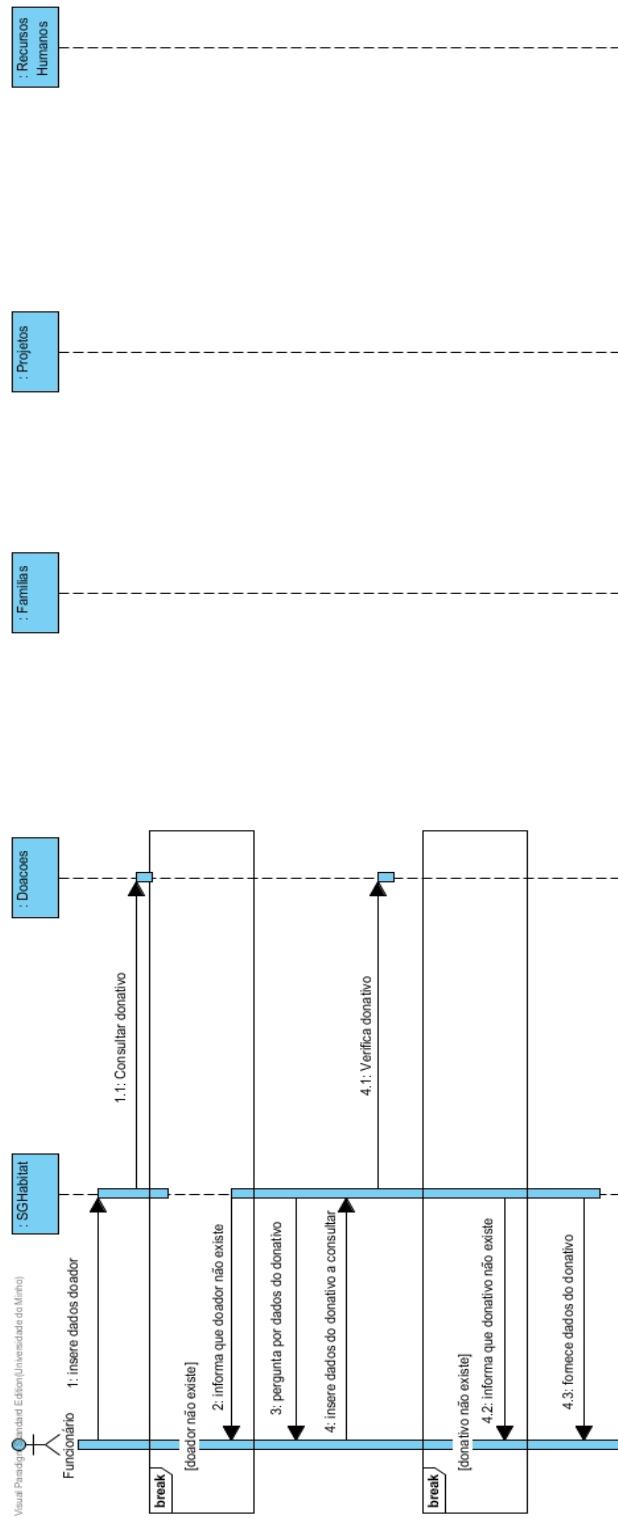


Figura 73: Diagrama de sequência de subsistemas de Consulta Donativo.

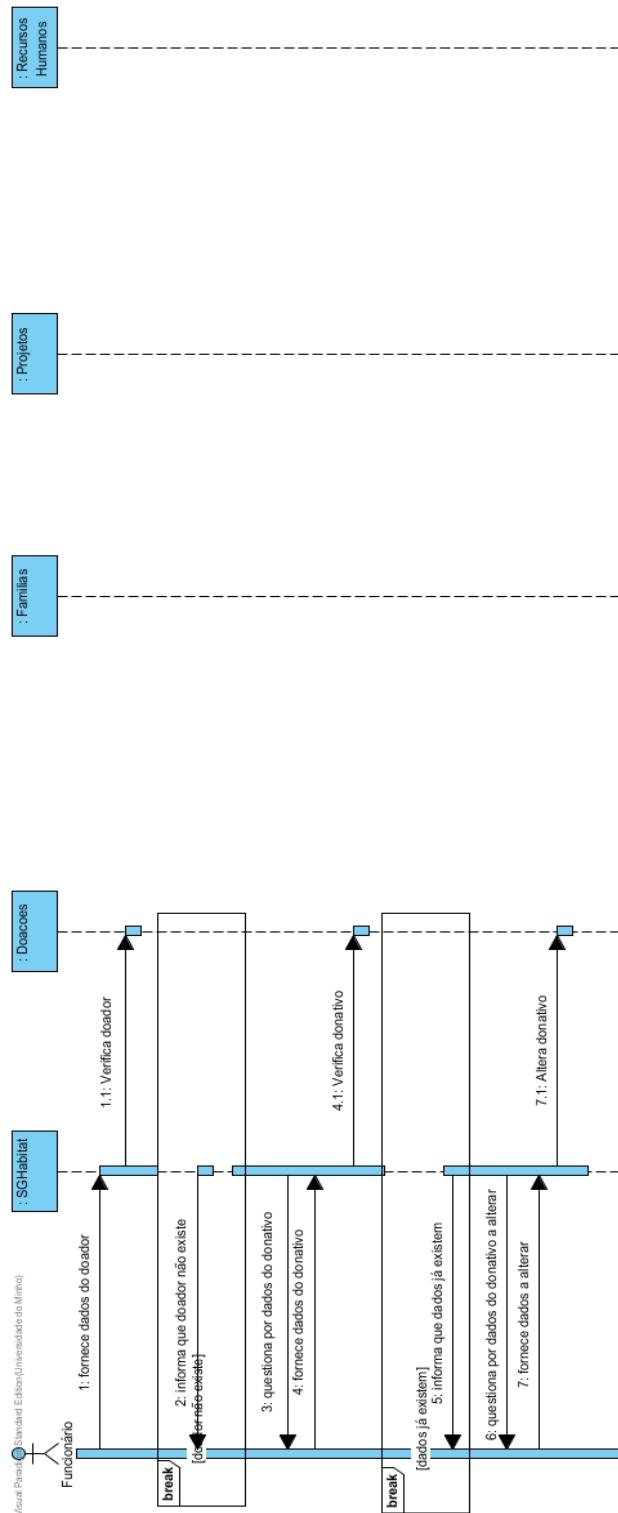


Figura 74: Diagrama de sequência de subsistemas de Editar Donativo.

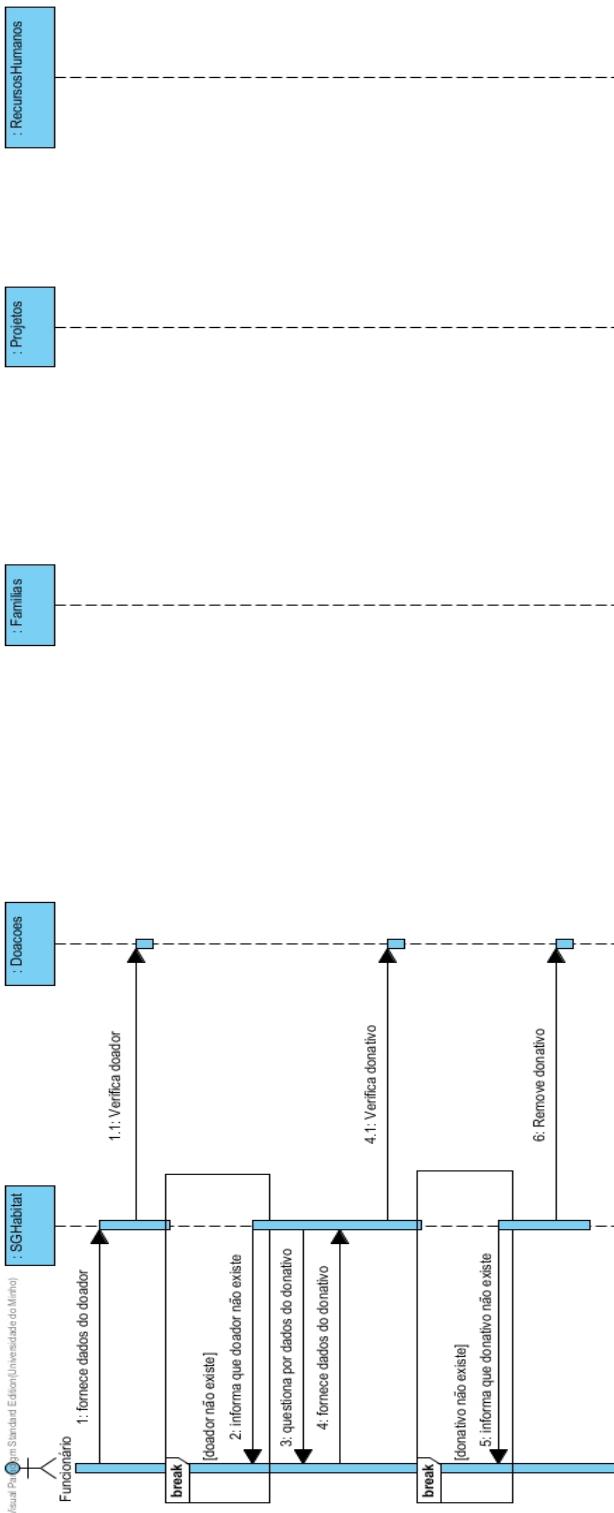


Figura 75: Diagrama de sequência de subsistemas de Remove Domativo.

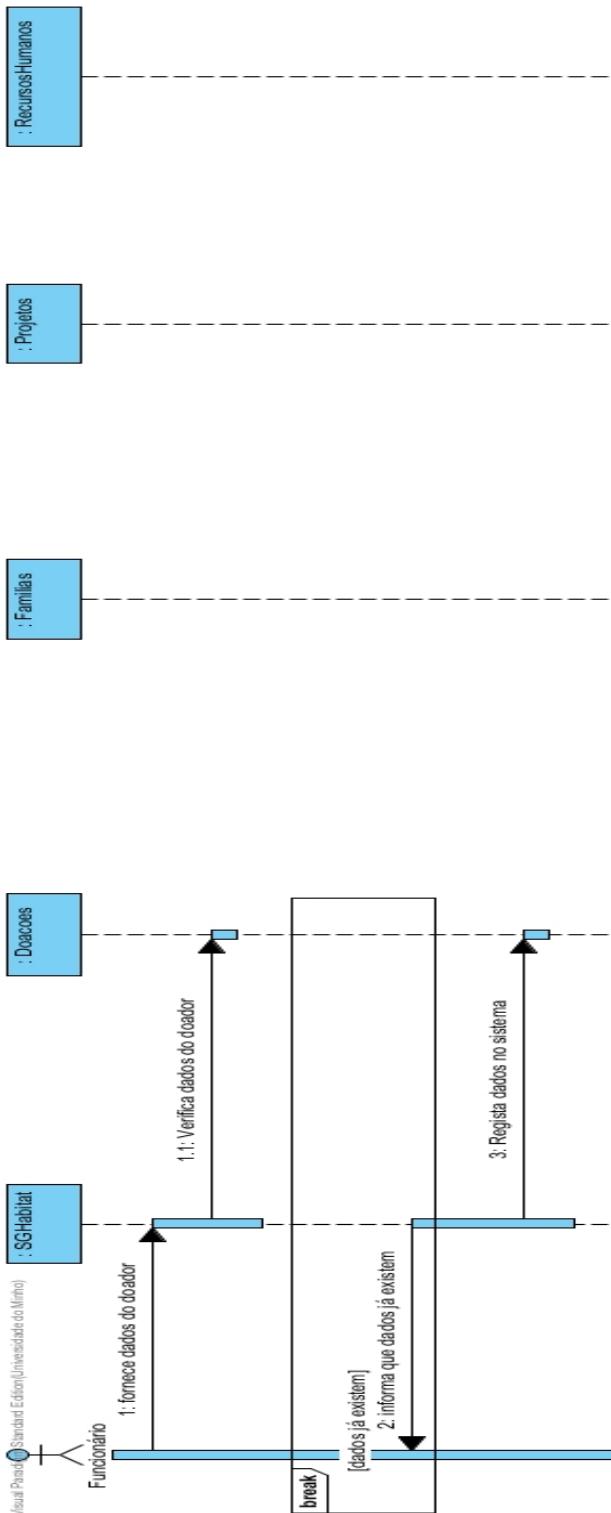


Figura 76: Diagrama de sequência de subsistemas de Regista Doador.

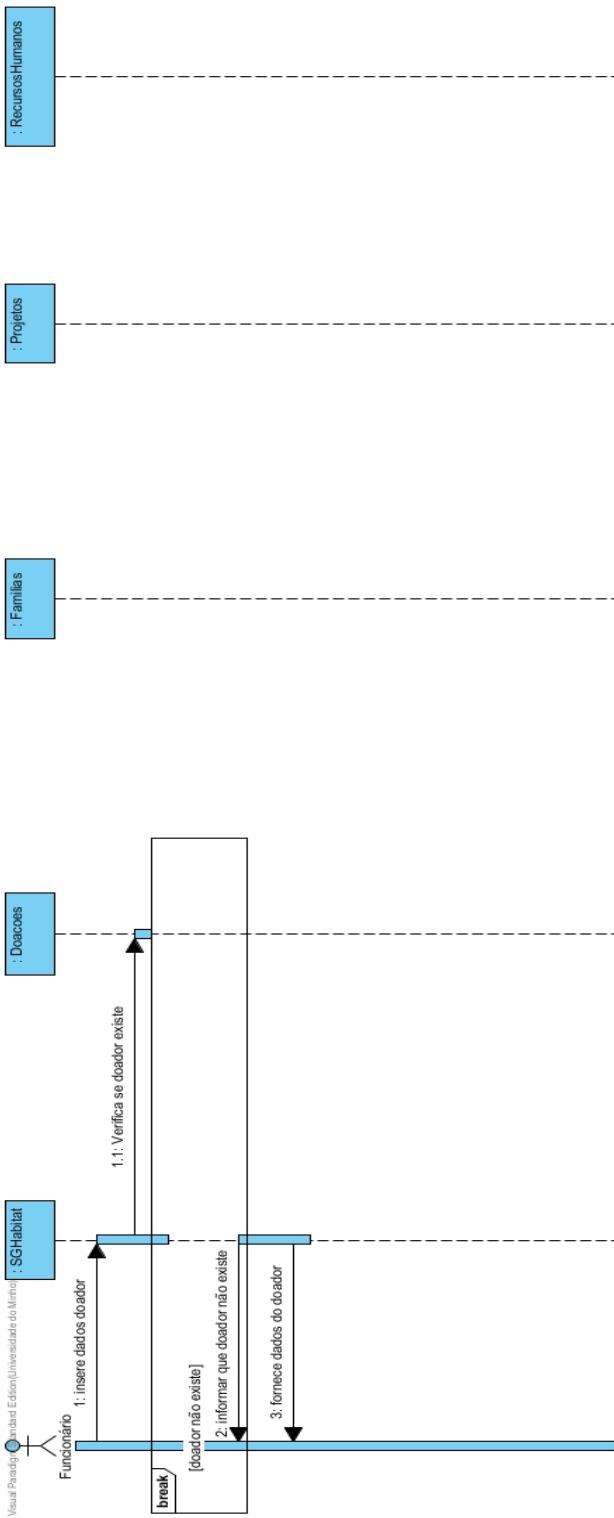


Figura 77: Diagrama de seqüência de subsistemas de Consulta Donativo.

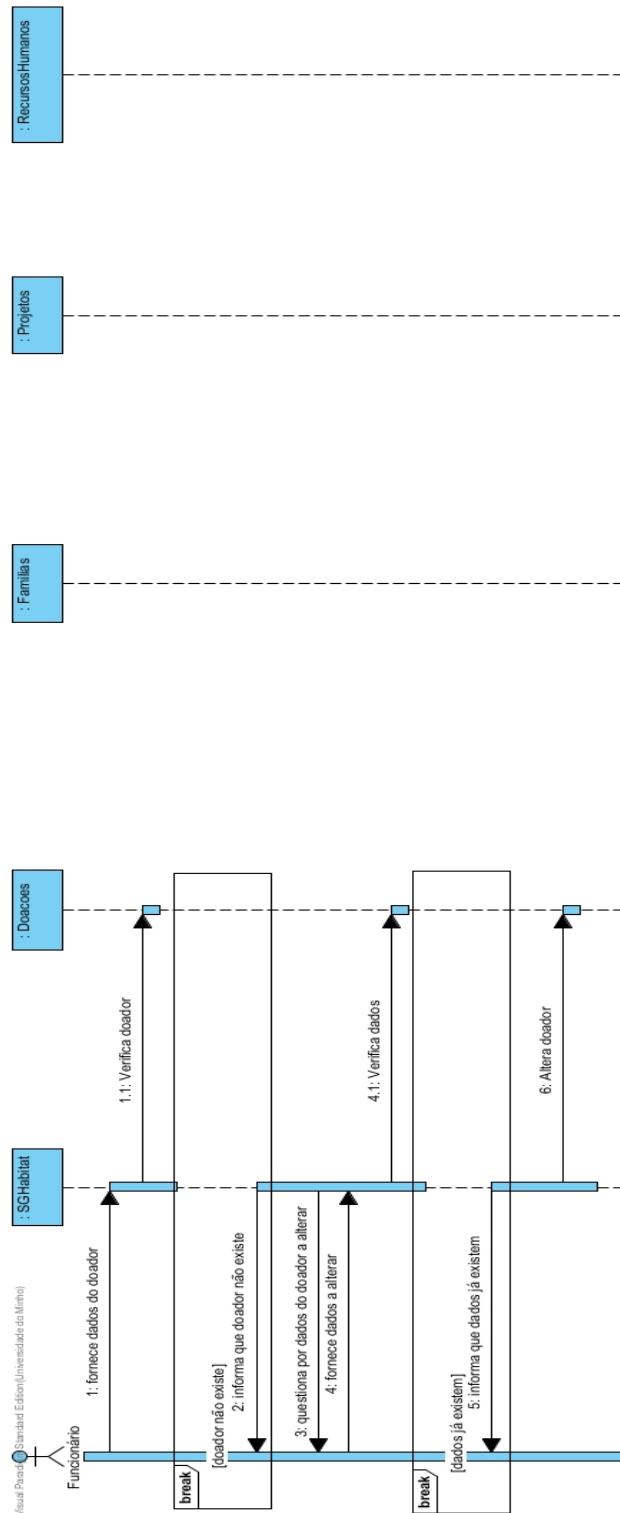


Figura 78: Diagrama de sequência de subsistemas de Editar Doador.

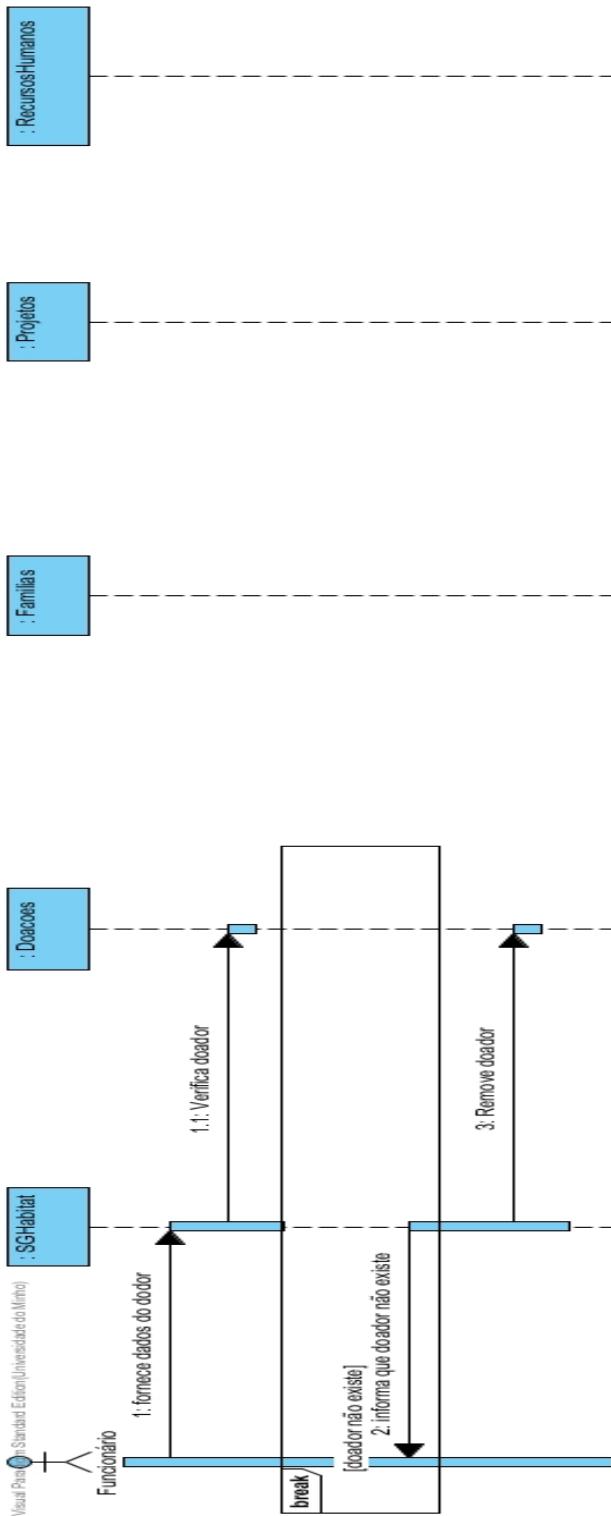


Figura 79: Diagrama de sequência de subsistemas de Remove Doador.

6.3 Famílias

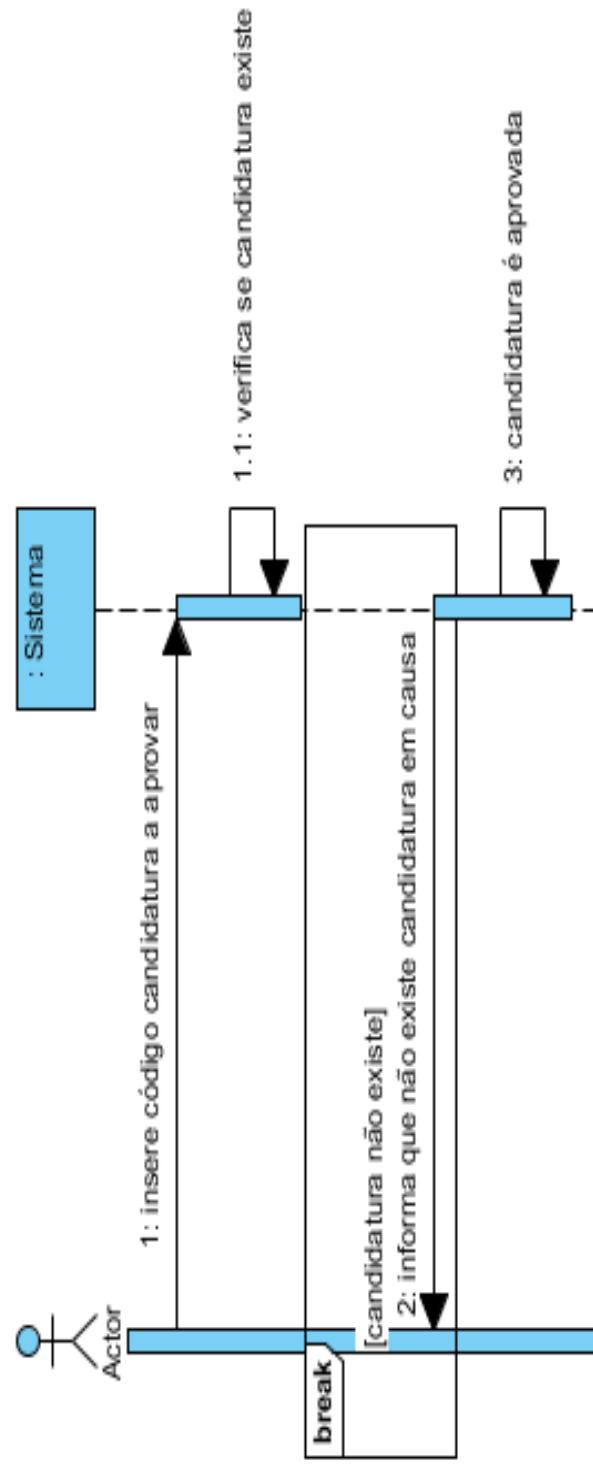


Figura 80: Diagrama de sequência de subsistemas de *Registrar Candidatura*.

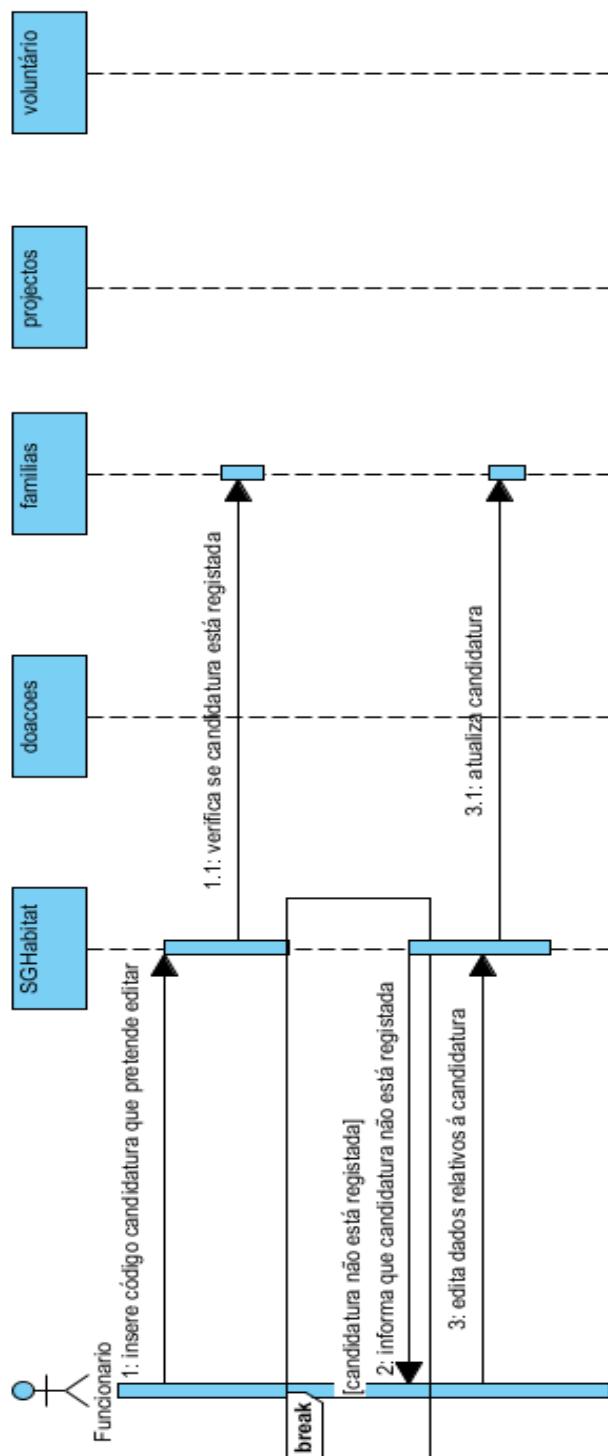


Figura 81: Diagrama de sequência de subsistemas de Editar Candidatura.

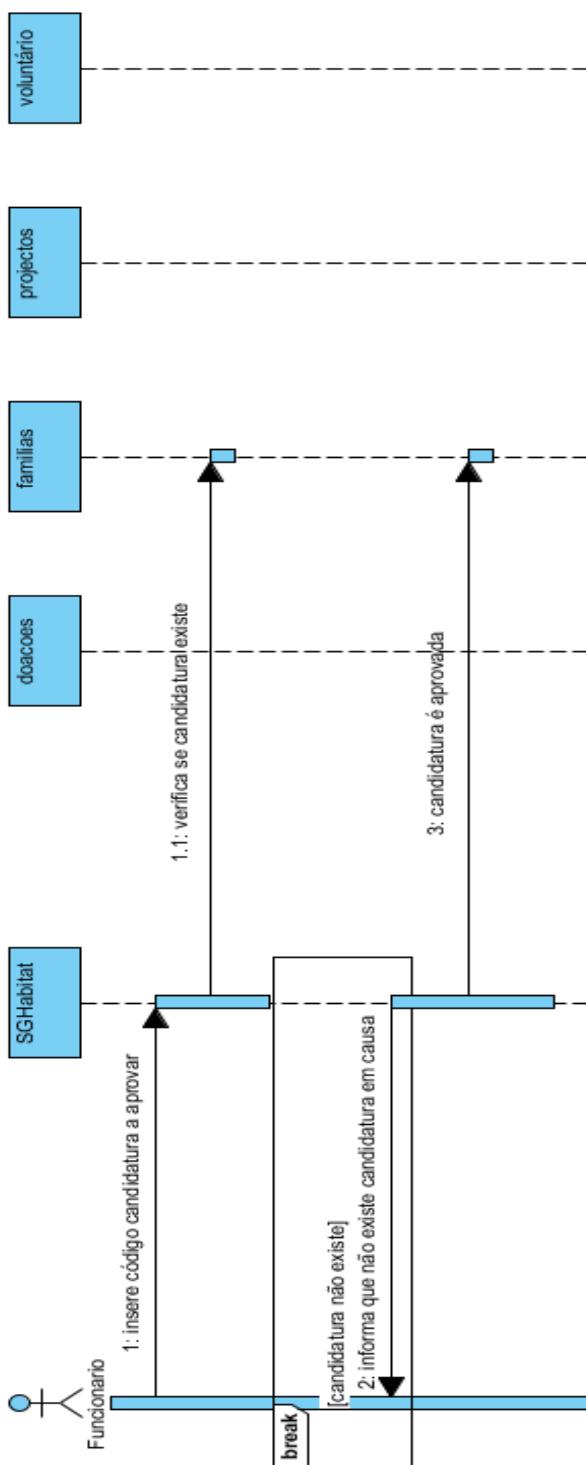


Figura 82: Diagrama de sequência de subsistemas de Aprovar Candidatura.

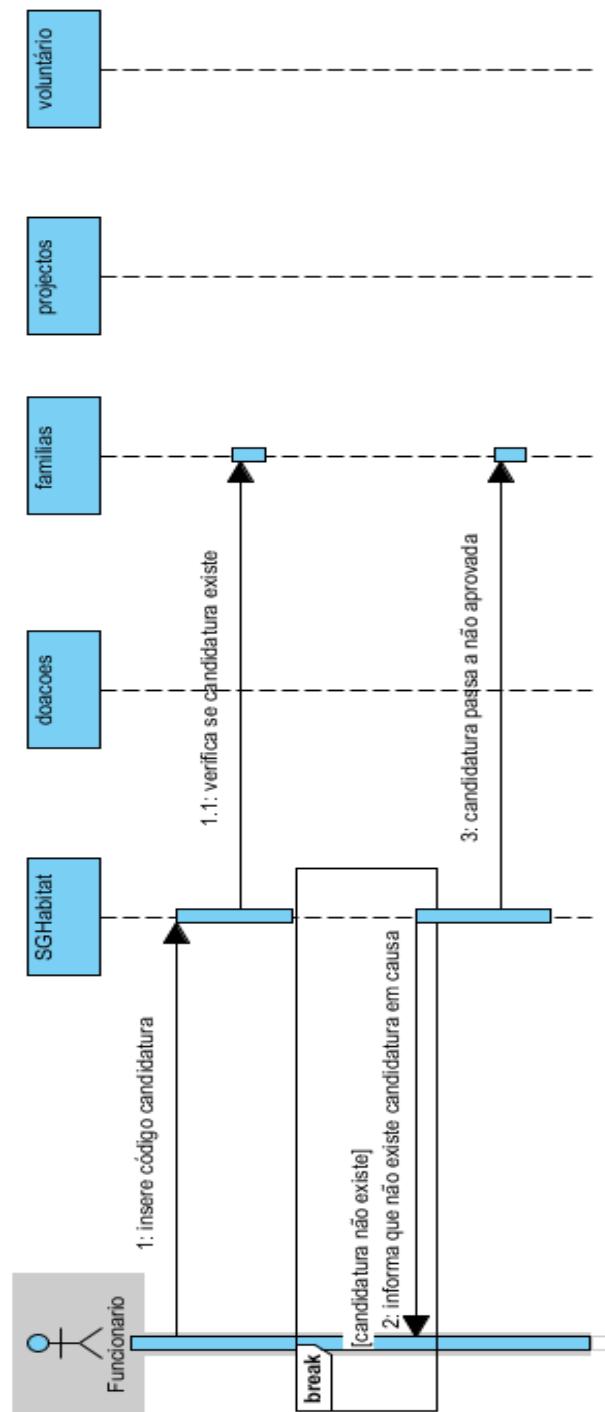


Figura 83: Diagrama de sequência de subsistemas de Rejetar Candidatura.

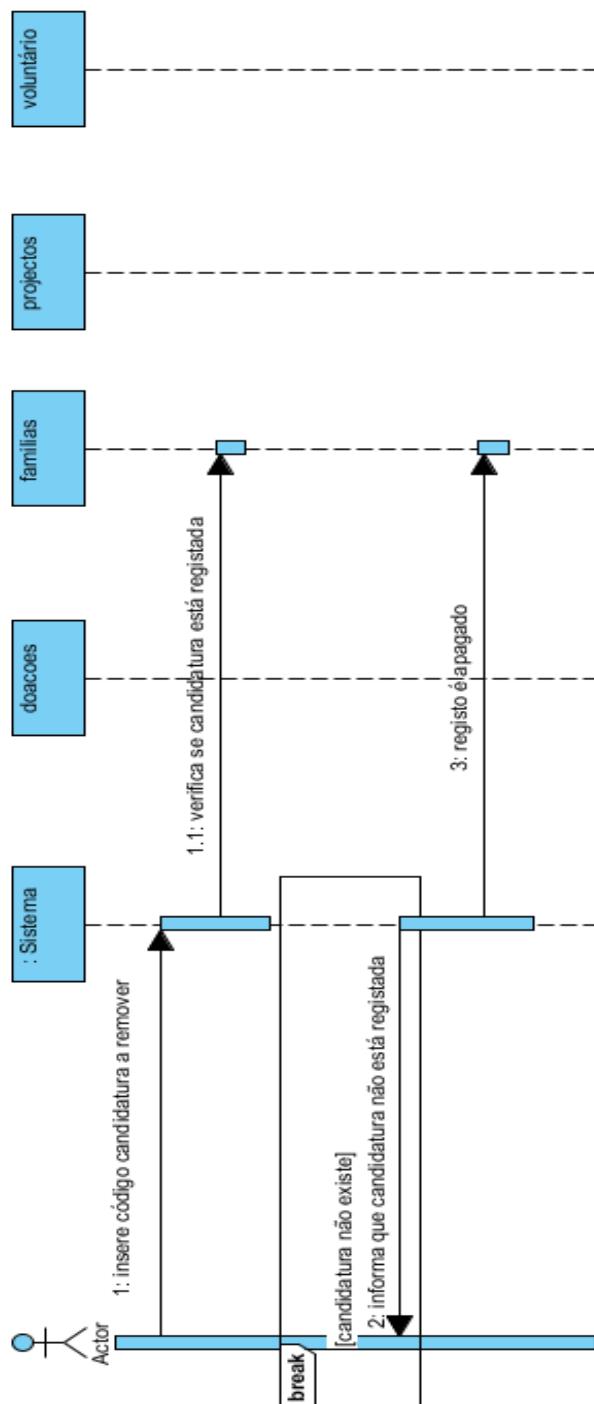


Figura 84: Diagrama de sequência de subsistemas de Remove Candidatura.

6.4 Projetos

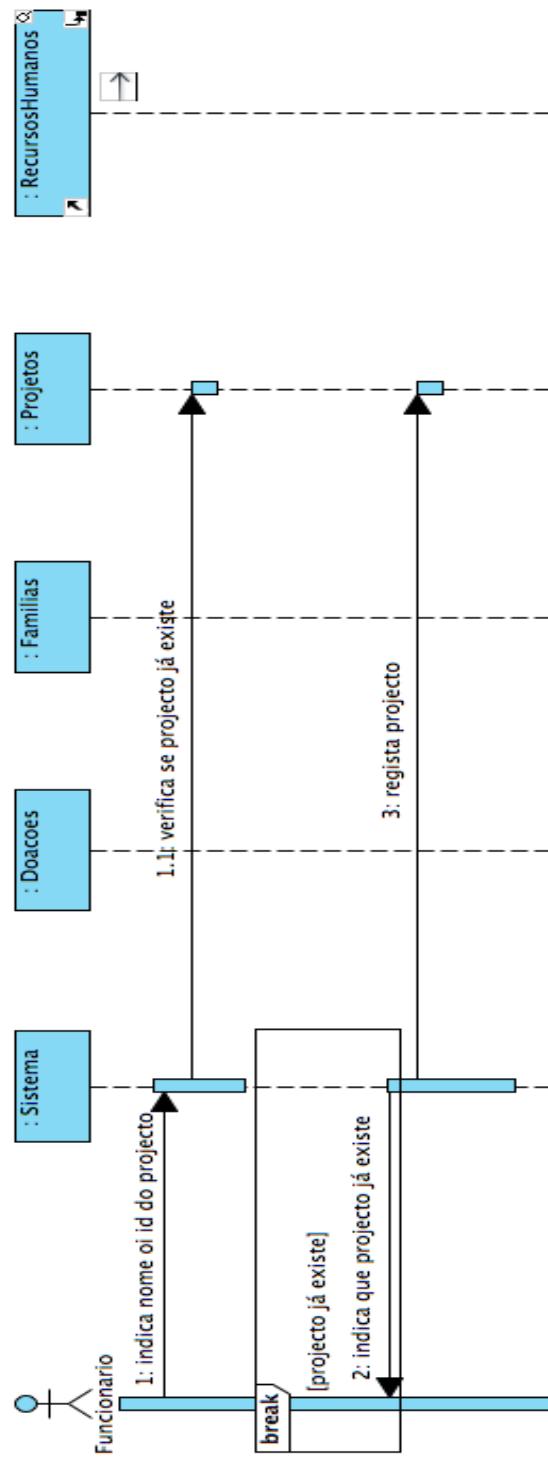


Figura 85: Diagrama de sequência de subsistemas de Regista Projeto.

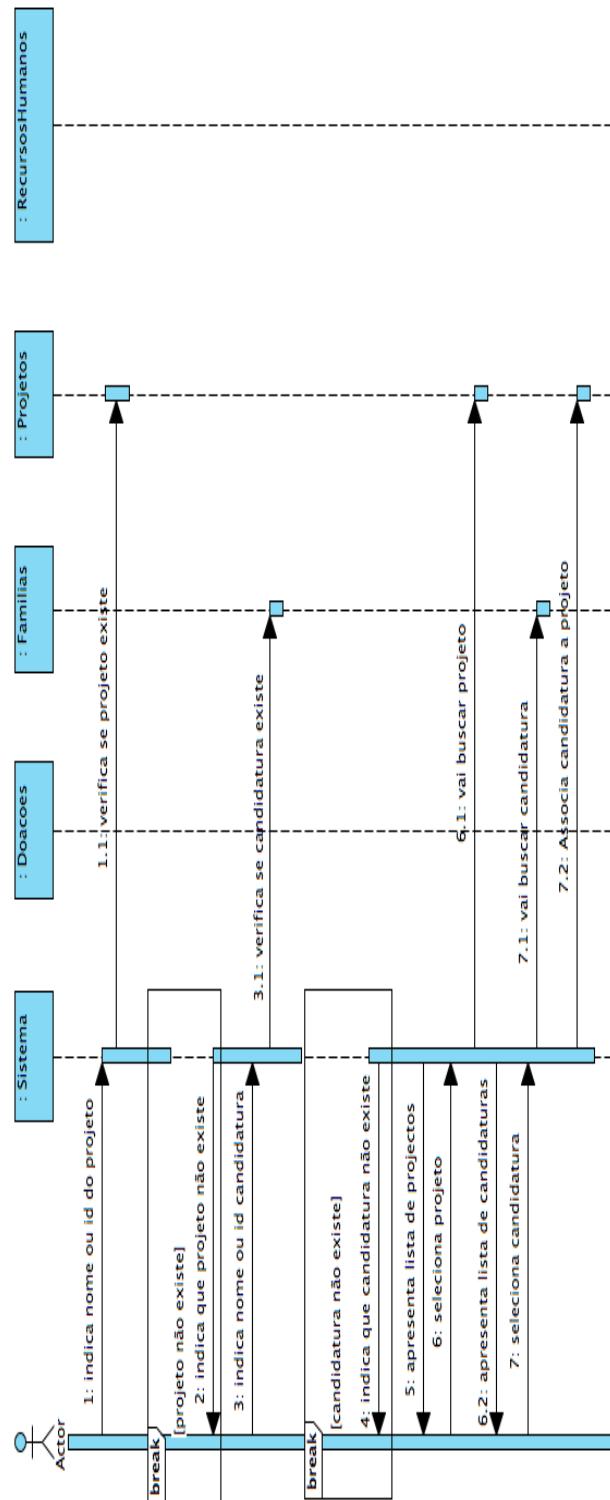


Figura 86: Diagrama de sequência de subsistemas de Associação Candidatura a Projeto.

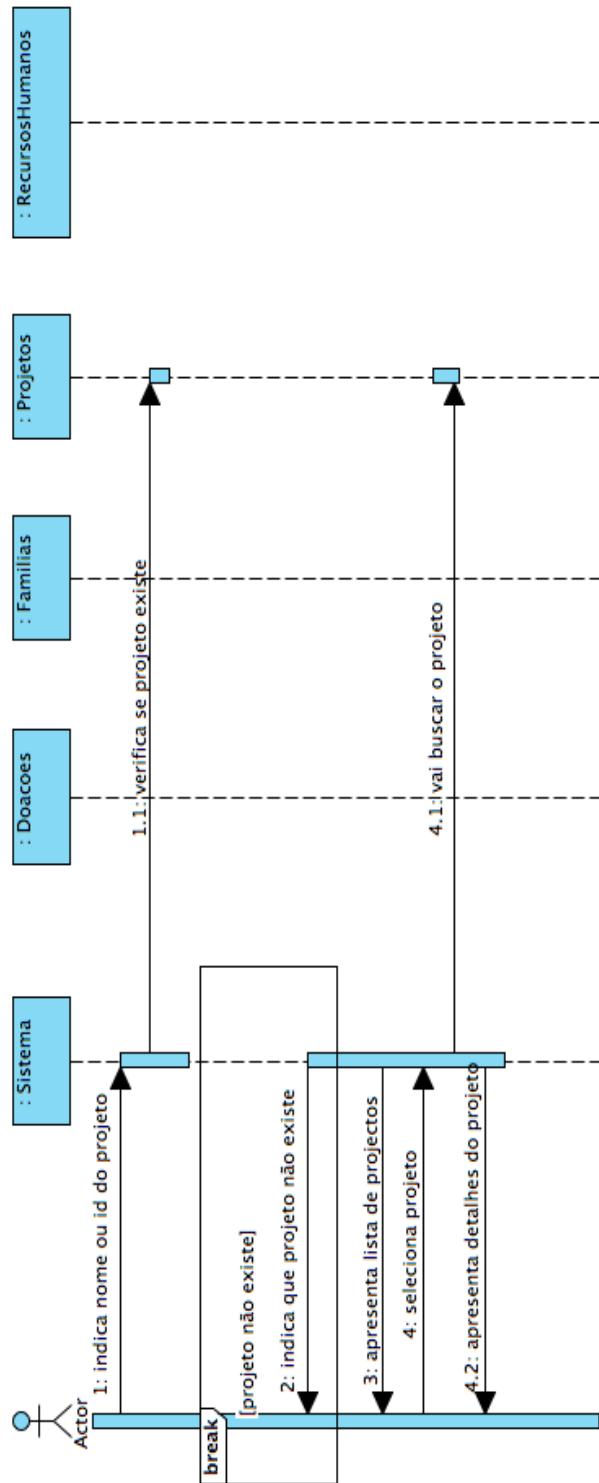


Figura 87: Diagrama de sequência de subsistemas de Consulta Projeto.

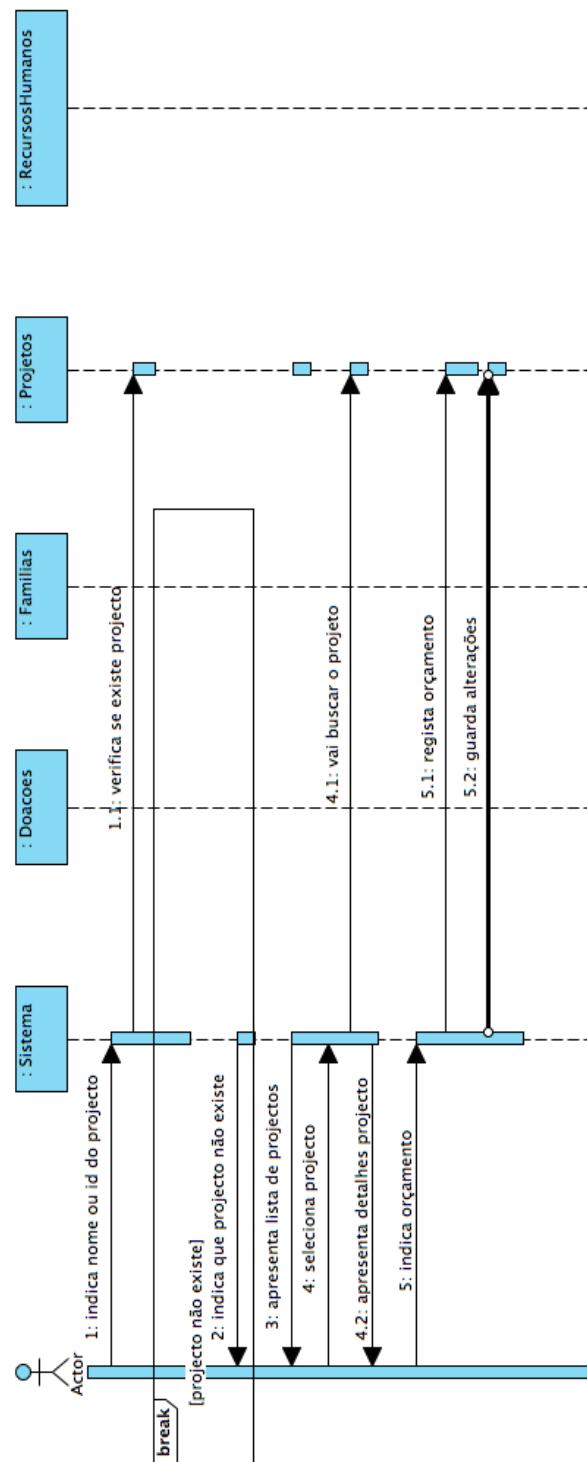


Figura 88: Diagrama de sequência de subsistemas de Regista Orçamento.

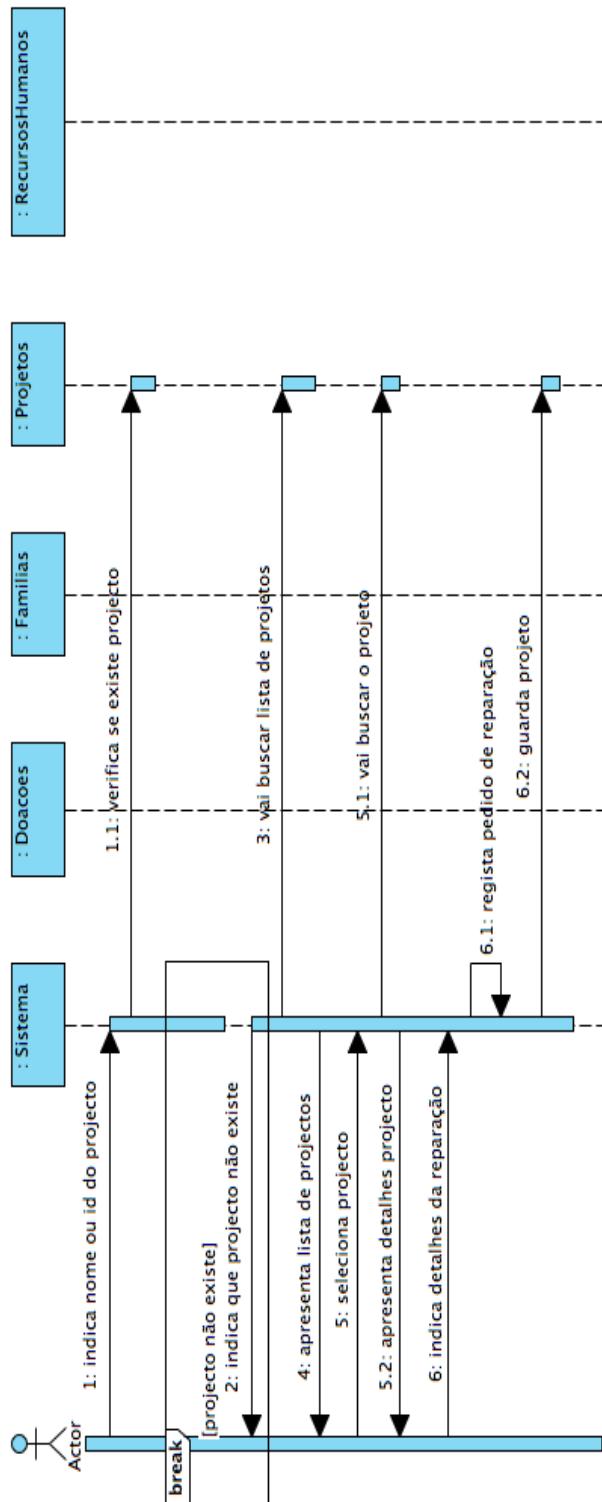


Figura 89: Diagrama de seqüência de subsistemas de Regista Reparação.

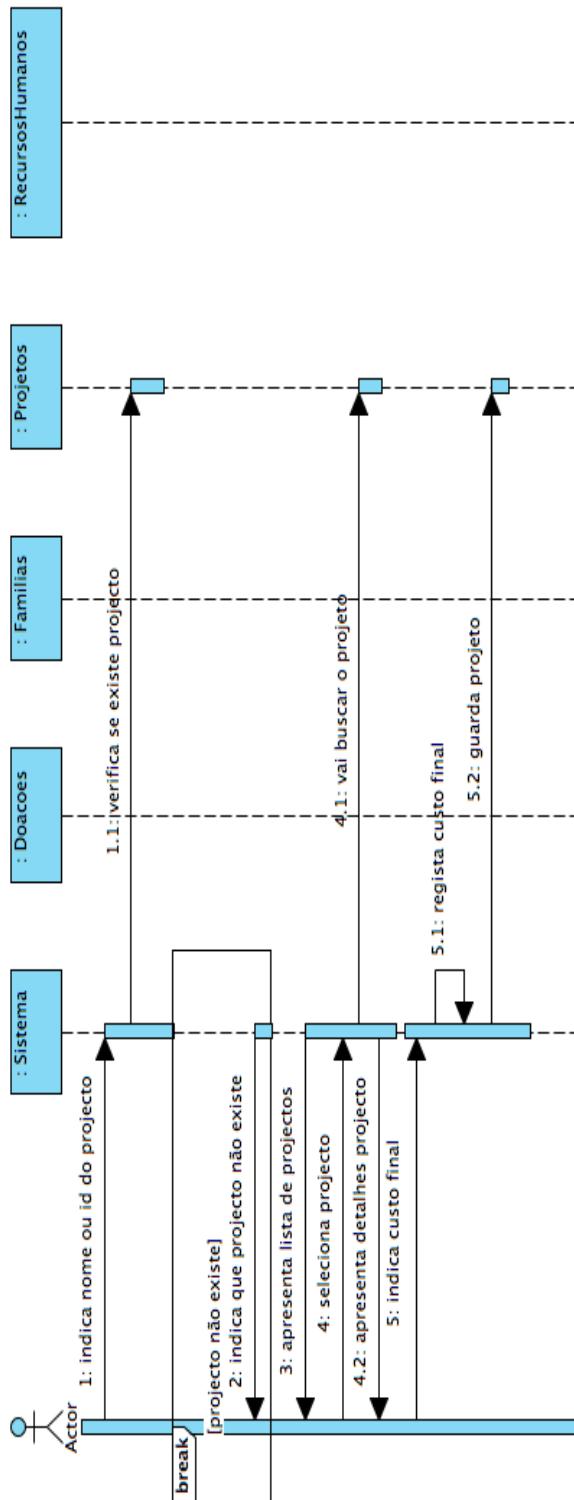


Figura 90: Diagrama de sequência de subsistemas de Regista Custo Final.

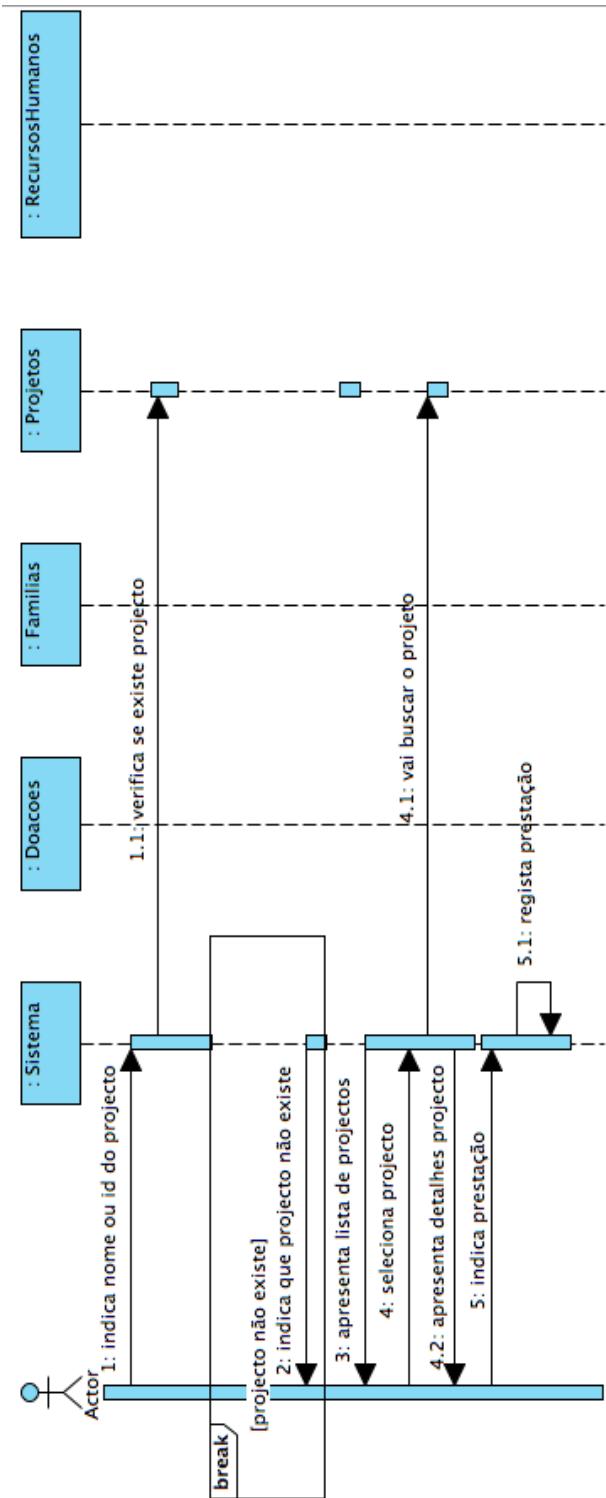


Figura 91: Diagrama de sequência de subsistemas de Regista Prestação.

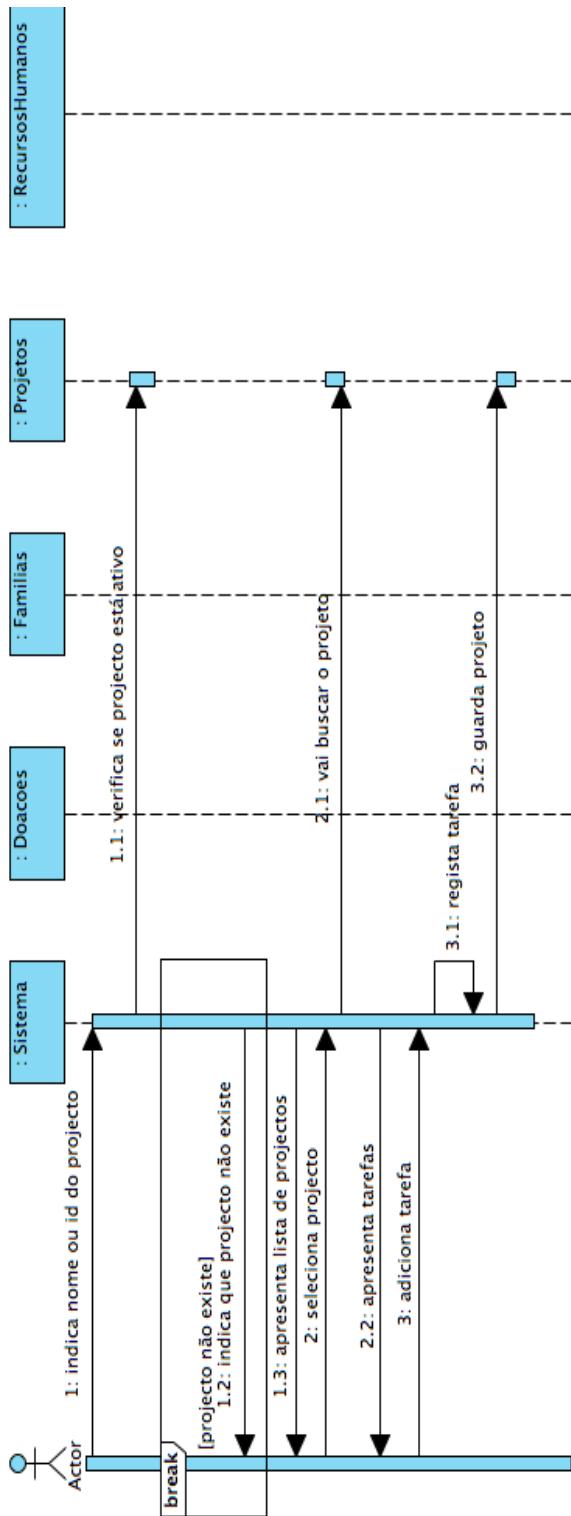


Figura 92: Diagrama de seqüência de subsistemas de Regista Tarefa.

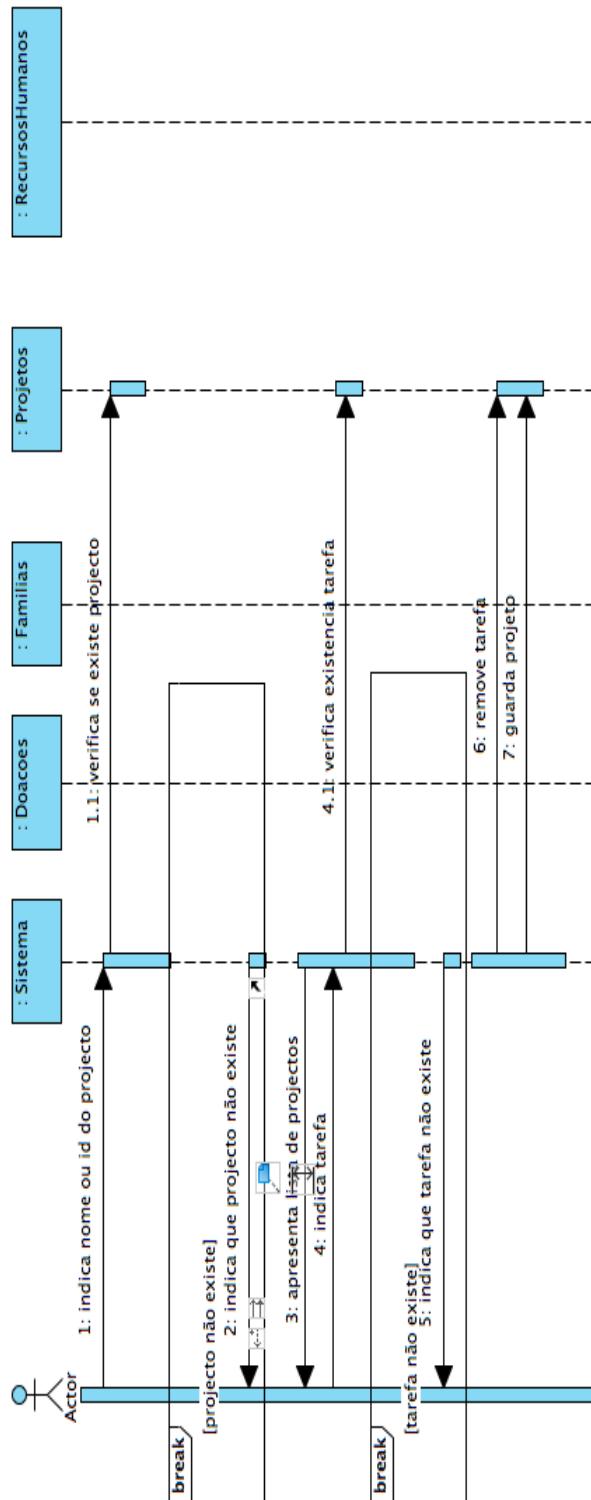


Figura 93: Diagrama de sequência de subsistemas de Remove Tarefa.

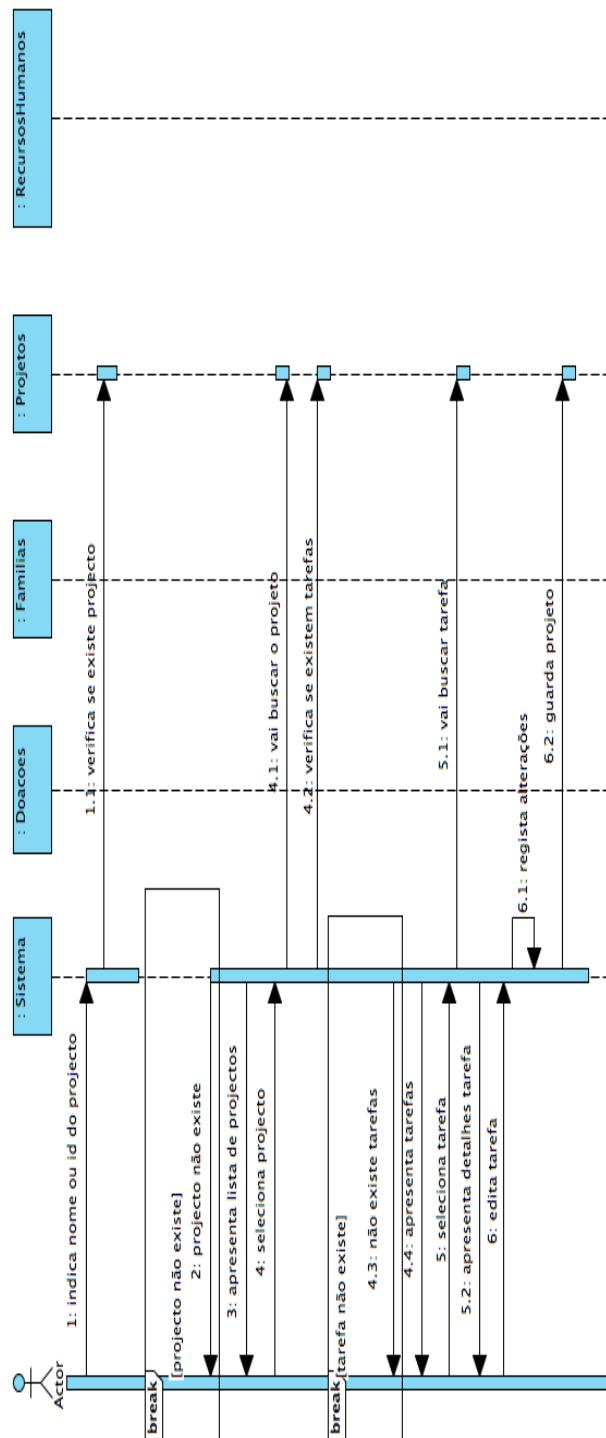


Figura 94: Diagrama de sequência de subsistemas de Edita Projeto.

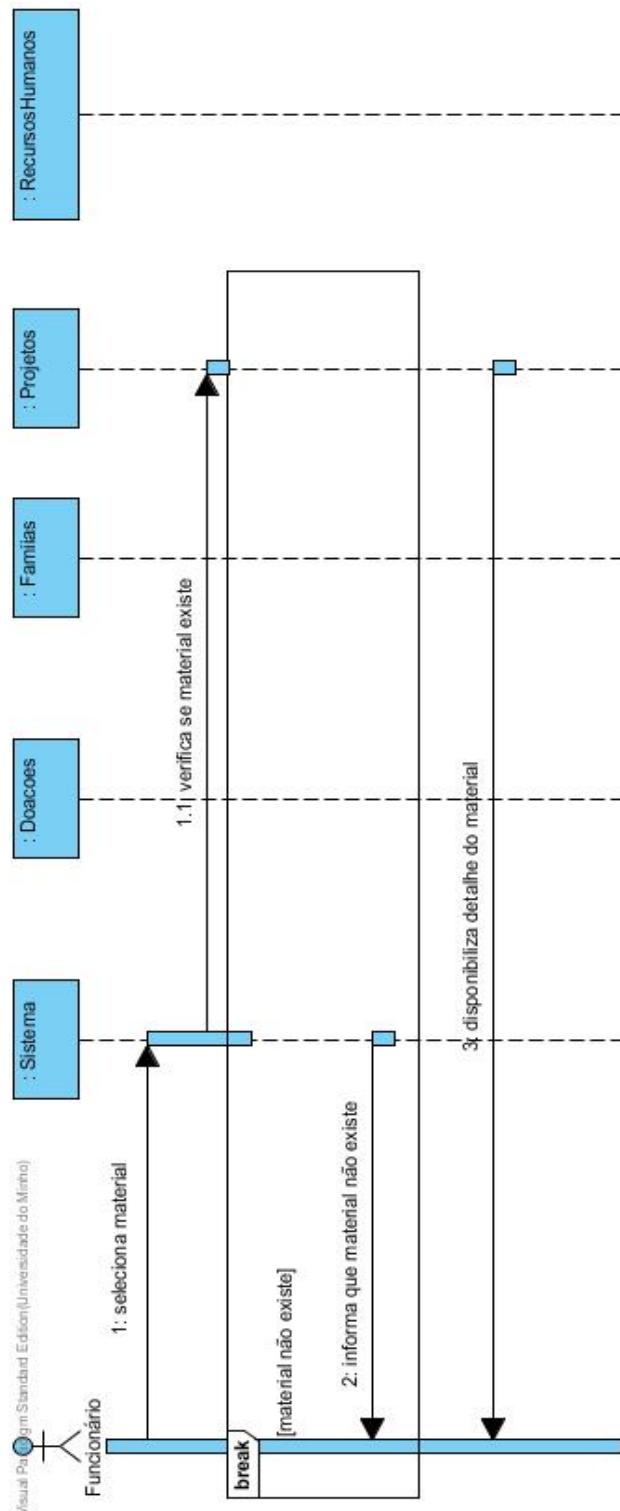


Figura 95: Diagrama de sequência de subsistemas de Consulta Material.

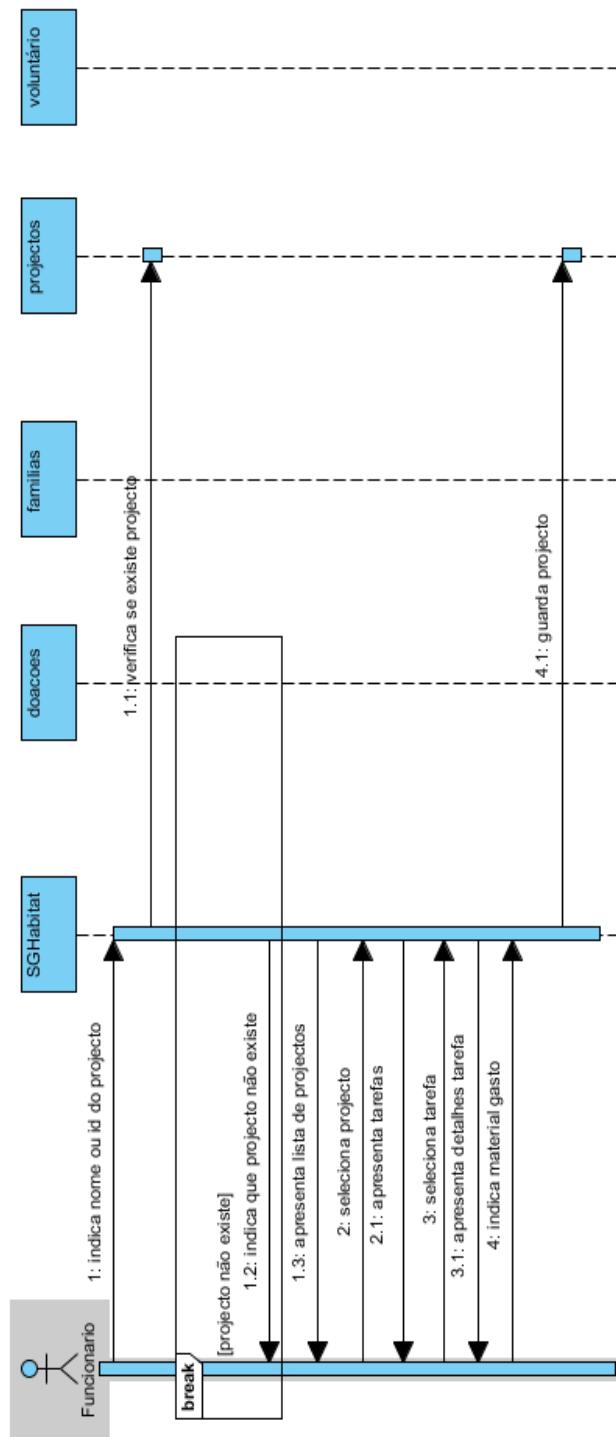


Figura 96: Diagrama de sequência de subsistemas de Regista Material Gasto.

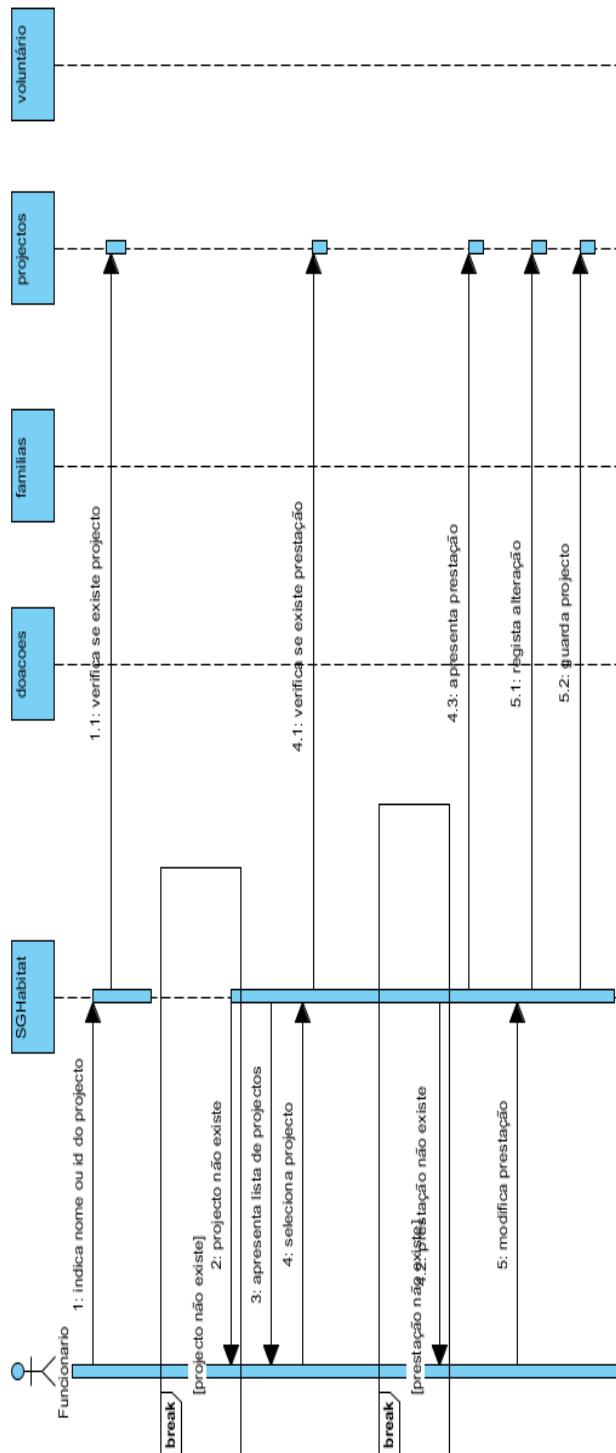


Figura 97: Diagrama de sequência de subsistemas de Edita Prestação.

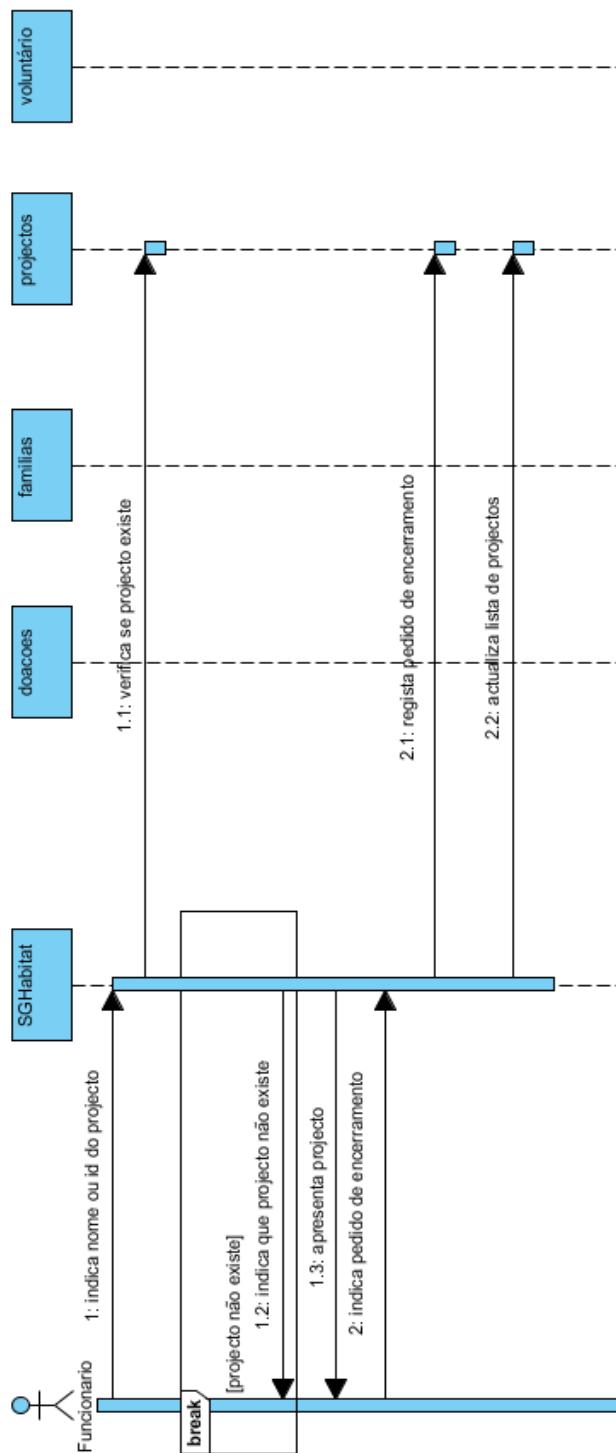


Figura 98: Diagrama de sequência de subsistemas de Encerra Projeto.

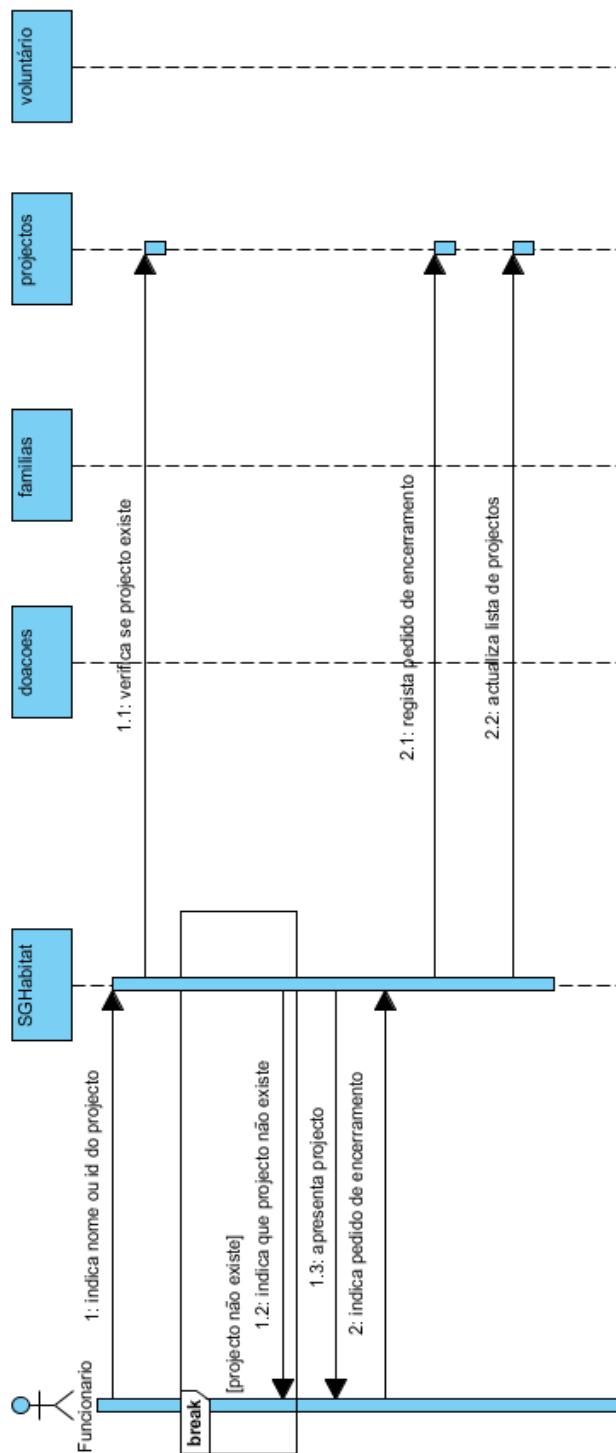


Figura 99: Diagrama de sequência de subsistemas de Encerra Projeto.

6.5 Recursos Humanos

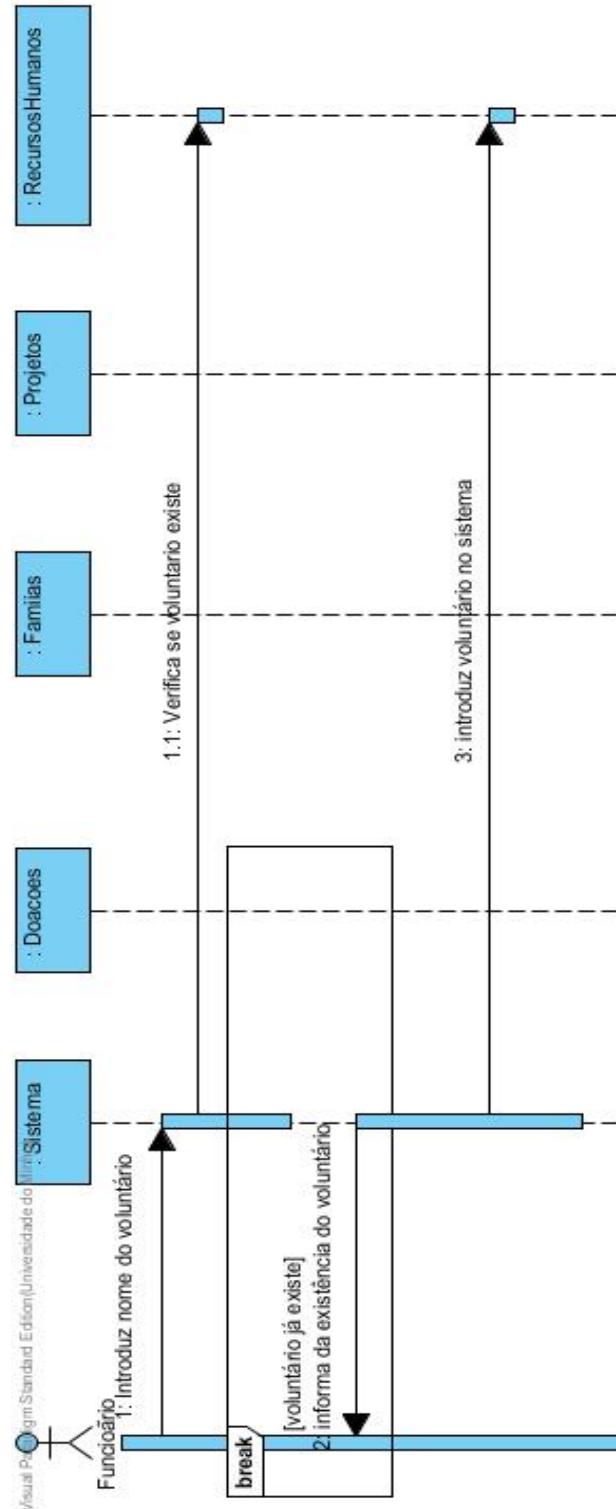


Figura 100: Diagrama de sequência de subsistemas de Regista Voluntário.

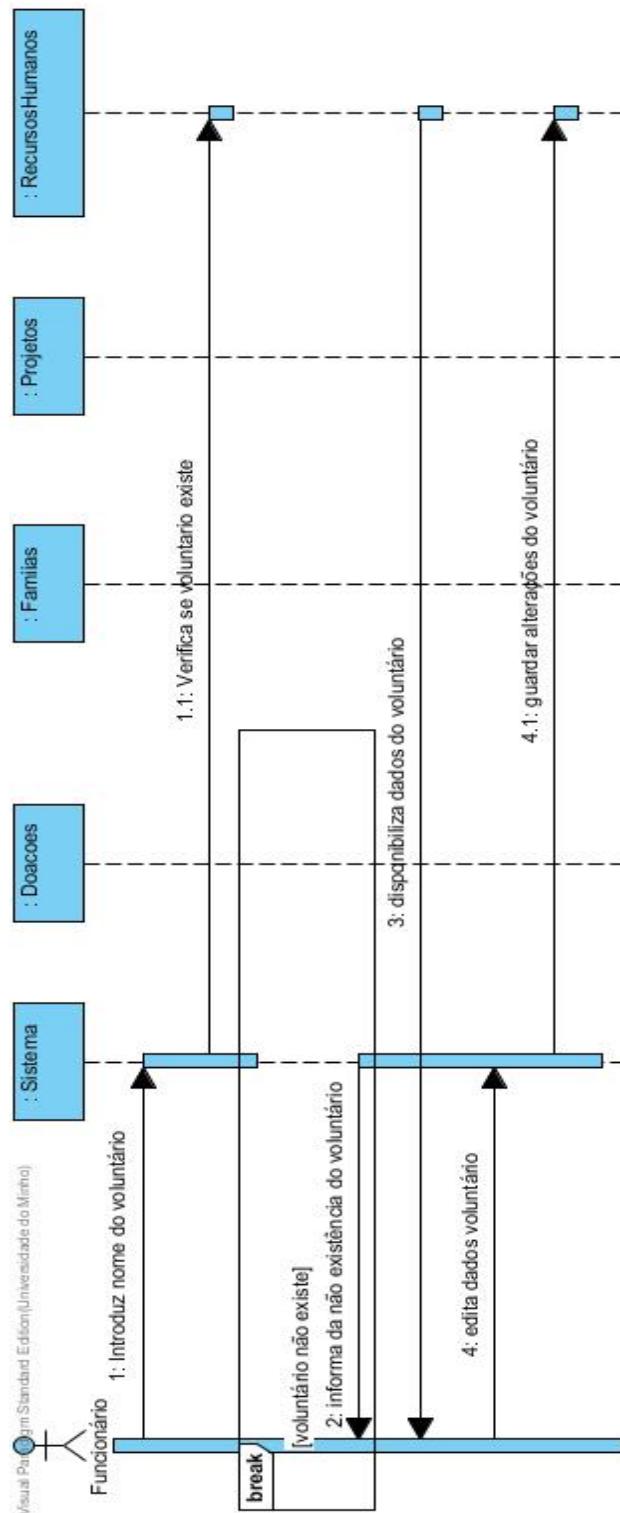


Figura 101: Diagrama de sequência de subsistemas de Edita Voluntário.

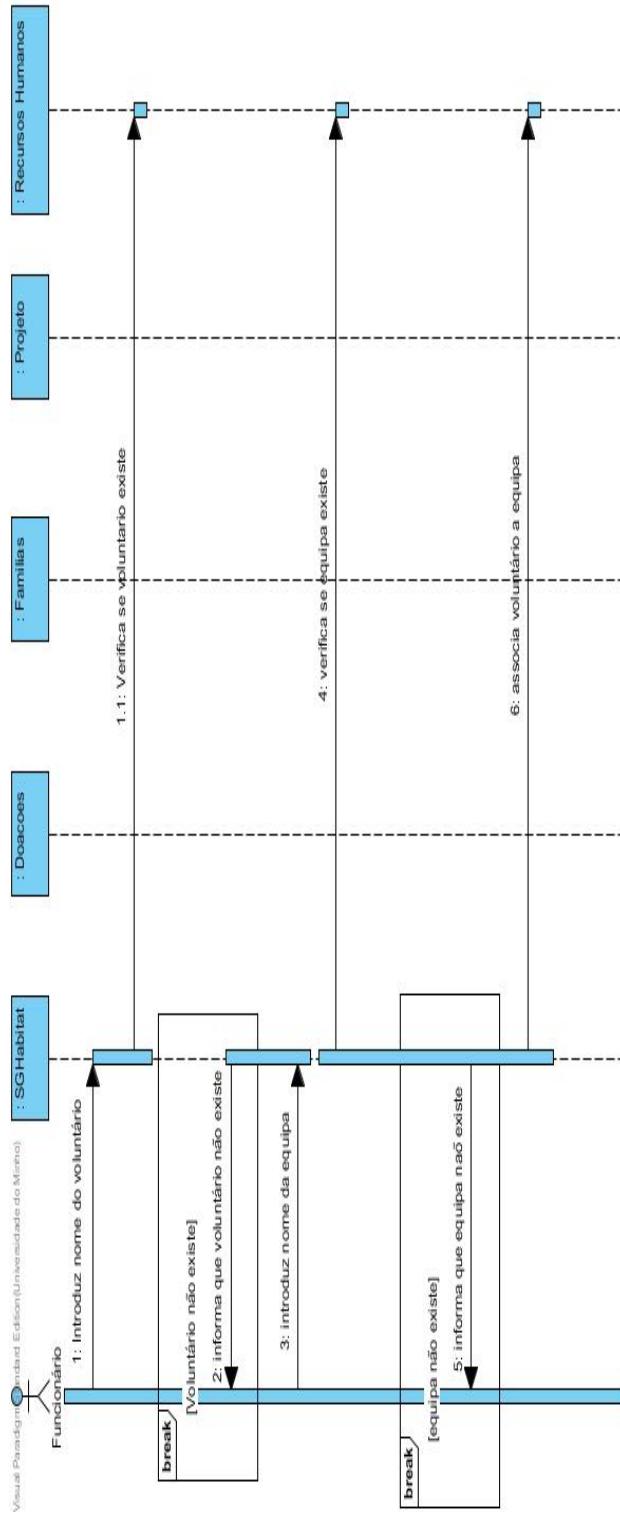


Figura 102: Diagrama de sequência de subsistemas de Associação Voluntário a Equipa.

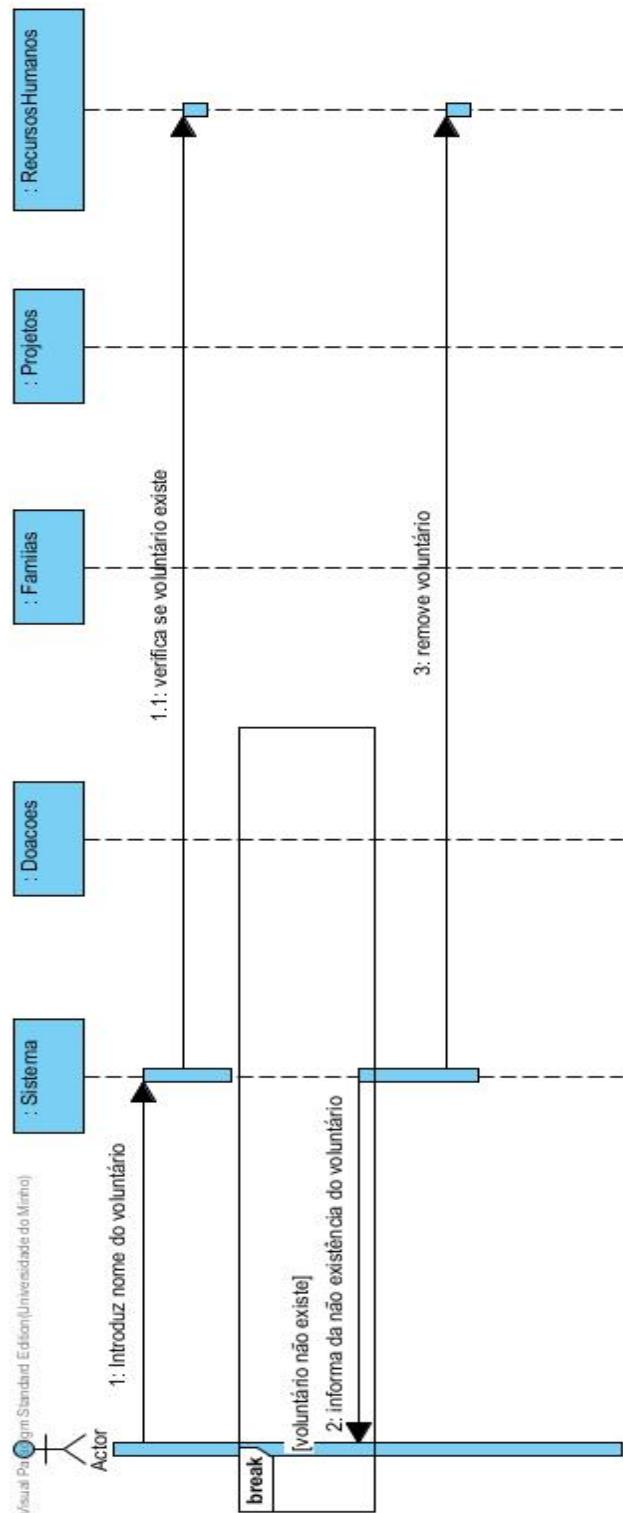


Figura 103: Diagrama de sequência de subsistemas de Remove Voluntário.

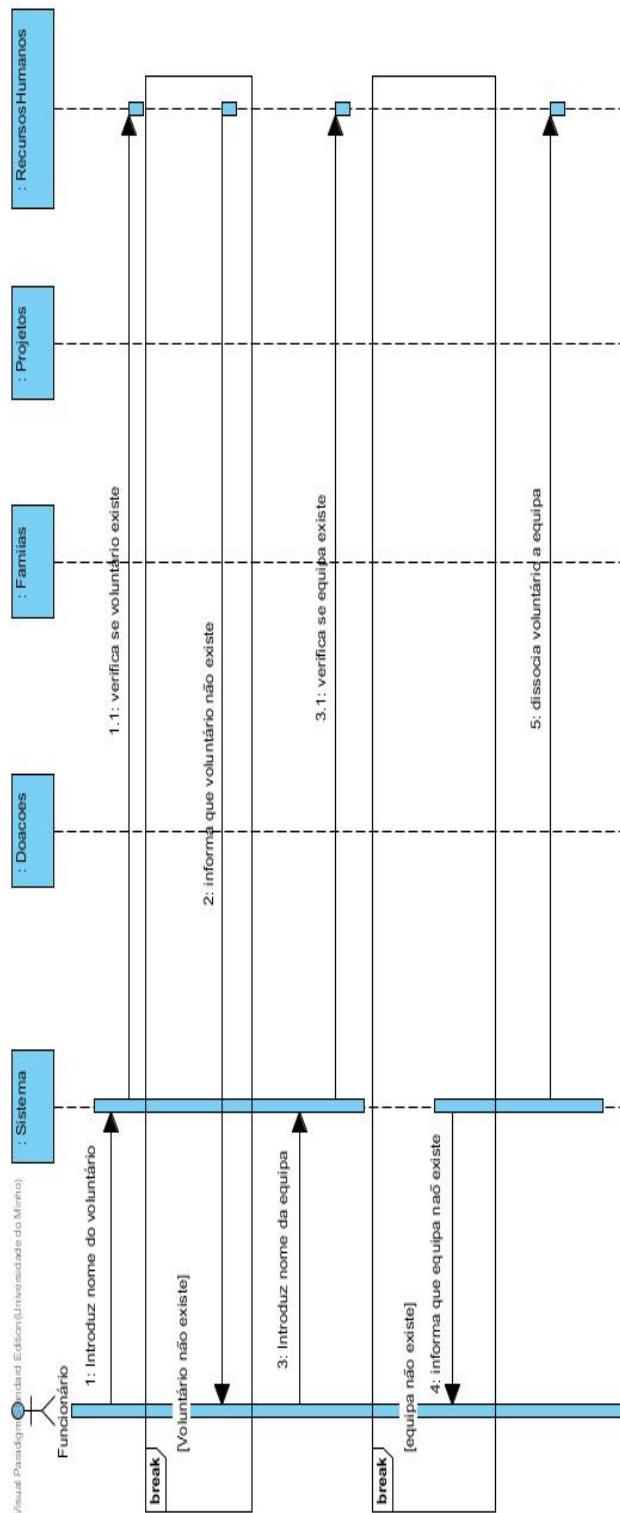


Figura 104: Diagrama de sequência de subsistemas de Associação Voluntário de Equipa.

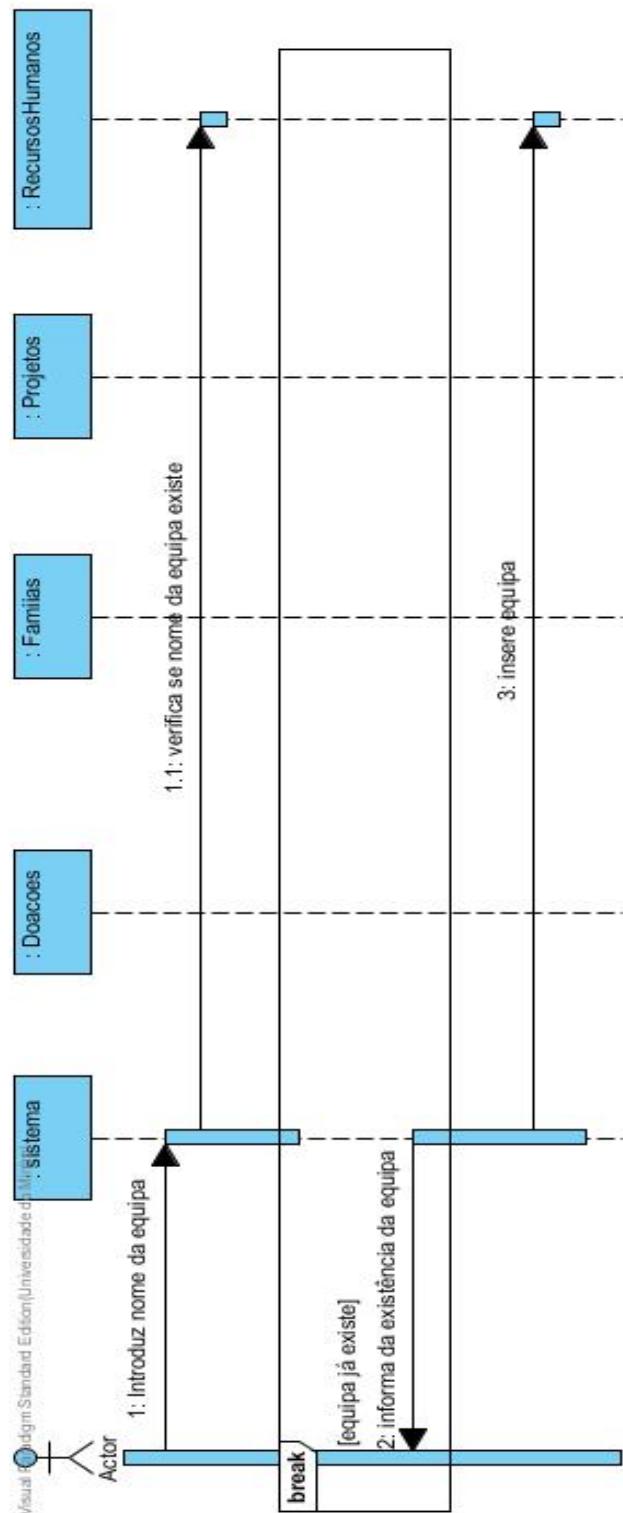


Figura 105: Diagrama de sequência de subsistemas de Regista Equipa.

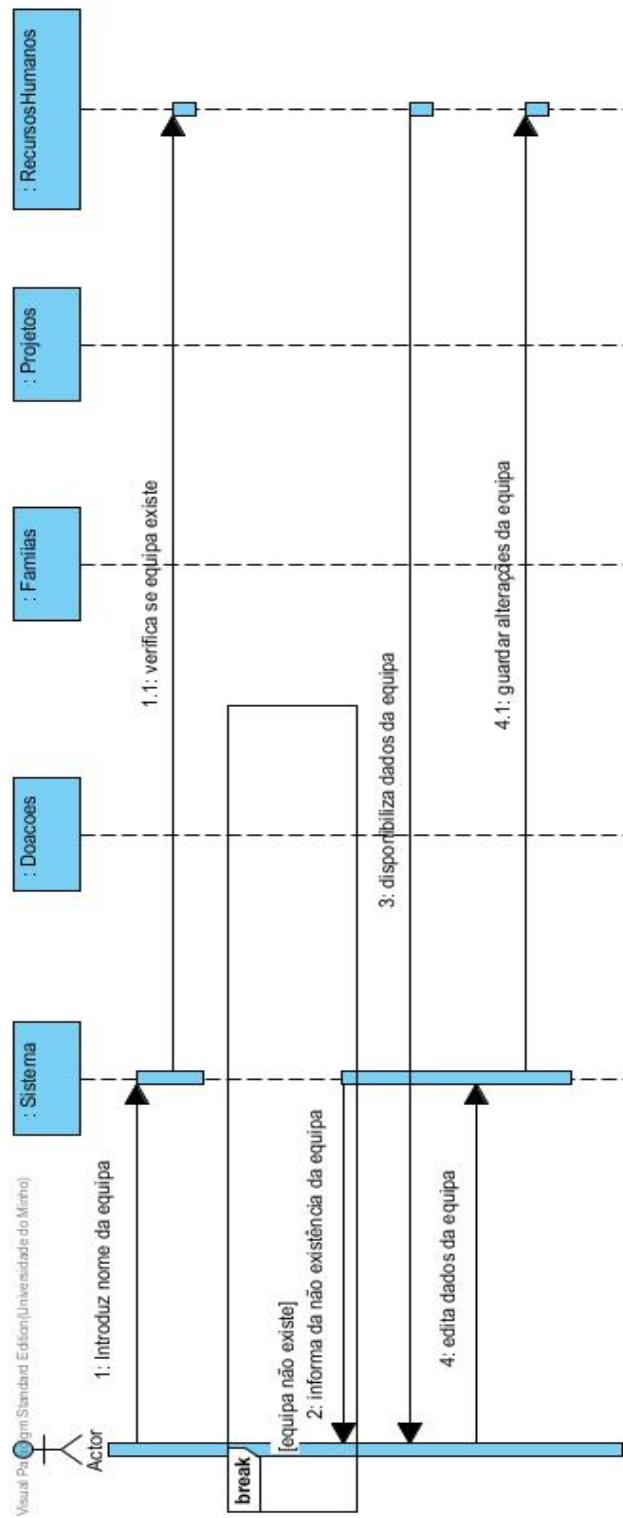


Figura 106: Diagrama de sequência de subsistemas de Edita Equipa.

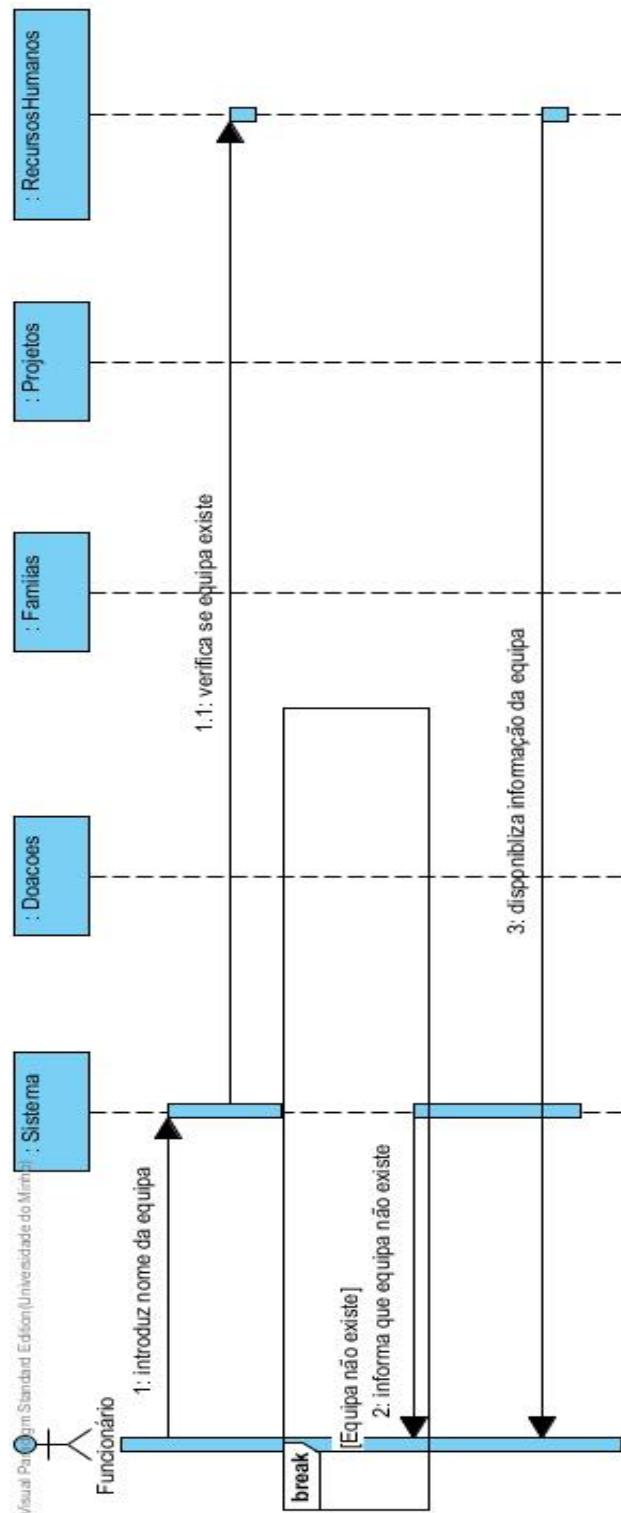


Figura 107. Diagrama de sequência de subsistemas de Consulta Equipa.

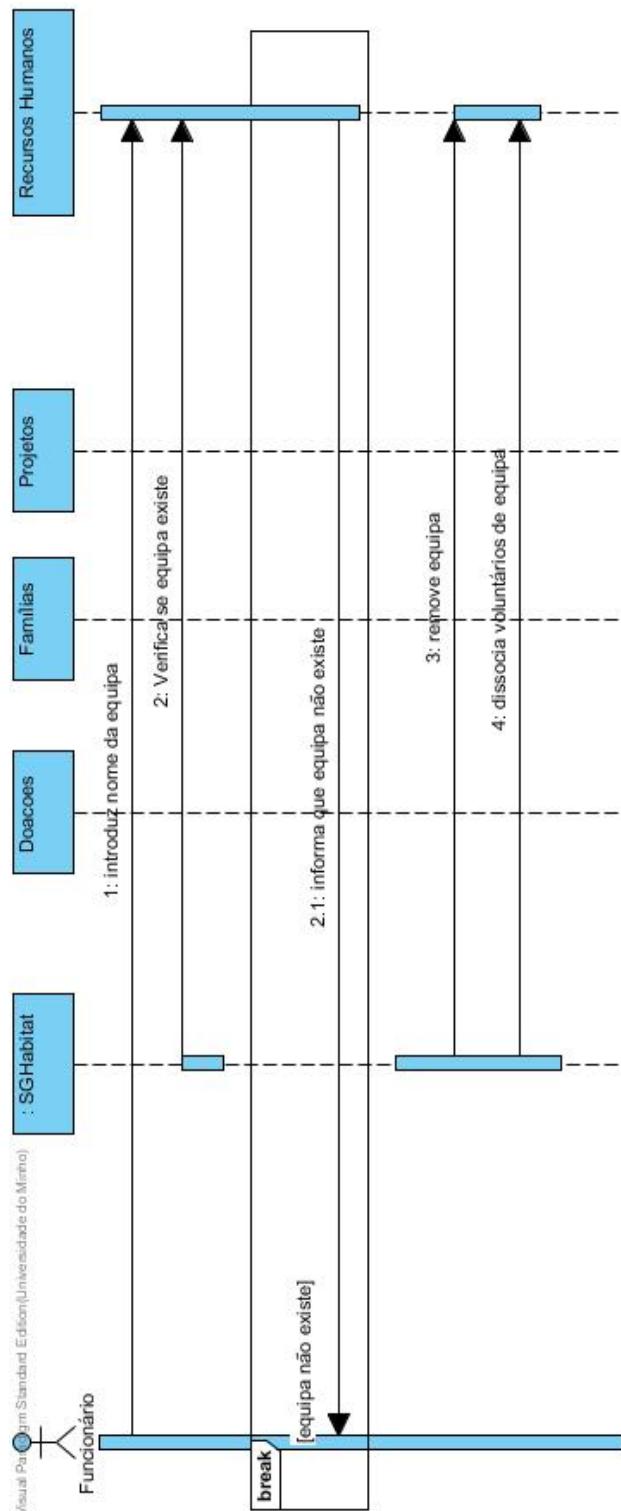


Figura 108: Diagrama de seqüência de subsistemas de Remove Equipa.

7 Diagramas Seq. Implementação

Nesta quase última secção do relatório entramos numa fase final, já muito próxima da implementação. Iremos *desdobrar* alguns daqueles *use cases* que achamos mais relevantes, em diagramas de implementação que mostram quase exatamente o detalhe a nível da implementação do código das operações.

7.1 Autenticação

O log in é sem dúvida, apesar de trivial, um momento de extrema importância no contexto desta aplicação.

Encontra-se implementado no método *public int logIn(String username, String password)*.

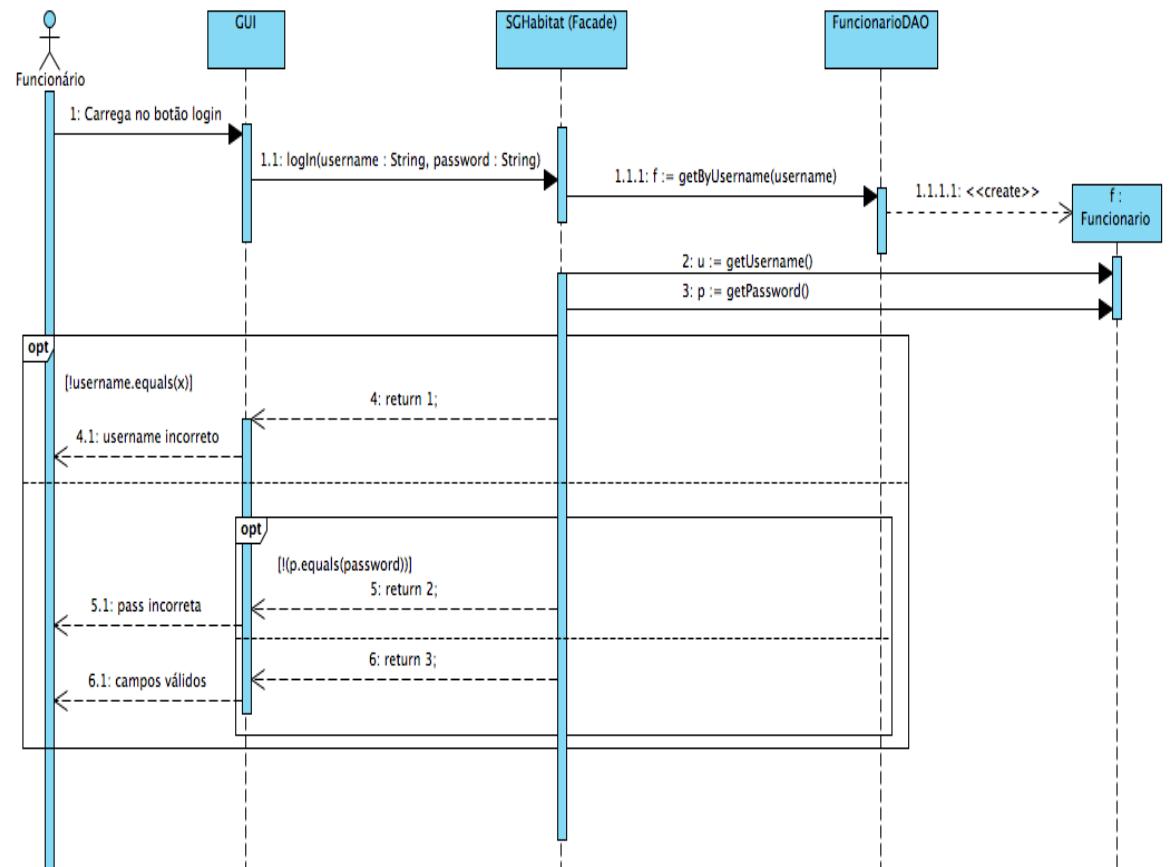


Figura 109: Diagrama de sequência de implementação **Log in**

7.2 Consultar candidatura

Através do método `public ICandidatura fm_getCandidatura(int nr)` disponibilizado no facade **SGHabitat** conseguimos obter uma candidatura a partir do seu identificador.

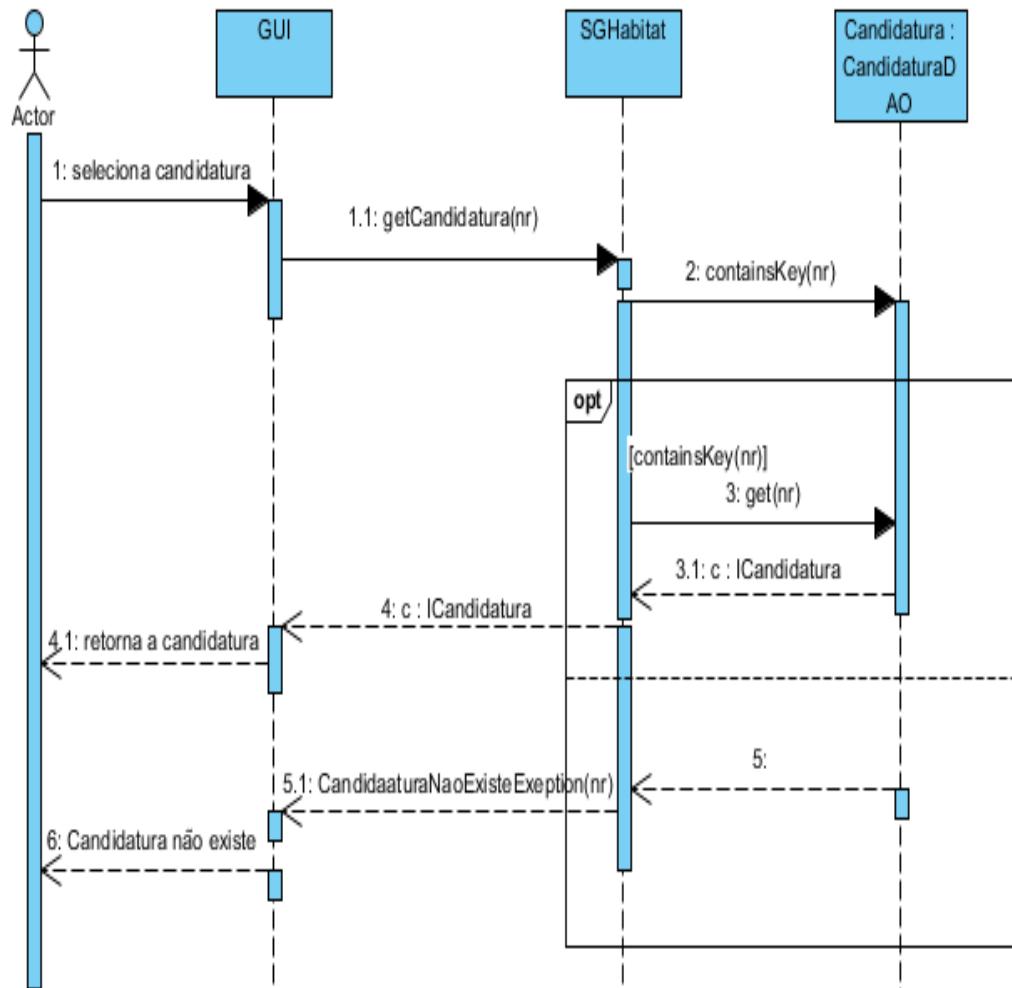


Figura 110: Diagrama de sequência de implementação **Consultar Candidatura**

7.3 Procura Representante

Neste diagrama temos como objetivo implementar um método de pesquisa de um ou vários representantes de uma família pelo seu nome ou apelido, o mesmo se aplica a todas as outras classes.

Método `public Set<IRepresentante> fm_searchRepresentante(String searchinput)`.

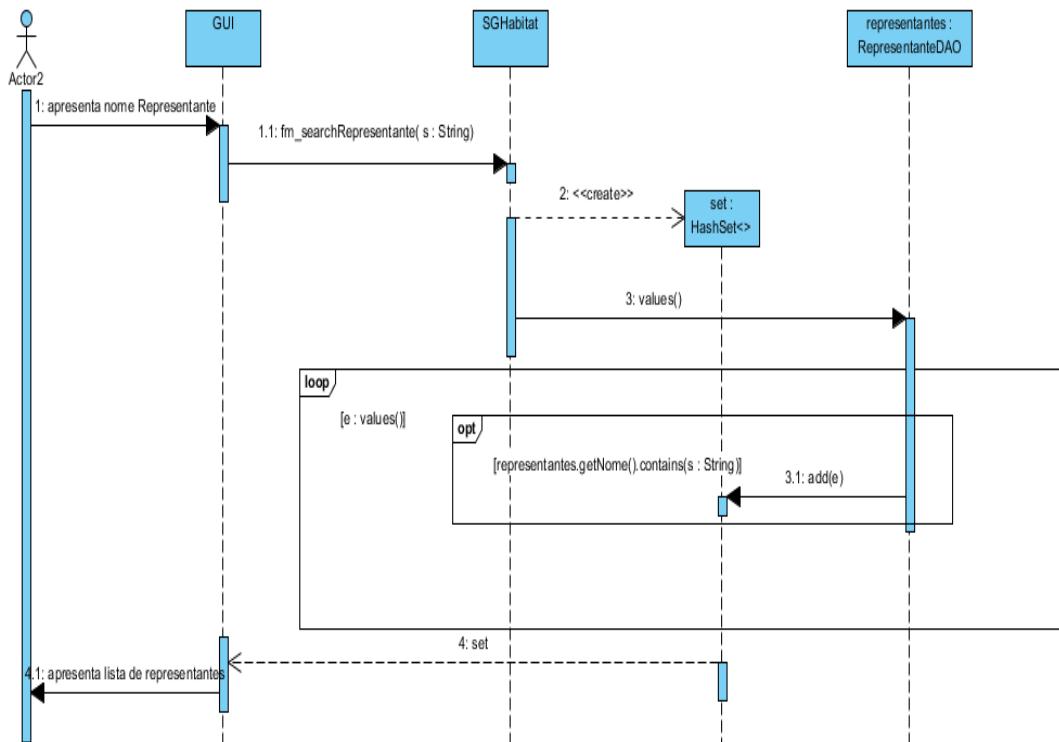


Figura 111: Diagrama de sequência de implementação **Search Representante**

7.4 Insere Projeto

Não podíamos certamente deixar de demonstrar uma inserção nestes diagramas.

O mesmo podemos aplicar a todas as restantes classes do projeto.

Método `public boolean saveProjeto(IProjeto p)`.

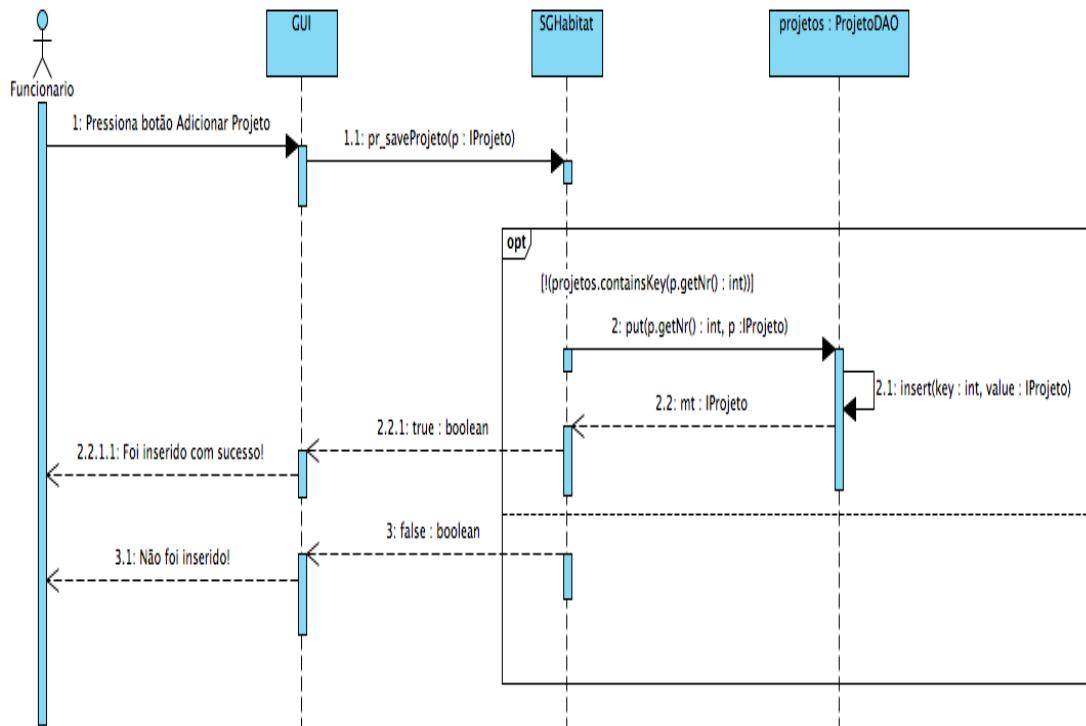


Figura 112: Diagrama de sequência de implementação **Insere Projeto**

7.5 Total Doado por um Doador *DIAGRAMAS SEQ. IMPLEMENTAÇÃO*

7.5 Total Doado por um Doador

Este diagrama é de extrema importância pois permite-nos saber qual o total monetário doado por um doador ou parceria da instituição.

Método ***public float do_totalDoadoPorUmDoador(String nif)*** (Diagrama na página seguinte).

7.5 Total Doador por um Doador

DIAGRAMAS SEQ. IMPLEMENTAÇÃO

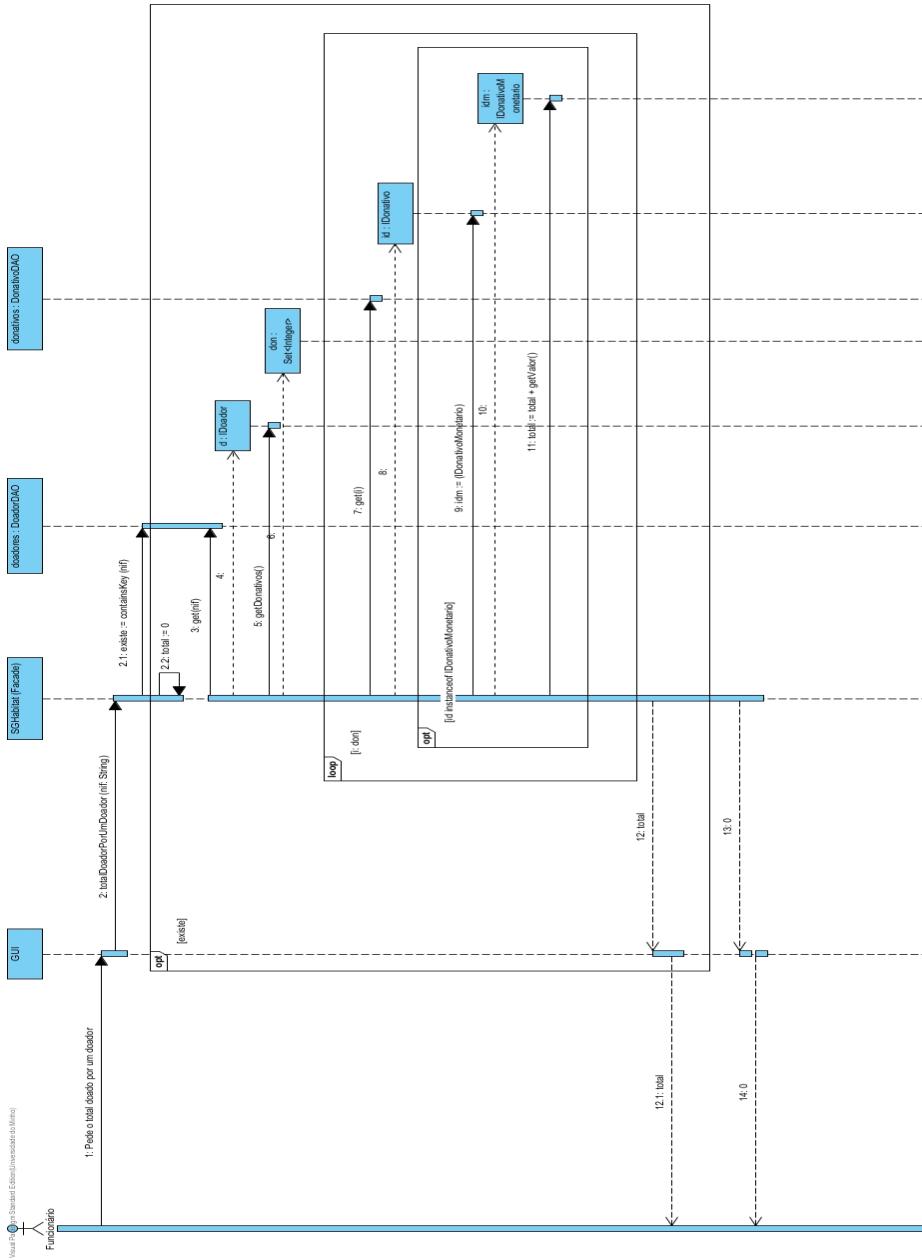


Figura 113: Diagrama de sequência de implementação Total doado por um doador

8 Máquinas de estado

Nesta secção apresentamos as máquinas de estados de algumas entidades do nosso projeto.

- Candidatura;
- Candidatura e Família;
- Projeto;
- Voluntário;

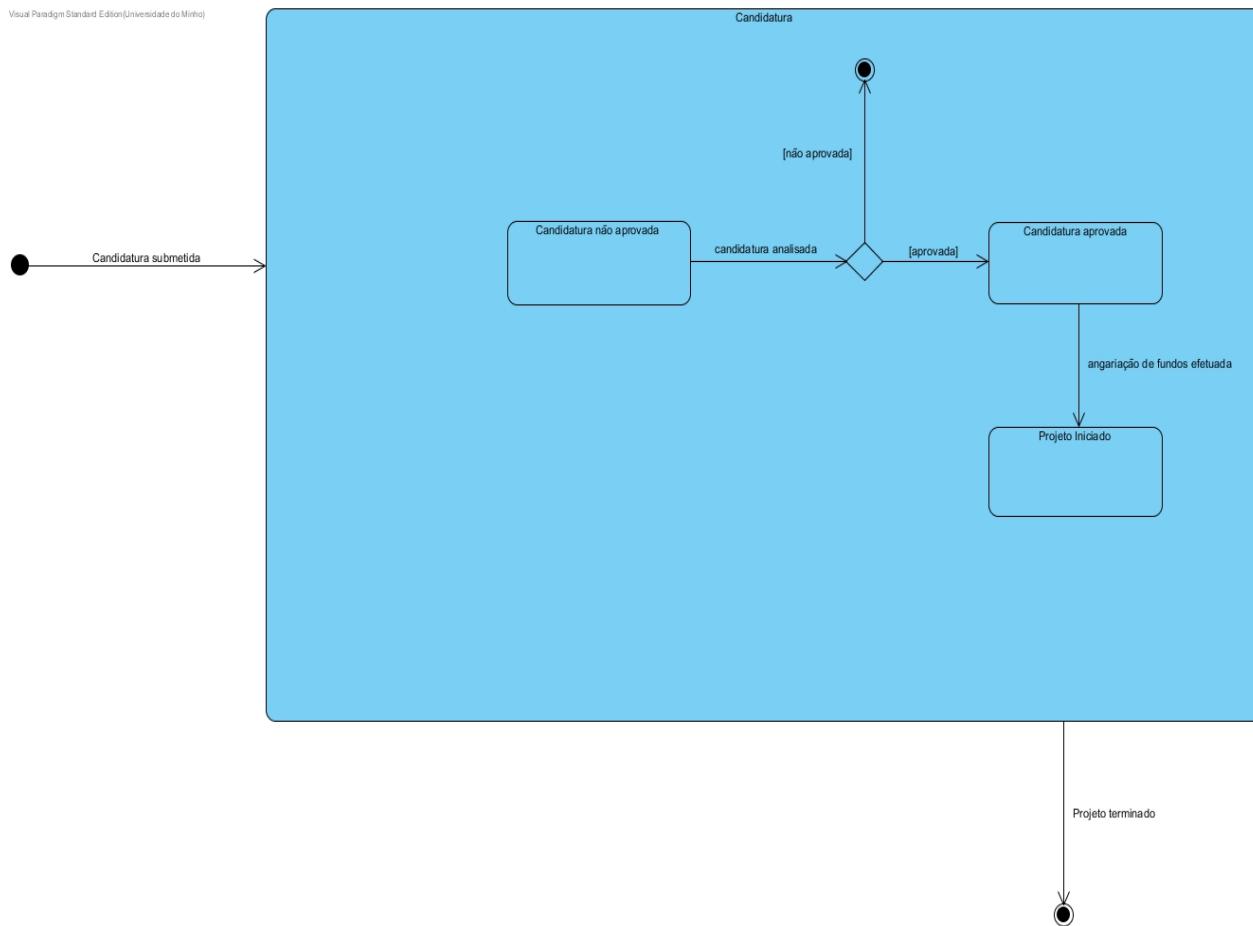


Figura 114: Máquina de estado de **Candidatura**

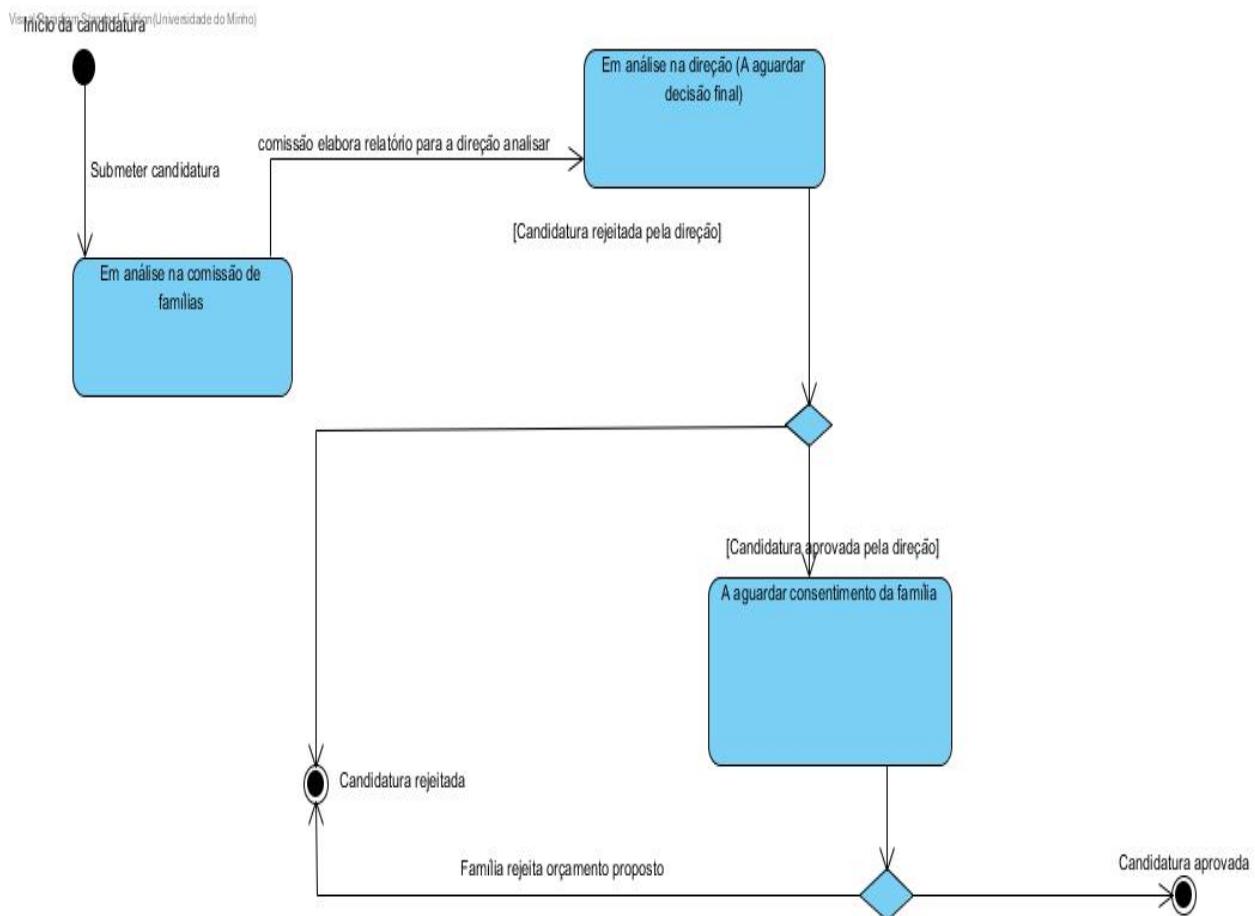


Figura 115: Máquina de estado de **Candidatura e Família**

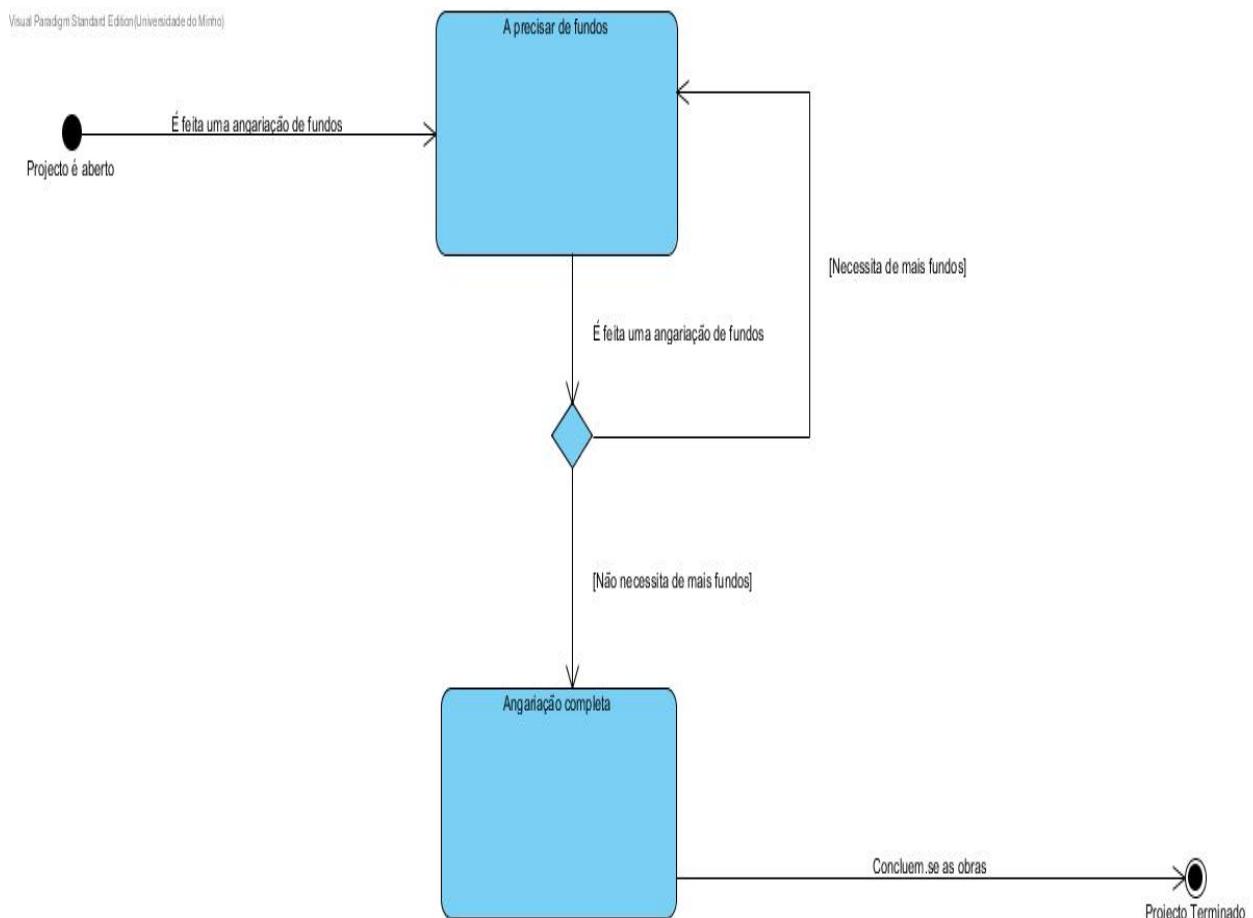


Figura 116: Máquina de estado de **Projeto**

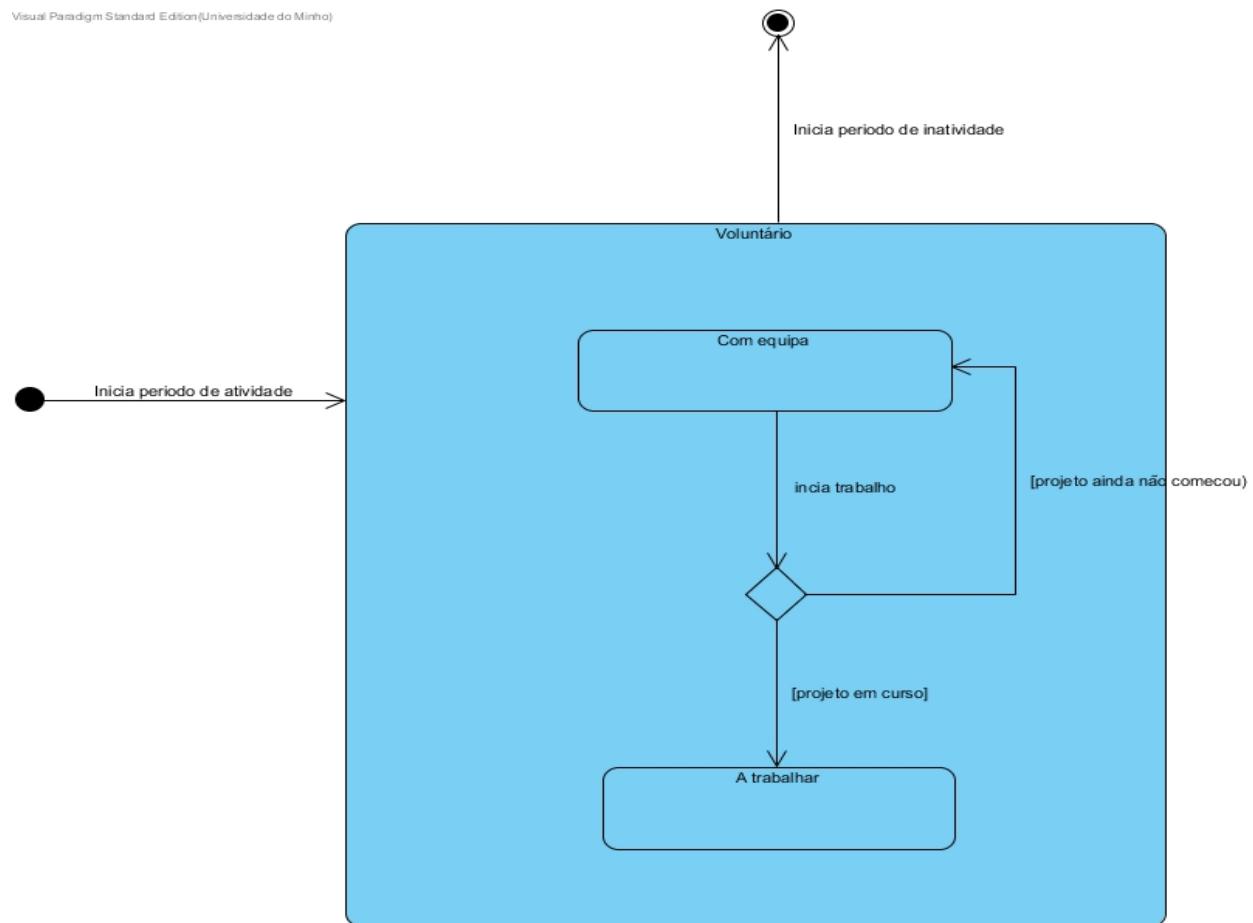


Figura 117: Máquina de estado de **Voluntário**

9 Modelo controlo de diálogo

Nesta secção apresentamos dois diagramas de estados que implementamos na aplicação. Estes diagramas descrevem o processo de registar ou consultar um voluntário e a autenticação no sistema.

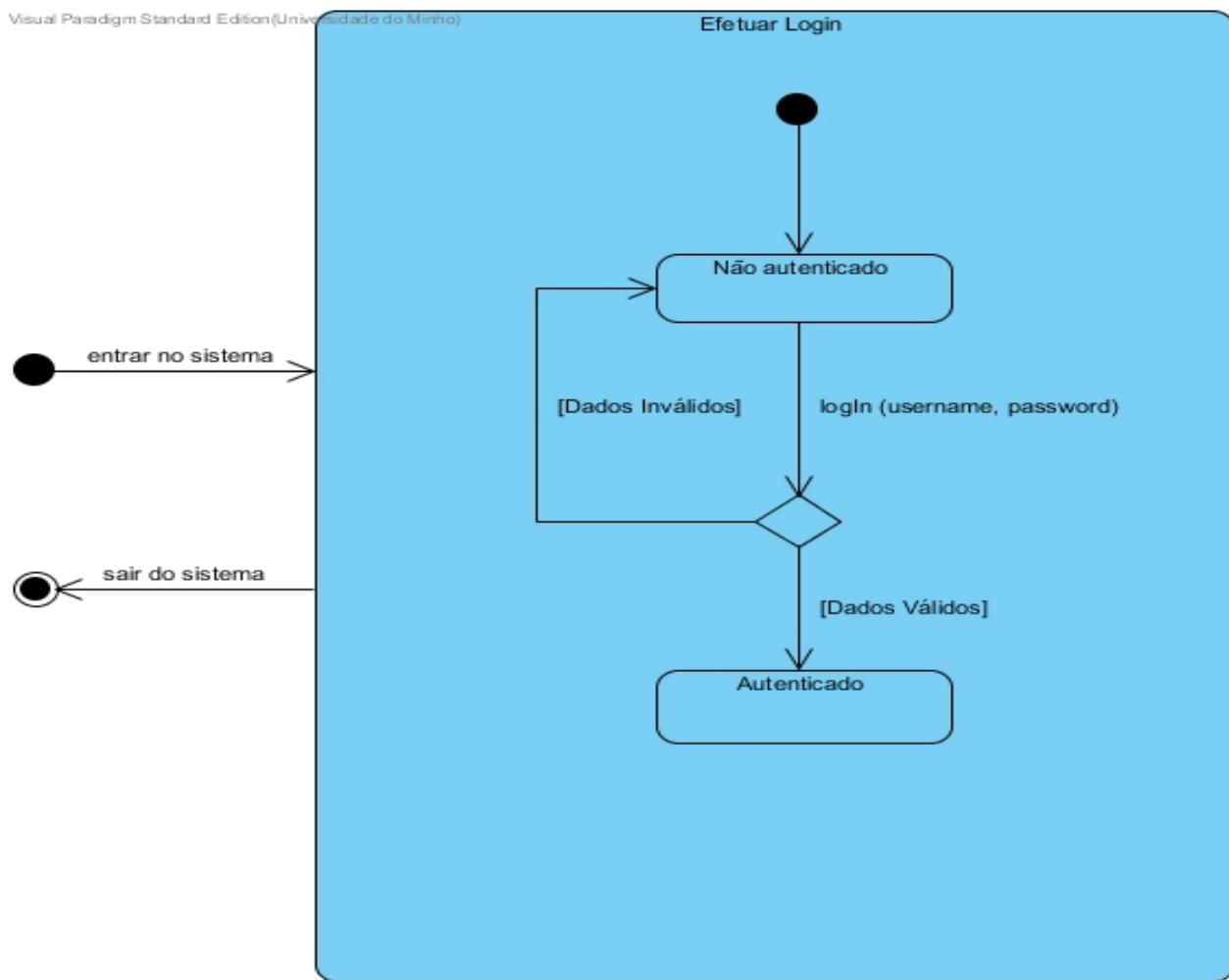


Figura 118: Diagrama de estado de **Registrar/Consultar Voluntário**.

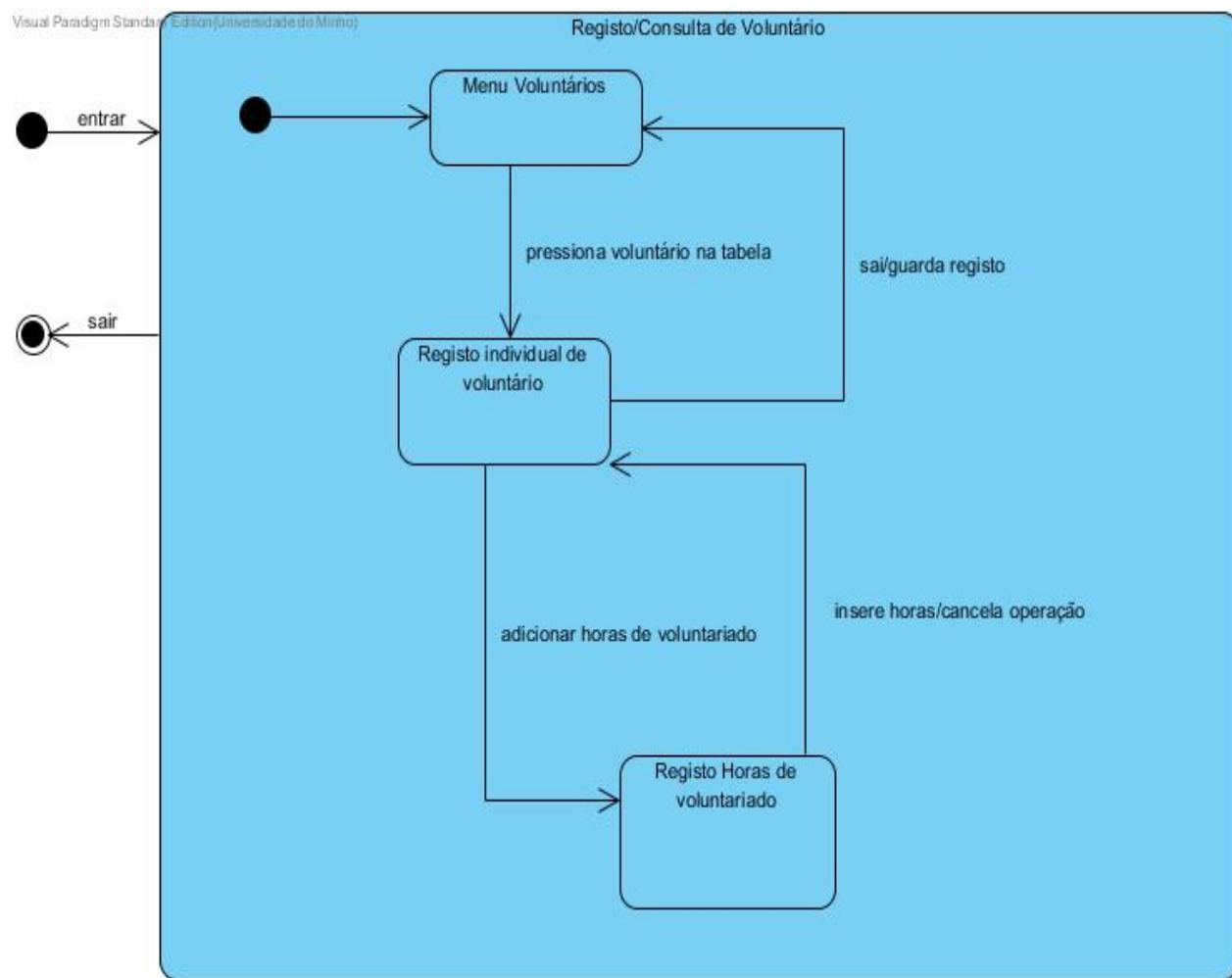


Figura 119: Diagrama de estado de **Autenticação**.

10 Aplicação GestHabitat

10.1 Detalhes de implementação

Por forma a obter código 100% reutilizável o grupo de trabalho estudou para a camada de negócio um método de tornar todas partes lógicas independentes. Como vimos no diagrama de packages em secções anteriores do relatório, existem quatro subpackages na camada de negócio onde para cada um implementamos um ***factory pattern***, fazendo com que exteriormente as classes sejam acedidas apenas por *interfaces*. Vantagens que podemos deduzir:

- Cada subpackage é totalmente independente assim como cada classe dentro dos mesmos;
- O **detalhe de implementação** de cada classe é totalmente escondido visto que podemos **"personalizar o acesso"** a cada classe apenas modificando a interface, escondendo ou mostrando mais métodos;
- Não instanciar classes elimina a fadiga de utilização de construtores, visto que para instanciar objectos com um número razoável de variáveis temos de passar os parametros numa ordem pré-estabelecida, sendo essa ordem alterada todo o código seria afetado, o que não se passa com o uso de *factories*;
- Podemos como vemos na figura em baixo, instanciar através de *factories* objetos na camada de interface gráfica;

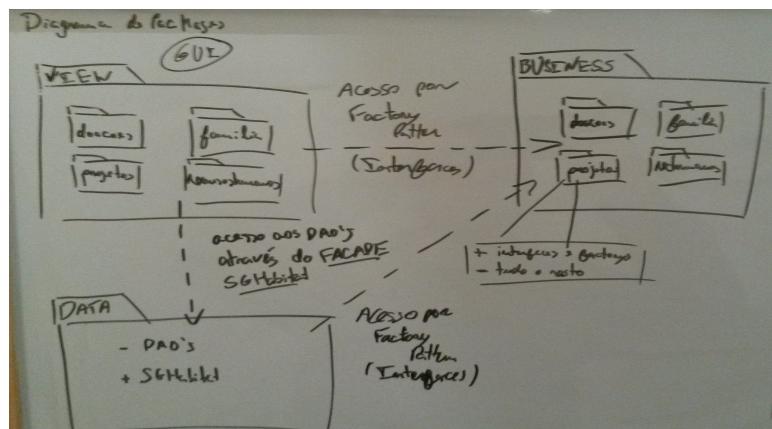


Figura 120: Um rascunho do estudo do diagrama genérico de *packages* e as interações entre os mesmos. Na figura em baixo podemos ver a interação entre *packages* no nosso projeto **GestHabitat**.

10.2 GUI final

Na seguinte figura podemos ver o menu principal da aplicação no seu estado final. Os diferentes botões abrem diferentes áreas de gestão da instituição dentro do *desktoppane*, desta forma a nossa interface oferece uma navegabilidade simples sem *pop-ups* e prática.



Figura 121: Janela log in da aplicação onde o funcionário deve preencher os campos de autenticação.

Mesmo tendo o grupo de trabalho tendo implementado toda a camada de negócio, não implementá-mos toda a camada de interface gráfica como veremos na seguinte secção.



Figura 122: Menu principal e sempre acessível da aplicação, podemos observar as diferentes áreas do lado esquerdo.

10.3 Trabalho efetuado

Após a implementação da **camada de negócio** e com base num protótipo de baixa fidelidade construído na primeira etapa, o grupo implementou uma pequena interface para gerir os **voluntários** da instituição. Focamo-nos mais em aprimorar esta parte da interface pelo que não implementámos as restantes áreas.

10.3.1 Ferramentas da interface

Nr	Nome	Horas
1	Alonso Rodrigues	766
2	Filipe Mena	300
3	Cristina Lopes	48
4	Sara Filipa	69
5	Josefino Fernandes	66
6	Roberto Carlos	0
7	Anderson Talisca	0
8	Lula Silva	2
9	Alex Sandro	35
10	Júlio César	2
11	Paul Pogba	15
12	Napoléon Bonaparte	60
13	Zinedine Zidane	55
14	Charlie Gaulle	133
15	René Descartes	5
16	Yakuza Suzuka	3
17	Shikaku Kaiizo	8
18	Kiriba Fernandes	13
19	Carlitos Afonso	0

Figura 123: Menu voluntários.

O que é que implementámos na interface:

- Tabela simples para visualização simples e abrangente de voluntários;
- **RowSorters** para que se possam vizualizar a tabela de voluntários ordenada por: nº de voluntário, nome e horas totais de voluntariados (**vermelho**);
- Linhas selecionáveis para que aquando de um *double-click* numa linha do voluntário, seja aberto o registo pessoal do mesmo;
- Registo de campos manuais simples para o voluntário com detalhe nas horas de voluntariado pelo nº do projeto (pequena tabela);
- Barra de pesquisa no topo do *desktop pane* por forma a pesquisar voluntários com um dado nome próprio, apelido ou até mesmo o **número** do

voluntário (**abre diretamente o registo a amarelo**).

- A ferramenta mais complexa, um *parser* de **.docx** que permite carregar uma ou mais **fichas de voluntário** através de um **file chooser (azul)**.
- A criação de um relatório **username.txt** no formato texto, de todas as operações efetuadas por um dado funcionário. A hora e data estão sempre junto a cada operação (**um histórico**).
- Lançamento dos diálogos apropriados para informações e remoção de um voluntário.

Observemos as imagens da aplicação em execução...

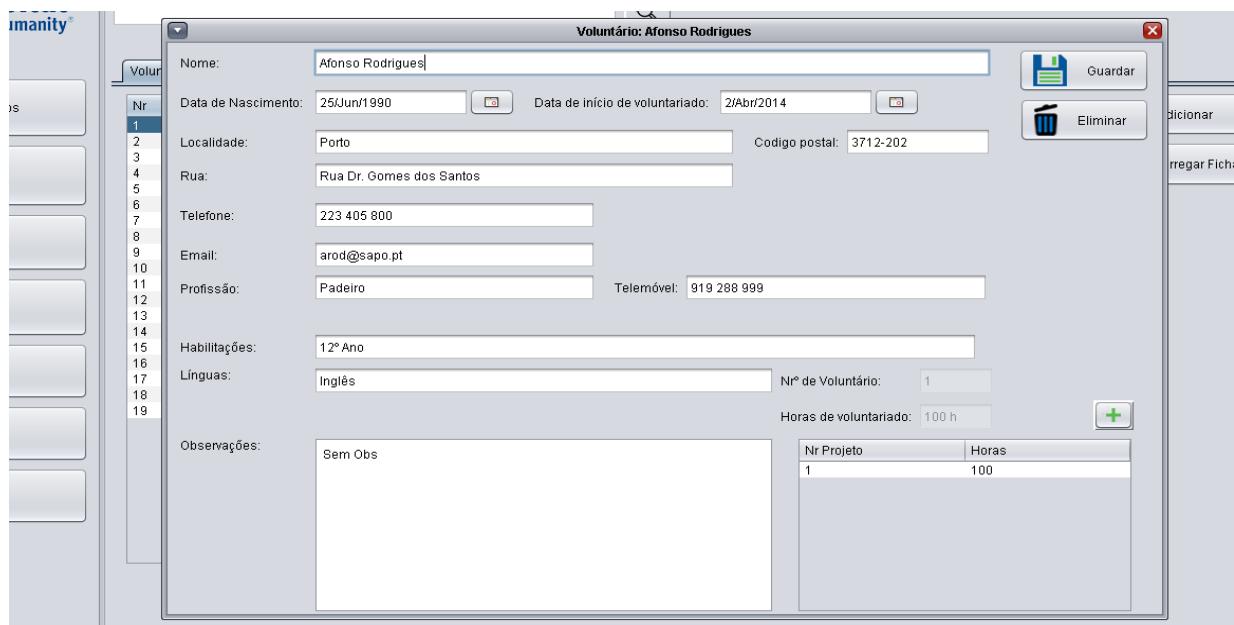


Figura 124: Registo individual de um voluntário.

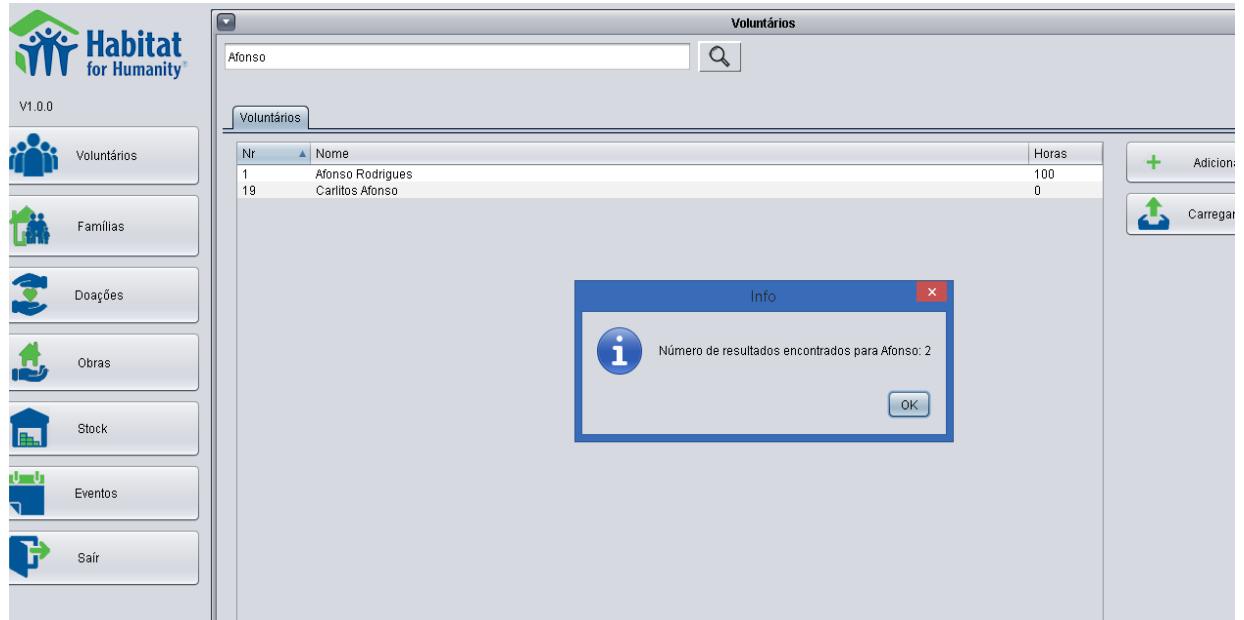
Figura 125: Resultado da pesquisa de voluntário pelo nome **Afonso**.

Figura 126: Ficheiros relatários sublinhados na figura.

```

Bem-vindo!
jdc
2015/01/06 15:02:13

Voluntário editado: Lula Silva Nº: 8
Funcionário: jdc
Data: 2015/01/06 15:07:23

Voluntário editado: Afonso Rodrigues Nº: 1
Funcionário: jdc
Data: 2015/01/06 15:07:29

Voluntário eliminado: Carlitos Afonso Nº:19
Funcionário: jdc
Data: 2015/01/06 15:07:32

```

Figura 127: Excerto de ficheiro .txt que contém relatório do utilizador *jdc* da aplicação.

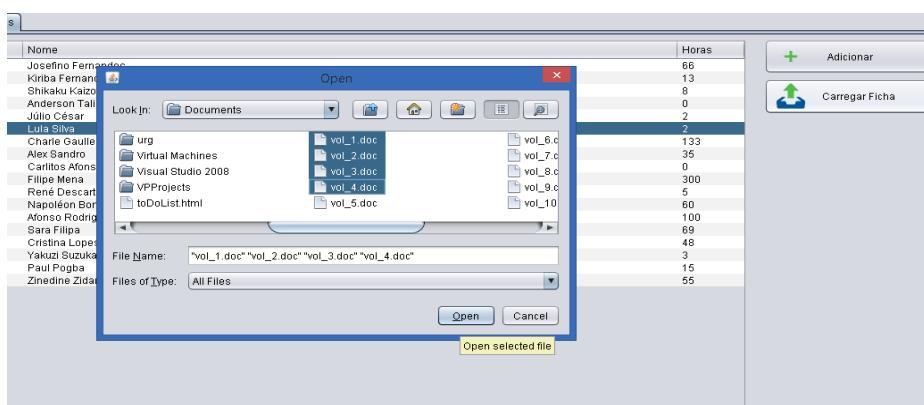


Figura 128: Demonstração da escolha de múltiplas fichas de voluntários para submeter ao sistema, o clique do botão open desencadeia uma série de abertura de janelas dos registos individuais com o resultado do *parse*, dando assim uma oportunidade ao utilizador de alterar algum resultado defeituoso.

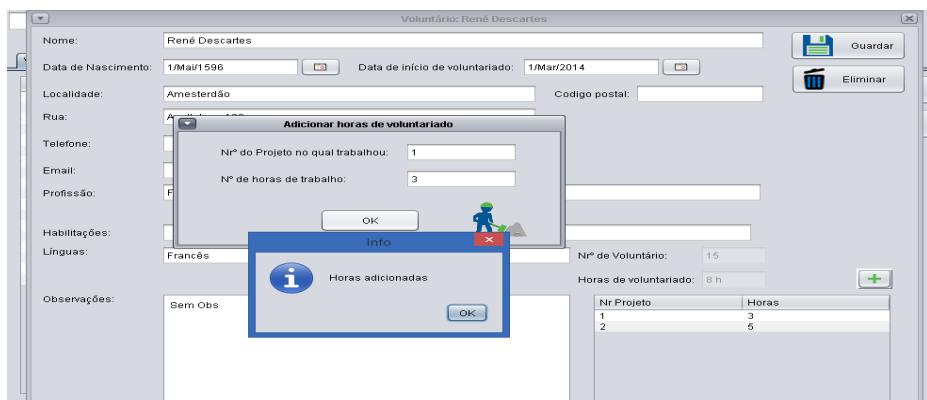


Figura 129: Exemplo de inserção de horas de voluntariado.

10.3.2 Entrega



Figura 130: Aspetto do menu principal da interface gráfica final, com os campos que não implementamos desmarcados.

11 Conclusão

11.1 Comentário

Apesar de não termos completado a aplicação, conseguimos modelar todo o problema e implementar a camada de negócio com acesso à base de dados. Relativamente à modelação aprendemos que esta é muito exigente e exaustiva pelo que exige muito rigor, rigor que falta trabalhar, e é uma lição que retirámos deste trabalho.

Pensamos que a conceptualização do problema e separação lógica foi bem pensada por parte dos elementos do grupo.

11.2 Aprendizagem

- Adquirimos conhecimentos base em UML, na modelação de *software* que nos permitem sistematizar o processo de implementação de aplicações e pensar sobre elas de uma forma estruturada.
- Modelar problema e estudar o domínio do mesmo.
- Definir *use cases* que traduzem interações.
- Usar diversas ferramentas como diagramas de sequência para aproximar *use cases* da implementação final.
- Melhor percepção da programação por camadas com o suporte de **Diagramas de sequência de subsistemas**.
- Programação por camadas.
- Encapsulamento de packages com implementação de **factories**.
- Implementação de DAO's, objetos de conexão a uma base de dados com o suporte do **JDBC**.
- Conhecimentos base de java swing (design) e programação de uma interface gráfica.

12 Anexos

12.1 Documentação javadoc

É enviado na pasta de entrega a documentação do código do projeto na pasta de entrega.

12.2 Scripts MySQL

São enviadas scripts de criação de ambiente de teste para a base de dados da aplicação na pasta de entrega.