Desenvolvimento de Sistemas de Software

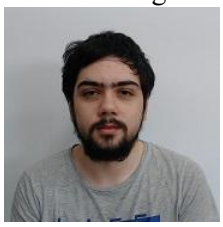
Professores

José Francisco Creissac Freitas Campos

António Manuel Nestor Ribeiro

Trabalho realizado por:

Celso Rodrigues A83655



Carlos Afonso A82529

Gonçalo Nogueira A86617

Marco Sampaio A85508

Luís Pereira A77667

Conteúdo

1. Descrição do enunciado proposto

Este trabalho foi-nos proposto no âmbito da unidade curricular de DSS e tem como objetivo desenvolver uma aplicação de partilha de músicas/vídeo num apartamento.

Neste sentido, foi-nos apresentado um trabalho dividido em 3 fases. Na 1ªfase foi proposto aos alunos que construíssem os requisitos necessários para que a aplicação fosse implementada a par de um protótipo da interface, na 2ªfase era pretendida já uma arquitetura conceptual do sistema, capaz de suportar os Use Cases definidos.

Na 3ª e última fase do trabalho é pedida uma primeira versão da aplicação com implementação dos Use Cases: Iniciar sessão, terminar sessão, fazer upload de conteúdo, alterar categoria de conteúdo e reproduzir conteúdo. A aplicação deverá também suportar a persistência dos dados numa Base de Dados relacional.

Por fim, neste relatório procuramos mostrar tudo aquilo que nos foi pedido, para que possamos explicar como foi obtida a nossa solução para o trabalho proposto.

2. Modelo de Domínio

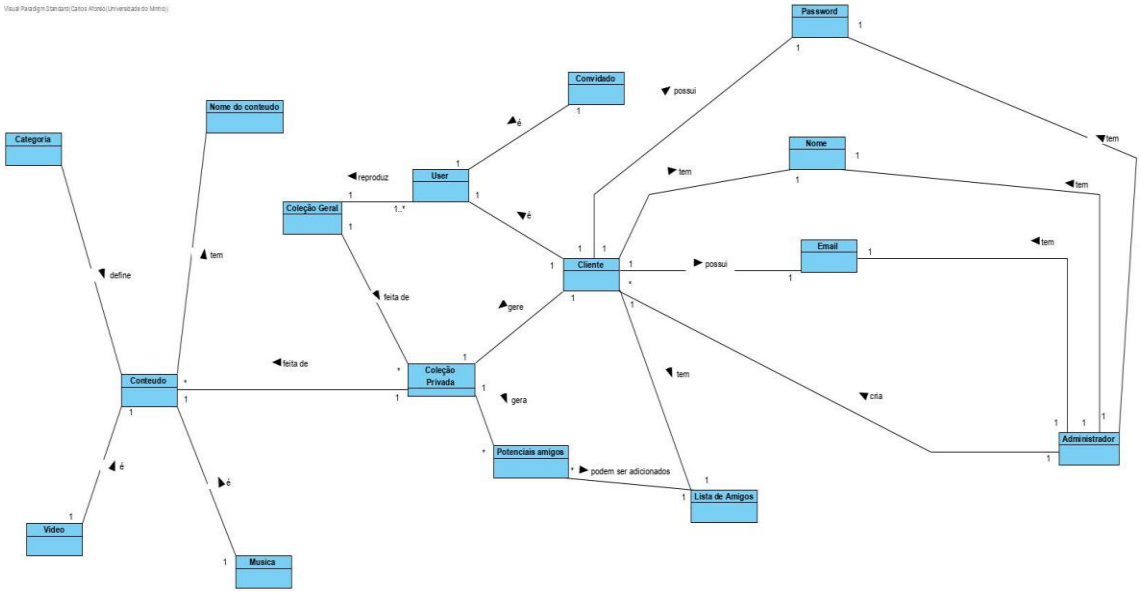


Figura 1 Modelo de Domínio

3. Diagrama de Use Cases

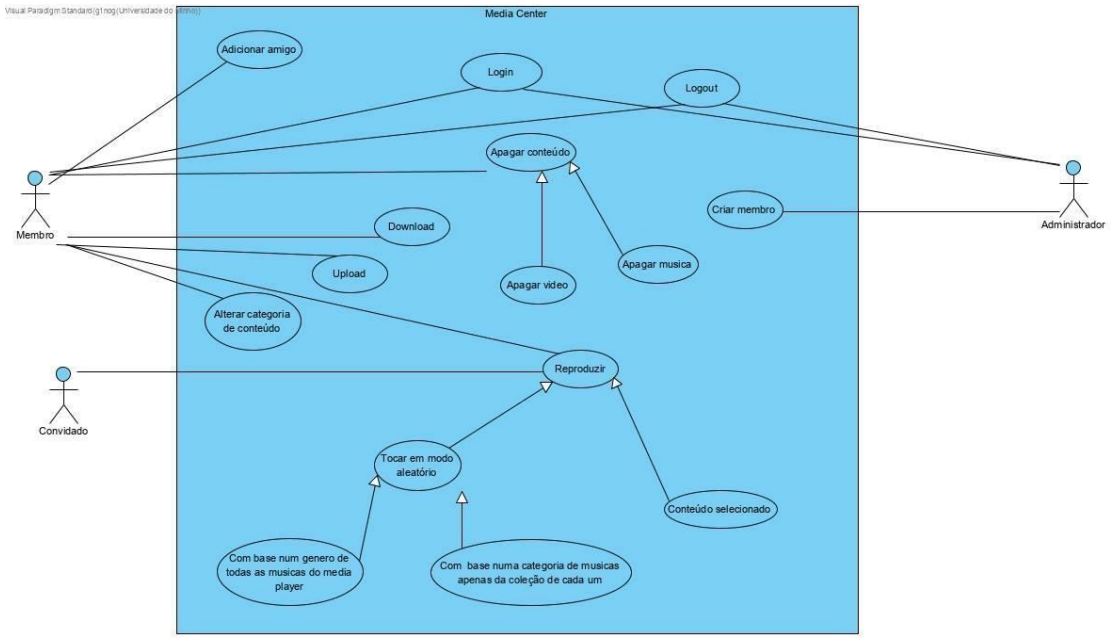


Figura 2 Diagrama de Use Cases

4. Use Cases e respetivas especificações

Use case: Autenticar User

Descrição: O utilizador faz login

Cenário: O João faz login na sua conta

Pré condição: O João não está autenticado

Pós condição: O João tem conta aberta e pode fazer aquilo a que tem direito

Fluxo normal: 1. João insere utilizador.

2. João insere password.

3. Máquina valida os dados. Fluxo de exceção:

3.1 Máquina diz que utilizador ou password estão incorretos.

3.2 João tenta outra vez.

Fluxo alternativo: 3.2.1 João sai.

Use case: Criar membro

Descrição: Administrador cria membro

Cenário: A Joana cria um utilizador para o João

Pré condição: A Joana já está autenticada.

Pós condição: O João fica com uma conta associada.

Fluxo Normal: 1. A Joana faz login.

2. A Joana indica que quer criar um novo utilizador.

3. A Joana indica o nome do utilizador.

4. A máquina salva os dados.

5. A Joana sai.

Fluxo de exceção 1: 2.1 Máquina avisa que utilizador ou password estão errados.

2.2 Volta ao 5.

Fluxo de exceção 2: 4.1 Máquina avisa que utilizador já existe.

4.2 Volta ao 5.

Use case: Adicionar amigo

Descrição: O utilizador adiciona um amigo à sua lista de amigos

Cenário: O João adiciona o Pedro aos seus amigos

Pré condição: O João esta autenticado

Pós condição: O Pedro fica na lista de amigos do João

Fluxo normal: 1. O João insere o utilizador do Pedro

2. A máquina salva os dados.

3. Volta atrás.

Fluxo de Exceção 1 (2º passo) [O João e o Pedro já são amigos] 2.1 A máquina diz que o Pedro e João já são amigos.

3. Volta atrás.

Use case: Apagar conteúdo

Descrição: O utilizador apaga conteúdo da cloud

Cenário: O João apaga conteúdo da cloud

Pré condição: O João está autenticado como membro. Pós-condição: O conteúdo é apagado

Fluxo normal: 1.O João entra em “My Collection”

2. O João seleciona uma música.

3. O João apaga a música.

4. O sistema exclui a música da coleção do João.

5. O João volta atrás.

Fluxo alternativo (passo 2) 2.1 O João seleciona um vídeo.

2.2 O João apaga o vídeo.

2.3 O sistema exclui a vídeo da coleção do João.

2.4 O João volta atrás.

Use case: Fazer download

Descrição: O utilizador faz download de conteúdo

Cenário: O João faz download de conteúdo

Pré condição: O João está autenticado como membro.

Pós condição: O João fica com o conteúdo descarregado.

Fluxo normal: 1. O João entra em “My Collection”.

2. O João seleciona uma música.

3. O João descarrega o ficheiro.

4. O sistema salva os dados.

5. O João volta atrás.

Fluxo alternativo: 2.1. O João descarrega um vídeo.

2.2. Volta ao 3.

Use case: Fazer upload

Descrição: O utilizador faz upload de conteúdo

Cenário: O João apaga faz download de uma música

Pré condição: O João está autenticado como membro.

Pós-condição: O João fica com a música na sua coleção.

Fluxo normal: 1. O João entra em “My Collection”.

2. O João dá upload de ficheiro.

3. O sistema guarda os dados.

Use case: Alterar categoria de conteúdo

Descrição: O utilizador altera categoria.

Cenário: O João altera categoria de uma música.

Pré condição: O João está autenticado como membro.

Pós condição: A categoria da música fica alterada.

Fluxo normal: 1. O João entra em “My Collection”.

2. O João seleciona a música cuja categoria quer alterar e escreve a nova categoria.

3. O João dá “save”.

4. O sistema guarda os dados.

Use case: Reproduzir conteúdo.

Descrição: O utilizador reproduz conteúdo.

Cenário: O João reproduz conteúdo.

Pré condição: O João está autenticado como membro.

Pós-condição: O João ouviu/visualizou o conteúdo.

Fluxo normal: 1. O João entra em “My Collection”.

2. O João reproduz conteúdo selecionado.

3. O João Sai do sistema.

Fluxo alternativo: 2.1 O João reproduz conteúdo em modo aleatório com base numa categoria de todas as músicas do media player.

2.2. O João Sai do sistema.

Fluxo alternativo: 2.1.1. O João reproduz conteúdo em modo aleatório com base numa categoria de músicas apenas da coleção de cada um.

2.2.2. O João Sai do sistema.

5. Proposta da interface com o utilizador

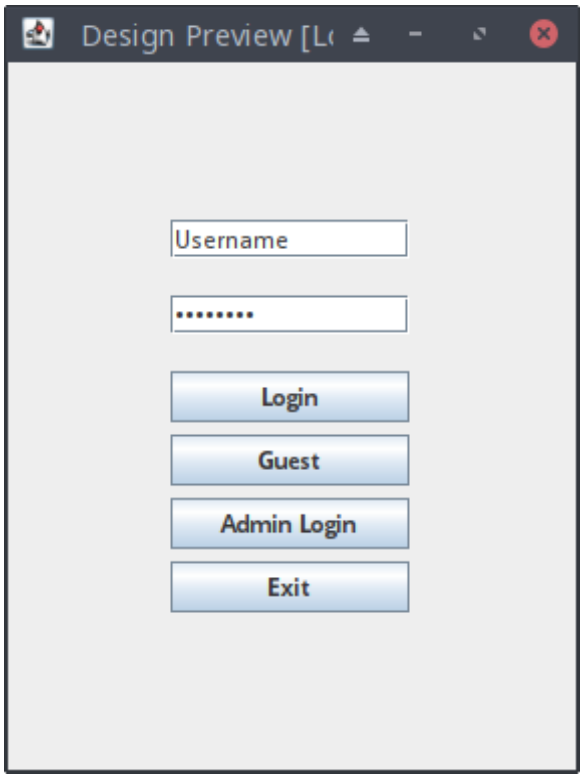


Figura 3 Login

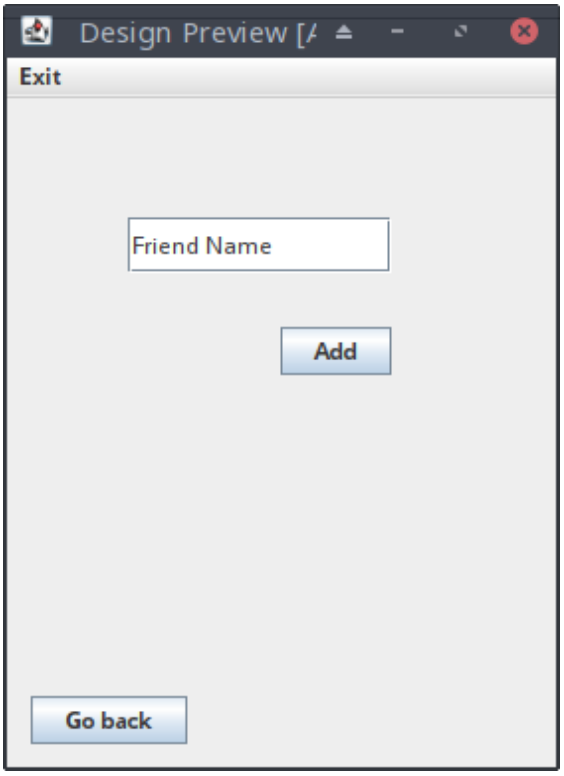


Figura 4 Adicionar Amigos

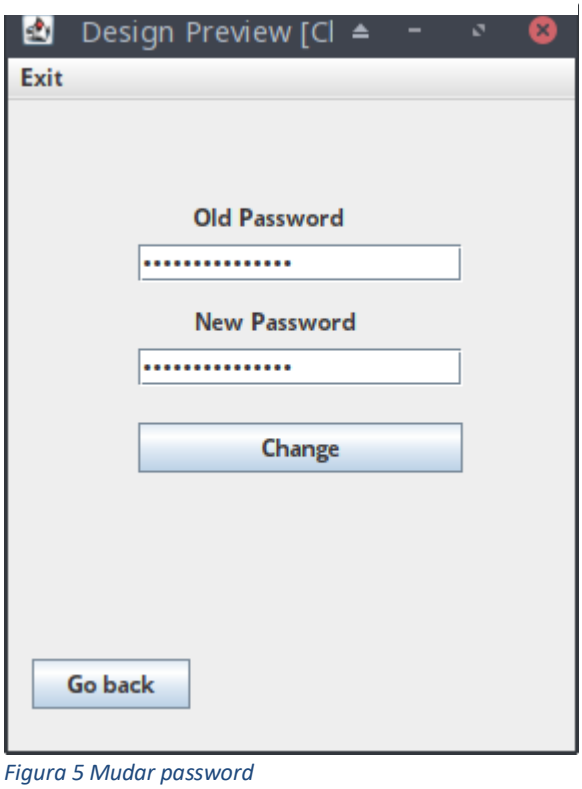


Figura 5 Mudar password

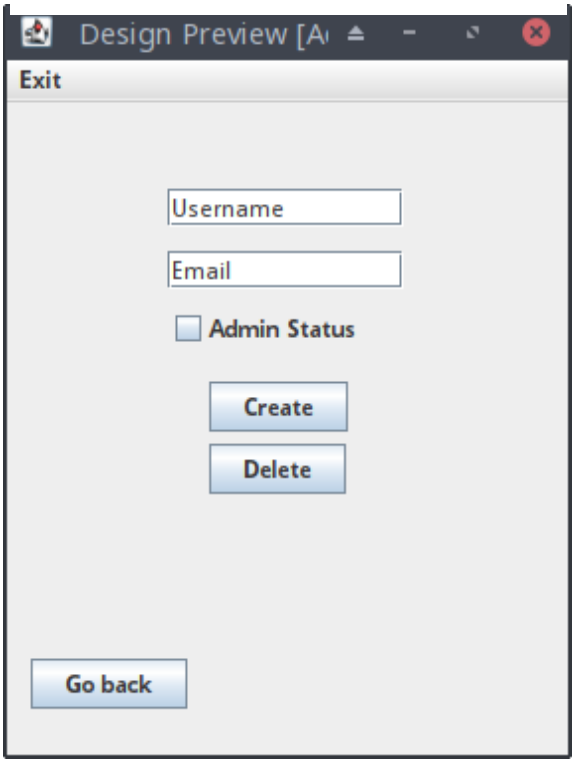


Figura 6 Criar Conta

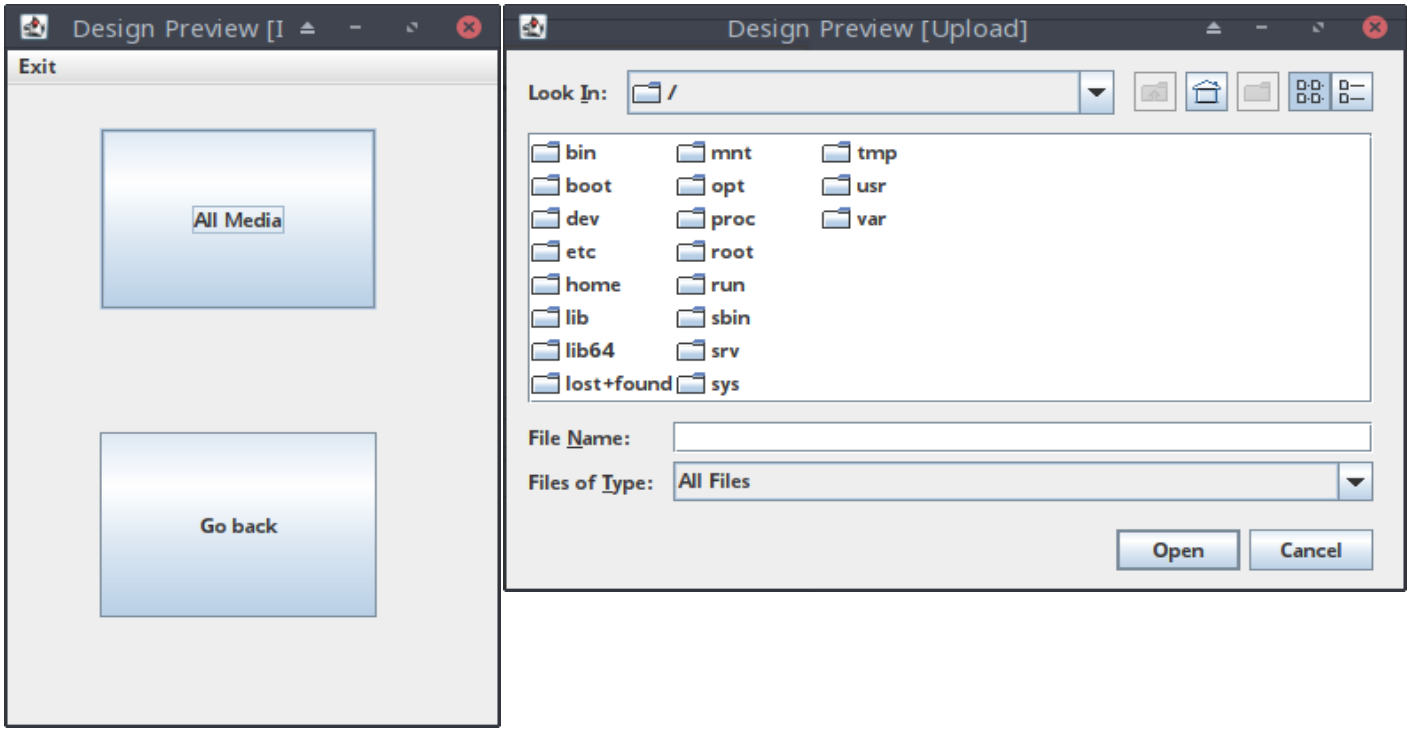


Figura 7 Menu convidado e upload de media

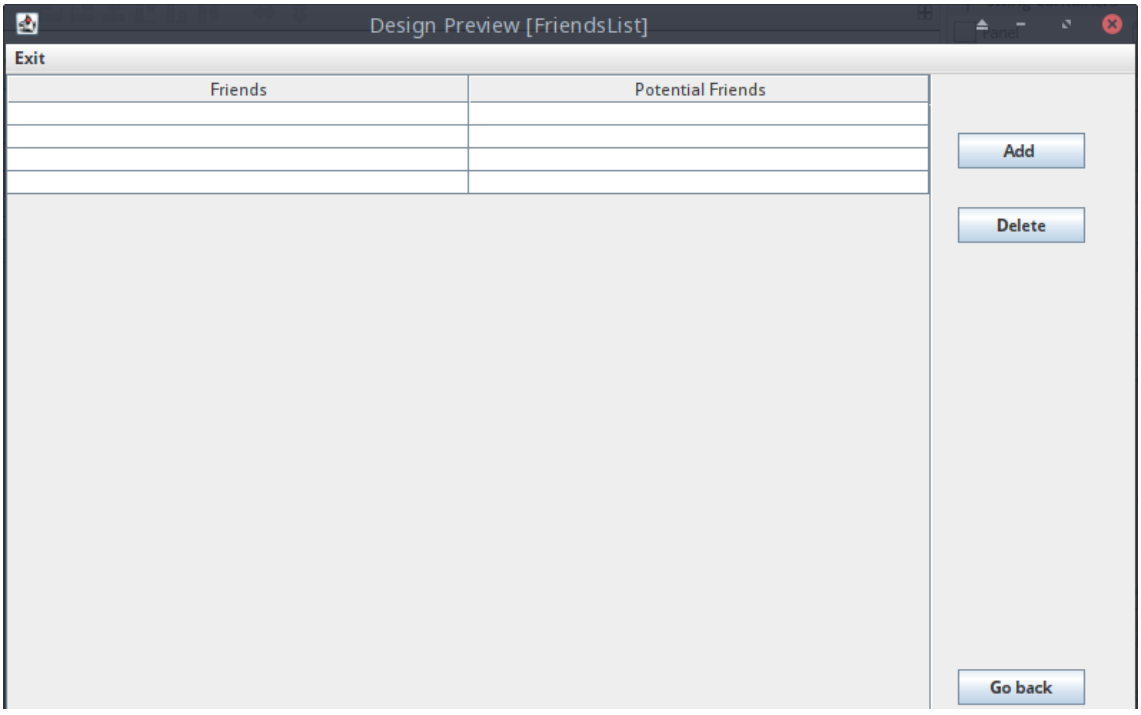


Figura 8 Lista de Amigos e potenciais amigos

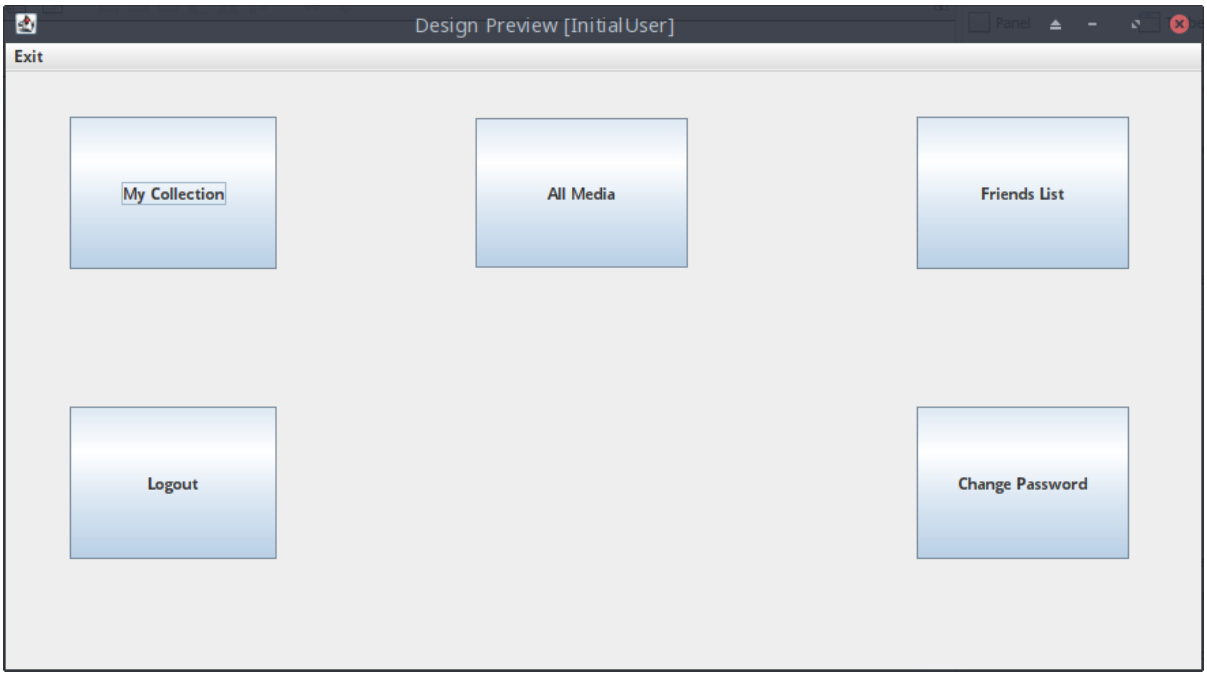


Figura 9 Menu do usuário

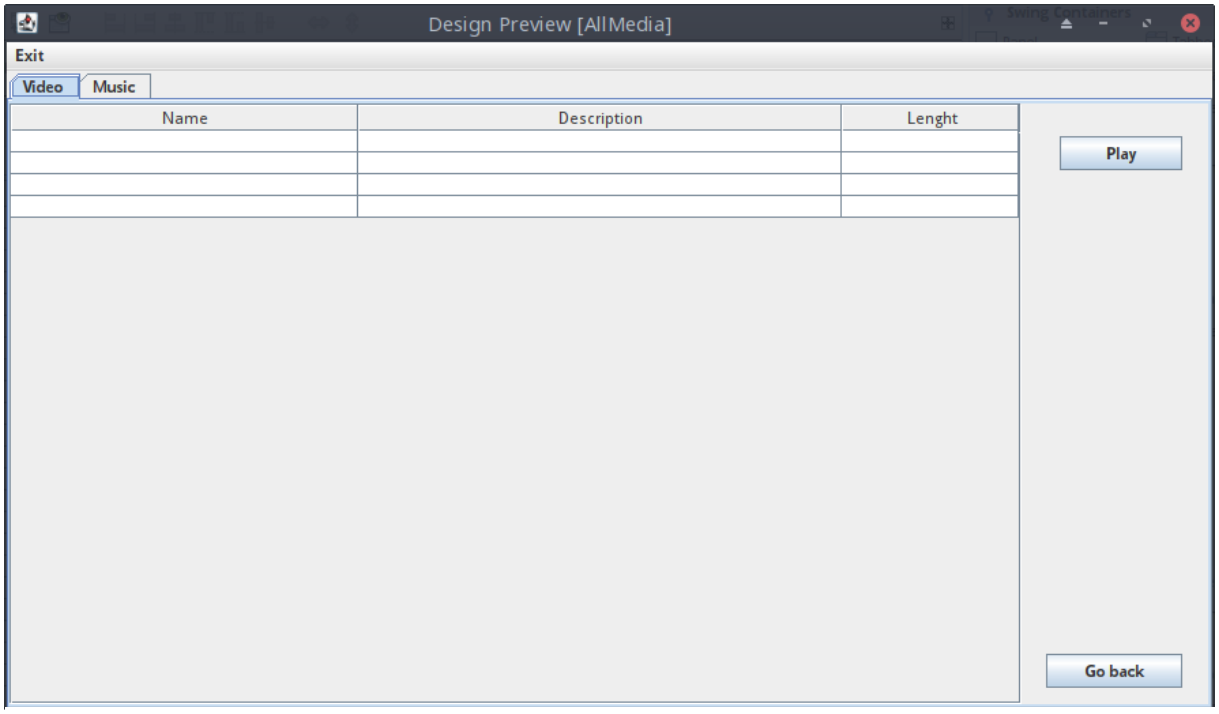


Figura 10 Vídeos de todas as coleções

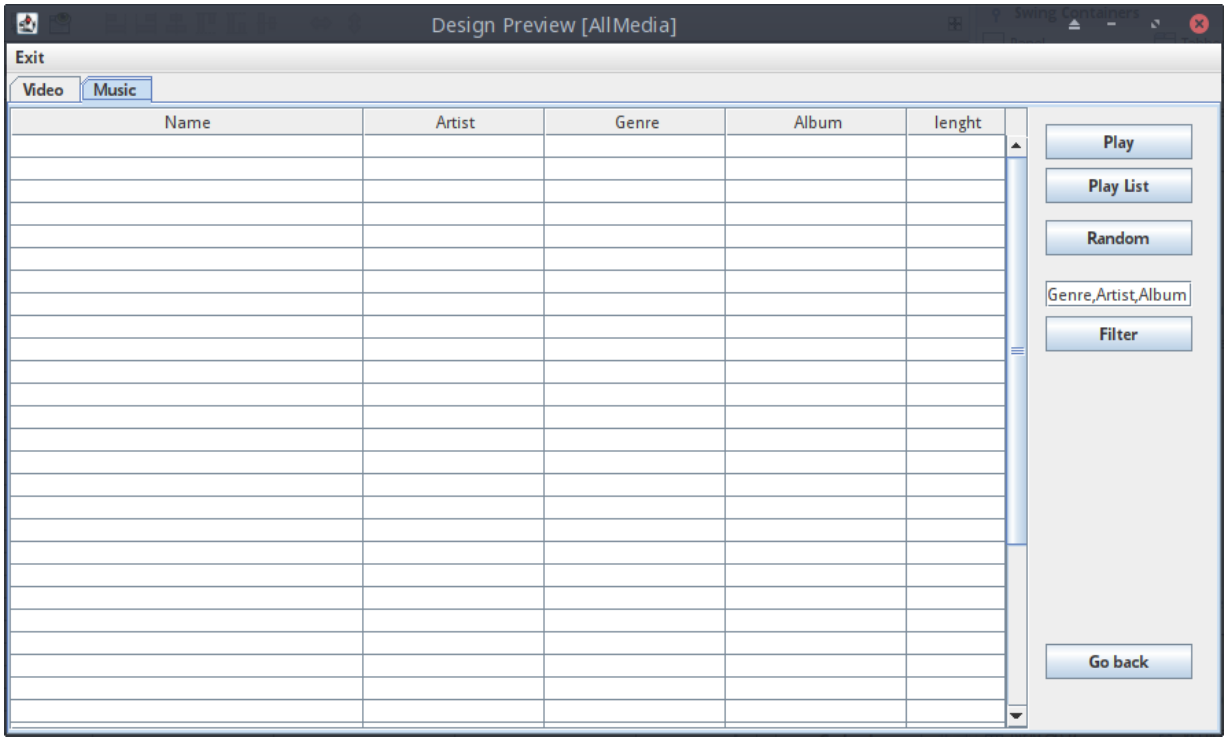


Figura 11 Músicas de todas as coleções

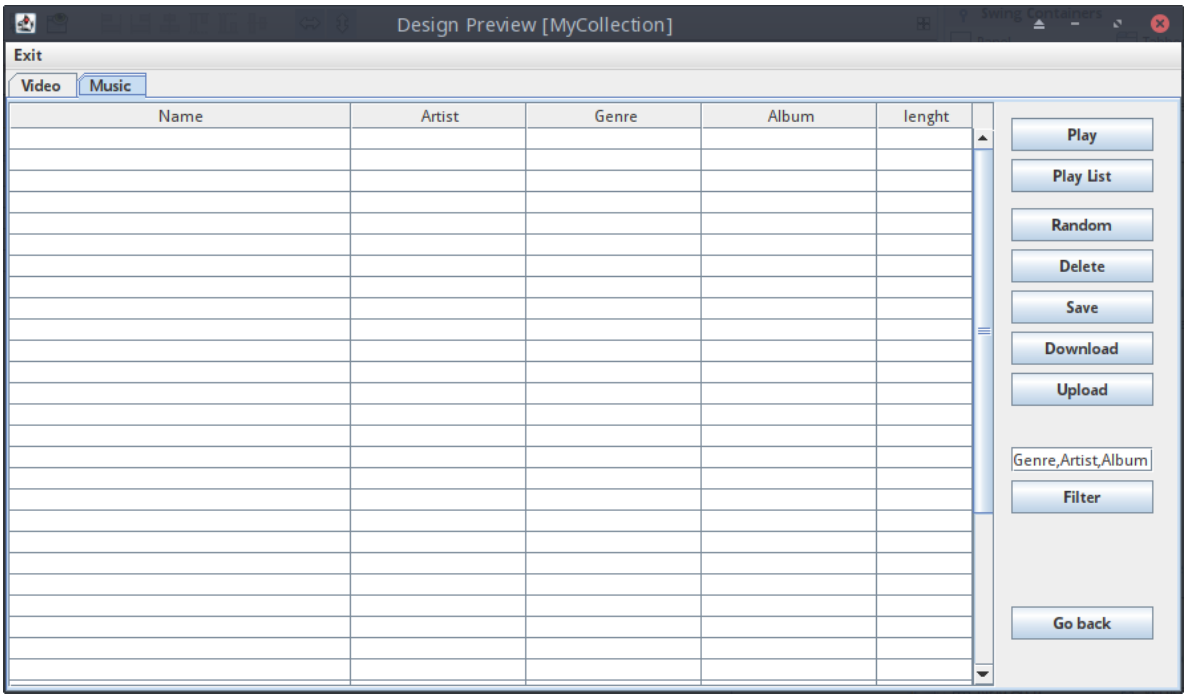


Figura 12 Músicas das minhas coleções

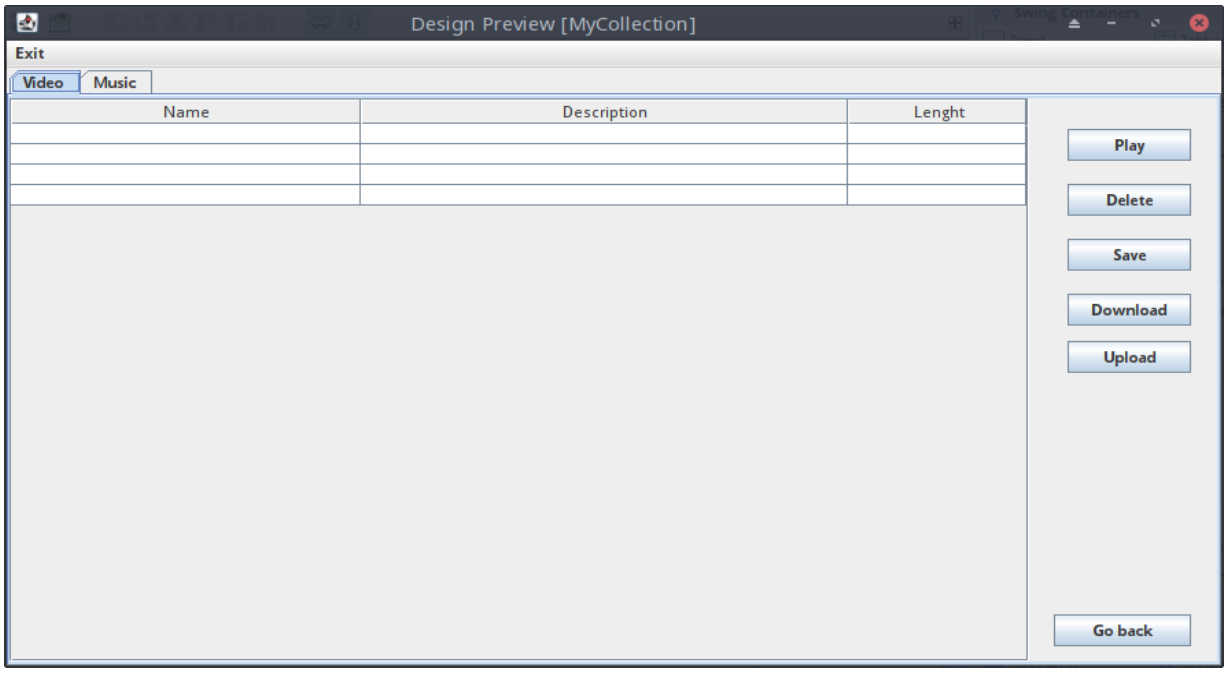


Figura 13 Vídeos nas minhas coleções

6. Alterações na Interface final

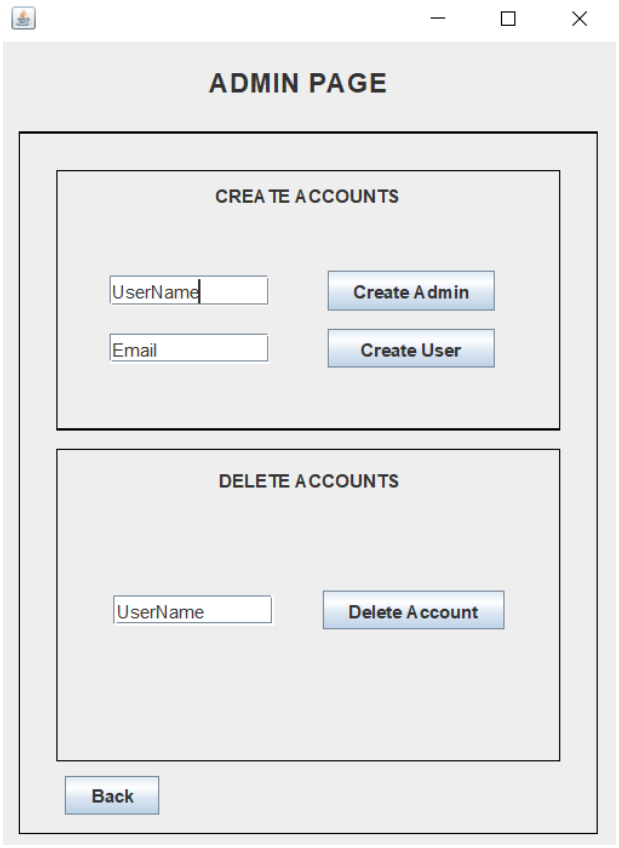


Figura 14 Página de Admin

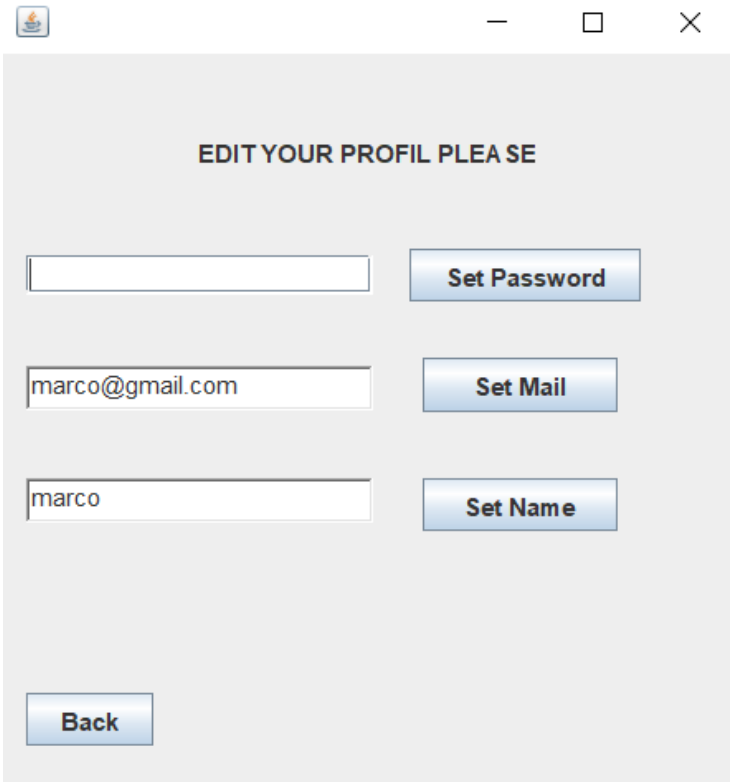


Figura 15 Editar Perfil

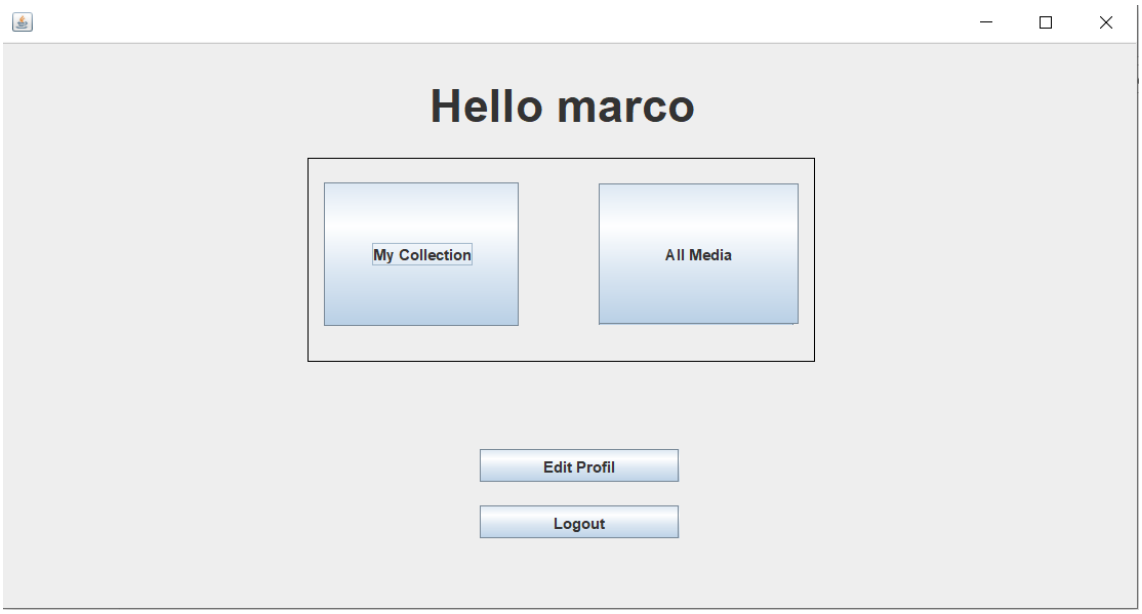


Figura 16 Menu de utilizador

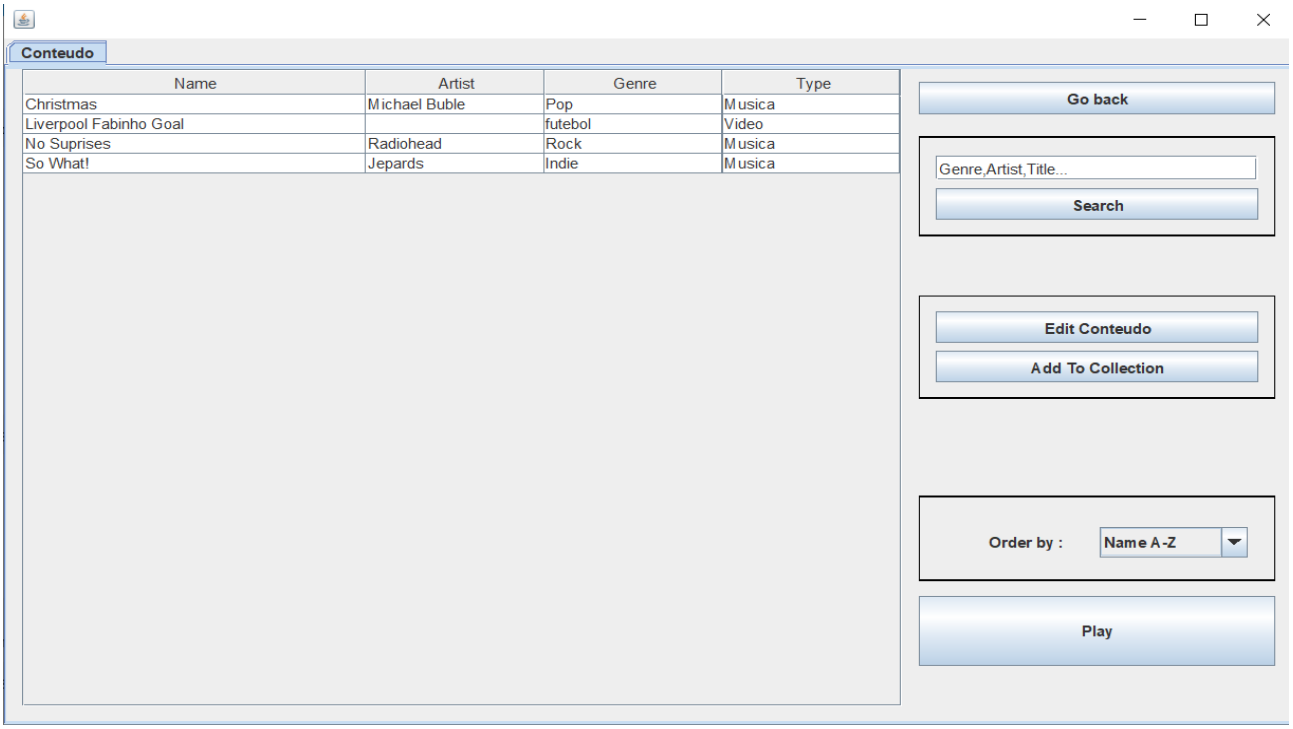


Figura 17 Conteúdo geral

Figura 18 Editar conteúdo

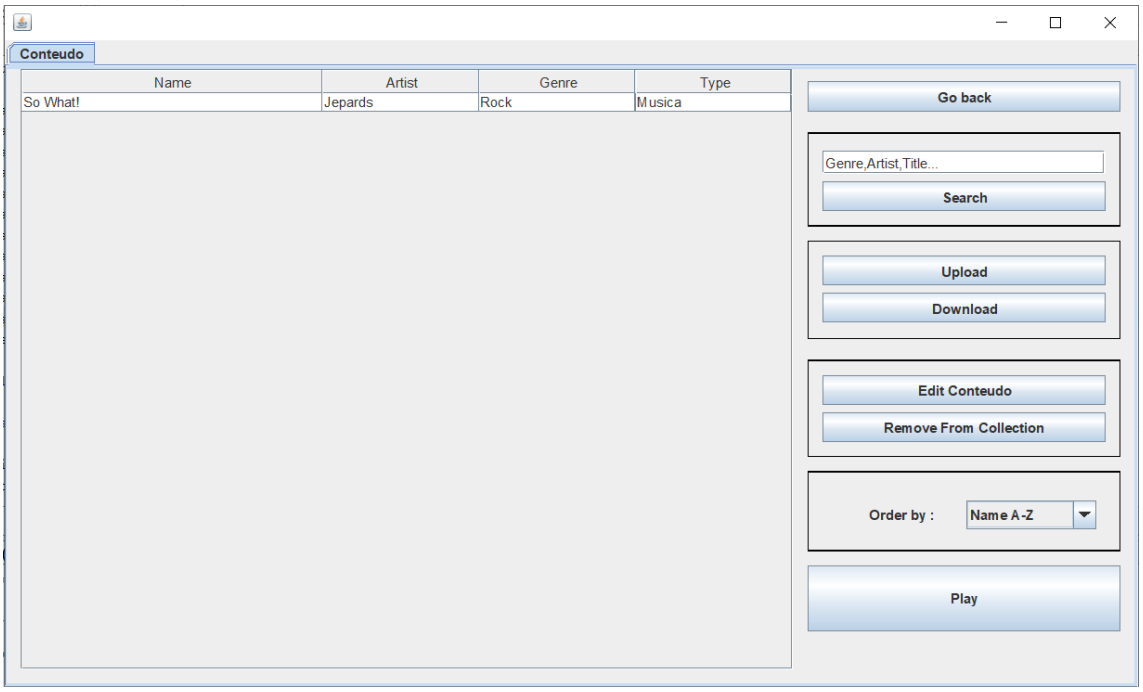


Figura 19 Biblioteca privada

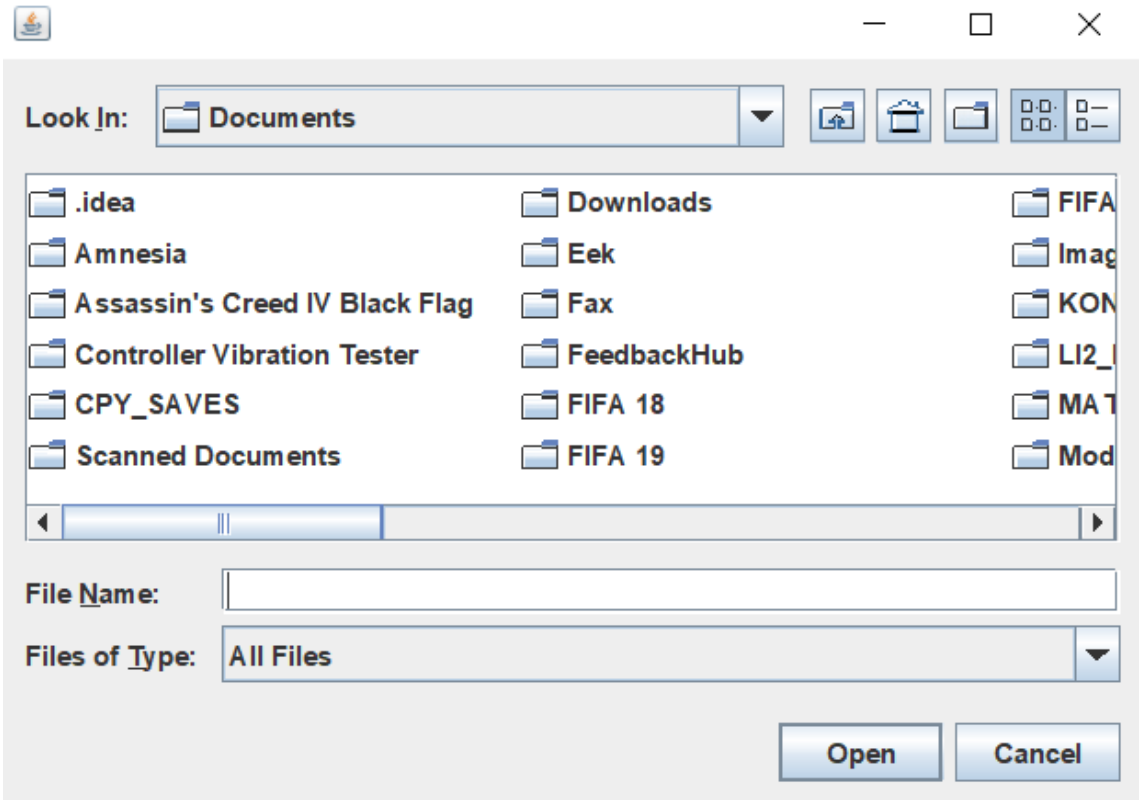


Figura 20 Upload e Download

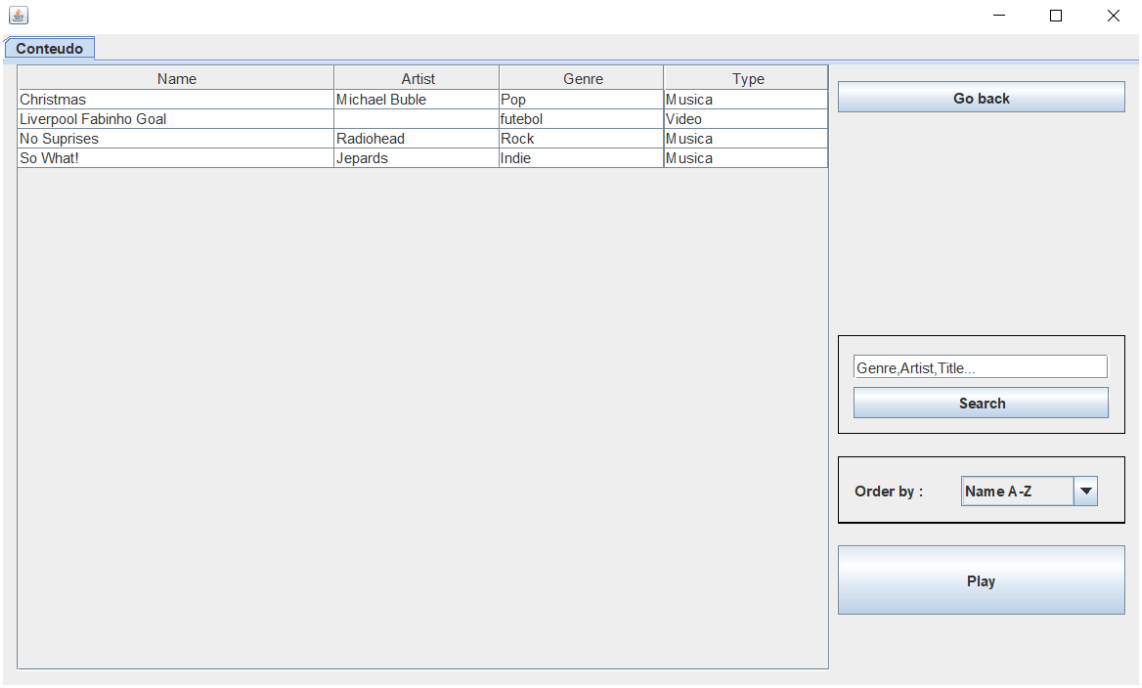


Figura 21 Apresentação de conteúdo para o Guest

7. Diagramas de Pacotes

O Diagrama de Package permite-nos agrupar as classes em pacotes de acordo com o seu papel no desenvolvimento da aplicação, identificando-se as relações de dependência entre estes, isto torna-se importante devido ao elevado número de classes em que a gestão das mesmas nem sempre é fácil.

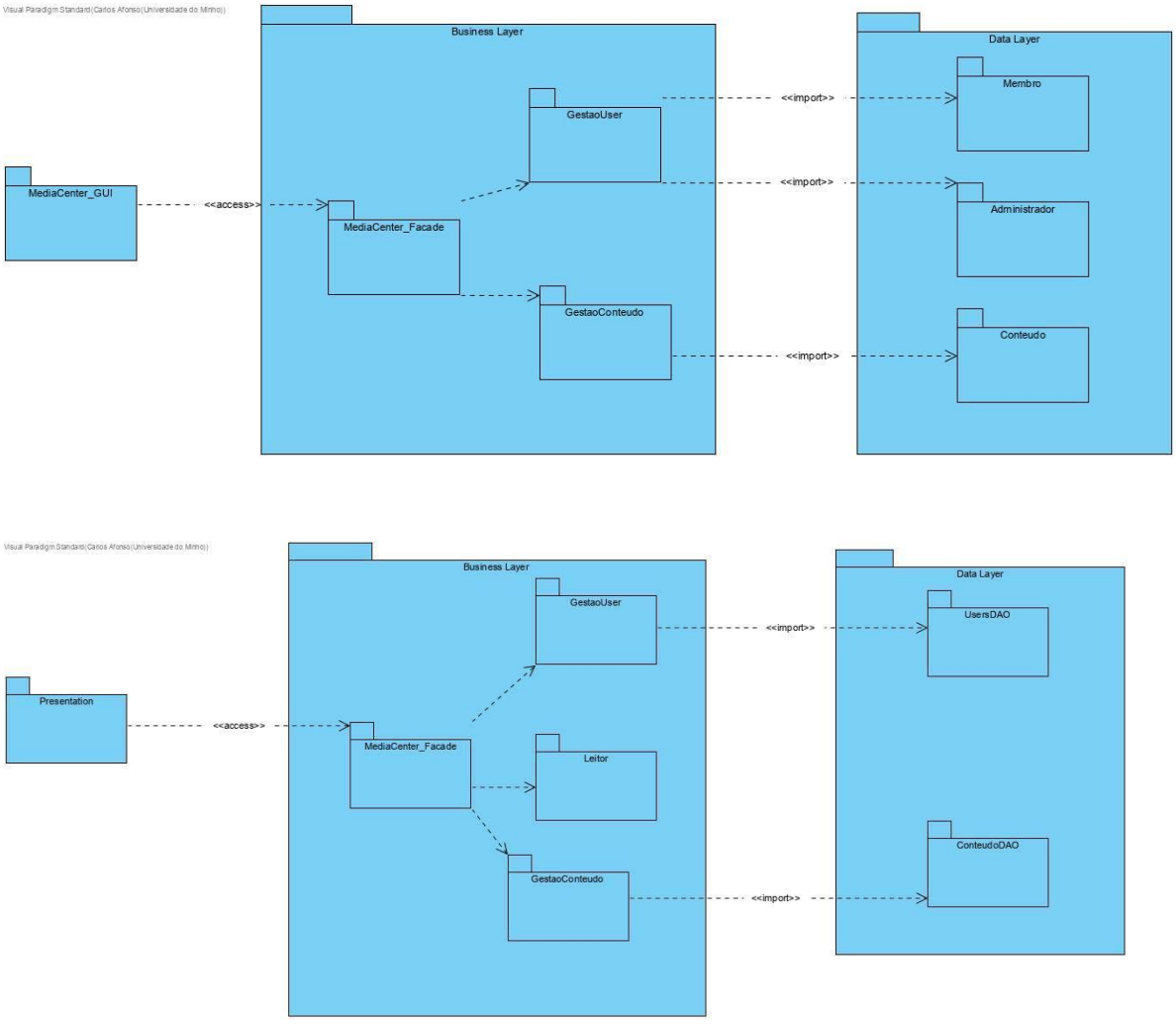


Figura 22 Diagramas de pacotes inicial e final

8. Diagrama de Classes

Os Diagramas de Classes são a base de um qualquer sistema orientado a objetos, apresentando as mais variadas classes que correspondem a conjuntos de objetos com a mesma estrutura de dados e comportamento.

Em cada classe estão definidos os atributos e as operações que correspondem às variáveis de instância de cada classe e aos métodos nela definidos.

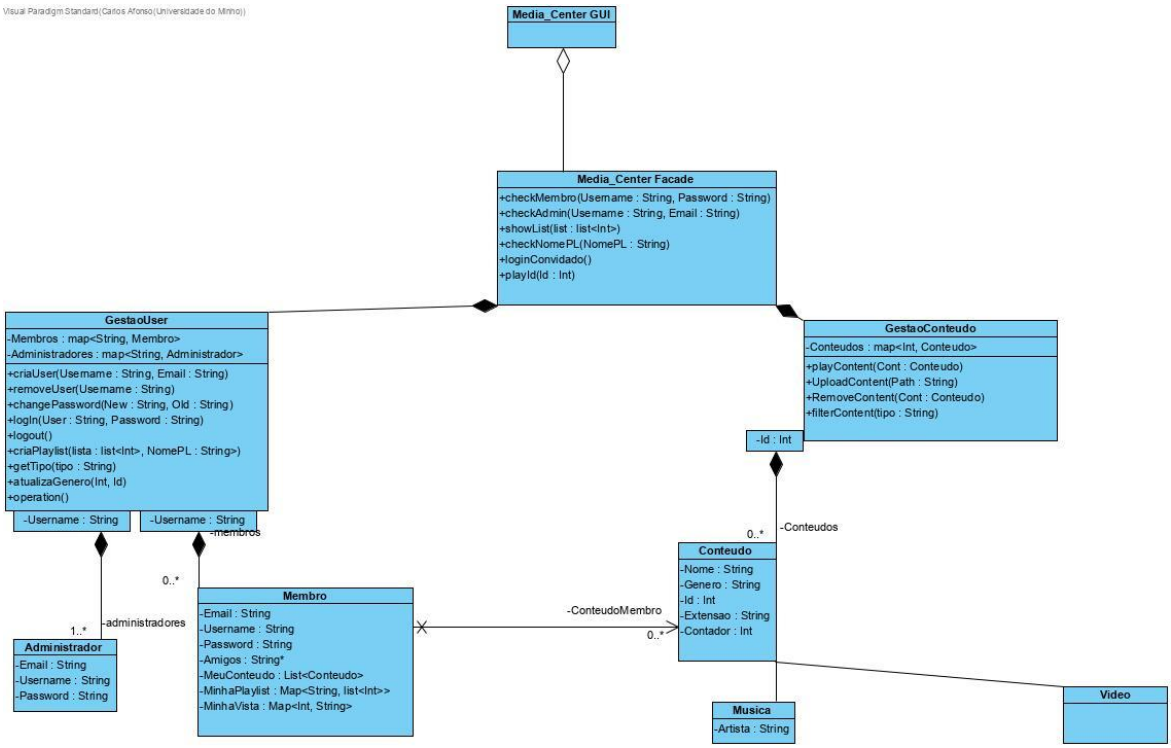


Figura 23 Diagrama de classes

9. Diagramas de sequência

Os diagramas de sequência representam a sequência de processos (mais especificamente, de mensagens passadas entre objetos) num programa. Como o nosso projeto tem uma grande quantidade de métodos em classes diferentes, pode ser difícil determinar a sequência global do comportamento. O diagrama de sequência representa essa informação de uma forma simples e lógica.

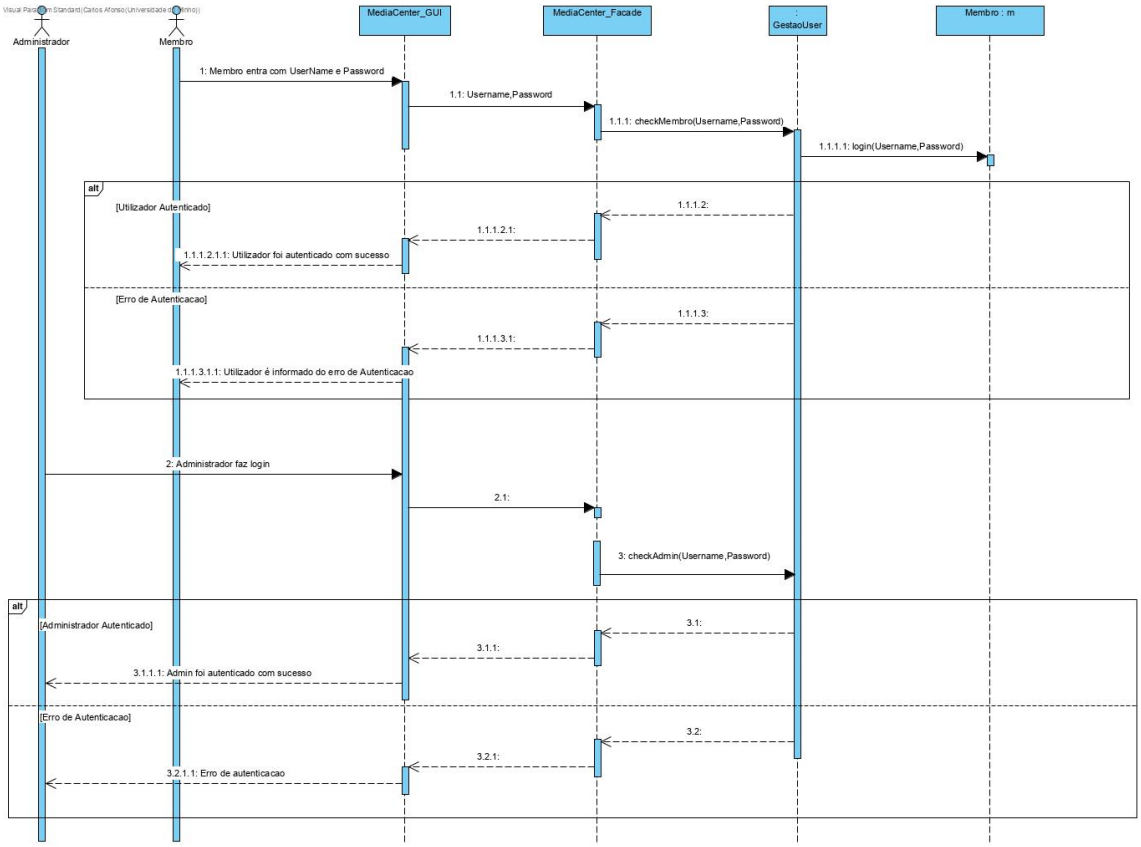


Figura 24 Login User

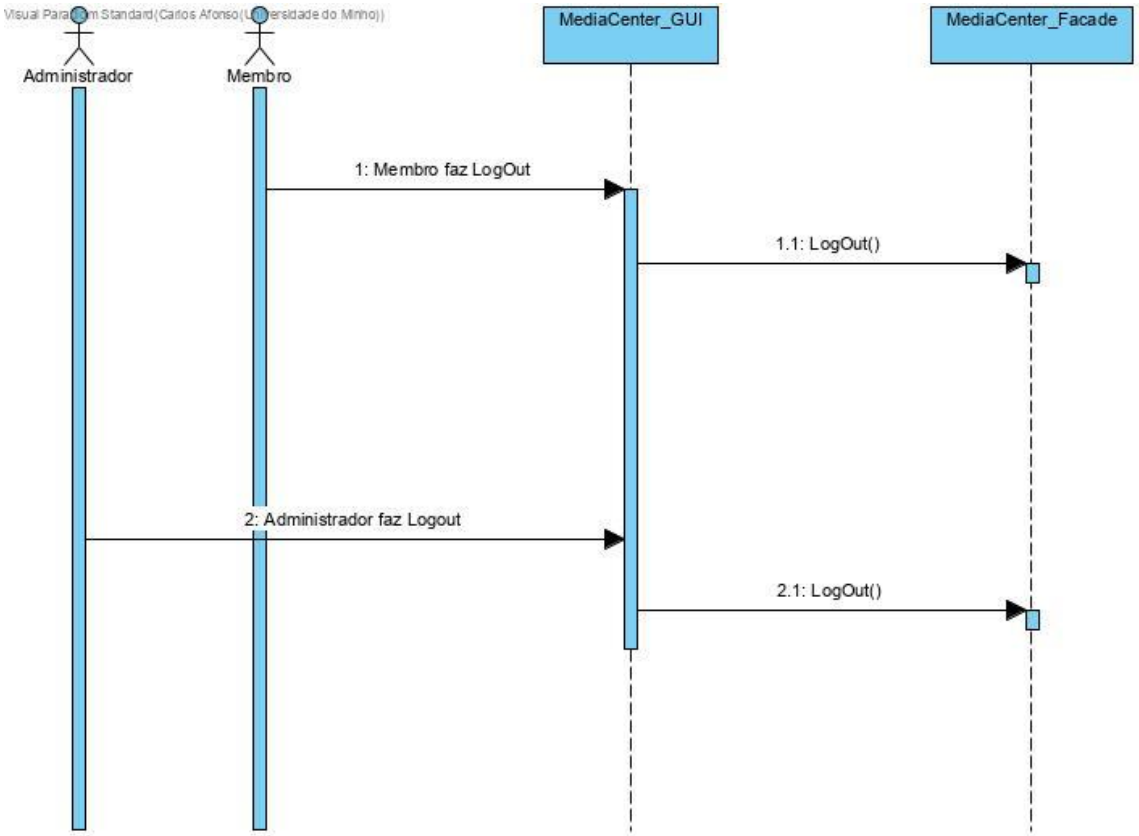


Figura 25 Logout

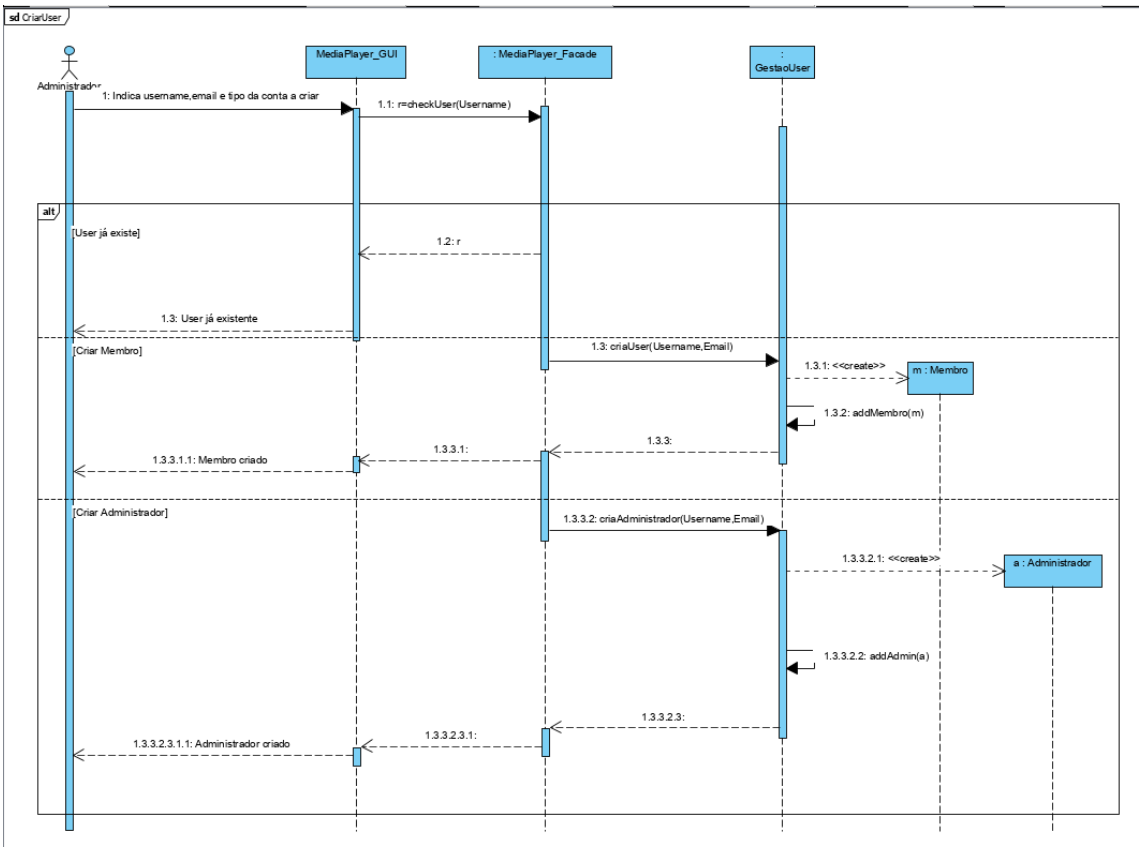


Figura 26 Criar User

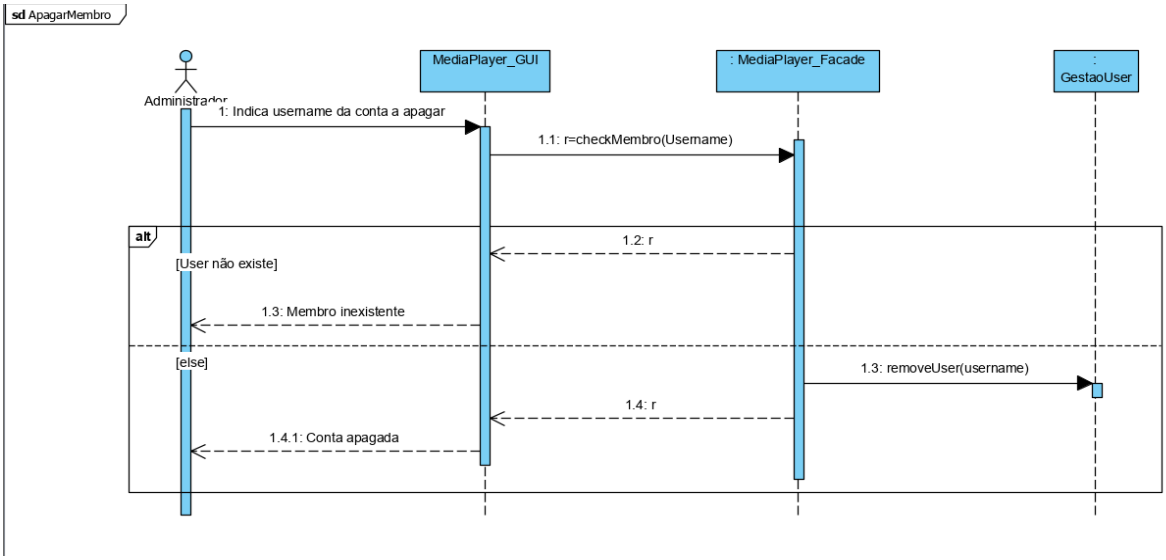


Figura 27 Apagar membro

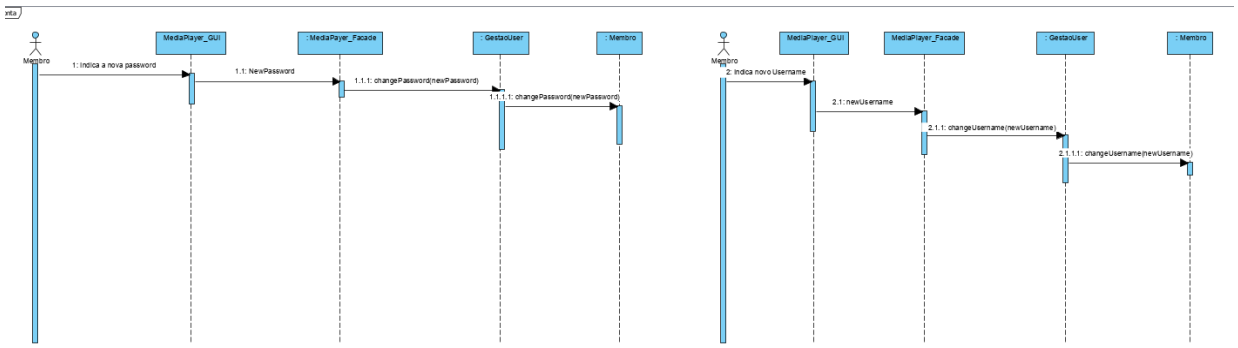


Figura 28 Editar conta

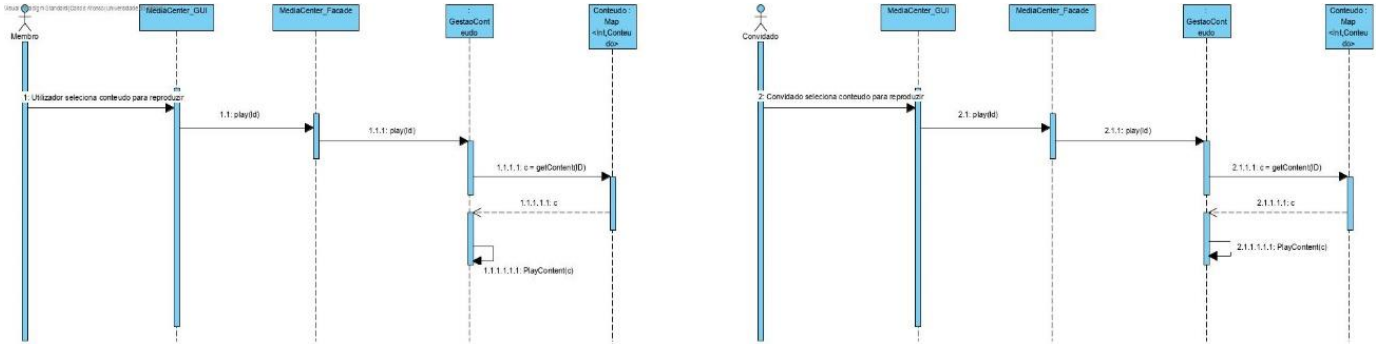


Figura 29 Reproduzir conteúdo

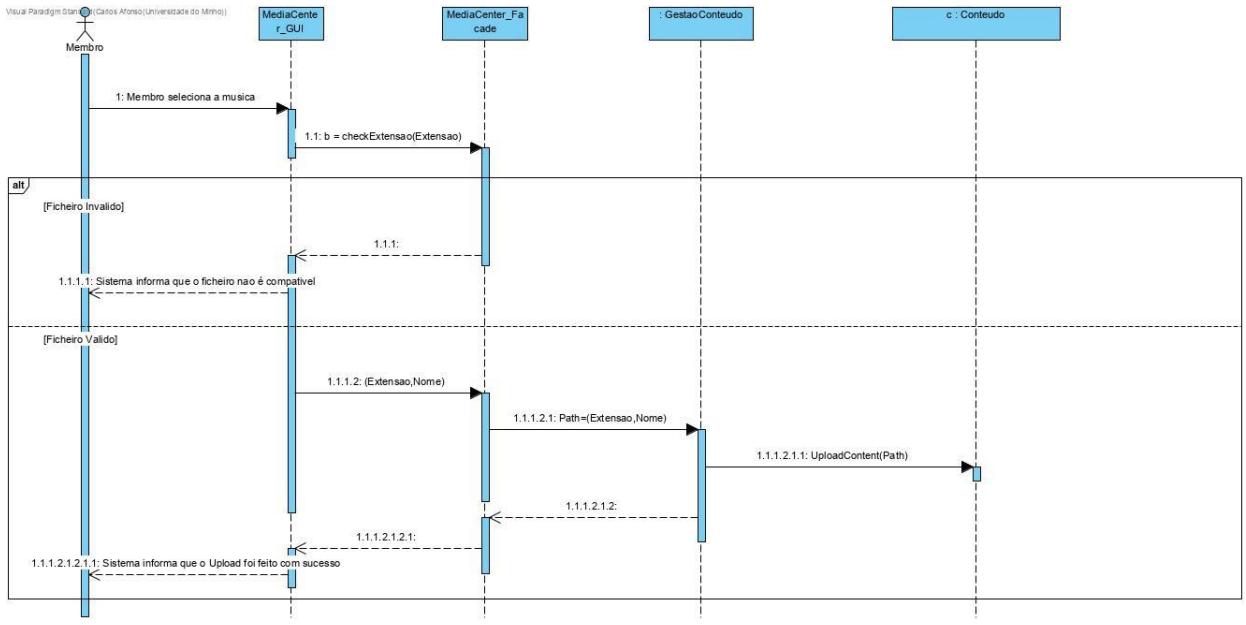


Figura 30 Upload conteúdo

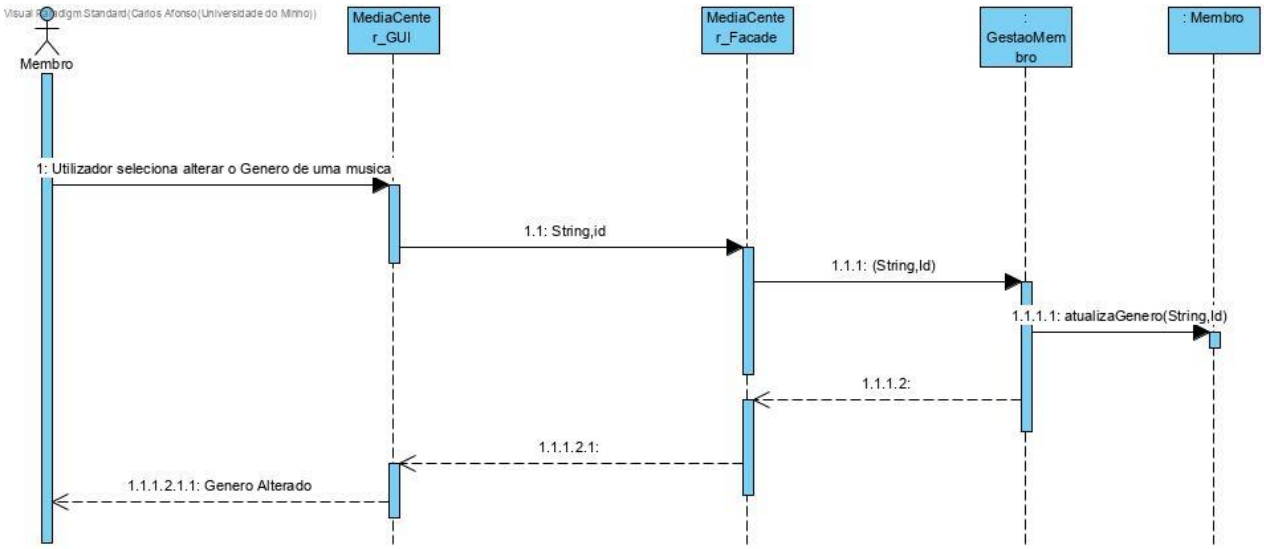


Figura 31 Alterar categoria

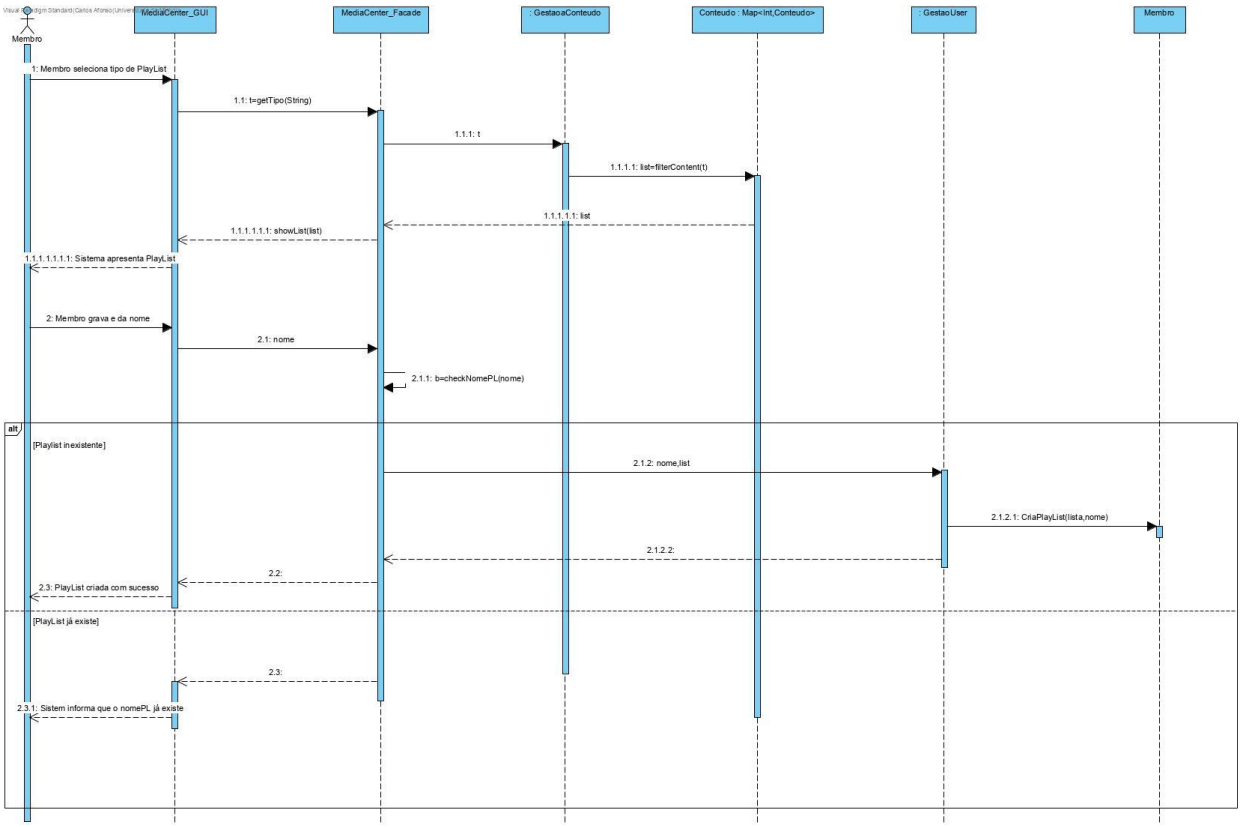


Figura 32 Criar playlist

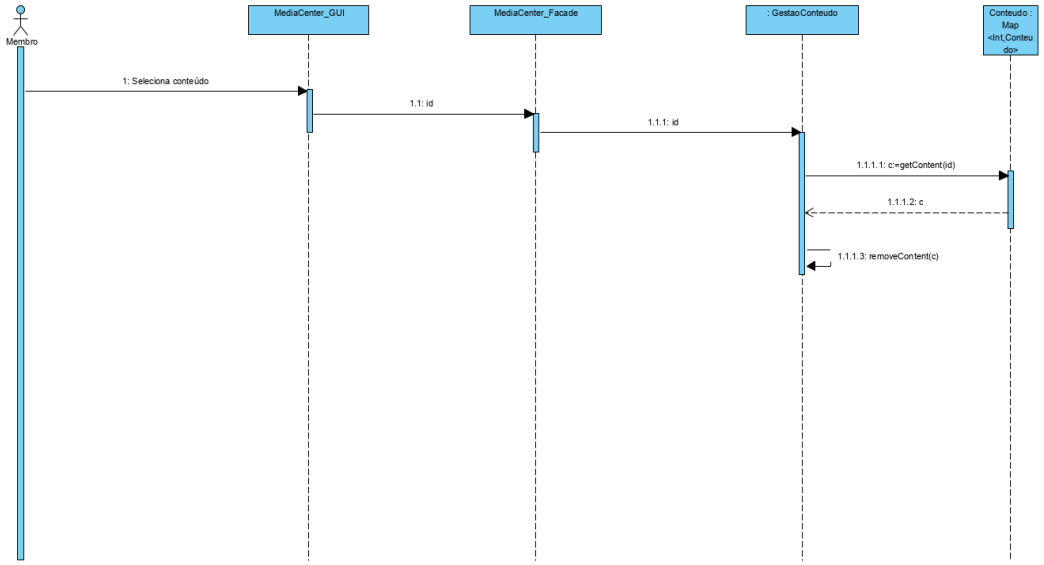


Figura 33 Apagar conteúdo

10. Diagrama de Classes com DAO's

Neste Diagrama de Classe, temos presentes as classes da camada de persistência de dados (*data layer*) que se encarregam de guardar os mais diversos dados na aplicação.

Este diagrama surgiu da adaptação feita ao Diagrama de Classes original que resultou na substituição dos *maps* por DAO´s. Estas classes garantem a persistência dos dados, a independência entre a camada de negócio e a camada de dados, fazendo com que a implementação de uma não dependa da outra o que é vantajoso quando queremos alterar algo em apenas uma das camadas.

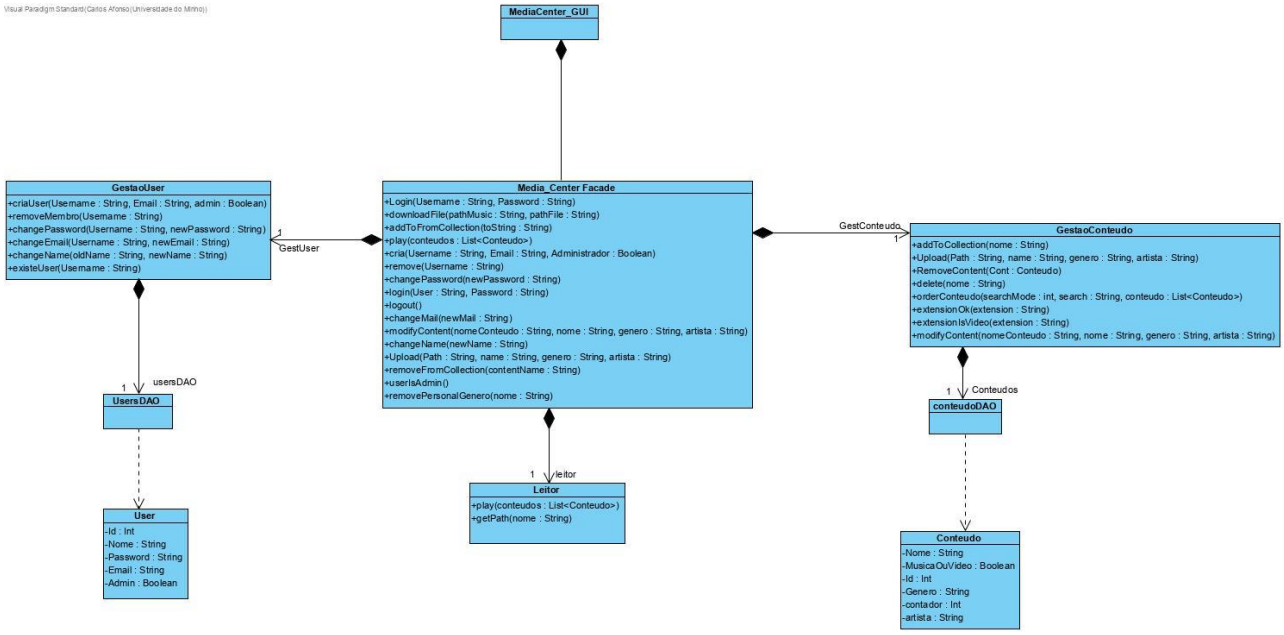


Figura 34 Diagrama de Classes com DAOs

11. Diagramas de Sequências com DAOs

Nesta fase do projeto os diagramas de sequência foram alterados com a inclusão dos DAOs, tendo em conta o novo diagrama de classes.

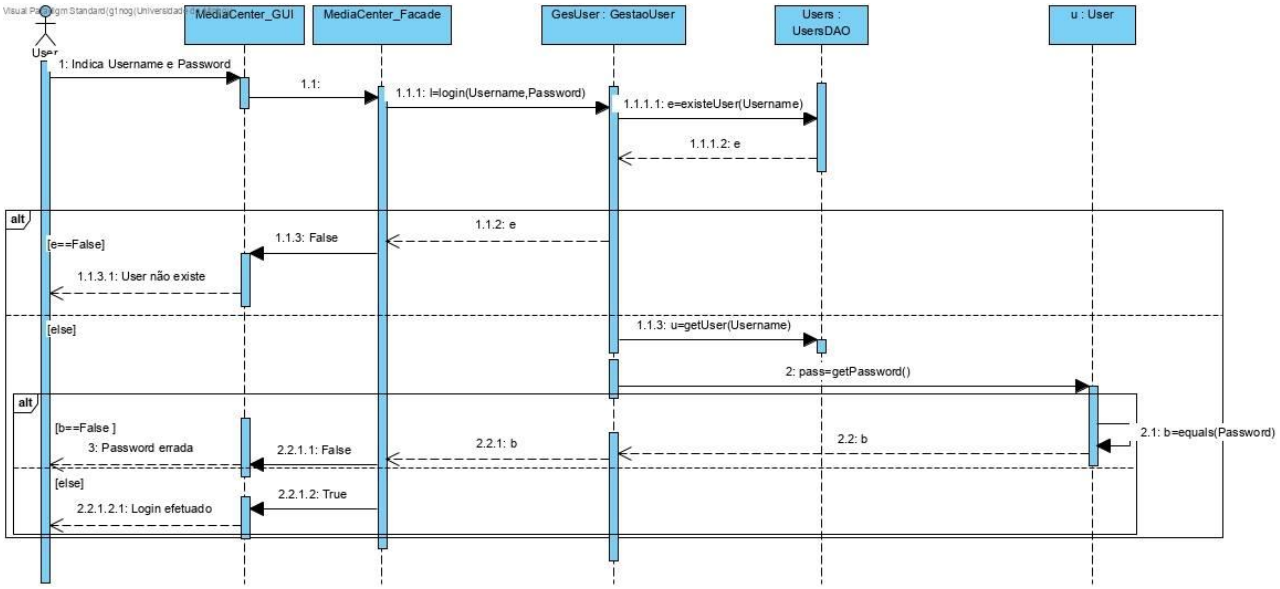


Figura 35 Login

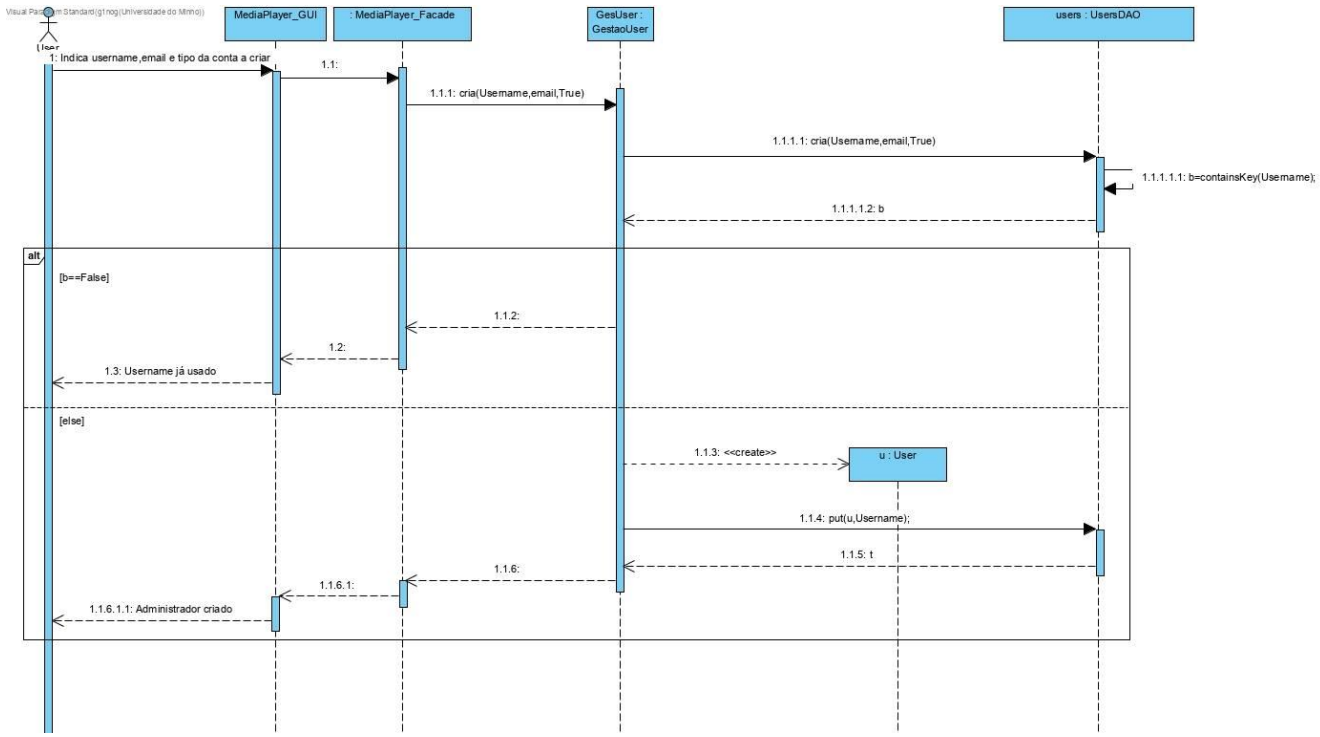


Figura 36 Criar Admin

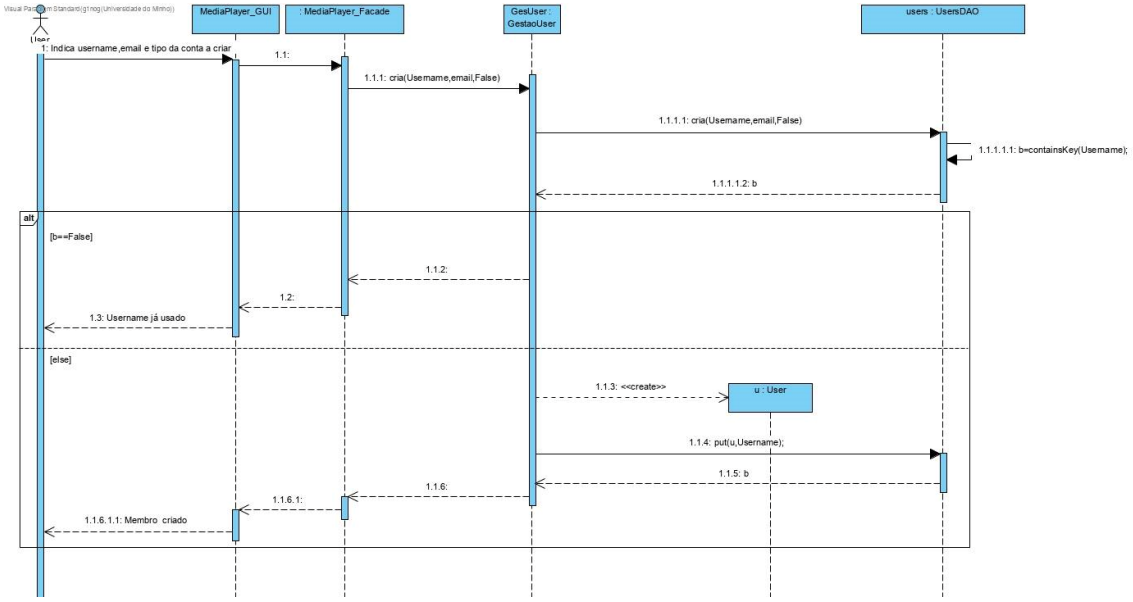


Figura 37 Criar Membro

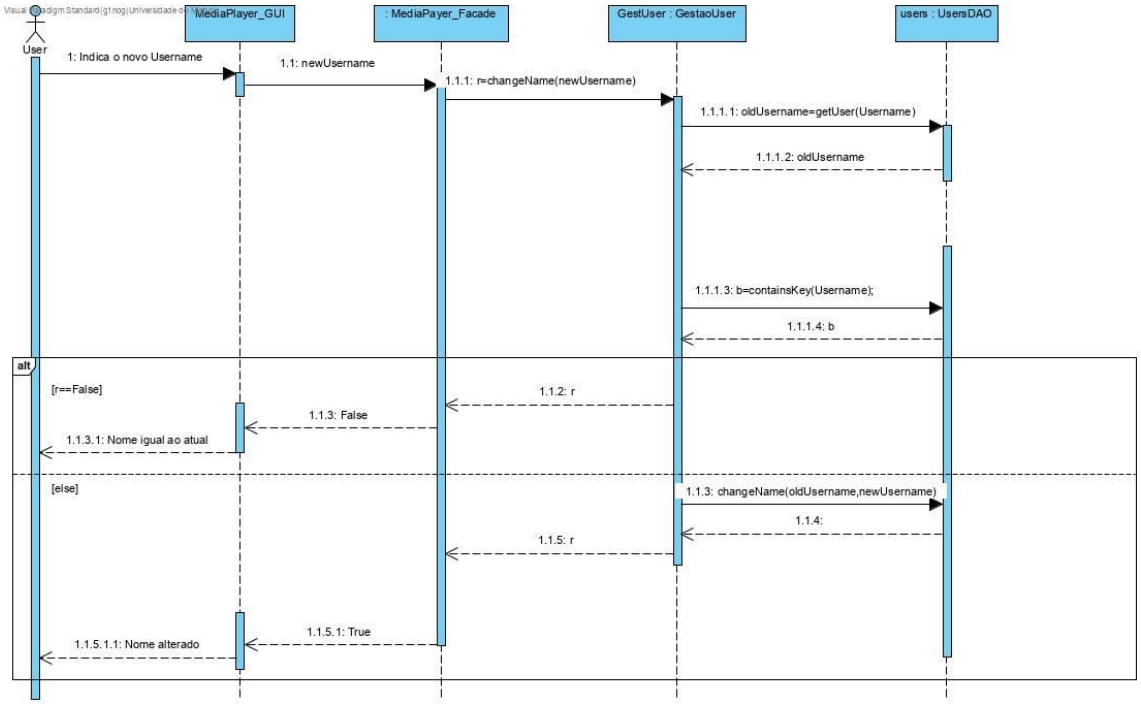


Figura 38 Alterar nome do User

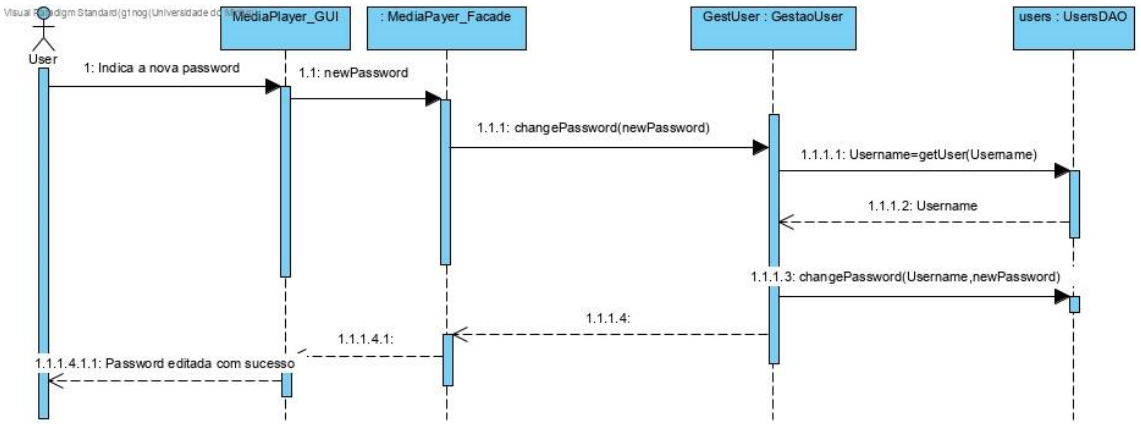


Figura 39 Alterar Password

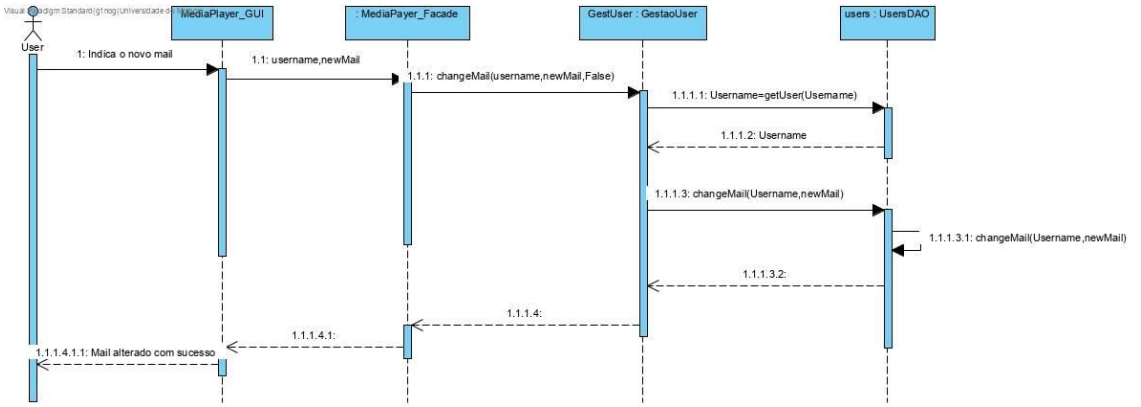


Figura 40 Alterar e-mail

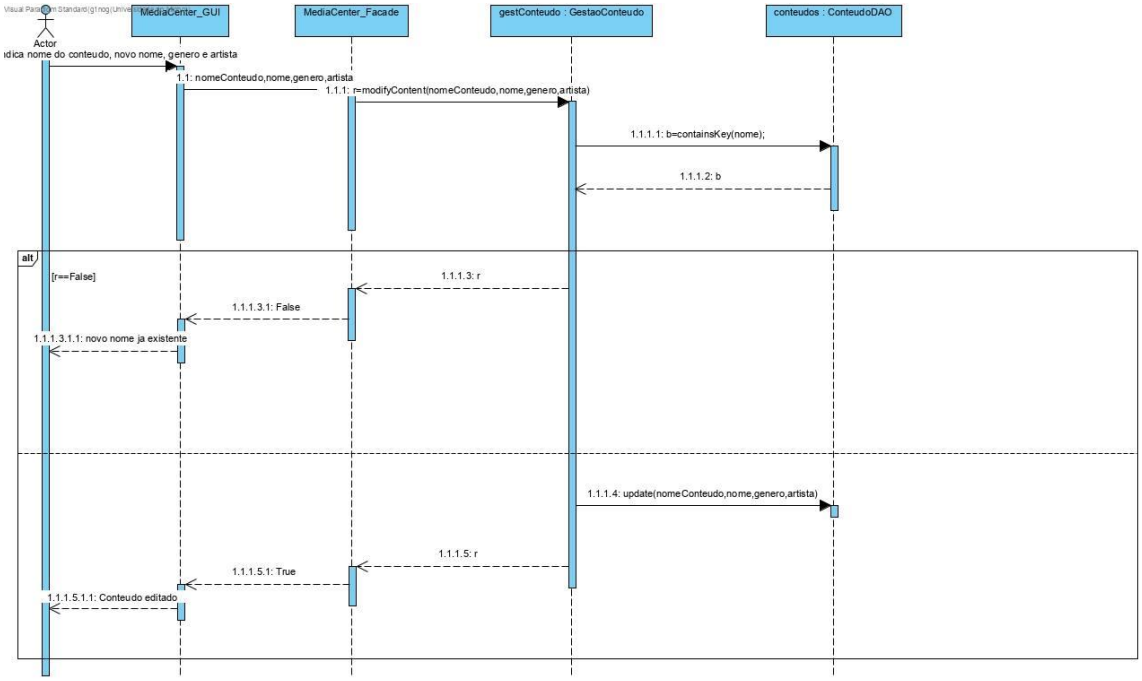


Figura 41 Editar Conteúdo

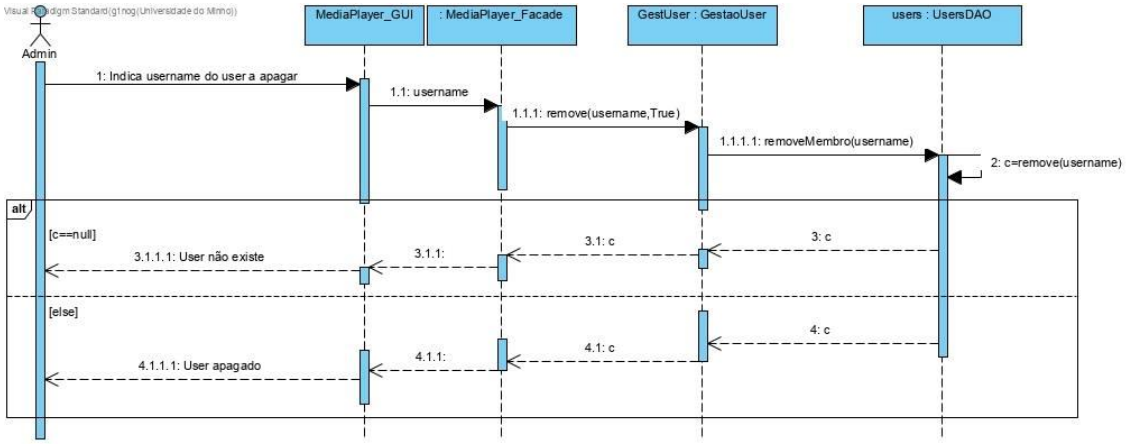


Figura 42 Apagar User

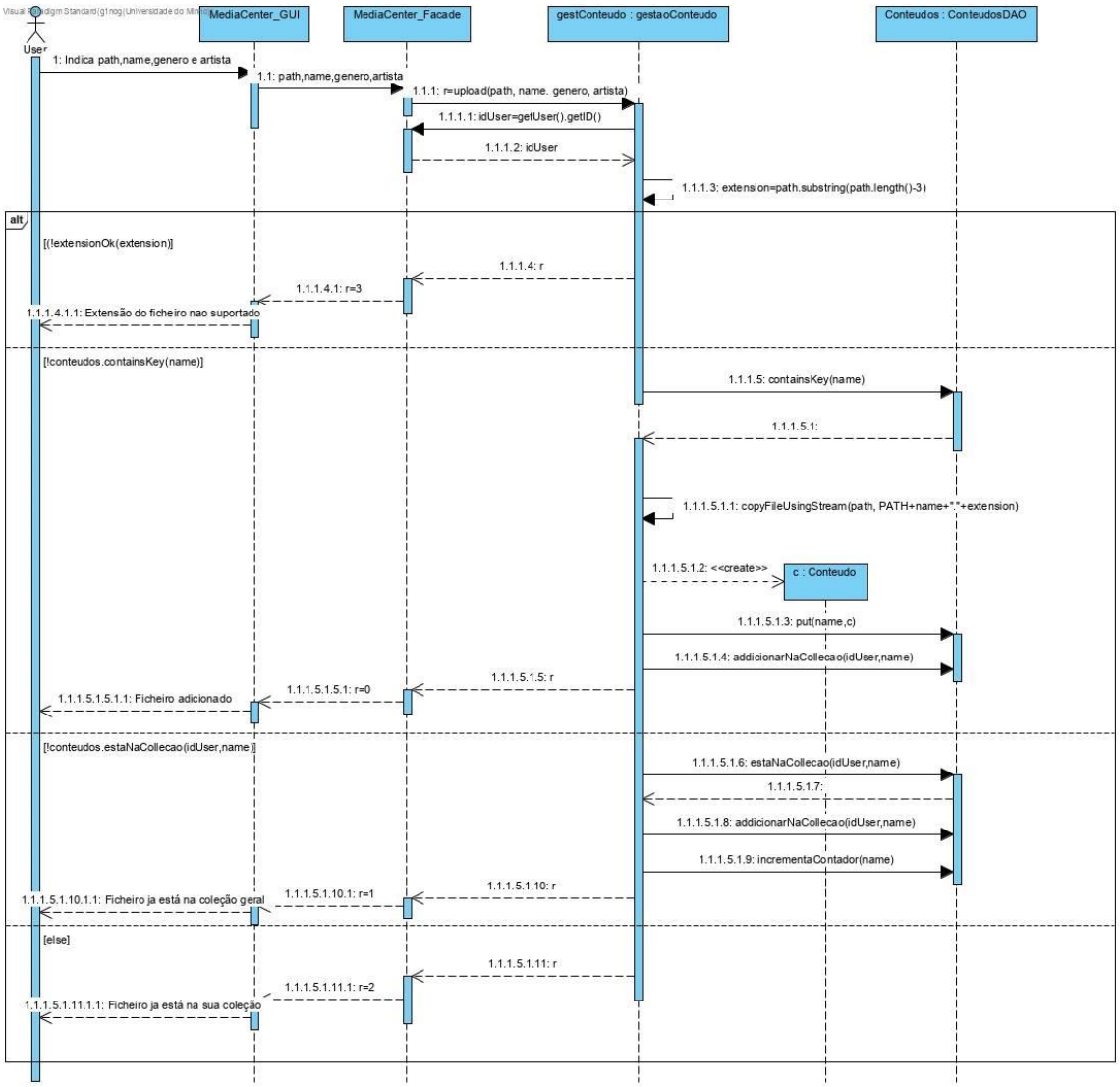


Figura 43 Upload

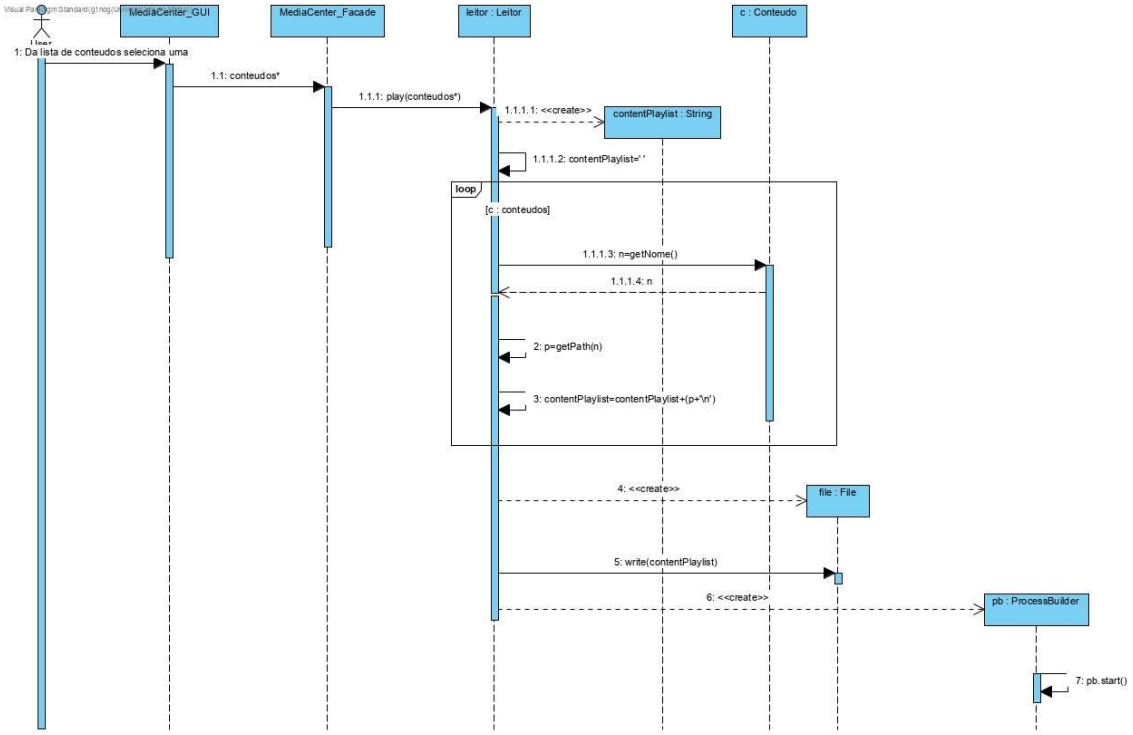


Figura 44 Play

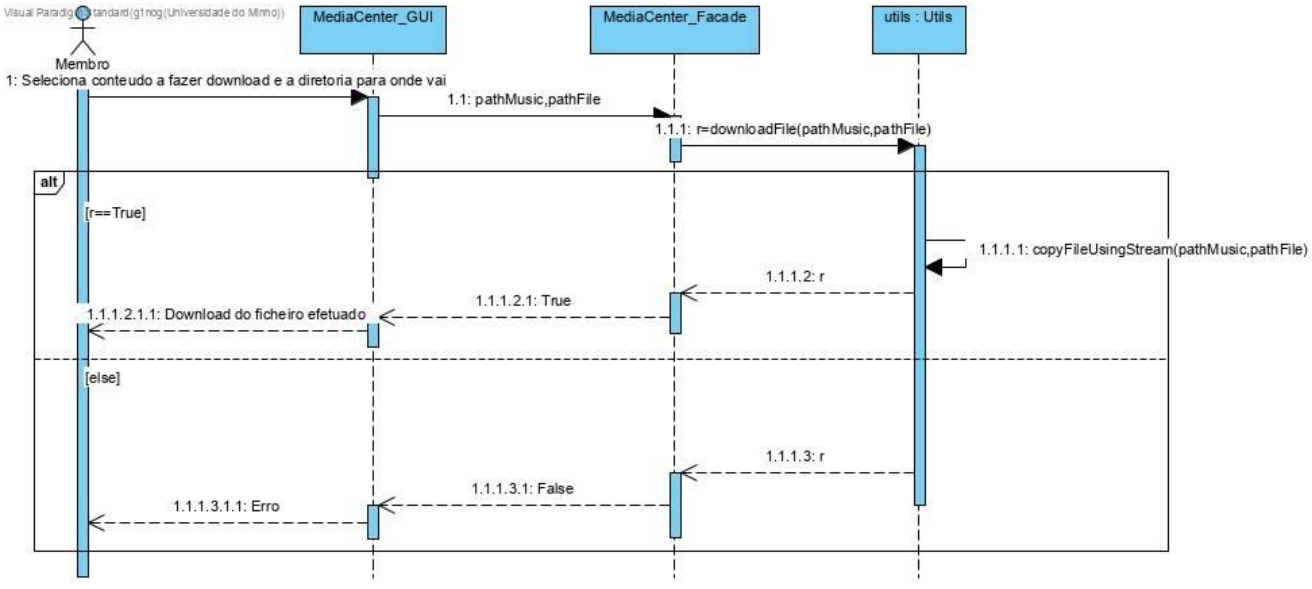


Figura 45 Download

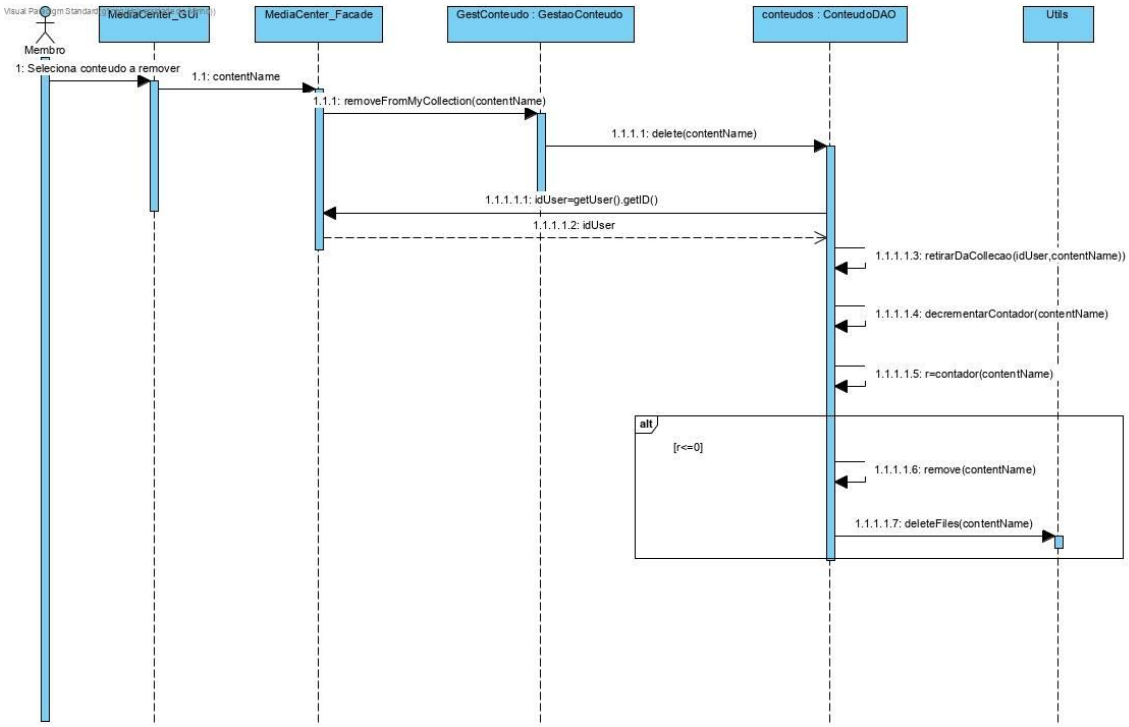


Figura 46 Apagar conteúdo

12. Base de Dados desenvolvida em SQL

O auxílio da base de dados permite isolar os dados de forma a que a aplicação não fique dependente da origem ou estrutura onde os dados estão armazenados. Isto permite-nos que métodos acedam e guardem dados que, em alternativa, seriam voláteis.

No nosso projeto temos as classes UserDAO e conteudoDAO, estas classes são necessárias para guardar, consultar e remover dados relativos a aplicação.

Toda e qualquer modificação feita, altera só as classes relativas à execução que for pretendida, não alterando outras classes.

Como não fazemos uso de querys como joins a nossa base de dados não tem ligações, algo que poderia ser modificado, pois, melhoraria a integridade da base de dados.

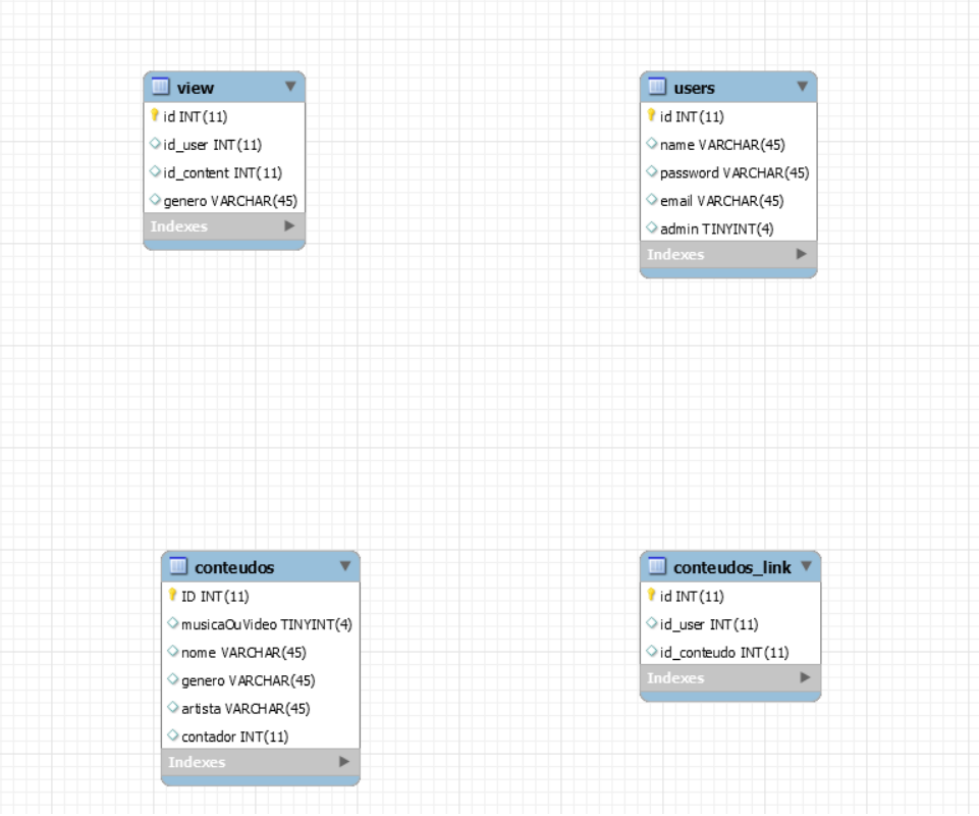


Figura 47 Base de Dados MySQL

13. Implementação em Java

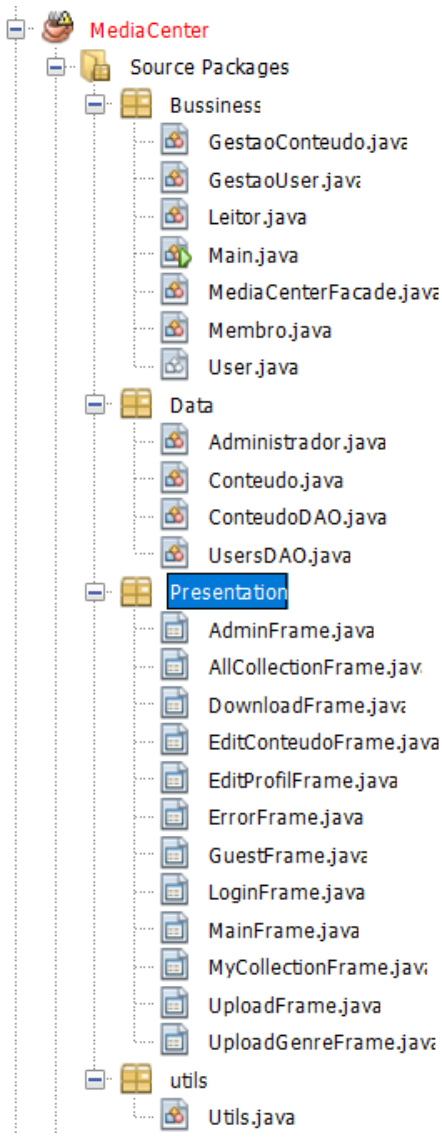


Figura 48 Implementação

14. Conclusões

Nesta fase final do projeto, continuamos a elaborar o programa que tínhamos planeado baseando-nos no modelo e diagramas que tínhamos contruído anteriormente.

As duas primeiras fases do trabalho foram relevantes, pois a elaboração dos modelos e dos diagramas facilitaram-nos todo o processo de implementação, tendo em conta que já tínhamos uma base e ideias já estruturadas. No entanto, tivemos de nos adaptar às dificuldades com que nos deparamos na altura de implementação de código e então fomos revendo e alterando modelos anteriores sem nunca fugirmos àquilo que tínhamos em mente numa fase primordial.

O Visual Paradigm foi uma ferramenta muito útil na construção dos mais diversos modelos e diagramas que suportaram a codificação. A utilização do modelo de camadas foi também essencial no desenvolvimento do projeto no sentido em que contribuiu fortemente para um trabalho mais sólido.

Por final, podemos dizer que foi um projeto bastante enriquecedor que nos deu uma visão daquilo que será um bocado do nosso futuro, aprendemos ferramentas e ganhamos hábitos importantes para o desenvolvimento de uma aplicação com planeamento, sem saltar etapas.