

UNIVERSIDADE DO MINHO



Mestrado Integrado em Engenharia Informática

DESENVOLVIMENTO DE SISTEMAS DE SOFTWARE – MEDIA CENTER

dezembro, 2019



GRUPO 12

Ana Afonso, A85762 | Carolina Cunha, A80142 | Hugo Faria, A81283 | João Diogo Mota, A80791 | Pedro Silva, A82522

ÍNDICE

ÍNDICE DE DIAGRAMAS	2
ÍNDICE DE FIGURAS	3
BREVE DESCRIÇÃO DO ENUNCIADO PROPOSTO	4
MODELAÇÃO	5
MODELO DE DOMÍNIO	5
MODELO DE USE CASE	7
ESPECIFICAÇÃO USE CASES	8
DIAGRAMAS DE SEQUÊNCIA DE SISTEMA	14
DIAGRAMA DE PACKAGES	20
DIAGRAMA DE CLASSES	21
DIAGRAMAS DE SEQUÊNCIA COM SUBSISTEMAS	22
DIAGRAMAS DE SEQUÊNCIA DE IMPLEMENTAÇÃO	26
OBJECT RELATIONAL MAPPING - ORM	29
DIAGRAMA DE CLASSES COM DAO	29
DIAGRAMA DE PACKAGES COM DAO	30
DIAGRAMAS DE SEQUÊNCIA COM DAO	31
PROTÓTIPO INTERFACE	33
PRIMEIRA IMPLEMENTAÇÃO DOS USE CASES E EXPLORAÇÃO DE BIBLIOTECAS	43
CONCLUSÕES	47
REFERÊNCIAS BIBLIOGRÁFICAS	48

ÍNDICE DE DIAGRAMAS

DIAGRAMA 1: DIAGRAMA DO MODELO DE DOMÍNIO	6
DIAGRAMA 2: DIAGRAMA DE USE CASE	7
DIAGRAMA 3: DSS INICIAR SESSÃO	14
DIAGRAMA 4: DSS TERMINAR SESSÃO	15
DIAGRAMA 5: DSS EDITAR DADOS	15
DIAGRAMA 6: DSS - REGISTAR UTILIZADOR	16
DIAGRAMA 7: DSS REMOVER UTILIZADOR	16
DIAGRAMA 8: DSS - ALTERAR CATEGORIA	17
DIAGRAMA 9: DSS - CRIAR NOVA LISTA DE REPRODUÇÃO	17
DIAGRAMA 10: DSS - DOWNLOAD DE CONTEÚDO	18
DIAGRAMA 11: DSS - UPLOAD DE CONTEÚDO	18
DIAGRAMA 12: DSS - REMOVER CONTEÚDO	19
DIAGRAMA 13: DSS - REPRODUZIR CONTEÚDO	19
DIAGRAMA 14: DIAGRAMA DE PACKAGES	20
DIAGRAMA 15: DIAGRAMA DE CLASSES	21
DIAGRAMA 16: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS : INICIAR SESSÃO	22
DIAGRAMA 17: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - TERMINAR SESSÃO	22
DIAGRAMA 18: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - REGISTAR UTILIZADOR	23
DIAGRAMA 19: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS : REMOVER UTILIZADOR	23
DIAGRAMA 20: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - UPLOAD CONTEÚDO	24
DIAGRAMA 21: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - REMOVER CONTEÚDO	24
DIAGRAMA 22: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - EDITAR DADOS	25
DIAGRAMA 23: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - ALTERAR CATEGORIA DE CONTEÚDO	25
DIAGRAMA 24: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - CRIAR LISTA DE REPRODUÇÃO	26
DIAGRAMA 25: DIAGRAMA DE SEQUÊNCIA COM SUBSISTEMAS - REPRODUZIR CONTEÚDO	26
DIAGRAMA 26: DIAGRAMA DE SEQUÊNCIA DE IMPLEMENTAÇÃO - INÍCIO DE SESSÃO	27
DIAGRAMA 28: DIAGRAMA DE SEQUÊNCIA DE IMPLEMENTAÇÃO – REPRODUZIR CONTEÚDO	28
DIAGRAMA 36: DIAGRAMA DE CLASSES COM DAO	29
DIAGRAMA 37: DIAGRAMA DE PACKAGES COM DAO	30
DIAGRAMA 38: DIAGRAMA DE SEQUÊNCIA COM DAO - INICIAR SESSÃO	31
DIAGRAMA 40: DIAGRAMA DE SEQUÊNCIA COM DAO - REGISTAR UTILIZADOR	32

ÍNDICE DE FIGURAS

FIGURA 1: PÁGINA INICIAL MEDIA CENTER	33
FIGURA 2: LOGIN ADMINISTRADOR	33
FIGURA 3: ERRO TENTATIVA INÍCIO SESSÃO ADMINISTRADOR	33
FIGURA 4: LOGIN DE UTILIZADOR	34
FIGURA 5: ERRO TENTATIVA LOGIN	34
FIGURA 6: MENU ADMINISTRADOR	35
FIGURA 7: REGISTRAR UTILIZADOR	35
FIGURA 8: ERRO NO REGISTO DO UTILIZADOR	36
FIGURA 9: REMOVER UTILIZADOR	36
FIGURA 10: ERRO NA TENTATIVA DE REMOÇÃO DE UTILIZADOR	37
FIGURA 11: MEDIA CENTER	37
FIGURA 12: MEDIA CENTER DO UTILIZADOR CONVIDADO	38
FIGURA 13: A MINHA COLEÇÃO	38
FIGURA 14: LISTA DE REPRODUÇÃO	39
FIGURA 15: ADICIONAR CONTEÚDO À COLEÇÃO	39
FIGURA 16: REMOVER CONTEÚDO DE UMA COLEÇÃO	40
FIGURA 17: TRANFERIR CONTEÚDOS DE UMA COLEÇÃO	40
FIGURA 18: LISTA DE AMIGOS DE UM UTILIZADOR	41
FIGURA 19: ADICIONAR AMIGO	41
FIGURA 20: EDITAR DADOS PESSOAIS	42
FIGURA 21: EDITAR EMAIL	42
FIGURA 22: EDITAR PALAVRA-PASSE	42

BREVE DESCRIÇÃO DO ENUNCIADO PROPOSTO

Prevê-se, com este trabalho, o desenvolvimento de uma aplicação que represente um *mediacenter* para partilha de conteúdos (media) num apartamento. Esta aplicação terá por base dois cenários disponibilizados, de modo a facilitar a elaboração da mesma. A conceção e desenvolvimento da aplicação seguirá uma abordagem suportada por UML e a aplicação será desenvolvida utilizando uma arquitetura multicamada e a linguagem de programação orientada aos objetos, Java.

A primeira fase do projeto consistirá na análise de requisitos. Deste modo, será apresentado o **modelo de domínio**, que incluirá as entidades mais revelantes, o **modelo de use cases** com as funcionalidades propostas bem como as suas especificações e um protótipo da interface proposta.

Na segunda fase do projeto, é pedida a **modelação arquitetural e comportamental** de um sistema capaz de suportar os *use case* indicados pelos docentes. Para que a sua realização seja possível, serão apresentados os **diagramas de package, diagramas de classe**, bem como os **diagramas de sequência**. Por fim, é pedida uma primeira implementação de alguns *use case* relevantes, com a intuito de perceber a relação entre os modelos e o código, assim como a exploração de bibliotecas capazes de solucionar os objetivos finais.

No que diz respeito à última fase do projeto, é pedida a implementação dos *use case* definidos pelos docentes, garantindo que a aplicação irá suportar a persistência dos dados numa Base de Dados relacional. É de notar que esta fase corresponde a uma primeira iteração do desenvolvimento, cujo intuito seria prosseguir até ser atingido o objetivo final.

MODELAÇÃO

MODELO DE DOMÍNIO

A conceção do modelo apresentado teve por base uma primeira escolha de algumas das principais entidades presentes no mesmo. Seguidamente, estabeleceram-se as respetivas ligações e cardinalidades analisadas como necessárias. Entidades adicionais foram introduzidas conforme a progressão do modelo.

Assim sendo, decidiu-se que cada utilizador, seja este administrador, utilizador residente ou utilizador convidado é um utilizador. No entanto, apenas o administrador e o utilizador residente usufruem de credenciais, bem como tudo que estas possuem (nome e email). Definiu-se ainda que, não obstante ao potencial de cada utilizador na plataforma, é o utilizador residente que apresenta um maior domínio no aproveitamento das funcionalidades da mesma, como tal, um utilizador residente pode aceder à biblioteca, desfrutar da sua coleção, criar e, posteriormente, aceder às suas listas de reprodução (tendo a possibilidade de reproduzir os conteúdos nesta presentes, assim como os conteúdos existentes na biblioteca, em modo aleatório, ordenado, modo repetição, ou até mesmo, a junção do modo aleatório e repetição) e ainda escolher a sua lista de amigos (quer os potenciais amigos sejam sugestão do sistema quer não).

Numa outra vertente, evidencia-se a pequenez e limitação dos restantes utilizadores, sendo que o administrador apenas é capaz de efetuar o registo de um utilizador residente, bem como a sua remoção e o utilizador convidado só pode reproduzir conteúdo pertencente à biblioteca.

Relativamente ao conteúdo cujo *upload* é feito para a biblioteca, e consequentemente, atualizado na coleção do utilizador residente feitor, este pode ser uma música ou um vídeo, que abrange um nome, duração, tamanho e organiza-se em categorias.

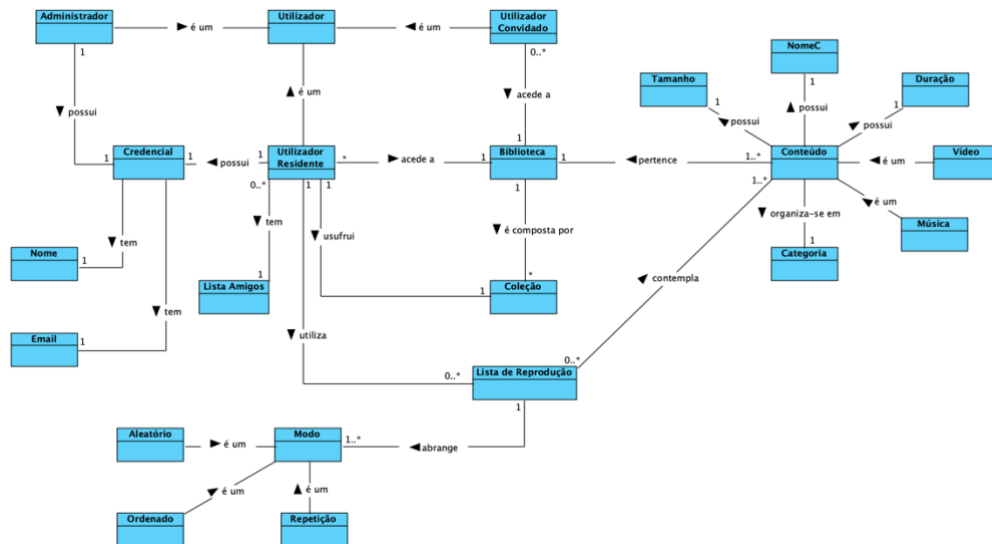


Diagrama 1: Diagrama do Modelo de Domínio

MODELO DE USE CASE

Um diagrama de *use case* representa o levantamento de requisitos de um sistema. Por sua vez, um requisito é uma funcionalidade ou característica considerada relevante na ótica do utilizador e representa o comportamento esperado do sistema [1].

Juntamente com a representação do diagrama de *use case*, serão apresentadas as especificações de cada use case com o objetivo de expor as possíveis ações executadas pelos diferentes atores do sistema.

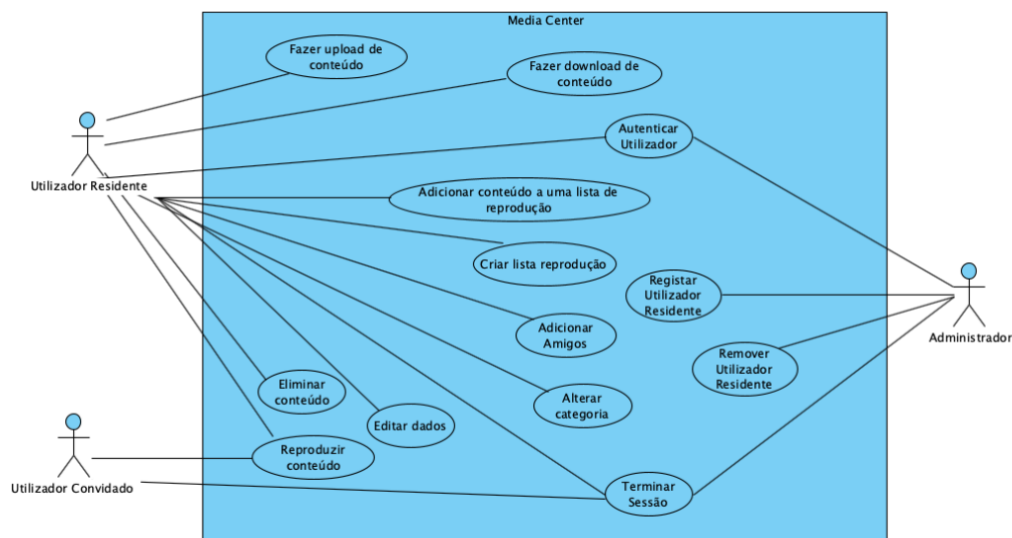


Diagrama 2: Diagrama de Use Case

ESPECIFICAÇÃO USE CASES

1. Reprodução de conteúdo

- **Use Case:** Reprodução de conteúdo.
 - **Descrição:** O utilizador escolhe o conteúdo de uma biblioteca que deseja reproduzir.
 - **Cenários:** O Ricardo reproduz um conteúdo num dado modo; O Rui acedeu como convidado, após o Ricardo ter terminado sessão;
 - **Pré-condição:** Existe conteúdo na biblioteca e o utilizador encontra-se autenticado.
 - **Pós-condição:** Sistema reproduz o conteúdo pretendido.
 - **Fluxo Normal:**
 1. O utilizador acede à biblioteca;
 2. O utilizador escolhe um conteúdo singular a reproduzir;
 3. O sistema reproduz o conteúdo escolhido.
 - **Fluxo Alternativo 1:** [Escolher lista de reprodução] (passo 2)
 - 2.1 O utilizador residente escolhe uma lista de reprodução;
 - 2.2 O utilizador residente escolhe o modo de reprodução ordenado;
 - 2.3 Regressa a 3.
 - **Fluxo Alternativo 2:** [Escolher lista para reprodução aleatória] (passo 2)
 - 2.1 O utilizador residente escolhe uma lista de reprodução;
 - 2.2 O utilizador residente escolhe o modo de reprodução aleatório;
 - 2.3 Regressa a 3.
 - **Fluxo Alternativo 3:** [Escolher lista de reprodução com repetição] (passo 2)
 - 2.1 O utilizador residente escolhe uma lista de reprodução;
 - 2.2 O utilizador residente escolhe o modo de reprodução com repetição;
 - 2.3 Regressa a 3.

2. Registar utilizador residente

- **Use Case:** Registar de utilizador.
 - **Descrição:** O administrador procede ao registo do utilizador residente.
 - **Cenários:** A Paula pretende registar um utilizador residente, o Manuel.
 - **Pré-condição:** Tem que ser efetuada a autenticação do administrador.
 - **Pós-condição:** O sistema guarda o registo do novo utilizador.
 - **Fluxo Normal:**
 1. O administrador insere o nome do residente;
 2. O administrador insere o email do residente;
 3. O sistema guarda o registo do novo utilizador.
 - **Fluxo Alternativo:** [Email já se encontra registado] (passo 2)
 - 2.1 O sistema informa que o email inserido já se encontra registado;
 - 2.2 O administrador insere um novo email;
 - 2.3 Regressa a 3.

3. *Upload* do conteúdo

- **Use Case:** *Upload* de conteúdo.
 - **Descrição:** O utilizador residente envia conteúdo para o *media center*.
 - **Cenários:** O Manuel pretende fazer *upload* do conteúdo para a sua coleção no *media center*.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** O sistema guarda na biblioteca e na coleção do utilizador o conteúdo.
 - **Fluxo Normal:**
 1. O sistema reconhece o meio físico;
 2. O utilizador residente escolhe o conteúdo a transferir para a sua coleção;
 3. O sistema procede à transferência do conteúdo;
 4. O sistema altera o estado da coleção do utilizador e, consequentemente, da biblioteca com o novo conteúdo;
 5. O sistema desconecta do meio físico.
 - **Fluxo Alternativo:** [O conteúdo escolhido já se encontra na biblioteca] (passo 3)
 - 3.1 O sistema verifica que o conteúdo já existe na biblioteca;
 - 3.2 O sistema não transfere o conteúdo existente para a biblioteca;
 - 3.3 Regressa a 4.

4. *Download* de conteúdo

- **Use Case:** *Download* de conteúdo.
 - **Descrição:** O utilizador residente descarrega conteúdos da sua coleção para um meio físico.
 - **Cenários:** A Isabel faz *download* de conteúdos presentes na sua coleção.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** Os conteúdos são transferidos para o meio físico.
 - **Fluxo Normal:**
 1. O sistema reconhece o meio físico;
 2. O utilizador residente escolhe os conteúdos a transferir da coleção;
 3. O sistema procede à transferência dos conteúdos;
 4. O sistema desconecta do meio físico.
 - **Fluxo de Exceção:** [Acesso a outras coleções] (passo 2)
 - 2.1 O utilizador residente escolhe conteúdos da coleção de outro residente;
 - 2.2 O sistema não permite a transferência dos conteúdos desejados.

5. Remover conteúdo

- **Use Case:** Remover conteúdo.
 - **Descrição:** O utilizador residente remove um conteúdo da sua coleção.
 - **Cenários:** O Ricardo decidiu remover um conteúdo da sua coleção.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente e o residente encontra-se na sua coleção.
 - **Pós-condição:** Os conteúdos são removidos da coleção.
 - **Fluxo Normal:**
 1. O utilizador residente escolhe o conteúdo a remover da sua coleção;
 2. O sistema procede à remoção do conteúdo da sua coleção;
 3. O sistema guarda o novo estado da sua coleção.

6. Criar uma nova lista de reprodução

- **Use Case:** Criar uma nova lista de reprodução.
 - **Descrição:** Criar uma nova lista de reprodução.
 - **Cenários:** O Manuel pretende criar uma lista de reprodução.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** O sistema guarda a lista de reprodução criada.
 - **Fluxo Normal:**
 1. O utilizador residente seleciona conteúdos da biblioteca;
 2. O utilizador residente atribui um nome à nova lista;
 3. O sistema transfere os conteúdos selecionados para a lista criada;
 4. O sistema guarda a lista de reprodução.

7. Adicionar conteúdo a uma lista de reprodução

- **Use Case:** Adicionar conteúdo a uma lista de reprodução
 - **Descrição:** Adicionar um conteúdo a uma lista de reprodução.
 - **Cenários:** O Manuel pretende adicionar um novo conteúdo a uma lista de reprodução existente.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** O sistema atualiza a lista de reprodução.
 - **Fluxo Normal:**
 1. O utilizador residente seleciona o conteúdo a adicionar à lista;
 2. O sistema transfere o conteúdo selecionado para a lista;
 3. O sistema atualiza a lista de reprodução.
 - **Fluxo de Exceção:** [Conteúdo já existente na lista] (passo 2)
 - 2.2 O sistema verifica que o conteúdo já se encontra na lista;
 - 2.3 O sistema não transfere o conteúdo para a lista.

8. Alterar a categoria de um conteúdo

- **Use Case:** Alterar a categoria de um conteúdo.
 - **Descrição:** O utilizador residente edita o nome da categoria de um conteúdo.
 - **Cenários:** O Ricardo pretende alterar a categoria de um conteúdo.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** O sistema guarda novo nome da categoria.
 - **Fluxo Normal:**
 1. O utilizador residente edita o nome da categoria do conteúdo;
 2. O sistema altera o estado das categorias afetadas.

9. Adicionar amigos

- **Use Case:** Adicionar amigos.
 - **Descrição:** O utilizador residente tem a possibilidade de adicionar outros utilizadores residentes à sua lista de amigos.
 - **Cenários:** O Manuel pretende adicionar à sua lista de amigos, a Paula, a Isabel e o Ricardo.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** O utilizador residente tem adicionados utilizadores residentes à sua lista de amigos.
 - **Fluxo Normal:**
 1. Sistema sugere adição de amigo com base nas preferências do utilizador;
 2. O utilizador residente aceita sugestão de adição de outro utilizador residente à sua lista de amigos;
 3. O sistema envia ao outro utilizador residente o aviso de pedido de amizade;
 4. O utilizador residente que recebeu o pedido aceita o mesmo;
 5. O sistema altera o estado da lista de amigos de ambos os utilizadores residentes.
 - **Fluxo Alternativo:** [Utilizador adiciona amigo] (passo 1)
 - 1.1 O utilizador residente procura um utilizador registado;
 - 1.2 Regressa a 3.
 - **Fluxo de Exceção 1:** [Utilizador recusa a sugestão de adição de amigo] (passo 2)
 - 1.1 O utilizador residente recusa sugestão de adição do amigo por parte do sistema.
 - **Fluxo de Exceção 2:** [Utilizador rejeita pedido de amizade] (passo 4)
 - 4.1 O utilizador residente que recebeu o pedido rejeita o mesmo.

10. Remover utilizador residente

- **Use Case:** Remover utilizador residente.
 - **Descrição:** O administrador procede à eliminação do utilizador residente.
 - **Cenários:** A Paula pretende remover a Isabel.
 - **Pré-condição:** Tem que ser efetuada a autenticação do administrador.
 - **Pós-condição:** O sistema guarda registo da eliminação do novo utilizador.
 - **Fluxo Normal:**
 1. O administrador insere o email do utilizador residente;
 2. O sistema elimina o utilizador da lista de utilizadores.
 - **Fluxo de Exceção:** [Utilizador não se encontra registado] (passo 2)
 - 2.1 O sistema informa que o email inserido não se encontra registado.

11. Editar dados

- **Use Case:** Editar dados de utilizador residente.
 - **Descrição:** O utilizador residente procede à edição dos seus dados pessoais.
 - **Cenários:** A Isabel pretende editar os seus dados.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador residente.
 - **Pós-condição:** O sistema guarda registo da alteração efetuada do utilizador.
 - **Fluxo Normal:**
 1. O utilizador residente altera a sua palavra-passe atual;
 2. O sistema atualiza os dados referentes ao utilizador.
 - **Fluxo Alternativo:** [Utilizador altera o seu email] (passo 1)
 - 1.1 O utilizador residente altera o seu email atual;
 - 1.2 Regressa a 2.

12. Iniciar sessão

- **Use Case:** Iniciar sessão.
 - **Descrição:** O utilizador efetua o início de sessão.
 - **Cenários:** A Isabel introduz a sua palavra-passe para completar o seu registo. A Paula inicia sessão como administrador após a Isabel terminar sessão. O Rui inicia sessão como convidado quando nenhum utilizador se encontra autenticado.
 - **Pré-condição:** Existe conteúdo na biblioteca e nenhum utilizador se encontra autenticado.
 - **Pós-condição:** O utilizador efetuou o início de sessão com sucesso.
 - **Fluxo Normal:**
 1. O utilizador residente introduz o seu email e palavra-passe;
 2. O sistema valida os dados introduzidos pelo utilizador;
 - **Fluxo Alternativo 1:** [Utilizador inicia sessão como administrador] (passo 1)
 - 1.1 O utilizador insere a palavra-passe;
 - 1.3 Regressa a 2.
 - **Fluxo Alternativo 2:** [Utilizador inicia sessão como convidado] (passo 1)
 - 1.1 O utilizador opta pelo início de sessão como convidado;
 - **Fluxo Alternativo 3:** [Inserção da palavra-passe] (passo 1)
 - 1.1 O utilizador residente introduz a palavra-passe no primeiro início de sessão;
 - 1.2 Regressa a 2.
 - **Fluxo de Exceção:** [Tentativa de início de sessão inválida] (passo 1)
 - 1.1 O utilizador residente tenta iniciar sessão no sistema;
 - 1.2 O sistema informa que o utilizador ou a palavra-passe são inválidos.

13. Terminar sessão

- **Use Case:** Terminar sessão.
 - **Descrição:** O utilizador termina sessão.
 - **Cenários:** A Isabel termina sessão.
 - **Pré-condição:** Tem que ser efetuada a autenticação do utilizador.
 - **Pós-condição:** O utilizador efetuou o término de sessão com sucesso.
 - **Fluxo Normal:**
 1. O utilizador residente termina sessão;
 2. O sistema guarda o estado da sessão.
 - **Fluxo Alternativo 1:** [sessão iniciada como convidado] (passo 1)
 1. O utilizador convidado termina sessão;
 2. Regressa a 3.
 - **Fluxo Alternativo 2:** [sessão iniciada como administrador] (passo 1)
 3. O administrador termina sessão;
 4. Regressa a 3.

DIAGRAMAS DE SEQUÊNCIA DE SISTEMA

Em UML são utilizados diagramas de interação para modelar os aspectos dinâmicos do sistema em termos dos objetos e suas interações, tendo como base as mensagens trocadas entre objetos [1]. O diagrama de sequência é um tipo de diagrama de interação, que apresenta as interações entre objetos a partir do encadeamento temporal das mensagens, pelo que permite analisar a distribuição de “responsabilidade” pelas diferentes entidades.

1. Iniciar Sessão

O diagrama de sequência de sistema (DSS) que se segue representa a autenticação de um utilizador no *mediacenter*.

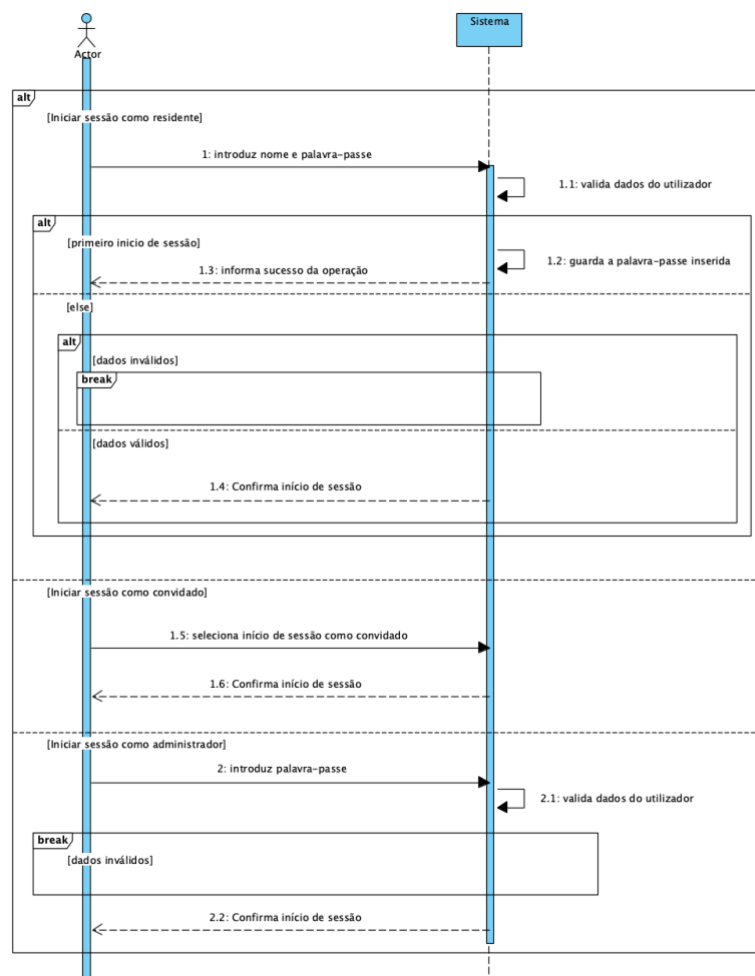


Diagrama 3: DSS iniciar sessão

2. Terminar Sessão

O DSS que se segue representa o término de sessão de um utilizador do *mediacenter*.

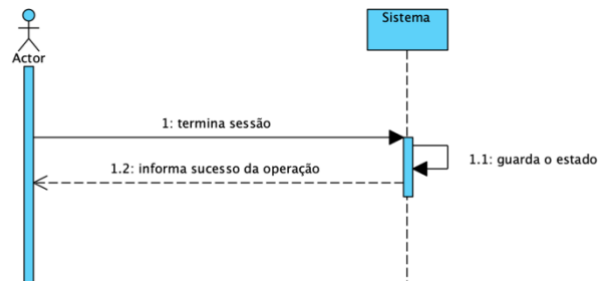


Diagrama 4: DSS terminar sessão

3. Editar Dados

No DSS apresentado, é demonstrada a alteração dos dados pessoais de um utilizador residente.

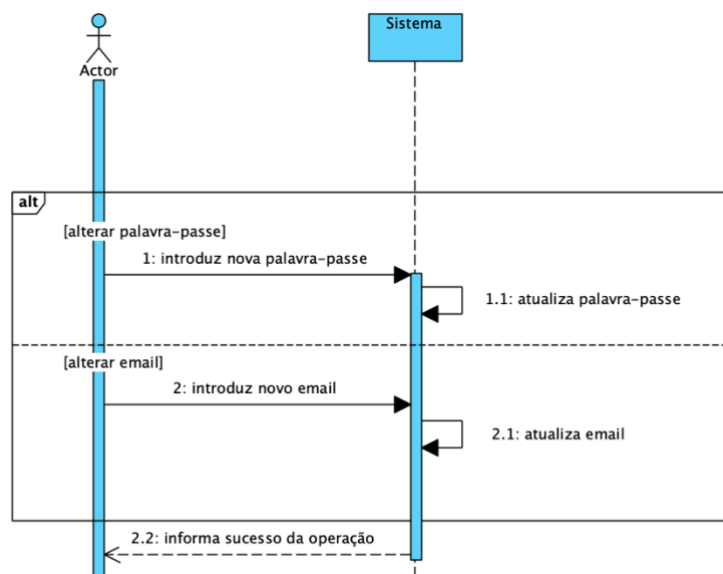


Diagrama 5: DSS editar dados

4. Registrar Utilizador

Diagrama onde se encontram representadas as interações entre o ator (administrador) e o sistema (*mediacenter*) no processo de registo de um utilizador no sistema.

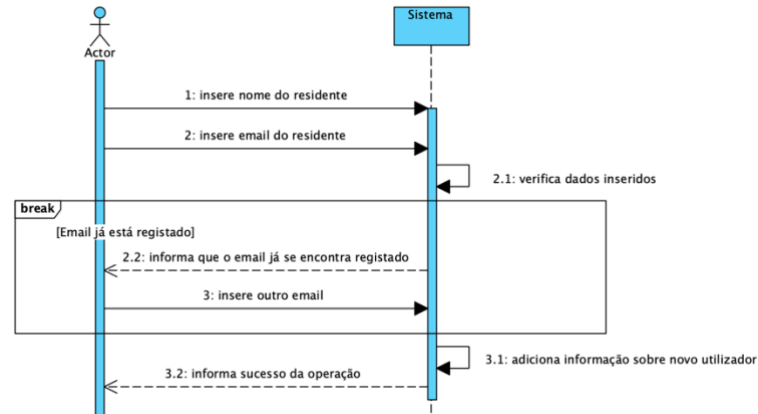


Diagrama 6: DSS - registrar utilizador

5. Remover Utilizador

A interação entre o sistema e o ator (administrador) para a concretização da remoção de um utilizador do sistema é representada pelo diagrama que se segue.

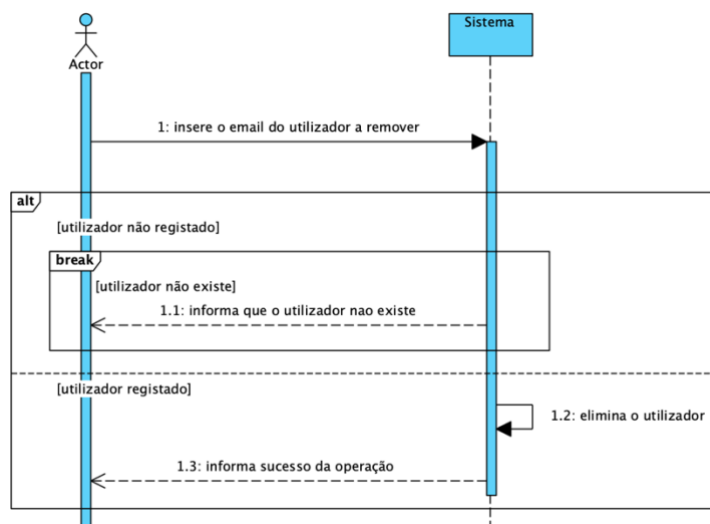


Diagrama 7: DSS remover utilizador

6. Alterar categoria de um conteúdo

No diagrama que se segue é representada a interação entre o utilizador e o sistema para a alteração da categoria de um conteúdo.

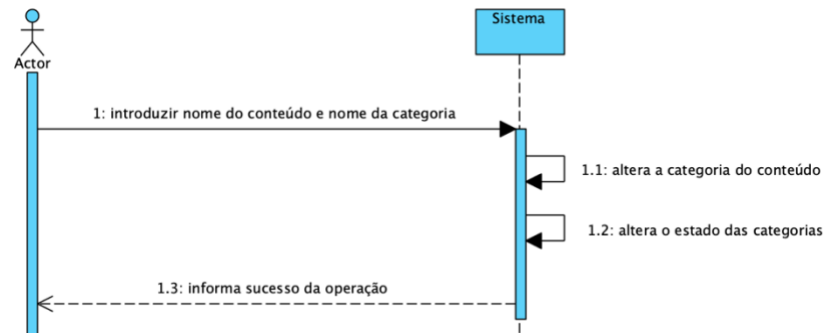


Diagrama 8: DSS - alterar categoria

7. Criar nova lista de reprodução

DSS que demonstra a interação necessária entre o sistema e o ator de forma a criar uma nova lista de reprodução.

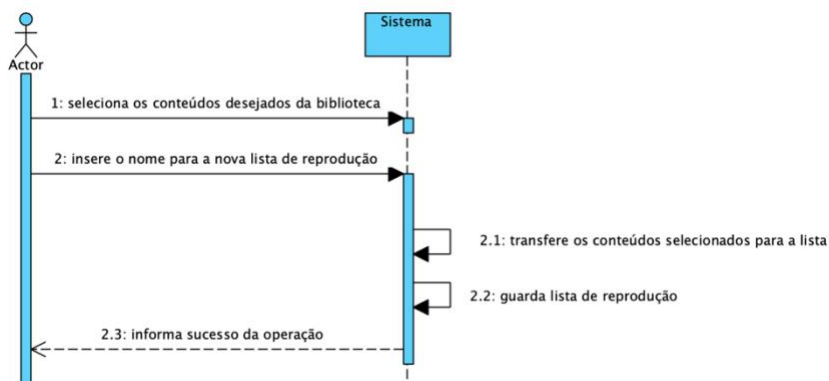


Diagrama 9: DSS - criar nova lista de reprodução

8. Download de conteúdo

O seguinte diagrama descreve a interação entre o sistema e o utilizador face à transferência de conteúdo da sua coleção para um meio físico.

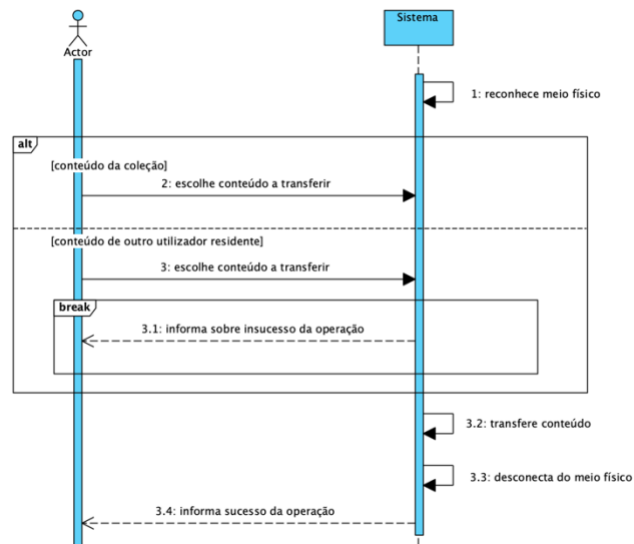


Diagrama 10: DSS - download de conteúdo

9. Upload conteúdo

A interação entre o utilizador residente (ator) e o *mediacenter* (sistema) face à ação de transferir conteúdo para a sua coleção é representada no DSS que se segue.

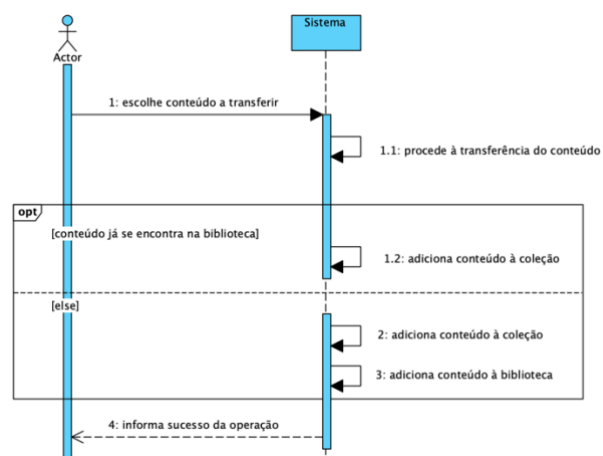


Diagrama 11: DSS - upload de conteúdo

10. Remover conteúdo

Diagrama que representa a remoção de conteúdo selecionado da coleção de um utilizador residente.

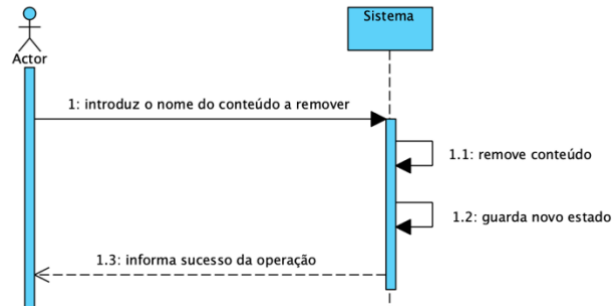


Diagrama 12: DSS - remover conteúdo

11. Reproduzir conteúdo

Neste último diagrama é visível a interação entre o utilizador (ator) e o *mediacenter* (sistema) que representa a ação de reprodução de conteúdo no sistema.

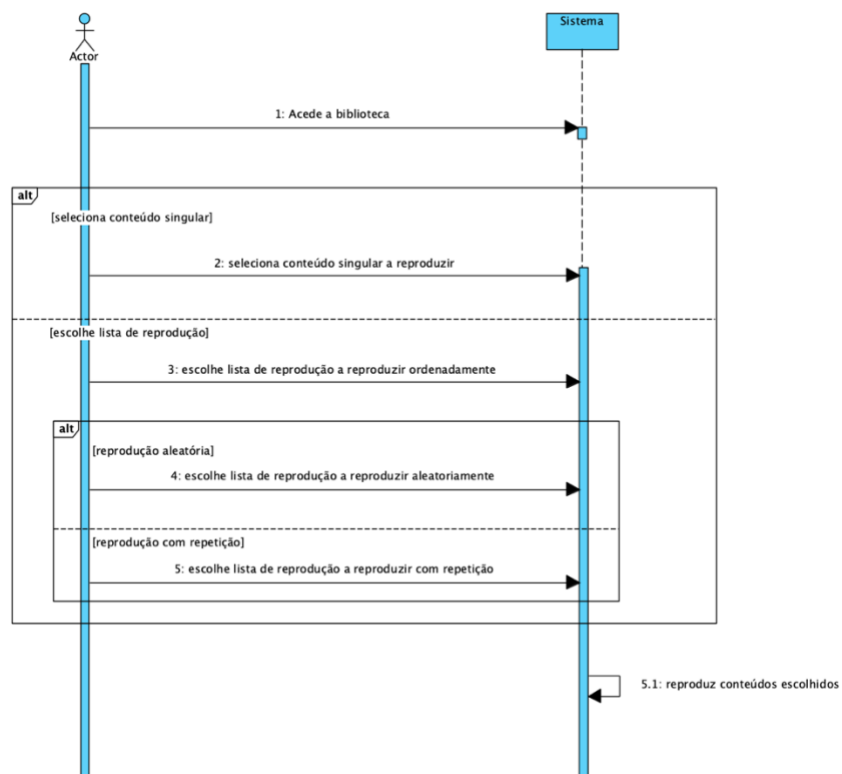


Diagrama 13: DSS - reproduzir conteúdo

DIAGRAMA DE PACKAGES

Os *packages* permitem dividir a complexidade do sistema em partes mais pequenas, permitindo agrupar elementos de modelação UML. Um diagrama de *packages* representa uma relação de dependência entre eles [1].

No *package* **Camada de Apresentação** é possível encontrar as classes que dizem respeito à interface gráfica. No que diz respeito ao *package* **Camada de Dados**, encontram-se presentes as classes que constituem a base de dados do projeto.

Na Lógica de Negócios verifica-se a existência de três *packages* distintos. A **MediaCenterFacade** apresenta uma dependência sobre o *package* **Utilizadores** - que contém a classe referente aos utilizadores - e **Biblioteca**, que diz respeito às classes biblioteca, conteúdo, lista de reprodução e coleção, e acede à **Camada de Dados**, com o intuito de garantir a persistência dos dados.

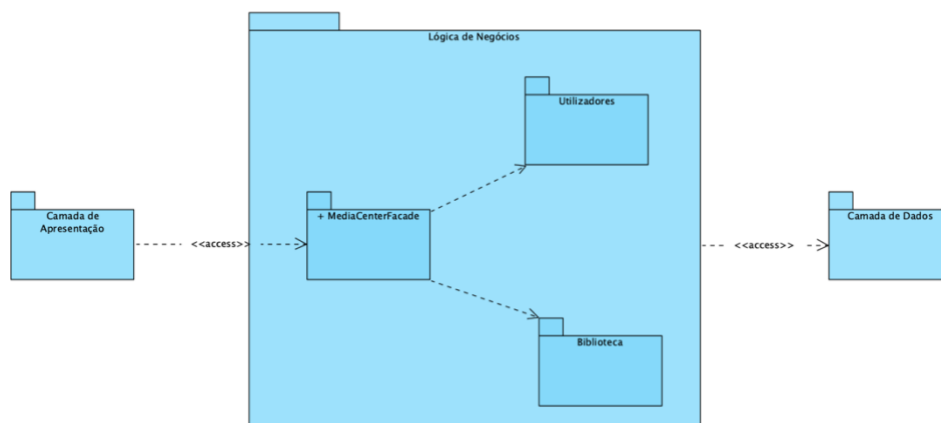


Diagrama 14: Diagrama de Packages

DIAGRAMA DE CLASSES

Uma classe descreve um conjunto de objetos com a mesma estrutura (atributos e relacionamentos) e comportamento (operações), podendo estar relacionada com outras classes através de dependências, associações ou generalizações.

Um diagrama de classes é, portanto, uma descrição formal da estrutura de objetos num sistema, descrevendo a sua identidade, os seus atributos e as suas operações, bem como o relacionamento com outros objetos [1].

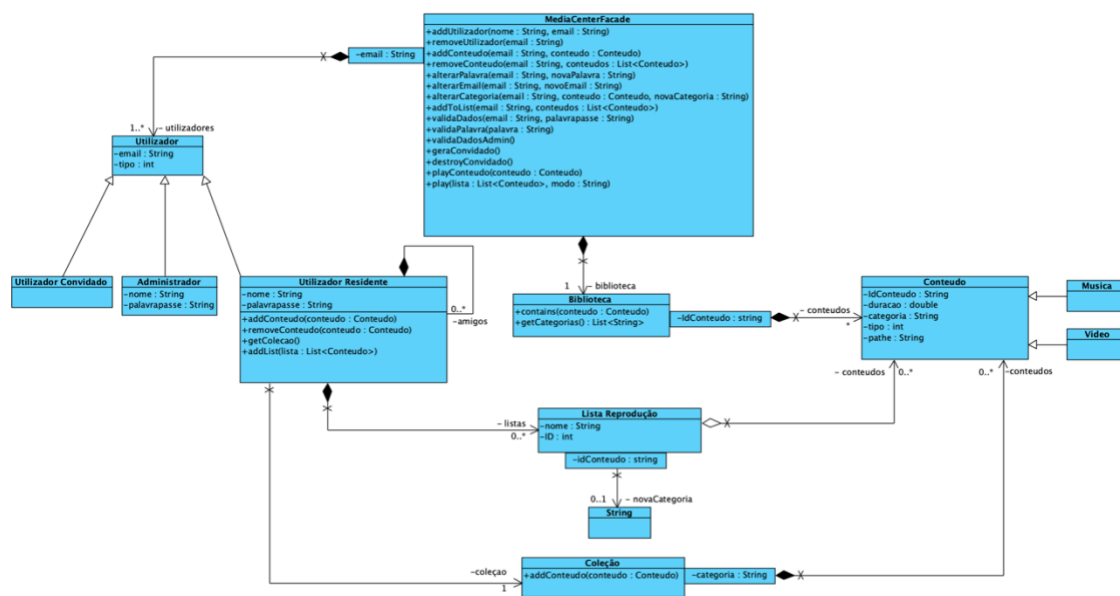


Diagrama 15: Diagrama de Classes

DIAGRAMAS DE SEQUÊNCIA COM SUBSISTEMAS

1. Iniciar sessão

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários *packages* do programa e o utilizador quando este inicia sessão no *mediacenter*.

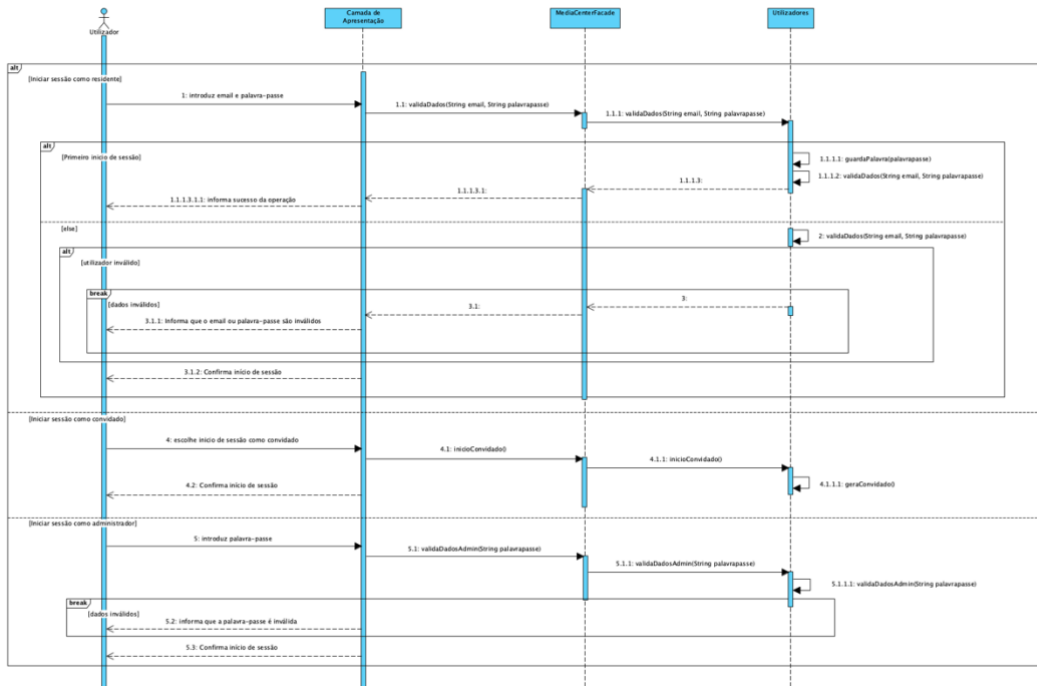


Diagrama 16: Diagrama de Sequência com Subsistemas : iniciar sessão

2. Terminar sessão

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários *packages* do programa e o utilizador quando este termina sessão no *mediacenter*.

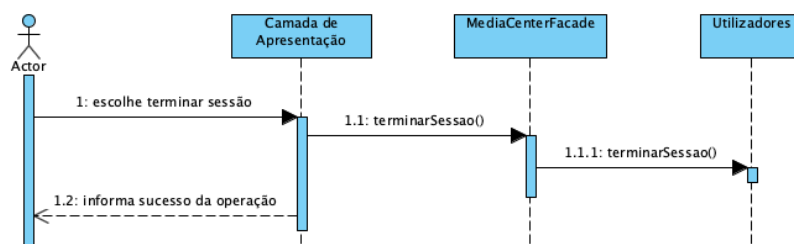


Diagrama 17: Diagrama de Sequência com Subsistemas - terminar sessão

3. Registrar Utilizador

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários packages do programa e o administrador quando este regista um novo utilizador no *mediacenter*.

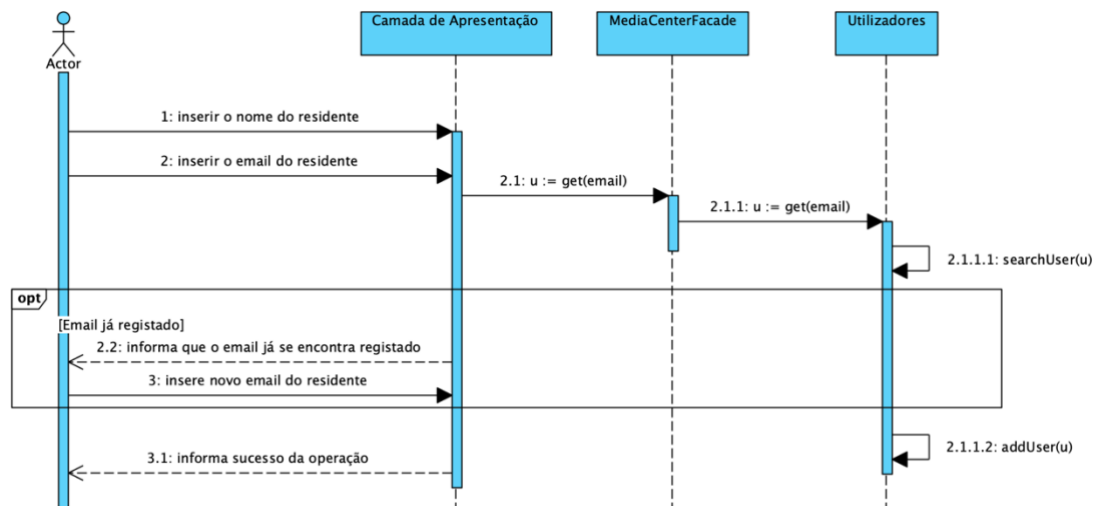


Diagrama 18: Diagrama de Sequência com Subsistemas - registar utilizador

4. Remover Utilizador

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários packages do programa e o administrador quando este remove um utilizador residente do *mediacenter*.

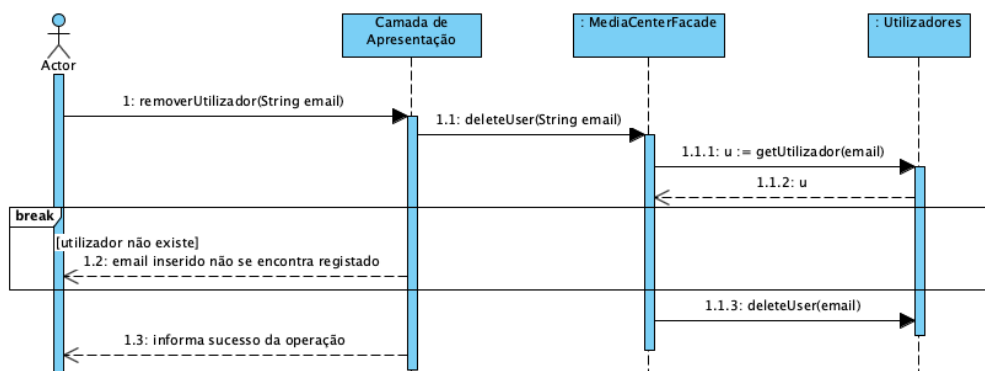


Diagrama 19: Diagrama de Sequência com Subsistemas : remover utilizador

5. Upload Conteúdo

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários *packages* do programa e o utilizador quando este procede ao *upload* de conteúdo no *mediacenter*.

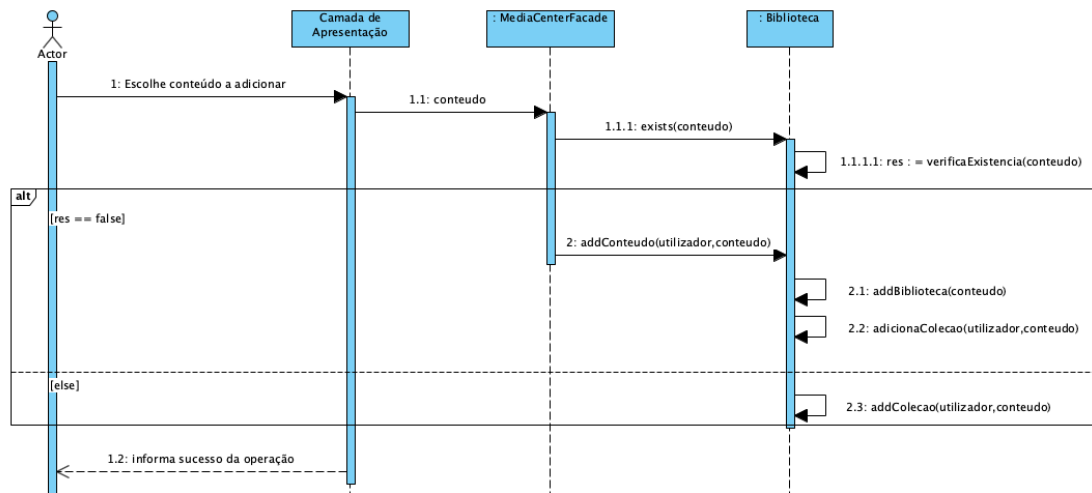


Diagrama 20: Diagrama de Sequência com Subsistemas - upload conteúdo

6. Remover Conteúdo

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários *packages* do programa e o utilizador quando este procede à remoção de conteúdo no *mediacenter*.

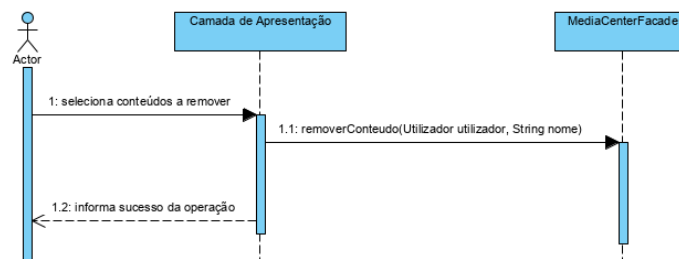


Diagrama 21: Diagrama de Sequência com Subsistemas - remover conteúdo

7. Editar dados

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários packages do programa e o utilizador quando este procede à edição dos seus dados pessoais no *mediacenter*.

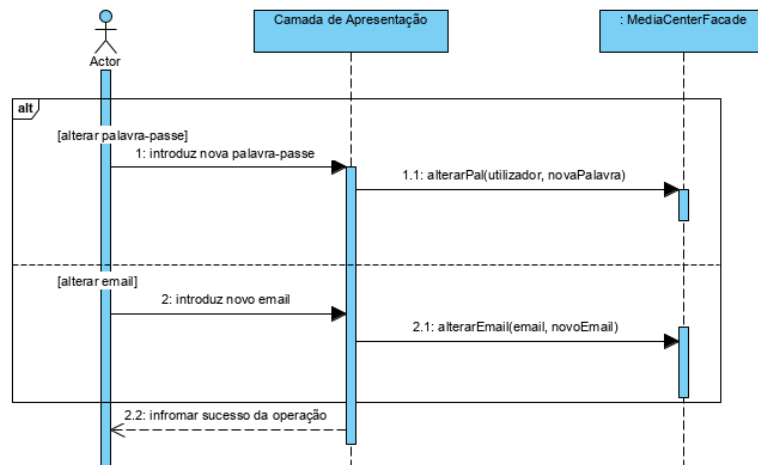


Diagrama 22: Diagrama de Sequência com Subsistemas - editar dados

8. Alterar a categoria de um conteúdo

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários packages do programa e o utilizador quando este realiza a alteração da categoria de um conteúdo no *mediacenter*.

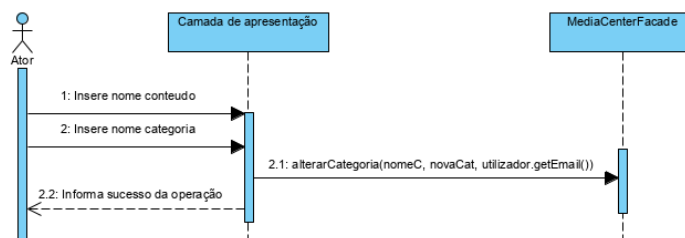


Diagrama 23: Diagrama de Sequência com Subsistemas - alterar categoria de conteúdo

9. Criar lista de reprodução

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários packages do programa e o utilizador quando este efetua a criação de uma nova lista de reprodução.

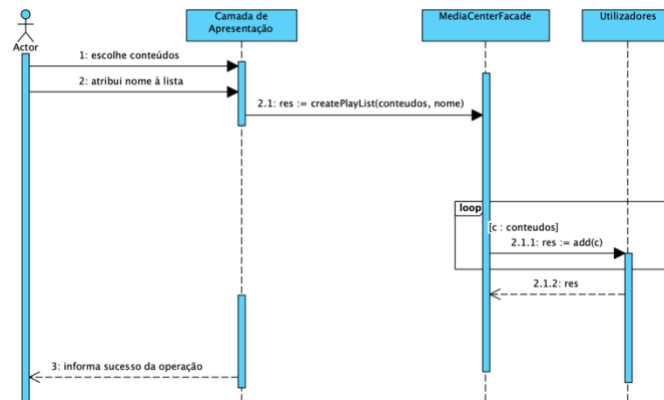


Diagrama 24: Diagrama de Sequência com Subsistemas - criar lista de reprodução

10. Reproduzir conteúdo

O diagrama de sequência com subsistemas que se segue representa as interações entre os vários packages do programa e o utilizador quando este reproduz conteúdos no *mediacenter*.

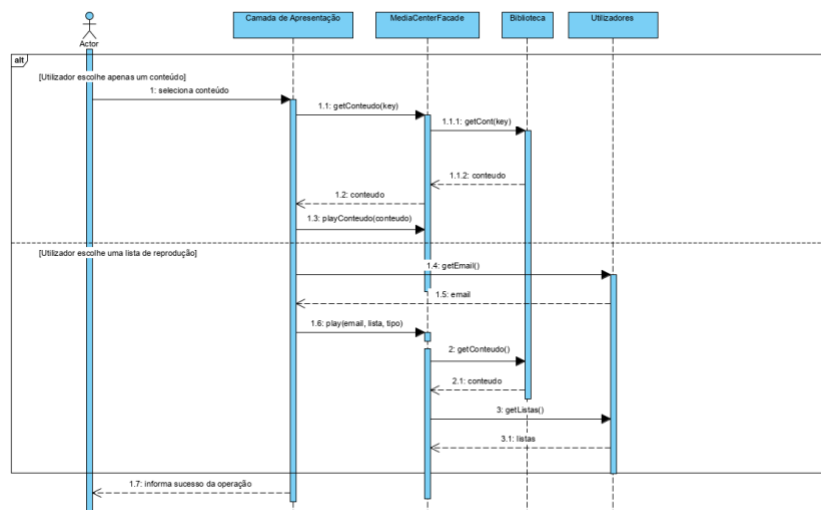


Diagrama 25: Diagrama de Sequência com Subsistemas - reproduzir conteúdo

Por sugestão dos docentes da unidade curricular e com o propósito de se demonstrar o seu funcionamento aplicacional serão apresentados dois diagramas de sequência de implementação a título de exemplo. Os restantes diagramas encontram-se juntamente com os ficheiros da aplicação.

1. Iniciar Sessão

O diagrama de sequência de implementação que se segue representa as interações entre as várias classes do programa e o utilizador, aquando do início de sessão no *mediacenter*.

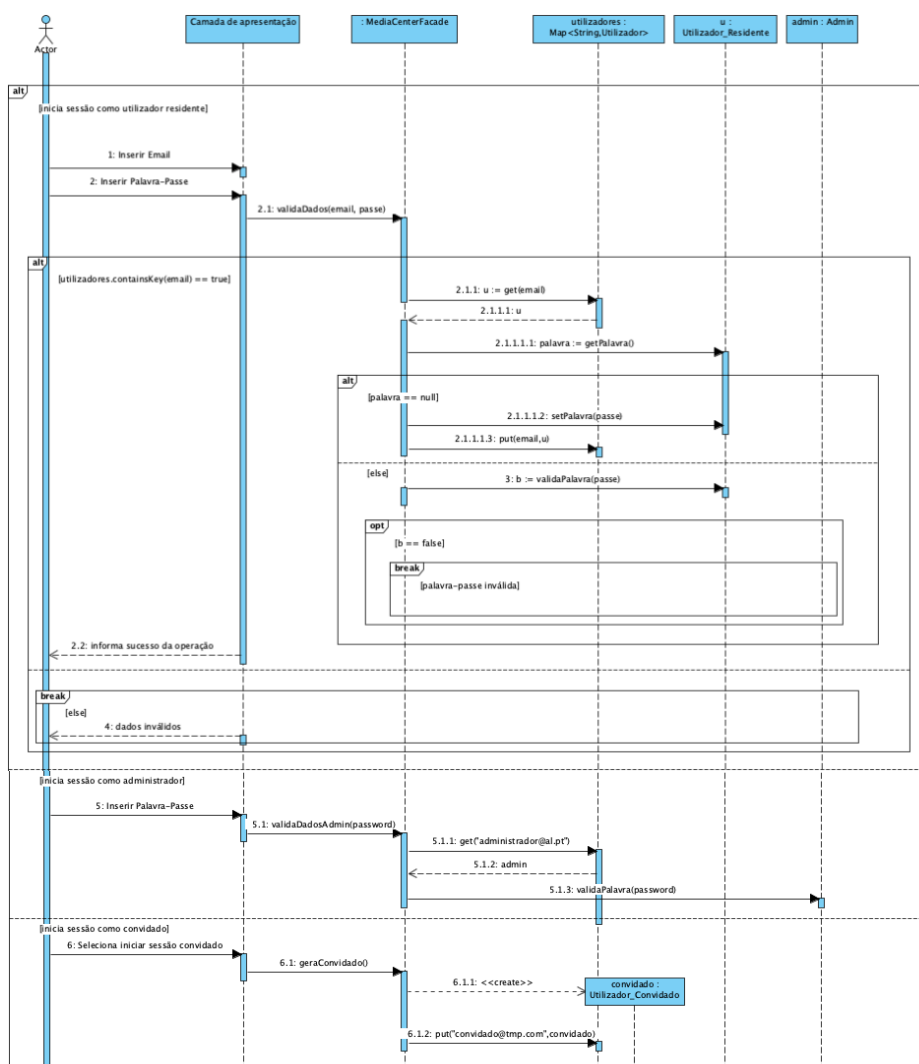


Diagrama 26: Diagrama de Sequência de Implementação - início de sessão

2. Reproduzir Conteúdo

O diagrama de sequência de implementação que se segue representa as interações entre as várias classes do programa e o administrador, aquando do registo de um novo utilizador residente no *mediacenter*.

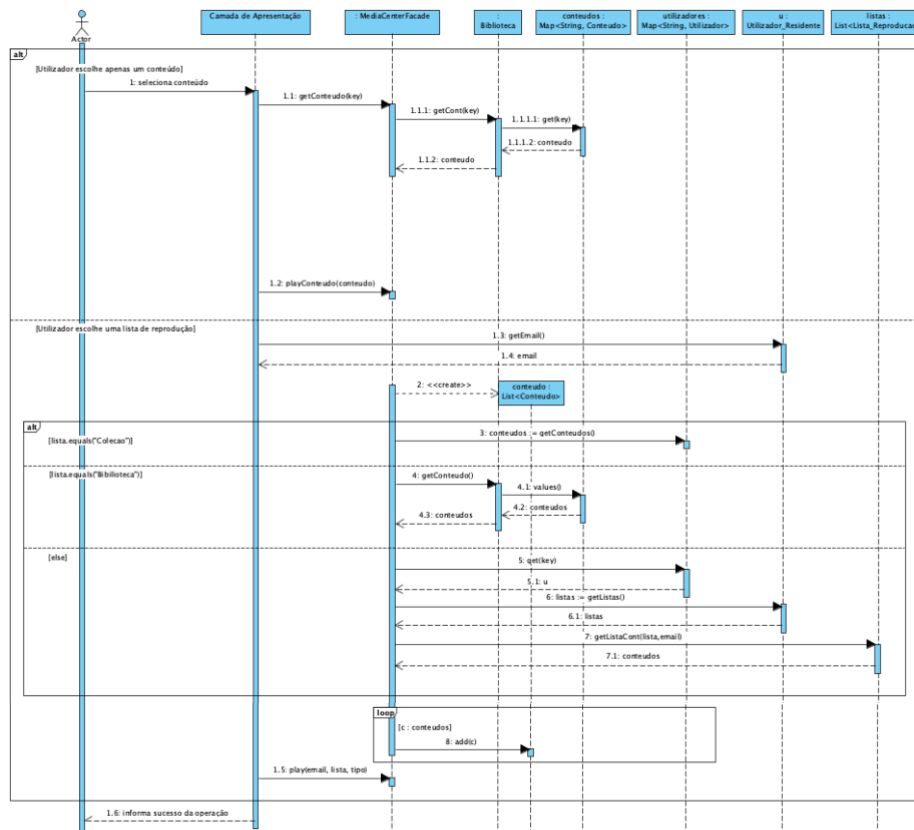


Diagrama 27: Diagrama de Sequência de Implementação – reproduzir conteúdo

OBJECT RELATIONAL MAPPING - ORM

Para a implementação desta aplicação será utilizada a linguagem de programação orientada aos objetos, Java, onde será desenvolvida toda a lógica de negócios do projeto.

De modo a garantir a persistência de dados, estes serão armazenados numa base de dados, onde será utilizado o *MySQL*.

No entanto, é necessário estabelecer um mapeamento entre os paradigmas orientado a objetos e relacional, uma vez que estes não são diretamente compatíveis.

De forma a ser possível este mapeamento, foi criado o ORM. Por este motivo, são implementados os DAOs, isto é, as operações que estabelecem a conexão entre a **Lógica de Negócios** e a base de dados da aplicação.

DIAGRAMA DE CLASSES COM DAO

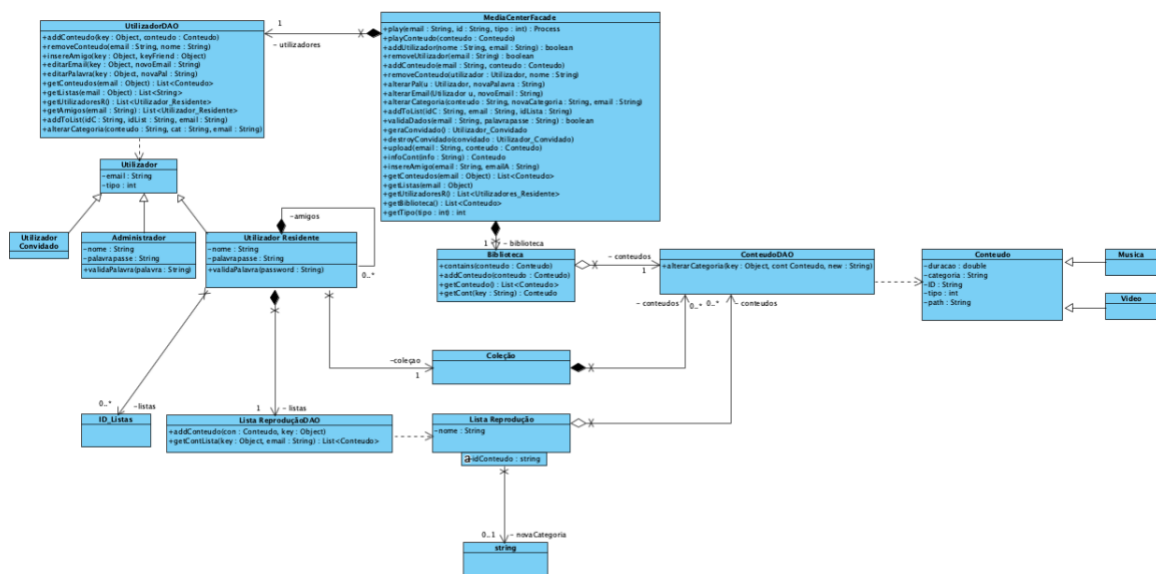


Diagrama 28: Diagrama de Classes com DAO

DIAGRAMA DE PACKAGES COM DAO

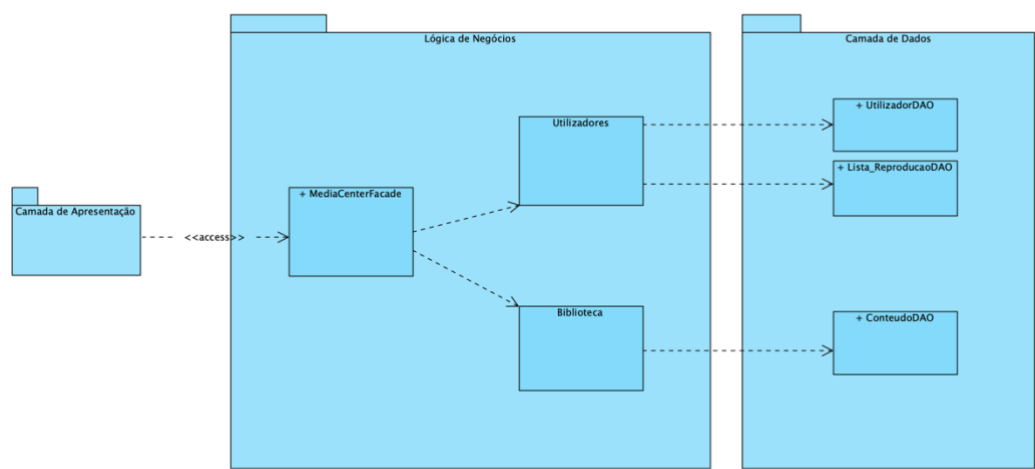


Diagrama 29: Diagrama de Packages com DAO

Assim como referido nos diagramas de sequência de implementação, serão também apresentados dois diagramas de sequência com DAO, com vista a uma melhor percepção de implementação dos mesmos.

1. Iniciar sessão

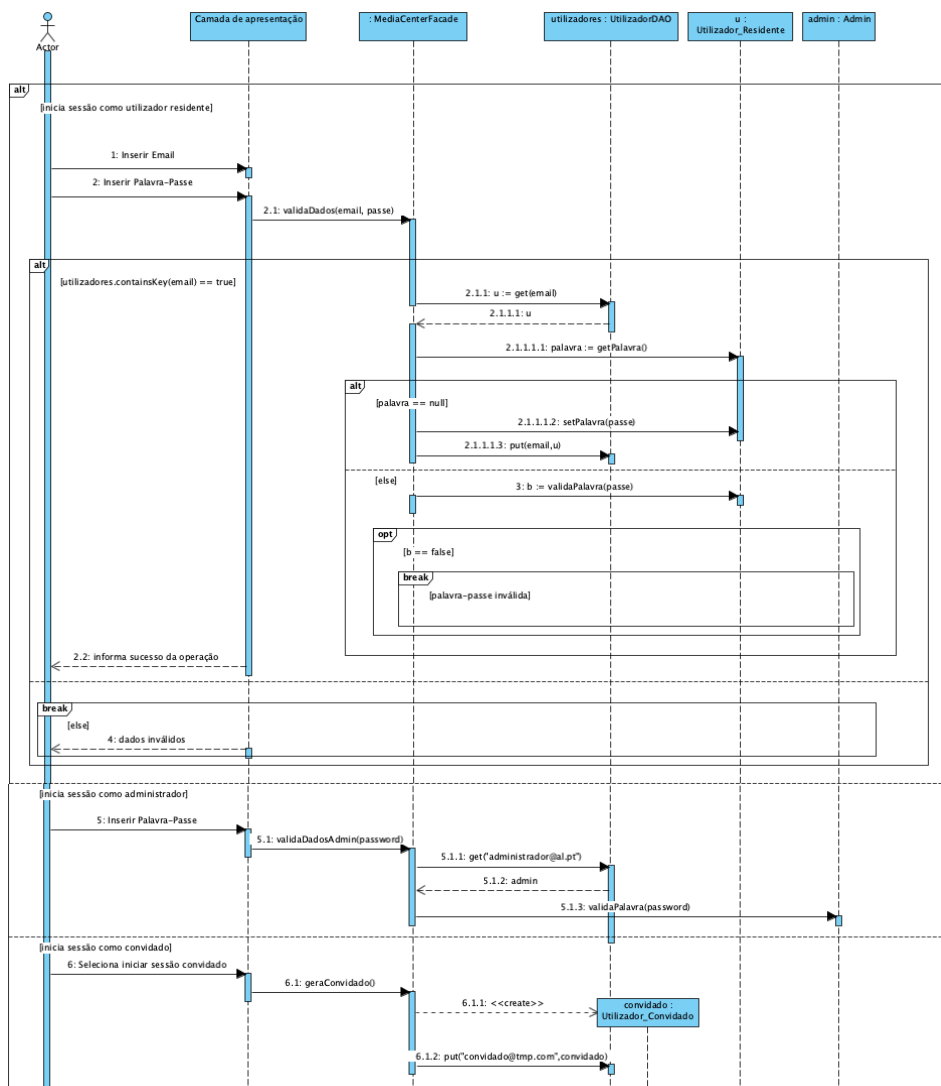


Diagrama 30: Diagrama de Sequência com DAO - iniciar sessão

2. Reproduzir Conteúdo

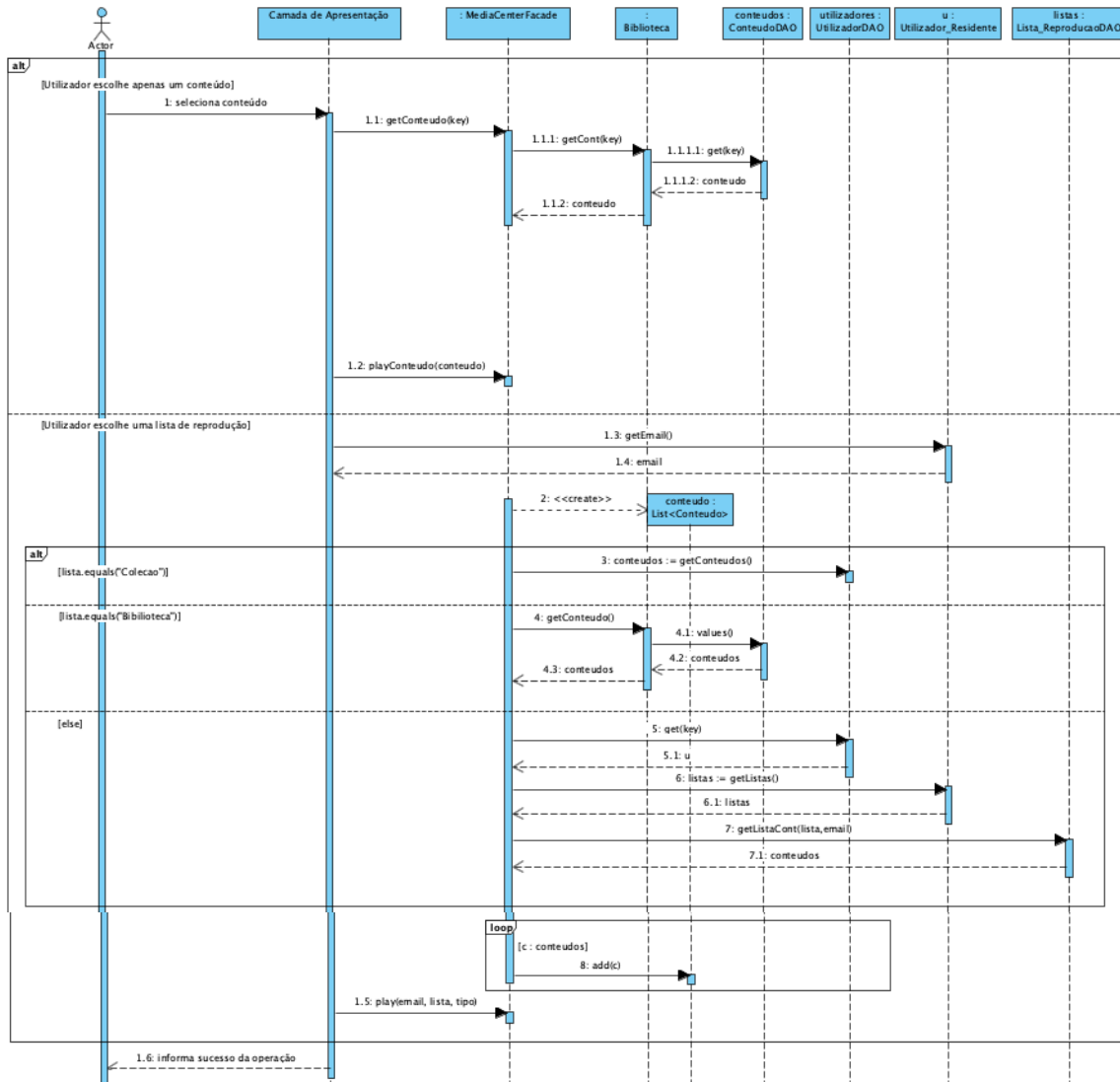


Diagrama 31: Diagrama de Sequência com DAO - registar utilizador

PROTÓTIPO INTERFACE

PÁGINA INICIAL



Figura 1: Página inicial Media Center

LOGIN ADMINISTRADOR

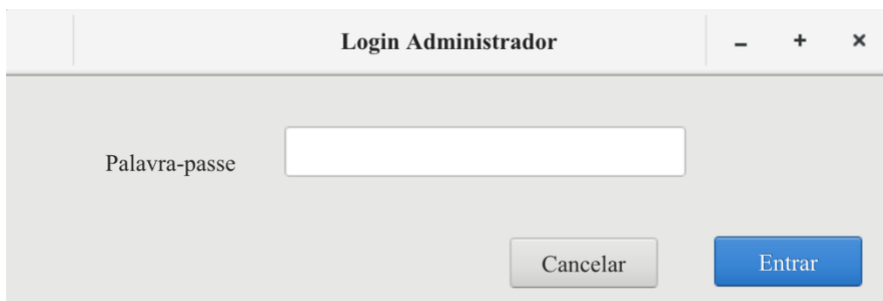


Figura 2: Login Administrador

ERRO LOGIN ADMINISTRADOR

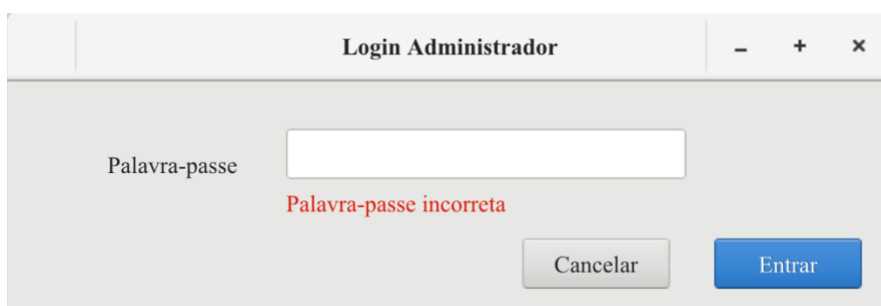



Figura 3: Erro tentativa início sessão administrador


LOGIN DE UTILIZADOR



The image shows a login window titled "Iniciar Sessão". It has a standard window header with minimize, maximize, and close buttons. The main area contains two input fields: "Email" and "Palavra-passe". Below the fields are two buttons: "Cancelar" (grey) and "Entrar" (blue).

Figura 4: Login de utilizador

ERRO LOGIN



The image shows the same login window as in Figure 4, but with an error message displayed in red text at the bottom left: "Email ou palavra-passe incorretos". The "Email" and "Palavra passe" fields are empty, and the "Cancelar" and "Entrar" buttons are still present.

Figura 5: Erro tentativa login

MENU ADMINISTRADOR

Nome	Email
Mário Silva	mario_silva15@gmail.com
Manuel Pereira	pereira.manu@gmail.com
Joana Costa	jujucosta@gmail.com

Página anterior Página seguinte

Registrar Utilizador Remover Utilizador Terminar Sessão

Figura 6: Menu Administrador

REGISTAR UTILIZADOR


Nome

Email

Cancelar Confirmar

Figura 7: Registrar utilizador

ERRO REGISTO UTILIZADOR



A dialog box titled "Registrar Utilizador" with standard window controls (minimize, maximize, close). It contains two input fields: "Nome" and "Email". Below the "Email" field, a red error message reads "Email introduzido já se encontra registado!". At the bottom right, there are two buttons: "Cancelar" (disabled) and "Confirmar" (active).

Figura 8: Erro no registo do utilizador

REMOVER UTILIZADOR



A dialog box titled "Remover Utilizador" with standard window controls (minimize, maximize, close). It contains a single input field labeled "Email". At the bottom right, there are two buttons: "Cancelar" (disabled) and "Confirmar" (active).

Figura 9: Remover utilizador

ERRO REMOÇÃO UTILIZADOR



The screenshot shows a window titled "Remover Utilizador" with standard window controls. Inside, there is a text input field labeled "Email". Below the field, a red error message reads "Email introduzido não se encontra registado!". At the bottom, there are two buttons: "Cancelar" and "Confirmar".

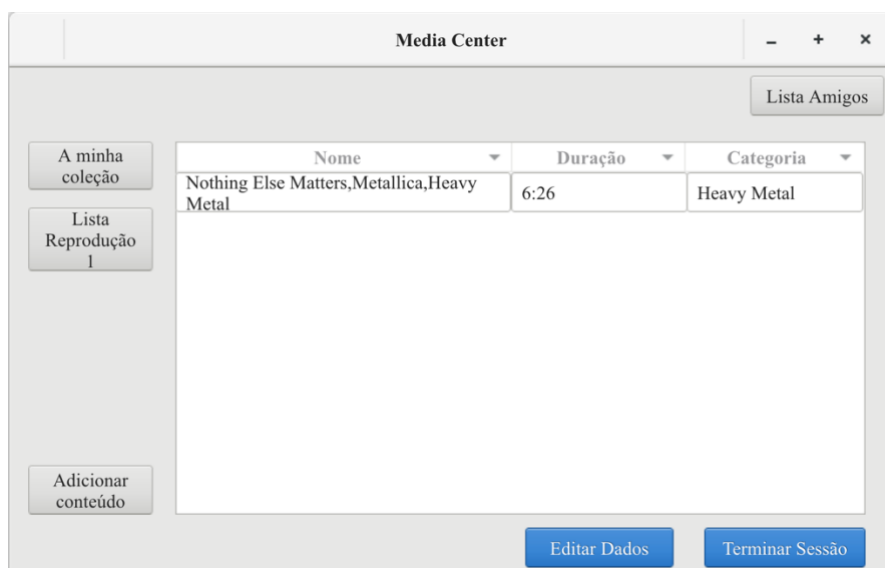
Email

Email introduzido não se encontra registado!

Cancelar Confirmar

Figura 10: Erro na tentativa de remoção de utilizador

MEDIA CENTER



The screenshot shows a window titled "Media Center" with standard window controls. On the left side, there are three buttons: "A minha coleção", "Lista Reprodução 1", and "Adicionar conteúdo". On the right side, there is a button labeled "Lista Amigos". The main area contains a table with three columns: "Nome", "Duração", and "Categoria". The table has one row of data. At the bottom right, there are two buttons: "Editar Dados" and "Terminar Sessão".

A minha coleção

Lista Reprodução 1

Adicionar conteúdo

Lista Amigos

Nome	Duração	Categoria
Nothing Else Matters,Metallica,Heavy Metal	6:26	Heavy Metal

Editar Dados Terminar Sessão

Figura 11: Media Center

CONVIDADO

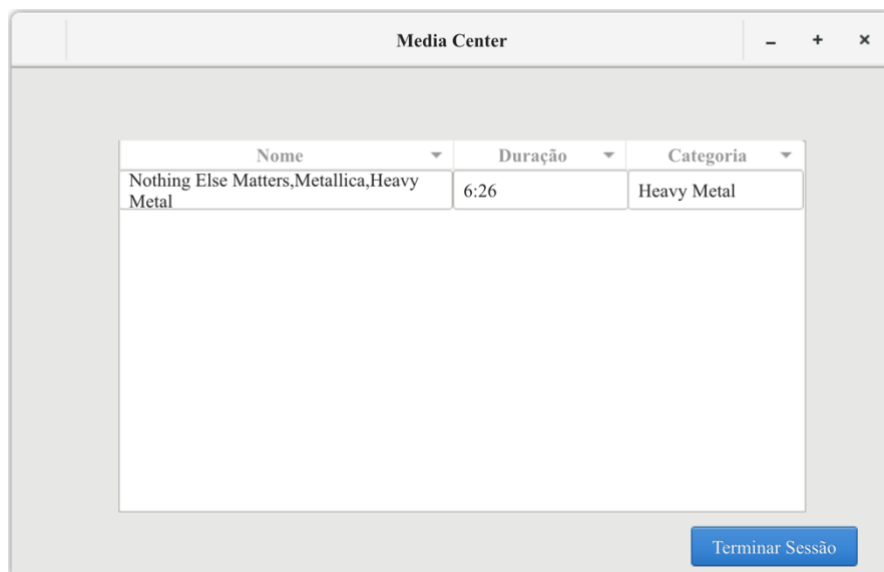


Figura 12: Media Center do utilizador convidado

A MINHA COLEÇÃO

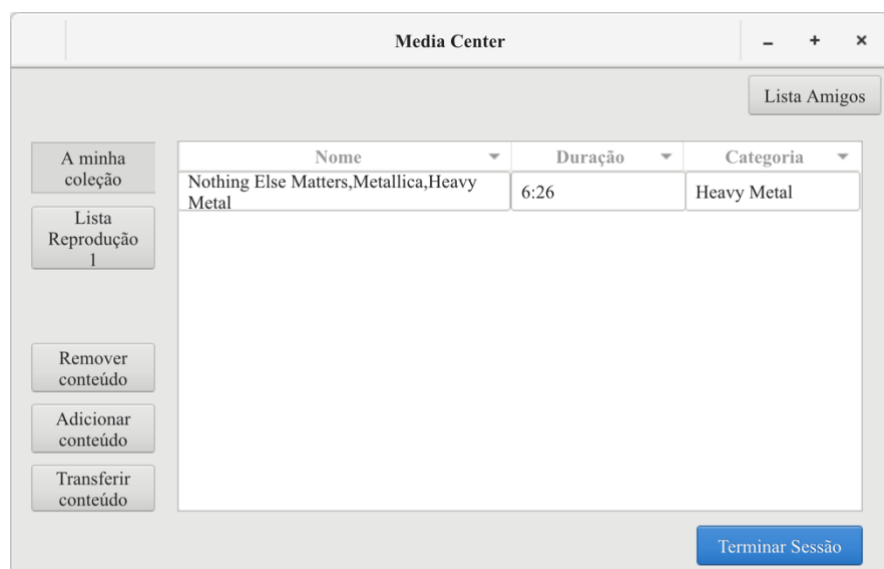


Figura 13: A minha coleção

LISTA DE REPRODUÇÃO

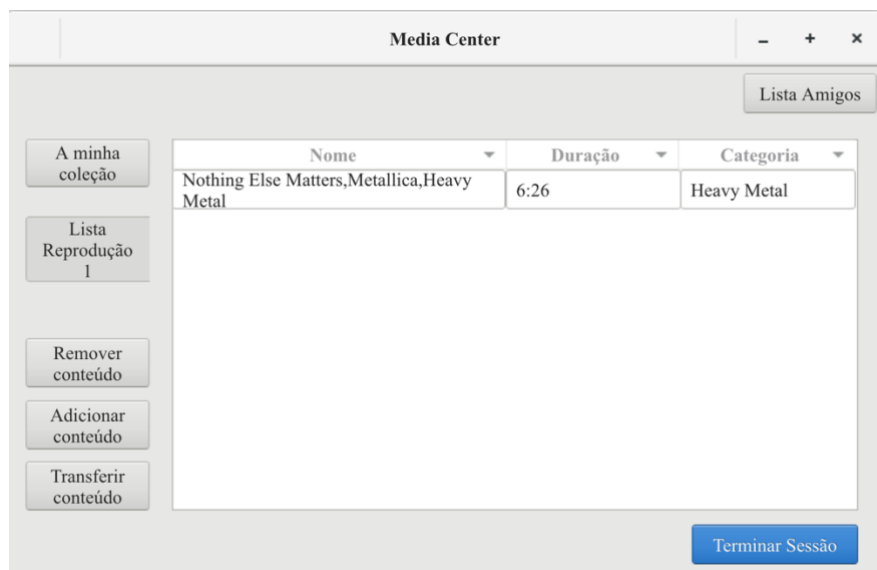


Figura 14: Lista de reprodução

ADICIONAR CONTEÚDO

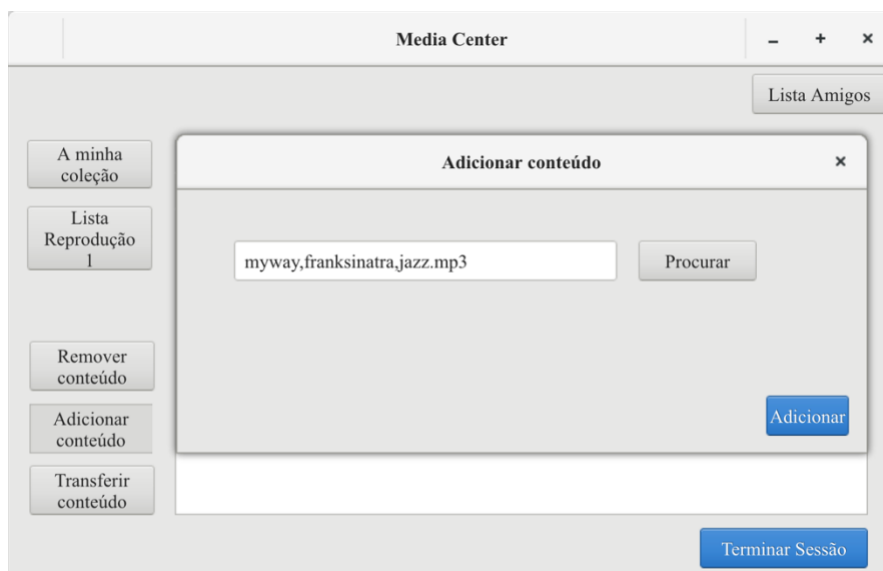


Figura 15: Adicionar conteúdo à coleção

REMOVER CONTEÚDO

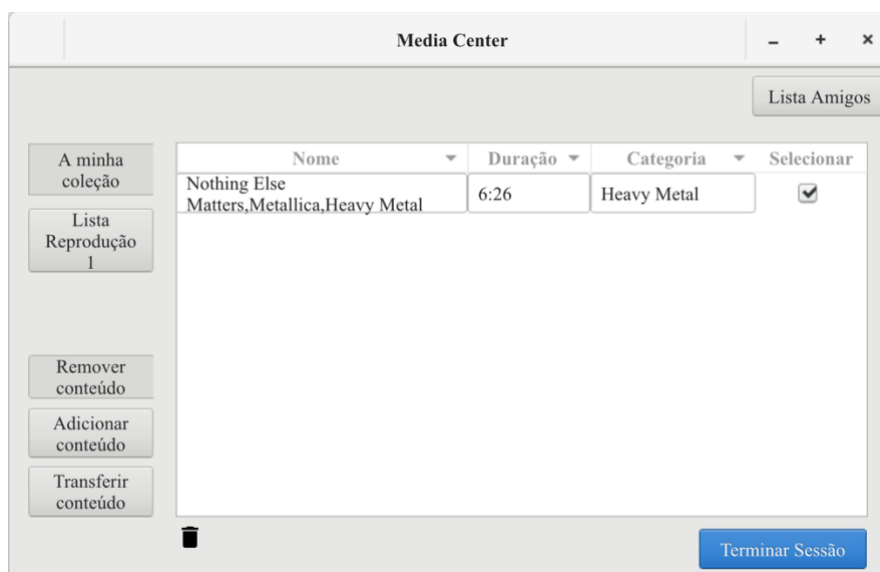


Figura 16: Remover conteúdo de uma coleção

TRANSFERIR CONTEÚDO

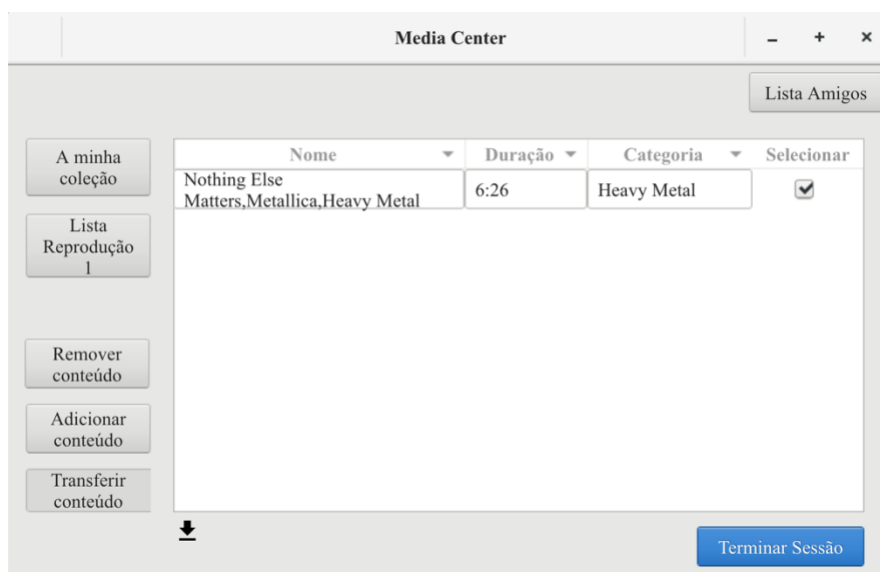
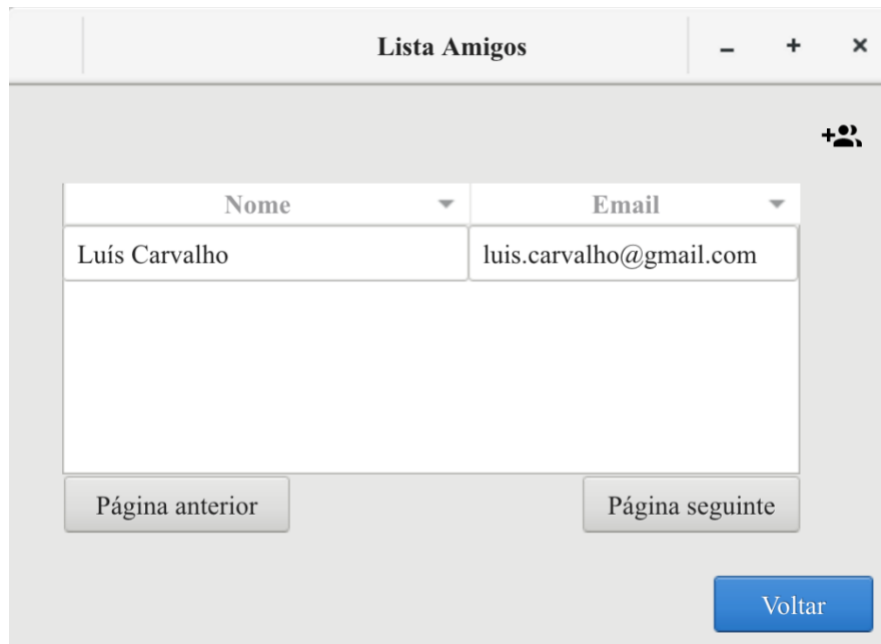


Figura 17: Transferir conteúdos de uma coleção

LISTA DE AMIGOS



The 'Lista Amigos' window displays a table with two columns: 'Nome' and 'Email'. It includes navigation buttons for 'Página anterior', 'Página seguinte', and 'Voltar'.

Nome	Email
Luís Carvalho	luis.carvalho@gmail.com

Página anterior Página seguinte

Voltar

Figura 18: Lista de amigos de um utilizador

ADICIONAR AMIGO



The 'Adicionar amigos' window displays a table with two columns: 'Nome' and 'Email'. It includes navigation buttons for 'Página anterior', 'Página seguinte', and a 'Cancelar' button.

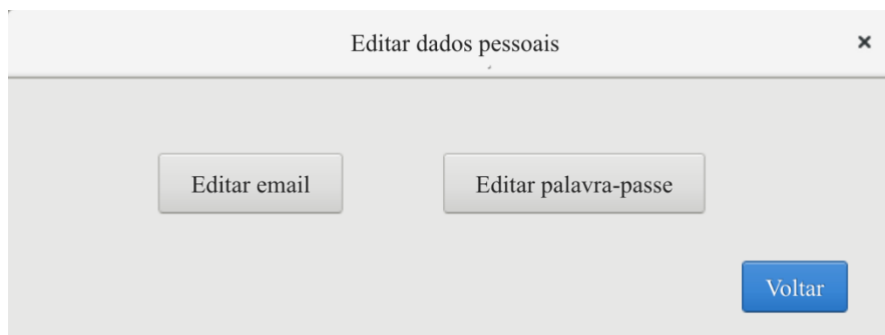
Nome	Email
Manuel Pereira	pereira.manu@gmail.com
Joana Costa	jujucosta@gmail.com

Página anterior Página seguinte

Cancelar

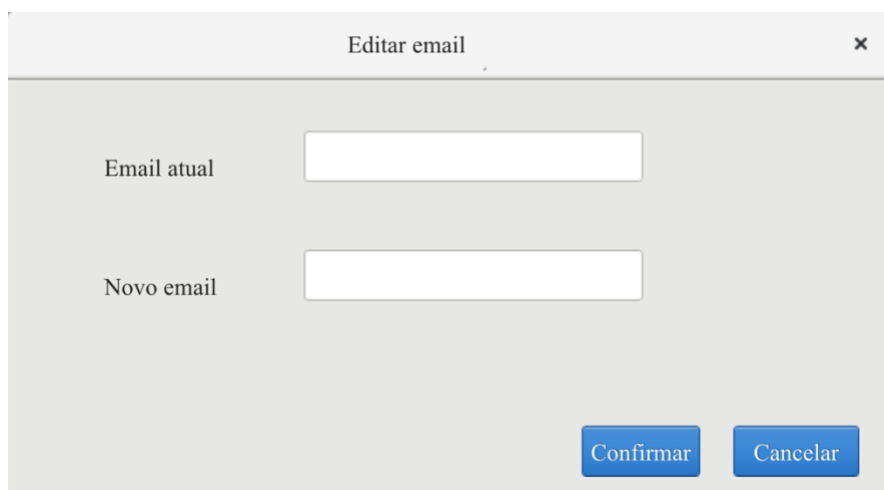
Figura 19: Adicionar amigo

EDITAR DADOS



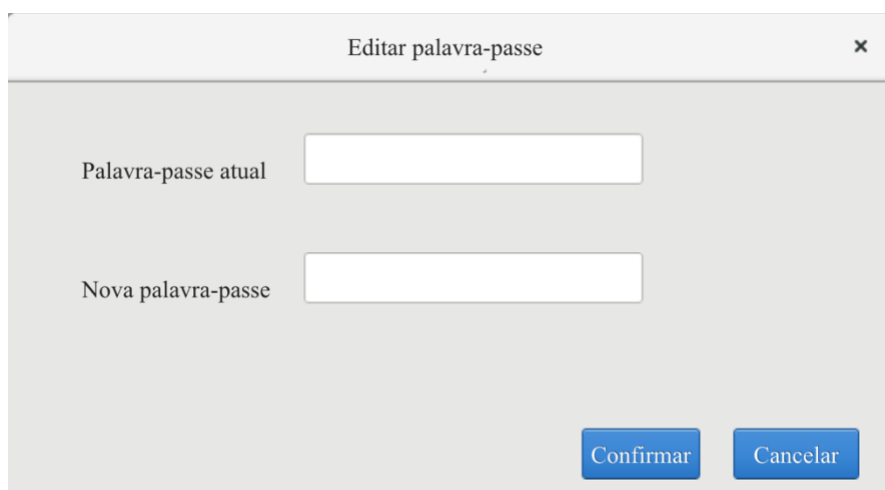
A dialog box titled "Editar dados pessoais" with a close button (X) in the top right corner. Inside the dialog, there are two buttons: "Editar email" and "Editar palavra-passe". At the bottom right, there is a blue button labeled "Voltar".

Figura 20: Editar dados pessoais



A dialog box titled "Editar email" with a close button (X) in the top right corner. It contains two text input fields. The first field is labeled "Email atual" and the second is labeled "Novo email". At the bottom right, there are two blue buttons: "Confirmar" and "Cancelar".

Figura 21: Editar email



A dialog box titled "Editar palavra-passe" with a close button (X) in the top right corner. It contains two text input fields. The first field is labeled "Palavra-passe atual" and the second is labeled "Nova palavra-passe". At the bottom right, there are two blue buttons: "Confirmar" and "Cancelar".

Figura 22: Editar palavra-passe

PRIMEIRA IMPLEMENTAÇÃO DOS USE CASES E EXPLORAÇÃO DE BIBLIOTECAS

Com o intuito de compreender o funcionamento de bibliotecas capazes de solucionar os objetivos finais, procedeu-se à criação de uma classe de teste, **MediaCenterFacade**, onde foi utilizado o *player* externo *VLC* para reproduzir músicas e vídeos. Desta forma, concretizou-se a implementação do *use case* ‘reproduzir conteúdo’, que permitiu a reprodução de um conteúdo singular ou de uma lista de reprodução. Com o intuito de averiguar a relação entre os modelos e o código, optou-se pela implementação de mais alguns dos *use case* apresentados, tais como ‘criar lista de reprodução’ e um protótipo de ‘registar utilizador’.

```
1  import java.util.Map;
2  import java.util.HashMap;
3  import java.util.List;
4  import java.util.ArrayList;
5
6  public class Administrador {
7      String nome;
8      String palavrapasse;
9
10     public Administrador(String nNome, String password){
11         nome = nNome;
12         palavrapasse = password;
13     }
14 }
15
16 public class Biblioteca {
17     Map<Integer, Conteudo> conteudos;
18
19     public Biblioteca(){
20         conteudos = new HashMap<Integer, Conteudo>();
21     }
22 }
23
24 public class Colecoes {
25     Map<String, Conteudo> conteudos;
26
27     public Colecoes(){
28         conteudos = new HashMap<String, Conteudo>();
29     }
30 }
31
32 public class Lista_Reproducao {
33     String nome;
34     String path;
35     List<Conteudo> conteudos;
36
37     public Lista_Reproducao(){
38         nome = "";
39         path = "";
40         conteudos = new ArrayList<Conteudo>();
41     }
42
43     public void setNome(String nNome){
44         nome = nNome;
45     }
46
47     public void setPath(String nPath){
48         path = nPath;
49     }
50
51     public List<Conteudo> getConteudos(){
52         return conteudos;
53     }
54
55     public String getPath(){
56         return path;
57     }
58 }
```

```

59
60 public class Utilizador_Residente {
61     String nome;
62     String email;
63     String palavrapasse;
64     List<Lista_Reproducao> listas;
65     Map<String,Utilizador_Residente> amigos;
66     Map<String,Utilizador_Residente> sugestoes;
67     Map<String,Utilizador_Residente> pedidos;
68
69     public Utilizador_Residente(){
70         nome = "";
71         email = "";
72         palavrapasse = null;
73         listas = new ArrayList<Lista_Reproducao>();
74         amigos = new HashMap<String,Utilizador_Residente>();
75         sugestoes = new HashMap<String,Utilizador_Residente>();
76         pedidos = new HashMap<String,Utilizador_Residente>();
77     }
78
79     public Utilizador_Residente(String nNome,String nEmail){
80         nome = nNome;
81         email = nEmail;
82         palavrapasse = "a";
83         listas = new ArrayList<Lista_Reproducao>();
84         amigos = new HashMap<String,Utilizador_Residente>();
85         sugestoes = new HashMap<String,Utilizador_Residente>();
86         pedidos = new HashMap<String,Utilizador_Residente>();
87     }
88
89     public void addList(Lista_Reproducao lista_nova){
90         listas.add(lista_nova);
91     }
92
93     public String getNome(){
94         return nome;
95     }
96
97     public List<Lista_Reproducao> getListas(){
98         return listas;
99     }
100 }
101
102 public class Conteudo {
103     String nome;
104     double tamanho;
105     double duracao;
106     int id;
107     String categoria;
108     String path;
109
110     public Conteudo(){
111         nome = "";
112         tamanho = 0.0;
113         duracao = 0.0;
114         id = 0;
115         categoria = "";
116         path = "";
117     }
118
119     public Conteudo(String nNome, String nCategoria, String nPath){
120         nome = nNome;
121         tamanho = 0.0;
122         duracao = 0.0;
123         id = 0;
124         categoria = nCategoria;
125         path = nPath;
126     }
127
128     public String getPath(){
129         return path;
130     }
131 }
132
133 import java.io.IOException;
134 import java.util.HashMap;
135 import java.util.Map;
136 import java.util.Scanner;
137 import java.util.List;
138 import java.util.ArrayList;
139 import java.util.stream.Collectors;
140 import java.lang.ProcessBuilder;
141 import java.lang.Process;
142
143 public class MediaCenterFacade {
144     private static transient Scanner input = new Scanner(System.in);
145     private static Map<String,Utilizador_Residente> utilizadores;
146     private static Biblioteca biblioteca;
147     private static String defaultPath;
148     private static Process runner;
149
150     public MediaCenterFacade(){
151         utilizadores = new HashMap<String,Utilizador_Residente>();
152         biblioteca = new Biblioteca();
153         defaultPath = "";
154     }
155
156     public void setPath(String nPath){
157         defaultPath = nPath;
158     }
159
160     public void addUtilizador(String nome, String email){
161         if(utilizadores.get(email) == null){
162             Utilizador_Residente user = new Utilizador_Residente(nome,email);
163             utilizadores.put(email,user);
164         }
165     }
166
167     public static Process playConteudo(String pathFile) throws IOException{
168         if(runner != null && runner.isAlive()) runner.destroy();
169         String pathVLC;
170         if (System.getProperty("os.name").contains("Windows")) pathVLC = "C:\\Program Files (x86)\\VideoLAN\\VLC\\vlc.exe";
171         else if (System.getProperty("os.name").contains("Mac")) pathVLC = "/Applications/VLC.app/Contents/MacOS/VLC";
172         else pathVLC = "vlc";
173         String play = "--play-and-exit";
174         ProcessBuilder pb;
175         pb = new ProcessBuilder(pathVLC, play, pathFile);
176         return pb.start();
177     }
178 }

```

```

46
47 public void createList(String email, List<Conteudo> l) throws IOException{
48     Lista_Reproducao lista_nova = new Lista_Reproducao();
49     System.out.println("Escreva o nome da lista");
50     String nome = input.nextLine();
51     lista_nova.setNome(nome);
52     ProcessBuilder pb = new ProcessBuilder();
53     String cmd;
54     String op;
55     String trueCmd;
56     String pathL;
57     if(System.getProperty("os.name").contains("Windows")){
58         pathL = defaultPath+"\\\\"+nome;
59         lista_nova.setPath(pathL);
60         cmd = "cmd.exe";
61         op = "/c";
62         trueCmd = "mkdir ";
63         pb.command(cmd , op, trueCmd, pathL);
64         pb.start();
65         trueCmd = "copy ";
66     }
67     else{
68         pathL = defaultPath+"/"+nome;
69         lista_nova.setPath(pathL);
70         cmd = "bash";
71         op = "-c";
72         trueCmd = "mkdir ";
73         pb.command(cmd , op, trueCmd + pathL);
74         pb.start();
75         trueCmd = "cp ";
76     }
77     for (Conteudo cont : l){
78         pb = new ProcessBuilder(cmd, op, trueCmd + cont.getPath() + " " + pathL);
79         pb.start();
80         lista_nova.getConteudos().add(cont);
81     }
82     utilizadores.get(email).addList(lista_nova);
83 }
84
85 public Map<String,Utilizador_Residente> getUsers(){
86     return utilizadores;
87 }
88
89 public static void main(String args[]){
90
91     MediaCenterFacade trial = new MediaCenterFacade();
92     System.out.println("Escreva o path onde quer criar listas de reprodução");
93     String firstPath = input.nextLine();
94     trial.setPath(firstPath);
95
96     System.out.println("Escreva o nome do utilizador");
97     String nome = input.nextLine();
98
99     System.out.println("Escreva o email do utilizador");
100    String email = input.nextLine();
101
102    trial.addUtilizador(nome,email);
103
104    System.out.println("Escreva o nome do conteudo");
105    nome = input.nextLine();
106
107    System.out.println("Escreva a categoria do conteudo");
108    String categoria = input.nextLine();
109
110    System.out.println("Escreva o path do conteudo");
111    String path = input.nextLine();
112
113    List<Conteudo> a = new ArrayList<Conteudo>();
114    a.add(new Conteudo(nome, categoria, path));
115
116    try{
117        trial.createList(email,a);
118    }
119
120    catch(IOException e){
121        e.printStackTrace();
122    }
123
124    try{
125        runner = trial.playConteudo(trial.getUsers().get(email).getListas().get(0).getPath());
126    }
127    catch(IOException e){
128        e.printStackTrace();
129    }
130 }
131

```

BASE DE DADOS

De forma a agrupar os dados num sistema de bases de dados, foi elaborado o seguinte modelo lógico, através do *MySQL Workbench*.

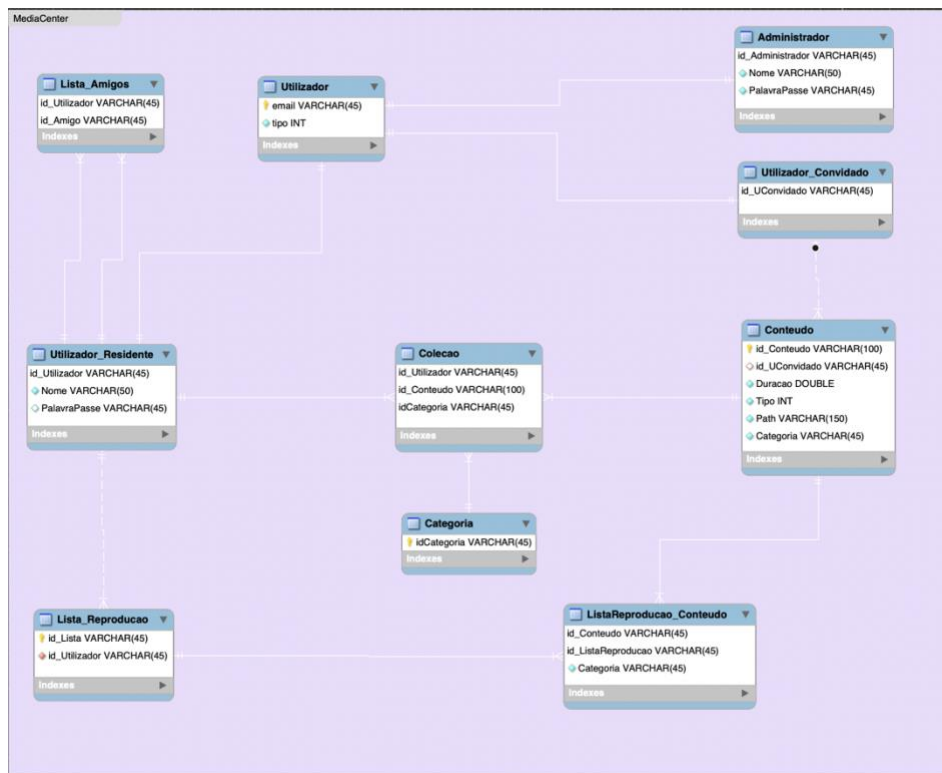


Figura 23: Modelo Lógico

A tabela `Lista_Amigos` foi criada devido à possibilidade de um utilizador residente possuir o contacto de outros utilizadores residentes, sendo por isso composta por uma chave primária (composta), o `id_Utilizador` e o `id_Amigo`.

Por sua vez, o relacionamento ternário entre `Utilizador_Residente`, `Categoria` e `Conteudo` deu origem a uma outra tabela, `Colecao`, que permite aos vários utilizadores atribuírem uma nova categoria a um conteúdo desejado, pelo que o atributo `Categoria` em `Conteudo` define a categoria originalmente introduzida.

CONCLUSÕES

Com a realização do presente trabalho foi possível abordar de uma forma mais realista e intrínseca uma problemática relacionada a um conceito aplicacional, o *mediacenter*.

Numa primeira fase, optou-se pela conceção dos modelos como se encontram detalhadamente explicados acima. É ainda crucial salientar que a solução para uma implementação eficaz dos mesmos emergiu de uma deliberação acentuada acerca das funcionalidades que uma aplicação como a apresentada deveria possuir.

A segunda fase carregou consigo uma inquietação adicional que originou a que o grupo se entregasse à consciencialização, apoderando-se do conceito abrangente a todo o projeto, o facto de se tratar de um método iterativo e incremental. Como tal, para dar resposta à busca incessante pelo rigor e excelência, o foco adquiriu um trajeto bidirecional, sendo este concentrado na alteração do previamente feito e na elaboração do proposto para esta fase em específico. Por conseguinte, de modo a satisfazer a necessidade de aprimorar o já realizado, procedeu-se à eliminação da entidade “palavra-passe” do modelo de domínio, uma vez que restringia futuramente o modo como se poderia implementar a aplicação, também se considerou imprescindível adicionar os use cases “editar dados”, “terminar sessão” e “iniciar sessão”.

Após a alteração efetuada com vista num objetivo futuro de implementação, o grupo centrou-se na elaboração dos diagramas dos use cases aventados pelo docente. Primeiramente, ponderou-se acerca do modo de execução faseada dos diagramas, de seguida realizaram-se os diagramas de sequência de sistema, posteriormente a atenção esteve nos diagramas de *packages* e por fim, fez-se o diagrama de classes (este apresenta uma particularidade, utilizou-se como chave do **Map** de utilizadores os seus emails – apesar do efetuado, o grupo dispõe de total consciência que, para uma aplicação de grandes dimensões, esta abordagem não seria a indicada, uma vez que a utilização de *Strings* como chaves tornar-se-ia ineficiente) e a partir destes realizaram-se os diagramas de sequência com subsistemas, que mais tarde evoluíram para diagramas de sequência de implementação.

O programa usado para começar a implementação a nível de código foi o *VLC*, uma vez que na visão do grupo este foi o que pareceu mais adequado relativamente à problemática em questão. Utilizou-se ainda o *ProcessBuilder*, visto que este permite a

abertura do VLC em modo semelhante a um terminal, o que oferece um maior controlo, além do que é possível fechar o VLC sempre que se considere necessário.

Numa terceira fase, imbuídos da mesma consciencialização relativamente ao método iterativo e incremental subjacente ao projeto, modificou-se o use case “alterar categoria”, outrora funcional apenas na coleção de cada utilizador residente e, neste momento, cumpre o requisito proposto inicialmente (isto é, um utilizador proceder à alteração de uma categoria unicamente na sua coleção, não interferindo assim, com a escolha prévia existente na biblioteca). No que diz respeito às dificuldades sentidas pelo grupo, este encontrou especial impacto na aprendizagem da conciliação de SQL com JAVA, percalço rapidamente ultrapassado após breves pesquisas serem efetuadas.

Enaltecidos pela vontade de querer ir mais além, o grupo reajustou/reorganizou o seu tempo a fim de conseguir não só concluir os cinco use cases obrigatórios, bem como realizar quatro use cases adicionais (tudo isto, dos dez use cases especificados na fase 2).

Por fim, é de sublinhar a forma organizadamente distributiva que o grupo descobriu para melhor demonstrar a relação de dependência entre conceitos teóricos e implementação prática dos mesmos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Nunes, M. and O'Neill, H. (2011). Fundamental de UML. 7th ed. Lisboa: FCA - Editora de informática, Lda.