CENTRALMUSIC - LDS

TEST DESIGN SPECIFICATION

Version 1.2

16/02/2021

Histórico de Versões

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*Samuel Cunha*	*15/02/2021*	*Samuel Cunha*	*15/02/2021*	Criação do documento e adaptação para o projeto
1.1	*Samuel Cunha Marcelo Carvalho*	*15/02/2021*	*Samuel Cunha*	*15/02/2021*	Adição da especificação dos testes de algumas componentes
1.2	*Samuel Cunha*	*15/02/20201*	*Samuel Cunha*	*16/02/2021*	Âmbito, finalização dos componentes UserDAO, LoginDAO, ChatDAO e PublicationDAO Revisão

Tabela de Conteúdos

1. Introdução

1.1. Identificador do documento

TDS15022021V1.2

Descrição do identificador do projeto:
TDS - TestDesignSpecification
15 - Número do mês (dois dígitos seguidos dos dois dígitos iniciais)
02 - Dia do mês (dois dígitos iniciais)
2021 - Ano de criação (quatro dígitos finais)
V1.2 - Número de versão do projeto (dois caracteres sendo "V" o identificador de versão através de sigla seguido do número de versão "1" e número de alteração ".2")

1.2. Âmbito

O produto a ser testado é uma aplicação mobile que tem como objetivo a troca e venda de instrumentos musicais.Este sistema será composto por uma aplicação para dispositivos móveis desenvolvida em Flutter e, para a disponibilização dos serviços de negócio, irá ter uma aplicação no *backend* desenvolvida em ASP.net core.

Nesta versão do documento apenas vão ser testadas as classes de *data access* da API. Os controladores podem ser testados utilizando a documentação *swagger*.

Nota: Esta versão do documento não representa o seu estado final. Este documento pode ser submetido a futuras atualizações.

Funcionalidades a testar:

- *UserDAO*
  - Create – Criar um utilizador com os seus dados pessoais;
  - Update – Atualizar os dados pessoais do utilizador;
  - UpdatePassword – Atualizar a password do utilizador;
  - GetPublications – Obter a lista de publicações do utilizador;
  - UpdatePublication – Atualizar os dados da publicação do utilizador;
  - DeletePublication – Eliminar a publicação do utilizador;
  - GetPublicationsFromFavorites – Obter as publicações que são adicionadas aos favoritos;
  - DeletePublicationFromFavorites – Eliminar uma publicação que esteja na lista de favoritos do utilizador;
  - FindUserByEmail – Procurar um utilizador pelo e-mail;
  - FindById – Procurar um utilizador pelo ID.
- *LoginDAO*
  - Authenticate – Autenticar um utilizador;
  - RecoverPassword – Recuperar a password do utilizador porque a esqueceu.
- *PublicationDAO*
  - Create – Criar uma publicação com os dados relativos à mesma;
  - `FindById` – Procurar uma publicação pelo seu ID;
  - GetPublications – Obter todas as publicações (com ou sem filtro);
  - AddPublicationToFavourites - Adicionar uma publicação à lista de favoritos de um utilizador.
- *ChatDAO*
  - AddMessage – Adicionar uma mensagem para permitir a comunicação entre dois utilizadores.
  - CreateChat – Criar *chats* com referências de dois utilizadores para estes poderem enviar mensagens entre si.
  - GetChatId– Obtém o Id do chat que um utilizador tem com outro.
  - GetMessageList – Visualizar uma lista de mensagens associadas a um *chat*.
  - GetChats – Visualizar um *array* de chats associados a um utilizador.

1.3. Referências

Lista de referências/outros documentos relevantes para este documento:

- Test Case Outline;
- https://gitlab.estg.ipp.pt/8160526/centralmusic

1.4 Glossário

MC – Marcelo Carvalho
SC – Samuel Cunha

TC – Test Case (Caso de Test)
ECP - Equivalence Class Partitioning
EC – Equivalence Class (Classe de Equivalência)
ECs – Classes de equivalência
BVA - Boundary Value Analysis

2. Features/Itens a testar

Item a testar	Descrição das features	Requisitos	Responsabilidade
UserDAO	**Create**– permite criar um utilizador com os seus dados pessoais	Criar um utilizador	SC
	Update – permite editar os dados pessoais de um utilizador	Editar utilizador	SC
	UpdatePassword – permite atualizar a password de um utilizador	Atualizar password	SC
	GetPublications – obter a lista de publicações do utilizador	Obter publicações do utilizador	SC
	UpdatePublication – permite editar uma publicação do utilizador	Editar publicação	SC
	DeletePublication – permite eliminar uma publicação do utilizador	Eliminar publicação	SC
	GetPublicationFromFavorites – obter as publicações favoritas de um utilizador	Obter publicações favoritas	SC
	DeletePublicationFromFavorites – eliminar uma publicação da lista de favoritos	Eliminar publicação dos favoritos	SC
	FindUserByEmail – procurar um utilizador pelo e-mail	Procurar utilizador pelo e-mail	SC
	FindById – procurar um utilizador pelo ID	Procurar utilizador pelo ID	SC
LoginDAO	***Authenticate*** – permite autenticar um utilizador	Autenticar utilizador	SC
	RecoverPassword– permite alterar a password do utilizador	Recuperar password	SC
PublicationDAO	***Create*** – permite criar uma publicação com os dados requiridos	Criar publicação	MC
	FindById– permite procurar uma publicação pelo seu ID	Obter publicação pelo ID	MC
	GetPublications – Obter todas as publicações (com ou sem filtro)	Obter publicações (com ou sem filtro)	MC
	AddPublicationToFavorites – permite adicionar uma publicação aos favoritos do utilizador	Adicionar publicação aos favoritos	MC
ChatDAO	***AddMessage*** – permite a comunicação entre dois utilizadores	Enviar mensagem	MC
	CreateChat – *chats* com referências de dois utilizadores para estes poderem enviar mensagens entre si	Chat para comunicação entre dois utilizadores	MC
	GetChatId – obter o ID do chat que um utilizador tem com o outro	Obter ID de chat	MC
	GetMessageList – permite visualizar a lista de mensagens associadas a um chat	Lista de mensagens de chat	MC
	GetChats – permite visualizar um array de chats associados a um utilizador	Chats associados a um utilizador	MC

3. Detalhes da abordagem aos testes

A abordagem a ser utilizada para os testes dos itens referidos na tabela da secção anterior é uma abordagem *BlackBox*. Esta estratégia avalia o comportamento externo de software sem considerar o comportamento interno do mesmo, ou seja, são apenas consideradas as entradas e as saídas como uma base para o desenho dos casos de teste. Dentro desta abordagem, usaram-se dois métodos que foram usados para o planeamento dos casos de teste, cujos métodos são: *Equivalence Class Partitioning* (Partição em classes de equivalência / ECP) e *Boundary Value Analysis* (Análise de valor limite / BVA), ambos lecionados em unidades curriculares anteriores.

Equivalence Class Partitioning: É uma técnica destinada a reduzir o número de testes necessários e que divide o domínio de entrada (ou saída) em classes de dados em que os casos de teste podem ser derivados.

Boundary Value Analysis: É uma técnica focada nos limites do domínio de entrada (ou saída) e imediatamente acima e abaixo (além de ou em vez de valores intermédios).

A inspeção visual através do IDE (*Visual Studio*) foi o método usado para a análise dos resultados de teste. Para o apoio da seleção de casos de teste foram criadas classes de equivalência válidas e inválidas que definiram os critérios de passagem e de falha dos *test cases*, bem como os tipos de *input*. O suporte à seleção de *test cases* também foi influenciado pelas partições do intervalo de valores com a técnica de BVA, com o objetivo de testar os limites desta coleção.

4. Identificação dos Testes

4.1 Casos de teste das features do componente *UserDAO*

4.1.1 Método *Create*

TC1 - CanRegisterUserTest:
O TC1 tem como objetivo testar se o método *Create* cria um utilizador na base de dados e retorna o utilizador criado.

4.1.2 Método *Update*

TC1 – CanUserUpdateProfile:
O TC1 tem como objetivo testar se o método *Update* consegue editar as informações do utilizador com sucesso, adicionando um utilizador, acrescentando-lhe de seguida dados atualizados e fazendo a comparação dos novos dados pessoais enviados para o método com os dados que foram recebidos pelo método. Neste caso, comparou-se o e-mail, primeiro nome, apelido, morada e imagem.

4.1.3 Método *UpdatePassword*

TC1 – CanUserChangePassword:
O TC1 tem como objetivo testar se o método *UpdatePassword* edita a palavra passe do utilizador com sucesso, verificando se o método retorna *True* quando se efetua a alteração da palavra passe, introduzido a antiga palavra passe corretamente e a nova palavra passe.

4.1.4 Método *GetPublications*

TC1 - CanUserGetHisOwnPublications:
O TC1 tem como objetivo testar se o método *GetPublications* vai buscar todas as publicações ativas do utilizador.

4.1.5 Método *UpdatePublication*

TC1 - CanUserUpdatePost:
O TC1 tem como objetivo testar se o método *UpdatePublication* edita os dados da publicação do utilizador.

4.1.6 Método *DeletePublication*

TC1 - CanUserDeletePublication:
O TC1 tem como objetivo testar se o método *DeletePublication* elimina permanentemente uma publicação ativa do utilizador.

4.1.7 Método *GetPublicationFromFavorites*

TC1 - CanUserListFavoritePublications:
O TC1 tem como objetivo testar se o método *GetPublicationFromFavorites* vai buscar todas as publicações que foram adicionadas à lista de favoritos.

4.1.8 Método *DeletePublicationFromFavorites*

TC1 - CanUserRemoveFavoritePublication:
O TC1 tem como objetivo testar se o método *DeletePublicationFromFavorites* elimina a publicação da lista de favoritos do utilizador.

4.1.9 Método *FindUserByEmail*

TC1 - FindUserByEmail:
O TC1 tem como objetivo testar se o método *FindUserByEmail* irá buscar um utilizador pelo e-mail com sucesso.

TC2 - FindUserByEmailNull:
O TC1 tem como objetivo testar se o método *FindUserByEmail* buscar um utilizador nulo pelo e-mail com sucesso.

4.1.10 Método *FindById*

TC1 - FindUserById:
O TC1 tem como objetivo testar se o método *FindByIdl* irá buscar um utilizador pelo Id de utilizador com sucesso.

4.2 Casos de teste das features do componente *LoginDAO*

4.2.1 Método *Authenticate*

TC1 – CanUserLoginWithCorrectPasswordTest:

O TC1 tem como objetivo testar se um utilizador consegue fazer o *login* com a *password* correta. É chamado o método *VerifyHash* para verificar se a *password* é a mesma com que o utilizador se encontra registado. Para o teste ser considerado como bem-sucedido, o método deve retornar um booleano de verdadeiro.

TC2 – CanUserLoginWithWrongPasswordTest:

O TC2 tem como objetivo testar se o utilizador consegue fazer o *login* com a *password* incorreta. É chamado o método *VerifyHash* para verificar se a *password* é a mesma com que o utilizador se encontra registado. Para o teste ser considerado como bem-sucedido, o método deve retornar um booleano de falso.

4.2.2 Método *RecoverPassword*

TC1 – CanEmployerUserRecoverPassword:

O TC1 tem como objetivo testar se um utilizador consegue recuperar a sua palavra-passe. É chamado o método *NewPasswordRequest* onde é enviado um e-mail ao utilizador para proceder à alteração da mesma. Para o teste ser considerado como bem-sucedido, o método deve retornar um booleano de verdadeiro.

4.3 Casos de teste das features do componente *PublicationDAO*

4.3.1 Método *Create*

TC1 – CanAddPublication:
O TC1 tem como objetivo testar se o método *Create* cria um uma publicação na base de dados.

4.3.2 Método *FindById*

TC1 – FindById:
O TC1 tem como objetivo testar se o método *FindById* procura uma publicação pelo ID.

4.3.3 Método *GetPublications*

TC1 – CanSearchPublicationByCategorie:
O TC1 tem como objetivo testar se o método *GetPublication* consegue procurar um anúncio/publicação através do filtro por categorias.

TC2 – CanSearchPublicationByDistance:
O TC2 tem como objetivo testar se o método *GetPublication* consegue procurar um anúncio/publicação através do filtro por distância.

TC3 – CanSearchPublicationByKeyWord:
O TC1 tem como objetivo testar se o método *GetPublication* consegue procurar um anúncio/publicação através de palavra chave, ou seja, pela barra de pesquisa.

4.3.4 Método *AddPublicationToFavorites*

TC1 – CanAddPublicationToFavorites:
O TC1 tem como objetivo testar se o método *AddToFavorites* consegue adicionar uma publicação à lista de favoritos do utilizador

4.4 Casos de teste das features do componente *ChatDAO*

4.4.1 Método AddMessage

TC1 – CanAddMessage:

O TC1 tem como objetivo testar se o método retorna *true* quando um utilizador envia uma mensagem a outro utilizador online, e o servidor tenta guardar a mensagem na BD. Para o teste passar é necessário o utilizador enviar a mensagem.

O teste é considerado com sucesso quando o método retorna *true* e a mensagem é adicionada a BD.

4.4.2 Método *CreateChat*

TC1 – CanAddChats:

O TC1 tem como objetivo testar se o método cria um chat, e testa também se o chat foi criado para dois utilizadores.

O teste passa quando o chat e criado e o id de chat é igual para dois utilizadores.

4.4.3 Métodos *GetMessageList*

TC1 – CanGetMessages:

O TC1 tem como objetivo testar se o método retorna uma lista de mensagens de um chat.

O teste passa quando o utilizador abre um chat ao qual pertence e recebe as mensagens todas.

4.4.4 Método GetChats

TC1 – CanGetChats:

O TC1 tem como objetivo testar se o método retorna todos os chats ao qual o utilizador pertence.

O teste passa quando o utilizador recebe a sua lista de chats.

5. Critérios de passagem ou falha das features

Os critérios de passagem e de falha das features foram definidos em classes de equivalência e em intervalos de input com a técnica de BVA. As subsecções a seguir pretendem mostrar para cada feature, as classes de equivalência válidas e inválidas e os intervalos de BVA (caso se aplique) de modo a definir se a feature passa ou não.

5.1 ECs para as features do componente *UserDAO*

5.1.1 Método Create

Classes de equivalência

EC1 – Válido: O input é o utilizador com as suas informações. O resultado é o utilizador retornado e adicionado á base de dados.

EC2 – Inválido: O input é o utilizador com as suas informações. O resultado é um objeto retornado diferente de utilizador e não ser adicionado á base de dados.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um objeto de utilizador	Um objeto diferente de utilizador
Valor Específico	EC1	EC2

5.1.2 Método Update

Classes de equivalência

EC1 – Válido: Os inputs são o utilizador com as informações atualizadas e o Id do utilizador que se pretende atualizar. O resultado é o utilizador com as informações atualizadas.

EC2 – Inválido: Os inputs são o utilizador com as informações atualizadas e o Id do utilizador que se pretende atualizar. O resultado é o utilizador com as informações antigas.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	2	0, >2 ou <2
Tipo de Entrada	Um número inteiro que representa um id do utilizador, e um objeto de utilizador	Um número inteiro menor que 1 ou nulo. Um objeto diferente de utilizador
Valor Específico	EC1	EC2

5.1.3 Método *UpdatePassword*

Classes de equivalência

EC1 – Válido: Os inputs são o utilizador com a password que pretende o Id do utilizador que se pretende atualizar a password. O resultado é o utilizador com a password atualizada.

EC2 – Inválido: Os inputs são o utilizador com a password que pretende e o Id do utilizador que se pretende atualizar a password. O resultado é o utilizador com a password antiga.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	2	0, >2 ou <2
Tipo de Entrada	Um número inteiro que representa um id do utilizador, e um objeto de atualização de pass do utilizador	Um número inteiro menor ou maior que 2 ou nulo. Um objeto diferente de atualização de pass do utilizador
Valor Específico	EC1	EC2

5.1.4 *Método GetPublications*

Classes de equivalência

EC1 – Válido: O input é Id do utilizador que pretende aceder à sua lista de anúncios/publicações. O resultado é uma coleção que representa a lista de publicações desse mesmo utilizador.

EC2 – Inválido: O input é Id do utilizador que pretende aceder à sua lista de anúncios/publicações. O resultado é uma coleção diferente daquela que deveria ser apresentada.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um número inteiro que representa um id do utilizador	Um número inteiro menor que 1 ou nulo
Valor Específico	EC1	EC2

5.1.5 Método *UpdatePublication*

Classes de equivalência

EC1 – Válido: O input é a publicação que se pretende atualizar. O resultado é a publicação devidamente atualizada.

EC2 – Inválido: O input é a publicação que se pretende atualizar. O resultado é a publicação antiga.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um objeto de atualização de publicação do utilizador	Um objeto diferente de atualização de publicação do utilizador
Valor Específico	EC1	EC2

5.1.6 Método *DeletePublication*

Classes de equivalência

EC1 – Válido: O input é o ID da publicação que se pretende eliminar. O resultado é a publicação devidamente eliminada.

EC2 – Inválido: O input é o ID da publicação que se pretende eliminar. O resultado é a publicação ainda ativa.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um número inteiro que representa o ID da publicação	Um número maior ou menor que 1 ou nulo
Valor Específico	EC1	EC2

5.1.7 Método *GetPublicationFromFavorites*

Classes de equivalência

EC1 – Válido: O input é o ID da publicação que se pretende visualizar. O resultado é a publicação devidamente apresentada.

EC2 – Inválido: O input é o ID da publicação que se pretende visualizar. O resultado é a não visualização da publicação pedida.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um número inteiro que representa o ID da publicação	Um número maior ou menor que 1 ou nulo
Valor Específico	EC1	EC2

5.1.8 Método *DeletePublicationFromFavorites*

Classes de equivalência

EC1 – Válido: Os inputs são o ID do utilizador referente à publicação e o ID da publicação que se pretende eliminar dos favoritos. O resultado é a publicação devidamente removida dos favoritos.

EC1 – Válido: Os inputs são o ID do utilizador referente à publicação e o ID da publicação que se pretende eliminar dos favoritos. O resultado é a publicação devida não ser removida dos favoritos.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	2	0, >2 ou <2
Tipo de Entrada	Um número inteiro que representa o ID da publicação e outro que representa o ID do utilizador	Um número maior ou menor que 2 ou nulo
Valor Específico	EC1	EC2

5.1.9 Método FindUserByEmail

Classes de equivalência

EC1 – Válido: O input é uma string com o e-mail do utilizador pretendido. O resultado é um objeto utilizador com a informação relativa ao mesmo.

EC2 – Inválido: O input é uma string com o e-mail do utilizador pretendido. O resultado é um objeto diferente do utilizador.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Uma string que representa o e-mail do utilizador	Um input maior ou menor que 1 ou nulo.
Valor Específico	EC1	EC2

5.1.10 Método FindById

Classes de equivalência

EC1 – Válido: O input é o Id do utilizador pretendido. O resultado é um objeto *User* com a informação relativa ao utilizador com o id de parâmetro.

EC2 – Inválido: O input é o Id do utilizador pretendido. O resultado é um objeto *User* com a informação diferente relativa ao utilizador pretendido.

EC3 – Válido: O input é o Id de um utilizador inexistente. O resultado é null.

EC4 – Inválido: O input é o Id de um utilizador inexistente. O resultado é um objeto *User* com a informação de um utilizador.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um número inteiro que representa um id de utilizador	Um número inteiro menor que 1 ou nulo.
Valor Específico	EC1, EC3	EC2, EC4

5.2 ECs para as features do componente *LoginDAO*

5.2.1 Método *Authenticate*

Classes de equivalência

EC1 – Válido: Os inputs são o email e a password do utilizador que pretende fazer login. O resultado esperado é um user válido com um email e password existentes.

EC2 – Inválido: O input é o email e a password do utilizador que pretende fazer login. O resultado esperado é um user nulo, uma vez que o utilizador não existe ou a password é nula/incorreta.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	2	0, >2 ou <2
Tipo de Entrada	Duas strings representado um email e password respetivamente	As duas strings serem inválidas ou nulas
Valor Específico	EC1	EC2

5.2.2 Método *RecoverPassword*

Classes de equivalência

EC1 – Válido: Os inputs são a password, confirmação de password e o token que foi enviado para o e-mail do utilizador que pretende fazer recuperar a pass. O resultado esperado é a alteração da password do utilizador

EC2 – Inválido: Os inputs são a password, confirmação de password e o token que foi enviado para o e-mail do utilizador que pretende fazer recuperar a pass. O resultado esperado é nulo, sendo que as password podem não combinar ou o token não ser o mesmo que o enviado ou inválido.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	3	0, >3 ou <3
Tipo de Entrada	Três strings representado duas password e o token	As strings serem inválidas ou nulas
Valor Específico	EC1	EC2

5.3 ECs para as features do componente *PublicationDAO*

5.3.1 Método *Create*

Classes de equivalência

EC1 – Válido: O input é a publicação com as suas informações. O resultado é a publicação retornado e adicionada à base de dados.

EC2 – Inválido: O input é a publicação com as suas informações. O resultado é um objeto retornado diferente da publicação e não ser adicionada à base de dados.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um objeto de publicaçáo	Um objeto diferente de publicação
Valor Específico	EC1	EC2

5.3.2 Método *FindById*

Classes de equivalência

EC1 – Válido: O input é Id da publicação que se pretende visualizar. O resultado é a publicação que se pretendia

EC2 – Inválido: O input é Id da publicação que se pretende visualizar. O resultado é uma publicação diferente ou nula.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1 ou <1
Tipo de Entrada	Um número inteiro que representa o ID da publicação	Um número inteiro menor que 1 ou nulo
Valor Específico	EC1	EC2

5.3.3 Método *GetPublications*

Classes de equivalência

EC1 – Válido: Os inputs são os filtros para a pesquisa da publicação. O resultado é obter uma ou mais publicações de acordo com os filtros estabelecidos.

EC2 – Inválido: Os inputs são os filtros para a pesquisa da publicação. O resultado é obter uma ou mais publicações diferentes dos filtros que foram estabelecidos.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	4	0, >4
Tipo de Entrada	Uma string contendo o nome do instrumento que se pretende encontrar, uma ou mais categorias, morada ou raio de distância	Mais do que quatro filtros ou nulo
Valor Específico	EC1	EC2

5.3.4 Método *AddPublicationToFavorites*

Classes de equivalência

EC1 – Válido: Os inputs são o ID do utilizador que pretende adicionar e o ID da publicação que pretende adicionar aos favoritos. O resultado é ter a publicação adicionada à lista de favoritos.

EC2 – Inválido: Os inputs são o ID do utilizador que pretende adicionar e o ID da publicação que pretende adicionar aos favoritos. O resultado é a publicação não ser adicionada à lista de favoritos.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	2	0, >2 ou <2
Tipo de Entrada	Um número inteiro que representa um id do utilizador, e um número inteiro que representa o ID da publicação	Um número inteiro maior ou menor que 1 ou nulo
Valor Específico	EC1	EC2

5.4 ECs para as features do componente *ChatDAO*

5.4.1 Método *AddMessage*

Classes de equivalência

EC1 – Válido: O input é um objeto de Message. O resultado é a mensagem aparecer no chat e ser guardada na BD.

EC2 – Inválido: O input é um objeto de Message. O resultado é a mensagem não aparecer no chat e não ser guardada na BD.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	1
Tipo de Entrada	Objeto de Message	Objeto diferente de Message
Valor Específico	EC1	EC2

5.4.2 Método *CreateChat*

Classes de equivalência

EC1 – Válido: O input é um objeto de Chat. O resultado é a criação de um chat.

EC2 – Inválido: O input é um objeto de Chat. O resultado é a não criação de um chat.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	1
Tipo de Entrada	Objeto de Chat	Objeto diferente de Chat
Valor Específico	EC1	EC2

5.4.3 Método *GetMessageList*

Classes de equivalência

EC1 – Válido: O input é o ID do chat que se pretende obter as mensagens. O resultado é a lista de mensagens contidas nesse chat.

EC2 – Inválido: O input é o ID do chat que se pretende obter as mensagens. O resultado é a não listagem das mensagens contidas nesse chat.

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1, <1
Tipo de Entrada	Número inteiro com o ID do chat	Número inteiro maior ou menor que um ou nulo.
Valor Específico	EC1	EC2

5.4.4 Método *GetChats*

Classes de equivalência

EC1 – Válido: O input é o ID do utilizador que pretende obter todos os chats. O resultado é uma lista com todos os chats desse utilizador.

EC2 – Inválido: O input é o ID do utilizador que pretende obter todos os chats. O resultado não é uma lista com os chats

Critério	Classe de equivalência válida	Classe de equivalência inválida
Nº de Inputs	1	0, >1, <1
Tipo de Entrada	Número inteiro com o ID do utilizador	Número inteiro maior ou menor que um ou nulo.
Valor Específico	EC1	EC2