Escola Superior de Tecnologia do Barreiro

Aplicação de Exercício de Desenvolvimento Muscular

CTeSP de Tecnologias e Programação de Sistemas de Informação

Unidade Curricular:

Base de Dados

Docente da Unidade Curricular:

Luís Damas

Alunos:

Diogo Veigas 202200879

Filipe Passarinho 202200885

Índice

Índice de imagens

Sumário

Este relatório descreve a criação de uma base de dados destinada à pesquisa de exercícios de desenvolvimento muscular. O documento apresenta uma análise detalhada do problema, a lógica de funcionamento da aplicação e uma descrição dos dados a serem armazenados. O diagrama de entidades e relações (DER) é abordado minuciosamente, juntamente com a sua conversão para um modelo relacional (MR), acompanhado de exemplos da sintaxe utilizada.

Também é fornecido um resumo conciso, porém descritivo, do procedimento adotado para o desenvolvimento da base de dados, incluindo as suas funcionalidades.

Por fim, o relatório conclui com uma análise do projeto e das competências adquiridas ao longo de seu desenvolvimento.

Introdução

No âmbito da Unidade Curricular de Base de Dados (BD), foi proposta a elaboração de um sistema de dados que permite realizar pesquisas personalizadas e inteligentes, através da construção de uma base de dados apoiada por um conjunto de perguntas e as suas respetivas respostas associadas. O projeto teve uma duração aproximada de três meses.

Os alunos têm como objetivo aplicar as competências adquiridas ao longo do semestre, como a capacidade de conceber um esquema de base de dados utilizando um Modelo de Entidades e Relações/Diagrama de Entidades e Associações (MER/DEA), além do uso de Sistemas de Gestão de Bases de Dados Relacionais (SGBDR). Ambos os estudantes têm o desejo de desenvolver maior autonomia, melhorar a eficiência na gestão do tempo dedicado a cada tarefa e aprimorar as habilidades de trabalho em equipa, fornecendo suporte aos colegas e ao professor.

Este relatório é composto por 11 capítulos, cada um abordando um aspeto importante do tema em questão. O primeiro capítulo contém o resumo, seguido de uma breve introdução que contextualiza o assunto a ser abordado. A descrição lógica oferece uma visão geral do esquema da base de dados e sua estrutura. Em seguida, são apresentadas as necessidades que motivaram a criação da base de dados. O modelo Entidade-Relacionamento (MER) é detalhado, seguido pelo modelo relacional (MR) derivado dele. É feita uma breve descrição do processo de criação da base de dados, inserção de informações e pesquisa dos dados. O capítulo de conclusão apresenta uma síntese das principais conclusões obtidas a partir do estudo. O capítulo de referências lista as fontes consultadas durante a elaboração do relatório e do projeto, enquanto a bibliografia web contextualiza essas referências.

Apresentação do tema

Como tema deste trabalho foi proposta uma aplicação especializada na prática de exercícios físicos focados no desenvolvimento muscular.

Na base de dados serão encontradas informações desde exercícios, boas práticas, planos de treino pré-definidos, bem como dúvidas frequentes que atletas possam ter, tais como a alimentação, tempo descanso entre séries, número de repetições e muito mais.

Descrição lógica

Pode-se analisar a descrição lógica da aplicação a partir de duas perspetivas distintas: a do utilizador e a do desenvolvedor de bases de dados. Além disso, serão apresentados exemplos dos tipos de pesquisa que um utilizador pode realizar, bem como as categorias e *tags* que podem vir a ser encontradas na base de dados.

Perspetiva de um utilizador

Um utilizador, quando acede à plataforma, tem de autenticar-se, ou seja, introduzir as credenciais da sua conta e criar ou aceder a uma sessão já existente.

Numa sessão, um utilizador pode escolher uma pergunta frequentemente perguntada, ou qualquer outro método de pesquisa. Independentemente do método, o mesmo poderá ou não receber respostas relacionadas com o que introduziu. O utilizador poderá também classificar o quão relevante foi qualquer resposta que tenha recebido.

Por fim, pode continuar na mesma sessão e faz mais pesquisas, criar ou aceder a outra sessão, ou até mesmo sair da aplicação, obviamente.

Perspetiva de um desenvolvedor

O presente projeto consiste no desenvolvimento de uma base de dados, para uma aplicação de pesquisa/conversação avançada acerca de exercícios de desenvolvimento muscular. E para que tal pesquisa seja efetuada, existem vários requisitos a satisfazer.

Primeiramente, o sistema deve ser capaz de armazenar perguntas e respostas, classificá-las por categoria/subcategoria, aprovar respostas, atribuir *tags* às respostas e permitir que os utilizadores atribuam uma classificação à qualidade de cada resposta obtida. Além disso, o sistema deve permitir pesquisas por pergunta, categoria/subcategoria, *tags* ou livre.

Existem também outros fatores que podem não parecer tão relevantes, porém, são essenciais para a lógica da aplicação. Sendo uma delas a autenticação de

um utilizador, saber quem é que está a tentar aceder à informação da base de dados. Outra, é a existência de sessões ou *chats*, que o utilizador, depois de autenticado, poderá usar para efetuar um dos quatro tipos de pesquisa acima mencionados.

Para além das necessidades acima mencionadas, é possível concluir que também é necessária a existência de permissões adicionais para que administradores possam autorizar, ou até mesmo, descontinuar respostas. Como também é fundamental desenvolver um método de permitir que as respostas obtidas através de qualquer tipo de pesquisa possam ser avaliadas por qualquer utilizador que as obtenha (as repostas).

Existe também outro problema relacionado às respostas que um utilizador pode receber, como o método de pesquisa usada para guardá-las num histórico de mensagens. Mais detalhes serão dados no próximo capítulo, Descrição das necessidades de informação.

Métodos de pesquisa

Considerando que a aplicação é voltada para o desenvolvimento muscular e fitness, é interessante apresentar exemplos de que tipos de pesquisa o utilizador poderá realizar. Mais tarde, serão também fornecidos exemplos de *tags* e categorias que farão parte da base de dados.

Existem quatro métodos de pesquisa disponíveis nesta aplicação, e este subcapítulo será dividido em quatro partes, uma para cada tipo de pesquisa.

O primeiro método de pesquisa abordado será a pesquisa por pergunta, pois todos os outros métodos de pesquisa acabam por se relacionar de alguma forma com essa modalidade.

Pesquisa por Pergunta

Um utilizador pode escolher uma das perguntas já inseridas na base de dados, isto é, uma pergunta por comumente perguntada.

Exemplo: “Quais é que são os benefícios de fazer flexões?”.

Pesquisa por Categoria

Já na pesquisa por categoria, o utilizador recebe uma ou mais respostas associadas às perguntas têm essa categoria/subcategoria, vale salientar que a pesquisa por categoria, é isso mesmo, apenas uma categoria.

Exemplo: “Exercícios sem equipamentos”.

No exemplo acima, o utilizador quer ver informação, ou melhor, uma lista de exercícios que podem ser realizados sem equipamentos.

Pesquisa por *Tag*

Quando se menciona uma *tag*, existem duas coisas a ter em atenção. Sendo a primeira que apenas as respostas possuem *tags*, e uma resposta pode ter mais do que uma *tag*.

Já o método de pesquisa, consiste em selecionar respostas com a *tag* escrita pelo utilizador.

Exemplo: “Dorsais”.

No exemplo acima, o utilizador quer receber informação relacionada à *tag* “Dorsais”.

Pesquisa Livre

Por fim, a pesquisa livre será, basicamente, para aquele utilizador que não sabem bem o que quer, portanto, o utilizador insere uma cadeia de caracteres e receberá uma resposta que esteja de alguma maneira associada à frase que introduziu. Essa pesquisa livre tentará encontrar uma resposta, comparando o conteúdo introduzido a *tags*, categorias e perguntas existentes na base de dados.

Exemplo: “Quero ter braços maiores, mais grossos”.

No exemplo acima, o utilizador quer receber informações relacionadas ao ganho de massa muscular nos braços.

Exemplos de *tags* e categorias

Este é o tão esperado subcapítulo que tem como intuito dar a conhecer algumas das categorias, subcategorias e *tags* que poderão ser usadas para selecionar e pedir informação à base de dados.

Exemplos de categorias

- Exercícios musculares
    - Levantamento de Peso
        - Halteres
        - Barras
        - Kettlebells
        - Máquinas
    - Exercícios com Peso Corporal
        - Flexões
        - Agachamentos
        - Elevações
        - Abdominais
    - Treinamento com Resistência
        - Bandas Elásticas
- Alimentação
    - Receitas
        - Bulking
        - Cutting
    - Macronutrientes

Exemplos de *tags*

- Upper Body
- Lower Body
- Push
- Pull
- Leg
- 6-pack
- Bíceps
- Tríceps
- Dorsais
- Peitorais
- Lombar
- Romboides
- Gémeo
- Tibial
- Glúteos

- Quadríceps
- Coxa

Descrição das necessidades de informação

A base de dados é fundamental para o bom funcionamento da aplicação, pois é nela que serão armazenadas todas as informações necessárias para o sistema. As perguntas e respostas devem ser registadas de maneira a que se possam relacionar, pois é a partir dessas associações que o utilizador será capaz de receber respostas.

As perguntas armazenadas devem conter o texto da pergunta, o idioma em que foi escrita, a data de entrada no sistema e a data da última atualização realizada. É importante atribuir categorias e subcategorias às perguntas para que a pesquisa seja mais fácil e precisa. Essas categorias podem ser por tipo de exercício, equipamento utilizado, entre outros.

As respostas devem conter o texto da resposta, a data de aprovação e o estado em que se encontra (Aprovada, em Aprovação, Descontinuada). Além disso, é importante adicionar *tags* que descrevam o conteúdo da resposta, como palavras-chave, para que seja mais fácil encontra-las durante a pesquisa.

Os utilizadores devem ter as suas informações pessoais armazenadas no sistema, como nome, e-mail, data de nascimento, género, password e nacionalidade. Essas informações serão utilizadas para autenticação do utilizador no sistema e para personalizar a experiência do utilizador.

As sessões são importantes para acompanhar o uso do sistema pelos utilizadores. Elas devem armazenar um *token* identificador, a data/hora de início e fim da sessão, o IP do dispositivo utilizado pelo utilizador e o tipo de dispositivo (telemóvel, tablet, computador, torradeira, etc.) e, obviamente, o tipo de pesquisa efetuado pelo utilizador. Essas informações serão utilizadas para monitorar o uso do sistema e para melhorar a experiência do utilizador.

Dentro de cada sessão, poderão ser relacionadas pesquisas, que normalmente terão uma data em que a pesquisa foi feita, como as respostas obtidas e o tipo de pesquisa que foi realizado. É também necessário garantir um relacionamento entre as respostas que o utilizador obteve, para além da categoria ou *tags* que ele usou para receber tais respostas, por exemplo.

Existe também mais um fator importantíssimo relacionado às respostas recebidas e parâmetros de pesquisa usados pelo utilizador. Especialmente as

respostas, devem ter uma maneira de ser guardadas, ou seja, persistir de forma completamente independente a seleções feitas na base de dados. Porquê?

Pesquisas feitas na base de dados, como deve saber, são realizadas à informação que está na base de dados, obviamente. Mas pense no seguinte, a informação que está armazenada hoje, pode não estar amanhã. O oposto também é verdade, a informação que lá estará amanhã, provavelmente não estará hoje.

O ponto é, não necessariamente com informação que é eliminada e adicionada, mas sim, mais concretamente, a respostas aprovadas e descontinuadas. Afinal de contas, o estado da resposta é o fator decisivo pelo qual foi necessário desenvolver um sistema de histórico tão específico, que preserva a informação de maneira tão detalhada.

Uma resposta que ainda estava por aprovar hoje, não será apresentada ao utilizador. No entanto, mesmo quando a resposta for aprovada, só deve ser associada às respostas obtidas do utilizador, caso o utilizador volte a realizar essa mesma pesquisa, caso contrário, nem sequer faz sentido incomodar o utilizador com uma pesquisa da qual já não está interessado.

Descrição do Diagrama de Entidades e Relações

Um diagrama de entidades e relações é uma representação visual de um modelo de dados que ilustra as entidades (objetos ou conceitos do mundo real) e as relações entre elas. É utilizado para comunicar a estrutura da informação que será armazenada numa base de dados de forma clara e concisa, mostrando as entidades e atributos que as definem e os relacionamentos entre elas.

abaixo, pode ser encontrado o diagrama proposto para o esquema da base de dados.

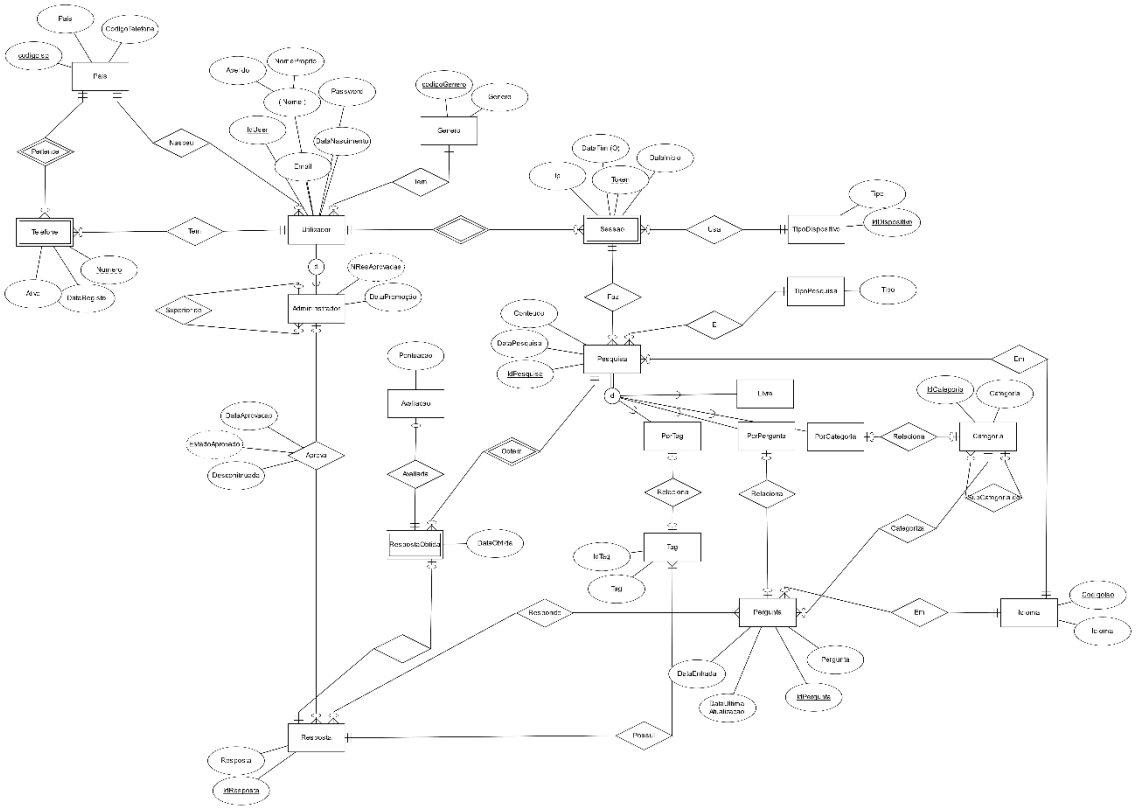

Figura 1 - Diagrama de Entidades e Relações

Entidades

As bases de dados são normalmente constituídas por muitas propriedades sendo a mais importante as entidades. As entidades são distinguíveis de todos os outros objetos sendo estas compostas por várias propriedades que permitem a sua caracterização.

As entidades que constituem esta base de dados são provenientes de duas situações, a primeira situação são as entidades referidas no enunciado do

projeto e a segunda situação são entidades que se revelaram essenciais para um bom funcionamento da base de dados.

Esta base de dados é constituída por entidades fortes, fracas, associativas e supertipo.

Como já deve estar à espera, todas as entidades são constituídas por atributos, pois estas têm de guardar informação relacionada às mesmas. Os atributos de uma entidade podem ser classificados como um atributo simples ou composto, *multivalor*, derivados e nulos.

Para além dos tipos referidos acima, existem também os atributos chave, que servem para distinguir registos entre eles mesmos.

Atributos

Os atributos de uma entidade são fundamentais para guardar informações relacionadas a ela. Existem vários tipos de atributos, incluindo simples ou compostos, *multivalor*, derivados e nulos.

Os atributos simples são aqueles que não possuem uma estrutura interna e podem ser representados por um único valor, como um número ou uma cadeia de caracteres. Já os atributos compostos são aqueles que possuem uma estrutura interna, sendo constituídos por mais de um valor, como por exemplo, um endereço que é formado por rua, número, bairro, cidade e conselho, por exemplo.

Os atributos *multivalor* são aqueles que podem possuir mais de um valor associado a eles, como por exemplo, os telefones de uma pessoa, que podem incluir vários números de telemóvel diferentes. Os atributos derivados são aqueles que podem ser obtidos a partir de outros atributos da mesma entidade, como por exemplo, a idade de uma pessoa, que pode ser calculada a partir da sua data de nascimento.

Por fim, os atributos nulos, também conhecidos como opcionais, são aqueles que não precisam obrigatoriamente ter um valor associado a eles, ou seja, não contêm informações relevantes sobre a entidade em questão. Além disso, há

também os atributos chave, que servem para distinguir registos entre si e são usados para garantir a unicidade de cada registo em uma ou entidade.

Entidades e atributos presentes no diagrama

Abaixo, estará uma lista extensiva de todas as entidades e respetivos atributos presentes no diagrama de entidades e relacionamentos.

Utilizador

Esta entidade é um supertipo, e é constituída por 6 atributos sendo estes, o nome próprio, apelido, *password*, data de nascimento, email e id. Todos os atributos são simples, exceto o nome, que é composto. O id do utilizador é a chave primária.

Telefone

Esta entidade é fraca, e é responsável por guardar apenas um número de telefone, a data de registo do telemóvel e verificar se está ou não ativo, por exemplo, se o utilizador introduziu o código que recebeu por SMS. Sendo esta entidade fraca, a chave parcial é o número, pois podem existir vários números de telefone, mas não com o mesmo código de telefone.

Pais

Esta entidade é constituída por três atributos, sendo estes, o país, o código ISO, e o código do telefone, todos atributos simples, sendo a chave o código ISO.

Género

Esta entidade é constituída por dois atributos sendo estes, o género e o código do género, ambos os atributos são simples, sendo o código a chave.

Sessão

Esta entidade fraca é constituída por quatro atributos sendo estes, a data de início da sessão, a data de fim da sessão, o *ip* do utilizador e o *token*. Esta entidade é fraca pois 2 sessões podem ter o mesmo *token*, e com um relacionamento fraco, será possível garantir que cada sessão é diferente da de cada utilizador.

Tipo de Dispositivo

Esta entidade é composta por dois atributos, o tipo e o id do dispositivo, são ambos atributos simples, sendo o id, a chave.

Pesquisa

Esta entidade é supertipo disjunta, ou seja, cada pesquisa só poderá ter um tipo. Ela é composta por três atributos, a data de pesquisa, o id da pesquisa e o conteúdo, são todos atributos simples, sendo o id a chave. O motivo pelo qual foi necessário criar um id identificador deve-se ao facto de que o utilizador pode pesquisar várias vezes a mesma coisa dentro de uma mesma sessão. Uma outra possível chave primária poderia ser o conteúdo com a data de pesquisa, até porque é difícil serem pesquisadas 2 coisas iguais ao mesmo tempo. No entanto, ao usar a data como tipo de pesquisa, a identificação da informação estaria sujeita à precisão do em que é guardada a data.

Tipo Pesquisa

A entidade do tipo de pesquisa, como o nome indica, refere-se ao tipo de uma pesquisa. Esta serve apenas para identificar, de forma mais rápido e eficaz, o tipo da pesquisa efetuada.

Tem como atributo o tipo que uma pesquisa pode ter, sendo a chave o próprio tipo.

Especializações da Pesquisa

As entidades PorTag, PorPergunta, PorCategoria e Livre são especializações da entidade pesquisa. Servem apenas para se relacionarem com os respetivos dados guardados, mais detalhes no subcapítulo das relações.

Categoria

Esta entidade guarda todos as categorias que uma pergunta poderá ter, sendo ela composta por 2 atributos, a categoria, e o id da mesma. A Categoria é um atributo simples, e o id a chave.

Pergunta

Esta entidade guarda todas as perguntas pela qual o utilizador poderá escolher, e é composta por quatro atributos, sendo estes, a data de entrada, a pergunta, a data da última atualização e o id da pergunta. Todos os atributos são atributos simples, sendo que o id da pergunta é a chave.

Idioma

Esta entidade é composta por dois atributos sendo estes, o idioma e o código ISO, ambos os atributos são simples, com o código, a chave.

Tag

Esta entidade guardará todas as *tags* que uma resposta poderá ter. Ela é composta por dois atributos simples, a *tag* e o id, novamente, o id é a chave.

Resposta obtida

Esta entidade serve para relacionar, ou seja, guardar a/as respostas que condizem com a pesquisa do utilizador. Esta entidade é fraca, pois irá precisar da chave da pesquisa para funcionar de maneira autónoma, mais detalhes sobre a relação mais abaixo, no subcapítulo das relações. E o atributo da data obtida é simples, obviamente.

Avaliacao

Esta entidade serve apenas para guardar a pontuação que um utilizador deu a uma resposta. A pontuação é um atributo simples.

Resposta

Esta entidade armazena todas as respostas da base de dados. Esta entidade é composta por dois atributos, sendo estes a resposta e o id da resposta. Ambos simples, sendo o id a chave primária.

Administrador

Esta entidade é um tipo de utilizador, para além de possuir um atributo derivado, o número de resposta aprovadas, possui também a data de promoção, isto é, quando é que o utilizador se tornou num administrador.

Aprova

Esta entidade é associativa, ou seja, foi criada devido à necessidade de associar um administrador a uma resposta. Ela é composta por 3 atributos, a data de aprovação e uma *flag* de descontinuação, que são simples, e o estado aprovado, que é derivado.

Relações

Numa base de dados as entidades relacionam-se entre si, havendo relações de um para um (1:1), de um para muitos (1:N) e de muitos para muitos (M:N), para além de existir também o tipo de participação, sendo este total ou parcial.

Abaixo, estarão descritas cada uma das relações presentes no diagrama de entidades e relacionamentos.

Relação tem

Um utilizador pode ter vários telemóveis, mas não tem de ter nenhum, já que o email é obrigatório. No entanto, um telemóvel só poderá estar associado a um utilizador, para garantir uma maior segurança aos nossos utilizadores.

Relação pertence

Esta relação consiste na associação de um número de telefone com o seu respetivo país. Será assumido que cada país só terá um código de discagem, dessa maneira, criando uma relação de 1 para muitos. A participação é obrigatória. Sendo também um relacionamento fraco para o lado do telefone, pois podem existir números de telefone iguais com código de discagem diferentes.

Relação nasceu

Esta relação entre o utilizador e o país criou a necessidade de existir uma nova entidade, pois cada utilizador é de um e apenas um país, enquanto que o mesmo país pode corresponder a vários utilizadores, como também podem não existir utilizadores nascidos em Butão, levando a que seja uma relação com participação parcial, com obrigatoriedade do utilizador.

Relação tem

Esta relação entre o utilizador e o género criou a necessidade de existir uma nova entidade, pois cada utilizador só pode ter um género, enquanto que o mesmo género pode corresponder a vários utilizadores, levando a que seja uma relação com uma participação parcial. E sim, um utilizador tem de ter um género, nem que o mesmo se designe como indefinido ou similar.

Supertipo utilizador

O utilizador é supertipo, ou seja, pode possuir especializações. Neste caso, a única especialização existente é a administrador, que permite que um utilizador aprove respostas. Este supertipo não é total, isto é, não tem de necessariamente

ser administrador, pois, qualquer utilizador, com privilégios ou não, poderá pesquisar.

Relação superior de

Para uma maior autonomia do cliente, existirão dois tipos de administradores, aqueles que o programador insere na base de dados, e aqueles que os administradores chefes adicionam como empregados. E para auxiliar essa funcionalidade, os administradores possuirão a data de promoção, como referido anteriormente, para além de estarem relacionados aos seus chefes hierárquicos.

Relação aprova

Esta relação serve apenas para aprovar respostas, ou seja, a partir do momento que em um registo é criado com a associação de uma resposta e um administrador, será possível determinar que essa resposta foi aprovada, e que o administrador aprovou mais uma resposta.

Relação possui

Esta relação é entre a entidade *tag* e a entidade resposta, nesta relação uma resposta pode ter várias *tags* e uma *tag* pode pertencer a várias respostas. Existe uma participação parcial nesta relação pois uma *tag* pode estar em várias respostas, não sendo obrigatório estar nalguma, no entanto, uma resposta tem de ter pelo menos uma *tag*.

Relação responde

As perguntas podem ter várias respostas associadas às mesmas, como até pode ainda não existir numa resposta aprovada. E o oposto também é quase verdade, uma resposta pode responder a várias perguntas, no entanto, uma resposta só existirá para responder pelo menos a uma pergunta. Levando a uma participação parcial.

Relação categoriza

A necessidade de relacionar uma pergunta a uma categoria exigiu a criação de uma nova entidade, pois cada pergunta deve corresponder a uma única categoria. No entanto, a mesma categoria pode ser atribuída a várias perguntas, o que resulta numa relação de participação parcial entre as entidades.

Relação escrita em

Todas as perguntas terão um idioma, o que levou à criação de uma nova entidade, onde cada pergunta só pode ter um idioma, mas o mesmo idioma pode ser usado em várias perguntas.

Relação subcategoria

Todas as categorias, não importa qual, podem ter descendentes, portanto serão associadas categorias como subcategorias entre elas, por exemplo, halteres pertencem ao levantamento de peso que por sua vez pertencem aos exercícios musculares.

Relação utilizador e a sessão

Esta relação é entre a entidade utilizador e a entidade sessão, nesta relação o utilizador poder ter várias sessões ou nem ter nenhuma, mas uma sessão só pertence a um utilizador, foi por isso que foi decidido criar uma relação fraca entre estas duas entidades. Portanto, a chave da sessão acabará por ser composta pela chave da entidade sessão e da entidade utilizador.

Esta relação tem uma participação parcial pois não é obrigatório um utilizador ter sequer uma sessão, pois quando o mesmo se regista na aplicação, não tem de necessariamente usá-la.

Relação usa

Uma relação entre a sessão e o tipo de dispositivo levou à criação de uma nova entidade, pois uma sessão precisa de ser aberta num dispositivo, no entanto, novos dispositivos podem ser adicionados a uma mesma sessão.

Esta relação tem uma participação parcial pois não é preciso um dispositivo ter uma sessão aberta, mas para o programa funcionar é obrigatório ter uma sessão aberta em pelo menos um dispositivo.

Relação faz

Na relação entre a entidade sessão e pesquisa, é possível realizar várias pesquisas dentro de uma mesma sessão. No entanto, é importante destacar que cada pesquisa realizada é única, já que é necessário identificar o utilizador que a executou, mesmo que o tipo, conteúdo e resposta da pesquisa sejam idênticos a outras pesquisas. Nessa relação, a entidade sessão tem uma participação parcial, já que não é obrigatório que exista uma pesquisa associada a cada sessão.

Relação É (tipo pesquisa)

Esta relação serve apenas para atribuir um tipo à pesquisa. Vale referir que uma pesquisa tem de obrigatoriamente possuir um tipo, nem que ele seja “livre”. No entanto, um tipo de pesquisa pode ter sido feito em diversas pesquisas, ou ainda em nenhuma, o que leva a uma participação parcial.

Supertipo pesquisa

A entidade pesquisa é um supertipo, sendo ele, disjunto e total, ou seja, deve existir sempre um tipo de pesquisa a cada pesquisa feita, para além de que cada pesquisa só terá um único tipo.

Especializações da pesquisa

Como uma pesquisa pode ter vários tipos, e cada tipo tem as suas particularidades, por exemplo, uma pesquisa por categoria só recebe uma categoria, não deve receber uma *tag* nem uma pergunta, logo, só poderá ser guardada informação proveniente da entidade das categorias.

É também relevante mencionar o facto de que uma pergunta da lista de perguntas não tem de ter necessariamente sido pesquisada, tal como uma

pesquisa pode não corresponder a nenhuma das perguntas guardadas na base de dados.

Relação obtém

Esta relação é fraca, para o lado da resposta obtida, pois é a única maneira de saber que respostas um utilizador obteve em determinada pesquisa. Um utilizador pode receber várias respostas numa mesma pesquisa, no entanto, uma resposta só será obtida, ou seja, selecionada, quando um utilizador procura por ela.

Relacionar uma pesquisa a uma resposta

Esta relação permite que uma reposta seja associada a uma pesquisa de qualquer um dos tipos e guardada para não existirem discrepâncias de informação ao passar do tempo. Sendo que todas as respostas obtidas têm de se relacionar a pelo menos uma pesquisa, enquanto que uma resposta obtida tem uma e só uma resposta e uma resposta na lista de respostas nem sequer tem de ter sido usada para responder a uma pesquisa.

Relação avaliada

Esta relação serve para atribuir uma nota (1 a 10) a quão bem a resposta se encaixa ao que o utilizador pesquisou.

Relações requisito

Neste subcapítulo serão também realçadas as relações pedidas no enunciado como obrigatórias.

Recursiva

Uma relação recursiva usa chaves de outros registos da mesma entidade para criar uma hierarquia. As relações recursivas presentes no esquema são a

Relação superior de e a Relação categoriza, das entidades administrador e categoria, respetivamente.

Ternária

Relações ternárias são entidades que possuem pelos menos 3 relações. No diagrama de entidades e relações pode encontrar a entidade Pergunta.

Generalização e especialização

Uma generalização ou entidade supertipo é uma entidade que pode ter vários subtipos, os quais podem também ser supertipos eles mesmos. A especificação de um subtipo pode ou não se sobrepor a outros subtipos da entidade supertipo. Nesta representação, a entidade Pesquisa e Utilizador são generalizações.

Relacionamento fraco

Um relacionamento fraco é criado quando a chave da própria entidade não é suficiente para identificar uma tupla da entidade. O que também faz com que a chave da entidade fraca de um relacionamento fraco deixe de ser a chave total, mas sim uma chave parcial.

E no diagrama de entidades e associações, pode ser encontrada a Relação , sendo a entidade fraca a sessão, tal como a **Erro! A origem da referência não foi encontrada.**, cuja entidade fraca é o número de telefone.

O que é um Modelo Relacional

O modelo relacional é uma forma de representação de uma base de dados através de tabelas, onde cada tabela representa uma entidade, os seus atributos, e os relacionamentos entre as entidades são representados por meio de chaves estrangeiras. Esse modelo tem como principal vantagem a sua simplicidade e facilidade de uso, sendo amplamente utilizado em sistemas de gestão de bases de dados relacionais (SGBDR).

O processo de conversão de um Modelo de Entidade e Relacionamento (MER) para um Modelo Relacional (MR) pode ser vantajoso, uma vez que o MR é mais simples e fácil de implementar num SGBDR. O MER é útil para representar a estrutura e as relações das entidades de uma base de dados de forma mais visual e conceitual, mas pode ser mais difícil de implementar diretamente num SGBDR. Ao converter o MER em MR, a base de dados é organizada em tabelas, o que facilita a conversão mais direta para um SGBDR, como o *MySQL*, por exemplo.

Modelo Relacional da Base de Dados

Neste subcapítulo é apresentado o modelo relacional elaborado pelos alunos do grupo, acompanhado de uma análise detalhada da sintaxe utilizada e da escolha das chaves primárias, chaves estrangeiras e tabelas adicionais.

A explicação da sintaxe usada no modelo é de grande importância, pois permite uma compreensão mais precisa do que foi planeado pelo grupo.

Além disso, o subcapítulo aborda o porquê da escolha das chaves primárias e estrangeiras, que são essenciais para garantir a integridade dos dados e estabelecer relacionamentos entre as tabelas.

Por fim, também é abordada a necessidade de criar tabelas adicionais para armazenar informações relacionadas que não se encaixam perfeitamente em nenhuma tabela existente.

Sintaxe usada para apresentar o modelo

Com o objetivo de facilitar a compreensão do modelo projetado, serão descritos os tipos de formatação utilizados para identificar cada componente de cada tabela.

Primeiramente, será realizada uma quebra de página antes e depois do modelo relacional. A fonte usada também será diferente para representar o MR, sendo ela a *Roboto Mono*, pois cada caractere tem o mesmo espaço horizontal, o que facilita a leitura de textos que precisam de alinhamento vertical, como código-fonte.

Os nomes das tabelas encontram-se em *UpperCamelCase* antes de um par de parênteses. Enquanto que os atributos referentes às tabelas encontram-se diretamente após ao respetivo nome entre parênteses, separado por vírgulas.

Surgiu também a necessidade de destacar as chaves, como a chave primária, candidata e estrangeira. A chave primária, será estilizada com negrito e sublinhado, enquanto que a chave estrangeira estará diretamente após as letras *CK* e a chave estrangeira após as letras *FK*.

A chave estrangeira possui também outra característica única, a sua referência. Uma chave estrangeira pode referir-se à chave primária da própria, ou de qualquer outra tabela, portanto, tal como em *MySQL*, é referido o nome do atributo na presente tabela, e o nome e atributo da tabela ao qual se está a referir.

Tão esperado modelo

Finalmente, abaixo encontra-se o modelo relacional desenvolvido pelo grupo.

pais(**codigo_iso**, código_telefone, pais)

CK:pais

genero(**codigo_genero**, genero)

CK: genero

user(**id_user**, email, data_nascimento, password, nome_proprio, apelido, codigo_iso, codigo_genero)

CK: email

FK: pais referencia pais(codigo_iso)

genero referencia genero(codigo_genero)

telefone(**numero**, **codigo_iso**, data_registo, ativo, id_user)

FK: código_iso referencia pais(codigo_iso)

id_user referencia user(id_user)

admin(**id_user**, data_promocao)

FK: id_user referencia user(user)

chefia(**id_user**, **id_chefe**)

FK: id_user referencia user(id_user)

id_chefe referencia user(id_user)

resposta(**id_resposta**, resposta)

CK: resposta

aprova(**id_user, id_resposta**, data_aprovacao, descontinuada)

FK: id_user referencia admin(id_user)

id_resposta referencia resposta(id_resposta)

tag(**id_tag**, tag)

CK: tag

reposta_possui_tag(**id_resposta**, **id_tag**)

FK: id_resposta referencia resposta(id_resposta)

id_tag referencia tag(id_tag)

categoria(**id_categoria**, categoria)

CK: categoria

sub_cat_cat(**sub_categoria**, **categoria**)

FK: sub_categoria referencia categoria(id_categoria)

categoria referencia categoria(id_categoria)

idioma(**codigo_idioma**, idioma)

CK: idioma

pergunta(**id_pergunta**, pergunta, data_entrada, ultima_atualizacao, idioma, categoria)

CK: pergunta

FK: idioma referencia idioma(codigo_iso)

categoria referencia categoria(id_categoria)

resposta_responde_pergunta(**id_resposta**, **id_pergunta**)

FK: id_resposta referencia resposta(id_resposta)

id_pergunta referencia pergunta(id_pergunta)

tipo_dispositivo(**id_dispositivo**, tipo)

CK: tipo

sessao(**token**, **id_user**, ip, inicio, fim, dipositivo)

FK: id_user referencia user(id_user)

dipositivo referencia tipo_dispositivo(id_dispositivo)

tipo_pesquisa(**id_tipo**, tipo)

CK: tipo

pesquisa(**id_pesquisa**, token, id_user, data_pesquisa, conteudo, idioma, id_tipo)

FK: token referencia sessao(token)

id_user referencia sessao(id_user)

idioma referencia idioma(codigo_iso)

id_tipo referencia tipo_pesquisa(id_tipo)

tag_pesquisada(**id_pesquisa**, **id_tag**)

FK: id_pesquisa referencia Pesquisa(id_pesquisa)

tag referencia tag(id_tag)

categoria_pesquisada(**id_pesquisa**, **id_categoria**)

FK: pesquisa referencia pesquisa(pesquisa)

id_categoria referencia categoria(id_categoria)

pergunta_pesquisada(**id_pesquisa**, **id_pergunta**)

FK: id_pesquisa referencia pesquisa(id_pesquisa)

id_pergunta referencia pergunta(id_pergunta)

resposta_obtida(**id_pesquisa**, **id_resposta**, data_obtida)

FK: id_pesquisa referencia pesquisa(id_pesquisa)

id_resposta referencia resposta(id_resposta)

avaliacao(**id_pesquisa**, **id_resposta**, pontuacao)

FK: id_pesquisa referencia pesquisa(id_pesquisa)

id_resposta referencia resposta(id_resposta)

Adição de tabelas e chaves estrangeiras

Na esquematização de uma base de dados, é considerada uma boa prática evitar relações de 1:1 com participação total, ou seja, obrigatoriedade por parte de ambas as entidades. Em vez disso, é preferível juntar essas duas tabelas e manter apenas a chave primária mais relevante. No entanto, no DER usado para clarificar o esquema da base de dados deste projeto, foram incluídas diversas relações de 1:1 obrigatórias para facilitar a compreensão da lógica por detrás da organização da informação. Por isso, os alunos acharam importante deixar claro esse aspeto.

Uma outra boa prática consiste em evitar que existam campos nulos, para isso, são criadas novas tabelas com certas propriedades. Quando existem duas tabelas relacionadas e uma delas não é obrigatória ou pode existir mais de uma instância, a chave estrangeira é atribuída a essa tabela. Em casos em que não há obrigatoriedade em nenhuma das tabelas ou a relação é de muitos para muitos, uma nova tabela é criada para gerir essa relação.

É importante lembrar que uma chave primária não costuma ser constituída por apenas uma chave estrangeira pois pode não ser o suficiente para identificar um registo, no entanto, existem casos como na generalização, que pode fazer sentido usar uma chave primária composta apenas pela chave estrangeira.

Numa relação de muitos para muitos, é comum que as duas chaves estrangeiras sejam combinadas para formar a chave primária dessa tabela. Além disso, em tabelas com relação fraca e forte, a parte fraca recebe a chave estrangeira e a chave parcial da própria tabela para formar a chave primária total. Essas práticas contribuem para a organização e consistência da informação armazenada na base de dados de dados.

Base de dados MySQL

Neste último grande capítulo, será abordado e descrito, de maneira breve, todo o desenvolvimento de uma base de dados de suporte a uma aplicação de Perguntas Frequentes cuja temática segue a apresentação realizada na 1ª fase.

Neste capítulo encontrará informação sobre a criação do esquema, ou seja, das tabelas da base de dados, a inserção de informação, consultas feitas à base de dados, pesquisas com o auxílio de *views*, a componente lógica constituída por funções e procedimentos, bem como o uso da base de dados e testes feitos às restantes componentes.

Criação da base de dados

Inicialmente, antes de qualquer manipulação de informação, obviamente, será necessário criar todo o esquema da base de dados, tal como, as tabelas e as suas respetivas colunas.

Tipos de tabelas

As tabelas criadas na base de dados podem ser divididas nalguns tipos diferentes, tais como, as tabelas simples, que estão encarregadas de guardar informação, tal como a pergunta, resposta, utilizador, etc. Como também existem as tabelas que se encarregam de associar tabelas ou pouco mais, tais como respostas que respondem a perguntas, ou respostas obtidas de uma pesquisa, por exemplo.

Hierarquia de informação

Existem duas tabelas que servem como hierarquia de informação, sendo elas *sub_cat_cat*, que guarda que categoria é subcategoria de que categoria, tal como a tabela *chefia*, que guarda a informação de administradores que chefiam e são chefiados.

Informação relacionada a uma pesquisa

Pesquisa, é uma tabela que possui referências através de chaves estrangeiras em diversas tabelas, pois ela é uma das tabelas mais importantes de todo o esquema.

E o que foi referido anteriormente deve-se ao facto de que a maneira mais fácil de associar as respostas obtidas numa pesquisa, as perguntas/categorias/*tags* que foram encontradas com o conteúdo introduzido pelo utilizador, como também a classificação que um utilizador atribuiu a uma resposta, tendo em conta a sua pesquisa com a ajuda da chave estrangeira proveniente da tabela das pesquisas, que por usa vez, passa por outras tabelas até essa chave deixar de ser necessária

Inserção de dados

Tal como o enunciado pedia, a todas as tabelas ao qual faz sentido, tiveram de ser inseridos pelo menos 20 registos, de modo a que a base de dados e as suas pesquisas sejam mais realistas e mais corretas. O que também levou à necessidade de introduzir mais registos nalgumas tabelas, devido às dependências que outras têm à mesma.

Existem também alguns factos que podem ser referidos neste subcapítulo, tal como a gestão de data e hora numa inserção. Para além de cadeias de caracteres e inteiros, também foi necessário o uso de *timestamp* e até mesmo *date_sub*, para de forma mais correta e/ou automática se obtivessem as datas para as respetivas colunas das tabelas.

Consultas à base de dados

Sendo esta a parte que demandou mais trabalho, e no qual foi investido mais tempo, logo a seguir das inserções, obviamente, foi feito o documento de consultas ou *selects* à informação presente na base de dados.

Em todas as tabelas, foi tentado ao máximo fornecer informação o mais detalhada possível, tal como usar *joins* a diversas tabelas, de modo a obter a informação em estado legível para que as seleções possam ser mais úteis. Um

exemplo são as consultas de perguntas, às quais foram relacionadas o texto das respetivas categorias pertencentes a cada categoria.

Duas coisas forma feitas para aumentar a qualidade das consultas, usar *aliases*, isto é, atribuir um nome mais fácil de ler e escrever para qualquer pessoa que faça essa consulta, bem como, o uso de *natural joins*, o que evita a necessidade de identificar a condição de junção de tabelas. E isto foi apenas possível, obviamente, após ter sido desenhada uma base de dados de forma correta.

Com a exceção do básico e comum, pode também ser mencionado o uso de uma função criada pelo grupo para calcular a idade de um utilizador de maneira mais rápida.

Pesquisas à base de dados

Neste documento, podem ser encontradas as *views* da base dados, isto é, tabelas imaginárias que guardam apenas informação dinâmica de uma ou mais tabelas.

O grupo concordou que as *views* mais importantes seriam aquelas responsáveis pelo tipo de pesquisa, sendo elas, pesquisa por *tag*, categoria, pergunta e livre.

Todas as tabelas recebem as respostas associadas a exatamente pesquisaram, com a exceção da pesquisa livre. Vale também apontar que todas as respostas devem estar associadas a uma pergunta, por isto, mesmo que uma *tag* esteja associada a uma resposta, a resposta deve responder a uma pergunta, obviamente. Bem como, mais importante ainda, essa resposta deve ter sido aprovada por um administrador, antes mesmo de ser pesquisada.

A pesquisa livre, ao invés de procurar por estritamente uma igualdade, deve procurar por uma palavra dentro de uma categoria, *tag*, pergunta, ou até mesmo uma resposta.

Uma descrição mais detalhada sobre o uso de cada uma das *views* pode ser encontrada no último ficheiro, contendo todas as instruções de teste.

Componente lógica

Esta componente encontra-se no documento chamado *logic.js*, e este contém a criação de todos os *stored procedures* e *functions*.

De acordo com os requisitos mínimos, foram desenvolvidos 2 procedimentos, um responsável por criar uma sessão, gerar um *token*, e devolver esse mesmo *token* que fora gerado, e o outro procedimento responsável por receber o id de uma sessão, cloná-la, e adicionar atrás do *token* o texto “TEMP “, como também, por fim, retornar esse mesmo *token* com o “TEMP “.

Para além das *stored procedures*, neste mesmo documento, estão presentes duas funções. Uma das funções considerada um requisito mínimo, e outra que foi implementada devido às vantagens que a mesma traz.

A primeira função consiste na pesquisa de uma palavra em todas as respostas, e retorna o número de respostas que possuem tal palavra. Como deve calcular, foi usado o *count(*)* com recurso a uma *subquery* com uma comparação *like* da concatenação da palavra recebida por parâmetro com os símbolos “%” à volta da palavra.

Já a segunda função, calcular uma idade a partir da data atual do sistema. Ainda que pareça uma função demasiado simples e inútil, ela é, na verdade, bastante útil. Até porque, se parar para pensar, sempre que é necessário agrupar informação ou simplesmente calcular uma idade é necessário escrever uma grande fórmula, e desta maneira, o processo de cálculo de idades é bastante simplificado.

Uso e teste de funcionalidades

Neste último ficheiro de funcionalidades, é onde é verificado se as funcionalidades implementadas realmente funcionam como deveriam.

Este documento é curto, no entanto, é uma das partes mais importantes para um cliente hipotético, cobrindo o meio de criar perguntas, respostas, *tags* e categorias, associar a informação de maneira correta, isto é, associar respostas a perguntas, categorias a perguntas e *tags* a respostas. É também salientado o método de aprovação de uma dada resposta, até porque, caso uma resposta

não tenha sido aprovada por um administrador, será impossível receber tal resposta a partir de uma pesquisa.

Neste documento são também exemplificadas condições que um cliente deve utilizar para efetuar uma pesquisa de um determinado tipo através das *views* anteriormente criadas, bem como, o uso das funções e procedimentos.

E por fim, mais abaixo, será exemplificado como realizar uma pesquisa, escolher o tipo, receber respostas, guardá-las, possivelmente, avaliá-las, e ainda apresentar, de forma clara e concisa, todos os dados mais relevantes de uma pesquisa.

Inserção de perguntas e respostas

Na primeira parte do documento, serão introduzidos alguns dados para fins de testes completamente dinâmicos, isto é, não importa quantas vezes sejam executados, nunca retornarão erros. Para tal fim, foi utilizado o *ignore* do comando *insert*, como também foram utilizadas as variáveis presentes no *MySQL* para trabalhar dinamicamente com as chaves dos registos.

Os alunos determinaram importante exemplificar, pelo menos, todas as operações básicas que um possível cliente pudesse querer realizar a partir da base de dados.

Pesquisas com *views*

Para efetuar uma pesquisa, uma coisa que o uso de todos os *views* têm em comum é a seleção de toda a informação presente na *view*. Ainda que não seja necessário selecionar toda a informação, é mais rápido e é o que faz mais sentido.

Após selecionar a informação da *view*, basta adicionar uma clausula *where* com a respetiva condição a pesquisar.

Sempre que pretender realizar uma pesquisa por *tag*, basta verificar se a *tag* é igual “=” ou como “like” a *tag* exata que deseja pesquisar. O mesmo acontece com a pesquisa por categoria e pergunta, sendo que deve substituir *tag* por pergunta ou categoria.

No entanto, tal como o procedimento da pesquisa livre é mais complexo, a cláusula de pesquisa também pode ser mais complexa. Já que a pesquisa livre pode querer obter informação em diversos tipos de dados, é recomendado o uso da função de concatenação do MySQL para concatenar todos os textos de cada um dos registos da *view*.

Desta maneira, através da função *concat*, pode não só decidir os campos aos quais quer pesquisar, como após isso, deve efetuar uma comparação “like” a toda essa concatenação, não esquecendo as percentagens, para que a palavra ou palavras sejam pesquisadas sobre todos os campos pretendidos de um registo.

Criação de registos com procedimentos

Atualmente, existem 3 procedimentos, sendo um responsável por criar uma nova sessão, outro, responsável por clonar uma sessão, e ainda outro, que estará responsável por terminar uma sessão.

Criar uma sessão

Quando quiser usar qualquer um dos dois procedimentos relacionados às sessões, terá de usar um recurso especial do MySQL que consiste no uso de variáveis, que podem ser usadas com o nome seguido de uma arroba.

EX: @token

E isto acontece pois, sempre que é criada uma nova sessão, a mesma recebe todos os dados, tal como ip, datas de início e fim, e id do utilizador, como também retornará o novo *token* gerado, até porque o mesmo foi criado através da função “UUID()” do MySQL para criar, de forma segura e automática, um *token*.

Após isso, pode tanto usar a variável e selecioná-la para ver o *token* aleatoriamente gerado, como pode também usá-lo como uma cláusula de condição, como o *where*, para, por exemplo, selecionar todas as sessões com esse *token*.

Como referido anteriormente, este procedimento de criação de sessão deve receber todos os dados que uma sessão normalmente receberia, com a exceção

do *token* e id do utilizador. No entanto, a inserção de datas só faz sentido no caso de que a sessão deveria ter sido criada e/ou terminada no passado, o que normalmente não será o caso.

Tendo em mente a maneria com que a base de dados será usada na aplicação, é também possível enviar ambos os *timestamps* vazios, isto é, nulos, para desta maneria o procedimento usar a data e hora atual do sistema para guarda-la como início de sessão, e o fim da sessão será mantido a nulo até que a mesma termine, ou seja, seja guardado uma nova data e hora.

Clonar uma sessão

Para clonar uma sessão, isto é, criar uma nova sessão com certas alterações, pode ser usado o segundo procedimento da base de dados.

O segundo procedimento, ao contrário do primeiro, tem apenas de receber o id de uma sessão já existente, para dessa maneira copiar apenas o conteúdo da outra sessão para esta.

Vale referir que o início da sessão clonada será sempre o instante atual do sistema, e o fim será marcado como nulo, até porque não faz sentido clonar uma sessão e mantê-la fechada.

O *token* da sessão clonada terá um prefixo “TEMP “, como também retornará esse *token* clonado.

Terminar uma sessão

Para terminar uma sessão, isto é, atribuir-lhe uma data de finalização, foi criado este último procedimento.

O mesmo recebe apenas o id da sessão que deseja terminar, como também retorna o momento em que foi terminada, cujo valor pode ser guardado tal como os *tokens* referidos anteriormente.

Realizar uma pesquisa através de um conteúdo

Ainda mais abaixo no documento de testes, logo após o teste de funções e procedimentos, estão presentes, alguns meios de pesquisa dinâmicos, com um tipo de pesquisa livre, que guardará como respostas obtidas toda a informação relevante para o conteúdo de pesquisa introduzido por um possível utilizador.

Todo o processo consiste na criação de uma nova pesquisa numa determinada sessão, bem como, a decisão de um conteúdo para pesquisa. Após isso, será necessário guardar todas as respostas obtidas nessa mesma pesquisa, com a ajuda da respetiva *view*, que para uma pesquisa do tipo livre, deve ser a pesquisa completa. Pode também ser adicionada uma avaliação com uma pontuação de 0 a 10, tal como poderá também verificar a integridade da informação através da seleção de informação realizada no final do documento.

Ordem de execução dos scripts

Por fim, é também importante ressaltar que os scripts devem ser executados na ordem correta, pois existem ficheiros que dependem de outros. Para garantir o funcionamento adequado, os ficheiros devem ser executados na seguinte sequência:

- *create.sql*
- *search.sql*
- *logic.sql*
- *populate.sql*
- *queries.sql*
- *test.sql*

A ordem acima não é uma regra absoluta, ou seja, a ordem dos scripts pode ser alterada entre si sem problemas, desde que alguns detalhes sejam observados. Antes de executar qualquer operação na base de dados, é necessário criar a base de dados, criar as tabelas correspondentes e, só então, realizar as operações desejadas.

Outra observação importante é que tanto no documento *test.sql* quanto o *queries.sql* dependem da execução do *logic.sql*, pois eles utilizam funções implementadas apenas no script de lógica.

O ficheiro de testes, como o próprio nome sugere, destina-se a testar as funcionalidades implementadas. No entanto, é necessário ter em mente que todas as funcionalidades só poderão ser testadas quando existir informação na base de dados (*populate.sql*) e quando as *views* de pesquisa (*views.sql*) forem criadas.

Conclusão

Com este trabalho, o grupo alcançou uma solução funcional para o problema proposto, reconhecendo a existência de outras soluções possíveis e potencialmente melhores. Todos os requisitos mínimos foram cumpridos e a explicação do projeto está completa e de fácil compreensão. A base de dados foi enriquecida com a adição de complexidade, o que representa um ponto positivo no desenvolvimento do projeto.

Os alunos estão satisfeitos com os progressos realizados nesta primeira fase do projeto, tendo aprendido e aplicado o conteúdo lecionado nas aulas. A gestão do tempo foi eficaz, priorizando o que era mais importante para obter uma boa avaliação.

Durante a realização do projeto, foram enfrentadas algumas dificuldades de impacto limitado. A primeira dificuldade foi a implementação de todas as relações requisitadas no enunciado do projeto, pois algumas delas não haviam sido lecionadas em sala de aula quando o DER começou a ser esquematizado. No entanto, essa dificuldade foi superada ao longo do tempo, especialmente durante o processo de criação da base de dados com o auxílio do SGBDR MySQL.

Outra dificuldade relevante foi a funcionalidade de armazenar as respostas e o tipo de pesquisa do utilizador de forma precisa ao longo do tempo. Para resolver esse problema, foi desenvolvido um sistema de histórico. Durante a criação das tabelas, foi implementado um método que evita a perda de informações relevantes, como uma tabela para armazenar as respostas encontradas e o método de pesquisa utilizado.

Como trabalho futuro, sugere-se a implementação desta base de dados MySQL em uma aplicação prática. Além disso, espera-se aprimorar a base de dados para que possa ser adicionada ao portfólio dos alunos como um projeto bem-sucedido. Em suma, o grupo acredita que este projeto foi uma oportunidade de crescimento em termos de conhecimento adquirido, experiência de trabalho em equipe e capacidade de enfrentar desafios e encontrar soluções.

Web grafia

Estes dois sites foram usados como maneira de interpretar e perceber melhor as relações que existem entre as entidades no MER:

- (Joel, 2023)
- (Oliveira, 2023)

As 3 páginas web abaixo serviram para resolver diversos problemas durante a implementação da base de dados em MySQL.

- (Oracle, 2023)
- (Data, 2023)
- (Atwood, 2023)

O (OpenAI, 2023) foi utilizado ao longo de todo o projeto, proporcionando aos alunos assistência durante a resolução de problemas relacionados às diversas formas de associações presentes entre as entidades, bem como na implementação das funcionalidades da base de dados. Além disso, o OpenAI auxiliou também os alunos a organizar e escolher a informação mais importante a ser apresentada no presente relatório.

Referências

Atwood, J. (06 de 01 de 2023). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. Obtido de Stack Overflow: https://stackoverflow.com/

Data, R. (10 de 06 de 2023). *MySQL Tutorial*. Obtido de W3Schools: https://www.w3schools.com/mysql/default.asp

Joel. (12 de 04 de 2023). *MER e DER: Modelagem de Bancos de Dados*. Obtido de DevMedia: https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332

Oliveira, D. (12 de 04 de 2023). *MER e DER: Definições, Banco de Dados e Exemplos | Alura*. Obtido de Alura: https://www.alura.com.br/artigos/mer-e-der-funcoes

OpenAI. (07 de 04 de 2023). *Introducing ChatGPT*. Obtido de OpenAi: https://openai.com/blog/chatgpt

Oracle. (01 de 06 de 2023). *MySQL*. Obtido de MySQL: https://www.mysql.com/