

**Bases de Dados
2021/2022**

Licenciatura em Engenharia Informática

Enunciado de Projeto – Gestão Integrada de Atividades Desportivas (2ª Fase)

Leia o enunciado com atenção!

Quaisquer dúvidas sobre o mesmo deverão ser colocadas no respetivo fórum no Moodle.

1 Introdução

A 2ª fase do projeto incidirá sobre as implicações da lógica aplicacional na implementação do modelo, gestão de dados e sua integridade (Figura 1).

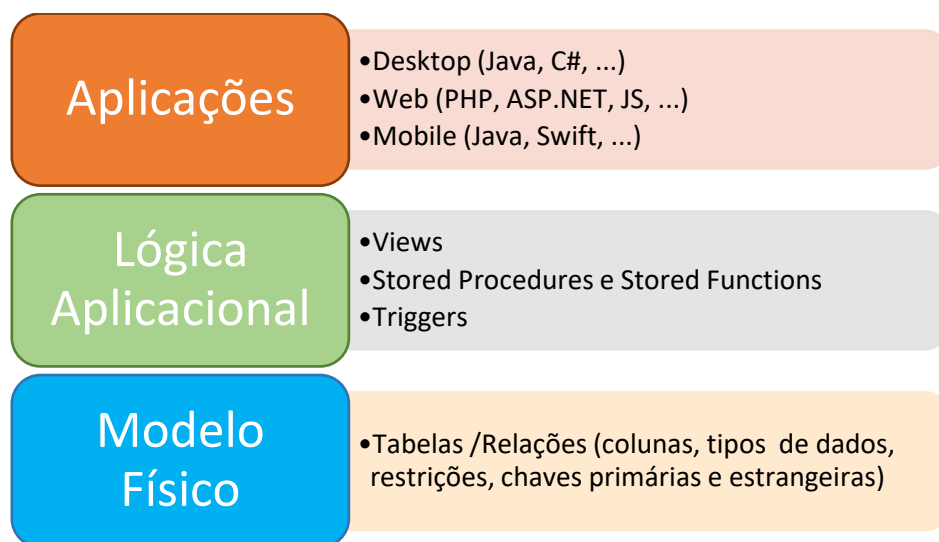


Figura 1 - Arquitetura do problema. A camada aplicacional será alvo de estudo noutras unidades curriculares.

O objetivo será desenvolver um conjunto de procedimentos e mecanismos que, para além de permitirem manter a integridade referencial da base de dados, permitam que uma aplicação desempenhe as funcionalidades básicas pressupostas pelo sistema preconizado com segurança, através de uma API ao nível dos dados.

Por exemplo, para criar um novo utilizador na aplicação, em vez de invocar o comando INSERT INTO (DML) (mais difícil de programar, risco de *SQL-Injection*, etc.), poderá invocar um procedimento disponível para o efeito, (exemplo: `sp_insert_user(...)`) evitando os riscos indicados anteriormente.

O mesmo princípio aplica-se a consultas disponibilizadas através de *Views*.

Dada a multiplicidade de variantes desportivas escolhidas pelos grupos, os requisitos a desenvolver são aqui especificados de uma forma genérica e deverão ser adaptados, por parte do grupo de trabalho, de acordo com o problema específico que pretendem resolver. Da mesma forma, um problema específico poderá levantar a necessidade de implementação de um ou mais procedimentos complementares de modo a completar a lógica aplicacional desenvolvida nesta fase.

1.1 Objetivos específicos

- Refinação do modelo de dados no seguimento da discussão do trabalho da 1ª fase;
- Criação da Base de dados e carregamento inicial de dados;
- Criação de Consultas (*Queries*), *Views*, *Functions*, *Stored Procedures* e *Triggers*;
- Desenvolvimento de um *script* de teste das funcionalidades implementadas.

1.2 Regras

- O trabalho deverá basear-se na submissão efetuada na 1ª fase, mantendo o tema e a constituição dos grupos;
- É permitido (e encorajado) o “refinamento” do trabalho desenvolvido na 1ª fase;
- O trabalho desenvolvido deverá ser comunicado em forma de relatório, revendo e estendendo o relatório da 1ª fase.

2 Requisitos mínimos

O projeto da disciplina consiste no desenvolvimento de uma base de dados de suporte a uma aplicação de **Gestão Integrada de Atividades Desportivas** cujo tema de aplicação já está definido desde a 1ª fase.

Nota preliminar sobre requisitos mínimos:

Dos requisitos presentes neste enunciado, destacam-se as seguintes obrigаторiedades: Os *scripts* desenvolvidos deverão ser os seguintes:

1.1 Criar a base de dados – (ficheiro: **create.sql**)

- Criar a base de dados e todas as estruturas de suporte indicando os tipos de dados de cada coluna, as restrições associadas às colunas ou tabelas e reforçando a integridade referencial, sempre que for possível e aconselhável.

1.2 Componente lógica – (ficheiro: **logic.sql**)

- Criação de, pelo menos, 5 *views*;
- Criação de, pelo menos, 2 *stored functions*;
- Criação de, pelo menos, 10 *stored procedures*;
- Criação de, pelo menos, 2 *triggers*.

1.3 Carregamento inicial de dados na base de dados – (ficheiro: **populate.sql**)

- Carregar a base de dados com um mínimo de 20 registos em cada tabela. Excetuam-se tabelas onde tal seja manifestamente irrealista ou inadequado (Exemplos: *tbl_estado_civil*, *tbl_genero*). Nesta componente prevê-se o recurso a instruções **DML** + definição de **Stored Procedures**.

1.4 Consultas à base de dados – (ficheiro: **queries.sql**)

- Implementação das consultas à base de dados (ver tópico sobre consultas).

1.5 Registo de resultados – (ficheiro: **results.sql**)

- As tabelas relacionadas com o registo de resultados das provas deverão ser populadas e manipuladas através deste *script* de teste onde seja ilustrado o funcionamento dos **stored procedures** que deverão ser implementados para tal.

1.6 Teste de funcionalidade – (ficheiro: test_triggers.sql)

- Criação de um *script* que permita testar todas as funcionalidades associadas aos *triggers* implementados.

1.7 Teste de funcionalidade – (ficheiro: test.sql)

- Criação de um *script* que permita testar todas as funcionalidades desenvolvidas.

Os requisitos a desenvolver deverão ser organizados por tipo de operação.

3 Consultas à base de dados – (Views e/ou Procedures) (ficheiro: queries.sql)

As consultas deverão apresentar a informação de forma legível, isto é, em vez de mostrar uma coluna com o nome do atributo respetivo, (exemplo: **user_name**), deverá apresentá-lo de forma compreensível para o utilizador (exemplo: **'Nome Utilizador'**).

Os comandos relativos a este ponto devem ser colocados no ficheiro **queries.sql** e, cada um deles, deve ser precedido por um comentário indicando o ponto a que corresponde.

Exemplo

```
-- Q1.1 (número identificador da consulta)
-- Listagem de participantes que competiram nas provas femininas (descrição)
SELECT * FROM Participante WHERE sexo='F' ORDER BY 1,3;
```

Devem ser implementados os comandos SQL que permitem obter os seguintes resultados:

1. Lista de participantes segundo, pelo menos, 2 critérios de consulta:
 - 1.1. Critério I;
 - 1.2. Critério II.
2. Lista de equipas e respetivos elementos segundo, pelo menos, 2 critérios de consulta:
 - 2.1. Critério I;
 - 2.2. Critério II.
3. Lista de provas do evento segundo dois critérios:
 - 3.1. Critério I;
 - 3.2. Critério II.
4. Lista de alojamentos segundo, pelo menos, 3 critérios:
 - 4.1. Critério I;
 - 4.2. Critério II;
 - 4.3. Critério III.
5. Lista de resultados de cada prova realçando os participantes que foram medalhados (3 primeiros de cada prova);
6. Lista de participantes individuais que não participaram em qualquer prova;
7. Lista, organizada por tipologia, dos alojamentos com indicação da área média, mínima, máxima e desvio padrão;
8. Lista com o número médio, mínimo, máximo e desvio padrão dos participantes por evento, segundo, pelo menos, 2 critérios:
 - 8.1. Critério I;
 - 8.2. Critério II.
9. Lista com o número médio, mínimo, máximo e desvio padrão dos participantes por prova, segundo, pelo menos, 2 critérios:
 - 9.1. Critério I;
 - 9.2. Critério II.

10. Lista de alojamentos cujo total de itens associados (micro-ondas, forno, A/C, etc.) é superior a 4. (**Nota:** Deve apresentar o número total de itens do alojamento). A listagem deve ser ordenada e apresentar no topo os alojamentos com maior número de itens;
11. Top 5 das provas com maior número de participantes;
12. Lista dos 5 participantes com mais medalhas (seja a participação individual ou coletiva);
13. Consulta adicional recorrendo a, pelo menos, 3 tabelas;
14. Consulta adicional recorrendo a, pelo menos, 3 tabelas que inclua WHERE e HAVING;
15. Consulta adicional usando descrições de dados existentes num relacionamento recursivo.

4 Registo de resultados – (ficheiro: results.sql)

O registo (inserção, alteração e remoção) de resultados nas provas deve ser feito através de *stored procedures* (**sp**). Cada *stored procedure* deve ser precedido por um comentário indicando o ponto a que corresponde.

Exemplo:

```
-- SP1 (número identificador do stored procedure)
-- criar registo de novo prato com origem na zona oeste
CREATE PROCEDURE sp_criar_prato(...)
```

Deverão ser implementados os seguintes *stored procedures*:

1. **sp_criar_prova** – Cria uma nova prova num evento determinado, enviando todos os dados necessários à definição da mesma;
2. **sp_adicionar_participante(id_prova, ...)** – Adiciona um atleta à lista de atletas/equipas que irão competir na prova indicada de acordo com o tipo de prova;
3. **sp_registar_resultado(id_prova, id_participante, ...)** – Regista o resultado do participante na prova indicada;
4. **sp_remover_prova(id_prova, force, ...)** – Remove a prova identificada no parâmetro, nas seguintes circunstâncias:
 - a. Caso não existam resultados associados à prova (ou outros registos que sejam dependentes);
 - b. Caso existam resultados associados à prova e tenha sido enviado "True" no parâmetro **force**;
 - c. Caso contrário, devolve um erro.
5. **sp_clonar_prova(id_prova, ...)** – Cria uma nova prova com uma cópia de todos os dados existentes na prova indicada como parâmetro. A única exceção é que, à descrição da prova, deverá ser adicionada a string " --- COPIA (a preencher)".

Exemplo: "100 metros" → "100 metros --- COPIA (a preencher)"

O mecanismo descrito por estes procedimentos pode ser alterado desde que devidamente justificado e validado antecipadamente à entrega pelo docente das práticas.

5 Remoção de dados

A remoção de dados deve ser encarada com a sensibilidade de manter a integridade referencial dos dados já existentes. Não devem, por isso, ser utilizadas chaves estrangeiras com a opção **ON DELETE CASCADE**. Como tal, devem ser desenvolvidos os **stored procedures** considerados apropriados para remoção de dados assegurando a integridade referencial.

6 Monitorização de falhas

Implemente os seguintes *triggers* que permitem fazer a monitorização de alterações nos resultados:

- **result_change** – Regista na tabela de **tbl_logs** (a criar na base de dados), pelos menos, os seguintes dados:
 - Data e hora da operação;
 - Identificação do atleta/equipa;
 - Identificação da prova;
 - Resultado anterior;
 - Novo resultado;
 - Posição (1º, 2º, ...).
- Outro *trigger* (nome e características a definir pelo grupo) que mantenha um log atualizado com informação sobre os resultados removidos da base de dados.

7 Entregas – Scripts

Deverão ser entregues *scripts* com os seguintes propósitos:

NOTA: Ficheiros com erros de sintaxe/execução não serão considerados nem as funcionalidades que, eventualmente, pudessem facultar, caso a sua implementação fosse a correta.

create.sql	Contendo a definição do modelo relacional;
logic.sql	Contendo todas as <i>views</i> , <i>funções</i> , <i>stored procedures</i> e <i>triggers</i> ;
populate.sql	Contendo todas as instruções utilizadas para popular a base de dados (correndo os <i>scripts</i> por esta sequência, poderão fazer uso dos <i>stored procedures</i>);
queries.sql	Contendo conjunto das consultas à base de dados;
results.sql	Contendo os testes à gestão de informação associada aos resultados das provas;
test_triggers.sql	Contendo os testes aos <i>triggers</i> ;
test.sql	Contendo instruções que permitam testar as funcionalidades desenvolvidas (é esperado que o <i>script</i> seja o resultado natural dos vossos testes). Este <i>script</i> será central na discussão do projeto.

O *script* de teste deve conter chamadas a todas as funcionalidades desenvolvidas, acompanhadas por consultas que permitam verificar a correção do funcionamento das funcionalidades.

8 Relatório

O relatório deverá conter toda a informação relativa à 1ª fase, acrescido da descrição das funcionalidades desenvolvidas na 2ª fase. Qualquer código-fonte e/ou *script* só poderá ser colocado em “anexo” e não como parte central do relatório. Dever-se-á privilegiar uma listagem/resumo das funcionalidades implementadas com uma pequena descrição individual e nas conclusões deverão relatar adicionalmente as limitações e/ou o que não foi desenvolvido.

9 Datas e Entrega

Haverá aula de laboratório dedicada a apoio ao projeto (não obstante da consideração dos restantes laboratórios se oportuno e dos horários de dúvidas) na semana de 09/05, 23/05 e 06/06 2022.

A entrega deverá ser efetuada por apenas um elemento do grupo no link respetivo relativo à turma do docente das aulas práticas a que pertence o grupo.

Todos os componentes do projeto devem ser entregues num único ficheiro ZIP cujo nome deve seguir a seguinte nomenclatura:

`nºaluno1_PrimeiroNomeUltimoNome_nºaluno2_PrimeiroNomeUltimoNome_Docente_TurnoLab.zip`

Exemplo: `202201234_AntonioSantos_RuteMorais_CS_5F14h.zip`

Em que:

- **Nome e apelido:** sem acentos/cedilhas e com a primeira letra maiúscula;
- **Números de aluno:** nºaluno1 < nºaluno2;
- **Docente:** Iniciais do primeiro e último nome;
- **TurnoLab:** Dia da semana e hora de início do Laboratório (Exemplo: 2F11h).

O relatório deverá ser submetido em formato PDF juntamente com os *scripts* desenvolvidos (em ficheiros separados, embora o código possa também ser parte integrante na respetiva secção de anexos do relatório), num ficheiro ZIP com a designação anteriormente indicada, até às 13h00 do dia 11 de junho de 2022 na plataforma Moodle.

Será aplicada uma penalização de 0.1 valores por cada hora de atraso na submissão, com o limite máximo de entrega a ser as 23h55 do dia 11/06). Após as 24h de 11/06 é bloqueada a submissão e o projeto só poderá ser entregue em época de recurso seguindo o respetivo enunciado dessa época.

As discussões terão lugar nos laboratórios das semanas de 13/06 e 20/06 de 2022.

10 Avaliação:


- **Relatório – 10%**
- **Revisão do Modelo apresentado na fase 1 – 10%**
- **Implementação dos requisitos (40%)**
 - Grau de cumprimento dos requisitos mínimos;
 - Nível de desenvolvimento da lógica do modelo face aos requisitos próprios do projeto;
 - Grau de cumprimento dos requisitos específicos.
- **Scripts de teste e demonstração de funcionalidades – 40%**


11 Convenções de escrita (opcional):


Embora cada grupo/indivíduo possa ter as suas preferências para a forma como denomina as tabelas, campos, *stored procedures*, etc., é importante acompanhar as (boas) práticas do sector. Assim, propõe-se a utilização dos seguintes critérios para as estruturas e elementos a criar na base de dados:

- **Usar sempre minúsculas:**
 - `concelho(cod_concelho, cod_distrito, concelho)`
 - vs.
 - `Concelho(CodConcelho, Cod_Distrito, CONCELHO)`



- **Separar palavras usando *underscore*(_):**
 - `aluno_disciplina_curso(id_aluno, nome_completo, data_prevista_fim, ...)` 
 - vs.**
 - `AlunoDisciplinaCurso(IdAluno, Nome_Completo, DataPrevista_fim, ...)`

- **Usar o singular (sempre que tal faça sentido):**
 - `fatura(nr_fatura, nome, nif, nota, observacoes, total, ...)` 
 - vs.**
 - `faturas(nr_fatura, nomes, NIF, notas, observacao, totais_globais, ...)`

- **Não usar caracteres acentuados ou cedilhados:**
 - `doenca(id_doenca, designacao, id_acao, observacoes, total, ...)` 
 - vs.**
 - `doença(id_doença, designação, id_ação, observações, total, ...)`

Nota: poderá optar por usar *camelCase* em vez de *underscore* na ligação entre palavras. Deve, contudo, ser consistente ao longo do projeto.

<https://medium.com/@centizennationwide/mysql-naming-conventions-e3a6f6219efe>

[Fim do enunciado]