Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores

Trabalho prático

(Fases 1 e 2)

Sistemas de Informação

Semestre de verão de 2022/2023

Docentes: Afonso Remédios, Nuno Leite e Walter Vieira

Planeamento

As datas importantes a recordar são:

• Lançamento do enunciado: 17 de março de 2023

• Entrega intermédia (Fase 1): 1 de maio de 2023

• Entrega intermédia (Fase 2): 29 de maio de 2023.

Cada entrega intermédia deve apresentar o relatório e código (se houver) referentes exclusivamente a essa fase. O relatório deve seguir um dos modelos fornecidos, obrigatoriamente, sob pena de penalização na nota. Este deve ser conciso e apresentar a justificação de todas as decisões tomadas. A capa do relatório deve indicar a composição do grupo, a unidade curricular e a fase do trabalho a que respeita. Caso tenha adendas e/ou correções a fazer a modelos já entregues, deve indicá-las de forma explícita no relatório seguinte.

O ficheiro pdf (ou, o zip) gerado deve seguir o padrão: `TPSISINF-2223SV-GrupoNNTFaseN.ext´. N representa um dígito, T a turma (D1, D2, D3, N) e `ext´ a extensão do ficheiro, exemplo: TPSISINF-2223SV-Grupo14D2Fase1.zip ou TPSISINF-2223SV-Grupo02NFase1.zip.

Primeira fase

Objetivos de aprendizagem

No final da primeira fase do trabalho, os alunos deverão ser capazes de:

- Desenvolver um modelo de dados adequado aos requisitos, normalizado até à 3NF;
- Conceber e implementar uma solução baseada em bases de dados dinâmicas, adequada aos requisitos;
- Utilizar corretamente controlo transacional;
- Utilizar corretamente níveis de isolamento;
- Utilizar corretamente vistas;
- Utilizar corretamente procedimentos armazenados;
- Utilizar corretamente gatilhos;
- Utilizar corretamente funções;
- Saber justificar o uso na solução apresentada de vistas, procedimentos armazenados, gatilhos e funções.
- Desenvolver código de teste, em PL/pgSQL, para cada uma das funcionalidades requeridas nos requisitos;
- Desenvolver código PL/pgSQL para criar todos os objetos necessários à solução, a partir de uma base de dados vazia;
- Escrever um relatório técnico sobre as decisões tomadas e o trabalho desenvolvido.

Enunciado do trabalho (documento de requisitos do sistema)

A empresa "GameOn" pretende desenvolver um sistema para gerir jogos, jogadores e as partidas que estes efetuam. O sistema deve registar os dados dos jogadores que são identificados por um id gerado pelo sistema, tendo também o email e o *username* como valores únicos e obrigatórios. O jogador toma um dos estados 'Ativo', 'Inativo' ou 'Banido' e pertence a uma determinada região. Para cada região apenas se deve registar o seu nome. Os jogos têm como identificador uma referência alfanumérica de dimensão 10, um nome obrigatório e único e um url para uma página com os detalhes do jogo. Interessa registar os jogadores que compraram determinado jogo, a data e o preço associados à compra. Cada vez que o jogo é jogado, é criada uma partida com um número sequencial e único para cada jogo, devendo ser guardadas as datas e horas de início e de fim da partida. A partida pode ser normal de apenas um jogador, ou multi-jogador. As partidas normais devem ter informação sobre o grau de dificuldade (valor de 1 a 5) e estar associadas ao jogador que as joga e à pontuação por ele obtida. As partidas multi-jogador devem estar associadas aos jogadores que as jogam sendo necessário guardar as pontuações obtidas por cada jogador em cada partida. Devem ainda conter informação sobre o estado em que se encontram, o qual pode tomar os valores 'Por iniciar', 'A aguardar jogadores', 'Em curso' e 'Terminada'. Assume-se a existência de um sistema que atualiza a pontuação e estado durante a execução do jogo. Cada partida está associada a uma região e apenas jogadores dessa região a podem jogar. De modo a recompensar os jogadores, cada jogo pode ter um conjunto de crachás que são atribuídos aos jogadores quando um limite de pontos nesse jogo é atingido. Para isso interessa registar o nome do crachá que é único para cada jogo, o limite de pontos e o url para a imagem do crachá. Devem ficar registados na base de dados os crachás que são atribuídos a cada jogador.

Deverão existir em tabelas próprias as estatísticas associadas aos jogadores e aos jogos. Interessa registar para cada jogador, o número de partidas que efetuou, o número de jogos diferentes que

jogou e o total de pontos de todos os jogos e partidas efetuadas. Para cada jogo interessa registar o número de partidas, o número de jogadores e o total de pontos.

Os jogadores podem adicionar outros jogadores como amigos. Portanto interessa registar essa relação de amizade. É também possível criar uma conversa entre vários jogadores com um identificador gerado pelo sistema e um nome. Associado à conversa existem mensagens com um número de ordem único e sequencial para cada conversa que serve de identificador, a data e hora da mensagem, o texto e qual o jogador que enviou a mensagem.

Nota:

-Deve utilizar os tipos de dados que achar adequados.

Resultados pretendidos

Tendo em conta os objetivos de aprendizagem, deverão ser produzidos os seguintes resultados:

1. O modelo de dados (conceptual e relacional), incluindo todas as restrições de integridade;
2. O código PL/pgSQL que permite:
   (a) Criar o modelo físico (1 *script* autónomo);

   (b) Remover o modelo físico (1 *script* autónomo);

   (c) Preenchimento inicial da base de dados (1 *script* autónomo);

   (d) Criar os mecanismos que permitam criar o jogador, dados os seus email, região e *username*, desativar e banir o jogador;

   (e) Criar, sem usar as tabelas de estatísticas, a função **totalPontosJogador** que recebe como parâmetro o identificador de um jogador e devolve o número total de pontos obtidos pelo jogador.

   (f) Criar, sem usar as tabelas de estatísticas, a função **totalJogosJogador** que recebe como parâmetro o identificador de um jogador e devolve o número total de jogos diferentes nos quais o jogador participou.

   (g) Criar a função **PontosJogoPorJogador** que recebe como parâmetro a referência de um jogo e devolve uma tabela com duas colunas (identificador de jogador, total de pontos) em que cada linha contém o identificador de um jogador e o total de pontos que esse jogador teve nesse jogo. Apenas devem ser devolvidos os jogadores que tenham jogado o jogo.

   (h) Criar o procedimento armazenado **associarCrachá** que recebe como parâmetros o identificador de um jogador, a referência de um jogo e o nome de um crachá desse jogo e atribui o crachá a esse jogador se ele reunir as condições para o obter.

   (i) Criar o procedimento armazenado **iniciarConversa** iniciar uma conversa (*chat*) dados o identificador de um jogador e o nome da conversa. O jogador deve ficar automaticamente associado à conversa. O procedimento deve devolver num parâmetro de saída o identificador da conversa criada.

   (j) Criar o procedimento armazenado **juntarConversa** que recebe como parâmetros os identificadores de um jogador e de uma conversa e junta esse jogador a essa conversa.

(k) Criar o procedimento armazenado **enviarMensagem** que recebe como parâmetros os identificadores de um jogador e de uma conversa e o texto de uma mensagem e procede ao envio dessa mensagem para a conversa indicada, associando-a ao jogador também indicado.

(l) Criar a vista **jogadorTotalInfo** que permita aceder à informação sobre identificador, estado, email, *username*, número total de jogos em que participou, número total de partidas em que participou e número total de pontos que já obteve de todos os jogadores cujo estado seja diferente de "Banido". Deve implementar na vista os cálculos sem aceder às tabelas de estatísticas.

(m) Criar os mecanismos necessários para que, de forma automática, quando uma partida termina, se proceda à atribuição de crachás do jogo a que ela pertence.

(n) Criar os mecanismos necessários para que a execução da instrução DELETE sobre a vista jogadorTotalInfo permita colocar os jogadores envolvidos no estado "Banido".

(o) Criar um *script* autónomo com os testes das funcionalidades de 2d a 2n para cenários normais e de erro. Este *script*, ao ser executado, deve listar, para cada teste, o seu nome e indicação se ele correu ou não com sucesso;

Ex: "teste 1: Inserir jogador com dados bem passados: Resultado OK"

Ou caso falhe: "teste 1: Inserir Cliente com dados bem passados: Resultado FAIL"

Nota:

Todos os procedimentos armazenados devem ter tratamento de erros e gestão transacional utilizando o nível de isolamento adequado.

Data limite para entrega (após extensão do prazo): 8 de maio de 2023 até às 23:50.

A entrega tem de incluir um relatório (em formato PDF) e o código PL/pgSQL, entregues via moodle. O relatório deverá obedecer ao padrão fornecido pelos docentes.

Segunda fase

Objetivos de aprendizagem

No final da segunda fase do trabalho, os alunos devem ser capazes de:

- Desenvolver uma camada de acesso a dados, que use uma implementação de JPA e um subconjunto dos padrões de desenho DataMapper, Repository e UnitOfWork;
- Desenvolver uma aplicação em Java, que use adequadamente a camada de acesso a dados;
- Utilizar corretamente processamento transacional, através de mecanismos disponíveis no JPA;
- Garantir a correta libertação de ligações e recursos, quando estes não estejam a ser utilizados;
- Garantir a correta implementação das restrições de integridade e/ou lógica de negócio;

Resultados pretendidos

Tendo em conta os objetivos de aprendizagem, deverão ser produzidos os seguintes resultados:

1. Criação de uma aplicação Java que permita:

   (a) Aceder às funcionalidades 2d a 2l, descritas na fase 1 deste trabalho;

   (b) Realizar a funcionalidade 2h, descrita na fase 1 deste trabalho, sem usar qualquer procedimento armazenado nem qualquer função pgSql;

   (c) Realizar a funcionalidade 2h, descrita na fase 1 deste trabalho, reutilizando os procedimentos armazenados e funções que a funcionalidade original usa;

2. (a) Usando *optimistic locking*, aumentar em 20% o número de pontos associados a um crachá, dados o nome do crachá e o identificador do jogo a que ele pertence.

   (b) testar a alínea anterior, apresentando uma mensagem de erro adequada em caso de alteração concorrente conflituante que inviabilize a operação. No relatório deve estar descrita a forma como as situações de erro foram criadas para teste desta alínea;

   (c) realizar o mesmo que em (a), mas usando controlo de concorrência pessimista.

Notas:

Todas as alíneas devem ter tratamento de erros e, nos casos relevantes, gestão transacional utilizando o nível de isolamento adequado de forma explícita.

O código a desenvolver deverá estar organizado em vários projetos que reflitam a estrutura modular da aplicação, a qual deverá reduzir o acoplamento entre módulos e facilitar a identificação das dependências entre eles.

Data limite para entrega: 12 de junho de 2023 até às 23:50.

A entrega tem de incluir um relatório (em formato PDF), os projetos *MAVEN*, o código Java e o código PL/pgSQL, entregues via moodle. O relatório deverá obedecer ao padrão fornecido pelos docentes.