ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Sistema de Gestão de Jogos Fase 2

48268 – Marçorio Fortes
48315 – Rafael Costa
47539 – Bernardo Serra

Orientadores: Doutor Nuno Leite

Relatório para a Unidade Curricular de Sistemas de Informação da Licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2022/2023

8 de maio de 2023

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Sistema de Gestão de Jogos Fase 2

48268 Marçorio Fortes

48315 Rafael Costa

47539 Bernardo Serra

Orientadores: Doutor Nuno Leite

Relatório para a Unidade Curricular de Sistemas de Informação da Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2022/2023

8 de maio de 2023

Resumo

Este projeto está dividido em duas fases, sendo esta a segunda fase. Este relatório tem como objetivo a elaboração da camada de acesso a dados para a base dados desenvolvida na primeira fase deste trabalho (base de dados para um sistema de gestão de jogos).

Nesta fase do trabalho com o objetivo de desenvolver uma camada de acesso a dados, que use uma implementação de JPA em conjunto com um subconjunto dos padrões de desenvolvimento DataMapper, Repository e UnitOfWork. Desenvolver uma aplicação em Java, que use adequadamente a camada de acesso a dados, e queremos usar corretamente o processamento transacional neste caso com os mecanismos disponíveis em JPA.

Também, temos como objetivo garantir a correta implementação das restrições de integridade e logica de negócio.

Este relatório parte do pressuposto do acesso por parte do leitor ao código desenvolvido no âmbito do mesmo, não sendo assim necessário enunciá-lo em extensão, bastando apenas mencionar trechos do mesmo.

Abstract

This project is divided into two phases, this being the second phase. This report aims to elaborate the data access layer for the database developed in the first phase of this work (database for a game management system).
In this phase of the work, the objective is to develop a data access layer, which uses a JPA implementation in conjunction with a subset of the DataMapper, Repository and UnitOfWork development patterns. Develop an application in Java, which properly uses the data access layer, and we want to correctly use transactional processing in this case with the mechanisms available in JPA.
Also, we aim to ensure the correct implementation of integrity constraints and business logic.
This report is based on the assumption that the reader has access to the code developed within the scope of the same, therefore it is not necessary to state it in length, just mentioning excerpts from it.

Agradecimentos

Queremos agradecer ao Professor Doutor Nuno Leite pelo grande apoio que nos foi proporcionado em aulas e fora das aulas.

Índice

1. Introdução

Nesta parte do relatório falaremos de aspetos teóricos que foram necessários para a execução do trabalho.

1.1 Data Mapper

Para os casos mais simples, o acesso a dados pode ser realizado com base num conjunto de objetos designados mappers (um por cada entidade) que são responsáveis por realizar as operações CRUD da entidade a que estão associados.

1.2 Repository

Por vezes, torna-se necessário manipular as entidades de forma mais complexa, por exemplo, lendo coleções de entidades com determinado critério. Nestes casos, o padrão Repository pode ajudar.

As implementações dos repositórios podem usar os Mappers já existentes, criando uma nova camada de abstração que, basicamente, virtualiza as fontes de dados.

1.3 UnitOfWork

Estpadrão simplifica a gestão do ciclo de vida das entidades e facilita, sem obrigar, a utilização desligada (as ligações à base de dados apenas são estabelecidas durante a leitura das entidades e quando se pretende propagar valores para a BD, ficando, entretanto, as entidades armazenadas no objeto que implementa o padrão)**.**

1.4 Entity Manager

Estes objetos permitem manter os contextos de persistência, controlando os acessos à base de dados, as transações e o ciclo de vida das instâncias das entidades geridas (managed).

Existem vários tipos, mas apenas veremos os application-managed entity managers. A propriedade jakarta.persistence.EntityTransaction getTransaction() permite obter uma instância que possibilita a realização de controlo transactional (begin, commit, rollback, …).

1.5 Mapeamento de Entidades

Os mapeamentos de entidades são uma parte fundamental da camada de acesso a dados ao utilizar a JPA. Através desses mapeamentos, é possível definir como as entidades da aplicação são persistidas e recuperadas do banco de dados. Existem diferentes formas de realizar esses mapeamentos, dependendo das necessidades e complexidade do sistema.

A JPA permite definir relações entre entidades, como relações um-para-um, um-para-muitos e muitos-para-muitos. Essas relações são mapeadas através de anotações, como **@OneToOne**, **@OneToMany** e **@ManyToMany**. É possível especificar o tipo de relacionamento, a chave estrangeira, as colunas envolvidas, entre outros detalhes. Para relações muitos-para-muitos com atributos temos que fazer uma relação associativa.

1.6 Controlo de Concorrência

O controle de concorrência é uma técnica utilizada em bancos de dados para gerenciar o acesso simultâneo a recursos compartilhados, como tabelas ou registos, por múltiplas transações concorrentes. Duas abordagens comuns de controle de concorrência são o pessimista e o otimista.

1. **Controle de Concorrência Pessimista:**

   No controle de concorrência pessimista, assume-se que conflitos entre transações são frequentes e, portanto, medidas de prevenção devem ser tomadas para evitar que problemas ocorram. Nessa abordagem, bloqueios são adquiridos em recursos compartilhados para garantir que apenas uma transação possa modificá-los por vez. Isso significa que outras transações que desejam acessar os mesmos recursos precisam esperar até que o bloqueio seja liberado.

   O controle de concorrência pessimista é geralmente implementado através do uso de bloqueios de leitura (para transações que apenas leem dados) e bloqueios de escrita (para transações que modificam dados). Esses bloqueios são mantidos durante toda a duração da transação, o que pode resultar em um desempenho mais baixo quando há um alto nível de concorrência.

2. **Controle de Concorrência Otimista:**

   No controle de concorrência otimista, assume-se que conflitos entre transações são menos prováveis de ocorrer, e portanto, medidas de prevenção não são tomadas inicialmente. Nessa abordagem, as transações podem acessar e modificar os recursos compartilhados livremente, sem bloqueios. No entanto, quando uma transação tenta confirmar suas modificações, ocorre uma verificação para garantir que nenhum conflito tenha ocorrido.

   Essa verificação pode ser feita comparando a versão dos dados que a transação leu inicialmente com a versão atual dos dados no banco de dados. Se não houver conflitos, a transação é confirmada. Caso contrário, ela precisa ser revertida e as modificações são descartadas.

O controle de concorrência otimista é especialmente útil em situações onde conflitos são raros, pois não há necessidade de bloqueios durante a execução das transações. No entanto, se ocorrerem conflitos com frequência, a taxa de reversão e execução das transações pode diminuir o desempenho do sistema.

Cada abordagem possui suas vantagens e desvantagens, e a escolha entre controle de concorrência pessimista e otimista depende do cenário de uso e das características da aplicação. É importante avaliar as necessidades de consistência, desempenho e concorrência da aplicação para determinar a melhor abordagem a ser adotada.

2. Formulação do Problema

Para desenvolver uma aplicação Java robusta e eficiente, é fundamental contar com uma camada de acesso a dados bem estruturada. Para isso, é recomendado utilizar uma implementação da API de Persistência do Java (JPA), juntamente com um subconjunto dos padrões de desenho DataMapper, Repository e UnitOfWork.

A camada de acesso a dados é responsável por interagir com a fonte de dados subjacente, seja ela um banco de dados relacional, um sistema de arquivos ou qualquer outro meio de armazenamento. Ao utilizar uma implementação da JPA, podemos aproveitar as vantagens oferecidas por esse framework, como a facilidade de mapeamento entre objetos e recursos transacionais.

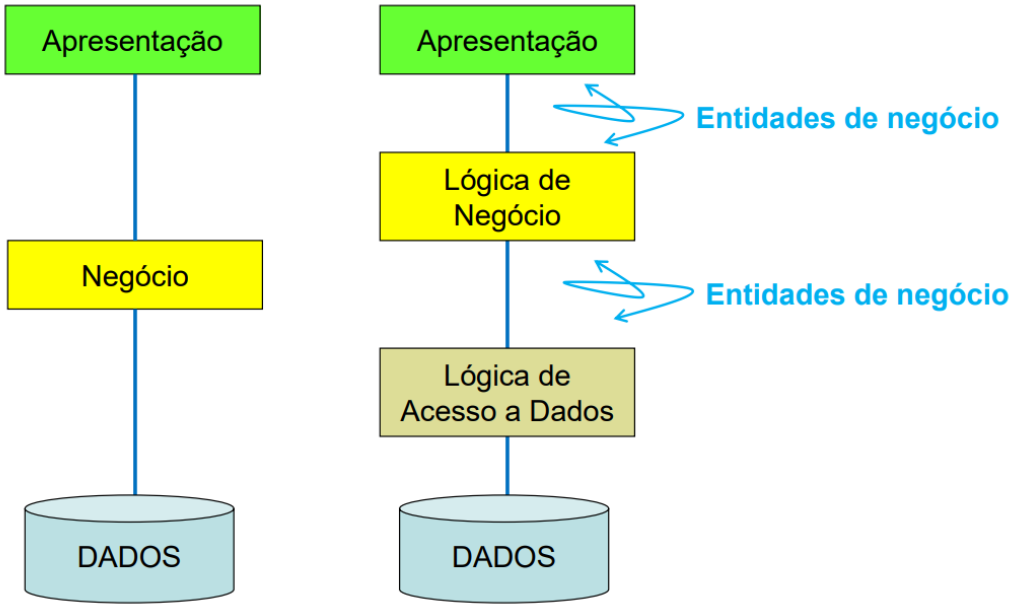
A arquitetura baseada nos padrões de desenho DataMapper, Repository e UnitOfWork permite uma organização e estruturação mais eficientes do código relacionado ao acesso a dados. O DataMapper é responsável por mapear os objetos de domínio para o esquema de armazenamento subjacente, enquanto o Repository fornece uma interface abstrata para interagir com os dados. Por sua vez, o UnitOfWork gerencia as transações e garante a consistência dos dados durante as operações.

Além disso, é essencial utilizar corretamente o processamento transacional disponível na JPA. As transações garantem a integridade e a consistência dos dados, permitindo que as operações sejam realizadas atomicamente e revertidas em caso de falhas. É importante compreender e utilizar adequadamente as anotações e os métodos transacionais fornecidos pela JPA para garantir a execução correta das transações.

Outro aspecto fundamental é a correta libertação de ligações e recursos quando estes não estão sendo utilizados. Recursos como conexões de banco de dados, arquivos ou outros meios de acesso a dados devem ser libertados de forma apropriada para evitar vazamentos e otimizar o desempenho da aplicação.

Por fim, a implementação das restrições de integridade e/ou lógica de negócio é um requisito indispensável para garantir a consistência e a qualidade dos dados manipulados pela aplicação. Isso pode envolver a validação dos dados, a verificação das chaves estrangeiras, as restrições dos valores aceitáveis, entre outros aspetos. É importante implementar essas restrições tanto no nível do banco de dados quanto no código da camada de acesso a dados.

Em resumo, o desenvolvimento de uma camada de acesso a dados que utiliza uma implementação da JPA e os padrões de desenho mencionados, com a correta utilização do processamento transacional, a libertação adequada de recursos e a implementação correta das restrições de integridade e/ou lógica de negócio, contribuirá para uma aplicação Java robusta, eficiente e confiável.



3. Detalhes de Implementação

3.1 Nota Prévia

Uma vez mais, são apenas referidos detalhes da implementação que achámos relevantes para compreensão de como o trabalho foi efetuado e a razão das decisões tomadas no mesmo.
Não sendo mencionado aqui alíneas inteiras pedidas no relatório, por considerarmos que a logica das mesmas é trivial ou semelhante à de outras alíneas anteriores ou posteriores já explicadas.

3.2 Modelo.

Foi criado uma classe para cada entidade do modelo conceptual, e também foi criado uma classe extra para a relação de plays_multi porque tinha um atributo e era uma reação muitos-para-muitos, desta forma foi feita uma classe que representa uma relação associativa. Da mesma forma, para as tabelas que tinham chave compostas foi criado uma classe extra para representar a chave.

3.3 Acesso as Funcionalidades

Para aceder as funcionalidades em pgsql (procedimentos e funções), foi necessário usar mecanismos do JPA. Todos foram de fácil acesso. Tivemos apenas que realizar um encapsulamento do procedimento iniciar conversa em uma função para que fosse possível receber o novo id da conversa.

Também, para realização do procedimento associar crachá em JPA foi necessário a criação de Mappers e Repositorys utilizando o AccessScope para as seguintes classes: Badge, Player, PlaysMulti e SinglePlayerMatch. Como era de se esperar a classe Badge e Player seriam necessários para a atribuição do crachá. PlaysMulti e SinglePlayerMatch serviram para fazer a contabilização da pontuação.

3.4 Optimistic Locking e Pessimist Locking

Para a realização do optimistic locking foi necessário adicionar a classe **Badge** uma nova anotação @OptimisticLocking(cascade=true,type=OptimisticLockingType.CHANGED_COLUMNS).
Para além de ter os parâmetros gameRef e badgeName que são respetivamente a referência do jogo o nome do crachá, também tem um parâmetro extra chamado attempts, este parâmetro serve para indicar quantas tentativas queremos até a operação lançar uma exceção.
Uma forma de causar o erro no optimistic locking, seria correndo a aplicação java em modo debug. E executar um UPTADE na tabela badge durante o debug.

3.5 Alterações e Atualizações realizadas

Na primeira fase do trabalho houve um engano com a chave da tabela single_player_match e multi_player_match e plays_multi. No entanto, esse esse erro foi corrigido.

4. Conclusões

Neste trabalho tratou-se o problema da gestão do sistema de gerenciamento de jogos onde tivemos de aplicar conhecimentos anteriormente adquiridos e conhecimentos adquiridos ao longo do semestre.

Em resumo, ao desenvolver uma aplicação em Java com uma camada de acesso a dados, é importante utilizar uma implementação de JPA (Java Persistence API) e seguir um subconjunto dos padrões de desenho DataMapper, Repository e UnitOfWork.

A camada de acesso a dados permite que a aplicação interaja com o banco de dados de forma eficiente e organizada. Utilizando o JPA, é possível mapear objetos Java para entidades do banco de dados e realizar operações CRUD (Create, Read, Update, Delete) de forma simplificada.

Além disso, é essencial utilizar corretamente o processamento transacional, aproveitando os mecanismos disponíveis no JPA. Transações garantem a integridade dos dados e permitem que operações sejam realizadas de forma consistente e atomicamente.

Além disso, é fundamental implementar corretamente as restrições de integridade e/ou lógica de negócio definidas para o sistema. Isso envolve a validação de dados inseridos ou atualizados, a aplicação de regras de negócio específicas e a garantia de que as operações respeitem as restrições definidas.

Com esta primeira segunda do trabalho conseguimos obter conhecimentos muito importantes para o nosso curso, e como foi referido no primeiro relatório, neste momento estamos aptos para realizar o projeto final com uma melhor facilidade.

A solução obtida atingiu resultados satisfatórios.

Referências

[1] Material de estudo fornecido pelo Prof. Walter Vieira (slides com a matéria).

[2]