

**Segunda Fase Do Trabalho De Sistemas De Informação – T43D – G05**

|  |  |
| --- | --- |
|  | Nuno Bartolomeu |
|  | João Viegas  Miguel Moreira |

|  |  |  |
| --- | --- | --- |
| Professor: | Nuno Leite |  |
|  |  | |

Relatório final realizado no âmbito de Sistemas de Informação,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2022/2023

Junho de 2023

**Instituto Superior de Engenharia de Lisboa**  
Licenciatura em Engenharia Informática e de Computadores

**Segunda Fase Do Trabalho De Sistemas De Informação – T43D – G05**

|  |  |
| --- | --- |
| 47233 | Nuno António Oliveira Bartolomeu |
| 47208 | João Francisco Nunes Viegas |
| 46092 | Miguel Sousa Moreira |

|  |  |
| --- | --- |
| Professor: | Dr. Nuno Miguel da Costa de Sousa Leite |
|  |  |

Relatório final realizado no âmbito de Sistemas de Informação,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2022/2023

Junho de 2023

# Resumo

A segunda fase do trabalho de sistemas de informação visa perceber as capacidades dos alunos para desenvolver uma camada de acesso a dados, que use uma implementação JPA e um subconjunto de padrões de desenho, nomeadamente DataMapper, Repository e UnitOfWork.

Também verifica as capacidades de desenvolver uma aplicação Java que use os padrões de desenho referidos anteriormente, garantindo que o processo transacional é feito corretamente através de mecanismos disponíveis no JPA.

Finalmente, serão verificadas as técnicas para a correta libertação de ligações e recursos, assim como a correta implementação de restrições de integridade / lógica de negócio.

**Índice**

Resumo 4

1. Problema 1
2. Solução 2

2.1 Passagem de Tabelas para Classes 2

2.2 Lógica de negócio 3

2.3 Modelo 3

2.4 Funcionalidades 4

2.5 Testes 4

1. Conclusões 5

Referências 6

# 

# Problema

O problema a resolver nesta fase é a criação de uma aplicação que utilize a base de dados criada na primeira fase, atualizando a informação através de uma camada de acesso a dados, nomeadamente o JPA.

# Solução

A nossa solução é apresentada passo-a-passo neste capítulo.

## 2.1 Passagem de Tabelas para Classes

Após uma cuidadosa verificação, o grupo chegou a um total de 13 classes que serão necessárias para representar o modelo de dados :

* Jogadores;
* Regiões;
* Jogos;
* Compras;
* Partidas Normais;
* Partidas Multi-jogador;
* Pontuação Multi-Jogador;
* Crachás;
* Estatísticas Jogador;
* Estatisticas Jogo;
* Amigos;
* Conversas;
* Mensagens;

O modelo da primeira fase teve de sofrer algumas alterações, nomeadamente:

* A tabela pontuações foi alterada para pontuações multi-jogador, esta nova tabela apenas contém informações relativas às pontuações das partidas multi-jogador.
* Acréscimo do atributo pontos à tabela partidas normais, esta mudança e a mudança anterior, servem para corrigir um erro da fase anterior onde a pontuação não estava de acordo com o diagrama o que causou problemas nesta fase.
* Acréscimo de algumas restrições da lógica de negócio (e.g. O email em jogador passa a ser “unique”).

## 2.2 Lógica de negócio

Foram implementados alguns requisitos da lógica de negócio no código Java:

* Adicionou-se uma verificação para garantir que um jogador que esteja banido não possa comprar jogos, ou jogar partidas.
* Garantiu-se que os jogadores só podem jogar partidas que sejam feitas na mesma região que eles.
* Jogadores inativos não conseguem criar conversas ou enviar mensagens.

## 2.3 Modelo

O modelo está presente na pasta de código na subdiretoria: /src/main/java/model. Neste modelo é possível ver que cada entidade tem a sua própria pasta, onde está sempre implementada a entidade com uma classe com o mesmo nome, nesta classe estão definidos todos os atributos e os “getters” e “setters” respetivos.

Também está implementada uma classe com “Embeddable” para as classes cuja chave primária é constituída por 2 ou mais atributos. E em alguns casos está implementado um Mapper ou um Repositório da classe para facilitar o acesso a dados.

## 2.4 Funcionalidades

As funcionalidades do projeto estão na pasta de código na subdiretoria: /src/main/java/exercises. A classe Exercise1\_A contem o código de acesso às funcionalidades desenvolvidas na primeira fase. A classe Exercise1\_B e Exercise1\_C contém o código para a associação de um crachá a um jogador caso este tenha os pontos necessários, utilizando apenas código JPA. A diferença entre o B e o C é a utilização de procedimentos armazenados que foram utilizados no código original, o B não faz uso de nenhum procedimento armazenado enquanto que o C utiliza o procedimento “PontosJogoPorJogador”.  
 Também existe uma classe chamada Exercise2, esta classe contém o código necessário para realizar o segundo exercício proposto. Neste exercício é necessária a criação de uma função que aumente o número de pontos de um cracha em 20% de forma Otimista e Pessimista.   
 Para a forma otimista foi adicionada a anotação “

@OptimisticLocking(type = OptimisticLockingType.CHANGED\_COLUMNS)” à classe Cracha, o que garante que todas as operações que utilizem a entidade terão um comportamento Otimista a não ser que seja especificado o oposto. Para a versão pessimista foi utilizado o find com a especificação pessimista “ Cracha cracha = em.find(Cracha.class, crachaId, LockModeType.PESSIMISTIC\_READ);”

## 2.5 Testes

Os testes de todas as funcionalidades podem ser encontrados na subpasta: /src/test/java/exercises. Algo a notar sobre os testes é a existência de um delay de 100 ms presente nas funções do exercício 2. Este delay é necessário para garantir que ambas as threads leem a mesma informação da base de dados, garantindo colisão.

# Conclusões

Neste trabalho recebemos um problema que pode ser facilmente apresentado a um programador no mundo profissional. Aprendemos a idealizar o código primeiro para podermos ter um caminho concreto a seguir e a verificar as funcionalidades após ser implementadas para garantir que fazem o que é suposto. Como esta é a última parte do trabalho fizemos alterações à primeira fase para garantir o bom funcionamento do código.

# Referências

[1] Postgres, home page, https://www.postgresql.org/.

[2] Postgres Tutorial, https://www.postgresqltutorial.com/.

[3] Ramez Elmasri, Shamkant B. Navathe, Pearson Education, Fundamentals of Database Systems” *7th Edition 2015*.