



Trabalho Prático Da Disciplina de
Sistemas De Informação
Fase 2

José Silva

Aureliu Iurcu

Henrique Fontes

Professor Walter Vieira

Relatório realizado no âmbito da disciplina de Sistemas de Informação,
do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2021/2022

Junho de 2022

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Trabalho Prático da Disciplina de
Sistemas de Informação
Fase 2

48349 José Abiúd Manuel Da Silva

48285 Aureliu Iurcu

48295 Henrique Figueiredo Mota Duarte Fontes

Professor: Walter Vieira

Relatório realizado no âmbito da disciplina de Sistemas de Informação,
do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2021/2022

Junho de 2022

Resumo

Em sequência do desenvolvimento de um modelo de dados adequado aos requisitos fornecidos, tais como os produzidos na primeira fase de desenvolvimento do trabalho prático, a saber: conceção e implementação de uma solução baseada em bases de dados dinâmicas, adequada aos requisitos; utilização correta do controlo transacional; utilização correta de níveis de isolamento, vistas, procedimentos armazenados, gatilhos e funções. Nesta segunda fase do trabalho tivemos como objetivos o desenvolvimento uma camada de acesso a dados, que use uma implementação de *JPA* e um subconjunto dos padrões de desenho *DataMapper*, *Repository* e *UnitOfWork*; o desenvolvimento de uma aplicação em *Java*, que use adequadamente a camada de acesso a dados; utilização correta de um processamento transacional, através de mecanismos disponíveis no *JPA*; garantir a correta libertação de ligações e recursos, quando estes não estejam a ser utilizados; e finalmente, garantir a correta implementação das restrições de integridade e/ou lógica de negócio.

Tendo em conta os objetivos de aprendizagem nesta corrente fase, devíamos criar uma aplicação em *Java* que acesse algumas das funcionalidades da fase anterior, criação de uma aplicação que não utilizasse procedimentos armazenados, reutilizar os procedimentos armazenados, a reimplementação de uma das funcionalidades utilizando o método *optimistic locking* e o teste do *optimistic locking*.

Índice

Introdução

1. Introdução.....	1
1.1 Tema	1
1.2 Organização em Etapas.....	1
2. Abordagem.....	3
2.1 Camada de acesso a dados	3
2.2 Desenvolvimento dos objetivos a realizar	4
3. Teste da implementação do Optimistic Locking	7
4. Conclusões.....	10

Lista de Figuras

Figura 1 – <i>IMapper</i>	3
Figura 2 – <i>IRepository</i>	4
Figura 3 – Resultado do <i>Optimistic Lock</i>	7

1. Introdução

O presente trabalho tem como base a continuação do desenvolvimento do modelo de dados utilizado na primeira fase do trabalho, porém desta vez foi concebido através da aplicação *Java Persistence API (JPA)*, que foi um dos objetivos de estudos no terceiro módulo da matéria que tem sido lecionada.

1.1 Tema

O tópico presente advém da intenção de uma empresa denominada de “OnTrack” desenvolver um sistema para gerir e registar a localização de automóveis e camiões. Deste modo, surge a necessidade de guardar informação relativa a, por exemplo, clientes, veículos e equipamentos.

1.2 Organização em Etapas

De forma a abordar os temas base desta fase do trabalho, o mesmo foi produzido em etapas. A primeira etapa visou a construção da camada de acesso a dados que auxiliou o desenvolvimento do trabalho.

De seguida seguiu-se a ordem estipulada dos exercícios a realizar, tendo em contas os objetivos a concluir.

2. Abordagem

Aprofundou-se o acesso e controlo da base de dados através da *API* a utilizar.

2.1 Camada de acesso a dados

O desenvolvimento desta fase tem como pressuposto o aprofundamento do domínio da camada de acesso a dados, que facilita o controlo e elaboração de novas funcionalidades estabelecidas. Para isso, através do estabelecimento de um *DataScope* foram implementadas duas interfaces: *IMapper* e *IRepository*.

Na *Figura 1* é possível observar a implementação da interface *IMapper*, que possui as operações *CRUD* (*Create*, *Read*, *Update* e *Delete*) que são independentes das características específicas da *SGBD*.

```
public interface IMapper<Table, TId> {  
    10 implementations  
    void create(Table e) throws Exception;  
    10 implementations  
    Table read(TId id) throws Exception;  
    10 implementations  
    void update(Table e) throws Exception;  
    10 implementations  
    void delete(Table e) throws Exception;  
}
```

Figura 1- IMapper

Já na *Figura 2* é possível observar a implementação da interface *IRepository*, que contém as funções *getAll*, que permite obter sob a forma de lista todos os objetos de uma entidade; *find*, que através de uma chave-primária passada como parâmetro retorna o objeto entidade; *add*, para adicionar um objeto entidade na base de dados da entidade em causa; a função *delete* que permite apagar um objeto entidade da base de dados; e a função *save* que permite guardar na base de dados um objeto entidade.

```
10 implementations
public interface IRepository<Entity, Key> {
    10 implementations
    List<Entity> getAll() throws Exception;
    10 implementations
    Entity find(Key k) throws Exception;

    10 implementations
    void add(Entity entity) throws Exception;
    10 implementations
    void delete(Entity entity) throws Exception;
    10 implementations
    void save(Entity entity) throws Exception;
}
```

Figura 2 - IRepository

2.2 Desenvolvimento dos objetivos a realizar

De forma a possibilitar o acesso à base de dados, foi estabelecida uma ligação, obtendo a informação necessária como tabelas, relações, funções e procedimentos. Após este procedimento e da resolução da camada de dados, procedeu-se ao desenvolvimento das funcionalidades pretendidas nesta fase do trabalho.

Os objetivos propostos têm como objetivo a criação das seguintes funcionalidades:

No exercício *1a* concebemos uma forma de aceder as funcionalidades de cada exercício da alínea *D* a alínea *I* criados na primeira fase do trabalho através da aplicação de conceitos como *Entity Manager* e *Queries*. O exercício *1b* implicou a tradução da implementação do exercício 2h constituído anteriormente para a linguagem *Java* dando uso as funcionalidades do *JPA*, tal como das funcionalidades estabelecidas aquando da criação da camada de acesso a dados. A última alínea do exercício 1, *1c*, tem como resolução a reutilização do procedimento desenvolvido na primeira fase do trabalho na alínea *2h*, fazendo uma chamada a este procedimento por parte do *JPA*.

O exercício *2a* visou a implementação das funcionalidades da alínea *2f* através dos modelos de dados criados e da implementação de um *Optimistic Locking*. Esta implementação foi por sua vez testada na alínea *2b* tal como será explicado mais adiante.

3. Teste da implementação do *Optimistic Locking*

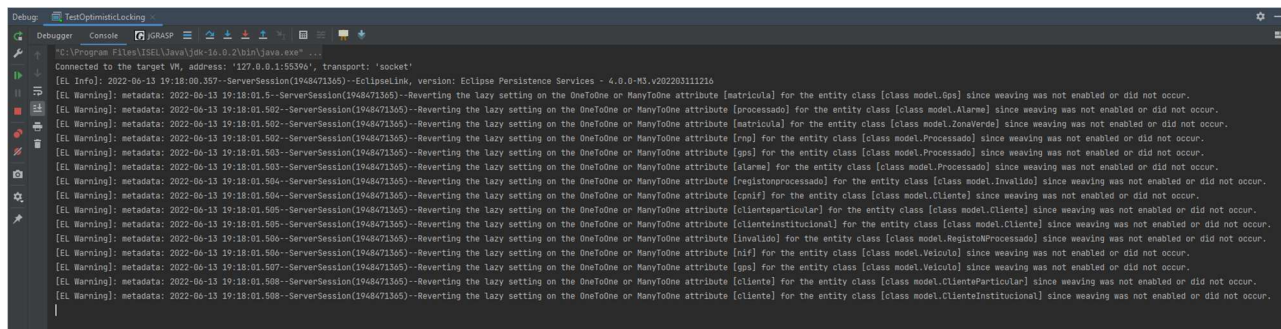
O *Optimistic Locking* aplicado no exercício 2 desta fase do trabalho visa solucionar o uso das funcionalidades *atualização* e *remoção* aquando da existência de concorrência.

Esta solução passou por adicionar uma coluna *version* (a qual denominamos de *vrs* nos nossos *scripts sql*) a cada tabela, que indica o histórico de atualizações da mesma, sendo que por definição começa com um valor '0' (zero) para cada tuplo das tabelas. Esta coluna é incrementada automaticamente cada vez que é feito uma atualização aos dados e é verificada sempre que isso acontece.

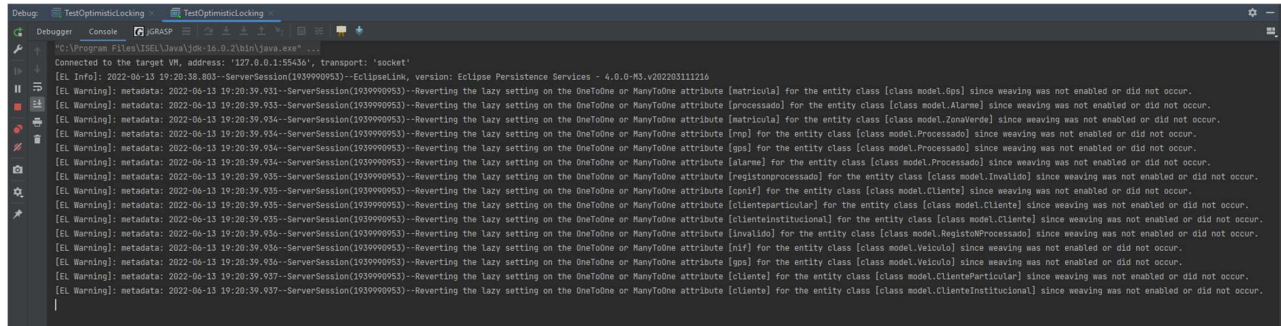
Quando a atualização está pronta a decorrer, é feita uma verificação que ocorre na coluna *vrs* que passa por uma comparação entre o valor da mesma e o valor esperado. Caso este não seja o esperado, ocorre um *lock* (bloqueio) originado pelo *Optimistic Locking*.

De forma a testar esta nova funcionalidade, a saber, o *Optimistic Locking*, procedemos à avaliação da aplicação em questão em modo de depuração (*debug*) pela chamada à função que utiliza este método (na aplicação, esta função encontra-se na pasta denominada *b*, que está dentro da pasta *exc2*) em duas instâncias distintas, sendo que a primeira em modo de depuração, já a segunda em tempo de execução (*running time*).

Na *Figura 3* é possível observar a consola do primeiro utilizador, a correr em modo *debug*, cujo *break-point* está numa das linhas da função *processUnprocessedRegisters* da *class F* da pasta *a* da diretoria *exc2*.

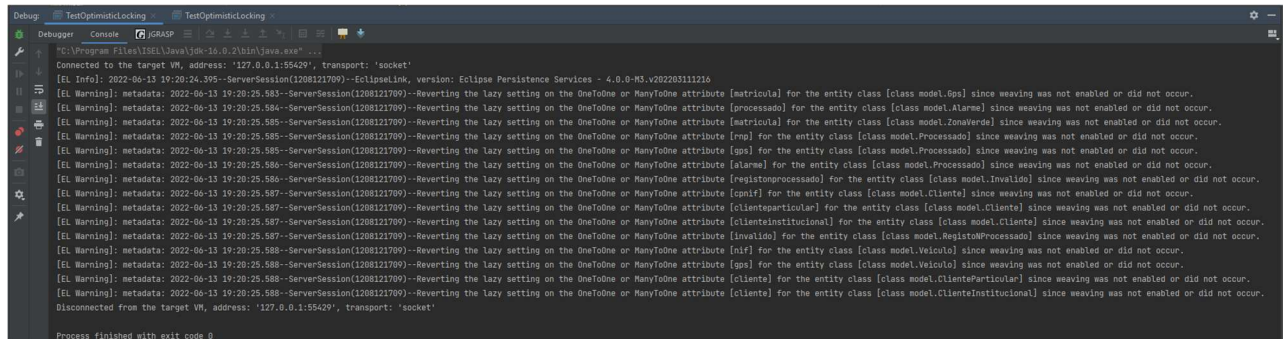


Já na *Figura 4* vemos a consola do segundo utilizador (que correr em tempo de execução). É de suma importância mencionar que o segundo utilizador apenas começou a correr a aplicação alguns segundos



depois do primeiro utilizador.

Como observado na figura acima, a consola do segundo utilizador está num ciclo que apenas é quebrado após a execução da primeira instância feita pelo primeiro utilizador, uma vez que esta instância, correspondente a uma transação, ainda não foi terminada. Logo, será correto afirmar que a transação do segundo utilizador está bloqueada, conforme previsto pelo método *Optimistic Locking*.



Após o primeiro utilizador terminar a sua transação, não é evidenciado nenhum tipo de erro, o que pode ser observado na *Figura 5*.

Figura 5 – Consola do primeiro utilizador após terminar de correr o programa.

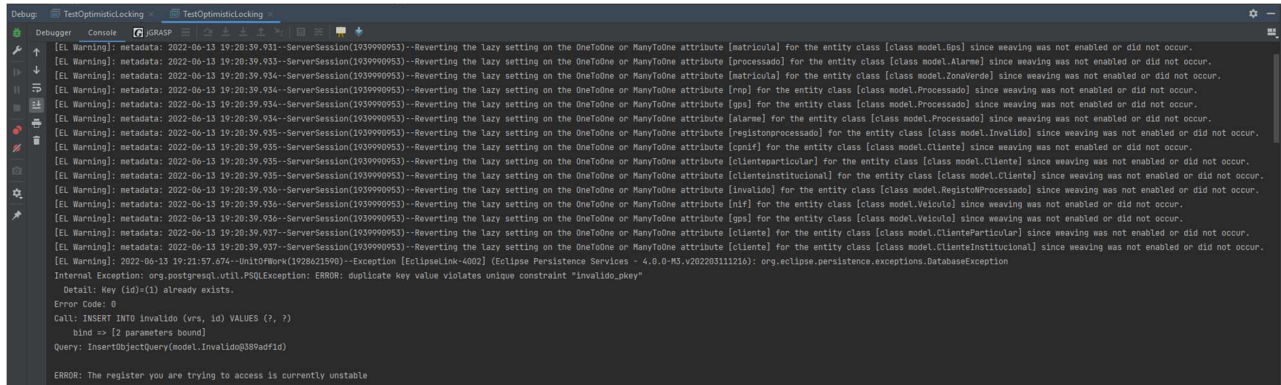


Figura 6 – Consola do segundo utilizador após o termino da execução da transação iniciada pelo primeiro utilizador.

O erro ocorre na consola do utilizador que acede por último à base de dados, neste caso, o segundo utilizador, devido ao acesso concorrente a função. O primeiro utilizador a fazer este acesso é o que consegue completar a transação como esperado. Porém o segundo utilizador já não consegue até a transação iniciada pelo primeiro utilizador ser terminada.

Quando o segundo utilizador, por fim conseguir correr a sua transação, receberá uma mensagem de erro, pois irá trabalhar com uma base de dados já atualizada, tal como vê-se na *Figura 6*. Ou seja, os dados sob os quais inicialmente o segundo utilizador começou a trabalhar acabam por violar a restrição de identidade por já terem sido adicionados a base de dados pelo primeiro utilizador. E desta forma verificamos a funcionalidade pedida.

4. Conclusões

Esta fase do trabalho resultou na necessidade de se ir buscar certos temas assimilados na disciplina *Introdução aos Sistemas De Informação*, tal como na consolidação dos recém-adquiridos conceitos da disciplina de *Sistemas de Informação*.

Outro facto distinto deste trabalho foi a familiarização com plataformas como a *API JPA* e o aprofundamento da compreensão da camada de acesso de dados. As controvérsias incorporadas neste trabalho possibilitaram a aquisição de novos conhecimentos, nomeadamente, a utilização de *DataMappers*, *Repository* e novos tipos de *Locks*.

Igualmente como no trabalho passado, podemos aprofundar os nossos conhecimentos sobre a cadeira *Sistemas de Informação* neste trabalho, pelo que foi muito útil para nós, pois muito do que não conseguimos entender nas aulas, os conseguimos durante a realização deste trabalho bem como nos episódios de esclarecimento de dúvidas.