Departamento de Informática
Plano de Trabalho Prático
Sistema de Gestão de Base de Dados 2020/21
Mestrado em Engenharia Informática

Controlo de Transações em Base de Dados

Professor Doutor João Manuel da Silva Fernandes Muranho

David Alexandre^{M11137}
Douglas Amante^{M10624}
Thiago Machado^{M10733}

[1]Departamento de Informática
Universidade Beira Interior - UBI
R. Marquês de Ávila e Bolama, Covilhã 6201-001, Portugal
+351 275 319 700
31 de outubro de 2020

Conteúdo

1 Sumário **5**

2 Introdução **6**

3 Descrição do Trabalho Prático **7**

3.1 Base De Dados 7

3.2 Ferramentas Utilizadas 7

3.2.1 Netbeans 8

3.2.2 SQL Server Management Studio 8

3.3 Conexão 8

4 Aplicações **8**

4.1 Login 9

4.2 Browser 9

4.3 Edit 10

4.4 Log 11

4.5 Log Time 12

5 Transações **13**

5.1 Controlo 13

5.2 Gestão 14

5.3 Isolamento 14

5.4 Recuperação 15

6 Concorrência **16**

6.1 *Block* 16

6.2 *DeadLock* 17

6.3 *Lock* 17

7 Diagramas **18**

7.1 Diagrama de Caso de Uso 18

7.2 Diagrama de Pacotes 19

7.3 Diagrama de Classe 19

8 Testes e Análises de Resultados **21**

8.1 *Browser* 21

8.2 *Edit* 21

9 Conclusão **22**

Lista de Figuras

1 Conectividade entre os arquivos SQL. 7
2 Arquitetura Cliente-Servidor. 8
3 Divisão das aplicações. 9
4 Interface *Login*. 9
5 Aplicação *Browser*. 10
6 Aplicação parte 1 *Edit*. 10
7 Aplicação parte 2 *Edit*. 11
8 Aplicação *Log Operation*. 12
9 Aplicação *Log Time*. 12
10 Tipos de interferências que afetam uma transação. 14
11 Situações de parada para o processamento de uma transação. 14
12 Sequência de tratamento sobre as falhas para a recuperação da base de dados. 16
13 Ocorrência de um *Block*. 16
14 Ocorrência de um *deadlock*. 17
15 Ocorrência de um *Lock*. 18
16 Diagrama de Caso de Uso. 19
17 Diagrama de Pacotes da aplicação. 19
18 Diagrama de Classes parte 1 referente à aplicação. 20
19 Diagrama de Classes parte 2 referente à aplicação. 20

Lista de Tabelas

1 Propriedades do Conjunto de Transações. 13
2 Níveis de Isolamento. 15
3 Variáveis Globais 21
4 Acontecimentos da Aplicação *Browser* 21

1. Sumário

Este documento reporta os resultados relacionados a análise dos dados das encomendas do cenário das aplicações *Browser*, *Edit*, *Log* e *LogTime* de acordo com os modelos e *scripts* desenvolvidos pelo docente, João Manuel da Silva Fernandes Muranho e as transações desenvolvidas. O objetivo é conhecer e entender as transações, considerando as metodologias da própria, tais como: gestão, controlo, níveis de isolamento e recuperação, tipos de concorrência, como: *block*, *locks* e *deadlocks*. O indicador físico dos testes e análise de resultados está sujeito ao nível de isolamento das aplicações, detalhando o resultado das transações observadas de acordo com os testes. Este relatório é dividido em dez Seções. As Seções 1 e 2 descrevem o Sumário e sua versão em inglês. A seção 3 apresenta uma introdução do desenvolvimento de sistemas de informação operando com uma base de dados propicia à várias aplicações acendendo aos dados partilhados. A seção 4 aborda a descrição do trabalho prático. A seção 5 apresenta as informações contextuais referentes às aplicações. A seção 6 descreve as transações e suas particularidades. A seção 7 exibe a metodologia de concorrência dos dados. A seção 8 expõe três diagramas *Unified Modeling Language - UML*, sendo eles caso de uso, pacotes e classe. A seção 9 evidência os testes e análises de resultados das transações sobre as aplicações, e por fim, a seção 10 aborda as considerações finais sobre o trabalho prático realizado.

Resumo. *Este documento manifesta os resultados relacionados ao estudo dos dados das encomendas do cenário das seguintes aplicações Browser, Edit, Log e LogTime de acordo com os modelos propostos. O objetivo do trabalho é compreender as transações, levando em conta as organizações da própria, tais como: gestão, controlo, níveis de isolamento e recuperação, tipos de concorrência, como: block, locks e deadlocks.*

Palavras-Chave — Base de Dados, Consistência, Impasse, Níveis de Isolamento, Sistema de Gestão de Base de Dados, Trincos, Transações.

2. Introdução

Os dados estão com um crescimento exponencial nas últimas décadas. Por conta deste crescimento ser exponencial é necessário ter uma estrutura muito bem definida para desempenhar uma eficiência nos dados. Estas estruturas são gerenciadas por um sistema de gestão de base dados, no qual são programas que permitem criar e manipular as base de dados, em que dados estão estruturados com independência relativamente aos programas da aplicação que os manipulam. Assim, o seu objetivo é registar e manter a informação que for considerada necessária ao sistema, disponibilizando-a automaticamente para os mais diversos fins [1].

No contexto de um sistema de gerenciamento de base dados é necessário ter consciência sobre uma unidade lógica, às vezes composta de várias operações, denominada transação [2]. A transação no ambiente de base de dados possui dois propósitos principais, sendo: fornecer unidades de trabalho confiáveis que permitam a recuperação correta de falhas, mantendo uma consistência e fornecer isolamento entre programas que acessam uma base de dados simultaneamente [3].

Qualquer base de dados que seja utilizada por mais de um usuário, terá que administrar o controle de concorrência entre as informações que estão sendo acessadas pelos usuários. Controle de concorrência é quando, em um banco de dados, usuários distintos tentam acessar a mesma informação e então é feito um controle entre essas transações. E para a solução deste problema existem diversas técnicas de controle de concorrência que são utilizadas como forma de assegurar a propriedade de não interferência entre uma operação e outra, ou o isolamento das transações executadas ao mesmo tempo [4].

3. Descrição do Trabalho Prático

O trabalho prático foi desenvolvido por partes. Primeiramente, foi feito uma revisão profunda na literatura sobre as metodologias da base de dados e o caminho da entrega final, compartilhado pelo docente. Com isso, foi realizado uma divisão do sistema como um todo. No qual, utilizou uma linguagem alto nível para a iteração com os usuários, uma base de dados sql para armazenamento e gerenciamento dos dados, e, consequentemente, a conexão entre ambas. Estes tópicos estão descritos nas subseções 4.2.1, 4.2.2 e 4.3, respectivamente.

3.1. Base De Dados

A base de dados é um conjunto de informações que dizem a respeito de um determinado domínio com a função principal de facilitar a gestão das informações. No qual, estas bases podem ser estruturadas ou não-estruturadas. Dados estruturados são aqueles que possuem relações entre si e possuem mesmos atributos para registros diferentes, como por exemplo, cadastros de *websites*, onde existem campos a serem preenchidos. Já os dados não-estruturados não possuem uma organização prévia, o que significa que os atributos não são devidamente organizados, como por exemplo, áudios, arquivos de texto [5].

No caso deste trabalho prático, a base de dados foi constituída por alguns arquivos que se relacionam de alguma forma com o intuito de gerenciar as encomendas e consequentemente os produtos feito pelos clientes, como demonstrado na Figura 1.

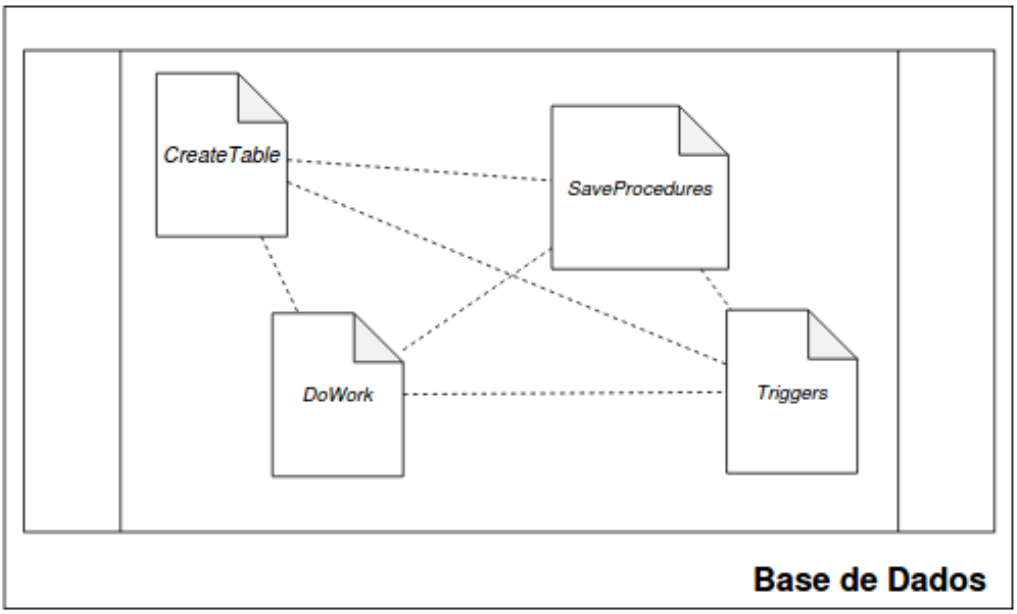

Figura 1. Conectividade entre os arquivos SQL.

3.2. Ferramentas Utilizadas

Para o desenvolvimento da aplicação foi necessário recorrer a um conjunto de tecnologias e ferramentas:

3.2.1. Netbeans

As aplicações foram implementadas na linguagem de programação *Java*, por meio do ambiente de desenvolvimento integrado *Netbeans*, que serve de sustentação a todo o código da aplicação. A escolha é devido a convergência entre ambos e também pela sua praticidade na integração de base de dados com *Structure Query Language -SQL Server*.

3.2.2. SQL Server Management Studio

Pelo fato de configurar e administrar as ligações entre as tabelas da base de dados criada pelo docente, utilizou-se o *SQL Server Management Studio - SSMS*, aplicação da *Microsoft*. Esta ferramenta possui editores de *scripts* e ferramentas gráficas que permitiram a edição e a análise dos componentes da base de dados. Esta ferramenta é também muito importante na otimização do desempenho da base de dados.

3.3. Conexão

Foi utilizado a arquitetura cliente-servidor, ilustrada na Figura 2. No qual, o cliente solicita recursos e o servidor processa os pedidos e disponibiliza os recursos. Na manutenção desta arquitetura é necessário identificar as funções do cliente e as funções do servidor. Ao nível das bases de dados relacionais, o sistema de gerenciamento de base de dados encarrega-se da gestão das transações, do controlo de acesso, da recuperação em caso de falhas.

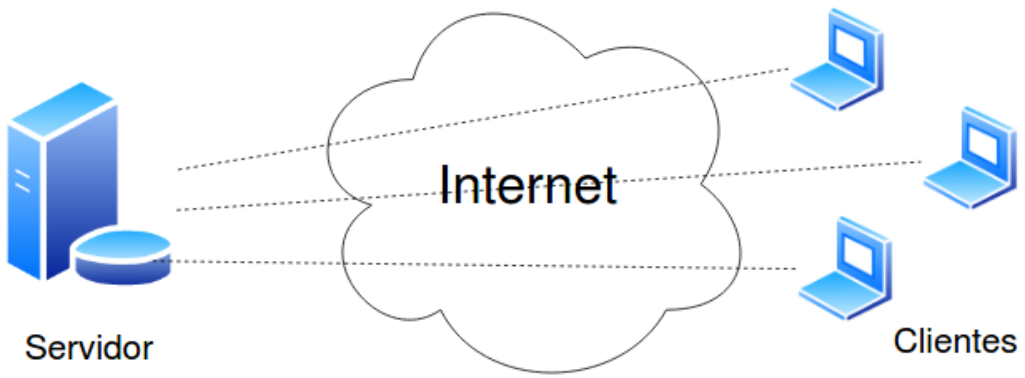

Figura 2. Arquitetura Cliente-Servidor.

4. Aplicações

A arquitetura da aplicação referente ao trabalho prático é divida em quatro aplicações distintas, como ilustrado na Figura 3.

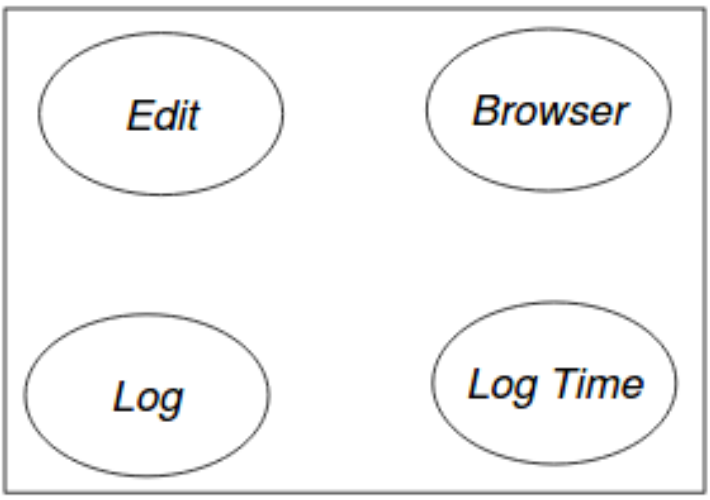

Figura 3. Divisão das aplicações.

4.1. Login

O *Login* é realizado com base em quatro campos. Todos os campos são convergentes à conexão com a base de dados. A Figura 4 indica a primeira iteração entre o cliente e o servidor.

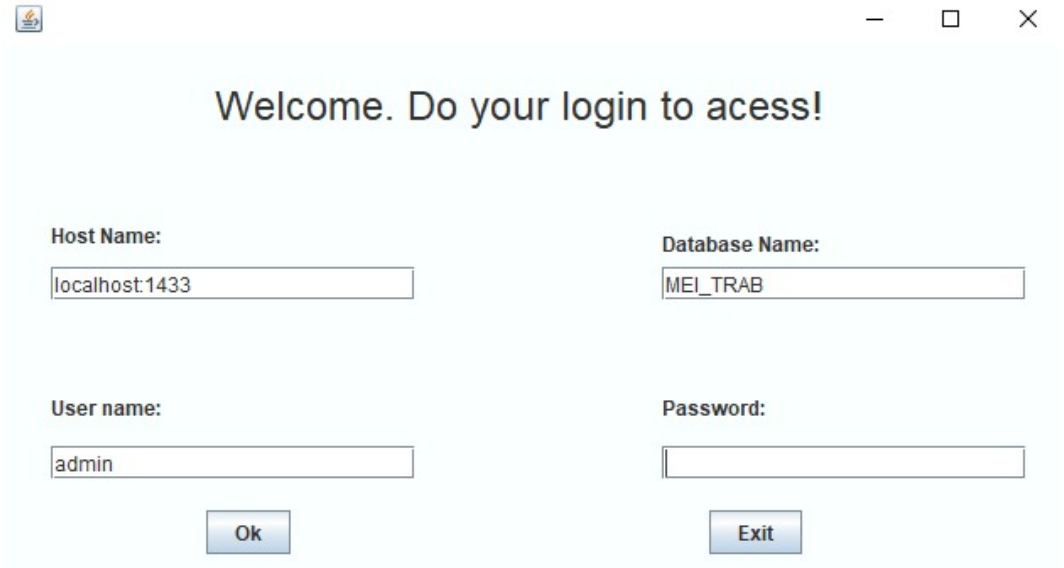

Figura 4. Interface *Login*.

4.2. Browser

No módulo *Browser* pretende-se visualizar as encomendas e os seus produtos. Esta interface possui duas grelhas, uma com as encomendas e a outra com os produtos da encomenda selecionada. A Figura 5 mostra a interface *Browser*.

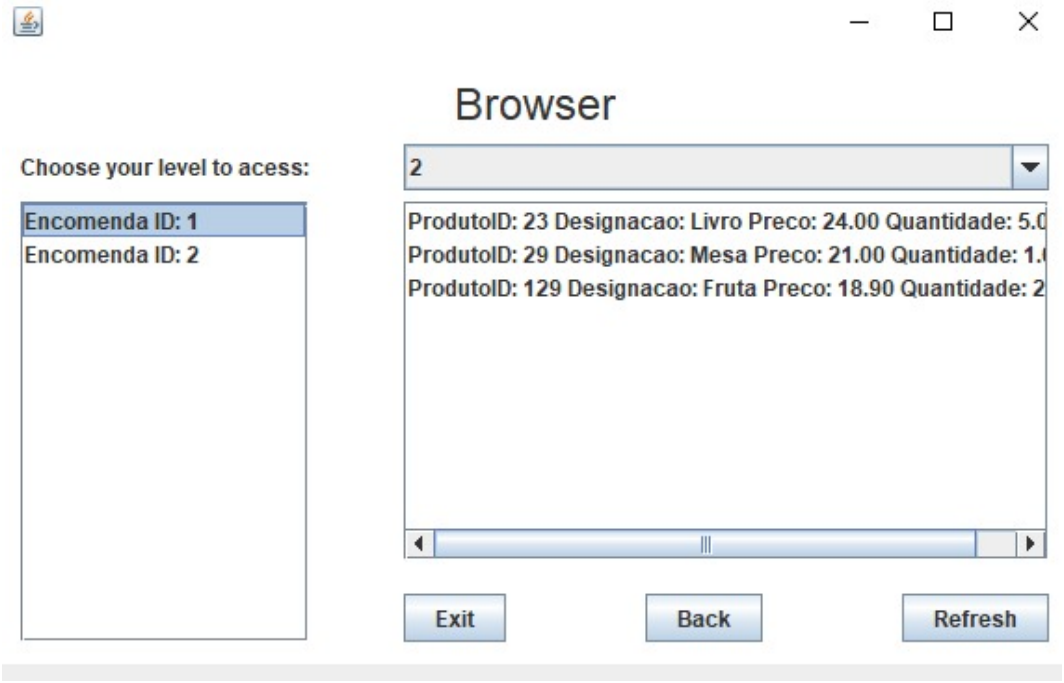

Figura 5. Aplicação *Browser*.

4.3. Edit

A aplicação *Edit* possui a finalidade de proporcionar ao usuário um campo para que ele possa indicar o *ID* da encomenda com o objetivo de editar/alterar. A interface mostra os dados da encomenda, incluindo as linhas, e permitindo ao utilizador alterá-los. Com a afirmação que o usuário não irá inserir novos produtos na encomenda nem eliminar os produtos já encomendados. As Figuras 6 e 7 apresentam as respectivas interfaces da aplicação *Edit*.

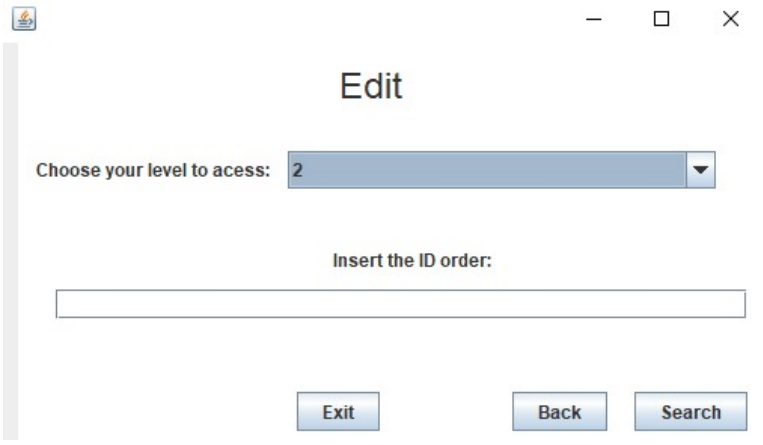

Figura 6. Aplicação parte 1 *Edit*.

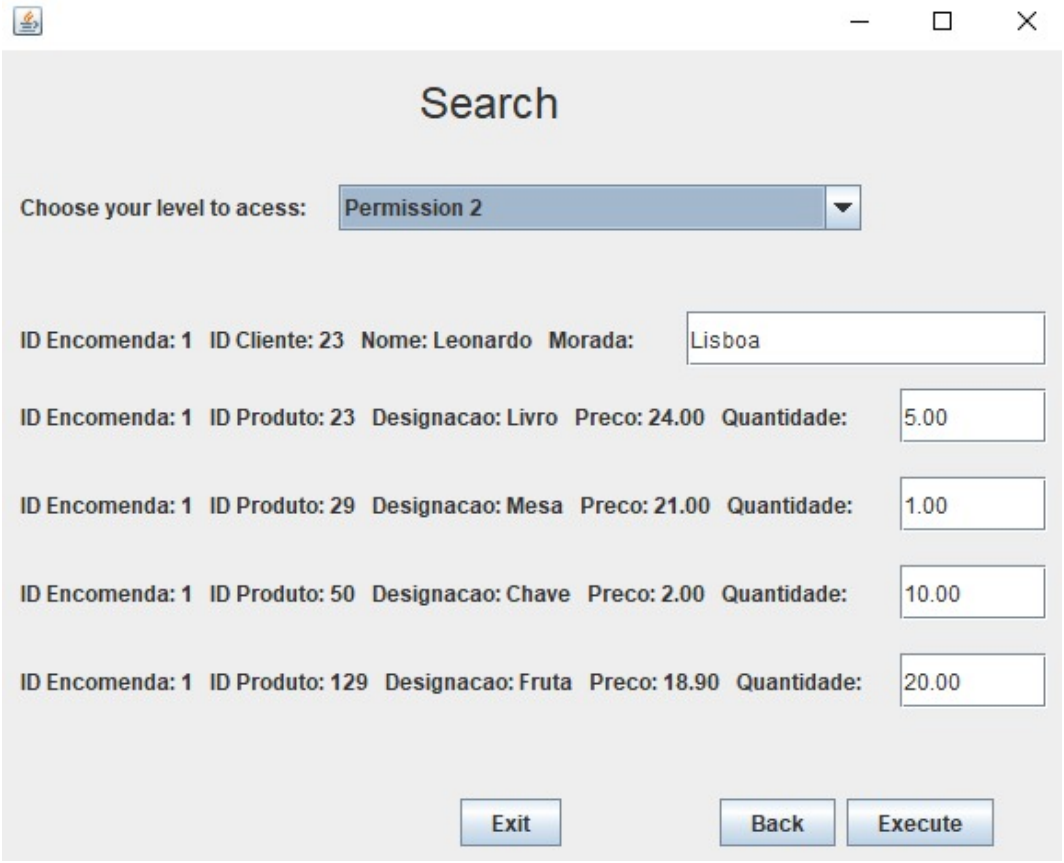

Figura 7. Aplicação parte 2 *Edit*.

4.4. Log

A interface do *Log* contém uma grelha onde é possível visualizar todos as linhas mais recentes referente à tabela *Log Operations*. Porém, os tipos de eventos conhecidos são apenas: **I**, **U** e **D**. A Figura 8 aponta a interface *Log Operation*.

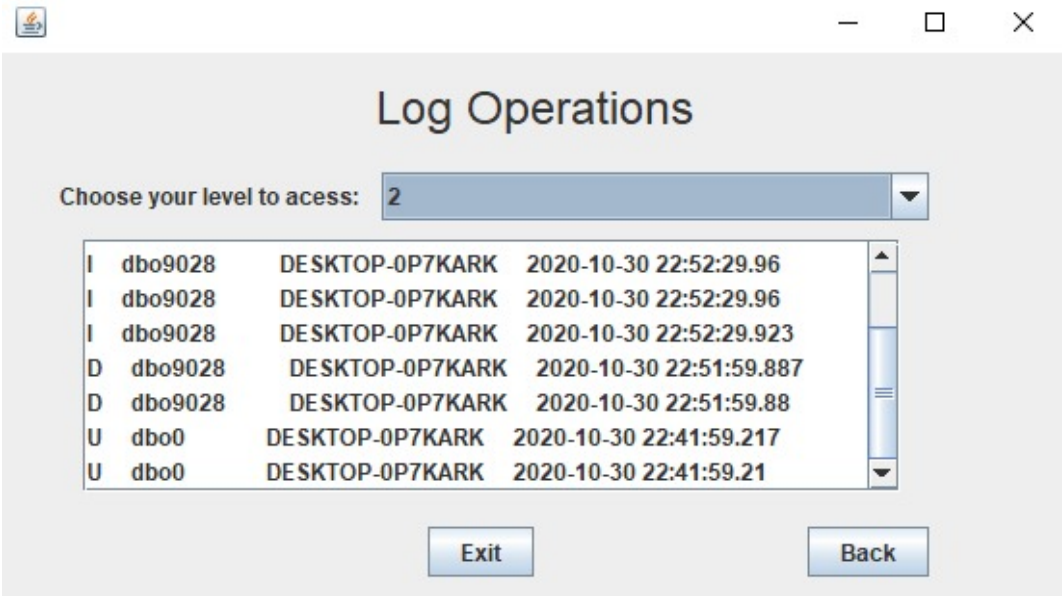

Figura 8. Aplicação *Log Operation*.

4.5. Log Time

Por fim, a aplicação referente ao *Log Time* possui a utilidade de mostrar os registros do tipo:

$$\textit{EventType} = \textbf{'O'} =' Outro'$$

Cada linha da grelha contém as seguintes colunas: *UserId, EncId* e Tempo. A Figura 9 exibe a interface *Log Time Operations*.

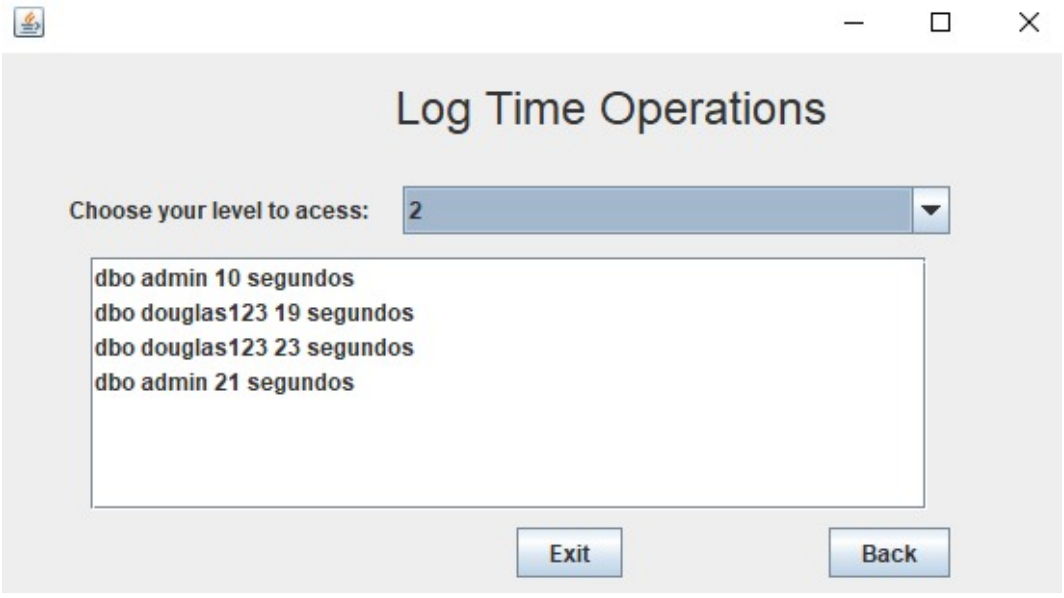

Figura 9. Aplicação *Log Time*.

5. Transações

Uma transação é a sequência de operações tratada como um bloco único e indivisível, denominado na metodologia como atômico. No qual esta sequência de operações atua durante a recuperação de falhas e que isolam entre acessos simultâneos na mesma massa de dados. Uma transação é executada na sua totalidade ou não é executada [6].

Uma transação é uma única unidade, suas ações não podem ser mescladas com outras operações da base de dados que não participam da transação. Mesmo um único comando SQL envolve vários acessos da base de dados separados, e uma transação pode consistir em vários comandos SQL.

Para garantir que o sistema de base de dados não perca uma transação concluída com êxito de uma falha posterior, as ações de uma transação devem persistir entre falhas. Por fim, uma transação deve preservar a consistência da base de dados.

A Tabela 1 representa as quatro particularidades das transação.

Tabela 1. Propriedades do Conjunto de Transações.

Propriedade	**Definição**
Atomicidade	Caso a transação falhar devido a qualquer motivo, seja qual for a alteração na base de dados que a transação possa ter feito anteriormente é desfeita.
Consistência	Uma transação deve preservar a consistência da base de dados, ou seja, uma transação deve respeitar as regras de integridade dos dados .
Durabilidade	Os efeitos de uma transação em caso de sucesso devem persistir na base de dados mesmo em casos de quedas de energia, travamentos ou erros.
Isolamento	A transação deve funcionar sem interferência de comandos da base de dados sendo executados simultaneamente.

5.1. Controlo

O processo de coordenação da execução simultânea, e sem interferências, das operações das transações, é conhecido como controlo de concorrência. O principal objetivo do controlo de concorrência é garantir a sequência das transações, pois o acesso concorrente a dados partilhados podem dar origem a diversos problemas de integridade e consistência.O acesso simultâneo é benéfico quando envolve apenas leitura de dados. Os problemas de consistência podem ocorrer quando há operações de escrita envolvidas.

Uma transação, ainda que correta, se executada individualmente, pode ser afetada por três tipos de interferências, apresentadas na Figura 10.

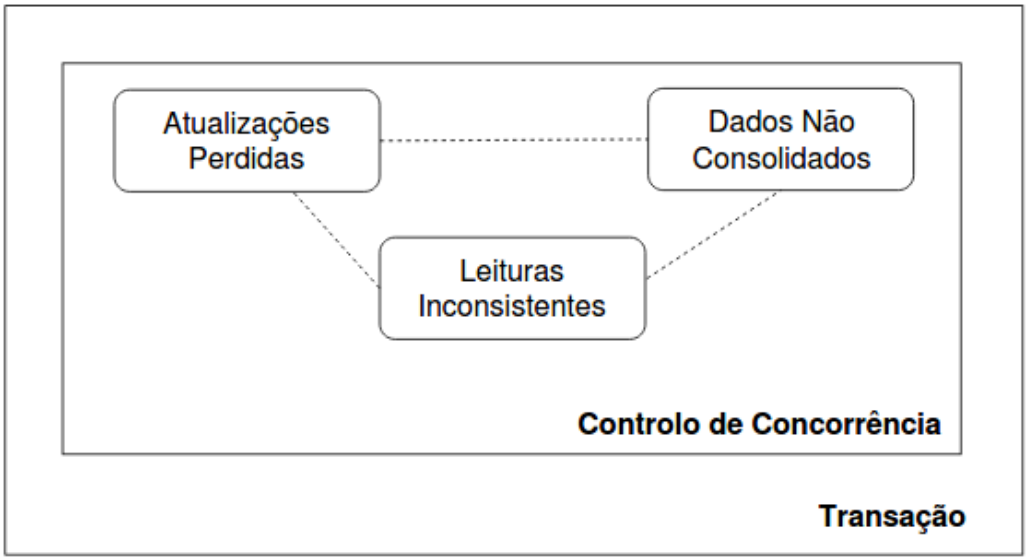

Figura 10. Tipos de interferências que afetam uma transação.

5.2. Gestão

Segundo o *American National Standards Institute - ANSI*, a finalização de uma transação é concretizado com recurso a um dos comandos `COMMIT` (transação confirmada) ou `ROLLBACK` (transação abortada). O processamento de uma transação faz-se de modo sequencial, operação a operação, até que ocorra uma das seguintes situações apresentadas na Figura 11.

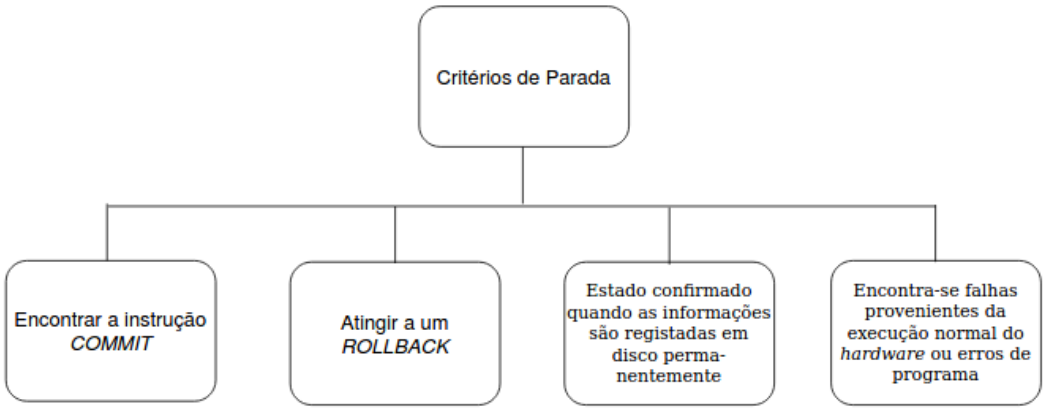

Figura 11. Situações de parada para o processamento de uma transação.

5.3. Isolamento

O desenvolvimento de aplicações sobre bases de dados implica o conhecimento dos níveis de isolamento das transações. O nível de isolamento define em que grau uma transação está exposta à modificações feitas por outras transações concorrentes. Os sistemas de gerenciamento de base de dados usam níveis de isolamento e trincos para controlar

o grau de concorrência [7], consequentemente, fazendo que o programador decida qual é o nível que mais se ajusta à sua aplicação [8].

A maior parte das aplicações desenvolvidas não utiliza o nível de isolamento mais restrito, permitindo assim uma concorrência mais elevada [9]. Quanto mais restritivo for o nível de isolamento utilizado, menor será a concorrência efetiva e maior será a probabilidade de o sistema entrar em *deadlock*. Segundo Canolly, um *deadlock* , constitui um impasse em que duas ou mais transações esperam indefinidamente por recursos bloqueados por outras transações [10].

As especificações do SQL definem quatro níveis de isolamento,, sendo eles: Leitura Não Comprometida, Leitura Comprometida, Leitura Repetível e Serializáveis [11], apresentado na Tabela 2. A escolha do nível de isolamento depende do uso previsto para o mesmo. Um nível de isolamento mais baixo, está mais sujeito a efeitos colaterais e, portanto, com mais leituras sujas e mais atualizações perdidas, mas permite mais concorrência. Um nível de isolamento mais elevado reduz concorrência, requer mais recursos do sistema e aumenta as hipóteses de *deadlock* [12].

Tabela 2. Níveis de Isolamento.

Nível	**Definição**
Read Uncommited	Constitui o menor nível da hierarquia. Neste nível, uma transação T_x pode ler alterações feitas por transações que ainda estão a decorrer. Naturalmente, os dados lidos podem ainda sofrer outras alterações enquanto T_x está em progresso.
Read Commited	Este nível assegura que uma transação T_x só pode ler dados confirmados. Para além disso assegura que nenhum valor escrito por T_x é alterado por outra transação até T_x terminar. Note-se que um valor lido por T_x pode ser modificado por outra transação durante o período de vida de T_x.
Repetable Read	Há semelhança do nível leitura comprometida, este nível assegura que a transação T_x só faz leituras de dados confirmados. Para além disso assegura que nenhum valor lido ou escrito por T_x é alterado por outra transação até T_x terminar.
Serializable	Neste nível, todas as transações são isoladas e executadas de modo sequencial. As transações não podem ler dados modificados ainda não confirmados por outras transações.

5.4. Recuperação

No ambiente de recuperação, o objetivo é restaurar a base de dados ao seu último estado consistente antes de uma falha considerável. Porém, esta recuperação deve levar em consideração as propriedades das transações, detalhadas na Tabela 1.

A recuperação é realizada com base nos dois tipos de falhas referentes à base dados, sem danos físicos ou com danos físicos. No qual, a recuperação é feita atráves da metodologia apresentada na Figura 12.

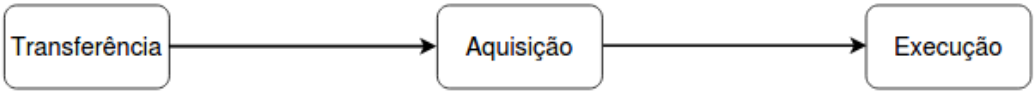

Figura 12. Sequência de tratamento sobre as falhas para a recuperação da base de dados.

6. Concorrência

No contexto de escalabilidade, com o decorrer do tempo a base de dados poderá sofrer com múltiplos acessos simultâneos. No qual, a base de dados terá que ser capaz de suportar os acessos simultâneos. Com isso, é apresentados nas subseções 6.1, 6.2 e 6.3 algumas metodologias sobre estes problemas.

6.1. *Block*

Os *blocks* ocorrem justamente quando um processo tenta acesso a um fragmento de dados. Porém, não pode pois já há um outro processo em execução. O $Processo_B$ fica então, bloqueado até que o $Processo_A$ encerre a atividade, como é demonstrado na Figura 13.

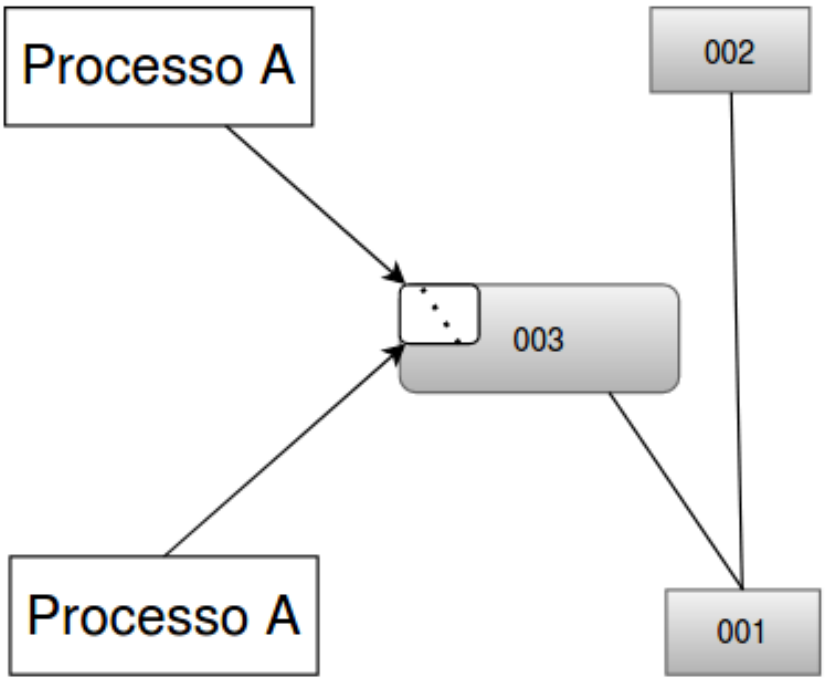

Figura 13. Ocorrência de um *Block*.

6.2. *DeadLock*

Os *deadlocks* ocorrem quando há mais de dois processos bloqueados. Quando o $Processo_A$ está bloqueado por um *lock* referente ao $Processo_B$, enquanto o $Processo_B$ está bloqueado de acessar um *lock* do $Processo_A$, ou seja, existe uma dependência cruzada entre eles que nunca será resolvida, como é apresentado na Figura 14. Em uma situação de *deadlocks*, os processos estão bloqueando um ao outro, por esse motivo, há a necessidade de uma ação externa para resolver esse *deadlock*.

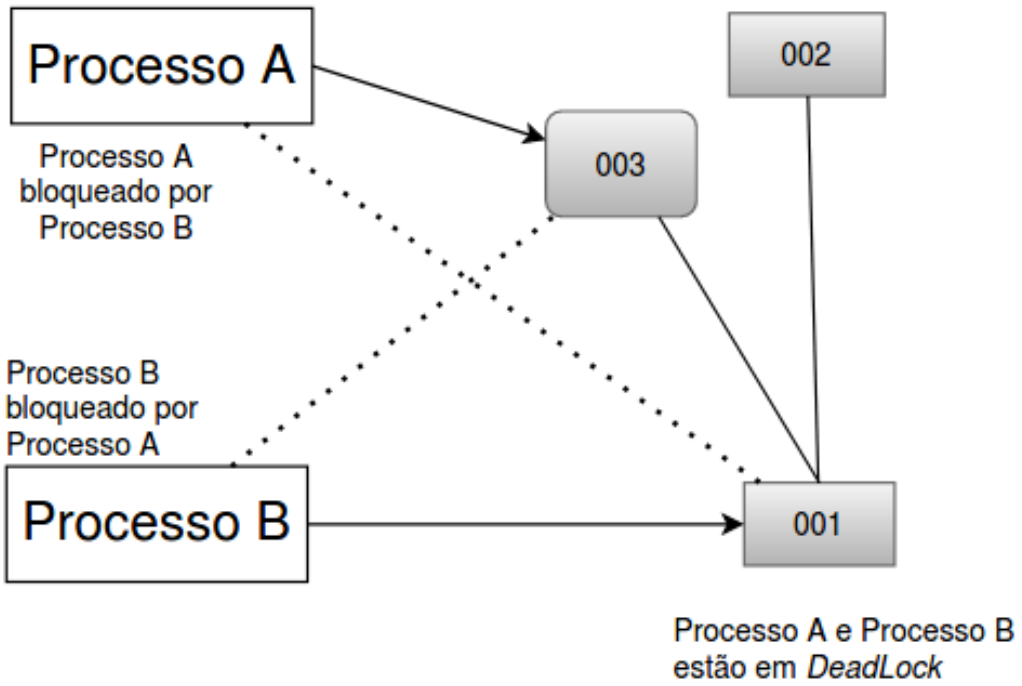

Figura 14. Ocorrência de um *deadlock*.

6.3. *Lock*

Os *locks* ocorrem quando um processo acessa parte dos dados onde existe a possibilidade de que outro processo concorrente tenha acesso aos mesmos dados. Com isso, o *lock* age de forma a não permitir que nenhuma informação relacionada à fila em execução seja alterada.

Neste cenário, o $Processo_A$, está com um *lock* ao acessar o registro 003, como demonstrado na Figura 15.

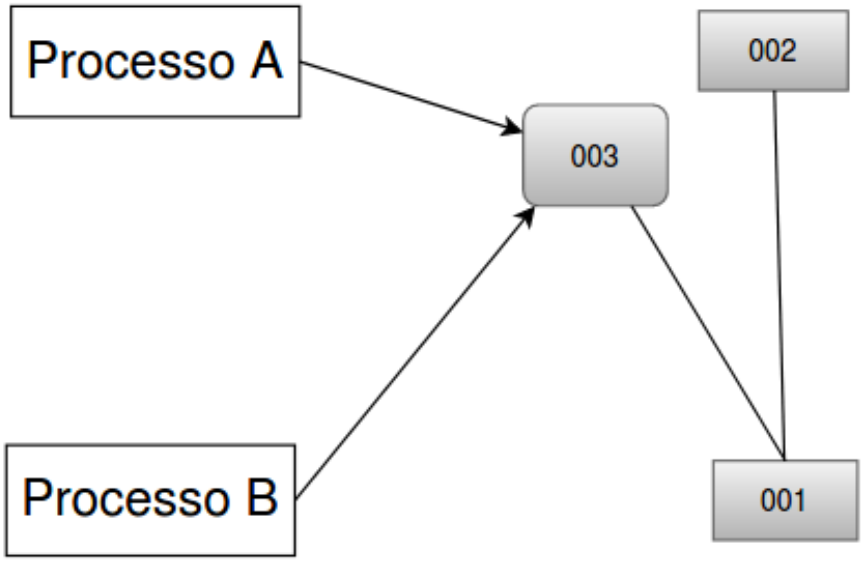

Figura 15. Ocorrência de um *Lock*.

7. Diagramas

Os diagramas possuem uma visão abstrata sobre a aplicação. No contexto do uso de *UML* na produção de *software* se aplica: 20% dos recursos da *UML* atendem em 80% dos cenários que precisamos especificar. Os diagramas descritos nas subseções 7.1, 7.2 e 7.3 deste trabalho são derivados da literatura *UML* e possuem o intuito de compartilhar a metodologia codificada de uma maneira eficaz para leigos sobre a codificação de *software* [13]. Foram introduzidos os conceitos de diagramas de caso de uso, pacotes e classes.

7.1. Diagrama de Caso de Uso

Com uma maneira simplificada da iteração entre usuário e sistema, este é o objetivo do diagrama de caso de uso. Além, de mostra o relacionamento entre os diferentes casos de uso envolvidos no sistema. Na Figura 16 é apresentado o caso de uso da aplicação.

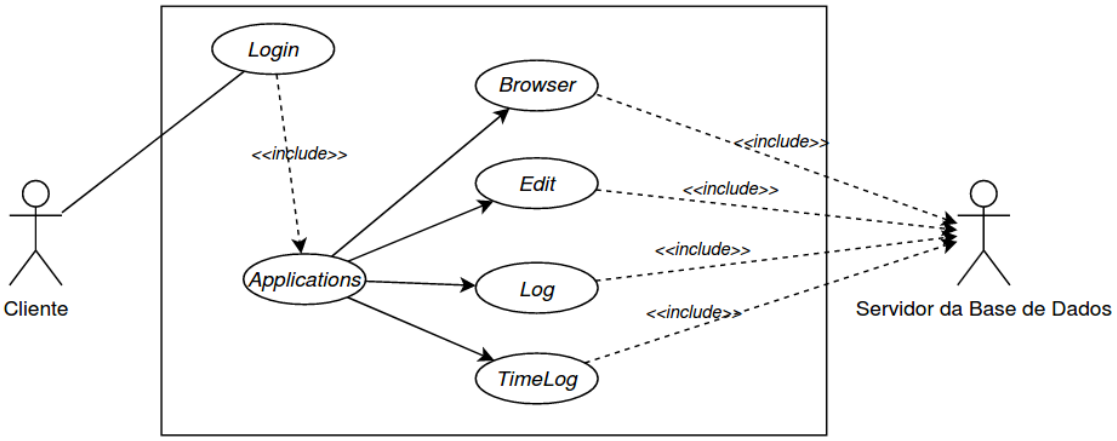

Figura 16. Diagrama de Caso de Uso.

7.2. Diagrama de Pacotes

A representação do diagrama de pacotes é apresentar o relacionamento entre todos os pacotes do sistema, juntamente com um relacionamento mais alto nível das classes envolvidas nas coleções dos pacotes [14]. Com isso, a Figura 17 apresenta o diagrama de pacotes do sistema.

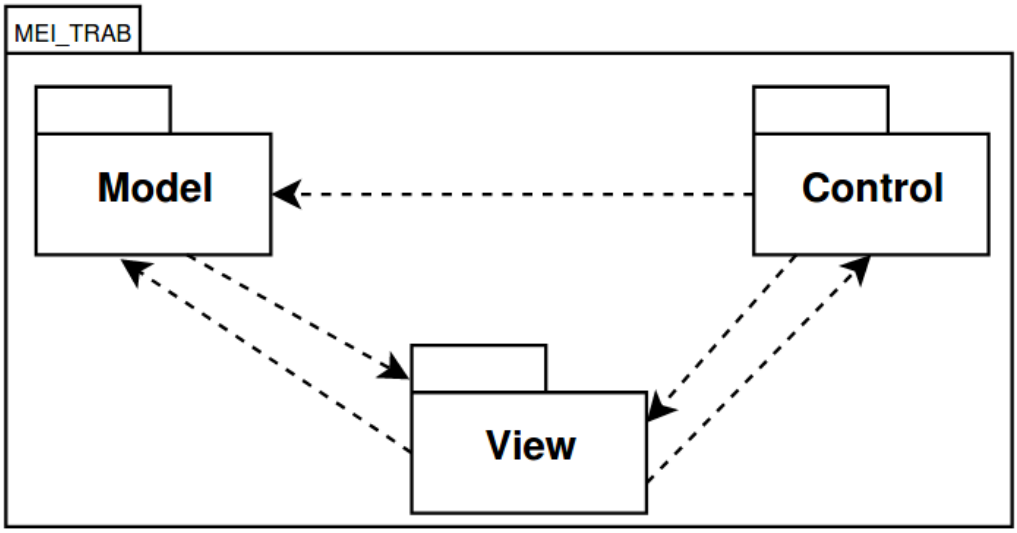

Figura 17. Diagrama de Pacotes da aplicação.

7.3. Diagrama de Classe

O diagrama de classe possui uma estrutura que descreve a modelagem conceitual dos objetos referentes às classes do sistema. No qual, introduz a ideia sobre atributos, métodos e os relacionamentos dos objetos [15].

Contudo, a apresentação do diagrama de classe do sistema foi divido em duas partes. No qual, a divisão em dois subsistemas se deve ao relacionamento `0...0 para 1...*` à classe `Users`, apresentada na Figura 18. Entretanto, a classe `Auth` se relaciona diretamente com as interfaces `Application` e `Login`, mostrado na Figura 19.

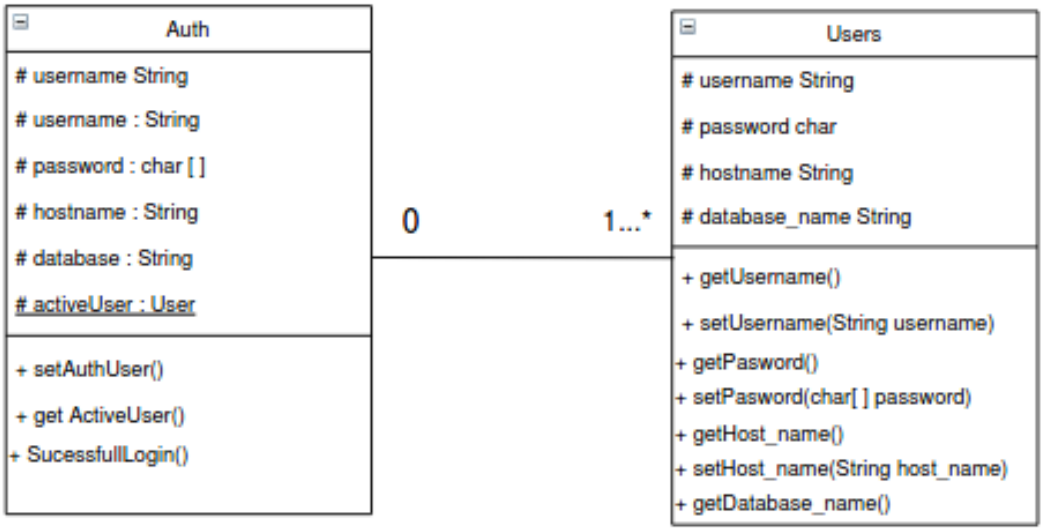

Figura 18. Diagrama de Classes parte 1 referente à aplicação.

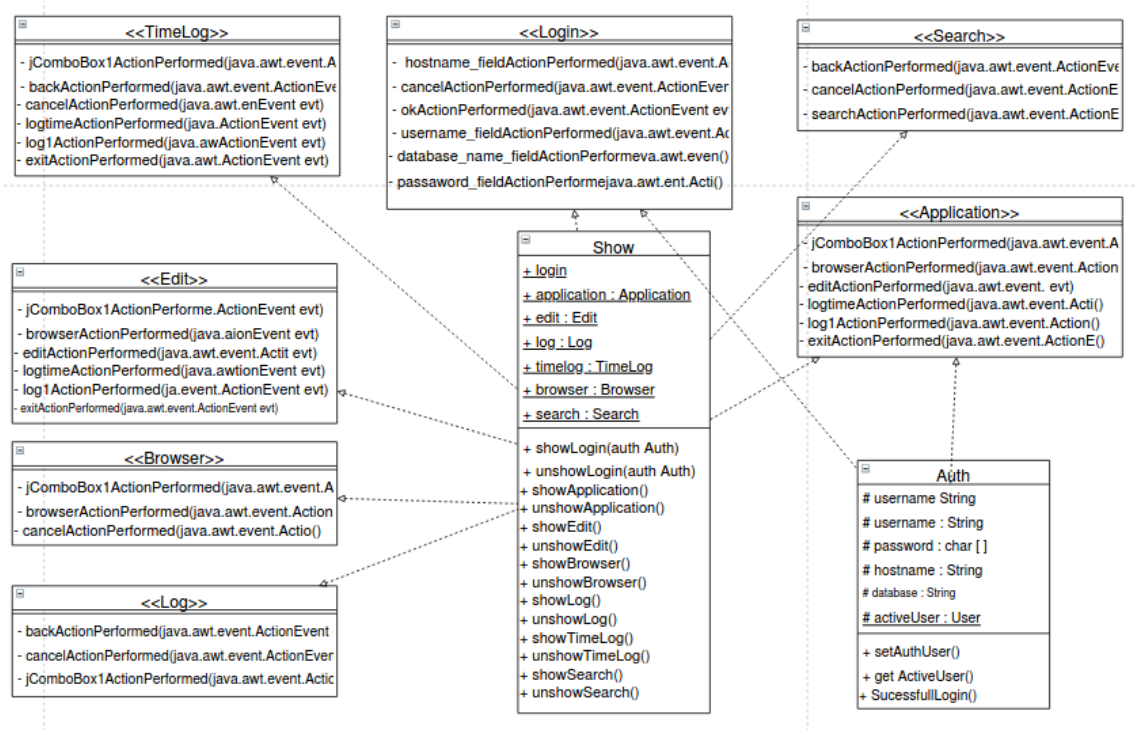

Figura 19. Diagrama de Classes parte 2 referente à aplicação.

8. Testes e Análises de Resultados

A programação das aplicações demonstradas nesta seção foi auxiliada pela criação de uma classe de memória de alguns elementos de passagem entre janelas. A enumeração desses elementos pode ser verificada na Tabela 3:

Tabela 3. Variáveis Globais

Nome da Variável	Designação
AppConnection	Guarda a conexão à base de dados
username	Guarda o Nome de Utilizador
encomendaID	Guarda o Número de Encomenda
connectionLevel	Guarda o Nível de Transação
referenceEdit	Guarda a referência a guardar na base de dados

Os testes aplicados ao *software* criado, consistem na variação quer dos tempos de *refresh*, quer dos níveis de isolamento aplicados. Tarefas como a edição concorrente de itens da base de dados, assim como a verificação de informação desses mesmos itens foram aplicadas, mais uma vez com a variação dos níveis de transação aplicados.

8.1. *Browser*

O estabelecimento dos níveis de transação na aplicação *Browser*, foi bem sucedido sendo sempre aplicado à transação em questão.

Quaisquer leituras de dados cruciais à elaboração das duas listas geradas em auxílio do utilizador, pareceram de alterações conforme o nível de transação atual à data da execução. A Tabela 4 apresenta os acontecimentos na aplicação *Browser*. No qual, a Tabela 2 serviu como base.

Tabela 4. Acontecimentos da Aplicação *Browser*

Nível de Transação	Acontecimento
Read Uncommitted	Leitura dos dados desatualizada
Read Commited	Re-leitura dos dados bem sucedida
Repetable Read	Leitura de tuplos repetidos
Serializable	Leitura correta e lenta dos dados

8.2. *Edit*

A aplicação *Edit* demonstrou-se bem sucedida na leitura dos dados necessários à edição da interface. Quer a edição de qualquer encomenda assim como o sucesso no comando *save* ou descarte das alterações executadas foi também bem sucedido.

Contudo, a realização das ações anteriores em concorrência, mencionados na seção 6 retornaram resultados variados conforme cada nível de transação escolhido, descrito na Tabela 3.

9. Conclusão

A partir do trabalho prático desenvolvido conclui-se que a metodologia sobre propriedades, apresentado na Tabela 1, são de extrema importância para o gerenciamento da base de dados, no qual as propriedades atomicidade, consistência, durabilidade e isolamento possuem o objetivo de garantir a validade dos dados, pelo fato de haver a possibilidade de erros no armazenamento e mesmo assim as quatro propriedades convergirem para a facilitação de processamento de transações na base de dados.

Por outro lado, quanto maior a quantidade de execuções simultâneas maior a probabilidade de uma transação ser afetada pelas interferências, descritas na Figura 1, ou seja, essas interferências podem gerar problemas na integridade e na consistência na base de dados do sistema.

Os resultados obtidos nos testes das aplicações podem demonstrar a eficiência dos níveis de isolamento das transações. No qual, as transações não estão expostas à modificações de outras transações concorrentes de nível mais elevada, permitindo a menor probabilidade do sistema entrar em *deadlock*.

Para trabalhos futuros sugere-se criar uma imagem docker para a comunicação dos *containers* cliente-servidor. No qual a comunicação responsável pelo servidor seria gerenciada pela plataforma *Azure Microsoft*, armazenada na nuvem.

Referências

[1] M. Hardt and G. N. Rothblum, "A multiplicative weights mechanism for privacy-preserving data analysis," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 2010, pp. 61–70.

[2] J. Groff and P. Weinberg, *SQL The Complete Reference, 3rd Edition*, 3rd ed. USA: McGraw-Hill, Inc., 2009.

[3] R. Cattell, "Scalable sql and nosql data stores," *SIGMOD Record*, vol. 39, pp. 12–27, 12 2010.

[4] M. Martins and G. Müller, "Uma ferramenta case e-learning para transações e controle de concorrência em sgbdr," in *Anais do XXVII Workshop sobre Educação em Computação*. Porto Alegre, RS, Brasil: SBC, 2019, pp. 463–472. [Online]. Available: https://sol.sbc.org.br/index.php/wei/article/view/6651

[5] L. P. Monteiro, "Structured data and unstructured data," https://universidadedatecnologia.com.br/dados-estruturados-e-nao-estruturados/, [Acessado em 20-Outubro-2020].

[6] K. Ren, D. Li, and D. J. Abadi, "Slog: Serializable, low-latency, geo-replicated transactions," PVLDB 12(11), pp. 1747–1761, 2019.

[7] D. J. Abadi, "Introduction to transaction isolation levels," DBMS Musings. http://dbmsmusings.blogspot.com/2019/05/introduction-to-transaction-isolation.html, 2019.

[8] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A critique of ansi sql isolation levels," *CoRR*, vol. abs/cs/0701157, 01 2007.

[9] D. J. Abadi and M. Freels, "Serializability vs "strict"serializability: The dirty secret of database isolation levels," Fauna Corporate Blog. https://fauna.com/blog/serializability-vs-strict-serializability-the-dirty-secret-of-database-isolation-levels, 2019.

[10] T. Connolly and C. Begg, *Database Systems: practical approach to design, implementation, and management*, 6th ed. United Kingdom: Pearson Education Limited, 2015.

[11] D. Chamberlin, *SQL*. Boston, MA: Springer US, 2009, pp. 2753–2760. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1091

[12] A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha, "Making snapshot isolation serializable," *ACM Transactions on Database Systems*, vol. 30, no. 2, pp. 492–528, Jun. 2005.

[13] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, 2nd ed., ser. Object Technology Series. Boston, MA: Addison-Wesley, 2004. [Online]. Available: https://www.safaribooksonline.com/library/view/unified-modeling-language/0321245628/

[14] K. Hameed, I. S. Bajwa, and M. A. Naeem, "A novel approach for automatic generation of uml class diagrams from xmi," in *Emerging Trends and Applications in Information Communication Technologies*, B. S. Chowdhry, F. K. Shaikh, D. M. A. Hussain, and M. A. Uqaili, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 164–175.

[15] H. Herchi and W. B. Abdessalem, "From user requirements to uml class diagram," *CoRR*, vol. abs/1211.0713, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1211.html#abs-1211-0713