

Universidade da Beira Interior

Departamento de Informática



**Departamento de
Informática**

Controlo de Transações

Elaborado por:

Pedro Moreira, M11052

Gabriel Esteves, M11151

Nuno Salvado

Sistemas de Gestão de Bases de Dados

Professor:

Prof. Dr. João Muranho

30 de Outubro de 2020

Conteúdo

Conteúdo	1
Lista de Figuras	3
Lista de Tabelas	4
1 Introdução	7
1.1 Enquadramento	7
1.2 Motivação	7
1.3 Objetivos	7
1.4 Constituição do Grupo	8
1.5 Organização do Documento	8
2 Tecnologias e Ferramentas Utilizadas	10
2.1 Introdução	10
2.2 <i>Visual Studio Code</i>	10
2.3 Linguagem <i>Python</i>	11
2.4 <i>Qt</i>	12
2.4.1 <i>PyQt</i>	12
2.5 <i>Python Package Index</i>	13
2.5.1 <i>pyodbc</i>	13
2.6 <i>Structured Query Language (SQL)</i>	14
2.7 <i>Microsoft SQL Server</i>	15
2.8 <i>SQL Server Management Studio</i>	15
2.9 Conclusões	16
3 Gestão e Controlo de Transações	17
3.1 Introdução	17
3.2 Transações	17
3.3 Níveis de Isolamento	18
3.4 Recuperação (<i>Rollback</i>)	19
3.5 Tipos de Trincos (<i>Locks</i>) e de Impasse (<i>Deadlock</i>)	20

3.6	Conclusões	21
4	Aplicações	22
4.1	Introdução	22
4.2	Conexão à Base de Dados Remota	22
4.3	Menu da Aplicação	23
4.4	Aplicação <i>Edit</i>	24
4.5	Aplicação <i>Browser</i>	26
4.6	Aplicação <i>Log Tempo</i>	27
4.7	Aplicação <i>Log</i>	29
4.8	Conclusões	31
5	Testes e Análise de Resultados	32
5.1	Introdução	32
5.2	Testes	32
5.2.1	Alteração da morada da encomenda e da quantidade de um artigo	32
5.3	Conclusões	34
6	Conclusões e Trabalho Futuro	35
6.1	Conclusões Principais	35
6.2	Trabalho Futuro	36
	Bibliografia	37

Lista de Figuras

2.1	Logo do <i>VS Code</i>	10
2.2	Logótipo do <i>Python</i>	11
2.3	Logótipo do <i>PyQt</i>	13
2.4	Logo do <i>SQL Server</i>	15
2.5	Logo do <i>SSMS</i>	16
3.1	Representação de um <i>deadlock</i>	20
4.1	Janela de Conexão à Base de Dados.	23
4.2	Menu da Aplicação.	23
4.3	Aplicação <i>Edit</i>	24
4.4	Aplicação <i>Browser</i>	26
4.5	Aplicação <i>Log Tempo</i>	28
4.6	Aplicação <i>Log</i>	30
5.1	Aplicação <i>Browser</i> com as quantidades e a morada originais. . . .	33
5.2	Aplicação <i>Edit</i> da encomenda com o <i>ID 2</i> com a morada e a quantidade do produto 131 alterados.	33
5.3	Aplicação <i>Log</i> em que é possível ver que a encomenda com o <i>ID 2</i> foi editada.	34

Lista de Tabelas

2.1	Tabela de alguns módulos presentes no <i>PyQt</i>	13
3.1	Classificação dos Níveis de Isolamento.	19

Lista de Excertos de Código

2.1	Exemplo de uma conexão a uma Base de Dados (BD) [15].	14
4.1	Transação efetuada após o clique no botão "Aceitar" na Edição. . .	25
4.2	Transação efetuada para o preenchimento dos campos na aplicação <i>Browser</i>	26
4.3	Diferentes parâmetros de <i>DATEDIFF</i>	27
4.4	Excerto da função que faz <i>refresh</i> à aplicação <i>Log Tempo</i>	28
4.5	Excerto de código da função <i>Log</i>	30

Acrónimos

ACID	<i>Atomic, Consistent, Isolated, Durable</i>
ANSI	<i>American National Standards Institute</i>
BD	Base de Dados
GUI	<i>Graphics User Interface</i>
IDE	<i>IDE</i>
ISO	<i>International Organization for Standardization</i>
ODBC	<i>Open Database Connectivity</i>
PyPI	<i>Python Package Index</i>
RDBMS	<i>Relational Database Management Systems</i>
SEQUEL	<i>Structured English Query Language</i>
SGBD	Sistema de Gestão de Base de Dados
SQL	<i>Structured Query Language</i>
SQL Server	<i>Microsoft SQL Server</i>
SSMS	<i>SQL Server Management Studio</i>
UBI	Universidade da Beira Interior
UC	Unidade Curricular
VSCode	<i>Visual Studio Code</i>

Capítulo 1

Introdução

1.1 Enquadramento

O presente trabalho de grupo foi realizado no âmbito da Unidade Curricular (UC) de Sistema de Gestão de Base de Dados (SGBD) que se enquadra no primeiro ano do 2º Ciclo de Estudos em Engenharia Informática na Universidade da Beira Interior (UBI).

1.2 Motivação

Durante o desenvolvimento de Sistemas de Informação, é frequente lidar com casos onde várias aplicações acedem, de forma concorrente, a dados partilhados. O SGBD é a entidade responsável pela gestão destes acessos.

Este trabalho coloca em prática, e aprofunda, os conhecimentos adquiridos nas aulas teóricas e práticas do presente ano letivo, onde a sua concretização nos permitiu a compreensão de como é efetuada a manipulação dos dados e das suas transações.

1.3 Objetivos

Este trabalho visa conhecer os detalhes do controlo de transações, particularmente os níveis de isolamento, os tipos de trincos (*Locks*) e impasse (*Deadlock*).

Depois de ser criada a Base de Dados (BD) e construídas as aplicações, foi possível o estudo das transações em ambientes concorrentes, onde assim, tivemos a oportunidade de testar os diferentes níveis de isolamento das transações bem como os fenómenos a estes associados.

1.4 Constituição do Grupo

A realização deste documento, e do respetivo trabalho prático, ficou ao encargo dos seguintes elementos constituintes do grupo:

- Pedro Moreira, M11052
- Gabriel Esteves, M11151
- Nuno Salvado

1.5 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se dividido em seis capítulos principais:

1. O capítulo 1 – **Introdução** – apresenta uma descrição geral do trabalho, com o enquadramento, objetivos e motivação do mesmo, a constituição do grupo de trabalho e a organização do documento.
2. O capítulo 2 – **Tecnologias e Ferramentas Utilizadas** – faz uma descrição das tecnologias e ferramentas utilizadas neste trabalho.
3. O capítulo 3 – **Gestão e Controlo de Transações** – explica em que consiste a gestão e o controlo de transações.
4. O capítulo 4 – **Aplicações** – faz uma descrição das aplicações desenvolvidas que constituem este projeto.
5. O capítulo 5 – **Testes e Análise de Resultados** – apresenta os testes efetuados em relação ao controlo de transações em ambiente concorrente e os níveis de isolamento.

6. O capítulo 6 – **Conclusões e Trabalho Futuro** – inclui uma análise do trabalho realizado e algumas ideias para possíveis futuras implementações.

Capítulo 2

Tecnologias e Ferramentas Utilizadas

2.1 Introdução

Para a realização deste projeto foram utilizadas diversas ferramentas e tecnologias. Desta forma serão abordadas com detalhe aquelas que suportam a BD, bem como o *IDE* (IDE) e linguagem de programação para o desenvolvimento das aplicações.

2.2 *Visual Studio Code*

O IDE utilizado para o desenvolvimento da aplicação foi o *Visual Studio Code* (VSCode), representado na figura 2.1, tendo sido este lançado em 2015 pela *Microsoft* [13].

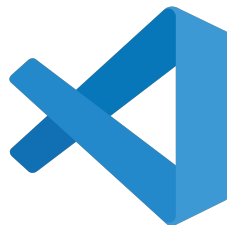


Figura 2.1: Logo do VS Code.

Este IDE apresenta diversas vantagens, entre as quais:

- **Gratuito**, mesmo sendo um programa da *Microsoft* com o nome *Visual Studio*;
- **Open-source**, garantindo assim facilidade na criação de extensões e uma menor taxa de *bugs*;
- **Multi-Plataforma**;
- **IntelliSense**, sendo esta uma assistência inteligente à codificação;
- **Integração com o Git**;
- **Suporte para várias linguagens de programação** através do uso de extensões;
- **Funcionalidades para *debugging*** [2].

2.3 Linguagem *Python*

A linguagem de programação Python, estando o seu logótipo representado na figura 2.2, começou a ser desenvolvida por *Guido van Rossum*, no ano 1989, no *Centrum Wiskunde & Informatica* (CWI), situado na Holanda. Este foi, no entanto, publicado em 1991 e posteriormente desenvolvido pela *Python Software Foundation*.

O *Python* distingue-se das restantes linguagens de programação pela sua fácil interpretação de código (por parte do programador) e pela sua sintaxe simples que permite exprimir conceitos e algoritmos complexos, geralmente, em poucas linhas de código [14].



Figura 2.2: Logótipo do *Python*.

Podemos considerar, como principais vantagens do *Python*, as seguintes:

- **Simples e fácil de aprender**, devido à sua sintaxe, quantidade de documentação e comunidade;
- **Realização de testes**, onde apresenta variadas estruturas de teste integrados;
- Suporte a mecanismos de **Automatização e Scripting**;
- **Big Data e Data Science**, sendo amplamente utilizado nestas áreas [10].

2.4 Qt

O Qt é uma *framework cross-plataform* baseada em C++ e que permite o desenvolvimento de *Graphics User Interface* (GUI)s para *Android*, *iOS*, *Windows*, *Linux* e *macOS*. Para além disso, inclui também abstrações de *sockets* de rede, *threads*, expressões regulares, BD *Structured Query Language* (SQL) e um extenso conjunto de *widgets* para GUIs. As suas classes aplicam um sistema de *signal/slot* para comunicarem entre objetos, tornando mais fácil criar componentes reutilizáveis de *software*.

Este inclui também o *Qt Designer* que é uma GUI para efeitos de *design* [18].

2.4.1 PyQt

O *PyQt* é um conjunto de módulos da linguagem *Python* que permite a ligação à *framework Qt* executando em todas as plataformas por esta suportadas.

Os módulos mais comumente utilizados estão representados na tabela 2.1 [17].

Nome do módulo	Descrição
<i>QtCore</i>	Contém as classes básicas, incluindo o mecanismo de sinal e <i>slot</i> , <i>threads</i> , expressões regulares e definições de aplicação e utilizador.
<i>QtGui</i>	Contém as classes para lidar com eventos, gráficos 2D, imagens, fontes e texto.
<i>QtSql</i>	Contém as classes que integram BD SQL. Inclui uma implementação do <i>SQLite</i> .

<i>QtWidgets</i>	Contém as classes para criar GUIs no estilo de <i>desktop</i> .
------------------	---

Tabela 2.1: Tabela de alguns módulos presentes no *PyQt*.

O *PyQt*, estando o seu logótipo representado na figura 2.3, combina as vantagens do *Qt* e *Python*, dando origem a um programa com o poder do *Qt* e a simplicidade do *Python* [16] [8].

Figura 2.3: Logótipo do *PyQt*.

No excerto de código abaixo, podemos observar a criação de uma janela através do uso do *PyQt*, onde para tal, utilizamos os módulos *Widgets* e *QtCore*.

2.5 Python Package Index

O *Python Package Index* (PyPI) é um repositório de *software* para a linguagem de programação *Python*. Este serve como fonte de dependências e pacotes ao *pip*. Mais de 113 mil pacotes *Python* podem ser acedidos através do PyPI [20] [6].

2.5.1 *pyodbc*

O *pyodbc* é um módulo *Python* de código aberto que torna simples o acesso às bases de dados *Open Database Connectivity* (ODBC). Este módulo implementa a especificação *BD API 2.0* [7]. No excerto de código 2.1 podemos observar os passos necessários para se efetuar uma conexão à BD.

```
import pyodbc

# Dados necessarios para a ligacao
server = 'tcp:myserver.database.windows.net'
database = 'mydb'
username = 'myusername'
password = 'mypassword'

# String de conexao
cnxn = pyodbc.connect('DRIVER={ODBC Driver 13 for SQL Server};SERVER='+
    server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()

# Exemplo de uma query SELECT
query = cursor.execute("SELECT * FROM MEI_TRAB.dbo.Encomendas;")
aux = []
for row in query:
    aux.append(str(row))
# Faz display da query
print(aux)
```

Excerto de Código 2.1: Exemplo de uma conexão a uma BD [15].

2.6 SQL

A linguagem *Structured English Query Language* (SEQUEL) começou a ser desenvolvida no ano de 1970 nos laboratórios da *IBM* pelos investigadores *Donald D. Chamberlin* e *Raymond F. Boyce*, com a finalidade de demonstrar a viabilidade do modelo relacional proposto pelo Dr. *E. F. Codd* na sua publicação intitulada de "*A Relational Model of Data for Large Shared Data Banks*" [12] [3].

Mais tarde passaria a chamar-se de SQL, sendo aceite como modelo padrão da linguagem *Relational Database Management Systems* (RDBMS), tendo sido padronizada pela *American National Standards Institute* (ANSI) em 1986 e pela *International Organization for Standardization* (ISO) em 1987 [1].

Para concluir, a linguagem SQL diferencia-se das outras linguagens de consulta à BD, na medida em que cada consulta especifica a forma do resultado, e não o caminho até ele.

2.7 Microsoft SQL Server

O *Microsoft SQL Server* (SQL Server), cujo o logótipo se encontra na figura 2.4, é um SGBD desenvolvido pela *Sybase* em parceria com a *Microsoft*.



Figura 2.4: Logo do SQL Server.

Uma vez que se trata de um servidor de BD, tem como principal função o armazenamento e a recuperação de dados conforme requisitados por outros *softwares*, que podem ser executados no mesmo computador, em servidor local ou em rede.

Existem várias edições do SQL Server destinadas a diferentes públicos-alvo, sendo que para a realização deste trabalho, foi utilizada a versão *Developer* numa máquina com o Sistema Operativo *Windows 10*.

2.8 SQL Server Management Studio

O *SQL Server Management Studio* (SSMS), cujo logótipo se encontra representado na figura 2.5, é um ambiente integrado, lançado pela primeira vez com o SQL Server 2005, para configurar, gerir e controlar todos os componentes do SQL Server [11] [22].



[h]

Figura 2.5: Logo do SSMS.

Na realização deste projeto, foi utilizada a versão SSMS 18, lançada em Outubro de 2020.

2.9 Conclusões

Neste capítulo foram caracterizadas e definidas todas as ferramentas e tecnologias utilizadas no desenvolvimento e implementação deste projeto. Todas elas foram essenciais para o cumprimento dos objetivos propostos.

Foram assim descritas a linguagem utilizadas para o desenvolvimento da aplicação, tanto na vertente de *frontend* como de *backend*, bem como os *s* utilizados referentes ao *IDE* e *SGBD*.

Capítulo 3

Gestão e Controlo de Transações

3.1 Introdução

Neste capítulo serão abordados alguns assuntos que são relevantes para o desenvolvimento deste projeto.

3.2 Transações

Uma transação retrata uma unidade lógica ou de trabalho executada dentro de um SGBD, sobre uma BD. Esta representa, geralmente, qualquer alteração numa BD.

As transações são um conjunto de instruções SQL. Estas são sequências de trabalho que podem ser executadas seguindo uma ordem lógica, onde para tal, recorrem a blocos para preservar a integridade dos dados durante a sua execução. Se uma transação terminar corretamente, esta pode utilizar o comando *COMMIT*, para assim armazenar permanentemente as suas modificações. Caso contrário, recorre-se ao comando *ROLLBACK*, para cancelar todas as modificações efetuadas e restaurar a BD para uma instância anterior.

As transações num ambiente de BD têm duas objetivos principais:

- Fornecer unidades de trabalho confiáveis que permitam a recuperação correta de falhas e manter uma BD consistente, mesmo em casos de falha

do sistema;

- Fornecer isolamento entre programas que acessem a uma BD em simultâneo, caso contrário os resultados dos programas possivelmente serão imprevisíveis.

A integridade de uma transação depende de quatro propriedades, denominadas propriedades *Atomic, Consistent, Isolated, Durable* (ACID):

- **Atomicidade** - Todas as ações que compõem a unidade lógica ou de trabalho devem ser concluídas com sucesso, para que a transação seja efetuada;
- **Consistência** - Todas as restrições e regras definidas na base de dados devem ser cumpridas;
- **Isolamento** - Cada transação funciona completamente à parte de outras estações (as operações são parte de uma transação única);
- **Durabilidade** - Significa que os resultados de uma transação são permanentes e, podem vir a ser desfeitos exclusivamente por uma transação posterior [19].

3.3 Níveis de Isolamento

No ramo das BD, o isolamento, uma propriedade ACID, é também, uma propriedade que define como e quando uma alteração feita por uma transação se torna visível para outras transações concorrentes.

A maior parte dos SGBD permitem definir o nível de isolamento das transações, controlando assim o grau de concorrência da BD. Assim, fica ao encargo do programador decidir qual o nível de isolamento mais adequado para cada aplicação, sendo que a maioria das aplicações não pedem um nível de isolamento muito restritivo, permitindo, assim, uma elevada concorrência e evitando a possibilidade de ocorrerem *deadlocks* (quanto mais alto o nível de isolamento, menos concorrência permite e mais probabilidade do sistema entrar em *deadlock* de recursos).

Existem quatro níveis de isolamento, sendo estes: o *Read Uncommitted*, *Read Committed*, *Repeatable Read* e o *Serializable*. Estes são classificados segundo a

possibilidade de ocorrer determinados fenômenos não desejados, que podem ser “*Dirty reads*”, “*Non-repeatable reads*” e “*Phantoms*”. A tabela 3.1 compara os diferentes níveis de isolamento com os fenômenos a estes associados.

	<i>Dirty Read</i>	<i>Non-Repeatable Read</i>	<i>Phantom</i>
<i>Read Uncommitted</i>	Sim	Sim	Sim
<i>Read Committed</i>	Não	Sim	Sim
<i>Repeatable Read</i>	Não	Não	Sim
<i>Serializable</i>	Não	Não	Não

Tabela 3.1: Classificação dos Níveis de Isolamento.

- ***Read Uncommitted*** - É o primeiro nível e o menos proibitivo de todos os níveis de isolamento. Neste nível podem ocorrer os três fenômenos (*Dirty Reads*, *Non-Repeatable Read* e *Phantom*), uma vez que não é adquirido qualquer tipo de *lock* de dados;
- ***Read Committed*** - É o nível que é usado por defeito no *Oracle* e no *SQL Server*. Neste nível, os *dirty reads* não ocorrem porque são usados *locks* partilhados que asseguram que nenhuma informação corrompida ou alterada por outra transação (que ainda não tenha sido *committed*) é lida. Este nível não assegura que os dados não sejam alterados antes do fim da transação, por isso, permite a ocorrência de *non-repeatable reads*;
- ***Repeatable Read*** - É o nível onde são adquiridos *locks* de leitura, prevenindo a ocorrência de *dirty reads* e de *non-repeatable reads*, mas permite a ocorrência de *phantoms*, pois não são adquiridos “*range-locks*”;
- ***Serializable*** - É o nível onde todas as transações ocorrem num meio fechado, isto é, todas elas são executadas de modo sequencial. O SGBD pode executar transações concorrentemente apenas se a ilusão de serialização for mantida, ou seja, se uma transação não partilhar qualquer tipo de dados com a outra. Neste nível são usados *range locks*, não permitindo, portanto, qualquer ocorrência de *phantoms* ou de outros fenômenos [5].

3.4 Recuperação (*Rollback*)

Sempre que uma transação for concluída com sucesso, a BD será alterada permanentemente, onde os dados afetados por esta transação serão guardados em

disco, através do comando *Commit*. Por outro lado, se houver alguma falha em qualquer uma das operações que compõem a transação, os dados anteriores à falha, são copiados novamente para a BD, através do comando *Rollback*.

A operação *Rollback* é importante para manter a integridade da BD, pois permite que esta possa ser restaurada para uma cópia anterior, mesmo após a execução de operações erradas.

Em *SQL*, a instrução *ROLLBACK* permite-nos carregar todas as modificações de dados desde o último *BEGIN WORK* ou *START TRANSACTION* [4] [21].

3.5 Tipos de Trincos (*Locks*) e de Impasse (*Deadlock*)

Diversas são as vezes em que ouvimos falar de *locks* e de *deadlocks*, da forma menos correta.

Um *lock* ocorre quando qualquer processo acede a uma parte dos dados e outro processo, em simultâneo, precisa dessa exata parte dos dados ao mesmo tempo. *Lock* é o mecanismo que o SQL Server usa para proteger a integridade dos dados durante as transações, onde a BD ativa *LOCKS* que por sua vez impedem a modificação, em simultâneo dos dados, por transações concorrentes.

Um *deadlock* ocorre quando um processo é bloqueado e fica à espera que um segundo processo finalize o seu trabalho e liberte os *locks*. Simultaneamente o segundo processo é bloqueado e fica à espera que o primeiro liberte o *lock*. Os *deadlocks* são considerados um ponto chave no que toca às BD, pois os processos são eliminados automaticamente e, por isso, podem e devem ser evitados [9].

Na figura 3.1, está representado um caso de um *deadlock* entre 2 processos concorrentes.

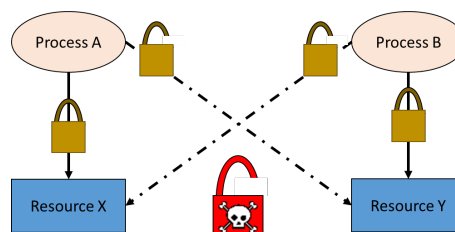


Figura 3.1: Representação de um *deadlock*.

3.6 Conclusões

Este capítulo visou a apresentação do que se entende por gestão e controlo de transações, onde se abordaram os quatro níveis de isolamento e os fenómenos a estes associados, o conceito de *rollback* e os tipos de trincos e de impasse.

Capítulo 4

Aplicações

4.1 Introdução

Neste capítulo são apresentadas todas as fases do programa e o seu respetivo funcionamento.

4.2 Conexão à Base de Dados Remota

Ao iniciar o programa é necessário fazer a conexão à BD remota, onde para tal é necessário preencher alguns campos, sendo estes os seguintes:

- **Host Name** – necessita do *IP* da máquina a correr o SQL Server, bem como a porta *TCP*;
- **Database Name** – nome dado à BD onde o SQL Server está a executar;
- **Username** – nome de utilizador contido na BD;
- **Password** – palavra-passe correspondente ao utilizador.

Como se pode observar na figura 4.1, estes campos encontram-se preenchidos de forma correta, para que assim haja sucesso na conexão à BD remota. Como tal, e se for verificado que existe conexão, avançamos para o Menu da Aplicação.

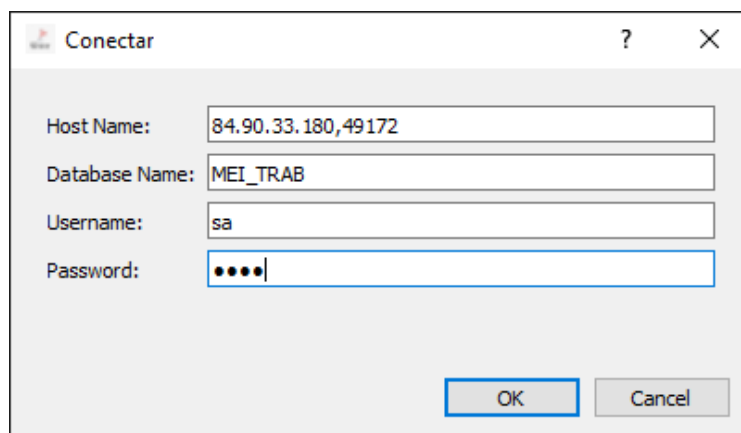


Figura 4.1: Janela de Conexão à Base de Dados.

4.3 Menu da Aplicação

No Menu da Aplicação, é possível escolher o nível de isolamento que se pretende, sendo que estes são os quatro níveis abordados no capítulo 3.3. Por *default* está selecionado o *Read Uncommitted*.

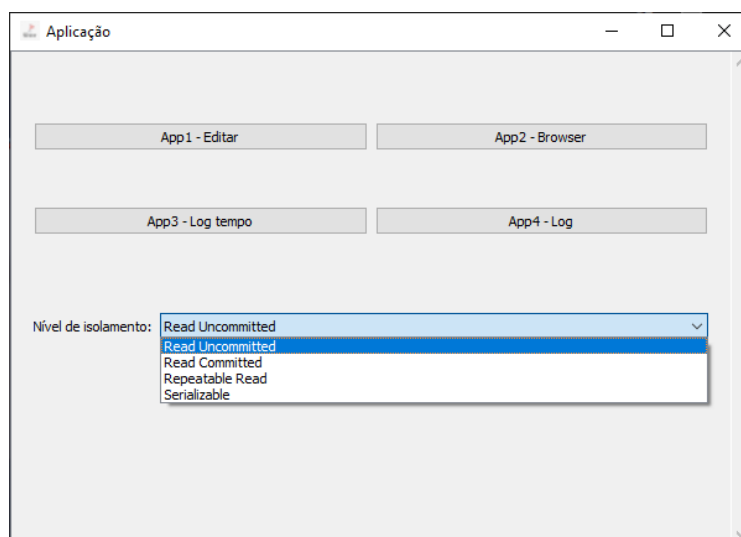


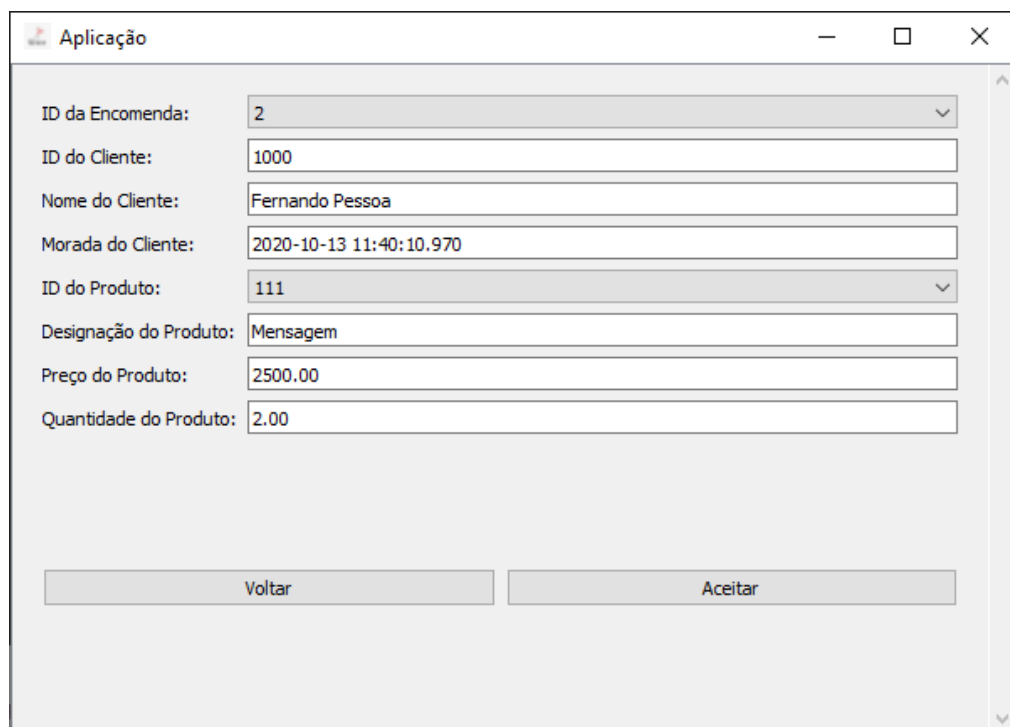
Figura 4.2: Menu da Aplicação.

Como podemos visualizar na figura 4.2, é possível, através do presente menu,

aceder a uma das quatro aplicações existentes.

4.4 Aplicação *Edit*

Nesta aplicação é possível escolher o *ID* de uma encomenda, sendo que os parâmetros abaixo deste serão preenchidos de forma automática. Para tal a aplicação irá criar e executar uma transação com 2 *queries* à BD, sendo que o nível de isolamento foi escolhido anteriormente. Os dados retornados pelas *queries* serão armazenados em variáveis auxiliares, para que assim estes possam ser utilizados sem ser necessário e execução excessiva de pedidos à BD.



The screenshot shows a window titled "Aplicação" with a standard Windows title bar (minimize, maximize, close buttons). The window contains a form with the following fields:

- ID da Encomenda: 2 (dropdown menu)
- ID do Cliente: 1000 (text input)
- Nome do Cliente: Fernando Pessoa (text input)
- Morada do Cliente: 2020-10-13 11:40:10.970 (text input)
- ID do Produto: 111 (dropdown menu)
- Designação do Produto: Mensagem (text input)
- Preço do Produto: 2500.00 (text input)
- Quantidade do Produto: 2.00 (text input)

At the bottom of the form, there are two buttons: "Voltar" (Back) and "Aceitar" (Accept).

Figura 4.3: Aplicação *Edit*.

Como podemos observar na figura 4.3, estarão disponíveis para alterar a Morada do Cliente e a Quantidade do Produto, onde esta ultima só aceita, como *input* valores numéricos inteiros.

Aquando o clique no botão "Aceitar", será criada e executada uma transação, a qual irá conter 3 *queries* à BD. O excerto de código 4.1, demonstra o nível de

isolamento selecionado pelo utilizador, bem como a criação e execução dos pedidos à BD. Após a correta execução dos mesmos, será feito o comando *COMMIT*, de modo a submeter as alterações efetuadas. Caso contrário será feito, de forma automática pelo SQL Server, um *ROLLBACK*.

```
def EditFunction( self ):
    SetIsolationLevel( self.conn, self.isolationComboBox.currentText() )
    query = "UPDATE " + self.dbName + ".dbo.Encomenda SET ClienteID = " +
        self.clientIDField.text() + ", Nome = '" + self.clientNameField.
        text() + "', Morada = '" + self.clienteMoradaField.text() + "'
    WHERE EncID = " + self.encIDField.currentText()
    self.cursor.execute(query)

    query = "UPDATE " + self.dbName + ".dbo.EncLinha SET Designacao = '"
        + self.productDesignationField.text() + "', Preço = " + self.
        productPriceField.text() + ", Qtd = " + self.productQtyField.text
        () + " WHERE EncId = " + self.encIDField.currentText() + " AND
        ProdutoID = " + self.productIDField.currentText()
    self.cursor.execute(query)

    query = 'SELECT Referencia FROM ' + self.dbName + ".dbo.LogOperations
        WHERE Objecto = '" + self.encIDField.currentText() + "'"
    query = self.cursor.execute(query)

    aux = []
    for row in query:
        aux.append(row)

    date = datetime.now()
    dtString = date.strftime( '%Y-%m-%d %H:%M:%S%f' )
    if not aux:
        uniqueString = 'G1-' + dtString
    else:
        aux1 = aux[0]
        uniqueString = aux1[0]
    query = "INSERT INTO " + self.dbName + ".dbo.LogOperations (EventType
        , Objecto, Valor, Referencia) VALUES ('O', " + self.encIDField.
        currentText() + ", '" + dtString + "', '" + uniqueString + "'" )
    self.cursor.execute(query)
    self.conn.commit()
```

Excerto de Código 4.1: Transação efetuada após o clique no botão "Aceitar" na Edição.

4.5 Aplicação *Browser*

Nesta aplicação será possível seleccionar um *ID* válido da tabela *Encomenda*. Após esta selecção, serão listados todos os parâmetros das tabelas *Encomenda* e *EncLinha* referentes a esse *ID*, como podemos observar na figura 4.4.

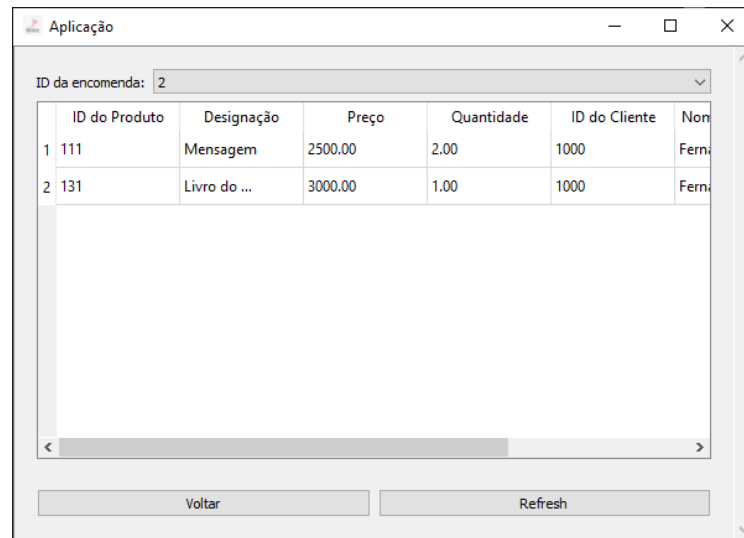


Figura 4.4: Aplicação *Browser*.

O excerto de código abaixo, 4.2, podemos observar a criação de uma transacção com três pedidos à BD.

```
def DisplayBrowser ( self ) :
    self.stacked.setCurrentIndex(2)
    SetIsolationLevel(self.conn, self.isolationComboBox.currentText())
    query = 'SELECT EncID FROM ' + self.dbName + '.dbo.Encomenda'
    query = self.cursor.execute(query)

    for row in query:
        self.encIDs.append(str(row[0]))

    self.choosedEnc.addItem(self.encIDs)
    self.choosedEnc.currentIndexChanged.connect(self.RefreshBrowser)

    query = 'SELECT * FROM ' + self.dbName + '.dbo.Encomenda WHERE EncID'
    query = self.cursor.execute(query)
```

```
for row in query:
    clienteID = str(row[1])
    clienteNome = str(row[2])
    clienteMorada = (row[3])

self.ClearAuxiliar(2)

query = 'SELECT * FROM ' + self.dbName + '.dbo.EncLinha WHERE EncID = ' + self.choosedEnc.currentText()
query = self.cursor.execute(query)
for row in query:
    self.produtoIDs.append(str(row[1]))
    self.designacao.append(str(row[2]))
    self.preco.append(str(row[3]))
    self.qtd.append(str(row[4]))
self.conn.commit()

aux = []
# Vai preencher a lista auxiliar
for i in range(len(self.qtd)):
    aux.append([self.produtoIDs[i], self.designacao[i], self.preco[i], self.qtd[i], clienteID, clienteNome, clienteMorada])

data = np.matrix(aux)
modal = TableModel(data, 0)
self.browserTable.setModel(modal)
```

Excerto de Código 4.2: Transação efetuada para o preenchimento dos campos na aplicação *Browser*.

É também possível, caso pretendido, atualizar os dados apresentados na tabela, onde para tal será executada uma função semelhante à do excerto acima explicado.

4.6 Aplicação *Log Tempo*

Nesta aplicação, existirá a opção de selecionar qual o intervalo pretendido. O excerto de código 4.3, permite a seleção de sete dos onze parâmetros possíveis. Por *default* está selecionado o intervalo em horas.

```
# Trata de como queremos o Tempo
# https://www.w3schools.com/sql/func_sqlserver_datediff.asp
if self.timeField.currentText() == 'Ano(s)':
```

```

        interval = 'YEAR'
    elif self.timeField.currentText() == 'M s /Meses':
        interval = 'MONTH'
    elif self.timeField.currentText() == 'Dia(s)':
        interval = 'DAY'
    elif self.timeField.currentText() == 'Hora(s)':
        interval = 'HOUR'
    elif self.timeField.currentText() == 'Minuto(s)':
        interval = 'MINUTE'
    elif self.timeField.currentText() == 'Segundo(s)':
        interval = 'SECOND'
    else:
        interval = 'MILLISECOND'

```

Excerto de Código 4.3: Diferentes parâmetros de *DATEDIFF*.

Na figura 4.5, podemos selecionar um outro intervalo, mas para que este faça efeito, é necessário clicar no botão de "Refresh".

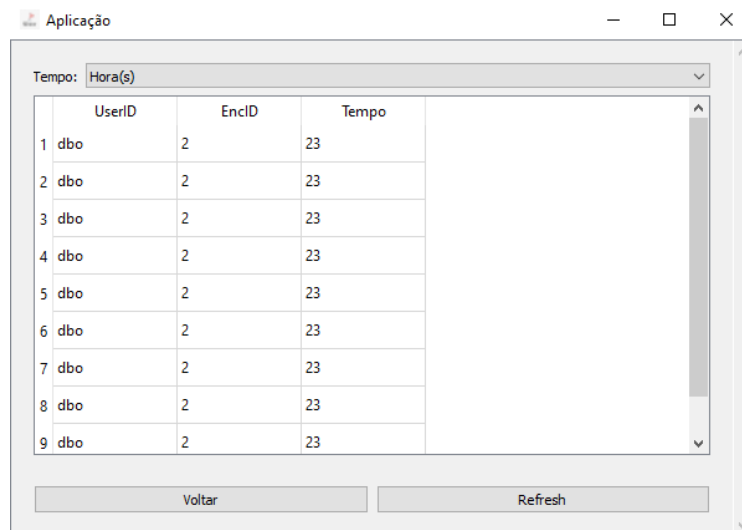


Figura 4.5: Aplicação *Log Tempo*.

O excerto de código 4.4, cria uma transação com 2 pedidos à BD. O primeiro pedido trata da transformação dos valores das datas em string para o formato necessário para o bom funcionamento do *DATEDIFF*.

```

SetIsolationLevel(self.conn, self.isolationComboBox.currentText())
# Tenho que tratar do Valor para yyyy/mm/dd hh:mm:ss:ms
query = "SELECT L1.Valor, L2.Valor FROM " + self.dbName + ".dbo.
        LogOperations L1, " + self.dbName + ".dbo.LogOperations L2 WHERE L1.

```

```

        EventType = 'O' and L1.EventType = L2.EventType and L1.Referencia = L2
        .Referencia and L1.Valor < L2.Valor"
query = self.cursor.execute(query)
for row in query:
    aux.append(row)
if aux:
    fillAux = []
for i in range(len(aux)):
    l_aux = aux[i]
    l1Referencia = 'G1-' + l_aux[0]
    l1Valor = l_aux[0]
    l2Valor = l_aux[1]
    l1Valor = l1Valor[:4] + '/' + l1Valor[4:6] + '/' + l1Valor[6:8] + ' '
        + l1Valor[8:10] + ':' + l1Valor[10:12] + ':' + l1Valor[12:14] + '
        .' + l1Valor[14:]
    l2Valor = l2Valor[:4] + '/' + l2Valor[4:6] + '/' + l2Valor[6:8] + ' '
        + l2Valor[8:10] + ':' + l2Valor[10:12] + ':' + l2Valor[12:14] + '
        .' + l2Valor[14:]

    query = "SELECT L1.UserID, L1.Objecto AS EncId, DATEDIFF(" + interval
        + ", '" + l1Valor + "', '" + l2Valor + "') AS Tempo FROM " + self.
        dbName + ".dbo.LogOperations L1, " + self.dbName + ".dbo.
        LogOperations L2 WHERE L1.EventType = 'O' and L1.EventType = L2.
        EventType and L1.DCricao < L2.DCricao and L1.Referencia = '" +
        l1Referencia + "'"
    query = self.cursor.execute(query)
    for row in query:
        self.userID.append(str(row[0]))
        self.objeto.append(str(row[1]))
        tempo.append(str(row[2]))

# Vai preencher a lista auxiliar
for i in range(len(tempo)):
    fillAux.append([self.userID[i], self.objeto[i], tempo[i]])
self.conn.commit()

```

Excerto de Código 4.4: Excerto da função que faz *refresh* à aplicação Log Tempo.

4.7 Aplicação Log

Nesta aplicação será possível escolher o número de linhas pretendidas a serem apresentadas, onde cada linha será uma entrada na tabela *LogOperations*, sendo que tal é possível observar-se na figura 4.6. Por *default* serão apresentadas todas as entradas dessa tabela.

	NumReg	EventType	Objecto	Valor	Referencia
1	1	I	Encomenda	1	None
2	2	I	EncLinha	1	111
3	3	I	EncLinha	1	131
4	4	D	EncLinha	1 131	Livro do ...
5	5	D	EncLinha	1 111	Mensagem ...
6	6	D	Encomenda	1 Fernando ...	None
7	7	I	Encomenda	1	None
8	8	I	EncLinha	1	111

Figura 4.6: Aplicação Log.

O excerto de código 4.5, apresenta a transação com uma *query*, onde será feito o pedido de todo o conteúdo contido na tabela *LogOperations*. Os dados serão então armazenados em variáveis locais, para que assim possam ser apresentados ao utilizador.

```
SetIsolationLevel(self.conn, self.isolationComboBox.currentText())
query = 'SELECT * FROM ' + self.dbName + '.dbo.LogOperations'
query = self.cursor.execute(query)
for row in query:
    self.numReg.append(str(row[0]))
    self.eventType.append(str(row[1]))
    self.objeto.append(str(row[2]))
    self.valor.append(str(row[3]))
    self.referencia.append(str(row[4]))
    self.userID.append(str(row[5]))
    self.terminalID.append(str(row[6]))
    self.terminalName.append(str(row[7]))
    self.dataCriacao.append(str(row[8]))
self.conn.commit()
```

Excerto de Código 4.5: Excerto de código da função Log.

4.8 Conclusões

Este capítulo apresentou o funcionamento do trabalho prático, nomeadamente das quatro aplicações que o compõem, bem como o menu principal da aplicação e a conexão à BD remota.

Capítulo 5

Testes e Análise de Resultados

5.1 Introdução

Neste capítulo são apresentadas diversas capturas de ecrã de testes efetuados ao trabalho desenvolvido pelo grupo e a análise dos resultados desses mesmos testes.

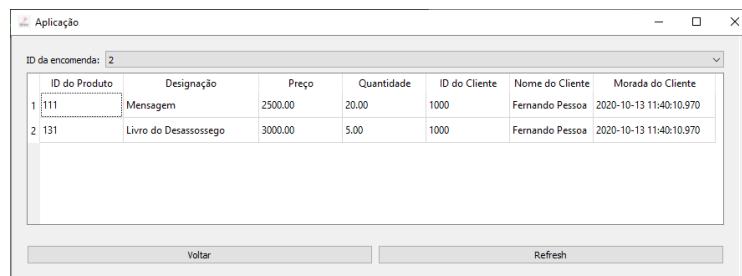
5.2 Testes

Nesta secção irão ser apresentados os diversos testes realizados às aplicações.

5.2.1 Alteração da morada da encomenda e da quantidade de um artigo

Neste teste podemos verificar que é possível fazer alterações à encomenda, e que em caso de sucesso, os registos ficam armazenados na BD, sendo que fica registado quando e quem a alterou.

Na figura 5.1 conseguimos observar as informações originais referentes à encomenda com o *ID* 2, onde existem 2 artigos.

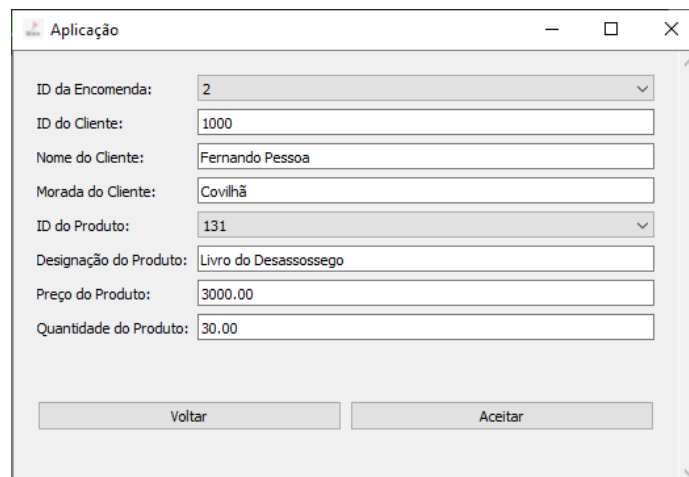


The screenshot shows a web application window titled 'Aplicação'. At the top, there is a dropdown menu labeled 'ID da encomenda:' with the value '2' selected. Below this is a table with 7 columns: 'ID do Produto', 'Designação', 'Preço', 'Quantidade', 'ID do Cliente', 'Nome do Cliente', and 'Morada do Cliente'. The table contains two rows of data. Below the table, there are two buttons: 'Voltar' and 'Refresh'.

	ID do Produto	Designação	Preço	Quantidade	ID do Cliente	Nome do Cliente	Morada do Cliente
1	111	Mensagem	2500.00	20.00	1000	Fernando Pessoa	2020-10-13 11:40:10.970
2	131	Livro do Desassossego	3000.00	5.00	1000	Fernando Pessoa	2020-10-13 11:40:10.970

Figura 5.1: Aplicação *Browser* com as quantidades e a morada originais.

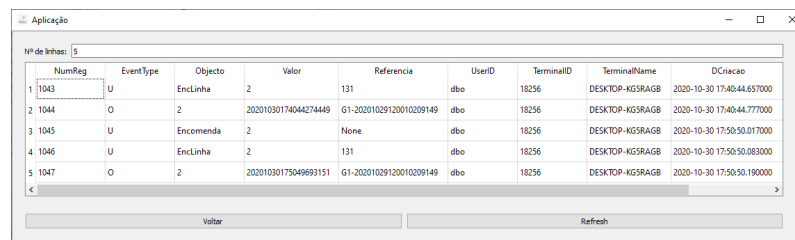
Após a alteração e submissão dos novos valores, pode-se observar que a aplicação *Edit* já os apresenta, como podemos observar na figura 5.2.



The screenshot shows the 'Aplicação' window in edit mode. It features several input fields for editing order details. The fields are: 'ID da Encomenda:' (dropdown with '2'), 'ID do Cliente:' (text box with '1000'), 'Nome do Cliente:' (text box with 'Fernando Pessoa'), 'Morada do Cliente:' (text box with 'Covilhã'), 'ID do Produto:' (dropdown with '131'), 'Designação do Produto:' (text box with 'Livro do Desassossego'), 'Preço do Produto:' (text box with '3000.00'), and 'Quantidade do Produto:' (text box with '30.00'). At the bottom, there are two buttons: 'Voltar' and 'Aceitar'.

Figura 5.2: Aplicação *Edit* da encomenda com o *ID* 2 com a morada e a quantidade do produto 131 alterados.

Na figura 5.3 é possível observar, na linha 5, que realmente existiu uma alteração do tipo *O - Outra*, do objeto 2 e que nela consta a data da alteração e quem fez a alteração.



	NumReg	EventType	Objecto	Valor	Referencia	UserID	TerminalID	TerminalName	D/Criacao
1	1043	U	EncLinha	2	131	dbo	18256	DESKTOP-KG5RAGB	2020-10-30 17:40:44.657000
2	1044	O	2	20201030174044274449	G1-20201029120010209149	dbo	18256	DESKTOP-KG5RAGB	2020-10-30 17:40:44.777000
3	1045	U	Encomenda	2	None	dbo	18256	DESKTOP-KG5RAGB	2020-10-30 17:50:50.017000
4	1046	U	EncLinha	2	131	dbo	18256	DESKTOP-KG5RAGB	2020-10-30 17:50:50.083000
5	1047	O	2	20201030175049693151	G1-20201029120010209149	dbo	18256	DESKTOP-KG5RAGB	2020-10-30 17:50:50.190000

< >

Voltar Refresh

Figura 5.3: Aplicação *Log* em que é possível ver que a encomenda com o *ID* 2 foi editada.

5.3 Conclusões

Este capítulo apresentou essencialmente o *output* das aplicações desenvolvidas durante o projeto, sendo que estes foram executados tendo o *script Do Work* a funcionar na máquina quem contém a BD.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

Durante o desenvolvimento deste trabalho foi possível rever os assuntos lecionados na Unidade Curricular de Bases de Dados, mas também foi possível adquirir novos conhecimentos inseridos na Unidade Curricular de Sistemas de Gestão de Bases de Dados.

Este trabalho permitiu aprofundar o conhecimento em relação aos níveis de isolamento da base de dados e perceber as especificações de cada um dos níveis usados, sendo eles: o *"Read Uncommitted"*, o *"Read Committed"*, o *"Repeatable Read"* e o *"Serializable"*. Também permitiu interagir com as transações e ver como elas funcionam em cada nível de isolamento, assim como as recuperações dessas mesmas transações.

O presente trabalho permitiu, ainda, utilizar tecnologias que ainda não tinham sido usadas por todos os membros do grupo de trabalho, tais como o *Qt* e o *PyQt*, que permitem criar a interface gráfica que o utilizador vê, e o módulo *pyodbc*, que torna simples o acesso à base de dados. Foi possível rever tecnologias que já tinham sido usadas anteriormente, na Unidade Curricular de Bases de Dados, sendo elas: o *Microsoft SQL Server*, que é um sistema de gestão de bases de dados relacional e o *SQL Server Management Studio* que é um ambiente integrado.

O desenvolvimento deste trabalho enriqueceu o conhecimento do grupo de trabalho na área de Base de Dados e na área de Sistemas de Gestão de Bases

de Dados, e permitiu superar vários desafios que foram ocorrendo ao longo do seu desenvolvimento.

6.2 Trabalho Futuro

Com vista à complementação do trabalho apresentado, algumas melhorias e/ou enriquecimentos poderiam ser feitos.

Tais como acrescentar mais campos à tabela das encomendas, por exemplo um campo do estado da encomenda, para se saber se já foi enviada, está pendente, em entrega ou recebida.

Outra funcionalidade que se poderia adicionar na aplicação seria de saber a entidade que entrega cada encomenda, e os utilizadores conseguirem classificar quais as entregadoras mais rápidas, quais não entregam as encomendas danificadas ou fora de prazo, de forma a perceber quais as melhores entregadoras e usar preferencialmente essas, por exemplo.

Bibliografia

- [1] Allyn Grey de Almeida Lima. Padrão SQL e sua Evolução, 2019. [Online] <http://www.ic.unicamp.br/~geovane/mo410-091/Ch05-PadraoSQL-art.pdf>. Último acesso a 20 de Outubro de 2020.
- [2] ByLearn. 11 Motivos para migrar para o VS Code, 2019. [Online] <https://medium.com/@bylearn/11-motivos-para-migrar-para-o-vs-code-5b9574a057f5>. Último acesso a 28 Outubro de 2020.
- [3] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–389, 1970.
- [4] Guilherme Tavares de Assis. Processamento de Transações. [Online] http://www.decom.ufop.br/guilherme/BCC441/geral/bd2_processamento-de-transacoes.pdf. Último acesso a 28 Outubro de 2020.
- [5] Bruno Gomes ErrorStream. Níveis de isolamento da Base de Dados, 2009. [Online] <https://errorstream.wordpress.com/2009/04/01/niveis-de-isolamento-da-base-de-dados/>. Último acesso a 28 Outubro de 2020.
- [6] Python Software Foundation. Find, install and publish Python packages with the Python Package Index, 2020. [Online] <https://pypi.org/>. Último acesso a 28 Outubro de 2020.
- [7] Python Software Foundation. pyodbc 4.0.30, 2020. [Online] <https://pypi.org/project/pyodbc/>. Último acesso a 28 Outubro de 2020.
- [8] Python Software Foundation. PyQt5 5.14.2, 2020. [Online] <https://pypi.org/project/PyQt5/>. Último acesso a 19 de Outubro de 2020.

- [9] Pythian Group. Locks, Blocks e Deadlocks – Qual a diferença?, 2020. [Online] <https://dbabrasil.net.br/locks-blocks-deadlocks/>. Último acesso a 28 Outubro de 2020.
- [10] Mastertech. 10 motivos para você aprender Python, 2018. [Online] <https://www.hostgator.com.br/blog/10-motivos-para-voce-aprender-python/>. Último acesso a 30 de Outubro de 2020.
- [11] Microsoft. Download SQL Server Management Studio (SSMS), 2020. [Online] <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>. Último acesso a 28 Outubro de 2020.
- [12] Oracle. History of SQP, 2003. [Online] https://docs.oracle.com/cd/B12037_01/server.101/b10759/intro001.htm. Último acesso a 20 de Outubro de 2020.
- [13] Pedro Pinto. Visual Studio Code: O melhor editor para programadores?, 2017. [Online] <https://pplware.sapo.pt/software/visual-studio-code-melhor-editor-programadores/>. Último acesso a 28 Outubro de 2020.
- [14] SohomPramanick. History of Python. [Online] <https://geeksforgeeks.org/history-of-python/>. Último acesso a 30 de Outubro de 2020.
- [15] *mihaelablendea*. *Azure Data SQL Samples Repository*. [Online] <https://github.com/microsoft/sql-server-samples>. Último acesso a 28 de Outubro de 2020.
- [16] *Riverbank Computing*. *Modules*. [Online] <https://riverbankcomputing.com/software/pyqt/intro>. Último acesso a 28 Outubro de 2020.
- [17] *Riverbank Computing*. *What is PyQt?* [Online] https://www.riverbankcomputing.com/static/Docs/PyQt5/module_index.html. Último acesso a 28 de Outubro de 2020.
- [18] *The Qt Company*. *One framework. One codebase. Any platform*. [Online] <https://www.qt.io/>. Último acesso a 6 de Agosto de 2020.
- [19] Wikipedia. Transação (banco de dados), 2019. [Online] [https://pt.wikipedia.org/wiki/Transa%C3%A7%C3%A3o_\(banco_de_dados\)](https://pt.wikipedia.org/wiki/Transa%C3%A7%C3%A3o_(banco_de_dados)). Último acesso a 28 Outubro de 2020.

- [20] Wikipedia. Python Package Index, 2020. [Online] https://pt.wikipedia.org/wiki/Python_Package_Index. Último acesso a 28 Outubro de 2020.
- [21] Wikipedia. Rollback (data management), 2020. [Online] [https://en.wikipedia.org/wiki/Rollback_\(data_management\)](https://en.wikipedia.org/wiki/Rollback_(data_management)). Último acesso a 28 Outubro de 2020.
- [22] Wikipedia. SQL Server Management Studio, 2020. [Online] https://en.wikipedia.org/wiki/SQL_Server_Management_Studio. Último acesso a 28 Outubro de 2020.