

Universidade da Beira Interior

Departamento de Informática



Departamento de
Informática

Zurrapa – Drinks & Coffee

Licenciatura em Engenharia Informática

Base de Dados



Elaborado por:

Maria Pais, a45564

Guilherme Nunes, a45894

Sara Inácio, a46228

Prof. Doutor João Muranho

Covilhã, janeiro de 2022

Agradecimentos

A entrega deste projeto não teria sido possível sem a colaboração, auxílio, responsabilidade e dedicação total por parte de todos os elementos do grupo.

Queremos agradecer às nossas famílias e aos nossos amigos que respeitaram e compreenderam o nosso trabalho, especialmente nas épocas festivas.

Por fim, agradecer ao Prof. João Muranho, regente da Unidade Curricular de Base de Dados, da Universidade da Beira Interior, pela orientação, pela sua disponibilidade para nosso auxílio, assim como pela oportunidade de desenvolvimento e evolução de *soft-skills*, e aprendizagem de novos conhecimentos na área informática, através da realização deste projeto.

Resumo

Este projeto foi desenvolvido no âmbito da unidade curricular (UC) de Base de Dades da Licenciatura em Engenharia Informática da Universidade de Beira Interior (UBI), a fim de consolidar os conhecimentos adquiridos na mesma, assim como adquirir capacidades de trabalho de equipa e aperfeiçoar técnicas informáticas para o futuro.

O projeto consiste na implementação de um sistema de informação para a gerência de uma empresa designada por “Zurrapa - Drinks & Coffee, Lda.” que se dedica à exploração de *snack-bares* em estabelecimentos de grandes dimensões e possui filiais em vários pontos do país. O sistema de informação deve permitir á gerência conhecer as vendas das suas filiais.

Para além do desenvolvimento da aplicação que trata da caixa e do balcão, que lida com os clientes (receção de pedidos, pagamentos de pedidos, e consulta de pedidos), existe também uma parte deste *software* desenvolvido para gerir o armazém (permite a consulta do stock do armazém e dos bares) e para gerir as estatísticas (que faculta informação sobre despesas e lucros).

O Sistema de Gestão de Bases de Dados (SGBD) usado foi o SQL Server Management Studio 18 (SSMS), e as três aplicações foram desenvolvidas em Visual Basic (utilizando o Visual Studio 2022 para a escrita de código e desenvolvimento das interfaces do utilizador).

Palavras-chave: aplicações, base de dados, modelo, organização, projeto, SQL, SSMS.

Índice

1	Introdução.....	5
1.1	Enquadramento	5
1.2	Motivação	5
1.3	Objetivos.....	5
1.4	Organização do documento	5
2	Desenvolvimento de aplicações cliente/servidor sobre base de dados.....	6
2.1	Introdução.....	6
2.2	Aplicações cliente/servidor	6
2.3	SQL server	6
2.4	Configuração do acesso ao servidor	7
2.5	Visual Studio 2022 e Visual Basic	7
3	Modelação	7
3.1	Introdução.....	7
3.2	Descrição da organização	8
3.3	Modelo Conceptual	8
3.4	Modelo Lógico	9
3.5	Considerações.....	10
4	Aplicações	10
4.1	Distribuição das tarefas.....	10
4.1.1	Descrição precisa das tarefas	11
4.2	Acesso à base de dados	11
4.3	Funcionalidade.....	11
4.3.1	Descrição Geral	11
4.3.2	Aplicações	12
5	Conclusões	22
	Epílogo	22
	Referências Bibliográficas	22
	Anexos	22
	Apêndices	23

Lista de Abreviaturas

- **UC** - Unidade Curricular
- **SQL** - Structured Query Language
- **UBI** - Universidade da Beira Interior
- **SGBD** - Sistema de Gestão de Base de Dados

Lista de Figuras

- **Figura 1** - Modelo conceptual da Sede
- **Figura 2** - Modelo Conceptual da Filial
- **Figura 3** – Modelo Lógico da Filial
- **Figura 4** – Modelo Lógico da Sede
- **Figura 5** - Esquema de funcionamento das aplicações
- **Figura 6** – Aplicação Bar – fazer pedido
- **Figura 7** – Aplicação Bar – consultar pedidos
- **Figura 8** – Aplicação Bar – consultar stock
- **Figura 9** - Aplicação Bar – fechar a caixa
- **Figura 10** - Aplicação Armazém – consultar produtos
- **Figura 11** - Aplicação Armazém - atualizar stock
- **Figura 12** - Aplicação Armazém – transferir stock
- **Figura 13** - Aplicação Sede – consultar estatísticas por dia
- **Figura 14** - Aplicação Sede – consultar estatísticas por dia e por bar

Lista de Tabelas

Sem tabelas.

1 Introdução

1.1 Enquadramento

O presente relatório, elaborado no âmbito da UC de Base de Dados da UBI, descreve três aplicações conectadas com uma base de dados. Este projeto é um exemplo prático, que simula uma futura proposta de trabalho, sendo o desenvolvimento de uma situação que se aproxima bastante da realidade (comércio de bens alimentares), sendo assim uma componente essencial de aprendizagem, que permite aos alunos o aperfeiçoamento de boas práticas e o desenvolvimento de perícia para atividades futuras.

1.2 Motivação

A principal motivação para a realização deste projeto foi aplicar e consolidar todo o conhecimento adquirido ao longo de um semestre, e com isto aperfeiçoar boas práticas na área do SQL, assim como fazer o design de aplicações.

1.3 Objetivos

O projeto tem como principal objetivo o desenvolvimento de uma plataforma que torne a informação da “Zurrapa-Drinks & Coffee, Lda.” fácil de gerir e de certa forma, os empregados mais eficientes. Pretende-se que a gerência tenha conhecimento das vendas das suas filiais. Esta aplicação irá permitir à gerência um controlo sobre a parte contabilística do negócio, o que será benéfico para o seu desenvolvimento.

1.4 Organização do documento

A estrutura que sintetiza o trabalho elaborado é a que seguir se descreve:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua realização, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento;
2. O segundo capítulo – **Desenvolvimento de aplicações cliente/servidor** – relata as aplicações cliente/servidor, conhecimento sobre SQL Server, configuração do acesso ao servidor e o ambiente de desenvolvimento (Visual Studio);
3. O terceiro capítulo – **Modelação** – descreve a notação usada, as opções tomadas para situações não especificadas no enunciado, o modelo elaborado e algumas considerações sobre o mesmo;
4. O quarto capítulo – **Aplicações** – menciona a distribuição de tarefas e quem ficou encarregue de as realizar, o acesso à base de dados, e a descrição geral das aplicações;
5. O quinto capítulo – **Conclusões** – retrata o que foi conseguido e os objetivos não atingidos, assim como o seu porquê.

2 Desenvolvimento de aplicações cliente/servidor sobre base de dados

2.1 Introdução

Nas linhas abaixo relata-se o desenvolvimento de aplicações cliente/servidor.

As aplicações cliente/servidor criam a ponte entre os funcionários e o SGBD (e, posteriormente, a base de dados (BD)). O cliente faz um pedido no balcão e a caixa vai realizar os vários pedidos (através de *queries*) aos servidores e aguardar até os receber corretamente.

Todas estas aplicações foram desenvolvidas em Visual Basic, tal como mencionado e desenvolvido nas secções seguintes.

2.2 Aplicações cliente/servidor

Para realizar este projeto foram necessárias 3 aplicações cliente/servidor, cada uma com uma função diferente, de forma a satisfazer distintas tarefas.

A aplicação Caixa permite fazer pedidos, consultar pedidos, fechar a caixa e consultar o stock existente naquele bar.

A aplicação Armazém permite visualizar os produtos que existem (inventário), atualizar o stock do armazém e dos bares, e transferir stock.

A aplicação Sede permite fazer as estatísticas sobre os totais gastos e totais recebidos, de acordo com o dia e o bar.

Todas as aplicações têm *queries* diferentes prontas a enviar ao servidor de forma a receber as informações necessárias e ajustadas a cada uma.

2.3 SQL server

O Microsoft SQL Server é um sistema de gestão de base de dados (SGBD), permite desempenhar as tarefas de armazenamento de e manipulação de dados, fornecendo-os aos utilizadores finais como eles são pedidos, e ainda recuperar todos os dados em caso de falha.

Tem como linguagens de consulta (*queries*) o SQL (*Structured Query Language*), que tem 2 componentes sendo essas a DDL (*Data Definition Language*) - que permite a definição de estruturas de dados e controlo de acesso - e a DML (*Data Manipulation Language*) - que consulta e atualiza os dados.

Nesta ferramenta escrevemos os scripts necessários para criar a base de dados, criar as tabelas e suas restrições e, por fim, inserir alguns dados iniciais, a fim de podermos executar alguns testes nas aplicações por nós desenvolvidas.

2.4 Configuração do acesso ao servidor

Não houve qualquer necessidade de configuração para aceder ao servidor.

2.5 Visual Studio 2022 e Visual Basic

Microsoft Visual Studio 2022 Community é um ambiente de desenvolvimento integrado da Microsoft para desenvolvimento de software especialmente dedicado ao .NET Framework e às linguagens Visual Basic (VB), C, C++, C# (C Sharp) e F# (F Sharp). Esta ferramenta é gratuita e útil para principiantes, visto que é bastante intuitiva.

O Visual Basic é uma linguagem de programação, integrante do pacote Microsoft Visual Studio.

3 Modelação

3.1 Introdução

Os modelos de dados facilitam a interação entre os programadores e os utilizadores finais. Um modelo bem projetado promove uma melhor compreensão da organização.

A definição do modelo de dados é feita a três níveis: conceptual (representação fiel da realidade), lógico (adaptação do modelo conceptual a um modelo de dados específico) e físico (adaptação do modelo logico as características do sistema informático).

As técnicas de modelação dividem-se em dois grupos: do particular para o geral (*Bottom-up*), parte dos atributos e agrupa-os usando dependências funcionais e do geral para o particular (*Top-down*): parte das entidades identificando as suas inter-relações.

A modelação de dados foi a primeira fase realizada neste projeto, para melhor perceber o que era pretendido.

A técnica de modelação usada foi *Top-down*, uma vez que partimos das entidades identificando as suas inter-relações e só depois são especificados os atributos para cada uma.

Para representar o modelo entidade-associação que permitisse representar graficamente as entidades e as suas associações, usou-se a notação de Chen, em que as entidades são representadas por retângulos e as associações por losangos. Estes diagramas ajudam na compreensão da estrutura da base de dados.

3.2 Descrição da organização

A “Zurrapa - Drinks & Coffee, Lda.” tem uma sede que gerir os bares da UBI, que por sua vez têm um armazém, onde existem diferentes produtos, em stock. Os empregados que trabalham na empresa estão atribuídos a um bar que é o seu local de trabalho pré-definido, os empregados podem ser transferidos para bares diferentes dos seus pré-definidos ou até para o armazém. De acordo com esta informação, criámos as seguintes tabelas para o nosso modelo conceptual:

Filial:

- Empregado (id_emp, nome, idade, cargo, id_localPredef);
- Trabalha (hora_Entrada, hora_Saida, id_emp, id_local);
- Local (id_local, tipo_de_local, localizacao, tempo_repor);
- Pedido (id_pedido, dia, hora, estado, total_preco, total_custo, id_emp, id_local);
- Prod_Local (Qty, id_local, id_produto, id_UM);
- Produto (id_produto, designacao, preco_venda_unidade, preco_custo);
- Prod_pedido (quantidade_servida, quantidade_paga, id_pedido, id_produto);
- Unidade (id_UM, designacao);
- Conversão (De_UM, Para_UM, fator);

Sede:

- Filial (id_filial, endereço, designacao, email, telefone, gerente);
- Filial_Dia (id_dia, dia, total_gasto, total_recebido, lucro, id_filial);
- Filial_Dia_Bar (id_bar, nome_bar, dia_bar, total_gasto_bar, total_recebido_bar, lucro_bar, id_filial);

3.3 Modelo Conceptual

Sede:

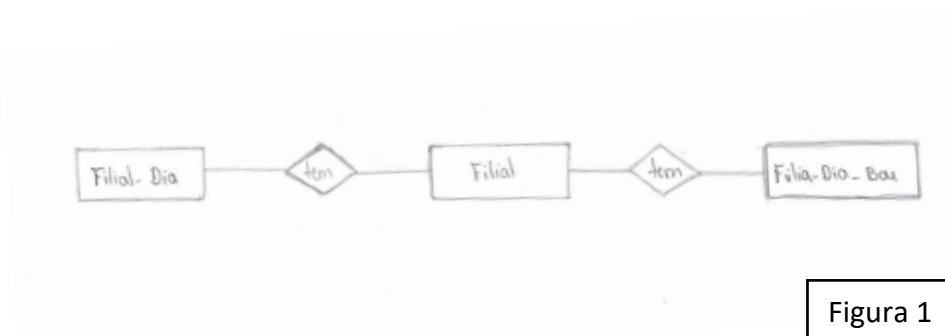


Figura 1

Filial:

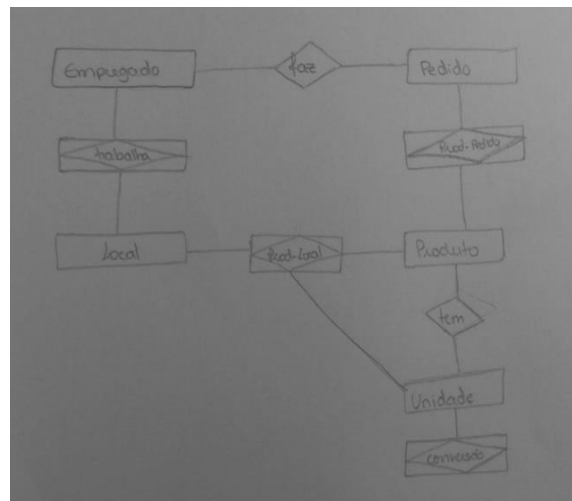


Figura 2

3.4 Modelo Lógico

Filial:

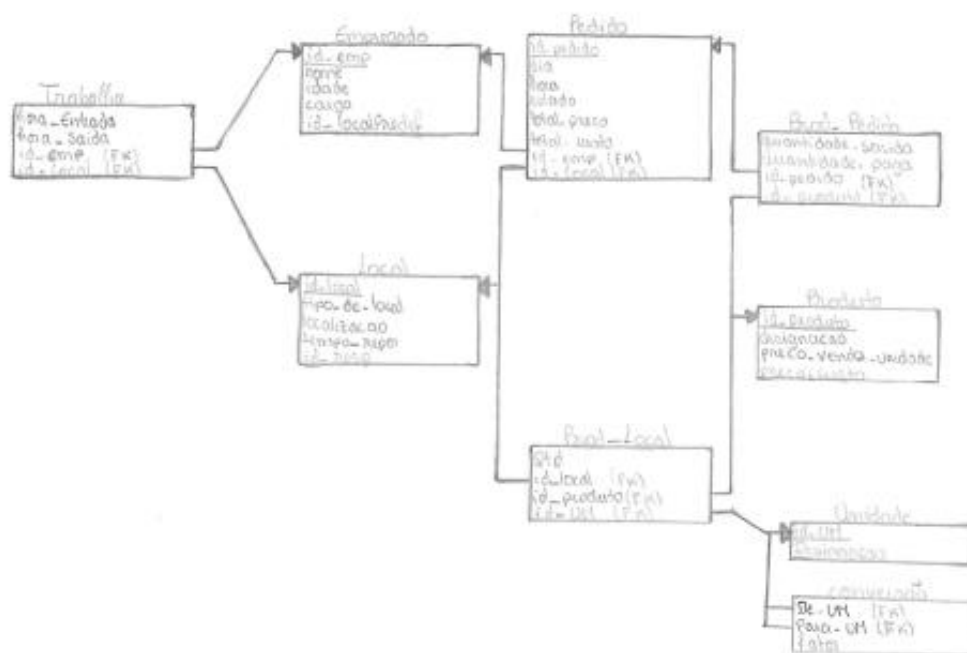


Figura 3

Sede:

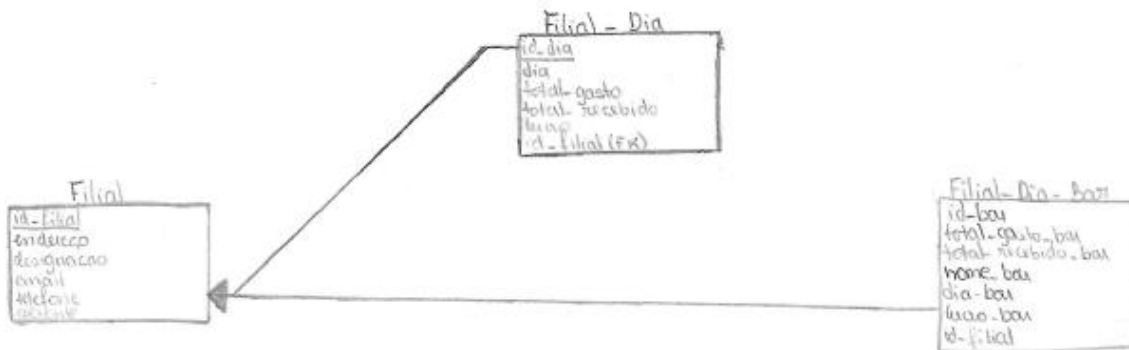


Figura 4

3.5 Considerações

Durante as várias fases da modelação verificámos que este é um processo que causa bastantes alterações no mesmo. Assim, o nosso modelo foi adotando, dia após dia, novas melhorias com raciocínios mais lógicos e mais adequados ao enunciado disponibilizado. Foi através desta melhoria que foram eliminadas as redundâncias e informações não relevantes do enunciado, que anteriormente tinha sido aplicado no diagrama entidade-associação.

4 Aplicações

4.1 Distribuição das tarefas

- I. Elaborar o modelo conceptual.
- II. Produzir o modelo lógico.
- III. Escrever o script SQL para criar a base de dados.
- IV. Escrever o script SQL para criar as diferentes tabelas necessárias e as suas restrições.
- V. Escrever o script SQL para introduzir dados iniciais nas tabelas.
- VI. Desenvolver a aplicação Bar e fazer a ligação com a base de dados.
- VII. Desenvolver a aplicação Armazém e fazer a ligação com a base de dados.
- VIII. Desenvolver a aplicação Sede e fazer a ligação com a base de dados.

4.1.1 Descrição precisa das tarefas

4.1.1.1 *Guilherme Nunes*

Na realização deste projeto fui responsável por desenvolver a aplicação do Bar e colaborei na realização das restantes tarefas, produção dos modelos e elaboração dos scripts.

4.1.1.2 *Maria Pais*

Na realização deste projeto fui responsável por desenvolver a aplicação do Armazém e colaborei na realização das restantes tarefas, produção dos modelos e elaboração dos scripts.

4.1.1.3 *Sara Inácio*

Na realização deste projeto fui responsável por desenvolver a aplicação da Sede e colaborei na realização das restantes tarefas, produção dos modelos e elaboração dos scripts.

4.2 Acesso à base de dados

De forma a conseguir conectar a base de dados com o Visual Studio criámos um script, que está disponibilizado no apêndice G.

4.3 Funcionalidade

4.3.1 Descrição Geral

As 3 aplicações criadas no Visual Basic estão igualmente conectadas à base de dados, desta forma, conseguem aceder aos dados inicialmente.

Depois da ligação estar a funcionar, na aplicação Armazém conseguimos listar todos os produtos existentes, ou produtos específicos através do nome, a fim de saber toda a sobre os mesmos. Nesta aplicação, conseguimos ainda atualizar stocks, transferir produtos e obter o valor do inventário.

Na aplicação Bar, o empregado pode fazer o pedido, consultar os pedidos já existentes, verificar o stock existente e ainda fechar a caixa. Quando faz o pedido, o empregado terá que preencher o ID do pedido, o Produto e a quantidade pretendida. No consultar pedido, o empregado consegue ver todos os pedidos do dia inserido, e ainda procurá-lo pelo ID, assim como modificar o seu estado. No consultar stock, consegue ver todos os produtos e ainda procurar pelo nome. No fecho de caixa, é somado todo o dinheiro recebido ao longo do dia, para posteriormente fazer as estatísticas na sede.

Na aplicação Sede, é possível consultar o total gasto, o total recebido e o lucro por dia ou por bar.

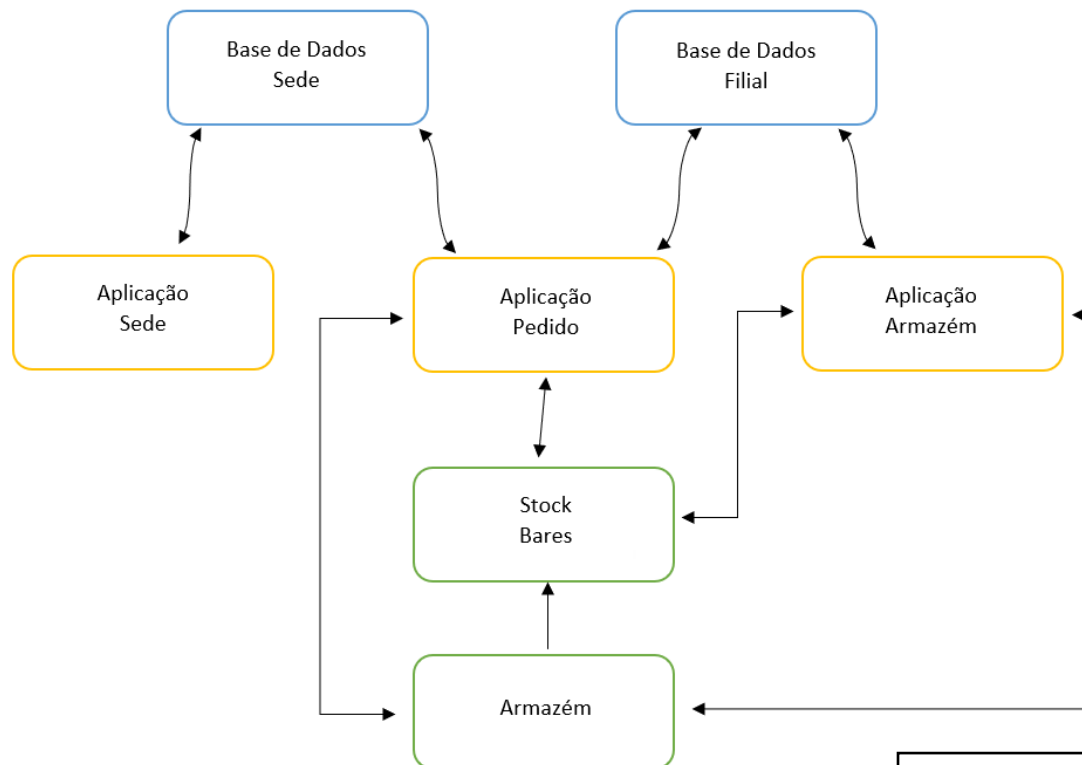


Figura 5

4.3.2 Aplicações

4.3.2.1 Trata Bar

A aplicação Bar permite fazer um pedido, consultar os pedidos feitos, ver o stock do bar e fechar a caixa.

A captura de tela mostra a interface da aplicação 'FzrPedido'. No topo, há um menu 'Local' com uma seta para baixo e um botão 'Procura'. Abaixo, há campos de entrada para 'Dia', 'Hora', 'ID Empregado', 'ID Local', 'Produto', 'Quantidade' e 'Id Pedido'. Há também botões 'Começar', 'Adicionar', 'Finalizar' e 'Atualizar'. À direita, há uma tabela com os seguintes dados:

	Id Pedido	Empregado	Id Emp	Id Local	Produto
▶	1	Emanuel Pacheco	3	1	Pao
	1	Emanuel Pacheco	3	1	Cerveja
	2	Soraia Tavares	4	2	Prego
	2	Soraia Tavares	4	2	Refrigerante
	3	Maria Nunes	9	4	Agua
	4	Joao Alvez	1	1	Pizza
	5	Andre Romero	5	2	Cerveja
	5	Andre Romero	5	2	Prego
	6	Ruben Sousa	7	4	Pao
	1	Emanuel Pacheco	3	1	Pao
	1	Emanuel Pacheco	3	1	Cerveja
	2	Soraia Tavares	4	2	Prego
	2	Soraia Tavares	4	2	Refrigerante
	3	Maria Nunes	9	4	Agua
	4	Joao Alvez	1	1	Pizza

Figura 6

```

Public Sub Finalizar()
    Dim totalCusto As Double
    Dim totalPreco As Double
    Dim idpedido As Integer
    Dim count As Double
    Dim count2 As Double

    idpedido = txtIDPEDIDO.Text
    count = 0
    count2 = 0

    For i As Byte = 0 To dgvPedido.Rows.Count - 1
        If idpedido = (CDBl(dgvPedido.Rows(i).Cells(0).Value)) Then

            totalPreco = (((CDBl(dgvPedido.Rows(i).Cells(5).Value)) * (CDBl(dgvPedido.Rows(i).Cells(7).Value)))) + (count))
            count = totalPreco

        End If
    Next

    For i As Byte = 0 To dgvPedido.Rows.Count - 1
        If idpedido = (CDBl(dgvPedido.Rows(i).Cells(0).Value)) Then

            totalCusto = (((CDBl(dgvPedido.Rows(i).Cells(5).Value)) * (CDBl(dgvPedido.Rows(i).Cells(8).Value)))) + (count2))
            count2 = totalCusto

        End If
    Next

    SQL_Executar("UPDATE Pedido SET total_preco =" & count & " WHERE id_pedido =" & idpedido & "")
    SQL_Executar("UPDATE Pedido SET total_custo =" & count2 & " WHERE id_pedido =" & idpedido & "")

End Sub

```

consultar

Dia

	Id Pedido	Dia	Hora	Estado	Total	Empregado
▶	1	01/01	17:32	0	5,2	Emanuel Pacheco
	2	01/01	12:27	0	4,2	Soraia Tavares
	3	01/01	19:57	0	6,6	Maria Nunes
	4	01/01	14:53	0	5	Joao Alvez
	5	01/01	10:30	0	14,4	Andre Romero
	6	01/01	12:57	0	0,5	Ruben Sousa
	7	01/01	17:32	0	5,2	Emanuel Pacheco
	8	01/01	12:27	0	4,2	Soraia Tavares
	9	01/01	19:57	0	6,6	Maria Nunes
	10	01/01	14:53	0	5	Joao Alvez

ID Pedido 1=aberto
Estado 0=fechado

Figura 7

```

Public Sub LoadGrid()
    cdxItems.Items.Clear()
    cdxItems.Items.Add("01/01")

    SQL.Executar("SELECT p.id_pedido, p.dia, p.hora, p.estado, p.total_preco, e.nome FROM Pedido p, Empregado e WHERE p.id_emp = e.id_emp and p.dia LIKE '%'" & txtSearch.Text & "%'")
    If SQL.HasException(True) Then Exit Sub

    dgvConsultar.DataSource = SQL DRDT
    dgvConsultar.Columns(0).HeaderText = "Id Pedido"
    dgvConsultar.Columns(1).HeaderText = "Dia"
    dgvConsultar.Columns(2).HeaderText = "Hora"
    dgvConsultar.Columns(3).HeaderText = "Estado"
    dgvConsultar.Columns(4).HeaderText = "Total"
    dgvConsultar.Columns(5).HeaderText = "Empregado"

    dgvConsultar.Columns(0).Width = 75
    dgvConsultar.Columns(1).Width = 75
    dgvConsultar.Columns(2).Width = 75
    dgvConsultar.Columns(3).Width = 70
    dgvConsultar.Columns(4).Width = 70
    dgvConsultar.Columns(5).Width = 130
End Sub

Private Sub consultar_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Me.Parent = Form1
    LoadGrid()
End Sub

Private Sub cmdAtualizar_Click(sender As Object, e As EventArgs) Handles cmdAtualizar.Click
    Atualizar()
    LoadGrid()
End Sub

Private Sub Label1_Click(sender As Object, e As EventArgs) Handles Label1.Click
End Sub

Public Sub Atualizar()
    Dim id As Integer
    Dim estado As Integer
    Dim qtd As Integer

    id = txtIdPedido.Text
    estado = txtEstado.Text
    qtd = 0

    SQL.Executar("UPDATE Pedido SET estado = " & estado & " WHERE id_pedido = " & id & " ")
    If SQL.HasException(True) Then Exit Sub
End Sub

```

	id_produto	preco_venda_unidade	Qtd	designacao	fator
▶	1	0,5	5	Sacos de Pao	10
	2	0,6	3	Grades de Aguas	15
	3	0,9	2	Grades de Cerveja	25
	4	1,5	1	Packs de Batatas	25
	5	3	3	Packs de Pregos...	8
	6	1	1	Packs de Pastilhas	11
	7	2,5	2	Packs de Pizzas	12
	8	1,2	2	Grades de Refrig...	25
	1	0,5	3	Sacos de Pao	10
	2	0,6	2	Grades de Aguas	15
	3	0,9	1	Grades de Cerveja	25
	4	1,5	2	Packs de Batatas	25
	5	3	3	Packs de Pregos...	8
	6	1	1	Packs de Pastilhas	11
	7	2,5	2	Packs de Pizzas	12
	8	1,2	2	Grades de Refrig...	25

Valor de Inventario 3097

Figura 8

```

Public Class stock
    Public SQL As New SqlConnection

    2 referências
    Public Sub LoadGrid()
        cbxItems.Items.Clear()
        cbxItems.Items.Add("Sineiro")
        cbxItems.Items.Add("Avenida")
        cbxItems.Items.Add("Sexta Fase")
        cbxItems.Items.Add("Medicina")
        cbxItems.Items.Add("Biblioteca")
        SQL.Executar("SELECT p.id_produto, p.preco_venda_unidade, pl.Qtd, u.designacao, c.fator FROM Produto as p, Prod_Local as pl, Local as l, Unidade as u, Conversao as c
WHERE p.id_produto=pl.id_produto and pl.id_um=u.id_um and u.designacao LIKE '%" & txtProduto.Text & "%' and u.id_um=c.De_um and pl.id_local=l.id_local and l.localizacao LIKE '%" & cbxItems.Text & "%'")
        If SQL.HasException(True) Then Exit Sub
        dgvProdutos.DataSource = SQL.DBDT
        soma()
    End Sub

    0 referências
    Private Sub stock_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Mdiparent = Form1
        LoadGrid()
    End Sub

    1 referência
    Private Sub soma()
        Dim soma As Double

        For i As Byte = 0 To dgvProdutos.Rows.Count - 1
            soma = soma + (Cdbl(dgvProdutos.Rows(i).Cells(1).Value) * Cdbl(dgvProdutos.Rows(i).Cells(2).Value) * Cdbl(dgvProdutos.Rows(i).Cells(4).Value))
        Next
        txtValorInv.Text = soma
    End Sub

    0 referências
    Private Sub btnprocurar_Click(sender As Object, e As EventArgs) Handles btnprocurar.Click
        LoadGrid()
    End Sub
End Class

```

fecho

Dia

Id Bar

	Id Pedido	Dia	Hora	Id Local	Total	Cu
▶	1	01/01	17:32	1	5,2	1.7
	2	01/01	12:27	2	4,2	1.5
	3	01/01	19:57	4	6,6	3.3
	4	01/01	14:53	1	5	2
	5	01/01	10:30	2	14,4	2.9
	6	01/01	12:57	4	0,5	0.2
	7	01/01	17:32	1	5,2	1.7
	8	01/01	12:27	2	4,2	1.5
	9	01/01	19:57	4	6,6	3.3
	10	01/01	14:53	1	5	2
	11	01/01	10:30	2	14,4	2.9
	12	01/01	12:57	4	0,5	0.2
*						

Fechar Caixa

Figura 9

```

Public Sub LoadGrid()
    SQL_Escutar("SELECT p.id_pedido, p.dia, p.hora, p.id_local, p.total_preco, p.total_custo FROM Pedido p")
    If SQL.HasException(True) Then Exit Sub

    dgvConsultar1.DataSource = SQL_DS07
    dgvConsultar1.Columns(0).HeaderText = "Id Pedido"
    dgvConsultar1.Columns(1).HeaderText = "Dia"
    dgvConsultar1.Columns(2).HeaderText = "Hora"
    dgvConsultar1.Columns(3).HeaderText = "Id Local"
    dgvConsultar1.Columns(4).HeaderText = "Total"
    dgvConsultar1.Columns(5).HeaderText = "Custo"

    dgvConsultar1.Columns(0).Width = 100
    dgvConsultar1.Columns(1).Width = 100
    dgvConsultar1.Columns(2).Width = 100
    dgvConsultar1.Columns(3).Width = 100
    dgvConsultar1.Columns(4).Width = 100
    dgvConsultar1.Columns(5).Width = 100
End Sub

Public Sub dia()
    Dim dia As Double
    Dim totalGasto As Double
    Dim totalRecebido As Double
    Dim lucro As Double
    Dim count As Double
    Dim count2 As Double

    dia = diaGrid.Text

    For i As Byte = 0 To dgvConsultar1.Rows.Count - 1
        If CInt(dgvConsultar1.Rows(i).Cells(1).Value) = dia Then
            count = CInt(dgvConsultar1.Rows(i).Cells(5).Value) + (totalGasto)
            totalGasto = count
            count2 = CInt(dgvConsultar1.Rows(i).Cells(6).Value) + (totalRecebido)
            totalRecebido = count2
        End If
    Next
    lucro = totalRecebido - totalGasto

    SQL2_Escutar("INSERT INTO Filial_Dia(dia, total_gasto, total_recebido, lucro, id_filial) VALUES('" & dia & "', " & totalGasto & ", " & totalRecebido & ", " & lucro & ", 1)")
End Sub

Public Sub bar()
    Dim dia As Double
    Dim bar As Double
    Dim totalGasto As Double
    Dim totalRecebido As Double
    Dim lucro As Double
    Dim count As Double
    Dim count2 As Double
    Dim nome As String

    dia = diaGrid.Text
    bar = barGrid.Text

    For i As Byte = 0 To dgvConsultar1.Rows.Count - 1
        If (CInt(dgvConsultar1.Rows(i).Cells(1).Value)) = dia And (CInt(dgvConsultar1.Rows(i).Cells(4).Value)) = bar Then
            count = CInt(dgvConsultar1.Rows(i).Cells(5).Value) + (totalGasto)
            totalGasto = count
            count2 = CInt(dgvConsultar1.Rows(i).Cells(6).Value) + (totalRecebido)
            totalRecebido = count2
        End If
    Next
    If bar = 1 Then
        nome = "Gusta Faze"
    End If
    If bar = 2 Then
        nome = "Modicina"
    End If
    If bar = 3 Then
        nome = "Biblioteca"
    End If
    lucro = totalRecebido - totalGasto

    SQL2_Escutar("INSERT INTO Filial_Bar(id_bar, nome_bar, dia_bar, total_gasto_bar, total_recebido_bar, lucro_bar, id_filial) VALUES('" & bar & "', '" & nome & "', " & dia & "', " & totalGasto & ", " & totalRecebido & ", " & lucro & ", 1)")
End Sub

Private Sub fecho_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Me.Parent = Form1
    LoadGrid()
End Sub

Private Sub fecharcaixa_Click(sender As Object, e As EventArgs) Handles fecharcaixa.Click
    dia()
    bar()
End Sub

```


4.3.2.2 Trata Armazém

Na aplicação Armazém, podemos consultar os produtos, atualizar o stock e transferi-lo, de forma simples.

	id_produto	preco_venda_unid	Qtd	designacao	fato
▶	1	0,5	5	Sacos de Pao	10
	5	3	7	Packs de Pregos...	8

Valor do Inventário: 193

Figura 10

```
Public Class Inventario
    Public SQL As New SqlConnection

    Private Sub LoadGrid()
        cbItems.Items.Clear()
        cbItems.Items.Add("sineiro")
        cbItems.Items.Add("avenida")
        cbItems.Items.Add("sexta fase")
        cbItems.Items.Add("medicina")
        cbItems.Items.Add("biblioteca")
        SQL.Execute("SELECT p.id_produto, u.designacao, p.preco_venda_unidade, pl.qtd, c.fator FROM Produto as p, Prod_Local as pl, Local as l, Unidade as u, Conversao as c
        WHERE p.id_produto=pl.id_produto and pl.id_unid=u.id_unid and u.designacao LIKE '%" & txtProcura.Text & "%' and u.id_unid=pl.id_unid and pl.id_local=l.id_local and l.localizacao LIKE '%" & cbItems.Text & "%'")
        If SQL.HasException(True) Then Exit Sub

        dgv.DataSource = SQL.DBDT
        dgv.Columns(0).HeaderText = "ID Produto"
        dgv.Columns(1).HeaderText = "Nome Produto"
        dgv.Columns(2).HeaderText = "Preço"
        dgv.Columns(3).HeaderText = "Quantidade"
        dgv.Columns(4).HeaderText = "Fator"

        dgv.Columns(0).Width = 75
        dgv.Columns(1).Width = 130
        dgv.Columns(2).Width = 60
        dgv.Columns(3).Width = 70
        dgv.Columns(4).Width = 50
        soma()
    End Sub

    Private Sub Inventario_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MdiParent = Form1
    End Sub

    Private Sub Procurar_Click(sender As Object, e As EventArgs) Handles Procurar.Click
        LoadGrid()
    End Sub

    Private Sub soma()
        Dim soma As Double

        For i As Integer = 0 To dgv.Rows.Count - 1
            soma = soma + (Cdbl(dgv.Rows(i).Cells(2).Value) * Cdbl(dgv.Rows(i).Cells(3).Value) * Cdbl(dgv.Rows(i).Cells(4).Value))
        Next
        txtSoma.Text = soma
    End Sub
End Class
```

Figura 11

```
Public SQL As New SQLControl()
Dim connection As New SqlConnection("Server=LAPTOP-HH1B8343\SQLEXPRESS;Database=ZurrapaFilial;Trusted_Connection=True")

O referências
Private Sub AtualizarArmazem_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    'TODO: esta linha de código carrega dados na tabela 'ZurrapaFilialDataSet.Prod_Local'. Você pode movê-la ou removê-la conforme necessário.
    Me.Prod_LocalTableAdapter.Fill(Me.ZurrapaFilialDataSet.Prod_Local)
    MdiParent = Form1
End Sub

O referências
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim command As New SqlCommand("UPDATE Prod_Local SET Qty= " & txtQTD.Text & " WHERE id_produto=" & txtID.Text & "", connection)

    If txtID.Text <= 0 Or txtQTD.Text <= 0 Or txtID.Text = "" Or txtQTD.Text = "" Or txtArm.Text < 3 Or txtArm.Text > 5 Or txtArm.Text = 4 Then
        MessageBox.Show("Valores Inválidos!")
    Else
        command.Parameters.Add("@num", SqlDbType.Int).Value = txtID.Text
        command.Parameters.Add("@qty", SqlDbType.Int).Value = txtQTD.Text

        connection.Open()

        If command.ExecuteNonQuery() = 1 Then
            MessageBox.Show("Dados Não Atualizados!")
            connection.Close()
        Else
            MessageBox.Show("Dados Atualizados!")
            connection.Close()
        End If
    End If

    Dim id As Integer
    id = txtID.Text
    Dim arm As Integer
    arm = txtArm.Text
    SQL.ExecuteNonQuery("Select pl.* From Produto p, Prod_Local pl Where pl.id_produto=" & id & " and pl.id_local=" & arm & "")
    If SQL.HasException(True) Then Exit Sub
    dgvData.DataSource = SQL.DBDT
End Sub
End Class
```

Figura 12

```
Public Sub LoadGrid()
    SQL.Executar("SELECT l.tipo_de_local, l.localizacao, l.id_local, p.id_produto, u.designacao, p.preco_venda_unidade, pl.qtd FROM Produto p, Prod_Local pl, Local l, Unidade u WHERE p.id_produto=pl.id_produto and pl.id_produto=u.id_un and pl.id_local=l.id_local")
    If SQL.HasException(Now) Then Exit Sub
    dgv.DataSource = SQL.Rows
    dgv.Columns(0).HeaderText = "Tipo Local"
    dgv.Columns(1).HeaderText = "Nome Local"
    dgv.Columns(2).HeaderText = "Codigo do Local"
    dgv.Columns(3).HeaderText = "Codigo Produto"
    dgv.Columns(4).HeaderText = "Nome Produto"
    dgv.Columns(5).HeaderText = "Preco"
    dgv.Columns(6).HeaderText = "Quantidade"

    dgv.Columns(0).Width = 75
    dgv.Columns(1).Width = 75
    dgv.Columns(2).Width = 75
    dgv.Columns(3).Width = 70
    dgv.Columns(4).Width = 130
    dgv.Columns(5).Width = 70
    dgv.Columns(6).Width = 75
End Sub

Imports
Public Sub Tracar()
    Dim id As Integer
    Dim bar As Integer
    Dim arm As Integer
    Dim qtd As Integer
    Dim a As Integer
    Dim soma As Integer
    Dim subtr As Integer

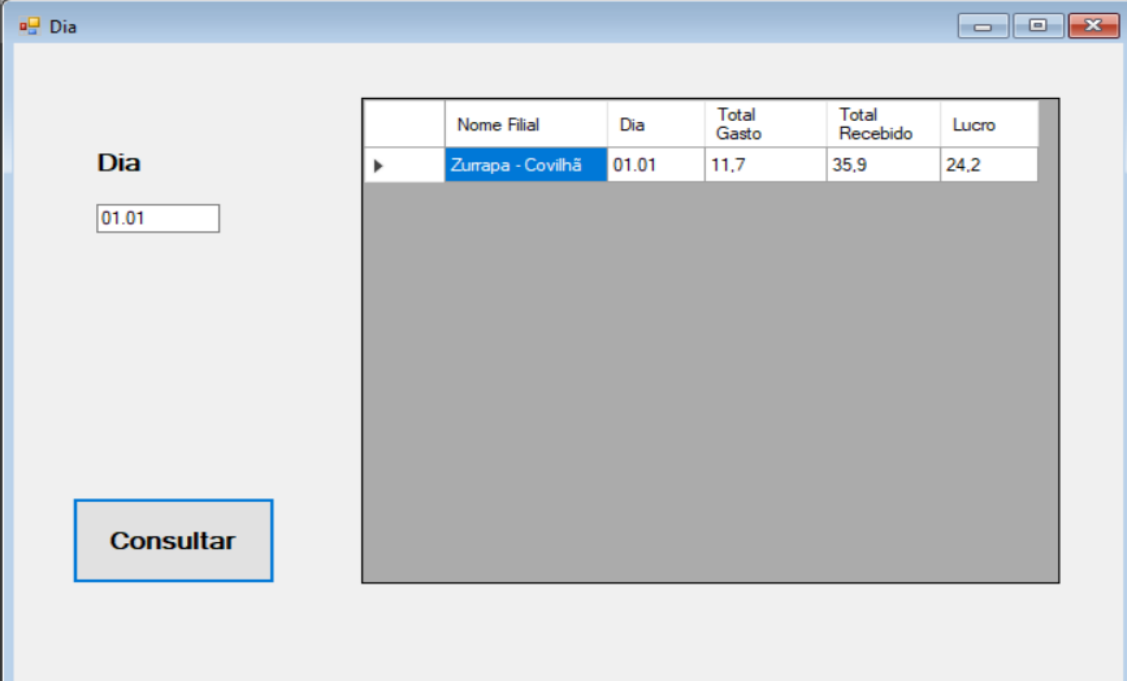
    id = txtID.Text - 1
    bar = txtBar.Text
    arm = txtArm.Text
    qtd = txtQtd.Text

    For i As Byte = 0 To dgv.Rows.Count - 1
        If id = i Then
            a = CInt(dgv.Rows(i).Cells(6).Value)
            End If
        Next

    If a < qtd Then
        MessageBox.Show("Nao temos essa quantidade!")
    Else
        soma = bar + a
        subtr = arm - a
        SQL.Executar("UPDATE Prod_Local SET Prod_Local.Qtd= & subtr & " WHERE Prod_Local.id_produto= & txtID.Text & " and Prod_Local.id_local= & txtArm.Text & ")
        SQL.Executar("UPDATE Prod_Local SET Prod_Local.Qtd= & soma & " WHERE Prod_Local.id_produto= & txtID.Text & " and Prod_Local.id_local= & txtBar.Text & ")
        MessageBox.Show("Transferencia Concluida!")
    End If
End Sub
```

4.3.2.3 Trata Sede

A Sede permite a consulta do total gasto, total recebido e o lucro de cada dia e em cada bar.



	Nome Filial	Dia	Total Gasto	Total Recebido	Lucro
▶	Zurapa - Covilhã	01.01	11,7	35,9	24,2

Figura 13

```
Public Class Dia
    Private SQL As New SQLControl

    0 referências
    Private Sub Dia_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MdiParent = Form1
        LoadGrid()
    End Sub

    2 referências
    Public Sub LoadGrid()
        SQL.Executar("SELECT f.designacao, d.dia, d.total_gasto, d.total_recebido, d.lucro FROM Filial as f, Filial_Dia as d
        WHERE d.id_filial = f.id_filial and d.dia like '%" & txtDia.Text & "%'")

        If SQL.HasException(True) Then Exit Sub
        dgv.DataSource = SQL.DBDT
        dgv.Columns(0).HeaderText = "Nome Filial"
        dgv.Columns(1).HeaderText = "Dia"
        dgv.Columns(2).HeaderText = "Total Gasto"
        dgv.Columns(3).HeaderText = "Total Recebido"
        dgv.Columns(4).HeaderText = "Lucro"

        dgv.Columns(0).Width = 100
        dgv.Columns(1).Width = 60
        dgv.Columns(2).Width = 75
        dgv.Columns(3).Width = 70
        dgv.Columns(4).Width = 60
    End Sub

    0 referências
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        LoadGrid()
    End Sub
End Class
```

The application window 'Bar' displays a search interface. On the left, there are two input fields: 'Dia' with the value '1.01' and 'ID Bar' with the value '4'. Below these is a button labeled 'Consultar'. To the right is a data grid with the following columns: Nome Filial, ID Bar, Nome Bar, Dia, Total Gasto, and Total Recebido. The grid contains one data row: Zurapa - Covilhã, 4, biblioteca, 1.01, 3.5, 7.1.

	Nome Filial	ID Bar	Nome Bar	Dia	Total Gasto	Total Recebido
▶	Zurapa - Covilhã	4	biblioteca	1.01	3.5	7.1

Figura 14

```
Public Class Bar
    Private SQL As New SQLControl

    0 referências
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        LoadGrid()
    End Sub

    2 referências
    Public Sub LoadGrid()
        SQL.Executar("SELECT f.designacao, b.id_bar, b.nome_bar, b.dia_bar, b.total_gasto_bar, b.total_recebido_bar, b.lucro_bar FROM Filial as f, Filial_Dia_Bar as b
        WHERE b.id_filial = f.id_filial and b.id_bar like '%" & txtBar.Text & "%' and b.dia_bar like '%" & txtDia.Text & "%'")

        If SQL.HasException(True) Then Exit Sub
        dgv.DataSource = SQL.DBDT
        dgv.Columns(0).HeaderText = "Nome Filial"
        dgv.Columns(1).HeaderText = "ID Bar"
        dgv.Columns(2).HeaderText = "Nome Bar"
        dgv.Columns(3).HeaderText = "Dia"
        dgv.Columns(4).HeaderText = "Total Gasto"
        dgv.Columns(5).HeaderText = "Total Recebido"
        dgv.Columns(6).HeaderText = "Lucro"

        dgv.Columns(0).Width = 100
        dgv.Columns(1).Width = 60
        dgv.Columns(2).Width = 75
        dgv.Columns(3).Width = 60
        dgv.Columns(4).Width = 75
        dgv.Columns(5).Width = 75
        dgv.Columns(6).Width = 60
    End Sub

    0 referências
    Private Sub Bar_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        LoadGrid()
        MdIParent = Form1
    End Sub
End Class
```

5 Conclusões

Ao realizar este projeto, pensamos ter conseguido atingir os principais objetivos, sendo esses a criação de um modelo concetual e lógico, com determinadas restrições, estas descritas no enunciado disponibilizado, a criação de uma base de dados funcional com tabelas e valores inseridos, três aplicações capazes de fazer gestão do serviço de um bar, de um armazém e de uma sede, uma conexão entre as aplicações, o servidor e a BD, e um relatório com todo o progresso alcançado, bem como os vários passos e programas que foram necessários para que todo o nosso trabalho fosse concretizado.

Devido ao prazo de entrega do trabalho e à acumulação de vários outros trabalhos não foi possível realizar tanto quanto queríamos neste projeto. Alguns itens ficaram por resolver, podendo estes serem resolvidos num trabalho futuro.

Epílogo

Como grupo, achamos que a disciplina teve um impacto positivo nos estudantes, no entanto houve pontos que talvez pudessem ser modificados.

Começando pelos positivos e a manter:

- O contexto do programa lecionado foi adequado à UC e achamos extremamente necessário aprender sobre o funcionamento de base de dados, uma vez que é imprescindível a um bom informático saber trabalhar com elas.

Nos pontos a alterar:

- O caderno autónomo é de grande importância, uma vez que incentiva estudo contínuo da matéria, semanalmente, hábito que é difícil para muitos manter quando não há motivação para tal, especialmente quando existe uma época de avaliações mais “apertada”, com isto seria importante dar mais flexibilidade na realização do mesmo.

Referências Bibliográficas

- “*Notas de apoio às aulas*” – Versão 5.5, J. Muranho, H. Proença, P. Prata, R. Cardoso. 2021-11-02.

- <https://www.youtube.com/watch?v=u9DqI0oAzmk>

- <https://www.youtube.com/watch?v=71ZwwzqzPeA>

- <https://www.youtube.com/watch?v=SAbdkMEHXdA>

- <https://www.youtube.com/watch?v=7Z4BGEHD-JQ>

Anexos

Sem anexos.

Apêndices

Apêndice A – Criar Base de Dados da Filial

```
USE master
IF ( EXISTS( SELECT * FROM [dbo].[sysdatabases] Where name = 'ZurrapaFilial'))
Begin
    DROP DATABASE ZurrapaFilial
end
IF (NOT EXISTS( SELECT * FROM [dbo].[sysdatabases] Where name='ZurrapaFilial'))
Begin
    CREATE DATABASE ZurrapaFilial
    ON
    ( NAME = 'Filial_dat',
    FILENAME = 'C:\Users\sarao\Desktop\Scripts\ZurrapaFilialDb.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
    LOG ON
    ( NAME = 'Filial_log',
    FILENAME = 'C:\Users\sarao\Desktop\Scripts\ZurrapaFilialDb.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB )
End
```

Apêndice B – Criar Base de Dados da Sede

```
USE master
IF ( EXISTS( SELECT * FROM [dbo].[sysdatabases] Where name = 'ZurrapaSede')
)
Begin
    DROP DATABASE ZurrapaSede
end
IF (NOT EXISTS( SELECT * FROM [dbo].[sysdatabases] Where name =
'ZurrapaSede') )
Begin
    CREATE DATABASE ZurrapaSede
    ON
    ( NAME = 'Sede_dat',
    FILENAME = 'C:\Users\guinu\OneDrive\Ambiente de Trabalho\Script\ZurrapaSedeDb.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
    LOG ON
    ( NAME = 'Sede_log',
    FILENAME = 'C:\Users\guinu\OneDrive\Ambiente de Trabalho\Script\ZurrapaSedeDb.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB )
End
```

Apêndice C – Criar tabelas da Filial e restrições

Use ZurrapaFilial

```
if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Produto]'))
begin
    CREATE TABLE Produto (
        id_produto int NOT NULL IDENTITY(1,1)
        CHECK (id_produto >= 1),
        designacao nvarchar(80),
        preco_venda_unidade float NOT NULL,
        preco_custo float NOT NULL,
        CONSTRAINT PK_Produto PRIMARY KEY (id_produto));
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Empregado]'))
begin
    CREATE TABLE Empregado (
        id_emp int NOT NULL IDENTITY(1,1)
        CHECK (id_emp >=1),
        nome nvarchar(80),
        idade int NOT NULL,
        cargo nvarchar(20),
        id_localPredef int NOT NULL,
        CONSTRAINT PK_Empregado PRIMARY KEY (id_emp));
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Unidade]'))
begin
    CREATE TABLE Unidade (
        id_UM int NOT NULL IDENTITY(1,1)
        CHECK (id_UM >=1),
        designacao nvarchar(80),
        CONSTRAINT PK_Unidade PRIMARY KEY (id_UM));
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Local]'))
begin
    CREATE TABLE Local (
        id_local int NOT NULL IDENTITY(1,1)
        CHECK (id_local >=1),
        tipo_de_local nvarchar(80) NOT NULL,
        localizacao nvarchar(80) NOT NULL,
        tempo_repor int,
        id_resp int NOT NULL,
        CONSTRAINT PK_Local PRIMARY KEY (id_local));
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Trabalha]'))
begin
    CREATE TABLE Trabalha (
        hora_Entrada nvarchar(10) NOT NULL,
        hora_Saida nvarchar(10) NOT NULL,
        id_emp INT NOT NULL,
        id_local INT NOT NULL,
        FOREIGN KEY (id_emp)
            REFERENCES Empregado(id_emp)
            ON UPDATE CASCADE

        ON DELETE CASCADE,
        FOREIGN KEY (id_local)
```



```
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Pedido]'))
begin
    CREATE TABLE Pedido (
        id_pedido INT NOT NULL IDENTITY(1,1)
        CHECK (id_pedido >= 1),
        dia nvarchar(80) NOT NULL,
        hora nvarchar(10) NOT NULL,
        estado nvarchar(80) NOT NULL,
        total_preco float NOT NULL,
        total_custo nvarchar(10) NOT NULL,
        id_emp INT NOT NULL,
        id_local INT NOT NULL,
        CONSTRAINT PK_Pedido PRIMARY KEY (id_pedido),
        FOREIGN KEY (id_emp)
            REFERENCES Empregado(id_emp)
            ON UPDATE CASCADE
            ON DELETE CASCADE,
        FOREIGN KEY (id_local)
            REFERENCES Local(id_local),);
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Prod_Pedido]'))
begin
    CREATE TABLE Prod_Pedido (
        quantidade_servida INT NOT NULL,
        quantidade_paga INT NOT NULL,
        id_pedido INT NOT NULL,
        id_produto INT NOT NULL,
        FOREIGN KEY (id_pedido)
            REFERENCES Pedido(id_pedido)
            ON UPDATE CASCADE,
        FOREIGN KEY (id_produto)
            REFERENCES Produto(id_produto),);
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Prod_Local]'))
begin
    CREATE TABLE Prod_Local (
        Qtd INT NOT NULL,
        id_local INT NOT NULL,
        id_produto INT NOT NULL,
        id_UM INT NOT NULL,
        FOREIGN KEY (id_local)
            REFERENCES Local(id_local)
            ON UPDATE CASCADE
            ON DELETE NO ACTION,
        FOREIGN KEY (id_produto)
            REFERENCES Produto(id_produto),
        FOREIGN KEY (id_UM)
            REFERENCES Unidade(id_UM),);
end

if not exists (select * from dbo.sysobjects
               where id = object_id(N'[dbo].[Conversao]'))
begin
    CREATE TABLE Conversao (
        De_UM INT NOT NULL,
        Para_UM INT NOT NULL,
```

```
fator INT NOT NULL,  
FOREIGN KEY (De_UM)  
REFERENCES Unidade(id_UM)  
ON UPDATE CASCADE,  
FOREIGN KEY (Para_UM)  
REFERENCES Unidade(id_UM));
```

End

Apêndice D – Criar tabelas da sede e restrições

Use ZurrapaSede

```
if not exists (select * from dbo.sysobjects  
               where id = object_id(N'[dbo].[Filial]'))  
begin  
    CREATE TABLE Filial (  
        id_filial INT NOT NULL IDENTITY(1,1)  
        CHECK (id_filial >=1),  
        endereco nvarchar(80) NOT NULL,  
        designacao nvarchar(80) NOT NULL,  
        email nvarchar(80) NOT NULL,  
        telefone nvarchar(30) NOT NULL,  
        gerente nvarchar(80) NOT NULL,  
        CONSTRAINT PK_Filial PRIMARY KEY (id_filial),;  
end  
  
if not exists (select * from dbo.sysobjects  
               where id = object_id(N'[dbo].[Filial_Dia]'))  
begin  
    CREATE TABLE Filial_Dia (  
        id_dia INT NOT NULL IDENTITY(1,1)  
        CHECK (id_dia >=1),  
        dia nvarchar(80) NOT NULL,  
        total_gasto FLOAT NOT NULL,  
        total_recebido FLOAT NOT NULL,  
        lucro FLOAT NOT NULL,  
        id_filial INT NOT NULL,  
        CONSTRAINT PK_Filial_Dia PRIMARY KEY (id_dia),  
        FOREIGN KEY (id_filial)  
        REFERENCES Filial(id_filial)  
        ON UPDATE CASCADE,);  
end  
  
if not exists (select * from dbo.sysobjects  
               where id = object_id(N'[dbo].[Filial_Dia_Bar]'))  
begin  
    CREATE TABLE Filial_Dia_Bar (  
        id_db INT NOT NULL IDENTITY (1,1)  
        CHECK (id_db >=1),  
        id_bar INT NOT NULL,  
        nome_bar nvarchar(80) NOT NULL,  
        dia_bar nvarchar(50) NOT NULL,  
        total_gasto_bar FLOAT NOT NULL,  
        total_recebido_bar FLOAT NOT NULL,  
        lucro_bar FLOAT NOT NULL,  
        id_filial INT NOT NULL,  
        CONSTRAINT PK_Filial_Dia_Bar PRIMARY KEY (id_db),  
        FOREIGN KEY (id_filial)  
        REFERENCES Filial(id_filial));  
end
```

Apêndice E – Inserir dados iniciais da Filial

```
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Joao Alvez',  
29, 'caixa', 1)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Claudia Brito',  
35, 'balcao', 2)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Emanuel  
Pacheco', 26, 'balcao', 1)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Soraia  
Tavares', 47, 'balcao', 2)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Andre  
Romero', 45, 'caixa', 2)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Sara  
Mourato', 59, 'caixa', 1)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Ruben  
Sousa', 23, 'balcao', 4)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Guilherme  
Pais', 29, 'balcao', 3)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Maria  
Nunes', 37, 'caixa', 4)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Eduardo  
Amarelo', 28, 'balcao', 4)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Tiago  
Relvado', 46, 'caixa', 5)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'André Pais',  
25, 'balcao', 3)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Miguel  
Pinheiro', 51, 'balcao', 3)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Diogo  
Ferreira', 31, 'balcao', 5)  
INSERT INTO Empregado( nome, idade, cargo, id_localPredef) VALUES ( 'Muhammed  
Habib', 43, 'balcao', 5)
```

```
INSERT INTO Local(tipo_de_local, localizacao, tempo_repor, id_resp) VALUES('Bar',  
'Sexta Fase',10, 6)  
INSERT INTO Local(tipo_de_local, localizacao, tempo_repor, id_resp) VALUES('Bar',  
'Medicina',15, 2)  
INSERT INTO Local(tipo_de_local, localizacao, tempo_repor, id_resp)  
VALUES('Armazem', 'Sineiro',NULL, 8)  
INSERT INTO Local(tipo_de_local, localizacao, tempo_repor, id_resp) VALUES('Bar',  
'Biblioteca',7, 10)  
INSERT INTO Local(tipo_de_local, localizacao, tempo_repor, id_resp) VALUES(  
'Armazem', 'Avenida',NULL, 11)
```

```
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('12:00',  
'20:30', 6, 1)  
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('8:00',  
'20:00', 1, 1)
```

```
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('8:00',  
'20:00', 3, 1)
```

```
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local)
VALUES('11:00', '20:30', 2, 2)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('9:00',
'20:00', 4, 2)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('9:00',
'20:00', 5, 2)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('8:00',
'19:30', 8, 3)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('7:00',
'19:00', 12, 3)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('7:00',
'19:00', 13, 3)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('10:30',
'20:00', 10, 4)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('10:00',
'19:30', 7, 4)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('10:00',
'19:30', 9, 4)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('7:30',
'19:00', 11, 5)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('7:00',
'19:00', 14, 5)
INSERT INTO Trabalha(hora_Entrada, hora_Saida, id_emp, id_local) VALUES('7:00',
'19:00', 15, 5)
```

```
INSERT INTO Pedido(dia, hora, estado, total_preco, total_custo, id_emp, id_local)
VALUES('01/01', '17:32', '0', 5.2, 1.75, 3, 1)
INSERT INTO Pedido(dia, hora, estado, total_preco, total_custo, id_emp, id_local)
VALUES('01/01', '12:27', '0', 4.20, 1.5, 4, 2)
INSERT INTO Pedido(dia, hora, estado, total_preco, total_custo, id_emp, id_local)
VALUES('01/01', '19:57', '0', 6.60, 3.3, 9, 4)
INSERT INTO Pedido(dia, hora, estado, total_preco, total_custo, id_emp, id_local)
VALUES('01/01', '14:53', '0', 5.0, 2, 1, 1)
INSERT INTO Pedido(dia, hora, estado, total_preco, total_custo, id_emp, id_local)
VALUES('01/01', '10:30', '0', 14.4, 2.95, 5, 2)
INSERT INTO Pedido(dia, hora, estado, total_preco, total_custo, id_emp, id_local)
VALUES('01/01', '12:57', '0', 0.50, 0.2, 7, 4)
```

```
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(5, 1, 1, 1)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 1, 2, 2)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 1, 3, 3)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(1, 1, 4, 4)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 1, 5, 5)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(1, 1, 6, 6)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 1, 7, 7)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 1, 8, 8)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 2, 1, 1)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 2, 2, 2)
```

```
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(1, 2, 3, 3)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 2, 4, 4)
```

Base de Dados

```
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 2, 5, 5)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 2, 6, 6)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 2, 7, 7)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(1, 2, 8, 8)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 4, 1, 1)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(1, 4, 2, 2)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 4, 3, 3)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(0, 4, 4, 4)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 4, 5, 5)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(3, 4, 6, 6)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(1, 4, 7, 7)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(2, 4, 8, 8)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(5, 3, 1, 1)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(6, 3, 2, 2)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(6, 3, 3, 3)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(5, 3, 4, 4)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(7, 3, 5, 5)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(8, 3, 6, 6)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(7, 3, 7, 7)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(6, 3, 8, 8)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(8, 5, 1, 1)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(6, 5, 2, 2)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(6, 5, 3, 3)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(8, 5, 4, 4)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(6, 5, 5, 5)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(7, 5, 6, 6)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(5, 5, 7, 7)
INSERT INTO Prod_Local(Qtd, id_local, id_produto, id_UM) VALUES(7, 5, 8, 8)
```

```
INSERT INTO Unidade(Designacao) VALUES('Sacos de Pao')
INSERT INTO Unidade(Designacao) VALUES('Grades de Aguas')
INSERT INTO Unidade(Designacao) VALUES('Grades de Cerveja')
INSERT INTO Unidade(Designacao) VALUES('Packs de Batatas')
INSERT INTO Unidade(Designacao) VALUES('Packs de Pregos no Pao')
INSERT INTO Unidade(Designacao) VALUES('Packs de Pastilhas')
INSERT INTO Unidade(Designacao) VALUES('Packs de Pizzas')
INSERT INTO Unidade(Designacao) VALUES('Grades de Refrigerantes')
INSERT INTO Unidade(Designacao) VALUES('Unidade')
```

```
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(1, 9, 10)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(2, 9, 15)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(3, 9, 25)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(4, 9, 25)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(5, 9, 8)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(6, 9, 11)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(7, 9, 12)
INSERT INTO Conversao(De_um, Para_um, fator) VALUES(8, 9, 25)
```

```
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo)
VALUES( 'Pao', 0.50, 0.20)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo)
VALUES('Agua', 0.60, 0.30)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo) VALUES(
'Cerveja', 0.90, 0.25)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo) VALUES(
'Batatas', 1.50, 0.33)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo)
VALUES( 'Prego', 3, 1.10)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo) VALUES(
'Pastilhas', 1, 0.45)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo)
VALUES( 'Pizza', 2.50, 1)
INSERT INTO Produto(designacao, preco_venda_unidade, preco_custo) VALUES(
'Refrigerantes', 1.20, 0.40)
```

```
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(5, 5, 1, 1)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(3, 3, 1, 3)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(1, 1, 2, 5)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(1, 1, 2, 8)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(11, 11, 3, 2)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(2, 2, 4, 7)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(3, 3, 5, 3)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(2, 2, 5, 5)
INSERT INTO Prod_Pedido(quantidade_servida, quantidade_paga, id_pedido,
id_produto) VALUES(1, 1, 6, 1)
```

Apêndice F – Inserir dados iniciais da Sede

```
INSERT INTO Filial(endereco, designacao, email, telefone, gerente) VALUES('nº 1 da  
Avenida Montes Hermínios, Covilhã', 'Zurrapa - Covilhã', 'zurrapa_covilha@gmail.com',  
'222 334 569', 'João Muranho')
```

```
INSERT INTO Filial_Dia(dia, total_gasto, total_recebido, lucro, id_filial) VALUES('01.01',  
11.7, 35.9, 24.2, 1)
```

```
INSERT INTO Filial_Dia_Bar(id_bar, nome_bar, dia_bar, total_gasto_bar,  
total_recebido_bar, lucro_bar, id_filial) VALUES(1,'sexta fase', 01.01, 3.75, 10.20, 6.45,  
1)
```

```
INSERT INTO Filial_Dia_Bar(id_bar, nome_bar, dia_bar, total_gasto_bar,  
total_recebido_bar, lucro_bar, id_filial) VALUES(2,'medicina', 01.01, 4.45, 18.60,  
14.15, 1)
```

```
INSERT INTO Filial_Dia_Bar(id_bar, nome_bar, dia_bar, total_gasto_bar,  
total_recebido_bar, lucro_bar, id_filial) VALUES(4, 'biblioteca', 01.01, 3.5, 7.10, 3.60, 1)
```

Apêndice G – Script de Apoio para conectar a base de dados ao Visual Basic

```
Imports System.Data.SqlClient  
Public Class SQLControl  
    Private DBCon As New SqlConnection("Server=LAPTOP-  
68AH0PGS\SQLEXPRESS;Database=ZurrapaSede;Trusted_Connection=True")  
    Private DBCmd As SqlCommand  
    'DB DATA  
    Public DBDA As SqlDataAdapter  
    Public DBDT As DataTable  
    'QUERY PARAMETERS  
    Public Params As New List(Of SqlParameter)  
    'QUERY STATISTICS  
    Public RecordCount As Integer  
    Public Exception As String  
    Public Sub New()  
    End Sub  
    'ALLOW CONNECTION STRING OVERRIDE  
    Public Sub New(ConnectionString As String)  
        DBCon = New SqlConnection(ConnectionString)  
    End Sub  
    'EXECUTE QUERY SUB  
    Public Sub Executar(Query As String)  
        ' RESET QUERY STATS  
        RecordCount = 0  
        Exception = ""  
        Try  
            DBCon.Open()  
            'DATABASE COMAND  
            ' CREATE DB COMAND  
            DBCmd = New SqlCommand(Query, DBCon)  
            ' LOAD PARAMS INTO DB COMAND  
            Params.ForEach(Sub(p) DBCmd.Parameters.Add(p))  
            'CLEAR PAARAM LIST  
            Params.Clear()  
            'EXECUTE COMAND AND FILL DATASET  
            DBDT = New DataTable  
            DBDA = New SqlDataAdapter(DBCmd)  
            RecordCount = DBDA.Fill(DBDT)
```



```
Catch ex As Exception
    'CAPTURE ERROR
    Exception = "ExecQuery Error : " & vbNewLine & ex.Message
Finally
    'CLOSE CONNECTION
    If DBCon.State = ConnectionState.Open Then DBCon.Close()
End Try
End Sub
'ADD PARMS
Public Sub AddParam(Name As String, Value As Object)
    Dim NewParam As New SqlParameter(Name, Value)
    Params.Add(NewParam)
End Sub
'ERROR CHECKING
Public Function HasException(Optional ByVal report As Boolean = False) As Boolean
    If String.IsNullOrEmpty(Exception) Then Return False
    If report = True Then MsgBox(Exception, MsgBoxStyle.Critical, "Exception:")
    Return True
End Function
End Class
```