IPV-Instituto Politécnico de Viseu

ESTGV-Escola Superior de Tecnologia e Gestão de Viseu

DI - Departamento de Informática

Relatório

Trabalho Prático – Gestão de Restaurantes da empresa R&R

Bases de Dados II

Docentes:

- Paulo Tomé
- José Henriques
- Pedro Martins

Trabalho elaborado por:

- Alexandre Rodrigues, 16063
- Inês Figueiredo, 16789
- João Martinho, 16086

Viseu, 2020

IPV-Instituto Politécnico de Viseu

ESTGV-Escola Superior de Tecnologia e Gestão de Viseu

DI - Departamento de Informática

Relatório - Trabalho Prático

Licenciatura em Engenharia Informática

Gestão de Restaurantes da empresa R&R

Base de Dados II

Ano Letivo: 2019/2020

Docentes:

- Paulo Tomé
- José Henriques
- Pedro Martins

Trabalho elaborado por:

- Alexandre Rodrigues, 16063
- Inês Figueiredo, 16789
- João Martinho, 16086

I. Índice

I. Índice II

II. Índice de Figuras III

1. Introdução 1

2. Dicionário de Dados 3

3. Modelo de Dados 7

4. Objetos Criados 9

4.1. Procedimentos 9

4.2. Vistas 11

4.3. Funções 12

4.4. *Triggers* 14

4.5. XML 15

5. Aplicação Web 17

5.1. Funcionamento da aplicação 17

5.2. Desenvolvimento da aplicação 22

6. Conclusão 25

7. Referências 26

8. Anexos 27

II. Índice de Figuras

Figura 1 - Modelo Físico de Dados 7

Figura 2 - Inserção de alergias 9

Figura 3 - Alteração de dados na tabela alergias 9

Figura 4 - Procedimento de remoção de alergia 10

Figura 5 - View referente à tabela de alergias 11

Figura 6 - View referente a todos os consumos registados 11

Figura 7 - View que visualiza todas as ementas registadas de cada restaurante 11

Figura 8 - Função Read para a tabela alergias 12

Figura 9 - Função repor stock de um dado item 12

Figura 10 - Função que determina a quantidade média consumida 13

Figura 11 - Função que determina o funcionário com mais consumos 14

Figura 12 - Trigger para atualizar stock de um item na ementa 14

Figura 13 - Trigger para as ementas só serem criadas ao domingo 15

Figura 14 - Query em XML sobre as Ementas dos Restaurantes 16

Figura 15 - Diagrama da Interação do Cliente com a Aplicação Web 17

Figura 16 - Páginas Inserir, Alterar e Apagar Dados 18

Figura 17 - Exemplos de Páginas de Procedimentos CRUD 19

Figura 18 - Inserção na Tabela Ementas - App Web 20

Figura 19 - Todas as Tabelas - Views App Web 20

Figura 20 - Página Média Consumo 21

Figura 21 - Página XML da Aplicação Web 21

Figura 22 - Ligação à base de dados pela aplicação - psycopg2 22

Figura 23 - Rotas entre páginas - App Web 23

Figura 24 - Código da página layout.html 24

III. Índice de Anexos

Anexo A – Modelo Físico de Dados 27

Anexo B - Diagrama da Interação Cliente - App Web 28

1. Introdução

O presente relatório feito no âmbito da Unidade Curricular de Base de Dados II, lecionada durante a Licenciatura em Engenharia Informática, na Escola Superior de Tecnologias e Gestão de Viseu, do Instituto Politécnico de Viseu.

Neste trabalho prático o pretendido é desenvolver uma aplicação Web capaz de gerir os dados e informações relativos a uma cadeia de restaurantes da empresa R&R, tendo em conta que esta empresa tem restaurantes em diversas cidades, vilas e aldeias de Portugal. Cada um dos restaurantes possui uma ementa diferente para cada refeição (almoço ou jantar) e dia da semana, sendo estas criadas no início da semana. Cada ementa está organizada por diferentes grupos como, por exemplo, bebidas, prato de peixe, entre outros. Os clientes podem consumir tanto ao balcão como na mesa.

O software utilizado para desenvolvimento do modelo de dados foi o *PostgreSQL*, que é uma tecnologia de gestão de base de dados de código aberto. Esta tecnologia é a primeira que implementa o recurso MVCC (controlo de concorrência em várias versões), mesmo antes do Oracle, onde este recurso é conhecido como isolamento de captura instantânea no Oracle. Além disso, permite adicionar funções personalizadas desenvolvidas utilizando diferentes linguagens de programação como C/C++, Java, entre outros. (PostgreSQL Tutorial, 2019)

Relativamente ao desenvolvimento da aplicação Web, criou-se uma API em *python*, responsável por interligar a base de dados (*PostgreSQL*) e a página web. A página web foi desenvolvida em HTML, com recurso a CSS e *Javascript*.

O *python* é uma linguagem de programação utilizada (1) num servidor para criar aplicações web, (2) junto com o software para criar fluxos de trabalho, (3) para a conexão entre sistemas de base de dados, podendo ler e modificar ficheiros, e (4) para lidar em *big data* e executar matemática complexa. Esta linguagem foi preparada para facilitar a leitura utilizando a linha abaixo para concluir um comando, ao contrário das outras linguagens que utilizam, por exemplo, o ponto e vírgula. (w3schools.com, 1999-2019)

No primeiro capítulo do relatório abordar-se-á o modelo de dados que foi desenvolvido pelo grupo, bem como a explicação do raciocínio do mesmo. No segundo capítulo analisar-se-

ão todos os objetos criados em *PostgreSQL*, que foram auxiliares ao sucesso do presente trabalho prático.

Por último explicar-se-á o desenvolvimento da aplicação web na linguagem de programação de *python*, bem como o funcionamento da mesma.

2. Dicionário de Dados

De forma a fornecer uma melhor perceção e compreensão do trabalho realizado, construiu-se a Tabela 1, onde se pode visualizar todas as tabelas criadas na Base de Dados e quais os seus atributos.

Tabela	Atributos	Tipo de Dados	Descrição
Restaurantes	ID_Restaurante	Number	Identificador de cada tipo de restaurante
	Horário	Date & Time	Horário de cada restaurante
	Nome	Text	Nome de cada restaurante
	Contacto	Number	Contacto de cada restaurante
	Morada	Text	Morada de cada restaurante
Ementas	ID_Ementa	Number	Identificador de cada tipo de ementa
	Descriçao	Text	Descrição de cada tipo de ementa
Datas_emendtas	ID_DatasEmendtas	Number	Identificador de cada DatasEmendtas
	Designacao	text	Descrição de cada datasEmenta
Alergias	IDAlergia	Number	Identificador de cada Alergia
	Designcao	Text	Descrição de cada Alergia
Produtos	ID_Produtos	Number	Identificador de cada produto
	Validade	Date	Validade de cada produto
	Quantidade	Number	Quantidade de produtos
Locais	ID_Local	Number	Identificador de cada local
	Designacao	Text	Descrição de cada local

Locais_Venda	ID_LocaiVendas	Number	Identificador de cada local de venda
	Descrição	Text	Descrição de cada local de venda
Vendas	ID_Vendas	Number	Identificador de cada venda
	Data	Date	Data de cada venda
Funcionários	ID_Funcionários	Number	Identificador de cada funcionario
	Nome	Text	Nome de cada funcionário
	Contacto	Number	Contacto de cada funcionário
	Idade	Number	Idade de cada funcionário
Cargo	ID_Cargos	Number	Identificador de cada cargo
	Descrição	Text	Descrição de cada cargo
	Função	Text	Função de cada cargo
	Nome	Text	Nome de cada cliente
	Idade	Number	Idade de cada cliente
	Contacto	Number	Contacto de cada cliente
	Morada	Text	Morada de cada cliente

Tabela 1 - Dicionário dos atributos das tabelas

Modelo de Dados

De forma a auxiliar a compreensão do modelo de dados desenvolvido para esta aplicação o Modelo Físico de Dados presente na Figura 1. Como se pode observar, criaram-se doze tabelas com chaves forasteiras inseridas. Este modelo de dados encontra-se disponível com maior detalhe nos Anexos, nomeadamente no Anexo A.

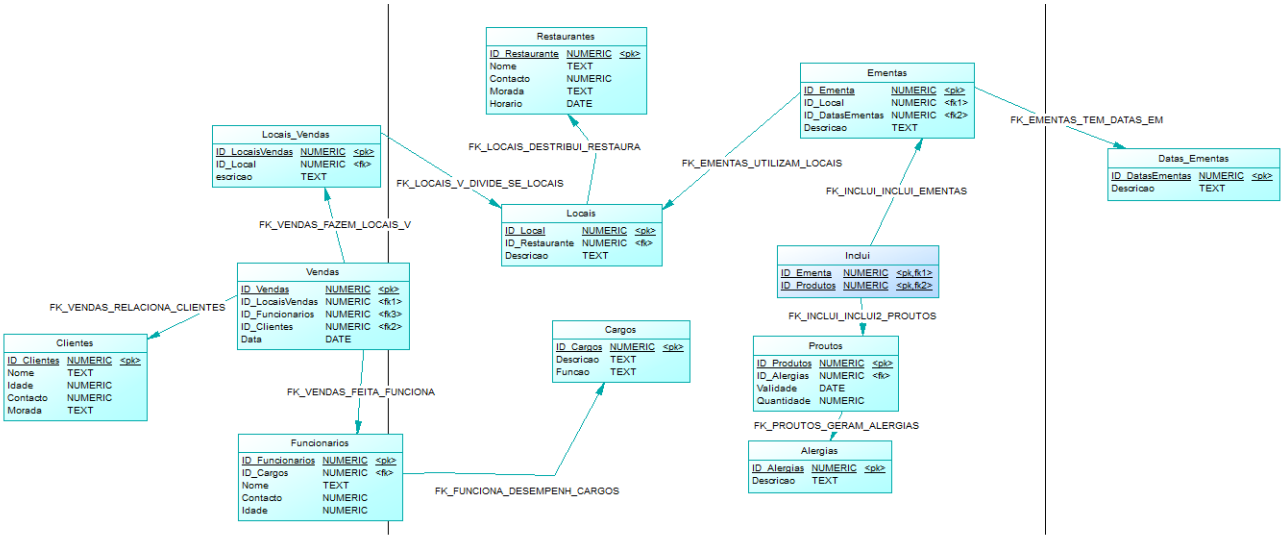


Figura 1 - Modelo Físico de Dados

Como o enunciado referiu e como mencionado na Introdução, existem vários restaurantes dentro da empresa R&R e, visto que é possível existirem restaurantes da mesma cadeia com nomes diferentes, criou-se a tabela "Restaurantes". Além disso, podem possuir localizações diferentes (cidades, vilas), daí a criação da tabela "Locais".

Cada local, de um dado restaurante, tem a sua própria ementa (tabela "Ementas"). Além disso, é atribuída, a cada ementa, uma data (tabela "DatasEmenta") e os produtos correspondentes (tabela "Produtos"), com a informação da quantidade em stock de cada produto em cada ementa (tabela "Inclui").

A tabela "Produtos" também possui a designação de quais as alergias associadas (tabela "Alergias").

Na tabela "Clientes" tem-se acesso aos dados essenciais de cada cliente, de forma a que seja possível efetuarem uma compra, isto é, "consumir" (tabela "Vendas") uma dada ementa, seja este consumo efetuado ao balcão ou à mesa (tabela "LocaisVendas", com os locais de consumo

que cada restaurante possui). Ao ser efetuado o consumo, o cliente pode ser atendido por um dado funcionário (tabela “Funcionarios”), onde se encontra armazenada a informação de todos os funcionários e quais os cargos desempenhados (tabela “Cargos”).

3. Objetos Criados

A criação de objetos numa base de dados é bastante vantajosa na medida em que torna o sistema de base de dados mais dinâmico e simples de inserir, ler, escrever ou remover. Assim, foram criados diversos objetos, entre eles, procedimentos, vistas, *triggers* e funções.

3.1. Procedimentos

Neste subcapítulo serão abordados os procedimentos armazenados que foram construídos, por forma a realizar todas as funções CRUD. Primeiramente, começou-se por criar um procedimento armazenado de inserção para todas as funções referidas no capítulo 3, como se demonstra na Figura 2. Este procedimento recebe como parâmetros de entrada todos os atributos de cada uma das tabelas exceto o identificador, pois esse será gerado automaticamente e incrementado, posteriormente, na inserção seguinte, uma vez que, o identificador é do tipo SERIAL.

```
CREATE OR REPLACE PROCEDURE inserir_alergias(nome varchar)
LANGUAGE SQL
AS $$
    INSERT INTO Alergias VALUES (nextval('IncrementaAlergia'), nome);
$$;
```

Figura 2 - Inserção de alergias

De seguida, criou-se um procedimento para a alteração de dados de cada tabela. Este, como se verifica na Figura 3, recebe o identificador de cada tabela e os atributos que serão alterados, ou seja, numa dada tabela, os atributos são alterados consoante o identificador recebido, assumindo os valores fornecidos, não alterando o identificador. Neste procedimento, existiram algumas tabelas em que não faria sentido alterar determinados atributos, tais como, o NIF na tabela CLIENTES, visto este atributo ser algo que se mantém inalterado durante a vida de um indivíduo.

```
CREATE OR REPLACE PROCEDURE atualiza_alergias(id int, nome varchar)
C:\Users\João Martinho\Desktop\Universidade\3°ano\BD\Procedimentos_Update.sql - not connected
AS $$
            UPDATE Alergias
            Set Alergia=nome
            Where ID_Alergia=id;
$$;
```

Figura 3 - Alteração de dados na tabela alergias

O último procedimento armazenado criado correspondeu ao procedimento de remoção de uma linha de cada tabela, o qual apenas recebe o identificador da tabela e procede à remoção, fazendo a verificação se o atributo é chave estrangeira numa outra tabela e se existe na tabela de onde se pretende remover, através da linha *"WHERE idtipoitem NOT IN (SELECT i_idtipoitem FROM Itens WHERE i_idtipoitem=id) and idtipoitem IN (SELECT idtipoitem FROM TipoItens WHERE idtipoitem=id);"* da Figura 4.

```
CREATE OR REPLACE PROCEDURE apagar_Alergias(id int) AS $$
        BEGIN
            DELETE FROM Alergias
            WHERE ID_Alergia NOT IN (SELECT Produto_IDAlergia FROM Produtos WHERE Produto_IDAlergia =id) and
            ID_Alergia IN (SELECT ID_Alergia FROM Alergias WHERE ID_Alergia =id);
        END; $$
        LANGUAGE plpgsql;
```

Figura 4 - Procedimento de eliminação de alergia

3.2. Vistas

Relativamente a vistas ou *views*, criou-se, primeiramente, uma vista para cada tabela, com todas as informações nela armazenadas, como se pode visualizar na Figura 5, fazendo apenas um *Select* à tabela que se pretende criar a vista.

```
--Alergias
CREATE VIEW VizualizaAlergias
    AS
    SELECT *
    from Alergias;
```

Figura 5 - View referente à tabela de alergias

Uma segunda vista foi criada, com o intuito de facilitar a análise dos consumos, armazenando, assim, apenas o nome do cliente, o nome do funcionário que registou o pedido, a designação do produto consumido e o custo total da fatura, como se observa na Figura 6.

```
CREATE VIEW VizualizaTodosConsumos (nomeCliente, nomeFuncionario, Ementa, TotaPago)
    AS
    SELECT cli.C_Nome,F_Nome, I_Designcao,CustoTotal
    From Consumos c join Clientes cli
        on c.C_IDCliente=cli.IDCliente join Funcionario f
            on c.C_IDFuncionario=f.IDFuncionario join Ementas e
                on c.C_IDEmenta=e.IDEmenta join ItensEmentas it
                    on e.IDEmenta=it.IE_IDEmenta join Itens i
                        on i.IDItem=it.IE_IDItem
```

Figura 6 - View referente a todos os consumos registados

A última vista foi gerada com o mesmo intuito do que a anterior, mas relativa às ementas, apresentando em detalhe as ementas criadas com os atributos relativos ao nome do restaurante, à data da ementa, à designação do produto, ao custo da ementa e à designação da alergia atribuída ao produto, como se analisa na Figura 7.

```
CREATE VIEW VizualizaEmentasCompletas(Restaurante, DataEmenta, Ementa, Preco, Alergias)
    AS
    SELECT R_Nome, Data, I_Designcao, Custo,A_Designacao
    From Restaurantes r join Locais l
        on r.IDRestaurante=l.L_IDRestaurante join Ementas e
            on l.IDLocal=e.E_IDLocal join DatasEmentas d
                on e.C_IDDataEmenta=d.IDDataEmenta join ItensEmentas it
                    on e.IDEmenta=it.IE_IDEmenta join Itens i
                        on i.IDItem=it.IE_IDItem join Alergias al
                            on al.IDAlergia=i.I_IDAlergia
```

Figura 7 - View que visualiza todas as ementas registadas de cada restaurante

3.3. Funções

A função criada teve como objetivo de informar o número de vendas de um funcionário. Nesta função foi criada uma variável *total*, onde se armazena o número de vendas de um funcionário. Recorreu-se ao Count para contabilizar o total de vendas da tabela vendas com o ID do funcionário pretendido. Por fim, será retornado o total de vendas desse funcionário, como se observa na Figura .

```
--Função
CREATE OR REPLACE FUNCTION NumeroVendasUmFuncionario(idfunc int)
RETURNS integer AS $$
DECLARE total numeric default 0;
BEGIN
    Select count(*) into total From Vendas
      Where IDFuncionario_Venda=idfunc;
    RETURN total;
END; $$
LANGUAGE plpgsql;
--Select NumeroVendasUmFuncionario(1);
```

Figura 11 - Função que determina o número de vendas de um funcionário.

3.4. Triggers

Quanto aos *triggers*, o primeiro *trigger* criado tem como objetivo ser ativado ao ocorrer uma venda, como se verifica na Figura 12. Este *trigger* é despoletado sempre que ocorra uma inserção na tabela VENDAS.

```
--Trigger
Drop Trigger if exists InsereVenda on Vendas;

CREATE TRIGGER InsereVenda
AFTER INSERT ON Vendas
FOR EACH ROW EXECUTE PROCEDURE InsereVenda();

CREATE OR REPLACE FUNCTION InsereVenda() returns trigger
AS $$
BEGIN
    UPDATE ProdutosEmentas
    SET Stock=Stock-new.Quantidade_consumida
    where IDEmenta_PE=new.IDEmenta_Venda;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Figura 12 – Trigger para atualizar as vendas

3.5. XML

Quanto ao XML, criou-se a *query* da Figura 14, de forma a permitir devolver um XML com os dados referentes a cada venda.

```
--XML
SELECT xmlelement(name Venda,
                 xmlattributes(ID_Venda as ID),
                 xmlelement(name Total, CustoTotal),
                 xmlelement(name DetalhesFatura,
                              (Select xmlagg(xmlflorest(PNome_Func,Produto,Alergia,PNome_Cliente)))
                               From Vendas V  join Funcionarios Func
                               on Func.ID_Funcionario =V.IDFuncionario_Venda join Ementas E
                               on V.IDEmenta_Venda =E.ID_Ementa join ProdutosEmentas PE
                               on PE.IDEmenta_PE = E.ID_Ementa join Produtos P
                               on P.ID_Produto = PE.IDProduto_PE join Alergias A
                               on A.ID_Alergia = P.Produto_IDAlergia
                             )
)From Vendas
```

Figura 14 - Query em XML sobre as vendas dos Restaurantes

4. Aplicação Web

Como referido na Introdução, foi pedida a criação de uma aplicação Web para a empresa R&R, de modo a gerir os dados da mesma, utilizando, para tal, a base de dados criada nos capítulos anteriores. Esta aplicação foi construída em *python*, com recurso a HTML, CSS e *Javascript*.

4.1. Funcionamento da aplicação

A aplicação desenvolvida encontra-se demonstrada na Figura 15, com um exemplo da interação do cliente com a mesma, estando visível, com maior detalhe, nos Anexos, nomeadamente no Anexo B.

Figura 15 - Diagrama da Interação do Cliente com a Aplicação Web

Como se pode visualizar na figura anterior, a página inicial é a *index.html* onde o utilizador tem acesso a um breve resumo sobre a empresa R&R e um menu lateral com ligação para as restantes páginas. O menu permite acesso às páginas (1) Inserir Dados, (2) Alterar Dados, (3) Eliminar Dados, (4) Tabelas Base de Dados, (5) Ver Consumos, (6) Ver Ementas, (7) Repor Stock, (8) Média Consumo por Restaurante, (9) Funcionário com Mais Vendas e (10)

xml Dados Restaurante. Os números 1, 2 e 3 correspondem a procedimentos CRUD, os números 4,5 e 6 a *Views*, os números 7,8 e 9 correspondem a funções e o número 10 a XML.

As páginas Inserir, Alterar e Apagar Dados, representadas na Figura 16, são semelhantes. Ao clicar num dos botões correspondentes às tabelas da Base de Dados, tem-se acesso a uma página com *inputs* para inserir dados de modo a interagir com a base de dados, como demonstrado na Figura 17.

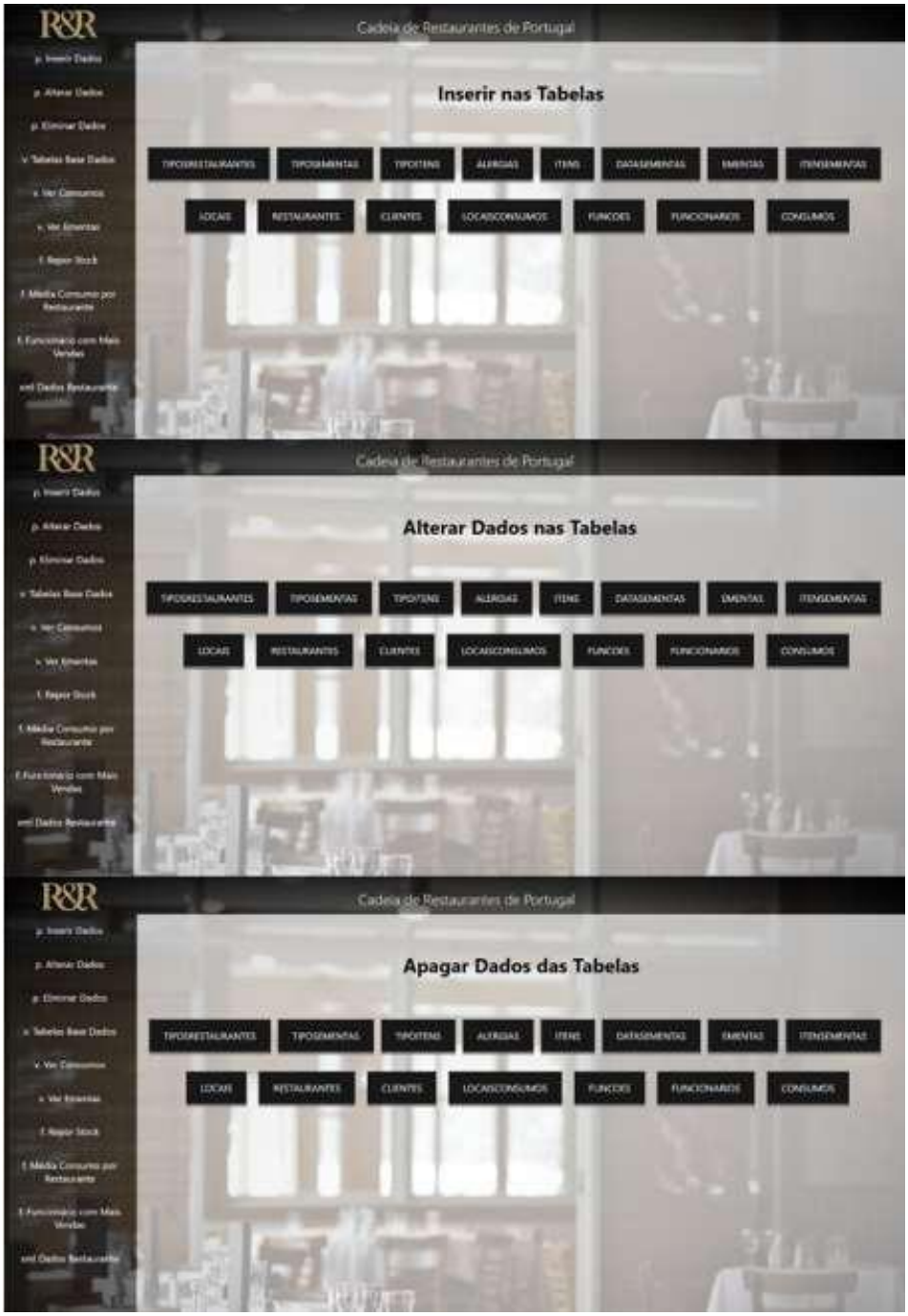


Figura 16 - Páginas Inserir, Alterar e Apagar Dados

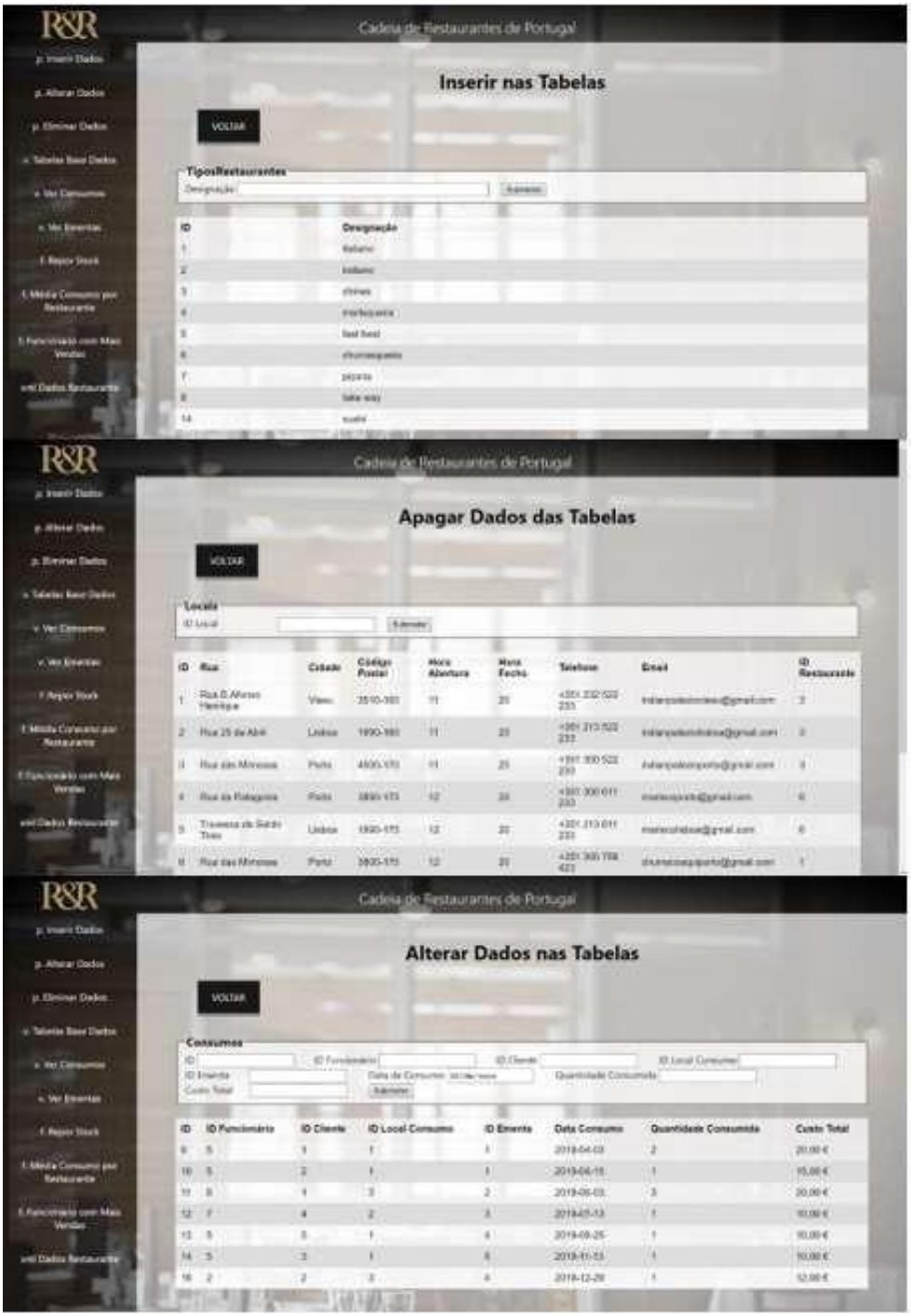


Figura 17 - Exemplos de Páginas de Procedimentos CRUD

No caso da inserção de dados na tabela Ementas, visto esta possuir um *trigger* que só permite a inserção de ementas ao domingo, colocou-se uma opção de ativar/desativar o *trigger*, de modo a poder demonstrar o *trigger* em funcionamento, como se demonstra na Figura 18.

Figura 18 - Inserção na Tabela Ementas - App Web

Nas páginas relativas às *Views* tem-se acesso às *Views* mencionadas anteriormente, sendo que, na Tabelas Base Dados, tem-se acesso a todas as tabelas da Base de Dados (Figura 19) e, nas outras duas, às *Views* criadas especialmente para visualizar Consumos e Ementas.

Figura 19 - Todas as Tabelas - Views App Web

Relativamente às funções criadas e à visualização do seu funcionamento em ação, tem-se as páginas Repor Stock, Média Consumo por Restaurante e Funcionário com Mais Vendas. Na primeira, para proceder à alteração do Stock, é necessário inserir o ID do funcionário, além dos atributos a alterar, e a alteração só é efetuada caso esse ID corresponda a um funcionário da gerência. Nas segunda e terceira páginas mencionadas basta inserir o ID do restaurante ou o ano, respetivamente, para obter o resultado desejado. Na Figura 20 encontra-se representada a página Média Consumo.

Figura 20 - Página Média Consumo

A última página que se tem acesso através do menu é a página do XML, onde se apresentam os ficheiros XML criados para cada restaurante e onde se pode transferir cada um ao clicar no botão GUARDAR TXT, obtendo um ficheiro de texto com os dados, como se visualiza na Figura 21.

Figura 21 - Página XML da Aplicação Web

4.2. Desenvolvimento da aplicação

Como já foi referido na Introdução, desenvolveu-se uma aplicação Web com uma API em *python*, que interliga o servidor do *PostgresSQL* com o site Web desenvolvido em HTML com CSS e *Javascript*. Para tal utilizou-se o *Psycopg2*, um adaptador para Base de Dados em *python*, e o *Flask* que consiste numa *framework* para aplicações web WSGI.

Através da criação de um ficheiro do tipo *.py* (*python*), declarou-se no mesmo a conexão à base de dados, demonstrada na Figura 22. Esta conexão é iniciada com *auto commit* e, em caso de erro, faz *rollback* à base de dados, de modo a evitar erros durante a execução da aplicação, visto que, sem o *rollback*, qualquer erro gerado obrigaria à interrupção do acesso da aplicação à base de dados até nova ligação ou reinício da mesma.

```
import os
import datetime
import psycopg2
from flask import Flask, render_template, request

#criar app web
app = Flask(__name__)

#conexao à base de dados
con = None
try:
    con = psycopg2.connect (
        host = "localhost",
        port = 5432,
        database = "empresaRR",     #nome base dados
        user = "postgres",
        password = "morgan"         #pass acesso ao pgadmin
    )

    con.set_session(autocommit=True)

    #executar app web
    if __name__ == '__main__':
        app.run(debug=True)

    con.save()


except psycopg2.DatabaseError as error:
    con.rollback()
finally:
    if con is not None:
        con.close()
```

Figura 22 - Ligação à base de dados pela aplicação - psycopg2

De modo a interligar as diferentes páginas, recorre-se a "*@app.route()*" com os dados do url e, em caso de páginas com formulários, com o método GET e/ou POST. Na Figura 23 encontra-se demonstrada a rota *home* e a rota para a página Inserir *TiposRestaurantes*, onde é executado o comando necessário na base de dados.

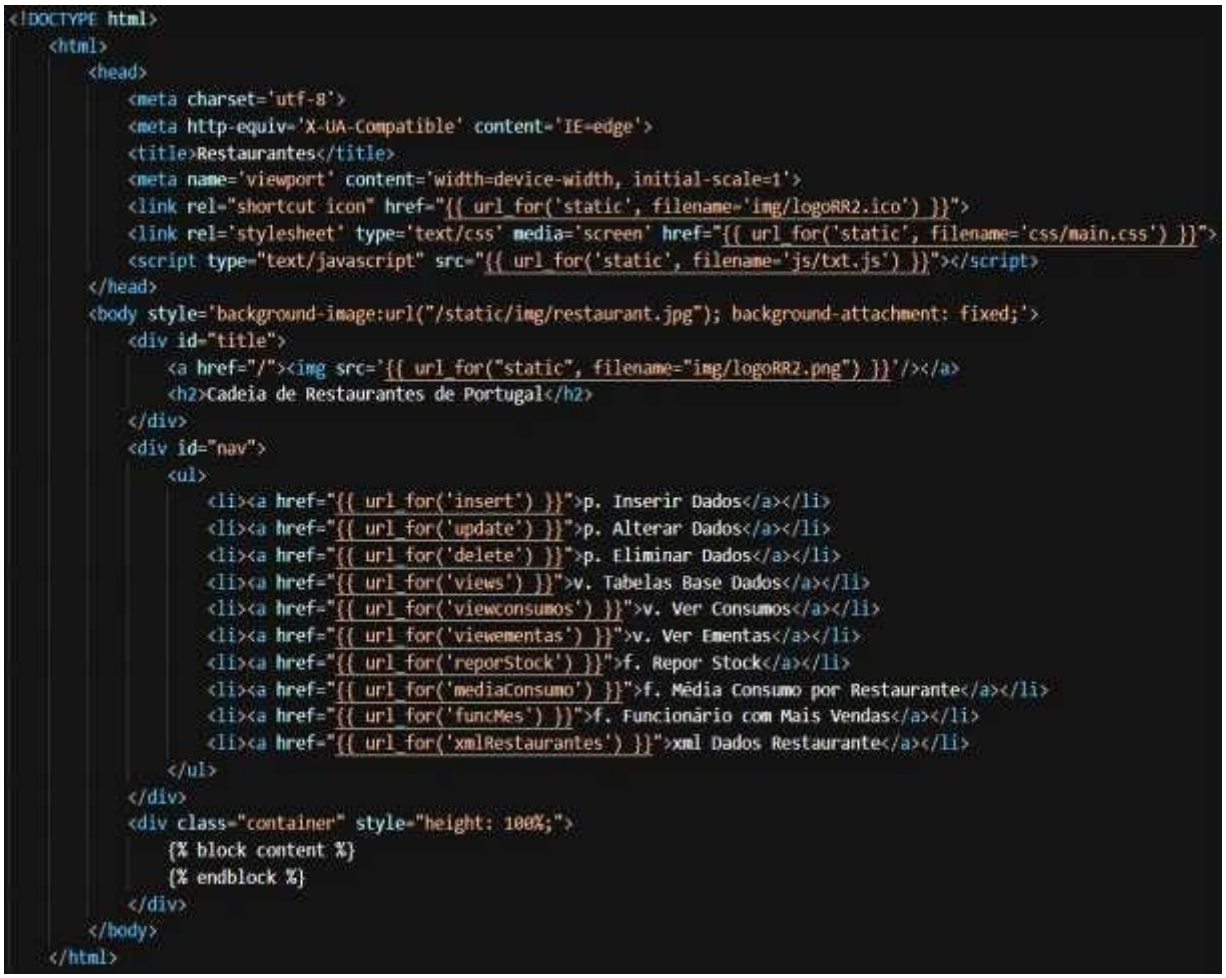
```python
# rotas entre paginas
@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('index.html')

#-------------------------------------------------------------------------------
#   inserir
#-------------------------------------------------------------------------------
@app.route('/ins_tipo_rest/', methods=['GET', 'POST'])
def ins_tipo_rest():
    if request.method == 'POST':
        #Tipo Restaurantes
        TRest = request.form.get("tRest_I")
        cur_TRest = con.cursor()
        cur_TRest.execute('CALL insert_TipoRestaurantes(%s)', [TRest])

    curViewtRest = con.cursor()
    curViewtRest.execute("SELECT * From TipoRestaurantes")
    arrayVtipoRest = curViewtRest.fetchall()
    return render_template('ins_tipo_rest.html', da = arrayVtipoRest)
```

Figura 23 - Rotas entre páginas - App Web

Além disso, todas as páginas utilizam a mesma página de base, a página *layout.html*, que contém a definição do ficheiro HTML, com *head*, *body*, caminhos para os ficheiros *css* e *javascript* e também inclui o menu de todo o site, tendo este sempre visível sem necessidade de carregar uma nova página sempre que se acede a uma nova ligação. Na Figura 24 encontra-se ilustrado o ficheiro *layout.html*. Como se visualiza no código da página *layout.html*, tem-se uma *div* onde as restantes páginas vão colocar o seu código, que possui o código *"{% block content %}{% endblock %}*". Nas restantes páginas, coloca-se o mesmo bloco de código incluindo "*{extends "layout.html"}*" antes, fazendo, deste modo, a interligação do código das novas páginas para a layout.



```
<!DOCTYPE html>
    <html>
        <head>
            <meta charset='utf-8'>
            <meta http-equiv='X-UA-Compatible' content='IE=edge'>
            <title>Restaurantes</title>
            <meta name='viewport' content='width=device-width, initial-scale=1'>
            <link rel="shortcut icon" href="{{ url_for('static', filename='img/logoRR2.ico') }}">
            <link rel='stylesheet' type='text/css' media='screen' href="{{ url_for('static', filename='css/main.css') }}">
            <script type="text/javascript" src="{{ url_for('static', filename='js/txt.js') }}"></script>
        </head>
        <body style='background-image:url("/static/img/restaurant.jpg"); background-attachment: fixed;'>
            <div id="title">
                <a href="/"><img src='{{ url_for("static", filename="img/logoRR2.png") }}'/></a>
                <h2>Cadeia de Restaurantes de Portugal</h2>
            </div>
            <div id="nav">
                <ul>
                    <li><a href="{{ url_for('insert') }}">p. Inserir Dados</a></li>
                    <li><a href="{{ url_for('update') }}">p. Alterar Dados</a></li>
                    <li><a href="{{ url_for('delete') }}">p. Eliminar Dados</a></li>
                    <li><a href="{{ url_for('views') }}">v. Tabelas Base Dados</a></li>
                    <li><a href="{{ url_for('viewconsumos') }}">v. Ver Consumos</a></li>
                    <li><a href="{{ url_for('viewementas') }}">v. Ver Ementas</a></li>
                    <li><a href="{{ url_for('reporStock') }}">f. Repor Stock</a></li>
                    <li><a href="{{ url_for('mediaConsumo') }}">f. Média Consumo por Restaurante</a></li>
                    <li><a href="{{ url_for('funcMes') }}">f. Funcionário com Mais Vendas</a></li>
                    <li><a href="{{ url_for('xmlRestaurantes') }}">xml Dados Restaurante</a></li>
                </ul>
            </div>
            <div class="container" style="height: 100%;">
                {% block content %}
                {% endblock %}
            </div>
        </body>
    </html>
```

Figura 24 - Código da página layout.html

5. Conclusão

A execução deste trabalho prático contribuiu para a aprendizagem e consolidação de conhecimentos adquiridos durante este semestre, tanto a nível da programação em *PostgreSQL*, como em *Python*, nomeadamente no desenvolvimento de procedimentos CRUD, *Triggers*, *Views*, XML, Funções e no desenvolvimento da aplicação web.

Este trabalho foi executado com sucesso, tendo sido possível concluir todos os objetivos pretendidos, isto é, o desenvolvimento de uma aplicação web para gestão dos dados da empresa de restauração R&R, cumprindo com os requisitos (1) da existência de uma ementa própria para cada refeição/dia da semana, (2) da criação das ementas ao início da semana, neste caso, no domingo, (3) da organização da ementa por bebidas, entradas, pratos de peixe, pratos de carne e sobremesas, (4) do registo de designação, preço e risco de alergia nas ementas, (5) da associação de clientes a locais (mesa ou balcão) e (7) do consumo associado às escolhas feitas da ementa.

O trabalho prático foi desenvolvido com empenho, tendo sempre em mente corresponder aos objetivos da empresa, tentando cumprir com as funcionalidades pedidas, implementando-as funcionalmente, e conseguindo executar funcionalidades extra.

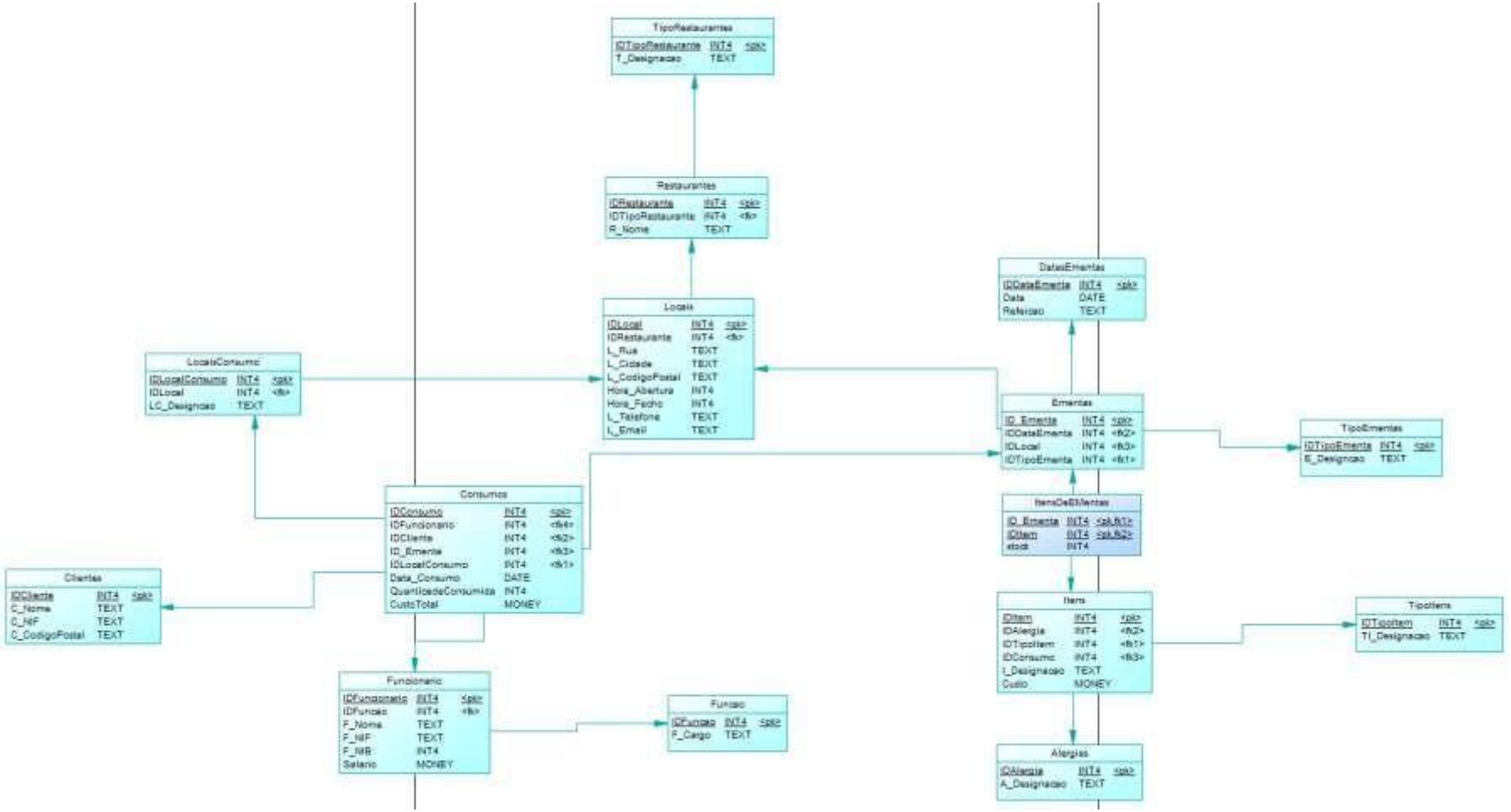
Apesar de toda a aplicação web e desenvolvimento da base de dados ter sido desenvolvido em duas novas linguagens, nunca antes dadas, com autoensino e dedicação foi-nos possível aprender o funcionamento básico das mesmas e desenvolver esta aplicação. Espera-se no futuro voltar a trabalhar com estas linguagens e ganhar mais prática no desenvolvimento e execução das mesmas.

6. Referências

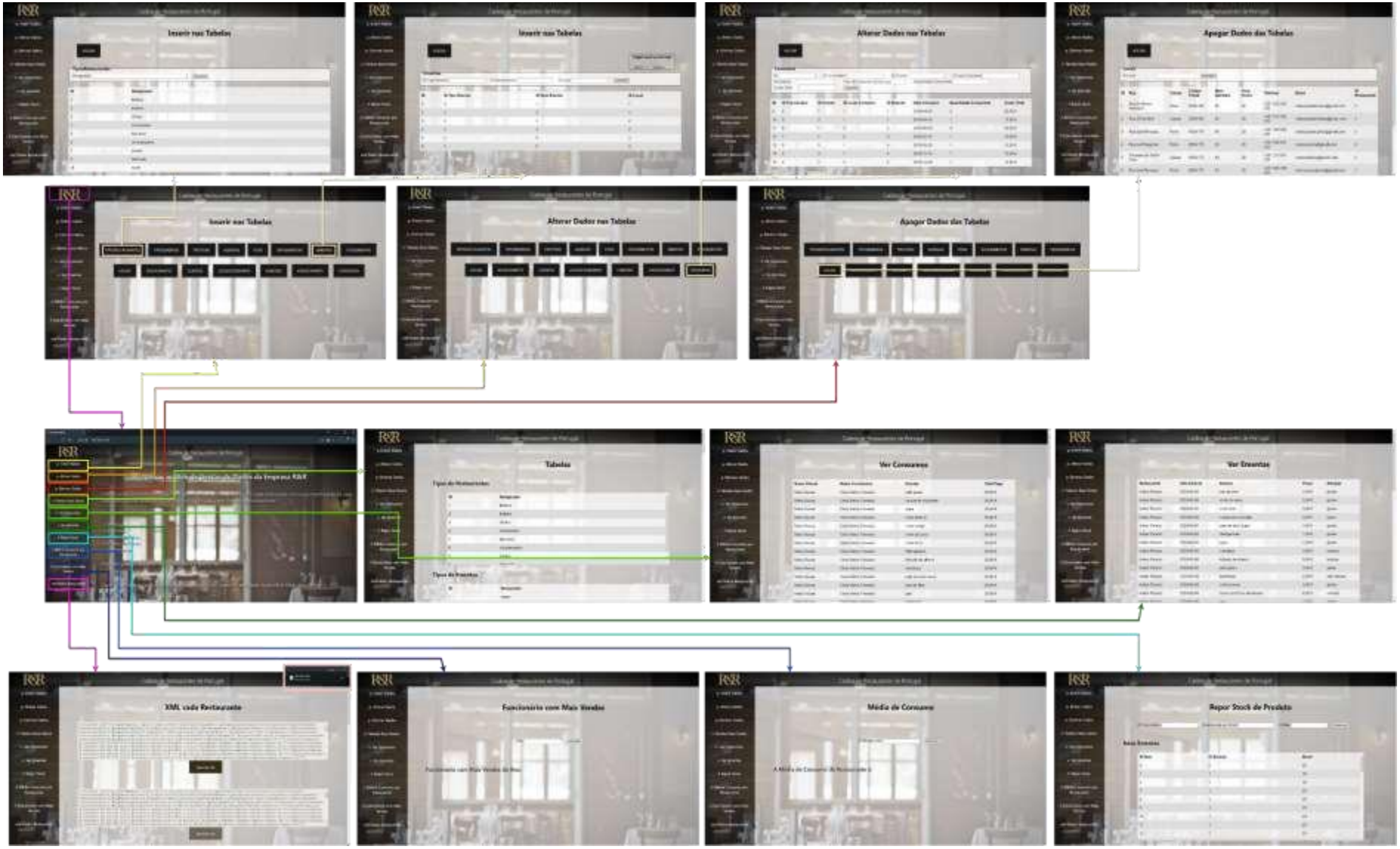
PostgreSQL Tutorial. (2019). *What is PostgreSQL?* Obtido de PostgreSQL Tutorial: http://www.postgresqltutorial.com/what-is-postgresql/

w3schools.com. (1999-2019). *Python Introduction.* Obtido de w3schools.com: https://www.w3schools.com/python/python_intro.asp

7. Anexos



Anexo A – Modelo Físico de Dados



Anexo B - Diagrama da Interação Cliente - App Web