Manual de usuário do ManagePy

Escrito por Bruno Baena,
Gustavo Herrero, Kauã Lucas,
Luana Barbosa e Vinicius Gabriel

Projeto desenvolvido como requisito para a conclusão
da disciplina *Computação Científica em Python* (LOM3260)
Escola de Engenharia de Lorena, Universidade de São Paulo

Novembro de 2021

Sumário

1 O que é o ManagePy?

O ManagePy é um software de código aberto desenvolvido por estudantes da Escola de Engenharia de Lorena, da Universidade de São Paulo, como requisito para a conclusão da disciplina *Computação Científica em Python* (LOM3260).

A proposta do projeto é a realização de um software que gerenciaria e automatizaria os principais processos que ocorrem em um estabelecimento comercial, como organização do fluxo de caixa, estoque de produtos, dados de funcionários, etc. E é isto que se propõe a fazer o software ManagePy.

Todas as funções que o programa realiza estão descritas neste documento. A construção deste software foi feita utilizando a linguagem de programação Python e está muito baseada na construção de um banco de dados a partir do sistema de gerenciamento de dados MySQL, que interage com o Python. Também está inserida no projeto a utilização do programa Power BI, um serviço de análise de negócios desenvolvido pela Microsoft, para a geração de estatísticas com os dados armazenados pelo ManagePy.

2 Instalação MySQL

Além do Python ser requerido para a utilização do ManagePy, é necessário a instalação do MySQL e do Power BI para que o programa funcione em sua plenitude. A seguir está descrito a maneira de como realizar a instalação destes softwares, para o MySQL, a instalação nos sistemas operacionais Windows e Linux Ubuntu.

2.1 Windows

Para a instalação do MySQL no Windows, vamos iniciar pela instalação da ferramenta WampServer, um software que efetua a instalação automática de um conjunto de softwares no computador, de modo a facilitar a configuração de um software interpretador de scripts local e um banco de dados no sistema Windows. Esta ferramenta permite ao usuário do Windows unificar a utilização do MySQL com PHP. Por mais que este recurso não seja necessário para o usuário do ManagePy, este adicional pode ser interessante para aplicações futuras e uma checagem diferente dos dados. Após a instalação do WampServer, mostramos como fazer a instalação do MySQL Workbench, uma ferramenta visual para o design e criação de um sistema de banco de dados de MySQL.

2.1.1 WampServer

Para realizar o download do pacote Wamp basta acessar o site atráves deste link. Já no site, o próximo passo é clicar na aba "downloads"na parte superios da tela, como mostra a Figura 1.

Figura 1: Aba de download do site do WampServer

Após selecionar a versão desejada, irá aparecer a seguinte tela, representada pela Figura 2.

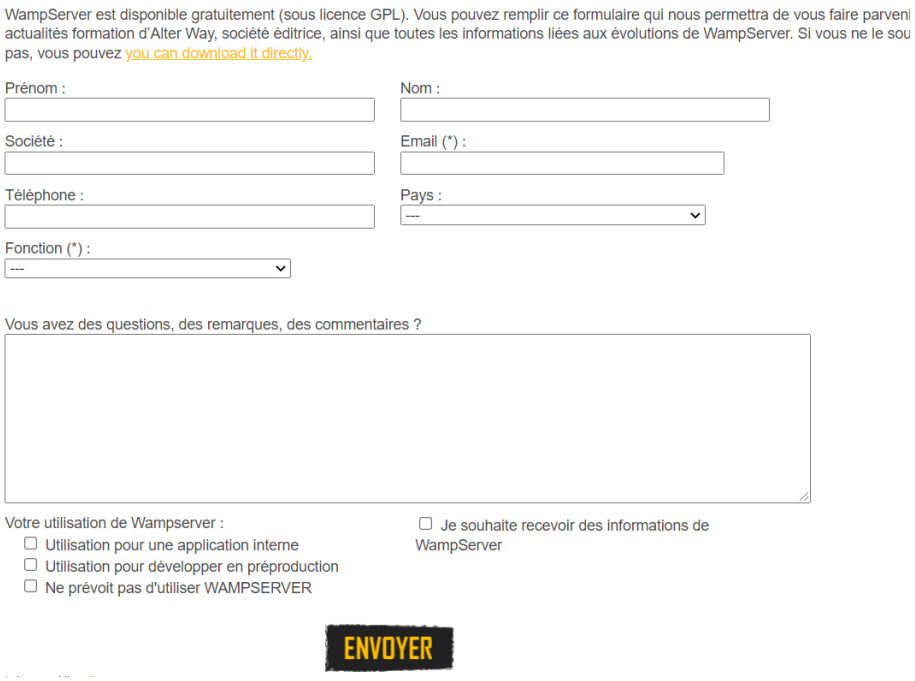

Figura 2: Próxima tela a aparecer no site do WampServer

Basta clicar em "you can download it directly", feito isso a página redirecionará para um outro site. Para seguir, o caminho agora é clicar em "download latest version", como está

mostrado na Figura 3.

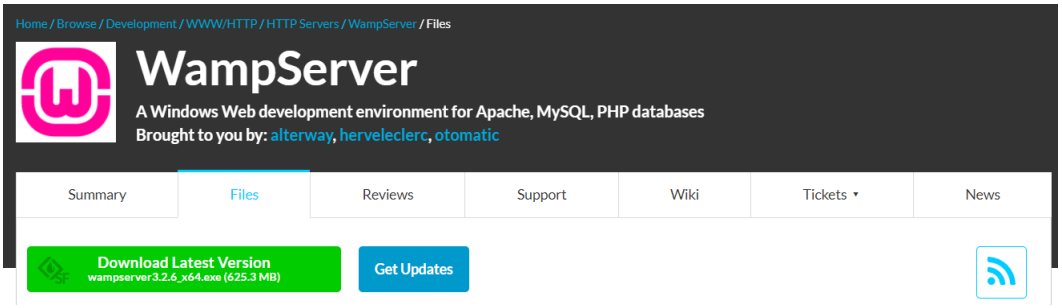

Figura 3: Área de download do WampServer

A última etapa para instalar o WampServer se dá após o término do download. Agora basta clicar no arquivo e seguir os passos dados pelo installer do WampServer.

2.1.2 MySQL Workbench

Para a realização da instalação do Workbench, o primeiro passo é acessar o site através deste segundo link e ir na área dos desenvolvedores. Para conseguir obter acesso a área dos desenvolvedores basta clicar em "DEVELOPER ZONE" como mostra a Figura 4.

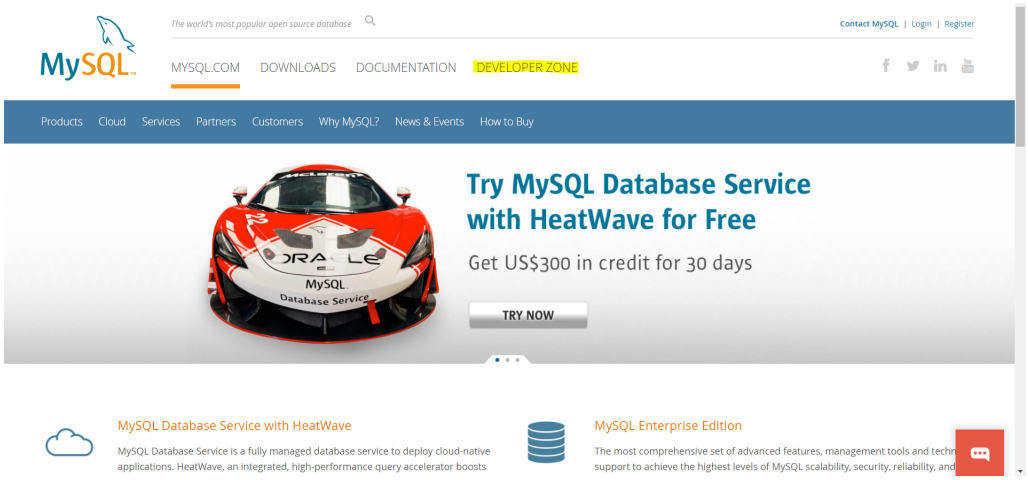

Figura 4: Página inicial do site do MySQL

Uma vez dentro da área dos desenvolvedores basta clicar em MySQL downloads no canto inferior esquerdo como demonstrado na Figura 5.

Feito isso irá aparecer diversos arquivos que podem ser selecionados basta selecionar o arquivo MySQL Workbench. Após ter selecionado o arquivo basta selecionar o sistema operacional desejado e clicar em download. Após ter terminado o download do arquivo basta clicar no arquivo e seguir os passos dados pelo installer do MySQL Workbench.

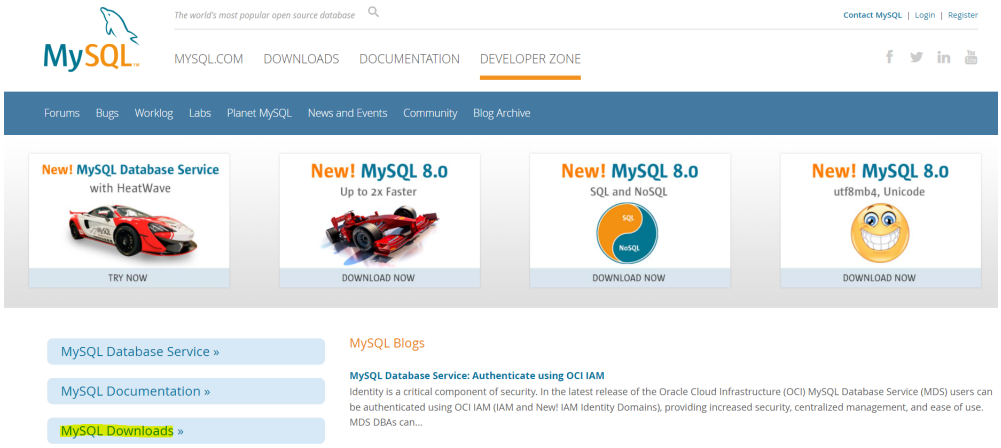

Figura 5: Área dos desenvolvedores do site do MySQL

2.2 Linux Ubuntu

A instalação efetiva do MySQL no Linux Ubuntu será mostrada via comandos no terminal. A primeira coisa a se fazer é acessar o site do MySQL neste terceiro link e verificar a versão mais atualizada disponível para download. Como exemplo, na data da confecção deste manual a versão mais atual é a 0.8.20, como mostra parte destacada em azul na Figura 6.

Figura 6: Área do site para obtenção da versão mais recente

Agora no terminal, começamos com o seguinte comando:

```
wget -c https://repo.mysql.com//mysql-apt-config_0.8.20-1_all.deb
```

Veja que o comando possui a parte 0.8.20, referente à versão utilizada. Após esta primeira etapa virão algumas seções pedindo confirmação, que devem ser realizadas sem problema. O próximo passo é fazer a atualização das listas de repositório do Linux. Ainda no terminal, digite agora o seguinte comando:

```
sudo apt update
```

Após esta atualização, digitamos o seguinte, ainda no terminal:

```
sudo apt install mysql-server
```

O sistema pedirá para o usuário definir uma senha para o ambiente MySQL, e aqui vai um recado: em alguns tutoriais pela internet é possível encontrar quem diga que não é necessário colocar senha nesta parte, o "OK"poderia ser o bastante, mas por experiência do escritor, recomendo colocar uma senha qualquer, pois o sistema pode exigir isto depois e uma senha vazia será bloqueada. Agora você terá que dar OK em mais alguns passos.

Agora precisamos configurar a instalação segura do MySQL, com o seguinte script:

```
sudo mysql_secure_installation
```

O sistema pedirá a senha configurada no passo anterior. Agora o sistema fará algumas perguntas. A primeira delas é sobre a validação de senha, nesta parte é suficiente apertar a tecla Enter, a próxima pergunta é sobre se o usuário gostaria de trocar a senha, Enter é novamente suficiente. A terceira pergunta é sobre usuários anônimos, pode-se digitar y (ou s) para removê-los. A próxima pergunta é sobre a desabilitação de login de root remoto, y (ou s) novamente. As próximas duas perguntas podem ser respondidas novamente usando y (ou s) e o sistema estará configurado da maneira como dos desenvolvedores do ManagePy.

Ainda no terminal, podemos agora verificar o status do serviço com o seguinte comando:

```
sudo systemctl status mysql
```

Caso seja encontrado a mensagem active (running), o sistema está pronto, caso contrário, digite o seguinte:

```
sudo systemctl start mysql
```

Agora habilitamos o início automático do MySQL Server com o script a seguir:

```
sudo systemctl enable mysql
```

Agora o MySQL está pronto para ser utilizado e já pode ser acessado pelo terminal com o seguinte comando:

```
mysql -h localhost -u root -p
```

A senha configurada pelo usuário no início das etapas será pedida, e agora o usuário já pode utilizar os comandos do sistema MySQL neste ambiente.

Mas também pode ser interessante a utilização de uma IDE, caso o usuário prefira. Esta IDE é o Workbench, o mesmo mostrado na seção de instalação para o Windows. Esta instalação é bem simples, basta digitar o seguinte comando no terminal:

```
sudo apt install mysql-workbench-community
```

E esta foi a última etapa de instalação do ambiente MySQL no Linux Ubuntu.

3 Principais comandos

3.1 Comandos MySQL

O arquivo com toda a estrutura necessária para utilizar o ManagePy está disponível, porém vamos fazer uma breve explicação sobre alguns comandos do ambiente do MySQL para que o leitor possa entender mais profundamente alguns comandos existentes no código em Python que são essenciais para o funcionamento do software e também caso o leitor queira fazer alguma modificação no código original.

O primeiro comando, ao estar em um ambiente MySQL, seja pelo terminal ou pela IDE Workbench, é seguinte:

```
create database nome_da_database;
```

Este comando servirá para criar uma database, ou seja, o arquivo que conterá toda a organização a qual os dados estarão submetidos. Uma database possui tabelas, o registro que conterá as inúmeras linhas de dados. Antes de criar uma tabela, é necessário entrar dentro do ambiente da database escolhida para editar, o comando para entrar neste ambiente é o mostrado a seguir.

```
use nome_da_database;
```

Perceba que os comandos devem todos terminar com ‘;’. Após o comando anterior, já é possível criar novas tabelas da seguinte forma:

```
create table nome_da_tabela(
coluna1 tipoDeDado,
coluna2 tipoDeDado,
coluna3 tipoDeDado,
....
);
```

O significado do que está escrito como ’coluna1’ ou ’coluna2’, por exemplo, é o nome do dado a ser guardado, como demonstração podemos citar: nome, telefone, idade, etc. Já os tipos de dado a serem guardados podem vários, não vamos nos aprofundar nisso aqui, mas como exemplo podemos citar: int, char, varchar, date, etc. Podemos também usar outras denominações no campo do tipo de dado, como ‘NOT NULL’, ‘PRIMARY KEY’, entre outras. Para mais informações recomendamos procurar na documentação do MySQL ou em outros diversos materiais de estudo pela internet. O mesmo serve para os próximos comandos mostrados aqui.

Estes são os comandos básicos para a construção da estrutura de um banco de dados. Vamos agora mostrar alguns que são bem importantes na construção do software e são recorrentes

dentro do código.

O primeiro deles é o ‘INSERT INTO’ (os comandos podem ser escritos minúsculo ou maiúsculo, vamos adotar aqui a convenção de usar minúsculo para escrever o código exemplificado). Este comando é o responsável por inserir dados no banco de dados e uma das maneiras mais gerais a qual ele pode ser usado é a seguinte:

```
insert into nome_da_tabela values (
  dado1,
  dado2,
  dado3,
  ....
);
```

O próximo comando essencial para o entendimento e funcionamento do código do ManagePy é o ‘SELECT’, este comando serve para que possamos visualizar os dados guardados em uma tabela do banco de dados. Existem diversas maneiras de utilizar este comando, mas a mais geral é a seguinte:

```
select * from nome_da_tabela;
```

A utilização deste comando não somente é crucial para o funcionamento do programa mas também no processo de confecção do código, para sabermos se os dados estão sendo atualizados. O asterisco ‘*’ presente na linha de código anterior serve para que o sistema retorne todas as colunas de todos os dados na tabela escolhida, mas podemos também utilizar outras combinações para filtrar dados quando conveniente. Algumas destas combinações usadas para o ManagePy foram a substituição do ‘*’ pelos nomes das colunas desejadas e a utilização do comando ‘WHERE’ para filtrar os dados. Como mostra o exemplo a seguir:

```
select dado1, dado2 from nome_da_tabela where
dado3 = 'valor3' and dado4 = 'valor4';
```

O parâmetro ‘AND’ é opcional. Existem diversas outras maneiras de se utilizar o comando ‘SELECT’, com outras combinações de código, para refinar a busca por dados. Mais uma vez recomendamos a procura em outras fontes.

O próximo comando necessário para a utilização do software é o ‘UPDATE’. Este comando modifica uma linha de dados já cadastrado, ou várias, dependendo do comando. A estrutura básica do comando para utilizar o ‘UPDATE’ é a seguinte:

```
update nome_da_tabela set coluna1 = 'valor1'
where coluna2 = 'valor2';
```

Estes são os comandos básicos que se fazem necessários para o entendimento do código e que estão presentes nesta versão do ManagePy.

3.2 Comandos Python

Neste tópico serão mostrados alguns comandos que servem para fazer a integração do Python com a base de dados no MySQL, isto é, com a base de dados criada, pode-se selecionar, inserir e modificar dados a partir de um script em Python.

A primeira coisa a se fazer é importar a biblioteca que permite fazer a interação.

```
import pymysql.cursors
```

Para fazer o script em Python conectar com a database desejada, é preciso ter o seguinte bloco de código:

```
conexao = pymysql.connect(
    host = 'localhost',
    user = 'root',
    password = '',
    db = '',
    charset = 'utf8',
    cursorclass = pymysql.cursors.DictCursor
)
```

Para o parâmetro *password* deve-se colocar a senha configurada na instalação do MySQL, caso a senha seja vazia, pode-se deixá-la com apenas as aspas. No parâmetro *db*, deve-se colocar o nome da database a ser modificada.

Para realizar alguma modificação na databse a partir do python é necessário ter conhecimento dos comandos de MySQL mostrados anteriormente. São necessárias três linhas de código para operar algum comando de interação com a database. O primeiro exemplo é de como utilizar o 'SELECT'.

```
with conexao.cursor() as cursor:
    cursor.execute('select * from funcionarios;')
    funcionarios = cursor.fetchall()
```

Este é um exemplo de como selecionar todos os dados da tabela de funcionários, presente na database do ManagePy. Todos os dados desta tabela estão agora dentro da variável *funcionários*, uma lista que contém vários dicionários, cada linha de dados contida na tabela funcionários é um dicionário na lista *funcionarios*. Em cada um destes dicionários, as chaves são representadas pelas colunas presentes na tabela e os valores são os dados contidos em cada linha. Existe ainda a possibilidade de se querer selecionar apenas uma linha de dados, nesse caso pode ser mais prático utilizar a função *fetchone*, que gera apenas um dicionário com esta linha de dados, ao invés de *fetchall*.

Para inserir ou modificar algum dado, pode-se utilizar os blocos de código a seguir, respectivamente.

```
with conexao.cursor() as cursor:
    cursor.execute('insert into produto values (default, ''leite'', ''110
   '', ''2'', ''3'', ''1'', ''10'')';)
    conexao.commit()
```

```
with conexao.cursor() as cursor:
    cursor.execute(f'update pedidos set data_entregue = "{data}" where id
   = "{n}";')
    conexao.commit()
```

Veja que a string dentro da função *execute* possui aspas duplas para os valores que serão inseridos na tabela produto, também presente na database do ManagePy. Isto acontece pela incapacidade de se colocar aspas simples dentro das aspas simples que limitam a string no todo, por isso deve-se fazer esta alternância. Outra coisa a se destacar é a última linha de código destes dois últimos blocos de código, que são diferentes da última linha da interação usando 'SELECT'. Para os comandos 'INSERT INTO' e 'UPDATE' é necessário utilizar a função *commit* na variável *conexão*.

É possível ainda colocar esta estrutura de código dentro da combinação *try except*, caso exista alguma falha na conexão ou o código esteja com algum erro é possível contornar este erro com alguma outra função.

Estes são basicamente os comandos usados no código do ManagePy para a interação com o banco de dados, com algumas variações destes códigos de MySQL, como mostrado na seção anterior. O resto da construção do software utiliza outros comandos bem conhecidos de Python para a construção das funções que o programa se dispõe a fazer e comandos de interface gráfica para a biblioteca *tkinter*.

4 Funções do programa

Para a preparação do código em Python do ManagePy foi necessário a construção de algumas funções que realizam as ações do programa. O código possui também algumas funções simples que servem apenas para cobrir alguns detalhes para o bom funcionamento do programa, mas vamos nos concentrar apenas nas sete principais funções essenciais para o seu funcionamento, estas serão esxplicadas nesta seção.

O programa está dividido em quatro arquivos *.py* e mais um que contém somente as funções para serem rodadas como ilustração das funcionalidades no terminal de algum editor de códigos. Dentre os quatro arquivos principais, o primeiro é o *login.py*, como diz o próprio nome, este arquivo é o que contém a parte inicial do programa, em que se realiza o login no software. O segundo arquivo, *caixa.py* é o que contém a parte do usuário que se denomina como caixa e possui as funções relativas à parte das vendas do estabelecimento. O terceiro e quarto arquivos,

gerente.py e *gerente_ 2.py* são iguais, à exceção da parte do código em que um arquivo importa o outro quando solicitado. Este recurso foi usado como solução para o problema de atualização da interface ao se inserir um dado novo na database, fica como meta para versões futuras encontrar uma alternativa mais elegante e prática.

4.1 login

Esta é a primeira função do programa. Ela direciona o usuário para as próximas páginas do programa, dependendo do qual opção de login escolher, o gerente ou caixa.

Para um usuário que usa o programa pela primeira vez, não existe nenhum funcionário cadastrado, por isso ninguém teria acesso. Para contornar este problema, o arquivo de banco de dados do ManagePy vem com uma linha de dados cadastrada na tabela *funcionarios*, e o usuário deverá efetuar o login com o nome 'inicio', cargo 'Gerente' e senha 'senha'.

4.2 cadastraFuncionario

Esta função, presente nos arquivos de gerente, é responsável por cadastrar funcionários novos no banco de dados, com todas as informações que podem ser relevantes. Esta função verifica se o banco de dados já não possui dados iguais aos do funcionário a ser cadastrado e impede que aconteca este duplicação de dados. Esta condição foi feita como mostra a pequena parte da função a seguir:

```
1  jaCadastrado = False
2
3  for linha in funcionarios:
4      if cpf == linha['cpf']:
5          jaCadastrado = True
6          break
7
8      if jaCadastrado == False:
9          with conexao.cursor() as cursor:
10             cursor.execute(f'insert into funcionarios values (default, "{
       nome}", "{cargo}", "{salario}", "{cpf}", "{telefone}", "{email}", "{
       data}", default, "{senha}");')
11         conexao.commit()
```

Veja que pode-se colocar valores de variáveis dentro dos comandos para MySQL.

4.3 pagamentos

Esta função realiza o cadastro dos pagamentos dos funcionários, as datas em que ocorrem os pagamentos ficam guardadas na coluna *hist_ pagamento* na tabela os funcionários, com o uso do comando 'UPDATE'. A função também adiciona o valor do pagamento na tabela *fluxo_ caixa* na database, esta tabela contém todas as transações que ocorrem no estabelecimento, organizando assim a quantidade de dinheiro que entra e sai.

4.4 cadastraProduto

A função *cadastraProduto* é bem simples, ela apenas insere os dados de produtos novos que são adicionados ao catálogo do estabelecimento com os dados de custo e preço, os preços de compra e venda, respectivamente. Também é necessário guardar o valor do código do produto, este será utilizado para identificá-lo ao ser vendido no caixa. É necessário cadastrar os códigos em ordem, sem pular números, a partir do 100, pois dessa forma é possível garantir que os códigos tenha 3 dígitos, uma maneira mais agradável de se organizar os dados.

4.5 cadastraFornecedor

Cada produto possui um número existente no estoque do estabelecimento e conforme vão ocorrendo as vendas o estoque vai diminuindo, por isso se faz necessário repor este estoque. O ManagePy possui uma função que cadastra os fornecedores do estabelecimento, com os dados que podem ser relevantes para verificação.

4.6 fazPedido

Com o objetivo de resolver o problema do estoque, a função *fazPedido* realiza os pedidos a serem feitos pelo estabelecimento. Deve-se utilizar os dados de algum fornecedor previamente cadastrado e de produtos também já presentes na database, já com os valores de código, custo e preço de venda. Ao término das ações requeridas pela função, ela envia um email ao fornecedor com os produtos pedidos e insere na tabela *pedidos* os dados do pedido realizado, com o valor da data de entrega vazio. Esta função não altera os dados de estoque e fluxo de caixa.

4.7 atualizaPedido

Esta foi a função mais complicada de se preparar, possui uma estrutura de listas confusa por causa da maneira como os dados dos pedidos realizados são armazenados. Cada produto a ser pedido está dentro de uma lista que contém o nome do produto, seu custo e a quantidade

pedida, e cada pedido é representado por uma lista que possui várias listas dos produtos individuais. Por fim, todas as listas com os pedidos ficam armazenados dentro de uma outra lista que contém todos estes pedidos, esta lista mais geral ainda é representada por uma string, nos obrigando a usar a função *eval* para transformá-la em uma lista, já que sua estrutura é de lista.

O propósito desta função é dar baixa ao pedido realizado pela função anterior e atualizar o estoque dos produtos e o fluxo de caixa do estabelecimento. Esta dificuldade na estrutura dos dados fica evidente quando se tenta conseguir acesso ao valores individuais de cada produto em um pedido, como a quantidade, para se atualizar o estoque, mas a função funciona perfeitamente.

```
1 with conexao.cursor() as cursor:
2     cursor.execute(f"select valor from pedidos where id = '{n}';")
3     valor = cursor.fetchone()
4
5 with conexao.cursor() as cursor:
6      cursor.execute(f"insert into fluxo_caixa values (default, 'pagamento
    de pedido {n}', '{data}', '-{valor['valor']}');")
7      conexao.commit()
```

No trecho de código mostrado acima, vemos como a função atualiza o fluxo de caixa, adicionando um valor negativo para o valor de uma transação, e também um exemplo de uso do *fetchone* no lugar do *fetchall*, além de uma filtragem de código usada com o comando 'SELECT'.

4.8 Funções do caixa

A parte do login do caixa possui cinco funções, *salvaDados*, *mostraTotal*, *mostraTroco*, *atualizaEstoque* e *finExpediente*. A primeira função serve para inserir os dados da cada produto em uma compra, a próximas duas são bem auto-explicativas, mostram o total da compra e o troco a ser feito, dependendo do valor a ser pago pelo cliente. As duas últimas têm papel crucial, após o término de alguma compra a função *atualizaEstoque* diminui o valor de estoque dos produtos passados na compra e a função *finExpediente* atualiza o fluxo de caixa do estabelecimento após o término do expediente do funcionário que utilizou o programa como caixa.

5 Considerações sobre o uso

Como explicado anteriormente, ao se fazer o cadastro na aba de login (o arquivo *login.py* deve ser o compilado), o usuário pode ser redirecionado para uma das duas abas existentes, a do gerente e a do caixa, dependendo de seu cargo na empresa. Aqui serão feitas algumas

considerações sobre o uso de cada uma das duas abas.

Como o leitor pode ter imaginado, não há como efetuar o login no programa sem haver o registro na database anteriormente. Por esta razão, para o usuário conseguir acesso pela primeira vez, é necessário efetuar o login com o nome 'inicio', cargo ´Gerente' e senha como 'senha', este é o registro presente na database disponibilizada. Logo após, o usuário pode efetuar o registro de mais funcionários, como caixas, que terão acesso à página feita para que os caixas tenham acesso.

5.1 Página do gerente

Esta página esta organizada com o auxílio de abas, a ferramenta *Notebook* da biblioteca *tkinter*. Cada aba possui uma das funções citadas na seção anterior, com exceção do login e das funções do caixa.

A primeira consideração a se fazer é sobre a coluna 'PAGAMENTOS' da treeview na aba 'VER/PAG FUNCIONÁRIOS', esta coluna contém as datas em que foram realizados pagamentos para determinado funcionário, e após algum tempo os dados serão maiores que o espaço reservado na treeview. Para resolver este problema basta colocar o ícone do mouse na lateral direita do widget em que está escrito 'PAGAMENTOS', em cinza. Isto expandirá a coluna para o lado e os dados podem ser acessados com a ajuda do scrollbar horizontal presente na treeview. O mesmo vale para as outras colunas, caso necessário e também para outras treeviews presentes nas outras abas, como a coluna pedidos na aba 'VERIFICAR PEDIDOS', que também pode exceder o limite proposto.

Outro ponto importante de se lembrar é sobre a parte em que se realiza um novo pedido de produtos. Antes de se realizar algum, é importante lembrar que o fornecedor do pedido já deve estar previamente cadastrado, assim como os produtos que se deseja comprar.

Assim que é inserido um novo dado em qualquer uma das tabelas, para sua visualização se faz necessário reiniciar o programa, ou entaão utilizar o botão 'ATUALIZAR', presente em mais de uma aba.

5.2 Pagina do caixa

Esta parte do software é bem simples de se usar, mas o não cumprimento da ordem certa das ações pode comprometer o uso do programa. Ao iniciar a passagem dos produtos pelo caixa em uma compra, o usuário deve primeiro digitar os campos relativos ao código e a quantidade do produto a ser passada. Logo depois, ao clicar no botão 'INSERIR', os dados do produto aparecerão na treeview ao lado e o mesmo processo pode ser repetido, até que todos os produtos sejam passados.

Após a inserção de todos os produtos, o botão ‘TOTAL’ deve ser apertado, é importante salientar que este passo é **obrigatório** para a continuação do procedimento. Caso o cliente queira realizar o pagamento da compra por dinheiro, pode-se usar o botão ‘TROCO’ para verificar o troco a ser dado ao cliente, se o pagamento for feito de outro modo não há necessidade de seu uso.

Após estes procedimentos, deve-se usar o botão ‘FINALIZAR COMPRA’ e o ciclo da compra esta finalizado, o estoque atualizado e pode-se começar uma nova compra. Ao término do expediente do caixa, este deve usar o botão ‘FINALIZAR EXPEDIENTE’ e o programa será fechado. Sua utilização também é **obrigatória** pois somente neste passo o fluxo de caixa é atualizado.

6 Power BI

O Power Bi é uma ferramenta que tem crescido na área de Business Intelligence, uma de suas principais funções é coletar dados que não estejam relacionados, até mesmo em diferentes linguagens, e transformá-los em informações coerentes com uma visualização fácil e interativa.

Um de seus diferenciais é capacidade do tratamento de quantidades massivas de dados e processamento mais rápido que de outros softwares comumente utilizados para fazer análises com gráficos. Além disso, seus relatórios podem ser acessados em diversos dispositivos e plataformas, o que também confere mobilidade e flexibilidade aos usuários.

Nosso objetivo com o Power BI é apresentar informações importantes obtidas a partir do banco de dados que desenvolvemos, permitindo que o cliente consiga acompanhar o desenvolvimento a curto e médio prazo do seu negócio.

Como já vimos, o Python será responsável por toda a interação com o programa, então será onde ocorrerá a inclusão, exclusão ou transformação dos dados. O MySQL será responsável por armazenar os diferentes tipos de dados que forem enviados. Por fim, o Power BI irá receber o banco de dados, tratar as informações e apresentá-las a partir de gráficos e indicadores.

À medida que o python alterar os dados do MySQL, as informações apresentados no power BI também sofrerão essas modificações.

6.1 Atualização dos dados

Essa plataforma normalmente é usada aliada da computação em nuvem, isso permite que os dados sejam armazenados em um banco de dados online e o Power BI possa utilizar essas informações com mais liberdade. Como o nosso projeto utiliza um servidor local no próprio computador, para que os dados possam ser atualizados, é necessário que o equipamento esteja

ligado e conectado à internet.

Como o nosso projeto realiza uma simulação, ele atende muito bem as suas necessidades, mas para a aplicação em um projeto real, a utilização da computação em nuvem é uma alternativa mais viável.

Para entender um pouco mais sobre o Power BI, ele possui uma plataforma para desktop, que é onde se trabalha com os dados e desenvolvimento dos painéis e também possui o servidor, que é onde o relatório é publicado e a equipe consegue visualizar.

A atualização do relatório no servidor pode ocorrer de duas principais maneiras, a primeira é periodicamente, nesse modelo é possível definir um horário que ocorrerá a atualização, por exemplo diariamente às 18:00, nesse cenário não haveriam informações das vendas até o horário definido. Optamos por utilizar a segunda maneira, que é a atualização incremental, assim podemos atualizar as informações a cada período de tempo, por exemplo a cada 30 minutos, ou a cada vez que é realizado um update.

6.2 Como acessar

O Relatório de vendas contendo alguns indicadores que podem ser relevantes e úteis para o tomador de decisão estará publicado em nuvem e para acessá-lo, o usuário deverá informar um email com o qual exista uma conta Microsoft. Ou seja, é importante que o usuário possua uma conta Microsoft. Dessa forma, o relatório será enviado ao email cadastrado.

6.3 Funcionamento de cada aba

O Relatório está dividido em duas partes que podem ser acessadas a partir do menu principal. Para acessar cada um dos relatórios, basta clicar sobre o ícone desejado.

No relatório mensal, o tomador de decisão poderá encontrar alguns indicadores de como o negócio desempenhou no mês anterior. Os indicadores mostram os dias da semana em que houveram mais vendas, bem como o total vendido e lucro total do mês.

Na Figura 7 são apresentados dois cartões e um gráfico.

O primeiro cartão indica o valor total vendido no último mês, o cartão logo abaixo apresenta quando foi o lucro desse mesmo mês e, por fim, o gráfico mostra quais os dias da semana que mais vendem no mês.

No relatório diário, é mostrado ao tomador de decisão como o negócio desempenhou no dia em questão. É possível visualizar como está o estoque, bem como quais foram as horas mais movimentadas do dia e quais produtos foram vendidos.

O gráfico de barras empilhadas na Figura 8 também possui um gradiente que indica a

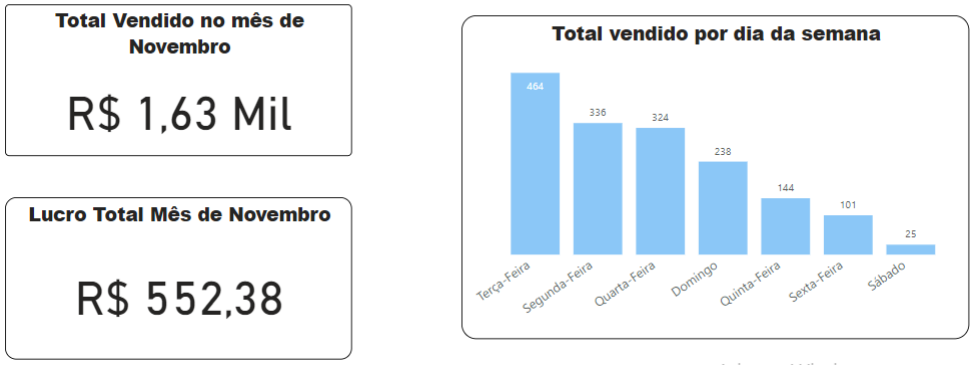

Figura 7: Mostra do gráfico gerado para o relatório mensal

quantidade de estoque de cada produto, sendo que em verde estão os produtos em maior quantidade e a medida que ficam mais avermelhados diminui a quantidade, o que é um indicativo para repor determinado produto.

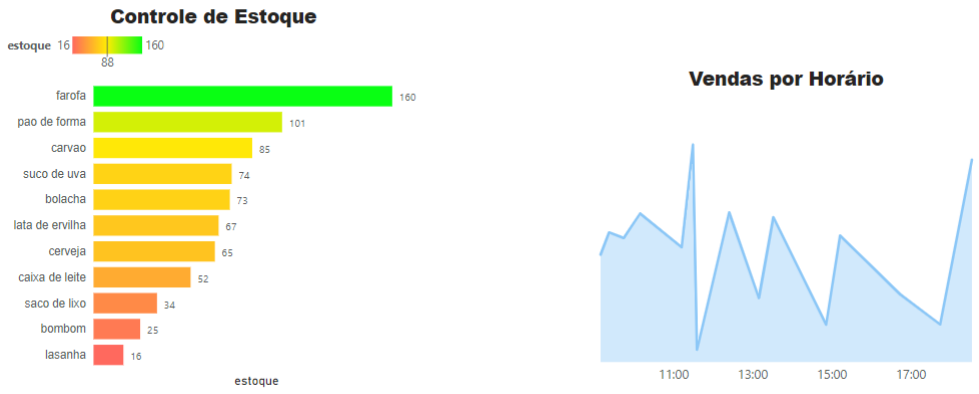

Figura 8: Mostra do gráfico gerado para o relatório diário