

Padaria do seu Victor

Trabalho de Programação Orientada a Objetos
UFSC 2021.1

Alunos e matrículas:

Rafael Luis Sol Veit Vargas - 21100138

Thiago Augusto Bewiahn - 21104162

Victor Rodrigues Gouvêa - 21104164

Funcionamento da aplicação

Resumo: A aplicação consiste em um sistema que controla as vendas de uma padaria, tanto na parte do caixa, adicionando e removendo produtos em um carrinho baseado utilizando seu respectivo código pré-cadastrado e na sua disponibilidade no estoque, quanto na parte administrativa, cadastrando novos produtos, atualizando o estoque e/ou valor dos produtos e podendo ver as informações de estoque e histórico de vendas. Para entrar na parte administrativa é necessário um login com senha, e dentro da aba do ADM é possível adicionar novos logins e senhas para o acesso.

Janelas: Para todas as funcionalidades citadas acima, foram criadas 3 janelas na aplicação:

- Principal:



CAIXA DA PADARIA DO SEU VICTOR

Código:

Quantidade:

[→ Acessar ADM](#)

Código	Nome	Quantidade	Preço Unitário	Preço Total
--------	------	------------	----------------	-------------

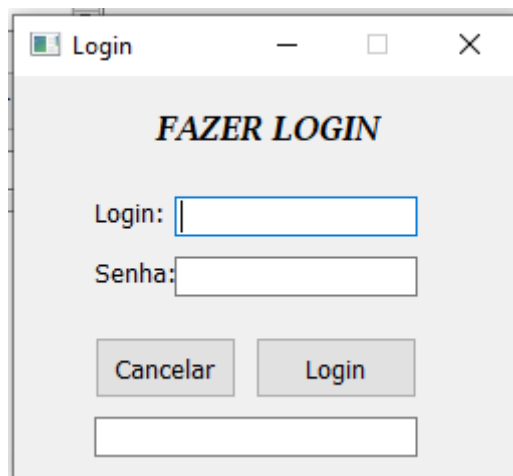
Nessa aba é feito o controle do caixa da padaria, que vai armazenar as compras do cliente que está sendo atendido. Aqui o atendente insere o código e a quantidade dos produtos, que vão sendo adicionados a tabela juntamente com seu nome, preço unitário e preço total(preço unitário * quantidade) após o clique no botão “Adicionar”. Além de que foi elaborado metodologias para que um produto só possa ser adicionado ao carrinho caso exista uma quantidade suficiente cadastrada no estoque.

Também temos os botões de “Limpar Carrinho” e “Remover Produto”, que têm a funcionalidade de remover todos os produtos do carrinho e remover apenas o produto selecionado com o mouse, respectivamente.

Por último temos o botão “Concluir Compra”, que limpa todo o carrinho e gera uma mensagem com o custo total da compra e também o botão para acessar a parte administrativa, que vai abrir a janela de login.

Vale ressaltar que, como a aplicação possui um banco de dados relacional SQLite, todas as ações descritas acima atualizam e verificam constantemente o estoque do banco de dados, para observar se cada ação efetuada pelo usuário é válida e pode ser executada não, e assim, dando o respectivo retorno para o usuário pelas caixas de mensagem.

- Login:

A screenshot of a login window titled "Login" in the title bar. The window has a light gray background. At the top, the text "FAZER LOGIN" is displayed in a bold, italicized, black font. Below this, there are two input fields: the first is labeled "Login:" and the second is labeled "Senha:". Below the input fields, there are two buttons: "Cancelar" and "Login". At the bottom of the window, there is an empty rectangular box.

Nesta aba é feito apenas o login do usuário que, se estiver cadastrado no banco de dados, tem acesso a parte administrativa da aplicação.

Caso algum dos dados de login esteja errado, aparece uma mensagem alertando que os dados são inválidos

- Administração:

The screenshot shows a software window titled 'ADMINISTRAÇÃO' with three main sections for product management and a purchase history table.

Cadastrar Novo Produto

Código:
Nome:
Preço Unitário:
Quantidade:

Atualizar Produto

Código:
Nome:
Preço Unitário:
Quantidade:

Adicionar Novo Usuário

Login:
Senha:

HISTÓRICO DE COMPRAS:

Código	Nome	Data	Quantidade	Valor Total
12345	Arroz	2021-08-21 11:23	10	R\$459.8
11111	Bolacha	2021-08-21 13:38	10	R\$500.0
33333	Balinha	2021-08-21 13:38	20	R\$800.0
11231	Refri	2021-08-21 13:38	7	R\$420.0
11111	Bolacha	2021-08-21 13:38	10	R\$500.0
33333	Balinha	2021-08-21 13:38	20	R\$800.0
11231	Refri	2021-08-21 13:38	7	R\$420.0
12345	Arroz	2021-09-06 09:43	3	R\$150.9
12345	Arroz	2021-09-06 09:43	3	R\$150.9
12345	Arroz	2021-09-06 09:44	3	R\$150.9

Nessa janela temos diversas funções relacionadas a administração da padaria, como a de cadastrar um novo produto, a qual basta o usuário inserir os dados do novo produto e usar o botão de cadastrar para um novo produto ser adicionado ao estoque.

Também temos uma funcionalidade parecida ao lado, onde conseguimos atualizar os dados de um produto. Basta inserir o código, e ao clicar em buscar ele verifica no banco de dados se existe um produto cadastrado com aquele código e, caso exista, o usuário consegue mudar as informações sobre esse produto e clicar no botão “Atualizar” para efetivar as mudanças.

Nesta aba também é possível adicionar um novo usuário a parte do ADM, apenas inserindo o login e a senha do novo usuário e clicando em “Adicionar”.

Por último, conseguimos visualizar o histórico de compras e os dados do estoque da padaria usando os botões ao lado da caixa onde essas informações aparecerão. A imagem acima mostra um exemplo de visualização do histórico de compras, já a abaixo mostra a visualização do estoque.

DADOS DO ESTOQUE:			
Código	Nome	Quantidade	Preço Unitário
11111	Coca-Cola	20	R\$ 11,65
12345	Arroz	2	R\$ 15,00
2323	Pão	55	R\$ 4,50
5000	Bolacha	8	R\$ 7,25
0	Suco	9	R\$ 6,25
7856	Doce de leite	7	R\$ 12,00
1212	Pão doce	10	R\$ 6,00

Buscar Histórico
Buscar Estoque

Instruções para executar a aplicação:

- Ter a linguagem python instalada no seu computador, juntamente com as seguintes bibliotecas: PyQt5, sqlite3 e bcrypt
- Verificar se todos os arquivos do repositório estão instalados
- Executar o arquivo main.py

OBS.: Caso algum erro na execução do código persista, é recomendado criar um ambiente virtual com o pip-venv, instalar todas as dependências necessárias e tentar rodar novamente.

Segue um link mostrando como efetuar esse procedimento:

<https://docs.python.org/pt-br/3/tutorial/venv.html>

Sobre o código

Os códigos da aplicação estão organizados da seguinte forma: há uma pasta separada chamada designs, que contém os layouts de cada tela do programa, feitas com o PyQt5. Desse modo, não precisamos descrever a GUI no arquivo principal, apenas importamos o layout necessário, facilitando a leitura e organização do código.

Os arquivos principais são:

- adm.py (controlador da tela do administrador);
- login.py (controlador da tela de login);
- main.py (controlador da tela principal de compras);
- conexoes.py (contém todas as funções que fazem uma ligação com o banco de dados).

Alguns trechos do código que gostaríamos de dar destaque:

- Forma de conectar ao banco de dados:

```
databasePath = 'padaria.db'

@contextmanager
def Conectar(databasePath):
    try:
        conn = sqlite3.connect(databasePath)
        cursor = conn.cursor()
        yield conn, cursor

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()

with Conectar(databasePath) as (conn, cursor):
```

- Função para verificar login e senha de um usuário, e função para cadastrar um usuário.

```
def Cadastrar_User(login, senha):  
    """  
    Essa função irá adicionar um usuário na tabela, para isso é necessário que seja passado o login e a senha  
    em string, a senha em si será criptografada antes de ser guardada (fonte: https://zetcode.com/python/bcrypt/)  
    """  
  
    senha_byte = str.encode(senha)  
    hashed = bcrypt.hashpw(senha_byte, bcrypt.gensalt())  
  
    with Conectar(databasePath) as (conn, cursor):  
        cursor.execute(  
            'INSERT INTO usuarios (login, senha) VALUES (?, ?)', (login, hashed))  
        conn.commit()
```

Utilizamos a biblioteca bcrypt do Python para dar um mínimo de segurança às senhas dos usuários, já que eles terão permissões de administrador no sistema da padaria, e qualquer acesso de terceiros poderia comprometer o estoque e portanto o funcionamento do caixa da padaria.

```
def Verificar_User(login, senha):  
    """  
    Verifica se o login e senha passados estão cadastrados como usuários, retorna True caso encontre  
    um cadastro em que o login e senha passados batam com o registro, retorna False caso não encontre  
    nenhum caso que os dados batam, a senha deve ser passada como string  
    """  
  
    with Conectar(databasePath) as (conn, cursor):  
        senha_byte = str.encode(senha)  
  
        cursor.execute(  
            f'SELECT login, senha FROM usuarios WHERE login=?', (login,))  
        retornos = cursor.fetchall()  
        conn.commit()  
  
        for user in retornos:  
            if bcrypt.checkpw(senha_byte, user[1]):  
                return True  
        return False
```

A senha é convertida para UTF-8 nos dois casos (cadastrar e verificar), já que esse é o tipo de dado que o bcrypt utiliza em suas funções. Ao verificar, selecionamos todos os usuários que estão na tabela por meio da query SQL e percorremos cada um deles, checando se a senha passada corresponde à senha daquele cadastro.

Conceitos de Orientação a Objeto utilizados:

- Classes e heranças: a classe Main (tela principal) herda das classes QMainWindow (do PyQt5) e Ui_MainWindow (contém as informações do layout dessa tela);

```
class Main(QMainWindow, Ui_MainWindow):
```

Também utilizamos construtores para inicializar cada tela e adicionar as conexões de cada botão com sua respectiva função, como no exemplo abaixo:

```
class Adm(QMainWindow, Ui_AdmWindow):
    def __init__(self):
        super().__init__() # Chama o construtor padrão da biblioteca PyQt5
        super().setUpUi(self) # Chama o construtor setUpUi do design.py

        # Conexão dos botões da tela do ADM com suas respectivas funções
        self.btnAdmCadastrar.clicked.connect(self.CadastrarProduto)
        self.btnAdmAtualizar.clicked.connect(self.AtualizarProduto)
        self.btnAdmBuscarAtualizar.clicked.connect(self.BuscarProduto)
        self.btnAdmAdicionarUser.clicked.connect(self.CadastrarUser)
        self.btnAdmHistorico.clicked.connect(self.BuscarHistorico)
        self.btnAdmEstoque.clicked.connect(self.BuscarEstoque)
```

- Métodos de classe: criamos métodos específicos para que as classes Adm, Main e Login pudessem retirar, manipular e tratar os dados do frontend do sistema;

```
self.btnLoginCancelar.clicked.connect(self.Fechar)
self.btnLoginLogin.clicked.connect(self.Verificar)

def Verificar(self):
    login = self.txtLoginLogin.text()
    senha = self.txtLoginSenha.text()

    if Verificar_User(login, senha):
        self.Abrir_Adm()
    else:
        self.txtLoginMessage.setText("Senha ou Login inválido")
```

Neste exemplo, a classe Login possui o método Verificar, que é chamado quando o botão Login é pressionado. Esse método guarda o valor dos campos de login e senha como strings e os envia para a função de verificação. Se a função voltar verdadeira, o método chama outro método da classe Login, Abrir_Adm, para fechar a tela de login e mostrar a tela do Adm ao usuário.

Papel de cada integrante

Rafael: Desenvolvimento dos designs das telas no framework Qt Designer e criação das conexões com o banco de dados por meio do context manager apresentado no vídeo.

Thiago: Desenvolvimento do código para gerenciar as telas de Adm e Login.

Victor: Desenvolvimento do código para gerenciar as telas de Main e Login.

Links

Link do Github da aplicação: <https://github.com/RafaelSolVargas/BakerySystem>

Link do vídeo da apresentação: <https://youtu.be/5WHmVwhNo1s>