Sistesma de gestão de padaria

O sistema criado pelos alunos da escola técnica do COTUCA, vem com novas páginas na interface

- Home (inicial)
- Cadastro de Funcionarios
- Vendas
- Sobre
- Desenvolvedores

Atualizações

Para cada página de gestão, foi criado suas funções dentro do projeto "main.py" que será inicializada na interface e depois caso precise de alguma conexão ou ação do banco de dados, é inicializada a conexão do banco de dados para fazer suas ações, como:

Module

Funções do arquivo module.py

- Login (Para inicializar a Conexão com banco de dados)
- closeConnection (para fechar a conexão com o banco de dados)
- LoginAuthentication (Para validar o usuario com seu registro para acessar a interface)
- insertTableFun (Cadastrar um funcionario no banco de dados)
- addProduct (adicionar produtos na tabela "padaria.venda")
- search (Pesquisar o valor de um produto)
- showTable (Inserir valor/dados em qualquer tabela)
- deleteFun (Deletar um funcionario)
- updateTableFun (Recarregar a janela de funcionario)

Main

Funções do arquivo main.py

- connectDatabase (Ele irá iniciar uma conexão com o banco de dados quando clicar no botão de "Venda" ou "Cadastra funcionario")
- deleteRows (Deleta a quantidade de produtos dentro carrinho)
- search (Ao clicar no botão de pesquisar ele irá pesquisar valores que o usuario colocou)
- confirmSale (Confirmar a venda e coloca na tabela de padaria.venda no banco de dados)
- cancelSale (Cancela a venda e exclui os produtos dentro do carrinho)
- addProduct (adiciona um produto para dentro do carrinho)
- calculeTotal (Calcula o valor de acordo com o que existe dentro do carrinho e mostra para o funcionario)
- employeeRegistration (Registrar funcionario de acordo com o que foi colocado)
- deleteFun (deletar um funcionario no banco de dados)
- updateTableFun (Atualizar a tabela de funcionario para mostrar para o funcionario)
- refreshTable (atualiza todas as tabelas do banco de dados e mostra para o funcionario)

- closeWindow (Além de fechar a janela, ela fecha a janela com o banco de dados.)

Ferramentas

Ferramentas utilizada para criação do software

- Python - Linguagem de programação utilizada para criar esse software
- VScode - IDE utilizada para fazer a criação do sistema
- QTdesigner - Aplicativo para criar interfaces em formato .UI
- pyodbc - pyodbc é um módulo Python de código aberto que simplifica o acesso a bancos de dados ODBC
- PySide6 - PySide6 é o módulo Python oficial do projeto Qt for Python, que fornece acesso ao framework Qt 6.0+ completo.
- SQL Server Management Studio - O SQL Server Management Studio (SSMS) é um ambiente integrado para gerenciar qualquer infraestrutura SQL.

Instalação

Vá até o site do python Python, faça a instalação do software

Faça um clone do nosso projeto no Github para sua máquina local pelo terminal.

```
git clone https://github.com/projeto-padaria/Padaria.git
```

Instale as dependecias usando o pip

```
pip install -r requirements.txt
```

Logo após isso inicie o projeto digitado isso

```
python .\source\middlewares\login.py
```

> *Nota do desenvolvedor Não criamos um arquivo executavel do Windows, poís seria complciado.*

Banco de dados

A criação do banco de dados é inicializada antes de fazer a requisições utilizada, para isso utilizado o servidor de banco de dados do COTUCA.

```
-- Está em ordem de criação das tabelas e relacionamentos

create schema padaria;

create table padaria.endereco
(
    idEndereco int identity(1,1),
    bairro varchar(100)  null,
    rua varchar(100)  null,
    numero int  null,
    cidade varchar(50)  null,
    UF char(2)  null,
    cep varchar(8)  null,
    PRIMARY KEY (idEndereco)
```

```
);


create table  padaria.fornecedor
(
    idFornecedor int identity(1,1),

    cnpj varchar(14),
    status varchar(50),
    nome varchar(50),
    telefone char(11),
    idEndereco int null,
    UNIQUE(cnpj),
    PRIMARY KEY (idFornecedor),

    FOREIGN KEY (idEndereco) REFERENCES padaria.endereco(idEndereco) on update CASCADE
);

create table  padaria.cliente
(
    idCliente int identity(1,1),

    nome varchar(50),
    sobrenome varchar(50),
    cpf varchar(11),
    idEndereco int  null,
    UNIQUE(cpf),
    PRIMARY KEY (idCliente),

    FOREIGN KEY (idEndereco) REFERENCES padaria.endereco(idEndereco) on update CASCADE
);

create table  padaria.funcionario
(
    idFuncionario int identity(1,1),

    cpf varchar(11),
    nome varchar(50),
    sobrenome varchar(50),
    senha varchar(40),
    cargo varchar(50),
    salario decimal(10,2),
    telefone varchar(11),
    idEndereco int null,
    UNIQUE(cpf),

    PRIMARY KEY (idFuncionario),

    FOREIGN KEY (idEndereco) REFERENCES padaria.endereco(idEndereco) on update CASCADE
);

create table  padaria.produto
```

```
(
    idProduto int identity(1,1),
    idFornecedor int ,

    nome varchar(50),
    marca varchar(50),
    preco decimal(10,2),
    quantidade int,
    datadeemissao date,
    datadevalidade date,
    UNIQUE(nome),

    PRIMARY KEY (idProduto),

    FOREIGN KEY (idFornecedor) REFERENCES padaria.fornecedor(idFornecedor) on update
CASCADE
);

create table  padaria.venda
(
    idVenda int,
    idCliente int,
    idProduto int,
    idFuncionario int,

    quantidade int,
    data datetime,

    PRIMARY KEY (idVenda, idCliente, idFuncionario, idProduto,data),

    FOREIGN KEY (idCliente) REFERENCES padaria.cliente(idCliente) on update CASCADE,

    FOREIGN KEY (idFuncionario) REFERENCES padaria.funcionario(idFuncionario) on
update NO ACTION,

    FOREIGN KEY (idProduto) REFERENCES padaria.produto(idProduto) on update NO ACTION,

)
```

Debug

Foi criado uma classe de debug para enviar alguns comandos apenas para desenvolvedores para ver os logs.

```
class libDebug():
    def __init__(self) -> None:
        self.BOLD = "\033[1m"
        self.RED = "\033[31m"
        self.GREEN = "\033[32m"
        self.RESET = "\033[0m"
        self.ORANGE = "\033[33m"
```

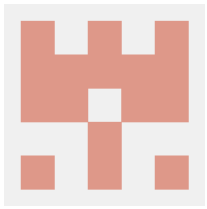
```
    def printError(self, error) -> None:
        return print(f"{self.BOLD}{self.RED}ERROR: {self.RESET}{error}")

    def printWarning(self, error) -> None:
        return print(f"{self.BOLD}{self.ORANGE}WARNING: {self.RESET}{error}")

    def printSuccess(self, success) -> None:
        return print(f"{self.BOLD}{self.GREEN}SUCCESS: {self.RESET}{success}")

    def printTitle(self, title) -> None:
        menubar = "—" * len(title)
        return print(f"{self.BOLD}{menubar} {title} {menubar} {self.RESET}")
```

Colaboradores

			
Rafael Moreira	**Gabriel Oliveira**	**Pedro Henrique**	**Lucas Siqueira**

Licença

Este projeto foi desenvolvido pelos alunos do Colégio Técnico de Campinas (Cotuca) e é licenciado sob a Licença MIT.

A Licença MIT é uma licença de software permissiva que permite a utilização, modificação e distribuição do código fonte deste projeto, seja para fins comerciais ou não, desde que a nota de direitos autorais e a licença sejam incluídas em todas as cópias do software.

Para mais informações, consulte o arquivo LICENSE.