UNIFEOB
ESCOLA DE NEGÓCIOS

2024

PROJETO INTEGRADO

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

DESENVOLVIMENTO DE SOLUÇÕES CONSOLE INTEGRADAS PARA EDUCAÇÃO, SUSTENTABILIDADE, INCLUSÃO SOCIAL E EMPREENDEDORISMO

GA PEREIRA MATERIAIS DE CONSTRUÇÃO

SÃO JOÃO DA BOA VISTA, SP

NOVEMBRO 2024

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

DESENVOLVIMENTO DE SOLUÇÕES CONSOLE INTEGRADAS PARA EDUCAÇÃO, SUSTENTABILIDADE, INCLUSÃO SOCIAL E EMPREENDEDORISMO

GA PEREIRA MATERIAIS DE CONSTRUÇÃO

MÓDULO MODELAGEM E DESENVOLVIMENTO DE SISTEMAS

Business Intelligence – Profª. Mariângela Martimbianco Santos

Programação Orientada a Objeto – Prof. Nivaldo de Andrade

Lógica de Programação – Prof. Marcelo Ciacco Almeida

Modelagem de Dados – Prof. Max Streicher Vallim

Projeto de Modelagem e Desenvolvimento de Sistemas – Profª. Mariângela M. Santos

Estudantes:

Enzo Daniel Abreu, RA 24001463
Gabriel da Silva Freitas, RA 24001078
José Carlos Pereira Neto, RA 24000209
Lucas Paulino Gomes, RA 24000580
Thierry Antonello Pengo, RA 24000073
Vinícius Tenti de Araújo Perri, RA 24000350

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2024

SUMÁRIO

1. INTRODUÇÃO

Gomes (2023) define CLI como a sigla para "Command-Line Interface" ou "Interface de Linha de Comando". Essencialmente, CLIs são interfaces baseadas em texto que permitem aos usuários interagir diretamente com um sistema, executando tarefas por meio de comandos. Originadas entre as décadas de 1960 e 1970, quando os recursos gráficos eram limitados, as CLIs eram as principais formas de interação com as máquinas.

Entretanto, com o avanço da tecnologia, as interfaces gráficas (GUIs) começaram a ganhar espaço a partir dos anos 1980, especialmente com o lançamento de sistemas como o Apple Macintosh e o Microsoft Windows. Essas interfaces, que utilizam ícones, menus e janelas, se tornaram populares por serem mais intuitivas e acessíveis para o usuário comum, permitindo a interação com computadores sem a necessidade de conhecer comandos complexos.

Apesar do crescimento e popularidade das GUIs, as CLIs continuam sendo ferramentas poderosas e altamente versáteis. Em um console de desenvolvimento, que é um ambiente de texto onde comandos são digitados, os usuários podem realizar tarefas de forma eficiente e precisa. Embora para alguns essa abordagem possa parecer menos moderna, ela oferece vantagens importantes, como maior controle, automação e personalização. Por isso, as CLIs ainda são amplamente usadas em contextos técnicos, como desenvolvimento de software, administração de sistemas e, em especial, na gestão de informações empresariais, onde a eficiência e o controle detalhado são essenciais.

Nesse sentido, a solução proposta para a GA Pereira consiste no desenvolvimento de uma aplicação via console, utilizando uma interface de linha de comando (CLI) intuitiva escrita na linguagem de programação Python. O console de desenvolvimento servirá como a interface principal para os usuários, onde os comandos da CLI serão executados. Essa abordagem permite uma interação direta e eficiente com o sistema, facilitando a gestão de informações e a automatização de tarefas. O aplicativo de console também fornecerá uma conexão a um banco de dados via SQL, possibilitando o controle e registro de produtos e compras de forma eficiente.

2. DESCRIÇÃO DA EMPRESA

A empresa para a qual nosso grupo prestará assistência tem como razão social GA Pereira Materiais para Construção LTDA, registrada sob o CNPJ 04.282.801/0001-48, com sede localizada na Rua José Rosa Pereira, nº 301, no município de Tapiratiba, estado de São Paulo.

A principal atividade da empresa consiste na comercialização de materiais de construção, além da produção de blocos e postes. Os materiais são adquiridos de fornecedores terceirizados, e a entrega dos produtos é realizada pela própria empresa. Para clientes que realizam compras a prazo na loja, é criada uma ficha em que todas as transações são registradas. Essas fichas podem ser transferidas para o sistema informatizado, onde são utilizadas para a emissão diária das notas fiscais. A partir das notas fiscais, é possível gerenciar as vendas, associando os produtos aos respectivos clientes.

Diante desse cenário, a loja necessita de uma solução mais eficiente para registrar as fichas dos clientes, gerenciar o estoque de produtos e acessar essas informações de forma ágil e organizada. Assim, nosso grupo propõe o desenvolvimento de uma aplicação em console que permitirá a execução de todas as ações mencionadas.

3. PROJETO INTEGRADO

3.1 PROGRAMAÇÃO ORIENTADA A OBJETO

Segundo Brookshear (2013, p. 216), o paradigma da programação orientada a objetos é um sistema no qual se utilizam entidades denominadas objetos, que representam elementos do mundo real ou fictício e podem interagir com outros objetos, modificando-os para solucionar um problema. Essa abordagem permite a modelagem de sistemas complexos de forma mais intuitiva, facilitando a compreensão e manutenção do código. Por meio de conceitos como encapsulamento, herança e polimorfismo, os desenvolvedores podem criar soluções mais flexíveis e reutilizáveis. Assim, a programação orientada a objetos se torna uma ferramenta poderosa para resolver problemas práticos, promovendo a organização e a modularidade do software.

Outrossim, é importante notar a relação entre POO (Programação Orientada a Objetos) e outras disciplinas deste projeto, como BI (Business Intelligence) e Modelagem de Dados. Segundo Almeida (2023), o uso de POO no contexto da Engenharia de Dados permite a organização de sistemas complexos e a reutilização de código. Em soluções de BI, a POO facilita o desenvolvimento de sistemas modulares, flexíveis e de fácil manutenção, o que é crucial para o tratamento de grandes volumes de dados e para responder de maneira ágil às necessidades do negócio.

3.1.1 CLASSES E OBJETOS

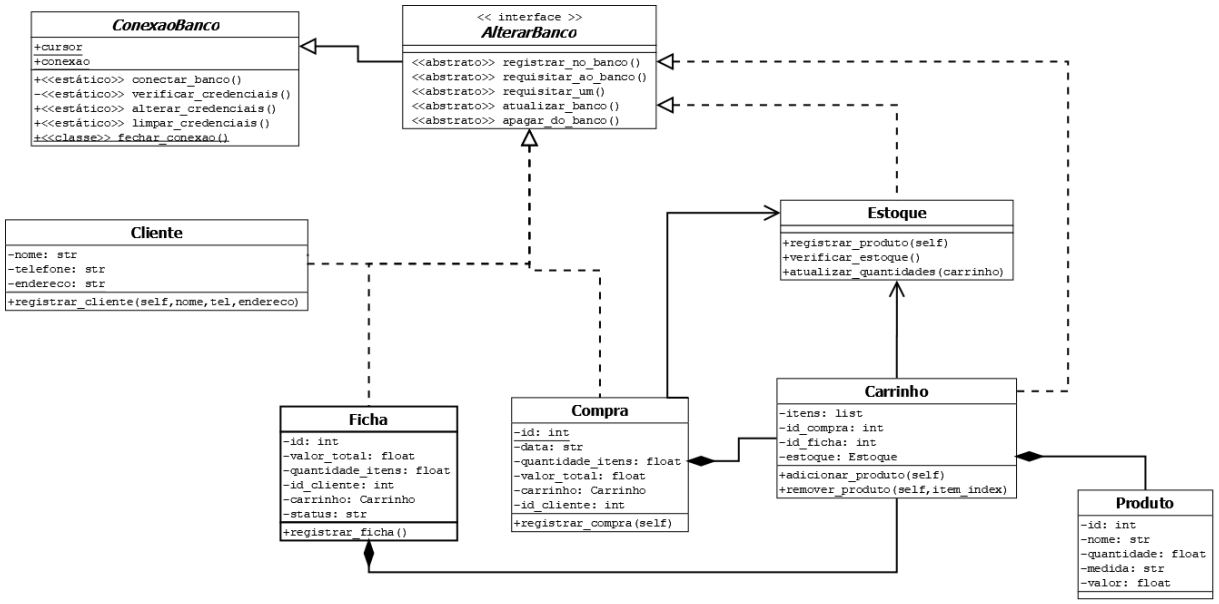
De acordo com Brookshear (2013, p. 249), classe é o termo utilizado para se referir a um modelo a partir do qual os objetos são criados.

Em nosso console de desenvolvimento, na codificação orientada a objetos, foram construídas várias classes, cada uma representando uma entidade presente no sistema da empresa selecionada.

Para melhor representar as classes do código e as relações entre si foi utilizado um diagrama UML que, segundo VanZandt (2022),

> É definido como um modelo dinâmico que facilita a visualização de processos e sequências. Essa representação visual captura meticulosamente os elementos essenciais de um sistema, incluindo atores, funções, ações e artefatos.
> Com o objetivo principal de não apenas aprimorar a compreensão, mas também de facilitar alterações, manutenção e documentação abrangente de informações cruciais do sistema, os diagramas UML são uma ferramenta indispensável no design e desenvolvimento de software moderno.

Figura 1 - Diagrama UML das classes utilizadas



Fonte: Autores

Cada caixa no diagrama acima representa uma classe, exceto a AlterarBanco. A classe ConexaoBanco é responsável por conectar o programa a um banco de dados, além de ser abstrata, isto é, um tipo de classe que, segundo Pinheiro (2020), não pode ser instanciada, apenas herdada. Já AlterarBanco é uma interface, pois, segundo DevMedia (2010), representa um "contrato" entre uma classe e o ambiente externo e ao implementar uma interface, a classe compromete-se a oferecer o comportamento definido por essa interface, cumprindo assim as especificações e funcionalidades que ela estabelece; os métodos em questão representam as operações que serão realizadas com o banco de dados. As demais classes atuam diretamente no programa, em contato com o usuário, pois elas são as responsáveis por gerenciar, registrar e consultar seus itens homônimos.

As setas representam os relacionamentos entre as classes e demonstram a estrutura do programa, a qual, em resumo, apresenta-se da seguinte maneira: a interface AlterarBanco herda de "ConexaoBanco", para que as classes que implementarem a interface possam

partilhar da conexão com o banco de dados, porque ao implementá-la, também herdam. As classes Cliente, Ficha[1], Compra, Estoque e Carrinho implementam a AlterarBanco, pois são classes que realizam operações no banco de dados. As classes Compra e Carrinho estão associadas à classe Estoque, ou seja, possuem métodos que utilizam um objeto do tipo Estoque. Isso significa que há um relacionamento entre duas ou mais classes, no qual uma classe usa objetos de outra como parte de suas operações (Oliveira, 2023). Por fim, a classe Produto compõe a classe Carrinho, ou seja, um carrinho é formado por produtos. Da mesma forma, a classe Carrinho compõe as classes Compra e Ficha, pois ambas dependem de um carrinho para existir. Isso exemplifica o conceito de composição, uma relação onde as partes não podem existir sem o todo, ou seja, as partes são totalmente dependentes do objeto principal (Damião, 2018). Assim, quando o carrinho é removido, seus produtos e as compras ou fichas associadas também deixam de existir, refletindo a forte dependência entre os componentes e o objeto principal.

Note que, nesse contexto, o carrinho é análogo aos objetos de mesmo nome utilizados nos supermercados: um contêiner no qual são inseridos todos os produtos de uma determinada compra.

3.1.2 ATRIBUTOS, MÉTODOS, ENCAPSULAMENTO, HERANÇA E POLIMORFISMO

De acordo com Henrique (2023), atributos são as características de um objeto, ou seja, os dados associados a ele. Já os métodos correspondem aos comportamentos realizados pelo objeto. Além disso, é interessante fazer uso de alguns pilares da POO.

Segundo Clemente (2024), herança, na programação orientada a objetos, é um conceito que permite a uma classe herdar atributos e comportamentos de outra. Esse recurso favorece o reaproveitamento do código, torna a manutenção mais simples e possibilita a criação de uma estrutura hierárquica entre os objetos, organizando-os de maneira lógica e eficiente. Também, segundo ele, o polimorfismo permite que objetos de diferentes classes sejam manipulados como se pertencessem a uma classe comum, mantendo a capacidade de cada um de realizar suas próprias operações específicas.

[1] Uma ficha, no contexto da empresa beneficiada por este projeto, é uma compra na qual o cliente recebe os produtos, mas cujo pagamento é adiado por um tempo indefinido, sendo registrada num documento que contém os produtos retirados e os dados do cliente para contato.

“Outra característica associada à programação orientada a objetos é o encapsulamento, que se refere à restrição de acesso às propriedades internas de um objeto.” (Brookshear 2013, p. 254).

Em nosso projeto, os pilares essenciais da POO foram implementados no código. A herança foi explicada no tópico anterior. A maior parte das classes contém atributos privados, o que demonstra o uso do encapsulamento, sendo a classe ConexaoBanco a única com atributos públicos, por serem atributos de classe, ou seja, disponíveis para toda instância da classe em questão, ou de suas subclasses. Já o polimorfismo foi trabalhado de duas formas: a primeira é a implementação dos métodos da interface AlterarBanco, apresentada na figura 2, pois cada classe tem o mesmo método operando de forma ligeiramente diferente; a segunda é através de um método presente nas classes que interagem com o banco de dados diretamente, chamado menu(). Esse método é executado através de uma função abrir_menu(), que recebe um objeto qualquer e chama o menu, como mostrado no anexo C.

Figura 2 - Código da interface

```python
from abc import ABC, abstractmethod

from .conexao_banco import ConexaoBanco

# ^ interface
class AlterarBanco(ConexaoBanco, ABC):
    @abstractmethod
    def registrar_no_banco(self):
        pass

    @abstractmethod
    def requisitar_ao_banco(self):
        pass

    @abstractmethod
    def requisitar_um(self):
        pass

    @abstractmethod
    def atualizar_banco(self):
        pass

    @abstractmethod
    def deletar_do_banco(self):
        pass
```

Fonte: Autores

3.1.3 MÉTODOS ESTÁTICOS, PÚBLICOS E PRIVADOS

DevMedia (2012) define métodos públicos como elementos que podem ser acessados livremente por qualquer parte do código, tanto dentro quanto fora da classe. Eles são utilizados para expor funcionalidades que precisam estar disponíveis para outros módulos ou classes, permitindo que a classe interaja com o restante do sistema. A definição de métodos privados, por sua vez, descreve elementos da classe que só podem ser acessados pelo código interno da própria classe. Esses métodos são projetados para encapsular a lógica e os dados que não devem ser expostos diretamente, garantindo que funcionalidades internas fiquem protegidas contra modificações externas. Além disso, DevMedia define métodos estáticos como aqueles utilizados para criar uma variável ou função que pode ser acessada diretamente pela classe, sem a necessidade de criar uma instância de objeto. Isso significa que a variável será a mesma para todas as instâncias da classe e qualquer alteração nela afetará todas as instâncias. Quanto aos métodos, o fato de serem estáticos implica que não é necessário criar um objeto para chamá-los, pois eles pertencem à classe e podem ser acessados diretamente por ela. Isso facilita o acesso a comportamentos comuns à classe como um todo.

Os métodos responsáveis por realizar operações no banco de dados são públicos, bem como os métodos de registro das classes Compra, Estoque, Cliente e Ficha. Além disso, o método menu() mencionado anteriormente também é público. As classes ConexaoBanco e Carrinho têm métodos privados: __mesclar_produtos_iguais() e __verificar_credenciais(), respectivamente, por não precisarem (nem poderem) ser chamados fora da classe.

A ConexaoBanco não contém nenhum método de instância, ao contrário das outras classes. O seu método conectar_ao_banco() é um método estático, responsável por criar a conexão e definir seus dois atributos de classe: banco, que armazena a conexão, e cursor, por onde serão realizadas as operações no banco. Além deste, há outros métodos estáticos responsáveis por ler e gerenciar as credenciais utilizadas para acessar o sistema do banco: o já mencionado __verificar_credenciais(), alterar_credenciais() e limpar_credenciais(). Apenas o primeiro é privado, pois ele ocorre apenas dentro da classe, enquanto que os dois últimos podem ser acessados de outras partes do código. Por fim, há um método de classe chamado fechar_conexao(), que desliga a conexão entre o programa e o banco de dados. A classe ConexaoBanco pode ser conferida no anexo A.

3.2 LÓGICA DE PROGRAMAÇÃO

Segundo Bessa (2023), a ordem que deve ser respeitada para alcançar o resultado desejado, ou seja, essa cadeia lógica é denominada lógica de programação. Essa lógica é fundamental no desenvolvimento de software, pois permite a criação de algoritmos que definem claramente os passos a serem seguidos para resolver um problema. Além disso, conceitos como variáveis, que armazenam dados; estruturas de controle, que permitem a tomada de decisões e a repetição de ações; e operadores, que realizam operações sobre os dados, são essenciais para a organização e eficiência do código. Juntos, esses elementos formam a base para a construção de programas que são não apenas funcionais, mas também escaláveis e manuteníveis.

3.2.1 CONCEITOS FUNDAMENTAIS DO DESENVOLVIMENTO DE SOFTWARE

Citando Alves (2013, p.12)

> O algoritmo pode ser entendido como uma descrição textual de uma solução para um determinado problema. Ele descreve um processo em termos de outros processos básicos previamente definidos. Um algoritmo pode ainda ser descrito como uma sequência lógica de etapas ou procedimentos que transformam uma entrada de dados em uma saída válida.

Ou seja, o algoritmo é um conjunto de instruções claras e organizadas que visa à solução de um problema específico, orientando o processamento de informações de maneira eficiente. Ele determina, de forma estruturada, as ações necessárias para converter os dados iniciais em um resultado final, garantindo a precisão e a consistência na execução das tarefas.

Além disso, Alves (2013, p.29) define funções como estruturas que recebem valores como parâmetros e retornam um resultado baseado nas operações realizadas sobre esses valores.

Tipos de dados primitivos são as unidades básicas de armazenamento de dados em linguagens de programação. Eles representam valores simples e ocupam um espaço fixo na memória, variando conforme o tipo. Esses tipos são usados para armazenar informações essenciais e são manipulados diretamente pelo processador, o que garante maior eficiência.Os principais tipos de dados primitivos incluem:

- **Inteiros (int)**: usados para armazenar números inteiros, positivos ou negativos.
- **Ponto flutuante (float, double)**: utilizados para representar números com parte decimal.
- **Caractere (char)**: armazena um único caractere, como uma letra ou símbolo.
- **Booleano (boolean)**: representa valores lógicos, como verdadeiro ou falso.

Na memória, os inteiros e caracteres geralmente ocupam 1 a 4 bytes, enquanto os valores de ponto flutuante ocupam 4 a 8 bytes, dependendo da precisão. Os booleanos ocupam 1 byte. Essas variáveis são essenciais para a realização de operações matemáticas, controle de fluxo lógico, e representação de dados simples dentro dos programas.

Outrossim, segundo Alves (2013, p. 30)

> Para armazenar esses valores, o computador utiliza posições específicas da memória RAM, a memória de uso temporário. Em nossos programas, declaramos variáveis que fazem referência a essas posições reservadas, atribuindo-lhes nomes e definindo os tipos de dados que elas podem receber. É como na Matemática, em que temos variáveis nas equações.

Dessa forma, o autor enfatiza a relação direta entre a programação e a lógica matemática, onde as variáveis desempenham o papel fundamental de armazenar e manipular informações, permitindo a execução de operações e cálculos de forma organizada e eficiente.

Ainda, segundo Alves (2013) não adiantaria armazenar dados em variáveis de memória ou em um banco de dados se os programas não fossem capazes de manipulá-los, ou seja, se não fosse possível realizar cálculos, comparações ou outras operações. Nesse contexto, ele menciona os operadores, que permitem a execução de diversas operações sobre os dados. Esses operadores incluem operações aritméticas, como soma, subtração, divisão e multiplicação; operações de comparação, como maior que, menor que, maior ou igual a, e menor ou igual a; e operações lógicas, como E, OU e NÃO.

A partir dessas operações, surgem as estruturas condicionais, que são fundamentais na programação. Elas utilizam os resultados das comparações e operações lógicas para permitir que um sistema tome decisões com base em diferentes cenários. Assim, guiam o fluxo de execução de um programa, alterando seu comportamento de acordo com as condições estabelecidas. Nesse sentido, conforme Alves (2013), “As principais estruturas de decisão são SE/ENTÃO, SE/ENTÃO/CASO CONTRÁRIO e FAÇA CASO.”

3.2.2 DESENVOLVIMENTO DE APLICAÇÕES

Segundo a IBM, “As regras de negócios orientam a tomada de decisões diárias dentro das empresas, delineando as relações entre objetos, como os nomes dos clientes e seus respectivos pedidos.”

Regras de negócios usam lógica formal (como "IF-THEN", "IF-ELSE") para automatizar decisões e são divididas em tipos principais: regras de restrição, que estabelecem condições para operações e estruturas, e regras de derivação, que inferem conclusões de outras informações. Juntas, essas regras permitem que organizações padronizem processos e decisões, otimizando operações como pedidos e envios de mercadorias.

No caso do nosso projeto integrado, foram criadas regras de restrição que visam garantir a consistência e segurança dos dados, bem como oferecer benefícios automáticos aos clientes e melhorar o controle do estoque da loja. Essas restrições foram implementadas com base nas necessidades específicas do negócio, assegurando que as operações sejam realizadas dentro dos parâmetros definidos.

Requisito: Cadastro de compras
Regras:

- As compras não podem ser vazias;
- As compras não podem exceder a quantidade de produtos disponíveis;
- Compras que valem mais de x reais e são pagas à vista recebem desconto.

Requisito: Cadastro de fichas
Regras:

- Aplica as regras das compras;
- Fichas não recebem desconto.

Requisito: Cadastro de produtos no estoque
Regras:

- É possível que não haja determinado produto no estoque;
- O estoque não pode ficar completamente vazio.

Requisito: Cadastro de clientes
Regras:

- Nenhum campo do cliente pode estar em branco;
- Clientes com mais de x compras na loja recebem desconto;
- Clientes com mais de x anos de frequência recebem desconto.

3.2.3 IMPLEMENTAÇÃO E VALIDAÇÃO

Durante a implementação do código, foi utilizado o conceito de modularidade, que segundo Brookshear (2013), é “a divisão de software em partes gerenciáveis”, tendo sido alcançada através da divisão das responsabilidades entre classes, de forma que uma classe não realize as tarefas de outra. Cada classe foi colocada em um arquivo separado para facilitar a organização. Além disso, foram feitos diversos testes para detectar erros, os quais foram resolvidos com a refatoração (reescrita) do código e com o uso de comandos para tratamento de exceções, como TRY-EXCEPT, em Python (Perkovic, 2016, p.230). O código implementado e testado é uma ferramenta eficiente para registrar e consultar dados relevantes para a empresa, como compras efetuadas, produtos vendidos e clientes.

3.3 MODELAGEM DE DADOS

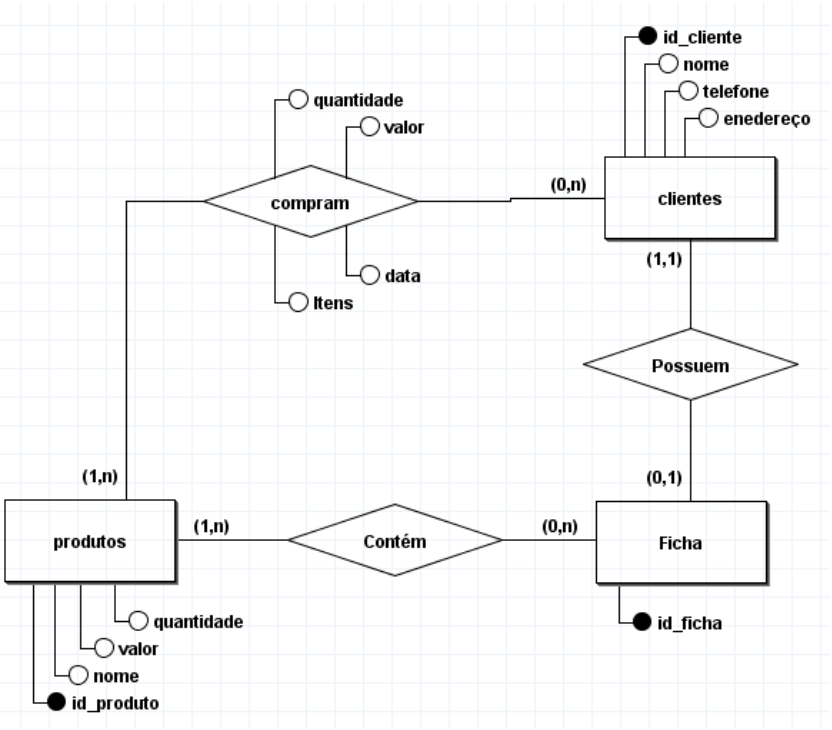
3.3.1 MODELO CONCEITUAL

De acordo com Devmedia (2014), “ modelos de dados conceituais são diagramas de alto nível que representam os conceitos de dados que suportam o negócio de uma empresa, uma área de negócio ou, por exemplo, um sistema de informações. Em projetos de TI, o objetivo principal de um modelo de dados conceitual é fornecer uma visão geral dos requisitos de informação envolvidos no projeto.” Nesse contexto, o Diagrama Entidade-Relacionamento (DER) é um exemplo de modelo conceitual que ilustra como as diferentes entidades (como clientes, produtos e compras) se relacionam entre si dentro de um sistema. O DER utiliza símbolos gráficos para representar essas entidades e as conexões (relacionamentos) entre elas, oferecendo uma visão clara e organizada dos dados que o sistema precisará gerenciar.

Para o projeto de nossa equipe, desenvolvemos um diagrama DER para ilustrar como o sistema do banco de dados irá funcionar, demonstrando as relações entre as entidades, seus atributos — características que descrevem as propriedades ou qualidades específicas de cada entidade, como nome, idade ou data — e as cardinalidades presentes em cada relação, ou seja,

o número de instâncias de uma entidade que podem estar associadas a instâncias de outra entidade.

Figura 3 - Modelo Conceitual



Fonte: Autores

Nesse modelo conceitual, há a presença de três entidades que representam as principais partes para o desenvolvimento do banco de dados. Além disso, há presença de atributos e cardinalidades.

Ao analisar o diagrama DER no qual o modelo se insere, percebe-se que os clientes devem comprar pelo menos um produto, mas podem adquirir vários, enquanto os produtos podem ser comprados por nenhum ou vários clientes. Além disso, os clientes podem possuir nenhuma ou no máximo uma ficha, sendo que cada ficha deve obrigatoriamente pertencer a um cliente.

Em relação aos atributos de cada entidade, é possível notar que clientes possuem nome, endereço e telefone, Produtos possui os atributos nome, valor e quantidade e Ficha só possui seu atributo de identificação (ID). Outrossim, a relação de ‘comprar’ entre clientes e produtos possui os atributos de quantidade, itens, valor e data.

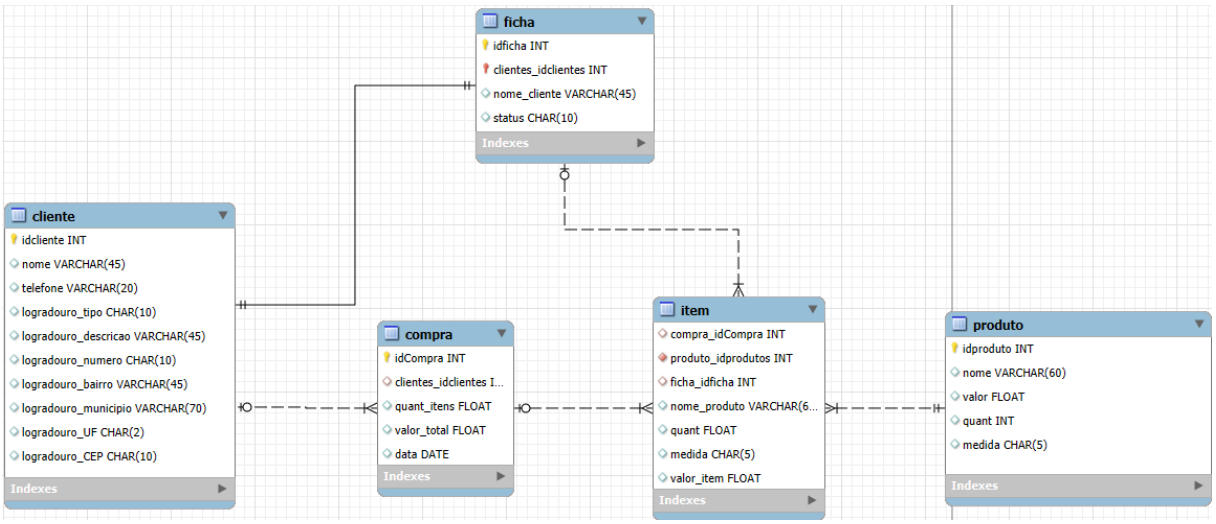
Cada uma das entidades mencionadas possui seu próprio atributo de identificação (ID), que funciona como chave primária, garantindo a unicidade de cada registro no banco de dados.

Conclui-se, portanto, que o modelo conceitual elaborado organiza eficientemente as principais entidades do sistema, suas características e as relações entre elas, facilitando a estruturação de um banco de dados robusto e funcional para a aplicação.

3.3.2 MODELO LÓGICO E FÍSICO

Barbosa e Freitas (2018, p. 109) definem o modelo lógico como a representação dos dados armazenados e seus relacionamentos, indicando as relações existentes entre os dados e, assim, apresentando a estrutura lógica dessas informações. Também definem o modelo físico como a descrição, no nível mais baixo de representação, da forma como os dados estão fisicamente armazenados.

Figura 4 - Modelo Lógico



Fonte: Autores

Conforme mencionado no tópico 3.3.1, apenas as entidades 'clientes', 'fichas' e 'produtos' foram utilizadas para otimizar o desenvolvimento do protótipo. No modelo lógico, as entidades são chamadas de tabelas, pois possuem um formato tabular com seus respectivos atributos. As entidades 'compra' e 'item' surgiram de relações (n,m) entre 'clientes' e 'item', e entre 'ficha' e 'produto', respectivamente.

Além disso, vale destacar a relação (1,1) entre 'clientes' e 'ficha', já que cada cliente possui uma ficha específica, e cada ficha pertence a um cliente exclusivo.

Conclui-se, portanto, que o modelo lógico reflete de forma clara e eficiente as relações entre os dados, permitindo uma estrutura coerente para o armazenamento e a manipulação das informações no sistema.

Após a criação do modelo lógico, criamos o modelo físico de nosso sistema, que consiste na implementação prática das estruturas e relações definidas anteriormente, traduzindo-as para um banco de dados real. O modelo físico descreve os detalhes técnicos de armazenamento, incluindo tabelas, colunas e tipos de dados, otimizando o uso dos recursos disponíveis para que o sistema funcione de maneira eficiente e confiável. Essa transição é essencial para assegurar que o modelo lógico se concretize em um ambiente de banco de dados capaz de suportar as operações do sistema com integridade e desempenho.

Figura 5 - Exemplo de Tabela do Modelo Físico

idproduto	nome	valor	quant	medida
1	areia fina	152	0	M
2	areia grossa	116	0	M
3	serrinha	5	0	UN
4	arame	9.5	0	UN
5	saco de pregos 18x27	13	0	UN
6	saco de cimento	36	10	UN
7	telha romana	2	646	UN
8	saco de prego 15x15	22	4	UN
9	saco de prego 19x36	13	0	UN
10	saco de prego 20x48	13	1	UN
11	tijolinho comum	1	7	UN
12	areia de barranco	86	0	M
13	pedra	116	2	M
14	saco de cal	20	0	UN
15	tijolinho bloquinho	2	0	UN
16	bara de ferro 3/16	35	0	UN
17	luva 3/4	1	20	UN
18	fita isolante	4.5	11	UN
19	canaleta de 15	2.5	1134	UN
20	bloco de 15	2.5	1827	UN
21	canaleta de 11	2	2195	UN
22	bloco de 11	2	1405	UN
23	canaleta de 20	2.9	2876	UN
24	bloco de 20	2.2	1488	UN

Fonte: Autores

A tabela apresentada na figura 5 representa o estoque da GA Pereira, contendo os produtos cadastrados com informações detalhadas, como ID do produto, nome, quantidade disponível e unidade de medida. Esses dados facilitam a gestão e o controle do estoque, permitindo um monitoramento preciso conforme o tipo e a quantidade de cada item armazenado.

Observa-se que alguns produtos apresentam quantidades igual a zero, o que indica a ausência desses itens no estoque.Por meio do comando INSERT, descrito no tópico 3.3.3, será

possível registrar novas entradas de estoque assim que os produtos estiverem disponíveis, integrando essas atualizações diretamente na aplicação. A operação será implementada usando programação orientada a objetos (POO), permitindo que o gerenciamento do estoque seja realizado de forma organizada e facilitando futuras modificações e manutenções no código.

3.3.3 SQL

Na criação de um banco de dados MySQL, e consequentemente do modelo físico, são utilizados comandos principais que desempenham funções essenciais para a construção e visualização do banco de dados.

O comando INSERT, conforme definido por Damas (2007), é usado para criar e armazenar novos registros no banco, permitindo a inserção de múltiplos registros de uma só vez ou de dados específicos em determinadas colunas. O comando UPDATE possibilita a alteração e atualização de dados de maneira simples, com base nas condições definidas pelo usuário. A instrução SELECT permite visualizar todos os dados armazenados e também selecionar apenas os dados que atendam a critérios específicos do usuário. Finalmente, o comando DELETE é utilizado para manter o banco de dados organizado e consistente, removendo dados desnecessários.

Para o projeto de nossa equipe, foram utilizados todos os comandos mencionados anteriormente; porém, como consta no anexo F do script SQL, percebe-se apenas o uso dos comandos INSERT, SELECT e UPDATE.

Inicialmente, utilizou-se o comando `INSERT` para registrar os produtos que estarão disponíveis para aquisição. As informações essenciais para essa inserção incluem o nome do produto, o preço e a quantidade disponível. Em seguida, o mesmo comando foi utilizado para inserir datas nas informações de compra e registrar dados nas fichas, que contém o identificador do cliente (ID) e o nome do cliente. Posteriormente, o comando foi aplicado novamente para incluir dados sobre os itens, sendo que a maioria dessas informações corresponde a chaves estrangeiras oriundas de outras tabelas, como “idcompra”, “idproduto” e “idficha”. Isso significa que é necessário preencher essas tabelas previamente antes de registrar os itens.

Após o preenchimento dessas tabelas, utilizou-se o comando “UPDATE” para modificar alguns valores na tabela de compras. Esse comando exige que o usuário especifique a tabela a ser atualizada, indique a coluna cujo valor será alterado e defina uma condição, por meio da cláusula “WHERE”, para que a alteração ocorra. A cláusula “WHERE” é essencial,

pois permite que o comando atualize apenas as linhas que atendem a uma condição específica, evitando alterações indesejadas em outros registros. Neste caso, a condição utilizada foi o `id` da compra, permitindo modificar a quantidade de itens e corrigir o `id` de alguns clientes que poderiam ter sido inseridos incorretamente.

Em seguida, utilizou-se o comando “SELECT” para exibir todas as colunas preenchidas das tabelas “produto”, “item" e “compra".

Por fim, o comando “INSERT” foi novamente utilizado para adicionar informações dos clientes, como nome, telefone e endereço, além de inserir novos produtos.

Vale ressaltar que esses comandos foram utilizados de forma pragmática na programação orientada a objetos com o auxílio da biblioteca mysql-connector que permitiu a integração direta entre o código Python e o banco de dados MySQL, facilitando a execução de operações de inserções, atualizações e consultas de forma estruturada e segura.

3.4 BUSINESS INTELLIGENCE

Segundo Sharda, Delen e Turban (2019, p. 15), "Inteligência de negócios (BI – business intelligence) é um termo guarda-chuva que combina arquiteturas, ferramentas, bases de dados, ferramentas analíticas, aplicativos e metodologias." O Business Intelligence (BI) utiliza dashboards em seus relatórios, os quais, de acordo com a G4 Educação, são ferramentas com uma interface gráfica que apresentam informações relevantes para o negócio, como métricas e indicadores. Geralmente desenvolvidos pela área de BI, os dashboards têm como principal objetivo facilitar a compreensão dos resultados numéricos da empresa.

No contexto do nosso projeto, criamos um dashboard no Power BI para organizar os dados relevantes ao sistema de banco de dados da GA Pereira, proporcionando uma análise mais eficiente do gerenciamento de compras, pedidos e estoque.

3.4.1 ORGANIZAÇÃO E IDENTIFICAÇÃO DAS INFORMAÇÕES

Para a criação do relatório no Power BI, foi necessário organizar os dados essenciais extraídos diretamente do banco de dados em MySQL. No contexto do nosso projeto, os dados a serem utilizados incluem: clientes, compras, produtos e fichas.

Os clientes representam as pessoas que realizam compras no estabelecimento. As compras correspondem aos produtos selecionados por eles. Os produtos correspondem às mercadorias adquiridas ou aos itens do estoque, enquanto os itens referem-se ao valor unitário de cada produto multiplicado pela quantidade escolhida. Por fim, as fichas registram as compras, que serão pagas posteriormente.

3.4.2 MANIPULAÇÃO E ANÁLISE DE DADOS

É importante ressaltar que os dados mencionados anteriormente também são as tabelas que serão utilizadas para realização do dashboard. Logo, eles apresentam relações entre si.

Os indicadores ‘id’ serão utilizados para criar as relações entre as tabelas. Cada tabela terá pelo menos um indicador ‘id’.

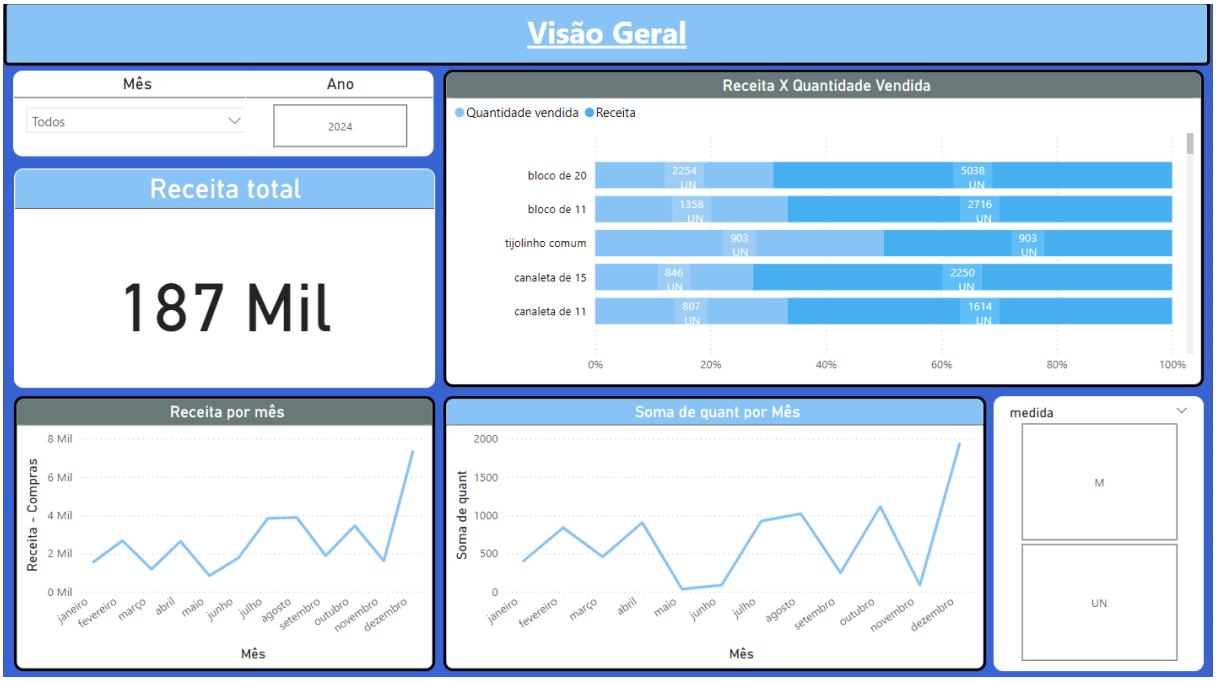
Na tabela Clientes, os atributos identificadores são: “idcliente”, “nome”, “telefone” e “endereço”. Na tabela Compras, os atributos identificadores incluem “idcompra”, “data_da_compra”, “quantidade” e “valor_total_dos_itens”, com a presença de “idcliente” como chave estrangeira, que estabelece a relação entre as tabelas Clientes e Compras. A tabela destinada ao armazenamento de produtos é denominada Estoque e possui os seguintes identificadores: “idproduto”, “nome_do_produto”, “valor”, “quantidade” e “unidade_de_medida”. Por fim, na tabela Ficha, os atributos identificadores são: “idficha”, “valor_total”, “quantidade_de_itens” e “status” (indicando se está pendente ou não), também incluindo “idcliente” como chave estrangeira para vinculação com a tabela Clientes.

3.4.3 CRIAÇÃO DE MODELOS DE ANÁLISE DE DADOS

A partir da criação do modelo físico do banco de dados, detalhado nos tópicos 3.3.2 e 3.3.3, foi gerado as tabelas que foram exportadas para o Power BI permitindo a criação da dashboard com gráficos, indicadores, tabelas, entre outros, que serão utilizados para fazer análises do desempenho da loja.

Em nosso projeto, foram criadas três páginas de relatórios que mostram uma visão geral, análise sobre as compras e análise sobre as fichas.

Figura 6 - Página de Visão Geral(Power BI)



Fonte: Autores

A página de “Visão geral” do nosso dashboard serve para mostrar um panorama da receita total gerada, receita gerada por mês, quantidade de produtos vendidos por mês e relação de custo dos produtos. Além disso, há dois filtros de texto que servem para filtrar essas informações por mês e por ano, permitindo uma análise detalhada e personalizada de acordo com o período desejado.

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: GERENCIANDO FINANÇAS

3.5.1 GERENCIANDO FINANÇAS

- **Tópico 1:** Introdução aos conceitos econômicos e financeiros básicos

Dinheiro é o meio pelo qual damos o valor para a troca de bens e serviços entre pessoas e empresas. Diante disso,a área de finanças tem forte relação com outras áreas de estudo como Contabilidade que estuda as variações de quantidade e qualidade do dinheiro e Economia que analisa a criação, a distribuição e o uso de bens e serviços destinados a suprir as necessidades humanas.

As pessoas comuns não devem gastar mais do que recebem para não enfrentarem problemas financeiros e prejuízos futuros.Sendo assim,gerenciar as despesas pessoais é fundamental para todos.Com esse controle,é possível definir metas de curto e longo prazo, arcar com despesas pessoais (como plano de saúde), despesas de ocupação (como aluguel), despesas com serviços profissionais (como manutenção) e outras despesas (como viagens, entretenimento, combustível etc).O controle financeiro garante uma situação econômica saudável, gerando excedentes financeiros e, a partir disso, torna-se possível decidir como esses recursos excedentes serão utilizados pelas empresas ou pessoas, principalmente com o auxílio do fluxo de caixa.

Exemplo disso é o desejo de adquirir um bem material, como um carro novo, sendo comum a muitas pessoas. No entanto, para tornar esse sonho realidade, é necessário um planejamento financeiro cuidadoso. Ao controlar seus gastos e definir metas claras, é possível direcionar seus recursos para o que realmente importa. Ao invés de gastar impulsivamente, o indivíduo passa a ter consciência de seus hábitos de consumo e a tomar decisões mais racionais. Com disciplina e organização, é possível alcançar a estabilidade financeira e realizar seus objetivos.

- **Tópico 2:** Entendendo o ambiente: independência financeira, o valor da minha riqueza e o registro do dia a dia

A gestão financeira busca liquidez, menores custos e otimização dos resultados, visando aumentar a riqueza. Para indivíduos, liquidez é o que resta após o pagamento de

dívidas. Tanto empresas quanto pessoas precisam gerar excedente financeiro: empresas por meio de vendas e serviços, e indivíduos por renda ou empreendedorismo.

Disciplina é crucial para alcançar a independência financeira, pois não basta apenas gerar receita, é preciso controlar as despesas para evitar que os gastos superem os ganhos. Além disso, acompanhar o fluxo de caixa, organizando entradas e saídas, facilita decisões de investimento.

Por fim, escolher investimentos confiáveis e instituições financeiras sólidas é essencial para proteger seus recursos. Essas regras valem para empresas e indivíduos.

Exemplo disso é um pequeno empresário que possui uma loja de roupas e decide implementar práticas de gestão financeira mais sólidas. Ele começa por analisar suas receitas e despesas, identificando que os custos com aluguel e funcionários estão elevados em relação ao lucro. Para aumentar a liquidez, ele negocia com o proprietário do imóvel um valor de aluguel mais baixo e otimiza a escala de trabalho dos funcionários.

Com o excedente de caixa gerado, ele investe em uma campanha de marketing digital para atrair novos clientes. Para garantir a segurança do dinheiro investido, ele escolhe uma plataforma de pagamento confiável e realiza um estudo sobre as diferentes opções de investimento disponíveis no mercado. Ao longo do tempo, o empresário percebe que a disciplina financeira e a busca por investimentos seguros são fundamentais para o crescimento sustentável do seu negócio. Ele passa a acompanhar de perto o fluxo de caixa, a analisar os indicadores financeiros da empresa e a tomar decisões mais estratégicas.

- **Tópico 3:** Dívidas e juros compostos, opções de empréstimo e alternativas ao endividado

No mundo financeiro, existe uma área de estudo chamada matemática financeira, responsável por estudar o valor do dinheiro no tempo. Um dos seus conceitos mais conhecidos denomina-se juros, e consistem num rendimento que é gerado por uma determinada quantia de dinheiro investida, da qual se espera um retorno igual ou, em alguns casos, superior. Para simplificar, pode-se pensar que, ao emprestar dinheiro para alguém, espera-se que a pessoa pague de volta dentro de um prazo. O atraso consiste em uma perda desse valor que foi investido, portanto é calculada uma taxa que compensa o tempo adicional aguardado por aquele que emprestou. Os juros se subdividem em juros simples e juros compostos. O primeiro é calculado apenas sobre o valor inicial, então o valor adicional pago se mantém constante, enquanto no segundo os juros são aplicados ao montante (valor inicial + juros) do período anterior, então eles crescem exponencialmente. O crédito é cobrado com juros se houver atraso, por isso é importante ter cuidado e pegar apenas o que puder ser pago.

Aqueles que oferecem o crédito sempre procuram informações a respeito do cliente para saber se o indivíduo (ou a empresa) terá condições de pagar de volta.

Deve haver organização nas finanças, tanto pessoais quanto empresariais, através do controle e definição de prioridades. Deve-se definir o que é importante, como contas e objetivos pessoais, e o que pode ser reduzido ou descartado, como gastos com produtos desnecessários. Além disso, sempre deve haver um espaço para a organização do dinheiro na rotina, pois o acompanhamento constante torna as operações financeiras mais conscientes, pelo controle contínuo e planejamento constante. Planejar um orçamento é uma atividade fundamental para uma vida financeira saudável, pois por meio dele e do controle os gastos podem ser feitos com intenção e não por impulso, permitindo a um indivíduo alcançar sonhos, e a uma empresa provocar crescimento.

- **Tópico 4:** Estabelecer metas para a realização de seus sonhos e como envolver o grupo a que você pertence para atingir seus objetivos

Como visto anteriormente, planejar e ter controle constante sobre as finanças são hábitos essenciais. Utilizar ferramentas como o fluxo de caixa para prever entradas e saídas oferece uma visão mais ampla que permite se planejar com mais exatidão para o futuro. Pode-se ter registrado, por exemplo, o dia em que a fatura do cartão fecha ou o dia do recebimento do salário. Além disso, também é importante manter reservas em caso de emergência. Com esses hábitos, realizar sonhos é menos trabalhoso, mas antes de realizá-los eles devem ser muito bem planejados, ou seja, eles devem ser transformados em projetos. Cada sonho deve ser priorizado de acordo com a sua importância e relevância, e então devem ser analisados para considerar todas as variáveis. Se for uma viagem, algumas variáveis seriam valor e local da hospedagem, valor das passagens, custo da alimentação, entre outros. Assim, é possível saber se o sonho pode ser realizado num futuro próximo ou se é necessário adiá-lo um pouco mais. É sempre importante ter atitude para delimitar objetivos claros e se comprometer com eles. Sem atitude, não há competência, e sem competência, não é possível alcançar a estabilidade financeira necessária para realizar sonhos.

Faz-se necessário algum cuidado e ceticismo para não cair em mitos existentes nas finanças: não é preciso muito dinheiro para investir, o tesouro direto e a poupança existem para os investidores menos radicais; o cartão de crédito não é nenhuma entidade sombria, basta ter consciência do valor gasto e estabelecer um limite que seja possível pagar; e gastar tudo o que se ganha em um mês não é aproveitar a vida, é jogar fora oportunidades de crescimento em troca de prazeres de curto prazo. Mitos como esses podem bloquear o crescimento financeiro e impedir um indivíduo de atingir objetivos. Além do que, eles podem

prejudicar a aposentadoria, pois criar investimentos seguros de longo prazo e manter reservas financeiras podem oferecer bons complementos aos fundos de aposentadoria privados e à previdência social.

3.5.2 ESTUDANTES NA PRÁTICA

O vídeo produzido pela nossa equipe traz dicas sobre controle financeiro, destacando a importância de administrar bem o dinheiro para evitar dívidas e perdas. Ele aborda o uso do fluxo de caixa como ferramenta para monitorar entradas e saídas de recursos, além de sugerir a separação entre renda e despesas para facilitar o planejamento. O texto também ressalta a importância de planejar com cuidado, priorizando os custos e avaliando as decisões de compra. Investir em ativos financeiros pode ser uma opção vantajosa, desde que os riscos sejam considerados. Por fim, reforça-se a necessidade de educação financeira para manter a disciplina no controle diário de gastos, evitar endividamentos e alcançar estabilidade, mesmo com recursos limitados.

O vídeo foi postado como 'não listado' no canal do YouTube de Enzo Daniel Abreu, chamado Ikarus com o título de ‘Finanças’. Ele pode ser acessado por meio deste link: https://youtu.be/igAtQniBEUw.

4. CONCLUSÃO

O projeto integrado da nossa equipe procurou auxiliar a empresa GA Pereira Materiais de Construção LTDA por meio de uma aplicação de console, com o objetivo de organizar o registro de compras, estoque, fichas e clientes. Para a criação desse console, foram utilizados conceitos e ferramentas de programação orientada a objetos (POO), lógica de programação, business intelligence (BI) e modelagem de dados. O console foi escrito na linguagem de programação Python.

A programação orientada a objetos (POO) facilita a organização do código por meio da representação de entidades como objetos, que interagem entre si para resolver problemas. Em nosso projeto, a POO foi aplicada na criação do código-fonte para separar as entidades principais do sistema em classes e objetos com suas respectivas funções, sendo que cada uma está conectada à outra. A lógica de programação, por sua vez, organiza a resolução de problemas por meio de sequências de passos, utilizando variáveis, funções, operadores e estruturas condicionais. Em toda a construção do código, a lógica de programação foi empregada de forma consistente para torná-lo limpo e eficiente. Além disso, aplicamos regras de negócios para garantir a consistência dos dados e automatizar processos como descontos e controle de estoque.

O modelo de dados conceitual, representado pelo Diagrama Entidade-Relacionamento (DER), facilita a visualização das entidades e seus relacionamentos no sistema. Desenvolvemos um DER para o projeto, com três entidades principais, seus atributos e cardinalidades, o que permite um mapeamento claro entre clientes, produtos e fichas. Em seguida, o modelo lógico transforma essas entidades em tabelas, enquanto o modelo físico detalha como os dados são armazenados, utilizando comandos SQL como INSERT, UPDATE e SELECT para manipular e consultar as informações, garantindo um banco de dados eficiente e bem estruturado para o sistema.

A Inteligência de Negócios (BI) permite analisar e organizar dados de maneira eficiente, utilizando dashboards para apresentar informações relevantes para o negócio. No nosso projeto, empregamos o Power BI para criar um dashboard que organiza dados sobre clientes, compras, produtos e fichas, exportados diretamente do banco de dados. Esses dados inter-relacionados são apresentados em gráficos e relatórios detalhados, com filtros personalizados, facilitando a análise do desempenho da loja em áreas como receita e vendas de produtos.

Adicionalmente, no conteúdo da formação para a vida, nossa equipe elaborou um vídeo que aborda a importância do controle financeiro, destacando o uso do fluxo de caixa, o planejamento cuidadoso e a educação financeira para evitar dívidas e alcançar estabilidade, além de sugerir a separação entre renda e despesas, a priorização dos custos e o investimento com cautela.

Durante o desenvolvimento do projeto, algumas dificuldades foram encontradas. A principal delas foi a estruturação do código, que, segundo o feedback do professor de lógica de programação, não estava seguindo as normas adequadas. Como resultado, o código precisou ser reformulado várias vezes até atingir o padrão ideal. Da mesma forma, surgiram "bugs" que prejudicaram o funcionamento do programa, que foram corrigidos no decorrer dos testes.

O resultado foi uma aplicação eficiente de registro e consulta de dados relacionados às transações da empresa, possibilitando melhor gerenciamento, armazenamento e análise dos mesmos. Possíveis melhorias futuras poderiam ampliar os dados armazenados para registrar fornecedores e serviços de terceiros, bem como dados relacionados ao transporte das mercadorias para entrega. Além disso, o código também poderia permitir uma comunicação em rede, para que o empresário possa acompanhar de perto a coleta dos dados remotamente.

REFERÊNCIAS

ALMEIDA, M. **Python: utilizando POO na Engenharia de Dados**. Alura 2023. Disponível em: https://www.alura.com.br/artigos/python-poo-engenharia-dados. Acesso em: 2 out. 2024.

ALVES, W. P. **Linguagem e Lógica de Programação**. Rio de Janeiro: Grupo GEN, 2013. E-book. ISBN 9788536519371. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536519371/. Acesso em: 20 set. 2024.

BARBOZA, F. F. M. FREITAS, P. H. C. **Modelagem e desenvolvimento de banco de dados**. Porto Alegre: SAGAH, 2018. E-book. p.109. ISBN 9788595025172. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595025172/. Acesso em: 08 out. 2024.

BESSA, A. **Algorítmos e Lógica de Programação:** O que são e qual a importância? Alura 2023. Disponível em: https://www.alura.com.br/artigos/algoritmos-e-logica-de-programacao. Acesso em: 27 set 2024.

BROOKSHEAR, J. G. **Ciência da computação**. Porto Alegre: Grupo A, 2013. E-book. ISBN 9788582600313. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582600313/. Acesso em: 25 set. 2024.

CLEMENTE, P. **Herança e Polimorfismo em Python:** Um guia detalhado. Rocketseat, 2024. Disponível em: https://blog.rocketseat.com.br/heranca-e-polimorfismo-em-python-um-guia-detalhado/. Acesso em: 3 out. 2024.

DAMAS, L. **SQL - Structured Query Language**. 6th ed. Rio de Janeiro: LTC, 2007. E-book. p.230. ISBN 9788521632450. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788521632450/. Acesso em: 08 nov. 2024.

DAMIÃO, T. **16 Conceitos de POO(Programação Orientada a Objeto)**. Medium 2018. Disponível em: https://medium.com/@TDamiao/16-conceitos-poo-programa%C3%A7%C3%A3o-orientada-a-objeto-6cdc72ac3ee2. Acesso em: 07 nov. 2024

DEVMEDIA. **Interfaces: Programação Orientada a Objetos**. Devmedia 2010. Disponível em: https://www.devmedia.com.br/interfaces-programacao-orientada-a-objetos/18695. Acesso em: 07 nov 2024.

DEVMEDIA. **Métodos, atributos e classes no Java**. Devmedia 2012. Disponível em: https://www.devmedia.com.br/metodos-atributos-e-classes-no-java/25404. Acesso em: 07 nov 2024.

DEVMEDIA. **Modelagem de Dados Conceitual:** construindo pontes entre dados e negócios. Devmedia: 2014. Disponível em: https://www.devmedia.com.br/modelagem-de-dados-conceitual-construindo-pontes-entre-dados-e-negocios/30597. Acesso em: 27 set 2024.

GOMES, G. **CLI: o que é, para que serve e como usar a Interface de Linha de Comandos.** Alura 2023. Disponível em: https://www.alura.com.br/artigos/cli-interface-linha-comandos. Acesso em: 17 set. 2024.

HENRIQUE, J. **POO: o que é programação orientada a objetos ?** Alura 2023. Disponível em:https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos?srsltid=AfmBOorZQ_k90LTR0ensko9ALXRwrvSbcrldT0jmTDBRML3VENBtQwTQ. Acesso em: 08 nov 2024

IBM. **O que são regras de negócios ?**. Disponível em: https://www.ibm.com/br-pt/topics/business-rules. Acesso em: 02 nov. 2024.

OLIVEIRA, J**. Programação Orientada a Objetos:** Herança x Associação**.** DIO 2023. Disponível em: https://www.dio.me/articles/programacao-orientada-a-objetos-heranca-x-associacao. Acesso em 07 nov. 2024

PERKOVIC, L. **Introdução à Computação Usando Python** - Um Foco no Desenvolvimento de Aplicações. Rio de Janeiro: LTC, 2016. E-book. p.230. ISBN 9788521630937. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788521630937/. Acesso em: 08 nov. 2024.

PINHEIRO, F. **Classes Abstratas vs Interfaces**. Treinaweb 2020. Disponível em: https://www.treinaweb.com.br/blog/classes-abstratas-vs-interfaces. Acesso em: 27 set 2024.

SHARDA, R; DELEN, D; TURBAN, E. **Business intelligence e análise de dados para gestão do negócio**. Porto Alegre: Bookman, 2019. E-book. ISBN 9788582605202. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582605202/. Acesso em: 02 out. 2024.

VANZANDT, P. **O que é diagrama UML?** Definição, casos de uso e como fazer. Ideascale: 2022. Disponível em: https://ideascale.com/pt-br/blogue/definicao-de-uml-diagrama/. Acesso em 07 nov. 2024

ANEXOS

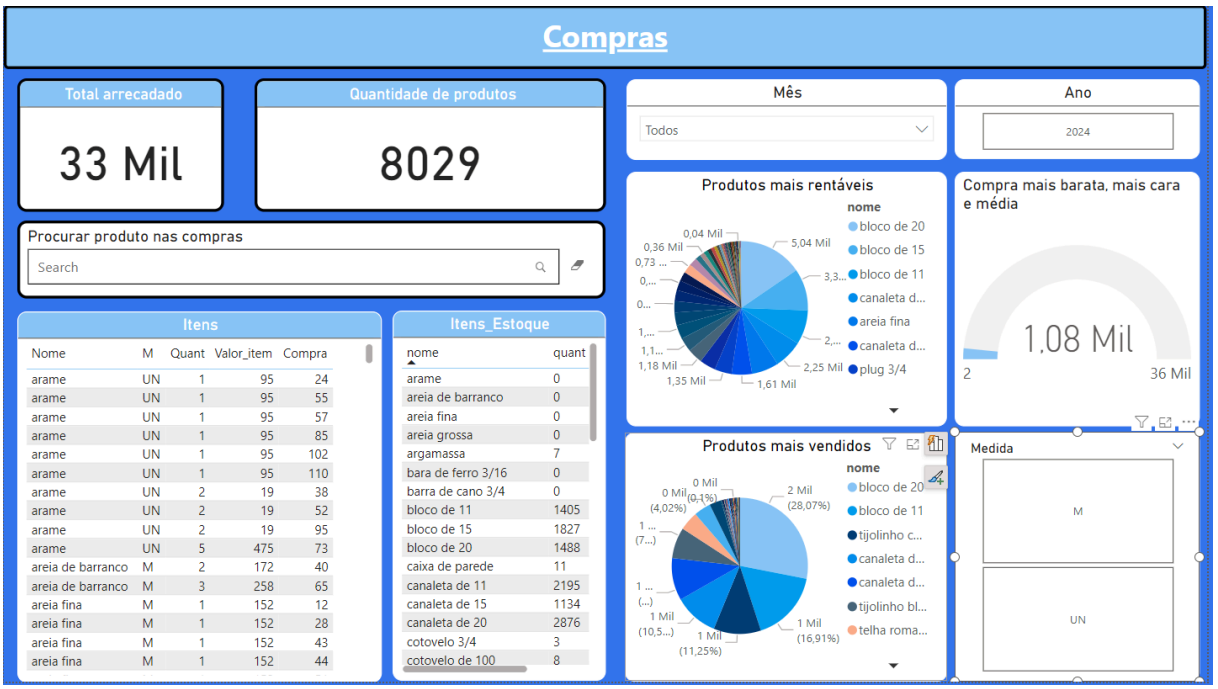
ANEXO A — Classe ConexaoBanco

```python
# bibliotecas padrão
from os import environ as env, path as pt
from abc import ABC

# bibliotecas externas
import mysql.connector
from mysql.connector import Error
from mysql.connector.abstracts import MySQLCursorAbstract, MySQLConnectionAbstract
from mysql.connector.pooling import PooledMySQLConnection
import dotenv as de


# ^ classe abstrata com método utilizado para se conectar ao banco de dados
class ConexaoBanco(ABC):
    cursor: MySQLCursorAbstract
    banco: PooledMySQLConnection | MySQLConnectionAbstract

    @staticmethod
    def conectar_ao_banco():
        if ConexaoBanco.__verificar_credenciais():
            while True:

                try:
                    ConexaoBanco.banco = mysql.connector.connect(
                        host = 'localhost', # servidor
                        user = 'root',
                        database = env['NOME_DO_BANCO'],
                        password = env['SENHA_MYSQL']
                    )

                    ConexaoBanco.cursor = ConexaoBanco.banco.cursor() # cria um cursor que interage com o BD para executar comandos CRUD e outros

                    break
                except Error as error:
                    print('Não foi possível conectar com o banco de dados. Verifique se o nome e a senha estão corretos. ', error)
                    ConexaoBanco.alterar_credenciais()

    @staticmethod
    def __verificar_credenciais() -> (bool | None):
        if 'NOME_DO_BANCO' in env and 'SENHA_MYSQL' in env:
            if len(env['NOME_DO_BANCO']) != 0 and len(env['SENHA_MYSQL']) != 0:
                return True
            else:
                return ConexaoBanco.alterar_credenciais()
        elif de.load_dotenv():
            try:
                ConexaoBanco.__verificar_credenciais()
            except RecursionError as error:
                print('Níveis de recursão perigosos. ', error)
        else:
            ConexaoBanco.alterar_credenciais()

    @staticmethod
    def alterar_credenciais() -> bool:
        try:
            # path = dotenv_path = pt.abspath(pt.join(pt.dirname(__file__), '.', '.env'))
            de.set_key(dotenv_path = de.find_dotenv(), key_to_set='NOME_DO_BANCO', value_to_set=input('Insira o nome do banco: ').strip())
            de.set_key(dotenv_path = de.find_dotenv(), key_to_set='SENHA_MYSQL', value_to_set=input('Insira a senha do MySQL: ').strip())
            de.load_dotenv(override=True)
            return True
        except Exception as error:
            print('Erro ao criar as senhas. ', error)
            return False

    @staticmethod
    def limpar_credenciais() -> None:
        try:
            # path = dotenv_path = pt.abspath(pt.join(pt.dirname(__file__), '.', '.env'))
            de.unset_key(dotenv_path = de.find_dotenv() , key_to_unset='NOME_DO_BANCO')
            de.unset_key(dotenv_path = de.find_dotenv(), key_to_unset='SENHA_MYSQL')
        except Exception as error:
            print(error)

    @classmethod
    def fechar_conexao(cls) -> None:
        cls.cursor.close()
        cls.banco.close()

        while True:
            match input('Deseja limpar suas credenciais (nome do banco e senha)? (S ou N) ').strip().lower():
                case 's':
                    cls.limpar_credenciais()
                    print('Credenciais limpas!')
                    input('Pressione ENTER para encerrar o programa.')
                    break
                case 'n':
                    break
                case _:
                    print('Digite sim ou não, por favor')
```

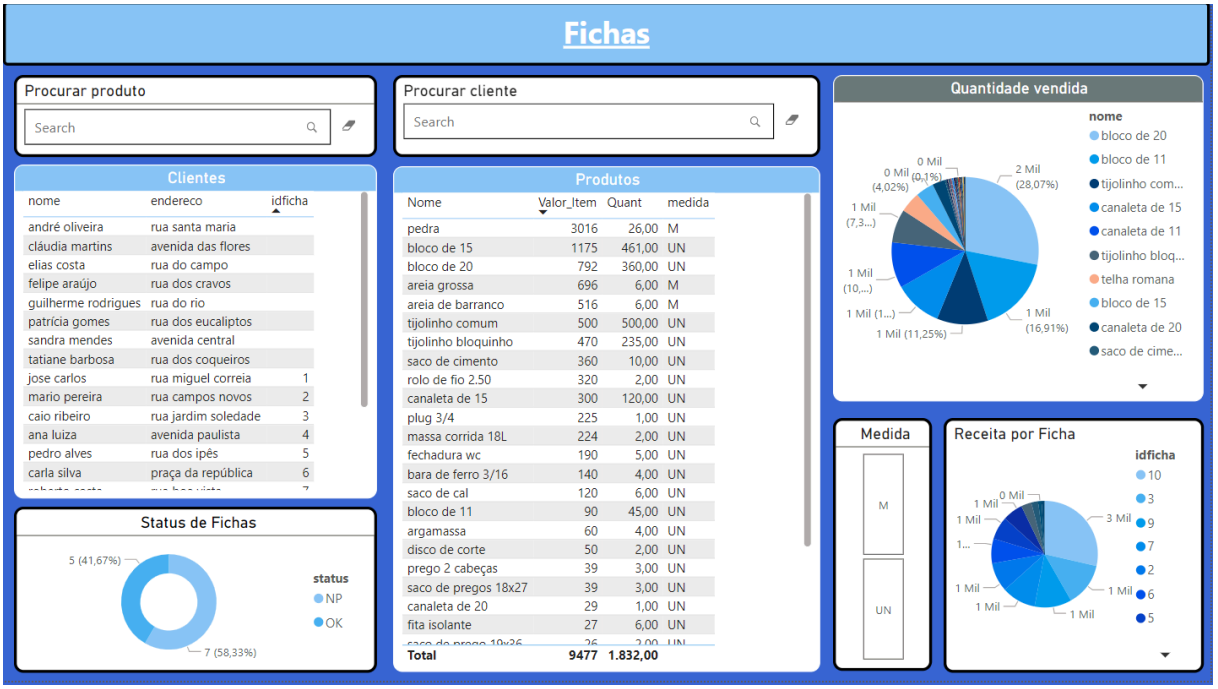
ANEXO B — Arquivo main.py

```python
from os import system
import classes as cl

if __name__ == '__main__':

    estoque = cl.Estoque()
    compra = cl.Compra()
    ficha = cl.Ficha()
    cliente = cl.Cliente()

    print('Banco de dados conectado com sucesso!')

    while True:
        input('\nPressione Enter para continuar') # permite que o usuário leia as informações antes de cada reinício
        system('cls') # limpa a tela

        match input('\n\rEscolha uma opção: \n'
                    '1 - Abrir menu das compras \t2 - Abrir menu do estoque \n'
                    '3 - Abrir menu das fichas \t4 - Abrir menu dos clientes \n'
                    '0 - Fechar programa\n\r>>> ').strip():
            case '1':
                cl.abrir_menu(compra) if estoque.verificar_estoque() == True else print('Estoque vazio, registre um produto antes.')

            case '2':
                cl.abrir_menu(estoque)

            case '3':
                cl.abrir_menu(ficha) if cliente.verificar_clientes() == True else print('Registre um cliente antes de preencher uma ficha.')

            case '4':
                cl.abrir_menu(cliente)

            case '0':
                if input('Tem certeza? \nDigite 0 para confirmar ou pressione ENTER para cancelar: ').strip() == '0':
                    system('cls')
                    compra.fechar_conexao()

                    break
                else:
                    print('Comando cancelado, continuando a execução.\n')
            case _:
                system('cls')
                print("Digite um dos números, por favor.")
```

ANEXO C — Página de Compras(Power BI)



ANEXO D — Página de Ficha(Power BI)



ANEXO E – Tabela de Clientes (MySQL Workbench)

idcliente	nome	telefone	endereco
1	jose carlos	1998278634021	rua miguel correia
2	mario pereira	1933367844423	rua campos novos
3	caio ribeiro	195454778991	rua jardim soledade
4	ana luiza	198112233445	avenida paulista
5	pedro alves	192334556677	rua dos ipês
6	carla silva	199876543210	praça da republica
7	roberto costa	198765432109	rua boa vista
8	larissa fernandes	193398765432	rua da alegria
9	paulo henrique	199823456789	avenida brasil
10	mariana santos	197612345678	rua das flores
11	lucas lima	199734567890	rua do sol
12	viviane monteiro	198123498765	rua verde
13	andré oliveira	197234567890	rua santa maria
14	sandra mendes	195678901234	avenida central
15	guilherme rodrig...	199852347810	rua do rio
16	tatiane barbosa	193456789012	rua dos coqueiros
17	felipe araujo	198765432987	rua dos cravos
18	claúdia martins	197823456123	avenida das flores
19	elias costa	199876544321	rua do campo
20	patrícia gomes	198543210987	rua dos eucaliptos
NULL	NULL	NULL	NULL

ANEXO F – Script SQL do Banco de Dados

```sql
-- INSERINDO DADOS EM PRODUTOS
insert into produto (nome,valor,quant) values ("saco de cal",60,30),
("areia grossa",86,30),
("saco de cimento",36,30),
("pedra",116,30),
("bloco de 15",2.10,300),
("canaleta de 15",2.20,300),
("areia fina",92,30),
("arame 1K",10.50,15),
("argamassa",20,30),
("massa corrida18L",112,10),
("lixa para parede",2,20),
("galão vedapen 18L",216,5);

-- INSERINDO DADOS EM COMPRA
insert into compra(data) values
('2024-08-12'), ('2024-09-23');

-- INSERINDO DADOS EM FICHA
insert into ficha(clientes_idclientes, nome_cliente) values (3, "caio ribeiro");

-- INSERINDO DADOS EM ITEM
insert into item (compra_idCompra, produto_idprodutos, nome_produto, quant, valor_item) values
(1, 3, "saco de cimento", 3, 108), (1, 5, "bloco de 15", 5, 10.5), (1, 7, "areia fina", 1, 92),
(2, 1,"saco de cal",3,180), (2, 4,"pedra",2,232), (2, 8,"arame 1K",4,42);
insert into item (ficha_idficha, produto_idprodutos, nome_produto, quant, valor_item) values
(1, 12, "galão vedapen 18L", 2, 432), (1, 11, "lixa para parede", 3, 6), (1, 10,"massa corrida 18L",1,112);

-- ATUALIZANDO DADOS DA TABELA COMPRA
update compra set quant_itens_un_kg = 9 where idCompra = 1;
update compra set valor_total = 210.10 where idCompra = 1;
update compra set quant_itens_un_kg = 9 where idCompra = 2;
update compra set valor_total = 454 where idCompra = 2;

 update compra set quant_itens_un_kg = 6 where idCompra = 3;
 update compra set valor_total = 550 where idCompra = 3;

 update compra set clientes_idclientes = 1 where idCompra = 1;
 update compra set clientes_idclientes = 1 where idCompra = 1;
 update compra set clientes_idclientes = 2 where idCompra = 2;
 update compra set clientes_idclientes = 2 where idCompra = 2;
 update compra set clientes_idclientes = 3 where idCompra = 3;
 update compra set clientes_idclientes = 3 where idCompra = 3;

 -- SELECIONANDO TODAS AS LINHAS DA TABELA PRODUTO,ITEM E COMPRA
 select *from produto;
 select *from item;
 select *from compra;

 -- INSERINDO DADOS EM CLIENTES
 insert into clientes (nome,telefone,endereco) values ("jose carlos",1998278634021,"rua miguel correia"),
 ("mario pereira",1933367844423,"rua campos novos"),
 ("caio ribeiro",195454778991,"rua jardim soledade");

 -- INSERINDO MAIS DADOS EM PRODUTO
 insert into produto (nome,valor,quant,medida) values
 ("areia fina",152,10,"M"),
 ("areia grossa",116,7,"M"),
 ("serrinha",5,5,"UN"),
 ("arame",9.50,10,"UN"),
 ("saco de pregos 18x27",13,5,"UN"),
 ("saco de cimento",36,30,"UN"),
 ("telha romana",2,1000,"UN"),
 ("saco de prego 15x15",22,8,"UN"),
 ("saco de prego 19x36",13,9,"UN"),
 ("saco de prego 20x48",13,10,"UN"),
 ("tijolinho comum",1,1200,"UN"),
```