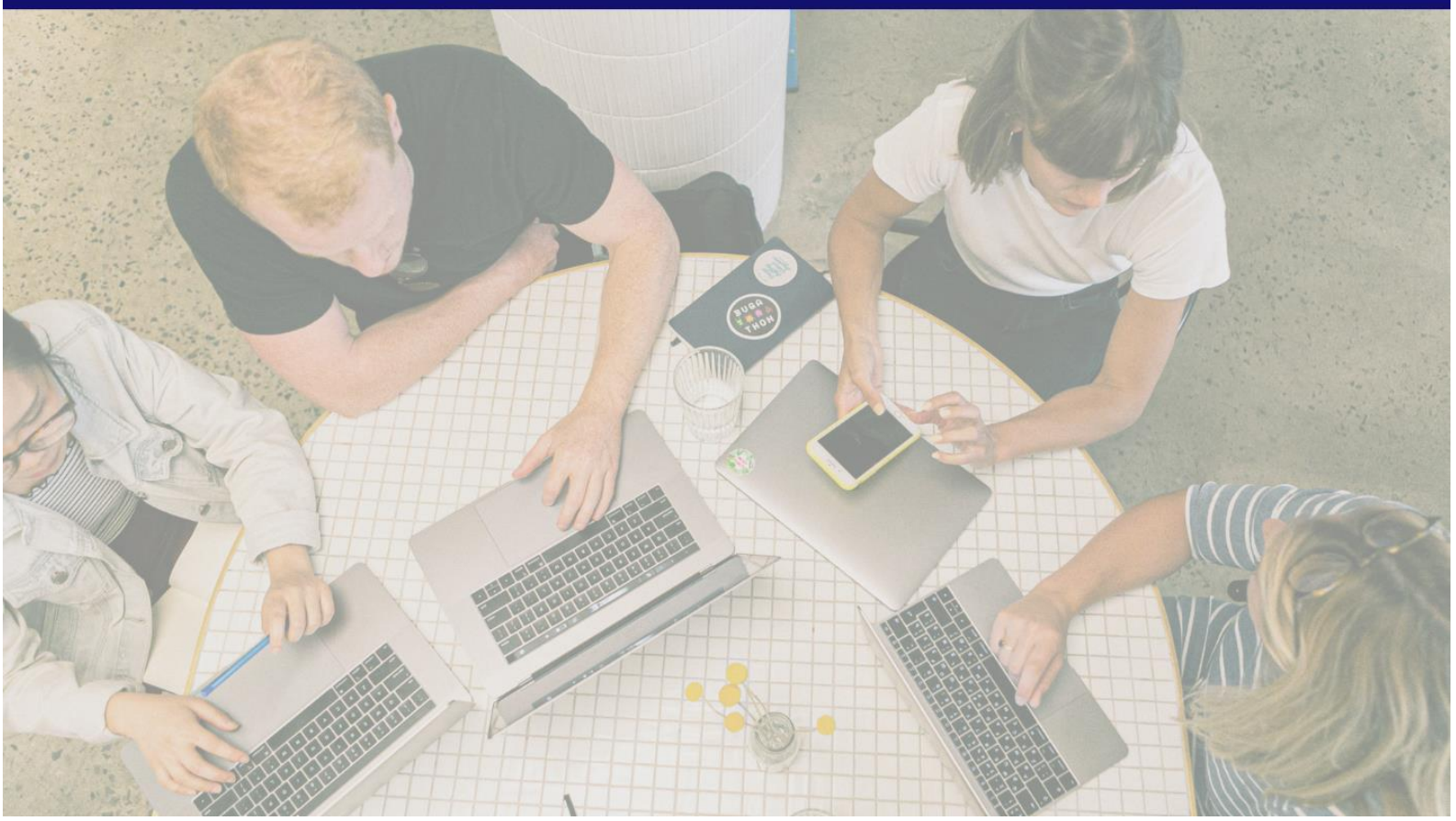
UNIFEOB
ESCOLA DE NEGÓCIOS

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

BOA VISTA FIT LTDA

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

BOA VISTA FIT LTDA

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:

Luiz Felipe, RA 24001022

Davi Pádua Medeiros de Carvalho, RA 24001211

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2024

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento de soluções tecnológicas tem se mostrado uma ferramenta essencial para enfrentar os desafios contemporâneos nas áreas de educação, sustentabilidade, inclusão social, e empreendedorismo. Neste cenário, a integração de conceitos de Business Intelligence, Programação Orientada a Objetos (POO), Lógica de Programação e Modelagem de Dados pode contribuir significativamente para a criação de soluções inovadoras que atendem a demandas sociais e ambientais. O projeto intitulado "Desenvolvimento de Soluções Console Integradas para Educação, Sustentabilidade, Inclusão Social e Empreendedorismo" busca explorar essas tecnologias para desenvolver uma aplicação via console que integre esses conceitos, visando a promoção de ações voltadas ao desenvolvimento sustentável e inclusivo.

O objetivo deste trabalho é criar uma aplicação prática que combine tecnologias de ponta e práticas inovadoras para resolver problemas reais, contribuindo para áreas essenciais como a educação e capacitação, sustentabilidade ambiental, inclusão social, tecnologia e inovação, desenvolvimento rural e empreendedorismo solidário. A proposta é desenvolver uma solução de fácil acesso e baixo custo, utilizando uma interface de console que permita o gerenciamento e análise de dados em diferentes áreas, facilitando a implementação de projetos educativos, ambientais e sociais. Além disso, o projeto visa proporcionar aos estudantes uma experiência prática e interdisciplinar, capacitando-os a utilizar a tecnologia de forma criativa e eficiente.

A metodologia adotada para o desenvolvimento da solução será dividida em três etapas principais: o levantamento e análise de requisitos, o desenvolvimento da aplicação utilizando os conceitos de POO, e a implementação de funcionalidades que promovam a análise e a visualização de dados com base em soluções de Business Intelligence. Na primeira etapa, serão definidos os dados necessários para cada área do projeto, como educação, meio ambiente, e empreendedorismo. Na segunda etapa, o sistema será projetado utilizando a POO, com a criação de classes e métodos específicos para lidar com as diferentes funcionalidades, como o cadastro de participantes, gestão de projetos e monitoramento de indicadores de impacto. Na terceira etapa, serão aplicadas técnicas de análise de dados para gerar relatórios e insights relevantes, que serão utilizados pelos gestores dos projetos para a tomada de decisões mais embasadas.

A justificativa deste trabalho se dá pela crescente necessidade de desenvolver soluções tecnológicas que possam contribuir de forma concreta para o enfrentamento de desafios sociais, ambientais e econômicos. As tecnologias desenvolvidas neste projeto serão voltadas para a promoção da inclusão social, da **s**ustentabilidade ambiental e do empreendedorismo em comunidades carentes e áreas rurais. Além disso, ao integrar soluções de Business Intelligence, a aplicação permitirá a análise eficiente de dados, potencializando os resultados de programas voltados à capacitação profissional, preservação ambiental e incentivo ao empreendedorismo. Com isso, o projeto busca não apenas fornecer uma solução técnica, mas também contribuir para um desenvolvimento mais justo, sustentável e inclusivo, alinhado com as demandas globais e os Objetivos de Desenvolvimento Sustentável (ODS) estabelecidos pela ONU.

ISSN 1983-6767

2. DESCRIÇÃO DA EMPRESA

Empresa (Razão Social): BOA VISTA FIT LTDA
Nome Fantasia: Track & Field
CNPJ: 38.477.845/0001-45
Endereço: Av. Dona Gertrudes, 51 - Centro, São João da Boa Vista - SP, 13870-110
Atividade: Comércio Varejista

A empresa BOA VISTA FIT LTDA, de nome fantasia Track&Field atua no comércio varejista de São João da Boa Vista, com venda de produtos diretamente ao consumidor final. A loja tem como principais produtos artigos esportivos masculinos e femininos.

ISSN 1983-6767

3. PROJETO INTEGRADO

3.1 PROGRAMAÇÃO ORIENTADA A OBJETO

Desenvolvimento de Soluções de Gestão Empresarial com Programação Orientada a Objetos

(Conjunto de Imagens 1)

O projeto em questão utiliza Programação Orientada a Objetos (POO) para criar uma aplicação de gestão empresarial focada em produtos, vendas, usuários e fornecedores. A proposta visa integrar soluções tecnológicas eficientes que ajudem na administração e controle de dados de vendas, estoque, clientes e funcionários, utilizando conceitos fundamentais de POO para resolver problemas práticos enfrentados por empresas de diversos segmentos.

Ao aplicar POO, o sistema é estruturado de forma modular e reutilizável, permitindo que as funcionalidades da aplicação sejam desenvolvidas de maneira escalável e de fácil manutenção. Os estudantes têm a oportunidade de aplicar conceitos avançados de programação, como classes, objetos, encapsulamento e herança, para criar soluções robustas que atendem às necessidades do mercado de trabalho, que demanda sistemas organizados e flexíveis.

A utilização de herança é um ponto chave na estrutura do sistema. A classe Usuario, por exemplo, pode servir como uma classe base para diferentes tipos de usuários, como Gerente e Vendedor, que herdam os atributos e comportamentos comuns da classe Usuario, mas com funcionalidades específicas para cada perfil. Isso torna o sistema altamente flexível e fácil de modificar à medida que novos requisitos surgem, como a inclusão de novos tipos de usuários ou mudanças nas regras de negócios de vendas.

A principal vantagem dessa abordagem é a criação de sistemas reutilizáveis e expansíveis. Com o uso de classes bem definidas, é possível adicionar novas funcionalidades ou modificar processos com o mínimo de impacto nas outras partes do sistema. Por exemplo, ao implementar a Venda, a adição de novos métodos ou alterações nas regras de cálculo de valores não afeta o restante do sistema, como o cadastro de produtos ou o gerenciamento de estoque.

Além disso, a POO facilita a integração com bancos de dados, como o MySQL, permitindo o armazenamento e a recuperação eficiente de informações relacionadas a produtos, vendas, fornecedores e usuários. A aplicação de conceitos de Modelagem de Dados ajuda a organizar as entidades do sistema de maneira que possam ser facilmente consultadas e manipuladas para gerar relatórios, como a listagem de produtos em estoque, histórico de vendas ou dados sobre os fornecedores.

A POO também permite a implementação de melhores práticas de desenvolvimento de software, como o encapsulamento e o polimorfismo. O encapsulamento garante que os dados sejam acessados e manipulados de maneira controlada, protegendo as informações sensíveis e garantindo que a lógica de negócio esteja sempre em conformidade. O polimorfismo, por sua vez, possibilita a criação de métodos que podem ser chamados de diferentes formas, dependendo da classe que os invoca, proporcionando maior flexibilidade no sistema.

Através desse projeto, os estudantes desenvolvem não apenas habilidades técnicas, mas também competências para resolver problemas reais em contextos empresariais, como a gestão de vendas e o controle de estoque. O uso de POO em um sistema de gestão empresarial prepara os alunos para enfrentar os desafios do mercado de trabalho, proporcionando uma compreensão profunda sobre como estruturar sistemas complexos e como utilizá-los para tomar decisões estratégicas.

O projeto também oferece uma oportunidade de aplicação de conceitos de inclusão social e empreendedorismo solidário, permitindo, por exemplo, a criação de módulos específicos para o cadastro de clientes em situações vulneráveis, ou a promoção de práticas sustentáveis, como a análise de dados de consumo e a adoção de políticas de redução de desperdício de estoque.

Ao aplicar a POO para resolver problemas do dia a dia de uma empresa, o projeto demonstra o poder dessa metodologia para criar soluções que são não apenas técnicas, mas também sociais e éticas. Os alunos, ao desenvolverem uma aplicação integrada para a gestão de produtos, vendas e usuários, aprendem a construir sistemas bem estruturados que podem contribuir diretamente para a transformação do mercado empresarial e para a criação de um futuro mais sustentável e inclusivo.

3.1.1 CLASSES E OBJETOS

Desenvolvimento de Soluções Console Integradas para Gestão Empresarial com Programação Orientada a Objetos

(Conjunto de Imagens 1)

O projeto "Desenvolvimento de Soluções Console Integradas para Gestão Empresarial" oferece uma oportunidade valiosa para que os estudantes apliquem os conhecimentos adquiridos sobre Programação Orientada a Objetos (POO) no desenvolvimento de uma aplicação robusta e eficiente para a gestão de produtos, vendas e usuários. Ao criar classes que representam entidades do mundo real, como Produto, Usuario, Venda e Fornecedor, os estudantes aprendem a estruturar dados e comportamentos de maneira organizada, promovendo a modularidade e a reutilização do código.

No contexto deste projeto, um dos primeiros passos será a criação de classes que representem as principais entidades envolvidas nas operações empresariais. Por exemplo, a classe Produto incluirá atributos como nome, descrição, preço, quantidade e categoria, enquanto a classe Venda pode armazenar dados como o valor da venda, cliente envolvido e data. Métodos como cadastrarProduto, registrarVenda e atualizarEstoque serão fundamentais para que o sistema não só organize as informações, mas também execute operações essenciais para o gerenciamento de um negócio, como registrar vendas, controlar o estoque e atualizar os dados dos produtos.

A prática de instanciar objetos a partir dessas classes será outro aspecto importante. Por exemplo, o estudante criará objetos como produto1 ou venda1, preenchendo-os com dados específicos, como "Produto: Monitor 24”, com preço de R$ 1500,00, ou "Venda de Cliente X", com um valor total de R$ 5000,00. Cada objeto, representando um produto ou uma venda, terá comportamentos distintos dependendo dos dados atribuídos a eles e dos métodos que serão invocados, como ajustar o estoque ou calcular o valor total de uma venda.

Além disso, a POO permite que esses objetos se comportem de forma modular e independente, tornando o código mais fácil de manter e expandir. Quando os estudantes criam classes e objetos, eles estão praticando a separação de preocupações e a organização do código em unidades lógicas que podem ser desenvolvidas, testadas e modificadas de maneira isolada. Isso é essencial quando se trabalha com sistemas complexos, como o gerenciamento de dados

de vendas, estoque e clientes, que podem precisar de ajustes e melhorias conforme o sistema cresce.

No projeto, o estudante também pode criar outras classes, como Usuario, que representa os funcionários da empresa, seja Gerente ou Vendedor, e a classe Fornecedor, para modelar as relações com os fornecedores de produtos. O uso de herança e polimorfismo permitirá que o código seja ainda mais eficiente. Uma classe base Usuario pode ser criada com atributos e métodos comuns, como login e senha, e as subclasses Gerente e Vendedor podem herdar esses métodos, mas com comportamentos específicos, como cálculo de salários para gerentes ou comissões para vendedores. Essa flexibilidade garante que, à medida que novos tipos de usuários ou funcionalidades sejam necessários, o código não precise ser refeito, mas sim estendido.

Além disso, a modularidade do código permitirá que o sistema se integre facilmente com outras áreas, como Business Intelligence e Modelagem de Dados. Por exemplo, a classe Produto pode conter informações sobre o preço, categoria e estoque de cada item, e essas informações podem ser coletadas e analisadas para gerar relatórios de vendas ou de estoque. Isso facilita a integração com ferramentas como o Power BI, permitindo a criação de dashboards interativos que ajudam na tomada de decisões estratégicas para o negócio.

A utilização de POO também ensina os estudantes a aplicar boas práticas de desenvolvimento, como encapsulamento e polimorfismo. O encapsulamento permite que os dados de cada entidade, como produtos ou vendas, sejam manipulados de forma controlada, garantindo que as informações sensíveis, como preços ou quantidade em estoque, sejam acessadas de maneira segura. O polimorfismo, por sua vez, facilita a criação de métodos que podem ser reutilizados em diferentes contextos, permitindo maior flexibilidade e reutilização do código.

Esse desenvolvimento não só ensina aos estudantes a construir soluções tecnológicas eficientes, mas também os prepara para enfrentar os desafios do mercado de trabalho. A capacidade de estruturar um sistema usando POO é uma habilidade essencial em qualquer carreira de programação, permitindo que os profissionais criem código mais limpo, modular e sustentável. Além disso, ao aplicar esses conceitos em um contexto de gestão empresarial, os alunos aprendem como criar soluções que podem melhorar processos de negócios, como o controle de estoque, a gestão de vendas e o relacionamento com clientes e fornecedores, o que os prepara para contribuir com soluções inovadoras no mundo corporativo.

3.1.2 ATRIBUTOS, MÉTODOS, ENCAPSULAMENTO, HERANÇA E POLIMORFISMO.

Desenvolvimento de Soluções Console Integradas para Gestão Empresarial com Programação Orientada a Objetos

(Conjunto de Imagens 1)

O projeto "Desenvolvimento de Soluções Console Integradas para Gestão Empresarial" oferece uma oportunidade prática para os estudantes aplicarem os conceitos fundamentais da Programação Orientada a Objetos (POO) na criação de uma aplicação robusta para a gestão de produtos, vendas, usuários e fornecedores. Ao longo do desenvolvimento, os alunos terão a chance de trabalhar com atributos, métodos, encapsulamento, herança e polimorfismo, conceitos essenciais para a construção de sistemas modulares, escaláveis e eficientes.

Atributos e Métodos

No contexto deste projeto, as classes são responsáveis por representar as principais entidades do sistema, como Produto, Usuario, Venda, Fornecedor, etc. Cada classe tem atributos e métodos que definem suas características e comportamentos.

Por exemplo, a classe Produto pode ter atributos como nome, descricao, preco, quantidade, e categoria. Esses atributos são fundamentais para descrever cada produto no sistema.

Encapsulamento

O encapsulamento é um princípio importante da POO que visa ocultar os detalhes internos de uma classe e expor apenas os métodos necessários para a interação com o sistema. Isso ajuda a proteger os dados e garantir que a manipulação dos mesmos seja feita de forma controlada.

No projeto, a classe Produto encapsula atributos como nome, preco e quantidade, tornando-os privados. Métodos públicos são fornecidos para acessar ou modificar esses dados de forma segura. Com esse encapsulamento, é possível controlar como os dados são acessados e alterados, garantindo que o sistema funcione de forma estável e segura.

Herança

A herança é uma das características mais poderosas da POO, permitindo criar novas classes (subclasses) baseadas em classes existentes (superclasses). Isso promove a reutilização de código e facilita a manutenção e expansão do sistema.

No contexto do projeto, por exemplo, a classe Usuario pode ser uma classe base que contém atributos e métodos comuns a todos os usuários do sistema (como nome, usuario, senha). Em seguida, as classes Gerente e Vendedor podem herdar de Usuario e adicionar comportamentos específicos, como calcular salários ou comissões.

Polimorfismo

O polimorfismo é um conceito que permite que métodos com o mesmo nome se comportem de maneiras diferentes dependendo da classe que os invoca. Esse conceito é essencial quando você tem subclasses que implementam um método de forma específica para seu contexto.

3.1.3 MÉTODOS ESTÁTICOS, PÚBLICOS E PRIVADOS

Métodos Públicos: Interagindo com Objetos e Realizando Operações

(Conjunto de Imagens 1)

Os métodos públicos são fundamentais para a interação com as instâncias das classes. Eles representam as operações que podem ser realizadas sobre ou com os objetos criados a partir dessas classes. No código enviado, temos diversos exemplos de métodos públicos que permitem manipular os dados relacionados a vendas, produtos e usuários. Aqui, temos métodos como getNome(), getPreco() e setQuantidade() que permitem acessar e modificar dados específicos de um produto. O método setQuantidade() permite ajustar a quantidade disponível de um produto no estoque, essencial para o controle de inventário.

Esses métodos garantem que o estoque e os preços dos produtos sejam sempre atualizados de maneira controlada e acessível, facilitando a gestão dentro do sistema.

Exemplo de método público em Venda:

A classe Venda gerencia as transações realizadas entre a empresa e o cliente. Ela possui métodos públicos que permitem acessar o valor total da venda e o cliente associado à transação. Aqui, os métodos getValor() e getCliente() são públicos e permitem que o sistema acesse os dados da venda, como o valor total e o cliente associado. Esse tipo de interação é essencial para o acompanhamento das vendas e a emissão de relatórios financeiros. Esse processo de consulta é vital para o gerenciamento financeiro e o rastreamento das vendas feitas ao longo do tempo.

Métodos Privados: Garantindo a Integridade dos Dados

Os métodos privados desempenham um papel importante ao esconder detalhes internos da implementação de uma classe e protegendo a integridade dos dados. Esses métodos podem ser usados para garantir que certos processos sejam realizados de forma controlada, sem que o código externo tenha acesso direto a essas funcionalidades internas.

Exemplo de método privado em Produto:

Na classe Produto, podemos imaginar um método privado que verifica se o preço de um produto é válido antes de ser alterado. Esse tipo de lógica interna ajuda a manter os dados consistentes e evitar modificações indesejadas no sistema. Neste exemplo, o método __validarPreco() é privado e verifica se o novo preço é maior que zero antes de atualizá-lo. Se o preço for inválido, o método setPreco() impede a atualização, garantindo que o sistema não aceite dados errados ou inconsistentes.

Esse tipo de proteção interna mantém a consistência dos dados e evita que ações incorretas sejam realizadas, como a atribuição de preços negativos.

Métodos Estáticos: Operações Globais e Utilitárias

Os métodos estáticos são úteis para realizar operações que não dependem de instâncias específicas de uma classe. Eles podem ser usados para realizar cálculos ou executar funções que não exigem acesso a dados de instância. O método calcularTotal() é estático porque realiza uma operação que não depende de uma instância específica, apenas dos parâmetros fornecidos. Ele pode ser chamado diretamente pela classe Item_venda sem necessidade de instanciar um objeto.

3.2 LÓGICA DE PROGRAMAÇÃO

(Conjunto de Imagens 1)

Algoritmos e Variáveis: Armazenando e Manipulando Dados

A construção de algoritmos começa com a definição de sequências de passos lógicos para resolver problemas específicos. No contexto do projeto de vendas, esses algoritmos podem ser usados para calcular o total de itens vendidos, verificar o estoque disponível ou até mesmo gerar relatórios de vendas.

As variáveis são usadas para armazenar dados temporários durante a execução dos algoritmos. Elas podem representar produtos, quantidades, valores de vendas ou informações do cliente. No código, variáveis como nome, preco, quantidade e total são essenciais para organizar e manipular as informações relacionadas aos produtos e vendas.

Funções: Modularizando o Código

As funções são blocos de código reutilizáveis, essenciais para a modularização do sistema. Elas permitem que tarefas repetitivas sejam organizadas em unidades de código que podem ser chamadas quando necessário, tornando o sistema mais eficiente e fácil de manter.

No projeto de vendas, funções podem ser usadas para calcular totais de vendas, verificar se o estoque é suficiente para uma venda ou até mesmo realizar a matrícula de produtos no estoque.

Estruturas Condicionais: Tomando Decisões no Fluxo de Trabalho

As estruturas condicionais em Python, como if, elif e else, são utilizadas para tomar decisões com base em condições específicas. Elas são essenciais para verificar condições antes de realizar ações no sistema.

No contexto do projeto de vendas, essas estruturas podem ser aplicadas para verificar a disponibilidade de estoque antes de processar uma venda ou verificar se um cliente está cadastrado no sistema. Elas ajudam a garantir que o sistema apenas execute operações válidas e consistentes.

Operadores Lógicos e de Comparação: Comparando Dados e Tomando Decisões

Operadores lógicos e operadores de comparação são fundamentais para realizar verificações complexas e tomar decisões com base em múltiplas condições. Os operadores de comparação como ==, !=, <, >, <=, >= são usados para comparar valores, enquanto os operadores lógicos como and, or e not ajudam a combinar múltiplas condições.

No contexto de vendas, operadores de comparação podem ser usados para verificar se um cliente já está registrado, comparar o preço de produtos ou avaliar o estoque disponível. Já os operadores lógicos podem ser usados para combinar várias condições, como verificar se a quantidade solicitada de um produto está disponível e se o cliente tem um limite de crédito suficiente.

3.2.1 CONCEITOS FUNDAMENTAIS DO DESENVOLVIMENTO DE SOFTWARE

Algoritmos: A Sequência Lógica para Resolver Problemas de Vendas

(Conjunto de Imagens 1)

Algoritmos são sequências de instruções que descrevem como resolver um problema de forma eficiente. No contexto do projeto de vendas, um algoritmo pode ser utilizado para calcular o total de uma venda, verificar a disponibilidade de um produto no estoque ou processar o pagamento.

Variáveis e Tipos de Dados: Organizando e Armazenando Informações

As variáveis são utilizadas para armazenar dados temporários que o programa manipula durante a execução. Em um sistema de vendas, as variáveis podem armazenar informações como o nome do cliente, o preço dos produtos, a quantidade de itens no estoque, entre outros.

Já os tipos de dados determinam o formato desses dados. Os tipos mais comuns incluem inteiros (para quantidades e valores inteiros), flutuantes (para valores monetários), strings (para nomes e descrições), e booleanos (para condições, como verificar se o pagamento foi aprovado). Essas variáveis são essenciais para controlar o cadastro de clientes, a quantidade de produtos disponíveis para venda, e o preço de cada item, permitindo que o sistema faça os cálculos e verifique a disponibilidade de produtos.

Funções: Modularizando Operações no Sistema de Vendas

As funções permitem dividir o código em blocos reutilizáveis, facilitando a manutenção e a organização do sistema. No projeto de vendas, funções podem ser utilizadas para realizar operações como cadastrar um cliente, verificar a disponibilidade de estoque, calcular o valor total de uma venda e validar pagamentos.

Estruturas Condicionais: Tomando Decisões no Fluxo de Vendas

As estruturas condicionais são fundamentais para a lógica de qualquer sistema, permitindo que o programa tome decisões com base em condições específicas. Em um sistema de vendas, essas estruturas são usadas para verificar se um produto está disponível no estoque, se o pagamento foi realizado com sucesso ou se o cliente já está cadastrado no sistema.

Operadores: Realizando Operações e Comparações

Os operadores são usados para realizar operações sobre variáveis e valores. Em Python, temos diferentes tipos de operadores, como:

- Operadores aritméticos (como +, -, *, /) para realizar operações matemáticas.
- Operadores de comparação (como ==, !=, >, <) para comparar valores.
- Operadores lógicos (como and, or, not) para combinar condições lógicas.

Em um sistema de vendas, os operadores de comparação são essenciais para verificar se um produto tem estoque suficiente ou se um pagamento foi autorizado. Já os operadores lógicos são úteis para combinar várias condições, como verificar se o estoque é suficiente E se o pagamento foi concluído.

3.2.2 DESENVOLVIMENTO DE APLICAÇÕES

No projeto de vendas, a implementação das regras de negócios e a modularização do código são essenciais para garantir que o sistema funcione corretamente, seja escalável e fácil de manter. As regras de negócios são as diretrizes que determinam como o sistema deve se comportar de acordo com os requisitos do projeto, e a modularização permite que cada parte do código seja organizada, reutilizável e independente, facilitando manutenções e atualizações futuras.

Regras de Negócio no Sistema de Vendas

As regras de negócios definem como o sistema deve processar as vendas e gerenciar o estoque. Algumas regras essenciais no contexto de vendas podem incluir:

1. Verificação de Estoque: A venda de um produto só pode ser realizada se a quantidade disponível no estoque for maior ou igual à quantidade solicitada pelo cliente.
2. Cálculo do Total da Venda: O preço total da venda deve ser calculado multiplicando o preço de cada item pela sua respectiva quantidade.
3. Validação do Pagamento: O pagamento do cliente deve ser validado antes de processar a venda, garantindo que o pagamento foi concluído com sucesso.

A implementação dessas regras de maneira clara e modular é fundamental para garantir que o sistema seja eficiente e livre de erros. Além disso, a modularização do código facilita a reutilização das funcionalidades em diferentes partes do sistema e contribui para a manutenibilidade do código.

Exemplo de Regra de Negócio para Verificação de Estoque

Uma das regras de negócios mais críticas em sistemas de vendas é garantir que a quantidade de um produto disponível no estoque seja suficiente para atender a uma solicitação de venda. Caso a quantidade solicitada seja maior que a quantidade em estoque, a venda não pode ser realizada.

Modularização e Funcionalidades do Sistema de Vendas

A modularização é crucial para manter o sistema de vendas bem estruturado e organizado. Cada módulo ou função do código deve ser responsável por uma tarefa específica. Isso facilita a reutilização de código e a manutenção do sistema a longo prazo.

O sistema pode ser dividido em módulos como:

- Módulo de Vendas: Responsável por calcular o valor total da venda, validar a disponibilidade de produtos no estoque e processar a venda.
- Módulo de Estoque: Responsável por verificar a quantidade disponível dos produtos e atualizar o estoque após a venda.
- Módulo de Clientes: Responsável pelo cadastro de clientes e gerenciamento de suas informações.

Exemplos de Módulos e Funções no Sistema de Vendas

Módulo de Gestão de Vendas

No módulo de vendas, uma função essencial pode ser o cálculo do valor total de uma venda, que depende do preço de cada item multiplicado pela sua quantidade.

Módulo de Gestão de Estoque

No módulo de estoque, podemos ter funções que verificam a quantidade disponível de um produto e atualizam o estoque conforme as vendas são realizadas.

Módulo de Gestão de Clientes

O módulo de clientes deve ser responsável pelo cadastro e atualização das informações dos clientes. Isso é fundamental para que o sistema possa associar as vendas aos clientes e manter um controle adequado.

Código Limpo, Eficiente e Manutenível

Para garantir que o sistema de vendas e gestão de estoque seja eficiente e fácil de manter, é fundamental escrever um código limpo e bem organizado. A modularização do código facilita a manutenção e permite que mudanças ou adições de funcionalidades possam ser feitas sem afetar outras partes do sistema.

Para alcançar um código limpo e eficiente, algumas boas práticas devem ser seguidas:

- Dividir o código em funções e módulos independentes: Cada função ou módulo deve ser responsável por uma tarefa específica e não deve acumular responsabilidades.
- Utilizar nomes descritivos: Variáveis, funções e módulos devem ter nomes claros que indiquem exatamente o que eles representam ou fazem.
- Evitar duplicação de código: Trechos de código que se repetem em várias partes do sistema devem ser transformados em funções reutilizáveis.

3.2.3 IMPLEMENTAÇÃO E VALIDAÇÃO

A fase de implementação e validação no projeto de vendas foca na integração dos módulos criados ao longo do desenvolvimento, garantindo que a aplicação final seja funcional, coesa e eficiente. Durante esta etapa, os estudantes devem testar a interação entre os diferentes

módulos do sistema e validar se todos os requisitos do projeto estão sendo atendidos. O objetivo é garantir que cada funcionalidade do sistema funcione conforme o esperado quando integrada ao sistema completo.

Implementação dos Módulos no Sistema de Vendas

Durante o desenvolvimento do sistema de vendas, foram criados vários módulos com funcionalidades específicas. Esses módulos precisam ser integrados de forma eficiente para que o sistema como um todo funcione corretamente. A integração envolve garantir que os dados fluam entre os módulos de forma eficiente e sem erros de lógica.

Os principais módulos envolvidos no projeto de vendas incluem:

- Módulo de Vendas: Gerencia o processo de venda, validando a disponibilidade dos produtos no estoque, calculando o valor total da venda e registrando o pagamento.
- Módulo de Estoque: Controla o estoque de produtos, garantindo que a quantidade disponível seja atualizada conforme as vendas.
- Módulo de Clientes: Armazena e gerencia os dados dos clientes, além de manter o histórico de compras para cada um.

A integração entre esses módulos deve ser feita de maneira que a interação entre eles seja fluida e sem falhas. Por exemplo, ao realizar uma venda no módulo de vendas, o sistema deve verificar a disponibilidade de produtos no módulo de estoque e registrar o histórico de compras no módulo de clientes.

Exemplo de Módulos Integrados

- Módulo de Vendas: Realiza o cálculo do total de vendas e processa o pagamento.
- Módulo de Estoque: Atualiza a quantidade disponível de um produto após cada venda.
- Módulo de Clientes: Registra a compra e mantém o histórico de compras de cada cliente.

Cada módulo precisa interagir com os outros de maneira eficiente, garantindo que as transações sejam realizadas sem erros e com dados consistentes.

Validação da Aplicação Final

Após os testes, é necessário realizar a validação final do sistema para garantir que todos os requisitos do projeto foram atendidos e que o sistema funcione corretamente. A validação envolve garantir que:

1. Requisitos funcionais: Todos os requisitos do sistema foram implementados corretamente. Por exemplo, ao registrar uma venda, o sistema deve verificar a disponibilidade do produto no estoque, atualizar a quantidade disponível e registrar o histórico de compras do cliente.
2. Integração entre os módulos: Os módulos interagem corretamente uns com os outros. Por exemplo, ao realizar uma venda, o módulo de vendas deve consultar o módulo de estoque para verificar a disponibilidade e o módulo de clientes para registrar o histórico da compra.
3. Desempenho do sistema: O sistema deve ser capaz de lidar com várias transações sem diminuir o desempenho. Isso pode envolver a realização de testes de desempenho para garantir que o sistema continue funcionando de maneira eficiente, mesmo com grandes volumes de dados.
4. Estrutura do código: O código deve ser bem estruturado, modular e de fácil manutenção, facilitando futuras atualizações ou modificações.

3.3 MODELAGEM DE DADOS

(Conjunto de Imagens 2)

O sistema foi projetado para gerenciar diversos aspectos operacionais de uma empresa, como cadastro de usuários (gerentes e vendedores), fornecedores, clientes, produtos, vendas e controle de estoque. A modelagem de dados é um dos pilares fundamentais do projeto, pois garante a organização, integridade e eficiência no armazenamento e na recuperação das informações.

Para garantir uma gestão eficiente, todos os dados cadastrais são armazenados em um banco de dados MySQL, que utiliza um esquema relacional. Cada entidade (usuários, fornecedores, clientes, produtos, vendas e estoque) é representada por uma tabela, com relacionamentos bem definidos entre elas para assegurar a consistência e facilitar consultas complexas. A identificação única de cada registro, por meio de IDs gerados automaticamente, permite a localização rápida e precisa das informações.

Além disso, essa estrutura de dados proporciona uma base robusta para análises futuras, como a integração com ferramentas como o Power BI, que permite a visualização do desempenho da empresa e a tomada de decisões estratégicas. Neste módulo, exploraremos como as tabelas foram projetadas, como as relações entre elas foram estabelecidas e como essas decisões impactam tanto a eficiência do sistema quanto as capacidades analíticas do projeto.

3.3.1 MODELO CONCEITUAL

(Conjunto de Imagens 2)

3.3.2 MODELO LÓGICO E FÍSICO

(Conjunto de Imagens 2)

3.3.3 SQL

(Conjunto de Imagens 2)

3.4 BUSINESS INTELLIGENCE

(Conjunto de Imagens 3)

3.4.1 ORGANIZAÇÃO E IDENTIFICAÇÃO DAS INFORMAÇÕES

No contexto deste projeto, a organização e identificação das informações desempenham um papel crucial para mensurar os atendimentos e a utilização do sistema de forma eficaz. Através do cadastro detalhado de usuários, clientes, produtos e vendas, é possível coletar dados relevantes sobre o desempenho da equipe de vendas, os produtos mais vendidos, os comportamentos de compra dos clientes e a eficiência do estoque.

Essas informações são organizadas em um banco de dados relacional, permitindo a análise dinâmica de métricas chave como volume de vendas. Integrar esses dados com ferramentas de Business Intelligence, como o Power BI, conseguimos criar dashboards interativos que visualizam tendências de desempenho, identificam áreas de melhoria e facilitam a tomada de decisões estratégicas, contribuindo para uma gestão mais eficiente e orientada por dados.

3.4.2 MANIPULAÇÃO E ANÁLISE DE DADOS

A base de dados deste projeto foi organizada de forma relacional, com tabelas distintas para cada entidade-chave do sistema: usuários, fornecedores, clientes, produtos, vendas e estoque. Cada tabela terá um campo de ID único, garantindo a identificação e integridade dos registros. Relacionamentos entre as tabelas, como vendas associadas a produtos e clientes, são estabelecidos por meio de chaves estrangeiras, permitindo uma navegação eficiente e consultas precisas. Além disso, a base de dados é estruturada para suportar a análise de dados, com dados históricos sobre vendas, compras e estoque, de modo a possibilitar a geração de relatórios e insights para decisões estratégicas.

3.4.3 CRIAÇÃO DE MODELOS DE ANÁLISE DE DADOS

(Imagem 3)

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: GERENCIANDO FINANÇAS

A Formação para a Vida é um dos eixos do Projeto Pedagógico de Formação por Competências da UNIFEOB.

Esta parte do projeto está diretamente relacionada com a extensão universitária, ou seja, o objetivo é que seja aplicável e que tenha real utilidade para a sociedade, de um modo geral.

3.5.1 GERENCIANDO FINANÇAS

Este projeto de gestão empresarial, focado na modelagem de dados e na integração de Business Intelligence para análise de desempenho, pode ser diretamente relacionado aos conceitos econômicos e financeiros descritos nos quatro tópicos propostos. A seguir, sintetizamos como cada tópico se conecta ao projeto e ao seu uso prático no dia a dia de uma empresa:

Tópico 1: Introdução aos conceitos econômicos e financeiros básicos.
No contexto do projeto, os conceitos econômicos e financeiros básicos são fundamentais para entender o funcionamento da empresa e a análise de dados. Por exemplo, o controle de vendas e a gestão de estoque diretamente afetam a rentabilidade da empresa. O sistema de vendas permite calcular o valor total de vendas, analisar margens de lucro e comparar o custo dos produtos com o preço de venda, permitindo uma avaliação constante da saúde financeira da empresa.

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

A utilização do sistema para registro diário de vendas, compras e controle de estoque contribui para a construção de um panorama financeiro da empresa. Com isso, é possível determinar a riqueza da empresa em termos de valor de estoque, faturamento e lucro líquido. A análise de dados financeiros gerados pelo sistema ajuda a identificar quais áreas da empresa geram mais receita e onde os custos podem ser reduzidos, promovendo uma busca pela independência financeira no longo prazo.

Tópico 3: Dívidas e juros compostos, opções de empréstimo e alternativas ao endividado
A gestão das finanças da empresa também envolve a análise de dívidas e financiamentos. Se o sistema de vendas indicar uma queda no fluxo de caixa, pode ser necessário recorrer a empréstimos ou opções de crédito. O sistema pode identificar o impacto dos juros compostos sobre dívidas existentes e simular as condições de empréstimo, ajudando a empresa a planejar o pagamento de dívidas e evitar o endividamento excessivo. Além disso, o controle de estoque e vendas a prazo pode impactar diretamente no endividamento da empresa, tornando fundamental um bom planejamento financeiro.

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

O sistema de gestão empresarial pode ser uma ferramenta importante para estabelecer metas financeiras e operacionais, como aumento de vendas, melhoria no controle de estoque ou redução de custos. O uso de Business Intelligence permite criar relatórios e indicadores que facilitam o acompanhamento do progresso em relação a essas metas. Ao envolver todos os membros da equipe, como gerentes e vendedores, na utilização desses dados, a empresa pode alinhar seus esforços para alcançar objetivos comuns, como a expansão de mercado ou a melhoria da satisfação do cliente.

Exemplos Práticos:

1. Meta de vendas: Um objetivo de aumentar o faturamento mensal em 10% pode ser monitorado através do sistema de vendas, com o acompanhamento contínuo do desempenho dos produtos e vendedores.
2. Gestão de estoque e fluxo de caixa: A integração entre estoque e vendas ajuda a evitar o excesso de estoque ou a falta de produtos que impactem as vendas, assegurando um fluxo de caixa mais saudável.
3. Planejamento de dívidas: Se a empresa precisar recorrer a crédito, o sistema pode calcular o impacto dos juros compostos sobre os empréstimos e sugerir alternativas para reduzir o endividamento.

3.5.2 ESTUDANTES NA PRÁTICA


GERENCIANDO
FINANÇAS
INTEGRAÇÃO COM MYSQL E POWERBI
INOVADOR E DINÂMICO
FLUXO DE DADOS EFICIENTE
Tópico 1: Conceitos econômicos se relacionam com a análise de dados.
Tópico 2: O registro diário de vendas e estoque permite maior independência.
Tópico 3: A integração com Power BI permite a análise e previsão, impedindo mal planejamento que resulta em dívidas.
Tópico 4: O sistema é perfeito para a criação de metas e ampliação de mercado, sendo elas em vendas, estoque ou metas internas com análises de dados.
MySQL
Dashboards que registram e analisam seu cotidiano, realize um planejamento de dívidas
Conceitos Econômicos simplificados
Escalabilidade do seu Banco com seu negócio:
Gerencie módulos e funcionalidades que se encaixam e valorize sua riqueza.
Estabeleça metas, gerencie seu estoque e fluxo de caixa
A solução perfeita para impulsionar seu negócio. Sistema integrado de cadastro, controle de estoque e vendas, aliado à análises do Power BI e MySQL. Você terá tudo que precisa para tomar as melhores decisões. Otimize suas operações e garanta os resultados! Vamos transformar sua gestão hoje!
Grupo 22 - Unifeob

4. CONCLUSÃO

O projeto de Gestão Empresarial, desenvolvido em Python com integração ao banco de dados MySQL e análise de dados no Power BI, buscou fornecer uma solução robusta para o gerenciamento de usuários, fornecedores, clientes, produtos, vendas e estoque de uma empresa. O sistema foi modelado para garantir a organização e a integridade das informações, criando uma estrutura de dados eficiente e interligada, capaz de gerar relatórios e indicadores para suportar decisões estratégicas.

Principais Pontos Abordados:

1. Modelagem de Dados: Criamos uma base de dados relacional com tabelas para cada entidade-chave do sistema (usuários, fornecedores, clientes, produtos, vendas e estoque), com IDs únicos e relacionamentos bem definidos. Isso permite a fácil recuperação e análise de informações, além de garantir a integridade e a consistência dos dados.
2. Integração com Business Intelligence: A integração com o Power BI possibilitou a criação de dashboards interativos, que ajudam na visualização do desempenho da empresa. Indicadores como volume de vendas, movimentação de estoque e desempenho de vendedores foram monitorados e analisados, permitindo a tomada de decisões mais informadas.
3. Gestão de Vendas e Estoque: O sistema possibilita o controle detalhado de vendas e estoque, com a geração de relatórios que facilitam a análise da rentabilidade dos produtos, a demanda dos clientes e o controle de inventário, assegurando que a empresa mantenha um fluxo de caixa saudável e evite perdas por falta ou excesso de estoque.
4. Desafios e Superações: Um dos maiores desafios enfrentados durante o desenvolvimento do projeto foi a saída de dois membros da equipe, o que reduziu o número de integrantes para apenas dois. Essa redução de equipe exigiu uma reorganização do trabalho e uma gestão mais eficiente do tempo e dos recursos disponíveis. Apesar disso, conseguimos manter o projeto dentro do cronograma, com a entrega de todas as funcionalidades principais e a integração de todas as partes do sistema.
5. Análise de Desempenho e Resultados: Através da coleta e análise de dados, conseguimos identificar áreas de melhoria, como o desempenho de vendas de certos produtos, o comportamento de compra dos clientes e a eficiência do estoque. Esses

dados foram fundamentais para gerar relatórios de desempenho e para estabelecer metas de vendas mais realistas e estratégias de crescimento para a empresa.

Dificuldades Encontradas:

Como mencionado, uma das maiores dificuldades foi a saída de dois colegas da equipe, o que reduziu significativamente nossa capacidade de dividir as tarefas. Essa mudança inesperada nos forçou a assumir mais responsabilidades individuais, além de aumentar a carga de trabalho. No entanto, conseguimos superar essa adversidade com boa comunicação e priorização das atividades mais críticas. Essa experiência também nos ensinou a importância de uma gestão de equipe ágil e da adaptação rápida a mudanças.

Considerações Finais:

O projeto foi uma excelente oportunidade para aplicar os conceitos de modelagem de dados, integração de sistemas e análise de desempenho em um ambiente real de negócios. Apesar dos desafios, conseguimos entregar uma solução funcional que não só organiza e gerencia as operações da empresa, mas também fornece insights valiosos para decisões estratégicas por meio de relatórios e indicadores baseados em dados. A experiência foi enriquecedora tanto do ponto de vista técnico quanto de trabalho em equipe, e certamente contribuiu para o nosso crescimento profissional.

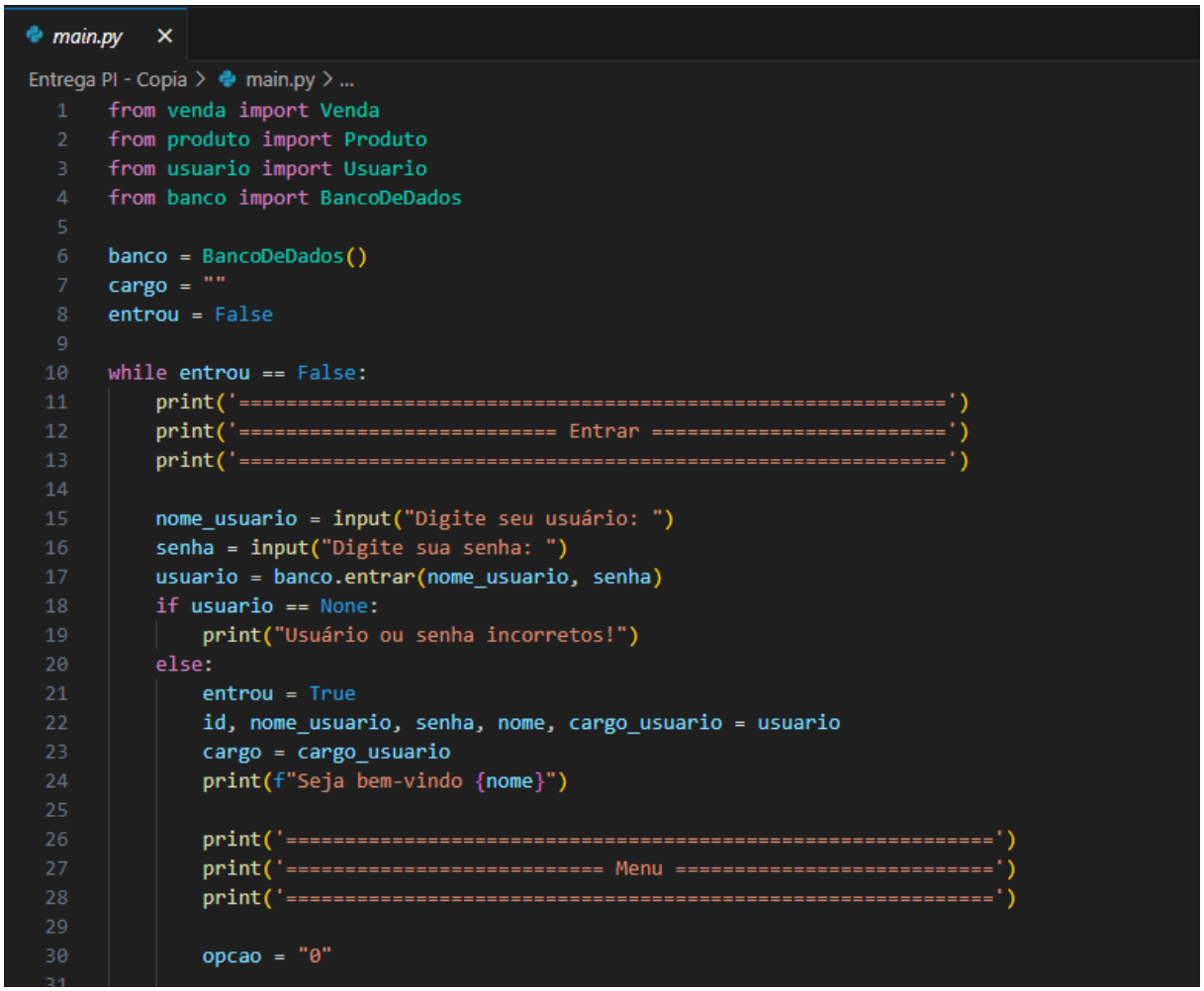
ISSN 1983-6767

REFERÊNCIAS

ISSN 1983-6767

ANEXOS

Conjunto de Imagens 1:



```python
from venda import Venda
from produto import Produto
from usuario import Usuario
from banco import BancoDeDados

banco = BancoDeDados()
cargo = ""
entrou = False

while entrou == False:
    print('============================================================')
    print('========================== Entrar ==========================')
    print('============================================================')

    nome_usuario = input("Digite seu usuário: ")
    senha = input("Digite sua senha: ")
    usuario = banco.entrar(nome_usuario, senha)
    if usuario == None:
        print("Usuário ou senha incorretos!")
    else:
        entrou = True
        id, nome_usuario, senha, nome, cargo_usuario = usuario
        cargo = cargo_usuario
        print(f"Seja bem-vindo {nome}")

        print('============================================================')
        print('========================== Menu ============================')
        print('============================================================')

        opcao = "0"
```

banco.py 2

Entrega PI - Copia > banco.py > ...

```python
import mysql.connector
from mysql.connector import Error

class BancoDeDados:
    def __init__(self):
        self.config = {
            'user': 'root',
            'password': 'sdf67fsadjk',
            'host': 'localhost',
            'database': 'mydb'
        }

        try:
            self.connection = mysql.connector.connect(**self.config)
            if self.connection.is_connected():
                print("Conexão bem-sucedida ao MySQL!")

                cursor = self.connection.cursor()
                cursor.execute("SELECT DATABASE();")
                record = cursor.fetchone()
                print("Conectado ao banco de dados: ", record)
                cursor.close()

        except mysql.connector.Error as err:
            print("Erro ao conectar com banco de dados: ", err)
            exit()

    def cadastrar_produto(self, nome_produto, descricao_produto, preco_produto, categoria_produto, quantidade_produto, id_fornecedor_produto):
        cursor = self.connection.cursor()
        sql = "INSERT INTO Produto (nome, descricao, preco, categoria, quantidade, id_fornecedor) VALUES (%s, %s, %s, %s, %s, %s)"
        values = (nome_produto, descricao_produto, preco_produto, categoria_produto, quantidade_produto, id_fornecedor_produto)
        cursor.execute(sql, values)
        self.connection.commit()
```

produto.py

Entrega PI - Copia > produto.py > Produto > getNome

```python
class Produto:
    def __init__(self, nome, descricao, preco, categoria, quantidade, id_fornecedor):
        self.__nome = nome
        self.__descricao = descricao
        self.__preco = preco
        self.__categoria = categoria
        self.__quantidade = quantidade
        self.__id_fornecedor = id_fornecedor


    def getNome(self):
        return self.__nome

    def getDescricao(self):
        return self.__descricao

    def getQuantidade(self):
        return self.__quantidade

    def getPreco(self):
        return self.__preco

    def getCategoria(self):
        return self.__categoria

    def getFornecedor(self):
        return self.__id_fornecedor

    def setNome(self, values):
        self.nome = values

    def setPreco(self, values):
        self.preco = values

    def setQuantidade(self, values):
        self.quantidade = values
```

usuario.py

Entrega PI - Copia > usuario.py > Usuario

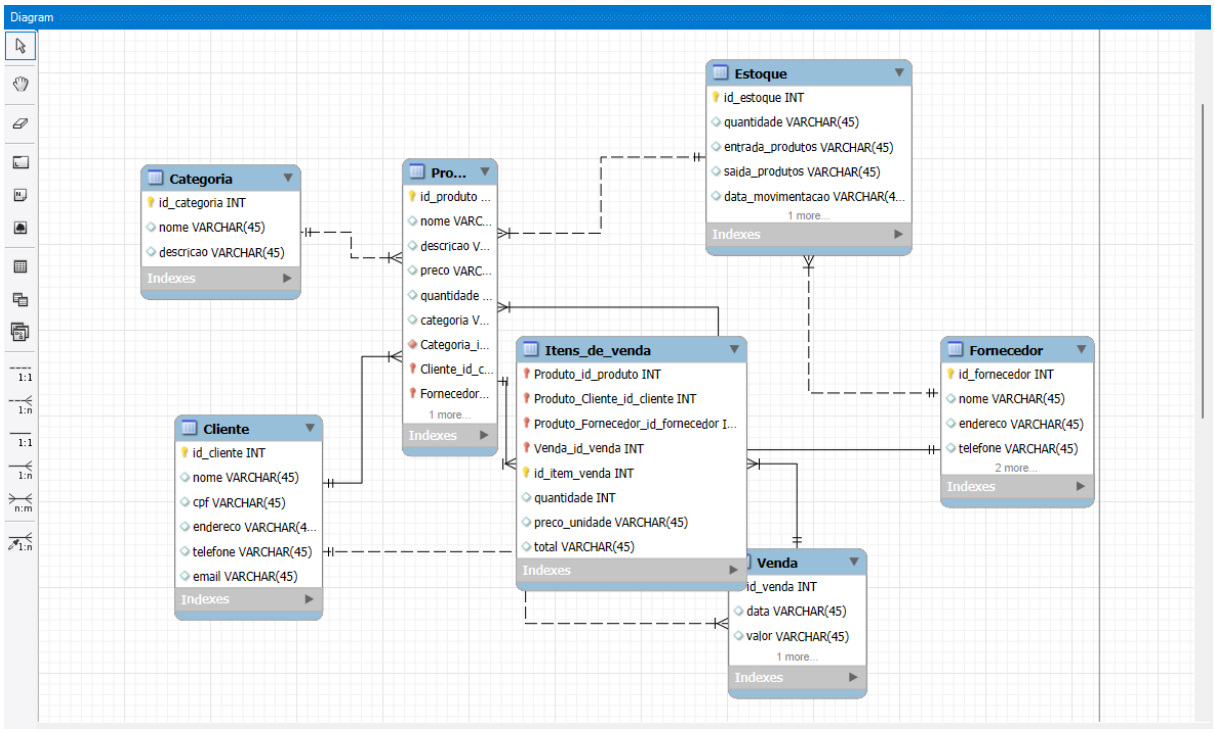
```python
class Usuario:
    def __init__(self, usuario, senha, nome, id_cargo):
        self.__usuario = usuario
        self.__senha = senha
        self.__nome = nome
        self.__id_cargo = id_cargo

    def getUsuario(self):
        return self.__usuario

    def getSenha(self):
        return self.__senha

    def getNome(self):
        return self.__nome

    def getidCargo(self):
        return self.__id_cargo
```

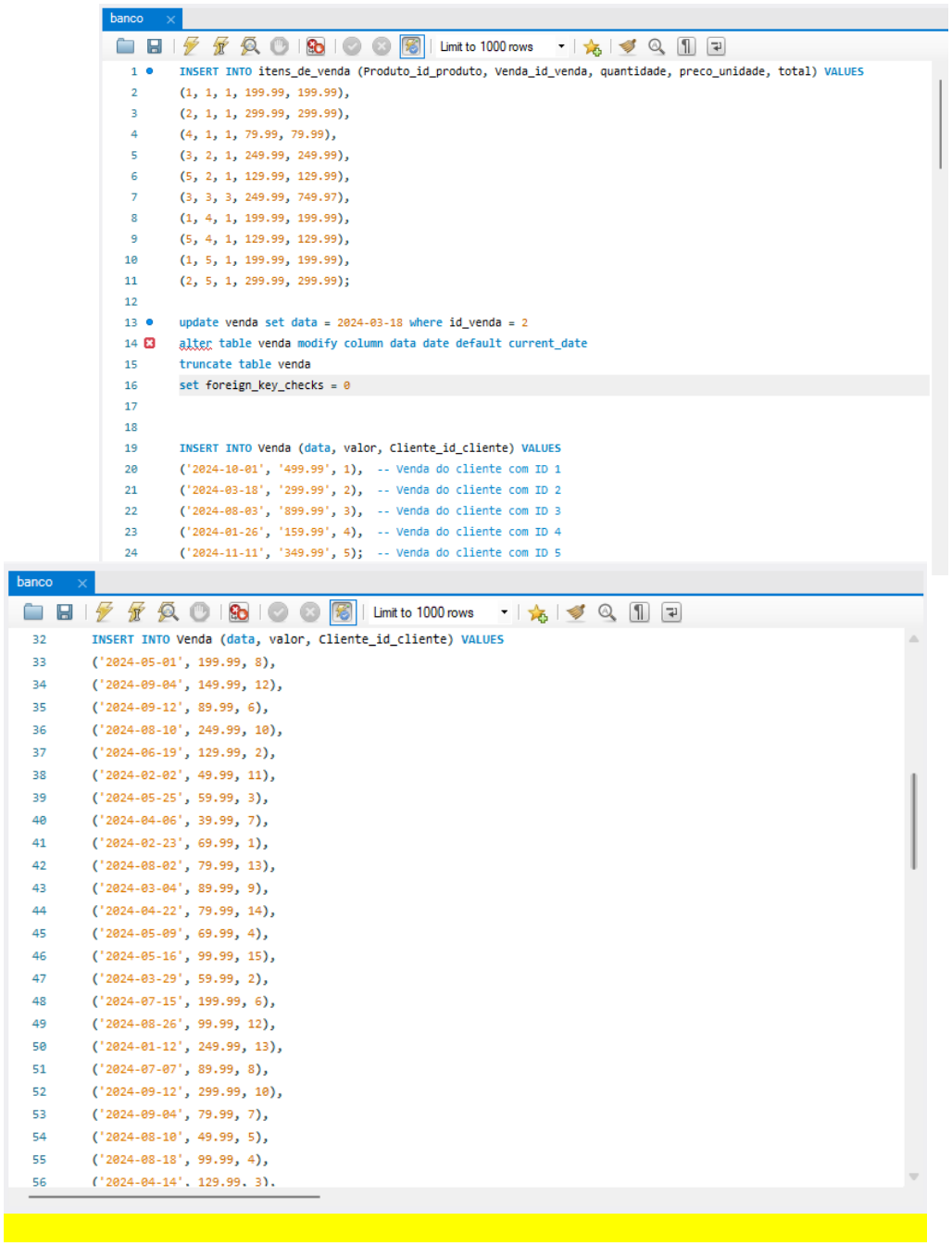
venda.py

Entrega PI - Copia > venda.py > Venda

```python
class Venda:
    def __init__(self, valor, cliente):
        self.__valor = valor
        self.__cliente = cliente

    def getCliente(self):
        return self.__cliente

    def getValor(self):
        return self.__valor

class Item_venda:
    def __init__(self, id_produto, id_venda, quantidade, preco_unidade, total):
        self.__id_produto = id_produto
        self.__id_venda = id_venda
        self._quantidade = quantidade
        self.__preco_unidade = preco_unidade
        self.__total = total
```

Conjunto de Imagens 2:




banco
Limit to 1000 rows
INSERT INTO itens_de_venda (Produto_id_produto, Venda_id_venda, quantidade, preco_unidade, total) VALUES
(1, 1, 1, 199.99, 199.99),
(2, 1, 1, 299.99, 299.99),
(4, 1, 1, 79.99, 79.99),
(3, 2, 1, 249.99, 249.99),
(5, 2, 1, 129.99, 129.99),
(3, 3, 3, 249.99, 749.97),
(1, 4, 1, 199.99, 199.99),
(5, 4, 1, 129.99, 129.99),
(1, 5, 1, 199.99, 199.99),
(2, 5, 1, 299.99, 299.99);
update venda set data = 2024-03-18 where id_venda = 2
alter table venda modify column data date default current_date
truncate table venda
set foreign_key_checks = 0
INSERT INTO Venda (data, valor, Cliente_id_cliente) VALUES
('2024-10-01', '499.99', 1), -- Venda do cliente com ID 1
('2024-03-18', '299.99', 2), -- Venda do cliente com ID 2
('2024-08-03', '899.99', 3), -- Venda do cliente com ID 3
('2024-01-26', '159.99', 4), -- Venda do cliente com ID 4
('2024-11-11', '349.99', 5); -- Venda do cliente com ID 5
banco
Limit to 1000 rows
INSERT INTO Venda (data, valor, Cliente_id_cliente) VALUES
('2024-05-01', 199.99, 8),
('2024-09-04', 149.99, 12),
('2024-09-12', 89.99, 6),
('2024-08-10', 249.99, 10),
('2024-06-19', 129.99, 2),
('2024-02-02', 49.99, 11),
('2024-05-25', 59.99, 3),
('2024-04-06', 39.99, 7),
('2024-02-23', 69.99, 1),
('2024-08-02', 79.99, 13),
('2024-03-04', 89.99, 9),
('2024-04-22', 79.99, 14),
('2024-05-09', 69.99, 4),
('2024-05-16', 99.99, 15),
('2024-03-29', 59.99, 2),
('2024-07-15', 199.99, 6),
('2024-08-26', 99.99, 12),
('2024-01-12', 249.99, 13),
('2024-07-07', 89.99, 8),
('2024-09-12', 299.99, 10),
('2024-09-04', 79.99, 7),
('2024-08-10', 49.99, 5),
('2024-08-18', 99.99, 4),
('2024-04-14', 129.99, 3),

banco

Limit to 1000 rows

```
INSERT INTO itens_de_venda (Produto_id_produto, Venda_id_venda, quantidade, preco_unidade, total) VALUES
(6, 6, 1, 199.99, 199.99),
(7, 7, 1, 149.99, 149.99),
(8, 8, 1, 89.99, 89.99),
(9, 9, 1, 249.99, 249.99),
(10, 10, 1, 129.99, 129.99),
(11, 11, 1, 49.99, 49.99),
(12, 12, 1, 59.99, 59.99),
(13, 13, 1, 39.99, 39.99),
(14, 14, 1, 69.99, 69.99),
(15, 15, 1, 79.99, 79.99),
(16, 16, 1, 89.99, 89.99),
(17, 17, 1, 79.99, 79.99),
(18, 18, 1, 69.99, 69.99),
(19, 19, 1, 99.99, 99.99),
(20, 20, 1, 59.99, 59.99),
(21, 21, 1, 199.99, 199.99),
(22, 22, 1, 99.99, 99.99),
(23, 23, 1, 249.99, 249.99),
(24, 24, 1, 89.99, 89.99),
(25, 25, 1, 299.99, 299.99),
(26, 26, 1, 79.99, 79.99),
(27, 27, 1, 49.99, 49.99),
(28, 28, 1, 99.99, 99.99),
(29, 29, 1, 129.99, 129.99),
```

banco

Limit to 1000 rows

```
INSERT INTO cliente (nome, cpf, endereco, telefone, email) VALUES
('Fernanda Lima', '112.233.445-66', 'Rua F, 303', '(11) 98765-4321', 'fernanda.lima@gmail.com'),
('Paulo Souza', '667.788.999-00', 'Rua G, 404', '(11) 92123-4567', 'paulo.souza@gmail.com'),
('Juliana Almeida', '334.556.778-99', 'Rua H, 505', '(11) 98765-9876', 'juliana.almeida@gmail.com'),
('Eduardo Costa', '998.877.665-44', 'Rua I, 606', '(11) 91123-4567', 'eduardo.costa@gmail.com'),
('Luciana Pereira', '223.344.556-77', 'Rua J, 707', '(11) 92345-6789', 'luciana.pereira@gmail.com'),
('Marcelo Rocha', '111.222.333-44', 'Rua K, 808', '(11) 93456-7891', 'marcelo.rocha@gmail.com'),
('Tatiane Santos', '444.555.666-33', 'Rua L, 909', '(11) 94567-8902', 'tatiane.santos@gmail.com'),
('Roberta Martins', '666.777.888-11', 'Rua M, 1010', '(11) 95678-9013', 'roberta.martins@gmail.com'),
('Fernando Oliveira', '333.444.555-88', 'Rua N, 1111', '(11) 96789-0123', 'fernando.oliveira@gmail.com'),
('Renata Souza', '555.777.888-22', 'Rua O, 1212', '(11) 97890-1234', 'renata.souza@gmail.com');


INSERT INTO Venda (data, valor, Cliente_id_cliente) VALUES
('2024-05-01', 579.97, 8),
('2024-09-04', 379.98, 12),
('2024-09-12', 749.97, 6),
('2024-08-10', 329.98, 10),
('2024-06-19', 499.98, 2),
('2024-02-02', 199.99, 11),
('2024-05-25', 149.99, 3),
('2024-04-06', 89.99, 7),
('2024-02-23', 249.99, 1),
('2024-08-02', 129.99, 13),
('2024-03-04', 49.99, 9),
```

Imagem 3:


Total de Faturamento
5,56 Mil
Vendas
id_venda Ano Mês Dia valor Cliente_id_cliente
1 2024 maio 1 579.97 8
2 2024 setembro 4 379.98 12
3 2024 setembro 12 749.97 6
4 2024 agosto 10 329.98 10
Produtos
id_produto nome descricao categoria
1 Tênis Esportivo Tênis leve e confortável, ideal para corrida. Sapatos
2 Sapato Social Sapato de couro legítimo, ideal para ocasiões formais. Sapatos
3 Bota de Caminhada Bota resistente, perfeita para trilhas e caminhadas. Sapatos
4 Sandália Casual Sandália confortável para o dia a dia. Sapatos
5 Sapatilha Feminina Sapatilha elegante, disponível em várias cores. Sapatos
6 Calça Jeans Calça jeans de corte reto, ideal para o dia a dia. Calças
7 Calça de Sarja Calça de sarja leve, confortável e prática. Calças
8 Calça de Moletom Calça de moletom macia e aconchegante. Calças
9 Calça Social Calça social perfeita para ocasiões formais. Calças
10 Calça Cargo Calça cargo com muitos bolsos, ideal para aventuras. Calças
11 Camiseta Básica Camiseta básica de algodão, ideal para o dia a dia. Camisetas
Total
Presença de Produtos em Vendas
Tênis Esportivo
Bota de Caminha...
Sapatilha Feminina
Sapato Social
Bermuda Cargo
Bermuda de Mol...
Bermuda de Praia
Bermuda de Sarja
Clientes
id_cliente nome cpf endereco telefone email
1 João Silva 123.456.789-00 Rua A, 123 (11) 91234-5678 joao.silva@gmail.com
2 Maria Oliveira 987.654.321-00 Rua B, 456 (11) 99876-5432 maria.oliveira@gmail.com
3 Carlos Pereira 555.666.777-00 Rua C, 789 (11) 93456-7890 carlos.pereira@gmail.com
4 Ana Costa 444.333.222-00 Rua D, 101 (11) 94567-8901 ana.costa@gmail.com
5 Roberto Santos 222.333.444-00 Rua E, 202 (11) 95678-9012 roberto.santos@gmail.com
Vendas X Mês
janeiro
março
abril
maio
junho
julho
agosto