# **UNIVERSIDADE AGOSTINHO NETO**

# **FACULDADE DE ENGENHARIA**

# **DEPARTAMENTO DE ENGENHARIA INFORMÁTICA**

# **RELATÓRIO: COMPRADOR-FORNECEDOR**

**Curso: Engenharia Informática**

**Cadeira: Estrutura de Dados**

**Ano: 2º   
Semestre: 2º**

**Docente**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Eng.º Dinho Chipongue**

# **Integrantes**

* Paulo Josué Victor Peso
* Manuel Bernardo Soares Ngola

# 

# **Sumário**

[**Sumário** 3](#_Toc201164559)

[**Introdução** 4](#_Toc201164560)

[**Java** 5](#_Toc201164561)

[**Estrutura de Dados** 5](#_Toc201164562)

[**Principais tipos de Estruturas de Dados:** 5](#_Toc201164563)

[**Base de Dados MySQL** 6](#_Toc201164564)

[**Arquitetura MVC** 7](#_Toc201164565)

[**Benefícios do uso do padrão MVC** 7](#_Toc201164566)

[**Relação entre Comprador e Fornecedor** 7](#_Toc201164567)

[**Sistema Comprador-Fornecedores** 8](#_Toc201164568)

[**Objetivos do Sistema** 8](#_Toc201164569)

[**Arquitetura do Sistema (MVC)** 8](#_Toc201164570)

[**Funcionalidades Principais** 9](#_Toc201164571)

[**Implementação e Estrutura do Código** 9](#_Toc201164572)

[**Persistência de Dados (Banco de Dados MySQL)** 9](#_Toc201164573)

[**Exemplo de Fluxo de Uso** 10](#_Toc201164574)

[**Análise Estatística com Dados Simulados** 10](#_Toc201164575)

[**Tabela de Frequência** 11](#_Toc201164576)

[**Interpretação dos Dados** 11](#_Toc201164577)

[**Benefícios da Análise Estatística** 12](#_Toc201164578)

[**Estimativa de Custo do Sistema** 12](#_Toc201164579)

[**Tela de Menu Principal:** 13](#_Toc201164580)

[**Cadastrar Produto:** 14](#_Toc201164581)

[**Tabela de Fornecedores:** 16](#_Toc201164582)

[**Tabela Ofertas:** 16](#_Toc201164583)

[**Tabela Pedidos itens:** 17](#_Toc201164584)

[**Tabela de Pedidos:** 17](#_Toc201164585)

[**Conclusão** 19](#_Toc201164586)

[**Referências Bibliográficas** 20](#_Toc201164587)

# **Introdução**

O sistema **"Comprador-Fornecedor"** é uma solução de software desenvolvida para automatizar e otimizar o processo de compras entre empresas e não só (compradores) e fornecedores. Seu principal objetivo é permitir que empresas possam realizar pedidos de produtos com base nas melhores ofertas disponíveis, considerando critérios como preço e prazo de entrega. O sistema torna o processo de tomada de decisão mais rápido, transparente e eficaz, promovendo uma melhor gestão de estoque, controle de fornecedores e eficiência nas compras corporativas.

A aplicação foi desenvolvida em **Java** com uso de **interface gráfica Swing,** conectada a uma base de dados **MySQL,** e estruturada segundo o **padrão MVC (Model-View-Controller)** para garantir separação de responsabilidades, modularidade e manutenção facilitada.

# **Java**

O sistema foi desenvolvido em **Java**, uma linguagem de programação orientada a objetos bastante utilizada para o desenvolvimento de sistemas corporativos, web e mobile devido à sua portabilidade, segurança e vasta comunidade. Java oferece ferramentas robustas para a construção de aplicações desktop e integração com bancos de dados.   
  
**Entre suas principais características estão:**

* **Portabilidade (WORA – Escreva uma vez, execute em qualquer lugar)**: O código Java é compilado para **bytecode** , que roda em qualquer sistema com uma **Java Virtual Machine (JVM)** .
* **Orientação a Objetos**: Java segue os **quatro pilares do POO** : encapsulamento, herança, polimorfismo e abstração.
* **Fortemente Tipada e Segura**: Java verifica erros em tempo de construção e execução,
* **Multithreading**: Permite a execução de **várias tarefas ao mesmo tempo** , ideal para sistemas complexos e aplicações em tempo real.
* **Bibliotecas Ricas e Comunidade Ativa**: Java possui **vastas bibliotecas e frameworks** (como Spring, Hibernate, JavaFX), além de uma **grande comunidade mundial de desenvolvedores .**
* **Independência de Plataforma**: Um programa Java pode rodar no **Windows, Linux, MacOS** e outros sistemas sem mudanças no código

# **Estrutura de Dados**

Estrutura de Dados é uma forma organizada de armazenar, gerenciar e manipular dados em um programa de forma eficiente. Ela define como os dados são organizados na memória do computador e quais operações podem ser realizadas sobre eles, como inserção, remoção, busca e ordenação  
  
**Por que Estrutura de Dados é importante?**

* **Eficiência** : Um bom uso de estruturas de dados torna o sistema mais rápido e leve.
* **Organização** : Permite representar dados de forma lógica, facilitando o desenvolvimento.
* **Otimização de recursos** : Ajuda a consumir menos memória e processamento.

## **Principais tipos de Estruturas de Dados:**

* **Vetores (Matrizes):**
  + Armazenam elementos em posições fixas e sequenciais.
  + Exemplo:int[] numeros = {1, 2, 3};
* **Listas (Lista, Lista de Matriz, Lista Encadeada):**
  + Armazenam coleções dinâmicas de elementos.
  + Permitem inserções e remoções com mais flexibilidade.
* **Pilhas:**
  + Seguem o princípio **LIFO** (Last In, First Out).
  + Exemplo: sistemas de navegação (voltar página).
* **Filas (Filas):**
  + Seguem o princípio **FIFO** (First In, First Out).
  + Exemplo: sistemas de atendimento ou impressão.
* **Mapa:**
  + Armazenam pares **chave-valor** .
  + Exemplo: associar um ID a um nome de produto.
* **Árvores e Grafos (avançado):**
  + Usadas em buscas hierárquicas e estruturas de redes.

# **Base de Dados MySQL**

Uma **base de dados**, ou **banco de dados**, é um sistema organizado de armazenamento de informações. Ela permite que dados sejam **inseridos, consultados, atualizados e deletados** com eficiência e segurança. O principal objetivo de uma base de dados é **organizar grandes volumes de informação**, de forma que possam ser acessados de maneira rápida e confiável por sistemas computacionais.  
  
O **MySQL** é um sistema de gerenciamento de banco de dados (SGBD) relacional, que **usa a linguagem SQL (Structured Query Language)** para manipular os dados.

Ele é **gratuito, de código aberto, robusto, rápido e amplamente utilizado** tanto em aplicações pequenas quanto em sistemas de grande escala.

No desenvolvimento de aplicações modernas, como sistemas de vendas, bancos, redes sociais e plataformas empresariais, as bases de dados são **indispensáveis**.  
  
**Por que usar o MySQL?**

* **Facilidade de uso:** é simples de instalar, configurar e operar.
* **Desempenho elevado:** é otimizado para alta performance em leitura e escrita de dados.
* **Suporte a dados relacionais:** permite criar relações entre tabelas usando **chaves primárias e estrangeiras,** garantindo integridade.
* **suporte:** possui vasta documentação, comunidade ativa e integração com diversas linguagens de programação, como **Java, PHP, Python**, entre outras.
* **Segurança e controle de acesso**: possibilita definir permissões para diferentes usuários.

A utilização do MySQL como base de dados no projeto foi essencial para garantir uma arquitetura robusta, organizada e escalável . Ele possibilitou que as funcionalidades do sistema fossem inovadoras com segurança, eficiência e aderência a boas práticas de desenvolvimento.

# **Arquitetura MVC**

O **MVC** , sigla para **Model-View-Controller** (Modelo-Visão-Controlador), é um padrão de arquitetura amplamente utilizado no desenvolvimento de sistemas, especialmente aplicações com interface gráfica. Seu objetivo principal é **separar a lógica da aplicação em três camadas distintas** , facilitando a organização, manutenção e escalabilidade do código

## **Benefícios do uso do padrão MVC**

* **Separação de responsabilidades** : facilita a manutenção e o entendimento do sistema.
* **Reutilização de código** : as partes do sistema são independentes e podem ser reutilizadas.
* **Facilidade para testes** : permite testar cada camada separadamente.
* **Melhor organização do projeto** : cada funcionalidade fica localizada em seu devido lugar (modelo, visão ou controle).

O uso do padrão MVC no projeto Comprador-Fornecedores trouxe **organização, clareza e robustez** à estrutura da aplicação. Ele permitiu um desenvolvimento mais limpo e facilitou a implementação de funcionalidades complexas, como a escolha da melhor oferta ou a atualização de dados diretamente na interface, mantendo o código modular e reutilizável.

# **Relação entre Comprador e Fornecedor**

No mundo dos negócios, a **relação entre compradores e fornecedores** é

fundamental para o funcionamento de qualquer cadeia de suprimentos. Essa relação envolve **trocas comerciais**,

onde um lado adquire bens ou serviços, e o outro os fornece.

Essa interação precisa ser eficiente, confiável e baseada em boas práticas para garantir o sucesso de ambas as partes.

Uma boa relação entre compradores e fornecedores traz diversos benefícios:

* **Redução de custos:** com melhores negociações e aproveitamento de ofertas.
* **Confiabilidade:** entregas no prazo e produtos de qualidade.
* **Eficiência operacional:** processos mais rápidos e automáticos.
* **Tomada de decisão estratégica:** com base em dados sobre preços, prazos e desempenho dos fornecedores.

A dinâmica entre compradores e fornecedores é um **pilar essencial** para o funcionamento de empresas,

com impactos diretos na qualidade do serviço prestado, nos custos e na satisfação do cliente final.

# **Sistema Comprador-Fornecedores**

## **Objetivos do Sistema**

O sistema **Comprador-Fornecedores** foi desenvolvido com o objetivo de **automatizar o processo de compras entre empresas e seus fornecedores,** tornando-o mais rápido, eficiente e econômico. Entre os principais objetivos estão:

* Permitir que o comprador visualize **múltiplas ofertas** de fornecedores para um mesmo produto.
* Facilitar a escolha do **melhor fornecedor** com base em **preço e prazo de entrega.**
* Registrar de forma clara e segura os **pedidos de compra**, mantendo o histórico e integridade das informações.
* Evitar erros manuais no processo de escolha e compras.

## **Arquitetura do Sistema (MVC)**

O sistema foi implementado com base no **padrão arquitetural MVC (Model-View-Controller)**, que divide a aplicação em três camadas principais:

* **Model (Modelo):** contém as classes que representam os dados da aplicação (como `Produto`, `Fornecedor`, `Oferta`, `Pedido`, etc.).
* **View (Visão):** são as interfaces gráficas feitas com Java Swing, onde o usuário interage com o sistema.
* **Controller (Controle):** lida com a lógica do sistema, conectando os dados (Model) com as ações da interface (View).

Essa separação ajuda a manter o sistema **organizado, reutilizável** e mais fácil de manter.

## **Funcionalidades Principais**

O sistema oferece as seguintes funcionalidades:

* **Cadastro de Fornecedores:** inserir, editar, listar e excluir fornecedores.
* **Cadastro de Produtos:** gerenciamento de produtos com controle de estoque.
* **Cadastro de Ofertas:** vínculo entre produto e fornecedor, com preço e prazo de entrega.
* **Criação de Pedidos de Compra:** permite ao comprador selecionar produtos, obter a melhor oferta automaticamente, e gerar o pedido.
* **Listagem de Pedidos:** exibe os pedidos realizados, com fornecedores envolvidos e valores totais.

## **Implementação e Estrutura do Código**

O sistema está dividido em pacotes Java conforme o padrão MVC:

* **‘Modelo’**: contém as classes de domínio como `Produto`, `Fornecedor`, `Oferta`, `Pedido`.
* **‘dao’:** classes responsáveis pela **persistência no banco de dados** (`ProdutoDao`, `FornecedorDao`, `PedidoDao`, `OfertaDao`).
* **‘controle’:** manipula a lógica e regras de negócio.
* **‘gui’:** telas gráficas com Swing (como a tela de cadastro de produtos, fornecedores e pedidos).

Cada DAO é responsável por executar as operações de **inserção, edição, remoção, consulta** no banco de dados com segurança e reutilização de código.

## **Persistência de Dados (Banco de Dados MySQL)**

O sistema usa o **MySQL** como sistema de gerenciamento de banco de dados. As informações são armazenadas em tabelas bem estruturadas:

* **‘fornecedores’** (id, nome, nif, email)
* **‘produtos’** (id, nome, qtd)
* **‘ofertas’** (produto\_id, fornecedor\_id, preco, prazo\_dias)
* **‘pedidos’** (id, data, total)
* **’pedido\_itens’** (pedido\_id, oferta\_id, quantidade)

As relações entre as tabelas são bem definidas com \*\*chaves estrangeiras\*\*. A aplicação faz uso de \*\*JOINs\*\* para extrair relatórios úteis, como pedidos com os fornecedores envolvidos usando `GROUP\_CONCAT`.

**Exemplo de Fluxo de Uso**

* 1. O usuário cadastra os produtos e fornecedores.
  2. O fornecedor oferece um produto (oferta com preço e prazo).
  3. O comprador seleciona os produtos desejados.
  4. O sistema escolhe automaticamente **a melhor oferta** para cada produto.
  5. O pedido é criado e salvo no banco com todos os dados.

# **Análise Estatística com Dados Simulados**

Para enriquecer o sistema "**Comprador-Fornecedores**" com uma visão analítica, foi feita uma simulação de dados com foco na **frequência de pedidos realizados por produto** ao longo de um período.

Essa abordagem permite identificar **produtos mais requisitados**, **padrões de consumo** e **tomar decisões estratégicas**, como renegociar ofertas com fornecedores ou ajustar o estoque.  
  
**Amostra Simulada**A amostra é composta por **30 pedidos de compra** simulados, distribuídos entre 5 produtos distintos, como mostrado abaixo:

| **Produto** | **Código** |
| --- | --- |
| Papel A4 | P01 |
| Toner Impressora | P02 |
| Teclado USB | P03 |
| Cadeira Escritório | P04 |
| Monitor LCD | P05 |

## **Tabela de Frequência**

| **Produto** | **Frequência (n)** | **Frequência Relativa (%)** | |
| --- | --- | --- | --- |
| Papel A4 | 10 | 33,3% |
| Toner Impressora | 6 | 20,0% |
| Teclado USB | 5 | 16,7% |
| Cadeira Escritório | 4 | 13,3% |
| Monitor LCD | 5 | 16,7% |
| **Total** | **30** | **100%** |

## **Interpretação dos Dados**

* O **produto mais requisitado** foi o **Papel A4** , com 33,3% dos pedidos. Isso sugere um alto consumo e pode indicar a necessidade de **estoque reforçado** ou **negociação com fornecedores** .
* Os produtos **Teclado USB e Monitor LCD** possuem demanda média (16,7%), indicando regularidade nos pedidos.
* A **Cadeira de Escritório** , com apenas 13,3% de frequência, representa a menor demanda da amostra. Pode ser comprado sob demanda.

## **Benefícios da Análise Estatística**

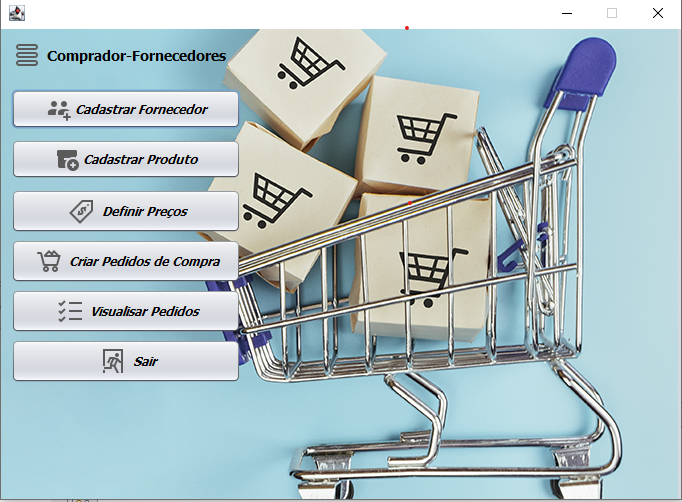
* **Identificação de padrões** de compra por tipo de produto.
* **Previsão de demanda** , otimizando estoque.
* **Apoio à decisão** na escolha de fornecedores e qualidades.
* Permite gerar **relatórios gerenciais** e escolhas específicas com base em dados.

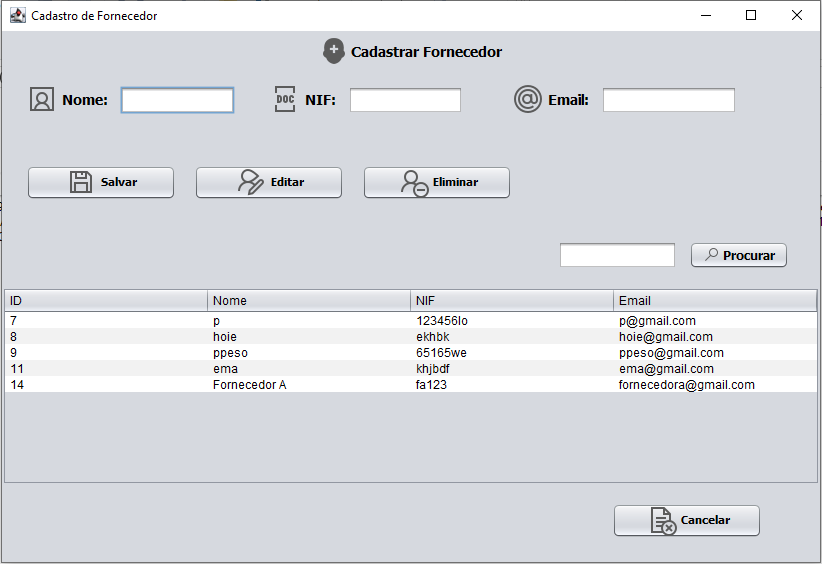
# **Estimativa de Custo do Sistema**

Com base nas funcionalidades desenvolvidas, arquitetura MVC, integração com banco de dados MySQL, lógica de seleção dos melhores fornecedores, interface gráfica Swing e documentação, o custo estimado de desenvolvimento do sistema "Comprador-Fornecedores" é de aproximadamente **$2.000 USD** .

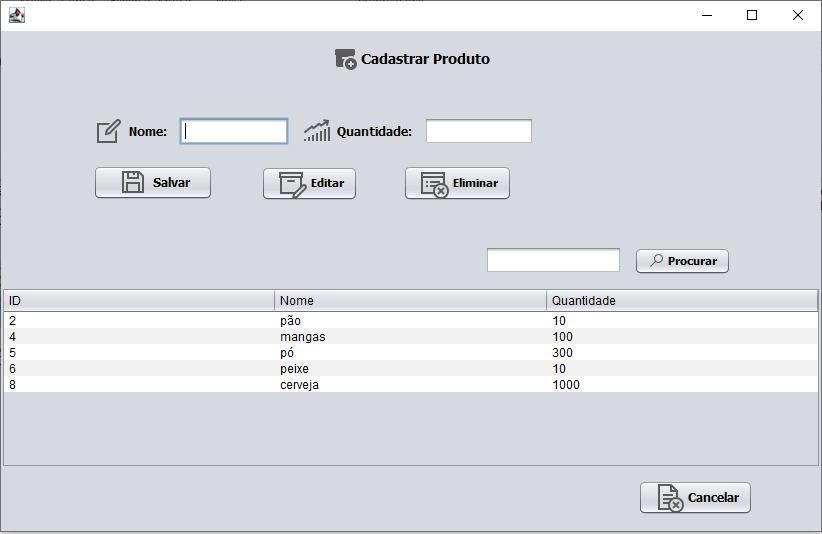
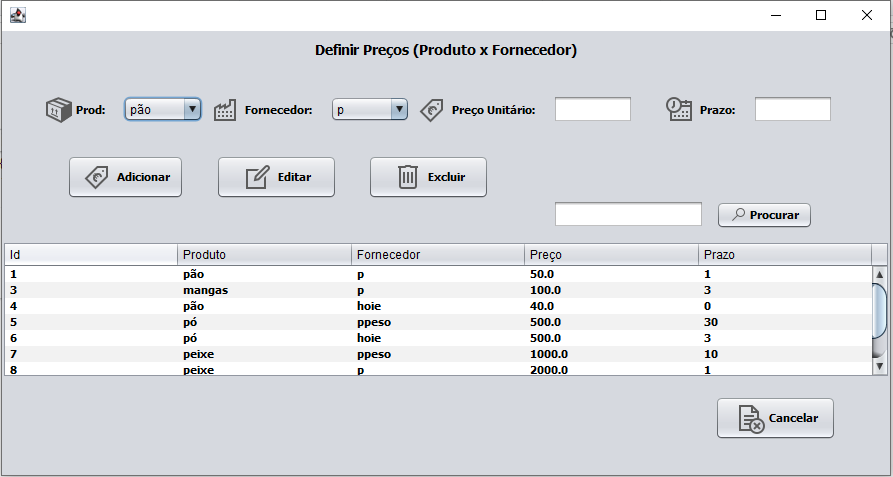
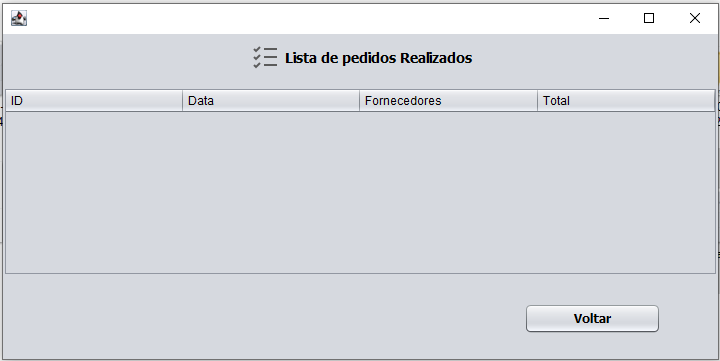
Esse valor considera cerca de **100 horas de trabalho** técnico, com uma média de **$20 por hora** , adequado para um sistema funcional, escalável e com valor de mercado real.   
Dependendo do cliente, o sistema pode ser comercializado entre **$1.500 e $2.500** , especialmente quando inclui instalação, suporte e possíveis adaptações

# **Tela de Menu Principal:**

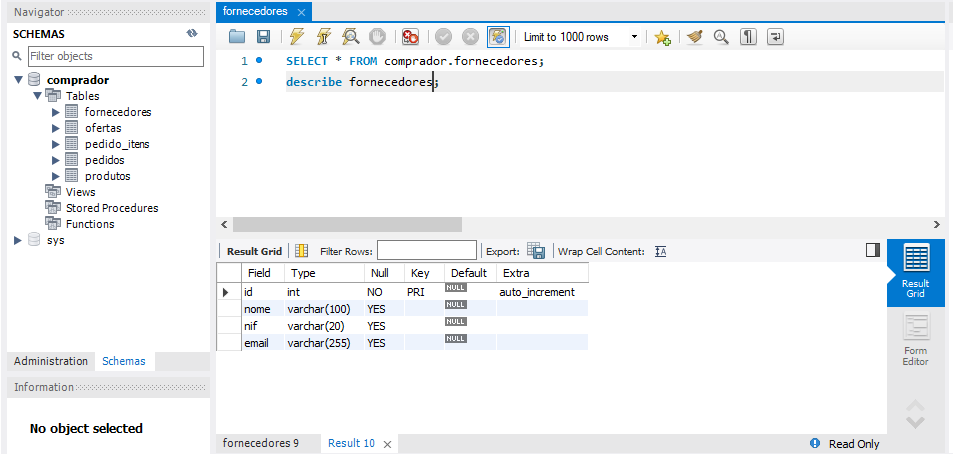
  
  
**Cadastrar Fornecedor:**



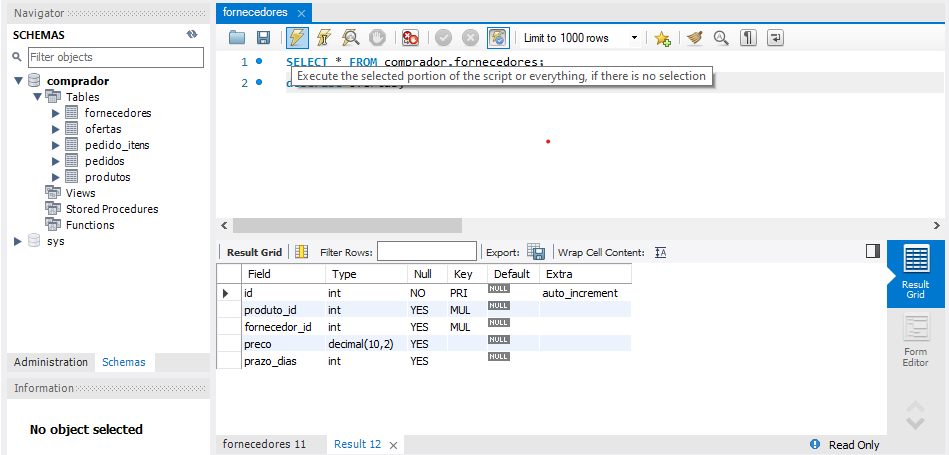
## **Cadastrar Produto:**

  
  
  
  
**Definir Preços:**  
  
  
  
  
  
  
  
  
  
  
**Criar pedidos:**  
  
  
  
**Lista de pedidos:**  
  


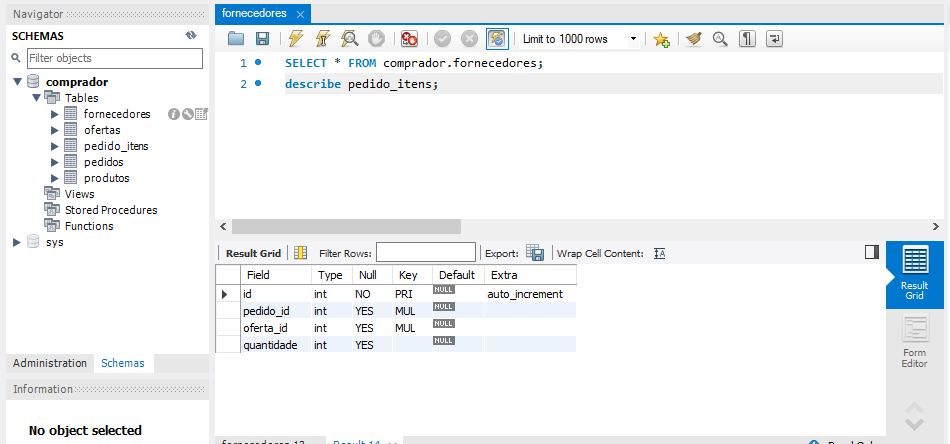
## **Tabela de Fornecedores:**



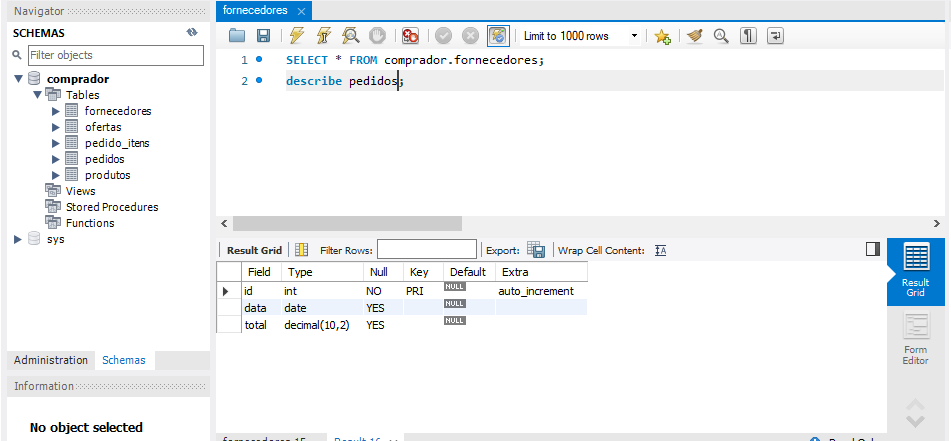
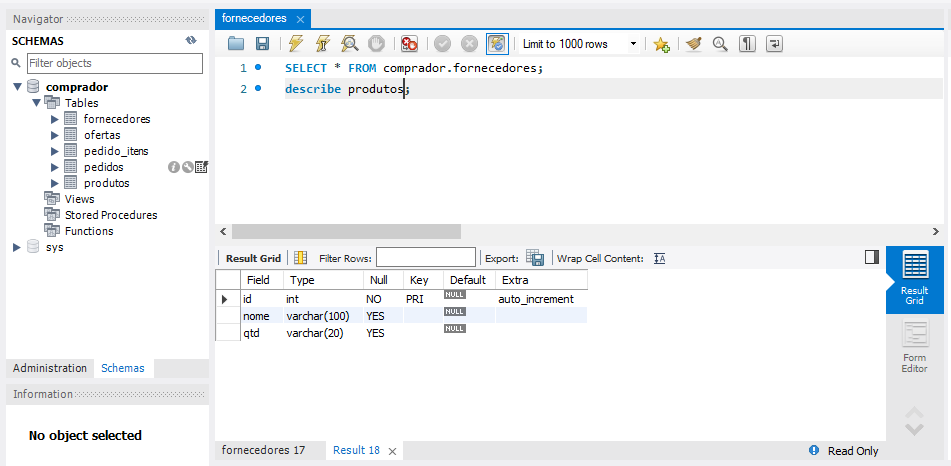
## **Tabela Ofertas:**



## **Tabela Pedidos itens:**



## **Tabela de Pedidos:**

  
  
**Tabela de produtos:**  


# **Conclusão**

O sistema Comprador-Fornecedores oferece uma solução prática e eficiente para automatizar o processo de compras, utilizando Java, MySQL e o padrão MVC para garantir organização, escalabilidade e facilidade de manutenção.

A integração de análise estatística, mesmo com dados simulados, contribuiu para identificar padrões de consumo e apoiar decisões estratégicas, como controle de estoque e seleção de fornecedores.

Com estrutura modular e código limpo, o sistema está preparado para futuras melhorias, como geração de relatórios ou integração com sistemas web.

# **Referências Bibliográficas**

* DEITEL, H.M.; DEITEL, PJ Java: Como programar . 10ª ed. São Paulo: Pearson, 2016.
* GOODRICH, MT; TAMASSIA, R.; GOLDWASSER, MH Estruturas de Dados e Algoritmos em Java . 6ª ed. Rio de Janeiro: LTC, 2014.
* DATE, CJ Introdução a Sistemas de Bancos de Dados . 8ª ed. Rio de Janeiro: LTC, 2004.
* MICROSSISTEMAS SOLARES. Plataforma Java™, Especificação de API Standard Edition 8 . Disponível em: <https://docs.oracle.com/javase/8/docs/api/> .
* MySQL. Documentação MySQL . Disponível em: <https://dev.mysql.com/doc/>
* FOWLER, M. Padrões de Arquitetura de Aplicações Corporativas . Addison-Wesley, 2002.
* GAMA, E. et al. Padrões de Projeto: Soluções Reutilizáveis ​​de Software Orientado a Objetos . Porto Alegre: Bookman, 2000.