



# GUIA DO PROJECTO

+

LOJA VIRTUAL KOMPRAKI

# INTRODUÇÃO

Este software consiste numa prototipagem e simulação de uma loja virtual denominada KOMPRAKI, Feito em PortugolStudio.

Este é um manual de acompanhamento onde estará a descrição, tecnologias, funções, fases e catalogagem de todo projecto.



# Desafios

Primeiro eu tinha que conseguir cumprir as exigências propostas pelo cliente.

Nomeadamente:

- + Cadastrar os Produtos
- + Listar os produtos
- + Detalhar os produtos
- + Eliminar os produtos
- + Editar os produtos



# Cadastrar Produtos

Para que fosse possível cadastrar os produtos, criei uma funcao `cadastrar_produto(inteiro i)`



Ela é responsável pela entrada dos produtos, recebe um valor do tipo inteiro como parâmetro, para identificar com precisão a posição atual do vetor[ ] que vai armazenar os valores se baseando na quantidade de produtos cadastrados. Essa função depende de mais 5 funções. A ideia é torna-lo mais inteligente possível, por isso ela depende de todas elas, nomeadamente:

`procurar_nome()`

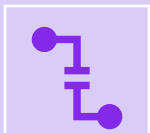
`ler_categoria_produto()`

`ler_quantidade_produto()`

`ler_preco_produto()`

`ler_descricao_produto()`

# Cadastrar Produtos/ procurar\_nome()



Para evitar redundância de nomes de produtos no sistema, criei esta função. O sistema não pode ter dois produtos com o mesmo nome se quiser mais de um basta aumentar a quantidade.



Ela funciona na base de outra função "ler\_nome\_produto( parâmetro)" ela recebe um nome como parâmetro e verifica a integridade do nome. Caso o nome não for valido, pedirá sempre que digite outro.



Uma vez que o sistema validar o nome é da responsabilidade da função procurar produto( ) verificar se já existe um produto com o mesmo nome no sistema. Caso tenha pedirá outro nome.

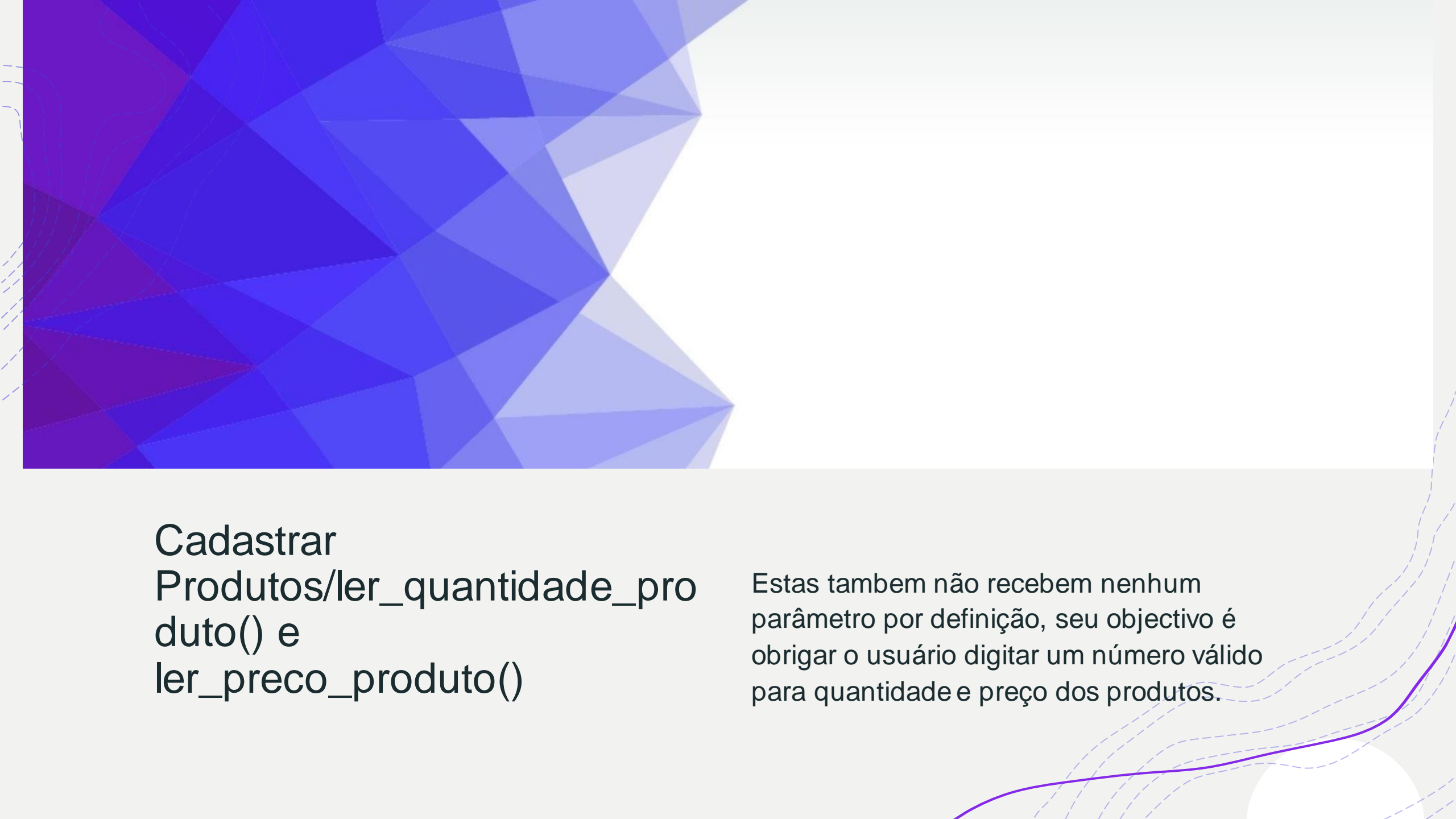
# Cadastrar

## Produtos/ler\_categoria\_produto( )

Esta não recebe nenhum parâmetro, porque atribui a categoria que o usuário escolher numa variável auxiliar(temporária).

Caso a categoria não for válida o a função pedirá sempre que o usuário digite uma válida.





Cadastrar  
Produtos/ler\_quantidade\_pro  
duto() e  
ler\_preco\_produto()

Estas também não recebem nenhum parâmetro por definição, seu objectivo é obrigar o usuário digitar um número válido para quantidade e preço dos produtos.

# Cadastrar Produtos/

ler\_descricao\_produto()

Esta funcao pede que o usuario digite uma descrição válida para o produto. Caso não fará questão de pedir sempre um válido.



As 5 funções auxiliares armazenam os dados em variáveis temporárias que depois serão verificadas.



Caso o sistema valide as variáveis temporárias, os dados poderão entrar na base de dados(vetores[ ])



Gravando também a data da entrada do produto no sistema. Caso o processo seja bem sucedido uma mensagem de aviso será emitida. Caso contrario uma mensagem de erro.

# listar\_produtos()

Uma das fases mais simples do projecto.  
Como eu precisava mostrar sempre apenas os produtos disponíveis...  
Considereei uma variável incrementadora que conta quantos produtos tem no sistema para que possa apresentar todos os produtos disponíveis. Apresentando:

- + Nome
- + ID
- + Preço

# Detalhar os produtos

Esta funcao é colaborativa tambem, opera com mais 2 funções:

- encontrar\_nome\_produto(cadeia nome, cadeia vetor[ ])
- ler\_nome\_produto(cadeia nome)

A segunda função já é conhecida, então vou abordar apenas a primeira.

Detalhar os  
produtos/encontrar\_nome\_produto(**cadeia**  
nome, **cadeia** vetor[ ])

Esta é a primeira função complexa criada neste projecto, ela recebe 2 parâmetros do tipo **cadeia** e retorna **lógico**. Ou seja ela retorna verdadeiro ou falso.


Primeiro parâmetro "**cadeia** nome" recebe algum valor do tipo **cadeia**

Segundo parâmetro " **cadeia** vetor[ ]" recebe um vetor do tipo **cadeia**.

Objectivo, ela recebe um nome no primeiro parâmetro e verifica se existe no vetor[ ] especificado no segundo parâmetro.

Se esse nome existe nesse vetor, retornará verdadeiro se não retornará falso





Após o retorno da função `encontrar_nome_produto()`, será feita a verificação.

Caso verdadeiro o retorno, o sistema apresentará os detalhes do produto.

- + Nome
- + ID
- + Categoria
- + Quantidade em Stok
- + Data de entrada no sistema

Caso falso uma mensagem de erro será exibida.





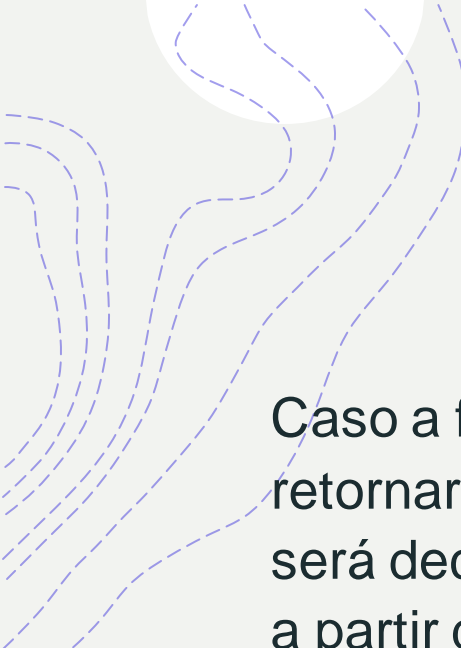
# Eliminar os produtos

Duas opções para o usuário ,

- Eliminar apenas um produto
- Eliminar todos os produtos

Nesta função também é colaborativa porque funciona na base de outras funções, caso o usuário escolha a primeira opção.

Após a leitura do ID do produto, será executada a função `encontrar_id_produtos( )`, semelhante a função `encontrar_nome_produto ( )` a única diferença é o primeiro parâmetro que em vez de pedir um nome pedirá um ID e fará as mesmas verificações no vetor especificado no segundo parâmetro.



Caso a função `encontrar_id_produtos()` retornar verdadeiro a quantidade dos produtos será decrementada e o vetor será reorganizado a partir da posição especificada.

Caso retornar falso será exibido um aviso.

Se o usuário escolher eliminar todos os produtos será exibida um aviso, caso sim, todos os produtos serão eliminados e os valores seriam resetados.

Caso não voltará ao menu.



# Editar Produtos


A última exigência do cliente, funciona primeiramente através de uma variável **lógica** "editando" que inicialmente é falso e sempre que o usuário entrar em "Editar" seu valor é alterado para **verdadeiro**. A ideia é fazer com que o sistema entenda que um produto está sendo editado, isso me ajudou a reaproveitar a **função** `cadastrar_produto( )`.

Utilizando a função paramétrica `encontrar_id_produtos( cadeia id, cadeia vetor[ ] )`, que já é conhecida para verificar se contém algum id no vetor[ ] especificado. Caso retorne verdadeiro as funções:

- + `procurar_nome( )`
- + `ler_categoria_produto( )`
- + `ler_quantidade_produto( )`
- + `ler_preco_produto( )`
- + `ler_descricao_produto( )`
- + `Cadastrar_produto( )`

São reaproveitadas.

Caso retorne falso, será exibida uma mensagem de aviso.



# Porque organizei desse jeito?

O projecto está dividido por funções, cada uma com seu objectivo e todas em geral deixar o sistema mais inteligente possível.

Desse jeito foi possível reaproveitar muitas funções que fez reduzir a enchente de redundâncias e deixar o código mais limpo possível.

O código e a documentação estão disponíveis no meu GitHub.

GitHub: [LeonardoDeveloper/UnyDevTech-Team-: Equipa UnyDevTech \(github.com\)](https://github.com/LeonardoDeveloper/UnyDevTech-Team-)