

# Documento de Desenho Arquitetural XPe

MÓDULO DESAFIO FINAL – ARQUITETO DE SOFTWARE

WILSON MISSINA FARIA - 0000139147

## Descritivo da Aplicação

---

Implementar uma aplicação REST Full utilizando GOlang para manipular dados de pedidos de usuários de um e-commerce. O esquema simplificado do banco de dados utilizado para ilustrar as interações se encontra abaixo:

Tabelas:

- **usuarios**
  - id = [integer] Id único de criação do usuário no banco
  - nome = [string] Nome do usuário
  - email = [string] Email do usuário
- **produtos**
  - id = [integer] Id única de registro do produto no banco
  - nome = [string] Nome do produto
  - valor = [float] Custo do produto em reais
  - quantidade = [integer] Quantidade em estoque
- **pedidos**
  - id = [integer] Id única da solicitação registrada em banco
  - id\_carrinho = [string] Agrupamento de produtos em uma única compra
  - id\_usuario = [integer] Referência ao ID do usuário
  - id\_produto = [integer] Referência ao ID do produto
  - qdt = [integer] Quantidade do produto inclusa no pedido

## Endpoints da Aplicação

---

A API conta com os seguintes métodos CRUD para a entidade pedidos:

Método	Rota	Função
GET	/pedidos	ListarPedidos
GET	/pedidos/{id}	BuscarPedidoPorID
GET	/pedidos/carrinho/{id_carrinho}	BuscarPorCarrinho
POST	/pedidos	CriarPedido
PUT	/pedidos/{id}	AtualizarPedido
DELETE	/pedidos/{id}	DeletarPedido

### 1. GET /pedidos – ListarPedidos

- O controlador chama `model.BuscarTodosPedidos()`.
- O model faz `SELECT ... FROM pedidos` e mapeia as linhas para `[]Pedido`.
- A resposta JSON retorna a lista completa com HTTP 200.

### 2. GET /pedidos/{id} – BuscarPedidoPorID

- O id é extraído da URL.
- Chama-se `model.BuscarPedidoPorID(id) → SELECT ... WHERE id = ?`.
- Se não houver registro, devolve 404; caso contrário, o pedido vem em JSON com 200.

### 3. GET /pedidos/carrinho/{id\_carrinho} – BuscarPorCarrinho

- Recebe o `id_carrinho`.
- Executa `model.BuscarPedidosPorCarrinho(idCarrinho) → SELECT ... WHERE id_carrinho = ?`.
- Retorna todos os itens que pertencem ao mesmo carrinho (útil para exibir o “carrinho de compras” do usuário).

### 4. POST /pedidos – CriarPedido

- O corpo da requisição (JSON) é desserializado em `model.Pedido`.
- O model executa `INSERT INTO pedidos (...) VALUES (...)`.
- Devolve 201 Created e `{ "id": <novo_id> }`.

### 5. PUT /pedidos/{id} – AtualizarPedido

- Lê tanto o id (URL) quanto o JSON atualizado.
- Chama `model.AtualizarPedido(id, pedido)` que faz `UPDATE pedidos SET ... WHERE id = ?`.
- Retorna 200 com mensagem de sucesso.

### 6. DELETE /pedidos/{id} – DeletarPedido

- Extraí o id.
- Executa `model.DeletarPedido(id) → DELETE FROM pedidos WHERE id = ?`.
- Responde 200 e confirma a exclusão.

### 7. GET /health

- Handler minimalista que devolve “OK”.
- Usado por load balancers ou plataformas de orquestração para saber se a instância está viva.

## Estrutura de Pastas

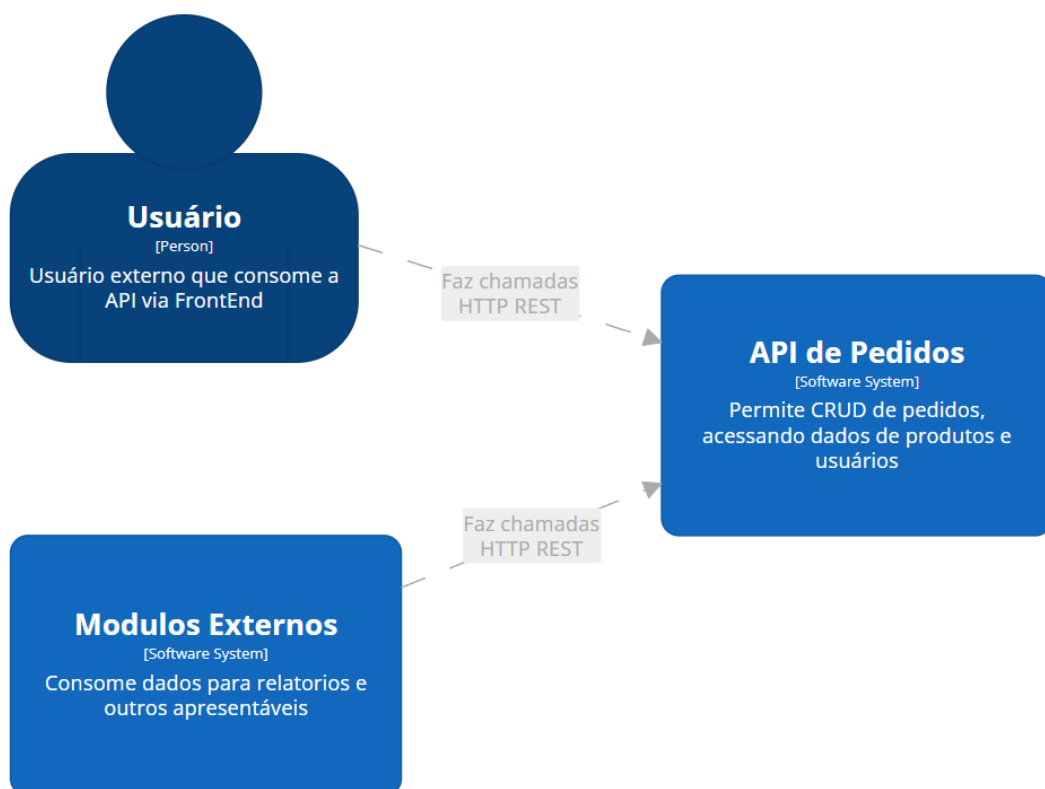
---

```
/mvc-pedidos/  
├── main.go  
├── database.sqlite          # Banco SQLite  
├── config/  
│   └── db.go              # conexão SQLite  
├── model/  
│   └── pedido.go          # struct + métodos de acesso a dados  
├── controller/  
│   └── pedido_controller.go # Mapeamento das regras de negócio  
├── view/  
│   └── response.go        # manipuladores de resposta JSON  
└── router/  
    └── routes.go          # mapeamento das rotas
```

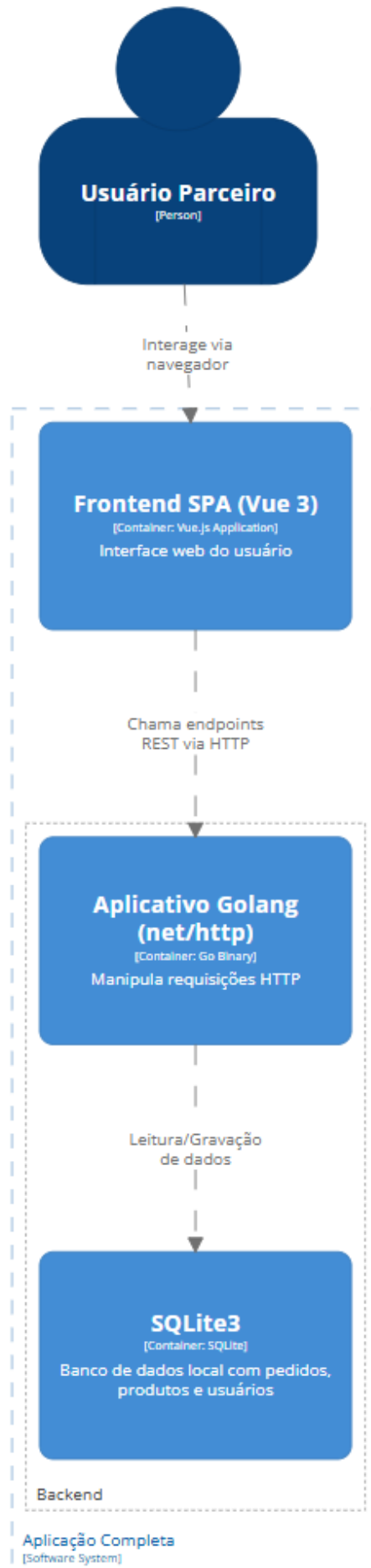
## Diagramas C4

---

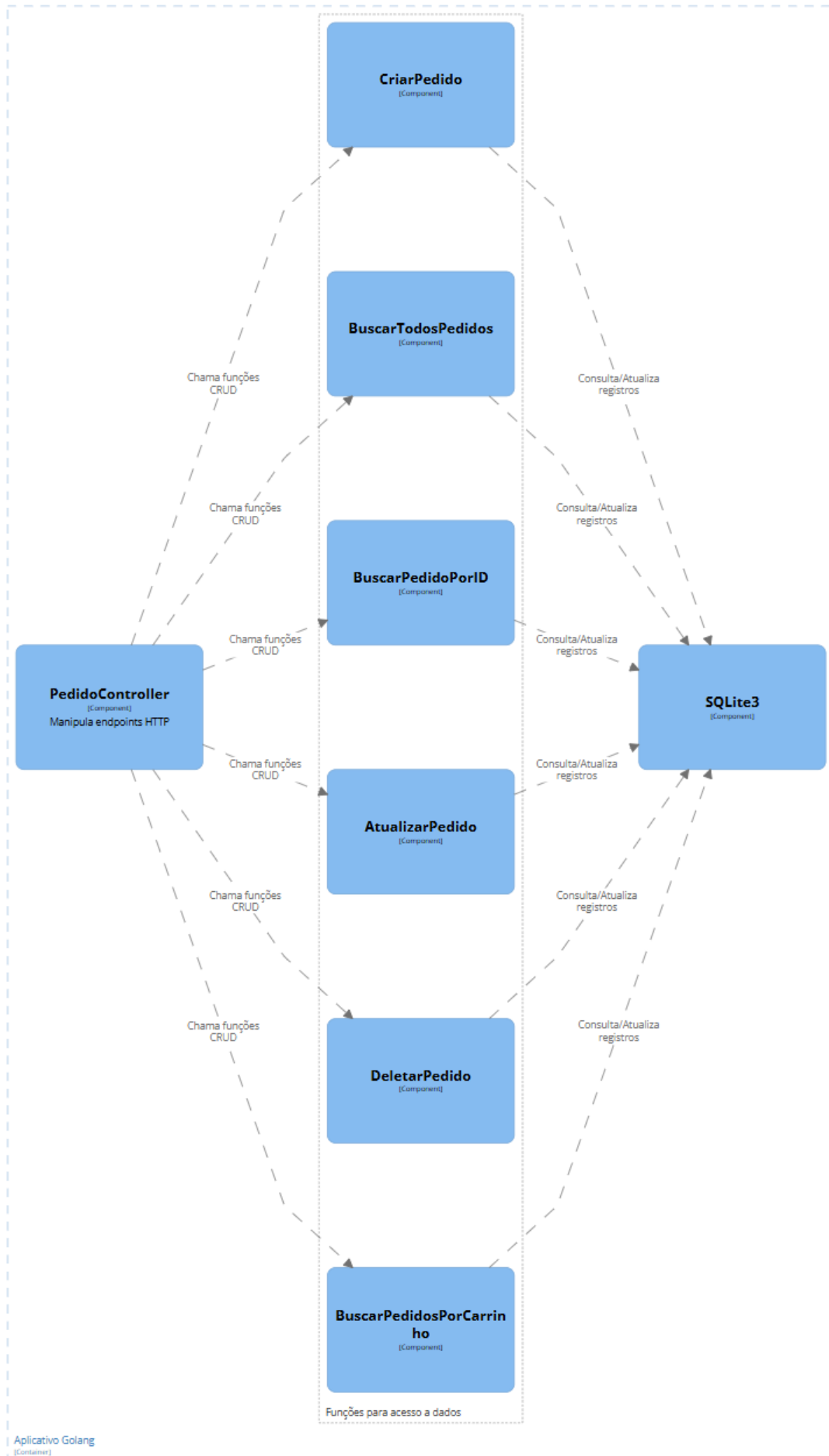
### Nível 1: Diagrama de Contexto



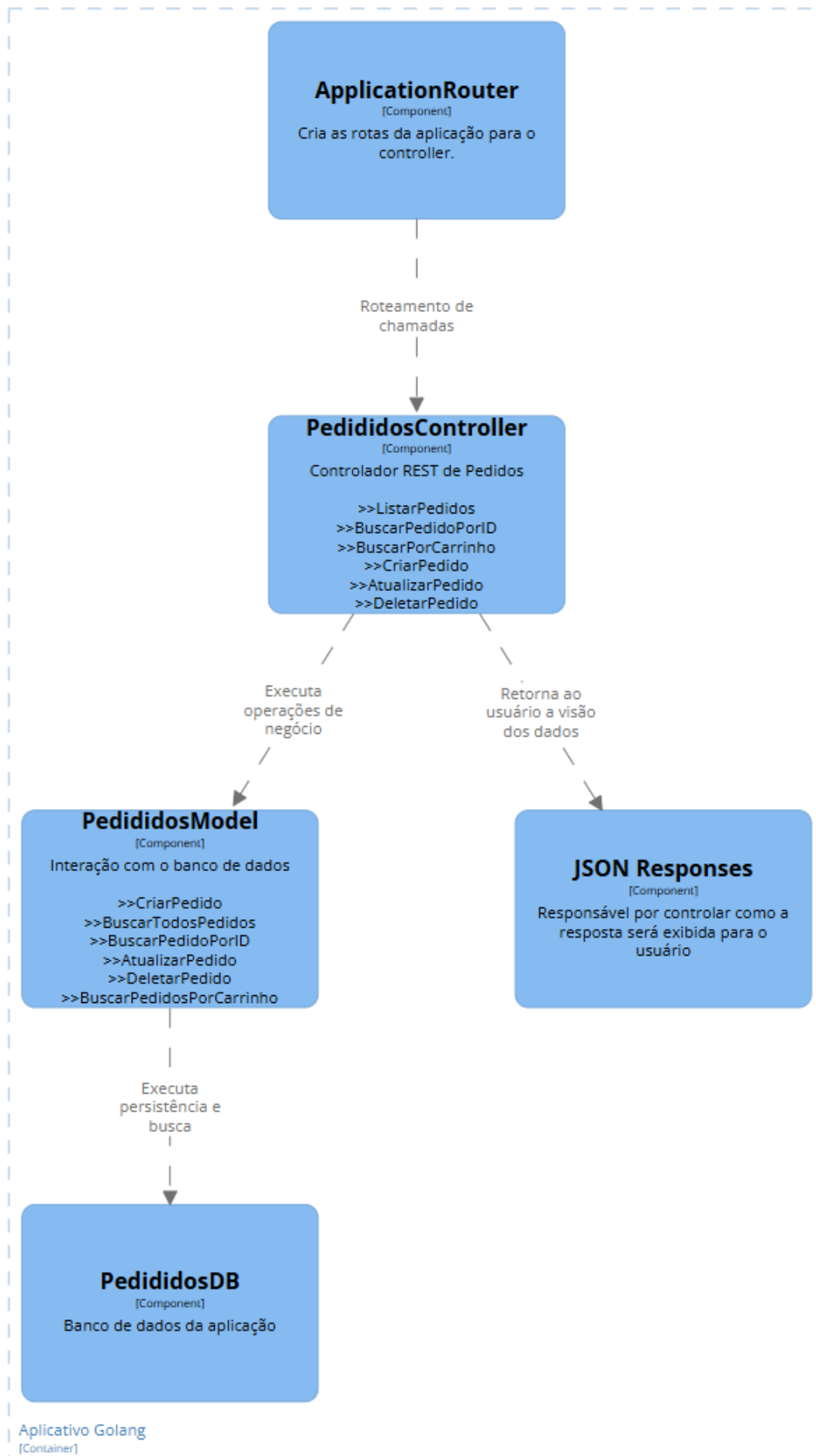
## Nível 2: Diagrama de Containers



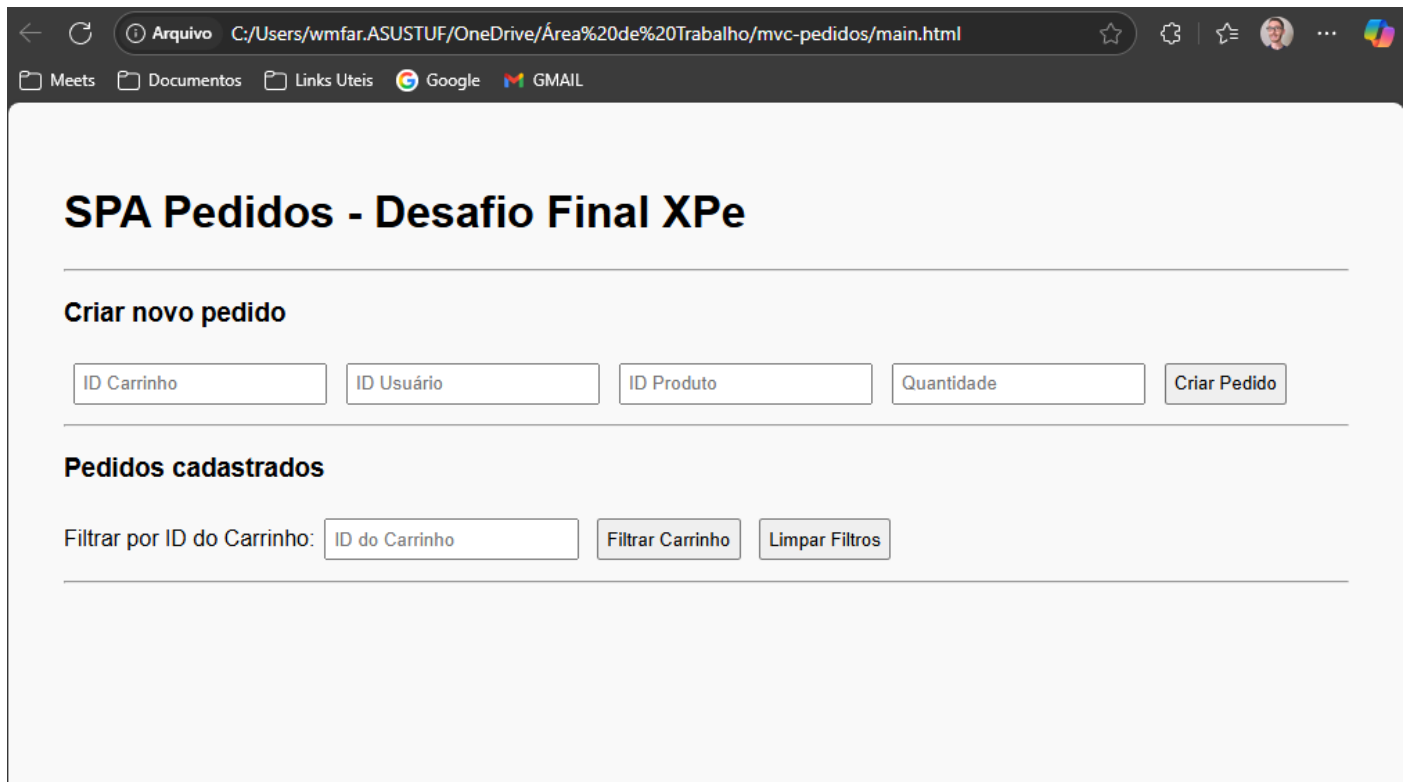
### Nível 3: Diagrama de Componentes



## Nível 4: Classes e sequências



## Playground em HTML Vue3 para testar a aplicação Go



The screenshot shows a web browser window with the address bar displaying the file path: `C:/Users/wmfar.ASUSTUF/OneDrive/Área%20de%20Trabalho/mvc-pedidos/main.html`. The browser's address bar also shows the text "Arquivo". Below the address bar, there are several icons for "Meets", "Documentos", "Links Uteis", "Google", and "GMAIL".

The main content area of the browser displays a web application titled "SPA Pedidos - Desafio Final XPe". The application has two main sections:

- Criar novo pedido**: This section contains four input fields labeled "ID Carrinho", "ID Usuário", "ID Produto", and "Quantidade", followed by a button labeled "Criar Pedido".
- Pedidos cadastrados**: This section contains a label "Filtrar por ID do Carrinho:" followed by an input field labeled "ID do Carrinho", a button labeled "Filtrar Carrinho", and a button labeled "Limpar Filtros".

Uma interface minimalista utilizando html + Vue3 foi criada para interagir com a aplicação em GO criando um ambiente "Playground" para simular as interações do usuário com a camada SPA. A camada SPA utiliza a biblioteca axios para se comunicar com o servidor REST Full feito em GO que deve estar executando em background durante a operação do SPA.

Um repositório contendo os códigos da aplicação Go juntamente com o HTML (Playground) está disponível em: [https://github.com/wilsonmfaria/crud\\_pedidosxpe](https://github.com/wilsonmfaria/crud_pedidosxpe)