Aplicação de pedidos

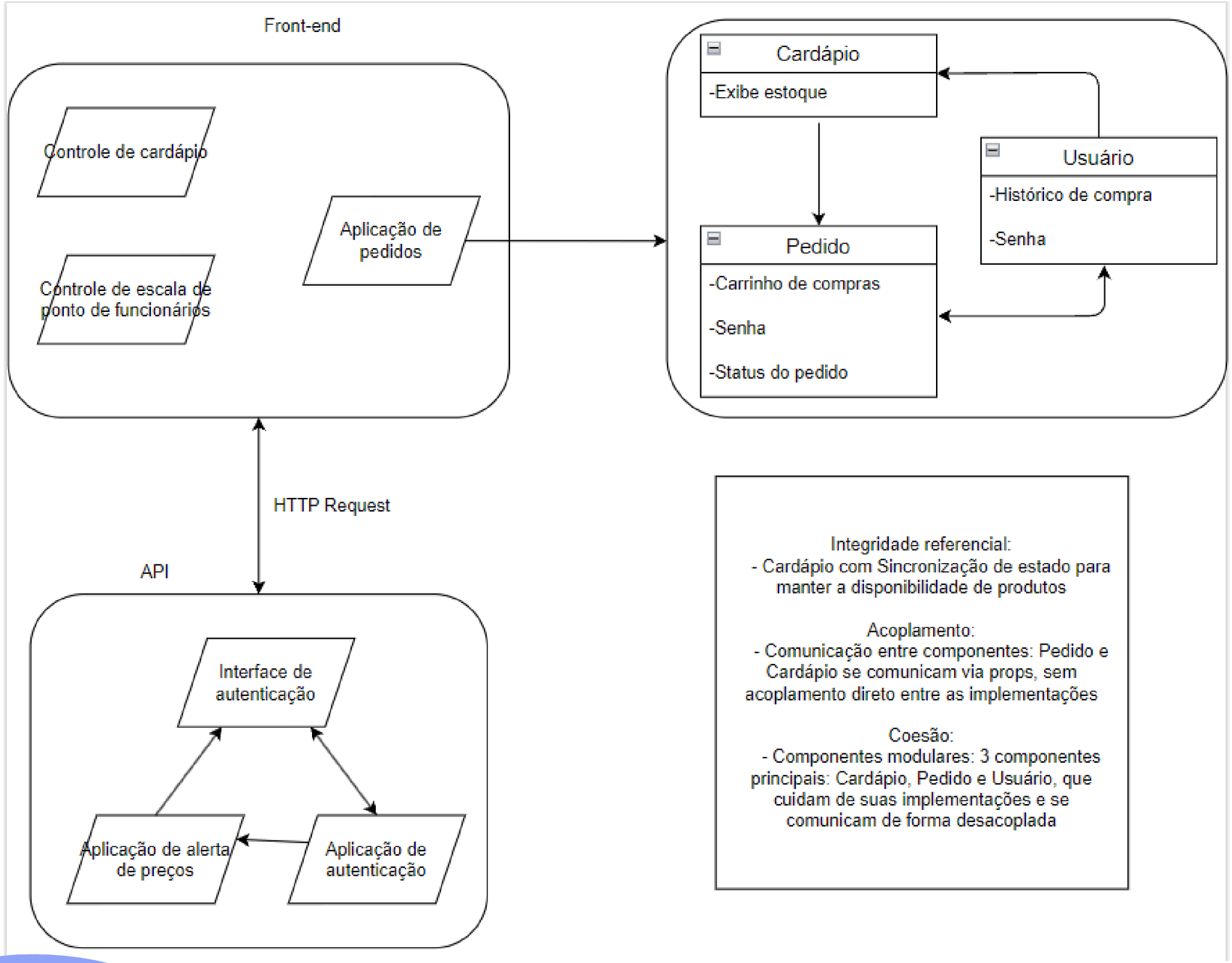
Integração entre frontend e backend

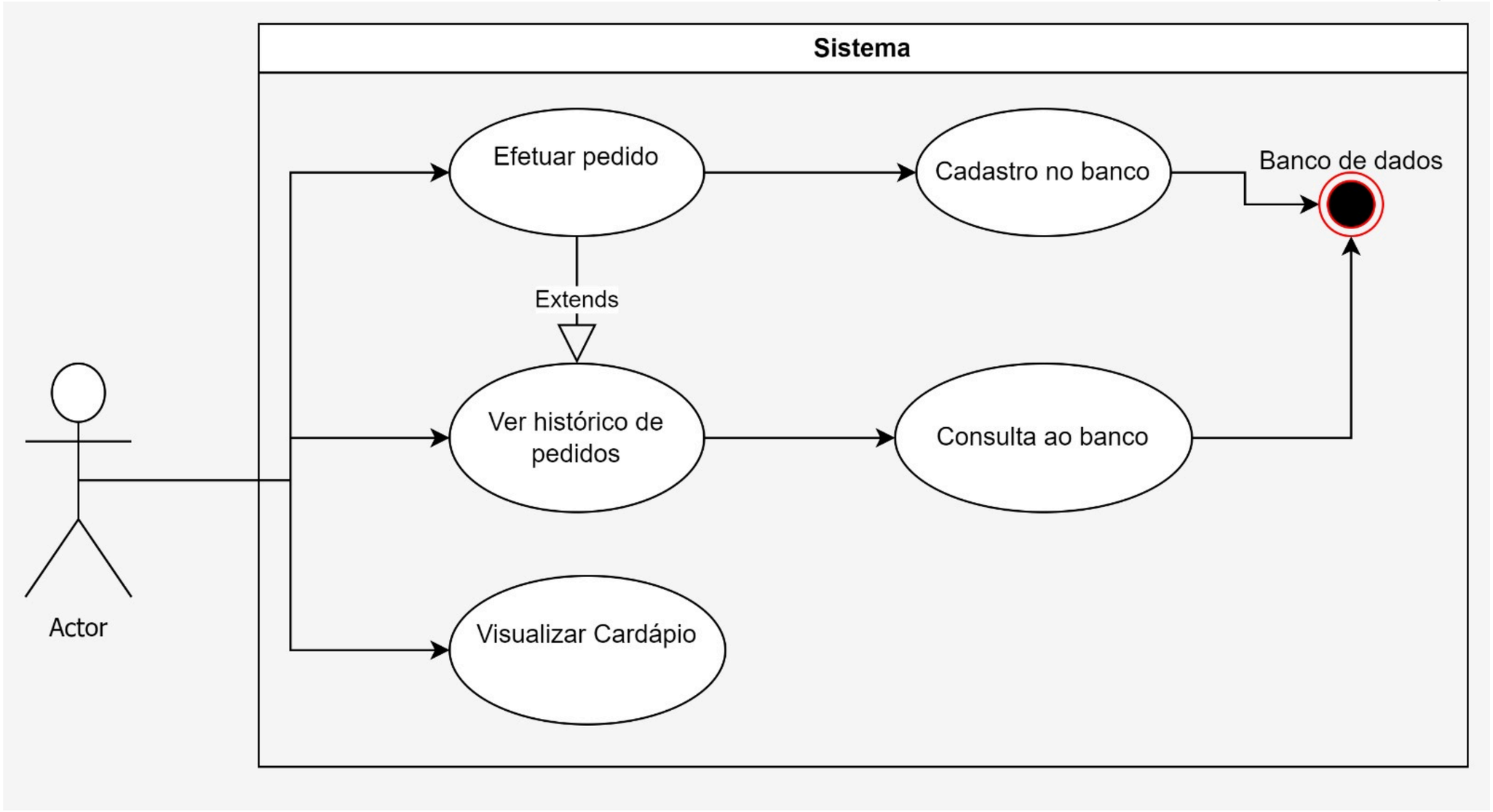
Aplicação de pedidos

front-end

Arquitetura escolhida: SPA

1. Sincronização eficiente de dados - Visando a integridade do sistema
2. Componentização - Com comunicação via props, não há acoplamento entre o pedido e o cardápio
3. Coesão - Componentes que cuidam de sua implementação de forma independente


Front-end
Controle de cardápio
Controle de escala de ponto de funcionários
Aplicação de pedidos
Cardápio
-Exibe estoque
Usuário
-Histórico de compra
-Senha
Pedido
-Carrinho de compras
-Senha
-Status do pedido
HTTP Request
API
Interface de autenticação
Aplicação de alerta de preços
Aplicação de autenticação
Integridade referencial:
- Cardápio com Sincronização de estado para manter a disponibilidade de produtos
Acoplamento:
- Comunicação entre componentes: Pedido e Cardápio se comunicam via props, sem acoplamento direto entre as implementações
Coesão:
- Componentes modulares: 3 componentes principais: Cardápio, Pedido e Usuário, que cuidam de suas implementações e se comunicam de forma desacoplada


Sistema
Efetuar pedido
Cadastro no banco
Banco de dados
Extends
Ver histórico de pedidos
Consulta ao banco
Visualizar Cardápio
Actor

Aplicação de pedidos

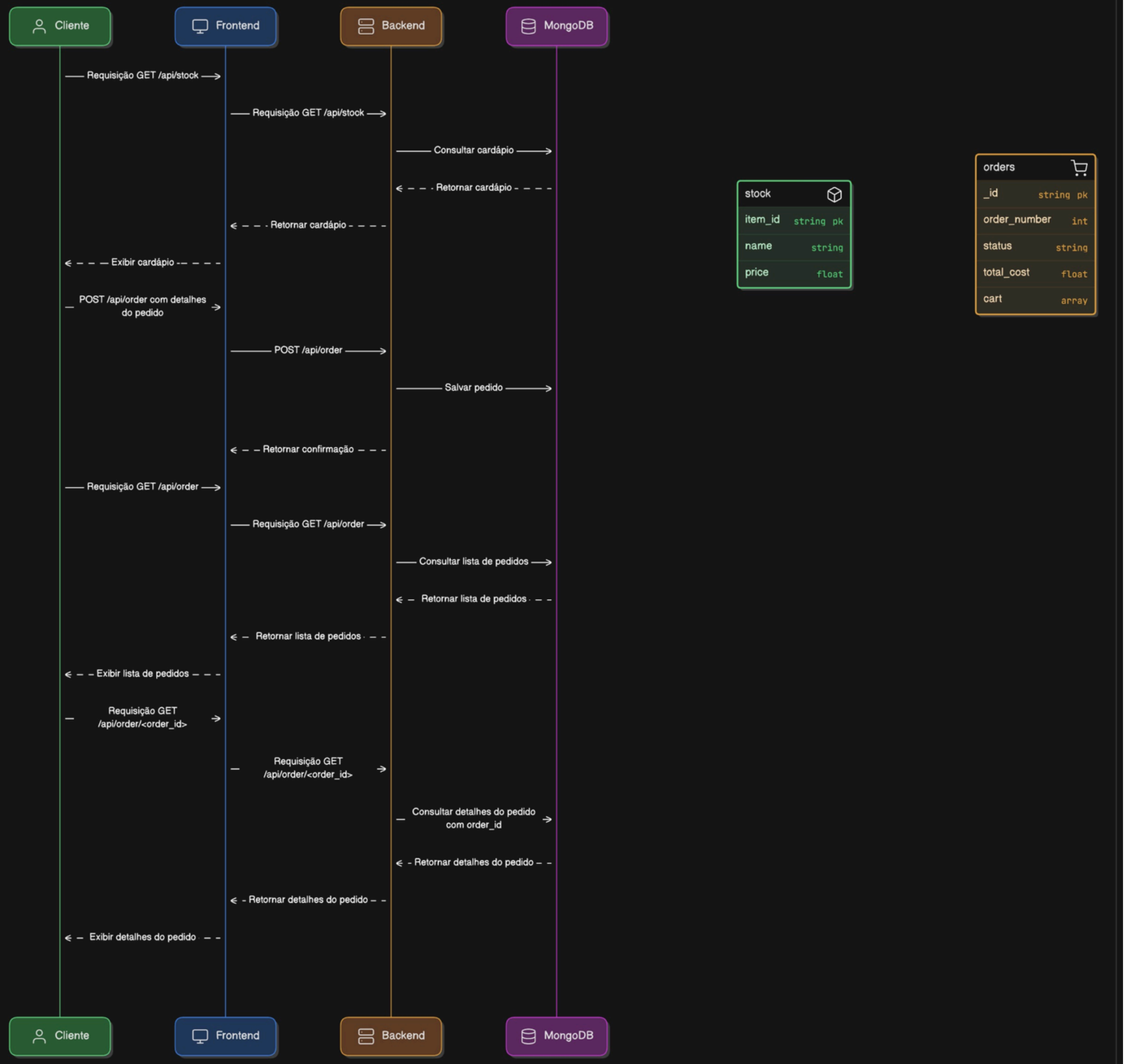
back-end

Arquitetura escolhida: SOA

1. Separação de responsabilidades - Cada camada da aplicação tem seu papel bem definido
2. Reutilização e abstração - Implementações reutilizáveis e com lógicas abstratas
3. Compatibilidade com a arquitetura SPA

Figure 1

Sistema de Pedido de Restaurante



stock	
item_id	string pk
name	string
price	float

orders	
_id	string pk
order_number	int
status	string
total_cost	float
cart	array

Estrutura do projeto

Model - Comunica com o o banco (MongoDB)

```
class OrderModel:
    @staticmethod
    def get_order():
        db = get_db()
        order_collection = db['PEDIDOS']
        return order_collection.find()
```

Service - Comunica com o Modelo para realizar a busca

```
def get_order_items():
    order = OrderModel.get_order()
    return order
```

Route - Define o endpoint a ser utilizado e comunica com a Service para buscar

```
@order_bp.route('/api/order', methods=['GET'])
def get_order():
    order_items = order_service.get_order_items()
    return dumps(order_items)
```