# ADR 002 — Estratégia de Persistência: SQL vs NoSQL

**Autor(es): Arquiteto ProntoBurger**

**Revisor(es): Arquitetos e Engenheiros**

**Status: Avaliação**

**Data: 21/09/2025**

## Contexto

A plataforma ProntoBurger possui múltiplos microsserviços com diferentes necessidades de persistência:

* **Cart/Checkout:** mantém carrinhos temporários, sessões de usuários e itens selecionados. E realiza cálculo de total, consulta estoque, catálogo e promoções, sem persistir pedidos.
* **Order Manager:** cria, atualiza e consulta pedidos confirmados, integrando pagamento, estoque, promoções e franquias.

Enquanto **Cart e Checkout** lidam com dados de curta duração e alta leitura/escrita rápida, o **Order Manager** precisa garantir **consistência forte e transações confiáveis.**

## Objetivo

Garantir persistência adequada de dados para cada serviço:

* **Cart/Checkout:** rápido acesso, flexibilidade de schema, escalabilidade horizontal.
* **Order Manager:** consistência, integridade transacional, consultas analíticas e relatórios gerenciais.

## Decisão

Adotamos **persistência híbrida:**

1. **MongoDB / Redis** para microsserviços de **Cart e Checkout**
   * Justificativa:
     + Alta performance para leitura/escrita de carrinhos temporários.
     + Flexibilidade de schema para promoções, variações de itens e cálculos dinâmicos.
     + Redis atua como cache rápido para acesso em tempo real pelo Checkout Service.
2. **AuroraDB** para **Order Manager**
   * Justificativa:
     + Garantia de consistência transacional (ACID) para pedidos e pagamentos.
     + Modelagem relacional adequada ao domínio de pedidos: clientes, itens, status, franquias.
     + Suporte a escalabilidade e alta disponibilidade via read réplicas, multi-AZ e failover automático.