Relatório de Execução de Testes – Backend ERP (Pedidos)

Visão Geral

Foi executada a suíte completa de testes automatizados do backend do módulo de **Pedidos (Orders)** utilizando **pytest** em ambiente containerizado via Docker.
Todos os testes foram concluídos com sucesso, totalizando **14 testes aprovados**, sem falhas ou erros de execução.

Ambiente identificado no runner:

- **Python:** 3.12.x
- **Framework Web:** Django 5.1.x
- **Test Runner:** pytest 8.x
- **Plugin:** pytest-django
- **Execução:** container Docker (`docker compose exec backend`)

Estrutura da Suíte de Testes

A suíte está dividida em dois grandes grupos:

1. Testes de Integração (`tests/integration/`)

Validam o comportamento completo da API, incluindo banco de dados, permissões e regras de negócio reais.

Arquivo: `test_orders_api.py`

Cobertura funcional:

- Listagem de pedidos (paginada)
- Detalhamento de pedido

- Permissões de cancelamento
- Bloqueio de criação por perfil viewer
- Idempotência de criação
- Atomicidade em falha de estoque
- Concorrência de reserva de estoque
- Alteração de status via rota alternativa (`/status`)

Objetivo principal:
Garantir que o contrato HTTP da API esteja correto e que regras críticas (estoque, status e permissões) sejam respeitadas.

Arquivo: `test_orders_integration.py`

Cobertura funcional:

- Reserva de estoque no momento da criação
- Cálculo correto do total do pedido
- Idempotência com mesma chave
- Cancelamento restaurando estoque
- Concorrência entre dois pedidos competindo pelo mesmo estoque

Objetivo principal:
Simular cenários reais de uso e validar consistência de dados sob carga e conflitos.

---

2. Testes Unitários (`tests/unit/`)

Focados na lógica de domínio isolada, sem dependência de HTTP ou banco completo.

Arquivo: `test_order_domain.py`

Cobertura funcional:

- Transições válidas de status
- Transições inválidas de status
- Códigos de erro de negócio

Objetivo principal:
Assegurar que o modelo de domínio respeite as regras internas independentemente da camada de API.

Resultados Consolidados

Categoria	Arquivo	Status
Integração API	test_orders_api.py	OK
Integração Core	test_orders_integration.py	OK
Unitário Domínio	test_order_domain.py	OK
Total	—	**14/14 OK**

Tempo médio de execução: **~20 segundos**

Conclusões Técnicas

- A API responde corretamente com paginação DRF.
- A lógica de idempotência está funcional.
- A atomicidade de estoque está preservada.
- Concorrência controlada corretamente.
- Permissões de usuário validadas.
- Regras de transição de status íntegras.
- Nenhuma regressão detectada.

A suíte demonstra **boa cobertura funcional** e **coerência de domínio**, sendo adequada para uso em pipelines de CI/CD.

Apêndice A — Comandos Pytest Utilizados

Execução Simples

```
docker compose exec backend pytest
```

Execução Verbosa (mostra cada teste)

```
docker compose exec backend pytest -vv
```

Execução com saída de prints/logs

```
docker compose exec backend pytest -s
```

Verboso + Logs

```
docker compose exec backend pytest -vv -s
```

Mostrar apenas testes de um grupo

```
docker compose exec backend pytest -m orders
```

Mostrar duração dos testes mais lentos

```
docker compose exec backend pytest --durations=10
```

Ativar logs de logging

```
docker compose exec backend pytest -o log_cli=true -o
log_cli_level=INFO
```

Execução de teste específico

```
docker compose exec backend pytest
tests/integration/test_orders_api.py::test_order_list_and_detail_fie
lds
```

Observação Final

A suíte atual é adequada para:

- Validação local de desenvolvimento
- Execução em CI/CD
- Prevenção de regressões críticas

A manutenção recomendada é apenas a expansão de cenários conforme novas features forem adicionadas ao módulo de pedidos.