

# Teste Hapvida Oracle Forms + PL/SQL

---

## Prova técnica — Dev Backend (Oracle Forms & PL/SQL)

Objetivo: construir uma **tela simples de cadastro (CRUD)** em **Oracle Forms** que utilize **procedures/functions PL/SQL** para **INSERT/UPDATE/DELETE/SELECT**, com foco em **boas práticas, transações, validações, tratamento de erros e documentação**.

Personas nas US: **Usuário, Integrador, Operador** (DBA/Suporte), **Mantenedor** (Dev), **Avaliador**.

---

## Escopo e Regras Gerais

- **Tempo sugerido:** 4–8 horas.
  - **Stack sugerida:** **Oracle Forms 12c** (ou 11g) + **Oracle Database XE** (19c/21c).
  - **Persistência:** Oracle XE
  - **Entrega:** Repositório público no **GitHub** com scripts SQL, fonte do Forms (**.fmb**) e binário (**.fmx**).
  - **Sem consumo de APIs externas.**
  - **Tratamento de erros:** usar **RAISE\_APPLICATION\_ERROR** com códigos **-20xxx** e mensagens padronizadas.
- 

## US01 — Modelagem e Objetos de Banco

**Como um Mantenedor, quero um esquema mínimo de clientes com chaves, constraints e sequência, para suportar o CRUD do Forms.**

**CA:**

- Criar tabela **TB\_CLIENTE** com, no mínimo: **ID\_CLIENTE** (PK), **NOME** (NOT NULL), **EMAIL** (UNIQUE), **CEP**, **LOGRADOURO**, **BAIRRO**, **CIDADE**, **UF** (2), **ATIVO** (DEFAULT 1), **DT\_CRIACAO** (DEFAULT SYSTIMESTAMP), **DT\_ATUALIZACAO**.
  - Criar **sequence** **SEQ\_CLIENTE** e **trigger** **TRG\_CLIENTE\_BI** (BEFORE INSERT) para popular **ID\_CLIENTE** e **DT\_CRIACAO**.
  - Constraints: **NOT NULL**, **UNIQUE(EMAIL)**, **CHECK(UF IN (...estados BR...))**.
  - Fornecer scripts **create.sql** (criação) e **drop.sql** (remoção).
- 

## US02 — Camada PL/SQL (Package de Negócio)

**Como um Integrador, quero procedures e functions para realizar o CRUD e validações, para que a UI (Forms) consuma somente a API PL/SQL.**

**CA:**

- Criar **package** **PKG\_CLIENTE** contendo:
  - **Function** **FN\_VALIDAR\_EMAIL(p\_email VARCHAR2) RETURN NUMBER** (1 válido, 0 inválido).

- Function `FN_NORMALIZAR_CEP(p_cep VARCHAR2) RETURN VARCHAR2` (somente dígitos; 8 chars).
  - Procedure `PRC_DELETAR_CLIENTE(p_id NUMBER)`.
  - Procedure `PRC_LISTAR_CLIENTES(p_nome VARCHAR2, p_email VARCHAR2, p_rc OUT SYS_REFCURSOR)` (filtros opcionais).
  - **Validações** no package (não apenas na UI):
    - **NOME** obrigatório; **EMAIL** válido; **CEP** com 8 dígitos se informado; **UF** em lista BR.
  - **Erros** via `RAISE_APPLICATION_ERROR(-20xxx, 'mensagem')` com códigos por tipo de falha.
  - **Transações:** não dar `COMMIT/ROLLBACK` dentro do package (deixar para a UI/orquestração).
- 

## US03 — Tela Oracle Forms (Cadastro de Cliente)

Como um Usuário, quero uma tela simples de cadastro que permita inserir, alterar, excluir e consultar clientes, para manter o cadastro atualizado.

CA:

- **Módulo** Forms `cliente.fmb` com:
  - **Data Block** (control block) e itens: `ID_CLIENTE` (read-only), `NOME`, `EMAIL`, `CEP`, `LOGRADOURO`, `BAIRRO`, `CIDADE`, `UF`, `ATIVO`.
  - **Botões:** Novo, Salvar, Excluir, Cancelar, Pesquisar.
  - **LOV** para `UF` (lista dos 27 estados).
- **Fluxo:**
  - **Novo:** limpa itens e posiciona para entrada.
  - **Salvar (Insert/Update):** Fazer via data block
    - `ID_CLIENTE` nulo → `PRC_INSERIR_CLIENTE` (preencher `p_id` no item).
    - `COMMIT` após sucesso; exibir mensagem amigável.
  - **Excluir:** confirmar → `PRC_DELETAR_CLIENTE` → `COMMIT`.
  - **Pesquisar:** filtros (`NOME/EMAIL`) → `PRC_LISTAR_CLIENTES` (Ref Cursor) → bloco multi-registro.
  - **Cancelar:** `ROLLBACK`.
- **Validações na UI (reforçadas no package):** `WHEN-VALIDATE-ITEM` para `EMAIL` (`FN_VALIDAR_EMAIL`), `CEP` (`FN_NORMALIZAR_CEP`) e `UF`.

NI (triggers sugeridas):

- `WHEN-BUTTON-PRESSED` (SALVAR/EXCLUIR/PESQUISAR).
- `WHEN-NEW-FORM-INSTANCE` (carregar LOV de UFs).
- `PRE-INSERT` / `PRE-UPDATE` (chamar validações do package).
- `ON-ERROR` (centralizar mensagens amigáveis).

---

## US04 — Tratamento de Erros e Logging

**Como um Operador, quero tratamento consistente de erros e logs básicos, para diagnosticar problemas em produção.**

**CA:**

- Padronizar códigos de erro: **-20001** validação de entrada, **-20002** violação de unicidade, **-20003** registro não encontrado, etc.
  - O Forms deve exibir mensagens derivadas de **SQLERRM** de forma amigável.
  - (Opcional) Criar **TB\_LOG\_ERRO** (**DT\_EVENTO**, **USUARIO**, **ORIGEM**, **MENSAGEM**) e registrar falhas críticas.
- 

## US05 — Regras e Validações de Dados

**Como um Integrador, quero validações de dados confiáveis, para garantir a integridade do cadastro.**

**CA:**

- **EMAIL** válido (regex simplificada em PL/SQL).
  - **UF** ∈ {AC, AL, AP, AM, BA, CE, DF, ES, GO, MA, MT, MS, MG, PA, PB, PR, PE, PI, RJ, RN, RS, RO, RR, SC, SP, SE, TO}.
  - **CEP** se informado → 8 dígitos.
  - **NOME** obrigatório; **ATIVO** ∈ {0,1}.
  - **UNIQUE(EMAIL)** com mensagem amigável quando violado.
- 

## US06 - Fluxo da Tela (Forms)

1. **Novo** → limpa campos → foco em **NOME**.
2. **Salvar**
  - **ID\_CLIENTE** nulo → **PRC\_INSERIR\_CLIENTE** → preencher **ID\_CLIENTE** retornado → **COMMIT**.
  - Caso contrário → **PRC\_ATUALIZAR\_CLIENTE** → **COMMIT**.
3. **Excluir** → confirmar → **PRC\_DELETAR\_CLIENTE** → **COMMIT** → limpar/atualizar tela.
4. **Pesquisar** → filtros (**NOME**, **EMAIL**) → **PRC\_LISTAR\_CLIENTES** (Ref Cursor) → bloco multi-registro.
5. **Cancelar** → **ROLLBACK**.
6. **Validações** nos itens (**WHEN-VALIDATE-ITEM**) chamando as **functions** do package.