**Faculdade de Informática e Administração Paulista**

**GLOBAL SOLUTION – WATERWISE**

**Logotipo

O conteúdo gerado por IA pode estar incorreto.**

**Projeto:** **Sistema Inteligente de Prevenção a Enchentes**

**MASTERING RELATIONAL AND NON-RELATIONAL DATABASE**

**INTEGRANTES (2TDSPS)**

Felipe Amador RM: 553528

Leonardo de Oliveira RM: 554024

Sara Sousa RM: 552656

**São Paulo**

**Junho/2025**

**SUMÁRIO**

Sumário

[**Descrição do Projeto** 3](#_Toc199757429)

[**Link para o vídeo explicativo da interface** 3](#_Toc199757430)

[**Arquitetura do Sistema(DB)** 4](#_Toc199757431)

[**Modelo de Dados e justificativa** 4](#_Toc199757432)

[**Implementação Package** 7](#_Toc199757433)

[**Integração MongoDB** 9](#_Toc199757434)

[**Código da Package** 10](#_Toc199757435)

[**Conclusão** 57](#_Toc199757436)

**WaterWise - Sistema Inteligente de Prevenção a Enchentes**

**Descrição do Projeto**

**Conceito**

O **WaterWise** é um ecossistema tecnológico inovador que previne enchentes urbanas através do monitoramento inteligente de propriedades rurais, transformando cada fazenda em uma "esponja natural" contra desastres hídricos.

**Problema Identificado**

* **Enchentes urbanas** causam bilhões em prejuízos anualmente
* **Degradação do solo rural** reduz capacidade de absorção de água
* **Falta de monitoramento** em tempo real das condições do solo
* **Ausência de sistemas integrados** entre zona rural e urbana

**Solução Proposta**

*"A enchente que alaga uma avenida pode começar com uma gota que o solo seco da zona rural não absorveu. Com WaterWise, cada metro de terra volta a ser uma esponja contra desastres."*

**Objetivos**

* **Monitorar** propriedades rurais em tempo real
* **Calcular** capacidade de absorção do solo
* **Prever** riscos de enchentes
* **Alertar** produtores e autoridades
* **Promover** práticas sustentáveis

**Link para o vídeo explicativo**

**Arquitetura do Sistema (DB)**

**Componentes Principais**

**Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.**

**Fluxo de Dados**

1. **Coleta: Sensores IoT capturam dados ambientais**
2. **Processamento: Oracle processa e calcula métricas**
3. **Análise: Algoritmos avaliam riscos e capacidades**
4. **Alertas: Sistema gera alertas automáticos**
5. **Visualização: Interface web apresenta dashboards**
6. **Logs: MongoDB armazena metadados e histórico**

**Modelo de Dados e Modelo Relacional**

**Entidades Principais**

**1. Produtor Rural**

* ID\_PRODUTOR (PK, Identity)
* NOME\_COMPLETO
* CPF\_CNPJ (Unique)
* EMAIL (Unique)
* TELEFONE
* SENHA
* DATA\_CADASTRO

**2. Propriedade Rural**

* ID\_PROPRIEDADE (PK, Identity)
* ID\_PRODUTOR (FK)
* ID\_NIVEL\_DEGRADACAO (FK)
* NOME\_PROPRIEDADE
* LATITUDE
* LONGITUDE
* AREA\_HECTARES
* DATA\_CADASTRO

**3. Sensor IoT**

* ID\_SENSOR (PK, Identity)
* ID\_PROPRIEDADE (FK)
* ID\_TIPO\_SENSOR (FK)
* MODELO\_DISPOSITIVO
* DATA\_INSTALACAO

**4. Leitura Sensor**

* ID\_LEITURA (PK, Identity)
* ID\_SENSOR (FK)
* TIMESTAMP\_LEITURA
* UMIDADE\_SOLO (0-100%)
* TEMPERATURA\_AR (-20 a 60°C)
* PRECIPITACAO\_MM (≥0)

**5. Alerta**

* ID\_ALERTA (PK, Identity)
* ID\_PRODUTOR (FK)
* ID\_LEITURA (FK)
* ID\_NIVEL\_SEVERIDADE (FK)
* TIMESTAMP\_ALERTA
* DESCRICAO\_ALERTA

**Tabelas de Apoio**

* **Tipo Sensor**: Categorização dos sensores
* **Nível Severidade**: BAIXO, MÉDIO, ALTO, CRÍTICO
* **Nível Degradação Solo**: EXCELENTE → CRÍTICO (1-5)

**Relacionamentos**

* Produtor → Propriedade (1:N)
* Propriedade → Sensor (1:N)
* Sensor → Leitura (1:N)
* Leitura → Alerta (1:N)
* Degradação Solo → Propriedade (1:N)
* Severidade → Alerta (1:N)

**Modelo Relacional**

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

**Implementação Package**

**Package Centralizada (PKG\_WATERWISE)**

A lógica do sistema está organizada em uma package Oracle com **29 procedimentos**:

**Procedures CRUD (8)**

* CRUD\_TIPO\_SENSOR
* CRUD\_NIVEL\_SEVERIDADE
* CRUD\_NIVEL\_DEGRADACAO\_SOLO
* CRUD\_PRODUTOR\_RURAL
* CRUD\_PROPRIEDADE\_RURAL
* CRUD\_SENSOR\_IOT
* CRUD\_LEITURA\_SENSOR
* CRUD\_ALERTA

**Funções de Cálculo (3)**

* CALCULAR\_RISCO\_ALAGAMENTO
* CALCULAR\_TAXA\_DEGRADACAO\_SOLO
* CALCULAR\_CAPACIDADE\_ABSORCAO

**Procedures de Análise (7)**

* ANALISAR\_ALERTAS\_DIARIOS
* VERIFICAR\_RISCO\_ENCHENTE
* STATUS\_SENSORES
* RESUMO\_DIARIO\_SISTEMA
* LISTAR\_ALERTAS\_RECENTES
* ESTADO\_GERAL\_SOLO
* PROPRIEDADES\_RISCO\_ENCHENTE

**Relatórios Executivos (7)**

* DASHBOARD\_METRICAS
* MELHORES\_PRODUTORES
* RISCO\_POR\_REGIAO
* SEVERIDADE\_ALERTAS
* MONITORAMENTO\_TEMPO\_REAL
* PRODUTIVIDADE\_POR\_REGIAO
* TENDENCIAS\_CLIMATICAS

**Utilitários (4)**

* INICIALIZAR\_SISTEMA
* VALIDAR\_INTEGRIDADE\_DADOS
* RELATORIO\_PROPRIEDADE
* BACKUP\_DADOS\_CRITICOS

**Triggers Automáticos**

1. **TRG\_ALERTA\_AUTOMATICO\_LEITURA**: Gera alertas baseados em condições críticas
2. **TRG\_VALIDAR\_DADOS\_SENSOR**: Valida dados antes da inserção
3. **TRG\_AUDITORIA\_DEGRADACAO\_SOLO**: Registra mudanças no estado do solo

**Cursores com Controle de Fluxo**

Implementação de cursores explícitos com estruturas condicionais:

CURSOR C\_ALERTAS\_RECENTES IS

SELECT a.timestamp\_alerta, ns.codigo\_severidade, prod.nome\_completo

FROM GS\_WW\_ALERTA a

JOIN GS\_WW\_NIVEL\_SEVERIDADE ns ON a.id\_nivel\_severidade = ns.id\_nivel\_severidade

WHERE a.timestamp\_alerta >= SYSDATE - 2

ORDER BY a.timestamp\_alerta DESC;

**Integração MongoDB**

**Estrutura de Documentos:**

**Logs de Atividade**

{

"\_id": ObjectId("..."),

"timestamp": ISODate("2025-06-02T10:30:00Z"),

"type": "oracle\_crud",

"user": "system",

"details": {

"entity": "PropriedadeRural",

"operation": "INSERT",

"name": "Fazenda São João"

},

"source": "streamlit\_interface"

}

**Relatórios Gerados**

{

"\_id": ObjectId("..."),

"timestamp": ISODate("2025-06-02T10:30:00Z"),

"type": "dashboard\_metrics",

"content": { */\* dados do relatório \*/* },

"metadata": {

"generated\_by": "waterwise\_system",

"format": "json",

"region": "centro-sul"

},

"status": "generated"

}

**Imagens e Metadados**

{

"\_id": ObjectId("..."),

"timestamp": ISODate("2025-06-02T10:30:00Z"),

"filename": "propriedade\_aereo\_001.jpg",

"metadata": {

"propriedade\_id": 1,

"tipo": "imagem\_aerea",

"coordenadas": [-23.5505, -46.6333]

},

"image\_data": "base64\_encoded\_string",

"size\_bytes": 2048576

}

**Código da Package**

**Estrutura de Documentos:**

CREATE OR REPLACE PACKAGE PKG\_WATERWISE\_SLIM IS

-- ============================================================================

-- 1. PROCEDURES CRUD

-- ============================================================================

PROCEDURE CRUD\_TIPO\_SENSOR(

v\_operacao IN VARCHAR2,

v\_id\_tipo\_sensor IN OUT NOCOPY GS\_WW\_TIPO\_SENSOR.id\_tipo\_sensor%TYPE,

v\_nome\_tipo IN GS\_WW\_TIPO\_SENSOR.nome\_tipo%TYPE DEFAULT NULL,

v\_descricao IN GS\_WW\_TIPO\_SENSOR.descricao%TYPE DEFAULT NULL,

v\_unidade\_medida IN GS\_WW\_TIPO\_SENSOR.unidade\_medida%TYPE DEFAULT NULL,

v\_valor\_min IN GS\_WW\_TIPO\_SENSOR.valor\_min%TYPE DEFAULT NULL,

v\_valor\_max IN GS\_WW\_TIPO\_SENSOR.valor\_max%TYPE DEFAULT NULL

);

PROCEDURE CRUD\_NIVEL\_SEVERIDADE(

v\_operacao IN VARCHAR2,

v\_id\_nivel\_severidade IN OUT NOCOPY GS\_WW\_NIVEL\_SEVERIDADE.id\_nivel\_severidade%TYPE,

v\_codigo\_severidade IN GS\_WW\_NIVEL\_SEVERIDADE.codigo\_severidade%TYPE DEFAULT NULL,

v\_descricao\_severidade IN GS\_WW\_NIVEL\_SEVERIDADE.descricao\_severidade%TYPE DEFAULT NULL,

v\_acoes\_recomendadas IN GS\_WW\_NIVEL\_SEVERIDADE.acoes\_recomendadas%TYPE DEFAULT NULL

);

PROCEDURE CRUD\_NIVEL\_DEGRADACAO\_SOLO(

v\_operacao IN VARCHAR2,

v\_id\_nivel\_degradacao IN OUT NOCOPY GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.id\_nivel\_degradacao%TYPE,

v\_codigo\_degradacao IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.codigo\_degradacao%TYPE DEFAULT NULL,

v\_descricao\_degradacao IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.descricao\_degradacao%TYPE DEFAULT NULL,

v\_nivel\_numerico IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.nivel\_numerico%TYPE DEFAULT NULL,

v\_acoes\_corretivas IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.acoes\_corretivas%TYPE DEFAULT NULL

);

PROCEDURE CRUD\_PRODUTOR\_RURAL(

v\_operacao IN VARCHAR2,

v\_id\_produtor IN OUT NOCOPY GS\_WW\_PRODUTOR\_RURAL.id\_produtor%TYPE,

v\_nome\_completo IN GS\_WW\_PRODUTOR\_RURAL.nome\_completo%TYPE DEFAULT NULL,

v\_cpf\_cnpj IN GS\_WW\_PRODUTOR\_RURAL.cpf\_cnpj%TYPE DEFAULT NULL,

v\_email IN GS\_WW\_PRODUTOR\_RURAL.email%TYPE DEFAULT NULL,

v\_telefone IN GS\_WW\_PRODUTOR\_RURAL.telefone%TYPE DEFAULT NULL,

v\_senha IN GS\_WW\_PRODUTOR\_RURAL.senha%TYPE DEFAULT NULL,

v\_data\_cadastro IN GS\_WW\_PRODUTOR\_RURAL.data\_cadastro%TYPE DEFAULT SYSDATE

);

PROCEDURE CRUD\_PROPRIEDADE\_RURAL(

v\_operacao IN VARCHAR2,

v\_id\_propriedade IN OUT NOCOPY GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE,

v\_id\_produtor IN GS\_WW\_PROPRIEDADE\_RURAL.id\_produtor%TYPE DEFAULT NULL,

v\_id\_nivel\_degradacao IN GS\_WW\_PROPRIEDADE\_RURAL.id\_nivel\_degradacao%TYPE DEFAULT NULL,

v\_nome\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.nome\_propriedade%TYPE DEFAULT NULL,

v\_latitude IN GS\_WW\_PROPRIEDADE\_RURAL.latitude%TYPE DEFAULT NULL,

v\_longitude IN GS\_WW\_PROPRIEDADE\_RURAL.longitude%TYPE DEFAULT NULL,

v\_area\_hectares IN GS\_WW\_PROPRIEDADE\_RURAL.area\_hectares%TYPE DEFAULT NULL

);

PROCEDURE CRUD\_SENSOR\_IOT(

v\_operacao IN VARCHAR2,

v\_id\_sensor IN OUT NOCOPY GS\_WW\_SENSOR\_IOT.id\_sensor%TYPE,

v\_id\_propriedade IN GS\_WW\_SENSOR\_IOT.id\_propriedade%TYPE DEFAULT NULL,

v\_id\_tipo\_sensor IN GS\_WW\_SENSOR\_IOT.id\_tipo\_sensor%TYPE DEFAULT NULL,

v\_modelo\_dispositivo IN GS\_WW\_SENSOR\_IOT.modelo\_dispositivo%TYPE DEFAULT NULL

);

PROCEDURE CRUD\_LEITURA\_SENSOR(

v\_operacao IN VARCHAR2,

v\_id\_leitura IN OUT NOCOPY GS\_WW\_LEITURA\_SENSOR.id\_leitura%TYPE,

v\_id\_sensor IN GS\_WW\_LEITURA\_SENSOR.id\_sensor%TYPE DEFAULT NULL,

v\_timestamp\_leitura IN GS\_WW\_LEITURA\_SENSOR.timestamp\_leitura%TYPE DEFAULT CURRENT\_TIMESTAMP,

v\_umidade\_solo IN GS\_WW\_LEITURA\_SENSOR.umidade\_solo%TYPE DEFAULT NULL,

v\_temperatura\_ar IN GS\_WW\_LEITURA\_SENSOR.temperatura\_ar%TYPE DEFAULT NULL,

v\_precipitacao\_mm IN GS\_WW\_LEITURA\_SENSOR.precipitacao\_mm%TYPE DEFAULT NULL

);

PROCEDURE CRUD\_ALERTA(

v\_operacao IN VARCHAR2,

v\_id\_alerta IN OUT NOCOPY GS\_WW\_ALERTA.id\_alerta%TYPE,

v\_id\_produtor IN GS\_WW\_ALERTA.id\_produtor%TYPE DEFAULT NULL,

v\_id\_leitura IN GS\_WW\_ALERTA.id\_leitura%TYPE DEFAULT NULL,

v\_id\_nivel\_severidade IN GS\_WW\_ALERTA.id\_nivel\_severidade%TYPE DEFAULT NULL,

v\_timestamp\_alerta IN GS\_WW\_ALERTA.timestamp\_alerta%TYPE DEFAULT CURRENT\_TIMESTAMP,

v\_descricao\_alerta IN GS\_WW\_ALERTA.descricao\_alerta%TYPE DEFAULT NULL

);

-- ============================================================================

-- 2. FUNÇÕES DE CÁLCULO

-- ============================================================================

FUNCTION CALCULAR\_RISCO\_ALAGAMENTO(

p\_id\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE

) RETURN VARCHAR2;

FUNCTION CALCULAR\_TAXA\_DEGRADACAO\_SOLO(

p\_id\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE

) RETURN VARCHAR2;

FUNCTION CALCULAR\_CAPACIDADE\_ABSORCAO(

p\_id\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE

) RETURN VARCHAR2;

-- ============================================================================

-- 3. PROCEDURES DE ANÁLISE E RELATÓRIOS

-- ============================================================================

PROCEDURE ANALISAR\_ALERTAS\_DIARIOS;

PROCEDURE VERIFICAR\_RISCO\_ENCHENTE(p\_id\_propriedade IN NUMBER DEFAULT NULL);

PROCEDURE ESTADO\_GERAL\_SOLO;

PROCEDURE DASHBOARD\_METRICAS;

PROCEDURE MELHORES\_PRODUTORES;

PROCEDURE RISCO\_POR\_REGIAO;

PROCEDURE RELATORIO\_PROPRIEDADE(p\_id\_propriedade IN NUMBER);

-- ============================================================================

-- 4. PROCEDURES UTILITÁRIAS

-- ============================================================================

PROCEDURE INICIALIZAR\_SISTEMA;

PROCEDURE VALIDAR\_INTEGRIDADE\_DADOS;

END PKG\_WATERWISE\_SLIM;

/

CREATE OR REPLACE PACKAGE BODY PKG\_WATERWISE\_SLIM IS

-- ============================================================================

-- 1. IMPLEMENTAÇÃO DAS PROCEDURES CRUD

-- ============================================================================

PROCEDURE CRUD\_TIPO\_SENSOR(

v\_operacao IN VARCHAR2,

v\_id\_tipo\_sensor IN OUT NOCOPY GS\_WW\_TIPO\_SENSOR.id\_tipo\_sensor%TYPE,

v\_nome\_tipo IN GS\_WW\_TIPO\_SENSOR.nome\_tipo%TYPE DEFAULT NULL,

v\_descricao IN GS\_WW\_TIPO\_SENSOR.descricao%TYPE DEFAULT NULL,

v\_unidade\_medida IN GS\_WW\_TIPO\_SENSOR.unidade\_medida%TYPE DEFAULT NULL,

v\_valor\_min IN GS\_WW\_TIPO\_SENSOR.valor\_min%TYPE DEFAULT NULL,

v\_valor\_max IN GS\_WW\_TIPO\_SENSOR.valor\_max%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_nome\_tipo IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nome do tipo de sensor é obrigatório para INSERT.');

END IF;

INSERT INTO GS\_WW\_TIPO\_SENSOR (nome\_tipo, descricao, unidade\_medida, valor\_min, valor\_max)

VALUES (v\_nome\_tipo, v\_descricao, v\_unidade\_medida, v\_valor\_min, v\_valor\_max)

RETURNING id\_tipo\_sensor INTO v\_id\_tipo\_sensor;

v\_mensagem := 'Tipo de sensor inserido com ID: ' || v\_id\_tipo\_sensor;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_tipo\_sensor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do tipo de sensor é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_TIPO\_SENSOR

SET nome\_tipo = NVL(v\_nome\_tipo, nome\_tipo),

descricao = NVL(v\_descricao, descricao),

unidade\_medida = NVL(v\_unidade\_medida, unidade\_medida),

valor\_min = NVL(v\_valor\_min, valor\_min),

valor\_max = NVL(v\_valor\_max, valor\_max)

WHERE id\_tipo\_sensor = v\_id\_tipo\_sensor;

v\_mensagem := 'Tipo de sensor atualizado com ID: ' || v\_id\_tipo\_sensor;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_tipo\_sensor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do tipo de sensor é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_TIPO\_SENSOR WHERE id\_tipo\_sensor = v\_id\_tipo\_sensor;

v\_mensagem := 'Tipo de sensor deletado com ID: ' || v\_id\_tipo\_sensor;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Tipo Sensor inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_TIPO\_SENSOR (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_TIPO\_SENSOR;

PROCEDURE CRUD\_NIVEL\_SEVERIDADE(

v\_operacao IN VARCHAR2,

v\_id\_nivel\_severidade IN OUT NOCOPY GS\_WW\_NIVEL\_SEVERIDADE.id\_nivel\_severidade%TYPE,

v\_codigo\_severidade IN GS\_WW\_NIVEL\_SEVERIDADE.codigo\_severidade%TYPE DEFAULT NULL,

v\_descricao\_severidade IN GS\_WW\_NIVEL\_SEVERIDADE.descricao\_severidade%TYPE DEFAULT NULL,

v\_acoes\_recomendadas IN GS\_WW\_NIVEL\_SEVERIDADE.acoes\_recomendadas%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_codigo\_severidade IS NULL OR v\_descricao\_severidade IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Código e Descrição da severidade são obrigatórios para INSERT.');

END IF;

INSERT INTO GS\_WW\_NIVEL\_SEVERIDADE (codigo\_severidade, descricao\_severidade, acoes\_recomendadas)

VALUES (v\_codigo\_severidade, v\_descricao\_severidade, v\_acoes\_recomendadas)

RETURNING id\_nivel\_severidade INTO v\_id\_nivel\_severidade;

v\_mensagem := 'Nível de severidade inserido com ID: ' || v\_id\_nivel\_severidade;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_nivel\_severidade IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do nível de severidade é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_NIVEL\_SEVERIDADE

SET codigo\_severidade = NVL(v\_codigo\_severidade, codigo\_severidade),

descricao\_severidade = NVL(v\_descricao\_severidade, descricao\_severidade),

acoes\_recomendadas = NVL(v\_acoes\_recomendadas, acoes\_recomendadas)

WHERE id\_nivel\_severidade = v\_id\_nivel\_severidade;

v\_mensagem := 'Nível de severidade atualizado com ID: ' || v\_id\_nivel\_severidade;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_nivel\_severidade IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do nível de severidade é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_NIVEL\_SEVERIDADE WHERE id\_nivel\_severidade = v\_id\_nivel\_severidade;

v\_mensagem := 'Nível de severidade deletado com ID: ' || v\_id\_nivel\_severidade;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Nível Severidade inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_NIVEL\_SEVERIDADE (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_NIVEL\_SEVERIDADE;

PROCEDURE CRUD\_NIVEL\_DEGRADACAO\_SOLO(

v\_operacao IN VARCHAR2,

v\_id\_nivel\_degradacao IN OUT NOCOPY GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.id\_nivel\_degradacao%TYPE,

v\_codigo\_degradacao IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.codigo\_degradacao%TYPE DEFAULT NULL,

v\_descricao\_degradacao IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.descricao\_degradacao%TYPE DEFAULT NULL,

v\_nivel\_numerico IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.nivel\_numerico%TYPE DEFAULT NULL,

v\_acoes\_corretivas IN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO.acoes\_corretivas%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_codigo\_degradacao IS NULL OR v\_descricao\_degradacao IS NULL OR v\_nivel\_numerico IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Código, Descrição e Nível Numérico da degradação são obrigatórios para INSERT.');

END IF;

INSERT INTO GS\_WW\_NIVEL\_DEGRADACAO\_SOLO (codigo\_degradacao, descricao\_degradacao, nivel\_numerico, acoes\_corretivas)

VALUES (v\_codigo\_degradacao, v\_descricao\_degradacao, v\_nivel\_numerico, v\_acoes\_corretivas)

RETURNING id\_nivel\_degradacao INTO v\_id\_nivel\_degradacao;

v\_mensagem := 'Nível de degradação inserido com ID: ' || v\_id\_nivel\_degradacao;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_nivel\_degradacao IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do nível de degradação é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_NIVEL\_DEGRADACAO\_SOLO

SET codigo\_degradacao = NVL(v\_codigo\_degradacao, codigo\_degradacao),

descricao\_degradacao = NVL(v\_descricao\_degradacao, descricao\_degradacao),

nivel\_numerico = NVL(v\_nivel\_numerico, nivel\_numerico),

acoes\_corretivas = NVL(v\_acoes\_corretivas, acoes\_corretivas)

WHERE id\_nivel\_degradacao = v\_id\_nivel\_degradacao;

v\_mensagem := 'Nível de degradação atualizado com ID: ' || v\_id\_nivel\_degradacao;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_nivel\_degradacao IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do nível de degradação é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_NIVEL\_DEGRADACAO\_SOLO WHERE id\_nivel\_degradacao = v\_id\_nivel\_degradacao;

v\_mensagem := 'Nível de degradação deletado com ID: ' || v\_id\_nivel\_degradacao;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Nível Degradação Solo inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_NIVEL\_DEGRADACAO\_SOLO (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_NIVEL\_DEGRADACAO\_SOLO;

PROCEDURE CRUD\_PRODUTOR\_RURAL(

v\_operacao IN VARCHAR2,

v\_id\_produtor IN OUT NOCOPY GS\_WW\_PRODUTOR\_RURAL.id\_produtor%TYPE,

v\_nome\_completo IN GS\_WW\_PRODUTOR\_RURAL.nome\_completo%TYPE DEFAULT NULL,

v\_cpf\_cnpj IN GS\_WW\_PRODUTOR\_RURAL.cpf\_cnpj%TYPE DEFAULT NULL,

v\_email IN GS\_WW\_PRODUTOR\_RURAL.email%TYPE DEFAULT NULL,

v\_telefone IN GS\_WW\_PRODUTOR\_RURAL.telefone%TYPE DEFAULT NULL,

v\_senha IN GS\_WW\_PRODUTOR\_RURAL.senha%TYPE DEFAULT NULL,

v\_data\_cadastro IN GS\_WW\_PRODUTOR\_RURAL.data\_cadastro%TYPE DEFAULT SYSDATE

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_nome\_completo IS NULL OR v\_cpf\_cnpj IS NULL OR v\_email IS NULL OR v\_senha IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Campos obrigatórios para Produtor (INSERT): nome, CPF/CNPJ, email e senha');

END IF;

INSERT INTO GS\_WW\_PRODUTOR\_RURAL (nome\_completo, cpf\_cnpj, email, telefone, senha, data\_cadastro)

VALUES (v\_nome\_completo, v\_cpf\_cnpj, v\_email, v\_telefone, v\_senha, NVL(v\_data\_cadastro, SYSDATE))

RETURNING id\_produtor INTO v\_id\_produtor;

v\_mensagem := 'Produtor rural inserido com ID: ' || v\_id\_produtor;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_produtor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do produtor é obrigatório para UPDATE');

END IF;

UPDATE GS\_WW\_PRODUTOR\_RURAL

SET nome\_completo = NVL(v\_nome\_completo, nome\_completo),

cpf\_cnpj = NVL(v\_cpf\_cnpj, cpf\_cnpj),

email = NVL(v\_email, email),

telefone = NVL(v\_telefone, telefone),

senha = NVL(v\_senha, senha),

data\_cadastro = NVL(v\_data\_cadastro, data\_cadastro)

WHERE id\_produtor = v\_id\_produtor;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Produtor com ID ' || v\_id\_produtor || ' não encontrado para UPDATE.');

END IF;

v\_mensagem := 'Produtor rural atualizado com ID: ' || v\_id\_produtor;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_produtor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do produtor é obrigatório para DELETE');

END IF;

DELETE FROM GS\_WW\_PRODUTOR\_RURAL WHERE id\_produtor = v\_id\_produtor;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Produtor com ID ' || v\_id\_produtor || ' não encontrado para DELETE.');

END IF;

v\_mensagem := 'Produtor rural deletado com ID: ' || v\_id\_produtor;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Produtor Rural inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_PRODUTOR\_RURAL (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_PRODUTOR\_RURAL;

PROCEDURE CRUD\_PROPRIEDADE\_RURAL(

v\_operacao IN VARCHAR2,

v\_id\_propriedade IN OUT NOCOPY GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE,

v\_id\_produtor IN GS\_WW\_PROPRIEDADE\_RURAL.id\_produtor%TYPE DEFAULT NULL,

v\_id\_nivel\_degradacao IN GS\_WW\_PROPRIEDADE\_RURAL.id\_nivel\_degradacao%TYPE DEFAULT NULL,

v\_nome\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.nome\_propriedade%TYPE DEFAULT NULL,

v\_latitude IN GS\_WW\_PROPRIEDADE\_RURAL.latitude%TYPE DEFAULT NULL,

v\_longitude IN GS\_WW\_PROPRIEDADE\_RURAL.longitude%TYPE DEFAULT NULL,

v\_area\_hectares IN GS\_WW\_PROPRIEDADE\_RURAL.area\_hectares%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_id\_produtor IS NULL OR v\_id\_nivel\_degradacao IS NULL OR v\_nome\_propriedade IS NULL OR

v\_latitude IS NULL OR v\_longitude IS NULL OR v\_area\_hectares IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Todos os campos (exceto ID da propriedade) são obrigatórios para INSERT de Propriedade.');

END IF;

INSERT INTO GS\_WW\_PROPRIEDADE\_RURAL (id\_produtor, id\_nivel\_degradacao, nome\_propriedade, latitude, longitude, area\_hectares, data\_cadastro)

VALUES (v\_id\_produtor, v\_id\_nivel\_degradacao, v\_nome\_propriedade, v\_latitude, v\_longitude, v\_area\_hectares, SYSDATE)

RETURNING id\_propriedade INTO v\_id\_propriedade;

v\_mensagem := 'Propriedade rural inserida com ID: ' || v\_id\_propriedade;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_propriedade IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID da propriedade é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_PROPRIEDADE\_RURAL

SET id\_produtor = NVL(v\_id\_produtor, id\_produtor),

id\_nivel\_degradacao = NVL(v\_id\_nivel\_degradacao, id\_nivel\_degradacao),

nome\_propriedade = NVL(v\_nome\_propriedade, nome\_propriedade),

latitude = NVL(v\_latitude, latitude),

longitude = NVL(v\_longitude, longitude),

area\_hectares = NVL(v\_area\_hectares, area\_hectares)

WHERE id\_propriedade = v\_id\_propriedade;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Propriedade com ID ' || v\_id\_propriedade || ' não encontrada para UPDATE.');

END IF;

v\_mensagem := 'Propriedade rural atualizada com ID: ' || v\_id\_propriedade;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_propriedade IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID da propriedade é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_PROPRIEDADE\_RURAL WHERE id\_propriedade = v\_id\_propriedade;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Propriedade com ID ' || v\_id\_propriedade || ' não encontrada para DELETE.');

END IF;

v\_mensagem := 'Propriedade rural deletada com ID: ' || v\_id\_propriedade;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Propriedade Rural inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_PROPRIEDADE\_RURAL (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_PROPRIEDADE\_RURAL;

PROCEDURE CRUD\_SENSOR\_IOT(

v\_operacao IN VARCHAR2,

v\_id\_sensor IN OUT NOCOPY GS\_WW\_SENSOR\_IOT.id\_sensor%TYPE,

v\_id\_propriedade IN GS\_WW\_SENSOR\_IOT.id\_propriedade%TYPE DEFAULT NULL,

v\_id\_tipo\_sensor IN GS\_WW\_SENSOR\_IOT.id\_tipo\_sensor%TYPE DEFAULT NULL,

v\_modelo\_dispositivo IN GS\_WW\_SENSOR\_IOT.modelo\_dispositivo%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_id\_propriedade IS NULL OR v\_id\_tipo\_sensor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID da Propriedade e ID do Tipo de Sensor são obrigatórios para INSERT de Sensor IoT.');

END IF;

INSERT INTO GS\_WW\_SENSOR\_IOT (id\_propriedade, id\_tipo\_sensor, modelo\_dispositivo, data\_instalacao)

VALUES (v\_id\_propriedade, v\_id\_tipo\_sensor, v\_modelo\_dispositivo, SYSDATE)

RETURNING id\_sensor INTO v\_id\_sensor;

v\_mensagem := 'Sensor IoT inserido com ID: ' || v\_id\_sensor;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_sensor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do sensor é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_SENSOR\_IOT

SET id\_propriedade = NVL(v\_id\_propriedade, id\_propriedade),

id\_tipo\_sensor = NVL(v\_id\_tipo\_sensor, id\_tipo\_sensor),

modelo\_dispositivo = NVL(v\_modelo\_dispositivo, modelo\_dispositivo)

WHERE id\_sensor = v\_id\_sensor;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Sensor com ID ' || v\_id\_sensor || ' não encontrado para UPDATE.');

END IF;

v\_mensagem := 'Sensor IoT atualizado com ID: ' || v\_id\_sensor;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_sensor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do sensor é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_SENSOR\_IOT WHERE id\_sensor = v\_id\_sensor;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Sensor com ID ' || v\_id\_sensor || ' não encontrado para DELETE.');

END IF;

v\_mensagem := 'Sensor IoT deletado com ID: ' || v\_id\_sensor;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Sensor IoT inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_SENSOR\_IOT (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_SENSOR\_IOT;

PROCEDURE CRUD\_LEITURA\_SENSOR(

v\_operacao IN VARCHAR2,

v\_id\_leitura IN OUT NOCOPY GS\_WW\_LEITURA\_SENSOR.id\_leitura%TYPE,

v\_id\_sensor IN GS\_WW\_LEITURA\_SENSOR.id\_sensor%TYPE DEFAULT NULL,

v\_timestamp\_leitura IN GS\_WW\_LEITURA\_SENSOR.timestamp\_leitura%TYPE DEFAULT CURRENT\_TIMESTAMP,

v\_umidade\_solo IN GS\_WW\_LEITURA\_SENSOR.umidade\_solo%TYPE DEFAULT NULL,

v\_temperatura\_ar IN GS\_WW\_LEITURA\_SENSOR.temperatura\_ar%TYPE DEFAULT NULL,

v\_precipitacao\_mm IN GS\_WW\_LEITURA\_SENSOR.precipitacao\_mm%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_id\_sensor IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do Sensor é obrigatório para INSERT de Leitura.');

END IF;

INSERT INTO GS\_WW\_LEITURA\_SENSOR (id\_sensor, timestamp\_leitura, umidade\_solo, temperatura\_ar, precipitacao\_mm)

VALUES (v\_id\_sensor, NVL(v\_timestamp\_leitura, CURRENT\_TIMESTAMP), v\_umidade\_solo, v\_temperatura\_ar, v\_precipitacao\_mm)

RETURNING id\_leitura INTO v\_id\_leitura;

v\_mensagem := 'Leitura de sensor inserida com ID: ' || v\_id\_leitura;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_leitura IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID da leitura é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_LEITURA\_SENSOR

SET id\_sensor = NVL(v\_id\_sensor, id\_sensor),

timestamp\_leitura = NVL(v\_timestamp\_leitura, timestamp\_leitura),

umidade\_solo = NVL(v\_umidade\_solo, umidade\_solo),

temperatura\_ar = NVL(v\_temperatura\_ar, temperatura\_ar),

precipitacao\_mm = NVL(v\_precipitacao\_mm, precipitacao\_mm)

WHERE id\_leitura = v\_id\_leitura;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Leitura com ID ' || v\_id\_leitura || ' não encontrada para UPDATE.');

END IF;

v\_mensagem := 'Leitura de sensor atualizada com ID: ' || v\_id\_leitura;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_leitura IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID da leitura é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_LEITURA\_SENSOR WHERE id\_leitura = v\_id\_leitura;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Leitura com ID ' || v\_id\_leitura || ' não encontrada para DELETE.');

END IF;

v\_mensagem := 'Leitura de sensor deletada com ID: ' || v\_id\_leitura;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Leitura Sensor inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_LEITURA\_SENSOR (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_LEITURA\_SENSOR;

PROCEDURE CRUD\_ALERTA(

v\_operacao IN VARCHAR2,

v\_id\_alerta IN OUT NOCOPY GS\_WW\_ALERTA.id\_alerta%TYPE,

v\_id\_produtor IN GS\_WW\_ALERTA.id\_produtor%TYPE DEFAULT NULL,

v\_id\_leitura IN GS\_WW\_ALERTA.id\_leitura%TYPE DEFAULT NULL,

v\_id\_nivel\_severidade IN GS\_WW\_ALERTA.id\_nivel\_severidade%TYPE DEFAULT NULL,

v\_timestamp\_alerta IN GS\_WW\_ALERTA.timestamp\_alerta%TYPE DEFAULT CURRENT\_TIMESTAMP,

v\_descricao\_alerta IN GS\_WW\_ALERTA.descricao\_alerta%TYPE DEFAULT NULL

) IS

v\_mensagem VARCHAR2(255);

BEGIN

IF UPPER(v\_operacao) = 'INSERT' THEN

IF v\_id\_produtor IS NULL OR v\_id\_nivel\_severidade IS NULL OR v\_descricao\_alerta IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do Produtor, ID Nível Severidade e Descrição são obrigatórios para INSERT de Alerta.');

END IF;

INSERT INTO GS\_WW\_ALERTA (id\_produtor, id\_leitura, id\_nivel\_severidade, timestamp\_alerta, descricao\_alerta)

VALUES (v\_id\_produtor, v\_id\_leitura, v\_id\_nivel\_severidade, NVL(v\_timestamp\_alerta, CURRENT\_TIMESTAMP), v\_descricao\_alerta)

RETURNING id\_alerta INTO v\_id\_alerta;

v\_mensagem := 'Alerta inserido com ID: ' || v\_id\_alerta;

ELSIF UPPER(v\_operacao) = 'UPDATE' THEN

IF v\_id\_alerta IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do alerta é obrigatório para UPDATE.');

END IF;

UPDATE GS\_WW\_ALERTA

SET id\_produtor = NVL(v\_id\_produtor, id\_produtor),

id\_leitura = NVL(v\_id\_leitura, id\_leitura),

id\_nivel\_severidade = NVL(v\_id\_nivel\_severidade, id\_nivel\_severidade),

timestamp\_alerta = NVL(v\_timestamp\_alerta, timestamp\_alerta),

descricao\_alerta = NVL(v\_descricao\_alerta, descricao\_alerta)

WHERE id\_alerta = v\_id\_alerta;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Alerta com ID ' || v\_id\_alerta || ' não encontrado para UPDATE.');

END IF;

v\_mensagem := 'Alerta atualizado com ID: ' || v\_id\_alerta;

ELSIF UPPER(v\_operacao) = 'DELETE' THEN

IF v\_id\_alerta IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID do alerta é obrigatório para DELETE.');

END IF;

DELETE FROM GS\_WW\_ALERTA WHERE id\_alerta = v\_id\_alerta;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Alerta com ID ' || v\_id\_alerta || ' não encontrado para DELETE.');

END IF;

v\_mensagem := 'Alerta deletado com ID: ' || v\_id\_alerta;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Operação CRUD para Alerta inválida: ' || v\_operacao);

END IF;

DBMS\_OUTPUT.PUT\_LINE(v\_mensagem);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro em CRUD\_ALERTA (' || v\_operacao || '): ' || SQLERRM);

ROLLBACK;

RAISE;

END CRUD\_ALERTA;

-- ============================================================================

-- 2. IMPLEMENTAÇÃO DAS FUNÇÕES

-- ============================================================================

FUNCTION CALCULAR\_RISCO\_ALAGAMENTO(

p\_id\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE

) RETURN VARCHAR2 IS

v\_precipitacao\_media NUMBER(8,2);

v\_umidade\_solo\_media NUMBER(5,2);

v\_nivel\_degradacao NUMBER(1);

v\_area\_hectares NUMBER(10,2);

v\_score\_risco NUMBER(5,2);

v\_nivel\_risco VARCHAR2(20);

v\_count\_leituras NUMBER;

BEGIN

SELECT pr.area\_hectares, nd.nivel\_numerico

INTO v\_area\_hectares, v\_nivel\_degradacao

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

WHERE pr.id\_propriedade = p\_id\_propriedade;

SELECT AVG(ls.precipitacao\_mm), AVG(ls.umidade\_solo), COUNT(\*)

INTO v\_precipitacao\_media, v\_umidade\_solo\_media, v\_count\_leituras

FROM GS\_WW\_LEITURA\_SENSOR ls

JOIN GS\_WW\_SENSOR\_IOT si ON ls.id\_sensor = si.id\_sensor

WHERE si.id\_propriedade = p\_id\_propriedade

AND ls.timestamp\_leitura >= SYSDATE - 1;

IF v\_count\_leituras = 0 OR v\_precipitacao\_media IS NULL THEN

RETURN 'INDETERMINADO - Dados insuficientes';

END IF;

v\_score\_risco := 0;

IF v\_precipitacao\_media > 50 THEN v\_score\_risco := v\_score\_risco + 40;

ELSIF v\_precipitacao\_media > 25 THEN v\_score\_risco := v\_score\_risco + 25;

ELSIF v\_precipitacao\_media > 10 THEN v\_score\_risco := v\_score\_risco + 15;

ELSE v\_score\_risco := v\_score\_risco + 5; END IF;

v\_score\_risco := v\_score\_risco + (v\_nivel\_degradacao \* 6);

IF v\_umidade\_solo\_media > 80 THEN v\_score\_risco := v\_score\_risco + 20;

ELSIF v\_umidade\_solo\_media > 60 THEN v\_score\_risco := v\_score\_risco + 15;

ELSIF v\_umidade\_solo\_media > 40 THEN v\_score\_risco := v\_score\_risco + 10;

ELSE v\_score\_risco := v\_score\_risco + 5; END IF;

IF v\_area\_hectares < 50 THEN v\_score\_risco := v\_score\_risco + 10;

ELSIF v\_area\_hectares < 150 THEN v\_score\_risco := v\_score\_risco + 6;

ELSE v\_score\_risco := v\_score\_risco + 3; END IF;

IF v\_score\_risco >= 80 THEN v\_nivel\_risco := 'CRÍTICO';

ELSIF v\_score\_risco >= 60 THEN v\_nivel\_risco := 'ALTO';

ELSIF v\_score\_risco >= 40 THEN v\_nivel\_risco := 'MÉDIO';

ELSIF v\_score\_risco >= 20 THEN v\_nivel\_risco := 'BAIXO';

ELSE v\_nivel\_risco := 'MÍNIMO'; END IF;

RETURN v\_nivel\_risco || ' (' || ROUND(v\_score\_risco, 1) || '%)';

EXCEPTION WHEN NO\_DATA\_FOUND THEN RETURN 'ERRO - Propriedade não encontrada'; WHEN OTHERS THEN RETURN 'ERRO - ' || SQLERRM;

END CALCULAR\_RISCO\_ALAGAMENTO;

FUNCTION CALCULAR\_TAXA\_DEGRADACAO\_SOLO(

p\_id\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE

) RETURN VARCHAR2 IS

v\_nivel\_atual NUMBER(1); v\_umidade\_media NUMBER(5,2);

v\_temperatura\_media NUMBER(4,1); v\_precipitacao\_total NUMBER(8,2);

v\_dias\_monitoramento NUMBER; v\_taxa\_degradacao NUMBER(8,4);

v\_tendencia VARCHAR2(50); v\_classificacao VARCHAR2(100);

BEGIN

SELECT nd.nivel\_numerico INTO v\_nivel\_atual

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

WHERE pr.id\_propriedade = p\_id\_propriedade;

SELECT AVG(ls.umidade\_solo), AVG(ls.temperatura\_ar), SUM(ls.precipitacao\_mm),

ROUND(SYSDATE - MIN(CAST(ls.timestamp\_leitura AS DATE)))

INTO v\_umidade\_media, v\_temperatura\_media, v\_precipitacao\_total, v\_dias\_monitoramento

FROM GS\_WW\_LEITURA\_SENSOR ls

JOIN GS\_WW\_SENSOR\_IOT si ON ls.id\_sensor = si.id\_sensor

WHERE si.id\_propriedade = p\_id\_propriedade AND ls.timestamp\_leitura >= SYSDATE - 30;

IF v\_dias\_monitoramento < 7 OR v\_umidade\_media IS NULL THEN RETURN 'Dados insuficientes - Mínimo 7 dias de monitoramento'; END IF;

CASE v\_nivel\_atual

WHEN 1 THEN v\_taxa\_degradacao := 0.1; WHEN 2 THEN v\_taxa\_degradacao := 0.3;

WHEN 3 THEN v\_taxa\_degradacao := 0.6; WHEN 4 THEN v\_taxa\_degradacao := 1.2;

WHEN 5 THEN v\_taxa\_degradacao := 2.5; ELSE v\_taxa\_degradacao := 0.5; END CASE;

IF v\_umidade\_media < 20 OR v\_umidade\_media > 85 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 1.5;

ELSIF v\_umidade\_media < 30 OR v\_umidade\_media > 75 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 1.2;

ELSIF v\_umidade\_media BETWEEN 40 AND 60 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 0.8; END IF;

IF v\_temperatura\_media > 35 OR v\_temperatura\_media < 5 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 1.4;

ELSIF v\_temperatura\_media > 30 OR v\_temperatura\_media < 10 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 1.1; END IF;

IF v\_precipitacao\_total > 200 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 1.3;

ELSIF v\_precipitacao\_total < 30 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 1.4;

ELSIF v\_precipitacao\_total BETWEEN 60 AND 120 THEN v\_taxa\_degradacao := v\_taxa\_degradacao \* 0.9; END IF;

IF v\_taxa\_degradacao <= 0.2 THEN v\_tendencia := 'ESTÁVEL/MELHORIA';

ELSIF v\_taxa\_degradacao <= 0.5 THEN v\_tendencia := 'DEGRADAÇÃO LENTA';

ELSIF v\_taxa\_degradacao <= 1.0 THEN v\_tendencia := 'DEGRADAÇÃO MODERADA';

ELSIF v\_taxa\_degradacao <= 2.0 THEN v\_tendencia := 'DEGRADAÇÃO ACELERADA';

ELSE v\_tendencia := 'DEGRADAÇÃO CRÍTICA'; END IF;

v\_classificacao := v\_tendencia || ' - ' || ROUND(v\_taxa\_degradacao, 2) || '%/mês';

RETURN v\_classificacao;

EXCEPTION WHEN NO\_DATA\_FOUND THEN RETURN 'ERRO - Propriedade não encontrada'; WHEN OTHERS THEN RETURN 'ERRO - ' || SQLERRM;

END CALCULAR\_TAXA\_DEGRADACAO\_SOLO;

FUNCTION CALCULAR\_CAPACIDADE\_ABSORCAO(

p\_id\_propriedade IN GS\_WW\_PROPRIEDADE\_RURAL.id\_propriedade%TYPE

) RETURN VARCHAR2 IS

v\_area\_hectares NUMBER(10,2); v\_nivel\_degradacao NUMBER(1);

v\_umidade\_atual NUMBER(5,2); v\_precipitacao\_recente NUMBER(8,2);

v\_capacidade\_base NUMBER(10,2); v\_capacidade\_atual NUMBER(10,2);

v\_reducao\_percentual NUMBER(5,2); v\_status\_absorcao VARCHAR2(100);

v\_count\_sensores NUMBER;

BEGIN

SELECT pr.area\_hectares, nd.nivel\_numerico INTO v\_area\_hectares, v\_nivel\_degradacao

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

WHERE pr.id\_propriedade = p\_id\_propriedade;

SELECT AVG(ls.umidade\_solo), SUM(ls.precipitacao\_mm), COUNT(DISTINCT si.id\_sensor)

INTO v\_umidade\_atual, v\_precipitacao\_recente, v\_count\_sensores

FROM GS\_WW\_LEITURA\_SENSOR ls

JOIN GS\_WW\_SENSOR\_IOT si ON ls.id\_sensor = si.id\_sensor

WHERE si.id\_propriedade = p\_id\_propriedade AND ls.timestamp\_leitura >= SYSDATE - 0.25;

IF v\_count\_sensores = 0 OR v\_umidade\_atual IS NULL THEN RETURN 'SEM DADOS - Sensores não detectados'; END IF;

CASE v\_nivel\_degradacao

WHEN 1 THEN v\_capacidade\_base := 12000; WHEN 2 THEN v\_capacidade\_base := 9500;

WHEN 3 THEN v\_capacidade\_base := 7000; WHEN 4 THEN v\_capacidade\_base := 4500;

WHEN 5 THEN v\_capacidade\_base := 3000; ELSE v\_capacidade\_base := 6000; END CASE;

IF v\_umidade\_atual >= 90 THEN v\_reducao\_percentual := 95;

ELSIF v\_umidade\_atual >= 80 THEN v\_reducao\_percentual := 75;

ELSIF v\_umidade\_atual >= 70 THEN v\_reducao\_percentual := 50;

ELSIF v\_umidade\_atual >= 60 THEN v\_reducao\_percentual := 30;

ELSIF v\_umidade\_atual >= 50 THEN v\_reducao\_percentual := 15;

ELSIF v\_umidade\_atual >= 40 THEN v\_reducao\_percentual := 5;

ELSE v\_reducao\_percentual := 0; END IF;

v\_capacidade\_atual := v\_capacidade\_base \* (100 - v\_reducao\_percentual) / 100;

v\_capacidade\_atual := v\_capacidade\_atual \* v\_area\_hectares;

IF v\_reducao\_percentual >= 90 THEN v\_status\_absorcao := 'SATURADO - Risco Alto de Alagamento';

ELSIF v\_reducao\_percentual >= 70 THEN v\_status\_absorcao := 'CAPACIDADE CRÍTICA - Monitoramento Urgente';

ELSIF v\_reducao\_percentual >= 50 THEN v\_status\_absorcao := 'CAPACIDADE REDUZIDA - Atenção Necessária';

ELSIF v\_reducao\_percentual >= 25 THEN v\_status\_absorcao := 'CAPACIDADE BOA - Funcionamento Normal';

ELSE v\_status\_absorcao := 'CAPACIDADE EXCELENTE - Esponja Natural Ativa'; END IF;

RETURN v\_status\_absorcao || ' - ' || ROUND(v\_capacidade\_atual/1000, 1) || 'k litros disponíveis (' || (100 - v\_reducao\_percentual) || '% da capacidade)';

EXCEPTION WHEN NO\_DATA\_FOUND THEN RETURN 'ERRO - Propriedade não encontrada'; WHEN OTHERS THEN RETURN 'ERRO - ' || SQLERRM;

END CALCULAR\_CAPACIDADE\_ABSORCAO;

-- ============================================================================

-- 3. IMPLEMENTAÇÃO DAS PROCEDURES DE ANÁLISE E RELATÓRIOS

-- ============================================================================

PROCEDURE ANALISAR\_ALERTAS\_DIARIOS IS

v\_alertas\_hoje NUMBER; v\_alertas\_criticos NUMBER;

v\_alertas\_automaticos NUMBER; v\_primeiro\_alerta TIMESTAMP;

v\_ultimo\_alerta TIMESTAMP; v\_situacao\_dia VARCHAR2(50);

v\_recomendacao VARCHAR2(200);

BEGIN

SELECT COUNT(\*), COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END),

COUNT(CASE WHEN a.descricao\_alerta LIKE 'ALERTA AUTOMÁTICO:%' THEN 1 END),

MIN(a.timestamp\_alerta), MAX(a.timestamp\_alerta)

INTO v\_alertas\_hoje, v\_alertas\_criticos, v\_alertas\_automaticos, v\_primeiro\_alerta, v\_ultimo\_alerta

FROM GS\_WW\_ALERTA a

LEFT JOIN GS\_WW\_NIVEL\_SEVERIDADE ns ON a.id\_nivel\_severidade = ns.id\_nivel\_severidade

WHERE a.timestamp\_alerta >= TRUNC(SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('=== ANÁLISE DE ALERTAS DE HOJE ===');

DBMS\_OUTPUT.PUT\_LINE('Data: ' || TO\_CHAR(SYSDATE, 'DD/MM/YYYY')); DBMS\_OUTPUT.PUT\_LINE(' ');

IF v\_alertas\_hoje = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('✅ DIA TRANQUILO: Nenhum alerta hoje');

DBMS\_OUTPUT.PUT\_LINE('Sistema funcionando normalmente');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Total de Alertas: ' || v\_alertas\_hoje);

DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos: ' || v\_alertas\_criticos);

DBMS\_OUTPUT.PUT\_LINE('Alertas Automáticos: ' || v\_alertas\_automaticos);

DBMS\_OUTPUT.PUT\_LINE('Primeiro Alerta: ' || TO\_CHAR(v\_primeiro\_alerta, 'HH24:MI'));

DBMS\_OUTPUT.PUT\_LINE('Último Alerta: ' || TO\_CHAR(v\_ultimo\_alerta, 'HH24:MI')); DBMS\_OUTPUT.PUT\_LINE(' ');

IF v\_alertas\_criticos >= 5 THEN v\_situacao\_dia := '🚨 DIA DE EMERGÊNCIA'; v\_recomendacao := 'Ativar protocolo de emergência geral!';

ELSIF v\_alertas\_criticos >= 2 THEN v\_situacao\_dia := '⚠️ DIA CRÍTICO'; v\_recomendacao := 'Monitoramento intensivo necessário';

ELSIF v\_alertas\_hoje >= 10 THEN v\_situacao\_dia := '🟨 DIA AGITADO'; v\_recomendacao := 'Verificar causas dos múltiplos alertas';

ELSIF v\_alertas\_automaticos = v\_alertas\_hoje THEN v\_situacao\_dia := '🤖 DIA AUTOMATIZADO'; v\_recomendacao := 'Sistema inteligente funcionando bem';

ELSE v\_situacao\_dia := '📊 DIA NORMAL'; v\_recomendacao := 'Acompanhar alertas conforme necessário'; END IF;

DBMS\_OUTPUT.PUT\_LINE('Situação: ' || v\_situacao\_dia); DBMS\_OUTPUT.PUT\_LINE('Recomendação: ' || v\_recomendacao);

IF v\_alertas\_criticos > 0 THEN DBMS\_OUTPUT.PUT\_LINE(' ');

DBMS\_OUTPUT.PUT\_LINE('⚠️ ATENÇÃO ESPECIAL: ' || v\_alertas\_criticos || ' alertas críticos hoje!');

DBMS\_OUTPUT.PUT\_LINE('Verificar propriedades em risco imediatamente');

END IF;

END IF;

END ANALISAR\_ALERTAS\_DIARIOS;

PROCEDURE VERIFICAR\_RISCO\_ENCHENTE(p\_id\_propriedade IN NUMBER DEFAULT NULL) IS

CURSOR c\_propriedades\_risco IS

SELECT pr.id\_propriedade, pr.nome\_propriedade, prod.nome\_completo, prod.telefone,

AVG(ls.umidade\_solo) as umidade\_media, MAX(ls.precipitacao\_mm) as precipitacao\_max

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_PRODUTOR\_RURAL prod ON pr.id\_produtor = prod.id\_produtor

JOIN GS\_WW\_SENSOR\_IOT si ON pr.id\_propriedade = si.id\_propriedade

JOIN GS\_WW\_LEITURA\_SENSOR ls ON si.id\_sensor = ls.id\_sensor

WHERE (pr.id\_propriedade = p\_id\_propriedade OR p\_id\_propriedade IS NULL)

AND ls.timestamp\_leitura >= SYSDATE - 0.25 -- Últimas 6 horas

GROUP BY pr.id\_propriedade, pr.nome\_propriedade, prod.nome\_completo, prod.telefone

HAVING AVG(ls.umidade\_solo) IS NOT NULL;

v\_prop\_rec c\_propriedades\_risco%ROWTYPE;

v\_nivel\_risco VARCHAR2(20);

v\_acao\_recomendada VARCHAR2(200);

v\_count\_processed NUMBER := 0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== VERIFICAÇÃO DE RISCO DE ENCHENTE ===');

IF p\_id\_propriedade IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Propriedade Específica ID: ' || p\_id\_propriedade);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Todas as Propriedades com Dados Recentes (últimas 6h)');

END IF;

DBMS\_OUTPUT.PUT\_LINE(' ');

FOR v\_prop\_rec IN c\_propriedades\_risco LOOP

v\_count\_processed := v\_count\_processed + 1;

DBMS\_OUTPUT.PUT\_LINE('Propriedade: ' || v\_prop\_rec.nome\_propriedade || ' (ID: ' || v\_prop\_rec.id\_propriedade || ')');

DBMS\_OUTPUT.PUT\_LINE('Produtor: ' || v\_prop\_rec.nome\_completo);

DBMS\_OUTPUT.PUT\_LINE('Umidade Média: ' || ROUND(v\_prop\_rec.umidade\_media, 1) || '%');

DBMS\_OUTPUT.PUT\_LINE('Precipitação Máxima: ' || ROUND(NVL(v\_prop\_rec.precipitacao\_max, 0), 1) || 'mm');

IF v\_prop\_rec.umidade\_media > 90 AND NVL(v\_prop\_rec.precipitacao\_max, 0) > 50 THEN

v\_nivel\_risco := 'EMERGÊNCIA'; v\_acao\_recomendada := 'Evacuar áreas baixas IMEDIATAMENTE!';

ELSIF v\_prop\_rec.umidade\_media > 85 OR NVL(v\_prop\_rec.precipitacao\_max, 0) > 40 THEN

v\_nivel\_risco := 'CRÍTICO'; v\_acao\_recomendada := 'Preparar evacuação e drenar área';

ELSIF v\_prop\_rec.umidade\_media > 70 OR NVL(v\_prop\_rec.precipitacao\_max, 0) > 25 THEN

v\_nivel\_risco := 'ALTO'; v\_acao\_recomendada := 'Monitorar de perto e preparar drenagem';

ELSIF v\_prop\_rec.umidade\_media > 50 OR NVL(v\_prop\_rec.precipitacao\_max, 0) > 15 THEN

v\_nivel\_risco := 'MÉDIO'; v\_acao\_recomendada := 'Continuar monitoramento normal';

ELSE

v\_nivel\_risco := 'BAIXO'; v\_acao\_recomendada := 'Situação normal, sem ações necessárias';

END IF;

DBMS\_OUTPUT.PUT\_LINE('🚨 NÍVEL DE RISCO: ' || v\_nivel\_risco);

DBMS\_OUTPUT.PUT\_LINE('📞 Contato: ' || v\_prop\_rec.telefone);

DBMS\_OUTPUT.PUT\_LINE('✅ Ação Recomendada: ' || v\_acao\_recomendada);

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

END LOOP;

IF v\_count\_processed = 0 THEN

IF p\_id\_propriedade IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('❌ ERRO: Sem dados dos sensores nas últimas 6 horas para a propriedade ID ' || p\_id\_propriedade || ' ou propriedade não encontrada.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nenhuma propriedade com dados de sensor suficientes nas últimas 6 horas encontrada.');

END IF;

END IF;

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('❌ ERRO: ' || SQLERRM);

END VERIFICAR\_RISCO\_ENCHENTE;

PROCEDURE ESTADO\_GERAL\_SOLO IS

CURSOR c\_estado\_solo IS SELECT CASE WHEN pr.latitude > -15 THEN 'NORTE' WHEN pr.latitude > -25 THEN 'CENTRO' ELSE 'SUL' END AS regiao,

nd.descricao\_degradacao, COUNT(\*) AS quantidade\_propriedades, SUM(pr.area\_hectares) AS area\_total,

AVG(CASE WHEN ls.umidade\_solo IS NOT NULL THEN ls.umidade\_solo END) AS umidade\_media

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

LEFT JOIN GS\_WW\_SENSOR\_IOT si ON pr.id\_propriedade = si.id\_propriedade

LEFT JOIN GS\_WW\_LEITURA\_SENSOR ls ON si.id\_sensor = ls.id\_sensor AND ls.timestamp\_leitura >= SYSDATE - 1

GROUP BY CASE WHEN pr.latitude > -15 THEN 'NORTE' WHEN pr.latitude > -25 THEN 'CENTRO' ELSE 'SUL' END, nd.descricao\_degradacao, nd.nivel\_numerico

ORDER BY regiao, nd.nivel\_numerico;

v\_solo c\_estado\_solo%ROWTYPE; v\_regiao\_anterior VARCHAR2(10) := '';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== ESTADO GERAL DO SOLO ==='); DBMS\_OUTPUT.PUT\_LINE('Por região geográfica'); DBMS\_OUTPUT.PUT\_LINE(' ');

FOR v\_solo IN c\_estado\_solo LOOP

IF v\_solo.regiao != v\_regiao\_anterior THEN IF v\_regiao\_anterior != '' THEN DBMS\_OUTPUT.PUT\_LINE(' '); END IF;

DBMS\_OUTPUT.PUT\_LINE('REGIÃO: ' || v\_solo.regiao); DBMS\_OUTPUT.PUT\_LINE('=================='); v\_regiao\_anterior := v\_solo.regiao;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Estado: ' || v\_solo.descricao\_degradacao);

DBMS\_OUTPUT.PUT\_LINE(' Propriedades: ' || v\_solo.quantidade\_propriedades);

DBMS\_OUTPUT.PUT\_LINE(' Área Total: ' || ROUND(NVL(v\_solo.area\_total,0), 1) || ' hectares');

IF v\_solo.umidade\_media IS NOT NULL THEN DBMS\_OUTPUT.PUT\_LINE(' Umidade Média: ' || ROUND(v\_solo.umidade\_media, 1) || '%'); END IF;

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Erro: ' || SQLERRM);

END ESTADO\_GERAL\_SOLO;

PROCEDURE DASHBOARD\_METRICAS IS

v\_total\_propriedades NUMBER; v\_area\_total NUMBER; v\_sensores\_ativos NUMBER; v\_alertas\_criticos NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== DASHBOARD DE MÉTRICAS ESTRATÉGICAS ===');

DBMS\_OUTPUT.PUT\_LINE('Data: ' || TO\_CHAR(SYSDATE, 'DD/MM/YYYY HH24:MI')); DBMS\_OUTPUT.PUT\_LINE(' ');

SELECT COUNT(DISTINCT pr.id\_propriedade), ROUND(SUM(pr.area\_hectares), 0) INTO v\_total\_propriedades, v\_area\_total FROM GS\_WW\_PROPRIEDADE\_RURAL pr;

SELECT COUNT(DISTINCT si.id\_sensor) INTO v\_sensores\_ativos FROM GS\_WW\_SENSOR\_IOT si

JOIN GS\_WW\_LEITURA\_SENSOR ls ON si.id\_sensor = ls.id\_sensor WHERE ls.timestamp\_leitura >= SYSDATE - 1;

SELECT COUNT(a.id\_alerta) INTO v\_alertas\_criticos FROM GS\_WW\_ALERTA a

JOIN GS\_WW\_NIVEL\_SEVERIDADE ns ON a.id\_nivel\_severidade = ns.id\_nivel\_severidade

WHERE a.timestamp\_alerta >= SYSDATE - 1 AND ns.codigo\_severidade = 'CRITICO';

DBMS\_OUTPUT.PUT\_LINE('MÉTRICAS GERAIS'); DBMS\_OUTPUT.PUT\_LINE('===============');

DBMS\_OUTPUT.PUT\_LINE('Total de Propriedades: ' || NVL(v\_total\_propriedades,0) || ' propriedades');

DBMS\_OUTPUT.PUT\_LINE('Área Total Monitorada: ' || NVL(v\_area\_total,0) || ' hectares');

DBMS\_OUTPUT.PUT\_LINE('Sensores Ativos (24h): ' || NVL(v\_sensores\_ativos,0) || ' sensores');

DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos (24h): ' || NVL(v\_alertas\_criticos,0) || ' alertas'); DBMS\_OUTPUT.PUT\_LINE(' ');

DBMS\_OUTPUT.PUT\_LINE('STATUS DAS MÉTRICAS'); DBMS\_OUTPUT.PUT\_LINE('==================');

IF NVL(v\_total\_propriedades,0) >= 50 THEN DBMS\_OUTPUT.PUT\_LINE('Propriedades: EXCELENTE'); ELSIF NVL(v\_total\_propriedades,0) >= 20 THEN DBMS\_OUTPUT.PUT\_LINE('Propriedades: BOM');

ELSIF NVL(v\_total\_propriedades,0) >= 10 THEN DBMS\_OUTPUT.PUT\_LINE('Propriedades: REGULAR'); ELSE DBMS\_OUTPUT.PUT\_LINE('Propriedades: INSUFICIENTE'); END IF;

IF NVL(v\_sensores\_ativos,0) >= 100 THEN DBMS\_OUTPUT.PUT\_LINE('Sensores: EXCELENTE'); ELSIF NVL(v\_sensores\_ativos,0) >= 50 THEN DBMS\_OUTPUT.PUT\_LINE('Sensores: BOM');

ELSIF NVL(v\_sensores\_ativos,0) >= 20 THEN DBMS\_OUTPUT.PUT\_LINE('Sensores: REGULAR'); ELSE DBMS\_OUTPUT.PUT\_LINE('Sensores: INSUFICIENTE'); END IF;

IF NVL(v\_alertas\_criticos,0) = 0 THEN DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos: EXCELENTE'); ELSIF NVL(v\_alertas\_criticos,0) <= 2 THEN DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos: BOM');

ELSIF NVL(v\_alertas\_criticos,0) <= 5 THEN DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos: ATENÇÃO'); ELSE DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos: CRÍTICO'); END IF;

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Erro: ' || SQLERRM);

END DASHBOARD\_METRICAS;

PROCEDURE MELHORES\_PRODUTORES IS

CURSOR c\_melhores\_produtores IS SELECT prod.nome\_completo AS produtor, prod.email,

COUNT(DISTINCT pr.id\_propriedade) AS total\_propriedades, ROUND(SUM(pr.area\_hectares), 1) AS area\_total\_hectares,

COUNT(DISTINCT si.id\_sensor) AS sensores\_instalados, COUNT(a.id\_alerta) AS total\_alertas\_90d,

COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) AS alertas\_criticos\_90d,

ROUND(AVG(nd.nivel\_numerico), 2) AS nivel\_medio\_degradacao, COUNT(CASE WHEN nd.nivel\_numerico <= 2 THEN 1 END) AS propriedades\_solo\_bom,

COUNT(ls.id\_leitura) AS total\_leituras\_30d,

ROUND(100 + (COUNT(CASE WHEN nd.nivel\_numerico <= 2 THEN 1 END) \* 10) + (CASE WHEN COUNT(ls.id\_leitura) / GREATEST(30.0,1) >= 5 THEN 20 ELSE (COUNT(ls.id\_leitura) / GREATEST(30.0,1) \* 4) END) +

(CASE WHEN COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) >= 2 THEN 15 ELSE (COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) \* 7.5) END) -

(COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) \* 15) - (COUNT(a.id\_alerta) \* 2) - (COUNT(CASE WHEN nd.nivel\_numerico >= 4 THEN 1 END) \* 10), 1) AS score\_eficiencia,

CASE WHEN (100 + (COUNT(CASE WHEN nd.nivel\_numerico <= 2 THEN 1 END) \* 10) + (CASE WHEN COUNT(ls.id\_leitura) / GREATEST(30.0,1) >= 5 THEN 20 ELSE (COUNT(ls.id\_leitura) / GREATEST(30.0,1) \* 4) END) +

(CASE WHEN COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) >= 2 THEN 15 ELSE (COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) \* 7.5) END) -

(COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) \* 15) - (COUNT(a.id\_alerta) \* 2) - (COUNT(CASE WHEN nd.nivel\_numerico >= 4 THEN 1 END) \* 10)) >= 120 THEN 'PRODUTOR EXEMPLAR'

WHEN (100 + (COUNT(CASE WHEN nd.nivel\_numerico <= 2 THEN 1 END) \* 10) + (CASE WHEN COUNT(ls.id\_leitura) / GREATEST(30.0,1) >= 5 THEN 20 ELSE (COUNT(ls.id\_leitura) / GREATEST(30.0,1) \* 4) END) +

(CASE WHEN COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) >= 2 THEN 15 ELSE (COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) \* 7.5) END) -

(COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) \* 15) - (COUNT(a.id\_alerta) \* 2) - (COUNT(CASE WHEN nd.nivel\_numerico >= 4 THEN 1 END) \* 10)) >= 100 THEN 'PRODUTOR EFICIENTE'

WHEN (100 + (COUNT(CASE WHEN nd.nivel\_numerico <= 2 THEN 1 END) \* 10) + (CASE WHEN COUNT(ls.id\_leitura) / GREATEST(30.0,1) >= 5 THEN 20 ELSE (COUNT(ls.id\_leitura) / GREATEST(30.0,1) \* 4) END) +

(CASE WHEN COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) >= 2 THEN 15 ELSE (COUNT(DISTINCT si.id\_sensor) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) \* 7.5) END) -

(COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) \* 15) - (COUNT(a.id\_alerta) \* 2) - (COUNT(CASE WHEN nd.nivel\_numerico >= 4 THEN 1 END) \* 10)) >= 80 THEN 'PRODUTOR REGULAR'

ELSE 'PRODUTOR NECESSITA MELHORIA' END AS classificacao\_final

FROM GS\_WW\_PRODUTOR\_RURAL prod

LEFT JOIN GS\_WW\_PROPRIEDADE\_RURAL pr ON prod.id\_produtor = pr.id\_produtor

LEFT JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

LEFT JOIN GS\_WW\_SENSOR\_IOT si ON pr.id\_propriedade = si.id\_propriedade

LEFT JOIN GS\_WW\_LEITURA\_SENSOR ls ON si.id\_sensor = ls.id\_sensor AND ls.timestamp\_leitura >= SYSDATE - 30

LEFT JOIN GS\_WW\_ALERTA a ON prod.id\_produtor = a.id\_produtor AND a.timestamp\_alerta >= SYSDATE - 90

LEFT JOIN GS\_WW\_NIVEL\_SEVERIDADE ns ON a.id\_nivel\_severidade = ns.id\_nivel\_severidade

GROUP BY prod.id\_produtor, prod.nome\_completo, prod.email

ORDER BY score\_eficiencia DESC, alertas\_criticos\_90d ASC, area\_total\_hectares DESC;

v\_produtor c\_melhores\_produtores%ROWTYPE; v\_posicao NUMBER := 1;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== RANKING DOS MELHORES PRODUTORES ===');

DBMS\_OUTPUT.PUT\_LINE('Baseado em práticas sustentáveis e eficiência'); DBMS\_OUTPUT.PUT\_LINE(' ');

FOR v\_produtor IN c\_melhores\_produtores LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_posicao || 'º LUGAR - ' || v\_produtor.classificacao\_final);

DBMS\_OUTPUT.PUT\_LINE('Produtor: ' || v\_produtor.produtor); DBMS\_OUTPUT.PUT\_LINE('Email: ' || v\_produtor.email);

DBMS\_OUTPUT.PUT\_LINE('Score de Eficiência: ' || NVL(v\_produtor.score\_eficiencia,0));

DBMS\_OUTPUT.PUT\_LINE('Propriedades: ' || NVL(v\_produtor.total\_propriedades,0));

DBMS\_OUTPUT.PUT\_LINE('Área Total: ' || NVL(v\_produtor.area\_total\_hectares,0) || ' hectares');

DBMS\_OUTPUT.PUT\_LINE('Sensores: ' || NVL(v\_produtor.sensores\_instalados,0));

DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos (90d): ' || NVL(v\_produtor.alertas\_criticos\_90d,0));

DBMS\_OUTPUT.PUT\_LINE('Propriedades Solo Bom: ' || NVL(v\_produtor.propriedades\_solo\_bom,0));

DBMS\_OUTPUT.PUT\_LINE('Leituras (30d): ' || NVL(v\_produtor.total\_leituras\_30d,0));

DBMS\_OUTPUT.PUT\_LINE('==========================================');

v\_posicao := v\_posicao + 1; EXIT WHEN v\_posicao > 10;

END LOOP;

IF v\_posicao = 1 THEN DBMS\_OUTPUT.PUT\_LINE('Nenhum produtor encontrado para o ranking.'); END IF;

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Erro: ' || SQLERRM);

END MELHORES\_PRODUTORES;

PROCEDURE RISCO\_POR\_REGIAO IS

CURSOR c\_risco\_regiao IS SELECT CASE WHEN pr.latitude > -10 THEN 'REGIÃO NORTE' WHEN pr.latitude > -20 THEN 'REGIÃO NORDESTE'

WHEN pr.latitude > -30 THEN 'REGIÃO CENTRO-SUL' ELSE 'REGIÃO SUL' END AS regiao\_geografica,

COUNT(DISTINCT pr.id\_propriedade) AS total\_propriedades, COUNT(DISTINCT prod.id\_produtor) AS total\_produtores,

ROUND(SUM(pr.area\_hectares), 1) AS area\_total\_hectares, ROUND(AVG(pr.area\_hectares), 1) AS area\_media\_hectares,

COUNT(DISTINCT si.id\_sensor) AS sensores\_instalados, ROUND(AVG(ls.umidade\_solo), 1) AS umidade\_media\_regiao,

ROUND(AVG(ls.temperatura\_ar), 1) AS temperatura\_media, ROUND(SUM(ls.precipitacao\_mm), 1) AS precipitacao\_total\_24h,

ROUND(AVG(nd.nivel\_numerico), 2) AS nivel\_degradacao\_medio, COUNT(CASE WHEN nd.nivel\_numerico >= 4 THEN 1 END) AS propriedades\_solo\_degradado,

COUNT(DISTINCT a.id\_alerta) AS total\_alertas\_7d, COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) AS alertas\_criticos\_7d,

ROUND((NVL(AVG(ls.umidade\_solo),0) \* 0.4 + NVL(AVG(nd.nivel\_numerico),0) \* 20 \* 0.3 + (NVL(SUM(ls.precipitacao\_mm),0) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) / 10) \* 0.3), 1) AS score\_risco\_regional,

CASE WHEN (NVL(AVG(ls.umidade\_solo),0) \* 0.4 + NVL(AVG(nd.nivel\_numerico),0) \* 20 \* 0.3 + (NVL(SUM(ls.precipitacao\_mm),0) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) / 10) \* 0.3) >= 80 THEN 'CRÍTICO'

WHEN (NVL(AVG(ls.umidade\_solo),0) \* 0.4 + NVL(AVG(nd.nivel\_numerico),0) \* 20 \* 0.3 + (NVL(SUM(ls.precipitacao\_mm),0) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) / 10) \* 0.3) >= 60 THEN 'ALTO'

WHEN (NVL(AVG(ls.umidade\_solo),0) \* 0.4 + NVL(AVG(nd.nivel\_numerico),0) \* 20 \* 0.3 + (NVL(SUM(ls.precipitacao\_mm),0) / GREATEST(COUNT(DISTINCT pr.id\_propriedade),1) / 10) \* 0.3) >= 40 THEN 'MÉDIO'

ELSE 'BAIXO' END AS classificacao\_risco\_regional

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_PRODUTOR\_RURAL prod ON pr.id\_produtor = prod.id\_produtor

JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

LEFT JOIN GS\_WW\_SENSOR\_IOT si ON pr.id\_propriedade = si.id\_propriedade

LEFT JOIN GS\_WW\_LEITURA\_SENSOR ls ON si.id\_sensor = ls.id\_sensor AND ls.timestamp\_leitura >= SYSDATE - 1

LEFT JOIN GS\_WW\_ALERTA a ON prod.id\_produtor = a.id\_produtor AND a.timestamp\_alerta >= SYSDATE - 7

LEFT JOIN GS\_WW\_NIVEL\_SEVERIDADE ns ON a.id\_nivel\_severidade = ns.id\_nivel\_severidade

GROUP BY CASE WHEN pr.latitude > -10 THEN 'REGIÃO NORTE' WHEN pr.latitude > -20 THEN 'REGIÃO NORDESTE'

WHEN pr.latitude > -30 THEN 'REGIÃO CENTRO-SUL' ELSE 'REGIÃO SUL' END

ORDER BY score\_risco\_regional DESC, total\_propriedades DESC;

v\_regiao c\_risco\_regiao%ROWTYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== ANÁLISE DE RISCO POR REGIÃO ===');

DBMS\_OUTPUT.PUT\_LINE('Relatório executivo de propriedades em risco'); DBMS\_OUTPUT.PUT\_LINE(' ');

FOR v\_regiao IN c\_risco\_regiao LOOP

DBMS\_OUTPUT.PUT\_LINE('REGIÃO: ' || v\_regiao.regiao\_geografica);

DBMS\_OUTPUT.PUT\_LINE('Classificação de Risco: ' || v\_regiao.classificacao\_risco\_regional);

DBMS\_OUTPUT.PUT\_LINE('Score de Risco: ' || NVL(v\_regiao.score\_risco\_regional,0)); DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Propriedades: ' || v\_regiao.total\_propriedades); DBMS\_OUTPUT.PUT\_LINE('Produtores: ' || v\_regiao.total\_produtores);

DBMS\_OUTPUT.PUT\_LINE('Área Total: ' || NVL(v\_regiao.area\_total\_hectares,0) || ' hectares'); DBMS\_OUTPUT.PUT\_LINE('Sensores: ' || NVL(v\_regiao.sensores\_instalados,0));

IF v\_regiao.umidade\_media\_regiao IS NOT NULL THEN DBMS\_OUTPUT.PUT\_LINE('Umidade Média: ' || v\_regiao.umidade\_media\_regiao || '%'); END IF;

IF v\_regiao.temperatura\_media IS NOT NULL THEN DBMS\_OUTPUT.PUT\_LINE('Temperatura Média: ' || v\_regiao.temperatura\_media || '°C'); END IF;

IF v\_regiao.precipitacao\_total\_24h IS NOT NULL THEN DBMS\_OUTPUT.PUT\_LINE('Chuva (24h): ' || v\_regiao.precipitacao\_total\_24h || 'mm'); END IF;

DBMS\_OUTPUT.PUT\_LINE('Solo Degradado: ' || NVL(v\_regiao.propriedades\_solo\_degradado,0) || ' propriedades');

DBMS\_OUTPUT.PUT\_LINE('Alertas (7d): ' || NVL(v\_regiao.total\_alertas\_7d,0));

DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos (7d): ' || NVL(v\_regiao.alertas\_criticos\_7d,0));

DBMS\_OUTPUT.PUT\_LINE('======================================='); DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Erro: ' || SQLERRM);

END RISCO\_POR\_REGIAO;

PROCEDURE RELATORIO\_PROPRIEDADE(p\_id\_propriedade IN NUMBER) IS

v\_nome\_propriedade VARCHAR2(100); v\_nome\_produtor VARCHAR2(100); v\_email\_produtor VARCHAR2(100);

v\_telefone\_produtor VARCHAR2(15); v\_area\_hectares NUMBER(10,2); v\_latitude NUMBER(10,8);

v\_longitude NUMBER(11,8); v\_descricao\_degradacao VARCHAR2(200); v\_data\_cadastro DATE;

v\_total\_sensores NUMBER; v\_sensores\_ativos NUMBER; v\_total\_alertas NUMBER; v\_alertas\_criticos NUMBER;

v\_risco\_alagamento VARCHAR2(200); v\_taxa\_degradacao VARCHAR2(200); v\_capacidade\_absorcao VARCHAR2(200);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== RELATÓRIO COMPLETO DA PROPRIEDADE ===');

DBMS\_OUTPUT.PUT\_LINE('ID da Propriedade: ' || p\_id\_propriedade);

DBMS\_OUTPUT.PUT\_LINE('Data do Relatório: ' || TO\_CHAR(SYSDATE, 'DD/MM/YYYY HH24:MI')); DBMS\_OUTPUT.PUT\_LINE(' ');

SELECT pr.nome\_propriedade, prod.nome\_completo, prod.email, prod.telefone, pr.area\_hectares, pr.latitude, pr.longitude,

nd.descricao\_degradacao, pr.data\_cadastro

INTO v\_nome\_propriedade, v\_nome\_produtor, v\_email\_produtor, v\_telefone\_produtor, v\_area\_hectares, v\_latitude, v\_longitude,

v\_descricao\_degradacao, v\_data\_cadastro

FROM GS\_WW\_PROPRIEDADE\_RURAL pr

JOIN GS\_WW\_PRODUTOR\_RURAL prod ON pr.id\_produtor = prod.id\_produtor

JOIN GS\_WW\_NIVEL\_DEGRADACAO\_SOLO nd ON pr.id\_nivel\_degradacao = nd.id\_nivel\_degradacao

WHERE pr.id\_propriedade = p\_id\_propriedade;

SELECT COUNT(\*), COUNT(CASE WHEN EXISTS (SELECT 1 FROM GS\_WW\_LEITURA\_SENSOR ls WHERE ls.id\_sensor = si.id\_sensor AND ls.timestamp\_leitura >= SYSDATE - 7) THEN 1 END)

INTO v\_total\_sensores, v\_sensores\_ativos FROM GS\_WW\_SENSOR\_IOT si WHERE si.id\_propriedade = p\_id\_propriedade;

SELECT COUNT(\*), COUNT(CASE WHEN ns.codigo\_severidade = 'CRITICO' THEN 1 END) INTO v\_total\_alertas, v\_alertas\_criticos

FROM GS\_WW\_ALERTA a

JOIN GS\_WW\_NIVEL\_SEVERIDADE ns ON a.id\_nivel\_severidade = ns.id\_nivel\_severidade

JOIN GS\_WW\_PRODUTOR\_RURAL prod\_alerta ON a.id\_produtor = prod\_alerta.id\_produtor

JOIN GS\_WW\_PROPRIEDADE\_RURAL pr\_alerta ON prod\_alerta.id\_produtor = pr\_alerta.id\_produtor

WHERE pr\_alerta.id\_propriedade = p\_id\_propriedade AND a.timestamp\_alerta >= SYSDATE - 30;

v\_risco\_alagamento := CALCULAR\_RISCO\_ALAGAMENTO(p\_id\_propriedade);

v\_taxa\_degradacao := CALCULAR\_TAXA\_DEGRADACAO\_SOLO(p\_id\_propriedade);

v\_capacidade\_absorcao := CALCULAR\_CAPACIDADE\_ABSORCAO(p\_id\_propriedade);

DBMS\_OUTPUT.PUT\_LINE('INFORMAÇÕES GERAIS'); DBMS\_OUTPUT.PUT\_LINE('==================');

DBMS\_OUTPUT.PUT\_LINE('Nome: ' || v\_nome\_propriedade); DBMS\_OUTPUT.PUT\_LINE('Produtor: ' || v\_nome\_produtor);

DBMS\_OUTPUT.PUT\_LINE('Email: ' || v\_email\_produtor); DBMS\_OUTPUT.PUT\_LINE('Telefone: ' || v\_telefone\_produtor);

DBMS\_OUTPUT.PUT\_LINE('Área: ' || v\_area\_hectares || ' hectares');

DBMS\_OUTPUT.PUT\_LINE('Coordenadas: ' || v\_latitude || ', ' || v\_longitude);

DBMS\_OUTPUT.PUT\_LINE('Estado do Solo: ' || v\_descricao\_degradacao); DBMS\_OUTPUT.PUT\_LINE('Cadastro: ' || TO\_CHAR(v\_data\_cadastro, 'DD/MM/YYYY')); DBMS\_OUTPUT.PUT\_LINE(' ');

DBMS\_OUTPUT.PUT\_LINE('MONITORAMENTO'); DBMS\_OUTPUT.PUT\_LINE('=============');

DBMS\_OUTPUT.PUT\_LINE('Total de Sensores: ' || v\_total\_sensores); DBMS\_OUTPUT.PUT\_LINE('Sensores Ativos (7 dias): ' || v\_sensores\_ativos);

DBMS\_OUTPUT.PUT\_LINE('Total Alertas (30 dias): ' || v\_total\_alertas); DBMS\_OUTPUT.PUT\_LINE('Alertas Críticos (30 dias): ' || v\_alertas\_criticos); DBMS\_OUTPUT.PUT\_LINE(' ');

DBMS\_OUTPUT.PUT\_LINE('ANÁLISES ESPECIALIZADAS'); DBMS\_OUTPUT.PUT\_LINE('=======================');

DBMS\_OUTPUT.PUT\_LINE('Risco de Alagamento: ' || v\_risco\_alagamento); DBMS\_OUTPUT.PUT\_LINE('Taxa de Degradação: ' || v\_taxa\_degradacao);

DBMS\_OUTPUT.PUT\_LINE('Capacidade de Absorção: ' || v\_capacidade\_absorcao);

EXCEPTION WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('❌ ERRO: Propriedade não encontrada (ID: ' || p\_id\_propriedade || ')');

WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('❌ ERRO em RELATORIO\_PROPRIEDADE: ' || SQLERRM);

END RELATORIO\_PROPRIEDADE;

-- ============================================================================

-- 4. IMPLEMENTAÇÃO DAS PROCEDURES UTILITÁRIAS

-- ============================================================================

PROCEDURE INICIALIZAR\_SISTEMA IS

v\_id\_temp NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== INICIALIZANDO SISTEMA WATERWISE ===');

BEGIN

PKG\_WATERWISE\_SLIM.CRUD\_TIPO\_SENSOR('INSERT', v\_id\_temp, 'Sensor de Umidade do Solo', 'Sensor capacitivo para medição da umidade do solo', '%', 0, 100);

PKG\_WATERWISE\_SLIM.CRUD\_TIPO\_SENSOR('INSERT', v\_id\_temp, 'Sensor de Temperatura', 'Sensor digital para medição da temperatura ambiente', '°C', -40, 85);

PKG\_WATERWISE\_SLIM.CRUD\_TIPO\_SENSOR('INSERT', v\_id\_temp, 'Sensor de Precipitação', 'Pluviômetro digital para medição de chuva', 'mm', 0, 500);

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Tipos de sensores já existem ou erro: ' || SQLERRM); END;

BEGIN

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_SEVERIDADE('INSERT', v\_id\_temp, 'BAIXO', 'Situação sob controle, monitoramento rotineiro', 'Continuar monitoramento regular. Verificar tendências.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_SEVERIDADE('INSERT', v\_id\_temp, 'MEDIO', 'Situação requer atenção, monitoramento intensificado', 'Aumentar frequência de monitoramento. Verificar causas.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_SEVERIDADE('INSERT', v\_id\_temp, 'ALTO', 'Situação preocupante, ação necessária', 'Ação corretiva imediata. Contatar responsável técnico.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_SEVERIDADE('INSERT', v\_id\_temp, 'CRITICO', 'Situação crítica, ação imediata necessária', 'Intervenção imediata. Contatar especialista. Implementar medidas corretivas urgentes.');

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Níveis de severidade já existem ou erro: ' || SQLERRM); END;

BEGIN

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_DEGRADACAO\_SOLO('INSERT', v\_id\_temp, 'EXCELENTE', 'Solo em excelente estado de conservação', 1, 'Manter práticas atuais. Monitoramento preventivo.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_DEGRADACAO\_SOLO('INSERT', v\_id\_temp, 'BOM', 'Solo em bom estado, pequenos sinais de desgaste', 2, 'Aplicar cobertura vegetal. Reduzir pisoteio.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_DEGRADACAO\_SOLO('INSERT', v\_id\_temp, 'MODERADO', 'Degradação moderada, perda média de fertilidade', 3, 'Análise de solo. Correção química. Rotação de culturas.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_DEGRADACAO\_SOLO('INSERT', v\_id\_temp, 'RUIM', 'Degradação avançada, perda significativa de fertilidade', 4, 'Recuperação intensiva. Análise detalhada. Plantio de recuperação.');

PKG\_WATERWISE\_SLIM.CRUD\_NIVEL\_DEGRADACAO\_SOLO('INSERT', v\_id\_temp, 'CRITICO', 'Degradação crítica, solo quase improdutivo', 5, 'Recuperação emergencial. Projeto técnico especializado.');

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Níveis de degradação já existem ou erro: ' || SQLERRM); END;

DBMS\_OUTPUT.PUT\_LINE('✅ Sistema WaterWise inicializado com sucesso!');

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Erro na inicialização: ' || SQLERRM); ROLLBACK;

END INICIALIZAR\_SISTEMA;

PROCEDURE VALIDAR\_INTEGRIDADE\_DADOS IS

v\_produtores\_sem\_propriedade NUMBER; v\_propriedades\_sem\_sensor NUMBER;

v\_sensores\_sem\_leitura NUMBER; v\_alertas\_sem\_nivel NUMBER; v\_total\_problemas NUMBER := 0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== VALIDAÇÃO DE INTEGRIDADE DOS DADOS ===');

DBMS\_OUTPUT.PUT\_LINE('Verificando consistência do banco de dados'); DBMS\_OUTPUT.PUT\_LINE(' ');

SELECT COUNT(\*) INTO v\_produtores\_sem\_propriedade FROM GS\_WW\_PRODUTOR\_RURAL prod WHERE NOT EXISTS (SELECT 1 FROM GS\_WW\_PROPRIEDADE\_RURAL pr WHERE pr.id\_produtor = prod.id\_produtor);

SELECT COUNT(\*) INTO v\_propriedades\_sem\_sensor FROM GS\_WW\_PROPRIEDADE\_RURAL pr WHERE NOT EXISTS (SELECT 1 FROM GS\_WW\_SENSOR\_IOT si WHERE si.id\_propriedade = pr.id\_propriedade);

SELECT COUNT(\*) INTO v\_sensores\_sem\_leitura FROM GS\_WW\_SENSOR\_IOT si WHERE NOT EXISTS (SELECT 1 FROM GS\_WW\_LEITURA\_SENSOR ls WHERE ls.id\_sensor = si.id\_sensor AND ls.timestamp\_leitura >= SYSDATE - 7);

SELECT COUNT(\*) INTO v\_alertas\_sem\_nivel FROM GS\_WW\_ALERTA a WHERE NOT EXISTS (SELECT 1 FROM GS\_WW\_NIVEL\_SEVERIDADE ns WHERE ns.id\_nivel\_severidade = a.id\_nivel\_severidade);

DBMS\_OUTPUT.PUT\_LINE('RESULTADOS DA VALIDAÇÃO:'); DBMS\_OUTPUT.PUT\_LINE('=======================');

IF v\_produtores\_sem\_propriedade > 0 THEN DBMS\_OUTPUT.PUT\_LINE('⚠️ ' || v\_produtores\_sem\_propriedade || ' produtores sem propriedades cadastradas'); v\_total\_problemas := v\_total\_problemas + 1; END IF;

IF v\_propriedades\_sem\_sensor > 0 THEN DBMS\_OUTPUT.PUT\_LINE('⚠️ ' || v\_propriedades\_sem\_sensor || ' propriedades sem sensores instalados'); v\_total\_problemas := v\_total\_problemas + 1; END IF;

IF v\_sensores\_sem\_leitura > 0 THEN DBMS\_OUTPUT.PUT\_LINE('⚠️ ' || v\_sensores\_sem\_leitura || ' sensores sem leituras na última semana'); v\_total\_problemas := v\_total\_problemas + 1; END IF;

IF v\_alertas\_sem\_nivel > 0 THEN DBMS\_OUTPUT.PUT\_LINE('❌ ' || v\_alertas\_sem\_nivel || ' alertas com referências inválidas'); v\_total\_problemas := v\_total\_problemas + 1; END IF;

DBMS\_OUTPUT.PUT\_LINE(' ');

IF v\_total\_problemas = 0 THEN DBMS\_OUTPUT.PUT\_LINE('✅ INTEGRIDADE OK - Nenhum problema encontrado'); DBMS\_OUTPUT.PUT\_LINE('Base de dados consistente e funcional');

ELSE DBMS\_OUTPUT.PUT\_LINE('🔧 AÇÃO NECESSÁRIA - ' || v\_total\_problemas || ' tipos de problemas encontrados'); DBMS\_OUTPUT.PUT\_LINE('Revisar dados e corrigir inconsistências'); END IF;

EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Erro na validação: ' || SQLERRM);

END VALIDAR\_INTEGRIDADE\_DADOS;

END PKG\_WATERWISE\_SLIM;

/

**Prints de execução**

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

**Conclusão**

**O WaterWise representa uma solução inovadora para um dos maiores desafios ambientais do século XXI. Através da integração de tecnologias modernas de banco de dados, IoT e análise de dados, o sistema transforma propriedades rurais em sentinelas inteligentes contra enchentes.**

**A implementação acadêmica demonstra a viabilidade técnica da solução, com arquitetura robusta, modelagem de dados consistente e funcionalidades abrangentes. O projeto estabelece as bases para um sistema que pode ser escalado para implementação real, contribuindo efetivamente para a prevenção de desastres naturais e sustentabilidade ambiental.**

*"Cada gota monitorada hoje é uma enchente prevenida amanhã."*