FIAP

Faculdade de Informática e Administração Paulista

Data Application & Data Science

Global Solution - BlueGather

FIAP

INTEGRANTES

RM (SOMENTE NÚMEROS)	NOME COMPLEMENTO (SEM ABREVIAR)
97068	Gustavo Sorrilha Sanches
97324	Natan Cruz
97092	Vitor Rubim Passos
97503	Mariana Santos Fernandes de Sousa
96466	Kaue Caponero Figueiredo

Sumário

01. DESCRIÇÃO DO PROJETO

O BlueGather é um sistema de gerenciamento de voluntários projetado para capacitar os usuários a coordenarem e participarem de iniciativas sociais, como por exemplo limpeza de praias, resgate de animais marinhos, etc... Funciona de maneira simples e eficaz: os usuários cadastram a um evento onde há necessidade de alguma ação, como por exemplo em áreas com alta concentração de lixo marinho. As pessoas interessadas em ajudar podem ingressar nesses eventos, comprometendo-se a participar da limpeza. Uma vez que a data para a limpeza é definida os voluntários se reúnem no local designado e trabalham em conjunto para realizar a limpeza, dedicando seu tempo e esforço para remover o lixo dos ambientes marinhos. Durante todo o processo é permitida a adição de imagens como fotos do local definidos pelo momento específico (antes, durante e depois) e após a finalização do evento é feita uma avaliação de cada voluntário para aquele evento, garantindo a autenticidade das informações compartilhadas e reconhecendo o esforço conjunto da comunidade em prol da preservação dos oceanos.

02. OBJETIVOS

O objetivo do BlueGather é engajar a comunidade em ações sociais que promovam a limpeza e a conservação dos oceanos. Através de um aplicativo móvel, os usuários podem criar e participar de eventos como limpeza de praias, resgate de animais marinhos, etc.. contribuindo ativamente para um ambiente marinho mais saudável.

Nosso público alvo são:

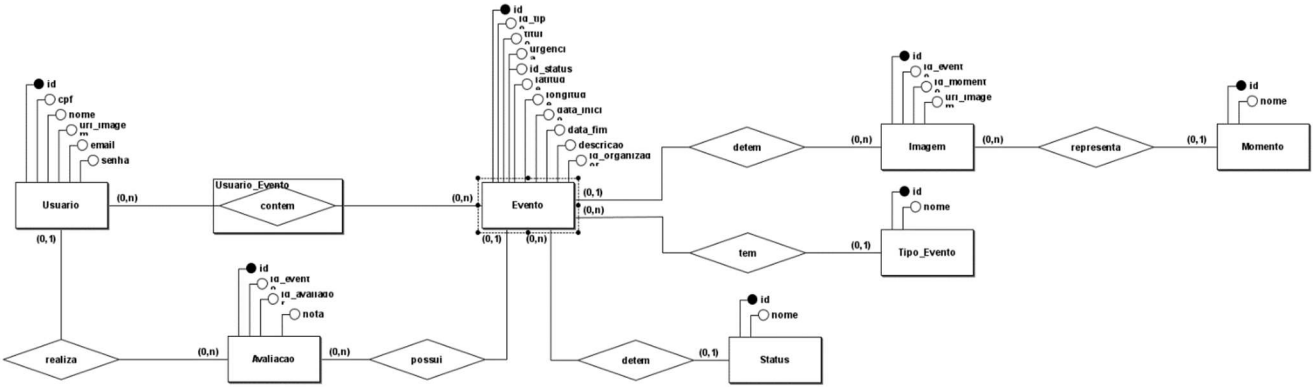
- Cidadãos conscientes e preocupados com o meio ambiente;

- Organizações não governamentais e instituições voltadas para a preservação ambiental;

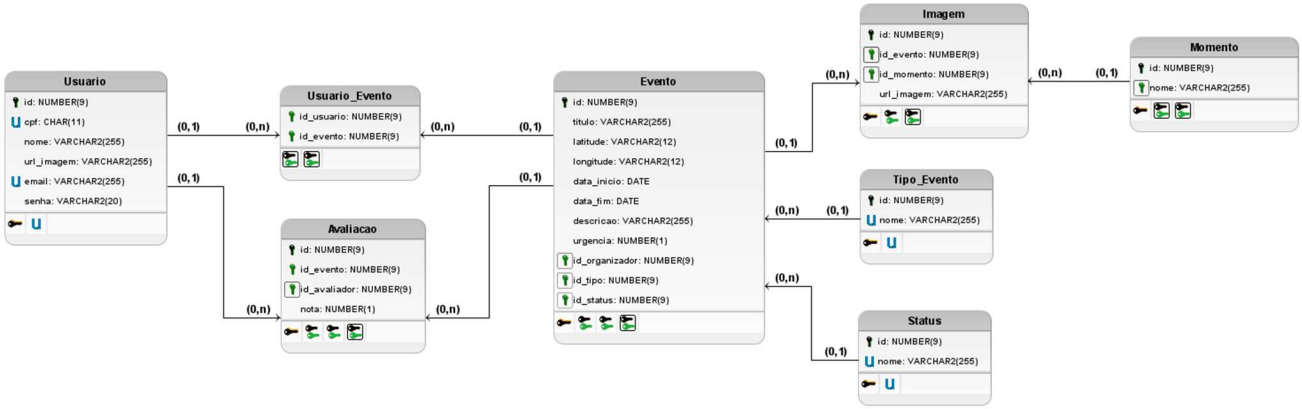
- Empresas que desejam engajar seus funcionários em atividades de responsabilidade social corporativa;

,

03. MODELO LÓGICO DO BANCO DE DADOS



04. MODELO FÍSICO DO BANCO DE DADOS



05. PRINTS DE EXECUÇÕES DOS SCRIPTS

05.01. CRIAÇÕES DAS TABELAS

```
CREATE TABLE usuario (
    id NUMBER(9) CONSTRAINT pk_id_usuario PRIMARY KEY,
    cpf CHAR(11) CONSTRAINT uk_cpf_usuario UNIQUE CONSTRAINT nn_cpf_usuario NOT NULL,
    nome VARCHAR2(255) CONSTRAINT nn_nome_usuario NOT NULL,
    url_imagem VARCHAR2(255),
    email VARCHAR2(255) CONSTRAINT uk_email_usuario UNIQUE CONSTRAINT nn_email_usuario NOT NULL,
    senha VARCHAR2(255) CONSTRAINT nn_senha_usuario NOT NULL
);
```

```
Saída do Script
Tarefa concluída em 0,057 segundos

Table USUARIO criado.
```

```
CREATE TABLE tipo_evento (
    id NUMBER(9) CONSTRAINT pk_id_tipo PRIMARY KEY,
    nome VARCHAR2(255) CONSTRAINT uk_nome_tipo UNIQUE CONSTRAINT nn_nome_tipo NOT NULL
);
```

```
Saída do Script
Tarefa concluída em 0,04 segundos

Table TIPO_EVENTO criado.
```

```
CREATE TABLE status (
    id NUMBER(9) CONSTRAINT pk_id_status PRIMARY KEY,
    nome VARCHAR2(255) CONSTRAINT uk_nome_status UNIQUE CONSTRAINT nn_nome_status NOT NULL
);
```

```
Saída do Script
Tarefa concluída em 0,031 segundos

Table STATUS criado.
```

```
CREATE TABLE momento (
    id NUMBER(9) CONSTRAINT pk_id_momento PRIMARY KEY,
    nome VARCHAR2(255) CONSTRAINT uk_nome_momento UNIQUE CONSTRAINT nn_nome_momento NOT NULL
);
```

```
Saída do Script
Tarefa concluída em 0,038 segundos

Table MOMENTO criado.
```

```
CREATE TABLE evento (
    id NUMBER(9) CONSTRAINT pk_id_evento PRIMARY KEY,
    titulo VARCHAR2(255) CONSTRAINT nn_titulo_evento NOT NULL,
    latitude VARCHAR2(12) CONSTRAINT nn_latitude_evento NOT NULL,
    longitude VARCHAR2(12) CONSTRAINT nn_longitude_evento NOT NULL,
    data_inicio DATE,
    data_fim DATE,
    descricao VARCHAR2(255),
    urgencia NUMBER(1) CONSTRAINT nn_urgencia_evento NOT NULL,
    id_organizador NUMBER(9) CONSTRAINT fk_id_organizador_evento REFERENCES usuario(id),
    id_tipo NUMBER(9) CONSTRAINT fk_id_tipo_evento REFERENCES tipo_evento(id) CONSTRAINT nn_tipo_evento NOT NULL,
    id_status NUMBER(9) CONSTRAINT fk_id_status_evento REFERENCES status(id) CONSTRAINT nn_status_evento NOT NULL
);
```

Saída do Script | Tarefa concluída em 0,042 segundos

```
Table EVENTO criado.
```

```
CREATE TABLE imagem (
    id NUMBER(9) CONSTRAINT pk_id_imagem PRIMARY KEY,
    id_evento NUMBER(9) CONSTRAINT fk_id_evento_imagem REFERENCES evento(id) CONSTRAINT nn_evento_imagem NOT NULL,
    id_momento NUMBER(9) CONSTRAINT fk_id_momento_imagem REFERENCES momento(id) CONSTRAINT nn_momento_imagem NOT NULL,
    url_imagem VARCHAR2(255) CONSTRAINT nn_url_imagem NOT NULL
```

Saída do Script | Tarefa concluída em 0,033 segundos

```
Table IMAGEM criado.
```

```
CREATE TABLE avaliacao (
    id NUMBER(9) CONSTRAINT pk_id_avaliacao PRIMARY KEY,
    id_evento NUMBER(9) CONSTRAINT fk_id_evento_avaliacao REFERENCES evento(id) CONSTRAINT nn_evento_avaliacao NOT NULL,
    id_avaliador NUMBER(9) CONSTRAINT fk_id_avaliador_avaliacao REFERENCES usuario(id) CONSTRAINT nn_avaliador_avaliacao NOT
    nota NUMBER(1) CONSTRAINT nn_nota NOT NULL
);
```

Saída do Script | Tarefa concluída em 0,038 segundos

```
Table AVALIACAO criado.
```

```
CREATE TABLE usuario_evento (
    id_usuario NUMBER(9) CONSTRAINT fk_id_usuario_evento REFERENCES usuario(id) CONSTRAINT nn_id_usuario_evento NOT NULL,
    id_evento NUMBER(9) CONSTRAINT fk_id_evento_usuario REFERENCES evento(id) CONSTRAINT nn_id_evento_usuario NOT NULL,
    CONSTRAINT pk_usuario_evento PRIMARY KEY (id_usuario, id_evento)
);
```

Saída do Script | Tarefa concluída em 0,034 segundos

```
Table USUARIO_EVENTO criado.
```

05.02. INSERTS

```
--- INSERTS
INSERT INTO usuario VALUES (sq_usuario.NEXTVAL, '11111111111', 'Gustavo Sanches', 'https://avatars.githubusercontent.com/u/111543305?v=
INSERT INTO usuario VALUES (sq_usuario.NEXTVAL, '22222222222', 'Kaue Caponero', 'https://avatars.githubusercontent.com/u/111543330?v=4'
INSERT INTO usuario VALUES (sq_usuario.NEXTVAL, '33333333333', 'Mariana Santos', 'https://avatars.githubusercontent.com/u/56116824?v=4'
INSERT INTO usuario VALUES (sq_usuario.NEXTVAL, '44444444444', 'Natan Cruz', 'https://avatars.githubusercontent.com/u/111809342?v=4', '
INSERT INTO usuario VALUES (sq_usuario.NEXTVAL, '55555555555', 'Vitor Rubim', 'https://avatars.githubusercontent.com/u/48107882?v=4', '

INSERT INTO tipo_evento VALUES (sq_tipo_evento.NEXTVAL, 'Limpeza de Praias');
INSERT INTO tipo_evento VALUES (sq_tipo_evento.NEXTVAL, 'Passeata de Conscientização Ambiental');
INSERT INTO tipo_evento VALUES (sq_tipo_evento.NEXTVAL, 'Resgate de Animais Marinhos');
INSERT INTO tipo_evento VALUES (sq_tipo_evento.NEXTVAL, 'Coleta de Lixo Reciclável');
INSERT INTO tipo_evento VALUES (sq_tipo_evento.NEXTVAL, 'Protesto');
```

Saída do Script

Tarefa concluída em 0,116 segundos

```
1 linha inserido.

1 linha inserido.

1 linha inserido.

1 linha inserido.

1 linha inserido.

1 linha inserido.

1 linha inserido.
```

```
INSERT INTO momento VALUES (1, 'Antes');
INSERT INTO momento VALUES (2, 'Durante');
INSERT INTO momento VALUES (3, 'Depois');

INSERT INTO imagem VALUES (sq_imagem.NEXTVAL, 1, 1, 'https://hardcore.com.br/wp-content/uploads/sites/21/2021/01/poluicao-plastica-em-
INSERT INTO imagem VALUES (sq_imagem.NEXTVAL, 1, 2, 'https://voiceoftheoceans.com/wp-content/uploads/2022/09/27e8fd00-c478-4522-88ad-f
INSERT INTO imagem VALUES (sq_imagem.NEXTVAL, 1, 3, 'https://turismo.ubatuba.sp.gov.br/wp-content/uploads/sites/29/2014/10/DSC01621.jp
INSERT INTO imagem VALUES (sq_imagem.NEXTVAL, 2, 2, 'https://f.i.uol.com.br/fotografia/2013/06/20/291240-970x600-1.jpeg');
INSERT INTO imagem VALUES (sq_imagem.NEXTVAL, 4, 2, 'https://camboriu.news/wp-content/uploads/2020/11/salvar-tartaruga.jpg');

INSERT INTO avaliacao VALUES (sq_avaliacao.NEXTVAL, 1, 2, 5);
INSERT INTO avaliacao VALUES (sq_avaliacao.NEXTVAL, 1, 3, 3);
INSERT INTO avaliacao VALUES (sq_avaliacao.NEXTVAL, 1, 4, 3);
INSERT INTO avaliacao VALUES (sq_avaliacao.NEXTVAL, 3, 5, 1);
INSERT INTO avaliacao VALUES (sq_avaliacao.NEXTVAL, 3, 4, 1);

INSERT INTO usuario_evento VALUES (1, 1);
INSERT INTO usuario_evento VALUES (4, 2);
INSERT INTO usuario_evento VALUES (1, 4);
INSERT INTO usuario_evento VALUES (2, 4);
INSERT INTO usuario_evento VALUES (3, 5);
```

Saída do Script

Tarefa concluída em 0,152 segundos

```
1 linha inserido.

1 linha inserido.

1 linha inserido.

1 linha inserido.

1 linha inserido.
```

05.03. PROCEDURES

```
CREATE OR REPLACE PROCEDURE inserir_usuario (
    p_id IN NUMBER,
    p_cpf IN CHAR,
    p_nome IN VARCHAR2,
    p_url_imagem IN VARCHAR2,
    p_email IN VARCHAR2,
    p_senha IN VARCHAR2
) IS
BEGIN
    INSERT INTO Usuario (id, cpf, nome, url_imagem, email, senha)
    VALUES (p_id, p_cpf, p_nome, p_url_imagem, p_email, p_senha);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: CPF ou email já cadastrados.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,101 segundos

```
Procedure INSERIR_USUARIO compilado
```

```
CREATE OR REPLACE PROCEDURE atualizar_usuario (
    p_id IN NUMBER,
    p_cpf IN CHAR,
    p_nome IN VARCHAR2,
    p_url_imagem IN VARCHAR2,
    p_email IN VARCHAR2,
    p_senha IN VARCHAR2
) IS
BEGIN
    UPDATE Usuario
    SET cpf = p_cpf,
        nome = p_nome,
        url_imagem = p_url_imagem,
        email = p_email,
        senha = p_senha
    WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Usuário não encontrado.');
```

Saída do Script

Tarefa concluída em 0,049 segundos

```
Procedimento PL/SQL concluído com sucesso.


Procedure ATUALIZAR_USUARIO compilado
```

```plsql
CREATE OR REPLACE PROCEDURE deletar_usuario (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Usuario WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Usuário não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,038 segundos

```
Procedure DELETAR_USUARIO compilado
```

```plsql
CREATE OR REPLACE PROCEDURE inserir_tipo_evento (
    p_id IN NUMBER,
    p_nome IN VARCHAR2
) IS
BEGIN
    INSERT INTO Tipo_Evento (id, nome)
    VALUES (p_id, p_nome);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Tipo de evento já cadastrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,048 segundos

```
Procedimento PL/SQL concluído com sucesso.


Procedure INSERIR_TIPO_EVENTO compilado
```

```plsql
CREATE OR REPLACE PROCEDURE atualizar_tipo_evento (
    p_id IN NUMBER,
    p_nome IN VARCHAR2
) IS
BEGIN
    UPDATE Tipo_Evento
    SET nome = p_nome
    WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Tipo de evento não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,027 segundos

```
Procedure ATUALIZAR_TIPO_EVENTO compilado
```

```sql
CREATE OR REPLACE PROCEDURE deletar_tipo_evento (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Tipo_Evento WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Tipo de evento não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,034 segundos

```
Procedure DELETAR_TIPO_EVENTO compilado
```

```sql
CREATE OR REPLACE PROCEDURE inserir_status (
    p_id IN NUMBER,
    p_nome IN VARCHAR2
) IS
BEGIN
    INSERT INTO Status (id, nome)
    VALUES (p_id, p_nome);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Status já cadastrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,05 segundos

```
Procedure INSERIR_STATUS compilado
```

```sql
CREATE OR REPLACE PROCEDURE atualizar_status (
    p_id IN NUMBER,
    p_nome IN VARCHAR2
) IS
BEGIN
    UPDATE Status
    SET nome = p_nome
    WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Status não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,047 segundos

```
Procedure ATUALIZAR_STATUS compilado
```

```plsql
CREATE OR REPLACE PROCEDURE deletar_status (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Status WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Status não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,039 segundos

```
Procedure DELETAR_STATUS compilado
```

```plsql
CREATE OR REPLACE PROCEDURE inserir_evento (
    p_id IN NUMBER,
    p_titulo IN VARCHAR2,
    p_latitude IN VARCHAR2,
    p_longitude IN VARCHAR2,
    p_data_inicio IN DATE,
    p_data_fim IN DATE,
    p_descricao IN VARCHAR2,
    p_urgencia IN NUMBER,
    p_id_organizador IN NUMBER,
    p_id_tipo IN NUMBER,
    p_id_status IN NUMBER
) IS
BEGIN
    INSERT INTO Evento (id, titulo, latitude, longitude, data_inicio, data_fim, descricao, urgencia, id_organizador, id_tipo, id_statu
    VALUES (p_id, p_titulo, p_latitude, p_longitude, p_data_inicio, p_data_fim, p_descricao, p_urgencia, p_id_organizador, p_id_tipo,

    COMMIT;
```

Saída do Script

Tarefa concluída em 0,07 segundos

```
Procedure INSERIR_EVENTO compilado
```

```plsql
CREATE OR REPLACE PROCEDURE atualizar_evento (
    p_id IN NUMBER,
    p_titulo IN VARCHAR2,
    p_latitude IN VARCHAR2,
    p_longitude IN VARCHAR2,
    p_data_inicio IN DATE,
    p_data_fim IN DATE,
    p_descricao IN VARCHAR2,
    p_urgencia IN NUMBER,
    p_id_organizador IN NUMBER,
    p_id_tipo IN NUMBER,
    p_id_status IN NUMBER
) IS
BEGIN
    UPDATE Evento
    SET titulo = p_titulo,
        latitude = p_latitude,
        longitude = p_longitude,
        data_inicio = p_data_inicio,
```

Saída do Script

Tarefa concluída em 0,047 segundos

```
Procedure ATUALIZAR_EVENTO compilado
```

```
CREATE OR REPLACE PROCEDURE deletar_evento (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Evento WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Evento não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,034 segundos

```
rocedure DELETAR_EVENTO compilado
```

```
CREATE OR REPLACE PROCEDURE inserir_momento (
    p_id IN NUMBER,
    p_nome IN VARCHAR2
) IS
BEGIN
    INSERT INTO Momento (id, nome)
    VALUES (p_id, p_nome);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Momento já cadastrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;

-- Teste para inserir momento
```

Saída do Script | Tarefa concluída em 0,046 segundos

```
Procedure INSERIR_MOMENTO compilado
```

```
CREATE OR REPLACE PROCEDURE atualizar_momento (
    p_id IN NUMBER,
    p_nome IN VARCHAR2
) IS
BEGIN
    UPDATE Momento
    SET nome = p_nome
    WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Momento não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,047 segundos

```
Procedure ATUALIZAR_MOMENTO compilado
```

```
CREATE OR REPLACE PROCEDURE deletar_momento (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Momento WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Momento não encontrado.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,035 segundos

```
Procedure DELETAR_MOMENTO compilado
```

```
CREATE OR REPLACE PROCEDURE inserir_imagem (
    p_id IN NUMBER,
    p_id_evento IN NUMBER,
    p_id_momento IN NUMBER,
    p_url_imagem IN VARCHAR2
) IS
BEGIN
    INSERT INTO Imagem (id, id_evento, id_momento, url_imagem)
    VALUES (p_id, p_id_evento, p_id_momento, p_url_imagem);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Imagem já cadastrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
```

Saída do Script | Tarefa concluída em 0,053 segundos

```
Procedure INSERIR_IMAGEM compilado
```

```
CREATE OR REPLACE PROCEDURE atualizar_imagem (
    p_id IN NUMBER,
    p_id_evento IN NUMBER,
    p_id_momento IN NUMBER,
    p_url_imagem IN VARCHAR2
) IS
BEGIN
    UPDATE Imagem
    SET id_evento = p_id_evento,
        id_momento = p_id_momento,
        url_imagem = p_url_imagem
    WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Imagem não encontrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
```

Saída do Script | Tarefa concluída em 0,045 segundos

```
Procedure ATUALIZAR_IMAGEM compilado
```

```plsql
CREATE OR REPLACE PROCEDURE deletar_imagem (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Imagem WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Imagem não encontrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,038 segundos

```
Procedure DELETAR_IMAGEM compilado
```

```plsql
CREATE OR REPLACE PROCEDURE inserir_avaliacao (
    p_id IN NUMBER,
    p_id_evento IN NUMBER,
    p_id_avaliador IN NUMBER,
    p_nota IN NUMBER
) IS
BEGIN
    INSERT INTO Avaliacao (id, id_evento, id_avaliador, nota)
    VALUES (p_id, p_id_evento, p_id_avaliador, p_nota);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Avaliação já cadastrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,055 segundos

```
Procedure INSERIR_AVALIACAO compilado
```

```plsql
CREATE OR REPLACE PROCEDURE atualizar_avaliacao (
    p_id IN NUMBER,
    p_id_evento IN NUMBER,
    p_id_avaliador IN NUMBER,
    p_nota IN NUMBER
) IS
BEGIN
    UPDATE Avaliacao
    SET id_evento = p_id_evento,
        id_avaliador = p_id_avaliador,
        nota = p_nota
    WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Avaliação não encontrada.');
```

Saída do Script

Tarefa concluída em 0,048 segundos

```
Procedure ATUALIZAR_AVALIACAO compilado
```

```
CREATE OR REPLACE PROCEDURE deletar_avaliacao (
    p_id IN NUMBER
) IS
BEGIN
    DELETE FROM Avaliacao WHERE id = p_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Avaliação não encontrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,041 segundos

```
Procedure DELETAR_AVALIACAO compilado
```

```
-- Procedures da tabela Usuario_Evento
CREATE OR REPLACE PROCEDURE inserir_usuario_evento (
    p_id_usuario IN NUMBER,
    p_id_evento IN NUMBER
) IS
BEGIN
    INSERT INTO Usuario_Evento (id_usuario, id_evento)
    VALUES (p_id_usuario, p_id_evento);

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Associação usuário-evento já cadastrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script

Tarefa concluída em 0,05 segundos

```
Procedure INSERIR_USUARIO_EVENTO compilado
```

```
-- Atualizar Usuario_Evento
CREATE OR REPLACE PROCEDURE atualizar_usuario_evento (
    p_id_usuario IN NUMBER,
    p_id_evento IN NUMBER,
    p_novo_id_evento IN NUMBER
) IS
BEGIN
    UPDATE Usuario_Evento
    SET id_evento = p_novo_id_evento
    WHERE id_usuario = p_id_usuario AND id_evento = p_id_evento;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Associacao usuario-evento nao encontrada.');
    END IF;

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erro: Associacao usuario-evento ja cadastrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,058 segundos

```
Procedure ATUALIZAR_USUARIO_EVENTO compilado
```

```
CREATE OR REPLACE PROCEDURE deletar_usuario_evento (
    p_id_usuario IN NUMBER,
    p_id_evento IN NUMBER
) IS
BEGIN
    DELETE FROM Usuario_Evento WHERE id_usuario = p_id_usuario AND id_evento = p_id_evento;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Erro: Associação usuário-evento não encontrada.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erro: ' || SQLERRM);
END;
```

Saída do Script | Tarefa concluída em 0,045 segundos

```
Procedure DELETAR_USUARIO_EVENTO compilado
```

05.04. RELATÓRIOS

```
-- Relatório de Participação de Usuários em Eventos
CREATE OR REPLACE PROCEDURE relatorio_participacao_usuarios
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Relatório de Participação de Usuários em Eventos');
    DBMS_OUTPUT.PUT_LINE('==========================================');
    FOR rec IN (SELECT u.nome AS usuario_nome, e.titulo AS evento_titulo, a.nota AS avaliacao
                FROM usuario u
                JOIN usuario_evento ue ON u.id = ue.id_usuario
                JOIN evento e ON ue.id_evento = e.id
                LEFT JOIN avaliacao a ON e.id = a.id_evento AND a.id_avaliador = u.id
                ORDER BY u.nome, e.titulo)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Usuário: ' || rec.usuario_nome || ', Evento: ' || rec.evento_titulo || ', Avaliação: ' || NVL(TO_C
    END LOOP;
END;
```

Saída do Script

Tarefa concluída em 0,051 segundos

```
Procedure RELATORIO_PARTICIPACAO_USUARIOS compilado
```

```
-- Relatório de Eventos por Status
CREATE OR REPLACE PROCEDURE relatorio_eventos_por_status
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Relatório de Eventos por Status');
    DBMS_OUTPUT.PUT_LINE('============================');
    FOR rec IN (SELECT e.titulo AS evento_titulo, s.nome AS status_nome, u.nome AS organizador_nome, e.data_inicio, e.data_
                FROM evento e
                JOIN status s ON e.id_status = s.id
                LEFT JOIN usuario u ON e.id_organizador = u.id
                ORDER BY s.nome, e.titulo)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Evento: ' || rec.evento_titulo || ', Status: ' || rec.status_nome || ', Organizador: ' || NV
                             ', Data Início: ' || NVL(TO_CHAR(rec.data_inicio, 'YYYY-MM-DD HH24:MI'), 'N/A') || ', Data Fi
                             ', Descrição: ' || NVL(rec.descricao, 'N/A') || ', Urgência: ' || rec.urgencia);
    END LOOP;
```

Saída do Script

Tarefa concluída em 0,062 segundos

```
Procedure RELATORIO_EVENTOS_POR_STATUS compilado
```

05.05. TRIGGERS

```sql
-- Criando tabela para registro de auditoria
CREATE TABLE auditoria (
    id NUMBER(9) CONSTRAINT pk_id_auditoria PRIMARY KEY,
    tabela VARCHAR2(255) NOT NULL,
    operacao VARCHAR2(50) NOT NULL,
    usuario VARCHAR2(255) NOT NULL,
    data_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    dados_anteriores CLOB,
    dados_posteriores CLOB
);
```

Saída do Script

Tarefa concluída em 0,054 segundos

```
Table AUDITORIA criado.
```

```sql
-- Trigger para a tabela usuario
CREATE OR REPLACE TRIGGER trg_auditoria_usuario
AFTER INSERT OR UPDATE OR DELETE
ON usuario
FOR EACH ROW
DECLARE
    v_operacao VARCHAR2(10);
    v_dados_anteriores CLOB;
    v_dados_posteriores CLOB;
BEGIN
    IF INSERTING THEN
        v_operacao := 'INSERT';
        v_dados_posteriores := 'ID: ' || :NEW.id || ', CPF: ' || :NEW.cpf || ', Nome: ' || :NEW.nome || ', URL Imagem: ' ||
    ELSIF UPDATING THEN
        v_operacao := 'UPDATE';
        v_dados_anteriores := 'ID: ' || :OLD.id || ', CPF: ' || :OLD.cpf || ', Nome: ' || :OLD.nome || ', URL Imagem: ' ||
        v_dados_posteriores := 'ID: ' || :NEW.id || ', CPF: ' || :NEW.cpf || ', Nome: ' || :NEW.nome || ', URL Imagem: ' ||
    ELSIF DELETING THEN
        v_operacao := 'DELETE';
```

Saída do Script

Tarefa concluída em 0,051 segundos

```
Trigger TRG_AUDITORIA_USUARIO compilado
```

```
-- Trigger para a tabela evento
CREATE OR REPLACE TRIGGER trg_auditoria_evento
AFTER INSERT OR UPDATE OR DELETE
ON evento
FOR EACH ROW
DECLARE
    v_operacao VARCHAR2(10);
    v_dados_anteriores CLOB;
    v_dados_posteriores CLOB;
BEGIN
    IF INSERTING THEN
        v_operacao := 'INSERT';
        v_dados_posteriores := 'ID: ' || :NEW.id || ', Titulo: ' || :NEW.titulo || ', Latitude: ' || :NEW.latitude || ', Longitude
    ELSIF UPDATING THEN
        v_operacao := 'UPDATE';
        v_dados_anteriores := 'ID: ' || :OLD.id || ', Titulo: ' || :OLD.titulo || ', Latitude: ' || :OLD.latitude || ', Longitude:
        v_dados_posteriores := 'ID: ' || :NEW.id || ', Titulo: ' || :NEW.titulo || ', Latitude: ' || :NEW.latitude || ', Longitude
    ELSIF DELETING THEN
        v_operacao := 'DELETE';
```

Saída do Script | Tarefa concluída em 0,059 segundos

```
Trigger TRG_AUDITORIA_EVENTO compilado
```

05.06. PACKAGES

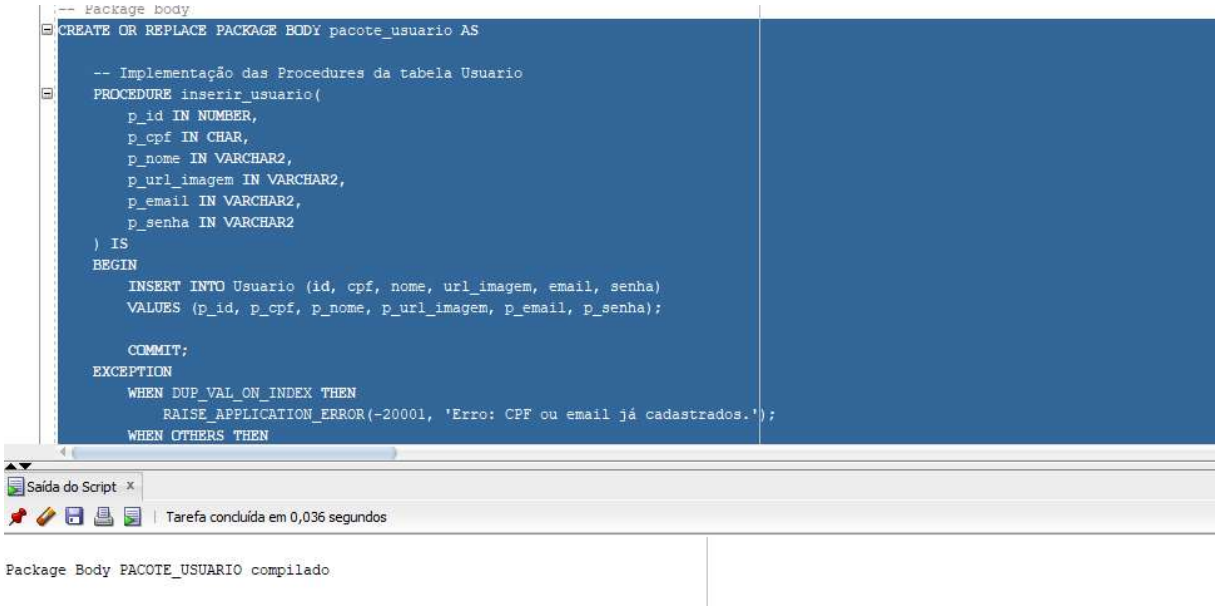
```
-- Declaração das procedures
CREATE OR REPLACE PACKAGE pacote_usuario AS

    -- Procedures da tabela Usuario
    PROCEDURE inserir_usuario(
        p_id IN NUMBER,
        p_cpf IN CHAR,
        p_nome IN VARCHAR2,
        p_url_imagem IN VARCHAR2,
        p_email IN VARCHAR2,
        p_senha IN VARCHAR2
    );

    PROCEDURE atualizar_usuario(
        p_id IN NUMBER,
        p_cpf IN CHAR,
        p_nome IN VARCHAR2,
        p_url_imagem IN VARCHAR2,
        p_email IN VARCHAR2,
        p_senha IN VARCHAR2
```

Saída do Script | Tarefa concluída em 0,035 segundos

```
Package PACOTE_USUARIO compilado
```



```
-- Package Body
CREATE OR REPLACE PACKAGE BODY pacote_usuario AS

    -- Implementação das Procedures da tabela Usuario
    PROCEDURE inserir_usuario(
        p_id IN NUMBER,
        p_cpf IN CHAR,
        p_nome IN VARCHAR2,
        p_url_imagem IN VARCHAR2,
        p_email IN VARCHAR2,
        p_senha IN VARCHAR2
    ) IS
    BEGIN
        INSERT INTO Usuario (id, cpf, nome, url_imagem, email, senha)
        VALUES (p_id, p_cpf, p_nome, p_url_imagem, p_email, p_senha);

        COMMIT;
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            RAISE_APPLICATION_ERROR(-20001, 'Erro: CPF ou email já cadastrados.');
        WHEN OTHERS THEN
```

Saída do Script

Tarefa concluída em 0,036 segundos

```
Package Body PACOTE_USUARIO compilado
```