INSTITUTO SUPERIOR TÉCNICO

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

PROJETO DE BASE DE DADOS – PARTE 2

Grupo 39 | 4ª Feira: 11:00h – 12:30h

Docente: André da Silva Pereira

ALUNO	NÚMERO	HORAS	PERCENTAGEM RELATIVA
Daniel Pereira	89425	9h	33,3(3)%
Tiago Gonçalves	89547	9h	33,3(3)%
Tiago Barroso	89549	9h	33,3(3)%

COMANDOS DE CRIAÇÃO DA BASE DE DADOS:

```
DROP TABLE IF EXISTS local_publico CASCADE;
DROP TABLE IF EXISTS item CASCADE;
DROP TABLE IF EXISTS anomalia CASCADE;
DROP TABLE IF EXISTS anomalia_traducao CASCADE;
DROP TABLE IF EXISTS duplicado CASCADE;
DROP TABLE IF EXISTS utilizador CASCADE;
DROP TABLE IF EXISTS utilizador_qualificado CASCADE;
DROP TABLE IF EXISTS utilizador_regular CASCADE;
DROP TABLE IF EXISTS incidencia CASCADE;
DROP TABLE IF EXISTS proposta_de_correcao CASCADE;
DROP TABLE IF EXISTS correcao CASCADE;
DROP FUNCTION IF EXISTS verificaUtilizador;
DROP FUNCTION IF EXISTS verificaAnomalia;

CREATE FUNCTION verificaUtilizador (emailAVerificar VARCHAR(255), qualificado INTEGER)
RETURNS BOOLEAN
AS
$$
BEGIN
  IF (qualificado = 1 AND EXISTS (SELECT email FROM utilizador_regular U WHERE U.email =
emailAVerificar)) OR (qualificado = 0 AND EXISTS (SELECT email FROM utilizador_qualificado U
WHERE U.email = emailAVerificar)) THEN
        RETURN FALSE;
  ELSE
        RETURN TRUE;
  END IF;
END;
$$
LANGUAGE plpgsql;

CREATE FUNCTION verificaAnomalia (id2 INTEGER, zona2 BOX, lingua2 VARCHAR(255))
RETURNS BOOLEAN
AS
$$
DECLARE z1p1 POINT;
DECLARE z1p2 POINT;
DECLARE z2p1 POINT;
DECLARE z2p2 POINT;
BEGIN
  SELECT zona[0] INTO z1p1 FROM anomalia WHERE id2 = id;
  SELECT zona[1] INTO z1p2 FROM anomalia WHERE id2 = id;
  SELECT zona2[0] INTO z2p1;
  SELECT zona2[1] INTO z2p2;

  IF ((z1p1[0] < z2p2[0] OR z2p1[0] < z1p2[0] OR z1p2[1] > z2p1[1] OR z2p2[1] > z1p1[1])
AND lingua2 <> (SELECT lingua FROM anomalia WHERE id2 = id)) THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
END;
$$
```

```
LANGUAGE plpgsql;

CREATE TABLE local_publico
  (latitude             FLOAT               NOT NULL,
   longitude            FLOAT               NOT NULL,
   nome                 VARCHAR(255)        NOT NULL,
   PRIMARY KEY(latitude, longitude),
   CHECK (latitude <= 90 AND latitude >= -90),
   CHECK (longitude <= 180 AND latitude >= -180));

CREATE TABLE item
  (id                   INTEGER             NOT NULL,
   descricao            VARCHAR(255)        NOT NULL,
   localizacao          VARCHAR(255)        NOT NULL,
   latitude             FLOAT               NOT NULL,
   longitude            FLOAT               NOT NULL,
   PRIMARY KEY(id),
   FOREIGN KEY(latitude, longitude) REFERENCES local_publico(latitude, longitude) ON
DELETE CASCADE);

CREATE TABLE anomalia
  (id                   INTEGER             NOT NULL,
   zona                 BOX                 NOT NULL,
   imagem               BYTEA               NOT NULL,
   lingua               VARCHAR(255)        NOT NULL,
   ts                   TIMESTAMP           NOT NULL,
   descricao            VARCHAR(255)        NOT NULL,
   tem_anomalia_redacao BOOLEAN             NOT NULL,
   PRIMARY KEY(id));

CREATE TABLE anomalia_traducao
  (id                   INTEGER             NOT NULL,
   zona2                BOX                 NOT NULL,
   lingua2              VARCHAR(255)        NOT NULL,
   PRIMARY KEY(id),
   FOREIGN KEY(id) REFERENCES anomalia(id) ON DELETE CASCADE ON UPDATE CASCADE,
   CHECK (verificaAnomalia(id, zona2, lingua2) = TRUE));

CREATE TABLE duplicado
  (item1                INTEGER             NOT NULL,
   item2                INTEGER             NOT NULL,
   PRIMARY KEY(item1, item2),
   FOREIGN KEY(item1) REFERENCES item(id) ON DELETE CASCADE,
   FOREIGN KEY(item2) REFERENCES item(id) ON DELETE CASCADE,
   CHECK (item1 < item2));

CREATE TABLE utilizador
  (email                VARCHAR(255)        NOT NULL,
   password             VARCHAR(255)        NOT NULL,
   PRIMARY KEY(email));

CREATE TABLE utilizador_qualificado
  (email                VARCHAR(255)        NOT NULL,
   PRIMARY KEY(email),
```

```
  FOREIGN KEY(email) REFERENCES utilizador(email) ON DELETE CASCADE,
  CHECK (verificaUtilizador(email, 1) = TRUE));

CREATE TABLE utilizador_regular
 (email                    VARCHAR(255)        NOT NULL,
  PRIMARY KEY(email),
  FOREIGN KEY(email) REFERENCES utilizador(email) ON DELETE CASCADE,
  CHECK (verificaUtilizador(email, 0) = TRUE));

CREATE TABLE incidencia
 (anomalia_id        INTEGER       NOT NULL,
  item_id            INTEGER       NOT NULL,
  email                    VARCHAR(255)        NOT NULL,
  PRIMARY KEY(anomalia_id),
  FOREIGN KEY(anomalia_id) REFERENCES anomalia(id) ON DELETE CASCADE ON UPDATE
CASCADE,
  FOREIGN KEY(item_id) REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY(email) REFERENCES utilizador(email) ON DELETE CASCADE);

CREATE TABLE proposta_de_correcao
 (email                    VARCHAR(255)        NOT NULL,
  nro                      INTEGER             NOT NULL,
  data_hora          TIMESTAMP           NOT NULL,
  texto                    VARCHAR(255)        NOT NULL,
  PRIMARY KEY(email, nro),
  FOREIGN KEY(email) REFERENCES utilizador_qualificado(email) ON DELETE CASCADE);

CREATE TABLE correcao
 (email                    VARCHAR(255)        NOT NULL,
  nro                      INTEGER             NOT NULL,
  anomalia_id        INTEGER       NOT NULL,
  PRIMARY KEY(email, nro, anomalia_id),
  FOREIGN KEY(email, nro) REFERENCES proposta_de_correcao(email, nro) ON DELETE
CASCADE,
  FOREIGN KEY (anomalia_id) REFERENCES incidencia(anomalia_id) ON DELETE
CASCADE ON UPDATE CASCADE);
```

CONSULTAS EM SQL:

```
SELECT nome
FROM local_publico NATURAL JOIN (
   SELECT latitude, longitude
   FROM incidencia INNER JOIN item ON item.id = incidencia.item_id
   GROUP BY latitude, longitude
   HAVING COUNT(incidencia.anomalia_id) = (
          SELECT MAX(anom_count) FROM (
             SELECT latitude, longitude, COUNT(incidencia.anomalia_id) as anom_count
             FROM incidencia INNER JOIN item ON item.id = incidencia.item_id
             GROUP BY latitude, longitude
          ) as x
   )
) as y;

SELECT email
FROM anomalia NATURAL JOIN anomalia_traducao INNER JOIN incidencia ON
anomalia_traducao.id = incidencia.anomalia_id NATURAL JOIN utilizador_regular
WHERE anomalia.ts >= '2019-01-01 00:00:00' AND anomalia.ts <= '2019-06-30 23:59:59'
GROUP BY email
HAVING COUNT(id) = (
   SELECT MAX(anom_count) FROM (
          SELECT email, COUNT(id) AS anom_count
          FROM anomalia NATURAL JOIN anomalia_traducao INNER JOIN incidencia ON
anomalia_traducao.id = incidencia.anomalia_id NATURAL JOIN utilizador_regular
          WHERE anomalia.ts >= '2019-01-01 00:00:00' AND anomalia.ts <= '2019-06-30
23:59:59'
          GROUP BY email
   ) as x
);

SELECT DISTINCT email
FROM incidencia I
WHERE NOT EXISTS (
   (SELECT local_publico.latitude, local_publico.longitude
    FROM anomalia INNER JOIN incidencia ON anomalia.id = anomalia_id INNER JOIN item ON
item.id = item_id RIGHT OUTER JOIN local_publico ON item.latitude = local_publico.latitude AND
item.longitude = local_publico.longitude
    WHERE (local_publico.latitude > (SELECT latitude FROM local_publico WHERE
local_publico.nome = 'Rio Maior')))
    EXCEPT
       (SELECT latitude, longitude
        FROM item INNER JOIN incidencia ON item.id = incidencia.item_id INNER JOIN anomalia
ON anomalia.id = anomalia_id
        WHERE email = I.email AND (ts IS NULL OR DATE_PART('year', ts) = 2019))
);
```

```
SELECT DISTINCT email
FROM (incidencia NATURAL JOIN utilizador_qualificado) A
WHERE EXISTS (
   (SELECT anomalia_id
    FROM anomalia INNER JOIN incidencia ON anomalia.id = incidencia.anomalia_id INNER JOIN
item ON incidencia.item_id = item.id
    WHERE email = A.email AND (item.latitude < (SELECT latitude FROM local_publico WHERE
local_publico.nome = 'Rio Maior') AND DATE_PART('year', ts) = DATE_PART('year',
CURRENT_DATE)))
   EXCEPT
   (SELECT correcao.anomalia_id
    FROM correcao INNER JOIN anomalia ON correcao.anomalia_id = anomalia.id INNER JOIN
incidencia ON anomalia.id = incidencia.anomalia_id INNER JOIN item ON item.id =
incidencia.item_id
    WHERE correcao.email = A.email)
);
```

EXPLICAÇÃO DA ARQUITETURA DA APLICAÇÃO PHP:

A aplicação PHP é inicializada numa página Início que pede ao utilizador para selecionar uma das opções que se encontram no menu no topo da página. Este menu encontra-se em todas as páginas da aplicação e contém opções para todos os tipos de dados que o utilizador tem acesso, mais especificamente Utilizadores (que o utilizador da aplicação pode ver), Locais, Itens, Anomalias (inserir e remover), Correções e Propostas de Correção (inserir, editar e remover), Incidências e Duplicados (inserir só). Também existe a opção de listar os dados correspondentes às alíneas e) e f) de "Desenvolvimento da Aplicação" e uma opção Início que volta para a página inicial.

Todas as opções, à exceção de Início e Listar, abrem uma página com o nome da opção selecionada no topo, seguido de uma zona para a inserção de uma entrada (exceto Utilizadores) e terminando com uma listagem dos dados respectivos à opção que se encontram na base de dados, aparecendo no final de cada linha as opções de Editar ou Remover quando aplicável. A opção de remoção elimina a entrada correspondente na base de dados, atualizando também a lista na aplicação, enquanto que Editar leva o utilizador para uma página onde ele pode alterar os atributos da entrada selecionada.

A zona de introdução de dados contém todos os campos relevantes para a base de dados, com um título e um placeholder para cada campo. Uma mensagem de erro aparece quando as operações de Inserir e Editar dão erro.

A opção Listar abre uma página onde se seleciona entre duas opções que listam dados. Cada opção leva a uma página para inserir os valores relevantes para a query dos dados e depois de inserir todos os valores e clicar no botão verde no final, aparece a lista com os dados pretendidos.

Quanto aos ficheiros PHP, existe um para cada opção no menu, mais dois para as duas opções da página Listar e um com os dados de login da base de dados (settings.php).