**Projeto de Bases de Dados**

**Parte 3**

Grupo 16

Turno BD225179577L05 – 2ªf 14h-15h30

Prof. Duarte Alexandre Galvão

|  |  |  |  |
| --- | --- | --- | --- |
| **Número** | **Nome** | **Esforço (horas)** | **Esforço (%)** |
| 89476 | João Fonseca | 15 | 33 |
| 89544 | Tiago Pires | 15 | 33 |
| 89552 | Tomás Lopes | 15 | 33 |

**Comandos de criação da BD (schema.sql)**

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;

CREATE TABLE local\_publico

(

latitude NUMERIC(8,6) NOT NULL,

longitude NUMERIC(9,6) NOT NULL,

nome VARCHAR(100) NOT NULL,

CONSTRAINT pk\_local\_publico PRIMARY KEY(latitude, longitude),

CONSTRAINT check\_latitude CHECK (latitude>=-90 AND latitude<=90),

CONSTRAINT check\_longitude CHECK (longitude>=-180 AND longitude<=180)

);

CREATE TABLE item

(

id SERIAL NOT NULL UNIQUE,

descricao TEXT NOT NULL,

localizacao VARCHAR(100) NOT NULL,

latitude NUMERIC(8,6) NOT NULL,

longitude NUMERIC(9,6) NOT NULL,

CONSTRAINT pk\_item PRIMARY KEY(id),

CONSTRAINT fk\_item\_local\_publico FOREIGN KEY(latitude, longitude)

REFERENCES local\_publico(latitude, longitude) ON DELETE CASCADE

);

CREATE TABLE anomalia

(

id SERIAL NOT NULL UNIQUE,

zona BOX NOT NULL,

imagem VARCHAR(1000) NOT NULL,

lingua VARCHAR(100) NOT NULL,

ts TIMESTAMP NOT NULL,

descricao TEXT NOT NULL,

tem\_anomalia\_redacao BOOLEAN NOT NULL,

CONSTRAINT pk\_anomalia PRIMARY KEY(id)

);

CREATE TABLE anomalia\_traducao

(

id INTEGER NOT NULL UNIQUE,

zona2 BOX NOT NULL,

lingua2 VARCHAR(100) NOT NULL,

CONSTRAINT pk\_anomalia\_traducao PRIMARY KEY(id),

CONSTRAINT fk\_anomalia\_traducao\_anomalia FOREIGN KEY(id) REFERENCES anomalia(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE duplicado

(

item1 INTEGER NOT NULL,

item2 INTEGER NOT NULL,

CONSTRAINT pk\_duplicado PRIMARY KEY(item1, item2),

CONSTRAINT fk\_duplicado\_item1 FOREIGN KEY(item1)

REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_duplicado\_item2 FOREIGN KEY(item2)

REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT check\_items CHECK (item1<item2)

);

CREATE TABLE utilizador

(

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

CONSTRAINT pk\_utilizador PRIMARY KEY(email),

CONSTRAINT check\_email CHECK (email LIKE '%@%.%')

);

CREATE TABLE utilizador\_qualificado

(

email VARCHAR(100) NOT NULL UNIQUE,

CONSTRAINT pk\_utilizador\_qualificado PRIMARY KEY(email),

CONSTRAINT fk\_utilizador\_qualificado\_utilizador FOREIGN KEY(email)

REFERENCES utilizador(email) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE utilizador\_regular

(

email VARCHAR(100) NOT NULL UNIQUE,

CONSTRAINT pk\_utilizador\_regular PRIMARY KEY(email),

CONSTRAINT fk\_utilizador\_regular\_utilizador FOREIGN KEY(email)

REFERENCES utilizador(email) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE incidencia

(

anomalia\_id INTEGER NOT NULL UNIQUE,

item\_id INTEGER NOT NULL,

email VARCHAR(100) NOT NULL,

CONSTRAINT pk\_incidencia PRIMARY KEY(anomalia\_id),

CONSTRAINT fk\_incidencia\_anomalia FOREIGN KEY(anomalia\_id)

REFERENCES anomalia(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_incidencia\_item FOREIGN KEY(item\_id)

REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_incidencia\_utilizador FOREIGN KEY(email)

REFERENCES utilizador(email) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE proposta\_de\_correcao

(

email VARCHAR(100) NOT NULL,

nro INTEGER NOT NULL,

data\_hora TIMESTAMP NOT NULL,

texto TEXT NOT NULL,

CONSTRAINT pk\_proposta\_de\_correcao PRIMARY KEY(email, nro),

CONSTRAINT fk\_proposta\_de\_correcao\_utilizador\_qualificado FOREIGN KEY(email)

REFERENCES utilizador\_qualificado(email) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE correcao

(

email VARCHAR(100) NOT NULL,

nro INTEGER NOT NULL,

anomalia\_id INTEGER NOT NULL,

CONSTRAINT pk\_correcao PRIMARY KEY(email, nro, anomalia\_id),

CONSTRAINT fk\_correcao\_proposta\_de\_correcao FOREIGN KEY(email, nro)

REFERENCES proposta\_de\_correcao(email,nro) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_correcao\_incidencia FOREIGN KEY(anomalia\_id)

REFERENCES incidencia(anomalia\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

**Consultas em SQL (queries.sql)**

/\*Query #1\*/

SELECT nome

FROM item NATURAL JOIN local\_publico NATURAL JOIN incidencia

WHERE id=item\_id

GROUP BY nome, latitude, longitude

HAVING COUNT(\*)>=ALL (

SELECT COUNT(\*)

FROM item NATURAL JOIN incidencia

WHERE id=item\_id

GROUP BY latitude, longitude

);

/\*Query #2\*/

SELECT email

FROM anomalia NATURAL JOIN anomalia\_traducao,incidencia NATURAL JOIN utilizador\_regular

WHERE ts>='2019-01-01 00:00:00' AND

ts<='2019-06-30 23:59:59' AND

id=anomalia\_id

GROUP BY email

HAVING COUNT(\*)>=ALL (

SELECT COUNT(\*)

FROM anomalia NATURAL JOIN anomalia\_traducao,

incidencia NATURAL JOIN utilizador\_regular

WHERE ts>='2019-01-01 00:00:00' AND

ts<='2019-06-30 23:59:59' AND

id=anomalia\_id

GROUP BY email

);

/\*Query #3\*/

SELECT email

FROM utilizador

WHERE email NOT IN (

SELECT DISTINCT email

FROM (

(SELECT email,latitude,longitude

FROM utilizador,local\_publico

WHERE latitude>39.336775)

EXCEPT

(SELECT DISTINCT email,latitude,longitude

FROM anomalia,incidencia,item NATURAL JOIN local\_publico

WHERE item\_id=item.id AND

anomalia\_id=anomalia.id AND

EXTRACT(year FROM ts)=2019 AND

latitude>39.336775)

) AS R

);

/\*Query #4\*/

SELECT DISTINCT email

FROM (

(SELECT email,anomalia\_id

FROM anomalia,incidencia,item NATURAL JOIN local\_publico

WHERE item.id=item\_id AND

anomalia.id=anomalia\_id AND

latitude<39.336775 AND

EXTRACT(year FROM ts)=EXTRACT(year FROM CURRENT\_DATE) AND

email IN (SELECT email FROM utilizador\_qualificado))

EXCEPT

(SELECT email,anomalia\_id FROM correcao)

) AS R;

**Explicação da arquitetura da aplicação PHP**

A aplicação PHP pode ser iniciada abrindo a página inicial (index.php), que contém *links* para cada uma das seis funcionalidades pedidas em diferentes páginas - por exemplo, alínea a)na página a.php, alínea b) na página b.php, etc.

A primeira página (a.php) contém três tabelas, correspondentes aos **Locais Públicos**, aos **Itens** e às **Anomalias**. É possível apagar uma entrada de qualquer uma das tabelas clicando no botão X da respetiva entrada a apagar, e é possível adicionar uma entrada a cada tabela clicando no botão + debaixo da respetiva tabela e preenchendo o formulário que se segue a esse clique. A página é atualizada depois de cada adição/remoção de entradas. Optou-se por guardar as imagens como forma de referência (em string), sendo que a imagem de exemplo para todas as anomalias está guardada localmente (image.jpeg), funcionando também com links para imagens online, estando apresentada efetivamente como imagem na aplicação e não como string.

\*\*\*\*\*POR FAZER: ALÍNEA b)\*\*\*\*\*

Na terceira página (c.php), são listados os *e-mails* de todos os **Utilizadores** presentes na base de dados, por ordem alfabética.

Na quarta página (d.php), são apresentadas as tabelas referentes às **Incidências** e aos **Duplicados**. É possível adicionar novas entradas a cada uma das tabelas de forma análoga ao que foi explicado para a primeira página.

A quinta página (e.php) redireciona o utilizador para um formulário, que depois de preenchido apresenta os resultados pretendidos, ou seja, lista todas as **Anomalias** de **Incidências** registadas na área compreendida entre os dois **Locais** **Públicos** introduzidos.

Finalmente, a última página (f.php) funciona de forma semelhante à página e.php, listando as **Anomalias** registadas nos últimos três meses a mais ou menos **(dX,dY)** graus de **(lat,long)**, sendo **dX**, **dY**, **lat** e **long** valores introduzidos pelo utilizador.

Em relação a detalhes de implementação, optou-se for usar modificações aos *links* de cada página como forma de obter a ação pretendida pelo utilizador (adicionar, remover, editar…) e os dados preenchidos nos formulários de forma a serem acessados como valores do array $\_GET. Garante-se a atomicidade de operações complexas que envolvam mais do que uma *query* usando *transactions*. Além disso, a base de dados está prevenida contra ataques *SQL INJECTION* devido ao uso de *prepared* *statements* para todas as *queries* executadas.