Projeto de Bases de Dados - Parte 3

Turno L08 - Quarta às 11:00 - André Pereira

Grupo 46

Daniel Gonçalves - 91004

Gabriel Almeida - 89446

Pedro Galhardo - 89522

Nome	Percentagem	Horas
Daniel	33,3%	10
Gabriel	33,3%	10
Pedro	33,3%	10

Comandos de criação da Base de Dados

```
-- Drop all tables
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 all tables
create table local_publico (
    latitude float not null,
    longitude float not null,
    nome varchar(50) not null,
    constraint pk_latitude_longitude primary key(latitude, longitude)
);

create table item (
    id serial,
    descricao varchar(200) not null,
    localizacao varchar(50) not null,
    latitude float not null,
    longitude float not null,
    constraint pk_item_id primary key(id),
    constraint fk_latitude foreign key(latitude, longitude) references
local_publico(latitude, longitude) on delete cascade
);

create table anomalia (
    id serial,
    zona box not null,
    imagem varchar(512) not null,
    lingua varchar(20) not null,
    ts timestamp(0) not null,
    descricao varchar(200) not null,
    tem_anomalia_redacao boolean,
```

```
    constraint pk_anomalia_id primary key(id)
);

create table anomalia_traducao (
    id serial,
    zona2 box not null,
    lingua2 varchar(20) not null,
    constraint pk_anomalia_traducao_id primary key(id),
    constraint fk_anomalia_traducao_id foreign key(id) references anomalia(id) on
delete cascade
);

create table duplicado (
    item1 serial,
    item2 serial check(item1 < item2),
    constraint pk_item_ids primary key(item1, item2),
    constraint fk_item1 foreign key(item1) references item(id) on delete cascade,
    constraint fk_item2 foreign key(item2) references item(id) on delete cascade
);

create table utilizador (
    email varchar(40) not null,
    psw varchar(15) not null,
    constraint pk_utilizador_email primary key(email)
);

create table utilizador_qualificado (
    email varchar(40) not null,
    constraint pk_utilizador_qualificado_email primary key(email),
    constraint fk_utilizador_qualificado_email foreign key(email) references
utilizador(email) on delete cascade
);

create table utilizador_regular (
    email varchar(40) not null,
    constraint pk_utilizador_regular_email primary key(email),
    constraint fk_utilizador_regular_email foreign key(email) references
utilizador(email) on delete cascade
);

create table incidencia (
```

```
    anomalia_id serial,
    item_id serial,
    email varchar(40) not null,
    constraint pk_incidencia primary key(anomalia_id),
    constraint fk_anomalia_id foreign key(anomalia_id) references anomalia(id) on
delete cascade,
    constraint fk_item_id foreign key(item_id) references item(id) on delete cascade,
    constraint fk_incidencia_email foreign key(email) references utilizador(email) on
delete cascade
);

create table proposta_de_correcao (
    email varchar(40) not null,
    nro serial,
    data_hora timestamp not null,
    texto varchar(200) not null,
    constraint pk_email_nro primary key(email, nro),
    constraint fk_proposta_de_correcao_email foreign key(email) references
utilizador_qualificado(email) on delete cascade,
    unique(nro)
);

create table correcao (
    email varchar(40) not null,
    nro integer not null,
    anomalia_id integer not null,
    constraint pk_email_nro_anomalia_id primary key(email, nro, anomalia_id),
    constraint pk_correcao_email foreign key(email, nro) references
proposta_de_correcao(email, nro) on delete cascade,
    constraint pk_correcao_anomalia_id foreign key(anomalia_id) references
incidencia(anomalia_id) on delete cascade
);
```

Consultas SQL

1.
```
SELECT L.nome
FROM incidencia
JOIN item ON incidencia.item_id = item.id
NATURAL JOIN local_publico AS L
GROUP BY L.latitude, L.longitude, L.nome
HAVING COUNT(anomalia_id) >= ALL (
        SELECT COUNT(anomalia_id)
        FROM incidencia
        JOIN item ON incidencia.item_id = item.id
        NATURAL JOIN local_publico AS L
        GROUP BY L.latitude, L.longitude, L.nome
)
```

2.
```
SELECT incidencia.email
FROM anomalia
JOIN incidencia ON anomalia.id = incidencia.anomalia_id
NATURAL JOIN utilizador_regular
WHERE anomalia.ts BETWEEN '2019-01-01 00:00:00' AND '2019-06-30 23:59:59'
GROUP BY incidencia.email
HAVING COUNT(*) >= ALL(
        SELECT COUNT(*)
        FROM anomalia JOIN incidencia ON anomalia.id = incidencia.anomalia_id
        WHERE anomalia.ts BETWEEN '2019-01-01 00:00:00' AND '2019-06-30 23:59:59'
        GROUP BY incidencia.email
)
```

3.
```
SELECT email
FROM incidencia JOIN item ON incidencia.item_id = item.id JOIN anomalia ON incidencia.anomalia_id
= anomalia.id
WHERE item.latitude > 39.336775 AND anomalia.ts BETWEEN '2019-01-01 00:00:00' AND
'2019-12-31 23:59:59'
GROUP BY email
HAVING COUNT( DISTINCT item.latitude, item.longitude) = (
        SELECT COUNT(*)
        FROM local_publico
        WHERE latitude > 39.336775
)
```

4.
```
SELECT X.email
FROM correcao AS C
CROSS JOIN (
        incidencia AS I
        JOIN anomalia AS A ON A.id = I.anomalia_id
        JOIN item ON item.id = I.item_id
) AS X
```

```
WHERE X.latitude < 39.336775 AND X.ts BETWEEN '2019-01-01 00:00:00' AND '2019-12-31
23:59:59'

EXCEPT

SELECT X.email
FROM correcao AS C
CROSS JOIN (
        incidencia AS I
        JOIN anomalia AS A ON A.id = I.anomalia_id
        JOIN item ON item.id = I.item_id
) AS X
WHERE C.anomalia_id = X.anomalia_id AND C.email = X.email
AND X.latitude < 39.336775 AND X.ts BETWEEN '2019-01-01 00:00:00' AND '2019-12-31 23:59:59'
```

Explicação da arquitetura da aplicação PHP

Na nossa aplicação para termos um bom nível de abstração, decidimos criar uma classe BD (db_class.php) onde a mesma realiza todo o tipo de operações diretamente com a base de dados, desde autenticação à realização de uma query, sendo apenas preciso criar uma instância dela nos restantes ficheiros php. Nessa mesma classe, ela obtém os dados para conectar-se à base de dados de um ficheiro chamado db_credentials.php.

Para se aceder à aplicação é necessário um login com credenciais de um utilizador presente na base de dados, mesmo que se tente carregar outra página diferente, se o login não for efetuado ele redireciona para a página de login, para o mesmo ser feito. Já dentro da aplicaçãMo tem 6 opções (presentes na barra de navegação): Home, que tem a opção para repor a base de dados com os ficheiros populate.sql e schema.sql; Inserir, onde permite inserir locais públicos, itens, anomalias, propostas de correção e correções; Editar, onde podemos remover locais, itens, anomalias, propostas de correção, correções e editar proposta de correção; Visualizar, em que permite ao utilizador ver todos os utilizadores (apenas o seu email), visualizar anomalias entre dois locais públicos ou numa vizinhança dando latitude, longitude e desvio; Registar, onde o utilizador pode registar uma incidência ou um item duplicado; e por fim, Logout, onde se pode terminar a sessão da conta ativa no momento.

Dependendo do tipo de utilizador pode-se realizar tarefas diferentes, nomeadamente um utilizador regular não pode efectuar correções, para efeitos de teste pode se usar a conta utilizador:“admin”, palavra-passe:“admin” que é um utilizador qualificado.