Projeto de Base de Dados

Parte 3

Nome	Número	Contribuição	Horas
Maria Inês Morais	83609	33	18
João Antunes	87668	33	18
Viviana Bernardo	87709	33	18

Grupo nº 7

Turno: Terça-feira das 08:30 às 10:00

Docente: Professor Carlos Mendes

Comandos de criação da base de dados

A criação da base de dados foi feita inteiramente de acordo com o Anexo A e tendo em consideração para que o tipo de dados escolhido para cada atributo fosse o mais adequado.

```
DROP TABLE IF EXISTS solicita CASCADE;
DROP TABLE IF EXISTS audita CASCADE;
DROP TABLE IF EXISTS coordenador CASCADE;
DROP TABLE IF EXISTS acciona CASCADE;
DROP TABLE IF EXISTS alocado CASCADE;
DROP TABLE IF EXISTS transporta CASCADE;
DROP TABLE IF EXISTS meioSocorro CASCADE;
DROP TABLE IF EXISTS meioApoio CASCADE;
DROP TABLE IF EXISTS meioCombate CASCADE;
DROP TABLE IF EXISTS meio CASCADE;
DROP TABLE IF EXISTS entidadeMeio CASCADE;
DROP TABLE IF EXISTS eventoEmergencia CASCADE;
DROP TABLE IF EXISTS processoSocorro CASCADE;
DROP TABLE IF EXISTS vigia CASCADE;
DROP TABLE IF EXISTS local CASCADE;
DROP TABLE IF EXISTS segmentoVideo CASCADE;
DROP TABLE IF EXISTS video CASCADE;
DROP TABLE IF EXISTS camara CASCADE;

CREATE TABLE camara(numCamara NUMERIC(4) NOT NULL, CONSTRAINT pk_camara PRIMARY KEY(numCamara));

CREATE TABLE video(dataHoraInicio TIMESTAMP NOT NULL, dataHoraFim TIMESTAMP NOT NULL, numCamara NUMERIC(4) NOT NULL, CONSTRAINT pk_video PRIMARY KEY (dataHoraInicio, numCamara), CONSTRAINT fk_video_camara FOREIGN KEY (numCamara) REFERENCES camara(numCamara) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE segmentoVideo(numSegmento NUMERIC(4) NOT NULL, duracao TIME NOT NULL, dataHoraInicio TIMESTAMP NOT NULL, numCamara NUMERIC(4) NOT NULL, CONSTRAINT pk_segmentoVideo PRIMARY KEY (numSegmento, dataHoraInicio, numCamara), CONSTRAINT fk_segmentoVideo_video FOREIGN KEY (dataHoraInicio, numCamara) REFERENCES video(dataHoraInicio, numCamara) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE local(moradaLocal VARCHAR(50) NOT NULL, CONSTRAINT pk_local PRIMARY KEY(moradaLocal));

CREATE TABLE vigia(moradaLocal VARCHAR(50) NOT NULL, numCamara numeric(4) NOT NULL, CONSTRAINT pk_vigia PRIMARY KEY(moradaLocal, numCamara), CONSTRAINT fk_vigia_local FOREIGN KEY(moradaLocal) REFERENCES local(moradaLocal) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_vigia_camara FOREIGN KEY(numCamara) REFERENCES camara(numCamara) ON DELETE CASCADE ON UPDATE CASCADE);
```

CREATE TABLE processoSocorro(numProcessoSocorro NUMERIC(9) NOT NULL, CONSTRAINT pk_processoSocorro PRIMARY KEY(numProcessoSocorro));

CREATE TABLE eventoEmergencia(numTelefone NUMERIC(9) NOT NULL, instanteChamada TIMESTAMP NOT NULL, nomePessoa VARCHAR(70) NOT NULL, moradaLocal VARCHAR(50) NOT NULL, numProcessoSocorro NUMERIC(9), CONSTRAINT pk_eventoEmergencia PRIMARY KEY (numTelefone, instanteChamada), CONSTRAINT fk_eventoEmergencia_local FOREIGN KEY (moradaLocal) REFERENCES local(moradaLocal) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_eventoEmergencia_processoSocorro FOREIGN KEY (numProcessoSocorro) REFERENCES processoSocorro(numProcessoSocorro) ON DELETE CASCADE ON UPDATE CASCADE, UNIQUE(numTelefone,nomePessoa));

CREATE TABLE entidadeMeio(nomeEntidade VARCHAR(50) NOT NULL, CONSTRAINT pk_entidadeMeio PRIMARY KEY (nomeEntidade));

CREATE TABLE meio(numMeio NUMERIC(4) NOT NULL, nomeMeio VARCHAR(50) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, CONSTRAINT pk_meio PRIMARY KEY (numMeio,nomeEntidade), CONSTRAINT fk_meio_entidadeMeio FOREIGN KEY (nomeEntidade) REFERENCES entidadeMeio(nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE meioCombate(numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, CONSTRAINT pk_meioCombate PRIMARY KEY (numMeio, nomeEntidade), CONSTRAINT fk_meioCombate_meio FOREIGN KEY (numMeio, nomeEntidade) REFERENCES meio(numMeio, nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE meioApoio(numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, CONSTRAINT pk_meioApoio PRIMARY KEY (numMeio, nomeEntidade), CONSTRAINT fk_meioApoio_meio FOREIGN KEY (numMeio, nomeEntidade) REFERENCES meio(numMeio, nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE meioSocorro(numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, CONSTRAINT pk_meioSocorro PRIMARY KEY (numMeio, nomeEntidade), CONSTRAINT fk_meioSocorro_meio FOREIGN KEY (numMeio, nomeEntidade) REFERENCES meio(numMeio, nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE transporta(numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, numVitimas NUMERIC(12) NOT NULL, numProcessoSocorro NUMERIC(9) NOT NULL, CONSTRAINT pk_transporta PRIMARY KEY (numMeio, nomeEntidade, numProcessoSocorro), CONSTRAINT fk_transporta_meioSocorro FOREIGN KEY (numMeio, nomeEntidade) REFERENCES meioSocorro(numMeio, nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_transporta_processoSocorro FOREIGN KEY (numProcessoSocorro) REFERENCES processoSocorro(numProcessoSocorro) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE alocado (numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, numHoras TIME NOT NULL, numProcessoSocorro NUMERIC(9) NOT NULL, CONSTRAINT pk_alocado PRIMARY KEY (numMeio, nomeEntidade, numProcessoSocorro), CONSTRAINT fk_alocado_meioApoio FOREIGN KEY (numMeio, nomeEntidade) REFERENCES

meioApoio(numMeio, nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_alocado_processoSocorro FOREIGN KEY (numProcessoSocorro) REFERENCES processoSocorro(numProcessoSocorro) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE acciona(numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, numProcessoSocorro NUMERIC(9) NOT NULL, CONSTRAINT pk_acciona PRIMARY KEY (numMeio, nomeEntidade, numProcessoSocorro), CONSTRAINT fk_acciona_meio FOREIGN KEY (numMeio, nomeEntidade) REFERENCES meio(numMeio, nomeEntidade) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_acciona_processoSocorro FOREIGN KEY (numProcessoSocorro) REFERENCES processoSocorro(numProcessoSocorro) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE coordenador(idCoordenador NUMERIC(3) NOT NULL, CONSTRAINT pk_coordenador PRIMARY KEY (idCoordenador));

CREATE TABLE audita(idCoordenador NUMERIC(3) NOT NULL, numMeio NUMERIC(4) NOT NULL, nomeEntidade VARCHAR(50) NOT NULL, numProcessoSocorro NUMERIC(9) NOT NULL, datahoraInicio TIMESTAMP NOT NULL, datahoraFim TIMESTAMP NOT NULL, dataAuditoria DATE NOT NULL, texto VARCHAR(255) NOT NULL, CONSTRAINT pk_audita PRIMARY KEY (idCoordenador, numMeio, nomeEntidade, numProcessoSocorro), CONSTRAINT fk_audita_acciona FOREIGN KEY (numMeio, nomeEntidade, numProcessoSocorro) REFERENCES acciona(numMeio, nomeEntidade, numProcessoSocorro) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_audita_coordenador FOREIGN KEY (idCoordenador) REFERENCES coordenador(idCoordenador) ON DELETE CASCADE ON UPDATE CASCADE, CHECK (datahoraInicio < datahoraFim));

CREATE TABLE solicita(idCoordenador NUMERIC(3) NOT NULL, dataHoraInicioVideo TIMESTAMP NOT NULL, numCamara NUMERIC(4) NOT NULL, dataHoraInicio TIMESTAMP NOT NULL, dataHoraFim TIMESTAMP NOT NULL, CONSTRAINT pk_solicita PRIMARY KEY (idCoordenador, dataHoraInicioVideo, numCamara), CONSTRAINT fk_solicita_coordenador FOREIGN KEY (idCoordenador) REFERENCES coordenador(idCoordenador) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_solicita_video FOREIGN KEY (dataHoraInicioVideo, numCamara) REFERENCES video(dataHoraInicio, numCamara) ON DELETE CASCADE ON UPDATE CASCADE);

Consultas em SQL

1. Qual é o processo de socorro que envolveu maior número de meios distintos.

```
SELECT numProcessoSocorro
FROM acciona
GROUP BY numProcessoSocorro
HAVING COUNT(numMeio) >= ALL(SELECT COUNT(numMeio)
                             FROM acciona
                             GROUP BY numProcessoSocorro);
```

2. Qual a entidade fornecedora de meios que participou em mais processos de socorro no Verão de 2018.

```
SELECT nomeEntidade
FROM acciona NATURAL JOIN eventoEmergencia
WHERE instanteChamada BETWEEN '2018-06-21 00:00:00' AND '2018-09-23 23:59:59' GROUP
BY nomeEntidade
HAVING COUNT(DISTINCT numProcessoSocorro) >= ALL(
        SELECT COUNT(DISTINCT numProcessoSocorro)
        FROM acciona NATURAL JOIN eventoEmergencia
        WHERE instanteChamada BETWEEN '2018-06-21 00:00:00' AND '2018-09-23 23:59:59'
        GROUP BY nomeEntidade);
```

3. Quais são os processos de socorro, referente a eventos de emergência em 2018 de Oliveira do Hospital, onde existe pelo menos um acionamento de meios que não foi alvo de auditoria.

```
SELECT DISTINCT numProcessoSocorro
FROM eventoEmergencia NATURAL JOIN acciona
WHERE moradaLocal = 'Oliveira do Hospital' AND
    instanteChamada BETWEEN '2018-01-01 00:00:00' AND '2018-12-31 23:59:59'
    AND (numProcessoSocorro, numMeio, nomeEntidade) NOT IN (
    SELECT numProcessoSocorro,numMeio,nomeEntidade
    FROM audita NATURAL JOIN eventoEmergencia
    WHERE morada = 'Oliveira do Hospital' AND instanteChamada BETWEEN '2018-01-01
    00:00:00' AND '2018-12-31 23:59:59');
```

4. Quantos segmentos de vídeo com duração superior a 60 segundos, foram gravados em câmeras de vigilância de Monchique durante o mês de Agosto de 2018.

```
SELECT COUNT(numSegmento)
FROM vigia NATURAL JOIN segmentoVideo NATURAL JOIN video
WHERE duracao > '00:00:60' AND moradaLocal = 'Monchique' AND dataHoraInicio >= '2018-08-
01 00:00:00' AND dataHoraFim <= '2018-08-31 23:59:59';
```

5. Liste os Meios de combate que não foram usados como Meios de Apoio em nenhum processo de socorro.

```
(SELECT numMeio, nomeEntidade FROM meioCombate)
EXCEPT
(SELECT numMeio, nomeEntidade FROM acciona NATURAL JOIN meioApoio);
```

6. Liste as entidades que forneceram meios de combate a todos os Processos de socorro que acionaram meios.

```
SELECT nomeEntidade
FROM meioCombate NATURAL JOIN acciona
GROUP BY nomeEntidade
HAVING COUNT (DISTINCT numProcessoSocorro) =
                                SELECT COUNT (DISTINCT numProcessoSocorro)
                                FROM acciona;
```

Aplicação PHP e Relações entre ficheiros

A nossa aplicação tem como estrutura um ficheiro index.html que informa o utilizador das diferentes ações que pode realizar na base de dados e a partir do qual o redirecionamos para um dos nossos ficheiros php, consoante a ação que este pretenda efetuar.

Os seis pontos pedidos para o desenvolvimento da aplicação foram feitos segundo a base de dados criada a partir dos ficheiros schema.sql e populate.sql.

1. *Inserir e remover Locais, Eventos de Emergência, Processos de Socorro, Meios e Entidades*.

Para inserir e remover os elementos na base de dados usamos os ficheiros inserir_local.php, inserir_evento.php, inserir_ processo.php, inserir_ meio.php, inserir_entidade.php, remover_entidade.php, remover_eventoEmergencia.php, remover_local.php, remover_meio.php e remover_processo.php que realizam a interação com o utilizador onde pedimos os valores necessários para efetuar as ação pretendida.

Para executar as alterações na base de dados utilizamos os ficheiros que têm o nome igual aos anteriores mas que começam com update.

Na inserção de locais, meios e entidades basta nos apenas inserir nas respectivas tabelas os atributos necessários. Aquando da inserção de um evento de emergência já é necessário verificar se o local e o processo de socorro atribuidos existem na base de dados, e se não, cria-los. Durante a inserção de um processo de socorro devido à restrição de integridade que dita que "todo o processo de socorro tem de estar associado a um ou mais evento de emergência" é necessário pedir ao utilizador que insira um evento de emergência a associar a este processo.

Na remoção de locais, eventos de emergência, processos de socorro, meios e entidades são apagadas as entradas das respetivas tabelas indicadas pelo utilizador e as entradas das tabelas que contêm uma foreign key para as tabelas anteriormente apagadas. Adicionalmente quando é removido um local ou um evento de emergência é ainda feita uma verificação para identificar processos de socorro que tenham ficado sem eventos de emergência associados e, caso isso se verifique, remover esses processos.

2. *Inserir, editar e remover Meios de Combate, Meios de Apoio, e Meios de Socorro.*

Para inserir, remover e editar os diferentes meios na base de dados usamos os ficheiros inserir_meioCombate.php, inserir_meioApoio.php, inserir_meioSocorro.php, remover_meioCombate.php, remover_meioApoio.php, remover_meioSocorro.php e editar_meio.php que realizam a interação com o utilizador onde pedimos os valores necessários para efetuar a ação pretendida.

Para executar a inserção e remoção na base de dados utilizamos os ficheiros que têm o nome igual aos anteriores mas que começam com update.

Na inserção de meios de apoio, meios de combate e meios de socorro é adicionada à respetiva tabela os respetivos atributos inseridos pelo utilizador. Para além disso é também feita uma inserção na tabela "meio" com os mesmos atributos.

Na remoção de meios de apoio, meios de combate e meios de socorro é apagada a entrada da respetiva tabela e feita uma verificação para confirmar se o meio que foi apagado existe como outro tipo de meio, se não existir, a mesma entrada da tabela "meio" é apagada.

Na edição de meios o atributo “nomeMeio” da tabela “meio” é atualizado com o novo nome inserido pelo utilizador e é inserida uma nova entrada nas tabela “meioApoio”, “meioCombate” e “meioSocorro” consoante o utilizador queira editar o meio.

3. *Listar Processos de Socorro e Meios.*

Para listar os Processos de Socorro e os Meios utilizamos os ficheiros listar_processo.php e listar_meio.php, onde efetuamos a ligação ao sistema PostgreSQL e executamos a query "SELECT numprocessosocorro FROM processoSocorro;" para o primeiro e a query "SELECT nummeio, nomemeio, nomeentidade FROM meio;" para o segundo.

Seguidamente construimos uma tabela onde representamos o resultado da query, percorrendo esses resultados um a um.

4. *Associar Processos de Socorro a Meios e Processos de Socorro a Eventos de Emergência.*

Para associar Processos de Socorro a Meios e Processos de Socorro a Eventos de Emergência na base de dados, usamos os ficheiros assoc_meio_processo.php e assoc_evento_processo.php, respetivamente, que realizam a interação com o utilizador onde pedimos os valores necessários para efetuar a ação pretendida.

Para executar as alterações na base de dados utilizamos os ficheiros update_assoc_meio_processo.php e update_assoc_evento_processo.php, onde efetuamos a ligação ao sistema PostgreSQL e iniciamos uma transação fazendo *$db->beginTransaction()*.

Para associarmos processos de socorro a meios inserimos os respetivos atributos do meio e do processo de socorro na tabela acciona. Para associarmos processos de socorro a eventos de emergência atualizamos o numero do processo de socorro do evento de emergência dado.

No final fazemos *$db->commit();* para alterar permanentemente os valores na nossa base de dados.

5. *Listar os Meios acionados num processo de socorro.*

Para listar os meios acionados num processo de socorro usamos o ficheiro listar_meios_accionados.php onde realizamos a interação com o utilizador e pedimos o número do processo de socorro.

De seguida, para listar os meios corretos, utilizamos o ficheiro update_listar_meios_accionados.php onde selecionamos os meios da tabela “acciona” com o numero de processo de socorro dado.

6. *Listar os meios de Socorro acionados em processos de socorro originados num dado local de incêndio.*

Para listar os meios de Socorro acionados em processos de socorro originados num dado local de incêndio usamos o ficheiro listar_meios_local.php onde realizamos a interação com o utilizador e pedimos o nome do local de incêndio.

De seguida, para listar os meios de Socorro corretos, utilizamos o ficheiro update_listar_meios_local.php onde efetuamos operações análogas às do ponto 5 exceptuando a execução da query que passa a ser "SELECT nummeio, nomeentidade FROM eventoEmergencia NATURAL JOIN (SELECT numMeio,nomeEntidade, numProcessoSocorro FROM meioSocorro NATURAL JOIN acciona) AS meiosAccionados WHERE moradaLocal = '$moradaLocal';”.