EDB: Especificação da base de dados

Neste segundo componente querem apresentar como o nosso projeto se vai agrupar e como certos modelos e esquemas vão ser realizados pelo nosso grupo.
Vamos realizar um modelo conceitual de dados (A4), um esquema relacional, validação e refinamento desse esquema (A5) e vamos indicar indexes, triggers e os dados da base de dados.

A4: Modelo de dados conceptuais

O Modelo de Domínio Conceitual contém a identificação e descrição das entidades do domínio e os relacionamentos entre elas em um diagrama de classes UML. A figura 1 representa as principais entidades organizacionais, as relações entre elas, os atributos e a multiplicidade de relações para o nosso projeto, "O Ardina".

1. Class diagram

> 

Figura 1: Diagrama UML

2. Additional Business Rules

Regras e restrições adicionais que não podem ser transmitidas no diagrama de classes UML.

- **BR01**- Utilizador não pode fazer report de elementos seus (próprio utilizador, notícias ou comentários) ;
- **BR02**- Utilizador não pode reagir (colocar gosto ou não) nos seus elementos (notícias ou comentários) ;
- **BR03**- Utilizador não pode colocar nos favoritos algo que ele é proprietário (tag não tem proprietário) ;
- **BR04**- Utilizador não se pode seguir a ele próprio ;
- **BR05**- Só o utilizador com permissão 'a' (Administrador) pode inserir FAQ, eliminar qualquer elemento do site (utilizador, notícia, comentário e tag) ;

A5: Esquema Relacional, validação e refinação do esquema

Este artefacto contém o esquema relacional obtido pelo mapeamento do modelo de dados conceptuais descrito no A4. O esquema relacional inclui o esquema de relação, atributos, domínios, PRIMARY KEY, FOREIGN KEY e outras regras de integridade tais como: UNIQUE, DEFAULT, NOT NULL, CHECK.

1. Esquema Relacional

Esquemas relacionais estão em especificados em notação compacta:

(Como este relatório está em formato Markdown, a ferramenta para sublinhar não funciona, então para as chaves primárias colocamos em negrito e todas as regras colocamos em capslock.)

Relação de referência	**Notação compacta da referência**
R01	utilizador(**id,** nome UK NN, email UK NN, password NN, foto, permissao NN CK permissao IN USER_TYPE, contacto NN)
R02	noticia(**id,** autor_id -> utilizador, titulo NN, descricao NN, data NN DF Today)
R03	tag(**id,** nome NN UK, prioridade NN CK prioridade IN PRIORIDADE)
R04	comentario(**id**, autor_id -> utilizador, not_id -> noticia NN, comentario NN, data NN DF Today)
R05	categoria(**not_id** -> noticia NN**, tag_id** -> tag NN)
R06	faq(**id**, autor_id -> utilizador, questao NN UK, resposta NN)
R07	imagem(**id,** legenda NN, path NN, not_id -> noticia NN)
R08	votCom(**id,** com_id -> comentario NN, autor_id -> utilizador, tipo NN CK tipo IN GOSTO)
R09	votNot(**id,** not_id -> noticia NN, autor_id -> utilizador, tipo NN CK tipo IN GOSTO)
R10	favNot(**autor_id** -> utilizador**, not_id** -> noticia NN)
R11	favCom(**autor_id** -> utilizador**, comentario_id** -> comentario NN)
R12	favTag(**autor_id** -> utilizador**, tag_id** -> tag NN)
R13	nSeguidor(**followed_id** -> utilizador, **infos_id** -> utilizador, lido NN DF FALSE, data NN DF Today)
R14	nVotNot(**uti_id** -> utilizador, **voto_id** -> votNot NN, lido NN DF FALSE, data NN DF Today)
R15	nVotCom(**uti_id** -> utilizador, **votoc_id** -> vot_com NN, lido NN DF FALSE, data NN DF Today)
R16	nComentario(**authornot_id** -> utilizador, **com_id** -> comentario NN, lido NN DF FALSE, data NN DF Today)
R17	nUtiBloq(**id**, bloq_id -> utilizador, motivo NN, lido NN DF FALSE, data NN DF Today)
R18	reportN(**id,** autor_id -> utilizador, not_id -> noticia NN, tiporesp_id -> textoReport NN, resolvido NN DF FALSE)
R19	reportC(**id,** autor_id -> utilizador, comentario_id -> comentario NN, tiporesp_id -> textoReport NN, resolvido NN DF FALSE)
R20	reportU(**id,** autor_id -> utilizador, uti_id -> utilizador, tiporesp_id -> textoReport NN, resolvido NN DF FALSE)
R21	textoReport(**id**, report NN)

Relação de referência	Notação compacta da referência
R22	infoSeguidor(**followed_id** -> utilizador, **infos_id** -> utilizador)
R23	publicidade(**id**, imagem NN)

Nota: UK significa UNIQUE, NN significa NOT NULL, CK significa CHECK.

1.1. Domínios

USER_TYPE	ENUM ('a', 'u', 'b')
PRIORIDADE	ENUM('1', '2')
LIDO	ENUM('lido', 'nao lido')
GOSTO	ENUM('like', 'dislike')

1.2. Validação de Esquema

Para validar o Esquema Relacional obtido do Modelo Conceitual, todas as dependências funcionais são identificadas e a normalização de todos os esquemas de relação é realizada.

Tabela R01 (utilizador)	
Keys : {id}, {email}, {nome}	
Dependências Funcionais	
FD0101	{id} -> {nome, email, password, foto, permissao, contacto}
FD0102	{email} -> {id, nome, password, foto, permissao, contacto}
FD0103	{nome} -> { id, email, password, foto, permissao, contacto}
Forma Normal	BCNF

Tabela R02 (noticia)	
Keys: {id}	
Dependências Funcionais	
FD0201	{id} -> {autor_id, titulo, descricao, data}
Forma Normal	BCNF

Tabela R03 (tag)	
Keys: {id}, {nome}	
Dependências Funcionais	
FD0301	{id} -> {nome, prioridade}
FD0302	{nome} -> {id, prioridade}

Tabela R03 (tag)	
Forma Normal	BCNF
Tabela R04 (comentario)	
Keys: {id}	
Dependências Funcionais	
FD0401	{id} -> {autor_id, not_id, comentario,data}
Forma Normal	BCNF
Tabela R05 (categoria)	
Keys: {not_id, tag_id}	
Dependências Funcionais	
FD0501	none
Forma Normal	BCNF
Tabela R06 (faq)	
Keys: {id}, {questao}	
Dependências Funcionais	
FD0601	{id} -> {autor_id, questao, resposta}
FD0602	{questao} -> {id, autor_id, resposta}
Forma Normal	BCNF
Tabela R07 (imagem)	
Keys: {id}	
Dependências Funcionais	
FD0701	{id} -> {legenda, path, not_id}
Forma Normal	BCNF
Tabela R08 (votCom)	
Keys: {id}, {com_id}	
Dependências Funcionais	
FD0801	{id} -> {com_id, autor_id, tipo}
FD0802	{com_id} -> {id, autor_id, tipo}
Forma Normal	BCNF
Tabela R09 (votNot)	
Keys: {id}, {not_id}	

Tabela R09 (votNot)	
Dependências Funcionais	
FD0901	{id} -> {not_id, autor_id, tipo}
FD0902	{not_id} -> {id, autor_id, tipo}
Forma Normal	BCNF

Tabela R10 (favNot)	
Keys: {autor_id, not_id}	
Dependências Funcionais	
FD1001	none
Forma Normal	BCNF

Tabela R11 (favCom)	
Keys: {autor_id, com_id}	
Dependências Funcionais	
FD1101	none
Forma Normal	BCNF

Tabela R12 (favTag)	
Keys: {autor_id, tag_id}	
Dependências Funcionais	
FD1201	none
Forma Normal	BCNF

Tabela R13 (nSeguidor)	
Keys: {users_id, follow_id}	
Dependências Funcionais	
FD1301	{users_id, follow_id) -> {notificacao}
Forma Normal	BCNF

Tabela R14 (nVotNot)	
Keys: {users_id, voto_id}	
Dependências Funcionais	
FD1401	{users_id, voto_id} -> {notificacao}
Forma Normal	BCNF

Tabela R15 (nVotCom)

Tabela R15 (nVotCom)	
Keys: {users_id, votoc_id}	
Dependências Funcionais	
FD1501	{users_id, votoc_id} -> {notificacao}
Forma Normal	BCNF

Tabela R16 (nComentario)	
Keys: {authornot_id, com_id}	
Dependências Funcionais	
FD1601	{authornot_id, com_id} -> {notificacao}
Forma Normal	BCNF

Tabela R17 (nUtiBloq)	
Keys: {id}, {bloq_id}	
Dependências Funcionais	
FD1701	{id} -> {bloq_id, motivo, lido}
FD1702	{bloq_id} -> {id, motivo, lido}
Forma Normal	BCNF

Tabela R18 (reportN)	
Keys: {id}, {not_id}	
Dependências Funcionais	
FD1801	{id} -> {autor_id, not_id, tiporesp_id, resolvido}
FD1802	{not_id} -> {id, , autor_id, tiporesp_id, resolvido}
Forma Normal	BCNF

Tabela R19 (reportC)	
Keys: {id}, {comentario_id}	
Dependências Funcionais	
FD1901	{id} -> {autor_id, comentario_id, tiporesp_id, resolvido}
FD1902	{comentario_id} -> {id, autor_id, tiporesp_id, resolvido}
Forma Normal	BCNF

Tabela R20 (reportU)	
Keys: {id}, {users_id}	
Dependências Funcionais	

Tabela R20 (reportU)	
FD2001	{id} -> {autor_id, users_id, tiporesp_id, resolvido}
FD2002	{users_id} -> {id, autor_id, tiporesp_id, resolvido}
Forma Normal	BCNF

Tabela R21 (textoReport)
Keys: {id}
Dependências Funcionais
FD2101 {id} -> {report}
Forma Normal BCNF

Tabela R22 (infoSeguidor)
Keys: {users_id, follow_id}
Dependências Funcionais
FD2201 none
Forma Normal BCNF

Tabela R23 (publicidade)
Keys: {id}
Dependências Funcionais
FD2301 {id} -> {imagem}
Forma Normal BCNF

Como todas as relações estão na forma normal de Boyce-Codd (BCNF), o esquema relacional também está no BCNF e, portanto, não há necessidade de ser definido usando a normalização.

A6: Índices, integridade e base de dados preenchida

Este artefacto contém:

- Uma previsão da carga de trabalho da base de dados;
- A identificação e caracterização dos índices;
- O modelo de dados físicos da base de dados;
- O suporte de regras de integridade com triggers;
- Transições da base de dados, necessárias para garantir a integridade dos dados na presença de acessos simultâneos;
- A definição de funções na base de dados definidas pelo utilizador;
- Código SQL necessário para a criação da base de dados, tal como para a definição de todas as restrições de integridade, índices e triggers.
- Código SQL para a povoação da base de dados.

1. Carga de trabalho da Base de Dados

É fundamental quantificarmos a utilização que a nossa aplicação irá ter para que dessa maneira sejamos capazes de construir uma base de dados adequada a essa carga.

Relação	Nome da relação	Ordem de magnitude	Crescimento estimado
R01	Utilizador	10k (dezenas de milhares)	10 (dezenas) / dia
R02	Noticia	100k (centenas de milhares)	10 (dezenas) / dia
R03	Tag	1k (milhares)	1 (unidades) / dia
R04	Comentario	1M (milhões)	1k (milhares) / dia
R05	VotN	10M (dezenas de milhões)	100k (centenas de milhares) / dia
R06	VotC	1M (milhões)	1k (milhares) / dia
R07	FavNoticia	10k (dezenas de milhares)	100 (centenas) / dia
R08	FavComentario	10k (dezenas de milhares)	100 (centenas) / dia
R09	FavTag	10k (dezenas de milhares)	100 (centenas) / dia
R10	Seguir	1M (milhões)	100 (centenas) / dia
R11	Not_Tags	10M (dezenas de milhões)	100 (centenas) / dia
R12	Not_Imagens	1M (milhões)	100 (centenas) / dia
R13	Noti_Seguidor	1M (milhões)	100 (centenas) / dia
R14	Noti_Comentario	100k (centenas de milhares)	100 (centenas) / dia
R15	Noti_VotN	10M (dezenas de milhões)	100k (centenas de milhares) / dia
R16	Notif_VotC	1M (milhões)	1k (milhares) / dia
R17	Noti_UtiBloq	1k (milhares)	10 (dezenas) / dia
R18	Report_N	1M (milhões)	100 (centenas) / dia
R19	Report_C	100k (centenas de milhares)	100 (centenas) / dia
R20	Report_U	10k (dezenas de milhares)	10 (dezenas) / dia
R21	TextoReport	1M (milhões)	1k (milhares) / dia
R22	FAQ	10 (dezenas)	1 / dia

2 - Indices Propostos

Índices são usados para reduzir os tempos de acesso numa base de dados, melhorando assim a sua performance. Apesar das vatagens, índices consomem recursos consideráveis, sendo assim necessário usa-los com algum cuidado.

2.1 Indices de Perfomance

Index	**ID01**
Relação	Users
Atributo	nome
Tipo	hash
Clustering	Não
Justificação	Obter a informação de um utilizador é executado várias vezes por isso tem que ser rápido; não precisa de suporte para consulta de intervalo; a cardinalidade é alta porque o nome de utilizador é uma chave exclusiva e, portanto, não é um bom candidato para cluster.

```
CREATE INDEX username_users_idx ON utilizador USING hash (nome);
```

Index	**ID02**
Relação	noticia
Atributo	data
Tipo	hash
Clustering	Yes
Justificação	Para permitir a busca das primeiras 25 notícias de uma determinada data que tenham a data inferior a um determinado valor mais rapidamente; É B-tree e agrupado para permitir consultas de alcance rápido. A cardinalidade é alta porque a data pode ter vários valores

```
CREATE INDEX noticias_date_idx ON noticia USING btree(data);
```

Index	**ID03**
Relação	noticia
Atributo	autor_id
Tipo	hash
Clustering	Yes
Justificação	O atributo autor_id é usado em várias consultas, portanto, é um bom candidato para a criação de um índice. A cardinalidade é média e, como um autor pode ter vários conteúdos, é um bom candidato para agrupamento.

```
CREATE INDEX autor_noticias_idx ON noticia USING hash(autor_id);
```

Index	**ID04**
Relação	comentario
Atributo	not_id
Tipo	hash

Index	ID04
Clustering	Não
Justificação	Obter os comentários de uma notícia é executado várias vezes, por isso tem de ser rápido. A cardinalidade é alta porque a notícia_id é uma chave exclusiva e, portanto, não é bom candidato para cluster.

```
CREATE INDEX comentarios_idx ON comentario USING hash(not_id)
```

2.2 - Indices de pesquisa Full-Text

Index	ID05
Relação	noticia
Atributo	title
Tipo	GIST
Clustering	Não
Justificação	Usado para melhorar o desempenho de pesquisas de texto completo ao pesquisar notícias, GIST porque as notícias são dados dinâmicos.

```
ALTER TABLE noticia ADD COLUMN search TSVECTOR;

CREATE INDEX search_noticias_idx ON noticia USING GIST (search);
```

Index	ID06
Relação	utilizador
Atributo	nome
Tipo	GIN
Clustering	Não
Justificação	Usado para melhorar o desempenho de pesquisas de texto completo ao pesquisar utilizadores, porque os dados dos utilizadores raramente são alterados.

```
ALTER TABLE utilizador ADD COLUMN search TSVECTOR;

CREATE INDEX search_users_idx ON utilizador USING GIST (search);
```

3.TRIGGERS

Trigger	TRIGGER01
Descrição	Acão de administrador
Justificação	Como algumas funcionalidades como bloquear users ou atualizar faqs são apenas permitidas administradores, este trigger é relevante.

```
CREATE  OR  REPLACE  FUNCTION acao_de_admin() RETURNS TRIGGER AS
$BODY$
BEGIN

IF  NOT (SELECT permissao FROM utilizador WHERE USER_TYPE =  'u' ) THEN
RAISE EXCEPTION 'Apenas administradores podem realizar esta operação.';
END  IF;
RETURN utilizador;

END

$BODY$

LANGUAGE plpgsql;



CREATE TRIGGER acao_de_admin

BEFORE  UPDATE

FOR EACH ROW

EXECUTE  PROCEDURE acao_de_admin();
```

Trigger	TRIGGER02
Descrição	Seguir próprio utilizador
Justificação	Deverá não ser possível para um utilizador se seguir a si próprio.

```
CREATE  OR  REPLACE  FUNCTION seguir_proprio() RETURNS TRIGGER AS

$BODY$

BEGIN

IF  infoSeguidor.autor_id  =  infoSeguidor.followed_id  THEN

RAISE EXCEPTION 'Nao se pode seguir a si próprio.';

END  IF;

RETURN infoSeguidor;

END

$BODY$
```

```
LANGUAGE plpgsql;


CREATE TRIGGER seguir_proprio

BEFORE  INSERT  ON infoSeguidor

FOR EACH ROW

EXECUTE  PROCEDURE seguir_proprio();
```

Trigger	TRIGGER03
Descrição	Notificação de novo seguidor
Justificação	Um user é notificado quando um outro utilizador segue a sua conta.

```
CREATE  OR  REPLACE  FUNCTION notificacao_seguidor(followed_id, infos_id)
RETURNS TRIGGER AS

$BODY$

BEGIN

INSERT  INTO nSeguidor

VALUES (followed_id, infos_id);


RETURN nSeguidor;

END

$BODY$

LANGUAGE plpgsql;


CREATE TRIGGER notificacao_seguidor

AFTER  INSERT  ON infoSeguidor

FOR EACH ROW

EXECUTE  PROCEDURE notificacao_seguidor();
```

Trigger	TRIGGER04

Trigger	TRIGGER04
Descrição	Notificação de comentário
Justificação	Um user é notificado quando um outro utilizador comenta a sua noticia.

```
CREATE  OR  REPLACE  FUNCTION notificacao_comentario(u_id,c_id) RETURNS
TRIGGER AS

$BODY$

BEGIN

INSERT  INTO nComentario

VALUES(u_id, c_id);

RETURN nComentario;

END

$BODY$

LANGUAGE plpgsql;


CREATE TRIGGER notificacao_comentario

AFTER  INSERT  ON comentario

FOR EACH ROW

EXECUTE  PROCEDURE notificacao_comentario();
```

Trigger	TRIGGER05
Descrição	Notificação de voto
Justificação	Um user é notificado quando um outro utilizador vota na sua noticia.

```
CREATE  OR  REPLACE  FUNCTION notificacao_voto(voto) RETURNS TRIGGER AS

$BODY$

BEGIN

INSERT  INTO nVotNot

SELECT  v.autor_id, v.id
```

```
FROM VotNot v

WHERE  v.id=voto;

RETURN nVotNot;

END

$BODY$

LANGUAGE plpgsql;


CREATE TRIGGER notificacao_voto

AFTER  INSERT  ON VotNot

FOR EACH ROW

EXECUTE  PROCEDURE notificacao_voto();
```

Trigger	TRIGGER06
Descrição	Atualização do feed
Justificação	Trigger usado sempre que a tabela de notícias é a taulizada. Ordena-as por ordem alfabetica

```
CREATE  OR  REPLACE  FUNCTION atualizar_feed() RETURNS TRIGGER AS

$BODY$

BEGIN

SELECT  *

FROM noticia

ORDER BY  data  DESC;

END

$BODY$

LANGUAGE plpgsql;


CREATE TRIGGER atualizar_feed
```

```
AFTER  UPDATE  ON noticia

FOR EACH ROW

EXECUTE  PROCEDURE atualizar_feed();
```

4. Transactions

Transações são usadas de forma a garantir a integridade dos dados quando estes são inseridos em simultâneo.

T01	Criação de Notícia
Justificação	Para manter a consistência do conteúdo, usamos uma transação para garantir que não haja vários conteúdos sendo inseridos ao mesmo tempo. Se ocorrer um erro, um ROLLBACK é emitido.
Nível de Isolamento	Leitura repetível.

```
BEGIN TRANSACTION;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

INSERT INTO noticia (autor_id, titulo, descricao, data)

VALUES ($autor_id, $titulo, $descricao, $data);

COMMIT;
```

T02	Inserir comentário
Justificação	Para manter a consistência do conteúdo, usamos uma transação para garantir que não haja vários conteúdos sendo inseridos ao mesmo tempo. Se ocorrer um erro, um ROLLBACK é emitido.
Nível de Isolamento	Leitura repetível.

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

INSERT INTO noticia (autor_id, titulo, descricao, data)

VALUES($autor_id, $titulo, $descricao, $data)

INSERT INTO comentario (autor_id, not_id, comentario)

VALUES($autor_id, $not_id, $comentario)

COMMIT;
```

T03	Reportar User
Justificação	Para manter a consistência do conteúdo, usamos uma transação para garantir que não haja vários conteúdos sendo inseridos ao mesmo tempo. Se ocorrer um erro, um ROLLBACK é emitido.
Nível de Isolamento	Leitura repetível.

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ:

INSERT INTO reportU (autor_id, users_id, tiporesp_id, resolvido)

VALUES($autor_id, $users_id, $tiporesp_id, $resolvido)

COMMIT;
```

T04	Reportar Noticia
Justificação	Para manter a consistência do conteúdo, usamos uma transação para garantir que não haja vários conteúdos sendo inseridos ao mesmo tempo. Se ocorrer um erro, um ROLLBACK é emitido.
Nível de Isolamento	Leitura repetível.

```
BEGIN TRANSACTION

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ:

INSERT INTO report_n (autor_id, not_id, tiporesp_id)

VALUES($autor_id, $not_id, $descricao, $data);

COMMIT;
```

T05	Reportar Comentário
Justificação	Para manter a consistência do conteúdo, usamos uma transação para garantir que não haja vários conteúdos sendo inseridos ao mesmo tempo. Se ocorrer um erro, um ROLLBACK é emitido.
Nível de Isolamento	Leitura repetível.

```
BEGIN TRANSACTION

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ:

INSERT INTO reportC (autor_id, comentario_id, tiporesp_id)

VALUES($autor_id, $comentario_id, $descricao, $data);

COMMIT;
```

Annex A. SQL Code

The database scripts are included in this annex to the EBD component.

The database creation script and the population script should be presented as separate elements. The creation script includes the code necessary to build (and rebuild) the database. The population script includes an amount of tuples suitable for testing and with plausible values for the fields of the database.

This code should also be included in the group's git repository and links added here.

A.1. Database schema

```
/* DROPs */
DROP TABLE IF EXISTS publicidade CASCADE;
DROP TABLE IF EXISTS reportC CASCADE;
DROP TABLE IF EXISTS reportN CASCADE;
DROP TABLE IF EXISTS reportU CASCADE;
DROP TABLE IF EXISTS textoReport CASCADE;
DROP TABLE IF EXISTS nUtiBloq CASCADE;
DROP TABLE IF EXISTS nVotCom CASCADE;
DROP TABLE IF EXISTS nVotNot CASCADE;
DROP TABLE IF EXISTS nComentario CASCADE;
DROP TABLE IF EXISTS nSeguidor CASCADE;
DROP TABLE IF EXISTS infoSeguidor CASCADE;
DROP TABLE IF EXISTS favTag CASCADE;
DROP TABLE IF EXISTS favCom CASCADE;
DROP TABLE IF EXISTS favNot CASCADE;
DROP TABLE IF EXISTS votNot CASCADE;
DROP TABLE IF EXISTS votCom CASCADE;
DROP TABLE IF EXISTS imagem CASCADE;
DROP TABLE IF EXISTS faq CASCADE;
DROP TABLE IF EXISTS categoria CASCADE;
DROP TABLE IF EXISTS comentario CASCADE;
DROP TABLE IF EXISTS tag CASCADE;
DROP TABLE IF EXISTS noticia CASCADE;
DROP TABLE IF EXISTS utilizador CASCADE;

DROP TYPE IF EXISTS USER_TYPE;
DROP TYPE IF EXISTS PRIORIDADE;
DROP TYPE IF EXISTS GOSTO;

DROP INDEX IF EXISTS nome_uti_idx;
DROP INDEX IF EXISTS noticia_date_idx;
DROP INDEX IF EXISTS autor_not_idx;
DROP INDEX IF EXISTS comentario_idx;
DROP INDEX IF EXISTS search_not_idx;
DROP INDEX IF EXISTS search_uti_idx;

-----------------------------------------
```

```
/* TIPOS */
CREATE TYPE USER_TYPE AS ENUM('a','u','b'); /* a - Administrador, u - User
Autenticado, b - User Banido*/
CREATE TYPE PRIORIDADE AS ENUM('0', '1', '2');
CREATE TYPE GOSTO AS ENUM('like', 'dislike');

-----------------------------------------

/* TABELAS */

/* T: dos Utilizadores*/
CREATE TABLE utilizador(
    id SERIAL PRIMARY KEY,
    nome VARCHAR(20) NOT NULL UNIQUE,
    email TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL,
    foto TEXT, /*url to text*/
    permissao USER_TYPE NOT NULL,
    contacto INTEGER NOT NULL,
    CONSTRAINT contacto_limites CHECK (contacto > 99999999 AND contacto <
1000000000)
);
/* T: das Noticias*/
CREATE TABLE noticia (
    id SERIAL PRIMARY KEY,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    titulo VARCHAR(90) NOT NULL,
    descricao TEXT NOT NULL,
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL
);

/* T: das Tags*/
CREATE TABLE tag (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(40) NOT NULL UNIQUE,
    prioridade PRIORIDADE NOT NULL
);
/* T: dos Comentários (Utilizador <-> Noticia)*/
CREATE TABLE comentario (
    id SERIAL PRIMARY KEY,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    not_id INTEGER NOT NULL REFERENCES noticia(id) ON DELETE CASCADE,
    texto TEXT NOT NULL,
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL
);
/* T: dos Categoria (Noticia <-> Tag)*/
CREATE TABLE categoria (
    not_id INTEGER NOT NULL REFERENCES noticia(id) ON DELETE CASCADE,
    tag_id INTEGER NOT NULL REFERENCES tag(id) ON DELETE CASCADE,
    PRIMARY KEY (not_id, tag_id)
);
/* T: dos FAQ (-> Utilizador)*/
CREATE TABLE faq (
    id SERIAL PRIMARY KEY,
```

```
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL, /*O
administrador que realizou ou alterou o faq*/
    questao TEXT NOT NULL UNIQUE,
    resposta TEXT NOT NULL
);
/* T: das Imagens (-> Noticia)*/
CREATE TABLE imagem (
    id SERIAL PRIMARY KEY,
    legenda VARCHAR(30) NOT NULL,
    path TEXT NOT NULL,
    not_id INTEGER NOT NULL REFERENCES noticia(id) ON DELETE CASCADE
);

/* T: Votaçao nos Comentários (Utilizador <-> Comentario)*/
CREATE TABLE votCom (
    id SERIAL PRIMARY KEY,
    com_id INTEGER NOT NULL REFERENCES comentario(id) ON DELETE CASCADE,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    tipo GOSTO NOT NULL
);

/* T: Votaçao nas Noticias (Utilizador <-> Noticia)*/
CREATE TABLE votNot (
    id SERIAL PRIMARY KEY,
    not_id INTEGER NOT NULL REFERENCES noticia(id) ON DELETE CASCADE,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    tipo GOSTO NOT NULL
);

/* T: Favoritos Noticias (Utilizador <-> Noticia)*/
CREATE TABLE favNot (
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    not_id INTEGER NOT NULL REFERENCES noticia(id) ON DELETE CASCADE,
    PRIMARY KEY(autor_id, not_id)
);

/* T: Favoritos Comentários (Utilizador <-> Comentario)*/
CREATE TABLE favCom (
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    com_id INTEGER NOT NULL REFERENCES comentario(id) ON DELETE CASCADE,
    PRIMARY KEY(autor_id, com_id)
);

/* T: Favoritos Tags (Utilizador <-> Tag)*/
CREATE TABLE favTag (
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    tag_id INTEGER NOT NULL REFERENCES tag(id) ON DELETE CASCADE,
    PRIMARY KEY(autor_id, tag_id)
);

/* T: Seguidores (Utilizador <-> Utilizador)*/
CREATE TABLE infoSeguidor (
    id SERIAL PRIMARY KEY,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
```

```
    followed_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL
);

/* T: Notificações Novos Seguidores (-> infoSeguidor)*/
CREATE TABLE nSeguidor (
    followed_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    infos_id INTEGER REFERENCES infoSeguidor(id)  ON DELETE SET NULL,
    lido BOOLEAN NOT NULL DEFAULT FALSE,
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    PRIMARY KEY(followed_id, infos_id)
);

/* T: Notificações Novos Comentários (-> Comentario)*/
CREATE TABLE nComentario (
    authornot_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    com_id INTEGER NOT NULL REFERENCES comentario(id) ON DELETE CASCADE,
    lido BOOLEAN NOT NULL DEFAULT FALSE,
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    PRIMARY KEY(authornot_id, com_id)
);

/* T: Notificações Votos Noticia (-> votNot)*/
CREATE TABLE nVotNot (
    uti_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    voto_id INTEGER NOT NULL REFERENCES vot_not(id) ON DELETE CASCADE,
    lido BOOLEAN NOT NULL DEFAULT FALSE,
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    PRIMARY KEY(uti_id, voto_id)
);

/* T: Notificações Votos Comentario (-> votCom)*/
CREATE TABLE nVotCom (
    uti_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    votoc_id INTEGER NOT NULL REFERENCES vot_com(id) ON DELETE CASCADE,
    lido BOOLEAN NOT NULL DEFAULT FALSE,
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    PRIMARY KEY(uti_id, votoc_id)
);

/* T: Notificações Utilizador Bloqueado (UreportU)*/
CREATE TABLE nUtiBloq (
    id SERIAL PRIMARY KEY,
    bloq_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    motivo TEXT NOT NULL,
    lido BOOLEAN NOT NULL DEFAULT FALSE
    data TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL
);

/* T: Possiveis Reports */
CREATE TABLE textoReport (
    id SERIAL PRIMARY KEY,
    report TEXT NOT NULL
);
/* T: Report Utilizador (Utilizador-> textoReport)*/
```

```
CREATE TABLE reportU (
    id SERIAL PRIMARY KEY,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    uti_id INTEGER REFERENCES utilizador(id)  ON DELETE SET NULL,
    tiporesp_id INTEGER NOT NULL REFERENCES textoReport(id)  ON DELETE
CASCADE,
    resolvido BOOLEAN NOT NULL DEFAULT FALSE
);

/* T: Report Noticia (Utilizador-> textoReport <-> Noticia)*/
CREATE TABLE reportN (
    id SERIAL PRIMARY KEY,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    not_id INTEGER NOT NULL REFERENCES noticia(id) ON DELETE CASCADE,
    tiporesp_id INTEGER NOT NULL REFERENCES textoReport(id) ON DELETE
CASCADE,
    resolvido BOOLEAN NOT NULL DEFAULT FALSE
);
/* T: Report Comentario (Utilizador-> textoReport <-> Comentario)*/
CREATE TABLE reportC (
    id SERIAL PRIMARY KEY,
    autor_id INTEGER REFERENCES utilizador(id) ON DELETE SET NULL,
    comentario_id INTEGER NOT NULL REFERENCES comentario(id) ON DELETE
CASCADE,
    tiporesp_id INTEGER NOT NULL REFERENCES textoReport(id) ON DELETE
CASCADE,
    resolvido BOOLEAN NOT NULL DEFAULT FALSE
);

/* T: Publicidade*/
CREATE TABLE publicidade (
    id SERIAL PRIMARY KEY,
    imagem TEXT NOT NULL
);

-----------------------------------------

/* Índices */

CREATE INDEX nome_uti_idx ON utilizador USING hash (nome);

CREATE INDEX noticia_date_idx ON noticia USING btree(data);

CREATE INDEX autor_not_idx ON noticia USING hash(autor_id);

CREATE INDEX comentario_idx ON comentario USING hash(not_id);

ALTER TABLE noticia ADD COLUMN search TSVECTOR;
CREATE INDEX search_not_idx ON noticia USING GIST (search);

ALTER TABLE utilizador ADD COLUMN search TSVECTOR;
CREATE INDEX search_uti_idx ON utilizador USING GIST (search);

-----------------------------------------------------------
```

A.2. Database population

INSERT INTO utilizador (nome,email,password,foto,contacto)

VALUES

('nec,','diam.nunc@hotmail.couk','semper','/src/photos/test.jpg','722325481'),

('Nunc','lacinia@icloud.edu','Lorem','/src/photos/test.jpg','303987142'),

('tortor,','lacus.ut.nec@aol.org','Nullam','/src/photos/test.jpg','550016487'),

('fringilla','aptent.taciti@outlook.net','senectus','/src/photos/test.jpg','817282354'),

('mauris','ligula@protonmail.org','nunc','/src/photos/test.jpg','667894517');

UPDATE utilizador /*Criar um administrador*/

SET permissao = 'a'

WHERE uti_id = 2;

UPDATE utilizador /*Criar um user banido*/

SET permissao = 'b';

WHERE uti_id = 5;

INSERT INTO noticia(autor_id,titulo,descricao)

VALUES

(3,'a felis','Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sed tortor. Integer aliquam adipiscing lacus. Ut nec urna et arcu imperdiet ullamcorper. Duis at lacus. Quisque purus sapien, gravida non, sollicitudin a, malesuada id, erat. Etiam vestibulum massa rutrum magna. Cras convallis convallis dolor. Quisque tincidunt pede ac urna. Ut tincidunt vehicula risus. Nulla eget metus eu erat semper rutrum. Fusce dolor quam, elementum at, egestas a, scelerisque sed, sapien. Nunc pulvinar arcu et pede. Nunc sed orci lobortis augue scelerisque mollis. Phasellus libero mauris, aliquam eu, accumsan sed, facilisis'),

(3,'pretium neque. Morbi','Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sed tortor. Integer aliquam adipiscing lacus. Ut nec urna et arcu imperdiet ullamcorper. Duis at lacus. Quisque purus sapien, gravida non, sollicitudin a, malesuada id, erat. Etiam vestibulum massa rutrum magna. Cras convallis convallis dolor. Quisque tincidunt pede ac urna. Ut tincidunt vehicula risus. Nulla eget metus eu erat semper rutrum. Fusce dolor quam, elementum at, egestas a, scelerisque sed, sapien. Nunc pulvinar'),

(2,'egestas blandit. Nam nulla','Lorem ipsum dolor sit'),

(4,'Class aptent taciti sociosqu','Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sed tortor. Integer aliquam adipiscing lacus. Ut nec urna et arcu imperdiet ullamcorper. Duis at lacus. Quisque purus sapien, gravida non, sollicitudin a, malesuada id, erat. Etiam vestibulum massa rutrum magna. Cras

convallis convallis dolor. Quisque tincidunt pede ac urna. Ut tincidunt vehicula risus. Nulla eget metus eu erat semper rutrum. Fusce dolor quam, elementum at, egestas a, scelerisque'),

(2,'massa. Mauris','Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sed tortor. Integer aliquam adipiscing lacus. Ut nec urna et arcu imperdiet ullamcorper. Duis at lacus. Quisque purus sapien, gravida non, sollicitudin a, malesuada id, erat. Etiam vestibulum massa rutrum magna. Cras convallis convallis dolor. Quisque tincidunt pede ac urna. Ut tincidunt vehicula risus. Nulla eget metus eu erat semper rutrum. Fusce dolor quam, elementum at, egestas a, scelerisque sed, sapien. Nunc pulvinar arcu et pede. Nunc sed');

UPDATE noticia

SET titulo = 'Ola Pessoa'

WHERE not_id = 3;

INSERT INTO tag(nome)

VALUES ('Desporto'), ('Nacional'), ('Local'), ('Mundo');

INSERT INTO tag(nome, prioridade)

VALUES ('FCPorto',2);

DELETE FROM tag

WHERE tag_id = 2;

INSERT INTO comentario (autor_id,not_id,texto)

VALUES

(3,3,'mi felis, adipiscing fringilla, porttitor'),

(3,4,'Nulla tempor augue ac ipsum. Phasellus vitae mauris'),

(1,4,'nibh. Phasellus nulla. Integer vulputate,'),

(1,1,'Morbi non sapien molestie orci'),

(1,3,'turpis vitae');

INSERT INTO categoria(not_id, tag_id)

VALUES (1,2), (2,1), (3,1), (4,3), (5,4);

INSERT INTO infoSeguidor(autor_id, followed_id)

VALUES (1,3),(3,2),(4,1),(3,1);

INSERT INTO textoReport(report)

VALUES ('O utilizador foi agressivo com outro e além de ser agressivo tentou ameaçar diversos utilizadores');

INSERT INTO reportU(autor_id, uti_id, tiporesp_id)

VALUES (3,2,1);

INSERT INTO faq(autor_id, question, resposta)

VALUES (2, 'Como inserir uma notícia ?', 'Nulla tempor augue ac ipsum. Phasellus vitae mauris');

Histórico de Revisão

Em relação à primeira submissão do trabalho (ER), fizemos as seguintes alterações:

1. Acrescentamos a informação inicial relativa ao projeto;
2. Modificamos a representação do Administrador no esquema de atores;
3. Acrescentamos a descrição do Administrador que nos faltava;

Grupo 63, Data : 30/11/2021

- António Ferreira Cabral de Barbosa Campelo, up201704987@fc.up.pt
- Edgar Miguel Pinto Lourenço, up201604910@fc.up.pt
- Manuel da Silva Sá, up201805273@fc.up.pt
- Patrícia Daniela Tavares Vieira, up201805238@fc.up.pt