Universidade de Coimbra
Faculdade de Ciências e Tecnologias
Departamento de Engenharia Informática
2012/2013
Base de dados
Relatório do projecto final

Relatório do Projecto

TxuxRum

Bernardo Marques	basimoes@student.dei.uc.pt	2010139226
Humberto Alves	hjalves@student.dei.uc.pt	2007108323
Tiago Levita	tlevita@student.dei.uc.pt	2009119525

Indíce

Sumário

O presente documento visa relatar resumidamente o decorrer do projecto de base de dados, contendo análises sobre os procedimentos, resultados e distribuição de trabalho.

O documento encontra-se dividido em 3 grandes partes, sendo a primeira o plano de desenvolvimento, a segunda os diagramas relacional e físico da base de dados, juntamente com a estimação do tamanho desta para um tempo conhecido e a terceira os *scripts* de criação da base de dados.

A auto-apreciação dos resultados obtidos e da execução e planeamento do trabalho é bastante positiva e serve-lhe de base os dados que irão ser apresentados ao longo deste documento.

Plano de desenvolvimento

Descrição do trabalho

O presente projecto visa implementar uma base de dados segundo um modelo relacional que sirva de base a um sistema de troca de mensagens, dados e relações entre utilizadores.

Problemas de concorrência

Os problemas de concorrência serão devidos a:
Mensagens no mesmo instante e na mesma sala por utilizadores diferentes
Ratings no mesmo instante ao mesmo tópico por utilizadores distintos
Apagar mensagens privadas que ainda não foram lidas pelo destinatário.

Funcionalidades a implementar

Gestão de utilizadores (registo, login, permissões)
Gestão de perfis (privacidade, inserir, remover e editar dados pessoais)
Motor de pesquisa de utilizadores
Relações entre utilizadores
Troca e gestão de mensagens de texto com ficheiros anexados
Gestão de TxuxRum's (criação, alteração, mensagens, gestão de utilizadores)
Sistema de Rating de TxuxRum's.
Interface gráfico

Plataforma de desenvolvimento e limitações

A aplicação será desenvolvida em plataforma web, de forma a tornar o projecto o mais próximo da realidade possível, sendo que não existem limitações impostas por este tipo de plataforma dado o âmbito do projecto.

Principais Tecnologias

Apache Server
PHP (código do lado do servidor)
PostgreSQL (motor de base de dados)
HTML, Javascript e CSS(código do lado do cliente)

Distribuição de esforço

Planeamento

3-10-2011	Bernardo Marquesscripts	plano de desenvolvimento	5-10-2012	1 dia
4-10-2011	Humberto Alves	plano de desenvolvimento (distribuição de esforço)	5-10-2012	1 dia
4-10-2011	Tiago Levita	plano de desenvolvimento (distribuição de esforço)	5-10-2012	1 dia
11-10-2011	Bernardo Marques	Esboço dos diagramas relacional e fisico, esboço da estimação do tamanho da base de dados	14-10-2012	2 dias
17-10-2011	Humberto Alves	Finalização dos diagramas, melhoria da estimação	19-10-2012	1 dia
17-10-2011	Tiago Levita	Finalização dos diagramas, melhoria da estimação	19-10-2012	1 dia
17-10-2011	Bernardo Marques	Finalização dos diagramas, melhoria da estimação	19-10-2012	1 dia

Execução do projecto

27-9-2012	Tiago Levita	estrutura das páginas html com php e css	24-10-2012	9 dias
6-11-2012	Tiago Levita	login/logout	6-11-2012	1 dia
8-11-2012	Tiago Levita	grande restruturação do código html, php e css. Implementação da edição de perfil	12-11-2012	5 dias
13-11-2012	Tiago Levita	paging e alguma reestruturação de código	13-11-2012	1 dia
13-11-2012	Tiago Levita	Implementou mensagens privadas com ajax e filtros de mensagens por remetente	18-11-2012	5 dias
14-12-2012	Tiago Levita	relatório	14-12-2012	1 dia

				22 dias
2-11-2012	Humberto Alves	estruturação da base de dados, ligação do php com a base de dados, mostrar mensagens, autenticação, mostrar utilizadores, tópicos	6-11-2012	4 dias
7-11-2012	Humberto Alves	correcção de bugs e melhorias, mostrar tópicos com variada informação	10-11-2012	4 dias
12-11-2012	Humberto Alves	querys SQL com várias finalidades, paging, descrição nas chatrooms, correção de bugs	13-11-2012	2 dias
20-11-2012	Humberto Alves	permissoes no geral, e alguma reestruturação de código	28-11-2012	6 dias
7-12-2012	Humberto Alves	funcionalidades dentro das chatrooms, anexação de ficheiros, melhorias nos triggers	11-12-2012	4 dias
14-12-2012	Humberto Alves	relatório	14-12-2012	1 dia
				21 dias
6-11-2012	Bernardo Marques	pesquisa de utilizadores, criação de posts	6-11-2012	1 dia
8-11-2012	Bernardo Marques	registo e pesquisa de utilizadores	9-11-2012	2 dias
13-11-2012	Bernardo Marques	trigger para criação de posts e ratings	13-11-2012	1 dia
15-11-2012	Bernardo Marques	algumas querys sanitized, postpone das mensagens privadas	16-11-2012	2 dias

11-12-2012	Bernardo Marques	actualização do trigger associado aos ratings, apagar conta, apagar mensagens privadas que ainda não foram lidas, privacidade	13-12-2012	3 dias
13-12-2012	Bernardo Marques	relatório	14-12-2012	2 dias
				11 dias

Análise da distribuição de esforço

Como se pode ver pela contagem do esforço empregue por cada um dos membros do grupo no projecto, todos os elementos tiveram uma participação importante ao longo da execução do trabalho.

É também de acrescentar que a contagem aqui feita se baseia em períodos em que os elementos fizeram “commits” para o repositório do projecto relacionados com o tema em questão e não no tempo que dedicaram a cada parte, sendo que possivelmente poderão existir falhas de precisão nos dados obtidos, no entanto, cremos que se trata de uma métrica aceitável.

Diagrama Conceptual

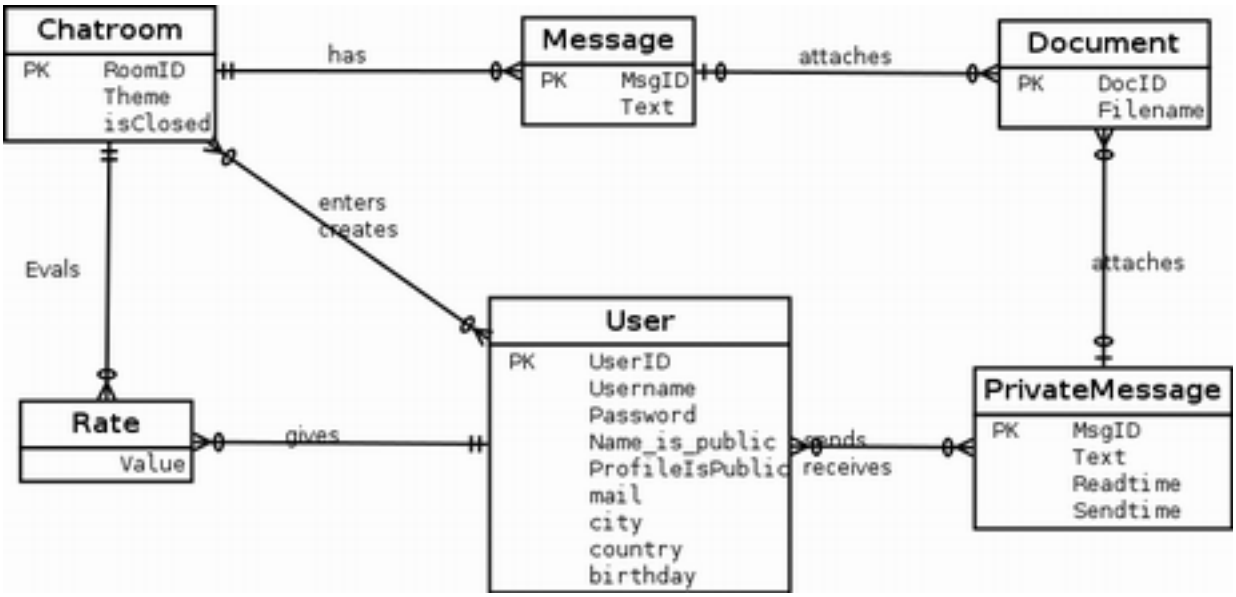
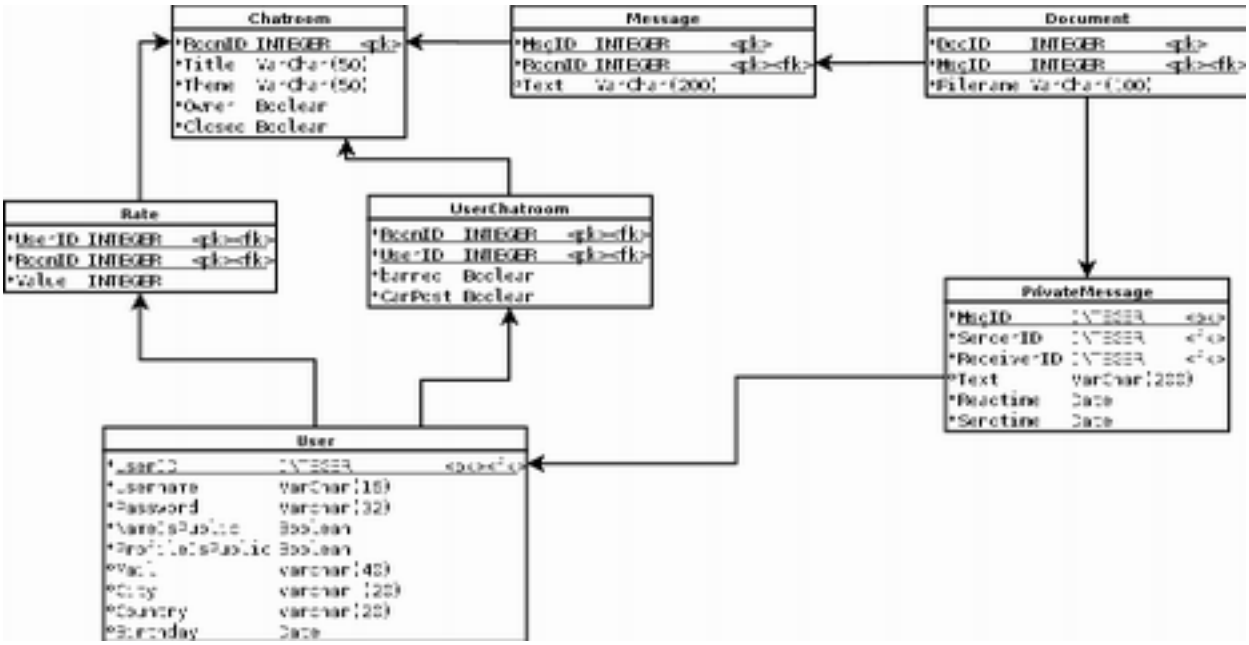


Diagrama Físico



Estimação do tamanho da base de dados

De forma a estimar o tamanho da base de dados vamos considerar que o número de utilizadores da aplicação vai crescer ao longo do tempo, inicialmente de forma rápida e depois de forma cada vez mais lenta, à semelhança da função logarítmica, deslocada uma unidade para a esquerda no eixo dos xx.

Através da análise da estrutura da base de dados, podemos concluir que esta cresce quando:
É criado um utilizador (*user*)
É criada uma chatroom (*room*)
É criada uma mensagem na chatroom (*msg*)
É criada uma mensagem privada (*pvtmsg*)
As permissões da chatroom são alteradas (*perm*)
É feito um rate à chatroom (*rate*)
É anexado um documento (*doc*)

Analisando as relações entre as diferentes operações conclui-se que todas elas se podem representar em relação ao número de utilizadores.
Consideremos então:
O número de utilizadores segue um crescimento logarítmico [log(t)]
Cada utilizador tem um período médio de actividade de 60 dias
Cada utilizador cria 1 room a cada 2 dias [0.5*log(t)]
Cada chatroom tem em média 4 mensagens por dia [2*log(t)]
Cada chatroom tem um período médio de actividade de 3 dias
Cada utilizador envia em média 2 mensagens privadas p/ dia [2*log(t)]
O número máximo de rates é o nº de rooms * nº de users [log^2(t)*0.5]
É anexado em média 1 documento a cada 4 dias por utilizador [log(t)*0.25]

Ou seja:

log(t)*user_size + nau*0.5*room_size + nar*4*msg_size +
nau*2*pvtmsg_size + nar*log(t)*rate_size + nau*0.25*doc_size

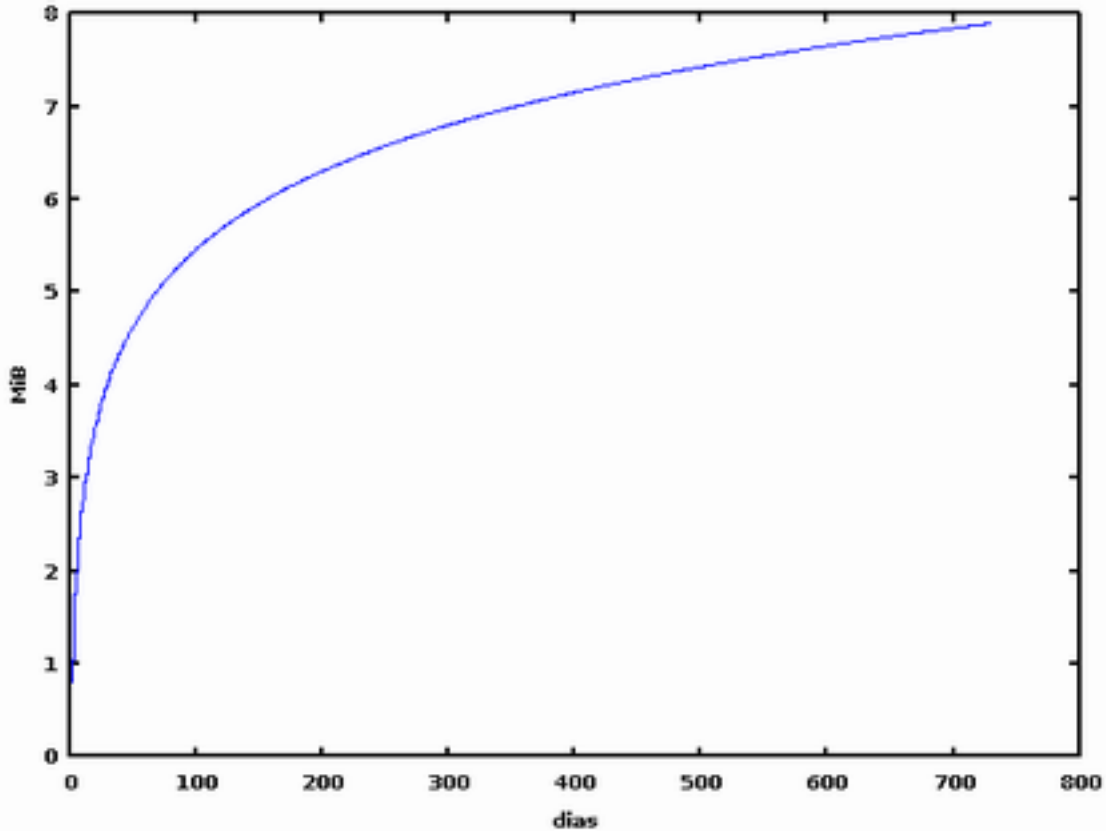
legenda: nau → nº de utilizadores: log(t);
nar → nº de rooms: 0.5*log(t)

Sabemos que:

user_size = 160B
room_size = 120B
msg_size=228B
pvtmsg_size=244B
rate_size = 12B

doc_size = 92B

O que resulta no seguinte gráfico:



Scripts de criação da base de dados

```
--Create Tables
CREATE TABLE users (
   UserID          serial      PRIMARY KEY,
   Username        varchar(16) UNIQUE NOT NULL,
   Password        varchar(32) NOT NULL,
   Deleted         boolean     DEFAULT FALSE,
   UsernamePublic  boolean     DEFAULT TRUE,
   ProfilePublic   boolean     DEFAULT TRUE,
   Name            varchar,
   Male            boolean,
   Mail            varchar,
   Location        varchar,
   Birthday        date,
   CHECK(UserID > 0)
);

CREATE TABLE chatrooms (
   RoomID          serial      PRIMARY KEY,
   OwnerID         integer     REFERENCES users (UserID),
   Title           varchar     NOT NULL,
   Closed          boolean     DEFAULT FALSE,
   CreationDate    timestamp   DEFAULT current_timestamp,
   Description     varchar,
   PostingPerm     boolean     DEFAULT TRUE,
   ReadingPerm     boolean     DEFAULT TRUE
);

CREATE TABLE permissions (
   UserID          integer     REFERENCES users,
   RoomID          integer     REFERENCES chatrooms,
   CanPost         boolean,
   CanRead         boolean,
   PRIMARY KEY (UserID, RoomID)
);

CREATE TABLE ratings (
```

```
    UserID          integer     REFERENCES users,
    RoomID          integer     REFERENCES chatrooms,
    Value           integer     NOT NULL,
    PRIMARY KEY (UserID, RoomID),
    CHECK(Value >= 0 AND Value <= 2)
);

CREATE TABLE messages (
    MsgID           serial      PRIMARY KEY,
    RoomID          integer     NOT NULL REFERENCES chatrooms,
    UserID          integer     REFERENCES users,
    MsgText         varchar     NOT NULL,
    PostTime        timestamp   DEFAULT current_timestamp
);

CREATE TABLE privatemessages (
    PvtMsgID        serial      PRIMARY KEY,
    SenderID        integer     REFERENCES users (UserID),
    ReceiverID      integer     REFERENCES users (UserID),
    MsgText         varchar     NOT NULL,
    SendTime        timestamp   DEFAULT current_timestamp,
    ReadTime        timestamp
);

CREATE TABLE documents (
    DocID           serial      PRIMARY KEY,
    MsgID           integer     REFERENCES messages,
    PvtMsgID        integer     REFERENCES privatemessages,
    Filename        varchar     NOT NULL
);

--Create views
-- Users com privacidade
CREATE VIEW usersprotected AS
  SELECT userid,
        CASE usernamepublic WHEN 't' THEN username ELSE '[hidden]' END "username",
        password, deleted, usernamepublic, profilepublic,
        CASE profilepublic WHEN 't' THEN name ELSE '[hidden]' END "name",
        CASE profilepublic WHEN 't' THEN male ELSE 'f' END "male",
        CASE profilepublic WHEN 't' THEN mail ELSE '[hidden]' END "mail",
        CASE profilepublic WHEN 't' THEN location ELSE '[hidden]' END "location",
        CASE profilepublic WHEN 't' THEN birthday ELSE null END "birthday"
```

```
  from users;

-- Último post por chatroom
CREATE VIEW lastmsg AS
  SELECT roomid, MAX(msgid) "msgid"
  FROM messages
  GROUP BY roomid;

-- Número de posts por chatroom
CREATE VIEW chattotalmsg AS
  SELECT roomid, COUNT(*) "numposts"
  FROM messages
  GROUP BY roomid;

-- Número de posts por user
CREATE VIEW usertotalmsg AS
  SELECT userid, COUNT(*) "numposts"
  FROM messages
  GROUP BY userid;

-- Média de rating por chatroom
CREATE VIEW chatrating AS
  SELECT roomid, SUM(value)/COUNT(*)::float "mean"
  FROM ratings
  GROUP BY roomid;

-- Número de documentos anexados às mensagens
CREATE VIEW msgdocuments AS
  SELECT msgid, COUNT(*) "numdocs"
  FROM documents
  GROUP BY msgid;

-- Chatrooms com muito mais informacao associada
CREATE VIEW chatrooms_extended AS
  SELECT chatrooms.*, owners.username "owner",
        posters.username "lastposter",
        messages.posttime "lastposttime",
        messages.msgtext "lastmsgtext",
        chatrating.mean "rating"
  FROM chatrooms
  LEFT JOIN lastmsg ON chatrooms.roomid = lastmsg.roomid
  LEFT JOIN messages ON lastmsg.msgid = messages.msgid
```

```
  LEFT JOIN usersprotected "owners" ON chatrooms.ownerid = owners.userid
  LEFT JOIN usersprotected "posters" ON messages.userid = posters.userid
  LEFT JOIN chatrating ON chatrooms.roomid = chatrating.roomid;

-- Vista das permissões atuais dos utilizadores
CREATE VIEW users_permissions AS
  SELECT u.userid "userid",
        c.roomid "roomid",
        (coalesce(pp.canread, c.readingperm, TRUE)
         OR c.ownerid = u.userid) "canread",
        (coalesce(pp.canpost, c.postingperm, TRUE)
         OR c.ownerid = u.userid) AND NOT c.closed "canpost"
  FROM chatrooms c
  CROSS JOIN users u
  LEFT JOIN (SELECT userid, roomid, canread, canpost
             FROM permissions) pp
  ON (c.roomid = pp.roomid AND u.userid = pp.userid);

-- Vista das permissões do convidado (user não autenticado)
CREATE VIEW guest_permissions AS
  SELECT 0 "userid",
        c.roomid "roomid",
        coalesce(c.readingperm, FALSE) "canread",
        FALSE "canpost"
  FROM chatrooms c;

-- Vista das permissões atuais
CREATE VIEW current_permissions AS
  SELECT * FROM users_permissions
    UNION ALL
  SELECT * FROM guest_permissions;

-- Chatrooms extended mais informacao respetiva do utilizador (permissions + rating)
CREATE VIEW chatrooms_extended_user AS
  SELECT cr.*, p.userid, p.canread, p.canpost
  FROM chatrooms_extended cr
  INNER JOIN current_permissions p
  ON p.roomid = cr.roomid;

--Create functions
CREATE FUNCTION getChatrooms(userid integer)
RETURNS SETOF chatrooms_extended_user
```

```
AS $$
   SELECT * FROM chatrooms_extended_user
   WHERE canread AND userid = COALESCE($1,0)
   ORDER BY coalesce(lastposttime, creationdate) DESC
$$ LANGUAGE SQL;


-- Efetua pesquisa às chatrooms em que user tem permissão para ler
-- TODO: devia ser mais elaborado
CREATE FUNCTION searchChatrooms(userid integer, title varchar, owner varchar)
RETURNS SETOF chatrooms_extended_user
AS $$
   SELECT * FROM getChatrooms($1)
   WHERE title ILIKE $2 AND owner ILIKE $3
$$ LANGUAGE SQL;


CREATE FUNCTION deleteUserData(uid integer)
RETURNS void
AS $$
   UPDATE users SET deleted='t',
                        usernamepublic='t',
                        profilepublic='t',
                        male=null,
                        name=null,
                        mail=null,
                        location = null,
                        birthday=null
                WHERE
                        userid = $1;
$$ LANGUAGE SQL;

--Create triggers
--trigger para verificar se o user tem permissões de postar
CREATE OR REPLACE FUNCTION post_permission_check()
RETURNS trigger
AS $$

DECLARE
   perm_rec current_permissions%ROWTYPE;

BEGIN
```

```
   SELECT * INTO perm_rec FROM current_permissions WHERE userid = NEW.userid AND roomid =
NEW.roomid;

   IF (FOUND AND (NOT perm_rec.canpost))
      THEN
         RAISE EXCEPTION 'Cannot post in this chatroom';
   END IF;

   RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER chatroom_post_trigger
BEFORE INSERT OR UPDATE ON messages
FOR EACH ROW EXECUTE PROCEDURE post_permission_check();

--trigger para verificar o limite de ratings
CREATE OR REPLACE FUNCTION rate_limit_check()
RETURNS trigger
AS $$

DECLARE
   max_val INTEGER;
   n_rates INTEGER;
   can_read BOOLEAN;

BEGIN
   LOCK TABLE ratings IN EXCLUSIVE MODE;

   select canread
                  from users_permissions into can_read
                  where userid = New.userid AND roomid = New.roomid;

   select COUNT(DISTINCT userid)
                  from messages into max_val
                  where roomid = NEW.roomid;

         select COUNT(DISTINCT userid)
                  from ratings into n_rates
                  where roomid = NEW.roomid;
```

```
    IF NOT can_read THEN
        RAISE EXCEPTION 'user has no permission to rate this chatroom';
    END IF;

    IF n_rates >= max_val THEN
            RAISE EXCEPTION 'rating limit for this room reached';
    END IF;

    RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER rate_trigger
BEFORE INSERT ON ratings
FOR EACH ROW
EXECUTE PROCEDURE rate_limit_check();

--trigger para verificar se quando se apaga uma mensagem esta não foi partilhada
CREATE OR REPLACE FUNCTION delete_message_check()
RETURNS trigger
AS $$

BEGIN

        LOCK TABLE privatemessages IN EXCLUSIVE MODE;

        IF (Old.readtime IS NOT NULL) THEN
                RAISE EXCEPTION 'message has already been shared, impossible to erase it';
        END IF;

    RETURN Old;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER delete_message_trigger
BEFORE DELETE ON privatemessages
FOR EACH ROW EXECUTE PROCEDURE delete_message_check();
```

Notas adicionais

A aplicação encontra-se online em wsn.dynip.sapo.pt.

A base de dados respectiva está bastante populada por diversas pessoas (o qual nem sempre é controlado).

Conclusão

Análise da execução do trabalho e dos resultados obtidos

Em suma o resultado final do trabalho produzido pelo grupo é altamente satisfatório, tanto pelo facto de a aplicação ser bastante usável, como pelo facto de ter existido uma preocupação em ter uma base de dados bastante populada, com pessoas diferentes, mais ou menos ativas, o que levou a que houvesse também uma necessidade de fazer algumas operações de manutenção sobre a base de dados, uma vez que esta foi sofrendo alterações à sua estrutura, entre outros motivos.

Por fim, há que referir que a aplicação está disponível online em wsn.dynip.sapo.pt. Outra possibilidade será pesquisar “txuxrum” no google e “sentir-se com sorte”.