**A6: Indexes, triggers, user functions**

**and population**

**1. Database workload**

A study of the predicted system load (database load), organized in subsections.

**1.1 Tuple Estimation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation Reference** | **Relation Name** | **Order of magnitude** | **Estimated growth** |
| R01 | user | hundreds | units per day |
| R02 | event | hundreds | units per day |
| R03 | localization | hundreds | units per day |
| R04 | image | hundreds | units per day |
| R05 | city | tens | units per week |
| R06 | country | tens | units per month |
| R07 | post | hundreds | dozens per day |
| R08 | poll | tens | units per day |
| R09 | option | tens | units per day |
| R10 | friendRequest | hundreds | dozens per day |
| R11 | friendActivity | hundreds | dozens per day |
| R12 | eventInvite | hundreds | dozens per day |
| R13 | owner | tens | units per day |
| R14 | participant | hundreds | units per day |
| R15 | done | hundreds | units per day |
| R16 | notDone | hundreds | units per day |
| R17 | admin | units | no growth |
| R18 | friendship | tens | units per day |
| R19 | eventDeleteWarning | tens | units per month |
| R20 | eventUpdateWarning | hundreds | units per day |
| R21 | currentDate | units | hundreds per day |
| R22 | rating | tens | units per day |

**1.2 Most frequent queries**

|  |  |
| --- | --- |
| **Querie reference** | **SELECT01** |
| **Querie description** | **User’s information** |
| **Querie frequency** | **Hundreds per day** |
| SELECT username, last\_name, first\_name, email, image\_path, city\_id  FROM users  WHERE users.id = $user\_id; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT02** |
| **Querie description** | **Event’s information** |
| **Querie frequency** | **Hundreds per day** |
| SELECT events.id, events.name, events.category,  events.description, events."date", users.username  FROM events, users  WHERE events.owner\_id = users.id AND events.id = $event\_id;    SELECT posts.description, posts.id, posts.image\_path, posts.user\_id  FROM posts,events  WHERE posts.event\_id = $event\_id;    SELECT users.username, users.image\_path  FROM participants  WHERE users.id = participants.user\_id AND participants.event\_id=$event\_id; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT03** |
| **Querie description** | **Search event** |
| **Querie frequency** | **Hundreds per day** |
| SELECT id, "name", "date", localization, category  FROM events  WHERE "name" LIKE %$search% OR localization LIKE %$search% AND event\_type = 'public'  ORDER BY "name"; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT04** |
| **Querie description** | **Search user** |
| **Querie frequency** | **Hundreds per day** |
| SELECT id, username, image\_path  FROM users  WHERE username LIKE %$search%  ORDER BY username; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT05** |
| **Querie description** | **Search by categories** |
| **Querie frequency** | **Hundreds per day** |
| SELECT id, "name", "date", localization, category  FROM events  WHERE "name" LIKE %$search% OR localization LIKE %$search% AND event\_type = 'public' AND events.category LIKE %$categories%  ORDER BY "name"; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT06** |
| **Querie description** | **User’s notifications** |
| **Querie frequency** | **Hundreds per day** |
| SELECT sender\_id  FROM friend\_requests  WHERE receiver\_id = $user\_id;  SELECT sender\_id, event\_id  FROM friend\_activities  WHERE receiver\_id = $user\_id;  SELECT owner\_id, event\_id  FROM event\_invites  WHERE receiver\_id = $user\_id;  SELECT event\_name  FROM event\_delete\_warnings  WHERE receiver\_id = $user\_id;  SELECT event\_id  FROM event\_update\_warnings  WHERE receiver\_id = $user\_id; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT07** |
| **Querie description** | **Owner of a specific event** |
| **Querie frequency** | **Hundreds per day** |
| SELECT event\_id  FROM owners  WHERE user\_id = $user\_id; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT08** |
| **Querie description** | **User’s events** |
| **Querie frequency** | **Hundreds per day** |
| SELECT event\_id  FROM participants  WHERE user\_id = $user\_id; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT09** |
| **Querie description** | **User’s friends** |
| **Querie frequency** | **Hundreds per day** |
| SELECT user\_id\_1, user\_id\_2  FROM friendships  WHERE user\_id\_1 = $user\_id OR user\_id\_2 = $user\_id; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT010** |
| **Querie description** | **Who can be invited to an event** |
| **Querie frequency** | **Hundreds per day** |
| SELECT users.username  FROM users, events  WHERE users.id!= event.owner\_id AND users.id NOT IN (  SELECT user\_id  FROM participants  WHERE user\_id IS NOT NULL AND participants.event\_id=$event\_id) ; | |

|  |  |
| --- | --- |
| **Querie reference** | **SELECT11** |
| **Querie description** | **Rating of a past event** |
| **Querie frequency** | **Hundreds per day** |
| SELECT rating  FROM dones  WHERE dones.event\_id= $event\_id; | |

**1.3 Most frequent modifications**

|  |  |
| --- | --- |
| **Querie reference** | **UPDATE01** |
| **Querie description** | **Update users information** |
| **Querie frequency** | **Hundreds per month** |
| UPDATE "users"  SET password = $password,  email = $email,  first\_name = $first\_name,  last\_name = $last\_name,  image\_path = $image\_path,  city\_id = $city\_id  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **UPDATE02** |
| **Querie description** | **Update events information** |
| **Querie frequency** | **Hundreds per month** |
| UPDATE events  SET name = $name,  date = $date,  description = $description,  localization\_id = $localization\_id,  event\_type = $event\_type,  category = $category  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **UPDATE03** |
| **Querie description** | **Update options description** |
| **Querie frequency** | **Tens per month** |
| UPDATE options  SET description = $description  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **UPDATE04** |
| **Querie description** | **Update posts information** |
| **Querie frequency** | **Hundreds per month** |
| UPDATE posts  SET description = $description,  date = $date,  image\_path = $image\_path  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **UPDATE05** |
| **Querie description** | **Update current\_date** |
| **Querie frequency** | **Thousands per month** |
| UPDATE "current\_date"  SET date = $date  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT01** |
| **Querie description** | **Regist new user** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO users  (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)  VALUES ($username, $password, $email, $regist\_date, $first\_name, $last\_name, $image\_path, $city\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT02** |
| **Querie description** | **Create new event** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO events  (name,date,description,owner\_id,localization\_id,type,category)  VALUES ($name,$date,$description,$owner\_id,$localization\_id, $type,$category); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT03** |
| **Querie description** | **Create new post** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO posts (description,date,event\_id, user\_id, image\_path)  VALUES ($description,$date,$event\_id, $user\_id, $image\_path); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT04** |
| **Querie description** | **Create new event done** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO dones (event\_id)  VALUES ($event\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT05** |
| **Querie description** | **Create new event not\_done** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO dones (event\_id)  VALUES ($event); | |

|  |  |
| --- | --- |
| **Querie reference** | INSERT06 |
| **Querie description** | Create new participant |
| **Querie frequency** | Tens per month |
| INSERT INTO participants (user\_id,event\_id)  VALUES ($user\_id,$event\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | INSERT07 |
| **Querie description** | Create new owner |
| **Querie frequency** | Units per month |
| INSERT INTO owners (user\_id,event\_id)  VALUES ($user\_id,$event\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT08** |
| **Querie description** | **Create new image** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO images (event\_id, path) VALUES ($event\_id, $path); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT09** |
| **Querie description** | **Create new localization** |
| **Querie frequency** | **Units per month** |
| INSERT INTO localizations (name,address,latitude,longitude,city\_id)  VALUES ($name,$address,$latitude,$longitude,$city\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT10** |
| **Querie description** | **Create new city** |
| **Querie frequency** | **Units per month** |
| INSERT INTO cities (name,country\_id) VALUES ($name,$country\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT11** |
| **Querie description** | **Create new country** |
| **Querie frequency** | **Units per month** |
| INSERT INTO countries (name) VALUES ($name); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT12** |
| **Querie description** | **Create new poll** |
| **Querie frequency** | **Units per month** |
| INSERT INTO polls (post\_id) VALUES ($posts\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT13** |
| **Querie description** | **Create new option** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO options (description,poll\_id) VALUES  ($description,$poll\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT14** |
| **Querie description** | **Create new friend request** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO friend\_requests (sender\_id,receiver\_id)  VALUES ($sender\_id,$receiver\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT15** |
| **Querie description** | **Create new friend activity** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO friend\_activities (sender\_id,receiver\_id,event\_id)  VALUES ($sender\_id,$receiver\_id,$event\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT16** |
| **Querie description** | **Create new event invite** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO event\_invites (event\_id,owner\_id,receiver\_id)  VALUES ($event\_id,$owner\_id,$receiver\_id); | |

|  |  |
| --- | --- |
| **Querie reference** | **INSERT17** |
| **Querie description** | **Create new friendship** |
| **Querie frequency** | **Tens per month** |
| INSERT INTO friendships (user\_id\_1, user\_id\_2)  VALUES ($user\_id\_1, $user\_id\_2); | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE01** |
| **Querie description** | **Delete an user** |
| **Querie frequency** | **Units per month** |
| DELETE FROM "users"  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE02** |
| **Querie description** | **Delete an event** |
| **Querie frequency** | **Units per month** |
| DELETE FROM events  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE03** |
| **Querie description** | **Delete a post** |
| **Querie frequency** | **Units per month** |
| DELETE FROM posts  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE04** |
| **Querie description** | **Delete an option** |
| **Querie frequency** | **Tens per month** |
| DELETE FROM options  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE05** |
| **Querie description** | **Delete a poll** |
| **Querie frequency** | **Units per month** |
| DELETE FROM polls  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE06** |
| **Querie description** | **Delete a friendship** |
| **Querie frequency** | **Units per month** |
| DELETE FROM friendships  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE07** |
| **Querie description** | **Delete a friend\_request** |
| **Querie frequency** | **Tens per month** |
| DELETE FROM friend\_requests  WHERE id = $id; | |

|  |  |
| --- | --- |
| **Querie reference** | **DELETE08** |
| **Querie description** | **Delete an event\_invite** |
| **Querie frequency** | **Tens per month** |
| DELETE FROM event\_invites  WHERE id = $id; | |

**2. Proposed indexes**

**2.1 Performance indexes**

|  |  |
| --- | --- |
| **Index reference** | IDX01 |
| **Related queries** | SELECT01 |
| **Index relation** | user |
| **Index atribute** | username |
| **Index type** | Hash |
| **Cardinality** | High |
| **Clustering** | No |
| **Justification** | Query SELECT01 has to be fast as it is executed many times; doesn't need range query support; cardinality is high because email is an unique key; it's not a good candidate for clustering. |
| CREATE INDEX user\_username ON users USING hash (username); | |

|  |  |
| --- | --- |
| **Index reference** | IDX02 |
| **Related queries** | SELECT02 |
| **Index relation** | events |
| **Index** **atribute** | owner\_id |
| **Index type** | Hash |
| **Cardinality** | high |
| **Clustering** | No |
| **Justification** | Query has to be fast as it is executed many times; doesn't need range query support; cardinality is high because owner\_id is an unique key; it's not a good candidate for clustering. |
| CREATE INDEX owner\_events ON events USING hash(owner\_id); | |

**2.2 Full-text search indexes**

|  |  |
| --- | --- |
| **Index reference** | IDX03 |
| **Related queries** | SELECT03 |
| **Index** **relation** | event |
| **Index atribute** | name |
| **Index type** | GIST |
| **Clustering** | No |
| **Justification** | To improve the performance of full text searches while searching for events and their names; GiST because it's better for dynamic data. |
| CREATE INDEX search\_events ON events USING GIST (to\_tsvector('english',  name)); | |

**3. Triggers**

|  |  |
| --- | --- |
| **Trigger reference** | TRIGGER01 |
| **Trigger description** | When the current\_date is updated, it is verified if it is greater than the event\_date. In that case, the event is added to the dones table and removed from the not\_dones. |
| CREATE OR REPLACE FUNCTION set\_event\_as\_done() RETURNS TRIGGER AS  $BODY$  BEGIN  IF EXISTS (SELECT event\_id FROM not\_done WHERE NEW.event\_id = id)  THEN  INSERT INTO dones VALUES (NEW.event\_id, NULL);  DELETE FROM not\_dones WHERE id = NEW.event\_id;  END IF;  RETURN NEW;  END  $BODY$  LANGUAGE plpgsql;    DROP TRIGGER IF EXISTS set\_event\_as\_done ON "current\_date";  CREATE TRIGGER set\_event\_as\_done  BEFORE UPDATE OF date ON "current\_date"  FOR EACH ROW  WHEN NEW.date = GETDATE()  EXECUTE PROCEDURE set\_event\_as\_done(); | |

|  |  |
| --- | --- |
| **Trigger reference** | TRIGGER02 |
| **Trigger description** | When an event is deleted a notification is sent to the participants of that event. |
| CREATE OR REPLACE FUNCTION notificate\_event\_delete() RETURNS TRIGGER AS  $BODY$  DECLARE  idx int;  BEGIN  FOR idx IN SELECT id FROM participants WHERE participants.event\_id = OLD.id  LOOP  INSERT INTO event\_delete\_warnings (event\_name, receiver\_id) VALUES (OLD.name, idx);  END LOOP;  RETURN OLD;  END;  $BODY$  LANGUAGE plpgsql;  DROP TRIGGER IF EXISTS notificate\_event\_delete ON "events";  CREATE TRIGGER notificate\_event\_delete  BEFORE DELETE ON events  FOR EACH ROW  EXECUTE PROCEDURE notificate\_event\_delete(); | |

|  |  |
| --- | --- |
| **Trigger reference** | TRIGGER03 |
| **Trigger description** | When an event is updated a notification is sent to the participants of that event. |
| CREATE OR REPLACE FUNCTION notificate\_event\_update() RETURNS TRIGGER AS  $BODY$  DECLARE  idx int;  BEGIN  FOR idx IN SELECT id FROM participants WHERE participants.event\_id = OLD.id  LOOP  INSERT INTO event\_update\_warnings (event\_id, receiver\_id) VALUES (OLD.id, idx);  END LOOP;  RETURN OLD;  END;  $BODY$  LANGUAGE plpgsql;  DROP TRIGGER IF EXISTS notificate\_event\_update ON "events";  CREATE TRIGGER notificate\_event\_update  BEFORE DELETE ON events  FOR EACH ROW  EXECUTE PROCEDURE notificate\_event\_update(); | |

|  |  |
| --- | --- |
| **Trigger reference** | TRIGGER04 |
| **Trigger description** | When a rating is inserted, the dones table is updated with the new rating average for that event. |
| CREATE OR REPLACE FUNCTION rating\_update() RETURNS TRIGGER AS  $BODY$  BEGIN  UPDATE dones SET rating = (SELECT AVG("value") FROM ratings WHERE New.event\_id = event\_id) WHERE event\_id = New.event\_id;  RETURN NULL;  END  $BODY$  LANGUAGE plpgsql;    CREATE TRIGGER rating\_update  AFTER INSERT ON ratings  FOR EACH ROW  EXECUTE PROCEDURE rating\_update(); | |

|  |  |
| --- | --- |
| **Trigger reference** | TRIGGER05 |
| **Trigger description** | When a friend request is accepted, a new friendship is added. |
| CREATE OR REPLACE FUNCTION accept\_friend\_request() RETURNS TRIGGER AS  $BODY$  BEGIN  IF New.answer = 'yes'  THEN  INSERT INTO friendships (user\_id\_1, user\_id\_2) VALUES (New.sender\_id, New.receiver\_id);  END IF;  RETURN NULL;  END  $BODY$  LANGUAGE plpgsql;    CREATE TRIGGER accept\_friend\_request  AFTER UPDATE ON friend\_requests  FOR EACH ROW  EXECUTE PROCEDURE accept\_friend\_request(); | |

|  |  |
| --- | --- |
| **Trigger reference** | TRIGGER06 |
| **Trigger description** | When an event invite is accepted, the user that had been invited is added to the participants of that event. |
| CREATE OR REPLACE FUNCTION accept\_event\_invite() RETURNS TRIGGER AS  $BODY$  BEGIN  IF New.answer = 'yes'  THEN  INSERT INTO participants (user\_id, event\_id) VALUES (New.receiver\_id, New.event\_id);  END IF;  RETURN NULL;  END  $BODY$  LANGUAGE plpgsql;    CREATE TRIGGER accept\_event\_invite  AFTER UPDATE ON event\_invites  FOR EACH ROW  EXECUTE PROCEDURE accept\_event\_invite(); | |

**4. SQL Code**

[Lbaw1765db.sql](https://github.com/Quaresma1997/lbaw1765/blob/gh-pages/lbaw1765_db.sql)

[Inserts.sql](https://github.com/Quaresma1997/lbaw1765/blob/gh-pages/inserts.sql)

--Types Enums

DROP TYPE IF EXISTS categories CASCADE;

CREATE TYPE categories AS ENUM(

'Music',

'Sports',

'Entertainment',

'Educational',

'Business',

'Other'

);

DROP TYPE IF EXISTS types\_of\_event CASCADE;

CREATE TYPE types\_of\_event AS ENUM(

'Public',

'Private'

);

--Tables

DROP TABLE IF EXISTS admins CASCADE;

CREATE TABLE admins (

id SERIAL NOT NULL,

username text NOT NULL,

password text NOT NULL,

email text NOT NULL,

CONSTRAINT admins\_pk PRIMARY KEY (id),

CONSTRAINT admins\_email\_uk UNIQUE (email)

);

DROP TABLE IF EXISTS cities CASCADE;

CREATE TABLE cities (

id SERIAL NOT NULL,

name text NOT NULL,

country\_id INTEGER NOT NULL,

CONSTRAINT cities\_pk PRIMARY KEY (id),

CONSTRAINT cities\_name\_uk UNIQUE (name)

);

DROP TABLE IF EXISTS "current\_date" CASCADE;

CREATE TABLE "current\_date" (

id SERIAL NOT NULL,

date TIMESTAMP WITH TIME zone NOT NULL,

CONSTRAINT current\_date\_pk PRIMARY KEY (id)

);

DROP TABLE IF EXISTS countries CASCADE;

CREATE TABLE countries (

id SERIAL NOT NULL,

name text NOT NULL,

CONSTRAINT countries\_pk PRIMARY KEY (id),

CONSTRAINT countries\_name\_uk UNIQUE (name)

);

DROP TABLE IF EXISTS dones CASCADE;

CREATE TABLE dones (

event\_id INTEGER NOT NULL,

rating FLOAT,

CONSTRAINT dones\_pk PRIMARY KEY (event\_id),

CONSTRAINT rating\_ck CHECK (((rating >= 1) AND (rating <= 5)))

);

DROP TABLE IF EXISTS events CASCADE;

CREATE TABLE events (

id SERIAL NOT NULL,

name text NOT NULL,

date TIMESTAMP WITH TIME zone NOT NULL,

description text NOT NULL,

owner\_id INTEGER NOT NULL,

localization\_id INTEGER,

type types\_of\_event NOT NULL,

category categories NOT NULL,

CONSTRAINT events\_pk PRIMARY KEY (id),

CONSTRAINT date\_ck CHECK ((date > now()))

);

DROP TABLE IF EXISTS event\_invites CASCADE;

CREATE TABLE event\_invites (

id SERIAL NOT NULL,

answer text,

event\_id INTEGER NOT NULL,

owner\_id INTEGER,

receiver\_id INTEGER NOT NULL,

CONSTRAINT event\_invites\_pk PRIMARY KEY (id)

);

DROP TABLE IF EXISTS event\_delete\_warnings CASCADE;

CREATE TABLE event\_delete\_warnings (

id SERIAL NOT NULL,

event\_name text NOT NULL,

receiver\_id INTEGER NOT NULL,

CONSTRAINT event\_delete\_warnings\_pk PRIMARY KEY (id)

);

DROP TABLE IF EXISTS event\_update\_warnings CASCADE;

CREATE TABLE event\_update\_warnings (

id SERIAL NOT NULL,

event\_id INTEGER NOT NULL,

receiver\_id INTEGER NOT NULL,

CONSTRAINT event\_update\_warnings\_pk PRIMARY KEY (id),

CONSTRAINT event\_update\_warnings\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS friend\_activities CASCADE;

CREATE TABLE friend\_activities (

id SERIAL NOT NULL,

sender\_id INTEGER NOT NULL,

receiver\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

CONSTRAINT friend\_activities\_pk PRIMARY KEY (id),

CONSTRAINT friend\_activities\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS friend\_requests CASCADE;

CREATE TABLE friend\_requests (

id SERIAL NOT NULL,

answer text,

sender\_id INTEGER NOT NULL,

receiver\_id INTEGER NOT NULL,

CONSTRAINT friend\_requests\_pk PRIMARY KEY (id)

);

DROP TABLE IF EXISTS friendships CASCADE;

CREATE TABLE friendships (

id SERIAL NOT NULL,

user\_id\_1 INTEGER NOT NULL,

user\_id\_2 INTEGER NOT NULL,

CONSTRAINT friendships\_users\_ids\_uk UNIQUE (user\_id\_1, user\_id\_2)

);

DROP TABLE IF EXISTS images CASCADE;

CREATE TABLE images (

id SERIAL NOT NULL,

event\_id INTEGER NOT NULL,

path text NOT NULL,

CONSTRAINT images\_pk PRIMARY KEY (id),

CONSTRAINT images\_path\_uk UNIQUE (path)

);

DROP TABLE IF EXISTS localizations CASCADE;

CREATE TABLE localizations (

id SERIAL NOT NULL,

name text NOT NULL,

address text NOT NULL,

latitude FLOAT,

longitude FLOAT,

city\_id INTEGER NOT NULL,

CONSTRAINT localizations\_pk PRIMARY KEY (id),

CONSTRAINT localizations\_city\_id\_fk FOREIGN KEY (city\_id) REFERENCES

cities(id) ON DELETE SET NULL

);

DROP TABLE IF EXISTS not\_dones CASCADE;

CREATE TABLE not\_dones (

event\_id INTEGER NOT NULL,

CONSTRAINT not\_dones\_pk PRIMARY KEY (event\_id),

CONSTRAINT not\_dones\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS options CASCADE;

CREATE TABLE options (

id SERIAL NOT NULL,

description text NOT NULL,

poll\_id INTEGER NOT NULL,

CONSTRAINT options\_pk PRIMARY KEY (id)

);

DROP TABLE IF EXISTS owners CASCADE;

CREATE TABLE owners (

id SERIAL NOT NULL,

user\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

CONSTRAINT owners\_pk PRIMARY KEY (id),

CONSTRAINT owners\_user\_id\_event\_id\_uk UNIQUE (user\_id, event\_id),

CONSTRAINT owners\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS participants CASCADE;

CREATE TABLE participants (

id SERIAL NOT NULL,

user\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

CONSTRAINT participants\_pk PRIMARY KEY (id),

CONSTRAINT participants\_user\_id\_event\_id\_uk UNIQUE (user\_id, event\_id),

CONSTRAINT participants\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS polls CASCADE;

CREATE TABLE polls (

id SERIAL NOT NULL,

post\_id INTEGER NOT NULL,

CONSTRAINT polls\_pk PRIMARY KEY (id)

);

DROP TABLE IF EXISTS posts CASCADE;

CREATE TABLE posts (

id SERIAL NOT NULL,

description text NOT NULL,

date TIMESTAMP WITH TIME zone NOT NULL,

event\_id INTEGER NOT NULL,

user\_id INTEGER NOT NULL,

image\_path text NOT NULL,

CONSTRAINT posts\_pk PRIMARY KEY (id),

CONSTRAINT posts\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS ratings CASCADE;

CREATE TABLE ratings (

id SERIAL NOT NULL,

"value" INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

user\_id INTEGER NOT NULL,

CONSTRAINT ratings\_pk PRIMARY KEY (id),

CONSTRAINT ratings\_user\_id\_event\_id\_uk UNIQUE (user\_id, event\_id),

CONSTRAINT ratings\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON DELETE CASCADE

);

DROP TABLE IF EXISTS users CASCADE;

CREATE TABLE users (

id SERIAL NOT NULL,

username text NOT NULL,

password text NOT NULL,

email text NOT NULL,

regist\_date TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,

first\_name text NOT NULL,

last\_name text NOT NULL,

image\_path text NOT NULL,

city\_id INTEGER,

CONSTRAINT users\_pk PRIMARY KEY (id),

CONSTRAINT users\_name\_uk UNIQUE (username),

CONSTRAINT users\_email\_uk UNIQUE (email),

CONSTRAINT users\_city\_id\_fk FOREIGN KEY (city\_id) REFERENCES

cities(id) ON DELETE SET NULL

);

ALTER TABLE ONLY cities

ADD CONSTRAINT cities\_country\_id\_fk FOREIGN KEY (country\_id) REFERENCES

countries(id) ON DELETE SET NULL;

ALTER TABLE ONLY dones

ADD CONSTRAINT dones\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

events(id) ON UPDATE CASCADE;

ALTER TABLE ONLY events

ADD CONSTRAINT events\_owner\_id\_fk FOREIGN KEY (owner\_id) REFERENCES

users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY events

ADD CONSTRAINT events\_localization\_id\_fk FOREIGN KEY (localization\_id) REFERENCES

localizations(id) ON DELETE SET NULL;

ALTER TABLE ONLY event\_invites

ADD CONSTRAINT event\_invites\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES

not\_dones(event\_id) ON DELETE CASCADE;

ALTER TABLE ONLY event\_invites

ADD CONSTRAINT event\_invites\_owner\_id\_fk FOREIGN KEY (owner\_id) REFERENCES

owners(id) ON DELETE SET NULL;

ALTER TABLE ONLY event\_invites

ADD CONSTRAINT event\_invites\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY event\_delete\_warnings

ADD CONSTRAINT event\_delete\_warnings\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY event\_update\_warnings

ADD CONSTRAINT event\_update\_warnings\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend\_activities

ADD CONSTRAINT friend\_activities\_sender\_id\_fk FOREIGN KEY (sender\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend\_activities

ADD CONSTRAINT friend\_activities\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend\_requests

ADD CONSTRAINT friend\_requests\_sender\_id\_fk FOREIGN KEY (sender\_id) REFERENCES

participants(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend\_requests

ADD CONSTRAINT friend\_requests\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friendships

ADD CONSTRAINT friendships\_user\_id\_1 FOREIGN KEY (user\_id\_1) REFERENCES

users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY friendships

ADD CONSTRAINT friendships\_user\_id\_2 FOREIGN KEY (user\_id\_2) REFERENCES

users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY options

ADD CONSTRAINT options\_poll\_id\_fk FOREIGN KEY (poll\_id) REFERENCES

polls(id) ON DELETE CASCADE;

ALTER TABLE ONLY owners

ADD CONSTRAINT owners\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES

users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY participants

ADD CONSTRAINT participants\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES

users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY polls

ADD CONSTRAINT polls\_post\_id\_fk FOREIGN KEY (post\_id) REFERENCES

posts(id) ON DELETE CASCADE;

ALTER TABLE ONLY posts

ADD CONSTRAINT posts\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES

users(id) ON DELETE CASCADE;

ALTER TABLE ONLY ratings

ADD CONSTRAINT ratings\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES

users(id) ON DELETE SET NULL;

CREATE OR REPLACE FUNCTION set\_event\_as\_done() RETURNS TRIGGER AS

$BODY$

BEGIN

IF EXISTS (SELECT event\_id FROM not\_done WHERE NEW.event\_id = id)

THEN

INSERT INTO dones VALUES (NEW.event\_id, NULL);

DELETE FROM not\_dones WHERE id = NEW.event\_id;

END IF;

RETURN NEW;

END

$BODY$

LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS set\_event\_as\_done ON "current\_date";

CREATE TRIGGER set\_event\_as\_done

BEFORE UPDATE OF date ON "current\_date"

FOR EACH ROW

WHEN NEW.date = GETDATE()

EXECUTE PROCEDURE set\_event\_as\_done();

CREATE OR REPLACE FUNCTION notificate\_event\_delete() RETURNS TRIGGER AS

$BODY$

DECLARE

idx int;

BEGIN

FOR idx IN SELECT id FROM participants WHERE participants.event\_id = OLD.id

LOOP

INSERT INTO event\_delete\_warnings (event\_name, receiver\_id) VALUES (OLD.name, idx);

END LOOP;

RETURN OLD;

END;

$BODY$

LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS notificate\_event\_delete ON "events";

CREATE TRIGGER notificate\_event\_delete

BEFORE DELETE ON events

FOR EACH ROW

EXECUTE PROCEDURE notificate\_event\_delete();

CREATE OR REPLACE FUNCTION notificate\_event\_update() RETURNS TRIGGER AS

$BODY$

DECLARE

idx int;

BEGIN

FOR idx IN SELECT id FROM participants WHERE participants.event\_id = OLD.id

LOOP

INSERT INTO event\_update\_warnings (event\_id, receiver\_id) VALUES (OLD.id, idx);

END LOOP;

RETURN OLD;

END;

$BODY$

LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS notificate\_event\_update ON "events";

CREATE TRIGGER notificate\_event\_update

BEFORE DELETE ON events

FOR EACH ROW

EXECUTE PROCEDURE notificate\_event\_update();

CREATE OR REPLACE FUNCTION rating\_update() RETURNS TRIGGER AS

$BODY$

BEGIN

UPDATE dones SET rating = (SELECT AVG("value") FROM ratings WHERE New.event\_id = event\_id) WHERE event\_id = New.event\_id;

RETURN NULL;

END

$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER rating\_update

AFTER INSERT ON ratings

FOR EACH ROW

EXECUTE PROCEDURE rating\_update();

CREATE OR REPLACE FUNCTION accept\_friend\_request() RETURNS TRIGGER AS

$BODY$

BEGIN

IF New.answer = 'yes'

THEN

INSERT INTO friendships (user\_id\_1, user\_id\_2) VALUES (New.sender\_id, New.receiver\_id);

END IF;

RETURN NULL;

END

$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER accept\_friend\_request

AFTER UPDATE ON friend\_requests

FOR EACH ROW

EXECUTE PROCEDURE accept\_friend\_request();

CREATE OR REPLACE FUNCTION accept\_event\_invite() RETURNS TRIGGER AS

$BODY$

BEGIN

IF New.answer = 'yes'

THEN

INSERT INTO participants (user\_id, event\_id) VALUES (New.receiver\_id, New.event\_id);

END IF;

RETURN NULL;

END

$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER accept\_event\_invite

AFTER UPDATE ON event\_invites

FOR EACH ROW

EXECUTE PROCEDURE accept\_event\_invite();

--> INDEXES

CREATE INDEX user\_username ON users USING hash (username);

CREATE INDEX owner\_events ON events USING hash(owner\_id);

CREATE INDEX search\_events ON events USING GIST (to\_tsvector('english', name));

INSERTS to populate the database

--> INSERTS

-- Here goes the SQL code - INSERTS

INSERT INTO countries (name) VALUES ('Portugal');

INSERT INTO countries (name) VALUES ('Espanha');

INSERT INTO countries (name) VALUES ('USA');

INSERT INTO cities (name,country\_id) VALUES ('Braga',1);

INSERT INTO cities (name,country\_id) VALUES ('Porto',1);

INSERT INTO cities (name,country\_id) VALUES ('Lisboa',1);

INSERT INTO localizations (name,address,latitude,longitude,city\_id) VALUES ('Restaurante O Pirata','Rua da Isabelinha',41.452993,-8.5775364,1);

INSERT INTO localizations (name,address,latitude,longitude,city\_id) VALUES ('FEUP','Rua Roberto Frias',41.1779401,-8.5998763,2);

INSERT INTO localizations (name,address,latitude,longitude,city\_id) VALUES ('Parque da BelaVista','Av. Arlindo Vicente',38.7507558,-9.1265431,3);

INSERT INTO localizations (name,address,latitude,longitude,city\_id) VALUES ('Passeio Maritimo de Alges','Alges',38.697318,-9.2375993,3);

INSERT INTO localizations (name,address,latitude,longitude,city\_id) VALUES ('Hotel Douro','Rua de Agramonte',41.1564707,-8.6288115,2);

INSERT INTO localizations (name,address,latitude,longitude,city\_id) VALUES ('Norte shopping','Matosinhos',41.1825143,-8.6803795,2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('sodales','GUL95ZXR9EX','sodales.at@curae.co.uk',NOW(),'Zeph','Griffin','/imgs/natu.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('uso1','12345','aliquam.iaculis.lacus@amet.co.uk',NOW(),'Ben','Warren','/imgs/natur.jpg',1);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('robin','pass123','amet.ante@faucibusleo.net',NOW(),'Robin','Wright','/imgs/natur.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('bar123','semper123','ut.dolor@gmail.com',NOW(),'Barry','Allen','/imgs/natur.jpg',3);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('reddevil','LBAW','Nulla@et.net',NOW(),'Andrew','Irons','/imgs/november.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('rpedro10','lbaw1765','rpedro10@iol.pt',NOW(),'Rui','Araujo','/imgs/fer.jpg',3);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('joss123','CKB15AAW5MM','ante@fleo.com',NOW(),'Joss','Stone','/imgs/natur.jpg',1);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('top123','SXT16TTW3MH','cursus.et@orciUt.co.uk',NOW(),'Chris','Harris','/imgs/natur.jpg',1);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('roland1','ZYS24FHN5GR','eget.dictum@orciDonec.edu',NOW(),'Roland','Schitt','/imgs/natur.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('david123','ZUS29FRTGVJ','amet@faucibusleo.net',NOW(),'David','Rose','/imgs/natur.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('catones', 'QQQ8EFHNGNR','amec.donec@faucibusleo.net',NOW(),'Carlos','Antonio','/imgs/november.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('emanem','ASDFGHJKL','donex@sapo.net',NOW(),'Raheem','Sterling','/imgs/natur.jpg',3);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('ufoExtis','ZXCVBNM','amet@iol.net',NOW(),'Delle','Alli','/imgs/pyr.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('ragnar','QAZWSXEDC','risus.In.mi@egestas.com',NOW(),'Thor','Ragnarok','/imgs/fer.jpg',1);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('seth','QWERTYUIOP','aliquet.diam.Sed@tinciduntnibh.co.uk',NOW(),'Seth','Byers','/imgs/natu.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('edNorton','TYT71DOD7YN','scelerisque.scelerisque.dui@arcuiaculisenim.ca',NOW(),'Ed','Norton','/imgs/natur.jpg',1);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('Pacquiao','SXT16TTW3MH','magnis@cursuset.edu',NOW(),'Paky','Barret','/imgs/natur.jpg',3);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('steven','MPS10QPK6UE','arcu.Vestibulum@amet.org',NOW(),'Donovan','Stevenson','/imgs/pyr.jpg',1);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('porter123','QIG24ZOK3EM','dui.nec@ultriciesadipiscing.co.uk',NOW(),'Porter','Osborn','/imgs/natu.jpg',2);

INSERT INTO users (username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)

            VALUES ('human','ZYG87WQA6FX','facilisis.magna.tellus@sociis.net',NOW(),'Hu','Randolphe','/imgs/pyr.jpg',2);

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('Antonys Birthday Party', '2018-12-04 12:30:19.000000', 'nunc ac mattis ornare, lectus',1,1,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('ENEI','2018-04-24 12:30:19.000000','Nunc quis arcu vel quam',2,2,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('RockInRio','2018-06-04 12:30:19.000000','tempus eu, ligula. Aenean euismod',3,3,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('Nos Alive','2018-08-04 12:30:19.000000','dignissim pharetra. Nam ac nulla.',3,4,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('Christmas Dinner','2018-10-04 12:30:19.000000','dignissim pharetra. Nam ac nulla.',4,5,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('Mark Birthday Party','2018-04-13 12:30:19.000000','dignissim pharetra. Nam ac nulla.',1,1,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('WebSummit','2018-04-12 12:30:19.000000','lorem lorem, luctus ut, pellentesque',6,6,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('Ted Talk','2018-04-24 12:30:19.000000','sollicitudin orci sem eget massa.',12,2,'Public','Sports');

INSERT INTO events (name,date,description,owner\_id,localization\_id,type,category)

            VALUES ('Teaches Conference','2018-04-13 12:30:19.000000','dignissim pharetra. Nam ac nulla.',1,1,'Private','Business');

INSERT INTO admins (username,password,email) VALUES ('admin1','password','sapo@iol.pt');

INSERT INTO images (event\_id,path) VALUES (1,'/imgs/natur.jpg');

INSERT INTO images (event\_id,path) VALUES (2,'/imgs/natu.jpg');

INSERT INTO images (event\_id,path) VALUES (3,'/imgs/pyr.jpg');

INSERT INTO images (event\_id,path) VALUES (4,'/imgs/november.jpg');

INSERT INTO images (event\_id,path) VALUES (5,'/imgs/taj.jpg');

INSERT INTO images (event\_id,path) VALUES (6,'/imgs/fer.jpg');

INSERT INTO images (event\_id,path) VALUES (7,'/imgs/fa1.jpg');

INSERT INTO images (event\_id,path) VALUES (8,'/imgs/fa2.jpg');

INSERT INTO dones (event\_id)

            VALUES (2);

INSERT INTO dones (event\_id)

            VALUES (7);

INSERT INTO not\_dones (event\_id)

            VALUES (1);

INSERT INTO not\_dones (event\_id)

            VALUES (3);

INSERT INTO not\_dones (event\_id)

            VALUES (4);

INSERT INTO not\_dones (event\_id)

            VALUES (5);

INSERT INTO not\_dones (event\_id)

            VALUES (6);

INSERT INTO not\_dones (event\_id)

            VALUES (8);

INSERT INTO not\_dones (event\_id)

            VALUES (9);

INSERT INTO ratings ( "value", event\_id, user\_id)

VALUES (4, 2, 2);

INSERT INTO participants (user\_id,event\_id)

            VALUES (1,1);

INSERT INTO participants (user\_id,event\_id)

            VALUES (2,1);

INSERT INTO participants (user\_id,event\_id)

            VALUES (2,2);

INSERT INTO participants (user\_id,event\_id)

            VALUES (3,1);

INSERT INTO participants (user\_id,event\_id)

            VALUES (4,1);

INSERT INTO participants (user\_id,event\_id)

            VALUES (5,8);

INSERT INTO participants (user\_id,event\_id)

            VALUES (6,8);

INSERT INTO participants (user\_id,event\_id)

            VALUES (7,3);

INSERT INTO participants (user\_id,event\_id)

            VALUES (12,3);

INSERT INTO participants (user\_id,event\_id)

            VALUES (13,3);

INSERT INTO participants (user\_id,event\_id)

            VALUES (14,3);

INSERT INTO participants (user\_id,event\_id)

            VALUES (15,1);

INSERT INTO participants (user\_id,event\_id)

            VALUES (16,2);

INSERT INTO participants (user\_id,event\_id)

            VALUES (17,3);

INSERT INTO participants (user\_id,event\_id)

            VALUES (18,2);

INSERT INTO participants (user\_id,event\_id)

            VALUES (19,1);

INSERT INTO participants (user\_id,event\_id)

            VALUES (20,3);

INSERT INTO participants (user\_id,event\_id)

            VALUES (12,6);

INSERT INTO participants (user\_id,event\_id)

            VALUES (12,5);

INSERT INTO participants (user\_id,event\_id)

            VALUES (10,4);

INSERT INTO owners (user\_id,event\_id)

            VALUES (1,1);

INSERT INTO owners (user\_id,event\_id)

            VALUES (2,2);

INSERT INTO owners (user\_id,event\_id)

            VALUES (3,3);

INSERT INTO owners (user\_id,event\_id)

            VALUES (3,4);

INSERT INTO owners (user\_id,event\_id)

            VALUES (5,5);

INSERT INTO owners (user\_id,event\_id)

            VALUES (1,6);

INSERT INTO owners (user\_id,event\_id)

            VALUES (6,7);

INSERT INTO owners (user\_id,event\_id)

            VALUES (12,8);

INSERT INTO owners (user\_id,event\_id)

            VALUES (20,9);

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,1,'/img/new.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,3,'/img/new.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,4,'/img/new.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ',NOW(),1,16,'/img/new.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',2,18,'/img/new.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',4,10,'/img/panda.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',3,17,'/img/panda.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',3,14,'/img/panda.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',4,5,'/img/panda.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-01-12 15:55:12',2,16,'/img/panda.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',3,12,'/img/sports.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,15,'/img/sports.jpg');

INSERT INTO posts (description,date,event\_id, user\_id, image\_path) VALUES ('Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',5,12,'/img/sports.jpg');

INSERT INTO polls (post\_id) VALUES (1);

INSERT INTO polls (post\_id) VALUES (2);

INSERT INTO polls (post\_id) VALUES (3);

INSERT INTO polls (post\_id) VALUES (4);

INSERT INTO options (description,poll\_id) VALUES ('Bar',1);

INSERT INTO options (description,poll\_id) VALUES ('Cafe',1);

INSERT INTO options (description,poll\_id) VALUES ('Club',1);

INSERT INTO options (description,poll\_id) VALUES ('Home',1);

INSERT INTO options (description,poll\_id) VALUES ('12/05/2018',2);

INSERT INTO options (description,poll\_id) VALUES ('13/05/2018',2);

INSERT INTO options (description,poll\_id) VALUES ('Great',3);

INSERT INTO options (description,poll\_id) VALUES ('Good',3);

INSERT INTO options (description,poll\_id) VALUES ('Available',4);

INSERT INTO options (description,poll\_id) VALUES ('Not Available',4);

INSERT INTO friend\_requests (sender\_id, receiver\_id) VALUES (1, 2);

INSERT INTO friend\_requests (sender\_id, receiver\_id) VALUES (3, 4);

INSERT INTO event\_invites (event\_id, owner\_id, receiver\_id) VALUES (1, 1, 10);

INSERT INTO event\_invites (event\_id, owner\_id, receiver\_id) VALUES (1, 1, 12);

**Revision History**

Changes made to the first submission:

1. Tuple estimation correction;
2. Added and corrected triggers;
3. Corrected tables size and position in the PDF;
4. Added and corrected some queries;

**GRUPO1765, 11/04/2018**

* Mariana Duarte Guimarães, [up201307777@fe.up.pt](mailto:up201307777@fe.up.pt)
* Rui Emanuel Cabral de Almeida Quaresma, [up201503005@fe.up.pt](mailto:up201503005@fe.up.pt)
* Rui Pedro Machado Araújo, [up201403263@fe.up.pt](mailto:up201403263@fe.up.pt)
* Tiago Duarte Carvalho, [up201504461@fe.up.pt](mailto:up201504461@fe.up.pt)