A6: Indexes, triggers, user functions

|  |  |  |  |
| --- | --- | --- | --- |
| Relation Reference | Relation Name | Order of magnitude | Estimated growth |
| R01 | user | tens | Units per day |
| R02 | event | tens | Units per day |
| R03 | location | tens | Units per day |
| R04 | image | tens | Units per day |
| R05 | city | tens | Units per day |
| R06 | country | tens | Units per day |
| 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 | tens | Units per day |
| R15 | done | tens | Units per day |
| R16 | notDone | tens | Units per day |
| R17 | admin | units | No growth |
| R18 | friendship | tens | Units per day |
| R19 | eventWarning | tens | Tens per day |
| R20 | currentDate | units | Hundreds per day |

and population

1. Database workload

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

* 1. Tuple Estimation
  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.image\_id, 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\_id, 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", image\_id, "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** username, email, image\_path  **FROM** users  **WHERE** username **LIKE** %$search% **OR** email **LIKE** %$search%  **ORDER** **BY** username; | |

|  |  |
| --- | --- |
| Querie reference | SELECT05 |
| Querie description | Search category |
| Querie frequency | Hundreds per day |
| SELECT id, "name", image\_id, "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\_id, "message"  **FROM** event\_warnings  **WHERE** receiver\_id = $user\_id | |

|  |  |
| --- | --- |
| Querie reference | SELECT07 |
| Querie description | Search category |
| Querie frequency | Hundreds per day |
| **SELECT** id, "name", image\_id, "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 | SELECT08 |
| Querie description | User’s events |
| Querie frequency | Hundreds per day |
| **SELECT** event\_id  **FROM** participants  **WHERE** user\_id = $user\_id  **SELECT** event\_id  **FROM** owners  **WHERE** user\_id = $user\_id | |

|  |  |
| --- | --- |
| Querie reference | SELECT09 |
| Querie description | User’s friends |
| Querie frequency | Hundreds per day |
| **SELECT** event\_id  **FROM** participants  **WHERE** user\_id = $user\_id  **SELECT** event\_id  **FROM** owners  **WHERE** user\_id = $user\_id | |

|  |  |
| --- | --- |
| Querie reference | SELECT010 |
| Querie description | Who can be invited to the 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** **AVG**(rating)  **FROM** dones  **WHERE** dones.event\_id= $event\_id; | |

* 1. 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,  image\_id = $image\_id,  event\_type = $event\_type,  category = $category  WHERE id = $id; | |

|  |  |
| --- | --- |
| Querie reference | UPDATE03 |
| Querie description | Update dones rating |
| Querie frequency | Tens per month |
| UPDATE dones  SET rating = $rating  WHERE event\_id = $event\_id; | |

|  |  |
| --- | --- |
| Querie reference | UPDATE04 |
| Querie description | Update options description |
| Querie frequency | Tens per month |
| UPDATE options  SET description = $description  WHERE id = $id; | |

|  |  |
| --- | --- |
| Querie reference | UPDATE05 |
| Querie description | Update posts information |
| Querie frequency | Hundreds per month |
| UPDATE posts  SET description = $description.  date = $date,  image\_id = $image\_id  WHERE id = $id; | |

|  |  |
| --- | --- |
| Querie reference | UPDATE06 |
| 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 (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)              VALUES ($id,$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 (id,name,date,description,owner\_id,localization\_id,image\_id,type,category) VALUES ($id,$name,$date,$description,$owner\_id,$localization\_id,$image\_id, $type,$category); | |

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

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

|  |  |
| --- | --- |
| 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 (id,user\_id,event\_id)              VALUES ($id,$user\_id,$event\_id); | |

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

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

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

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

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

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

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

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

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

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

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

|  |  |
| --- | --- |
| Querie reference | INSERT18 |
| Querie description | Create new event warning |
| Querie frequency | Tens per month |
| INSERT INTO event\_warnings (id, event\_id, receiver\_id, message) VALUES ($id, $event\_id, $receiver\_id, $message); | |

|  |  |
| --- | --- |
| 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; | |

1. Proposed Indexes
   1. Performance Indexes

|  |  |
| --- | --- |
| Index reference | IDX01 |
| Related queries | SELECT01 |
| Índex relation | user |
| Índex atribute | email |
| Index type | Hash |
| Cardinality | High |
| Clustering | No |
| Justification | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX |
|  | |

|  |  |
| --- | --- |
| Index reference | IDX01 |
| Related queries | SELECT01 |
| Índex relation | user |
| Índex atribute | email |
| Index type | Hash |
| Cardinality | High |
| Clustering | No |
| Justification | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX |
|  | |

|  |  |
| --- | --- |
| Index reference | IDX01 |
| Related queries | SELECT01 |
| Índex relation | user |
| Índex atribute | email |
| Index type | Hash |
| Cardinality | High |
| Clustering | No |
| Justification | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX |
|  | |

* 1. Full-text search indexes

|  |  |
| --- | --- |
| Index reference | IDX01 |
| Related queries | SELECT01 |
| Índex relation | user |
| Índex atribute | email |
| Index type | Hash |
| Cardinality | High |
| Clustering | No |
| Justification | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX |
|  | |

1. 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 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 (NEW.event\_id, NULL);  DELETE FROM not\_dones WHERE id = NEW.event\_id;  END IF;  RETURN NEW;  END  $BODY$  LANGUAGE plpgsql;    CREATE TRIGGER set\_event\_as\_done  BEFORE UPDATE OF date ON current\_date  FOR EACH ROW  WHEN NEW.date = now()  EXECUTE PROCEDURE set\_event\_as\_done(); | |

|  |  |
| --- | --- |
| Trigger reference | TRIGGER02 |
| Trigger description | When an event is deleted a notification is sent to the user |
| CREATE FUNCTION notificate\_event\_delete() RETURNS TRIGGER AS  $BODY$  WHILE( SELECT id FROM participants WHERE participants.event\_id = OLD.event\_id)  BEGIN  INSERT INTO event\_warnings(OLD.event\_id, id)  END  $BODY$  LANGUAGE plpgsql;    CREATE TRIGGER notificate\_event\_delete  FOR DELETE OR UPDATE ON events  FOR EACH ROW  EXECUTE PROCEDURE notificate\_event\_delete(); | |

1. SQL code
2. --Types Enums
3. DROP TYPE IF EXISTS categories CASCADE;
4. CREATE TYPE categories AS ENUM(
5. 'Music',
6. 'Sports',
7. 'Entertainment',
8. 'Educational',
9. 'Business',
10. 'Other'
11. );
12. DROP TYPE IF EXISTS types\_of\_event CASCADE;
13. CREATE TYPE types\_of\_event AS ENUM(
14. 'Public',
15. 'Private'
16. );
17. --Tables
18. DROP TABLE IF EXISTS admins CASCADE;
19. CREATE TABLE admins (
20. id SERIAL NOT NULL,
21. username text NOT NULL,
22. password text NOT NULL,
23. email text NOT NULL,
24. CONSTRAINT admins\_pk PRIMARY KEY (id),
25. CONSTRAINT admins\_email\_uk UNIQUE (email)
26. );
27. DROP TABLE IF EXISTS cities CASCADE;
28. CREATE TABLE cities (
29. id SERIAL NOT NULL,
30. name text NOT NULL,
31. country\_id INTEGER NOT NULL,
32. CONSTRAINT cities\_pk PRIMARY KEY (id),
33. CONSTRAINT cities\_name\_uk UNIQUE (name)
35. );
36. DROP TABLE IF EXISTS current\_date CASCADE;
37. CREATE TABLE current\_date (
38. id SERIAL NOT NULL,
39. date TIMESTAMP WITH TIME zone NOT NULL,
40. CONSTRAINT current\_date\_pk PRIMARY KEY (id),
41. );
42. DROP TABLE IF EXISTS countries CASCADE;
43. CREATE TABLE countries (
44. id SERIAL NOT NULL,
45. name text NOT NULL,
46. CONSTRAINT countries\_pk PRIMARY KEY (id),
47. CONSTRAINT countries\_name\_uk UNIQUE (name)
48. );
49. DROP TABLE IF EXISTS dones CASCADE;
50. CREATE TABLE dones (
51. event\_id INTEGER NOT NULL,
52. rating INTEGER,
53. CONSTRAINT dones\_pk PRIMARY KEY (event\_id),
54. CONSTRAINT rating\_ck CHECK (((rating >= 1) AND (rating <= 5)))
55. );
56. DROP TABLE IF EXISTS events CASCADE;
57. CREATE TABLE events (
58. id SERIAL NOT NULL,
59. name text NOT NULL,
60. date TIMESTAMP WITH TIME zone NOT NULL,
61. description text NOT NULL,
62. owner\_id INTEGER NOT NULL,
63. localization\_id INTEGER NOT NULL,
64. image\_id INTEGER NOT NULL,
65. type event\_type NOT NULL,
66. category categories NOT NULL,
67. CONSTRAINT events\_pk PRIMARY KEY (id),
68. CONSTRAINT date\_ck CHECK ((date > now()))
69. );
70. DROP TABLE IF EXISTS event\_invites CASCADE;
71. CREATE TABLE event\_invites (
72. id SERIAL NOT NULL,
73. answer text NOT NULL,
74. event\_id INTEGER NOT NULL,
75. owner\_id INTEGER NOT NULL,
76. receiver\_id INTEGER NOT NULL,
77. CONSTRAINT event\_invites\_pk PRIMARY KEY (id)
78. );
79. DROP TABLE IF EXISTS event\_warnings CASCADE;
80. CREATE TABLE event\_warnings (
81. id SERIAL NOT NULL,
82. event\_id INTEGER NOT NULL,
83. receiver\_id INTEGER NOT NULL,
84. message text NOT NULL,
85. CONSTRAINT event\_warnings\_pk PRIMARY KEY (id),
86. CONSTRAINT event\_warnings\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES events(id),
87. CONSTRAINT event\_warnings\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES users(id)
88. );
89. DROP TABLE IF EXISTS friend\_activities CASCADE;
90. CREATE TABLE friend\_activities (
91. id SERIAL NOT NULL,
92. sender\_id INTEGER NOT NULL,
93. receiver\_id INTEGER NOT NULL,
94. event\_id INTEGER NOT NULL,
95. CONSTRAINT friend\_activities\_pk PRIMARY KEY (id),
96. CONSTRAINT friend\_activities\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
97. events(id) ON DELETE CASCADE
98. );
99. DROP TABLE IF EXISTS friend\_requests CASCADE;
100. CREATE TABLE friend\_requests (
101. id SERIAL NOT NULL,
102. answer text NOT NULL,
103. sender\_id INTEGER NOT NULL,
104. receiver\_id INTEGER NOT NULL,
105. CONSTRAINT friend\_requests\_pk PRIMARY KEY (id)
106. );
107. DROP TABLE IF EXISTS friendships CASCADE;
108. CREATE TABLE friendships (
109. id SERIAL NOT NULL,
110. user\_id\_1 INTEGER NOT NULL,
111. user\_id\_2 INTEGER NOT NULL,
112. CONSTRAINT friendships\_users\_ids\_uk UNIQUE (user\_id\_1, user\_id\_2)
113. )
114. DROP TABLE IF EXISTS images CASCADE;
115. CREATE TABLE images (
116. id SERIAL NOT NULL,
117. path text NOT NULL,
118. CONSTRAINT images\_pk PRIMARY KEY (id),
119. CONSTRAINT images\_path\_uk UNIQUE (path)
120. );
121. DROP TABLE IF EXISTS localizations CASCADE;
122. CREATE TABLE localizations (
123. id SERIAL NOT NULL,
124. name text NOT NULL,
125. address text NOT NULL,
126. latitude FLOAT,
127. longitude FLOAT,
128. city\_id INTEGER NOT NULL,
129. CONSTRAINT localizations\_pk PRIMARY KEY (id),
130. CONSTRAINT localizations\_city\_id\_fk FOREIGN KEY (city\_id) REFERENCES
131. cities(id) ON DELETE SET NULL
132. );
133. DROP TABLE IF EXISTS not\_dones CASCADE;
134. CREATE TABLE not\_dones (
135. event\_id INTEGER NOT NULL,
136. CONSTRAINT not\_dones\_pk PRIMARY KEY (event\_id),
137. CONSTRAINT not\_dones\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
138. events(id) ON UPDATE CASCADE
139. );
140. DROP TABLE IF EXISTS options CASCADE;
141. CREATE TABLE options (
142. id SERIAL NOT NULL,
143. description text NOT NULL,
144. poll\_id INTEGER NOT NULL,
145. CONSTRAINT options\_pk PRIMARY KEY (id)
146. );
147. DROP TABLE IF EXISTS owners CASCADE;
148. CREATE TABLE owners (
149. id SERIAL NOT NULL,
150. user\_id INTEGER NOT NULL,
151. event\_id INTEGER NOT NULL,
152. CONSTRAINT owners\_pk PRIMARY KEY (id),
153. CONSTRAINT owners\_user\_id\_event\_id\_uk UNIQUE (user\_id, event\_id),
154. CONSTRAINT owners\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
155. events(id) ON DELETE SET NULL
156. );
157. DROP TABLE IF EXISTS participants CASCADE;
158. CREATE TABLE participants (
159. id SERIAL NOT NULL,
160. user\_id INTEGER NOT NULL,
161. event\_id INTEGER NOT NULL,
162. CONSTRAINT participants\_pk PRIMARY KEY (id),
163. CONSTRAINT participants\_user\_id\_event\_id\_uk UNIQUE (user\_id, event\_id),
164. CONSTRAINT participants\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
165. events(id) ON DELETE SET NULL
166. );
167. DROP TABLE IF EXISTS polls CASCADE;
168. CREATE TABLE polls (
169. id SERIAL NOT NULL,
170. post\_id INTEGER NOT NULL,
171. CONSTRAINT polls\_pk PRIMARY KEY (id)
172. );
173. DROP TABLE IF EXISTS posts CASCADE;
174. CREATE TABLE posts (
175. id SERIAL NOT NULL,
176. description text NOT NULL,
177. date TIMESTAMP WITH TIME zone NOT NULL,
178. event\_id INTEGER NOT NULL,
179. user\_id INTEGER NOT NULL,
180. image\_id INTEGER NOT NULL,
181. CONSTRAINT posts\_pk PRIMARY KEY (id),
182. CONSTRAINT posts\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
183. events(id) ON DELETE CASCADE,
184. CONSTRAINT posts\_image\_id\_fk FOREIGN KEY (image\_id) REFERENCES
185. images(id) ON DELETE SET NULL
186. );
187. DROP TABLE IF EXISTS users CASCADE;
188. CREATE TABLE users (
189. id SERIAL NOT NULL,
190. username text NOT NULL,
191. password text NOT NULL,
192. email text NOT NULL,
193. regist\_date TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
194. first\_name text NOT NULL,
195. last\_name text NOT NULL,
196. image\_path text NOT NULL,
197. city\_id INTEGER NOT NULL,
198. CONSTRAINT users\_pk PRIMARY KEY (id),
199. CONSTRAINT users\_name\_uk UNIQUE (username),
200. CONSTRAINT users\_email\_uk UNIQUE (email),
201. CONSTRAINT users\_city\_id\_fk FOREIGN KEY (city\_id) REFERENCES
202. cities(id) ON DELETE SET NULL
203. );
204. ALTER TABLE ONLY cities
205. ADD CONSTRAINT cities\_country\_id\_fk FOREIGN KEY (country\_id) REFERENCES
206. countries(id) ON DELETE SET NULL;
207. ALTER TABLE ONLY dones
208. ADD CONSTRAINT dones\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
209. events(id) ON UPDATE CASCADE;
210. ALTER TABLE ONLY events
211. ADD CONSTRAINT events\_owner\_id\_fk FOREIGN KEY (owner\_id) REFERENCES
212. users(id) ON UPDATE CASCADE;
213. ALTER TABLE ONLY events
214. ADD CONSTRAINT events\_localization\_id\_fk FOREIGN KEY (localization\_id) REFERENCES
215. localizations(id) ON DELETE SET NULL;
216. ALTER TABLE ONLY events
217. ADD CONSTRAINT events\_image\_id\_fk FOREIGN KEY (image\_id) REFERENCES
218. images(id) ON DELETE SET NULL;
219. ALTER TABLE ONLY event\_invites
220. ADD CONSTRAINT event\_invites\_event\_id\_fk FOREIGN KEY (event\_id) REFERENCES
221. not\_dones(event\_id) ON DELETE CASCADE;
222. ALTER TABLE ONLY event\_invites
223. ADD CONSTRAINT event\_invites\_owner\_id\_fk FOREIGN KEY (owner\_id) REFERENCES
224. owners(id) ON DELETE SET NULL;
225. ALTER TABLE ONLY event\_invites
226. ADD CONSTRAINT event\_invites\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES
227. users(id) ON DELETE SET NULL;
228. ALTER TABLE ONLY friend\_activities
229. ADD CONSTRAINT friend\_activities\_sender\_id\_fk FOREIGN KEY (sender\_id) REFERENCES
230. users(id) ON DELETE SET NULL;
231. ALTER TABLE ONLY friend\_activities
232. ADD CONSTRAINT friend\_activities\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES
233. users(id) ON DELETE SET NULL;
234. ALTER TABLE ONLY friend\_requests
235. ADD CONSTRAINT friend\_requests\_sender\_id\_fk FOREIGN KEY (sender\_id) REFERENCES
236. participants(id) ON DELETE CASCADE;
237. ALTER TABLE ONLY friend\_requests
238. ADD CONSTRAINT friend\_requests\_receiver\_id\_fk FOREIGN KEY (receiver\_id) REFERENCES
239. users(id) ON DELETE CASCADE;
240. ALTER TABLE ONLY friendships
241. ADD CONSTRAINT friendships\_user\_id\_1 FOREIGN KEY (user\_id\_1) REFERENCES
242. users(id) ON UPDATE CASCADE;
243. ALTER TABLE ONLY friendships
244. ADD CONSTRAINT friendships\_user\_id\_2 FOREIGN KEY (user\_id\_2) REFERENCES
245. users(id) ON UPDATE CASCADE;
246. ALTER TABLE ONLY options
247. ADD CONSTRAINT options\_poll\_id\_fk FOREIGN KEY (poll\_id) REFERENCES
248. polls(id) ON DELETE CASCADE;
249. ALTER TABLE ONLY owners
250. ADD CONSTRAINT owners\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES
251. users(id) ON UPDATE CASCADE;
252. ALTER TABLE ONLY participants
253. ADD CONSTRAINT participants\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES
254. users(id) ON UPDATE CASCADE;
255. ALTER TABLE ONLY polls
256. ADD CONSTRAINT polls\_post\_id\_fk FOREIGN KEY (post\_id) REFERENCES
257. posts(id) ON DELETE CASCADE;
258. ALTER TABLE ONLY posts
259. ADD CONSTRAINT posts\_user\_id\_fk FOREIGN KEY (user\_id) REFERENCES
260. users(id) ON DELETE CASCADE;
261. UPDATE "users"
262. SET password = $password,
263. email = #email,
264. first\_name = $first\_name,
265. last\_name = $last\_name,
266. image\_path = $image\_path,
267. city\_id = $city\_id
268. WHERE id = $id;
269. UPDATE events
270. SET name = $name,
271. date = $date,
272. description = $description,
273. localization\_id = $localization\_id,
274. image\_id = $image\_id,
275. event\_type = $event\_type,
276. category = $category
277. WHERE id = $id;
278. UPDATE dones
279. SET rating = $rating
280. WHERE event\_id = $event\_id;
281. UPDATE options
282. SET description = $description
283. WHERE id = $id;
284. UPDATE posts
285. SET description = $description.
286. date = $date,
287. image\_id = $image\_id
288. WHERE id = $id;
289. UPDATE current\_date
290. SET date = $date
291. WHERE id = $id;
292. DELETE FROM "users"
293. WHERE id = $id;
294. DELETE FROM events
295. WHERE id = $id;
296. DELETE FROM posts
297. WHERE id = $id;
298. DELETE FROM options
299. WHERE id = $id;
300. DELETE FROM polls
301. WHERE id = $id;
302. DELETE FROM friendships
303. WHERE id = $id;
304. DELETE FROM friend\_requests
305. WHERE id = $id;
306. DELETE FROM event\_invites
307. WHERE id = $id;
308. CREATE FUNCTION set\_event\_as\_done() RETURNS TRIGGER AS
309. $BODY$
310. BEGIN
311. IF EXISTS (SELECT event\_id FROM not\_done WHERE NEW.event\_id = id)
312. THEN
313. INSERT INTO dones (NEW.event\_id, NULL);
314. DELETE FROM not\_dones WHERE id = NEW.event\_id;
315. END IF;
316. RETURN NEW;
317. END
318. $BODY$
319. LANGUAGE plpgsql;
321. CREATE TRIGGER set\_event\_as\_done
322. BEFORE UPDATE OF date ON current\_date
323. FOR EACH ROW
324. WHEN NEW.date = now()
325. EXECUTE PROCEDURE set\_event\_as\_done();
326. CREATE FUNCTION notificate\_event\_delete() RETURNS TRIGGER AS
327. $BODY$
328. WHILE( SELECT id FROM participants WHERE participants.event\_id = OLD.event\_id)
329. BEGIN
330. INSERT INTO event\_warnings(OLD.event\_id, id)
331. END
332. $BODY$
333. LANGUAGE plpgsql;
335. CREATE TRIGGER notificate\_event\_delete
336. FOR DELETE OR UPDATE ON events
337. FOR EACH ROW
338. EXECUTE PROCEDURE notificate\_event\_delete();
340. --> QUERIES
341. --> user profile
342. SELECT username, last\_name, first\_name, email, image\_path, city\_id
343. FROM users
344. WHERE users.id = $user\_id;
346. --> user friends
347. SELECT user\_id\_1, user\_id\_2
348. FROM friendships
349. WHERE user\_id\_1 = $user\_id OR user\_id\_2 = $user\_id;
350. --> user events
351. SELECT event\_id
352. FROM participants
353. WHERE user\_id = $user\_id
354. SELECT event\_id
355. FROM owners
356. WHERE user\_id = $user\_id
357. --> notifications
358. SELECT sender\_id
359. FROM friend\_requests
360. WHERE receiver\_id = $user\_id
361. SELECT sender\_id, event\_id
362. FROM friend\_activities
363. WHERE receiver\_id = $user\_id
364. SELECT owner\_id, event\_id
365. FROM event\_invites
366. WHERE receiver\_id = $user\_id
367. SELECT event\_id, "message"
368. FROM event\_warnings
369. WHERE receiver\_id = $user\_id
370. --> search user
371. SELECT id, username, image\_path
372. FROM users
373. WHERE username LIKE %$search%
374. ORDER BY username;
375. --> search event
376. SELECT id, "name", image\_id, "date", localization, category
377. FROM events
378. WHERE "name" LIKE %$search% OR localization LIKE %$search% AND event\_type = 'public'
379. ORDER BY "name";
380. --> filter by category
381. SELECT id, "name", image\_id, "date", localization, category
382. FROM events
383. WHERE "name" LIKE %$search% OR localization LIKE %$search% AND event\_type = 'public' AND events.category LIKE %$categories%
384. ORDER BY "name";
386. --> search user
387. SELECT username, email, image\_path
388. FROM users
389. WHERE username LIKE %$search% OR email LIKE %$search%
390. ORDER BY username;
391. --> event page
392. SELECT events.id, events.name, events.category, events.image\_id, events.description, events."date" users.username
393. FROM events, users
394. WHERE events.owner\_id = users.id AND events.id = $id;
396. SELECT posts.description, posts.id, posts.image\_id, posts.user\_id
397. FROM posts,events
398. WHERE posts.event\_id = events.id;
400. SELECT users.username, users.image\_path
401. FROM participants
402. WHERE users.id = participants.user\_id AND participants.event\_id=event.id;~
403. --> INSERTS
404. -- Here goes the SQL code - INSERTS
406. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
407. VALUES (1,'sodales','GUL95ZXR9EX','sodales.at@curae.co.uk','2018-03-07 22:23:34','Zeph','Griffin','/imgs/natu.jpg');

410. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
411. VALUES (2,'uso1','12345','aliquam.iaculis.lacus@amet.co.uk','2018-04-07 11:23:34','Ben','Warren','/imgs/natur.jpg');
413. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
414. VALUES (3,'robin','pass123','amet.ante@faucibusleo.net','2018-02-12 15:55:12','Robin','Wright','/imgs/natur.jpg');

417. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
418. VALUES (4,'bar123','semper123','ut.dolor@gmail.com','2017-12-12 23:55:12','Barry','Allen','/imgs/natur.jpg');
420. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
421. VALUES (5,'reddevil','LBAW','Nulla@et.net','2018-02-12 15:55:12','Andrew','Irons','/imgs/november.jpg');
423. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
424. VALUES (6,'rpedro10','lbaw1765','rpedro10@iol.pt','2018-01-01 23:55:12','Rui','Araujo','/imgs/fer.jpg');

427. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
428. VALUES (7,'joss123','CKB15AAW5MM','ante@fleo.com','2018-03-12 12:55:12','Joss','Stone','/imgs/natur.jpg');
430. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
431. VALUES (8,'top123','SXT16TTW3MH','cursus.et@orciUt.co.uk','2018-01-03 13:55:12','Chris','Harris','/imgs/natur.jpg');

434. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
435. VALUES (9,'roland1','ZYS24FHN5GR','eget.dictum@orciDonec.edu','2018-03-12 09:55:12','Roland','Schitt','/imgs/natur.jpg');
437. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
438. VALUES (10,'david123','ZUS29FRTGVJ','amet@faucibusleo.net','2018-01-22 15:07:12','David','Rose','/imgs/natur.jpg');
440. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
441. VALUES (11,'catones', 'QQQ8EFHNGNR','amec.donec@faucibusleo.net','2017-11-12 15:55:12','Carlos','Antonio','/imgs/november.jpg');
443. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
444. VALUES (12,'emanem','ASDFGHJKL','donex@sapo.net','2018-01-13 15:55:12','Raheem','Sterling','/imgs/natur.jpg');
446. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
447. VALUES (13,'ufoExtis','ZXCVBNM','amet@iol.net','2018-02-12 15:55:12','Delle','Alli','/imgs/pyr.jpg');
449. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
450. VALUES (14,'ragnar','QAZWSXEDC','risus.In.mi@egestas.com','2018-02-09 15:55:12','Thor','Ragnarok','/imgs/fer.jpg');
452. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
453. VALUES (15,'seth','QWERTYUIOP','aliquet.diam.Sed@tinciduntnibh.co.uk','2018-01-12 15:55:12','Seth','Byers','/imgs/natu.jpg');
455. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
456. VALUES (16,'edNorton','TYT71DOD7YN','scelerisque.scelerisque.dui@arcuiaculisenim.ca','2018-02-23 15:55:12','Ed','Norton','/imgs/natur.jpg');
458. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
459. VALUES (17,'Pacquiao','SXT16TTW3MH','magnis@cursuset.edu','2018-04-12 15:55:12','Paky','Barret','/imgs/natur.jpg');
461. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
462. VALUES (18,'steven','MPS10QPK6UE','arcu.Vestibulum@amet.org','2018-01-02 12:55:12','Donovan','Stevenson','/imgs/pyr.jpg');
464. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
465. VALUES (19,'porter123','QIG24ZOK3EM','dui.nec@ultriciesadipiscing.co.uk','2018-02-12 15:55:12','Porter','Osborn','/imgs/natu.jpg');
467. INSERT INTO users (id,username,password,email,regist\_date,first\_name,last\_name, image\_path,city\_id)
468. VALUES (20,'human','ZYG87WQA6FX','facilisis.magna.tellus@sociis.net','2018-04-01 11:55:12','Hu','Randolphe','/imgs/pyr.jpg');

471. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
472. VALUES (1,'Antonys Birthday Party', '2018-03-04 12:30:19.000000', 'nunc ac mattis ornare, lectus',1,1,1,'Public','Sports');
474. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
475. VALUES (2,'ENEI','2018-04-04 12:30:19.000000','Nunc quis arcu vel quam',2,2,2,'Public','Sports');
477. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
478. VALUES (3,'RockInRio','2018-06-04 12:30:19.000000','tempus eu, ligula. Aenean euismod',3,3,3,'Public','Sports');
480. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
481. VALUES (4,'Nos Alive','2018-08-04 12:30:19.000000','dignissim pharetra. Nam ac nulla.',3,4,4,'Public','Sports');
483. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
484. VALUES (5,'Christmas Dinner','2018-10-04 12:30:19.000000','dignissim pharetra. Nam ac nulla.',4,5,5,'Public','Sports');
486. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
487. VALUES (6,'Mark Birthday Party','2018-04-04 12:30:19.000000','dignissim pharetra. Nam ac nulla.',1,1,6,'Public','Sports');
489. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
490. VALUES (7,'WebSummit','2018-04-04 12:30:19.000000','lorem lorem, luctus ut, pellentesque',6,6,7,'Public','Sports');
492. INSERT INTO events (id,name,date,description,owner\_id,localization\_id,image\_id,type,category)
493. VALUES (8,'Ted Talk','2018-04-04 12:30:19.000000','sollicitudin orci sem eget massa.',12,2,8,'Public','Sports');


497. INSERT INTO dones (event\_id,rating)
498. VALUES (2,5);
499. INSERT INTO dones (event\_id,rating)
500. VALUES (7,3);
502. INSERT INTO not\_dones (event\_id)
503. VALUES (1);
504. INSERT INTO not\_dones (event\_id)
505. VALUES (3);
506. INSERT INTO not\_dones (event\_id)
507. VALUES (4);
508. INSERT INTO not\_dones (event\_id)
509. VALUES (5);
510. INSERT INTO not\_dones (event\_id)
511. VALUES (6);
512. INSERT INTO not\_dones (event\_id)
513. VALUES (8);

516. INSERT INTO participants (id,user\_id,event\_id)
517. VALUES (1,1,1);
518. INSERT INTO participants (id,user\_id,event\_id)
519. VALUES (2,2,1);
520. INSERT INTO participants (id,user\_id,event\_id)
521. VALUES (3,2,2);
522. INSERT INTO participants (id,user\_id,event\_id)
523. VALUES (4,3,1);
524. INSERT INTO participants (id,user\_id,event\_id)
525. VALUES (5,4,1);
526. INSERT INTO participants (id,user\_id,event\_id)
527. VALUES (6,5,8);
528. INSERT INTO participants (id,user\_id,event\_id)
529. VALUES (7,6,8);
530. INSERT INTO participants (id,user\_id,event\_id)
531. VALUES (8,7,3);
532. INSERT INTO participants (id,user\_id,event\_id)
533. VALUES (9,12,3);
534. INSERT INTO participants (id,user\_id,event\_id)
535. VALUES (10,13,3);
536. INSERT INTO participants (id,user\_id,event\_id)
537. VALUES (11,14,3);
538. INSERT INTO participants (id,user\_id,event\_id)
539. VALUES (12,15,1);
540. INSERT INTO participants (id,user\_id,event\_id)
541. VALUES (13,16,2);
542. INSERT INTO participants (id,user\_id,event\_id)
543. VALUES (14,17,3);
544. INSERT INTO participants (id,user\_id,event\_id)
545. VALUES (15,18,2);
546. INSERT INTO participants (id,user\_id,event\_id)
547. VALUES (16,19,1);
548. INSERT INTO participants (id,user\_id,event\_id)
549. VALUES (17,20,3);
550. INSERT INTO participants (id,user\_id,event\_id)
551. VALUES (18,12,6);
552. INSERT INTO participants (id,user\_id,event\_id)
553. VALUES (19,12,5);
554. INSERT INTO participants (id,user\_id,event\_id)
555. VALUES (20,10,4);


559. INSERT INTO owners (id,user\_id,event\_id)
560. VALUES (1,1,1);
561. INSERT INTO owners (id,user\_id,event\_id)
562. VALUES (2,2,2);
563. INSERT INTO owners (id,user\_id,event\_id)
564. VALUES (3,3,3);
565. INSERT INTO owners (id,user\_id,event\_id)
566. VALUES (4,3,4);
567. INSERT INTO owners (id,user\_id,event\_id)
568. VALUES (5,5,5);
569. INSERT INTO owners (id,user\_id,event\_id)
570. VALUES (6,1,6);
571. INSERT INTO owners (id,user\_id,event\_id)
572. VALUES (7,6,7);
573. INSERT INTO owners (id,user\_id,event\_id)
574. VALUES (8,12,8);

577. INSERT INTO images (id,path) VALUES (1,'/imgs/natur.jpg');
578. INSERT INTO images (id,path) VALUES (2,'/imgs/natu.jpg');
579. INSERT INTO images (id,path) VALUES (3,'/imgs/pyr.jpg');
580. INSERT INTO images (id,path) VALUES (4,'/imgs/november.jpg');
581. INSERT INTO images (id,path) VALUES (5,'/imgs/taj.jpg');
582. INSERT INTO images (id,path) VALUES (6,'/imgs/fer.jpg');
583. INSERT INTO images (id,path) VALUES (7,'/imgs/fa1.jpg');
584. INSERT INTO images (id,path) VALUES (8,'/imgs/fa2.jpg');
585. INSERT INTO localizations (id,name,address,latitude,longitude,city\_id) VALUES (1,'Restaurante O Pirata','Rua da Isabelinha',41.452993,-8.5775364,1);
586. INSERT INTO localizations (id,name,address,latitude,longitude,city\_id) VALUES (2,'FEUP','Rua Roberto Frias',41.1779401,-8.5998763,2);
587. INSERT INTO localizations (id,name,address,latitude,longitude,city\_id) VALUES (3,'Parque da BelaVista','Av. Arlindo Vicente',38.7507558,-9.1265431,3);
588. INSERT INTO localizations (id,name,address,latitude,longitude,city\_id) VALUES (4,'Passeio Maritimo de Alges','Alges',38.697318,-9.2375993,3);
589. INSERT INTO localizations (id,name,address,latitude,longitude,city\_id) VALUES (5,'Hotel Douro','Rua de Agramonte',41.1564707,-8.6288115,2);
590. INSERT INTO localizations (id,name,address,latitude,longitude,city\_id) VALUES (6,'Norte shopping','Matosinhos',41.1825143,-8.6803795,2);
591. INSERT INTO cities (id,name,country\_id) VALUES (1,'Braga',1);
592. INSERT INTO cities (id,name,country\_id) VALUES (1,'Porto',1);
593. INSERT INTO cities (id,name,country\_id) VALUES (1,'Lisboa',1);
594. INSERT INTO countries (id,name) VALUES (1,'Portugal');
595. INSERT INTO countries (id,name) VALUES (2,'Espanha');
596. INSERT INTO countries (id,name) VALUES (3,'USA');
597. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (1,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,1,1);
598. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (2,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,3,1);
599. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (3,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,4,1);
600. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (4,'Lorem ipsum dolor sit amet. ''2018-02-12 15:55:12',,1,16,1);
601. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (5,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',2,18,1);
602. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (6,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',4,10,2);
603. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (7,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',3,17,2);
604. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (8,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',3,14,2);
605. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (9,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',4,5,3);
606. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (10,'Lorem ipsum dolor sit amet. ','2018-01-12 15:55:12',2,16,2);
607. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (11,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',3,12,2);
608. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (12,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',1,15,2);
609. INSERT INTO posts (id,description,date,event\_id, user\_id, image\_id) VALUES (13,'Lorem ipsum dolor sit amet. ','2018-02-12 15:55:12',5,12,3);
610. INSERT INTO polls (id,post\_id) VALUES (1,1);
611. INSERT INTO polls (id,post\_id) VALUES (2,2);
612. INSERT INTO polls (id,post\_id) VALUES (3,3);
613. INSERT INTO polls (id,post\_id) VALUES (4,4);
614. INSERT INTO options (id,description,poll\_id) VALUES (1,'Bar',1);
615. INSERT INTO options (id,description,poll\_id) VALUES (2,'Cafe',1);
616. INSERT INTO options (id,description,poll\_id) VALUES (3,'Club',1);
617. INSERT INTO options (id,description,poll\_id) VALUES (4,'Home',1);
618. INSERT INTO options (id,description,poll\_id) VALUES (5,'12/05/2018',2);
619. INSERT INTO options (id,description,poll\_id) VALUES (6,'13/05/2018',2);
620. INSERT INTO options (id,description,poll\_id) VALUES (7,'Great',3);
621. INSERT INTO options (id,description,poll\_id) VALUES (8,'Good',3);
622. INSERT INTO options (id,description,poll\_id) VALUES (9,'Available',4);
623. INSERT INTO options (id,description,poll\_id) VALUES (10,'Not Available',4);
624. INSERT INTO admin (id,username,password,email) VALUES (1,'admin','admin123','amec.reset@edu.pt');
625. INSERT INTO friend\_requests (id,answer,sender\_id,receiver\_id) VALUES (1,'YES',1,2);
626. INSERT INTO friend\_requests (id,answer,sender\_id,receiver\_id) VALUES (3,'YES',1,3);
627. INSERT INTO friend\_requests (id,answer,sender\_id,receiver\_id) VALUES (4,'NO',1,4);
628. INSERT INTO friend\_requests (id,answer,sender\_id,receiver\_id) VALUES (5,'YES',1,5);
629. INSERT INTO friend\_activities (id,sender\_id,receiver\_id,event\_id) VALUES (1,1,2,1);
630. INSERT INTO friend\_activities (id,sender\_id,receiver\_id,event\_id) VALUES (2,2,3,2);
631. INSERT INTO event\_invites (id,answer,event\_id,owner\_id,receiver\_id) VALUES (1,'YES',1,1,2);
632. INSERT INTO event\_invites (id,answer,event\_id,owner\_id,receiver\_id) VALUES (2,'YES',1,1,3);
633. INSERT INTO event\_invites (id,answer,event\_id,owner\_id,receiver\_id) VALUES (3,'YES',1,1,4);
634. INSERT INTO event\_invites (id,answer,event\_id,owner\_id,receiver\_id) VALUES (4,'YES',1,1,19);
635. INSERT INTO friendships (id, user\_id\_1, user\_id\_2) VALUES (1,2,3);
636. INSERT INTO friendships (id, user\_id\_1, user\_id\_2) VALUES (2,4,7);
637. INSERT INTO event\_warnings (id, event\_id, receiver\_id, message) VALUES (1,4,1,"Canceled");
638. INSERT INTO current\_date (id, date) VALUES (1,'2018-04-04 12:30:19.000000');

**GRUPO1765, 21/03/2018**

* 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)