A9- Main Accesses to the database and transactions

This artefact shows the main accesses to the database, including the transactions.

For each transaction, the isolation level is explicitly stated and read-only transactions are identified to improve global performance. For each identified access, the SQL code and the reference of web resources (A7) are provided.

1. Main Accesses

Main accesses to the database.

1.1. M01: Authentication

SQL101	Creates a new user in the system
Web Resource	R104

```
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);
```

SQL102	Verifies if user/password exists
Web Resource	R102

```
SELECT *
FROM users
WHERE users.username = $user_username AND
users.password=$user_password;
```

1.2. M02: Users

SQL201	Show user profile
Web Resource	R201

```
SELECT username, last_name, first_name, email, image_path, city_id
FROM users WHERE users.id = $user_id;
```

SQL202	Edit user profile
Web Resource	R203

```
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;
```

SQL203	Show user's notifications
Web Resource	R207

```
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;
```

SQL204	Search users
Web Resource	R209

```
SELECT id, username, image_path FROM users
WHERE username LIKE %$search% ORDER BY username;
```

SQL205	View user's friends
Web Resource	

```
SELECT user_id_1, user_id_2 FROM friendships
WHERE user_id_1 = $user_id OR user_id_2 = $user_id;
```

SQL206	Add friend
Web Resource	R204

```
INSERT INTO friend_requests (sender_id, receiver_id)
  VALUES ($sender_id, $receiver_id);
```

SQL207	Remove friend
Web Resource	R206

```
 DELETE FROM friendships WHERE id = $id;
```

1.3. M03: Events

SQL301	View event's information
Web Resource	R301

```
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;
```

SQL302	Create event
Web Resource	R303

```
INSERT INTO events (name,date,description,
owner_id,localization_id,type,category)
VALUES ($name,$date,$description,$owner_id,$localization_id,
$type,$category);
```

SQL303	Search events
Web Resource	R308

```
SELECT id, "name", "date", localization, category
FROM events
WHERE "name" LIKE %$search% OR localization LIKE %$search% AND
event_type = 'public'
ORDER BY "name";
```

SQL304	Invite users to event
Web Resource	R309

```
INSERT INTO event_invites (event_id,owner_id,receiver_id)
VALUES ($event_id,$owner_id,$receiver_id);
```

SQL305	Create post on event
Web Resource	R315

```
INSERT INTO posts (description,date,event_id, user_id, image_path)
VALUES ($description,$date,$event_id, $user_id, $image_path);
```

1.4. M04: Administration

SQL401	Show all users
Web Resource	R401

```
SELECT username, email
FROM users;
```

SQL402	Show all events
Web Resource	R403

```
SELECT events.name, users.username
FROM events, users
WHERE users.id = events.owner_id;
```

2. Transactions

T01	Insert a new event
Isolation Level	REPEATABLE READ
Justification	In order to maintain consistency, it's necessary to use a transaction to ensure that the all the code executes without errors. If an error occurs, a ROLLBACK is issued (when the insertion of an event owner fails, per example). The isolation level is Repeatable Read, because, otherwise, an update of event_id could happen, due to an insert in the table events committed by a concurrent transaction, and as a result, inconsistent data would be stored.

```
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

-- Insert event
 INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
VALUES ($name,$date,$description,$owner_id,$localization_id,
$type,$category);

-- Insert owner
 INSERT INTO owners (user_id,event_id) VALUES ($user_id,$event_id);

COMMIT;
```

Grupo1765, 25/04/2018

- Mariana Duarte Guimarães, up201307777@fe.up.pt
- Rui Emanuel Cabral de Almeida Quaresma, up201503005@fe.up.pt
- Rui Pedro Machado Araújo, up201403263@fe.up.pt
- Tiago Duarte Carvalho, up201504461@fe.up.pt