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. M01: Authentication

|  |  |
| --- | --- |
| SQL101 | Creates a new user in the system |
| Web Resource | R103 |
| 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 | Log user into the system |
| Web Resource | R101 |
| SELECT username, last\_name, first\_name, email, image\_path, city\_id FROM users  WHERE users.username = $user\_username AND users.password=$user\_password; | |

* 1. M02: Users

|  |  |
| --- | --- |
| SQL103 | Show user profile |
| Web Resource | R201 |
| SELECT username, last\_name, first\_name, email, image\_path, city\_id  FROM users WHERE users.id = $user\_id; | |

|  |  |
| --- | --- |
| SQL104 | 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; | |

|  |  |
| --- | --- |
| SQL105 | Show user’s notifications |
| Web Resource |  |
| 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; | |

|  |  |
| --- | --- |
| SQL106 | Search users |
| Web Resource | R209 |
| SELECT id, username, image\_path FROM users  WHERE username LIKE %$search% ORDER BY username; | |

|  |  |
| --- | --- |
| SQL107 | 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; | |

|  |  |
| --- | --- |
| SQL108 | Add friend |
| Web Resource | R201 |
| INSERT INTO friend\_requests (sender\_id,receiver\_id)  VALUES ($sender\_id,$receiver\_id); | |

|  |  |
| --- | --- |
| SQL109 | Remove friend |
| Web Resource | R201 |

DELETE FROM friendships WHERE id = $id;

* 1. M03: Events

|  |  |
| --- | --- |
| SQL201 | View event’s information |
| Web Resource | UI10 |
| 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; | |

|  |  |
| --- | --- |
| SQL202 | Create event |
| Web Resource |  |
| INSERT INTO events (name,date,description,  owner\_id,localization\_id,type,category)  VALUES ($name,$date,$description,$owner\_id,$localization\_id, $type,$category); | |

|  |  |
| --- | --- |
| SQL203 | 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"; | |

|  |  |
| --- | --- |
| SQL204 | Show event’s rating |
| Web Resource |  |
| SELECT rating  FROM dones  WHERE dones.event\_id= $event\_id; | |

|  |  |
| --- | --- |
| SQL205 | Invite users to event |
| Web Resource |  |
| INSERT INTO event\_invites (event\_id,owner\_id,receiver\_id)  VALUES ($event\_id,$owner\_id,$receiver\_id); | |

|  |  |
| --- | --- |
| SQL206 | 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); | |

|  |  |
| --- | --- |
| SQL207 | Create poll on event |
| Web Resource |  |
| INSERT INTO polls (post\_id) VALUES ($posts\_id);  INSERT INTO options (description,poll\_id) VALUES ($description,$poll\_id); | |

|  |  |
| --- | --- |
| SQL208 | Invite users to event |
| Web Resource | R310 |
| INSERT INTO event\_invites (event\_id,owner\_id,receiver\_id)  VALUES ($event\_id,$owner\_id,$receiver\_id); | |

* 1. M04: Administration

|  |  |
| --- | --- |
| SQL301 | Show all users |
| Web Resource | UI04 |
| SELECT username  FROM users; | |

|  |  |
| --- | --- |
| SQL302 | Show all events |
| Web Resource | UI04 |
| SELECT name  FROM events; | |

|  |  |
| --- | --- |
| SQL303 | Ban user |
| Web Resource | R401 |
| DELETE FROM "users" WHERE id = $id; | |

1. 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 a 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; | |

|  |  |
| --- | --- |
| T02 |  |
| Isolation Level |  |
| Justification |  |
|  | |