A9: Main accesses to the database and transactions

Our project features an information system capable of supporting an online store, which would allow users to buy products from a wide range of categories. In this artefact we present its main accesses to the database, including transactions.

1 Main accesses to the database

Main accesses to the database.

1.1 M01: Authentication and Individual Profile

SQL101	Creates a new user in the platform
Web Resource	R105

```
INSERT INTO users(id, firstName, lastName, username, email, password, imageURL,
        dateCreated, dateModified, active, remememember_token)
VALUES ($id, $firstName, $lastName, $username, $email, $password, $imageURL,
        DEFAULT, DEFAULT, true, $token);
```

Table 1: Authentication and Individual Profile

1.2 M02: Search products

SQL102	Searches products by name and category
Web Resource	R207

```
SELECT p.id AS "ID",p.name AS "Name", p.quantityInStock AS "Quantity In Stock",
           p.dateCreated AS "Date Created", p.price AS "Price",
           p.imageURL AS "Image URL", p.bigDescription AS "Big Description",
       p.shortDescription AS "Short Description",
           (array_agg(pc.categoryName ORDER BY p.id DESC))[1] AS "Category",
       (array_agg(b.brandname ORDER BY p.id DESC))[1] AS "Brand",
       AVG(pr.rating) AS "Rating", ts_rank_cd(document, query) AS rank
FROM products p, categories pc, brands b, reviews pr,
to_tsvector(pc.categoryName || ' ' || p.name) AS document,
plainto_tsquery($query) AS query
WHERE p.id_brand = b.id_brand AND p.id_category = pc.id_category
AND pr.id_product = p.id AND document  @@ query
GROUP BY p.id, document.document, query.query
ORDER BY  rank DESC;
```

Table 2: Search products

1.3 M03: Cart

SQL103	Adds product to cart
Web Resource	R402

```
INSERT INTO carts (id_client, id_product, quantity)
VALUES ($id_client, $id_product, $quantity);
```

Table 3: Cart

1.4 M04: Consult purchased products

SQL103	Retrieve the products bought by an user
Web Resource	R404

```
SELECT products.name, products.price, products.imageURL,
purchaseproducts.quantity, purchaseproducts.cost
FROM purchases, products, purchaseproducts
        WHERE purchases.id = purchaseproducts.id_purchase
        AND products.id = purchaseproducts.id_product
        AND purchases.id_client = $id_client;
```

Table 4: Purchase

1.5 M05: Make a Purchase

SQL103	Retrieve the products bought by an user
Web Resource	R405

```
INSERT INTO purchases(id, id_client, id_address, purchaseDate, purchaseState,
        cost, paymentType, cardNumber, cardName, cardExpirationDate, nif)
VALUES ($id, $id_client,$id_address, DEFAULT, $purchaseState,
        $cost, $paymentType, $cardNumber, $cardName, $cardExpirationDate, $nif);
```

Table 5: Purchase

2 Transactions

Transactions needed to ensure the integrity of the data, with a proper justification.

T01	Retrieve the products bought by an user
Isolation Level	SERIALIZABLE READ ONLY
Justification	In order for the information retrieved in both SELECTS to be the same we must assure that no new rows can be inserted in the table *purchases*, that is, it must be assured that no *Phantom Read's* can occur. That's why the isolation level of this transaction must be SERIALIZABLE. It is also READ ONLY as only SELECTS are used.

```
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY;

--Get number of purchases
SELECT COUNT(*)
FROM purchases ;

--Get products purchased
SELECT products.name, products.price, products.imageURL,
purchaseproducts.quantity, purchaseproducts.cost
FROM purchases, products, purchaseproducts
        WHERE purchases.id = purchaseproducts.id_purchase
        AND products.id = purchaseproducts.id_product
        AND purchases.id_client = $id_client;

COMMIT;
```

Table 6: Purchase

T02	Inserts a new client
Isolation Level	REPEATABLE READ
Justification	In order to maintain consistency both INSERTS need to be executed. If an error occurs, a ROLLBACK is issued. The isolation level is REPEATABLE READ as it must be assured that an update of the attribute *id* in table *users* does not occur in at the same time of this transaction, because it would trample the data.

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

INSERT INTO users(id, firstName, lastName, username, email, password, imageURL,
        dateCreated, dateModified, active, remember_token)
VALUES ($id, $firstName, $lastName, $username, $email, $password, $imageURL,
        DEFAULT, DEFAULT, true, $token);

INSERT INTO clients($id, $cellphone);

COMMIT;
```

Table 7: Purchase

T03	Buy a product
Isolation Level	REPEATABLE READ
Justification	In order to maintain consistency both INSERTS need to be executed. If an error occurs, a ROLLBACK is issued. The isolation level is REPEATABLE READ as it must be assured that an update of the attribute *id* in table *purchases* does not occur in at the same time of this transaction, because it could trample the data.

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

INSERT INTO purchases(id, id_client, id_address, purchaseDate, purchaseState,
        cost, paymentType, cardNumber, cardName, cardExpirationDate, nif)
VALUES ($id, $id_client,$id_address, DEFAULT, $purchaseState,
        $cost, $paymentType, $cardNumber, $cardName, $cardExpirationDate, $nif);

INSERT INTO purchaseproducts (id_purchase, id_product, quantity, cost)
VALUES ($id, $id_product, $quantity, $cost);

COMMIT;
```

Table 8: Purchase

GROUP1736, 21/04/2018

- Group member 1 Beatriz de Henriques Martins, up201502858@fe.up.pt
- Group member 2 Francisco Tuna Andrade, up201503481@fe.up.pt
- Group member 3 Luís Miguel Santos Monteiro Saraiva, up201404302@fe.up.pt
- Group member 4 Ricardo Filipe Amaro Saleiro Abreu, up201304450@fe.up.pt