A9: Acessos principais à base de dados e transações

Este artefacto tem dois elementos principais: o primeiro é a identificação de, para cada módulo definido no artefacto A7, quais os acessos principais à base de dados realizados em cada módulo. Cada acesso foi documentado com o respectivo código SQL e a referência para o(s) recursos Web a que está associado. O segundo elemento é a definição de transações para acessos à base de dados que delas necessitem, tendo-se justificado, para cada transação, a razão pela qual é necessária e o motivo da escolha do seu nível de isolamento. É incluído também o código SQL necessário para a implementação de cada transação.

1. Acessos principais

M01: Autentificação e perfil individual

SQL101	Criar um novo utilizador
Recurso Web	R105

```
SELECT *
FROM member, image, comment
WHERE member.idImage = image.id AND member.id = $userID;

SELECT datePosted, liked, text, idParent, idSender, idReceiver, username
FROM comment, member
WHERE comment.idParent = $profileID AND is_removed = false AND comment.idSender =
member.id;
```

SQL102	Obter perfil de um utilizador
Recurso Web	R106

```
SELECT *
FROM member, image, comment
WHERE member.idImage = image.id AND member.id = $userID;

SELECT datePosted, liked, text, idParent, idSender, idReceiver, username
FROM comment, member
```

```
WHERE comment.idParent = $profileID AND is_removed = false AND comment.idSender =
member.id;
```

M02: Leilões

SQL201	Obter leilões usando múltiplos parâmetros
Recurso Web	R203

```
--Search by language
SELECT auction.*, image.source
FROM auction, language, image
WHERE auction.idLanguage = language.id AND language.language = $langName AND
image.idAuction = auction.idAuction;
```

```
--Search by fields
SELECT auction.*, image.source
FROM auction, image
WHERE isbn = $isbn OR author = $author OR title = $title;
```

SQL202	Obter a informação completa de um leilão
Recurso Web	R204

```
SELECT auction.*, member.username
FROM auction, member
WHERE auction.id = $auctionID AND idSeller = member.id;
```

```
SELECT source
FROM image
WHERE idAuction = $auctionID;
```

SQL203	Obter o valor atual das licitações num leilão
Recurso Web	R210

```
SELECT max(bidValue)
```

```
FROM bid
WHERE idAuction = $auctionID;
```

SQL204	Licitar num leilão
Recurso Web	R209

```
UPDATE bid
SET bidValue = $newValue, bidDate = now()
WHERE idBuyer = $userID AND idAuction = $auctionID;
```

M03: Listas do utilizador autenticado

SQL301	Marcar um leilão como favorito
Recurso Web	R302

```
INSERT INTO wishlist (idBuyer, idAuction)
VALUES ($userID, $idAuction);
```

M04: Comunicação

SQL401	Obter notificações não lidas
Recurso Web	R402

```
SELECT notifications.*
FROM notifications
WHERE idMember = $userID AND is_seen = false;
```

SQL402	Marcar uma notificação como lida
Recurso Web	R403

```
UPDATE notification
```

```
SET is_seen = TRUE, dateSeen = now()
WHERE idMember = $userID;
```

M05: Autentificação e área de gestão admin

SQL501	Mudar o *status* de um utilizador
Recurso Web	R506, R507, R508, R509, R510

```
UPDATE member
SET member_status = $status
WHERE idMember = $userID;
```

M06: Moderação de leilões

SQL601	Aprovar um leilão
Recurso Web	R602

```
UPDATE aution
SET auction_status = 'approved', dateApproved = now()
WHERE id = $auctionID;
```

2. Transações

T01	Criar um novo utilizador
Justificação	De modo a manter a consistência da base de dados, é necessário que todo o código execute sem erros e que ambos os tuplos sejam inseridos, sendo que se provoca um *Rollback* caso a inserção em pelo menos uma das tabelas falhe. O nível de isolamento é *READ COMMITTED* visto que não queremos que outra transação leia o tuplo inserido em *users* sem este ter o tuplo em *image* a ele associado também inserido e *committed* (ou seja, não queremos *dirty reads*).
Nível de isolamento	READ COMMITTED

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

-- Insert users
 INSERT INTO users (address, age, email, name, password, phone, postalCode,
username, idCountry) VALUES ($address, $age, $email, $name, $password, $phone,
$postalCode, $username, $idCountry);

-- Insert image
  INSERT INTO image (source, idusers) VALUES ($source, $idusers);

COMMIT;
```

T02	Terminar uma conta
Justificação	De modo a manter a consistência da base de dados, é necessário que todo o código execute sem erros e que ambos os tuplos sejam inseridos, sendo que se provoca um *Rollback* caso a inserção em pelo menos uma das tabelas falhe. O nível de isolamento é *READ UNCOMMITTED* visto que a camada lógica de negócio garante que não há problemas caso sejam lidos dados inconsistentes enquanto a transação decorre.
Nível de isolamento	READ UNCOMMITTED

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

-- Set user as terminated
  UPDATE users SET users_status = 'terminated' WHERE id = $userID;

-- Delete user profile image
  DELETE FROM requested_termination WHERE idusers = $idRequest;

COMMIT;
```

T03	Criar um novo leilão
	De modo a manter a consistência da base de dados, é necessário que todo o código execute sem erros e que ambos os tuplos sejam inseridos, sendo que se provoca um *Rollback* caso a inserção em pelo menos uma

Justificação	das tabelas falhe. O nível de isolamento é *SERIALIZABLE* porque não queremos permitir *queries* do tipo SELECT...WHERE (que são bastante pervalentes nestas tabelas) enquanto esta transação não estiver totalmente concluída. Isto evita a ocorrência de *phantom reads* na leitura de, por exemplo, todos os novos leilões.
Nível de isolamento	SERIALIZABLE

```
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

-- Insert auction
 INSERT INTO auction (author, description, duration, ISBN, title, idPublisher,
idLanguage, idSeller) VALUES ($author, $description, $duration, $ISBN, $title,
$idPublisher, $idLanguage, $userID);

-- Insert image
 INSERT INTO image (source, idAuction) VALUES ($source, $idAuction);

COMMIT;
```

T04	Criar um pedido de edição de leilão
Justificação	De modo a manter a consistência da base de dados, é necessário que todo o código execute sem erros e que ambos os tuplos sejam inseridos, sendo que se provoca um *Rollback* caso a inserção em pelo menos uma das tabelas falhe. O nível de isolamento é *READ COMMITTED* visto que não queremos que outra transação leia o tuplo inserido em *auction_modification* sem este ter o tuplo em *image* a ele associado também inserido e *committed* (ou seja, não queremos *dirty reads*).
Nível de isolamento	READ COMMITTED

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

-- Insert auction
  INSERT INTO auction_modification (newDescription, idApprovedAuction) VALUES
($newDescription, $idApprovedAuction);

-- Insert image
  INSERT INTO image (source, idAuctionModification) VALUES ($source,
```

```
$idAuctionModification);

COMMIT;
```

GROUP1726, 18/4/2018

> Daniel Vieira Azevedo, up201000307@fe.up.pt

> Nelson André Garrido da Costa, up201403128@fe.up.pt

> Rúben José da Silva Torres, up201405612@fe.up.pt

> Tiago Lascasas dos Santos, up201503616@fe.up.pt