2. EBD: Database Specification

EBD: Database Specification Component

A4: Conceptual Data Model

> The Conceptual Data Model represents all the relations between the different objects and how they are connected, through an UML Diagram

1. Class diagram

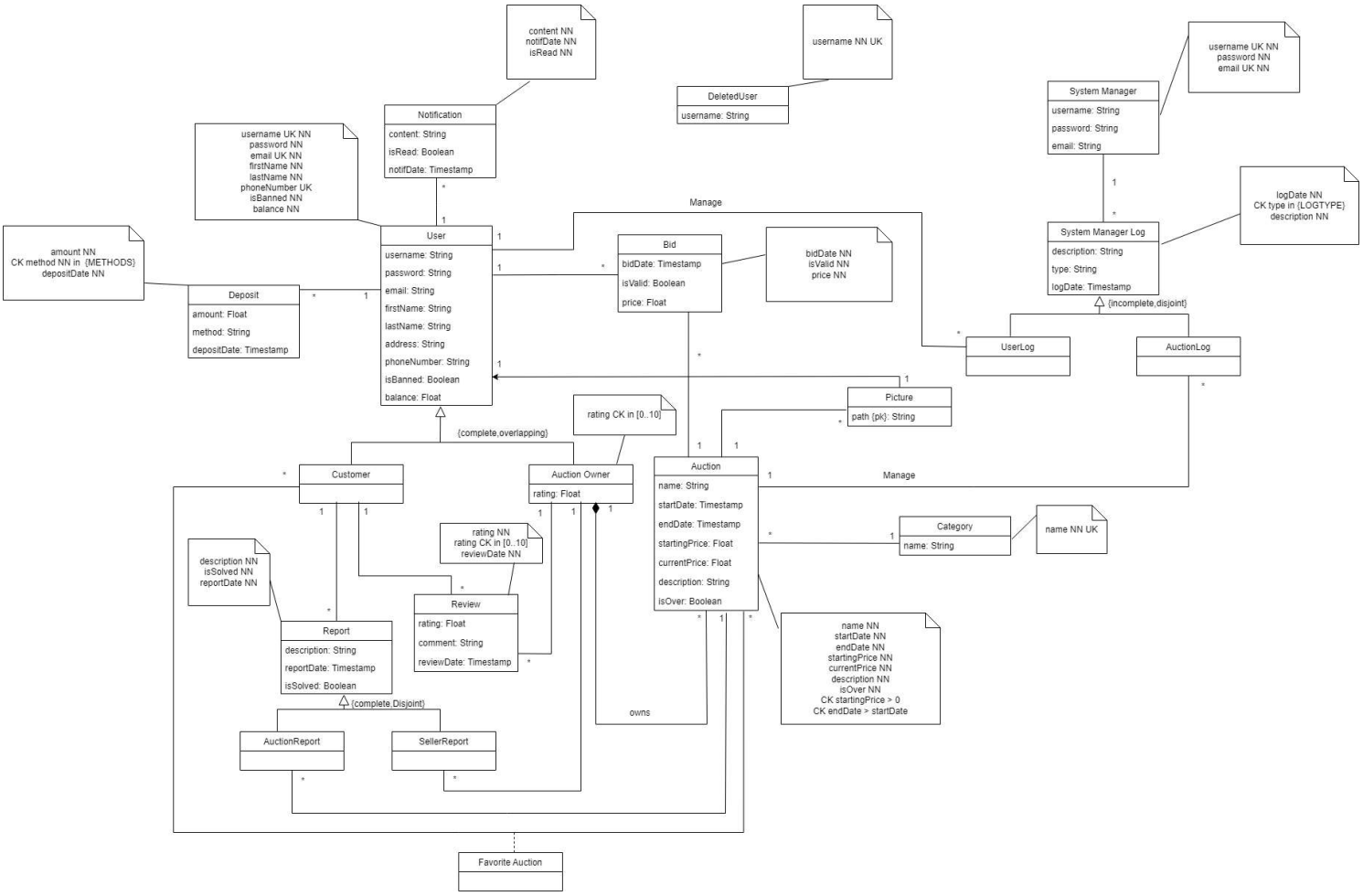

Fig1: WeBid Conceptual Model

2. Additional Business Rules

> One additional business rule was created.

Identifier	Name	Description
BR07	Active Bid or Auctions on Account Deletion	A User can't delete his account if he has active bids or auctions.

A5: Relational Schema, validation and schema refinement

> This artifact has the relation schema for the UML Diagram proposed above.

1. Relational Schema

Relation reference	Relation Compact Notation
R01	Client(idClient, username **NN UK**, email **NN UK**, password **NN**, firstName **NN**, lastName **NN**, address, phoneNumber **UK**, isBanned **NN**, balance **NN**)

Relation reference	Relation Compact Notation
R02	Category(idCategory, name **NN UK**)
R03	AuctionOwner(idClient → Client, rating **CK** (rating > 0 AND rating <= 10) OR rating is NULL)
R04	Auction(idAuction, name **NN**, startDate **NN**, endDate **NN CK** endDate > startDate, startingPrice **NN CK** startingPrice > 0, currentPrice **NN**, description **NN**, isOver **NN**, idCategory → Category, idOwner → AuctionOwner)
R05	Review(idReview, rating **NN CK** rating > 0 AND rating <= 10, comment, reviewDate **NN**, idUserReviewer → Client, idUserReviewed → AuctionOwner)
R06	Bid(idBid, bidDate **NN**, isValid **NN**, price **NN**, idClient → Client **NN**, idAuction → Auction **NN**)
R07	FavoriteAuction(idClient → Client, idAuction → Auction)
R08	SellerReport(idReport, reportDate **NN**, description **NN**, isSolved **NN**, idSeller → AuctionOwner, idReporter → Client
R08	AuctionReport(idReport, reportDate **NN**, description **NN**, isSolved **NN**, idAuction → Auction, idReporter → Client)
R09	SystemManager(idSystemManager, username **NN UK**, email **NN UK**, password **NN**)
R10	SystemManagerLog(idSysLog, logDate **NN**, logDescription **NN**, logType **NN CK** logType in {LogType}, idSysMan → SystemManager)
R11	Deposit(idDeposit, amount **NN**, method **NN CK** method in {Method}, depositDate **NN**, idClient → Client)
R12	Notification(idNotification, content **NN**, isRead **NN**, notifDate **NN**, idClient → Client)
R13	UserLog(idSysLog → SystemManagerLog, idClient → Client)
R14	AuctionLog(idSysLog → SystemManagerLog, idAuction → Auction)
R15	DeletedUser(idClient, username **NN UK**)

Caption:

- NN = NOT NULL
- UK = UNIQUE KEY
- CK = CHECK

2. Domains

> The following domains that were used can be defined as following:

Domain Name	Domain Specification
Methods	ENUM ('PAYPAL', 'MBWAY', 'CRYPTO', 'CREDIT CARD', 'BANK TRANSFER')
Logtype	ENUM ('Ban', 'Unban', 'Delete', 'Other')

3. Schema validation

> The identification of all functional dependencies and the normalization of all relation schemas are completed in order to validate the Relational Schema derived from the Conceptual Model. If normalizing is necessary to reach the correct BCNF, then the schema can be refined by applying additional transformations or by adding new relation types. The schema obtained through this process is stored and analyzed in order to identify the most important relations and attributes.

TABLE R01	**Client**
Keys	{idClient}, {email}, {username}, {phoneNumber}
Functional Dependencies:	
FD0101	idClient → {username, password, email, firstName, lastName, address, phoneNumber, isBanned, balance}
FD0102	email → {idClient, username, password, firstName, lastName, address, phoneNumber, isBanned, balance}
FD0103	username → {idClient, password, email, firstName, lastName, address, phoneNumber, isBanned, balance}

TABLE R01	Client
FD0103	phoneNumber → {idClient, username, password, email, firstName, lastName, address, isBanned, balance}
NORMAL FORM	BCNF

TABLE R02	Category
Keys	{idCategory}, {name}
Functional Dependencies:	
FD0201	idCategory → {name}
FD0202	name → {idCategory}
NORMAL FORM	BCNF

TABLE R03	AuctionOwner
Keys	{idClient}, {rating}
Functional Dependencies:	
FD0301	idClient → {rating}
NORMAL FORM	BCNF

TABLE R04	Auction
Keys	{idAuction}
Functional Dependencies:	
FD0401	idAuction → {name, startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner}
NORMAL FORM	BCNF

TABLE R05	Review
Keys	{idReview}
Functional Dependencies:	
FD0501	idReview → {rating, comment, reviewDate, idUserReviewer, idUserReviewed}
NORMAL FORM	BCNF

TABLE R06	Bid
Keys	{idBid}
Functional Dependencies:	
FD0601	idBid → {bidDate, isValid, price, idClient, idAuction}
NORMAL FORM	BCNF

TABLE R07	FavoriteAuction
Keys	{idClient, idAuction}
Functional Dependencies:	
NO FDS	

TABLE R07	FavoriteAuction
NORMAL FORM	BCNF

TABLE R08	SellerReport
Keys	{idReport}
Functional Dependencies:	
FD0801	idReport → {reportDate, description, isSolved, idSeller, idReporter}
NORMAL FORM	BCNF

TABLE R09	AuctionReport
Keys	{idReport}
Functional Dependencies:	
FD0901	idReport → {reportDate, description, isSolved, idAuction, idReporter}
NORMAL FORM	BCNF

TABLE R10	SystemManager
Keys	{idSysMan}, {username}, {email}
Functional Dependencies:	
FD1001	idSysMan → {username, email}
FD1002	username → {idSysMan, email}
FD1003	email → {idSysMan, username}
NORMAL FORM	BCNF

TABLE R11	SystemManagerLog
Keys	{idSysLog}
Functional Dependencies:	
FD1101	idSysLog → {logDate, logDescription, logType, idSysMan}
NORMAL FORM	BCNF

TABLE R12	Deposit
Keys	{idDeposit}
Functional Dependencies:	
FD1201	idDeposit → {idDeposit, amount, method, depositDate, idClient}
NORMAL FORM	BCNF

TABLE R13	Notification
Keys	{idNotification}
Functional Dependencies:	
FD1301	idNotification → {content, isRead, notifDate, idClient}

TABLE R13	Notification
NORMAL FORM	BCNF

TABLE R14	UserLog
Keys	{idSysLog, idClient}
Functional Dependencies:	
NO FDS	
NORMAL FORM	BCNF

TABLE R15	AuctionLog
Keys	{idSysLog, idAuction}
Functional Dependencies:	
NO FDS	
NORMAL FORM	BCNF

TABLE R16	DeletedUser
Keys	{idClient}, {username}
Functional Dependencies:	
FD1601	idClient → {username}
FD1602	username → {idClient}
NORMAL FORM	BCNF

A6: Indexes, triggers, transactions and database population

> The description of the database user-defined functions, the identification and characterization of the indexes, the support for data integrity rules using triggers, and the physical structure of the database are all contained in this item. This artifact also includes the workload for the database and the whole script for creating the database, and all SQL required to establish all integrity constraints, indexes, and triggers. Finally, a second script containing INSERT statements to fill the database is also included in this item.

1. Database Workload

Relation reference	Relation Name	Order of magnitude	Estimated growth
R01	Client	1k	10/day
R02	Auction	100	10/day
R03	Review	100	10/day
R04	AuctionOwner	100	10/day
R05	Bid	1k	100/day
R06	Category	10	0
R07	FavouriteAuction	100	10/day
R08	SellerReport	100	10/day
R09	AuctionReport	100	10/day
R01	SystemManager	10	1/day

Relation reference	Relation Name	Order of magnitude	Estimated growth
R11	SystemManagerLog	100	10/day
R12	Deposit	100	10/day
R13	Notification	1k	100/day

2. Proposed Indices

2.1. Performance Indices

IDX01

Index	IDX01
Relation	Notification
Attribute	idClient
Type	B-Tree
Cardinality	High
Clustering	No
Justification	Used to select the notifications that are related to a single user. Will have to be queried multiple times.

SQL CODE

```
CREATE INDEX id_client ON Notification(idClient);
```

IDX02

Index	IDX02
Relation	Auction
Attribute	idCategory
Type	B-Tree
Cardinality	Low
Clustering	No
Justification	Used to filter auctions by category.

SQL CODE

```
CREATE INDEX auction_category ON Auction(idCategory);
```

IDX03

Index	IDX03
Relation	Client
Attribute	username
Type	Hash
Cardinality	Low
Clustering	No
Justification	Used Hash indexing because it will be helpful to search for an exact username.

SQL CODE

```
CREATE INDEX user_username ON User USING hash(username);
```

2.2. Full-text Search Indices

Index	IDX01
Relation	Auction
Attribute	Name & Description
Type	GIN
Cardinality	Low
Clustering	No
Justification	Should include full-text search capabilities so users may seek for Auctions by Name or Description. Since the indexed fields are not anticipated to change frequently, the index type is GIN.

SQL CODE

```
ALTER TABLE Auction
ADD COLUMN tsvectors TSVECTOR;

DROP FUNCTION IF EXISTS Auction_search_update() CASCADE;

CREATE FUNCTION Auction_search_update() RETURNS TRIGGER AS $$
BEGIN
 IF TG_OP = 'INSERT' THEN
        NEW.tsvectors = (
         setweight(to_tsvector('english', NEW.name), 'A') ||
         setweight(to_tsvector('english', NEW.description), 'B')
        );
 END IF;
 IF TG_OP = 'UPDATE' THEN
         IF (NEW.name <> OLD.name OR NEW.description <> OLD.description) THEN
           NEW.tsvectors = (
             setweight(to_tsvector('english', NEW.name), 'A') ||
             setweight(to_tsvector('english', NEW.description), 'B')
           );
         END IF;
 END IF;
 RETURN NEW;
END $$
LANGUAGE plpgsql;

CREATE TRIGGER Auction_search_update
 BEFORE INSERT OR UPDATE ON Auction
 FOR EACH ROW
 EXECUTE PROCEDURE Auction_search_update();


CREATE INDEX search_idx ON Auction USING GIN (tsvectors);
```

3. Triggers

TRIGGER01

Trigger	TRIGGER01
Description	A user can't bid on a auction he owns.

SQL CODE

```
DROP FUNCTION IF EXISTS check_bid() CASCADE;

CREATE FUNCTION check_bid() RETURNS TRIGGER AS
$BODY$
BEGIN
    IF
        (NEW.idClient = (SELECT idOwner from Auction WHERE(Auction.idAuction = New.IdAuction)))
    THEN
        RAISE EXCEPTION 'Cannot bid on your own auction';
END IF;
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;


CREATE TRIGGER check_bid
BEFORE INSERT ON Bid
FOR EACH ROW
EXECUTE PROCEDURE check_bid();
```

TRIGGER02

Trigger	**TRIGGER02**
Description	A user can't review himself.

SQL CODE

```
DROP FUNCTION IF EXISTS create_review() CASCADE;

CREATE FUNCTION create_review() RETURNS TRIGGER AS
$BODY$

BEGIN
    IF
        (NEW.idUserReviewer = NEW.idUserReviewed)
    THEN
        RAISE EXCEPTION 'Cannot review yourself';
END IF;
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;


CREATE TRIGGER create_review
BEFORE INSERT ON Review
FOR EACH ROW
EXECUTE PROCEDURE create_review();
```

TRIGGER03

Trigger	**TRIGGER03**
Description	A user can't bid an amount lower than the current bid and, if a user bids on a auction, it updates the current price of that auction.

SQL CODE

```
DROP FUNCTION IF EXISTS create_bid() CASCADE;

CREATE FUNCTION create_bid() RETURNS TRIGGER AS
$BODY$
```

```
BEGIN
    IF
        (NEW.price < (SELECT currentprice FROM Auction
    WHERE(Auction.idAuction = NEW.idAuction)))
    THEN
        RAISE EXCEPTION 'Value of the bid is lower than the highest bid on the auction';
    ELSE
        UPDATE Auction SET currentprice = NEW.price WHERE (Auction.idAuction = NEW.idAuction);
END IF;
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;


CREATE TRIGGER create_bid
BEFORE INSERT ON Bid
FOR EACH ROW
EXECUTE PROCEDURE create_bid();
```

TRIGGER04

Trigger	TRIGGER04
Description	Inserts a client in the DeletedUser table after deleting his account.

SQL CODE

```
DROP FUNCTION IF EXISTS client_delete() CASCADE;

CREATE FUNCTION client_delete() RETURNS TRIGGER AS
$BODY$
BEGIN
    insert into DeletedUser (idClient, username) values (old.idClient, old.username);
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;


CREATE TRIGGER client_delete
AFTER DELETE ON Client
FOR EACH ROW
EXECUTE PROCEDURE client_delete();
```

TRIGGER05

Trigger	TRIGGER05
Description	Updates an Auction Owner's review score after he receives a new review.

SQL CODE

```
DROP FUNCTION IF EXISTS change_rating() CASCADE;

CREATE FUNCTION change_rating() RETURNS TRIGGER AS
$BODY$

BEGIN
    UPDATE AuctionOwner SET rating = (Select round(sum(rating * 1.0)/count(*),2) from Review where Review.idUserReviewed = New.idUserR
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;
```

```
CREATE TRIGGER change_rating
AFTER INSERT ON Review
FOR EACH ROW
EXECUTE PROCEDURE change_rating();
```

TRIGGER06

Trigger	TRIGGER06
Description	After an user is outbid, notify him.

SQL CODE

```
DROP FUNCTION IF EXISTS high_notif() CASCADE;

CREATE FUNCTION high_notif() RETURNS TRIGGER AS
$BODY$

BEGIN
    INSERT INTO Notification(content,isRead,notifDate,idClient)
    (SELECT CONCAT('Outbid on Auction "', Auction.name, '"'), False, NOW(), idClient from bid, Auction where bid.idauction = NEW.IdAuc
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER high_notif
AFTER INSERT ON Bid
FOR EACH ROW
EXECUTE PROCEDURE high_notif();
```

TRIGGER07

Trigger	TRIGGER07
Description	After a new deposit is made, increase that client's balance.

SQL CODE

```
DROP FUNCTION IF EXISTS balance_update() CASCADE;

CREATE FUNCTION balance_update() RETURNS TRIGGER AS
$BODY$
BEGIN
    UPDATE Client SET balance = (Select balance from Client where idClient = New.idClient) + New.amount WHERE Client.idClient = New.id
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER balance_update
AFTER INSERT ON Deposit
FOR EACH ROW
EXECUTE PROCEDURE balance_update();
```

TRIGGER08

Trigger	TRIGGER08
Description	Checks if an user has any active bids or auctions before deleting his account, not allowing the deletion if he does.

SQL CODE

```
DROP FUNCTION IF EXISTS check_del() CASCADE;

CREATE FUNCTION check_del() RETURNS TRIGGER AS
$BODY$
BEGIN
    IF EXISTS
        (select * from auction where auction.idOwner = OLD.idClient AND auction.endDate > NOW())
    THEN
        RAISE EXCEPTION 'Cannot delete user, he currently has active auctions';
    END IF;
    IF EXISTS
        (select from Bid where Bid.idClient = OLD.idClient AND Bid.Price = (Select currentprice from Auction where auction.idAuction =
    THEN
        RAISE EXCEPTION 'Cannot delete user, he currently has active bids';
    END IF;

RETURN OLD;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER check_del
BEFORE DELETE ON Client
FOR EACH ROW
EXECUTE PROCEDURE check_del();
```

TRIGGER09

Trigger	TRIGGER09
Description	Check if an Owner already exists before creating auction

SQL CODE

```
DROP FUNCTION IF EXISTS check_own() CASCADE;

CREATE FUNCTION check_own() RETURNS TRIGGER AS
$BODY$
BEGIN
    IF NOT EXISTS
        (select * from AuctionOwner where AuctionOwner.idClient = NEW.idOwner)
    THEN
        INSERT INTO AuctionOwner(idClient) values (New.idOwner);
    END IF;
RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER check_own
BEFORE INSERT ON Auction
FOR EACH ROW
EXECUTE PROCEDURE check_own();
```

4. Transactions

TRANSACTION01

Transaction	TRANS01
Description	Gets all the bids of an auction
Justification	In the middle of an auction, there can be new bids while you are trying to put a bid yourself. This guarantees that the current bid doesn't change when you bid.

Transaction	TRANS01
Isolation level	SERIALIZABLE READ ONLY

SQL CODE

```
BEGIN;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY;

select Bid.idBid, Bid.bidDate, Bid.isValid, Bid.price, Bid.idClient, Bid.idAuction
from Auction,Bid where bid.idAuction = Auction.idAuction order by Bid.price asc;

COMMIT;
```

TRANSACTION02

Transaction	TRANS02
Description	Insert a new auction
Justification	In the middle of the transaction, the insertion of new rows in the auction table can occur, which implies that the information retrieved in both selects is different, consequently resulting in a Phantom Read. It's READ ONLY because it only uses Selects.
Isolation level	SERIALIZABLE READ ONLY

SQL CODE

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

-- Get number of current auctions
SELECT COUNT(*)
FROM Auction
WHERE now() < endDate;

-- Get ending auctions (limit 8)
SELECT Auction.name, Auction.startDate, Auction.endDate, Auction.startingPrice, Auction.currentPrice, Auction.description, Category.nam
FROM Auction
INNER JOIN Category ON Category.idCategory = Auction.idCategory
INNER JOIN AuctionOwner ON AuctionOwner.idClient = Auction.idOwner
INNER JOIN Client ON Client.idClient = AuctionOwner.idClient
WHERE now () < Auction.endDate
ORDER BY Auction.endDate ASC
LIMIT 8;

COMMIT;
```

Annex A. SQL Code

All the SQL code can be found at src/db

A.1. Database schema

```
DROP TABLE IF EXISTS DeletedUser;
DROP TABLE IF EXISTS AuctionLog;
DROP TABLE IF EXISTS UserLog;
DROP TABLE IF EXISTS Notification;
DROP TABLE IF EXISTS Deposit;
DROP TABLE IF EXISTS SystemManagerLog;
DROP TABLE IF EXISTS SystemManager;
DROP TABLE IF EXISTS AuctionReport;
```

```
DROP TABLE IF EXISTS SellerReport;
DROP TABLE IF EXISTS FavoriteAuction;
DROP TABLE IF EXISTS Bid;
DROP TABLE IF EXISTS Review;
DROP TABLE IF EXISTS Auction;
DROP TABLE IF EXISTS AuctionOwner;
DROP TABLE IF EXISTS Category;
DROP TABLE IF EXISTS Client;

CREATE TABLE IF NOT EXISTS Client(
    idClient    SERIAL PRIMARY KEY,
    username    VARCHAR(30) NOT NULL UNIQUE,
    password    VARCHAR(256) NOT NULL,
    email       VARCHAR(50) UNIQUE NOT NULL,
    firstName   VARCHAR(30) NOT NULL,
    lastName    VARCHAR(30) NOT NULL,
    address     VARCHAR(70),
    phoneNumber VARCHAR(13) UNIQUE,
    isBanned    BOOLEAN NOT NULL,
    balance     FLOAT NOT NULL
);

CREATE TABLE IF NOT EXISTS Category(
    idCategory  SERIAL PRIMARY KEY,
    name        VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS AuctionOwner(
    idClient SERIAL PRIMARY KEY,
    rating   FLOAT,
    FOREIGN KEY (idClient) REFERENCES Client ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT validRating CHECK((rating BETWEEN 0 AND 10) OR (rating IS NULL))
);

CREATE TABLE IF NOT EXISTS Auction(
    idAuction      SERIAL PRIMARY KEY,
    name           VARCHAR(50) NOT NULL,
    startDate      TIMESTAMP NOT NULL,
    endDate        TIMESTAMP NOT NULL,
    startingPrice  FLOAT NOT NULL,
    currentPrice   FLOAT NOT NULL,
    description    VARCHAR(1000) NOT NULL,
    isOver         BOOLEAN NOT NULL,
    idCategory     INTEGER NOT NULL,
    idOwner        INTEGER NOT NULL,
    CONSTRAINT validEndDate CHECK (endDate > startDate),
    CONSTRAINT validStartingPrice CHECK (startingPrice > 0),
    FOREIGN KEY (idCategory) REFERENCES Category ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idOwner) REFERENCES AuctionOwner ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS Review(
    idReview       SERIAL PRIMARY KEY,
    rating         INTEGER NOT NULL,
    comment        VARCHAR(300),
    reviewDate     TIMESTAMP NOT NULL,
    idUserReviewer INTEGER NOT NULL,
    idUserReviewed INTEGER NOT NULL,
    CONSTRAINT validRating CHECK(rating BETWEEN 0 AND 10),
    FOREIGN KEY (idUserReviewer) REFERENCES Client(idClient) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idUserReviewed) REFERENCES AuctionOwner(idClient) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS Bid(
    idBid       SERIAL PRIMARY KEY,
    bidDate     TIMESTAMP NOT NULL,
    isValid     BOOLEAN NOT NULL,
```

```
    price       FLOAT NOT NULL,
    idClient    INTEGER NOT NULL,
    idAuction   INTEGER NOT NULL,
    FOREIGN KEY (idClient) REFERENCES Client ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idAuction) REFERENCES Auction ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS FavoriteAuction(
    idClient    INTEGER NOT NULL,
    idAuction   INTEGER NOT NULL,
    PRIMARY KEY(idClient, idAuction),
    FOREIGN KEY (idClient) REFERENCES Client ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idAuction) REFERENCES Auction ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS SellerReport(
    idReport       SERIAL PRIMARY KEY,
    reportDate     TIMESTAMP NOT NULL,
    description    VARCHAR(500) NOT NULL,
    isSolved       BOOLEAN NOT NULL,
    idSeller       INTEGER NOT NULL,
    idReporter     INTEGER NOT NULL,
    FOREIGN KEY (idSeller) REFERENCES AuctionOwner(idClient) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idReporter) REFERENCES Client(idClient) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS AuctionReport(
    idReport       SERIAL PRIMARY KEY,
    reportDate     TIMESTAMP NOT NULL,
    description    VARCHAR(500) NOT NULL,
    isSolved       BOOLEAN NOT NULL,
    idAuction      INTEGER NOT NULL,
    idReporter     INTEGER NOT NULL,
    FOREIGN KEY (idAuction) REFERENCES Auction ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idReporter) REFERENCES Client(idClient) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS SystemManager(
    idSysMan    SERIAL PRIMARY KEY,
    username    VARCHAR(30) NOT NULL UNIQUE,
    email       VARCHAR(30) NOT NULL UNIQUE,
    password    VARCHAR(30) NOT NULL
);

CREATE TABLE IF NOT EXISTS SystemManagerLog(
    idSysLog        SERIAL PRIMARY KEY,
    logDate         TIMESTAMP NOT NULL,
    logDescription  VARCHAR(500) NOT NULL,
    logType         VARCHAR(50) NOT NULL,
    idSysMan        INTEGER NOT NULL,
    FOREIGN KEY (idSysMan) REFERENCES SystemManager ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT TypeCheck CHECK (logType = 'Ban' OR logType = 'Unban' or logType = 'Delete' or logType = 'other')
);

CREATE TABLE IF NOT EXISTS Deposit(
    idDeposit   SERIAL PRIMARY KEY,
    amount      FLOAT NOT NULL,
    method      VARCHAR(30) NOT NULL,
    depositDate TIMESTAMP NOT NULL,
    idClient    INTEGER NOT NULL,
    FOREIGN KEY (idClient) REFERENCES Client ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT methodCheck CHECK (method = 'PAYPAL' OR method = 'MBWAY' or method = 'BANK TRANSFER' or method = 'CRYPTO' or method = '
);

CREATE TABLE IF NOT EXISTS Notification(
    idNotification  SERIAL PRIMARY KEY,
    content         VARCHAR(50) NOT NULL,
```

```
    isRead          BOOLEAN NOT NULL,
    notifDate       TIMESTAMP NOT NULL,
    idClient        INTEGER NOT NULL,
    FOREIGN KEY (idClient) REFERENCES Client ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS UserLog(
    idSysLog   INTEGER NOT NULL,
    idClient   INTEGER NOT NULL,
    PRIMARY KEY(idClient, idSysLog),
    FOREIGN KEY (idClient) REFERENCES Client ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idSysLog) REFERENCES SystemManagerLog ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS AuctionLog(
    idSysLog    INTEGER NOT NULL,
    idAuction   INTEGER NOT NULL,
    PRIMARY KEY(idAuction, idSysLog),
    FOREIGN KEY (idAuction) REFERENCES Auction ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (idSysLog) REFERENCES SystemManagerLog ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS DeletedUser(
    idClient INTEGER PRIMARY KEY NOT NULL, -- Has to be the same as a prior existing user
    username VARCHAR(30) UNIQUE NOT NULL
);
```

A.2. Database population

```
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (1, 'cmi
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (2, 'tma
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (3, 'eth
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (4, 'ipo
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (5, 'mgo
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (6, 'rge
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (7, 'epe
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (8, 'abi
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (9, 'mbr
insert into Client (idClient, username, email, password, firstName, lastName, address, phoneNumber, isBanned, balance) values (10, 'nc
insert into Category (idCategory, name) values (1, 'Desporto');
insert into Category (idCategory, name) values (2, 'Lazer');
insert into Category (idCategory, name) values (3, 'Veículos');
insert into Category (idCategory, name) values (4, 'Arte');
insert into Category (idCategory, name) values (5, 'Imobiliário');
insert into Category (idCategory, name) values (6, 'Moda');
insert into Category (idCategory, name) values (7, 'Tecnologia');
insert into Category (idCategory, name) values (8, 'Casa e Jardim');
insert into Category (idCategory, name) values (9, 'Animais');
insert into AuctionOwner (idClient, rating) values (1, 0);
insert into AuctionOwner (idClient, rating) values (2, 0);
insert into AuctionOwner (idClient, rating) values (3, 0);
insert into AuctionOwner (idClient, rating) values (4, 0);
insert into AuctionOwner (idClient, rating) values (5, 0);
insert into AuctionOwner (idClient, rating) values (6, 0);
insert into AuctionOwner (idClient, rating) values (7, 0);
insert into AuctionOwner (idClient, rating) values (8, 0);
insert into AuctionOwner (idClient, rating) values (9, 0);
insert into AuctionOwner (idClient, rating) values (10, 0);
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
```

```
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Auction (idAuction, name , startDate, endDate, startingPrice, currentPrice, description, isOver, idCategory, idOwner) valu
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (1, '2022-10-21 00:25:00', 'true', 95, 1, 10);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (2, '2022-10-22 10:00:30', 'true', 1400, 2, 9);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (3, '2022-10-20 21:00:00', 'true', 21000, 3, 8);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (4, '2022-10-22 14:30:00', 'true', 1151, 4, 7);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (5, '2022-10-19 10:00:00', 'true', 100, 5, 6);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (6, '2022-10-18 12:10:00', 'true', 20000, 6, 5);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (7, '2022-10-15 12:00:00', 'true', 220, 7, 4);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (8, '2022-10-20 22:00:30', 'true', 200, 8, 3);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (9, '2022-10-20 21:10:25', 'true', 4312, 9, 2);
insert into Bid (idBid, bidDate, isValid, price, idClient, idAuction) values (10, '2022-10-19 13:12:10', 'true', 150, 10, 1);
insert into FavoriteAuction (idClient, idAuction) values (1, 1);
insert into FavoriteAuction (idClient, idAuction) values (2, 2);
insert into FavoriteAuction (idClient, idAuction) values (3, 3);
insert into FavoriteAuction (idClient, idAuction) values (4, 4);
insert into FavoriteAuction (idClient, idAuction) values (5, 5);
insert into FavoriteAuction (idClient, idAuction) values (6, 6);
insert into FavoriteAuction (idClient, idAuction) values (7, 7);
insert into FavoriteAuction (idClient, idAuction) values (8, 8);
insert into FavoriteAuction (idClient, idAuction) values (9, 9);
insert into FavoriteAuction (idClient, idAuction) values (10, 10);
insert into SystemManager (idSysMan, username , email, password) values (1, 'ljedrych0', 'ljedrych0@blogtalkradio.com', '4HmZdUZG6kV')
insert into SystemManager (idSysMan, username , email, password) values (2, 'kcollihole1', 'kcollihole1@so-net.ne.jp', 'n5gLdwK');
```

Revision history

Changes made to the first submission:

GROUP2225, 23/10/2022

- Diogo Babo, up202004950@edu.fe.up.pt
- João Oliveira, up202004407@edu.fe.up.pt (editor)
- Gustavo Costa, up202004187@edu.fe.up.pt (editor)
- Ricardo Cavalheiro, up202005103@edu.fe.up.pt