POLITECNICO DI MILANO

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Progetto di Ingegneria del Software 2

Prof.ssa Raffaela Mirandola

A.A. 2014\15

Giuseppe Vergori

MATRICOLA: 824298

C.P. 10294634 824298

giuseppe.vergori@mail.polimi.it

Titolo Progetto: GUESSBID

Part II: DD

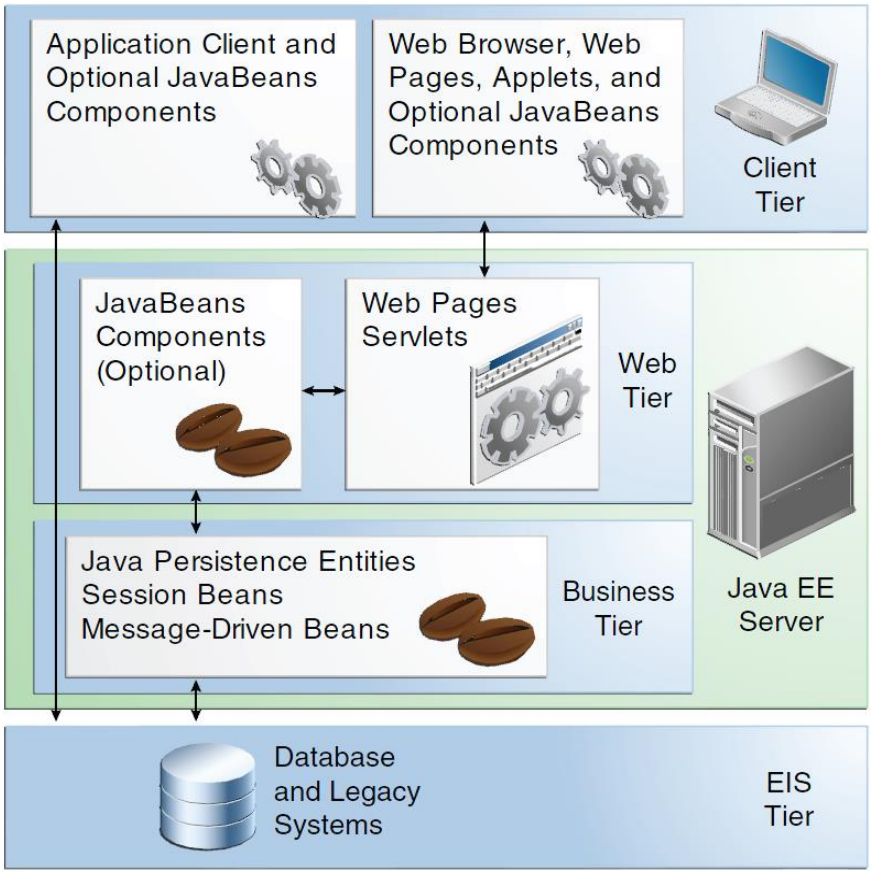
DESIGN DOCUMENT

SUMMARY

1.Architecture Description

1.1 JEE Architecture Overview

Before starting to explain the application's architecture let's focus on the JEE Architecture.



JEE has a four tiered architecture divided as:

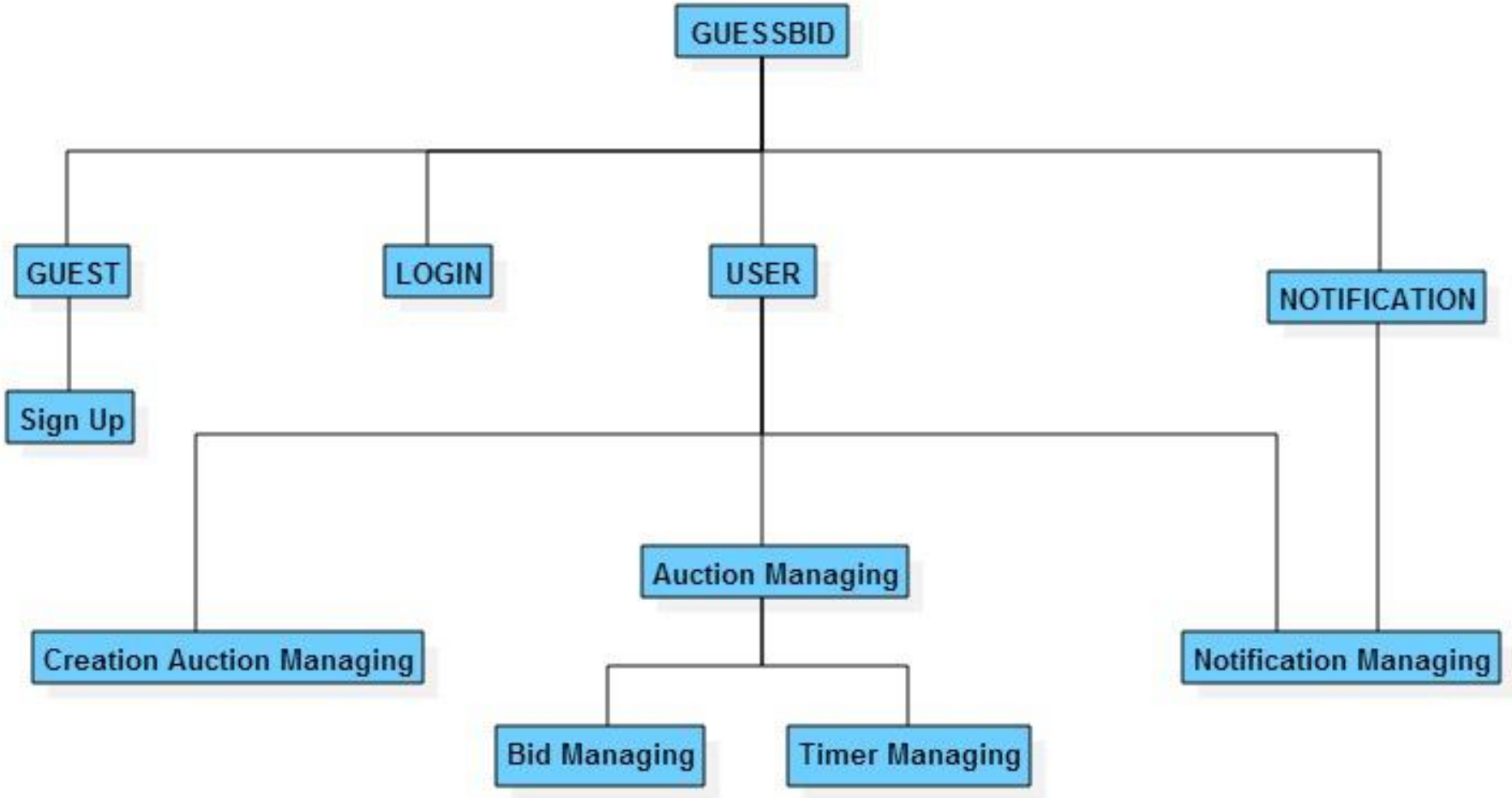
- **Client Tier**: it contains Application Clients and Web Browsers and it is the layer that interacts directly with the actors. The project will be a web application and the client will use a web browser to access pages;
- **Web Tier**: it contains the *Servlets* and Dynamic Web Pages that needs to be elaborated. This tier receives the requests from the client tier and forwards the pieces of data collected to the business tier waiting for processed data to be sent to the client tier, eventually formatted;

- **Business Tier**: it contains the Java Beans, that contain the business logic of the application, and Java Persistence Entities.
- **EIS Tier**: it is constituted by DBMS (in this case MySQL) and has the task of maintaining the data in a reliable way, preserving the ACID properties.

1.2. Identifying Sub-Systems

In order to easily understand the behaviour of the system, the application can be separated into these sub-systems:

- *Guest sub-system*, which includes
    - *sign up sub-system*, that manages the registration procedure for the new users;
- *Login sub-system*, that controls the validity of the data entered by users;
- *User sub-system*, also split in:
    - *Creation Auction Managing sub-system*, which allows users to create an auction;
    - *auction managing sub-system*, which allows users to manage their auctions and include:
        - *bid managing sub-system*, which allows users to bid for a good;
        - *timer managing sub-system*, which handles the auction time;
- *Notification sub-system*, which talk with the *bid managing sub-system* and the *timer managing sub-system* to retrieve information about the status of the auction, and includes:
    - *notification managing sub-system*, shared with the *user sub-system*. It controls that the notification massages are sent to the right users.
- *Data sub-system* (transparent), which manages user's data and ensures their correct storage.

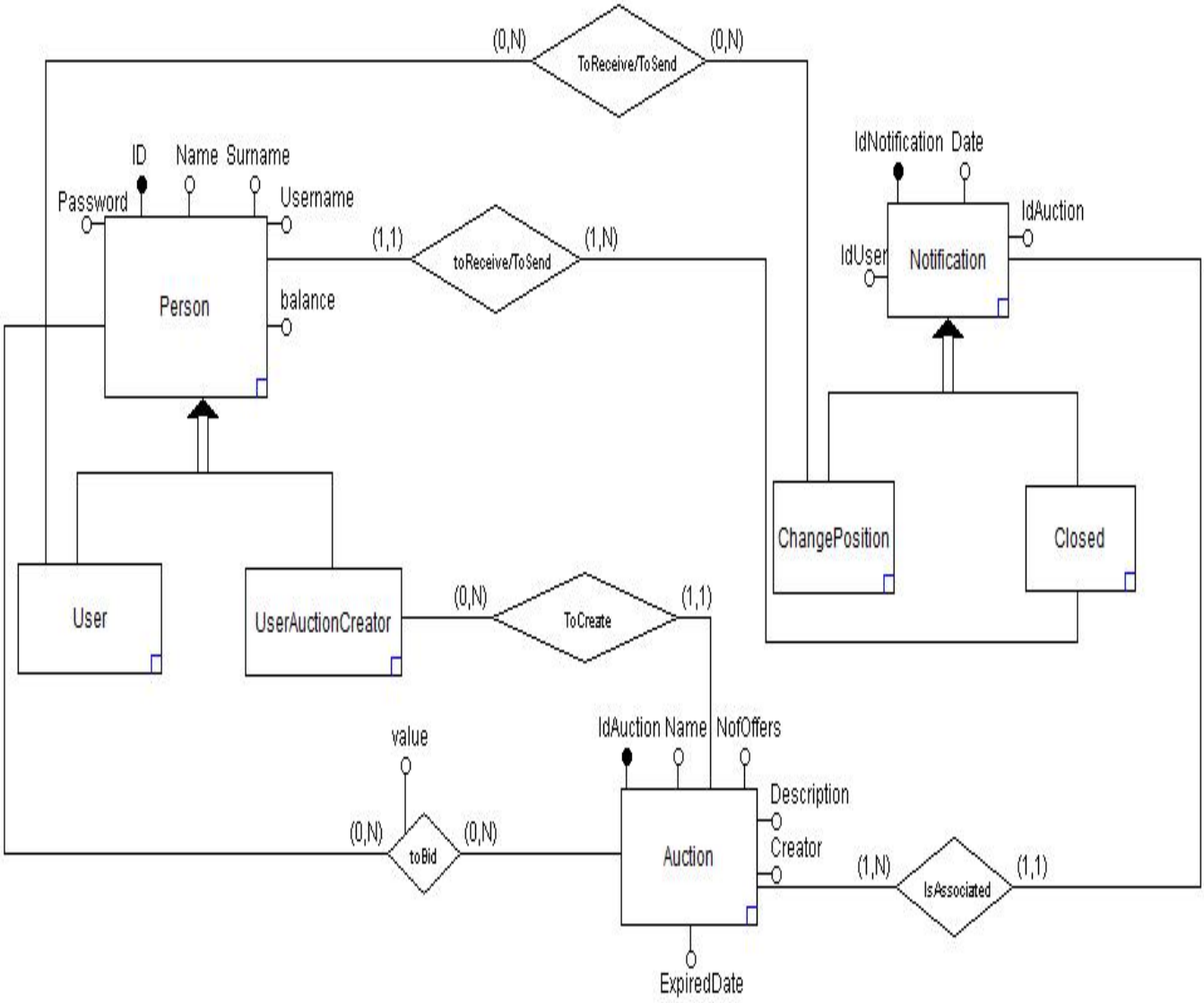

GUESSBID
GUEST
LOGIN
USER
NOTIFICATION
Sign Up
Auction Managing
Creation Auction Managing
Notification Managing
Bid Managing
Timer Managing

2. Persistent Data Management

The data is stored into a relation database. To better understand this process, let's see in this chapter entities, relations and in general the motivations for this design of the database

2.1 Conceptual Design

Conceptual design allows to start thinking about the data that will be stored into the database and about the relations between them. First, let's see a generic view of the database in terms of Entity-Relationship Diagram. It is thought respecting the GuessBid specifications with this interpretation, in order to understand clearly the behaviour of the system.



To better understand the E-R diagram above, let's examine in detailed the entities and the relation that compose it.

Let's start with the entities.

- *Person*: represents who uses the application and with its attributes identifies registered users. This entity is also divided to better understand GuessBid specification and the operations that can do:
    - *User*: all registered users;
    - *User Auction Creator*: a registered user that creates an auction;
- *Auction*: the attributes of this entity represents the data creates by registered users;
- *Notification*: represents, in general, the data that are generated after an auction creation. This entity is also divided to better understand GuessBid specification and the functions that can perform, into:
    - *Change Position*: in relation with the bids of an auction, is send to the users when their current position in the auction change.
    - *Closed*: in relation with the expired date, contains the information about the result of the auction.

Finally, the relations are:

To receive/To send: there are two type of this relation, but it represents the same relation. It has been split for better understand the GuessBid specifications. So, it means:

- between the entities Person and Closed Notification:
    - "A Person", intended as registered user like auction creator or auction participant, "can receive only 1 closed auction notification";
    - "A closed auction notification is sent to 1 or N users", user as written above.
- between the entities User and Change Position Notification:
    - "An user", intended as user who has bid in the auction, "can receive 0 o N change position notifications";
    - "A change position notification is sent to 0 or N users", user as written above.

To create: this relation, between the entities User Auction Creator and Auction, means:

- "An User Auction Creator can create 0 or N auctions";
- "An Auction can be created by 1 and only 1 User Auction Creator".

Is Associated: this relation, between the entities Auction and Notification, means:

- "An Auction can be associated 1 or N notifications";

- "A Notification is associated at 1 and only 1 Auction".

To bid: this relation, between the entities Person and Auction, means:

- "A Person", intended as general registered user, " can bid in 0 or N auctions";
- "An auction contains 0 or N bids, made by the persons", person as written above.

2.2 Logical Design

Logical Design has the aim to better represent the database structure of the system, but, in order to build this model from the ER diagram drawn above, must be made some transformations.

First of all, reducing several entity we obtain these complete entities:

- *User*: is the merge of the entity Person with its son. So, the entity with its attributes can identify any registered user.
- *Notification*: the merge of the entity Notification with its son. Now, thanks to the attribute "type" can identify and to distinguish the two types of notifications when they occur.

Now, let's examine the relations discussed in the previous paragraph:

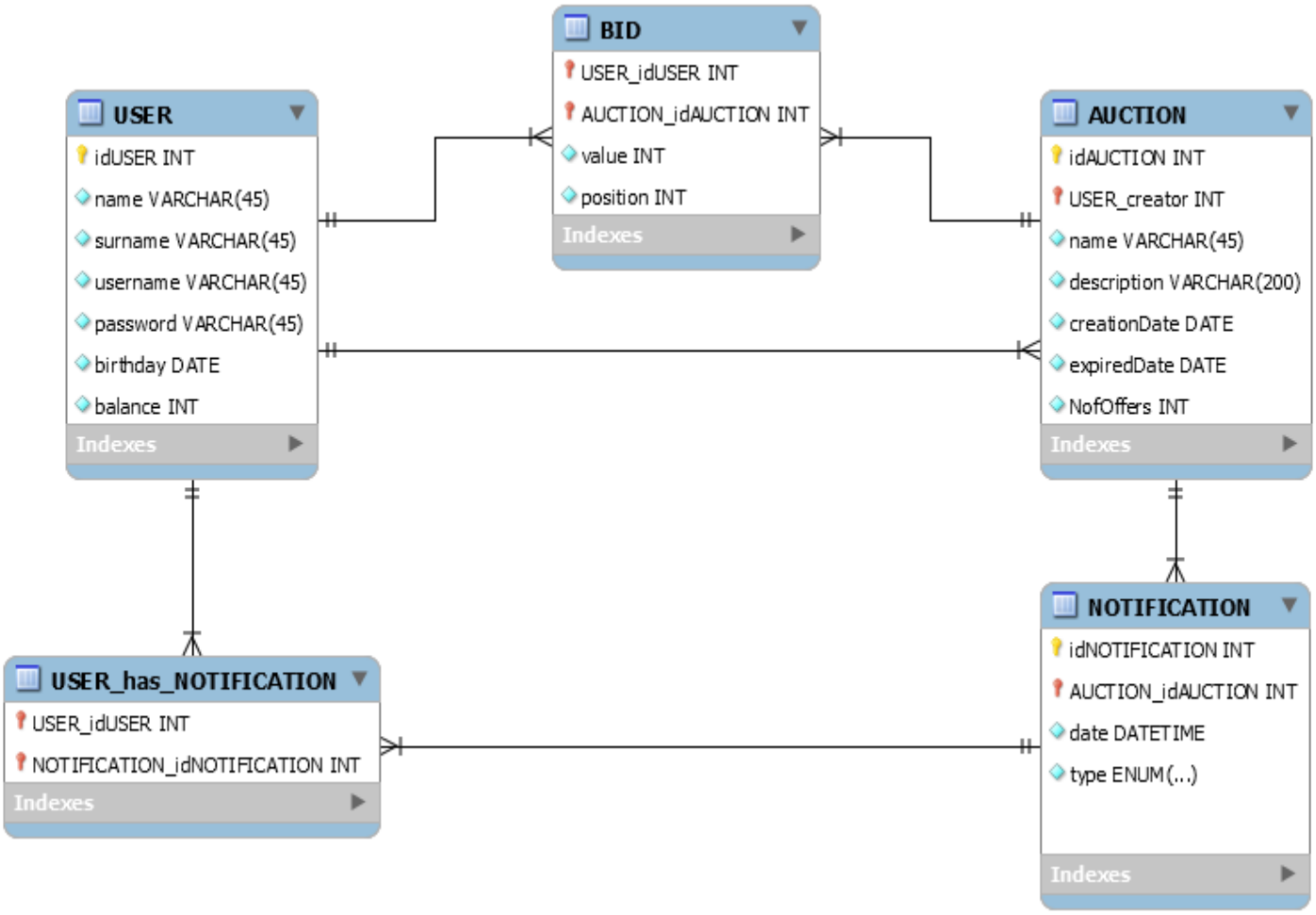
- relation "To create" between UserAuctionCreator and Auction is translated inserting the foreign key named "*User_Creator*" into the table Auction;

- relation "To bid", between Person and Auction and User, is translate inserting a new table "*Bid*" with the foreign keys that allow to identify the user and the auction. Also there is an attribute named "*Value*", that has information about user's bid, and "position", that keep track of the current position of user after the bid.

- the two relations "to Receive/to Send", between Person and Notification, are zero to many relations and are translated with a new table "*Notification_has_User*" that contain the foreign key "*idNotification*" and "*username*";

- relation "is Associate" between Auction and Notification is translated inserting the foreign key named "*idAuction*" into the table Notification.

Finally, this is the design of the logical model:



The final model has the following physical structure:

- **User** (Id User, Name, Surname, Username, Password, Birthday, Balance);
- **Auction** (Id Auction, User Creator, Name, Description, CreationDate, ExpiredDate, Number of Offers);
- **Bid** (Id Auction, Id User, Value, Position);
- **Notification** (Id Notification, Id Auction, Date, Type);
- **Notification_has_user** (Id Notification, Id User).

3. User Experience

Let's see in this paragraph the User Experience (U X) given by the system to its users.

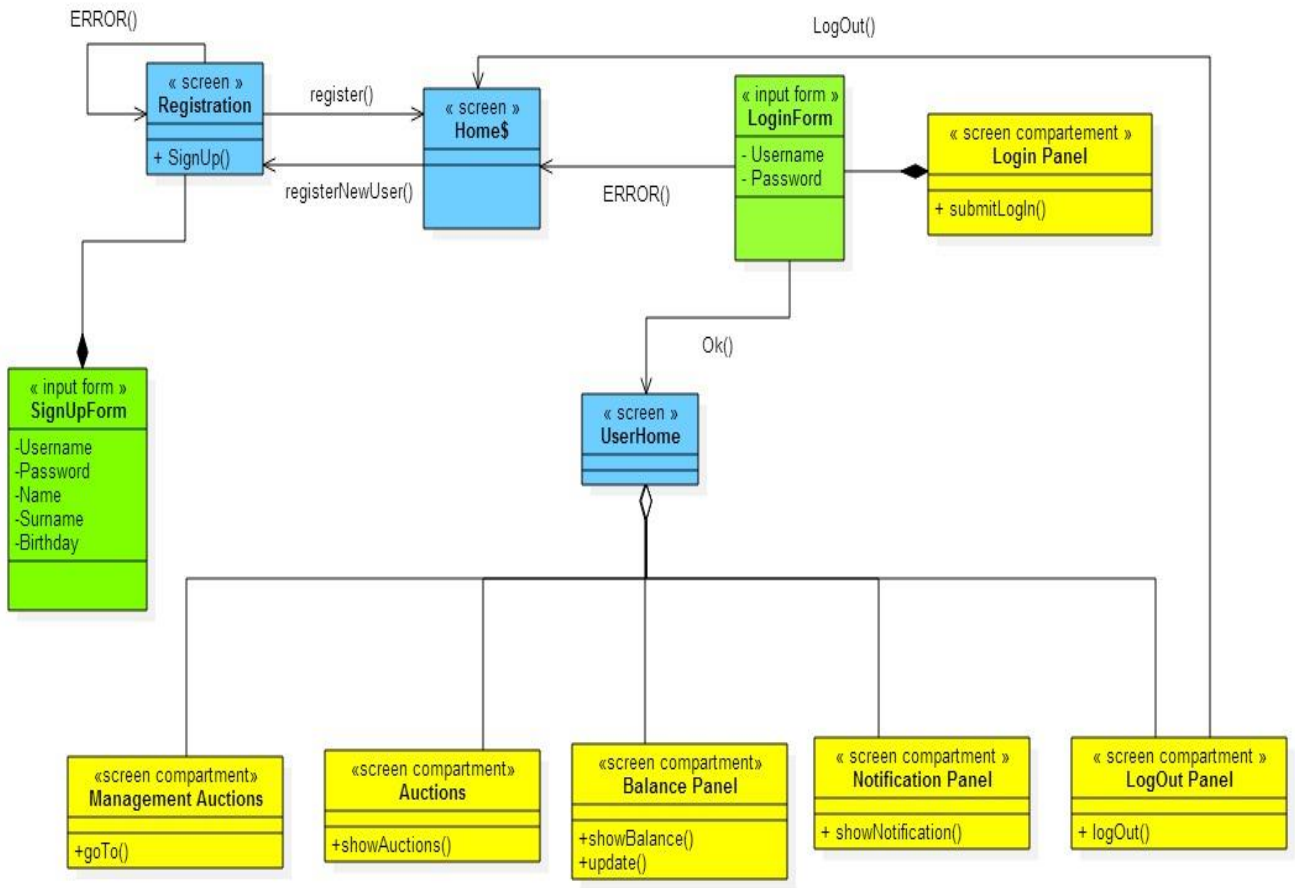
These U X contains class diagram with appropriate stereotypes, and are:

- <<*screen*>>, that represents pages;
- <<*screen compartment*>> that represents parts of the page, and the operations inside them are given just to explain the correct functioning of the system;
- <<*input form*>> that ,eventually, represents some input fields that can be fulfilled by a user (this information will be submitted to the system clicking on a button).

Also, the U X diagram refer to the Use Case of the RASD document in order to better understand the whole Diagram.

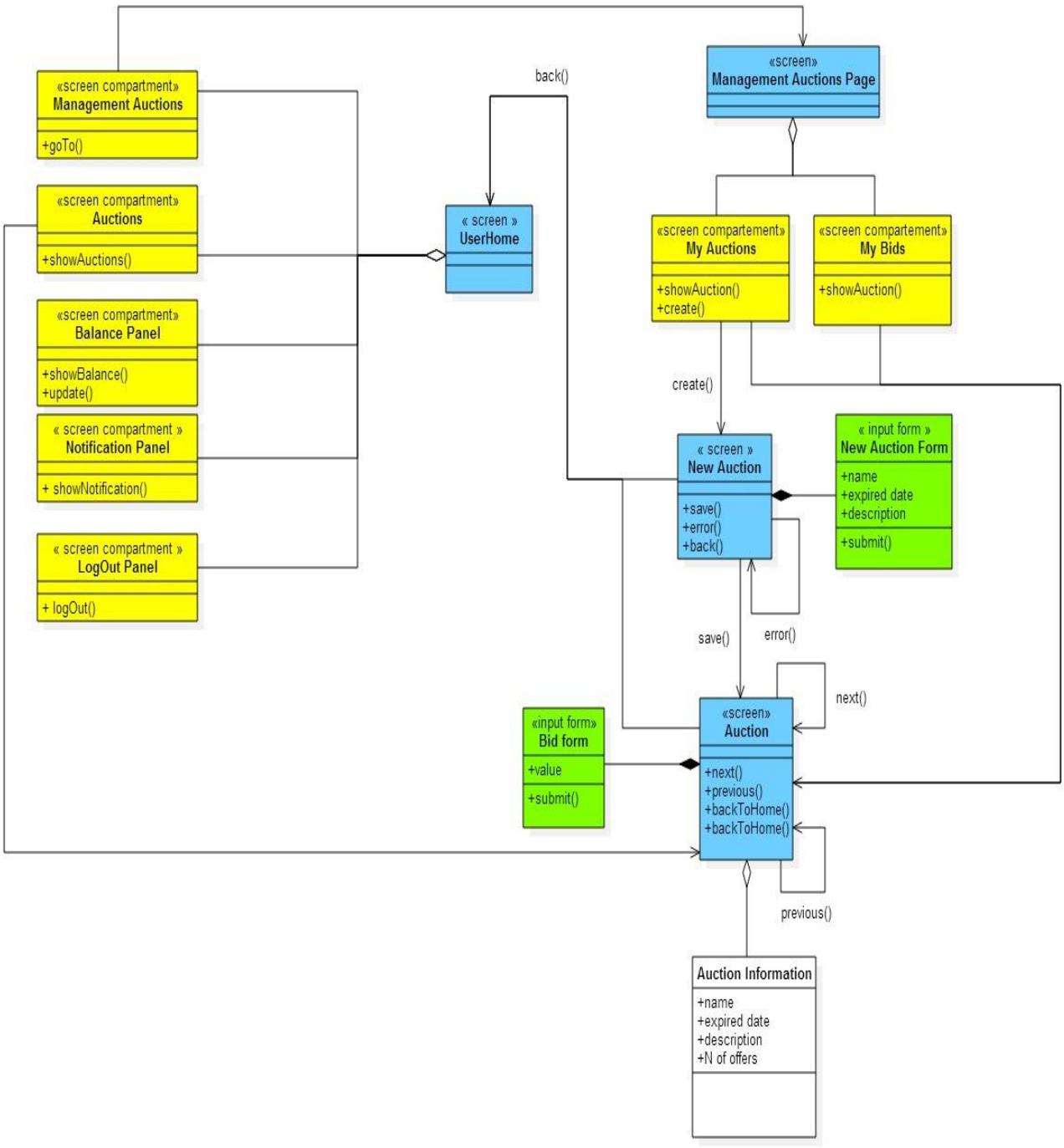
3.1 Homes

First, let's see the operations that can perform a guest user and what he see after registration and login.



3.2 Auction creation and *Bid*

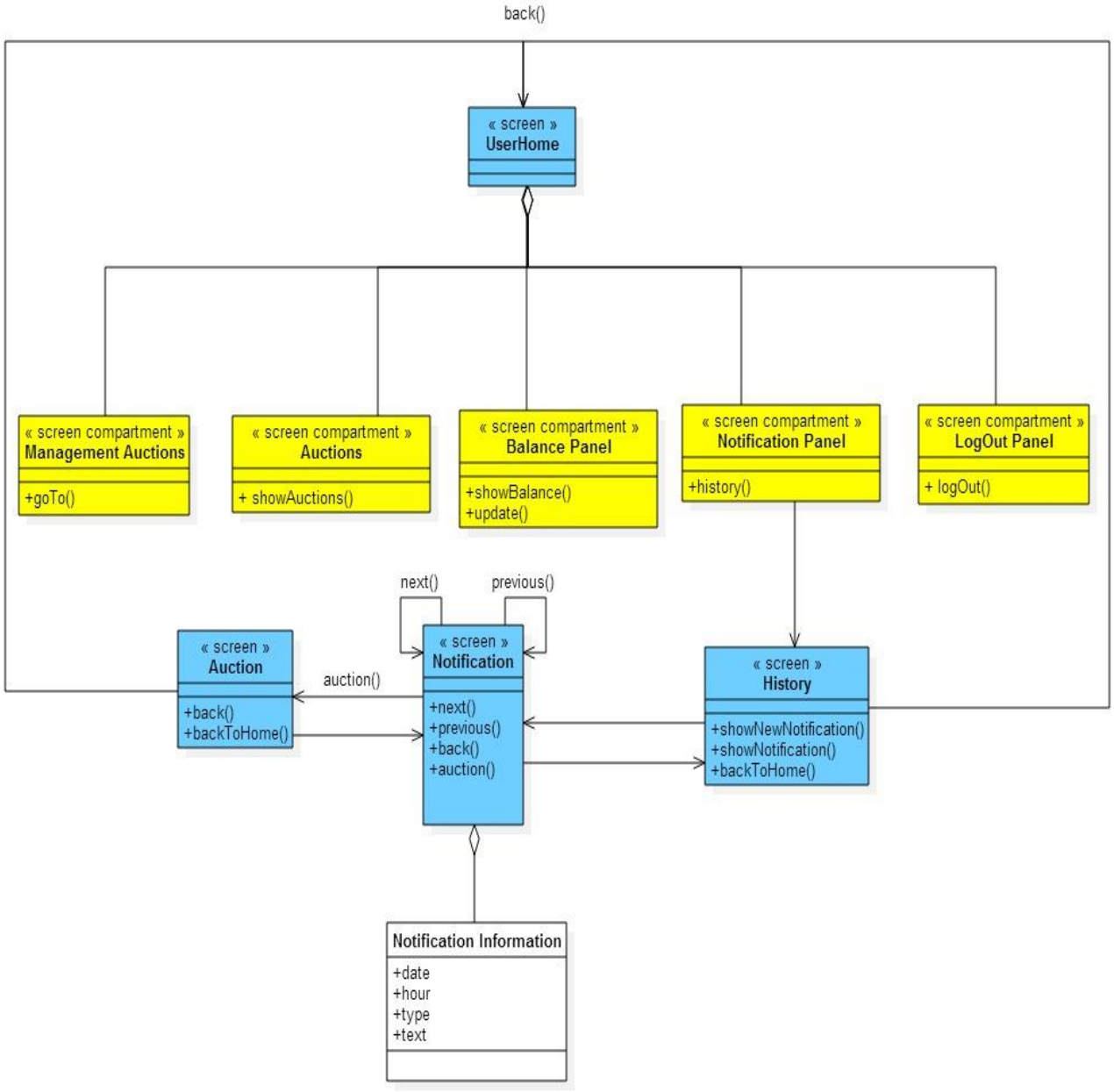
This part of the UX diagram allows to understand how the auction creation and to bid functionalities are offered to registered users. In particular, we can observe that a registered user can bid in the auction page.



3.3 Show Notifications

Finally, let's see the UX referred to registered users that can show the notifications.

A registered user who receives a notification alert clicks on the notification link and see a history of his notification received (optional), in chronological order. When he clicks on the notification, he sees the page of notification where there is a link to the referred auction. For instance, if the user receives a change position notification, when he click on the referred auction link he shows his new position in the ranking.

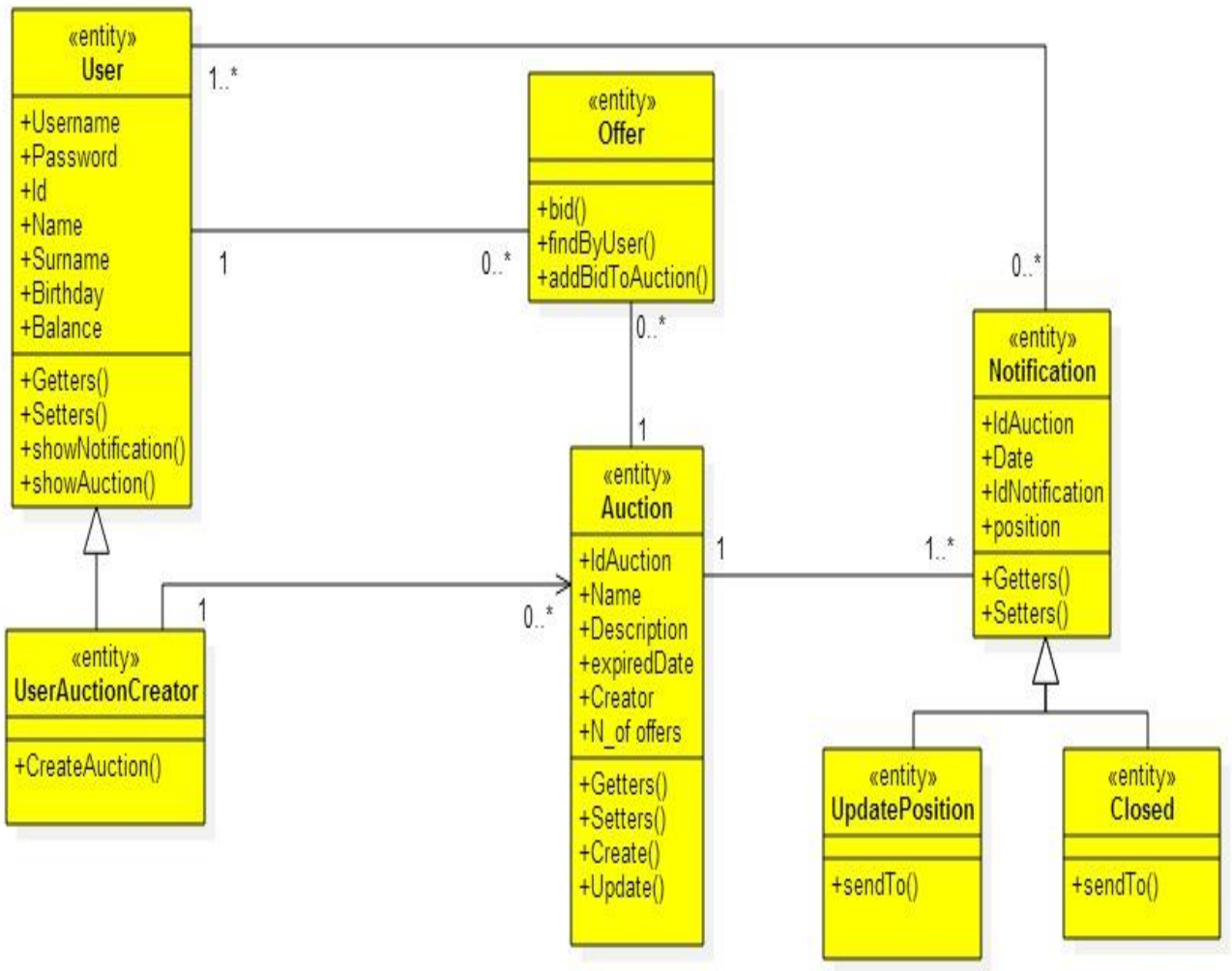


4. BCE Diagrams

Now, let's see a further design schema of GuessBid using the Boundary-Control-Entity pattern, because it is very close to the Model-View-Controller pattern (in general boundaries maps to the view, controls map to the controller and entities map to the model) and UML defines a standard to represent it.

4.1. Entity overview

First, let's see all entities of the BCE model. The entities Person and Notification are divided following the GuessBid specifications in order to better understand their functionalities.



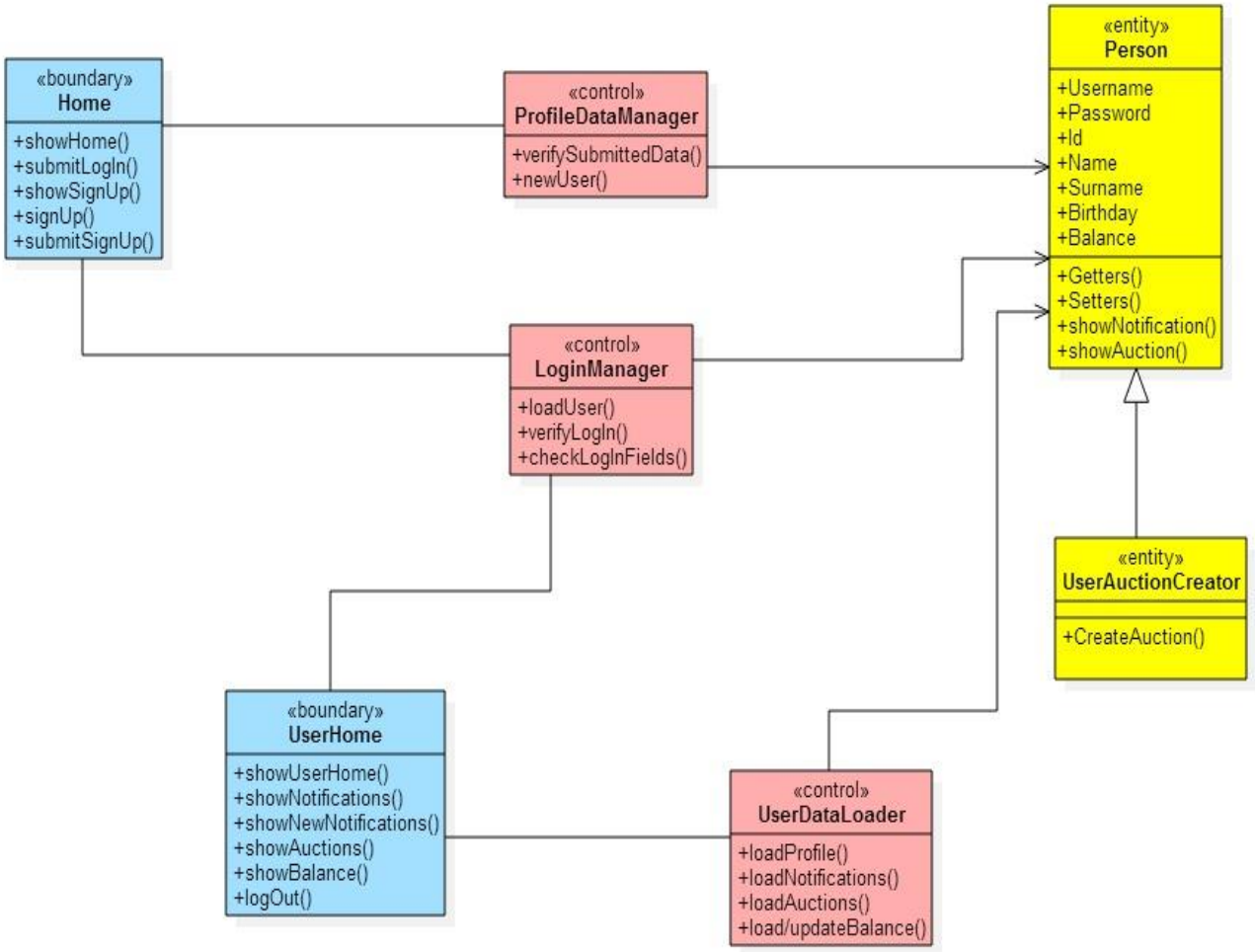
4.2. Sign up and Log in

The boundary *Home* represents the interface with guest users and the basic functionalities that they can reach at a first glance of the system, i.e. he can show the Home page and sign up to the application.

The boundary *UserHome* represents the interface with registered used and the functionalities that they can perform in according with the specifications on the system.

Also, there are three controllers:

- **Profile Data Manager:** this controller manages the verification of profile data in case of sign up. It also creates a new user in case of signing up correctly.
- **Log In Manager:** this controller manages the verification of log in fields.
- **User Data Loader:** this controller manages the data of the users.

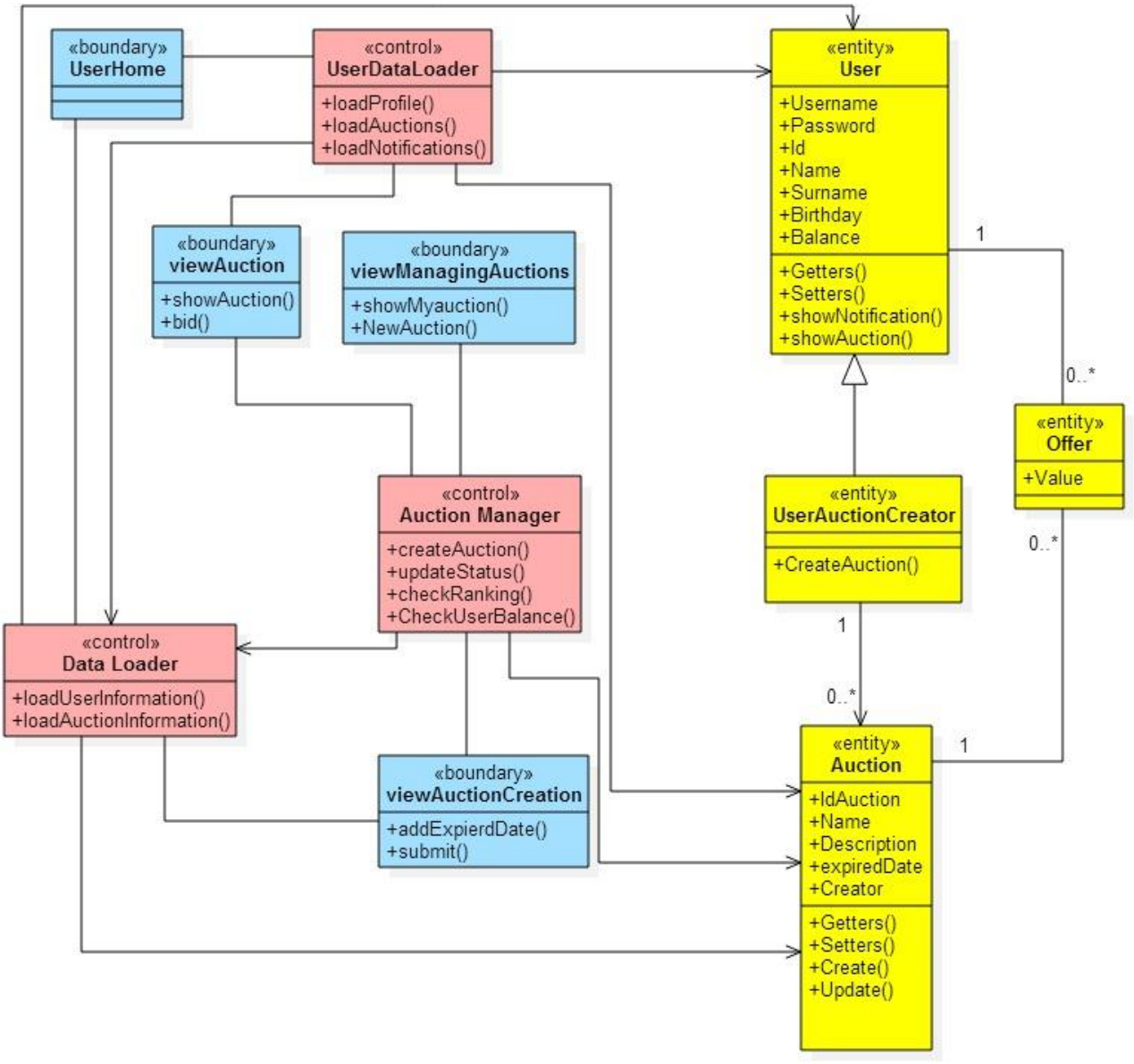


4.3 Auction Management

In this diagram there are the boundaries *viewAuction* , *viewManagingAuction* and *viewAuctionCreation* that represent the interfaces and functionalities that registered users can perform when show or create an auction.

There are also two new controllers:

- **Auction Manager**, which manages the part related to auctions;
- **Data Loader**, which loads information that are needed for example to compare all the bids, or to know the status of the auction.



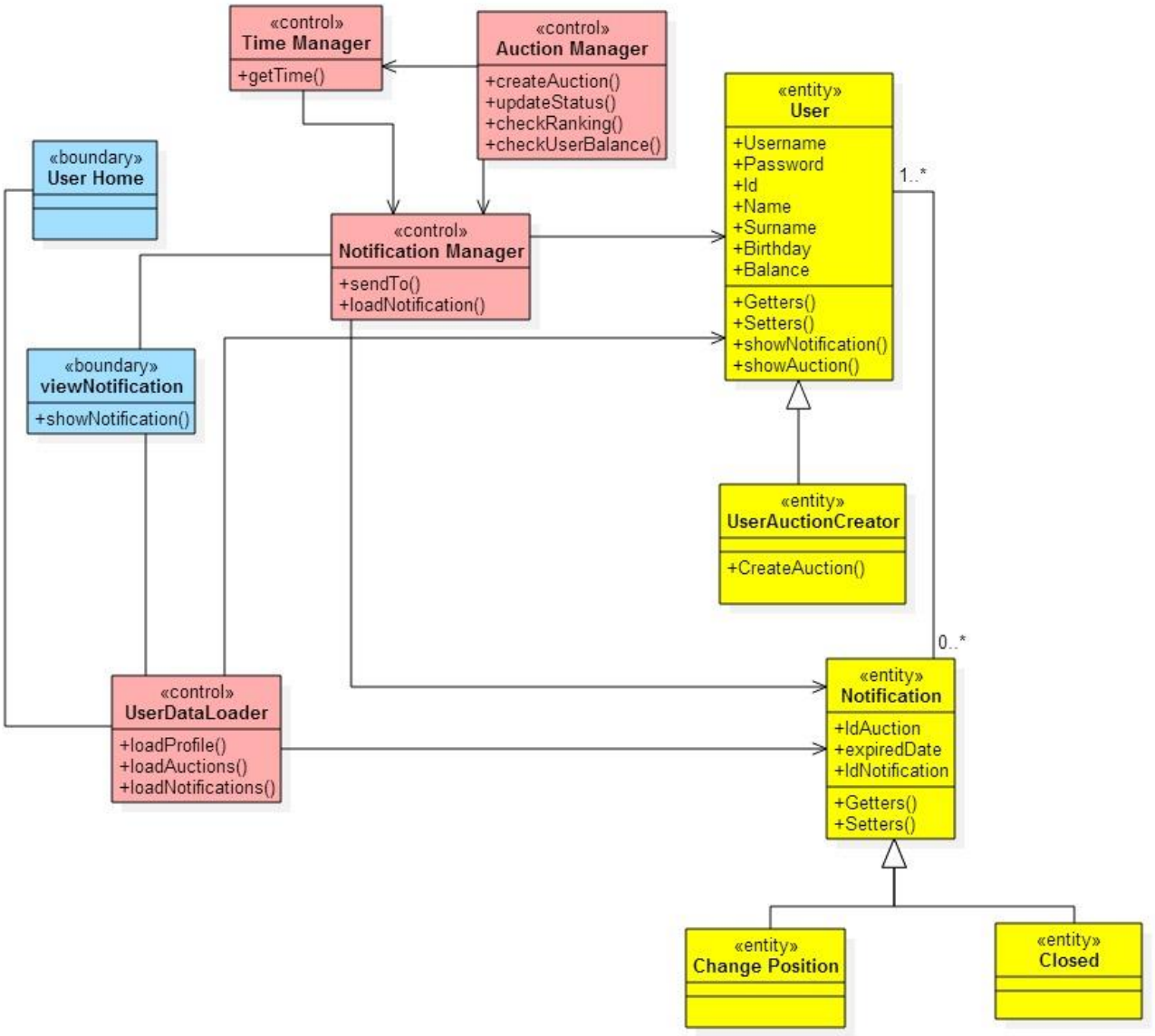
4.4 Notification Management

In this diagram the boundary *viewNotification* represents the interface and the functionalities that registered users can perform when show notifications.

There are also two new controllers:

- **Time Manager**, which checks the expired date of the auction.

- **Notification Manager**, that manages the notifications of the users and the information that it receives from the two controllers in order to send the notifications to the right users.



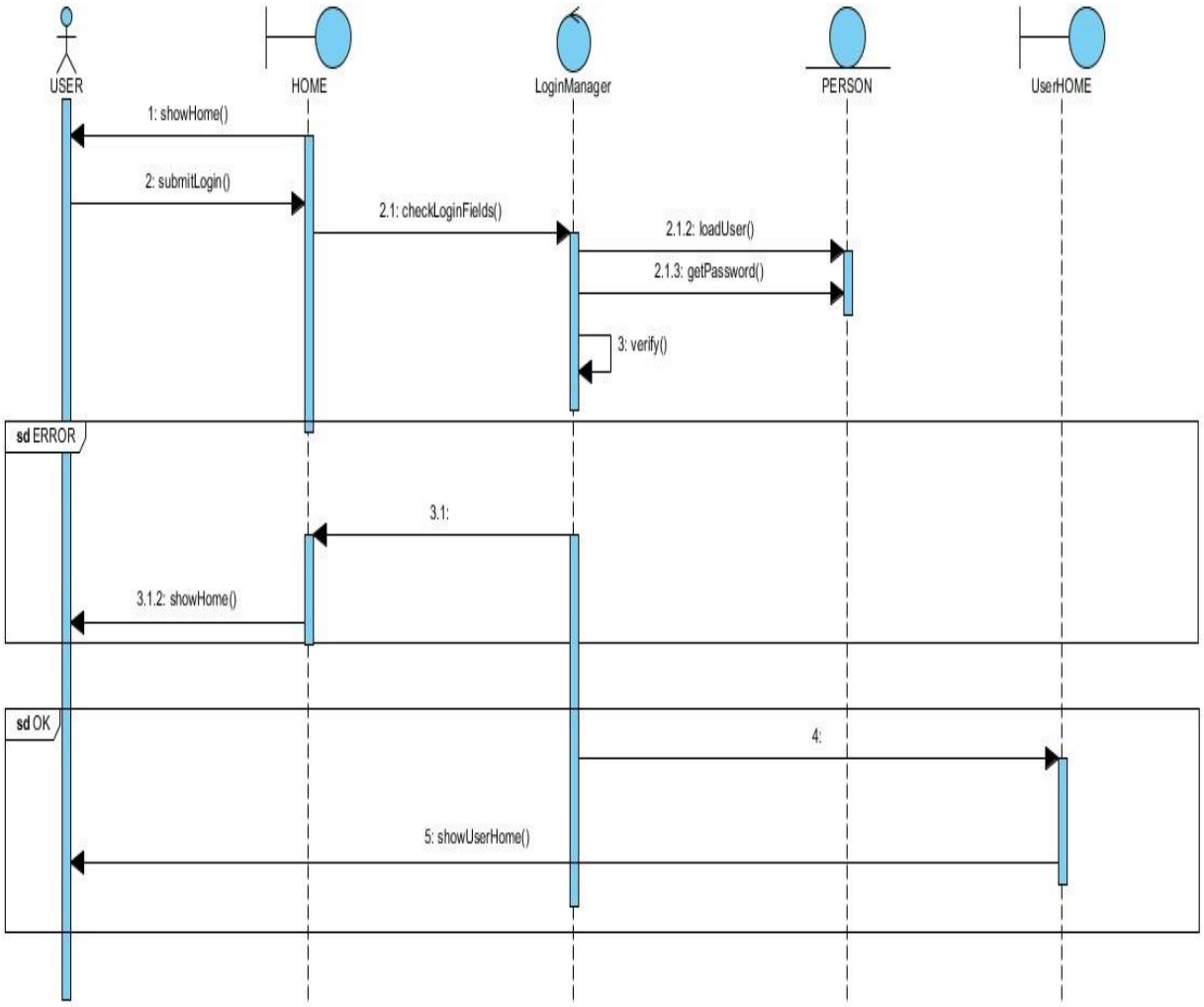
5. Sequence diagrams

Let's see in this paragraph some sequence diagrams for better understand the BCE diagrams shown previously.

All the methods used are the methods listed into the BCE in boundaries, controls and entities.

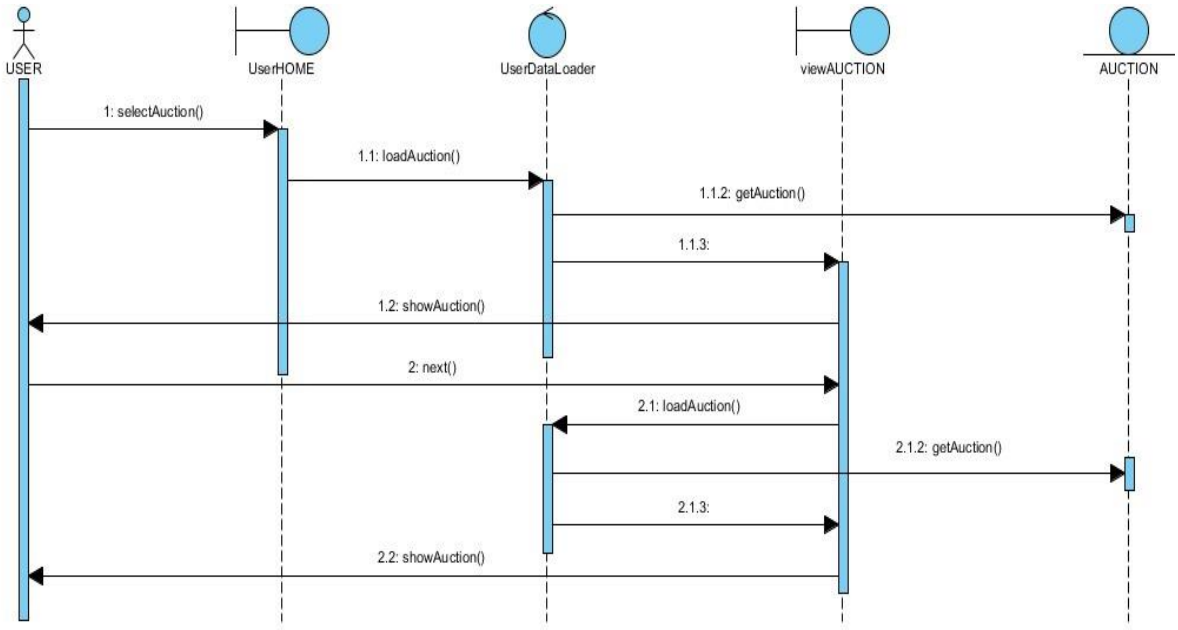
5.1. Log In

First, let's see how the system responds to the requests of users, with the relative exceptions, when these do the log in.



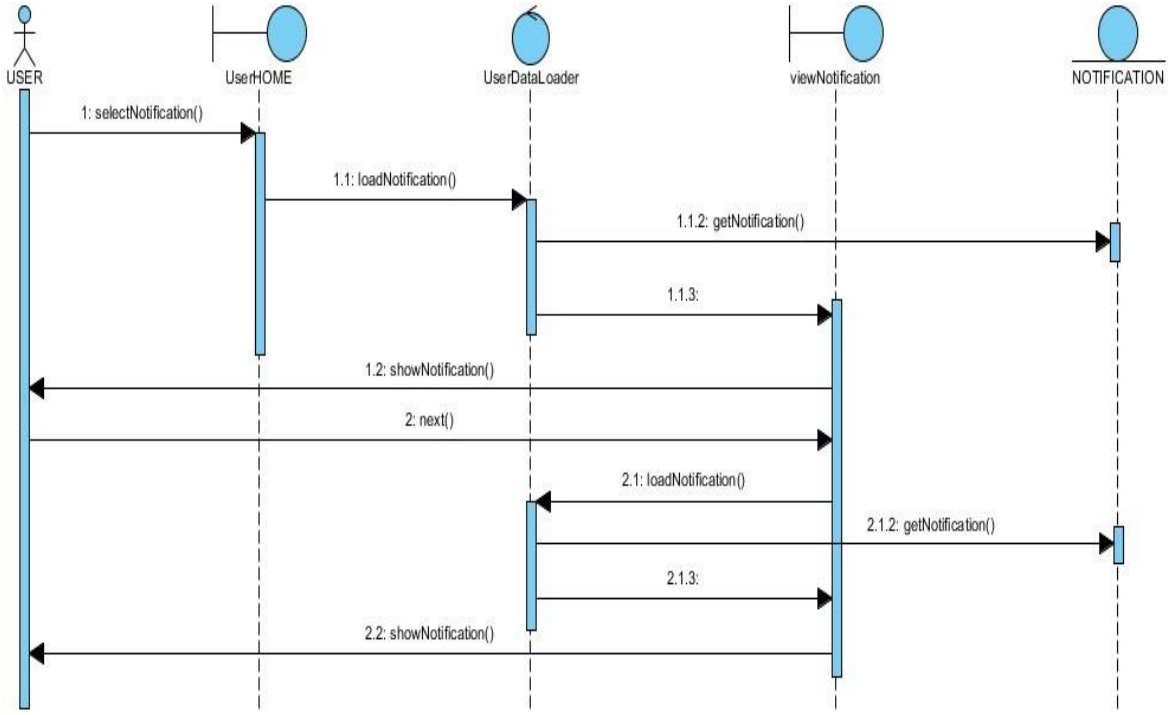
5.2 Show Auction

Let's see in this diagram the operations that performs the system to show the auction to the user.



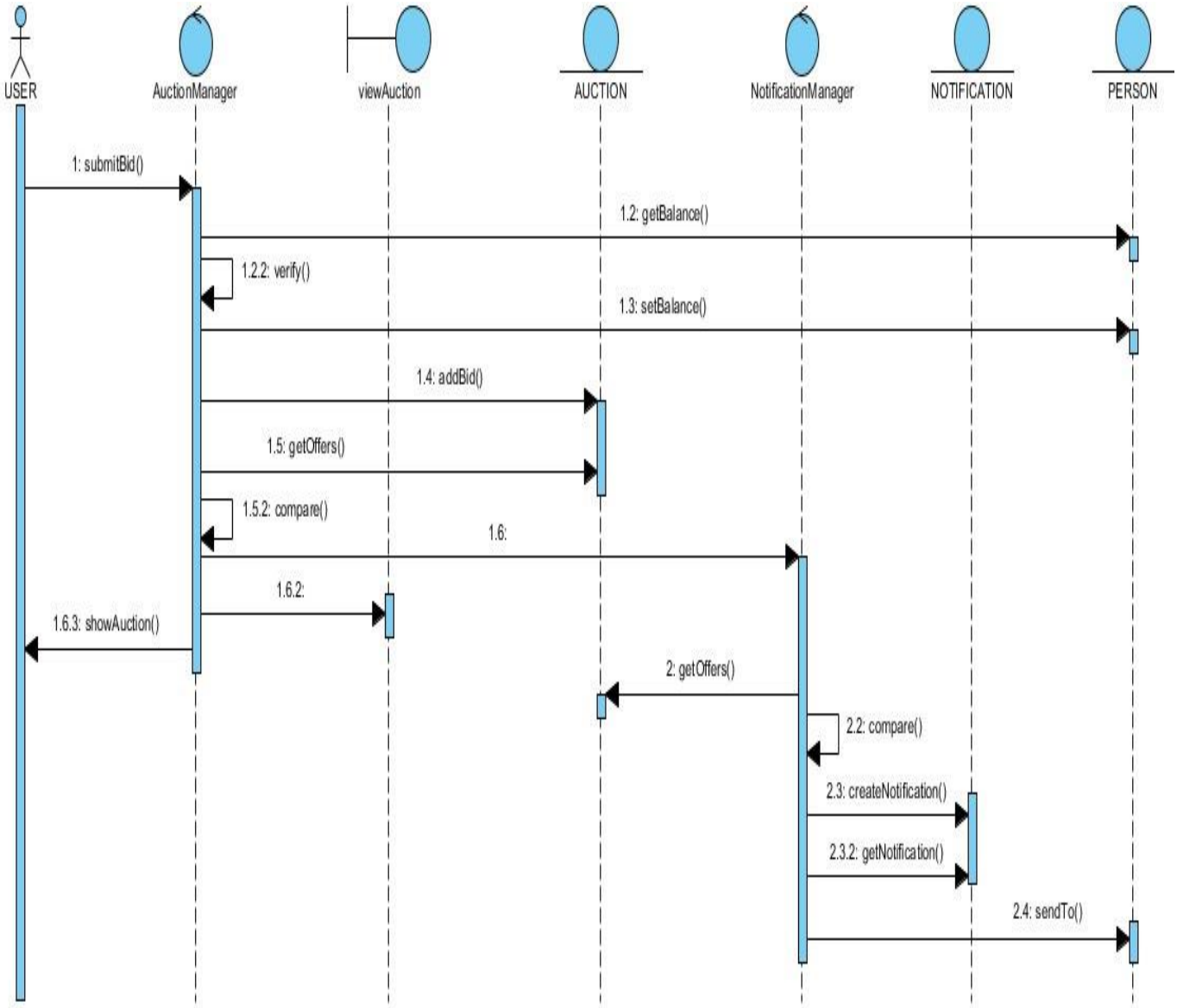
5.3 Show Notification

Let's see in this diagram the operations that performs the system to show the notifications to the user.



5.4 Bidding and sending notifications

Finally, let's see how the system, after an user bids, creates, if there is a change in the ranking, and sends the notification to the corresponding users.



6. USED TOOLS

The tools that I used to create this DD document are:

- **Microsoft Office Word 365**: to redact and to format this document;
- **Star UML**: to create Sub-System Diagram, UX Diagrams and BCE Diagrams;
- **Java Diagrams E-R (JDER)**: to create the entity-relation model of the DB.
- **MySQL Workbench**: to create the physical structure of the DB.
- **Visual Paradigm**: to create Sequence Diagrams.