
POLITECNICO
MILANO

SOFTWARE ENGINEERING 2

TEACHER:
PROF. RAFFAELA MIRANDOLA

GuessBid

REQUIREMENT ANALYSIS AND SPECIFICATION DOCUMENT

Authors:
Eliseo Sartori
Alessandro Picca

Sommario

1. INTRODUCTION

1.1 Aim

This document aims to describe the functionalities of "GuessBid" , a system that simulates a particular kind of auction: in order to win them, a user has to propose the lower unique bid.
The document will show main characteristics and issues of the system but also the design strategies, assumptions and constraints that make this software real and easy to implement. According to this target we want to intend it to managers and stakeholders who will finance the project, to make sure they understand all aspects and topics of the software, but also to software developers who are interested in its behaviors and entities.

1.2 General Description

The system we have to implement, GuessBid, is an online auction house that works in a particular way: the user who win the auction will be the one with the lowest unique bid, this means that if there is, for example, a user that bids for 3 and two or more users that bid for 2, the winner of this auction will be the former, because 2 is not a unique bid.
In order to use this web-application, a user has to register himself and create an account with his personal and mandatory information (surname, name nickname, city, age).
Once registered, the user receives by the system an amount of 100 virtual credits, that will be needed when he wants to bid; in fact, every time a user places a bet, 2 credits will be removed from his account.
In this web application, users can create a new auction for each good they have, defining an expiration date after which the auction expires.

In addition, they can also bid for an existing auction (obviously not created by themselves), after browsing them.
Every time a user places a bid, the system will inform him about his situation in the auction, and it will send notification if this situation changes before the auction expires.
A user can also keep track of all his past bids in a specific section of his page and every time an auction ends, he receives a notification by the system with the outcome of the auction.

More precise information about all possible options available in this application are given below.

1.3 Goals

In our conception of the application, GuessBid has to provide the following main features:

- Registration of a person in the system;
- Log-in the system;
- Creation of an auction for an own good with relative expiration date (Once a user is registered and logged);
- Bid for an auction not created by himself at a cost of 2 per bid (the only restriction about the number of bids placed by a user applies to the case where the credit of the user is lower than 2: in this case he can not bid);
- Check the story of previous auctions where the user was involved;
- Notify to users their position in the auction after each bid and the outcome of the auction once it is finished;

1.4 Domain properties

- Users should receive notifications by the system;

- Users can't spam or abuse other person, otherwise they will be banned;

- the part relating to the transfer of ownership of an asset after winning the same goods at an auction (methods and times of the delivery) is not handled by the application, but by an external system to which users must register separately;

- When a user creates his profile, he has to declare only true information about himself;

- 2 credits will be withdrawn from user's account every time he will bid;

- The system not allows users to bid for an auction if it closed or their credits are less than 2.

1.5 Assumptions

- Users have only to auction goods in acceptable conditions, and, if such goods should present faults and / or problems not compromising their use, they must report it in the auction description, so that the auction takes place in conditions of maximum transparency possible;

- People can interact with the web application only after registration and log-in;
- Notifications are received by users participating at an event only when they log in;
- It assumes that in no way users can collude each other and alter the outcome of an auction, so we refer to the common sense of all those who benefit from GuessBid;
- All users that had registered on GuessBid must be adult (the system allows the registration request only if the user is 18 or more, because this is the age of majority in Italy, the country where the software will be developed).

1.6 Glossary

Throughout this document , there will be used particular terms related to the specific application context. Below, it is shown the meaning in which these terms have to be understood :

- GuessBid: the name of the application itself

- Auction: series of bids that allows who set the minimum bid to take possession of the object to which the auction relates;

- Outcome: notification sent by system to all participants to an auction, informing them of any success or failure of the auction;

- Lowest Unique Bid: value that allows user to win a specific bid: if the minimum bid has been placed by more than one user, the lowest unique

bid becomes the one with the minimum value and such that only one user had placed it;

- User: a person already registered in GuessBid;

- Virtual Credit: amount of "credits" given to a user after he performs registration; the initial value of this credit is 100 and it will be decremented by 2 ever time the user bids;

- External user: user who interacts for the first time with the application or that is not registered yet;

- Historical: summary of the outcome of all past auctions in which the user has been involved and which is available from the user's page

- Notification: notice that a user receives every time he places a bid; this notification informs the user about his position in the specific auction. System sends a notification to a user every time his position changes before the expiration of an auction.

1.7 Reference Documents

- IEEE Recommended Practice for Software Requirements Specifications: http://standards.ieee.org/findstds/standard/830-1998.html

- UML class diagram attached to this paper: "GuessBidUML.jpeg" (see section 3.4 of this document)
- Alloy code attached to this paper: "GuessBid.als"

1.8 Overview

The rest of the paper is organized as follows:

Section 2, Actors Identifying: it identifies which are the main actors of the application;

Section 3, Requirements: describes the general factors that influence the production of the software and provides a detailed description of the functional and non-functional requirements, it also contains scenarios, use cases and modeling of the product;

Section 4, Appendix: provides modeling with Alloy.

2. ACTORS IDENTIFYING

- User: a person who has registered and so has provided his personal information.

- External User: a person who is not registered yet and can only register himself in the web-site.

3. REQUIREMENTS

The proposed system consists of a web application totally independent as of its functionality: it is not a part of a larger system, neither provides the specific components that extend the functions. It is not a stand-alone system, as it uses some software dependencies and hardware that will be listed.
The decision to make the system accessible to users as web application and not as an executable file has been driven primarily by the desire to make it accessible remotely (via internet) and regardless of the software installed on the user's device, which shall not be in any way "burdened" by the application.

3.1 Functional Requirements

3.1.1 Registration of a person in the system:

- The system has to provide a sign up functionality.

3.1.2 The possibility to create an auction (Once a user is registered):

- The system will make possible to create an auction for an own good
- The system force you to insert an expiration date for every auction you create
- The update or the delete of an auction that has yet been created is not allowed by the system

3.1.3 Bid for an existing auction

- The system allows user to place bids for an auction that has not yet expired
- System alerts user and don't allow him to bid if his virtual credit amount is less than 2
- Every bid has a cost of 2 virtual credits, that will be subtracted from the user's balance
- If the balance permits it, there is no limit for the number of bids that a user can place in a single auction

3.1.4 Keep track of the story of the previous auction

- User can check any time a brief summary of all auction where he's been involved in

3.1.5 Auction information

- User visualizes his position in the specific auction every time he bids
- User also receives a notification when an auction in which he has been involved expires
- User visualizes also when other users overtake his previous bid

3.2 Non functional requirements

3.2.1 Architectural Consideration

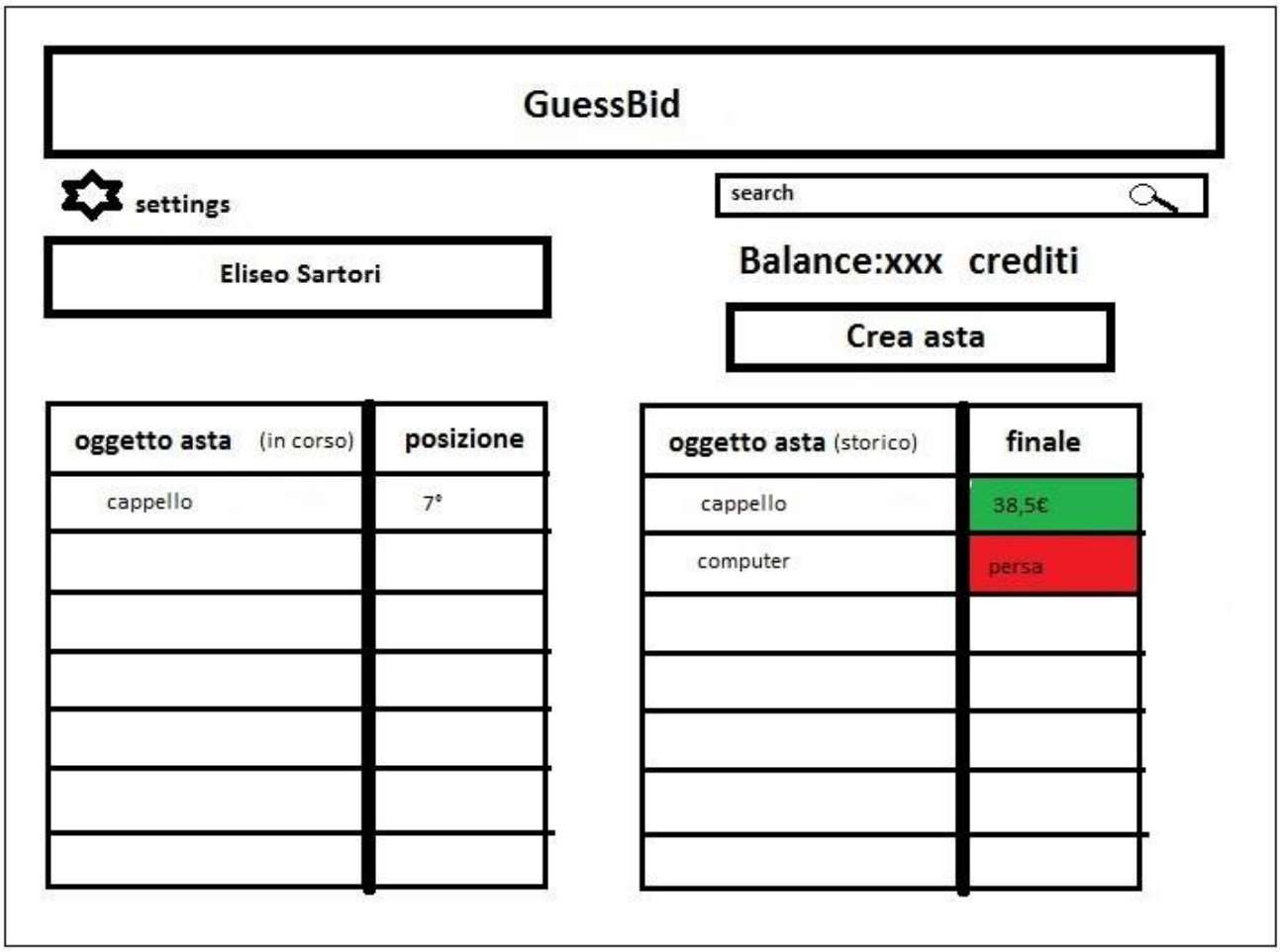
The application is based on a client-server architecture, that is divided in two separate and independent parts: the server side, which provides the request service, and the client side, which can be used by users in order to find the offered services.
We will use Java Enterprise Edition platform with a MySql database in which all system's information will be stored.
The Class Diagram given below would allow you to understand more about this information , because it could be considered as a basis for our database, but the precise ER Diagram will be drawn in the DD.
In order to use GuessBid, a stable Internet connection and also a recent web browser are required.

3.2.2 System Interfaces

For both client and server side, the system requires the use of standard input devices (keyboard and mouse) to interact with the GUI (client) and the character-based interface (server) that the system will provide. There are no specific extensions for touchscreen devices, nor we are thinking to their development in the future.

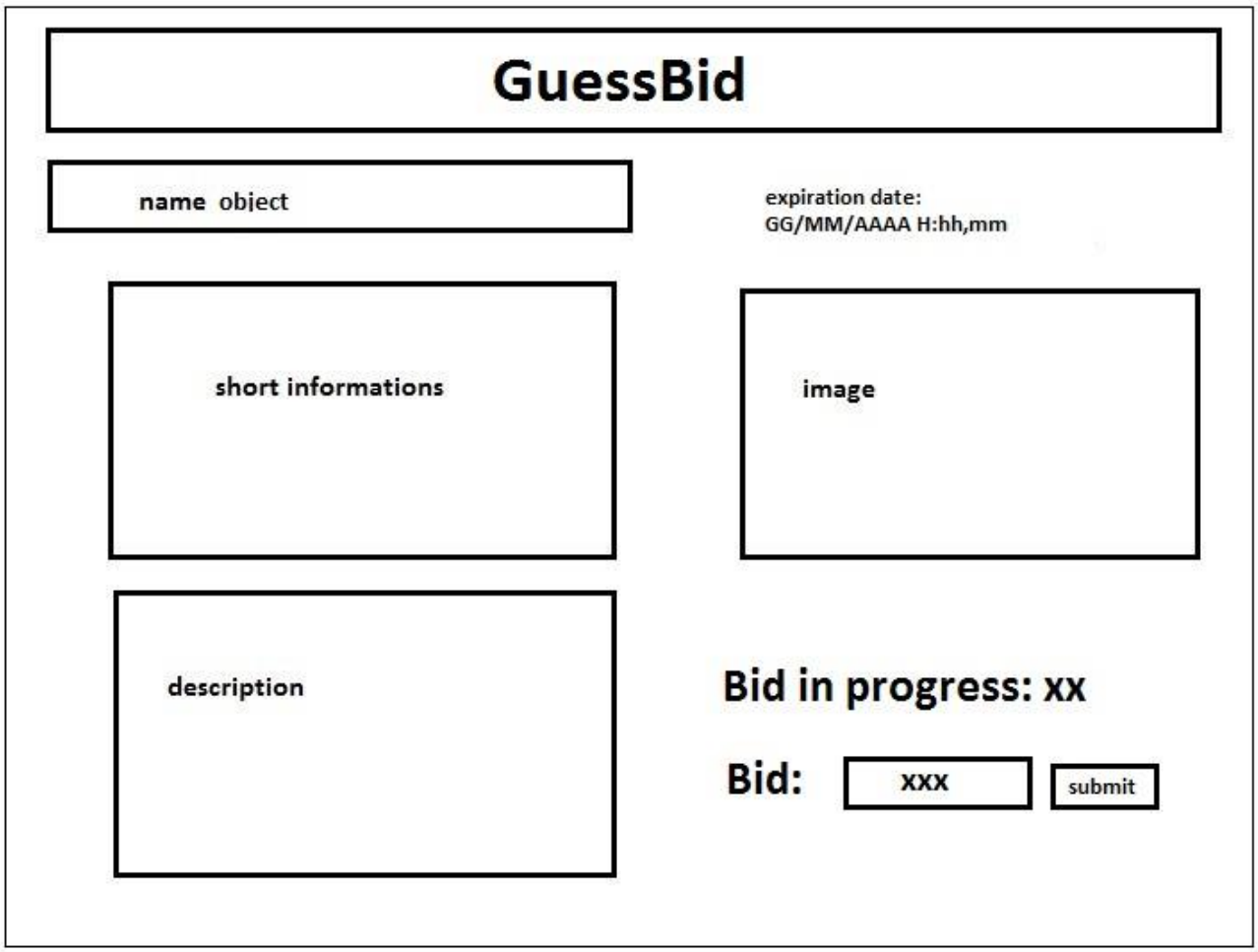
3.2.3 User Interface

Our purpose is to create a simple and easy website, that anyone can use; Our user interface has to provide a log in page that can connect each user directly to its profile page with a link, allowing him to find only the functionalities that he can effectively use.
The interface has also to show all the functionalities provided by the platform in the first page after the log in (like place a bid, check open auctions…); in particular the system would be able to generate dynamic web pages that have different functionalities depending on the type of user (e.g: an external user won't be able to see an auction created by a logged user).
Here we will list the main pages of the application and show a first draft of some of their sketches page :



This is an example of a user's personal page, here there are:

- User's name and surname (eventually the selected username)
- The section "Settings", where the user can change some information about his profile
- The amount of actual virtual credit balance
- A section that contains active auction in which user is involved, each of them with the current position of the user
- A section with all expired auction where user had placed at least one bid, with the outcome of every auction
- Search-bar, where a user can search auctions



This is an example of an auction page (of course not created by the user, otherwise he can't place bids), where the user can found all information about the auction and the related object; here there are:

- The name of the auctioned object;
- Short information, some brief information about the object
- A more detailed description, containing all the specifications and features of the object of the auction
- A picture/image of the object;
- The expiration date for this auction;
- The possibility to place a bid;
- The number of bids already placed for this auction.

3.2.4 Documentation

We will write these documents to organize better our work, both in terms of quality and time spent, and to make its understanding easier for those who will see this work in the future:

RASD: Requirement Analysis and Specification Document, which is drafted to well-understand the given problem and to analyze in a detailed way which are our goals and how to reach them defining requirements, specification, scenarios and use cases.

DD: Design Document, a written description of the software product, that we will draft in order to give an overall guidance to the architecture of the software project.

JavaDoc comments in the source code: comments added in order to permit to anyone that wants to develop the platform or do maintenance to better understand the code.

Testing Document: a report that describes our testing experience of another GuessBid project.

3.2.5 Database Requirements

GuessBid uses a MySql database and aims to ensure the correctness and consistency of its transactions; among other things it is necessary that the following conditions occur:

- There can be no more than one profile with the same username;
- All dates in the database must be later than April 1 2015;
- An organizer cannot set an expiration date for an auction in the past;

- In each table there must be a numeric identifier, which has the function of primary key;
- NULL value isn't admissible in any database record;
- The application server back-end must check the consistency of the data entered;
- Each attribute that is part of a WHERE clause of a query executed by the database must be associated with at least one index;
- There must be no duplicate records, nor inconsistent ones;
- Database operations that violate these constraints must be prevented by the system before they commit.

3.3 Possible scenarios:

Here we show some of the possible scenarios relating to GuessBid, in order to clarify the interactions between users and system

3.3.1 Signing up on GuessBid

- Ale knows that Paolo has auctioned his old GameBoy on GuessBid, so, since he's a Nintendo maniac, he decides to sign up and create an account, entering an username, a password and an email address in the appropriate section.
  The system sends a confirmation email to Ale containing the activation link; Ale clicks on the link and enters his personal page in the application.

3.3.2 Create an auction

- Davide bought a new hat, so he does not know what to make of the hat he had before, so, remembering that some time ago he registered on GuessBid, he decides to log on and auction his old hat, setting also an expiration date and confirming the operation.

3.3.3 Place a bid

- Eliseo, once he has logged on GuessBid, see that there is an open auction for the last model of smartphone, so he places his bid

3.3.4 Notice that the actual position in the auction has changed

- Danilo placed a bid for a cuckoo clock about two hours ago and after that he's gone out;
  When he re-logs on GuessBid, he notices that someone has placed a best bid for the clock, so he place a better offer than the previous

3.3.5 Check the story of your auctions

- Riccardo spent too much in the last month, so he want to visualize a summary of his auction on GuessBid; to do that he, after logging in, consults the section of his profile relative to the story of his auction and, unfortunately for him, he realizes that he really spent too much

3.3.6 Saw the notification

- Paul look at his homepage and notice that he win the auction of a Playstation, so decide to contact the external agency that control the payment and the delivery of the object.

3.3.7 Updating personal profile's information

- Paul remembers that when he signed up to GuessBid he was in a hurry, so he included in his profile only the essential information; today Paul is on holiday and he has more time, so he can add optional information to his profile, and he does it first by logging in, then selecting his own personal page and then adding "go fishing" and "collect small soldiers" in the sections "Hobbies" and "teacher" in the section "Work".

3.3.8 Receive a notification about the outcome of an auction

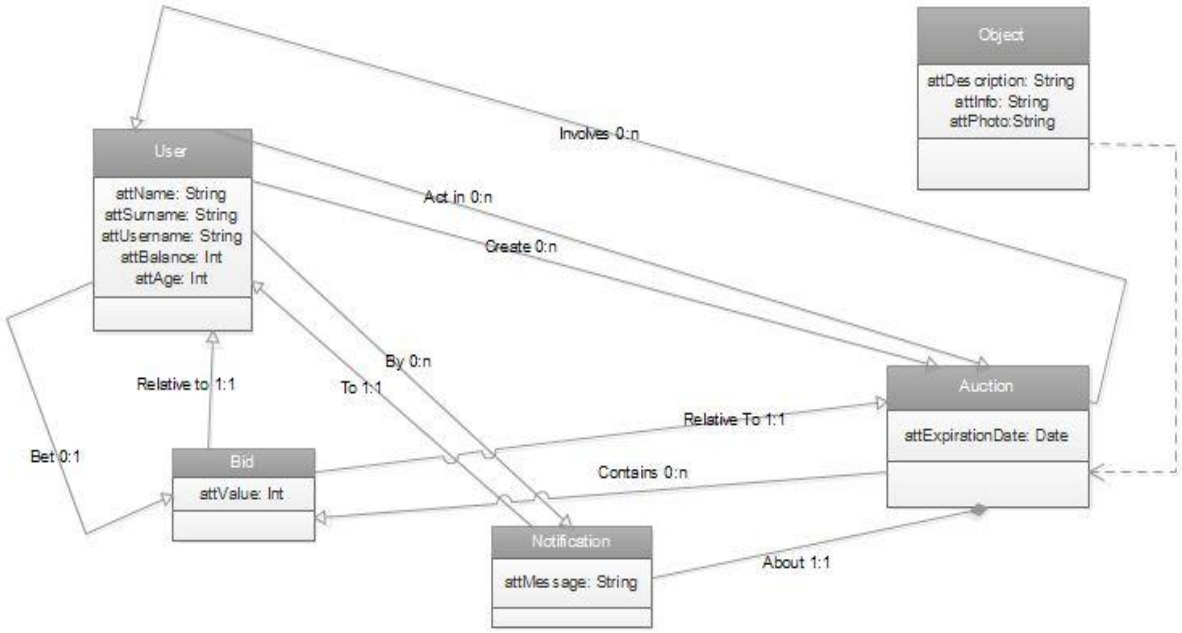
- The auction for cooking pots has expired and the system communicate Romeo that, unfortunately for him, he hasn't win.

3.3.9 Check balance amount

- Pablo placed some consecutive bids for an object that he likes very much, and has lost count of his virtual credits, so, from his home page, he check how many credits he has now.

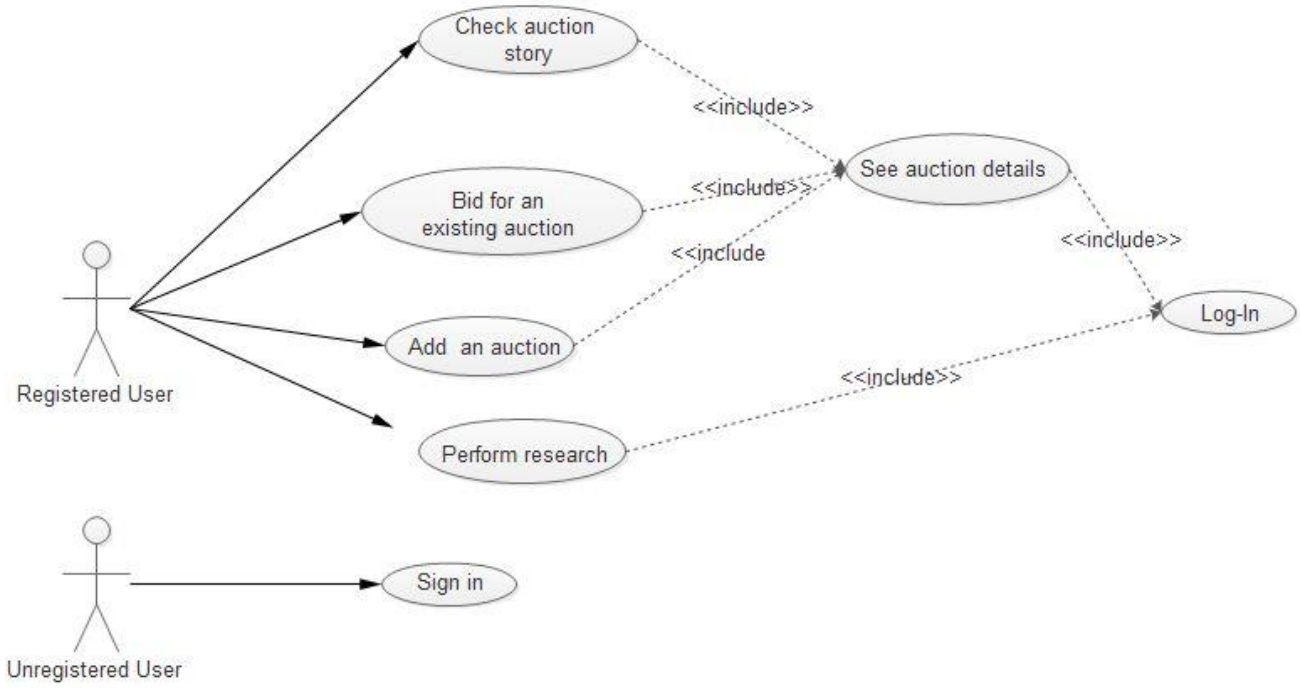
3.4 UML Class Diagram

This section shows the UML class diagram, used to model the classes that will be part of the system. Some of these methods will be exploited in the following sections for modeling use cases using sequence diagram.



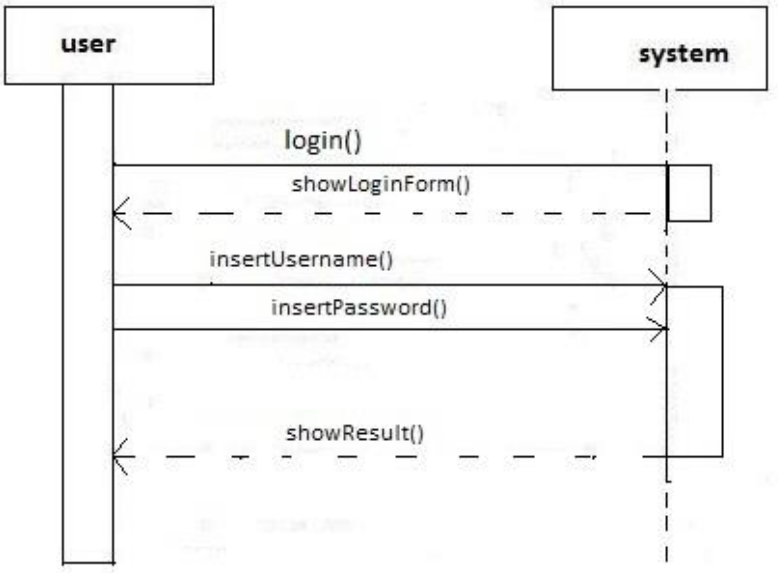
3.5 Use cases and sequence diagram

The following section presents the UML Use Case Diagram, then some use cases of the system will be listed with their relative Sequence Diagram.



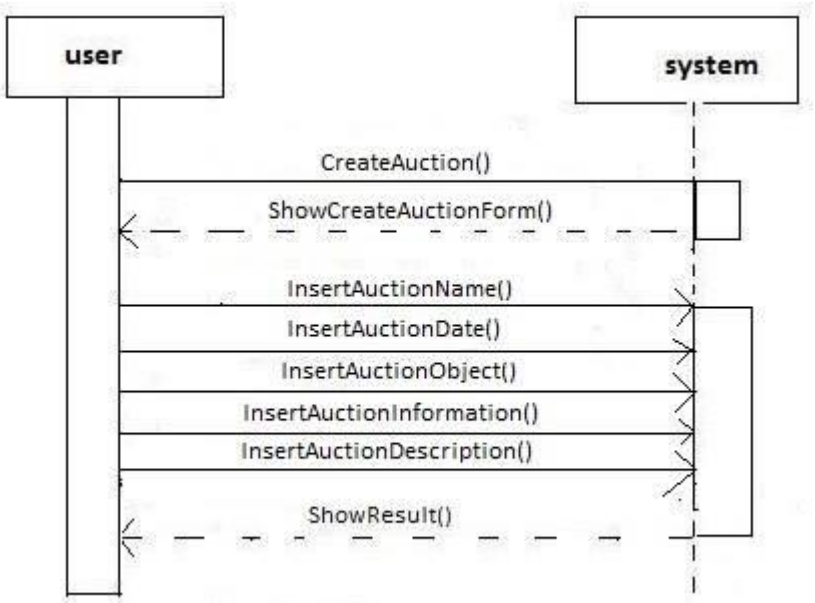
3.5.1 User Login

Title	User login
Actors	User
Input Conditions	The user is registered to the system
Flow of Events	• The user reaches the system; • The user clicks on "login"; • The system provides a login form; • The user enters the username and password he chose at the moment of the registration; • The system informs the user of the outcome of the operation
Output Conditions	The user enters his personal page
Exceptions	The data entered are incorrect



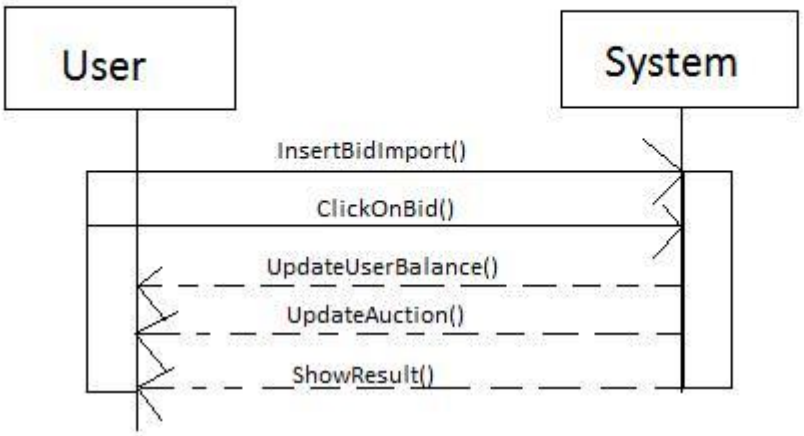
3.5.2 Creation of an auction

Title	Auction Creation
Actors	User
Input Conditions	The user is in his personal page
Flow of Events	• The user clicks on "Create auction"; • The system provides a form where the user has to insert auction's details; • The user enters description, information, a picture of the auctioned good and the expiration date • The system informs the user of the outcome of the operation
Output Conditions	The new auction will be created and other user can bid for the related object
Exceptions	The data entered are not consistent (e.g. expiration date prior to the current date)



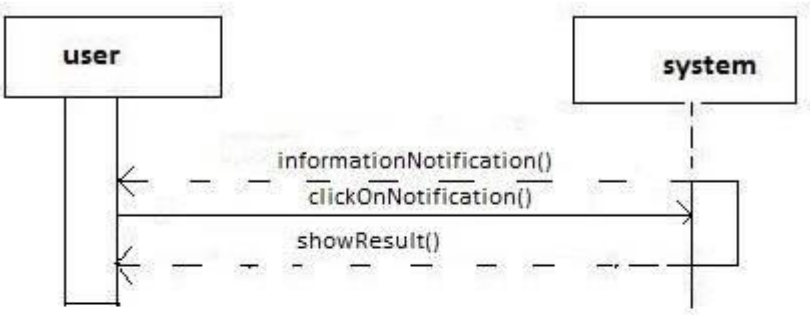
3.5.3 Bid for an active auction

Title	Bid for an active auction
Actors	User
Input Conditions	The user is on the auction's page
Flow of Events	• The user inserts the amount of his bid • User clicks on "Bid for this auction!" • The system updates the auction with the new bid; • The system withdrawn 2 credit from user's balance; • The system sends a notification to the user, informing him about his position in the auction;
Output Conditions	The user is informed about his position and his amount of credit is decreased by 2
Exceptions	User has less than 2 credits



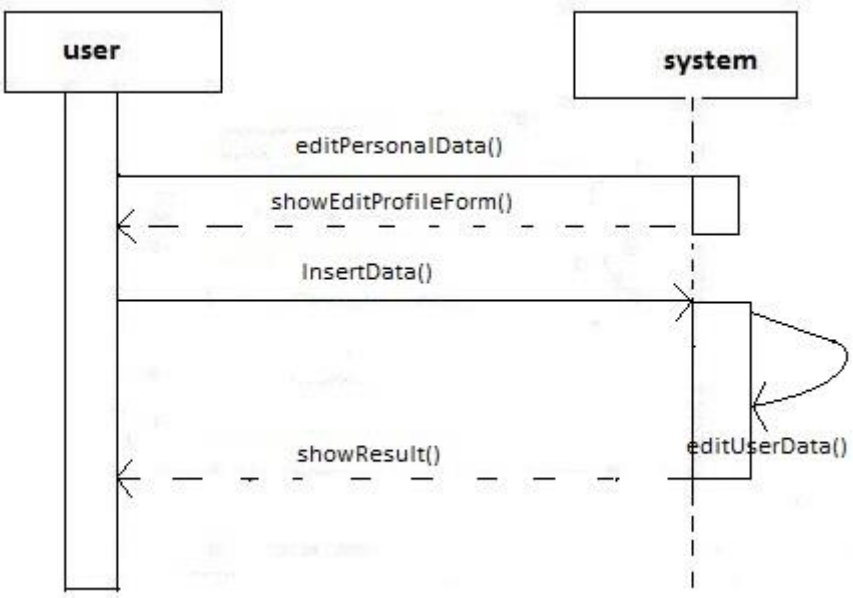
3.5.4 Notification of the outcome of an auction

Title	Notification of the outcome of an auction
Actors	User
Input Conditions	An auction has expired 2 seconds ago
Flow of Events	• The system informs users involved in the auction with the outcome • User clicks on “Ok” and go back to his page
Output Conditions	User is informed that he has won/lost the auction
Exceptions	There are no possible exceptions



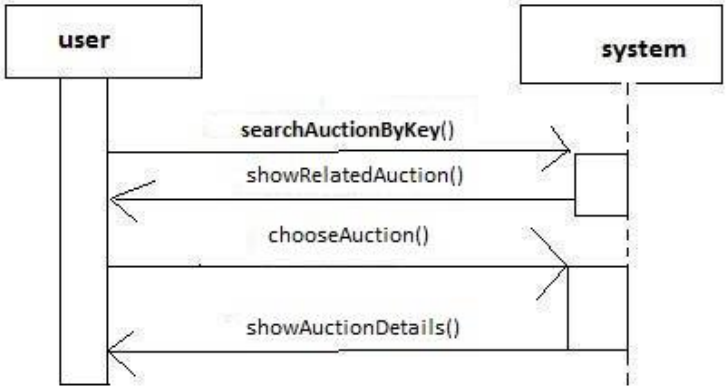
3.5.5. Modifying personal profile

Title	Modifying personal profile
Actors	User
Input Conditions	The user is logged and he's on his personal page
Flow of Events	• The user clicks on "Modify my profile" option; • The system shows a modify form; • The user modifies his personal data • The system informs the user of the outcome of the operation
Output Conditions	The system notifies the user of the data changes.
Exceptions	The new data entered are incorrect.



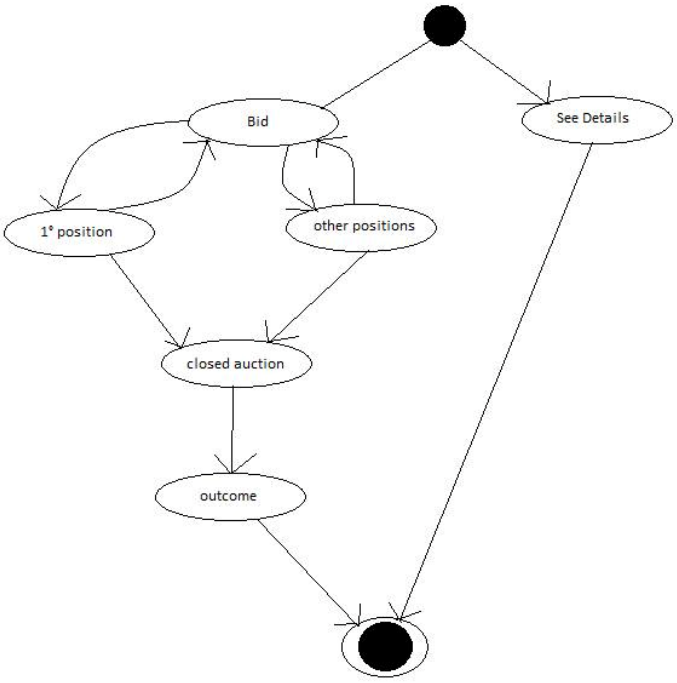
3.5.6 Search for an auction

Title	Search auction
Actors	User
Input Conditions	The user has just logged in
Flow of Events	• The user clicks on "Search"; • The system provides a search form; • The user enters the object or any detail about what he want to find; • The system show all results related with the keyword(s) (eventually nothing)
Output Conditions	Outcome of the research noticed to user
Exceptions	There are no possible exceptions

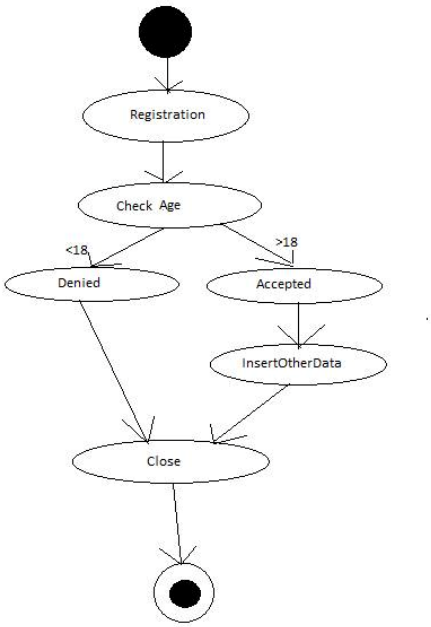

user
system
searchAuctionByKey()
showRelatedAuction()
chooseAuction()
showAuctionDetails()

3.6 State chart diagrams

3.6.1 Auction Lifecycle



3.6.2 Check validity of a registration



3.7 Performance Requirements

There aren't performance requirements.

3.8 Design Constraints

3.8.1 Standard compliance

The proposed system, particularly with regard to the web application presented to clients, is in compliance with the following international standards:

- HTML5 - http://www.w3.org/TR/html5/
- CSS3 - http://www.w3.org/Style/CSS/

3.9 Software system attributes

3.9.1 Reliability

GuessBid is designed to have, as far as possible, maximum reliability;
to do this we decided to place the central server in an area equipped with fire system, anti-flood system and a generator which maintains the server active even in case of electrical blackout, in order to prevent crash related to disasters or accidental events.
The system must also ensure that the transactions that take place on the database are consistent, atomic, isolated and persistent over time, so as not to allow the occurrence of conflicts between different transaction, that can cause a system crash.

3.9.2 Availability

The system must be available throughout all the day without interruption. The operating period of the server must be 100% to ensure efficiency and safety

3.9.4 Maintainability

Our goal is to create a software as stable as possible over time, so that it should not be modified in the short term.
Maybe there could be some changes to do and the implementation would be trivial, because we have decided to split the software's code in different classes, in order to maximize the reuse and make maintainability easier.

4. ALLOY

In order to show our system's consistency we have created its model in Alloy as follows:

```
module GuessBid

open util/boolean

////////////////////////////////////
//////////SIGNATURES//////////////
////////////////////////////////////

sig ExternalUser,Name, Surname, Username, Password, Message {}
//These signatures are considered empty for simplicity, because their attributes are
//not useful for the alloy modelization.

//String attributes are not considered for simplicity, because
//they're not useful for the alloy modelization

sig Date{
timestamp: one Int //every date corresponds to a specific timestamp
}{
timestamp>=0
}

sig Auction{
expdate: one Date,
creator: one User,
object: one Object,
bid: set Bid
}

sig User {
username: one Username,
password: one Password,
name: one Name,
surname: one Surname,
balance: one Int,
age: one Int,
createdauction: set Auction,
```

```
involvedauction: set Auction,
notification: set Notification,
bid: set Bid
}
{
all disjoint a1,a2: createdauction | a1.object != a2.object
//the same object can't be auctioned twice by a user

no(createdauction & involvedauction)
//a user can't bid for an auction (be involved in) that has created

}

sig Object{
description: one String,
photo: lone String,
information: one String,
}

sig Notification{
message: lone Message,
receiver: one User
}

sig Bid {
value: Int,
bidder: one User,
relatedauction: one Auction
}
```

```
////////////////////////////////////
////////////FUNCTIONS/////////////
////////////////////////////////////

fun UserAuction [u:User] : set Auction{
	u.involvedauction
}
//given a user, returns all auctions in which user bid

fun UserAuction [u:User] : set Auction{
	u.(involvedauction + createdauction)
}
//given a user, returns all auctions created by user or in which he had participated


////////////////////////////////////
///////////////FACTS////////////////
////////////////////////////////////


//Simmetries
fact simmetries{
all a:Auction | a in a.creator.createdauction
all u:User | all a: u.createdauction | u= a.creator
all a:Auction |all u:User | a in u.involvedauction implies (!u in a.creator)
all b:Bid | b in b.relatedauction.bid
all n:Notification | n in n.receiver.notification
all u:User | all n:u.notification | n.receiver =

u
all u:User | all b:u.bid | b.bidder=u
}
```

```
//Duplications
fact noDuplications{
all u1,u2: User | u1.username = u2.username implies u1=u2
all a1,a2: Auction | a1.object = a2.object implies a1=a2
all d1, d2: Date | d1.timestamp = d2.timestamp implies d1=d2
all b1,b2:Bid | (b1.value = b2.value && b1.bidder = b2.bidder) implies b1=b2
}


////////////////////////////////////
//////////ASSERTIONS//////////
////////////////////////////////////

assert UserAuctionSimmetry {
all a: Auction | a in a.creator.createdauction
all u:User | all a:u.createdauction | u=a.creator
}
check UserAuctionSimmetry


assert NoDuplications {
all u1,u2: User | u1.username = u2.username implies u1=u2
all a1,a2: Auction | a1.object = a2.object implies a1=a2
all d1, d2: Date | d1.timestamp = d2.timestamp implies d1=d2
all b1,b2: Bid | (b1.value = b2.value && b1.bidder = b2.bidder) implies b1=b2
}
check NoDuplications

assert Coherence {
#Object = #Auction
}
check Coherence
```

```
pred Show() {
#User = 2
}

run Show


////////////////////////////////////
/////SYSTEM FUNCTIONS////////
////////////////////////////////////

sig Database {
users : set User,
auctions : set Auction,
bids : set Bid,
notifications: set Notification
}{
#Database = 1
all u:User | u in users
all a:Auction | a in auctions
all b:Bid | b in bids
all n:Notification | n in notifications
}

//database predicates examples
pred addUser[d:Database, u:User] {
u !in d.users implies d.users = d.users+u
}
run addUser



pred login [eu:ExternalUser, un:Username, p:Password]{
one u:User | u.username = un && u.password = p
}
run login
```

Executing "Check UserAuctionSimmetry"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3639 vars. 414 primary vars. 6633 clauses. 488ms.
No counterexample found. Assertion may be valid. 18ms.

Executing "Check NoDuplications"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3641 vars. 414 primary vars. 7068 clauses. 112ms.
No counterexample found. Assertion may be valid. 10ms.

Executing "Check Coherence"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3654 vars. 414 primary vars. 6625 clauses. 68ms.
No counterexample found. Assertion may be valid. 37ms.

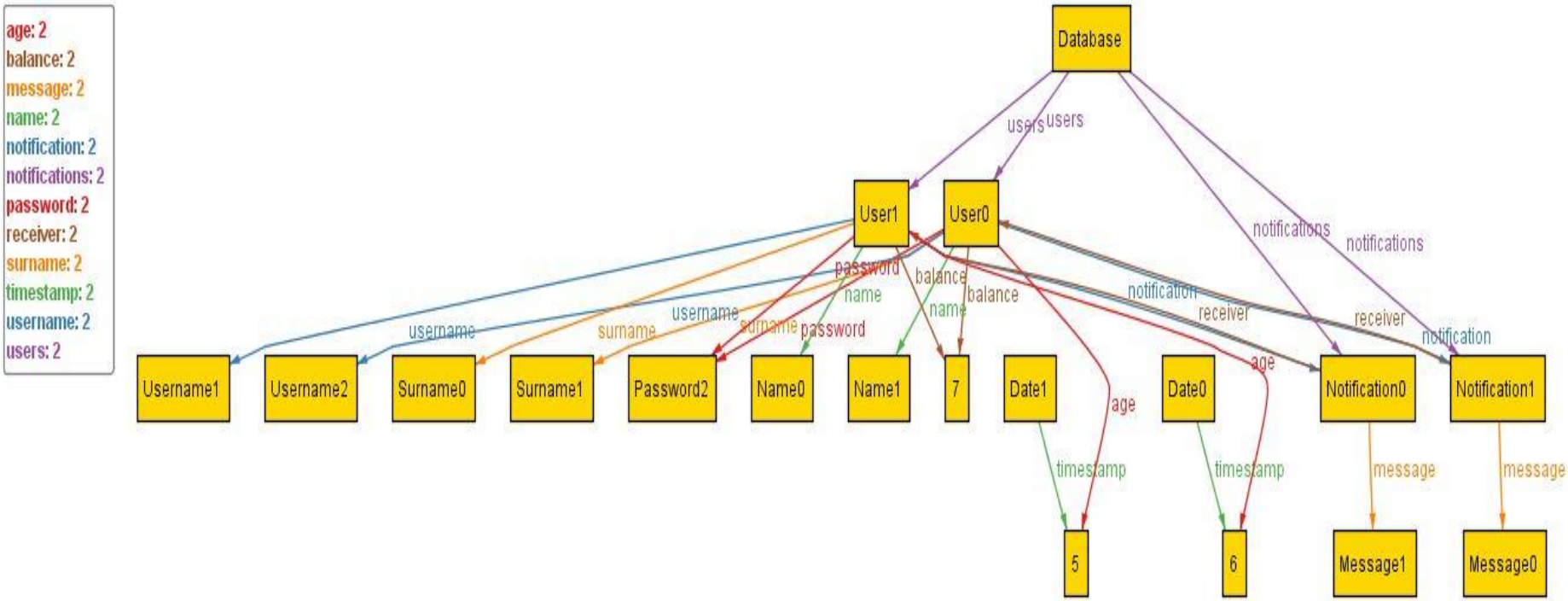
Executing "Run Show"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3632 vars. 411 primary vars. 6561 clauses. 77ms.
Instance found. Predicate is consistent. 41ms.

Executing "Run addUser"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3691 vars. 417 primary vars. 6661 clauses. 55ms.
Instance found. Predicate is consistent. 49ms.

Executing "Run login"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3754 vars. 420 primary vars. 6835 clauses. 63ms.
Instance found. Predicate is consistent. 53ms.

World Generated

Note for reader: The following two images have to be seen as one, it's not been possible show generated world in a single image because of the item's size.


age: 2
balance: 2
message: 2
name: 2
notification: 2
notifications: 2
password: 2
receiver: 2
surname: 2
timestamp: 2
username: 2
users: 2
Database
users
users
User1
User0
notifications
notifications
password
balance
name
balance
name
notification
receiver
receiver
notification
username
username
surname
surname
password
age
Username1
Username2
Surname0
Surname1
Password2
Name0
Name1
7
Date1
Date0
Notification0
Notification1
timestamp
timestamp
message
message
5
6
Message1
Message0