Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 19 Bratislava

Software modeling

Charity e-auctions

Model documentation

Matej Hrnčiar
3. ročník, 5. semester
Utorok, 14:00

Charity has always been a way for people to show philanthropy, but nowadays, the sole feeling of satisfaction from helping others is not enough for many people. Charity auction is a way to raise money to help people in need with also giving something back to people who contributed to the charity. Charity auctions can auction off physical objects that were gifted to the auction, like autograph or guitar of a famous musician interested in the auction, but they can also auction off non-physical objects and experiences, like meeting with a celebrity or given the opportunity to name a character in a book or a game. This works to some extent on platform kickstarter.com, where people can support interesting projects and by contributing specific amounts of money, can name a character or contribute to project in other ways.

Online environment is ideal to make the auction accessible to the as many people as possible. System would differentiate three main actors:

1. Organizers, who can create new auctions, give them names and choose charities that would be supported by said auctions.
2. Bidders, who can browse auctions and participate in the ones that interest them. In every auction, they can view available prizes and bid for the ones they want.
3. Operators, who oversee the system, create and edit charities, resolve issues with auctions and users and ban scammers.

Payment to charity is automatic, so after the end of e-auction and collecting all payments from bidders, almost all of the raised money is sent to charity, with small provision going to system to keep it running. bidders' payment could also be automatic, the user would enter their card details or bank details during registration and the system would automatically bill the bidder after winning the bid. If the payment is manual and the winner does not pay the price, the organizer is able to report the user and operator may ban him from participating in other auctions.

However, it can also be the case that the organizer does not sent a prize or donate the money for the purpose for which they were raised - in that case, other users can report the organizer, which can banned by operator as well.

UseCase Model

UC01: Register Organizer

Actor: Organizer
Goal: Registration of new organizer

The charity e-auction system is open to anyone, so anyone can become an organizer. Registration is similar to registration on any other site, but the organizer needs to understand that running an e-auction produces a legal liability. That's why the organizer has to be verified to make sure that he's not trying to become an organizer to scam bidders and charities. Registration requires basic information (name, address, e-mail, date of birth, …) and special information (ID number of person or identification number of organization - IČO).

Scenario:

1. Person who wishes to become an Organizer selects the option to register
2. System opens the registration window
3. Organizer inserts personal or organization information (name, OP/IČO, date of birth, address), username and password
4. System checks whether the username is available, password is correct and personal/organization information is correct

   Exception - Some information is incorrect:

   4.1 System produces warning that some inserted information is incorrect
   4.2 Organizer is given option to make changes
   4.3 *Return to step 3*

5. System checks if the information inserted don't appear in the list of banned Users

   Exception - User has been banned:

   5.1 System finds a match in the list of banned users
   5.2 System produces a warning that the registration cannot be completed since the User has been previously banned
   5.3 System returns to main screen

6. System registers new Organizer to database and logs him in

UC02: Create new e-auction

Actor: Organizer
Goal: Create new e-auction

Every registered organizer may create any number of e-auctions with charitable intent. When creating new e-auction, it is required to insert name, charity, description, based on which other users decide whether to participate in the e-auction or not, and date of start and end of e-auction.

Then, the organizer can insert biddings for prizes, which will be in the e-auction, and add their description, starting price, and start and end of every individual bidding. Traditionally, biddings are realised one after another, so the biddings will be in a list, which order can be changed by organizer, but the biddings may also overlap, so there can be two and more biddings running in parallel. After the start of e-auction, the biddings automatically follow one after another until all prizes are given out and after the end, a list of winners is generated with name and address of a winner of every bidding.

Scenario:

1. Organizer selects the option to create new e-auction
2. System opens the creation window
3. Organizer inserts basic information about e-auction (name, start and end, …)
4. System opens a dialogue window to search for available charities
5. Organizer inserts all biddings along with information about them (description, starting price, start and end of bidding)
6. System checks whether the start (end) of the first (last) bidding is the same as the start (end) of the e-auction

   Exception - the dates don't match:

   5.1 System produces a warning that the start (end) of the first (last) bidding has to be the same as the start (end) of the e-auction

   5.2 Organizer is given option to make changes

   5.3 *Return to step 4*

7. System produces a summary of the e-auction
8. Organizer confirms the e-auction

   Exception - no prizes were inserted:

   7.1 System produces a warning that the e-auction cannot start without biddings

   7.2 Organizer is given option to add biddings

   7.3 *Return to step 4*

Alternate - make changes:

7.1 Organizer decides to make changes in the list of prizes

7.2 *Return to step 4*

9. System plans the e-auction and shows a confirmation message

UC04: Make changes in e-auction

Actor: Organizer
Goal: Make changes to an existing e-auction

When there is an existing e-auction, it must be possible to make changes in it. It may involve changing the name, the date of e-auction, the list of biddings or even changing the purpose of the e-auction. Of course, the changes have some boundaries - organizer cannot change the date of e-auction that already started or remove prize that was already won.

Scenario:

1. Organizer selects the option to make changes to an existing e-auction
2. System opens the window with information about e-auction with list of biddings
3. Organizer makes changes to the e-auction (add/remove biddings, change date of e-auction, …)

   Alternate - cancel e-auction:

   3.1 Organizer decides to cancel the e-auction

   3.2 System invokes the UC06 Cancel e-auction

4. System makes a check whether the (end) of the first (last) bidding has to be the same as the start (end) of the e-auction

   Exception - the dates don't match:

   4.1 System produces a warning that the start (end) of the first (last) bidding has to be the same as the start (end) of the e-auction

   4.2 Organizer is given option to make changes

   4.3 *Return to step 4*

5. System produces a summary of changes
6. Organizer confirms the changes
7. System saves the changes and shows a confirmation message

UC14: Bid in e-auction

Actor: Bidder
Goal: Make a new bid in e-auction

After browsing e-auctions and finding the one that interests the bidder, he can make a new, higher, bid. Every auction can have more than one bidding in progress, so bidder can choose one with the prize he wants and make bid in hopes to win the prize. If the new bid is correct (the amount is higher that the previous one) the system shows the new highest bidder and amount.

Scenario:

1. User chooses an e-auction
2. System opens the e-auction screen with time remaining to the start of the auction or the list of biddings currently in progress
3. User selects a bidding from list
4. System opens the screen with details about the bidding, description and photos of prize, highest bidder and current price
5. User selects the option to make a bid
6. System opens a bidding screen
7. User inserts new amount and confirms the bid

   Alternate - cancel new bid:

   7.1 User decides to not make a new bid
   7.2 System produces confirmation query
   7.3. User confirms the cancellation
   7.4 System returns to auction screen

8. System checks the bid

   Exception - new bid is lower than previous:

   8.1 System produces a warning that the price of new bid is lower than the previous one, which is not allowed
   8.2 User is given option to modify the bid
   8.3 *Return to step 7*

9. System displays the new highest bidder and the new price and produces confirmation message to User

UC16: Report problem with e-auction

Actor: Bidder
Goal: Report e-auction

In case of error or suspicion of possible scam, all users can report a problem. The problem may have low (typo, incorrect description, missing picture) or high (organizer is known scammer, prize won never arrived) priority. That also decides, who should resolve the issue: low priority issues can be resolved by organizer and high priority issues require an operator to analyse the issue, ban users and so on.

Scenario:

1. User decides to report the e-auction
2. System opens the reporting screen
3. User selects whether the system is required to resolve the issue (prize won in bidding never arrived, Organizer is known scammer, …) or not (incorrect or missing description, missing picture of prize, typo, …)
4. User selects the reason from drop-down menu or inserts custom reason
5. User inserts description of problem, attaches pictures and confirms the report

   Alternate - cancel report:

   5.1 User decides to cancel the report
   5.2 System produces confirmation query
   5.3 User confirms the cancellation
   5.3 System returns to e-auction screen

6. System notifies the system operator or organizer based on nature of the report
7. System produces confirmation message
8. In case of problem resolution, system notifies the user

UC20: Update charities

Actor: Operator
Goal: Update list of charities

Every e-auction has a purpose for raising money. It can be to support children in underdeveloped parts of world, help cancer patients, help homeless people, support humanitarian aid to regions affected by natural disasters and many more people in need. Therefore, charities need to be periodically added to database and updated, which is a responsibility of operators.

Scenario:

1. Operator selects the option to update list of charities in database
2. System loads list from database and opens editing screen
3. Operator selects the option to add new charity

    Alternate - edit existing charity:

    3.1 Operator selects existing charity to edit
    3.2 System loads charity information and opens editing window
    3.3 Operator edits information about charity
    3.4 *Join to step 6*

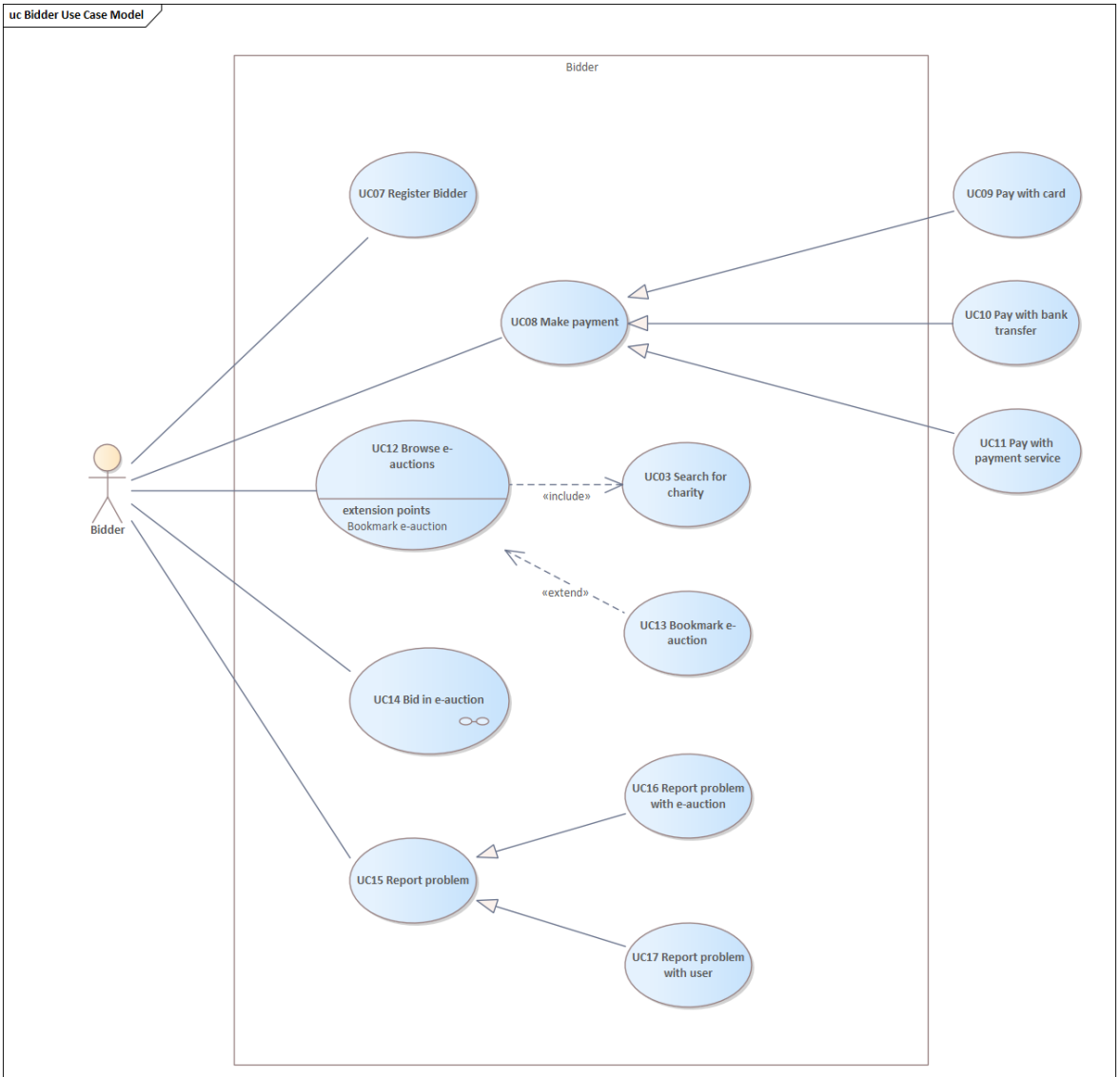
4. System opens the creation window
5. Operator inserts basic information (name, address, purpose of charity, …) as well as bank information to automatically send money from e-auction to charity
6. System checks whether the inserted information is correct

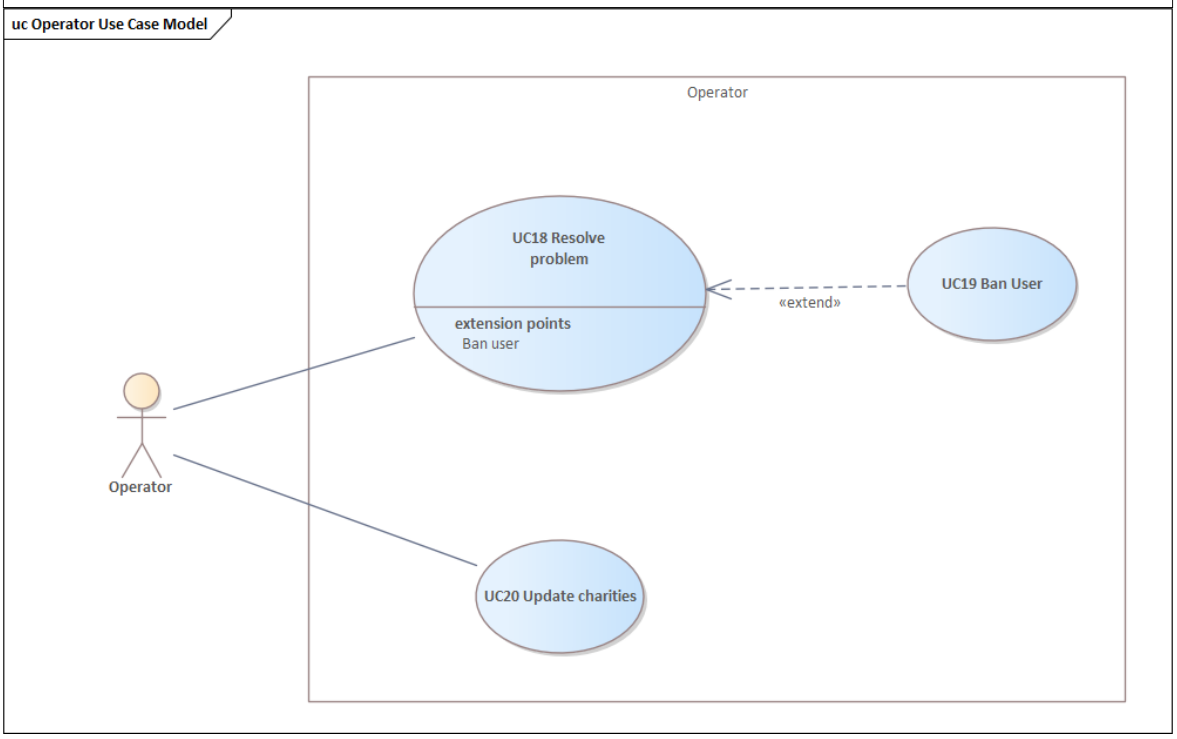
    Exception - Some information is incorrect:

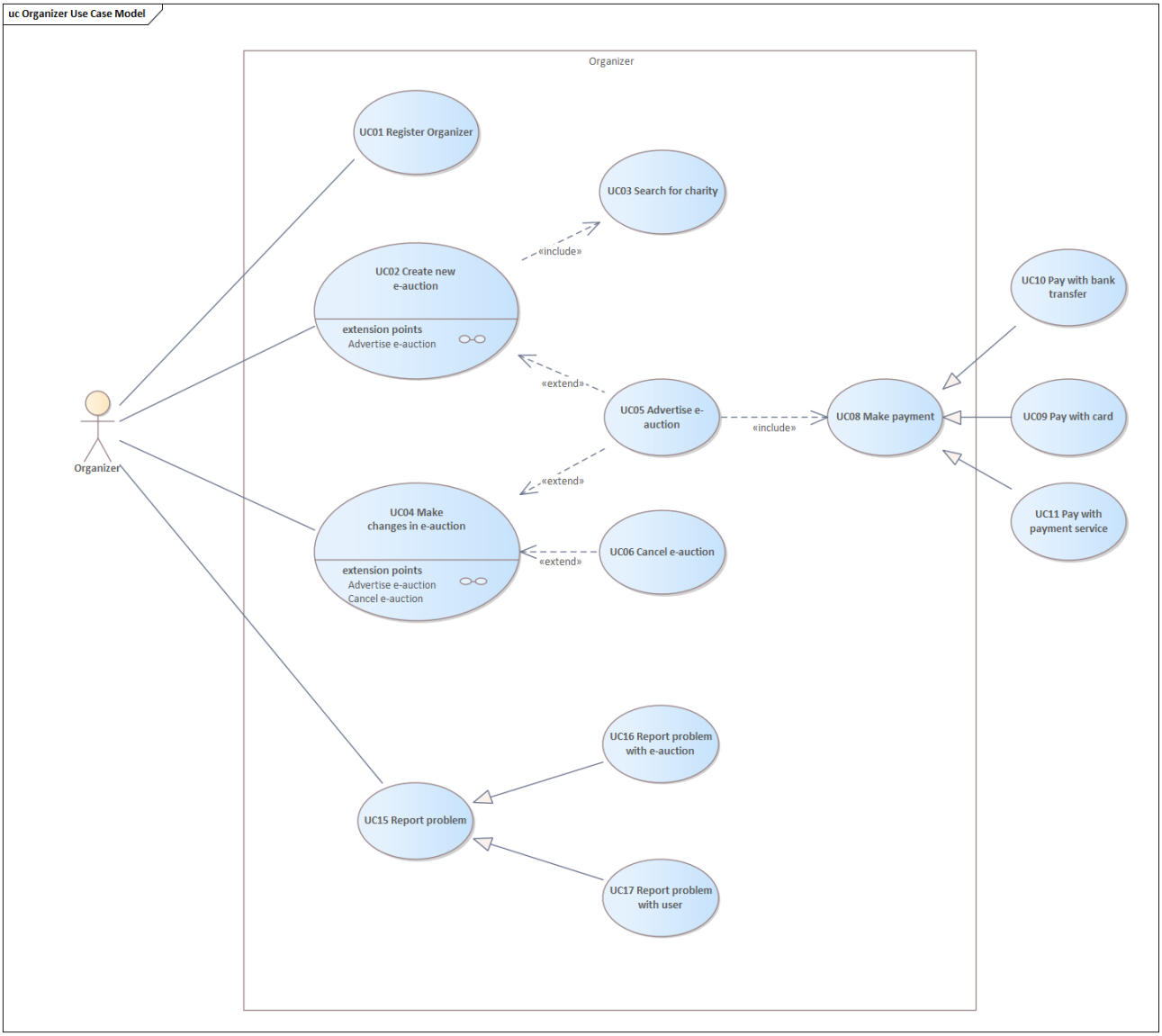
    6.1 System produces warning that some inserted information is incorrect
    6.2 Operator is given option to make changes
    6.3 *Return to step 5*

7. System saves the charity to database and shows confirmation message

In my project, I use Jacobson notation of Use Case model. Sequence and Activity diagrams are under respective Use Cases as a child diagrams and every Use Case described in this document has in Enterprise Architect included description in the Notes tab.


uc Bidder Use Case Model
Bidder
UC07 Register Bidder
UC09 Pay with card
UC08 Make payment
UC10 Pay with bank transfer
UC11 Pay with payment service
UC12 Browse e-auctions
extension points
Bookmark e-auction
«include»
UC03 Search for charity
«extend»
UC13 Bookmark e-auction
Bidder
UC14 Bid in e-auction
UC16 Report problem with e-auction
UC15 Report problem
UC17 Report problem with user


uc Operator Use Case Model
Operator
UC18 Resolve problem
extension points
Ban user
«extend»
UC19 Ban User
Operator
UC20 Update charities


uc Organizer Use Case Model
Organizer
UC01 Register Organizer
UC03 Search for charity
«include»
UC02 Create new e-auction
extension points
Advertise e-auction
UC10 Pay with bank transfer
«extend»
UC05 Advertise e-auction
«include»
UC08 Make payment
UC09 Pay with card
Organizer
«extend»
UC11 Pay with payment service
UC04 Make changes in e-auction
extension points
Advertise e-auction
Cancel e-auction
UC06 Cancel e-auction
«extend»
UC16 Report problem with e-auction
UC15 Report problem
UC17 Report problem with user

Sequence and Activity Diagrams

Sequence and Activity Diagrams model the same Use Cases so they can be compared if the functionality is modeled in the same way.

UC02: Create new e-auction

Use Case starts with Organizer selecting the option to create new e-auction. This creates new Auction entity, an object, that saves all information about an e-auction. First, user has to insert basic information - name, description and start and end of Auction. Organizer then has to select a Charity from list of charities saved in database. User may search the list to find the required Charity more easily. Auction has to support one real Charity, so it won't allow Organizer to continue unless one Charity is selected. After saving the information to Auction, it is time to create new Biddings. Each Auction has to contain at least one Bidding, so the first one has to be made, and then the Organizer gets to choose if he wants to create another, or confirm the list of already saved Biddings. The last step is validation - whether the start date is before the end date, if the start date isn't before the day of creation, the Biddings are valid and there isn't a gap between two Biddings, …

After successful validation, the Auction with Biddings is saved to database and scheduled for automatic start. Of course, Organizer can make subsequent changes to Auction through Use Case *UC04: Make changes in e-auction*.

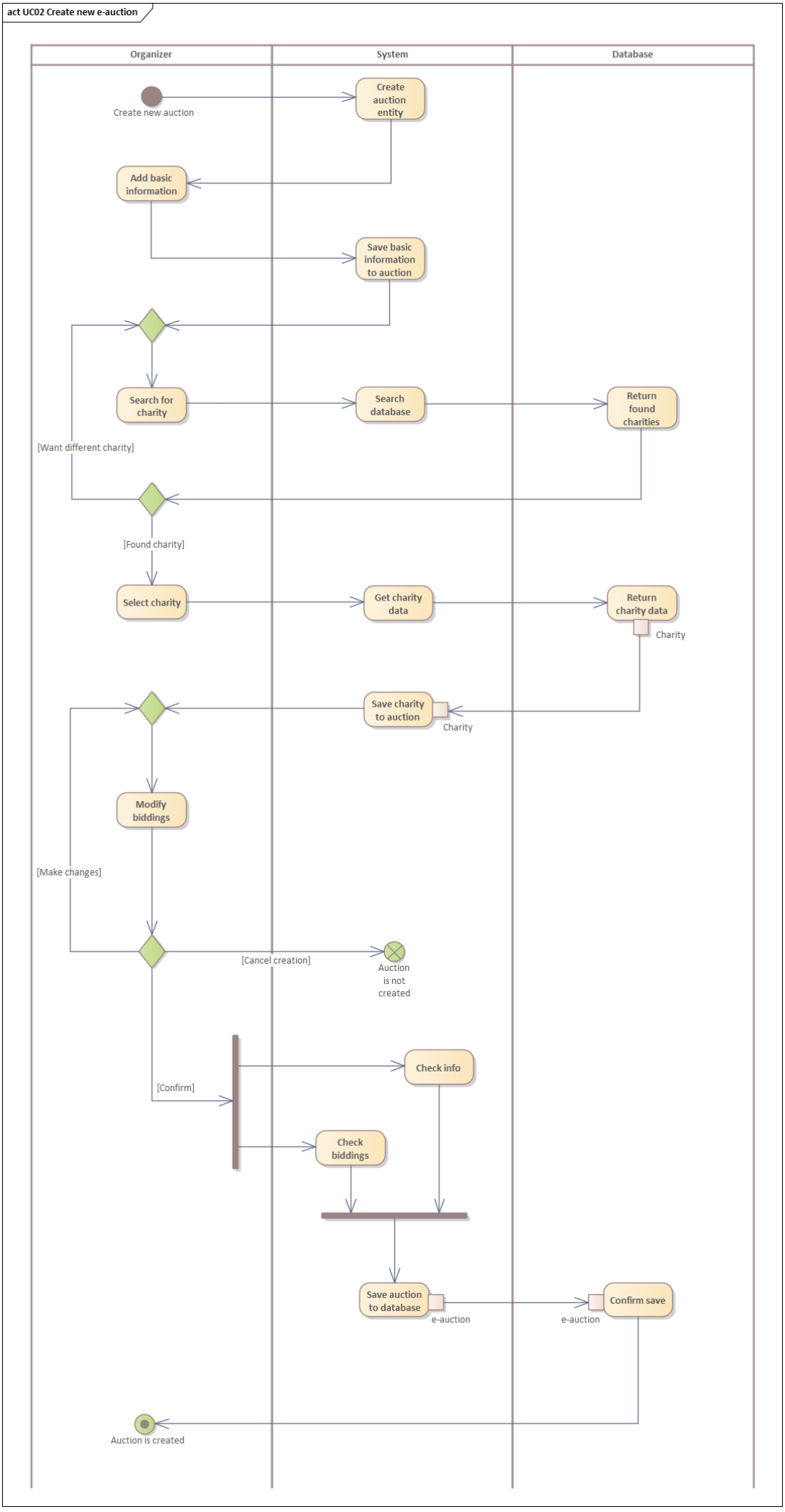
UC04: Make changes in e-auction

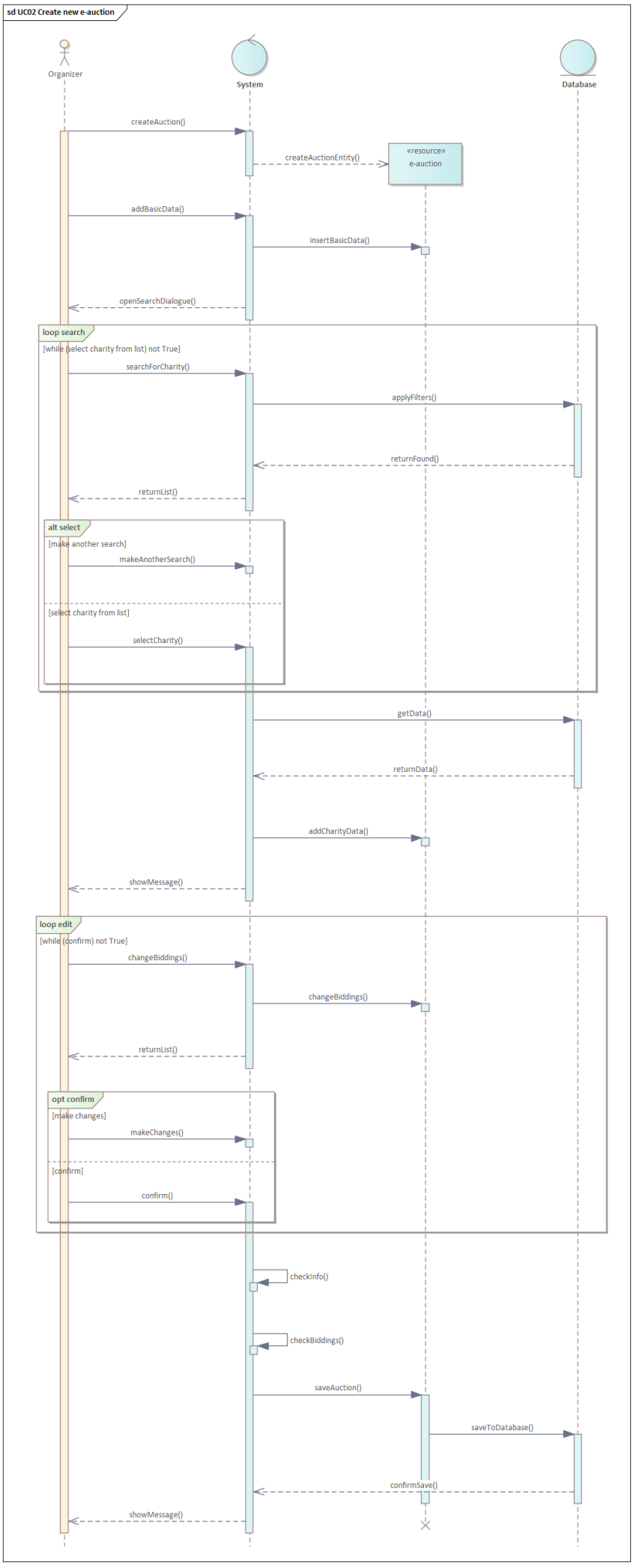
This Use Case has functionality similar to *UC02: Create new e-auction*, but it requires the Auction to already exist. The Auction is loaded from database and its information is displayed to Organizer, who can choose, whether he wants to make changes in Biddings or information. After committing changes, Organizer is again given to choice to make another changes or confirm. Just as with *UC02: Create new e-auction*, the last step is validation which check whether all changes are valid, and if they are, the information is updated in the database.

UC14: Bid in e-auction

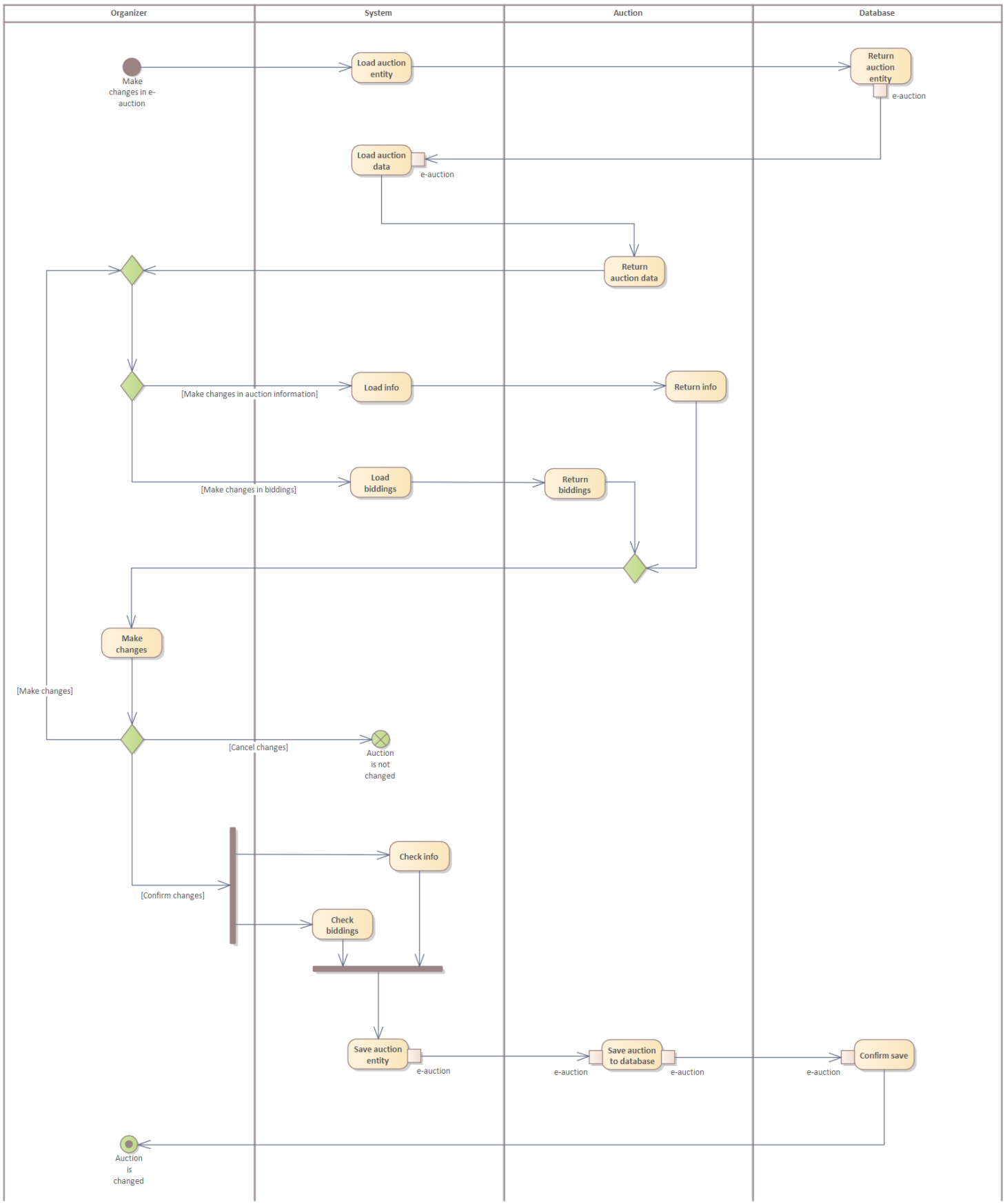
Bidding requires more information from database. First, Bidder selects an Auction from list of trending auctions in dashboard, or through search, and system displays all information about Auction as well as list of biddings. Bidder can select one of the active Biddings and after system displays all Bidding information (name, description, starting or highest bid, depending on whether someone has previously made a bid, and highest bidder), they can select an option to make a new bid. If the new bid is correct (higher than previous one), this bid is saved as the new highest along with the information about Bidder as the new highest bidder, and changes are uploaded to database.

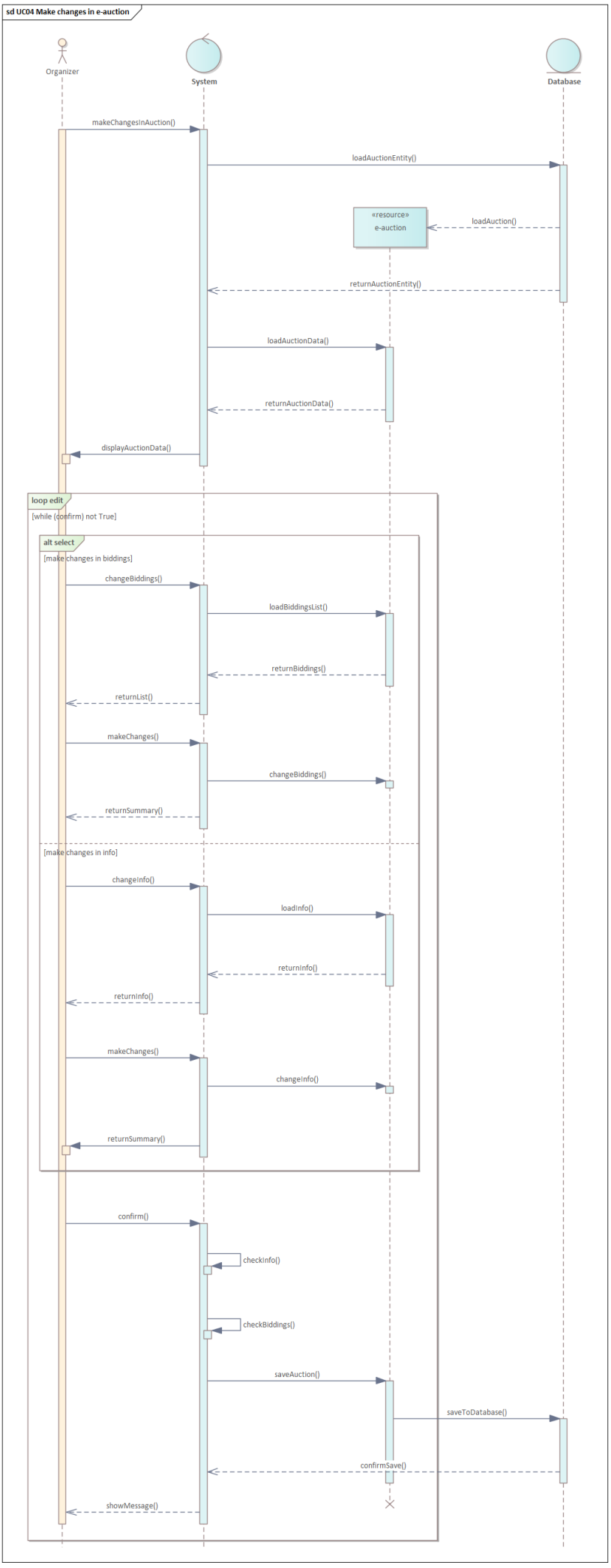
After the end of Bidding, the saved highest bidder is used in generating a list of winners for Organizer to send out prizes.

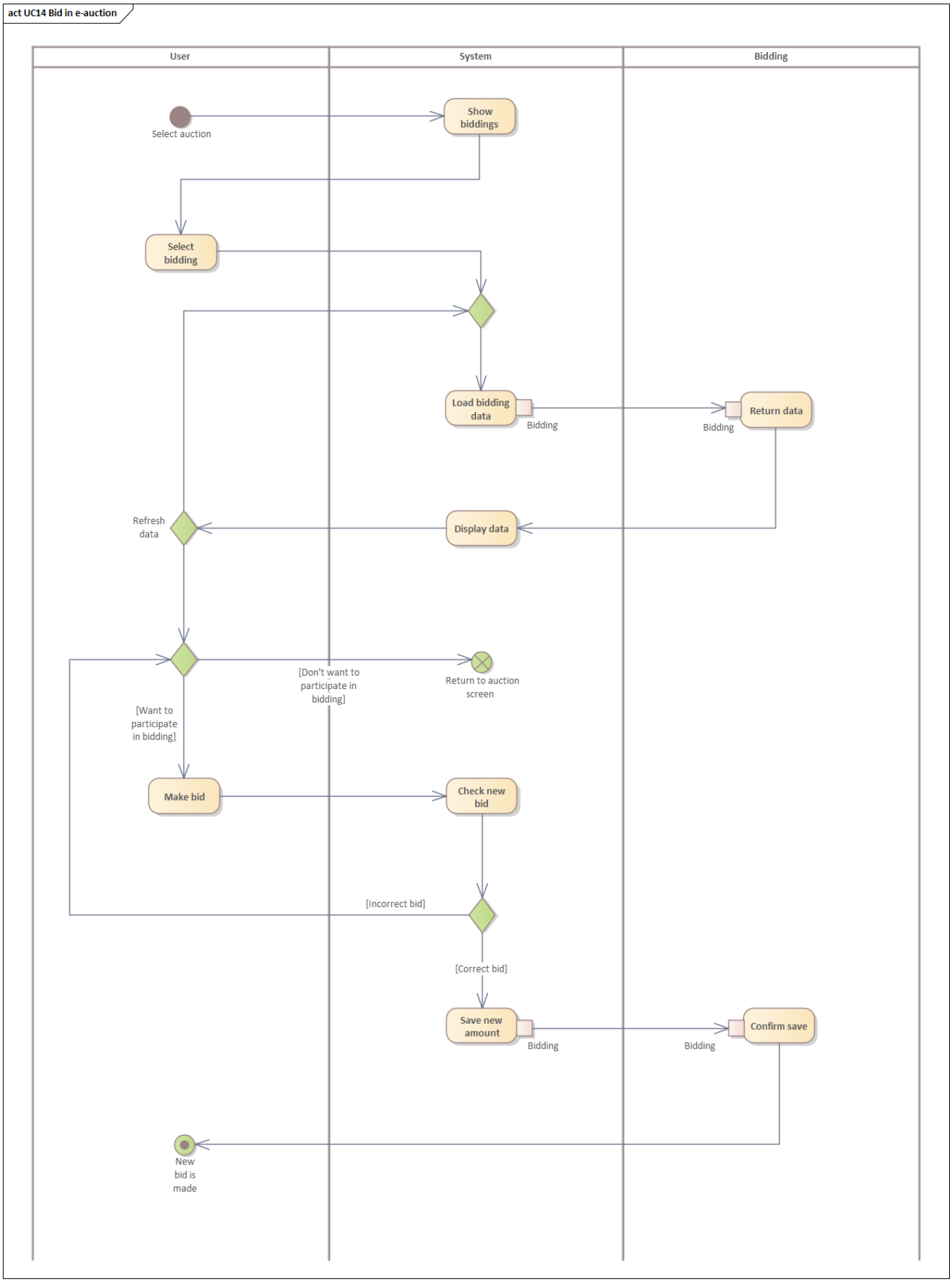

act UC02 Create new e-auction
Organizer
System
Database
Create new auction
Create auction entity
Add basic information
Save basic information to auction
Search for charity
Search database
Return found charities
[Want different charity]
[Found charity]
Select charity
Get charity data
Return charity data
Charity
Save charity to auction
Charity
Modify biddings
[Make changes]
[Cancel creation]
Auction is not created
[Confirm]
Check info
Check biddings
Save auction to database
e-auction
Confirm save
e-auction
Auction is created

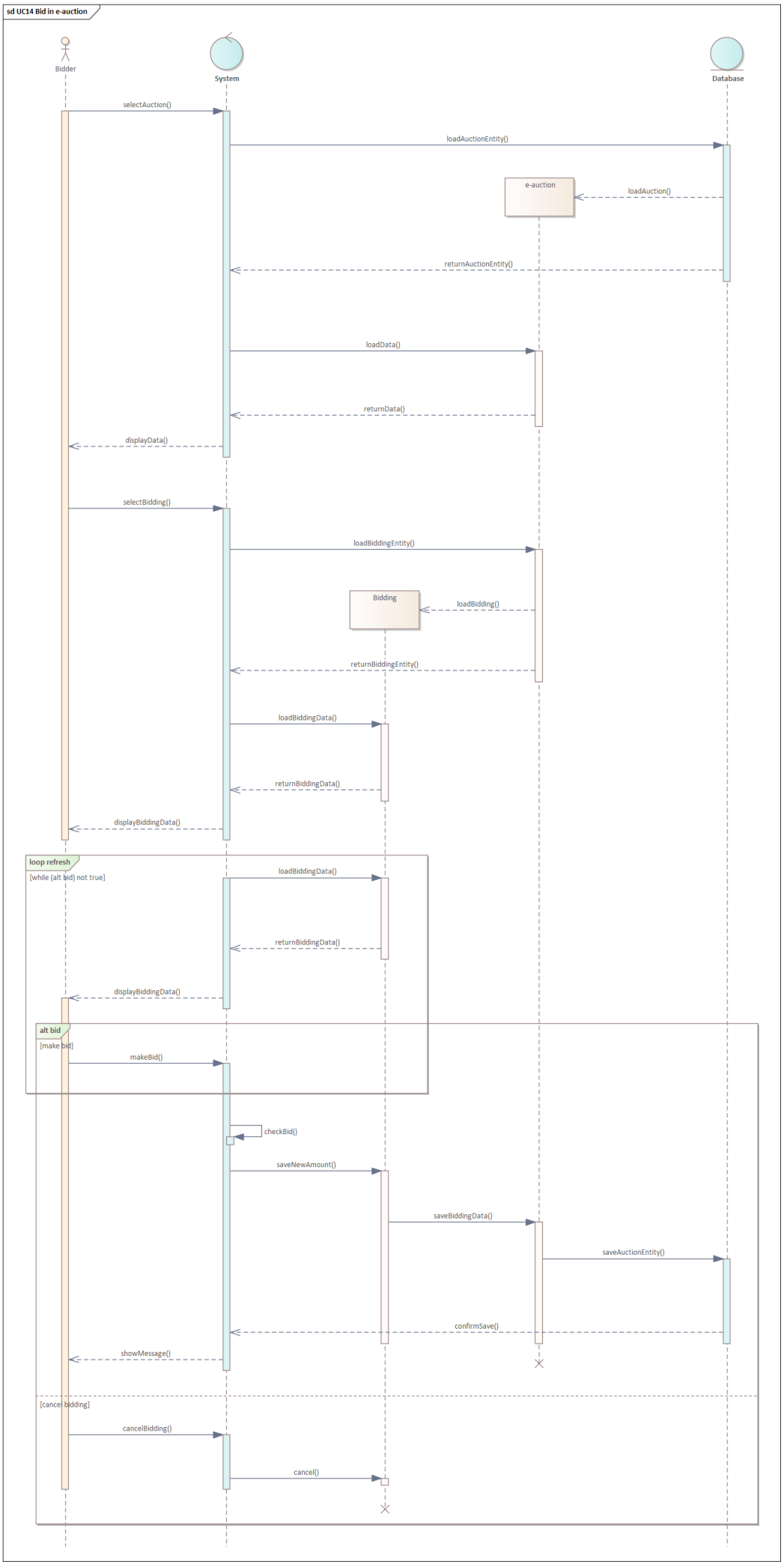

sd UC02 Create new e-auction
Organizer
System
Database
createAuction()
createAuctionEntity()
«resource»
e-auction
addBasicData()
insertBasicData()
openSearchDialogue()
loop search
[while (select charity from list) not True]
searchForCharity()
applyFilters()
returnFound()
returnList()
alt select
[make another search]
makeAnotherSearch()
[select charity from list]
selectCharity()
getData()
returnData()
addCharityData()
showMessage()
loop edit
[while (confirm) not True]
changeBiddings()
changeBiddings()
returnList()
opt confirm
[make changes]
makeChanges()
[confirm]
confirm()
checkInfo()
checkBiddings()
saveAuction()
saveToDatabase()
confirmSave()
showMessage()

act UC04 Make changes in e-auction




sd UC04 Make changes in e-auction
Organizer
System
Database
makeChangesInAuction()
loadAuctionEntity()
«resource»
e-auction
loadAuction()
returnAuctionEntity()
loadAuctionData()
returnAuctionData()
displayAuctionData()
loop edit
[while (confirm) not True]
alt select
[make changes in biddings]
changeBiddings()
loadBiddingsList()
returnBiddings()
returnList()
makeChanges()
changeBiddings()
returnSummary()
[make changes in info]
changeInfo()
loadInfo()
returnInfo()
returnInfo()
makeChanges()
changeInfo()
returnSummary()
confirm()
checkInfo()
checkBiddings()
saveAuction()
saveToDatabase()
confirmSave()
showMessage()


act UC14 Bid in e-auction
User
System
Bidding
Select auction
Show biddings
Select bidding
Load bidding data
Bidding
Return data
Bidding
Refresh data
Display data
[Don't want to participate in bidding]
Return to auction screen
[Want to participate in bidding]
Make bid
Check new bid
[Incorrect bid]
[Correct bid]
Save new amount
Bidding
Confirm save
Bidding
New bid is made


sd UC14 Bid in e-auction
Bidder
System
Database
selectAuction()
loadAuctionEntity()
e-auction
loadAuction()
returnAuctionEntity()
loadData()
returnData()
displayData()
selectBidding()
loadBiddingEntity()
Bidding
loadBidding()
returnBiddingEntity()
loadBiddingData()
returnBiddingData()
displayBiddingData()
loop refresh
[while (alt bid) not true]
loadBiddingData()
returnBiddingData()
displayBiddingData()
alt bid
[make bid]
makeBid()
checkBid()
saveNewAmount()
saveBiddingData()
saveAuctionEntity()
confirmSave()
showMessage()
[cancel bidding]
cancelBidding()
cancel()

Class Model

Class diagrams describe structure of the system. Important part of class diagram are constraints which define the boundaries of classes. The constraints are defined in every class under the tab “Constraints” in Enterprise Architect model.

Database

Since all information about system requires to be saved and archived, database is a vital part of the system. Information is saved into 8 interconnected tables:

Users saves information about each registered user in system, their full name, username, password, address, personal or company identification number (OP or IČO) and booleans whether the user is Bidder, Operator or Organizer.

Charities keeps information about every registered charity along with bank number, so the raised amount of money can be automatically sent to charity.

Auctions contains information about every auction, its name, description, start and end dates and foreign keys pointing to Charities to determine which charity is being supported by this auction, and Users to save the Organizer.

Biddings retain information about every bidding in system. Each Bidding has a foreign key to Auctions table to save which auction this bidding belongs to and also saves the id of highest bidder from Users table.

Bookmarks is a table for Bidders since is saves bookmarked auctions.

Reports keeps track of all issued reports in the system, whether and by whom they were resolved and what or who was the subject of the report.

Categories are selected from when issuing a report. Some categories require an Operator to resolve the issue, some problems may be resolved by Organizer, like typo or some other minor mistake in description of auction or bidding.

Services saves information about payment services - PayPal, Apple Pay, Google Pay, …

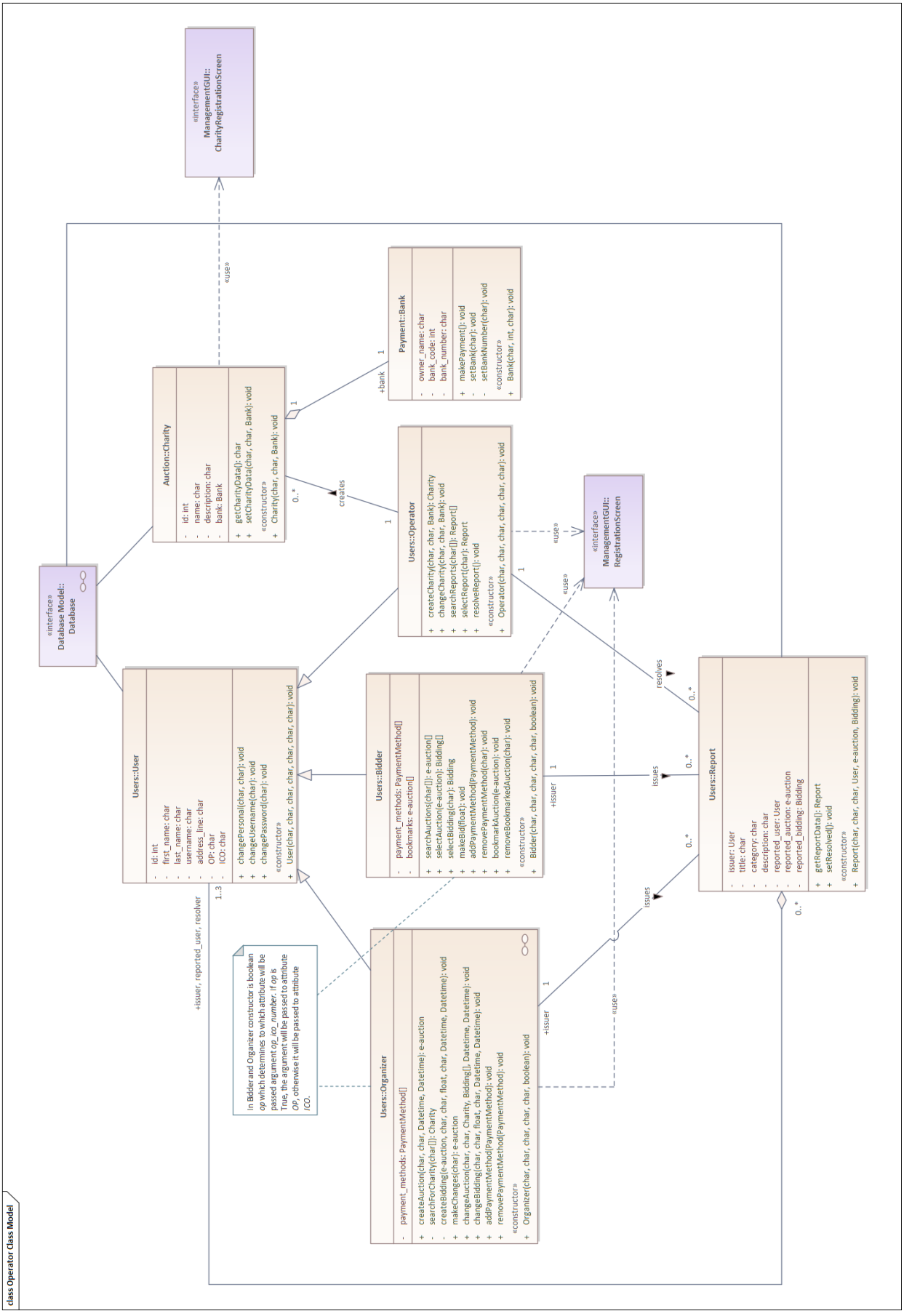
Operator

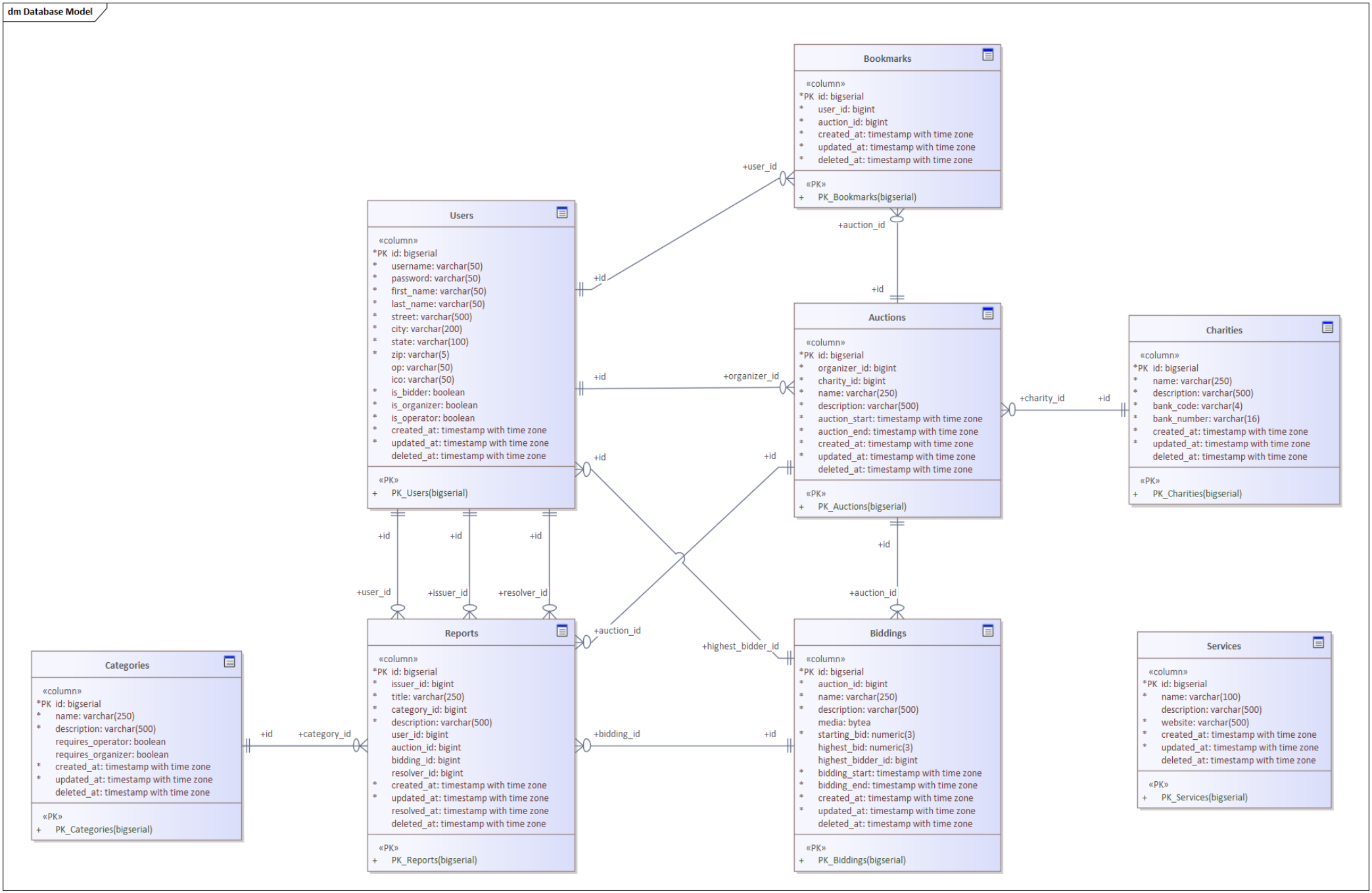
Operator Class diagram shows important hierarchy of users in system. A parent class User retains all basic information, like name, username, address, which are inherited by child classes, each with its own additional variables and functions.

Operator can create new Charities, which are subsequently saved to database, or modify existing. They can also view issued reports and resolve them:

- User report, where one user reports another for bad behaviour, attempt to scam other people, … These reports may end with Operator banning reported user, which means that his account will be suspended and he will be prevented from registering again.

• Auction report, where user reports an auction because of a mistake in description, which can be resolved by Organizer, but also more serious, like suspicion of fake auction in attempt to steal money from other users.
• Bidding report, which deals with the same problems as Auction report, but with Biddings.

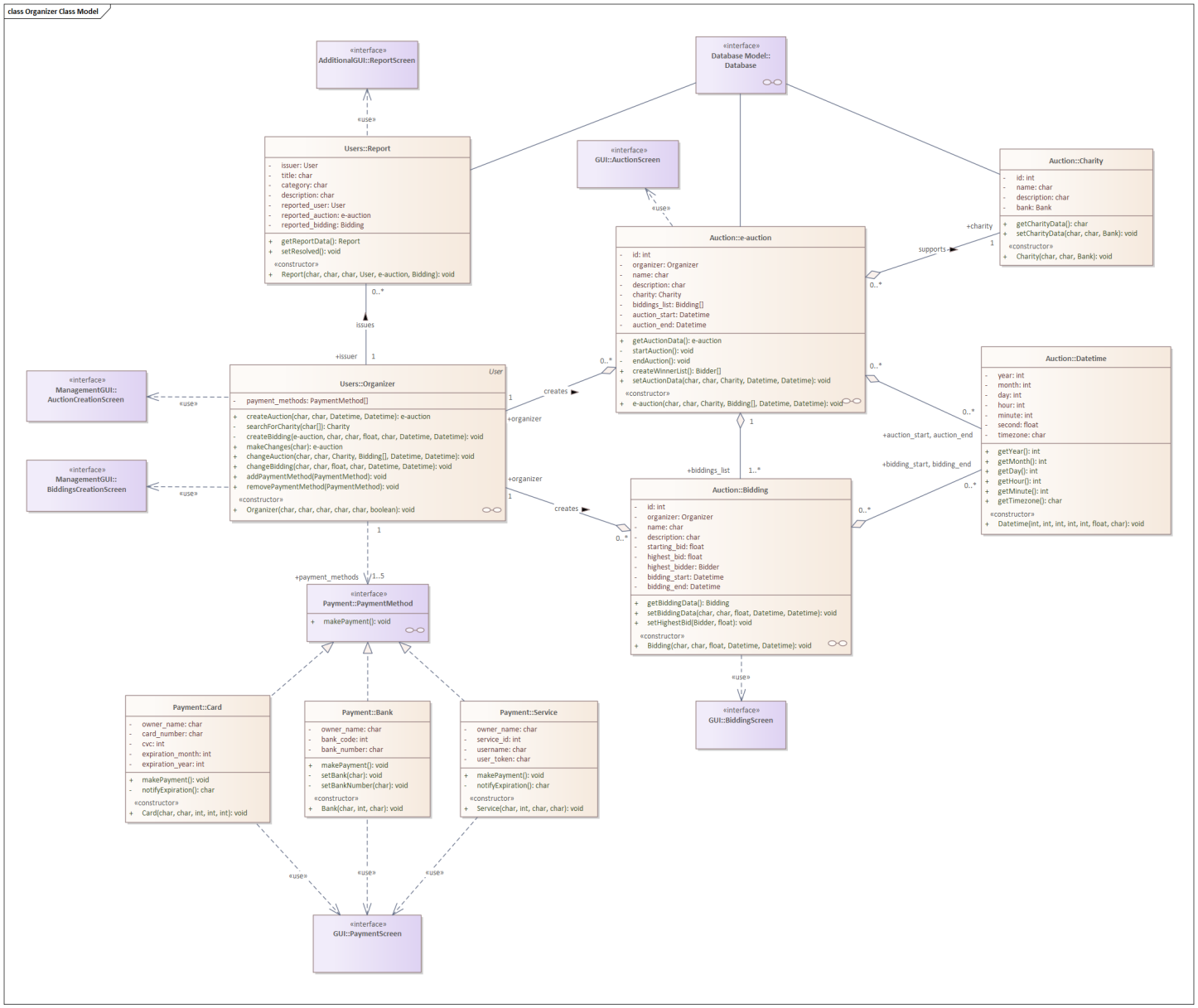



dm Database Model
Bookmarks
«column»
*PK id: bigserial
* user_id: bigint
* auction_id: bigint
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
* deleted_at: timestamp with time zone
«PK»
+ PK_Bookmarks(bigserial)
Users
«column»
*PK id: bigserial
* username: varchar(50)
* password: varchar(50)
* first_name: varchar(50)
* last_name: varchar(50)
* street: varchar(500)
* city: varchar(200)
* state: varchar(100)
* zip: varchar(5)
op: varchar(50)
ico: varchar(50)
* is_bidder: boolean
* is_organizer: boolean
* is_operator: boolean
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Users(bigserial)
Auctions
«column»
*PK id: bigserial
* organizer_id: bigint
* charity_id: bigint
* name: varchar(250)
* description: varchar(500)
* auction_start: timestamp with time zone
* auction_end: timestamp with time zone
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Auctions(bigserial)
Charities
«column»
*PK id: bigserial
* name: varchar(250)
* description: varchar(500)
* bank_code: varchar(4)
* bank_number: varchar(16)
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Charities(bigserial)
Categories
«column»
*PK id: bigserial
* name: varchar(250)
* description: varchar(500)
requires_operator: boolean
requires_organizer: boolean
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Categories(bigserial)
Reports
«column»
*PK id: bigserial
* issuer_id: bigint
* title: varchar(250)
* category_id: bigint
* description: varchar(500)
user_id: bigint
auction_id: bigint
bidding_id: bigint
resolver_id: bigint
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
resolved_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Reports(bigserial)
Biddings
«column»
*PK id: bigserial
* auction_id: bigint
* name: varchar(250)
* description: varchar(500)
media: bytea
* starting_bid: numeric(3)
highest_bid: numeric(3)
highest_bidder_id: bigint
* bidding_start: timestamp with time zone
* bidding_end: timestamp with time zone
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Biddings(bigserial)
Services
«column»
*PK id: bigserial
* name: varchar(100)
description: varchar(500)
* website: varchar(500)
* created_at: timestamp with time zone
* updated_at: timestamp with time zone
deleted_at: timestamp with time zone
«PK»
+ PK_Services(bigserial)
+user_id
+id
+auction_id
+id
+id
+organizer_id
+charity_id
+id
+id
+id
+id
+id
+id
+user_id
+issuer_id
+resolver_id
+auction_id
+auction_id
+highest_bidder_id
+id
+category_id
+bidding_id
+id

Organizer

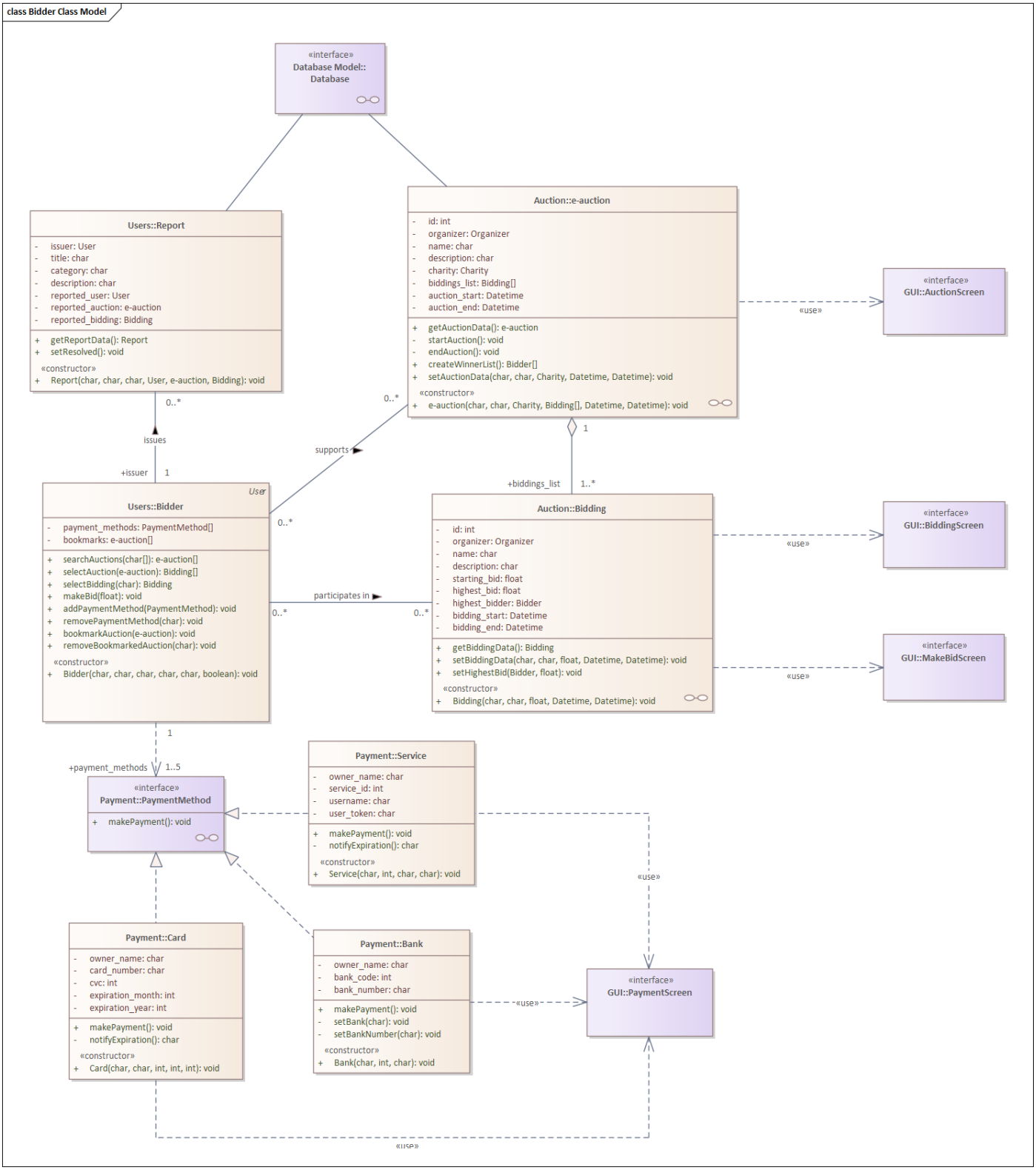
Organizer can create auctions, biddings, issue reports and make payments. Auctions and biddings need to save the start and end, which is represented by Datetime class. As was previously mentioned, auction also contains a list of biddings and it requires at least one existing bidding for an auction to be valid. Auction also contains information about information about charity.

If Organizer wants to advertise auction, he needs to have an existing payment method, to be able to pay for the advertisement. Payment methods may be of three kinds: bank, card and service. Bank requires the name of the owner of the account, account number and bank code. Card needs card number, cardholder's name, expiration date and cvc, which is a security code at the back of the card. Service performs payment through third-party payment gateway like PayPal, Apple Pay and many more. Actual payment information is saved on the payment service's side, but vendor obtains payment token, through which the they can obtain the money from the payment.



Bidder

Bidder operates with the same classes that were mentioned before, but in different ways. While Organizer can create and modify auctions and biddings, Bidder can participate in them, make new bids and win prizes. That, of course, requires an existing payment method, again it can be bank account, card or payment service. Bidder can also issue reports and they probably issue more reports than Organizers.

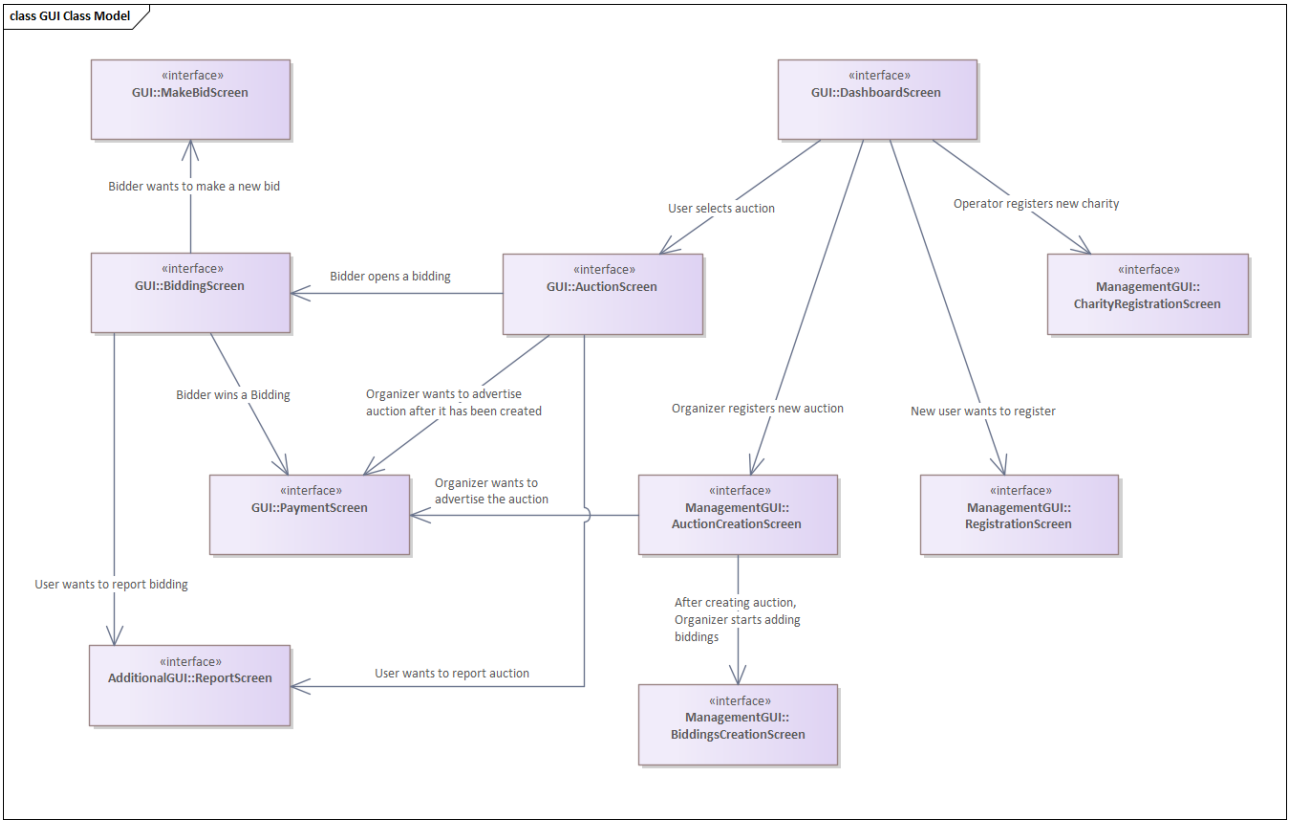


Graphical Interface

GUI is represented through a collections of screens, each displaying different information and involved with different part of system. The main screen is Dashboard, which contains sign in and sign up dialogues as well as list of trending (or advertised) auctions. Operator may invoke CharityRegistrationScreen for creating new charity. Organizer can invoke AuctionCreationScreen which later calls BiddingCreationScreen, or PaymentScreen, depending on whether Organizer creates a new auction with no biddings, or modifies existing one and wants to advertise it.

Bidder can open AuctionScreen and after selecting a bidding, a BiddingScreen is produced. When Bidder decides to make a new bid, MakeBidScreen dialogue is generated. Bidder also needs to pay for prize won, so they can also invoke PaymentScreen.

Bidder and Organizer can also issue a report which creates new dialogue window. ReportScreen is called from BiddingScreen or AuctionScreen, depending on the nature of the report.



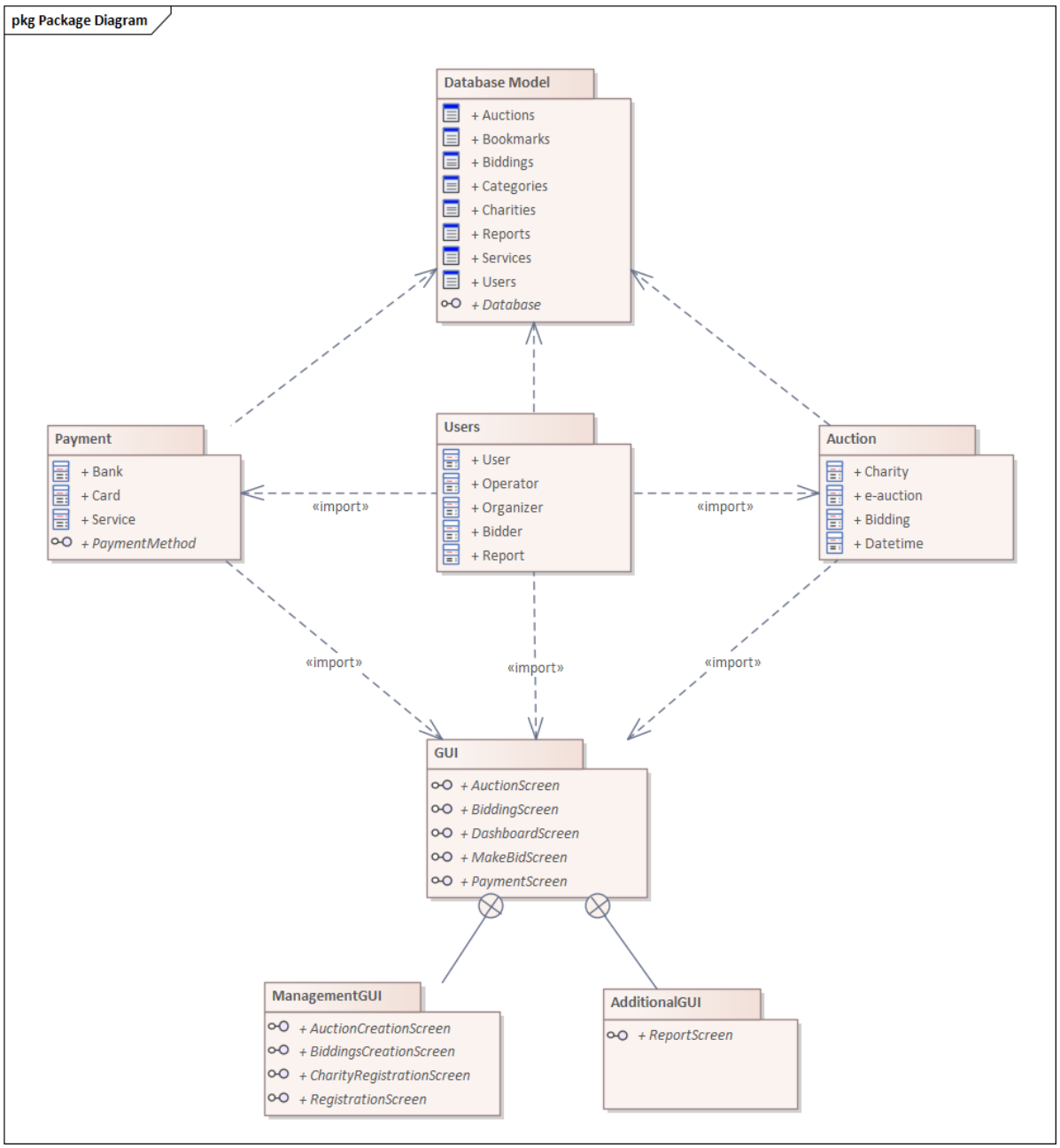
Packages

To make the system as organised as possible, classes are divided into packages. Database, of course, is whole in one package along with the database interface. Users are in separate package with Report class which is directly dependent on users, since they are the ones that issue them.

Another package is Auction which contains classes vital to correct functionality of auctions: e-auction which aggregates all auction information,

Bidding keeping information about one bidding, Charity which is supported by auction and Datetime which provides simple way to work with start and end dates and times of auctions and biddings.

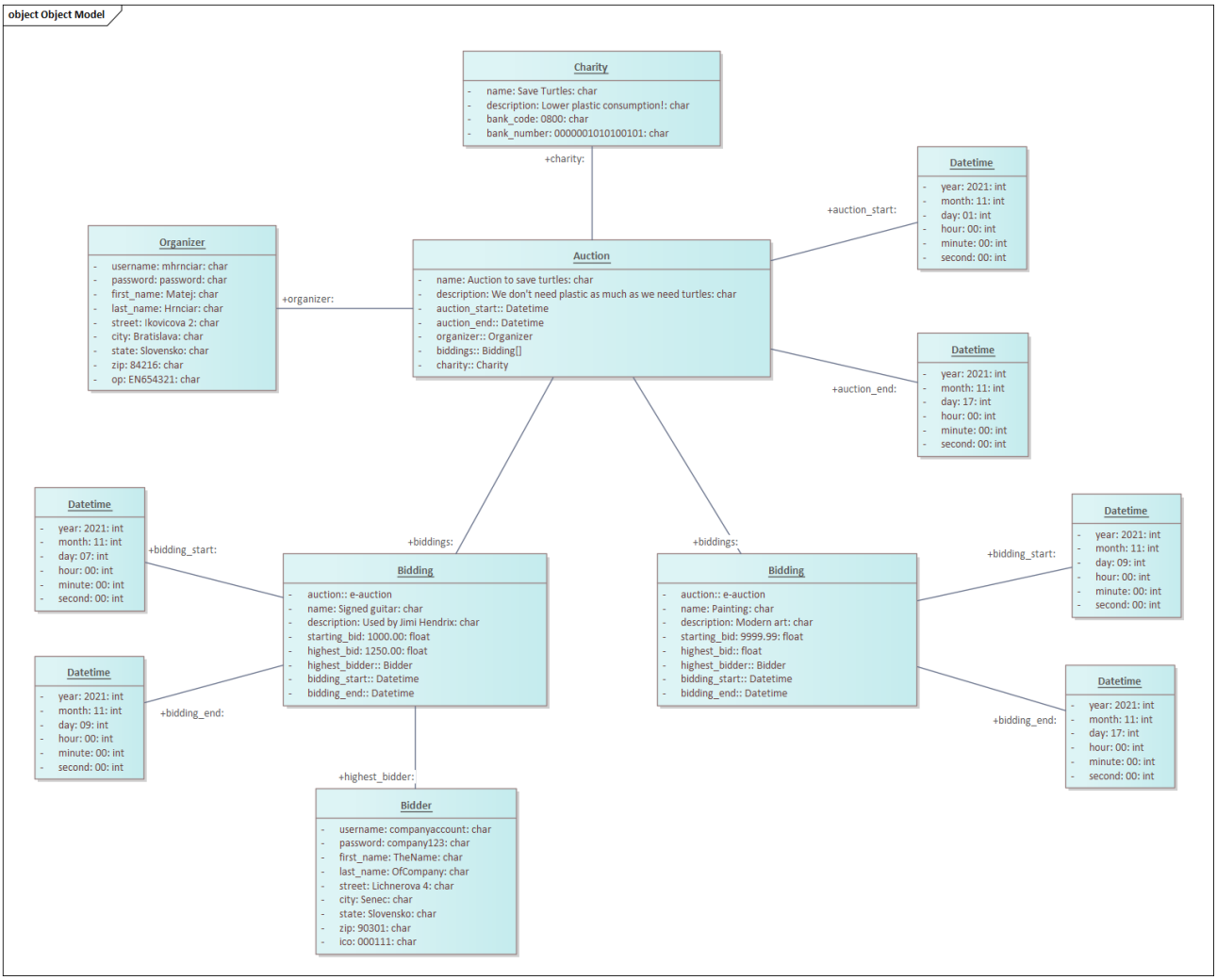
Another package is Payment which encompasses all payment methods and the last packages are for graphical interface. GUI package contains main screens, dashboard, auction, bidding, payment screens and dialogue to make a new bid. ManagementGUI aggregates all screens involved with creation and modification of system objects. AdditionalGUI provides additional functionality which isn't as vital to the correct functionality of the system, but may help with making the system better and safer.



Object Model

Following object model displays possible configuration of basic objects in system. At the top is a charity with saved information, which is supported by auction below it. Auction has its Organizer on the left and two datetimes representing start and end of auction. Auction also contains two biddings, both with datetimes of start and end respective bidding and starting bid. Bidding on the left also has highest bid and highest bidder, which is displayed below said bidding. The other bidding hadn't started yet or nobody made a bid, so there is only starting bid saved with no highest bidder.

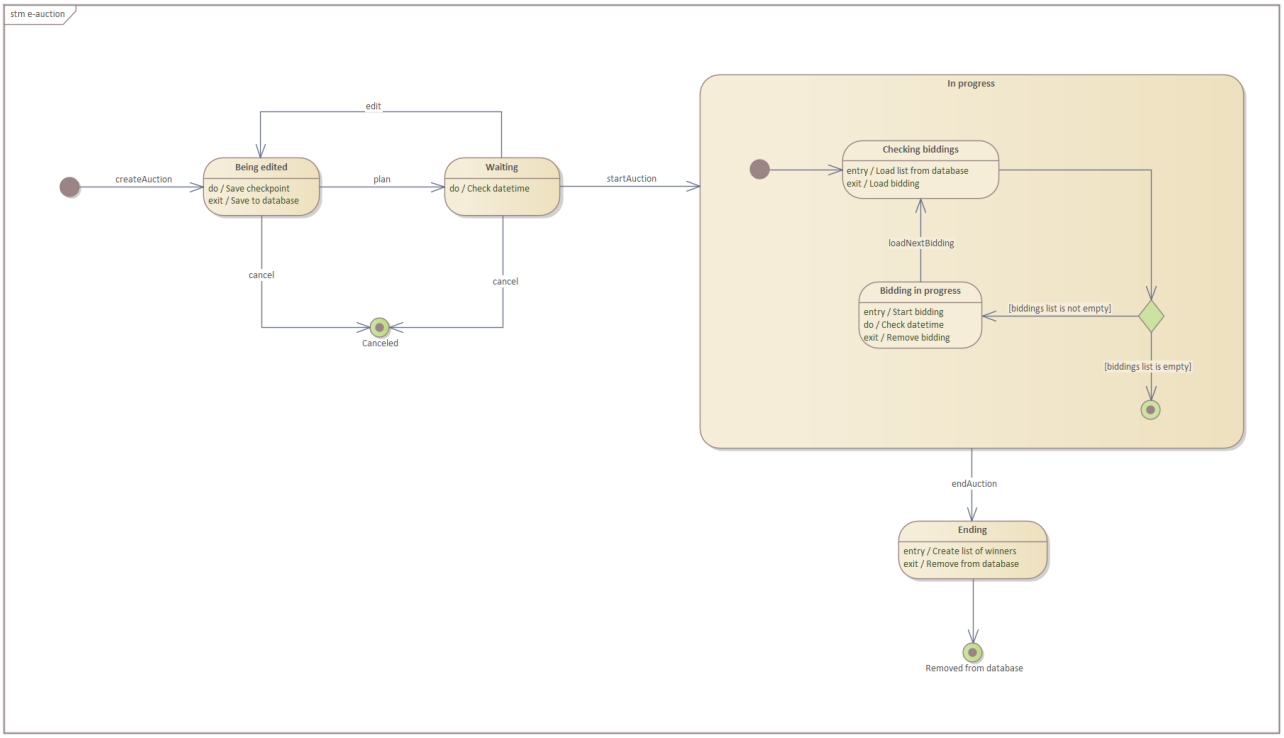
In case someone would make a new bid, the bidder object would be attached to bidding (in case on the right), or replace previously attached bidder object (on the left).



StateMachine Diagrams

Auction

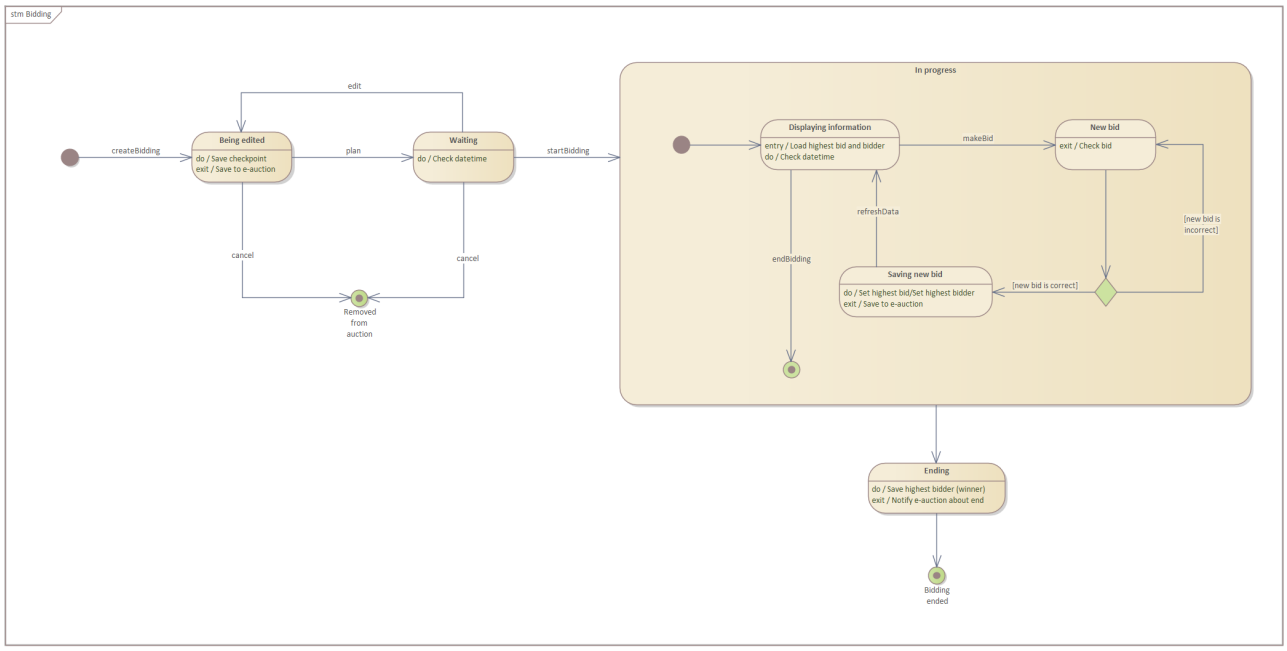
The Auction lifecycle begins with Organizer creating new Auction entity. After editing and saving, Auction waits until the start, during which the Auction may be modified or cancelled. After the start of Auction, system checks if there are biddings that weren't active yet, starts them on the datetime saved in bidding and subsequently ends them, also on the datetime saved in bidding. Auction ends when there are no more bidding in the list. After the end of the whole Auction, a list of winners is generated and the Auction is set as ended.



Bidding

Bidding is created in very same way as Auction - created, then waits until the start, but it can also be edited or removed. However, when the Bidding starts, it has different functionality.

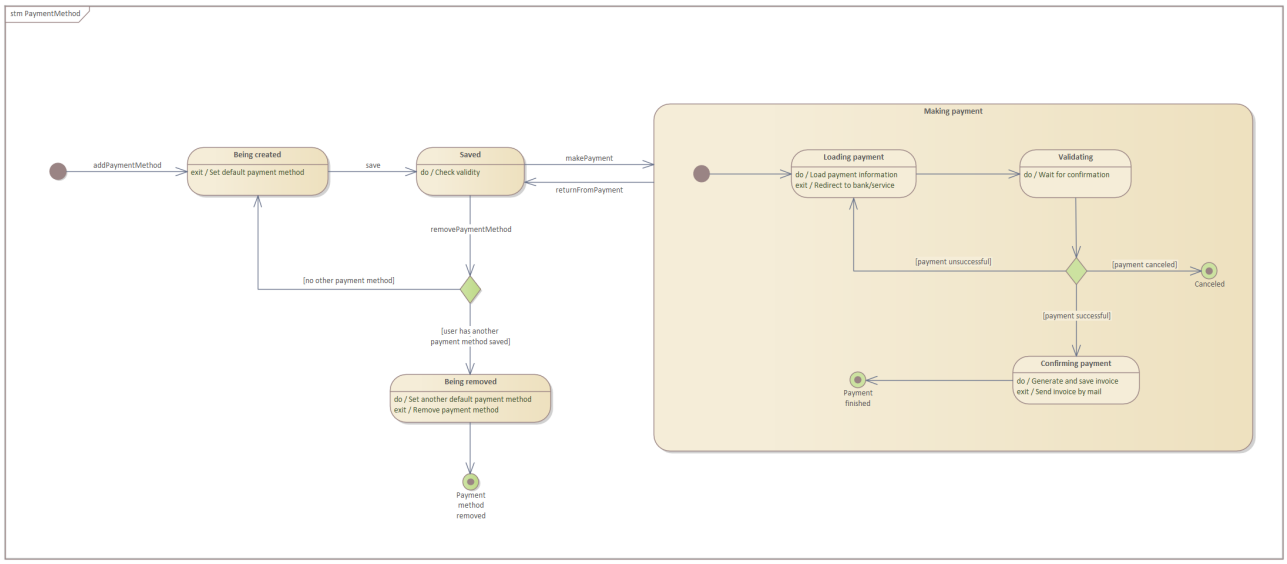
When the Bidding is started by Auction for the first time, it only displays basic information and starting bid. When Bidder makes a new bid, the bid has to be validated and if it's correct, it's saved as new highest bid along with the id of highest bidder. Bidding returns to state of displaying information until another Bidder makes a new bid, or the Bidding ends, when the id of the winner is saved, the Bidding is set as completed and Auction can load new Bidding from list.

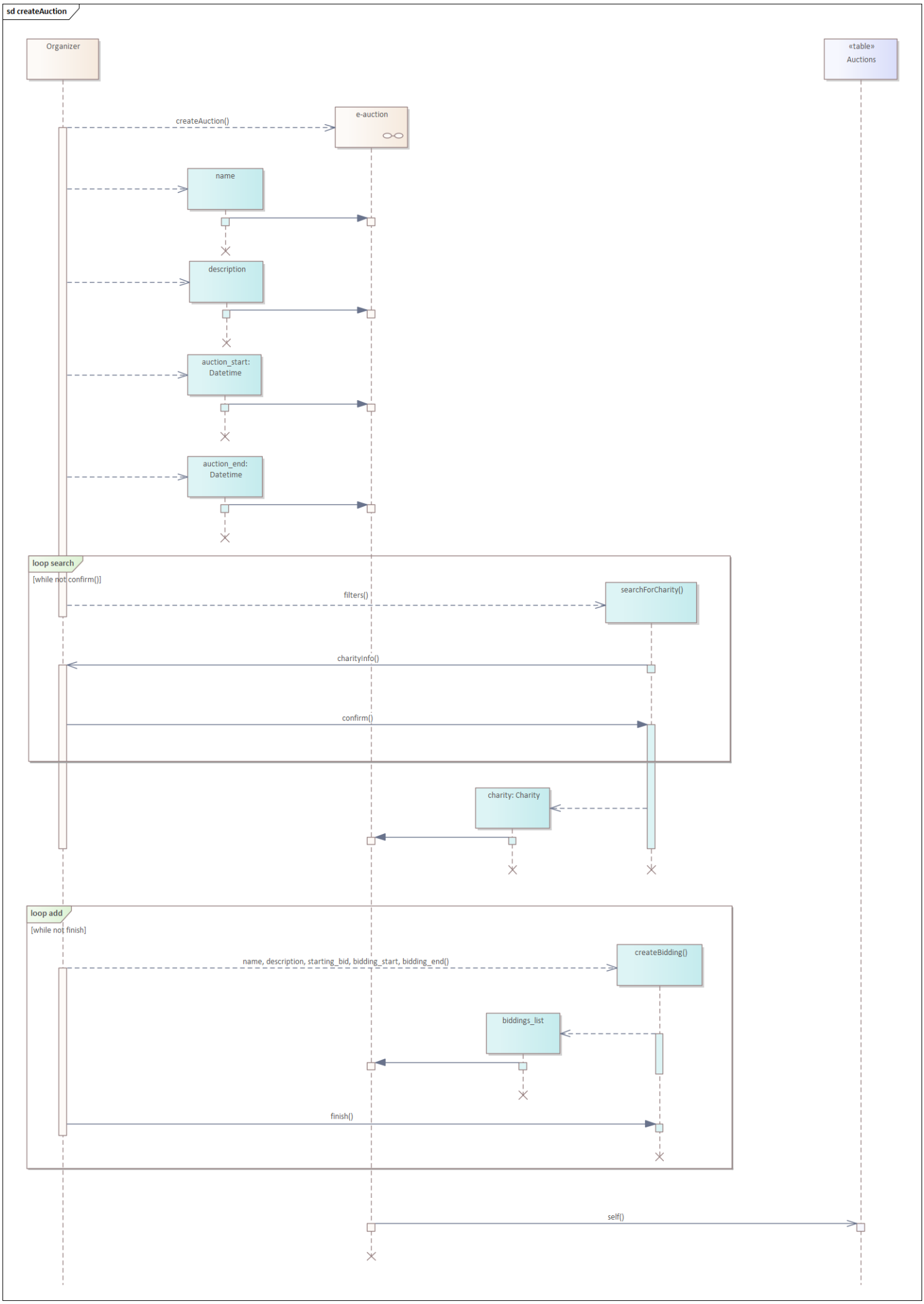


PaymentMethod

When Bidder or Organizer registers, they add a payment method to be able to pay for won bidding or advertisement. Payment method is saved in the database waiting to be used, but it can also be removed, however, if it was the only payment method, a new one has to be created. If the payment method wasn't the only one, it can be easily removed.

When user wants to perform payment, the method is loaded from database and user is redirected to respective bank gateway or payment service. System then waits for confirmation message, but in reality, three things can happen: payment successful - payment is performed, money is moved and the payment method returns to waiting state, payment unsuccessful - payment is loaded again and user may try to pay again, payment cancelled - user cancels payment and payment methods returns to waiting state.

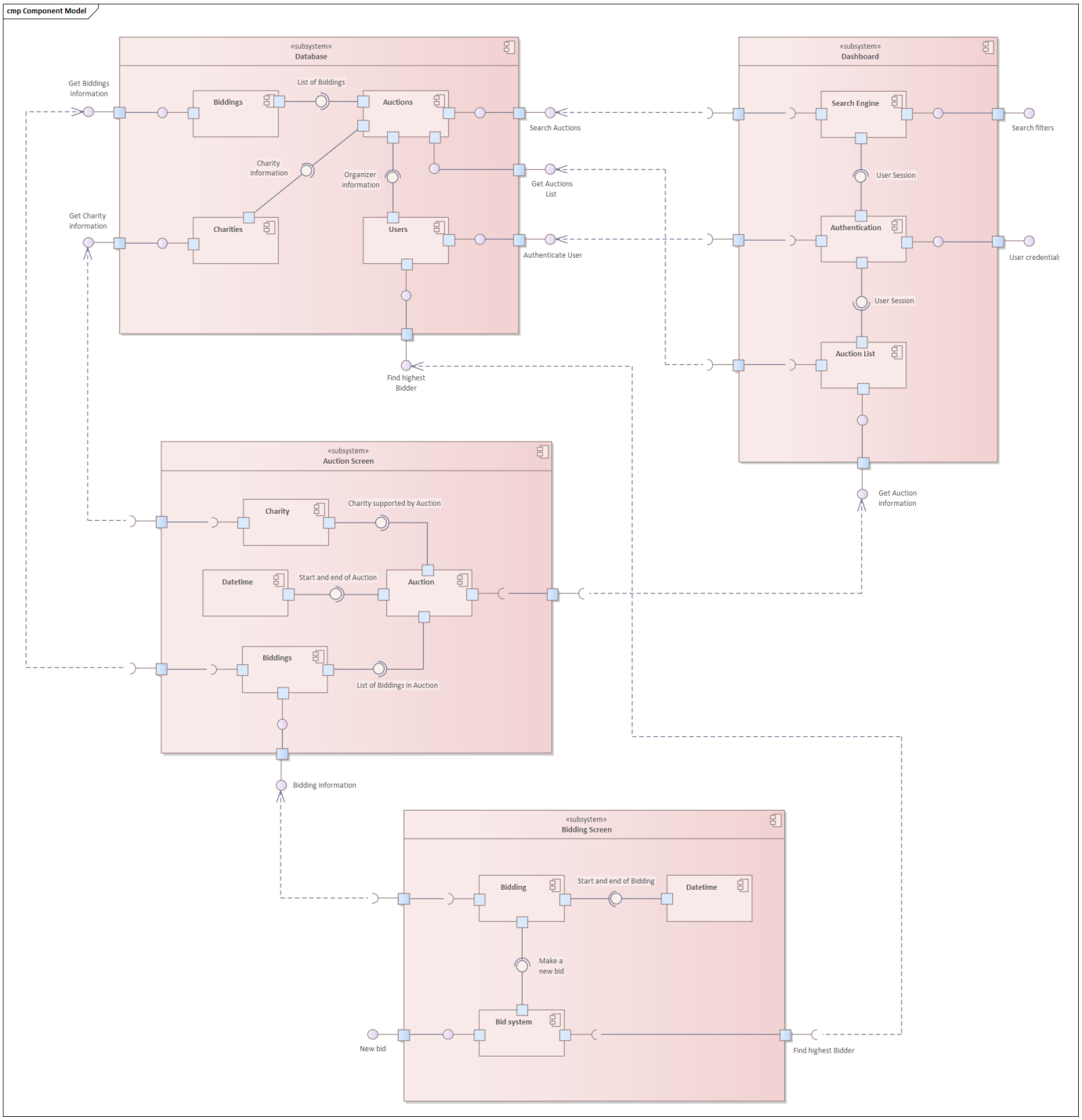



sd createAuction
Organizer
«table»
Auctions
createAuction()
e-auction
name
description
auction_start:
Datetime
auction_end:
Datetime
loop search
[while not confirm()]
filters()
searchForCharity()
charityInfo()
confirm()
charity: Charity
loop add
[while not finish]
name, description, starting_bid, bidding_start, bidding_end()
createBidding()
biddings_list
finish()
self()

Component Model

Component diagram contains four selected subsystems: Dashboard, Database, Auction Screen and Bidding Screen.

As was mentioned earlier, Dashboard is the first screen user sees. Dashboard subsystem queries database to obtain trending and advertised auctions, saved to auction list and displayed on screen. User can log in, meaning they insert user credentials which are validated against database and if the username and password match, user is logged in. Through user session, logged in



user can perform search, where they insert filters which are applied to database query and the result is again saved to auctions list and displayed on screen. When user selects an auction, Auction Screen subsystem is invoked.

Auction Screen depends on auction information from Dashboard. Remaining information is queried from database with the use of auction id. Separate information is saved to respective components which are connected to Auction. On this screen, user can view the basic information about auction, charity it supports and list of biddings. When one of the biddings is selected, system produces Bidding Screen subsystem.

Bidding Screen obtains almost all information from Auction Screen, but it needs to load the information about highest bidder from database. From Auction Screen is passed its id, which is used in query filter. Highest bidder, or at least his name, is displayed on the Bidding Screen. Bidding also uses Datetime component to keep track of the start and end of bidding, but it doesn't need to be loaded from database, since it was passed from Auction Screen along with all the other information about bidding. Making a new bid is performed through Bid system. Bidder inserts a new bid and after validation, highest bidder is overwritten with current Bidder and the bid is recursively saved to auction, and then to database.

Conclusion

This model encompasses the basic understanding of charity e-auction system. It describes basic boundaries of various classes of users in system and Use Cases related to them. Class model defines systems structure and component model explains how different subsystems are connected. All these models together create a draft of how such program should and would function if it was made a reality.

However, there are many things that weren't thought of and a lot of things would probably have to be implemented "on the fly" when real users would start to use it. The biggest challenge is security. Auction systems generally work with large amounts of money which requires thorough security checks. The weakness of this system is how "open" it is, meaning, anyone can register as an Organizer and create auctions, which allows for a potential fraud. Automatic transfer of money to selected charity, which can only be registered by Operators, lowers the risk of raising money for non-existing charities, or stealing of raised money, but it isn't completely secure and would require thorough consideration and analysis.

Of course, this model describes only basic functionality and if it were to be actually programmed, the system would be under much closer and stricter inspection than as it is now.

Notes

Code in Java was programmed in Java SDK 17, with the use of SQLite database driver. This driver, as well as instructions how to set it up in IntelliJ IDEA is included in dependencies folder and README file.