

Software Engineering Übung

Gruppe: 6

Übungsleiter: Dr. Martin Vasko

Designmodell v.2.0

## Projektname: u:Buy



Quelle: <http://www.mypennybids.com/category/penny-auctions-info/page/2/>

### Projektteam: (Team 4)

| Nachname    | Vorname   | Matrikelnummer | E-Mail-Adresse             |
|-------------|-----------|----------------|----------------------------|
| Andre       | Erwin     | 1329095        | a1329095@unet.univie.ac.at |
| Eggerth     | Cordula   | 0750881        | a0750881@unet.univie.ac.at |
| Pacher      | Christina | 1406933        | a1406933@unet.univie.ac.at |
| Regenfelder | Martin    | 1104500        | a1104500@unet.univie.ac.at |

### CEWebS-Teamseite:

[https://cewebs.cs.univie.ac.at/SWE/ws16/index.php?m=D&t=uebung&c=show&CEWebS\\_c=g050052-6t4](https://cewebs.cs.univie.ac.at/SWE/ws16/index.php?m=D&t=uebung&c=show&CEWebS_c=g050052-6t4)

Datum der Abgabe: 18.01.2017

# 1 Klassendesign

Das Projekt umfasst drei Packages, die gemäß dem Model-View-Controller-Pattern angelegt sind. Im Model befinden sich die Klassen „User“, „Auction“, „SearchTutorAd“ und „Message“. Im View befinden sich sämtliche Servletklassen, die das Handling der Benutzereingaben übernehmen. Im Controller sind die Klassen „UserManagement“, „AuctionManagement“, „StatisticManagement“, „SearchTutorAdManagement“, „MessageManagement“ und sämtliche zur persistenten Datenspeicherung notwendigen Klassen, wie z.B. „SerializedUserDAO“ und die zugehörigen Interfaces, hier z.B. Interface „UserDAO“ angelegt. Die Serialisierung wird für die jeweiligen Klassen getrennt vorgenommen, sodass die Übersichtlichkeit gegeben ist und die Implementierung erleichtert wird.

## **Model:**

### **Klasse User:**

#### Instanzvariablen (bzw. Attribute):

String username  
String password  
String firstname  
String lastname  
int usertype  
UUID id  
boolean active

#### Methoden:

nur Getter und Setter im Model

#### Beschreibung:

Die Klasse User enthält Instanzvariablen, sodass der User sich mit username, password, firstname und lastname registrieren kann. Der usertype bestimmt, ob der betreffende User ein normaler User, ein Admin oder ein Forscher ist – gespeichert werden die Informationen über den Typ als enum. Zusätzlich bekommt jeder User eine eindeutige ID. Active zeigt an, ob ein User der Plattform „active“ oder „banned“ ist – die Implementierung des Status erfolgt ebenfalls als boolean. Ein accountstatus auf „banned“ bedeutet, dass der User dauerhaft gesperrt ist und sich nicht mehr einloggen kann. Ein accountstatus „active“ bedeutet, dass der User gemäß seinen Rechten die Plattform nutzen kann – bei der Erstregistrierung ist der accountstatus per Default als „active“ zu setzen.

### **Klasse Auction:**

#### Instanzvariablen (bzw. Attribute):

UUID id  
String title  
String description  
Calendar end  
Calendar start  
ArrayList<Bid> bidList  
UUID createdByUserId  
boolean expired  
boolean sold  
String auctionGroup  
double startPrice

#### Methoden:

nur Getter und Setter im Model

Beschreibung:

Ein Objekt der Klasse Auction wird eindeutig durch eine ID klassifiziert. Damit der User leichter danach suchen kann, gibt es auch einen Titel und eine Beschreibung des Auktionsgegenstands. Der User, der die Auktion erstellt, muss hierbei auch ein Start- und Enddatum der Auktion eingeben, und von ihm wird auch die UserID gespeichert. Als Variablen vom Typ boolean werden die Eigenschaften, ob die Auktion abgelaufen (expired) und ob der Auktionsgegenstand verkauft wurde (sold) oder nicht, markiert. Zu jeder Auktion gibt es auch eine Auktionsgruppe, z.B. Lernmaterial, Nachhilfe oder Ähnliches. Bei der Erstellung der Auktion muss vom Anbieter auch ein Startpreis eingegeben werden – wenn die Auktion abgelaufen ist und Gebote vorliegen, die größer bzw. gleich Startpreis sind, dann wird die Variable sold auf true gesetzt.

**Klasse Bid:**

Instanzenvariablen (bzw. Attribute):

UUID bidId  
UUID userId  
double amount  
Calendar bidDate

Methoden:

nur Getter und Setter im Model

Beschreibung:

Die Klasse Bid erfasst als Instanz Gebote, die von Usern für eine bestimmte Auktion gemacht werden. Die Gebote werden im Konstruktor darauf geprüft, dass sie einen amount, der höher/gleich startPrice ist, haben. Jedes Bid wird in der bidList der dazugehörigen Auktion gespeichert.

**Klasse SearchTutorAd:**

Instanzenvariablen (bzw. Attribute):

UUID adId  
UUID userIdDemand  
UUID userIdSupply  
String course  
double maxPrice  
boolean adMatched

Methoden:

nur Getter und Setter im Model

Beschreibung:

Jeder User kann, wenn er eine Auktion gewonnen hat, dem Users, dem die Auktion gehörte, ein Rating geben. Dazu werden die zugehörigen User und die betroffenen Auktion gespeichert. Das Rating kann in Form von Schulnoten (1,2,3,4,5) sein und/oder kann ein Kommentar als String sein.

**Klasse Message:**

Instanzenvariablen (bzw. Attribute):

UUID id  
String sender  
String receiver  
Calendar sent  
String subject  
String text  
boolean feedback  
boolean read

Methoden:

nur Getter und Setter im Model

**View:**

Im View befinden sich **alle Servlet-Klassen**, die für das Handling der Benutzereingaben notwendig sind.

Beispielsweise sind dies folgende Servlets:

- LogoutServlet
- LoginServlet
- PlaceBidServlet
- AuctionSearchServlet
- AuctionListServlet
- addAuctionServlet
- deleteAuctionServlet
- etc.

Die Servlets implementieren alle die öffentlichen Methoden doPost() und doGet(), die den Returntype void haben.

Die Servlets verwenden die Methoden aus dem Controller, i.e. aus den Klassen UserManagement, StatisticManagement, SearchTutorAdManagement, MessageManagement und AuctionManagement, um die Benutzereingaben und Anforderungen abzuarbeiten.

**Controller:**

**Klasse UserManagement:**

Instanzvariablen (bzw. Attribute):

UserDAO userdao

AuctionManagement am

Methoden:

login(String username, String password): boolean

logout(String username): void

registerUser(String username, String password, String firstname, String lastname, int usertype): boolean

getUserByUsername(String username): User

getUserById(int id): User

deleteUser(User user): void

banUser(int id): void

editPassword(String password): void

getNumberAllUsers(): int

getNumberActiveUsers(): int

getNumberBannedUsers(): int

getNumberResearchers(): int

getNumberNormalUsers(): int

getNumberAdmins(): int

getAllUserRatings(): ArrayList<UserRating>

checkSessionKey(): boolean

getUserByUsername(String username): User

Beschreibung:

In der Klasse *UserManagement* werden alle Aktionen bzgl. Usern vorgenommen, die von der Erstregistrierung und dem Login bis zum Logout und über alle Prozesse, die dazwischen stattfinden, reichen. Die Klasse interagiert über das Interface *UserDAO* mit der Klasse *SerializedUserDAO*, in der die persistente Speicherung der User-Objekte vorgenommen wird.

#### **Klasse *StatisticManagement*:**

##### Instanzvariablen (bzw. Attribute):

*UserManagement* usermanagement

*AuctionManagement* auctionmanagement

##### Methoden:

*allBidsByUser*(int id): int

*getNumberAllUsers*(): int

*getNumberActiveUsers*(): int

*getNumberBannedUsers*(): int

*getNumberResearchers*(): int

*getNumberNormalUsers*(): int

*getNumberAdmins*(): int

*getNumberUserRatings*(): int

*getAverageUserRating*(): int

*getMostRatedUsers*(): ArrayList<User> // zeigt User an, die die meisten Ratings erhalten haben

*getMostRatingUsers*(): ArrayList<User> // zeigt User an, die die meisten Ratings verfasst haben

*getNumberRatedAuctions*(): int

*getMostRatedAuctions*(): ArrayList<Auction> // zeigt Auktionen, die am meisten Ratings erhielten

*getNumberAuctionGroups*(): int

*getNumberAuctionsperAuctionGroup*(String groupname): int

*getNumberAllBids*(): int

*getNumberExpiredAuctions*(): int

*getNumberActiveAuctions*(): int

*getNumberSoldAuctions*(): int

*getAverageAuctionStartPrice*(): int

*getAuctionOwnerUsername*() : String

##### Beschreibung:

Die Klasse *StatisticManagement* dient zur Erstellung gewisser Statistiken über User- und Auktionsverhalten auf der Plattform mittels verschiedener Methoden. Der Forscher kann sich diese Statistiken anzeigen lassen. Die Klasse verwendet das *UserManagement* und das *AuctionManagement* und kann Daten von dort erhalten.

#### **Klasse *AuctionManagement*:**

##### Instanzvariablen (bzw. Attribute):

ArrayList<String> auctionGroupList

AuctionGroupDAO auctiongroupdao

AuctionDAO auctiondao

##### Methoden:

*addAuctionGroup*(String group): void

*deleteAuctionGroup*(String group): void

*addAuction*(Auction auction): void

*deleteAuction*(Auction auction): void

*addBid*(Bid bid, Auction auction): void

*searchAuctionByKeyword*(String keyword): ArrayList<Auction>

*searchAuctionByCategory*(String category): ArrayList<Auction>

*auctionDaysLeft*(Auction auction): int

*searchAuctionByUser*(String user): ArrayList<Auction>

*getHighestBid(Auction auction): Bid*  
*getAllAuctions(): ArrayList<Auction>*  
*getAllBids(Auction auction): ArrayList<Bid>*  
*getAllExpiredAuctions(): ArrayList<Auction>*  
*getAllActiveAuctions(): ArrayList<Auction>*  
*getAllSoldAuctions(): ArrayList<Auction>*  
*checkExpired(Auction auction): boolean*  
*checkSold(Auction auction): Boolean*  
*getAuctionById(int id): Auction*  
*addBidToAuctionBidList(Auction auction, Bid bid): void*

**Beschreibung:**

Die Klasse AuctionManagement beinhaltet alle Methoden, die sich direkt auf die Auktionen oder Auktionsgruppen beziehen, und verwendet die Klassen SerializedAuctionDAO und SerializedAuctionGroupDAO (über die jeweiligen Interfaces) zur persistenten Speicherung der auktionsbezogenen Daten. Als Instanzvariable hat sie außerdem eine ArrayList mit Namen der auf der Plattform möglichen Auktionsgruppen vom Typ String.

**Klasse SerializedUserDAO:**

Instanzvariablen (bzw. Attribute):

String filename

Methoden:

*getUserList(): ArrayList<User>*  
*getUserById(int id): User*  
*addUser(User user): void*  
*deleteUser(User user): void*  
*addBidToUserBidList(Bid bid, User user): void*  
*modifyUserPassword(User user, String password): void*

**Beschreibung:**

In der Klasse SerializedUserDAO wird das Interface UserDAO implementiert. Es werden Methoden zur persistenten Datenspeicherung in einem File implementiert. Die Klasse wird von der Klasse UserManagement verwendet.

**<<interface>> UserDAO:**

Methoden:

*getUserList(): ArrayList<User>*  
*getUserById(int id): User*  
*addUser(User user): void*  
*deleteUser(User user): void*  
*addBidToUserBidList(Bid bid, User user): void*  
*modifyUserPassword(User user, String password): void*

**Beschreibung:**

Das Interface UserDAO fasst Methoden zusammen, die dann in der Klasse SerializedUserDAO implementiert werden.

**Genereller Kommentar zu den Klassen SerializedxxxDAO und zu Interfaces xxxDAO**

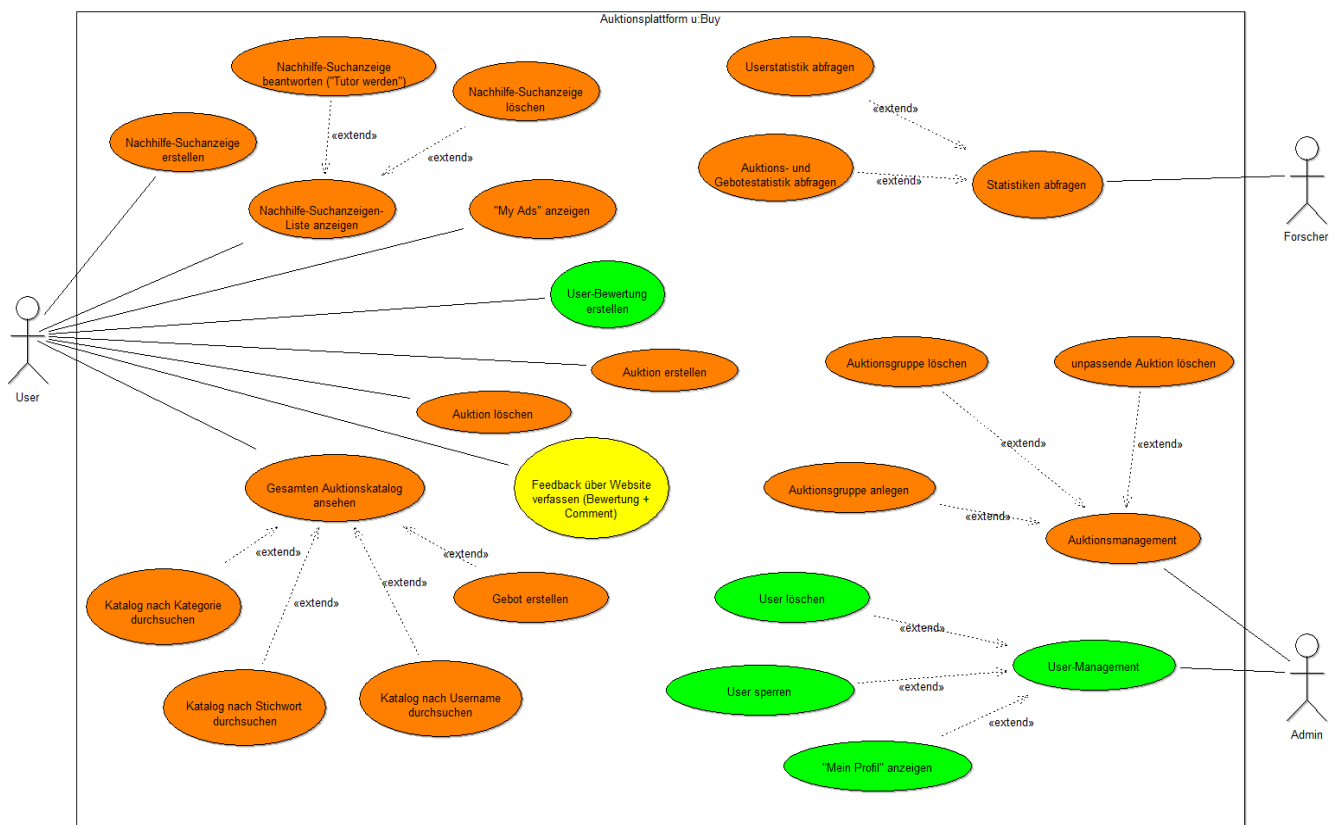
Diese Klassen dienen zur persistenten Datenspeicherung (in Dateien – die Angabe des Pfadname bzw. Dateiname erfolgt über die Instanzvariable filename). Im vorliegenden Projekt wird die Serialisierung der jeweiligen Instanzlisten der Klassen über die zugehörigen SerializedxxxDAO, die die Interfaces xxxDAO implementieren, vorgenommen. Die darin enthaltenen Methoden umfassen zumindest die folgenden Methoden (Anmerkung: XXX steht hier für den jeweiligen Klassennamen):

- *getXXXList(): ArrayList<XXX>*
- *getXXXById(int id): XXX*
- *addXXX(XXX xxx): void*
- *deleteXXX(XXX xxx): void*

Derartige Klassen- bzw. Interfacenamen sind im Projekt folgende:

- *<<interface>> UserDao*
- *class SerializedUserDao*
- *<<interface>> AuctionDao*
- *class SerializedAuctionDao*
- *<<interface>> AuctionGroupDao*
- *class SerializedAuctionGroupDao*
- *<<interface>> UserRatingDao*
- *class SerializedUserRatingDao*

## 2 Use Case Realization Design



Farbcodierung für Implementierung:

Erwin: **blau**

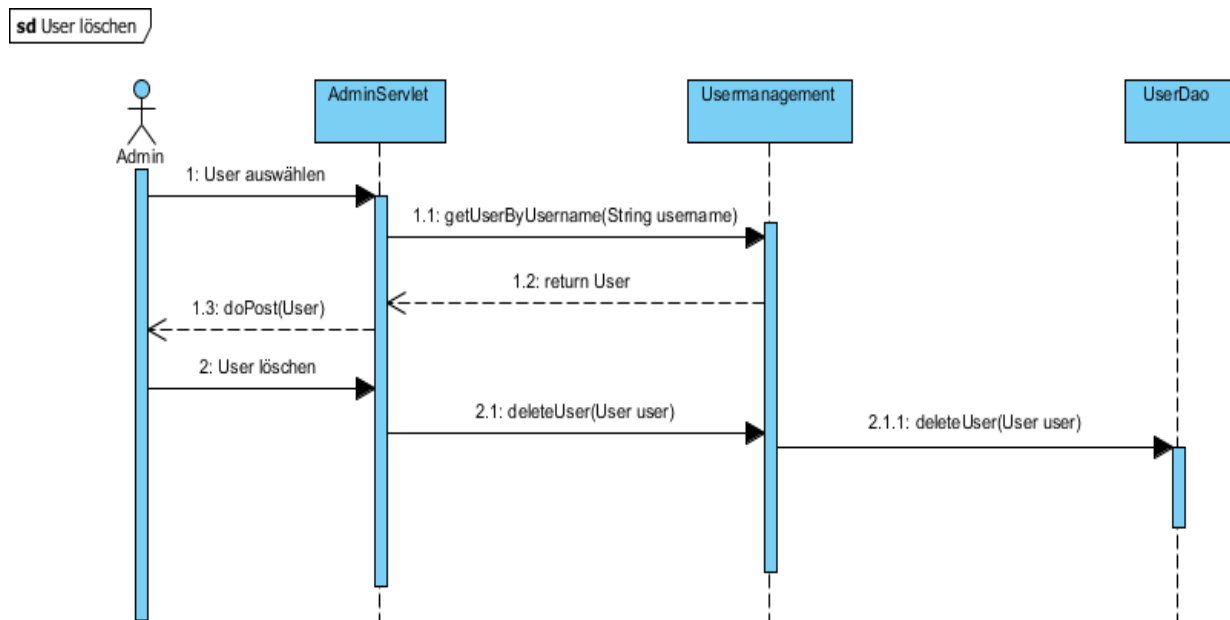
Cordula: **orange**

Christina: **gelb**

Martin: **grün**

## Beispielhafte Sequenzdiagramme von Use-Cases:

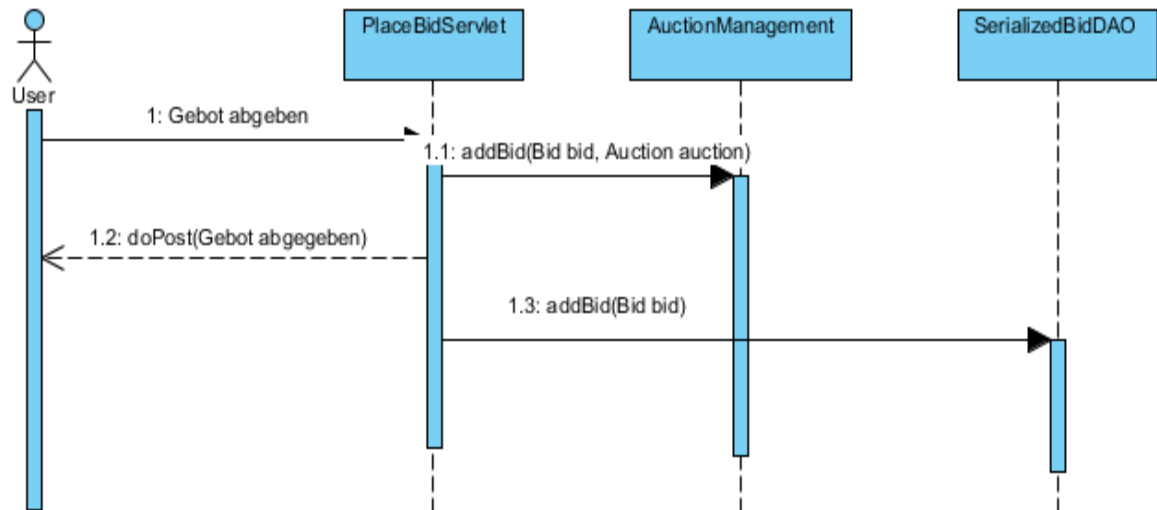
### User löschen:





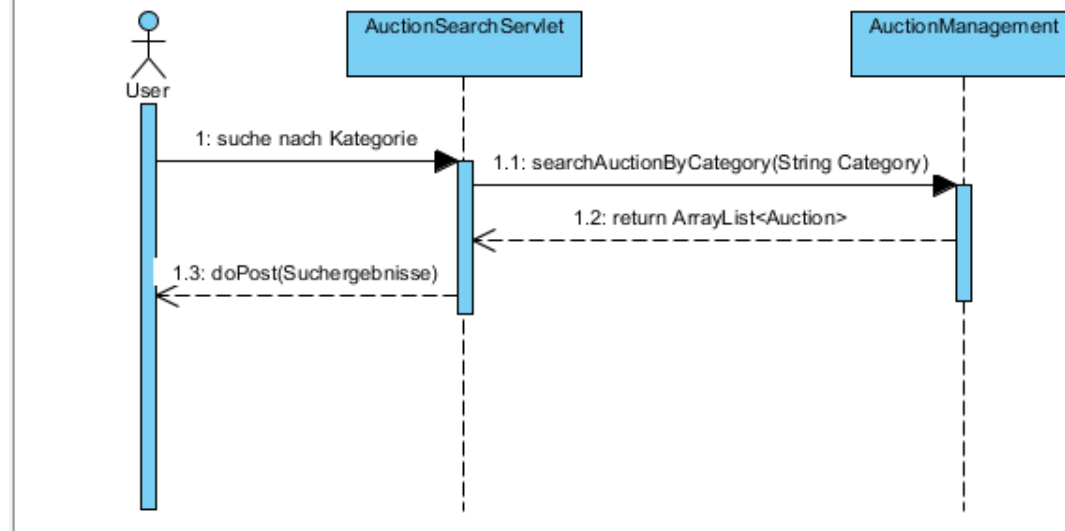
### Gebot erstellen:

sd Gebot erstellen

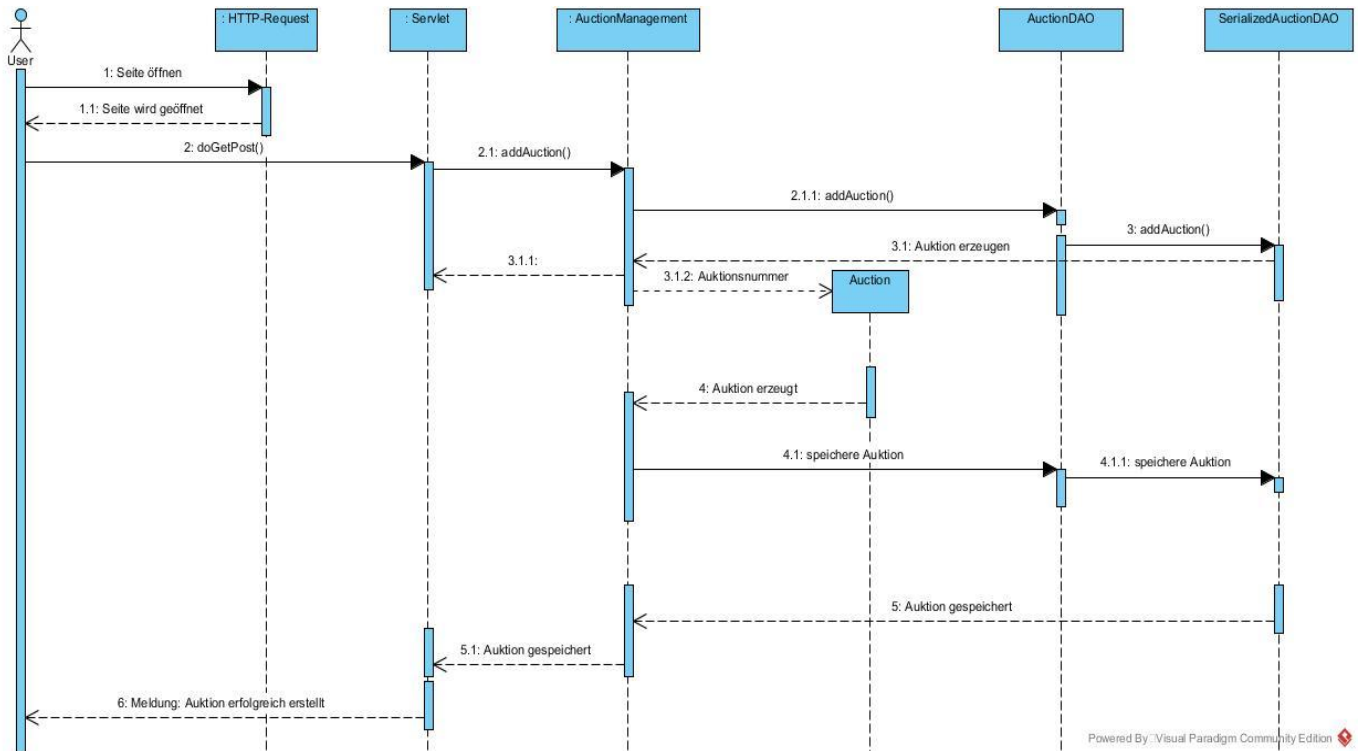


### Katalog nach Kategorie durchsuchen:

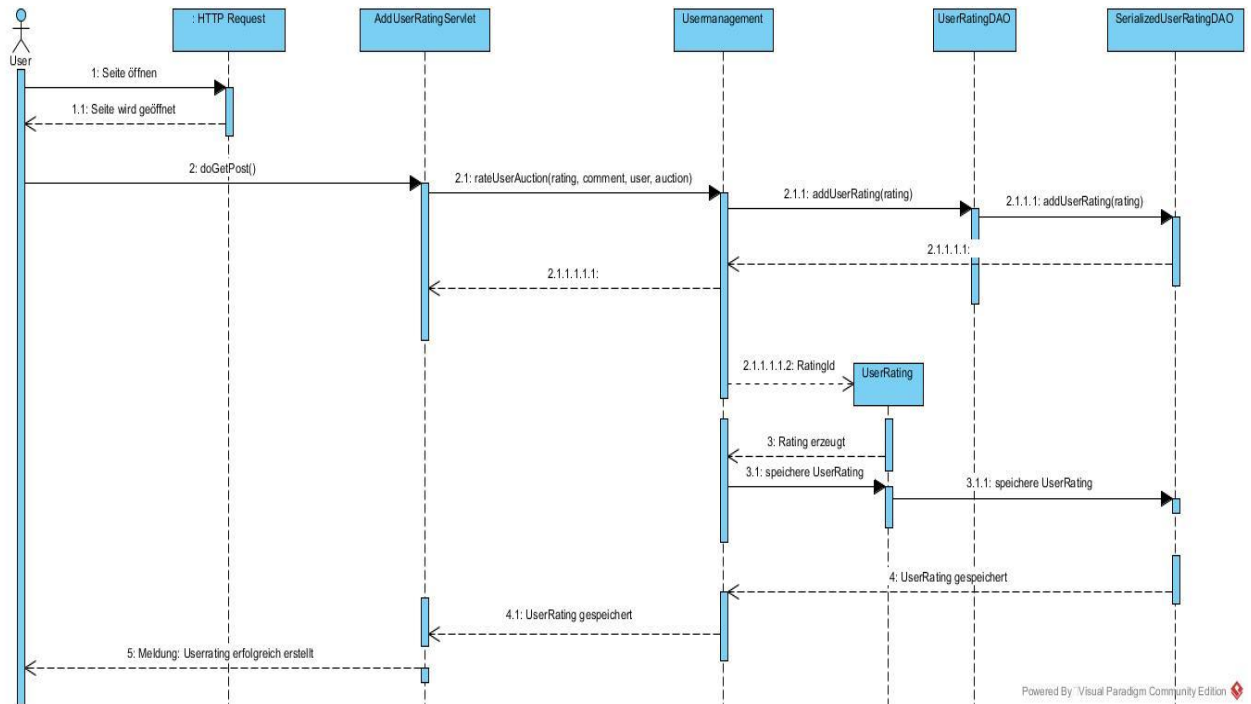
sd Katalog nach Kategorie durchsuchen



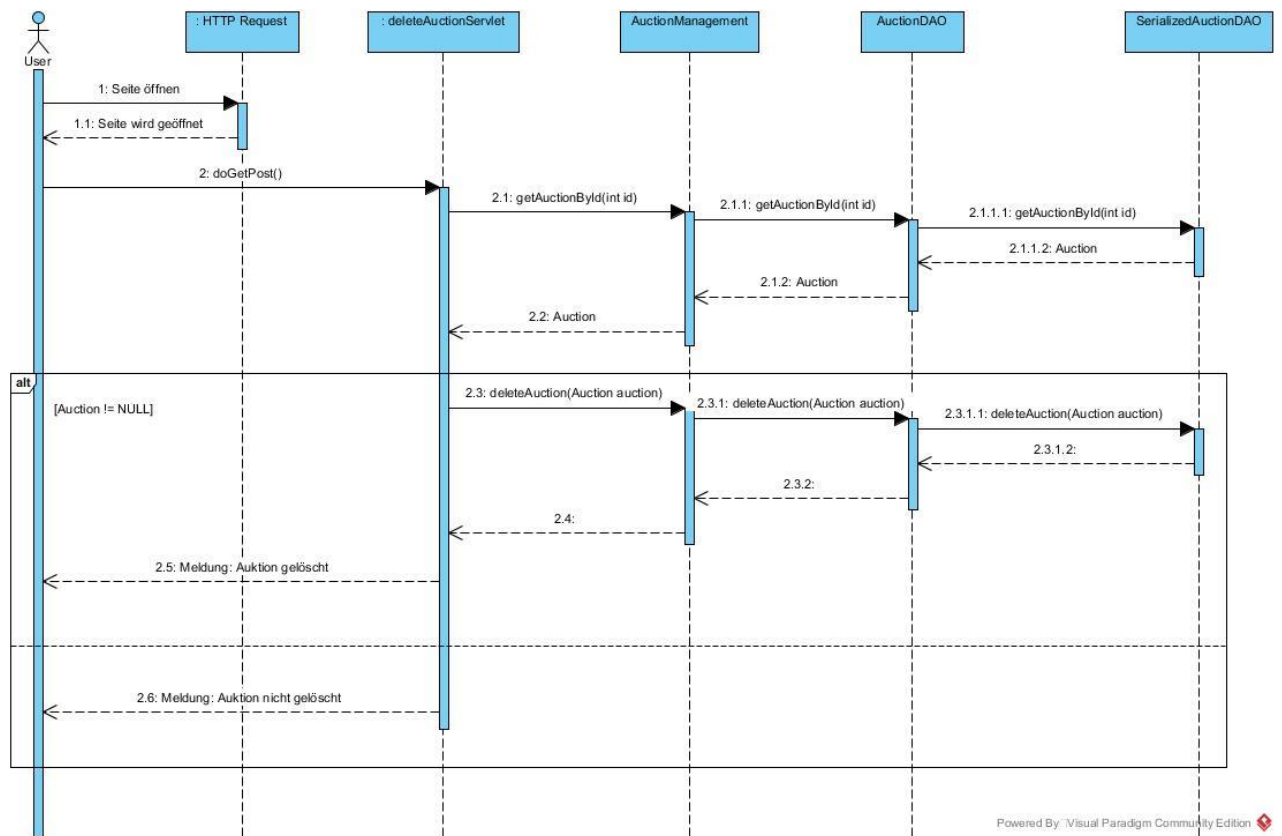
## Auktion erstellen:



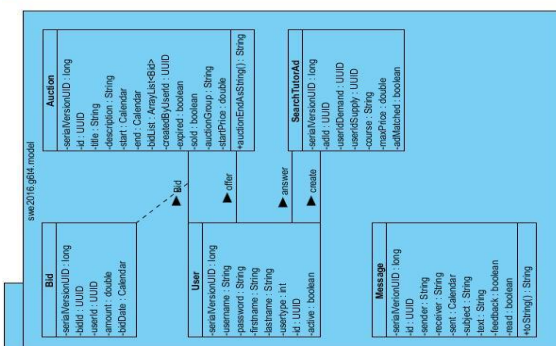
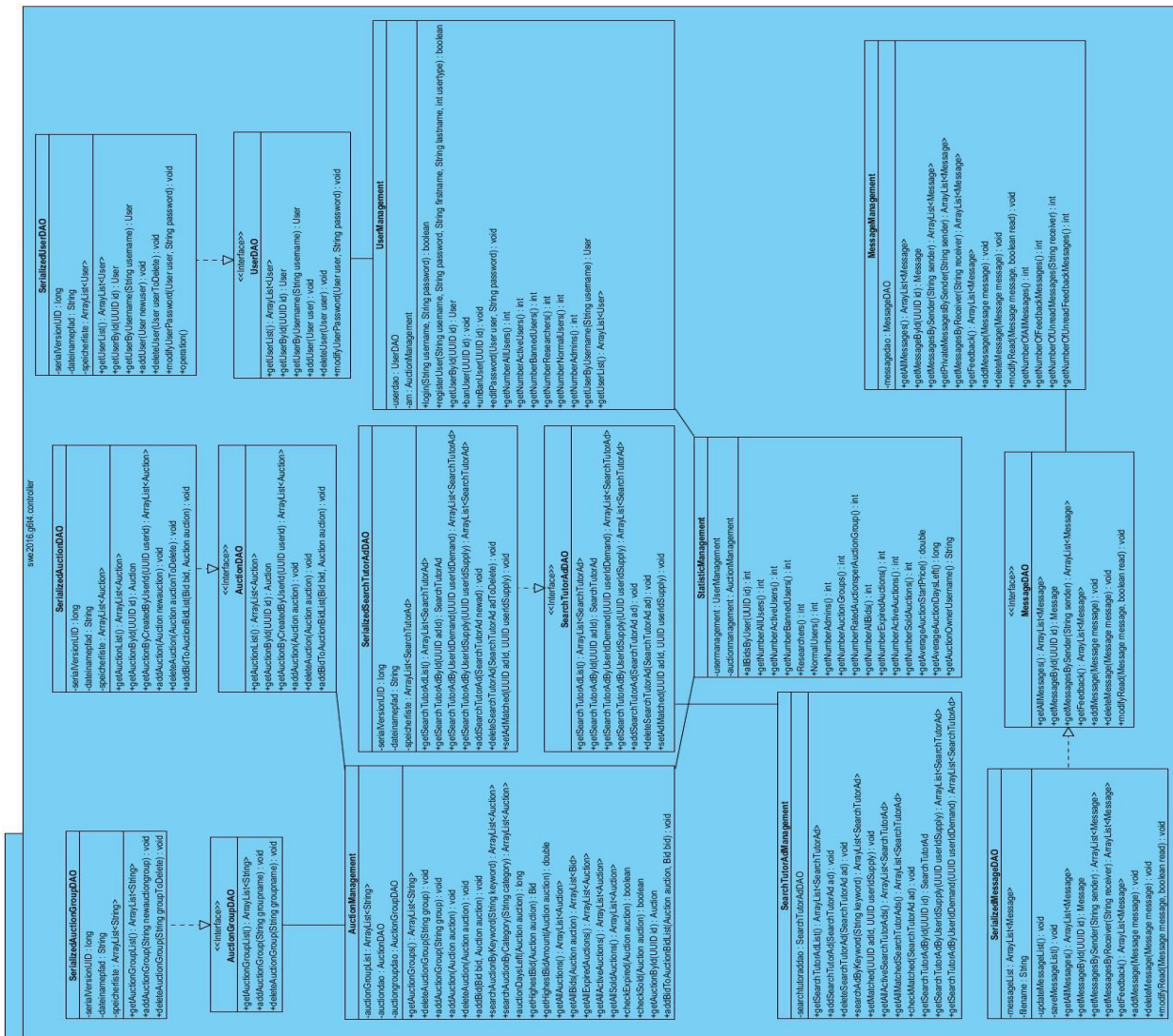
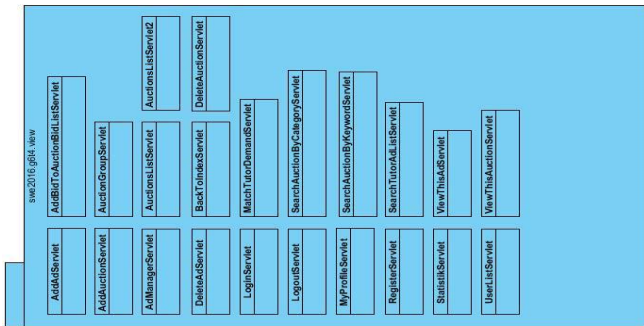
## Userbewertung erstellen:



## Auktion löschen:

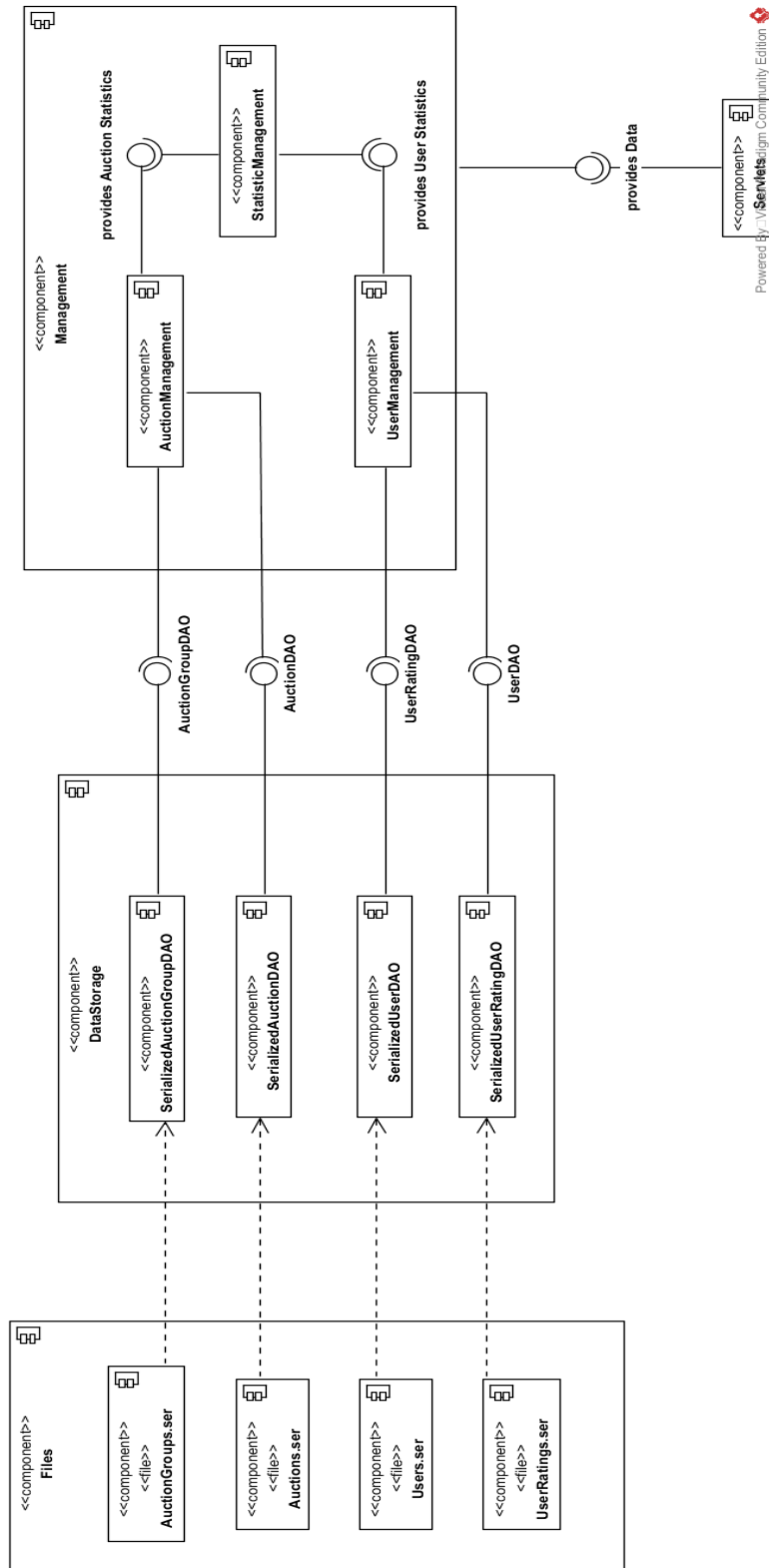


### 3 Übersichtsklassendiagramm



## 4 Architekturbeschreibung

### Komponentendiagramm:



**Deploymentdiagramm:**

