



Tiw Project

Documentazione



Specifiche

Versione HTML pura

Un'applicazione web consente la gestione di aste online. Gli utenti accedono tramite login e possono vendere e acquistare all'asta. La HOME page contiene due link, uno per accedere alla pagina VENDO e uno per accedere alla pagina ACQUISTO. La pagina VENDO mostra una lista delle aste create dall'utente e non ancora chiuse, una lista delle aste da lui create e chiuse e una form per creare un nuovo articolo e una nuova asta per venderlo. L'asta comprende l'articolo da mettere in vendita (codice, nome, descrizione, immagine), prezzo iniziale, rialzo minimo di ogni offerta e una scadenza (data e ora, es 19-04-2021 alle 23:00). La lista delle aste è ordinata per data+ora crescente e riporta: codice e nome dell'articolo, offerta massima, tempo mancante (numero di giorni e ore) tra il momento del login e la data e ora di chiusura dell'asta. Cliccando su un'asta compare una pagina DETTAGLIO ASTA che riporta per un'asta aperta i dati dell'asta e la lista delle offerte (nome utente, prezzo offerto, data e ora dell'offerta) ordinata per data+ora decrescente. Un bottone CHIUDI permette all'utente di chiudere l'asta se è giunta l'ora della scadenza (si ignori il caso di aste scadute ma non chiuse). Se l'asta è chiusa, la pagina riporta i dati dell'asta, il nome dell'aggiudicatario, il prezzo finale e l'indirizzo di spedizione dell'utente. La pagina ACQUISTO contiene una form di ricerca per parola chiave. Quando l'acquirente invia una parola chiave la pagina ACQUISTO si ricarica e mostra un elenco di aste aperte (la cui scadenza è posteriore all'ora dell'invio) il cui articolo contiene la parola chiave nel nome o nella descrizione. La lista è ordinata in modo decrescente in base al tempo (numero di giorni, ore e minuti) mancante alla chiusura. Cliccando su un'asta aperta compare la pagina OFFERTA che mostra i dati dell'articolo, l'elenco delle offerte pervenute in ordine di data+ora decrescente e un campo di input per inserire la propria offerta, che deve essere superiore all'offerta massima corrente. Dopo l'invio dell'offerta la pagina OFFERTA mostra l'elenco delle offerte aggiornate. La pagina ACQUISTO contiene anche un elenco delle offerte aggiudicate all'utente con i dati dell'articolo e il prezzo finale.

Javascript

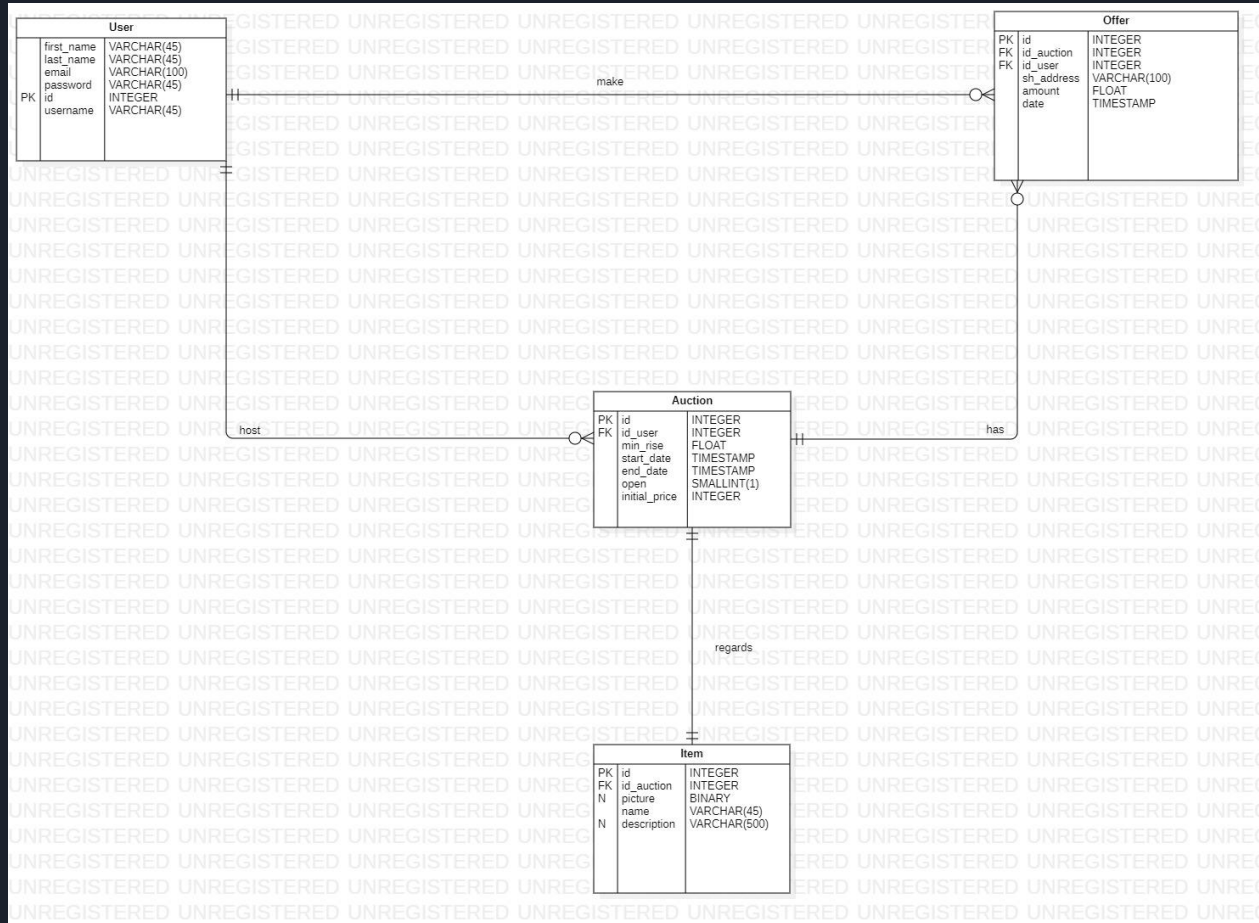
Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- Dopo il login, l'intera applicazione è realizzata con un'unica pagina.
- Se l'utente accede per la prima volta l'applicazione mostra il contenuto della pagina ACQUISTO. Se l'utente ha già usato l'applicazione, questa mostra il contenuto della pagina VENDO se l'ultima azione dell'utente è stata la creazione di un'asta; altrimenti mostra il contenuto della pagina ACQUISTO con l'elenco (eventualmente vuoto) delle aste su cui l'utente ha cliccato in precedenza e che sono ancora aperte. L'informazione dell'ultima azione compiuta e delle aste visitate è memorizzata a lato client per la durata di un mese.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica solo del contenuto da aggiornare a seguito dell'evento.

Database



EER Diagram



DDL

Auction

```
1 CREATE TABLE `auction` (  
2   `id` int NOT NULL AUTO_INCREMENT,  
3   `start_date` timestamp NOT NULL,  
4   `end_date` timestamp NOT NULL,  
5   `min_rise` float NOT NULL,  
6   `id_user` int NOT NULL,  
7   `open` tinyint(1) NOT NULL DEFAULT '1',  
8   `initial_price` float NOT NULL,  
9   PRIMARY KEY (`id`),  
10  KEY `id_idx` (`id_user`),  
11  CONSTRAINT `id_user` FOREIGN KEY (`id_user`) REFERENCES `user` (`id`)  
12 ) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Item

```
1 CREATE TABLE `item` (  
2   `id` int NOT NULL AUTO_INCREMENT,  
3   `name` varchar(45) NOT NULL,  
4   `description` varchar(500) DEFAULT NULL,  
5   `picture` blob,  
6   `id_auction` int NOT NULL,  
7   PRIMARY KEY (`id`),  
8   KEY `id_auction_idx` (`id_auction`),  
9   CONSTRAINT `id_auction` FOREIGN KEY (`id_auction`) REFERENCES `auction` (`id`) ON DELETE CASCADE ON UPDATE CASCADE  
10 ) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Offer

```
1 CREATE TABLE `offer` (  
2   `id` int NOT NULL AUTO_INCREMENT,  
3   `amount` float NOT NULL,  
4   `sh_address` varchar(100) NOT NULL,  
5   `id_user` int NOT NULL,  
6   `id_auction` int NOT NULL,  
7   `date` timestamp NOT NULL,  
8   PRIMARY KEY (`id`),  
9   KEY `id_user_idx` (`id_user`),  
10  KEY `id_auction_idx` (`id_auction`),  
11  CONSTRAINT `id_auction` FOREIGN KEY (`id_auction`) REFERENCES `auction` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,  
12  CONSTRAINT `id_user` FOREIGN KEY (`id_user`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE  
13 ) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

User

```
1 CREATE TABLE `user` (  
2   `id` int NOT NULL,  
3   `first_name` varchar(45) NOT NULL,  
4   `last_name` varchar(45) NOT NULL,  
5   `email` varchar(100) NOT NULL,  
6   `password` varchar(45) NOT NULL,  
7   `username` varchar(45) NOT NULL,  
8   PRIMARY KEY (`id`)  
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

View: auction_dashboard

```
1 • CREATE
2     ALGORITHM = UNDEFINED
3     DEFINER = `root`@`localhost`
4     SQL SECURITY DEFINER
5     VIEW `auction_proj`.`auction_dashboard` AS
6     SELECT
7         `auction_proj`.`auction`.`id` AS `id`,
8         `auction_proj`.`auction`.`start_date` AS `start_date`,
9         `auction_proj`.`auction`.`end_date` AS `end_date`,
10        `auction_proj`.`auction`.`id_user` AS `id_user`,
11        `auction_proj`.`auction`.`min_rise` AS `min_rise`,
12        `auction_proj`.`auction`.`initial_price` AS `initial_price`,
13        `auction_proj`.`auction`.`open` AS `open`,
14        `auction_proj`.`item`.`id` AS `item_id`,
15        `auction_proj`.`item`.`name` AS `name`,
16        `auction_proj`.`item`.`description` AS `description`,
17        `auction_proj`.`item`.`picture` AS `picture`
18    FROM
19        (`auction_proj`.`item`
20     JOIN `auction_proj`.`auction` ON ((`auction_proj`.`item`.`id_auction` = `auction_proj`.`auction`.`id`)))
```

Model



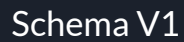


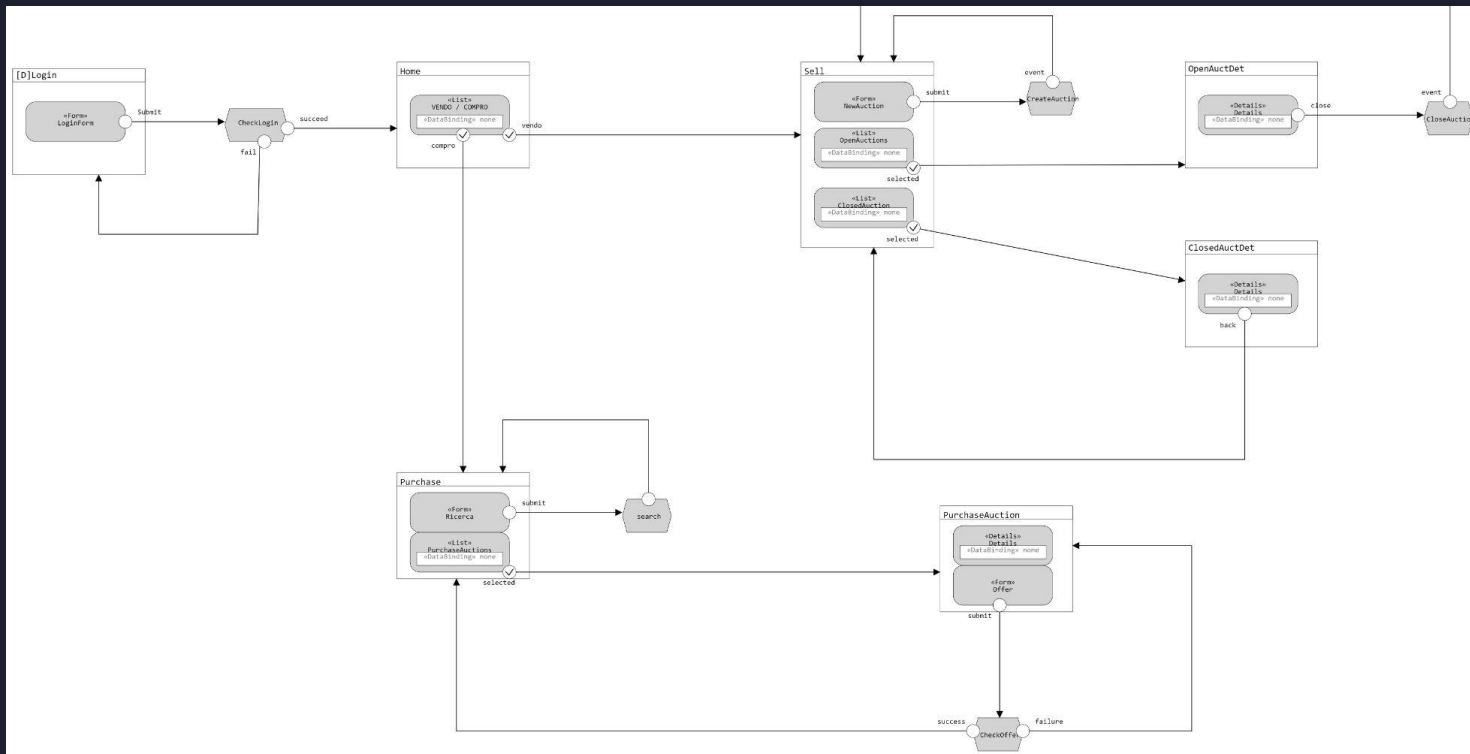
Beans - DAO

Bean	DAO	Table/View
Auction	AuctionDAO <ul style="list-style-type: none">• createAuction• closeAuction	auction
DashboardAuction	DashboardAuctionDAO <ul style="list-style-type: none">• getListOfOpenAuctionsByUser• getListOfClosedAuctionsByUser• getListOfOpenAuctionByKeystring• getAuctionDetail	auction_dashboard
Item	ItemDAO <ul style="list-style-type: none">• createItem	item
Offer	OfferDAO <ul style="list-style-type: none">• createOffer• offerListForAuction• winningBetForAuction	offer
User	UserDAO <ul style="list-style-type: none">• checkCredentials• getUserUsername	user

Application diagram








Schema v2 (IFML like)



Versione Pure HTML

Pages





General

- login.html
- Home.html

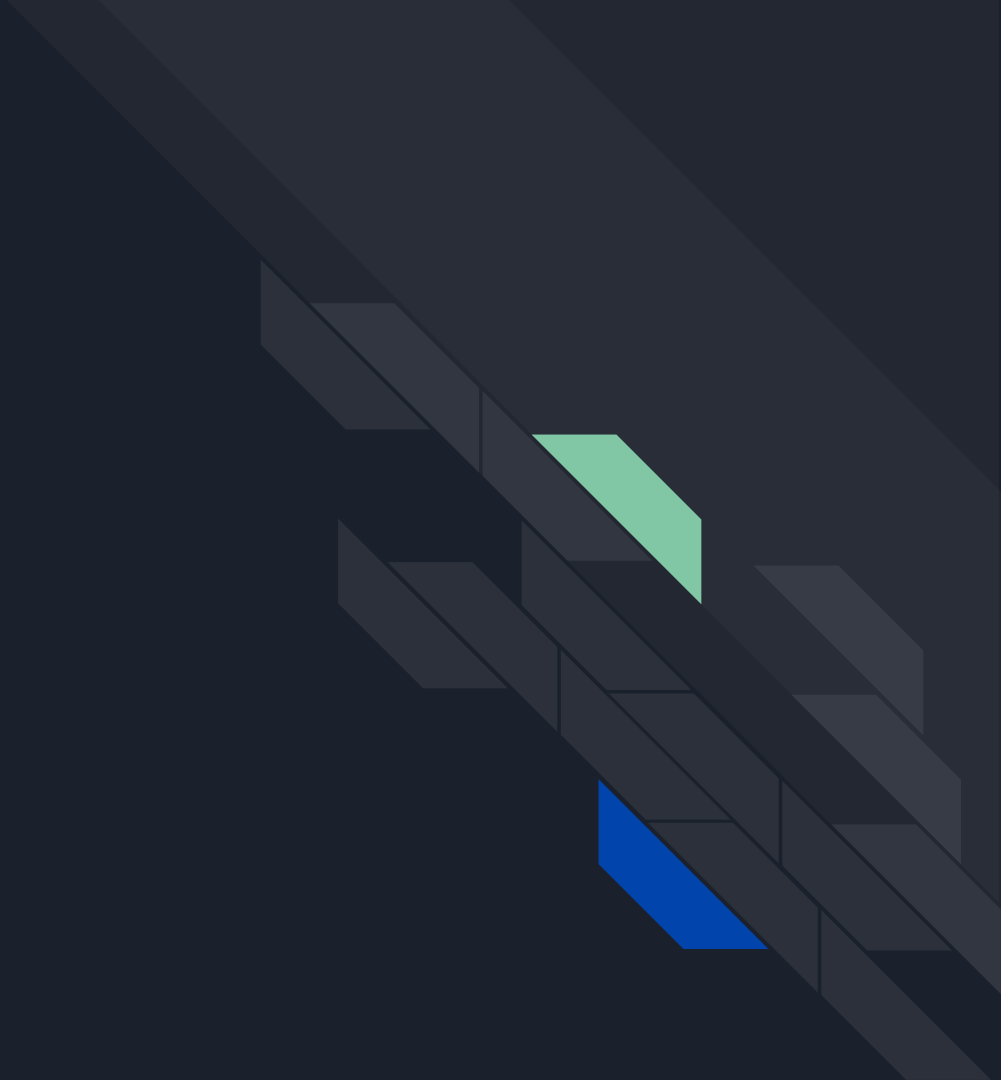
Sell

- Sell.html
- OpenAuctionDet.html
- ClosedAuctionDet.html

Purchase

- Purchase.html
- PurchaseAuction.html

Servlet





Render Servlets

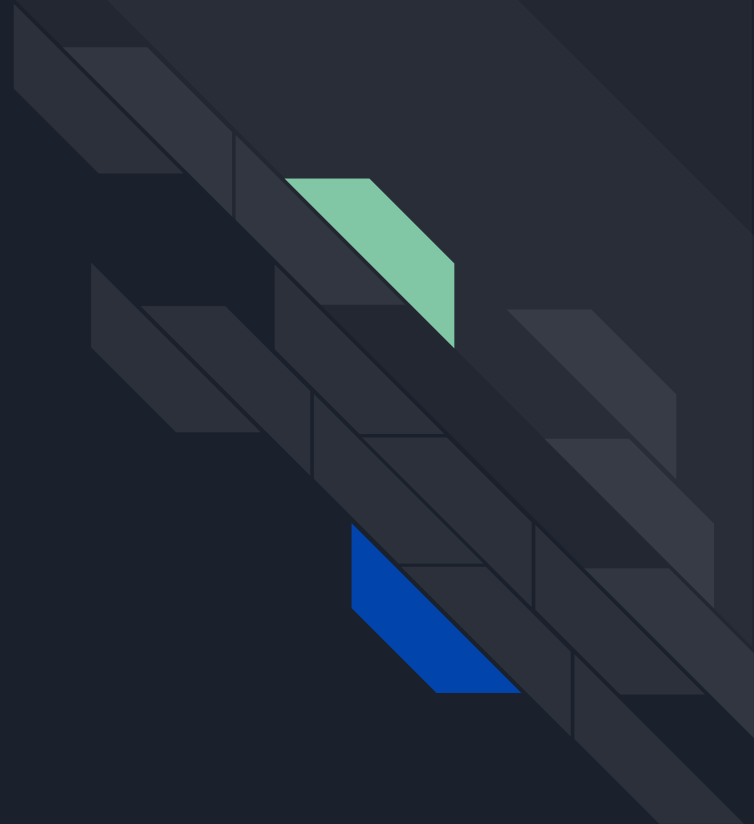
Servlet	renders this page
helloServlet	login.html
GoToHome	Home.html
GoToSell	Sell.html
GoToPurchase	Purchase.html
GetOpenAuction	OpenAuctionDet.html
GetClosedAuction	ClosedAuctionDet.html
GetPurchaseAuction	PurchaseAuction.html



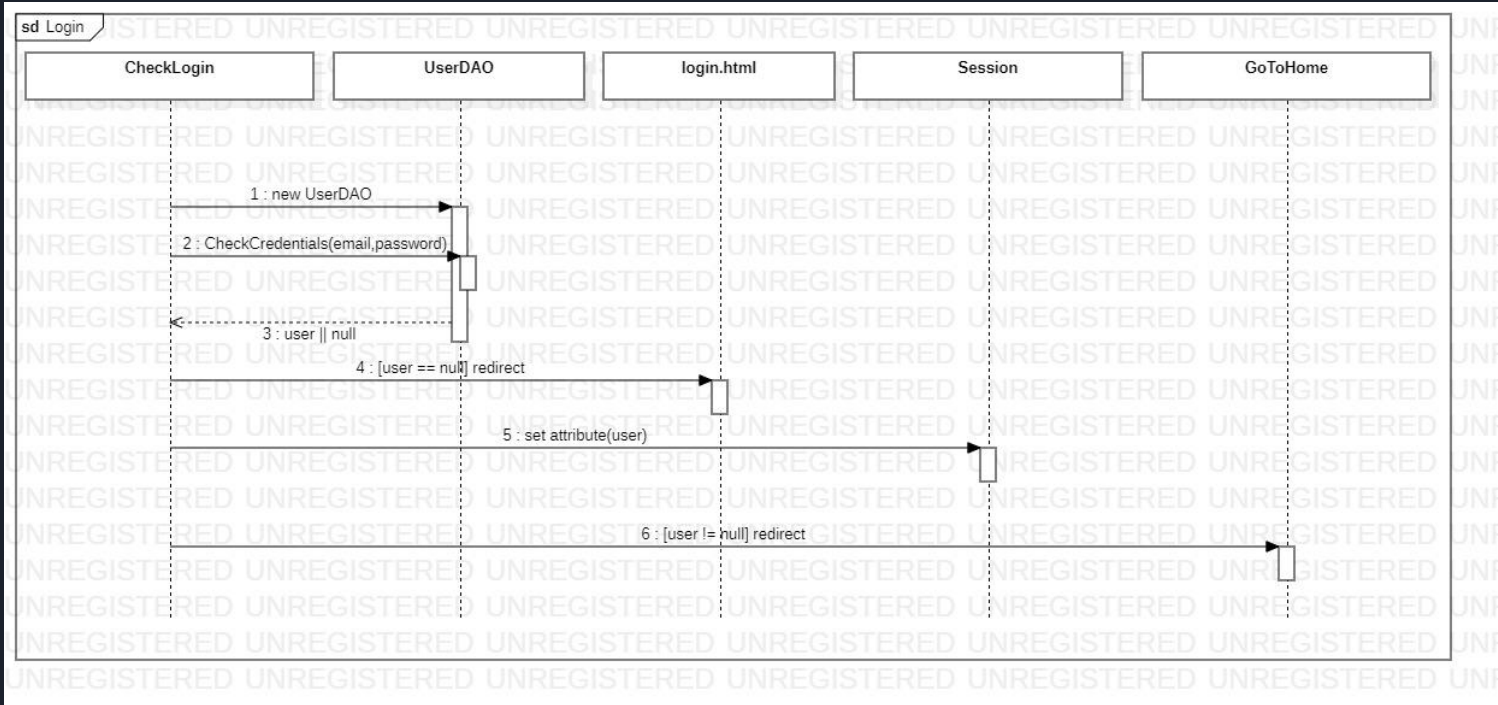
Action Servlets

- CheckLogin
- CreateAuction
- CloseAuction
- AddOffer
- Logout

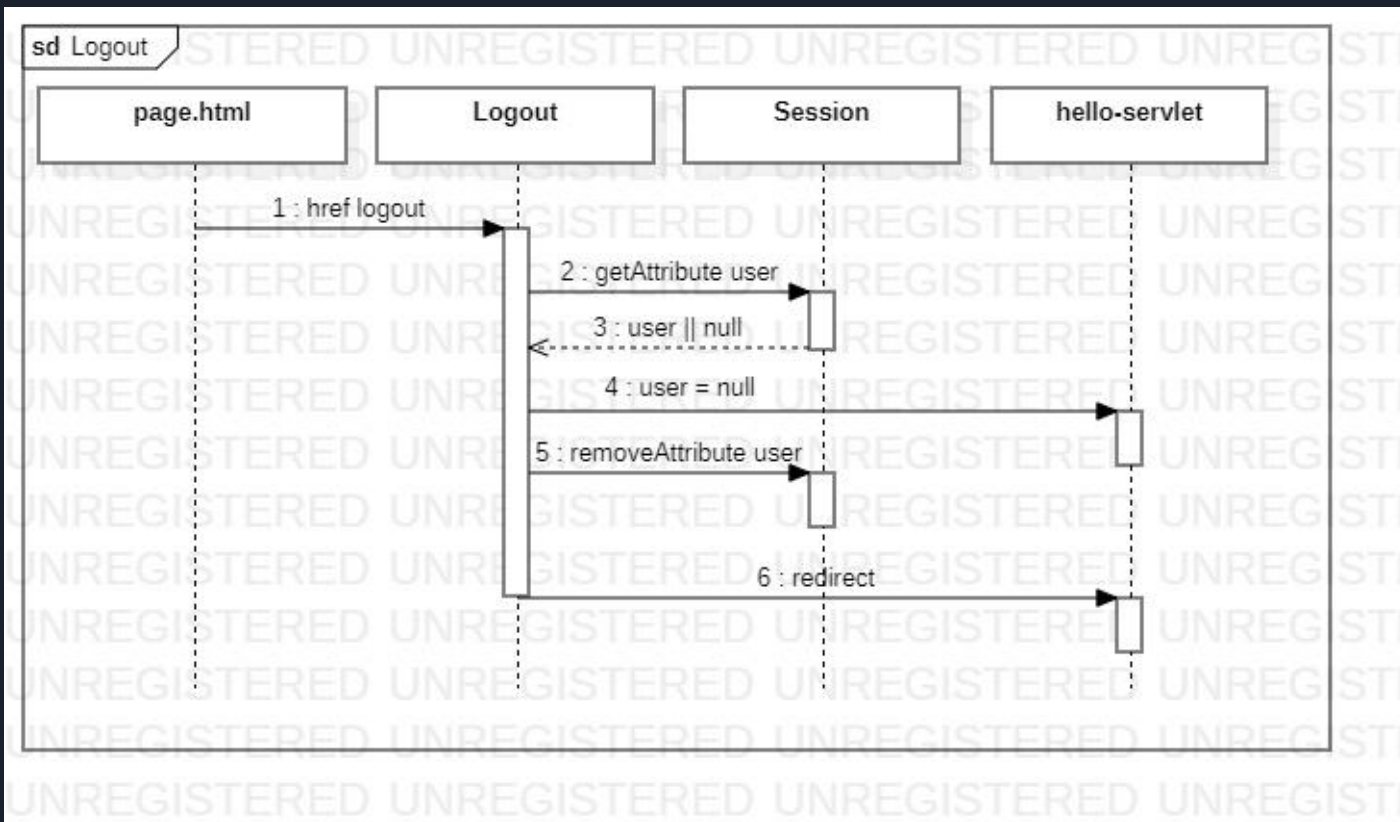
Sequence diagrams



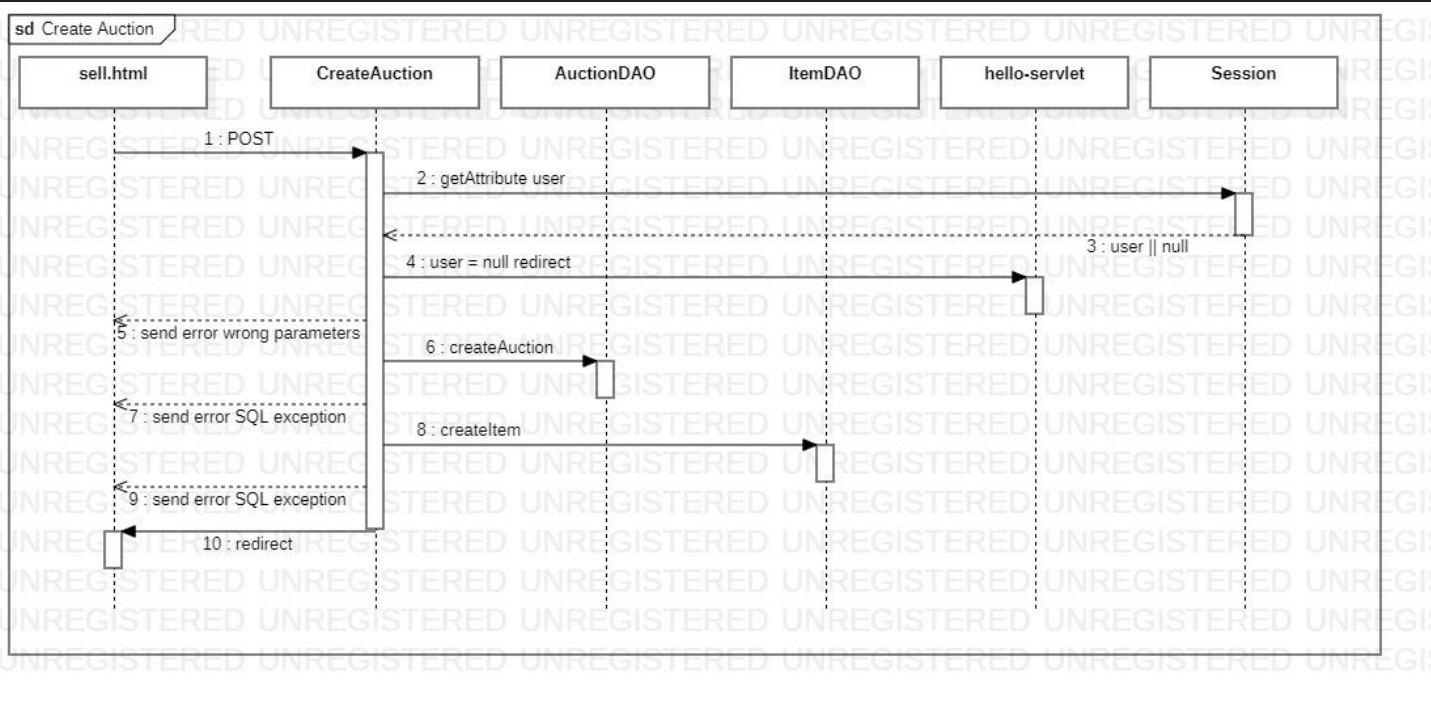
Login



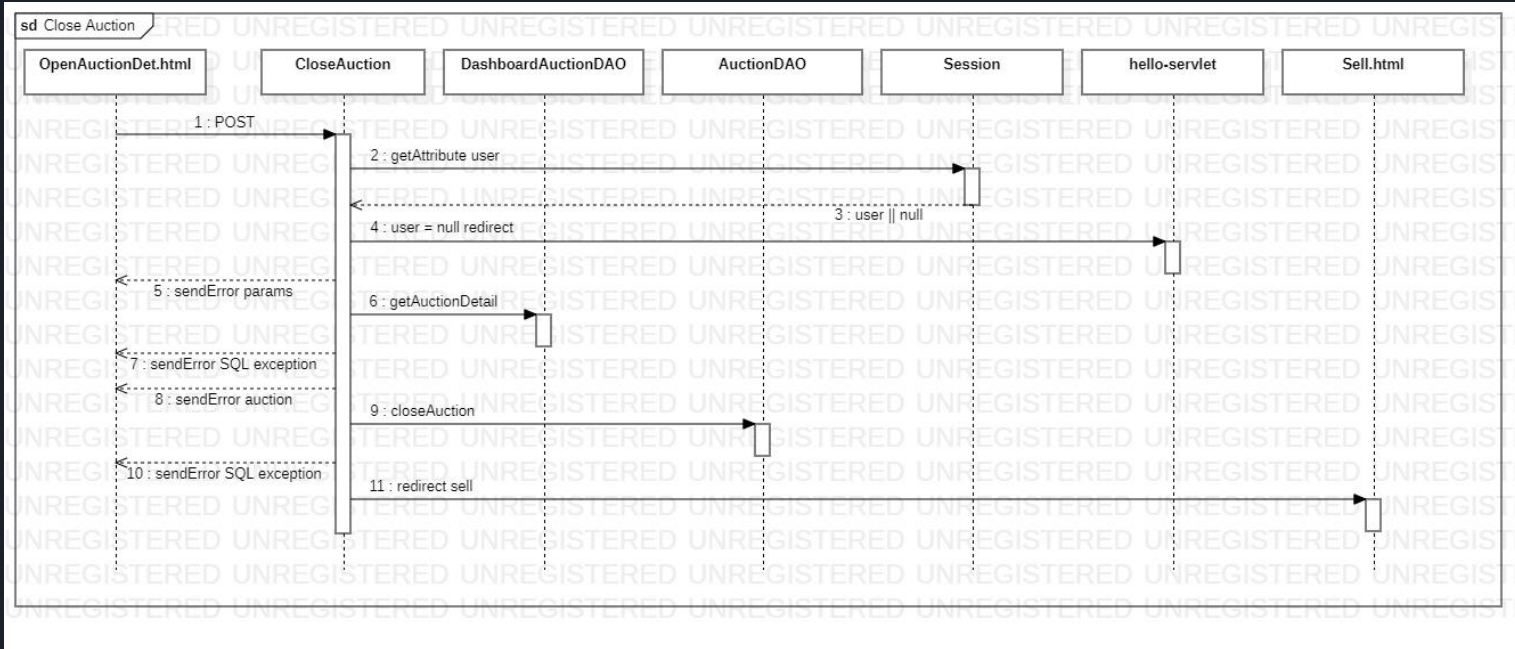
Logout



Create Auction



Close Auction





Versione RIA

API





/CheckLogin

Method	POST
Parameters	<ul style="list-style-type: none">• email => required, max 100• user-password => required, max 45
Return	<p>OK 200 => user.getUsername()</p> <p>KO 400 => errorMessage 500 => SQL error</p>
Note	<code>getSession().setAttribute("user", user);</code>



/GetAuctionData

Method	GET
Parameters	<p>OpenAuctionList</p> <ul style="list-style-type: none">• type => "OPEN", required <p>ClosedAuctionList</p> <ul style="list-style-type: none">• type => "CLOSED", required
Return	<p>OK</p> <p>200 => JSON(auctionList)</p> <p>KO</p> <p>403 => User is not logged in</p> <p>400 => errorMessage</p> <p>500 => SQL error</p>



/SearchAuction

Method	POST
Parameters	<ul style="list-style-type: none">• keystore => required
Return	<p>OK 200 => JSON(auctionList)</p> <p>KO 403 => User is not logged in 400 => errorMessage 500 => SQL error</p>



/CreateAuction

Method	POST
Parameters	<ul style="list-style-type: none">• itemName => required, max 45• itemDescription => max 500• itemPicture• end_date => required, must be future• initial_price => required, >= 0• min_rise => required, >=0
Return	<p>OK 200 => "Auction Created"</p> <p>KO 403 => User is not logged in 400 => errorMessage 500 => SQL error</p>



/CloseAuction

Method	POST
Parameters	<ul style="list-style-type: none">• auction_id => required
Return	<p>OK 200 => "Auction Closed"</p> <p>KO 403 => User is not logged in 400 => "Auction not found" 400 => errorMessage 500 => SQL error</p>



/AddOffer

Method	POST
Parameters	<ul style="list-style-type: none">• amount => required• id_auction => required• sh_address => required, max 100
Return	<p>OK 200 => "Offer Added"</p> <p>KO 403 => User is not logged in 400 => errorMessage 500 => SQL error</p>



/GetOfferData

Method	GET
Parameters	<p>OfferList</p> <ul style="list-style-type: none">• auction_id => required• type => "LIST" required <p>WinningOffer</p> <ul style="list-style-type: none">• auction_id => required• type => "WINNING" required
Return	<p>OK 200 => JSON(offerList) JSON(winningOffer)</p> <p>KO 403 => User is not logged in 400 => errorMessage 500 => SQL error</p>



/GetAuctionDetail

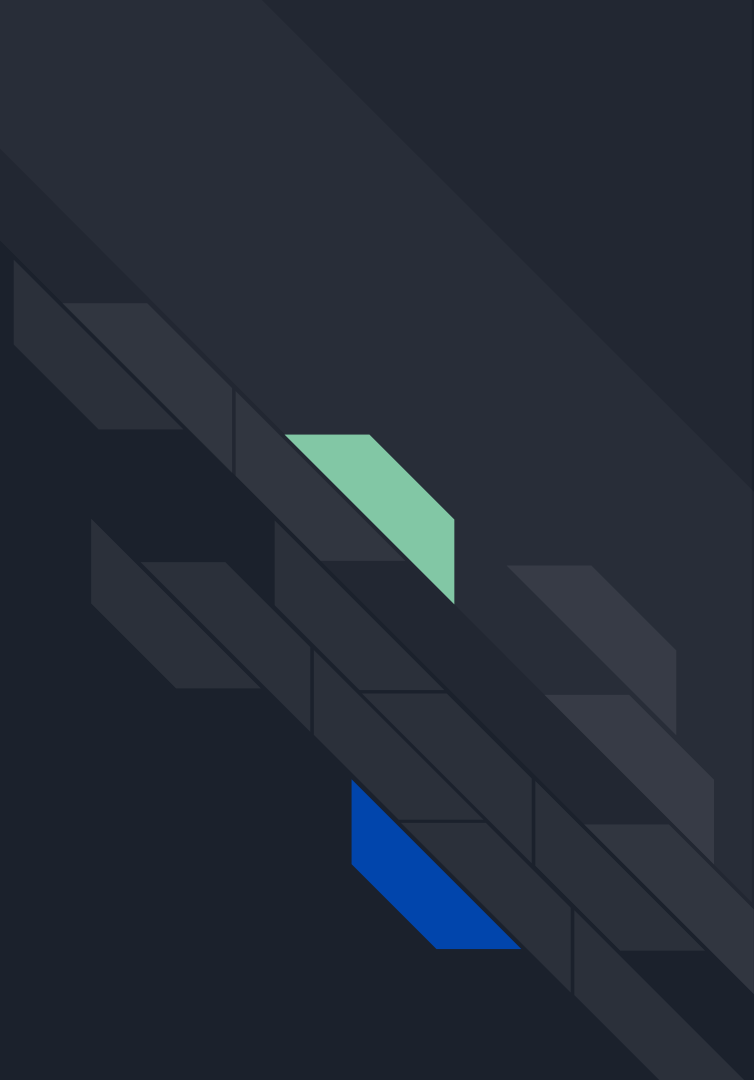
Method	GET
Parameters	<ul style="list-style-type: none">• auction_id => required
Return	<p>OK 200 => JSON(dashboardAuctionToGo)</p> <p>KO 403 => User is not logged in 400 => errorMessage 500 => SQL error</p>
Note	Return DashboardAuctionToGo



/Logout

Method	POST
Parameters	/
Return	OK 200 => "user logged out" KO 403 => User is not logged in

Elementi della pagina



- 
- Menu

Left Part

Purchase

- AuctionSearch
- PurchaseAuctionList

Sell

- OpenAuctionList
- ClosedAuctionList
- AuctionCreateForm

Right Part

- AlertBox
- AuctionWinnerDetail
- OfferForm
- AuctionCloseButton
- AuctionOfferList
- AuctionDetail



Left Part - Components methods

AuctionSearch

- `hide()`
- `show()`
- `registerEvent()`

PurchaseAuctionList

- `hide()`
- `show(auctionList)`
- `initialShow()`

OpenAuctionList

- `hide()`
- `show()`

ClosedAuctionList

- `hide()`
- `show()`

AuctionCreateForm

- `hide()`
- `show()`
- `registerEvent()`



Right Part - Components methods

AlertBox

- show(message)
- reset()

AuctionWinnerDetail

- hide()
- show(auctionId)

OfferForm

- hide()
- show(auctionId)
- registerEvent()

AuctionCloseButton

- hide()
- show(auctionId)
- resgisterEvent()

AuctionOfferList

- hide()
- show(auctionId)

AuctionDetail

- hide()
- show(auctionId)



Page Orchestrator

- start() -> initialize all the components

Left Part of the page

- showPurchase()
- showSell()

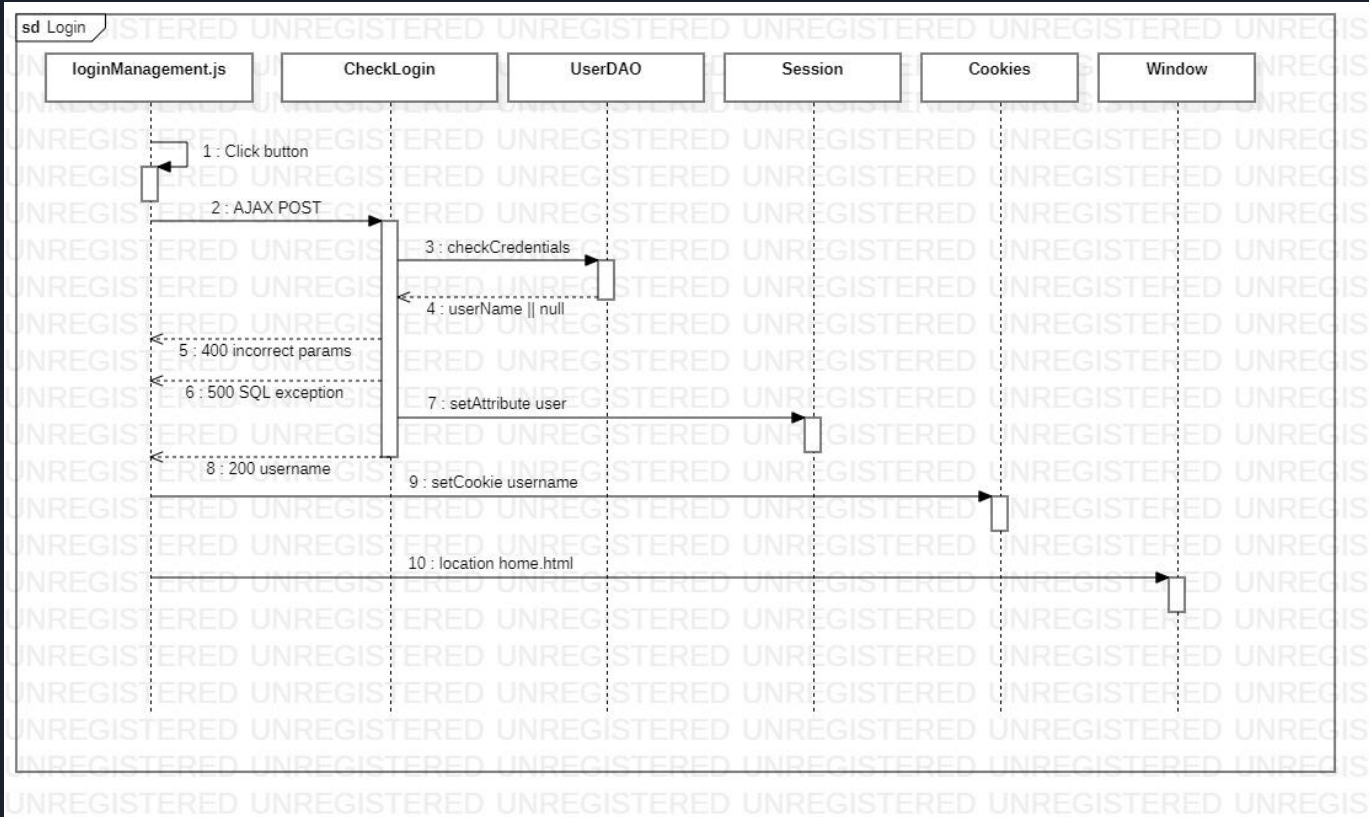
Right part of the page

- hideRightPart()
- showOpenAuction(auctionId)
- showOutDatedAuction(auctionId)
- showClosedAuction(auctionId)
- showPurchaseAuction(auctionId)

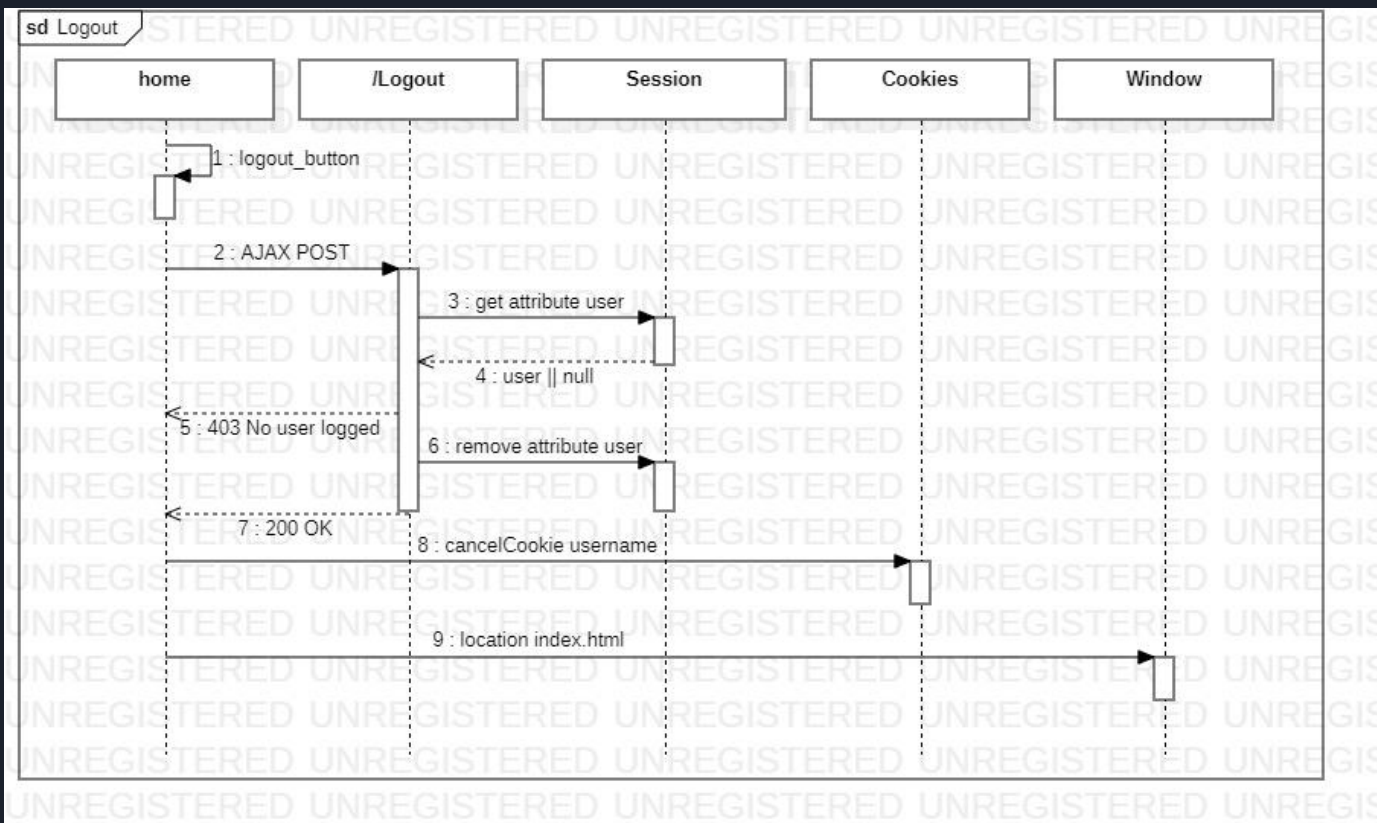
Sequence diagrams



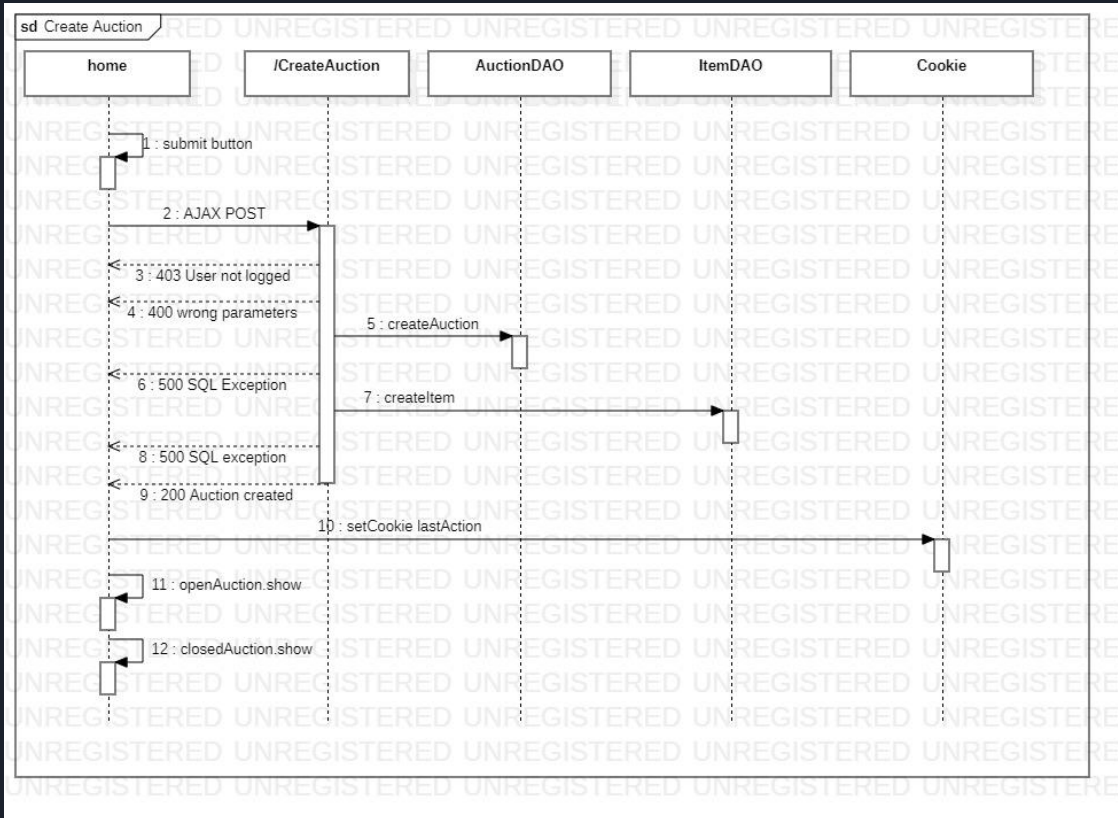
Login



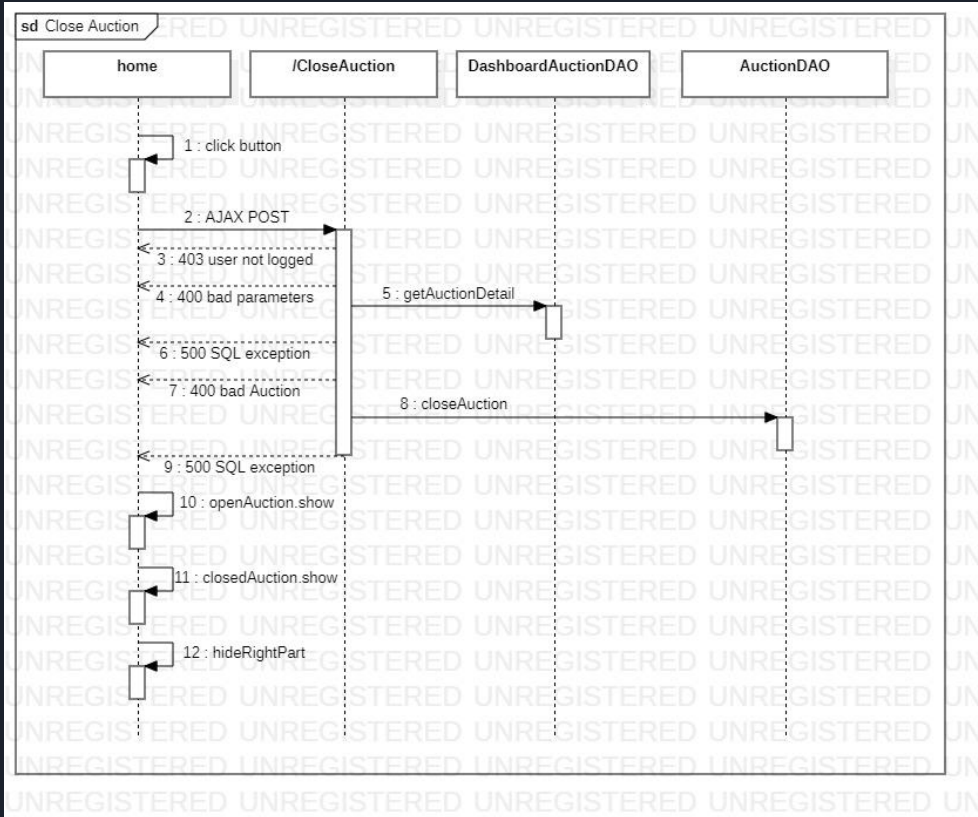
Logout



Create Auction



Close Auction



Add Offer

