Data requirement analysis

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 due form, una per creare un nuovo articolo e una per creare una nuova asta per vendere gli articoli dell’utente. ll primo form inserisce nuovi articoli nel database e il secondo mostra l'elenco degli articoli disponibili nel database e dà la possibilità di selezionarne più di uno. Un articolo ha codice, nome, descrizione, immagine e prezzo. Un’asta comprende uno o più articoli messi in vendita, il prezzo iniziale dell’insieme di articoli, il rialzo minimo di ogni offerta (espresso come un numero intero di euro) e una scadenza (data e ora, es. 19-04-2021 alle 24:00). Il prezzo iniziale dell’asta è ottenuto come somma del prezzo degli articoli compresi nell’offerta. Lo stesso articolo non può essere incluso in aste diverse. Una volta venduto, un articolo non deve essere più disponibile per l’inserimento in ulteriori aste. La lista delle aste è ordinata per data+ora crescente. L’elenco riporta: codice e nome degli articoli compresi nell’asta, offerta massima, tempo mancante (numero di giorni e ore) tra il momento (data ora) 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 tutti 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 dall’utente e non ci si occupi della chiusura automatica di aste dopo la scadenza). Se l’asta è chiusa, la pagina riporta tutti i dati dell’asta, il nome dell’aggiudicatario, il prezzo finale e l’indirizzo (fisso) 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 è aggiornata e mostra un elenco di aste aperte (la cui scadenza è posteriore alla data e ora dell’invio) per cui la parola chiave compare nel nome o nella descrizione di almeno uno degli articoli dell’asta. La lista è ordinata in modo decrescente in base al tempo (numero di giorni e ore) mancante alla chiusura. Cliccando su un’asta aperta compare la pagina OFFERTA che mostra i dati degli articoli, 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 di un importo pari almeno al rialzo minimo. 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 degli articoli e il prezzo finale.

Nota: il design del database non cambia tra versione pure HTML e RIA

Database design(DB ACID compliant)

A diagram of a computer

Description automatically generated

SQL Code

CREATE TABLE IF NOT EXISTS articolo (  
 id INTEGER PRIMARY KEY AUTO\_INCREMENT,  
 nome VARCHAR(200) NOT NULL,  
 descrizione VARCHAR(1000),  
 immagine blob,  
 prezzo INTEGER NOT NULL CHECK (prezzo > 0 AND prezzo < 500000),  
 venditore VARCHAR(100) NOT NULL,  
 FOREIGN KEY (venditore) REFERENCES utente(username),  
 CHECK(*length*(nome) >= 3)  
);  
  
CREATE TABLE IF NOT EXISTS asta (  
 id INTEGER PRIMARY KEY AUTO\_INCREMENT,  
 nome VARCHAR(200) NOT NULL,  
 descrizione VARCHAR(1000),  
 immagine blob,  
 scadenza VARCHAR(50) NOT NULL,  
 rialzo\_minimo INTEGER NOT NULL CHECK (rialzo\_minimo > 0 AND rialzo\_minimo < 100000),  
 venditore VARCHAR(100) NOT NULL,  
 aggiudicatario VARCHAR(100),  
 FOREIGN KEY (venditore) REFERENCES utente(username),  
 FOREIGN KEY (aggiudicatario) REFERENCES utente(username) ON DELETE SET NULL,  
 CHECK(*length*(nome) >= 4),  
 CHECK(*length*(scadenza) >= 4)  
);  
  
CREATE TABLE IF NOT EXISTS asta\_articoli (  
 id\_asta INTEGER NOT NULL,  
 id\_articolo INTEGER NOT NULL,  
 PRIMARY KEY (id\_asta, id\_articolo),  
 FOREIGN KEY (id\_asta) REFERENCES asta(id),  
 FOREIGN KEY (id\_articolo) REFERENCES articolo(id)  
);  
  
CREATE TABLE IF NOT EXISTS offerta (  
 id INTEGER PRIMARY KEY AUTO\_INCREMENT,  
 id\_asta INTEGER NOT NULL,  
 username VARCHAR(100) NOT NULL,  
 prezzo\_offerto INTEGER NOT NULL CHECK (prezzo\_offerto > 0 AND prezzo\_offerto < 1000000),  
 data\_offerta VARCHAR(50) NOT NULL DEFAULT *CURRENT\_TIMESTAMP*,  
 FOREIGN KEY (id\_asta) REFERENCES asta(id),  
 FOREIGN KEY (username) REFERENCES utente(username),  
 CHECK(*length*(data\_offerta) >= 4)  
);  
  
CREATE TABLE IF NOT EXISTS utente (  
 username VARCHAR(100) PRIMARY KEY,  
 pass\_hash VARCHAR(250) NOT NULL,  
 nome VARCHAR(100) NOT NULL,  
 cognome VARCHAR(100) NOT NULL,  
 indirizzo VARCHAR(400) NOT NULL,  
 CHECK(*length*(username) >= 1),  
 CHECK(*length*(pass\_hash) >= 10),  
 CHECK(*length*(nome) >= 2),  
 CHECK(*length*(cognome) >= 2),  
 CHECK(*length*(indirizzo) >= 6)  
);

Application requirements analysis

Pages(views), view components, events(user), actions(server)

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 due form, una per creare un nuovo articolo(submit form) e una per creare una nuova asta(submit form) per vendere gli articoli dell’utente. ll primo form inserisce nuovi articoli nel database e il secondo mostra l'elenco degli articoli disponibili nel database e dà la possibilità di selezionarne più di uno(db update). Un articolo ha codice, nome, descrizione, immagine e prezzo. Un’asta comprende uno o più articoli messi in vendita, il prezzo iniziale dell’insieme di articoli, il rialzo minimo di ogni offerta (espresso come un numero intero di euro) e una scadenza (data e ora, es. 19-04-2021 alle 24:00). Il prezzo iniziale dell’asta è ottenuto come somma del prezzo degli articoli compresi nell’offerta. Lo stesso articolo non può essere incluso in aste diverse. Una volta venduto, un articolo non deve essere più disponibile per l’inserimento in ulteriori aste. La lista delle aste è ordinata per data+ora crescente. L’elenco riporta: codice e nome degli articoli compresi nell’asta, offerta massima, tempo mancante (numero di giorni e ore) tra il momento (data ora) 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 tutti 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(click button)(db update) (si ignori il caso di aste scadute ma non chiuse dall’utente e non ci si occupi della chiusura automatica di aste dopo la scadenza). Se l’asta è chiusa, la pagina riporta tutti i dati dell’asta, il nome dell’aggiudicatario, il prezzo finale e l’indirizzo (fisso) 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 è aggiornata e mostra un elenco di aste aperte (la cui scadenza è posteriore alla data e ora dell’invio) per cui la parola chiave compare nel nome o nella descrizione di almeno uno degli articoli dell’asta. La lista è ordinata in modo decrescente in base al tempo (numero di giorni e ore) mancante alla chiusura. Cliccando su un’asta aperta compare la pagina OFFERTA che mostra i dati degli articoli, 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 di un importo pari almeno al rialzo minimo. 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 degli articoli e il prezzo finale.

**JavaScript additional requirements:**

-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.

Completamento delle specifiche:

Il login significa utente e password

I prezzi e le offerte minime sono strettamente positive, e hanno un limite massimo.

Aggiunta di un navbar che da’ all’utente l’opzione di tornare alla home, login, cerca aste, le mie aste, oltre anche al logout.

Le pagine che hanno un’asta fanno vedere anche l’autore.

Se utente non loggato prova ad andare alla pagina di vendo, verra’ indirizzato al login.

Gli errori e successi vengono visualizzati nella pagina.

MVC - Struttura dell'applicazione

**Beans**

-articolo

-utente

-Asta

-Articolo

**Controller**

-AcquistoServlet

-DettaglioAstaServlet

-HomeServlet

-LoginServlet

-LogoutServlet

-OffertaServlet

-VendoServlet

**Views**

-articoloCard

-astaCard

-messaggi

-navbar

-acquisto

-dettaglioAsta

-errore

-home

-login

-offerta

-vendo

**Accesso ai dati - Struttura dell'applicazione**

**DA COMPLETARE**

**Dao**

-DbManager

Da completare

(DAO nome) 🡨------------

Dao nome1 metodo 1 🡨------------

IFML DIAGRAM

A diagram of a computer

Description automatically generated

I diagrammi di sequenza sono semplificati: gli errori sono raggruppati tutti in ramo else per clarity, e le frecce di ritorno dal dbManager non sono mostrate.  
  
title Login Process  
 participant User  
 participant Browser  
 participant LoginServlet  
 participant DbManager  
 participant PasswordUtils  
 participant Session  
  
 User->>Browser: Enter login credentials  
 Browser->>LoginServlet: POST /login (username, password)  
 LoginServlet->>DbManager: getUtente(username)  
  
 alt Success  
 DbManager-->>LoginServlet: Return Utente object  
 LoginServlet->>PasswordUtils: verifyPassword(password, passwordHash)  
 alt Password valid  
 PasswordUtils-->>LoginServlet: true  
 LoginServlet->>Session: setAttribute("user", utente)  
 LoginServlet->>Session: setAttribute("isLoggedIn", true)  
 LoginServlet-->>Browser: Redirect to /home  
 Browser-->>User: Display home page  
 else Invalid credentials  
 PasswordUtils-->>LoginServlet: false  
 LoginServlet-->>Browser: Forward to login.jsp with error  
 Browser-->>User: Display error message  
 end  
 else Error or user not found  
 DbManager-->>LoginServlet: SQLException or null  
 LoginServlet-->>Browser: Forward to login.jsp with error  
 Browser-->>User: Display error message  
 end  
  
title Logout Process  
 participant User  
 participant Navbar  
 participant Browser  
 participant LogoutServlet  
 participant Session  
  
 Note over Navbar: Displays "Logout" link when user is logged in  
 User->>Navbar: Click "Logout" link  
 Navbar->>Browser: Request /logout  
 Browser->>LogoutServlet: GET /logout  
 LogoutServlet->>LogoutServlet: request.getSession(false)  
  
 alt Session exists  
 LogoutServlet->>Session: invalidate()  
 Session-->>LogoutServlet: Session terminated  
 else Session doesn't exist  
 Note over LogoutServlet: No action needed  
 end  
  
 LogoutServlet-->>Browser: Redirect to /login  
Browser-->>User: Display login page  
  
title Navigation to vendo page  
 participant User  
 participant JSP  
 participant VendoServlet  
 participant DbManager  
  
 User->>JSP: Click "Vendi un articolo" on home page  
 JSP->>VendoServlet: GET request to /vendo  
 VendoServlet->>VendoServlet: Check if user is logged in  
  
 alt User not logged in  
 VendoServlet-->>User: Redirect to login.jsp  
 else Error  
 VendoServlet-->>JSP: Forward to vendo.jsp with error(s)  
 JSP-->>User: Display error message(s)  
 else Success  
 VendoServlet->>DbManager: getArticoliDisponibiliPerUtente(username)  
 VendoServlet->>DbManager: getAsteUtente(username, false)  
 VendoServlet->>DbManager: getAsteUtente(username, true)  
 VendoServlet-->>JSP: Forward to vendo.jsp with attributes  
 JSP-->>User: Display vendo.jsp with forms and data  
 end  
  
title Create new article flow  
  
 participant User  
 participant JSP  
 participant VendoServlet  
 participant DbManager  
  
 User->>JSP: Submit "Nuovo Articolo" form  
 JSP->>VendoServlet: POST to /vendo with action=createArticolo  
 VendoServlet->>VendoServlet: Extract form parameters  
  
 alt Success  
 VendoServlet->>DbManager: inserisciArticolo(nome, descrizione, imageStream, prezzo, username)  
 VendoServlet-->>User: Redirect to /vendo with success message  
  
 else Error  
 VendoServlet-->>User: Redirect to /vendo with error message(s)  
 end  
  
title Create new auction flow  
 participant User  
 participant JSP  
 participant VendoServlet  
 participant DbManager  
  
 User->>JSP: Submit form  
 JSP->>VendoServlet: POST to /vendo with action=createAsta  
 VendoServlet->>VendoServlet: Extract form parameters  
  
 alt Success  
 loop For each selected article  
 VendoServlet->>DbManager: getPrezzoArticolo(id)  
 end  
 VendoServlet->>VendoServlet: Create Asta object with collected data  
 VendoServlet->>DbManager: inserisciAsta(asta)  
 DbManager-->>VendoServlet: Confirmation  
 VendoServlet-->>User: Redirect to /vendo with success message  
 else Error  
 VendoServlet-->>User: Redirect to /vendo with error message(s)  
 end  
  
title View Auction Details  
 participant User  
 participant JSP  
 participant DettaglioAstaServlet  
 participant DbManager  
  
 User->>JSP: Click on auction link  
 JSP->>DettaglioAstaServlet: GET request to /dettaglioAsta?id=X  
 DettaglioAstaServlet->>DettaglioAstaServlet: Check if user is logged in  
  
 alt User logged in  
 DettaglioAstaServlet->>DbManager: getAstaById(astaId)  
  
  
 alt Auction is closed (has winner)  
 DettaglioAstaServlet->>DbManager: getUtente(asta.getOffertaVincitrice().getUsername())  
  
 end  
  
 DettaglioAstaServlet-->>JSP: Forward to dettaglioAsta.jsp with attributes  
 JSP-->>User: Display auction details page  
 else User not logged in  
 DettaglioAstaServlet-->>User: Redirect to login.jsp  
 else Error  
 DettaglioAstaServlet-->>JSP: Forward to dettaglioAsta.jsp with error(s)  
 JSP-->>User: Display error message(s)  
 end  
  
title Close Auction  
 participant User  
 participant JSP  
 participant DettaglioAstaServlet  
 participant DbManager  
  
 User->>JSP: Click "Chiudi Asta" button  
 JSP->>DettaglioAstaServlet: POST to /dettaglioAsta with id and aggiudicatario  
  
 alt Success  
 DettaglioAstaServlet->>DbManager: chiudiAsta(astaId, aggiudicatario)  
 DettaglioAstaServlet->>DbManager: getAstaById(astaId)  
 DbManager-->>DettaglioAstaServlet: Return updated Asta object  
  
 alt Auction has winner  
 DettaglioAstaServlet->>DbManager: getUtente(asta.getOffertaVincitrice().getUsername())  
 end  
  
 DettaglioAstaServlet-->>User: Redirect to dettaglioAsta?id=X with success message  
 else Error  
 DettaglioAstaServlet-->>User: Redirect to dettaglioAsta?id=X with error message  
 end  
  
title Search and Display Auctions  
 participant User  
 participant JSP  
 participant AcquistoServlet  
 participant DbManager  
  
 User->>JSP: Visit acquisto page or submit search  
 JSP->>AcquistoServlet: GET request to /acquisto?parolaChiave=X  
 AcquistoServlet->>AcquistoServlet: Extract parolaChiave parameter  
  
 alt Success with search keyword  
 AcquistoServlet->>DbManager: getAstePerParolaChiave(parolaChiave, now)  
 AcquistoServlet->>AcquistoServlet: Sort auctions by remaining time  
  
 alt User is logged in  
 AcquistoServlet->>DbManager: getAsteVinteDaUtente(username)  
 end  
  
 AcquistoServlet-->>JSP: Forward to acquisto.jsp with attributes  
 JSP-->>User: Display open auctions and won auctions  
 else No search keyword  
 AcquistoServlet-->>JSP: Forward to acquisto.jsp with empty auctions list  
 JSP-->>User: Display search form only  
 else Error  
 AcquistoServlet-->>JSP: Forward to acquisto.jsp with error message(s)  
 JSP-->>User: Display error message(s)  
 end  
  
title View Auction for Offering  
 participant User  
 participant JSP  
 participant OffertaServlet  
 participant DbManager  
  
 User->>JSP: Click on auction to make offer  
 JSP->>OffertaServlet: GET /offerta?id=X  
  
 alt Success  
 OffertaServlet->>DbManager: getAstaById(astaId)  
 OffertaServlet-->>JSP: Forward to offerta.jsp with asta attribute  
 JSP-->>User: Display auction details with offer form  
 else Invalid auction ID  
 OffertaServlet-->>User: Redirect to acquisto.jsp  
 else Error  
 OffertaServlet-->>JSP: Forward to offerta.jsp with error message(s)  
 JSP-->>User: Display error message(s)  
 end  
  
title Submit Offer  
 participant User  
 participant JSP  
 participant OffertaServlet  
 participant DbManager  
  
 User->>JSP: Submit offer form with price  
 JSP->>OffertaServlet: POST /offerta with astaId and prezzo  
  
 alt User logged in  
 OffertaServlet->>DbManager: getAstaById(astaId)  
  
 OffertaServlet->>OffertaServlet: Validate offer amount  
  
 alt Valid offer amount  
 OffertaServlet->>OffertaServlet: Create new Offerta object  
 OffertaServlet->>OffertaServlet: asta.newOfferta(offerta)  
 OffertaServlet->>DbManager: registraOfferta(astaId, offerta)  
  
 OffertaServlet-->>User: Redirect to /offerta?id=X with success message  
 else Invalid offer amount  
 OffertaServlet-->>User: Redirect to /offerta?id=X with error message  
 end  
 else User not logged in  
 OffertaServlet-->>User: Redirect to login.jsp  
 else Error (invalid astaId or database error)  
 OffertaServlet-->>User: Redirect to "acquista" or /offerta?id=X with error  
 end

RIA MVC - Struttura dell'applicazione

**Beans**

-articolo

-utente

-Asta

-Articolo

**Controller**

-AcquistoApiServlet

-DettaglioAstaApiServlet

-LoginServlet

-LogoutServlet

-OffertaApiServlet

-VendoServlet

**Views**

-index.html

-articoloCard.js

-astaCard.js

-navbar.js

-messaggi.jsp

-messageManager.js

-dettaglioAsta.js

-login.jsp

-offerta.js

-dettaglioAsta.js

-gestioneAste.js

-ricercaAste.js

-app.js

Accesso ai dati - Struttura dell'applicazione

DAO

TODO

TODO

TODO

TODO

TODO

TODO

TODO

TODO

TODO

IFML

TODO

TODO

TODO

TODO

TODO

|  |  |  |  |
| --- | --- | --- | --- |
| Client |  | Server |  |
| Evento | **Azione** | **Evento** | **Azione** |
| Login form -> submit | Nullity check | Post username-password | Check credentials |
| Index -> Ricerca aste -> Load | View update | Get (username) | Return latest, won auctions if applicable |
| Index -> Ricerca aste -> Submit query | View update | Get (username) | Return latest, won and matching auctions if applicable |
| Index -> Ricerca aste -> click on auction | View update | Get (auction id) | Return auction info |
| Index -> Ricerca aste -> click on auction | Save info to cookie | - | - |
| Navbar -> Logout |  | Get | Terminate session |
| Index -> gestioneAste -> Load | View update | Get (username) | Return active, closed auctions, stored items if applicable |
| Index -> gestioneAste -> item form submit | Nullity check | Post (item data) | Db update |
| Index -> gestioneAste -> auction form submit | Nullity check | Post (actions data) | Db update |
| Index -> gestioneAste -> click on auction | View update | Get (auction id) | Return auction data, and winner data if applicable |
| Index -> dettagioAsta -> close auction | Show confirmation | Post | Db update |
| Index -> Offerta -> make offer | Nullity check | Post (offer amount) | Db update |

|  |  |  |  |
| --- | --- | --- | --- |
| Client |  | Server |  |
| Evento | **Controller** | **Evento** | **Controller** |
| Login form -> submit | Send post | Post (username, password) | LoginServlet |
| Index -> Ricerca asta -> Load | Function loadData | Get (query, visitedId) | AcquistoApiServlet |
| Index -> Ricerca aste -> click on auction | AstaCard.clickhandler() | Get(astaId) | OffertaApiServlet |
| Index -> Ricerca aste -> click on auction | Function registraAstaVisitata | - | - |
| Navbar -> Logout | Send get | Get | LogoutServlet |
| Index -> gestioneAste -> Load | Function loadData | Get (username) | VendoServlet |
| Index -> gestioneAste -> item form submit | Function handleFormArticolo | Post (item data) | VendoServlet |
| Index -> gestioneAste -> auction form submit | Function handleFormAsta | Post (action data) | VendoServlet |
| Index -> gestioneAste -> auction form submit | Function handleFormAsta  (Save info to cookie) | - | - |
| Index -> gestioneAste -> click on auction | AstaCard.clickhandler() | Get (auction id) | DettaglioAstaApiServlet |
| Index -> dettaglioAsta -> close auction | Function handleChiudiAsta | Post | DettaglioAstaApiServlet |
| Index -> Offerta -> make offer | Function handleFormOfferta | Pos (offer amount) | OffertaApiServlet |

title Login Process  
 participant "Client (User+Browser+App.js)" as Client  
 participant LoginServlet  
 participant DbManager  
 participant PasswordUtils  
 participant Session  
 participant AuthStatusServlet  
  
 Client->>LoginServlet: POST /login (username, password)  
 LoginServlet->>DbManager: getUtente(username)  
 alt Success  
 DbManager-->>LoginServlet: Return Utente object  
 LoginServlet->>PasswordUtils: verifyPassword(password, passwordHash)  
 alt Password valid  
 LoginServlet->>Session: setAttribute("user", utente)  
 LoginServlet->>Session: setAttribute("isLoggedIn", true)  
 LoginServlet-->>Client: Redirect to /index.html  
 Client->>Client: App.init()  
 Client->>AuthStatusServlet: GET /auth/status  
 AuthStatusServlet->>Session: getAttribute("user")  
 Session-->>AuthStatusServlet: Return user object  
 AuthStatusServlet-->>Client: Return user data  
 Client->>Client: Determine initial route  
 alt First visit or route check  
 Client->>Client: Check cookies (primaVisita, ultimaAzione)  
 Client->>Client: Set appropriate defaultRoute based on cookie values  
 end  
 Client->>Client: navigateTo(determinedRoute)  
 else Invalid credentials  
 LoginServlet-->>Client: Forward to login.jsp with error  
 end  
 else Error or user not found  
 DbManager-->>LoginServlet: SQLException or null  
 LoginServlet-->>Client: Forward to login.jsp with error  
 end  
  
  
 title Logout Process  
 participant "Client (User+Browser+App.js)" as Client  
 participant LogoutServlet  
 participant Session  
 Client->>LogoutServlet: GET /logout  
 LogoutServlet->>LogoutServlet: request.getSession(false)  
 alt Session exists  
 LogoutServlet->>Session: invalidate()  
 Session-->>LogoutServlet: Session terminated  
 else Session doesn't exist  
 note over LogoutServlet: No action needed  
 end  
 LogoutServlet-->>Client: Redirect to /login.jsp  
 Client->>Client: Display login page  
  
  
 title Load ricercaAste Page  
 participant "Client (User+Browser+App.js)" as Client  
 participant RicercaAstePage  
 participant "AuthStatusServlet" as API  
 participant AcquistoServlet  
 participant DbManager  
 Client->>API: GET /auth/status  
 alt User is logged in  
 API-->>Client: Return user data (null if not found)  
 else User is not logged in  
 API-->>Client: Return userdata = null  
 end  
 RicercaAstePage->>RicercaAstePage: loadData(parolaChiave)  
 RicercaAstePage->>API: GET /acquisto with params  
 API->>AcquistoServlet: GET request  
 AcquistoServlet->>AcquistoServlet: Extract parolaChiave parameter  
 alt User is logged in  
 AcquistoServlet->>DbManager: getAstePerParolaChiave(parolaChiave, now)  
 AcquistoServlet->>DbManager: getAsteVinteDaUtente(username)  
 AcquistoServlet->>AcquistoServlet: Process asteVisitate from cookies  
 AcquistoServlet-->>API: Return JSON {asteAperte, asteAggiudicate, asteVisitate}  
 else User not logged in  
 AcquistoServlet->>DbManager: getAstePerParolaChiave(parolaChiave, now)  
 AcquistoServlet-->>API: Return JSON {asteAperte}  
 end  
 API-->>RicercaAstePage: Return processed data  
 RicercaAstePage->>RicercaAstePage: Update page state  
 RicercaAstePage->>RicercaAstePage: Render HTML with auction data  
 RicercaAstePage->>RicercaAstePage: Attach event handlers  
 RicercaAstePage-->>Client: Display auctions page  
  
  
title View Auction Details Flow  
participant "Client(Browser+RicercaAstePage+App.js)" as Client  
participant "OffertaPage" as OffertaPage  
participant "OffertaApiServlet" as API  
participant "DbManager" as DB  
  
Client->>OffertaPage: navigateTo('offerta', {id: astaId})  
  
OffertaPage->>OffertaPage: Parse AstaId  
OffertaPage->>API: GET /api/offerta?id=astaId  
API->>API: Parse astaId parameter  
API->>DB: getAstaById(astaId)  
DB-->>API: Return Asta object  
API-->>OffertaPage: Return JSON {asta: astaObject}  
  
OffertaPage->>OffertaPage: Process auction data  
  
alt User is logged in  
 OffertaPage->>OffertaPage: registraAstaVisitata(astaId, username)  
 OffertaPage->>OffertaPage: Update asteVisitate cookie  
end  
  
alt User is logged in AND NOT auction.chiusa AND user != auction.venditore  
  
 OffertaPage->>OffertaPage: Setup bid form with minimum offer validation  
 OffertaPage->>OffertaPage: Attach submit event handler  
else Auction is closed  
 OffertaPage->>OffertaPage: Display "Quest'asta è chiusa" message  
else User is not logged in  
 OffertaPage->>OffertaPage: Display "Accedi per fare un'offerta" message  
else User is the seller  
 OffertaPage->>OffertaPage: Display "Non puoi fare offerte sulla tua stessa asta" message  
end  
  
OffertaPage->>OffertaPage: Render auction details, articles and offers table  
OffertaPage-->>Client: Display complete auction page  
  
  
title Submit Auction Bid Flow  
participant "User" as User  
participant "OffertaPage" as OffertaPage  
participant "API" as ClientAPI  
participant "OffertaApiServlet" as Servlet  
participant "DbManager" as DB  
  
User->>OffertaPage: Submit bid form with price  
OffertaPage->>OffertaPage: handleFormOfferta(event)  
OffertaPage->>OffertaPage: Client-side validation  
alt Validation fails  
 OffertaPage-->>User: Display error message  
else Validation passes  
 OffertaPage->>ClientAPI: API.post('/offerta', formData)  
 ClientAPI->>Servlet: POST /api/offerta (astaId, prezzo)  
  
 Servlet->>Servlet: Check user authentication  
 alt User not authenticated  
 Servlet-->>ClientAPI: Return 401 Unauthorized  
 ClientAPI-->>OffertaPage: Throw error  
 OffertaPage-->>User: Display authentication error  
 else User authenticated  
 Servlet->>DB: getAstaById(astaId)  
 DB-->>Servlet: Return Asta object  
  
 Servlet->>Servlet: Calculate minimum bid amount  
 alt Bid amount too low  
 Servlet-->>ClientAPI: Return 400 Bad Request  
 ClientAPI-->>OffertaPage: Throw error  
 OffertaPage-->>User: Display "offer too low" error  
 else Bid amount valid  
 Servlet->>DB: registraOfferta(astaId, offerta)  
 DB-->>Servlet: Success  
 Servlet-->>ClientAPI: Return success JSON  
 ClientAPI-->>OffertaPage: Return success response  
  
 OffertaPage->>OffertaPage: MessageManager.showSuccess()  
 OffertaPage->>OffertaPage: loadData(astaId)  
 OffertaPage->>ClientAPI: API.get('/offerta', {id: astaId})  
 ClientAPI->>Servlet: GET /api/offerta?id=astaId  
 Servlet->>DB: getAstaById(astaId)  
 DB-->>Servlet: Return updated Asta with new offer  
 Servlet-->>ClientAPI: Return updated auction data  
 ClientAPI-->>OffertaPage: Return updated data  
 OffertaPage->>OffertaPage: render(astaId)  
 OffertaPage-->>User: Display updated auction page with new bid  
 end  
 end  
end  
  
  
title Load GestioneAste Page  
 participant "Client (User+Browser+App.js)" as Client  
 participant GestioneAstePage  
 participant "API+VendoServlet" as ApiServlet  
 participant "DbManager" as DB  
 participant "Session" as Session  
  
 Client->>GestioneAstePage: navigateTo('gestioneAste')  
  
 GestioneAstePage->>ApiServlet: API.get('/vendo')  
  
 ApiServlet->>Session: getAttribute("user")  
 Session-->>ApiServlet: Return user object  
  
 alt User is logged in  
 ApiServlet->>DB: getArticoliDisponibiliPerUtente(username)  
 DB-->>ApiServlet: Return available articles  
 ApiServlet->>DB: getAsteUtente(username, false)  
 DB-->>ApiServlet: Return open auctions  
 ApiServlet->>DB: getAsteUtente(username, true)  
 DB-->>ApiServlet: Return closed auctions  
 ApiServlet-->>GestioneAstePage: Return JSON {articoliUtente, asteAperte, asteChiuse}  
 else User not logged in  
 ApiServlet-->>GestioneAstePage: Return 401 Unauthorized  
 end  
  
 GestioneAstePage->>GestioneAstePage: Update page state  
 GestioneAstePage->>GestioneAstePage: Render HTML with articles and auctions  
 GestioneAstePage->>GestioneAstePage: Attach event handlers for forms  
 GestioneAstePage-->>Client: Display auction management page  
  
  
title Submit Auction Bid Flow  
participant "User" as User  
participant "OffertaPage" as OffertaPage  
participant "API+OffertaApiServlet" as ApiServlet  
participant "DbManager" as DB  
participant "Session" as Session  
  
User->>OffertaPage: Submit bid form with price  
OffertaPage->>OffertaPage: handleFormOfferta(event)  
OffertaPage->>OffertaPage: Client-side validation  
OffertaPage->>ApiServlet: API.post('/offerta', formData)  
  
ApiServlet->>Session: getAttribute("user")  
Session-->>ApiServlet: Return user object  
  
alt Success path  
 ApiServlet->>ApiServlet: Parse parameters (astaId, prezzo)  
 ApiServlet->>DB: getAstaById(astaId)  
 DB-->>ApiServlet: Return Asta object  
 ApiServlet->>ApiServlet: Calculate minimum bid amount  
 ApiServlet->>ApiServlet: Validate bid price  
 ApiServlet->>DB: registraOfferta(astaId, offerta)  
 DB-->>ApiServlet: Success  
 ApiServlet-->>OffertaPage: Return success JSON  
 OffertaPage->>OffertaPage: MessageManager.showSuccess()  
 OffertaPage->>OffertaPage: loadData(astaId)  
 OffertaPage->>ApiServlet: API.get('/offerta?id=astaId')  
 ApiServlet->>DB: getAstaById(astaId)  
 DB-->>ApiServlet: Return updated Asta object  
 ApiServlet-->>OffertaPage: Return updated auction data  
 OffertaPage->>OffertaPage: render(astaId)  
 OffertaPage-->>User: Display updated auction page with new bid  
else Error conditions (400/401)  
  
 note over ApiServlet: 400: Invalid parameters, price too low, DB error  
 note over ApiServlet: 401: User not authenticated  
 ApiServlet->>ApiServlet: Add appropriate error message to errors list  
  
 ApiServlet-->>OffertaPage: Return error response with messages  
 OffertaPage->>OffertaPage: MessageManager.showErrors()  
 OffertaPage-->>User: Display appropriate error message  
end  
  
  
title Add New Auction Flow  
participant "User" as User  
participant "GestioneAstePage" as GestioneAstePage  
participant "API+VendoServlet" as ApiServlet  
participant "DbManager" as DB  
participant "Session" as Session  
  
User->>GestioneAstePage: Submit form  
GestioneAstePage->>GestioneAstePage: handleFormAsta(event)  
GestioneAstePage->>GestioneAstePage: Validate form (check articles selected)  
GestioneAstePage->>GestioneAstePage: Prepare FormData with action=createAsta  
GestioneAstePage->>ApiServlet: API.post('/vendo', formData)  
  
ApiServlet->>Session: getAttribute("user")  
  
  
alt Success Path  
 ApiServlet->>ApiServlet: Parse parameters (nome, descrizione, immagine, etc.)  
 ApiServlet->>ApiServlet: Parse articoliIds from request  
 ApiServlet->>DB: Get prices for all selected articles  
  
 ApiServlet->>ApiServlet: Calculate initial price (sum of article prices)  
 ApiServlet->>ApiServlet: Create Asta object with all parameters  
 ApiServlet->>DB: inserisciAsta(asta)  
 DB-->>ApiServlet: Success  
 ApiServlet-->>GestioneAstePage: Return success response  
 GestioneAstePage->>GestioneAstePage: MessageManager.showSuccess()  
 GestioneAstePage->>GestioneAstePage: Set cookie "ultimaAzione=creazione-asta"  
 GestioneAstePage->>GestioneAstePage: loadData()  
 GestioneAstePage->>ApiServlet: API.get('/vendo')  
 ApiServlet->>DB: getArticoliDisponibiliPerUtente(username)  
  
 ApiServlet->>DB: getAsteUtente(username, false)  
  
 ApiServlet->>DB: getAsteUtente(username, true)  
  
 ApiServlet-->>GestioneAstePage: Return updated data  
 GestioneAstePage->>GestioneAstePage: render()  
 GestioneAstePage-->>User: Display updated page with new auction  
else Error Conditions  
 note over ApiServlet: Possible errors:  
 note over ApiServlet: 401: User not authenticated  
 note over ApiServlet: 400: Missing required fields (name, description, etc.)  
 note over ApiServlet: 400: No articles selected for auction  
 note over ApiServlet: 400: Invalid auction parameters (dates, prices)  
 note over ApiServlet: 500: Database errors during insertion  
  
 ApiServlet->>ApiServlet: Identify error type and add to errors list  
 ApiServlet->>ApiServlet: Set appropriate status code (400/401/500)  
 ApiServlet-->>GestioneAstePage: Return error response with messages  
 GestioneAstePage->>GestioneAstePage: MessageManager.showErrors()  
 GestioneAstePage-->>User: Display appropriate error message  
end  
  
  
title View Auction Details Flow (DettaglioAsta)  
participant "User" as User  
participant "App.js" as App  
participant "DettaglioAstaPage" as DetailPage  
participant "API+DettaglioAstaApiServlet" as ApiServlet  
participant "DbManager" as DB  
participant "Session" as Session  
  
User->>App: Navigate to auction detail page  
App->>DetailPage: navigateTo('dettaglioAsta', {id: astaId})  
  
DetailPage->>ApiServlet: API.get('/dettaglioAsta', {id: astaId})  
  
ApiServlet->>Session: getAttribute("user")  
Session-->>ApiServlet: Return user object  
  
alt Success Path  
 ApiServlet->>ApiServlet: Validate user authentication  
 ApiServlet->>DB: getAstaById(astaId)  
  
 ApiServlet->>ApiServlet: Check if user.username == asta.venditore  
  
 alt Auction has winning offer  
 ApiServlet->>DB: getUtente(asta.getOffertaVincitrice().getUsername())  
  
 ApiServlet->>ApiServlet: Add winner to result  
 end  
  
 ApiServlet-->>DetailPage: Return JSON with asta and winner info  
  
 DetailPage->>App: Check current user (App.getCurrentUser())  
 App-->>DetailPage: Return current user  
 DetailPage->>DetailPage: Set isVenditore flag  
  
  
 alt Auction is closed and has winner  
 DetailPage->>DetailPage: Add winner section to HTML  
 end  
  
 alt User is seller and auction is open  
 DetailPage->>DetailPage: Add "Close Auction" form to HTML  
 end  
  
 DetailPage->>DetailPage: Prepare the page  
  
 DetailPage-->>User: Display complete auction details page  
else Error Conditions  
 note over ApiServlet: Possible errors:  
 note over ApiServlet: 401: User not authenticated  
 note over ApiServlet: 403: User is not the seller  
 note over ApiServlet: 400: Invalid auction ID  
 note over ApiServlet: 404: Auction not found  
 note over ApiServlet: 500: Database errors  
  
 ApiServlet->>ApiServlet: Identify error type  
 ApiServlet->>ApiServlet: Set appropriate status code  
 ApiServlet-->>DetailPage: Return error response  
 DetailPage->>DetailPage: MessageManager.showErrors()  
 DetailPage-->>User: Display appropriate error message  
end  
  
  
title Close Auction Flow  
participant "User" as User  
participant "DettaglioAstaPage" as DetailPage  
participant "API+DettaglioAstaApiServlet" as ApiServlet  
participant "DbManager" as DB  
participant "Session" as Session  
  
User->>DetailPage: Click "Chiudi Asta" button  
DetailPage->>DetailPage: Display confirmation dialog  
alt User cancels  
 DetailPage-->>User: Close dialog, no action taken  
else User confirms  
 DetailPage->>DetailPage: handleChiudiAsta(event)  
 DetailPage->>DetailPage: Prepare FormData with id and aggiudicatario  
 DetailPage->>ApiServlet: API.post('/dettaglioAsta', formData)  
  
 ApiServlet->>Session: getAttribute("user")  
 Session-->>ApiServlet: Return user object  
  
 alt Success Path  
 ApiServlet->>ApiServlet: Validate user authentication  
 ApiServlet->>ApiServlet: Parse auction ID and winner username  
 ApiServlet->>DB: chiudiAsta(astaId, aggiudicatario)  
 DB-->>ApiServlet: Success  
 ApiServlet->>DB: getAstaById(astaId)  
 DB-->>ApiServlet: Return updated Asta object  
  
 alt Auction has winning offer  
 ApiServlet->>DB: getUtente(asta.getOffertaVincitrice().getUsername())  
 DB-->>ApiServlet: Return winner user object  
 ApiServlet->>ApiServlet: Add winner to result  
 end  
  
 ApiServlet-->>DetailPage: Return success response  
 DetailPage->>DetailPage: MessageManager.showSuccess()  
 DetailPage->>DetailPage: Change the page to show the updated auction  
 DetailPage-->>User: Display updated auction details page  
 else Error Conditions  
 note over ApiServlet: Possible errors:  
 note over ApiServlet: 401: User not authenticated  
 note over ApiServlet: 403: User is not the seller  
 note over ApiServlet: 400: Invalid auction ID  
 note over ApiServlet: 400: Missing aggiudicatario  
 note over ApiServlet: 500: Database errors  
  
 ApiServlet->>ApiServlet: Identify error type  
 ApiServlet->>ApiServlet: Set appropriate status code  
 ApiServlet-->>DetailPage: Return error response  
 DetailPage->>DetailPage: MessageManager.showErrors()  
 DetailPage-->>User: Display appropriate error message  
 end  
end