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

A diagram of a computer

Description automatically generated

SQL Code

CREATE TABLE IF NOT EXISTS user (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 username TEXT NOT NULL UNIQUE,  
 password TEXT NOT NULL,  
 name TEXT NOT NULL,  
 surname TEXT NOT NULL,  
 address TEXT NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS item (  
 code INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT NOT NULL,  
 description TEXT NOT NULL,  
 image\_path TEXT NOT NULL,  
 price INTEGER NOT NULL,  
 owner\_id INTEGER NOT NULL,  
 is\_available BOOLEAN DEFAULT 1,  
 FOREIGN KEY (owner\_id) REFERENCES user(id)  
 ON UPDATE CASCADE ON DELETE NO ACTION  
);  
  
CREATE TABLE IF NOT EXISTS auction\_listing (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 seller\_id INTEGER NOT NULL,  
 initial\_price INTEGER NOT NULL,  
 min\_raise INTEGER NOT NULL,  
 deadline DATETIME NOT NULL,  
 closed BOOLEAN DEFAULT 0,  
 FOREIGN KEY (seller\_id) REFERENCES user(id)  
 ON UPDATE CASCADE ON DELETE NO ACTION  
);  
  
CREATE TABLE IF NOT EXISTS auction\_contains\_item (  
 auction\_id INTEGER NOT NULL,  
 item\_id INTEGER NOT NULL,  
 PRIMARY KEY (auction\_id, item\_id),  
 FOREIGN KEY (auction\_id) REFERENCES auction\_listing(id)  
 ON UPDATE CASCADE ON DELETE NO ACTION,  
 FOREIGN KEY (item\_id) REFERENCES item(code)  
 ON UPDATE CASCADE ON DELETE NO ACTION  
);  
  
CREATE TABLE IF NOT EXISTS bid (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 auction\_id INTEGER NOT NULL,  
 user\_id INTEGER NOT NULL,  
 price INTEGER NOT NULL,  
 timestamp TIMESTAMP DEFAULT *CURRENT\_TIMESTAMP*,  
 FOREIGN KEY (auction\_id) REFERENCES auction\_listing(id)  
 ON UPDATE CASCADE ON DELETE NO ACTION,  
 FOREIGN KEY (user\_id) REFERENCES user(id)  
 ON UPDATE CASCADE ON DELETE NO ACTION  
);

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.

SEQ diagrams  
  
LOGIN:

sequenceDiagram

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 Database error

DbManager-->>LoginServlet: SQLException

LoginServlet-->>Browser: Forward to login.jsp with database error

Browser-->>User: Display error message

else User found

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 Password invalid

PasswordUtils-->>LoginServlet: false

LoginServlet-->>Browser: Forward to login.jsp with "Invalid credentials" error

Browser-->>User: Display error message

end

else User not found

DbManager-->>LoginServlet: Return null

LoginServlet-->>Browser: Forward to login.jsp with "Invalid credentials" error

Browser-->>User: Display error message

end

LOGOUT

sequenceDiagram

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

Vendo 😊

sequenceDiagram

participant User as User

participant JSP as JSP Pages

participant VS as VendoServlet

participant DB as DbManager

participant Database as SQLite Database

*%% Navigation to vendo page*

User->>JSP: Click "Vendi un articolo" on home page

JSP->>VS: GET request to /vendo

VS->>VS: Check if user is logged in

alt User not logged in

VS-->>User: Redirect to login.jsp

else User logged in

VS->>DB: getArticoliDisponibiliPerUtente(username)

DB->>Database: SELECT from articolo

Database-->>DB: Return available articles

DB-->>VS: Return article list

VS->>DB: getAsteUtente(username, false)

DB->>Database: SELECT from asta (open auctions)

Database-->>DB: Return open auctions

DB-->>VS: Return open auctions list

VS->>DB: getAsteUtente(username, true)

DB->>Database: SELECT from asta (closed auctions)

Database-->>DB: Return closed auctions

DB-->>VS: Return closed auctions list

VS-->>JSP: Forward to vendo.jsp with attributes

JSP-->>User: Display vendo.jsp with forms and data

end

*%% Create new article flow*

User->>JSP: Fill "Nuovo Articolo" form

User->>JSP: Submit form

JSP->>VS: POST to /vendo with action=createArticolo

VS->>VS: Extract form parameters

VS->>DB: inserisciArticolo(nome, descrizione, imageStream, prezzo, username)

DB->>Database: INSERT INTO articolo

Database-->>DB: Confirmation

DB-->>VS: Confirmation

VS-->>User: Redirect to /vendo with success message

*%% Create new auction flow*

User->>JSP: Select articles and fill "Nuova Asta" form

User->>JSP: Submit form

JSP->>VS: POST to /vendo with action=createAsta

VS->>VS: Extract form parameters

loop For each selected article

VS->>DB: getPrezzoArticolo(id)

DB->>Database: SELECT prezzo FROM articolo

Database-->>DB: Return price

DB-->>VS: Return price

end

VS->>VS: Create Asta object with collected data

VS->>DB: inserisciAsta(asta)

DB->>Database: INSERT INTO asta

Database-->>DB: Return auction ID

loop For each article in auction

DB->>Database: INSERT INTO asta\_articoli

Database-->>DB: Confirmation

end

DB->>Database: INSERT INTO offerta (initial offer)

Database-->>DB: Confirmation

DB-->>VS: Confirmation

VS-->>User: Redirect to /vendo with success message

DettaglioAsta  
  
sequenceDiagram

participant User as User

participant JSP as JSP Pages

participant DAS as DettaglioAstaServlet

participant DB as DbManager

participant Database as SQLite Database

*%% View auction details flow*

User->>JSP: Click on auction link

JSP->>DAS: GET request to /dettaglioAsta?id=X

DAS->>DAS: Check if user is logged in

alt User not logged in

DAS-->>User: Redirect to login.jsp

else User logged in

DAS->>DB: getAstaById(astaId)

DB->>Database: SELECT from asta WHERE id = X

Database-->>DB: Return auction data

DB->>Database: SELECT from offerta WHERE id\_asta = X

Database-->>DB: Return offers data

DB->>Database: SELECT from articolo JOIN asta\_articoli

Database-->>DB: Return articles data

DB-->>DAS: Return complete Asta object

alt Auction is closed (has winner)

DAS->>DB: getUtente(asta.getOffertaVincitrice().getUsername())

DB->>Database: SELECT from utente

Database-->>DB: Return winner user data

DB-->>DAS: Return Utente object

end

DAS-->>JSP: Forward to dettaglioAsta.jsp with attributes

JSP-->>User: Display auction details page

end

*%% Close auction flow*

User->>JSP: Click "Chiudi Asta" button

JSP->>DAS: POST to /dettaglioAsta with id and aggiudicatario

DAS->>DB: chiudiAsta(astaId, aggiudicatario)

DB->>Database: UPDATE asta SET aggiudicatario = X WHERE id = Y

Database-->>DB: Confirmation

DB-->>DAS: Confirmation

DAS->>DB: getAstaById(astaId)

DB->>Database: SELECT from asta and related tables

Database-->>DB: Return updated auction data

DB-->>DAS: Return updated Asta object

DAS->>DB: getUtente(asta.getOffertaVincitrice().getUsername())

DB->>Database: SELECT from utente

Database-->>DB: Return winner user data

DB-->>DAS: Return Utente object

DAS-->>User: Redirect to dettaglioAsta?id=X with success message

Acquisto

sequenceDiagram

participant User

participant Browser

participant HomeJSP as Home Page (JSP)

participant NavbarJSP as Navbar (JSP)

participant AcquistoServlet

participant DbManager

participant AcquistoJSP as Acquisto Page (JSP)

participant MessaggiJSP as Messaggi (JSP)

participant AstaCardJSP as AstaCard (JSP)

User->>Browser: Navigate to home page

Browser->>HomeJSP: Request home page

HomeJSP-->>Browser: Display home page

Browser-->>User: Show home page

User->>Browser: Click "Acquisto" in navbar

Browser->>AcquistoServlet: GET /acquisto

AcquistoServlet->>AcquistoServlet: Check for search parameters (none)

alt User is logged in

AcquistoServlet->>DbManager: getAsteVinteDaUtente(username)

DbManager-->>AcquistoServlet: Return list of won auctions

end

AcquistoServlet->>AcquistoJSP: Forward request with attributes

AcquistoJSP->>NavbarJSP: Include navbar

NavbarJSP-->>AcquistoJSP: Return navbar content

AcquistoJSP->>MessaggiJSP: Include messages

MessaggiJSP-->>AcquistoJSP: Return messages content

alt User is logged in

AcquistoJSP->>AstaCardJSP: Include AstaCard for each won auction

AstaCardJSP-->>AcquistoJSP: Return auction card HTML

end

AcquistoJSP-->>Browser: Return complete HTML

Browser-->>User: Display Acquisto page with search form

User->>Browser: Enter search keyword "laptop"

User->>Browser: Click "Cerca" button

Browser->>AcquistoServlet: GET /acquisto?parolaChiave=laptop

AcquistoServlet->>DbManager: getAstePerParolaChiave("laptop", now)

DbManager->>DbManager: Execute SQL query with LIKE %laptop%

DbManager-->>AcquistoServlet: Return matching auctions

alt User is logged in

AcquistoServlet->>DbManager: getAsteVinteDaUtente(username)

DbManager-->>AcquistoServlet: Return list of won auctions

end

AcquistoServlet->>AcquistoJSP: Forward with search results and parameters

AcquistoJSP->>NavbarJSP: Include navbar

NavbarJSP-->>AcquistoJSP: Return navbar content

AcquistoJSP->>MessaggiJSP: Include messages

MessaggiJSP-->>AcquistoJSP: Return messages content

alt Search results found

AcquistoJSP->>AstaCardJSP: Include AstaCard for each open auction found

AstaCardJSP-->>AcquistoJSP: Return auction card HTML

end

alt User is logged in

AcquistoJSP->>AstaCardJSP: Include AstaCard for each won auction

AstaCardJSP-->>AcquistoJSP: Return auction card HTML

end

AcquistoJSP-->>Browser: Return complete HTML with search results

Browser-->>User: Display search results page