POLITECNICO
DI MILANO

Progetto di

Tecnologie Informatiche per il Web

a.a. 2024-2025

Prof. Fraternali Piero

Cristian Summa **Mathias Rodigari**

Sommario

Specifica

Versione HTML pura

Un'applicazione web consente la gestione di aste online.

Gli utenti accedono tramite login e possono vendere e acquistare articoli all'asta.

Ogni utente ha username, password, nome, cognome e indirizzo (quest'ultimo è usato per la spedizione degli articoli comperati).

Ogni articolo ha codice, nome, descrizione, immagine e prezzo.

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, uno per creare un nuovo articolo e uno per creare una nuova asta per vendere gli articoli dell'utente.
Il primo form consente di inserire nel database un articolo con tutti i suoi dati, che sono obbligatori.
Il secondo form mostra l'elenco degli articoli presenti nel database e disponibili per la vendita e dà la possibilità di selezionarne più di uno per creare un'asta che li comprende.

Un'asta è formata da 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 nella pagina VENDO 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 nell'elenco 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.

Versione con JavaScript

Si realizzi un'applicazione client server web che estende e/o 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.

Analisi della specifica

Analisi dei dati

Legenda:

- Entità
- Attributi
- Relazioni

Un'applicazione web consente la gestione di aste online.

Gli utenti accedono tramite login e possono vendere e acquistare articoli all'asta.

Ogni utente ha username, password, nome, cognome e indirizzo (quest'ultimo è usato per la spedizione degli articoli comperati).

Ogni articolo ha codice, nome, descrizione, immagine e prezzo.

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, uno per creare un nuovo articolo e uno per creare una nuova asta per vendere gli articoli dell'utente.
Il primo form consente di inserire nel database un articolo con tutti i suoi dati, che sono obbligatori.
Il secondo form mostra l'elenco degli articoli presenti nel database e disponibili per la vendita e dà la possibilità di selezionarne più di uno per creare un'asta che li comprende.

Un'asta è formata da 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 nella pagina VENDO 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 nell'elenco 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.

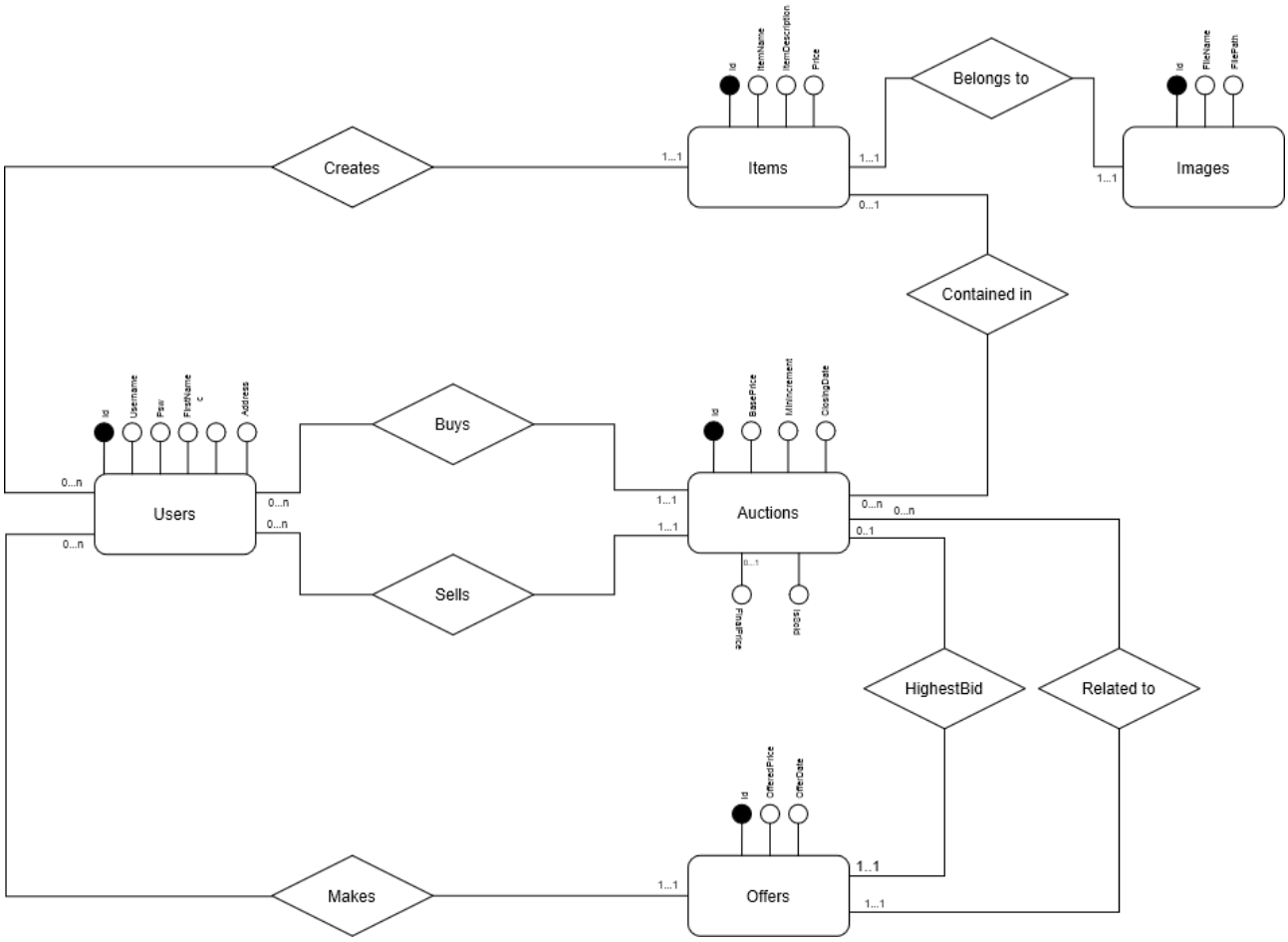
La pagina ACQUISTO contiene anche un elenco delle offerte aggiudicate all'utente con i dati degli articoli e il prezzo finale.

Nota: Durante la fase di progettazione si è scelto di rendere l'attributo *Immagine* dell'entità *Articolo* un'entità autonoma, stabilendo una relazione esplicita tra le due.
Questa scelta è stata motivata dalla volontà di prevenire possibili complicazioni nella gestione del salvataggio e del recupero delle immagini lato server.

In particolare, la separazione ha consentito di distinguere chiaramente due collegamenti: da un lato, quello tra l'*Articolo* e l'*Immagine* associata; dall'altro, quello tra l'*Immagine* memorizzata nel database e il file fisico presente nel filesystem.
Tale struttura rende più semplice l'adattamento futuro del sistema: nel caso in cui cambino le modalità o il luogo di archiviazione dei file, sarà sufficiente aggiornare le entry della tabella *Immagini*, senza dover modificare la tabella *Articoli*.

Nel contesto specifico di questo progetto, tale separazione si è rivelata solo parzialmente necessaria, ma si è deciso di mantenerla per garantire una struttura più solida e scalabile in vista di eventuali estensioni future.

Progetto del Database



Schema del Database

```sql
CREATE DATABASE IF NOT EXISTS Progetto_Tiw;
USE Progetto_Tiw;

-- Stores User's informations and login credentials
CREATE TABLE Users (
    Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -- Login Username
    Username VARCHAR(255) NOT NULL UNIQUE CHECK (LENGTH(TRIM(Username)) > 0),
    -- Password
    Psw VARCHAR(255) NOT NULL CHECK (LENGTH(TRIM(Psw)) > 0),
    -- Anagraphics
    FirstName VARCHAR(255) NOT NULL CHECK (LENGTH(TRIM(FirstName)) > 0),
    LastName VARCHAR(255) NOT NULL CHECK (LENGTH(TRIM(LastName)) > 0),
    -- User Address, used for shipping Auctioned Items
    Address VARCHAR(255) NOT NULL CHECK (LENGTH(TRIM(Address)) > 0)
);
```

```sql
-- Stores the File names and paths for Image resources of Items
CREATE TABLE Images (
    Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -- Name of the image file
    FileName VARCHAR(255) NOT NULL CHECK (LENGTH(TRIM(FileName)) > 0),
    -- Path inside the FileSystem to locate the file
    FilePath VARCHAR(255) NOT NULL UNIQUE CHECK (LENGTH(TRIM(FilePath)) > 0)
);

-- Stores all the Auctions created by Users, with all related data
CREATE TABLE Auctions (
    Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -- Starting Price for the Auction, calculated as the Sum of all the Item's Prices
    BasePrice INT NOT NULL CHECK (BasePrice > 0),
    -- Minimum Allowed Increment for Offers
    MinIncrement INT NOT NULL CHECK (MinIncrement > 0),
    -- The ID of the current Highest Offer for this Auction
    -- NULL if no Offers have been made yet
    HighestBidId INT,
    -- It will be added at the end of the SCRIPT via an ALTER TABLE
    -- Ending Date for the Auction
    ClosingDate DATETIME NOT NULL,
    -- The ID of the User who created the Auction
    SellerId INT NOT NULL,
    -- A Boolean value to determine whether the Auction has been closed or not
    IsSold BOOLEAN DEFAULT FALSE NOT NULL,
    -- The ID of the User who won the Auction
    BuyerId INT,
    -- The Price of the winning Offer
    FinalPrice INT CHECK (FinalPrice IS NULL OR FinalPrice >= 0),
    -- Foreign Keys
    FOREIGN KEY (BuyerId) REFERENCES Users(Id)
    -- If the buyer User is deleted, the Auction is not deleted
      ON DELETE SET NULL
      ON UPDATE CASCADE,
    FOREIGN KEY (SellerId) REFERENCES Users(Id)
       -- Users with Auctions can't delete their profile
       ON DELETE RESTRICT
       ON UPDATE CASCADE
    -- FOREIGN KEY (HighestBidId) REFERENCES Offers(Id)
       -- In case the Offer is deleted, the Auction's reference is set to NULL
       -- Added later via ALTER TABLE
       -- ON DELETE SET NULL
       -- ON UPDATE CASCADE
);
```

```sql
-- Stores the Offers made by the Users inside Auctions
CREATE TABLE Offers (
    Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -- The ID of the User who made the Offer
    UserId INT NOT NULL,
    -- The ID of the Auction where the Offered has been made
    AuctionId INT NOT NULL,
    -- The Price offered by the User
    OfferedPrice INT NOT NULL CHECK (OfferedPrice > 0),
    -- The Date and Time the Offer has been made
    OfferDate DATETIME NOT NULL,
    -- Foreign Keys
    FOREIGN KEY (UserId) REFERENCES Users(Id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY (AuctionId) REFERENCES Auctions(Id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);

/*
    Stores the Items created by the Users
    Items can be part of an Auction or not
*/
CREATE TABLE Items (
    Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -- The Name of the Item
    ItemName VARCHAR(255) NOT NULL CHECK (LENGTH(TRIM(ItemName)) > 0),
    -- The Description of the Item
    ItemDescription VARCHAR(1023) NOT NULL CHECK (LENGTH(TRIM(ItemDescription)) > 0),
    -- The Price of the Item
    Price INT NOT NULL CHECK (Price > 0),
    -- The ID of the Image of the Item, used to retrieve the image FilePath
    ImageId INT,
    -- The ID of the User who created the Item
    CreatorId INT NOT NULL,
    /*
       The ID of the Auction in which the Item is being sold.
       An Item can only reference 1 Auction, so no duplicates allowed.
       It can also be NULL, meaning the Item does not belong to any Auction yet.
    */
    AuctionId INT,
    -- Foreign Keys
    FOREIGN KEY (ImageId) REFERENCES Images(Id)
       -- Images can't be deleted if they belong to an Item
       ON DELETE RESTRICT
       ON UPDATE CASCADE,
    FOREIGN KEY (CreatorId) REFERENCES Users(Id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY (AuctionId) REFERENCES Auctions(Id)
       /*
           If an Auction is Deleted, the Items are not deleted,
           but marked as not in an Auction
       */
       ON DELETE SET NULL
       ON UPDATE CASCADE
);
```

```sql
-- Add Auctions.HighestBid
ALTER TABLE Auctions
    ADD CONSTRAINT fk_highest_bid
    FOREIGN KEY (HighestBidId) REFERENCES Offers(Id)
       ON DELETE SET NULL
       ON UPDATE CASCADE;

-- Indexes for performance
CREATE INDEX idx_offers_auction ON Offers(AuctionId);
CREATE INDEX idx_offers_user ON Offers(UserId);
CREATE INDEX idx_items_auction ON Items(AuctionId);
CREATE INDEX idx_items_creator ON Items(CreatorId);
CREATE INDEX idx_items_image ON Items(ImageId);
CREATE INDEX idx_auctions_seller ON Auctions(SellerId);
CREATE INDEX idx_auctions_buyer ON Auctions(BuyerId);
```

Analisi funzionale

Versione HTML pura

Analisi

Legenda:

- Pagine
- Componenti (View)
- Azioni
- Eventi

Un'applicazione web consente la gestione di aste online.

Gli utenti accedono tramite login e possono vendere e acquistare articoli all'asta.

Ogni utente ha username, password, nome, cognome e indirizzo (quest'ultimo è usato per la spedizione degli articoli comperati).

Ogni articolo ha codice, nome, descrizione, immagine e prezzo.

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, uno per creare un nuovo articolo e uno per creare una nuova asta per vendere gli articoli dell'utente.
Il primo form consente di inserire nel database un articolo con tutti i suoi dati, che sono obbligatori.
Il secondo form mostra l'elenco degli articoli presenti nel database e disponibili per la vendita e dà la possibilità di selezionarne più di uno per creare un'asta che li comprende.

Un'asta è formata da 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 nella pagina VENDO 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 nell'elenco 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.

Completamento delle specifiche

- La pagina LOGIN contiene un form.
- La pagina LOGIN contiene un link per navigare ad una pagina SIGN UP con un form dove l'utente può creare un account.
- La pagina SIGN UP contiene un link per tornare alla pagina LOGIN.
- Le pagine HOME, ACQUISTO, VENDITA, DETTAGLIO ASTA e OFFERTA contengono un bottone che permette all'utente, quando cliccato, di fare logout dal sito.
- Le pagine ACQUISTO e VENDITA contengono un link che permette di tornare alla pagina HOME.
- Le liste di Aste contengono un componente riutilizzabile, che mostra tutti i dati dell'asta. Questo componente contiene anche un link che, quando cliccato, fa comparire la pagina DETTAGLIO ASTA o OFFERTA.
- Le pagine DETTAGLIO ASTA e OFFERTA contengono un link che permette di tornare alla pagina precedente.

Riassunto e completamento dell'analisi funzionale

Pagine e componenti di interfaccia:

- Login
    - Form di login
    - Pulsante di login
    - Link per pagina SIGN UP
- Sign Up *
    - Form di registrazione *
    - Pulsante di invio
    - Link per tornare alla pagina LOGIN
- Home
    - Link per accedere alla pagina VENDITA
    - Link per accedere alla pagina ACQUISTO
    - Link per eseguire logout e tornare alla pagina di LOGIN
- Sell (Vendita)
    - Link per tornare alla pagina HOME
    - Link per eseguire il logout e tornare alla pagina di LOGIN
    - Lista delle aste create dall'utente e ancora aperte
    - Lista delle aste create dall'utente e chiuse
    - Form per creare un nuovo Articolo
    - Form per creare una nuova Asta
- Buy (Acquisto)
    - Link per tornare alla pagina HOME
    - Link per eseguire il logout e tornare alla pagina di LOGIN
    - Form per eseguire la ricerca
    - Lista delle aste trovate dalla ricerca **
    - Lista delle aste vinte dall'utente
- Auction Details (Dettaglio Asta e Offerta) ***
    - Link per tornare alla pagina precedente
    - Link per eseguire il logout e tornare alla pagina di LOGIN
    - Dati dell'asta
    - Dati dell'utente aggiudicatario (opzionale)
    - Lista degli Articoli contenuti nell'Asta
    - Bottone per chiudere l'asta (opzionale) ****
    - Lista delle Offerte fatte all'Asta (opzionale)
    - Form per inserire una nuova Offerta (opzionale)

* Per ragioni di *User Experience* si è deciso di dividere la fase di registrazione in due pagine separate, dunque anche il form risulta diviso in due.

** Per ragioni di *UX* si è deciso di nascondere la sezione dei risultati della ricerca se non è stato cercato nulla.
Nel momento in cui è presente il parametro *query* all'interno dell'URL viene mostrato anche questo componente, ma in sua assenza si suppone che l'utente abbia caricato la pagina da zero e non abbia, dunque, ancora eseguito alcuna ricerca.
Rimane sempre visibile, invece, il form che funge da barra di ricerca.

*** Per ragioni di *UX* e di miglior gestione del codice le due pagine DETTAGLIO ASTA e OFFERTA sono state unite in un'unica pagina.
Queste due pagine mostrano lo stesso contenuto, ovvero i dati dell'asta selezionata, con la differenza che la seconda mostra anche la lista delle Offerte fatte a quell'asta, nonché il form che permette all'utente di fare un'offerta a sua volta.
Questa sezione dell'interfaccia può essere mostrata o nascosta programmaticamente per ottenere l'una o l'altra vista a partire dalla stessa pagina originale.

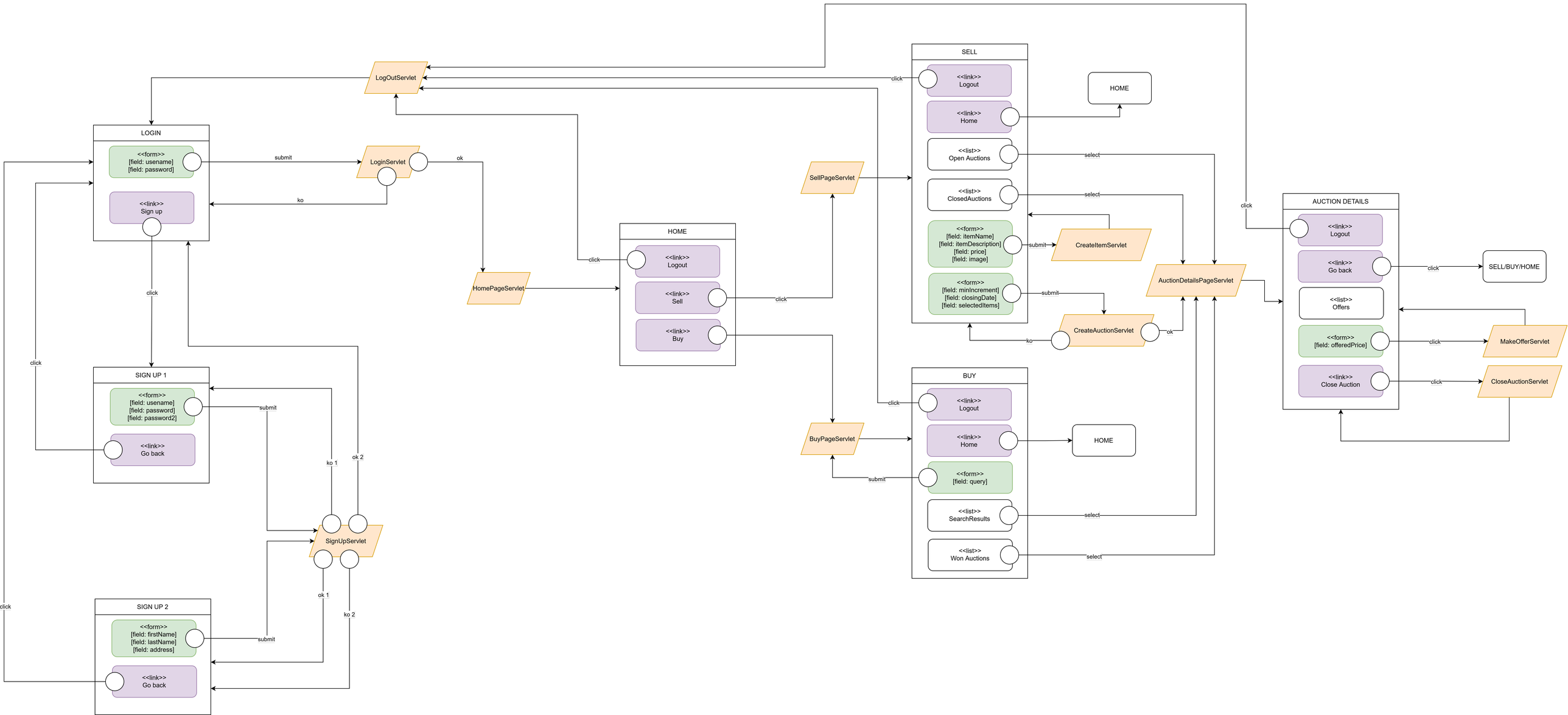
**** Mostrato solo se l'utente è anche il creatore dell'asta

Eventi ed azioni:

- Click del pulsante di login (o form submit)
    - Verifica delle credenziali ed autenticazione
- Click sul pulsante “Sign Up” sotto il form di Login
    - Reindirizzamento alla pagina SignUp 1
- Click del pulsante “Continua” nel primo form SignUp (o form submit)
    - Verifica disponibilità dell’username e validità della password scelta
    - Reindirizzamento alla pagina SignUp2
- Click del pulsante “SignUp” nel secondo form SignUp (o form submit)
    - Verifica disponibilità dell’username e validità della password scelta *
    - Reindirizzamento alla pagina Login
- Click del pulsante “Go back” nelle due pagine di SignUp
    - Reindirizzamento alla pagina Login
- Click del pulsante “Logout” nelle pagine Home, Sell. Buy, AuctionDetails
    - Invalidazione della sessione
    - Reindirizzamento alla pagina Login
- Click del pulsante “Sell” nella pagina Home
    - Reindirizzamento alla pagina Sell
- Click del pulsante “Buy” nella pagina Home
    - Reindirizzamento alla pagina Buy
- Click del pulsante “Home” nelle pagine Sell e Buy
    - Reindirizzamento alla pagina Home
- Click sul pulsante “Create Item” (Crea Articolo) nella pagina Sell (o form submit)
    - Invio dei dati a CreateItemServlet
    - Ricaricamento della pagina Sell con il nuovo Articolo o il messaggio di errore
- Click del pulsante “Create Auction” (Crea Asta) nella pagina Sell (o form submit)
    - Invio dei dati a CreateAuctionServlet
    - Ricaricamento della pagina Sell con la nuova Asta o il messaggio di errore
- Click del pulsante “Search” nella pagina Buy (o form submit)
    - Reindirizzamento alla pagina Buy (stessa pagina) con l’attributo query aggiunto alla richiesta
    - Ricaricamento della pagina con l’esito della ricerca (lista di risultati o messaggio di segnalazione)
- Click del pulsante “See more“ in una lista di Aste nelle pagine Sell e Buy
    - Reindirizzamento alla pagina AuctionDetails
- Click del pulsante “Send” nel form di invio offerta nella pagina AuctionDetails (o form submit)
    - Invio dei dati a MakeOfferServlet
    - Ricaricamento della pagina AuctionDetails con la nuova offerta o il messaggio di errore
- Click del pulsante “Close Auction” nella pagina AuctionDetails

- Invio dei dati a CloseAuctionServlet
- Ricaricamento della pagina AuctionDetails con nuova interfaccia o messaggio di errore

* Dopo la prima verifica i dati di username e password (entrambe) vengono inseriti in dei campi nascosti del secondo form, e vengono re-inviati al secondo submit.
Per evitare che l'utente possa modificare le credenziali in questo secondo passaggio ed arrecare danni al Database o al sistema, le credenziali vengono nuovamente controllate e validate, e qualora fosse identificato un errore legato ad esse, l'utente verrebbe reindirizzato nuovamente al primo form SignUp, di fatto annullando il tentativo di registrazione e costringendolo a ricominciare.
Un opportuno messaggio di errore sarebbe mostrato insieme al form.
Si segnala che questo secondo passaggio di validazione non tiene conto di quanto già controllato e verificato in precedenza. È dunque possibile che l'utente modifichi le proprie credenziali tra i campi nascosti del secondo form con delle altre credenziali altrettanto valide, e queste sarebbero accettate dal sistema. Lo scopo di questo secondo controllo non è garantire un processo di registrazione impossibile da manomettere, bensì proteggere il sistema dall'inserimento di credenziali non conformi alle specifiche o di username duplicati, che comprometterebbero la sicurezza del Login.
Pertanto, si considerano valide soltanto le credenziali contenute nel secondo submit, e sono quelle che vengono effettivamente inserite nel Database.


LOGIN
<<form>>
[field: usename]
[field: password]
<<link>>
Sign up
submit
LoginServlet
ok
ko
click
SIGN UP 1
<<form>>
[field: usename]
[field: password]
[field: password2]
<<link>>
Go back
submit
ko 1
ok 2
SignUpServlet
SIGN UP 2
<<form>>
[field: firstName]
[field: lastName]
[field: address]
<<link>>
Go back
submit
ok 1
ko 2
click
LogOutServlet
HOME
<<link>>
Logout
<<link>>
Sell
<<link>>
Buy
click
HomePageServlet
click
SellPageServlet
BuyPageServlet
SELL
<<link>>
Logout
<<link>>
Home
<<list>>
Open Auctions
<<list>>
ClosedAuctions
<<form>>
[field: itemName]
[field: itemDescription]
[field: price]
[field: image]
<<form>>
[field: minIncrement]
[field: closingDate]
[field: selectedItems]
HOME
select
select
submit
CreateItemServlet
submit
CreateAuctionServlet
ko
ok
BUY
<<link>>
Logout
<<link>>
Home
<<form>>
[field: query]
<<list>>
SearchResults
<<list>>
Won Auctions
HOME
submit
select
select
click
AuctionDetailsPageServlet
click
AUCTION DETAILS
<<link>>
Logout
<<link>>
Go back
<<list>>
Offers
<<form>>
[field: offeredPrice]
<<link>>
Close Auction
SELL/BUY/HOME
click
MakeOfferServlet
click
CloseAuctionServlet
click

Componenti

Model Objects (Beans):

- User (Utente)
- Item (Articolo)
- Auction (Asta)
- Offer (Offerta)
- Image (Immagine)

Data Access Objects (DAOs):

- UserDAO:
  - User checkCredentials(String username, String password)
  - void validateCredentials(String username, String password1, String password2)
  - User getUserById(int userId)
  - void insert(User user, String password)
- ItemDAO:
  - int insert(Item item)
  - void updateItemsAuctionId(List<Item> items, int auctionId)
  - void addImageToItem(int itemId, int imageId)
  - List<Item> getItemsInAuction(int auctionId)
  - List<Item> getItemsByIDs(List<Integer> itemIds)
  - List<Item> getAvailableItemsForUserId(int userId)
- AuctionDAO:
  - int insert(Auction auction)
  - void markAuctionAsClosed(Auction auction)
  - void updateAuctionsHighestBid(int auctionId, int offerId)
  - Auction getAuctionById(int id, long loginTime)
  - List<Auction> getAuctionsCreatedBy(User user, boolean closed, long loginTime)
  - List<Auction> getAuctionsNotCreatedBy(User user, long loginTime)
  - List<Auction> getAuctionsWonBy(User user, long loginTime)
  - List<Auction> getAuctionsForKeywords(String[] keywords, int userId, long loginTime)
- OfferDAO:
  - int insert(Offer offer)
  - Offer getOfferById(int offerId)
  - List<Offer> getOffersForAuction(int auctionId)
- ImageDAO:
  - Image getImageById(int id)
  - int insert(String fileName, String filePath)
  - void delete(int imageId)

Controller o Servlet:

- LoginServlet
- SignUpServlet
- LogOutServlet
- CloseAuctionServlet
- CreateAuctionServlet
- CreateItemServlet
- FrontServlet
- MakeOfferServlet

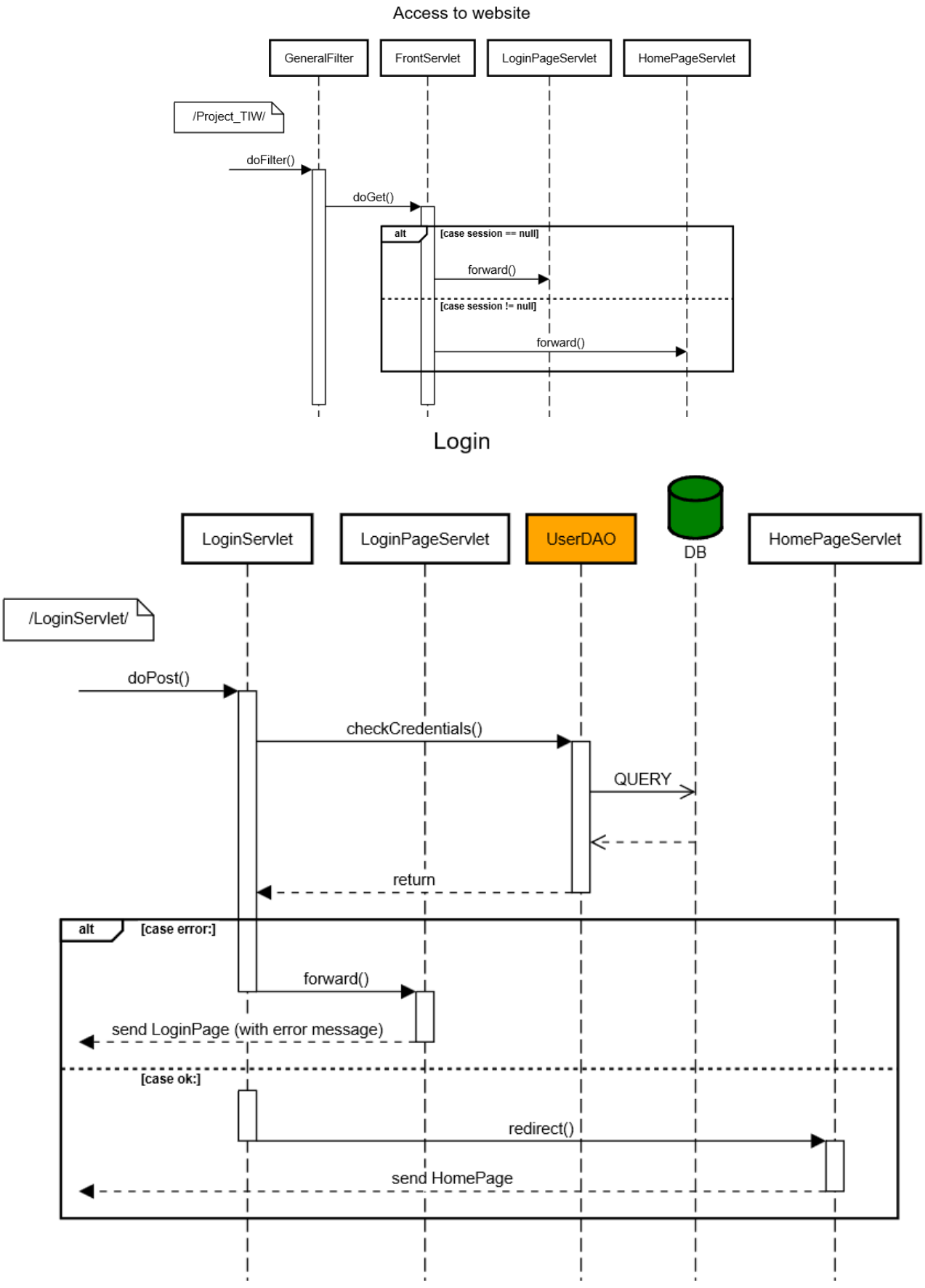
Servlet dedicate alle pagine (PageServlet):

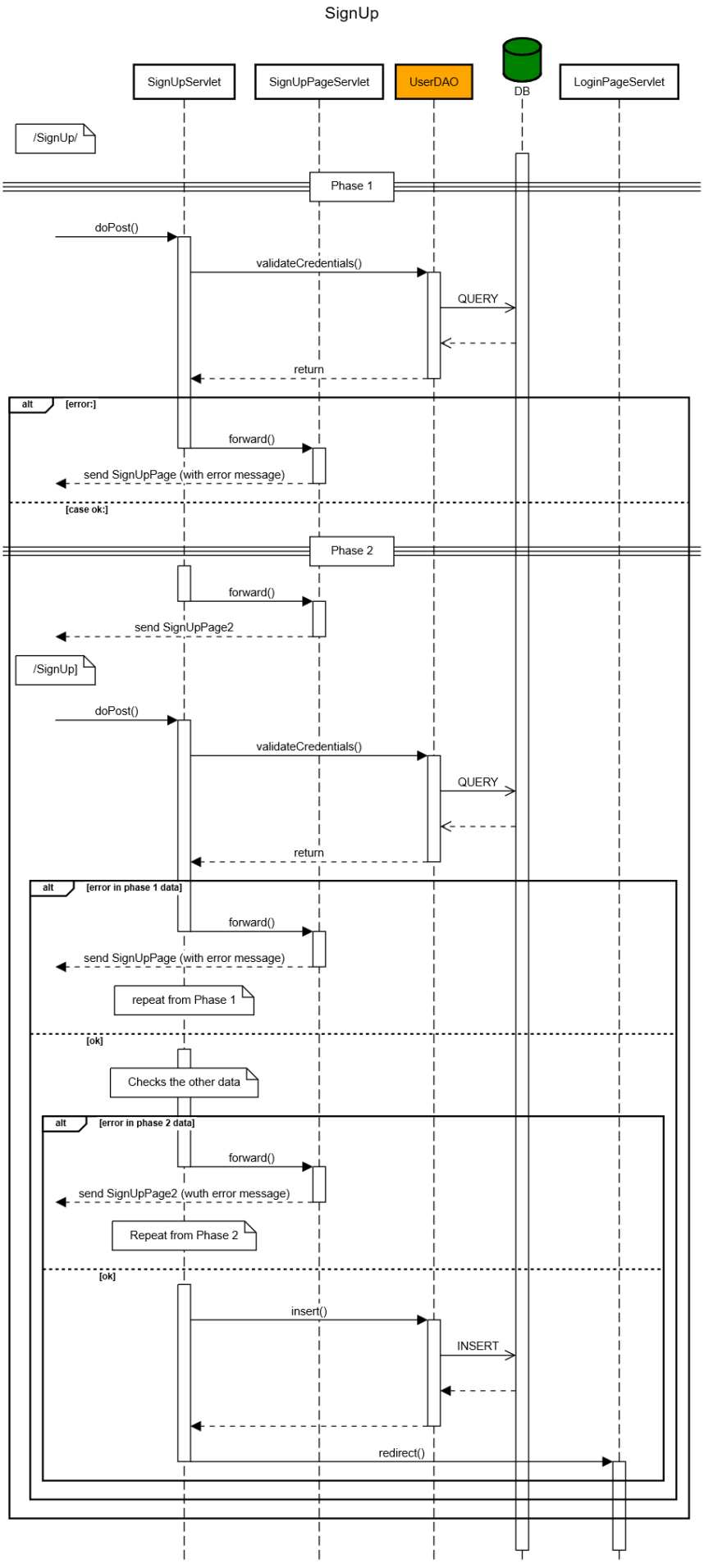
- AuctionDetailsPageServlet
- BuyPageServlet
- ErrorPageServlet
- HomePageServlet
- LoginPageServlet
- SellPageServlet
- SignUpServlet

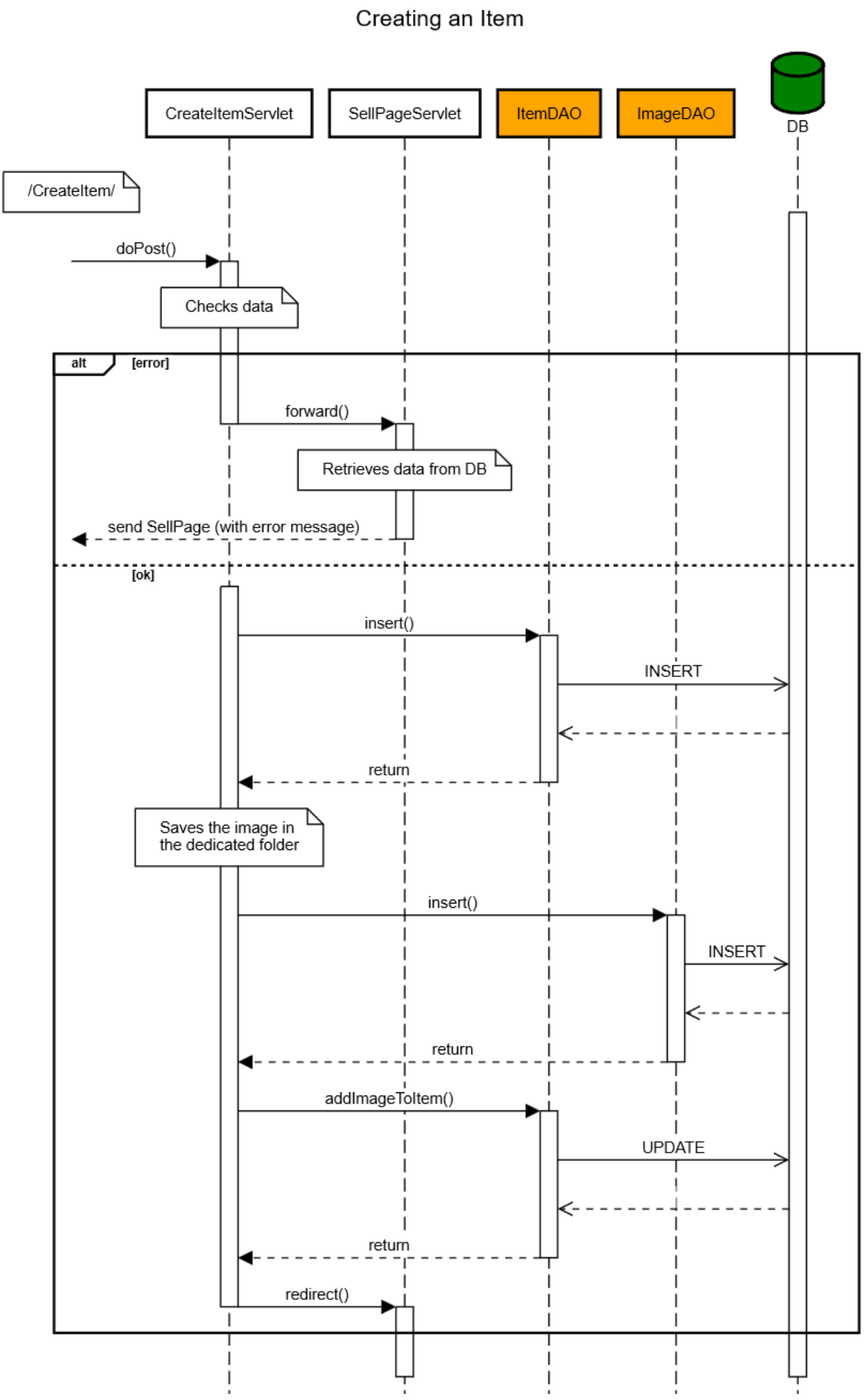
Pagine (template):

- AuctionDetailsPage
- BuyPage
- ErrorPage
- HomePage
- LoginPage
- SellPage
- SignUpPage
- SignUpPage2

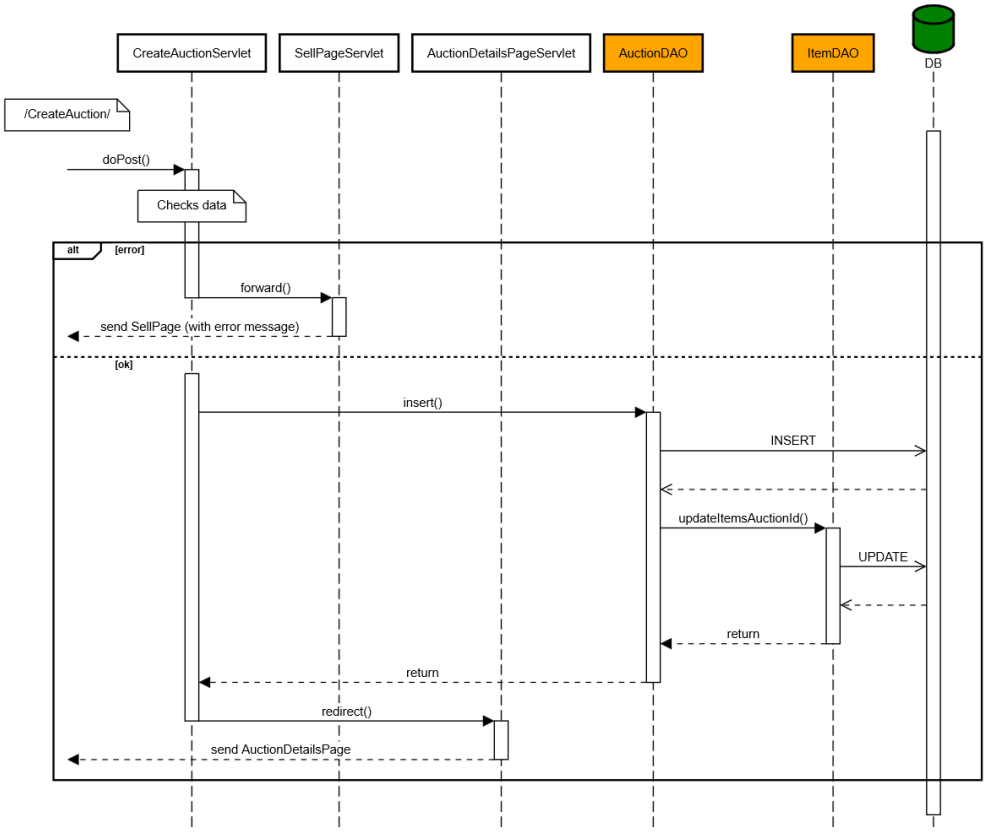
Sequence Diagrams



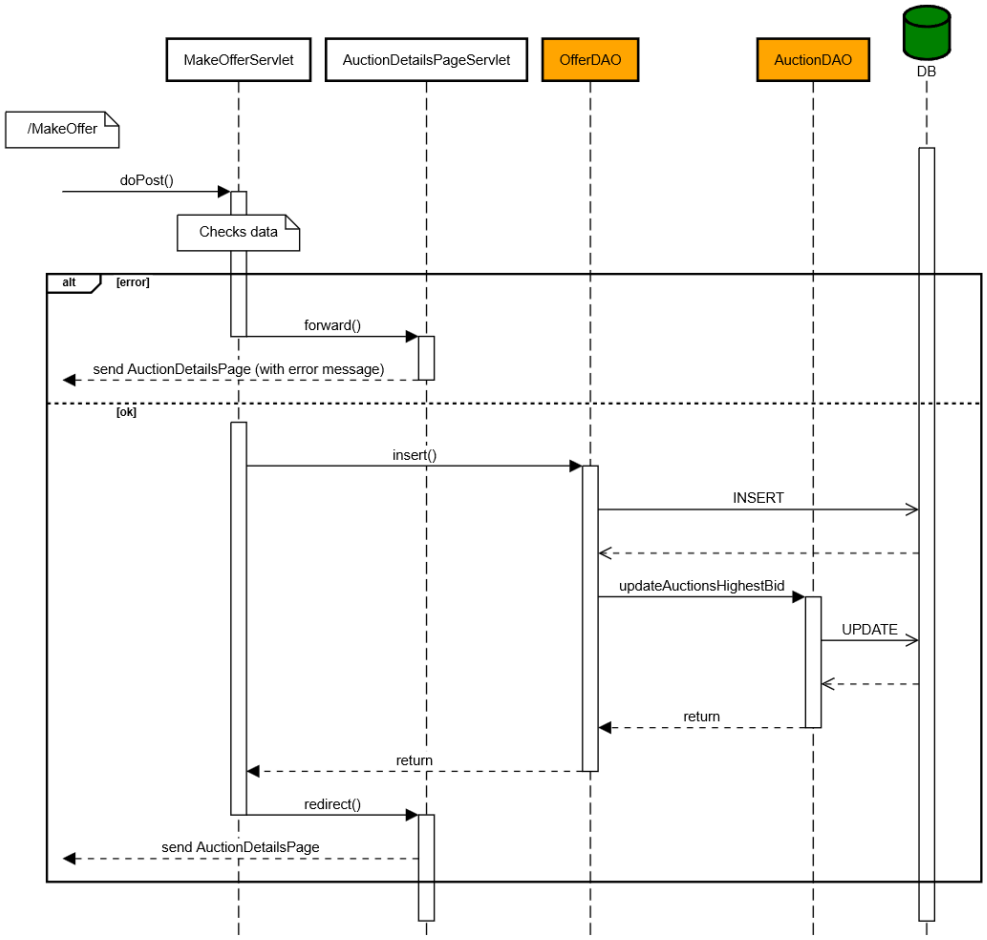

SignUp
SignUpServlet
SignUpPageServlet
UserDAO
DB
LoginPageServlet
/SignUp/
Phase 1
doPost()
validateCredentials()
QUERY
return
alt
[error:]
forward()
send SignUpPage (with error message)
[case ok:]
Phase 2
forward()
send SignUpPage2
/SignUp]
doPost()
validateCredentials()
QUERY
return
alt
[error in phase 1 data]
forward()
send SignUpPage (with error message)
repeat from Phase 1
[ok]
Checks the other data
alt
[error in phase 2 data]
forward()
send SignUpPage2 (wuth error message)
Repeat from Phase 2
[ok]
insert()
INSERT
redirect()


Creating an Item
CreateItemServlet
SellPageServlet
ItemDAO
ImageDAO
DB
/CreateItem/
doPost()
Checks data
alt
[error]
forward()
Retrieves data from DB
send SellPage (with error message)
[ok]
insert()
INSERT
return
Saves the image in
the dedicated folder
insert()
INSERT
return
addImageToItem()
UPDATE
return
redirect()

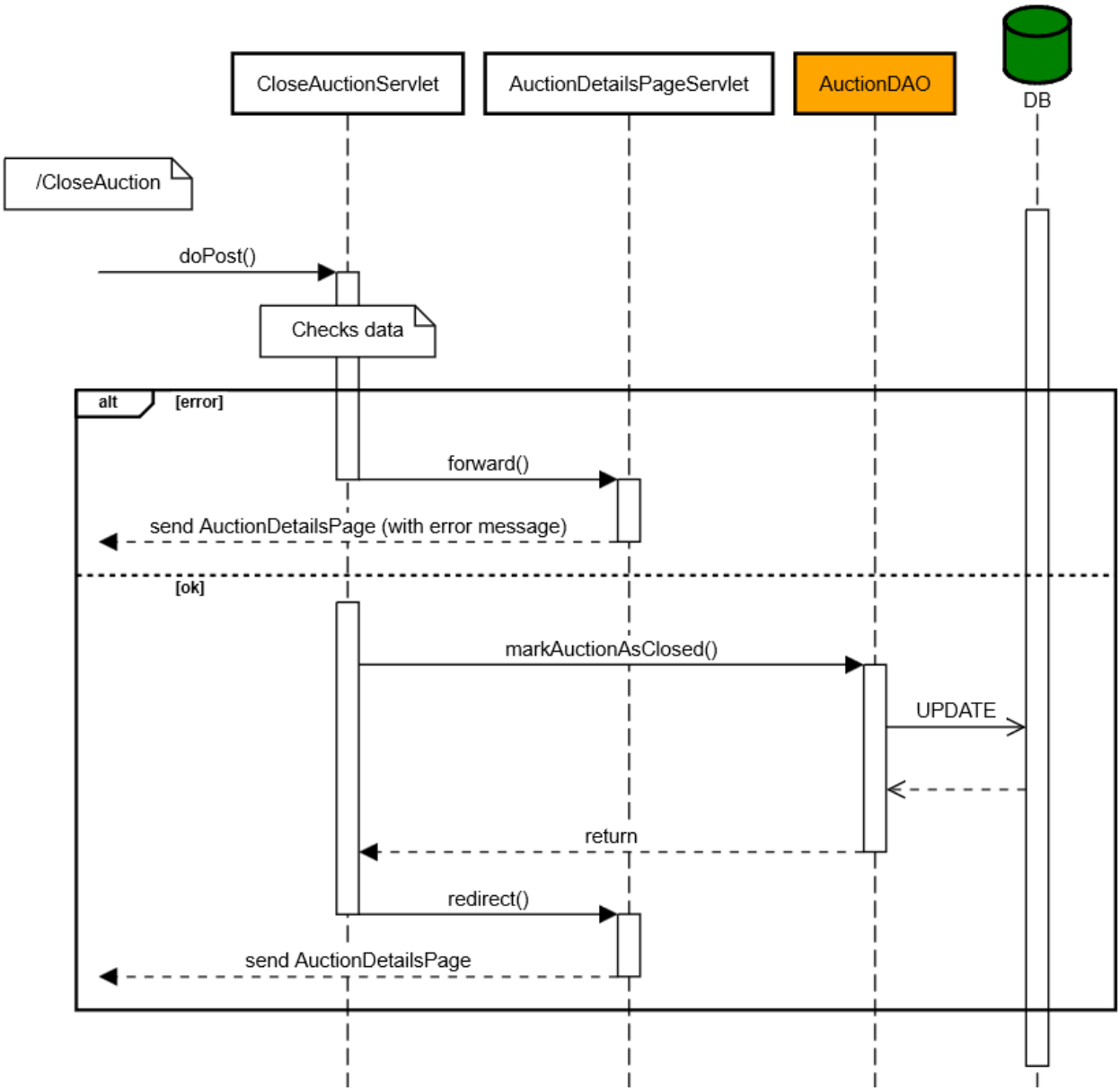
Creating an Auction



Making an Offer



Closing an Auction



View e interfaccia

Per questa versione del progetto era sufficiente fornire le varie pagine richieste in singole pagine web individuali, dunque sono state implementate le seguenti pagine:

- **LoginPage**: contiene un form in cui l'utente può inserire username e password per autenticarsi.
- **SignUpPage**: contiene un form in cui l'utente inserisce i propri username e password (la password viene inserita due volte, per conferma) per il suo nuovo account.
- **SignUpPage2**: contiene un form in cui l'utente inserisce i propri nome, cognome e indirizzo per completare la creazione dell'account.
- **HomePage**: contiene due link, uno a SellPage e uno a BuyPage.
- **SellPage**: contiene la lista delle Aste create dall'utente e ancora aperte, la lista delle Aste create dall'utente e chiuse, un form in cui può creare un nuovo Articolo e uno in cui può creare una nuova Asta.
- **BuyPage**: contiene una barra di ricerca, una lista di risultati di ricerca e una lista delle Aste vinte dall'utente.
- **AuctionDetailsPage**: Mostra tutti i dettagli di un'asta, adattando la propria UI allo stato dell'Asta.

Si è deciso di utilizzare *Thymeleaf* per assistere al processo di personalizzazione delle pagine. Dunque tutte queste pagine presentano innumerevoli tag provenienti da questo framework ed in grado di adattare la presentazione ed il contenuto delle pagine in base ai dati contenuti nella richiesta.

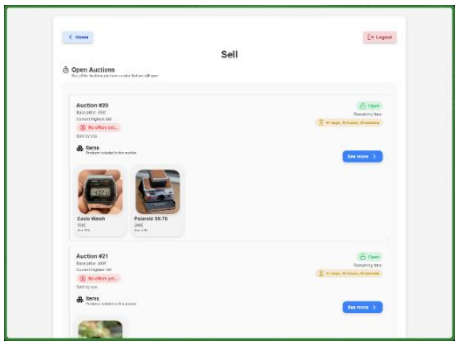
Per fare ciò, tuttavia, è necessario che il server invochi il metodo `process()` del *TemplateEngine* di *Thymeleaf* per renderizzare una pagina HTML completa prima di poterla inviare al client. Sono dunque state implementate delle Servlet dedicate, una per ogni pagina. Queste Servlet sono state opportunamente indicate dal prefisso "*PageServlet*", per coerenza, e sono:

- LoginPageServlet (/login)
- SignUpPageServlet (per entrambe le pagine) (/signup)
- HomePageServlet (/home)
- SellPageServlet (/sell)
- BuyPageServlet (/buy)
- AuctionDetailsPageServlet (/auction-details)

A queste Servlet sono stati assegnati degli URL semplici e distintivi, in quanto pensati per essere utilizzabili dall'utente.

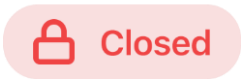
All’interno delle singole pagine troviamo un layout relativamente simile, con un riutilizzo di molte classi CSS per garantire continuità e coerenza.

Sempre in queste pagine sono presenti elementi costruiti dinamicamente, quali i componenti che rappresentano le singole Aste all’interno delle varie liste, le card che rappresentano gli Articoli delle Aste, o anche le bolle che contengono le Offerte.



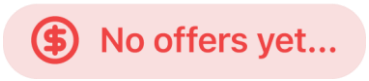
Esempio di pagina: SellPage

In questa versione del progetto, questi elementi sono stati scritti una volta in HTML e iterati tramite un `th:each` dal *TemplateEngine* di *Thymeleaf*.
Per i contenuti la cui presenza dipende dai dati rappresentati si è invece fatto uso di `th:if` e `th:unless`.



Indicatori dello stato dell'Asta

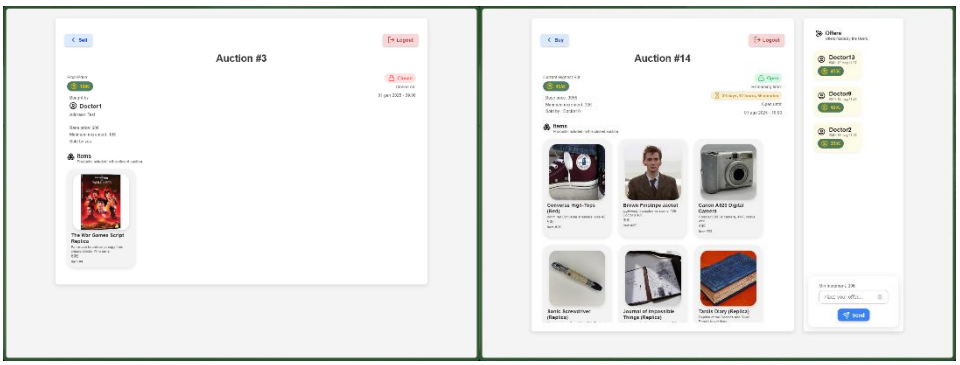
Questo permette di mostrare alternativamente uno o l’altro indicatore di Asta chiusa o aperta, oppure di mostrare l’indicatore di Offerta massima o prezzo finale al posto di quello che segnala l’assenza di Offerte.



In alto un esempio di indicatore di Offerta massima, in basso quello che segnala l'assenza di Offerte

Inoltre, il comando `th:if` ha consentito di ridurre il numero di pagine da quanto espresso nelle specifiche a quanto effettivamente implementato: infatti secondo le specifiche, aprire i dettagli di un’asta aperta deve mostrare una pagina contenente tutti i dati dell’asta e la lista delle offerte, mentre aprire un’asta già chiusa deve mostrare una pagina con solamente i dati dell’asta.

Essendo le due pagine accomunate per gran parte del contenuto, si è deciso di realizzare una sola schermata, composta da un contenitore con i dettagli dell’asta, la cui presenza, assenza e contenuto è controllata sulla base dei dati dell’Asta mostrata, e un secondo contenitore sulla destra dedicato alle Offerte, la cui presenza o assenza è gestita ad un `th:if` sul contenitore stesso.



Versione con JavaScript

Analisi

Legenda:

- Pagine
- Componenti (View)
- Azioni
- Eventi

Un'applicazione web consente la gestione di aste online.

Gli utenti accedono tramite login e possono vendere e acquistare articoli all'asta.

Ogni utente ha username, password, nome, cognome e indirizzo (quest'ultimo è usato per la spedizione degli articoli comperati).

Ogni articolo ha codice, nome, descrizione, immagine e prezzo.

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, uno per creare un nuovo articolo e uno per creare una nuova asta per vendere gli articoli dell'utente.
Il primo form consente di inserire nel database un articolo con tutti i suoi dati, che sono obbligatori.
Il secondo form mostra l'elenco degli articoli presenti nel database e disponibili per la vendita e dà la possibilità di selezionarne più di uno per creare un'asta che li comprende.

Un'asta è formata da 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 nella pagina VENDO 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 nell'elenco 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.

Si realizzi un'applicazione client server web che estende e/o 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.

Completamento delle specifiche

- La pagina LOGIN contiene un form.
- La sezione LOGIN della pagina contiene un pulsante per mostrare una seconda sezione SIGN UP, con al suo interno un form dove l'utente può creare un account.
- La sezione SIGN UP contiene un pulsante che mostra la sezione LOGIN della pagina.
- La pagina HOME contiene al suo interno le sezioni ACQUISTO, VENDITA, DETTAGLIO ASTA e OFFERTA.
- La pagina HOME contiene un bottone che permette all'utente, quando cliccato, di fare logout dal sito.
- La pagina HOME contiene un selettore che permette di navigare tra le sezioni VENDITA e ACQUISTO.
- Le sezioni ACQUISTO e VENDITA contengono un componente riutilizzabile, che mostra tutti i dati dell'asta. Questo componente contiene anche un pulsante che, quando cliccato, fa comparire la sezione DETTAGLIO ASTA o OFFERTA.
- Le sezioni DETTAGLIO ASTA e OFFERTA contengono un pulsante che permette di tornare alla vista precedente.

Riassunto e completamento dell'analisi funzionale

Pagine e componenti di interfaccia:

- Login
    - Login
        - Form di login
        - Pulsante di login
        - Pulsante per sezione SIGN UP
    - Sign Up *
        - Form di registrazione *
        - Pulsante di invio
        - Pulsante per sezione LOGIN
- Home
    - Link per eseguire logout e tornare alla pagina LOGIN
    - Selettore per selezionare tra VENDITA e ACQUISTO
    - Sell (Vendita)
        - Lista delle aste create dall'utente e ancora aperte
        - Lista delle aste create dall'utente e chiuse
        - Form per creare un nuovo Articolo
        - Form per creare una nuova Asta
    - Buy (Acquisto)
        - Form per eseguire la ricerca
        - Lista delle aste trovate dalla ricerca **
        - Lista delle aste visitate di recente
        - Lista delle aste vinte dall'utente
    - Auction Details (Dettaglio Asta e Offerta) ***
        - Pulsante per nascondere la sezione e mostrare quella precedente
        - Dati dell'Asta
        - Dati dell'utente aggiudicatario (opzionale)
        - Lista degli Articoli contenuti nell'Asta
        - Bottone per chiudere l'asta (opzionale) ****
        - Lista delle Offerte fatte all'Asta (opzionale)
        - Form per inserire una nuova Offerta (opzionale)

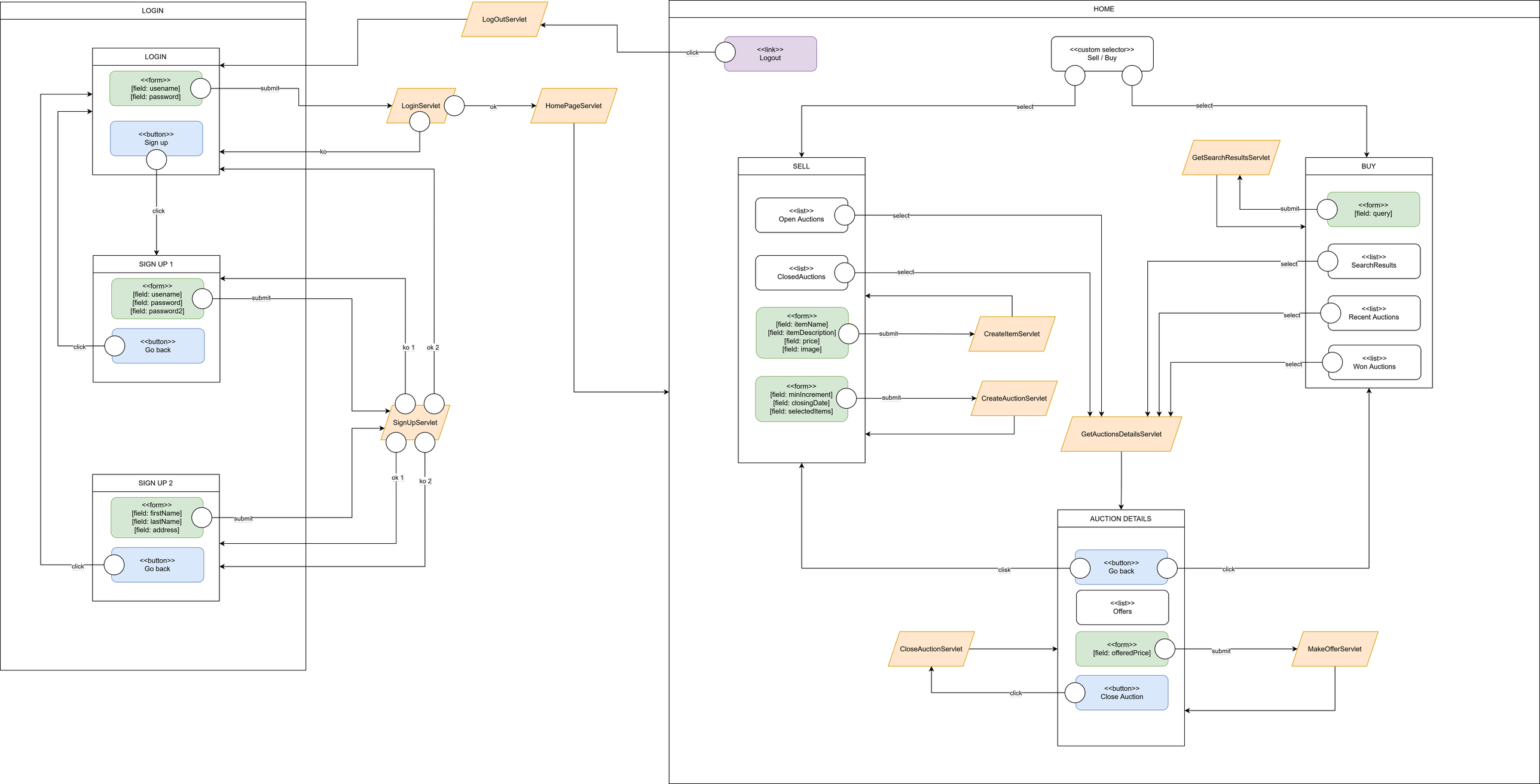
* Per ragioni di *User Experience* si è deciso di dividere la fase di registrazione in due sezioni separate, dunque anche il form risulta diviso in due.

** Per ragioni di *UX* si è deciso di nascondere la sezione dei risultati della ricerca se non è stato cercato nulla.
Non potendo più raggiungere questa pagina in modo specifico con una query nell'URL, si assume ora che la pagina sia sempre "pulita" al primo caricamento, ovvero senza alcuna ricerca già presente. La pagina si aggiorna in automatico quando la richiesta asincrona della

ricerca ritorna delle Aste, che vengono aggiunte alla pagina tramite JavaScript.
Risulta sempre visibile, invece, il form che funge da barra di ricerca.

*** Per ragioni di *UX* e di miglior gestione del codice le due sezioni DETTAGLIO ASTA e OFFERTA sono state unite in un'unica sezione.
Queste due sezioni mostrano lo stesso contenuto, ovvero i dati dell'asta selezionata, con la differenza che la seconda mostra anche la lista delle Offerte fatte a quell'asta, nonché il form che permette all'utente di fare un'offerta a sua volta.
Questa sotto-sezione dell'interfaccia può essere mostrata o nascosta programmaticamente per ottenere l'una o l'altra vista a partire dalla stessa sezione originale.

**** Mostrato solo se l'utente è anche il creatore dell'Asta.


LOGIN
LOGIN
<<form>>
[field: usename]
[field: password]
submit
<<button>>
Sign up
LoginServlet
ok
HomePageServlet
ko
click
SIGN UP 1
<<form>>
[field: usename]
[field: password]
[field: password2]
submit
click
<<button>>
Go back
ko 1
ok 2
SignUpServlet
ok 1
ko 2
SIGN UP 2
<<form>>
[field: firstName]
[field: lastName]
[field: address]
submit
click
<<button>>
Go back
LogOutServlet
click
<<link>>
Logout
HOME
<<custom selector>>
Sell / Buy
select
select
SELL
<<list>>
Open Auctions
select
<<list>>
ClosedAuctions
select
<<form>>
[field: itemName]
[field: itemDescription]
[field: price]
[field: image]
submit
CreateItemServlet
<<form>>
[field: minIncrement]
[field: closingDate]
[field: selectedItems]
submit
CreateAuctionServlet
GetSearchResultsServlet
BUY
<<form>>
[field: query]
submit
<<list>>
SearchResults
select
<<list>>
Recent Auctions
select
<<list>>
Won Auctions
select
GetAuctionsDetailsServlet
AUCTION DETAILS
<<button>>
Go back
clisk
click
<<list>>
Offers
CloseAuctionServlet
<<form>>
[field: offeredPrice]
submit
MakeOfferServlet
<<button>>
Close Auction
click

Componenti

Model Objects (Beans):

- User (Utente)
- Item (Articolo)
- Auction (Asta)
- Offer (Offerta)
- Image (Immagine)

Data Access Objects (DAOs):

- UserDAO:
  - User checkCredentials(String username, String password)
  - void validateCredentials(String username, String password1, String password2)
  - User getUserById(int userId)
  - void insert(User user, String password)
- ItemDAO:
  - int insert(Item item)
  - void updateItemsAuctionId(List<Item> items, int auctionId)
  - void addImageToItem(int itemId, int imageId)
  - List<Item> getItemsInAuction(int auctionId)
  - List<Item> getItemsByIDs(List<Integer> itemIds)
  - List<Item> getAvailableItemsForUserId(int userId)
- AuctionDAO:
  - int insert(Auction auction)
  - void markAuctionAsClosed(Auction auction)
  - void updateAuctionsHighestBid(int auctionId, int offerId)
  - Auction getAuctionById(int id, long loginTime)
  - List<Auction> getAuctionsCreatedBy(User user, boolean closed, long loginTime)
  - List<Auction> getAuctionsNotCreatedBy(User user, long loginTime)
  - List<Auction> getAuctionsWonBy(User user, long loginTime)
  - List<Auction> getAuctionsForKeywords(String[] keywords, int userId, long loginTime)
- OfferDAO:
  - int insert(Offer offer)
  - Offer getOfferById(int offerId)
  - List<Offer> getOffersForAuction(int auctionId)
- ImageDAO:
  - Image getImageById(int id)
  - int insert(String fileName, String filePath)
  - void delete(int imageId)

Controller o Servlet:

- LoginServlet
- SignUpServlet
- LogOutServlet
- CloseAuctionServlet
- CreateAuctionServlet
- CreateItemServlet
- FrontServlet
- GetAuctionsDetailsServlet
- GetAvailableItemsServlet
- GetClosedAuctionsServlet
- GetOpenAuctionsServlet
- GetSearchResultsServlet
- GetUserInfoServlet
- GetVisitedAuctionsServlet
- GetWonAuctionsServlet
- MakeOfferServlet

Servlet dedicate alle pagine (PageServlet) *:

- LoginPageServlet
- HomePageServlet

Pagine:

- LoginPage
- HomePage

* In questa versione del progetto si è deciso comunque di mantenere il meccanismo delle PageServlet, ovvero Servlet dedicate unicamente ad inviare file HTML, piuttosto che concedere l'accesso diretto ai file, in modo da poter fornire due URL più semplici (/login e /home).

Eventi ed azioni

N	Client-side		Server-side	
	Evento	Azione	Evento	Azione
1	Login -> login form -> submit	Login	POST (username, password)	Controllo credenziali ed autenticazione
2	Login -> button Sign Up	Visualizzazione form Sign Up 1	-	-
3	SignUp1 -> signUp form -> submit	Controllo credenziali + Visualizzazione form SignUp 2	POST (username, password, password2)	Controllo disponibilità username
4	SignUp 2 -> signUp form -> submit	Controllo dati + Visualizzazione form Login	POST (username, password, password2, firstName, lastName, address)	Controllo disponibilità username + registrazione
5	SignUp 1/2 -> button Go Back	Visualizzazione form Login	-	-
6	HomePage -> link Logout	Logout	GET()	Invalidazione della sessione + redirect a Loginpage
7	HomePage -> load	Recupero ID e nome utente	GET()	Invio dati utente
8	HomePage->load	Recupero articoli disponibili per nuova asta	GET()	Invio dati articoli
9	HomePage -> load	Recupero aste aperte, aste chiuse, aste vinte	GET()	Invio dati aste
10	HomePage->load	Visualizzazione aste recenti	POST(AuctionIds)	Invio dati aste
11	HomePage -> load	Lettura primo accesso / ultima azione eseguita e visualizzazione della sezione corretta	-	-
12	HomePage -> selezione Buy/Sell	Visualizzazione sezione Buy/Sell + refresh delle liste associate	GET() (per i refresh)	Invio dati
13	HomePage -> crea articolo form -> submit	Invio dei dati per creare l'articolo + refresh degli articoli disponibili nel form di creazione Asta	POST(itemName, itemDescription, price, image) (creazione Articolo) + GET() (refresh)	Controllo dati + aggiunta a DB (creazione Articolo) + invio dati (refresh)

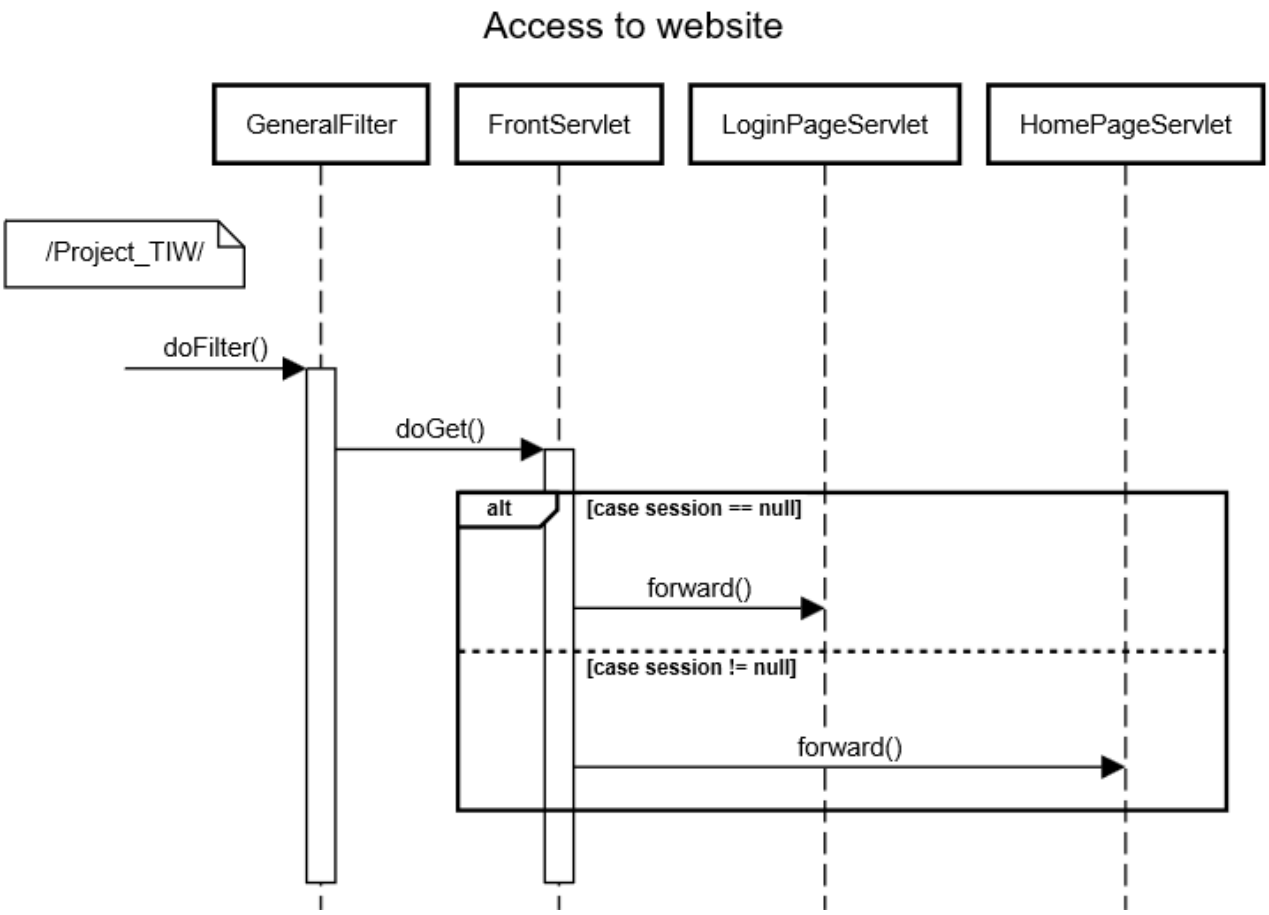
14	HomePage -> crea asta form -> submit	Invio dei dati per creare l'asta + refresh delle aste aperte e degli articoli disponibili nel form di creazione asta	POST(minIncrement, closingDate, items) (creazione Asta) + GET() (refresh)	Controllo dati + aggiunta a DB (creazione Asta), Invio dati (refresh)
15	HomePage -> selezione articolo in form di creazione asta checkbox -> check	Selezione della checkbox + cambio di estetica della <label> (tramite CSS)	-	-
16	HomePage -> search form -> submit	Invio della query + aggiornamento lista dei risultati (e comparsa)	GET(query)	Invio dati
17	HomePage -> apertura dettaglio asta (da qualunque lista) button -> click	Richiesta dati aggiornati dell'Asta + comparsa del popup + aggiornamento della UI + salvataggio in cookie dell'ID	GET(id)	Invio dati
18	HomePage -> chiusura del popup dettaglio asta button -> click	Scomparsa del popup	-	-
19	HomePage -> invio offerta form -> submit	Controllo dati + Invio + refresh UI del popup dettaglio asta	POST(userId, auctionId, offerId) (invio offerta) + GET(id) (refresh)	Controllo dati + inserimento in DB (invio offerta) + Invio dati (refresh)
20	HomePage -> chiusura asta button -> click	Controllo + Invio + refresh UI del popup dettaglio asta	POST(userId, auctionId) (chiusura) + GET(id) (refresh)	Controllo dati e validità + aggiornamento DB (chiusura) + Invio dati (refresh)

Controller ed event handlers

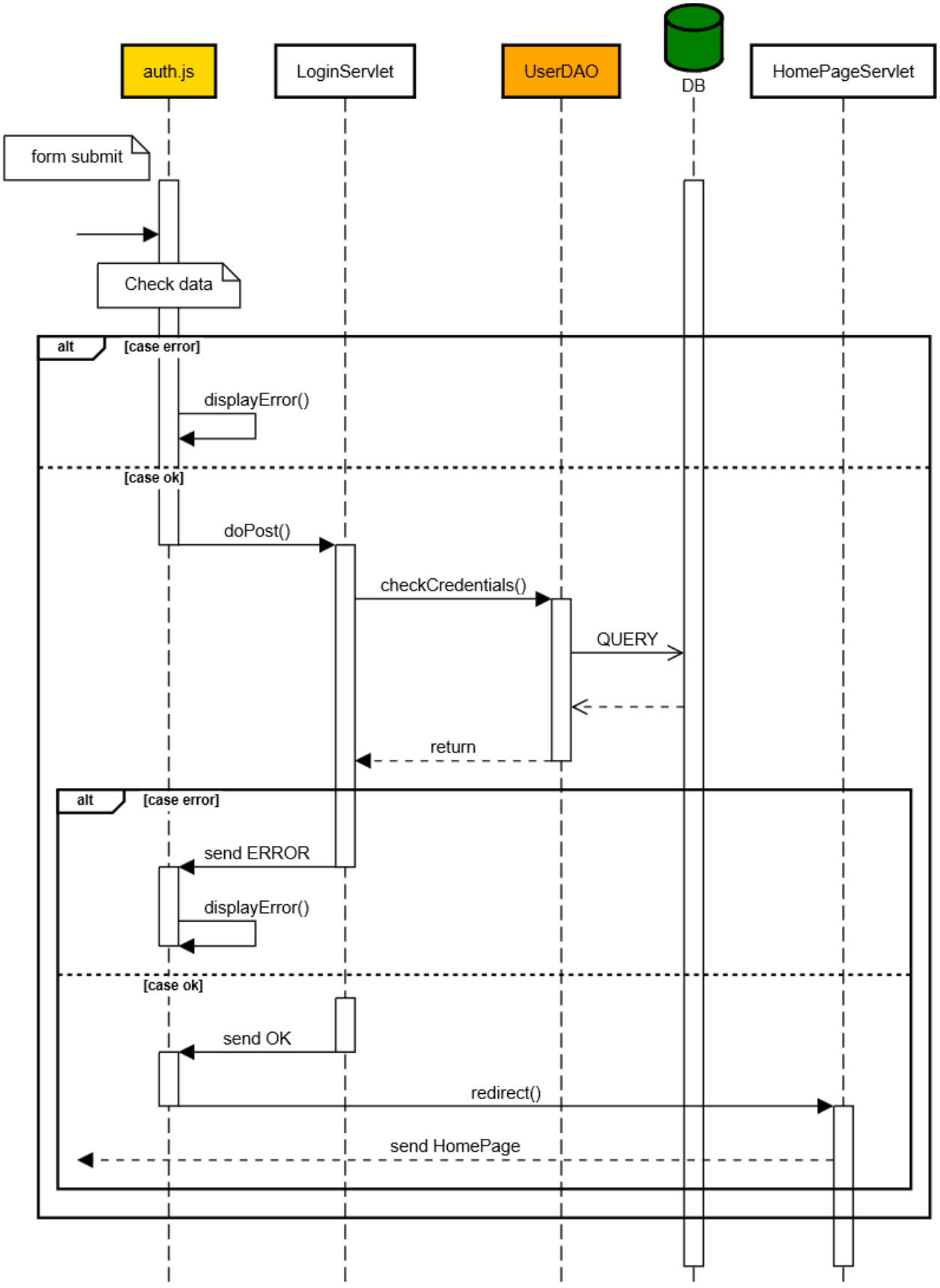
N	Client-side		Server-side	
	Evento	Controllore (funzione)	Evento	Controllore (Servlet)
1	Login -> login form -> submit	form.eventHandler	POST (username, password)	LoginServlet
2	Login -> button Sign Up	showForm()	-	-
3	SignUp1 -> signUp form -> submit	form.eventHandler	POST (username, password, password2)	SignUpServlet
4	SignUp 2 -> signUp form -> submit	form.eventHandler	POST (username, password, password2, firstName, lastName, address)	SignUpServlet
5	SignUp 1/2 -> button Go Back	showForm()	-	-
6	HomePage -> link Logout	-	GET()	LogOutServlet
7	HomePage -> load	fetchUser()	GET()	GetUserInfoServlet
8	HomePage->load	refreshAvailableItems()	GET()	GetAvailableItemsServlet
9	HomePage -> load	refreshOpenAuctions(), refreshClosedAuctions(), refreshWonAuctions()	GET()	GetOpenAuctionsServlet, GetClosedAuctionsServlet, GetWonAuctionsServlet
10	HomePage->load	refreshVisitedAuctions()	POST(AuctionIds)	GetVisitedAuctionsServlet
11	HomePage -> load	initPillTabBar()	-	-
12	HomePage -> selezione Buy/Sell	activateTab() (chiama le varie funzioni refresh...())	GET() (per i refresh)	Le varie Get...AuctionsServlet (per i refresh)
13	HomePage -> crea articolo form -> submit	form.eventHandler (settato da initItemCreationForm() on load), che chiama validateItemCreation() per il controllo dei dati, refreshAvailableItems()	POST(itemName, itemDescription, price, image) (creazione Articolo) + GET() (refresh)	CreateItemServlet, GetAvailableItemsServlet

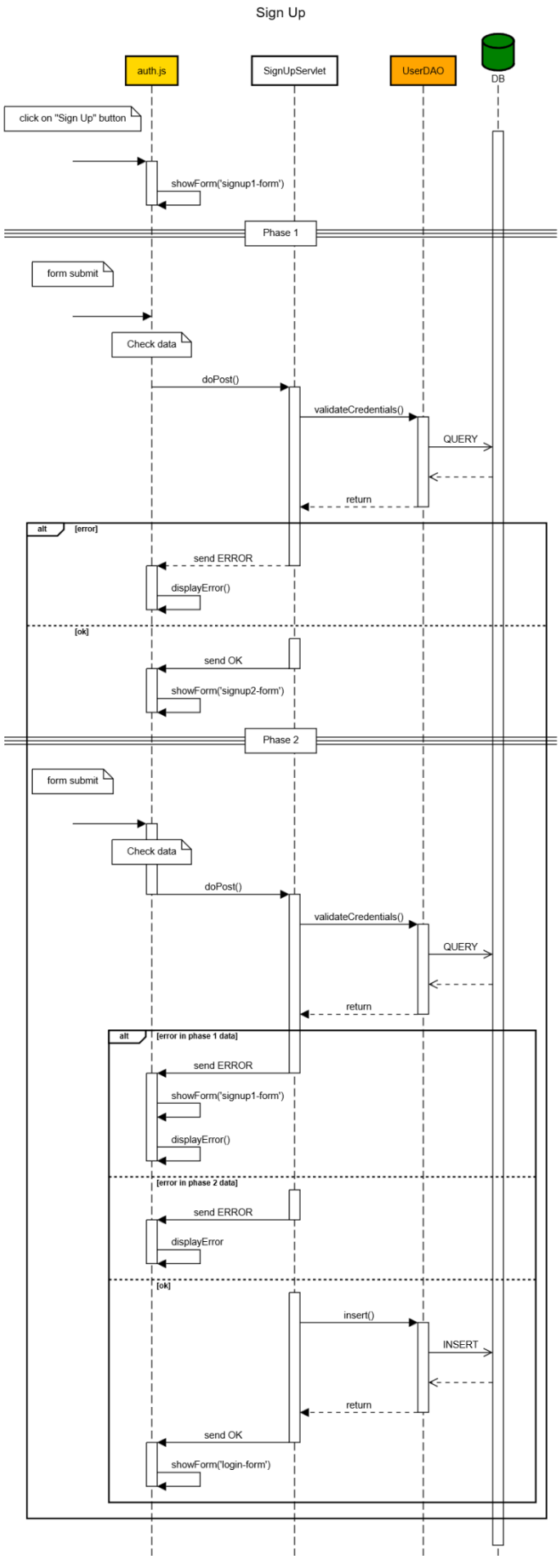
14	HomePage -> crea asta form -> submit	form.eventHandler(settato da initAuctionCreationForm()), che chiama validateAuctionCreationForm() per il controllo dei dati, refreshOpenAuctions(), refreshAvailableItems()	POST(minIncrement, closingDate, items) (creazione Asta) + GET() (refresh)	CreateAuctionsServlet, GetOpenAuctionsServlet, GetAvailableItemsServlet
15	HomePage -> selezione articolo in form di creazione asta checkbox -> check	-	-	-
16	HomePage -> search form -> submit	form.eventHandler	GET(query)	GetSearchResultsServlet
17	HomePage -> apertura dettaglio asta (da qualunque lista) button -> click	showAuctionPopup(), updateAuctionPopup(), addAuctionToVisited()	GET(id)	GetAuctionDetailsServlet()
18	HomePage -> chiusura del popup dettaglio asta button -> click	closePopup()	-	-
19	HomePage -> invio offerta form -> submit	form.eventHandler (settato da initMakeOfferForm()), che chiama validateOffer() per il controllo dei dati, updateAuctionPopup() per il refresh	POST(userId, auctionId, offerId) (invio offerta) + GET(id) (refresh)	MakeOfferServlet, GetAuctionsDetailsServlet
20	HomePage -> chiusura asta button -> click	button.eventHandler (settato da initCloseAuctionButton()), che chiama validateCloseAuction() per il controllo, updateAuctionPopup() per il refresh	POST(userId, auctionId) (chiusura) + GET(id) (refresh)	CloseAuctionServlet, GetAuctionsDetailsServlet

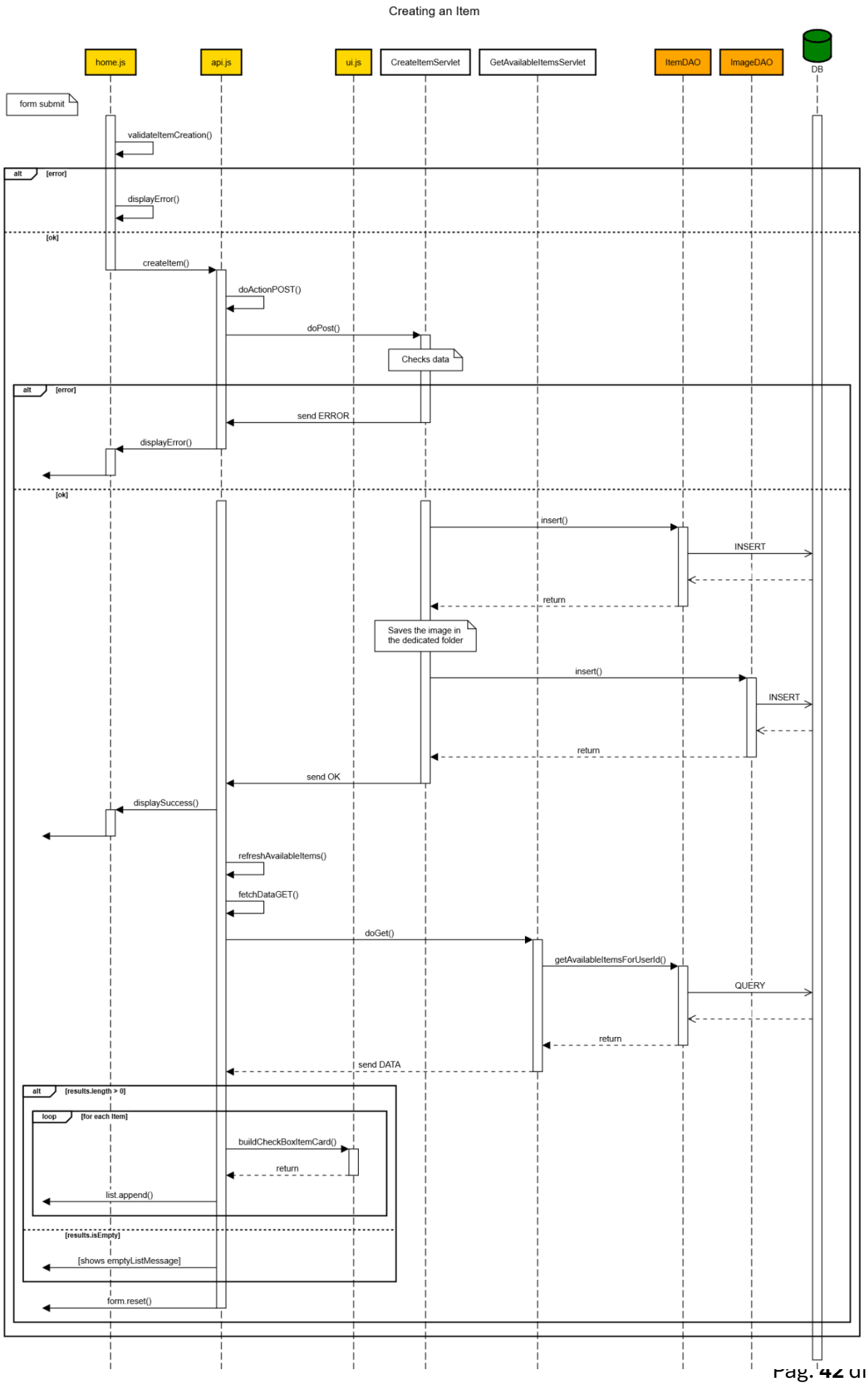
Sequence Diagrams



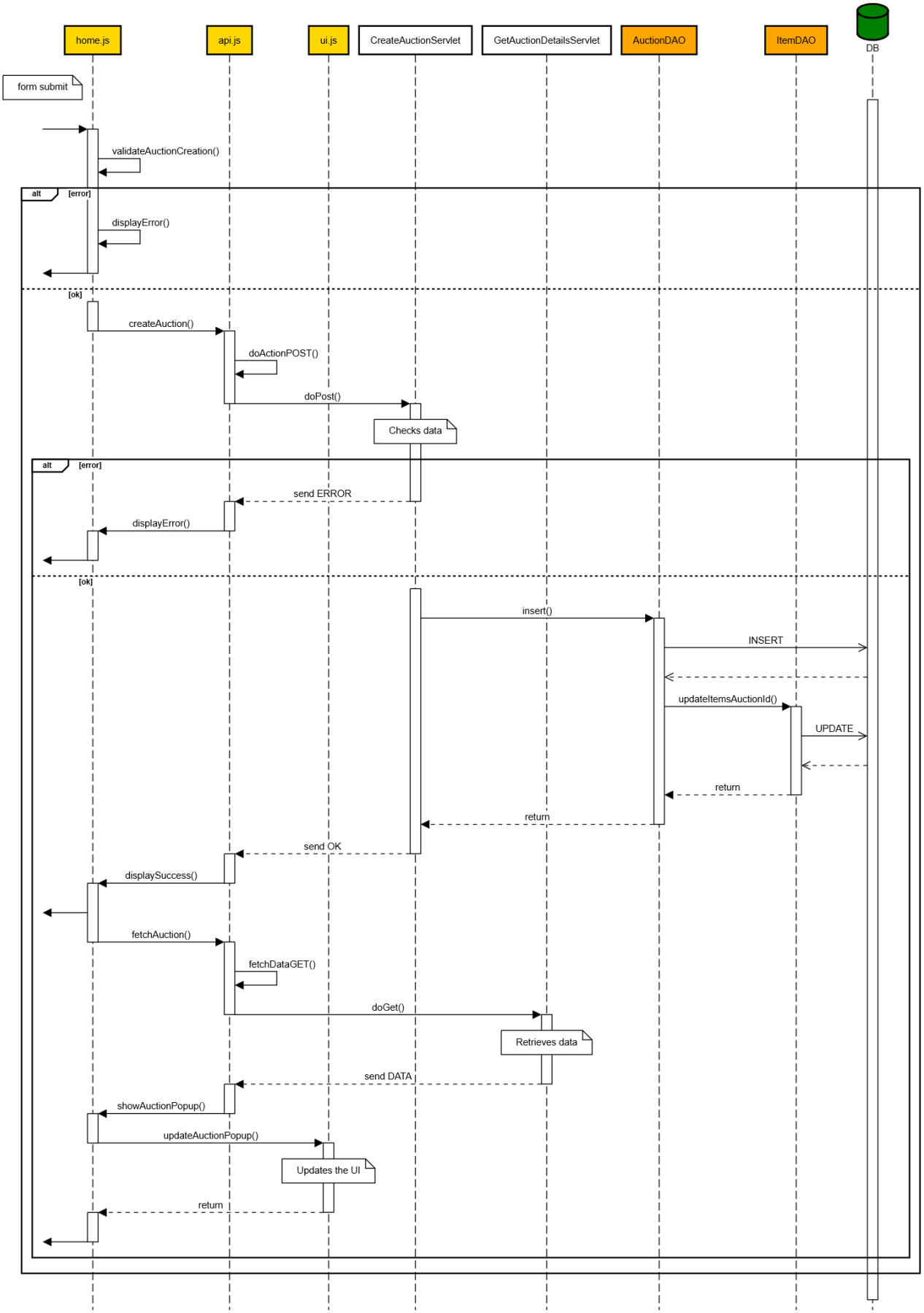
Login

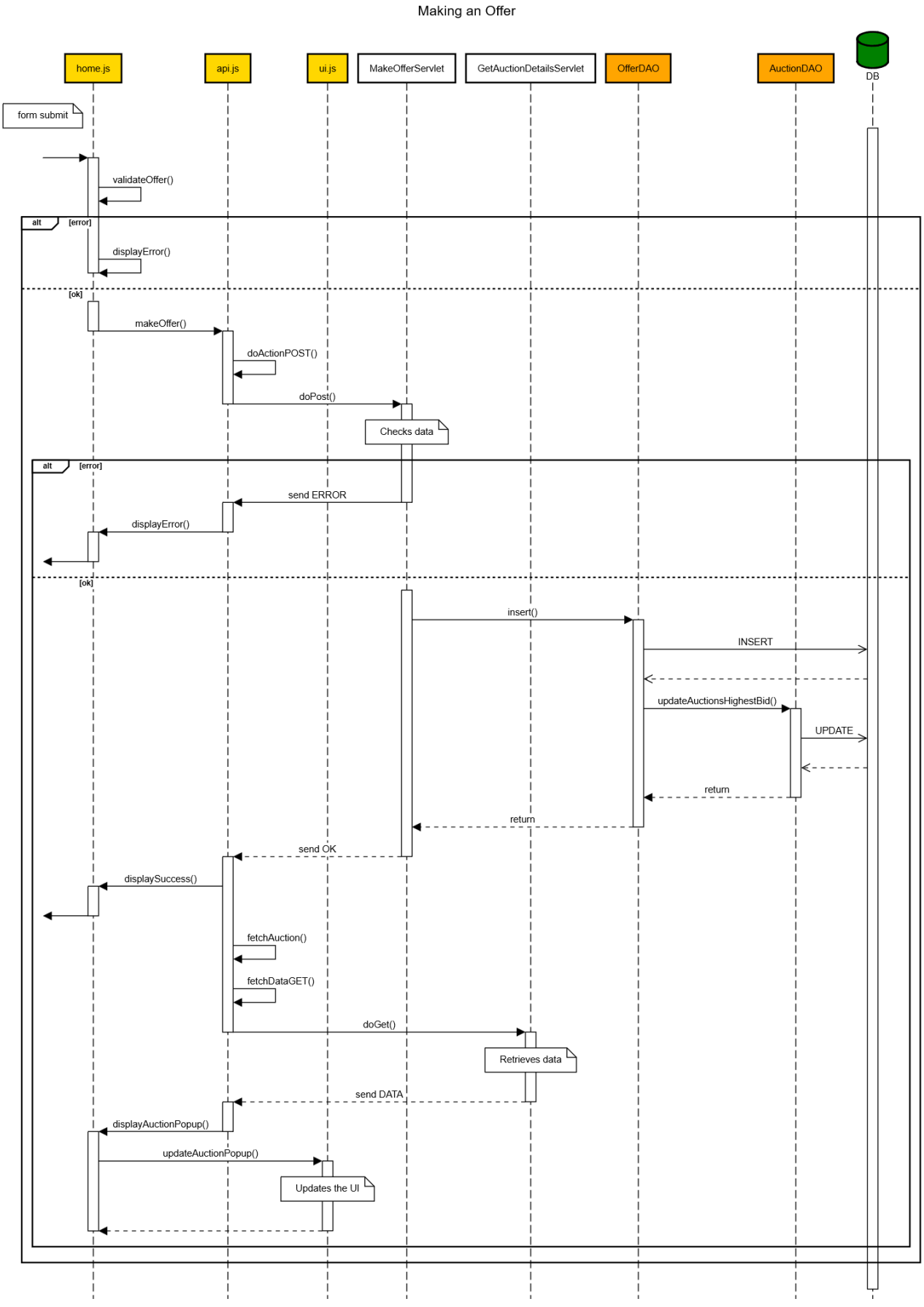

auth.js
LoginServlet
UserDAO
DB
HomePageServlet
form submit
Check data
alt
[case error]
displayError()
[case ok]
doPost()
checkCredentials()
QUERY
return
alt
[case error]
send ERROR
displayError()
[case ok]
send OK
redirect()
send HomePage


Sign Up
auth.js
SignUpServlet
UserDAO
DB
click on "Sign Up" button
showForm('signup1-form')
Phase 1
form submit
Check data
doPost()
validateCredentials()
QUERY
return
alt
[error]
send ERROR
displayError()
[ok]
send OK
showForm('signup2-form')
Phase 2
form submit
Check data
doPost()
validateCredentials()
QUERY
return
alt
[error in phase 1 data]
send ERROR
showForm('signup1-form')
displayError()
[error in phase 2 data]
send ERROR
displayError
[ok]
insert()
INSERT
return
send OK
showForm('login-form')

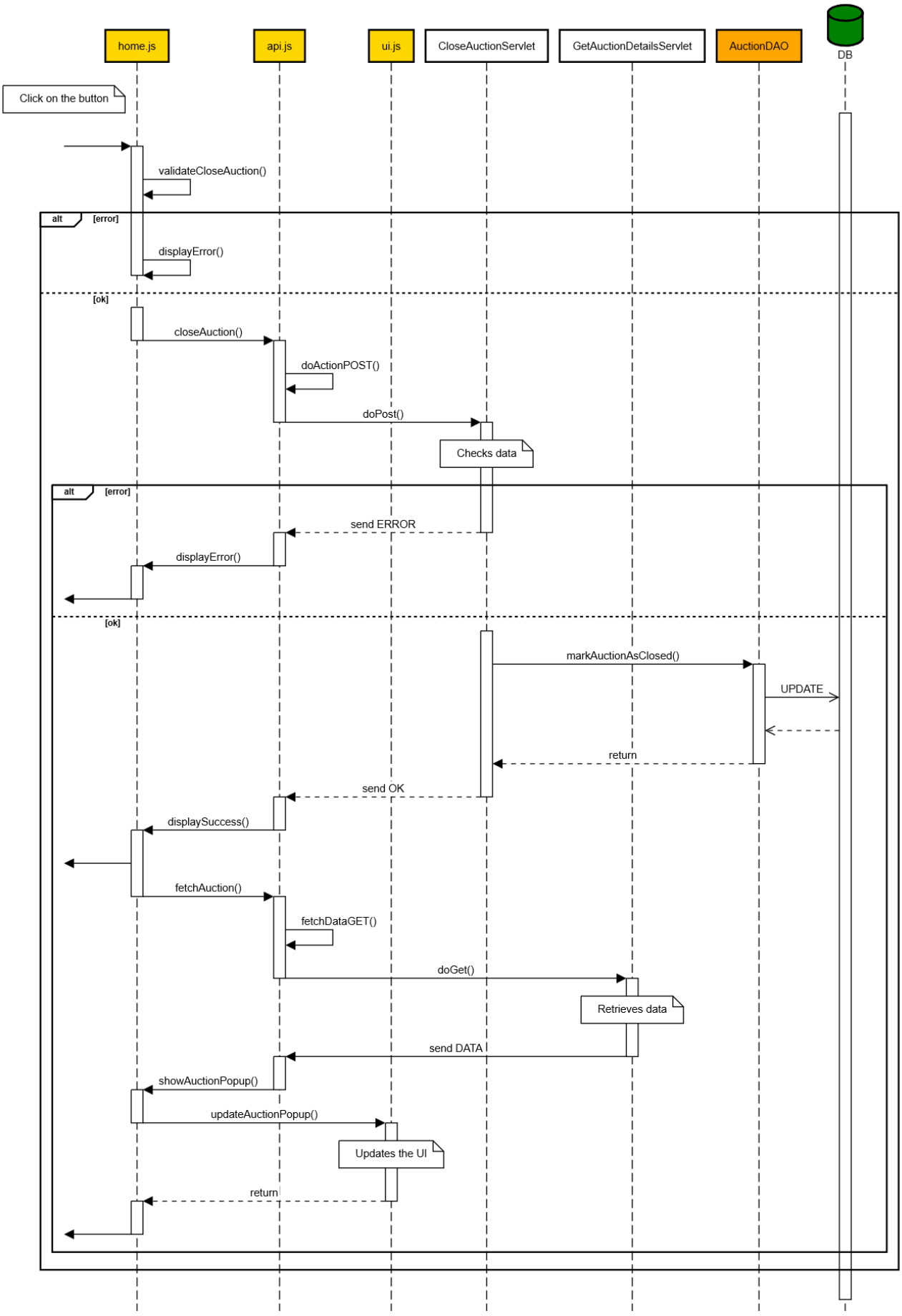

Creating an Item
home.js
api.js
ui.js
CreateItemServlet
GetAvailableItemsServlet
ItemDAO
ImageDAO
DB
form submit
validateItemCreation()
alt
[error]
displayError()
[ok]
createItem()
doActionPOST()
doPost()
Checks data
alt
[error]
send ERROR
displayError()
[ok]
insert()
INSERT
return
Saves the image in the dedicated folder
insert()
INSERT
return
send OK
displaySuccess()
refreshAvailableItems()
fetchDataGET()
doGet()
getAvailableItemsForUserId()
QUERY
return
send DATA
alt
[results.length > 0]
loop
[for each Item]
buildCheckBoxItemCard()
return
list.append()
[results.isEmpty]
[shows emptyListMessage]
form.reset()

Creating an Auction




Making an Offer
home.js
api.js
ui.js
MakeOfferServlet
GetAuctionDetailsServlet
OfferDAO
AuctionDAO
DB
form submit
validateOffer()
alt
[error]
displayError()
[ok]
makeOffer()
doActionPOST()
doPost()
Checks data
alt
[error]
send ERROR
displayError()
[ok]
insert()
INSERT
updateAuctionsHighestBid()
UPDATE
return
return
send OK
displaySuccess()
fetchAuction()
fetchDataGET()
doGet()
Retrieves data
send DATA
displayAuctionPopup()
updateAuctionPopup()
Updates the UI

Closing an Auction



View e interfaccia

Per questa seconda versione del progetto, a differenza della prima, era espressamente richiesto che tutto ciò che segue la fase di Login fosse contenuto in una singola pagina web. Per tale motivo questa versione implementa solamente due pagine: LoginPage e HomePage.

Nonostante in questa versione del progetto fosse sufficiente fornire accesso diretto ai file HTML delle due pagine, non essendoci necessità di pre-processare le pagine prima di inviarle all'utente (in altre parole, si poteva esporre l'indirizzo diretto ai file come "/HomePage.html"), si è comunque deciso di implementare un approccio simile alla prima versione del progetto, con due Servlet (LoginPageServlet e HomePageServlet) il cui unico scopo è recuperare i file HTML dalla cartella WEB-INF ed inviarle al client.

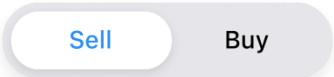
Questo è stato fatto per due motivi:

1. Evitare di esporre la struttura interna del server
2. Fornire anche in questo caso due URL comodi all'utente: "/login" e "/home", ritenuti più semplici ed intuitivi da utilizzare.

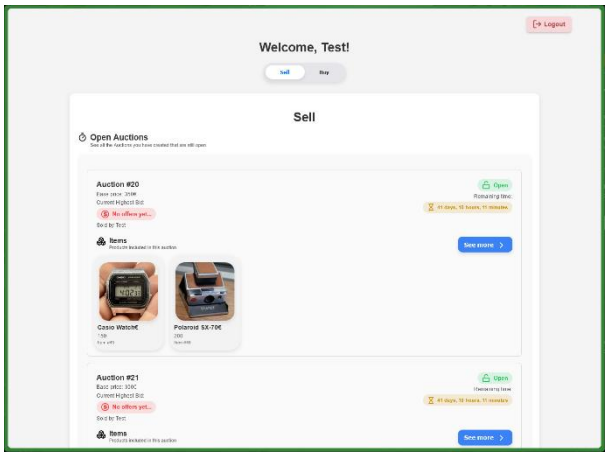
LoginPage gestisce internamente sia l'accesso che la registrazione: contiene al suo interno tutti e 3 i form che erano in precedenza divisi in 3 pagine separate, e li mostra e nasconde dinamicamente tramite JavaScript.

In maniera simile, la HomePage svolge tutti i ruoli prima divisi tra SellPage, BuyPage e AuctionDetailsPage.
A differenza della LoginPage, tuttavia, i due flow principali di questa schermata non sono propriamente esclusivi, dunque non si è ritenuto opportuno "nascondere" una delle due sezioni dietro un semplice pulsante a fondo pagina.

La soluzione che si è implementata è quella di una TabView, un componente mutuato dalle interfacce mobile e ampiamente utilizzato anche sul web per dividere più flow indipendenti ma equamente importanti.



Questo componente, situato nella parte alta della pagina ed immediatamente visibile, controlla la comparsa e scomparsa di due macro-contenitori *div*, uno contenente tutto ciò che prima si trovava in SellPage, l'altro contenente la BuyPage.

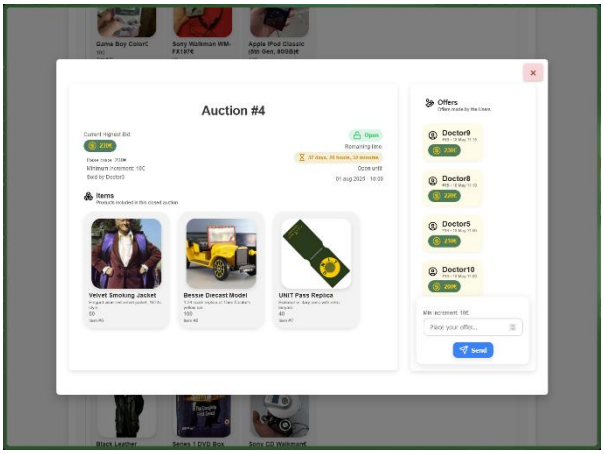


Esempio di pagina: SellPage

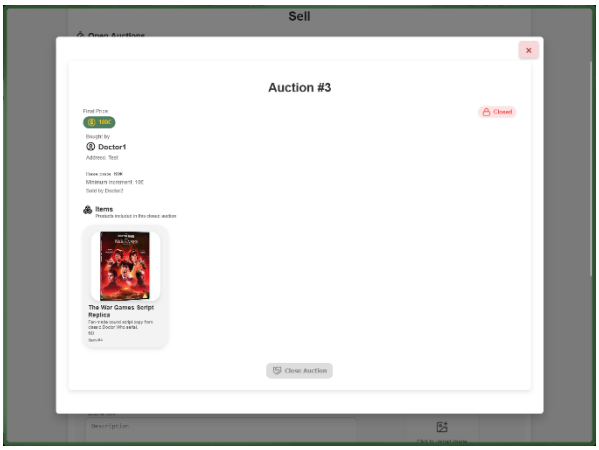
Per mostrare la pagina AuctionDetails, invece, si è optato per un'espediente diverso: invece di mostrarlo come pagina intera è stato inserito all'interno di un *popup*, mostrato al di sopra della vista principale della HomePage.

Questo consente di differenziare questa particolare sezione, il cui ruolo è di fatto secondario nella navigazione se comparato alle sezioni *Sell* e *Buy*, ed anche di unificare e centralizzare la logica dietro alla sua comparsa, aggiornamento e scomparsa.

Anche in questo caso la pagina presenta il pannello di destra, contenente la lista delle Offerte, che viene renderizzato solo in alcuni casi, come richiesto da specifica. In questa versione, però, la comparsa e scomparsa del pannello è gestita da JavaScript a lato client.



Pagina AuctionDetails nel popup



Popup AuctionDetails senza la lista delle Offerte