POLITECNICO
MILANO 1863

Traccia 1: aste online

Spangaro Francesco – 10734844

Ferlin Francesco – 10717750

Versione HTML pura (1/2)

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. 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 può essere incluso in aste diverse, da solo o con altri articoli.

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.

Versione HTML pura (2/2)

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).

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

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.

Data Requirement Analysis (1/2)

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. 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 può essere incluso in aste diverse, da solo o con altri articoli.

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.

Entities, attributes, relationships

Data Requirement Analysis (2/2)

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).

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.

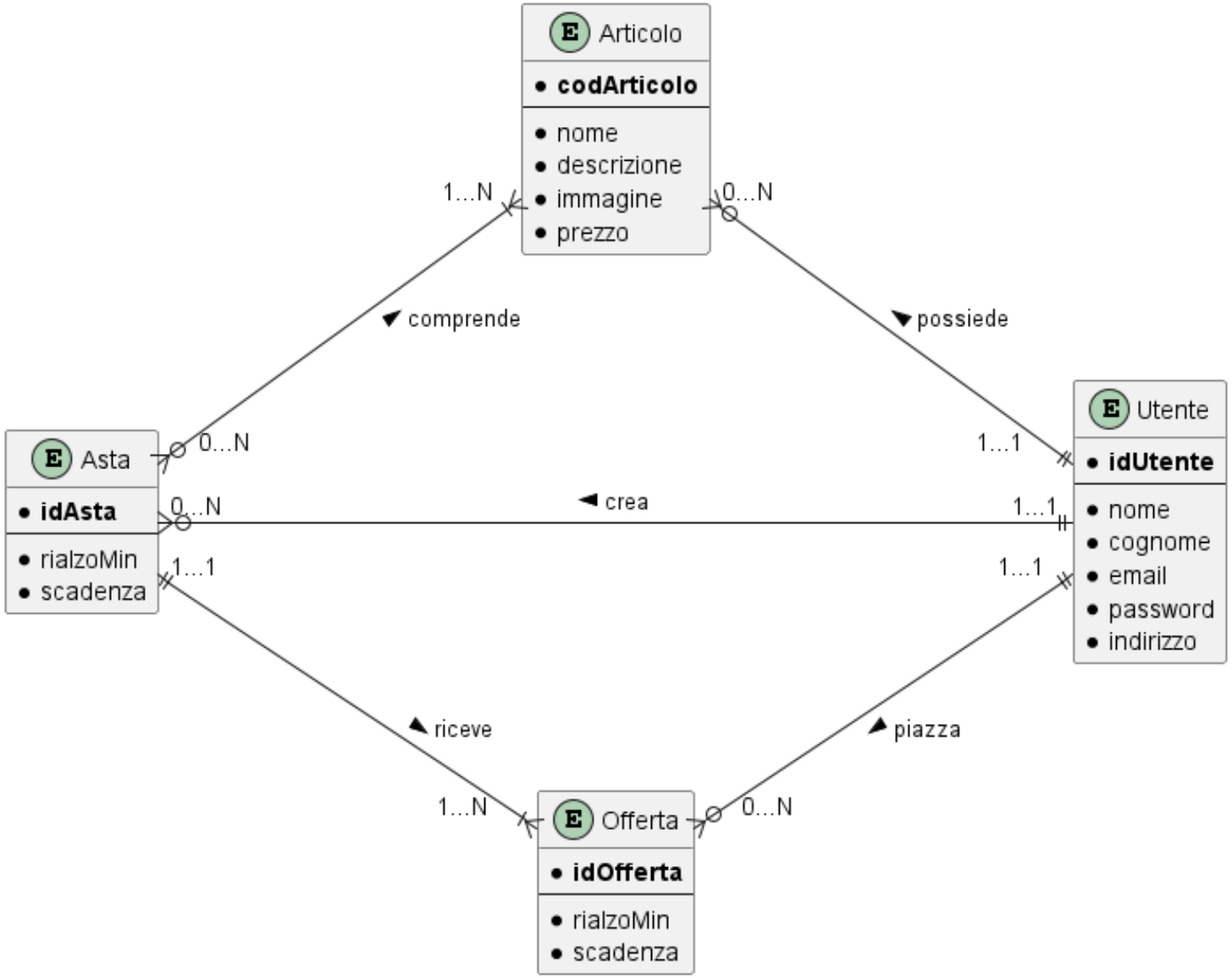
Dopo l'invio dell'offerta la pagina OFFERTA mostra l'elenco delle offerte aggiornate. La pagina ACQUISTO contiene anche un elenco delle offerte aggiudicate all'utente con i dati degli articoli e il prezzo finale.

Leggendo il testo notiamo inoltre che la tabella Utenti avrà come attributi, oltre all'indirizzo di spedizione, anche un nome e dei dati di login (saranno email e password, per scelta di design)

Entities, attributes, relationships

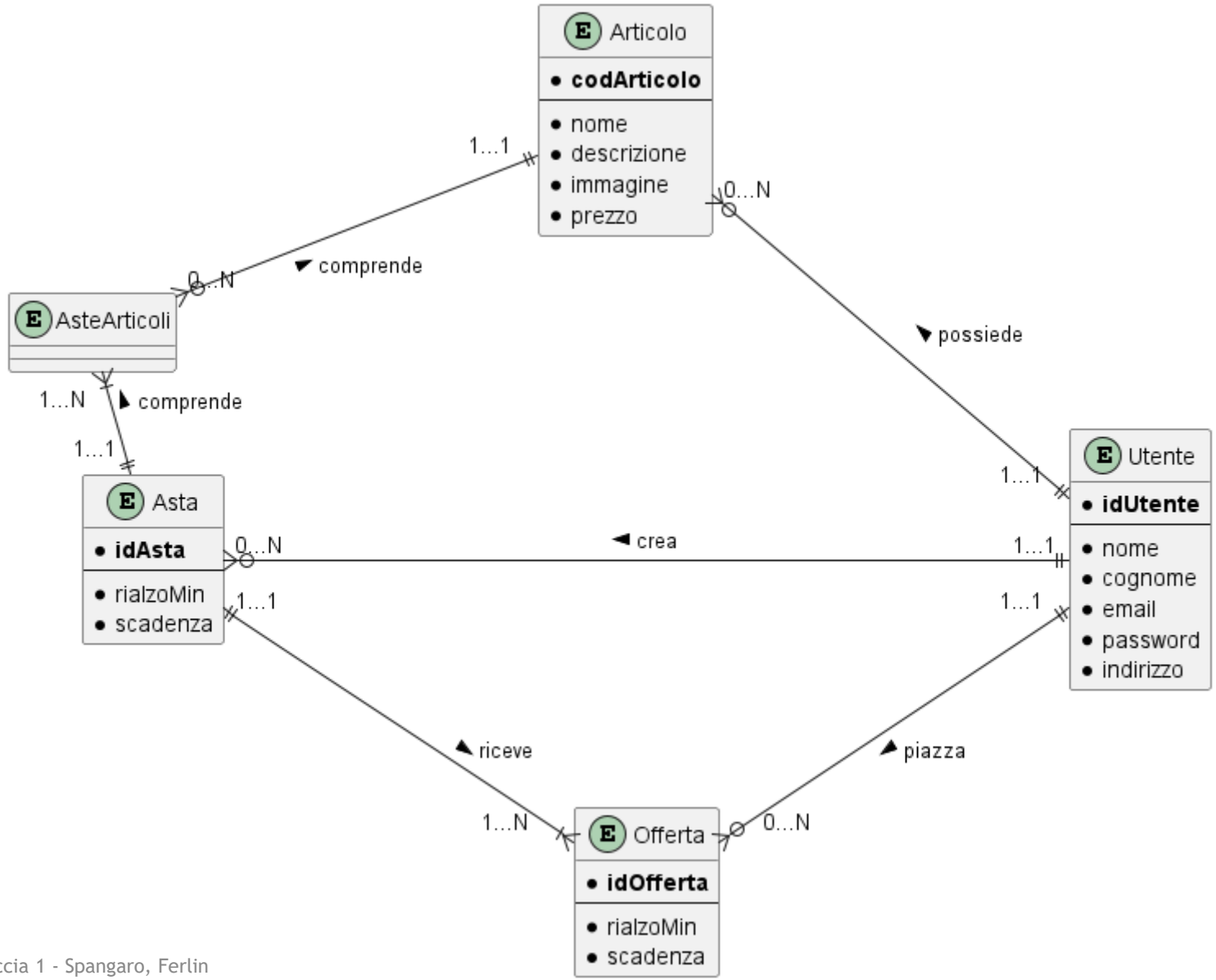
Tutti gli attributi sono non nulli

Schema Entità Relazioni (ER)



Tutti gli attributi sono non nulli

Schema Entità Relazioni Ristrutturato (ER)



Database Creation MySQL (1/3)

```
CREATE TABLE `articolo` (
        `codArticolo` int NOT NULL AUTO_INCREMENT,
        `nome` varchar(45) NOT NULL,
        `descrizione` text NOT NULL,
        `immagine` longblob NOT NULL,
        `prezzo` decimal(10,2) unsigned NOT NULL,
        `utente_idUtente` int NOT NULL,
        PRIMARY KEY (`codArticolo`),
        KEY `fk_articolo_utente1_idx` (`utente_idUtente`),
        CONSTRAINT `fk_articolo_utente1` FOREIGN KEY (`utente_idUtente`) REFERENCES `utente`(`idUtente`)
);


CREATE TABLE `asta` (
        `idAsta` int NOT NULL AUTO_INCREMENT,
        `rialzoMin` decimal(10,2) NOT NULL,
        `scadenza` datetime NOT NULL,
        `chiusa` tinyint(1) NOT NULL DEFAULT '0',
        PRIMARY KEY (`idAsta`)
);
```

Database Creation MySQL (2/3)

```
CREATE TABLE `astearticoli` (
	`articolo_codArticolo` int NOT NULL,
	`asta_idAsta` int NOT NULL,
	PRIMARY KEY (`articolo_codArticolo`,`asta_idAsta`),
	KEY `fk_asteArticoli_articolo_idx` (`articolo_codArticolo`),
	KEY `fk_asteArticoli_asta1_idx` (`asta_idAsta`),
	CONSTRAINT `fk_asteArticoli_articolo` FOREIGN KEY (`articolo_codArticolo`)
		REFERENCES `articolo` (`codArticolo`),
	CONSTRAINT `fk_asteArticoli_asta1` FOREIGN KEY (`asta_idAsta`) REFERENCES `asta` (`idAsta`)
);

CREATE TABLE `utente` (
	`idUtente` int NOT NULL AUTO_INCREMENT,
	`nome` varchar(45) NOT NULL,
	`cognome` varchar(45) NOT NULL,
	`email` varchar(45) NOT NULL,
	`password` varchar(255) NOT NULL,
	`indirizzo` varchar(45) NOT NULL,
	PRIMARY KEY (`idUtente`),
	UNIQUE KEY `email_UNIQUE` (`email`)
);
```

Database Creation MySQL (3/3)

```
CREATE TABLE `offerta` (
	`idOfferta` int NOT NULL AUTO_INCREMENT,
	`prezzoOfferto` decimal(10,2) NOT NULL,
	`dataOfferta` datetime NOT NULL,
	`utente_idUtente` int NOT NULL,
	`asta_idAsta` int NOT NULL,
	PRIMARY KEY (`idOfferta`),
	KEY `fk_offerta_utente1_idx` (`utente_idUtente`),
	KEY `fk_offerta_asta1_idx` (`asta_idAsta`),
	CONSTRAINT `fk_offerta_asta1` FOREIGN KEY (`asta_idAsta`) REFERENCES `asta` (`idAsta`),
	CONSTRAINT `fk_offerta_utente1` FOREIGN KEY (`utente_idUtente`) REFERENCES `utente` (`idUtente`)
);
```

Application Requirement Analysis (1/3)

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. 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 può essere incluso in aste diverse, da solo o con altri articoli.

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.

Pages(views), View components, Events, **Actions**

Application Requirement Analysis (2/3)

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).

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.

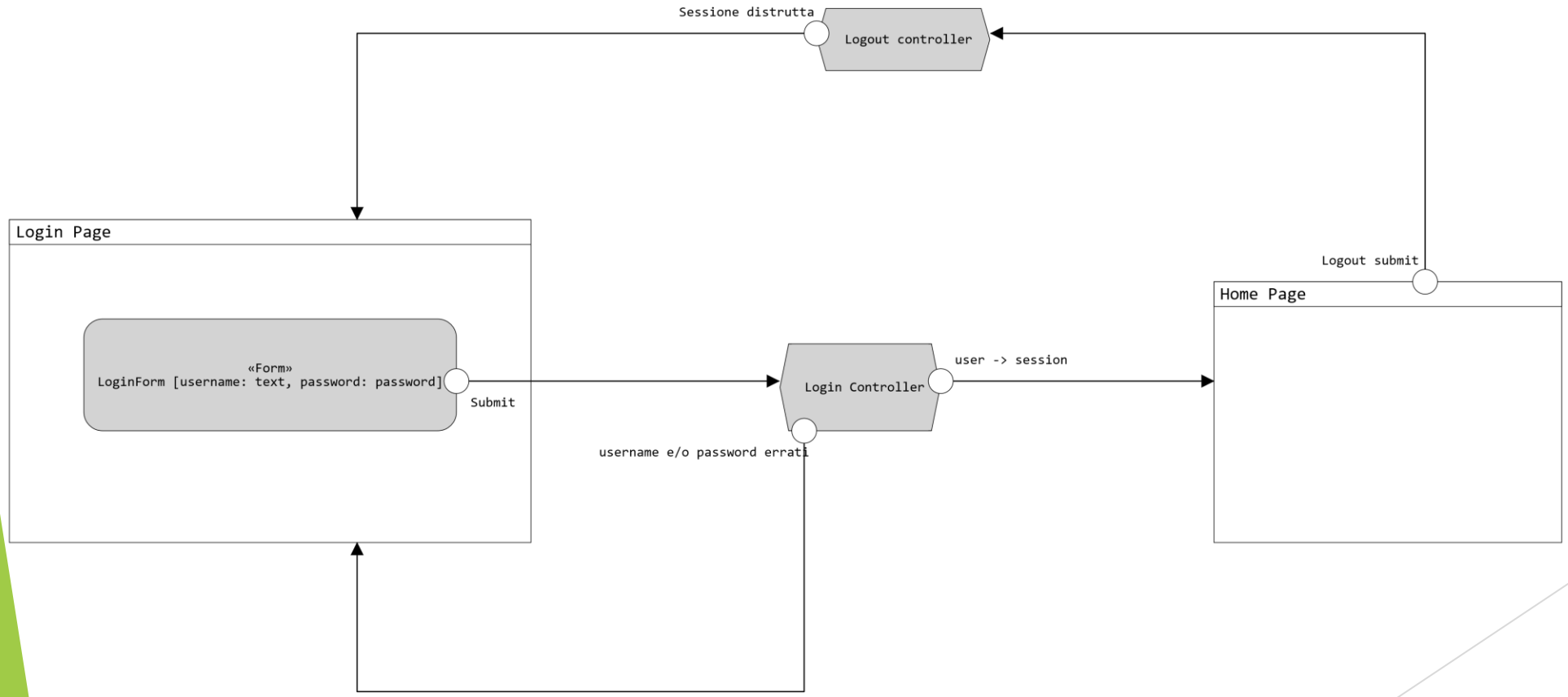
Dopo l'invio dell'offerta la pagina OFFERTA mostra l'elenco delle offerte aggiornate. La pagina ACQUISTO contiene anche un elenco delle offerte aggiudicate all'utente con i dati degli articoli e il prezzo finale.

Pages(views), View components, Events, **Actions**

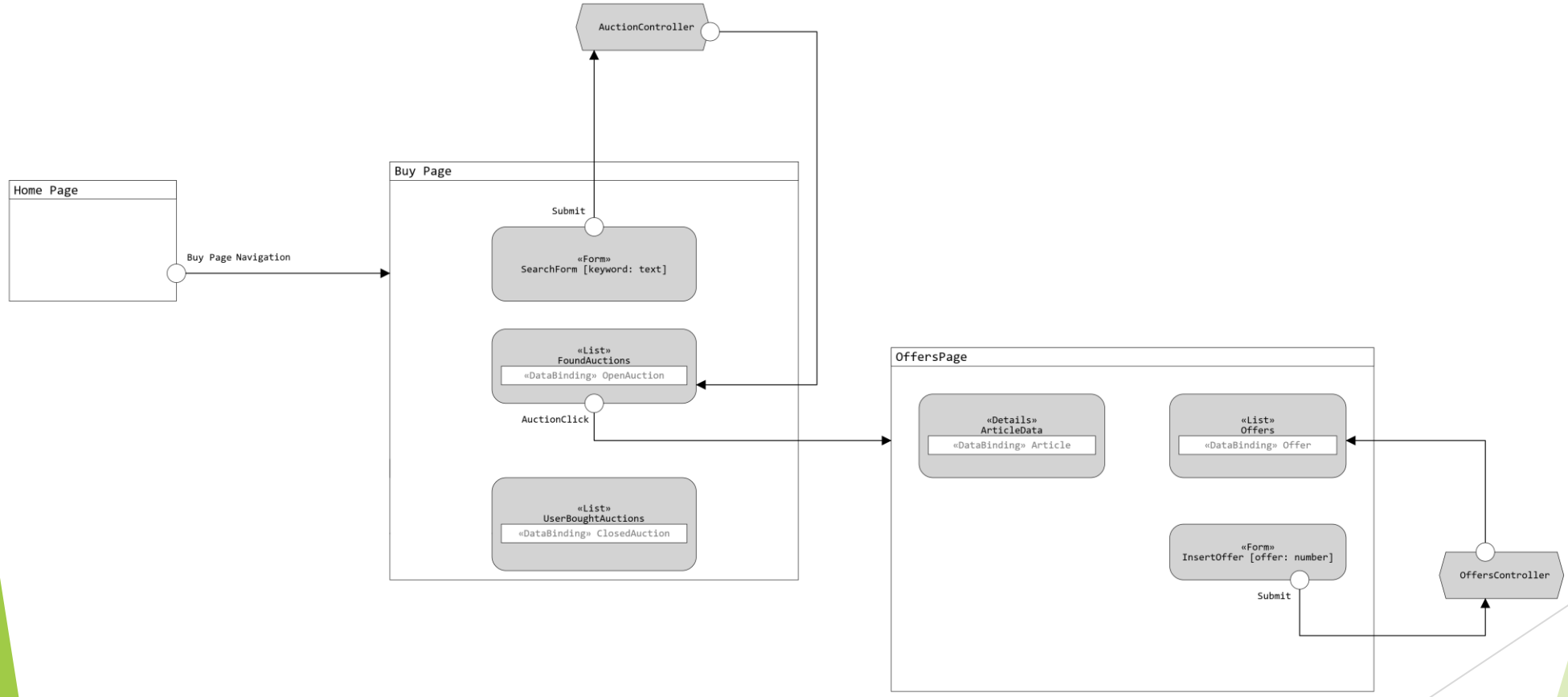
Application Requirement Analysis (3/3)

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.

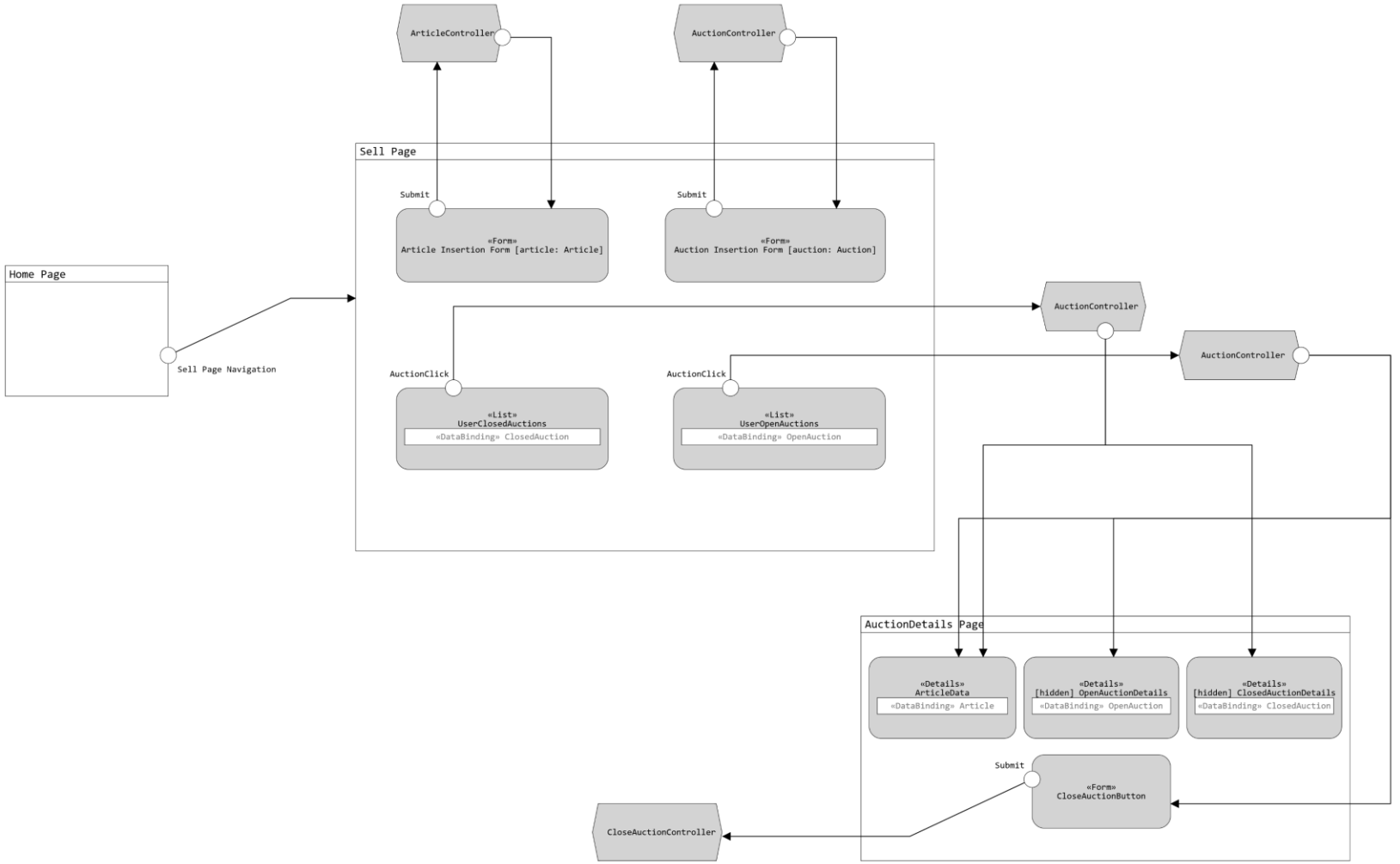
Design Applicativo (IFML - Login)



Design Applicativo (IFML – Buy and Offers Page)

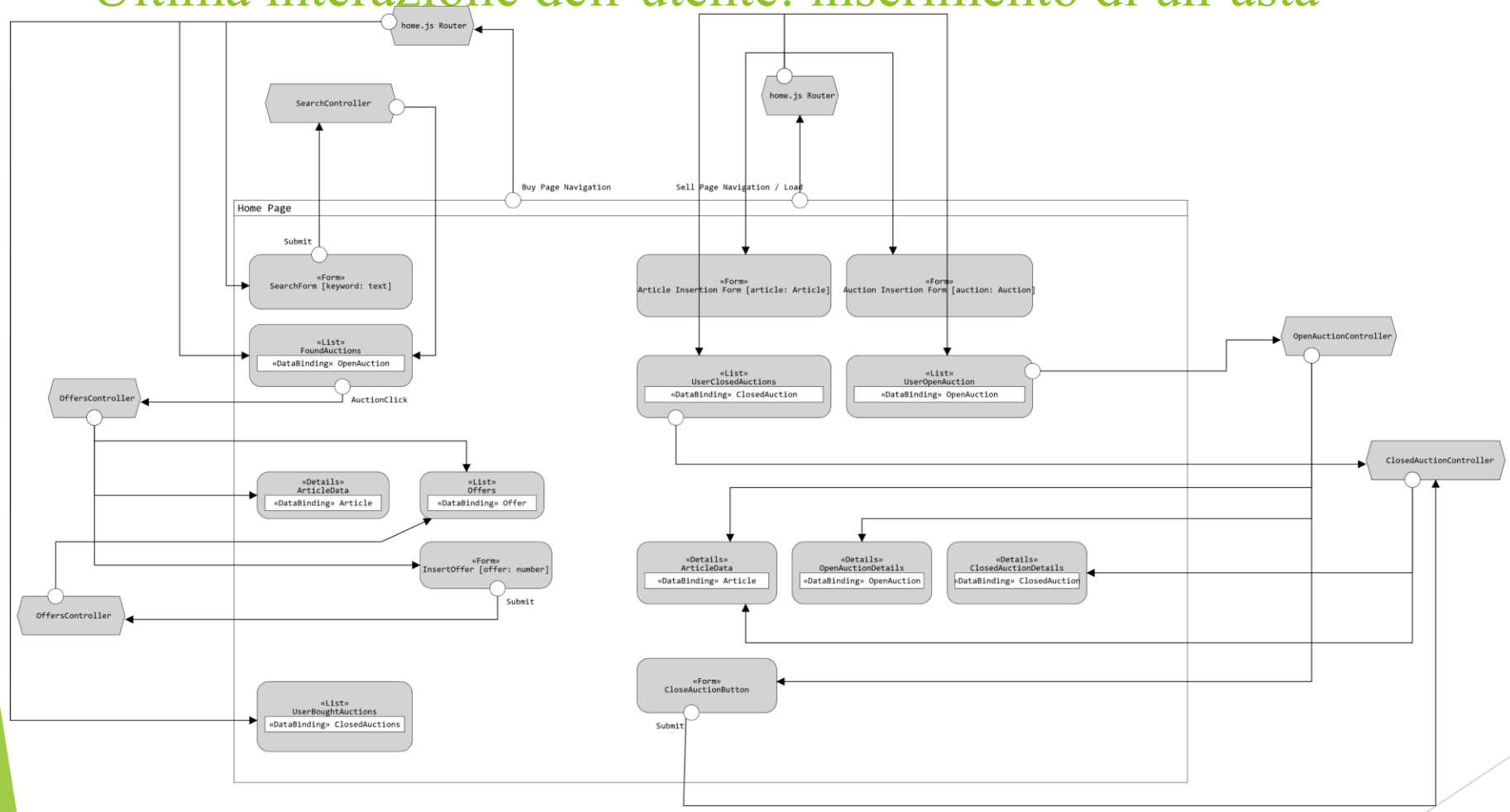


Design Applicativo (IFML – Sell and Auction Details Page)



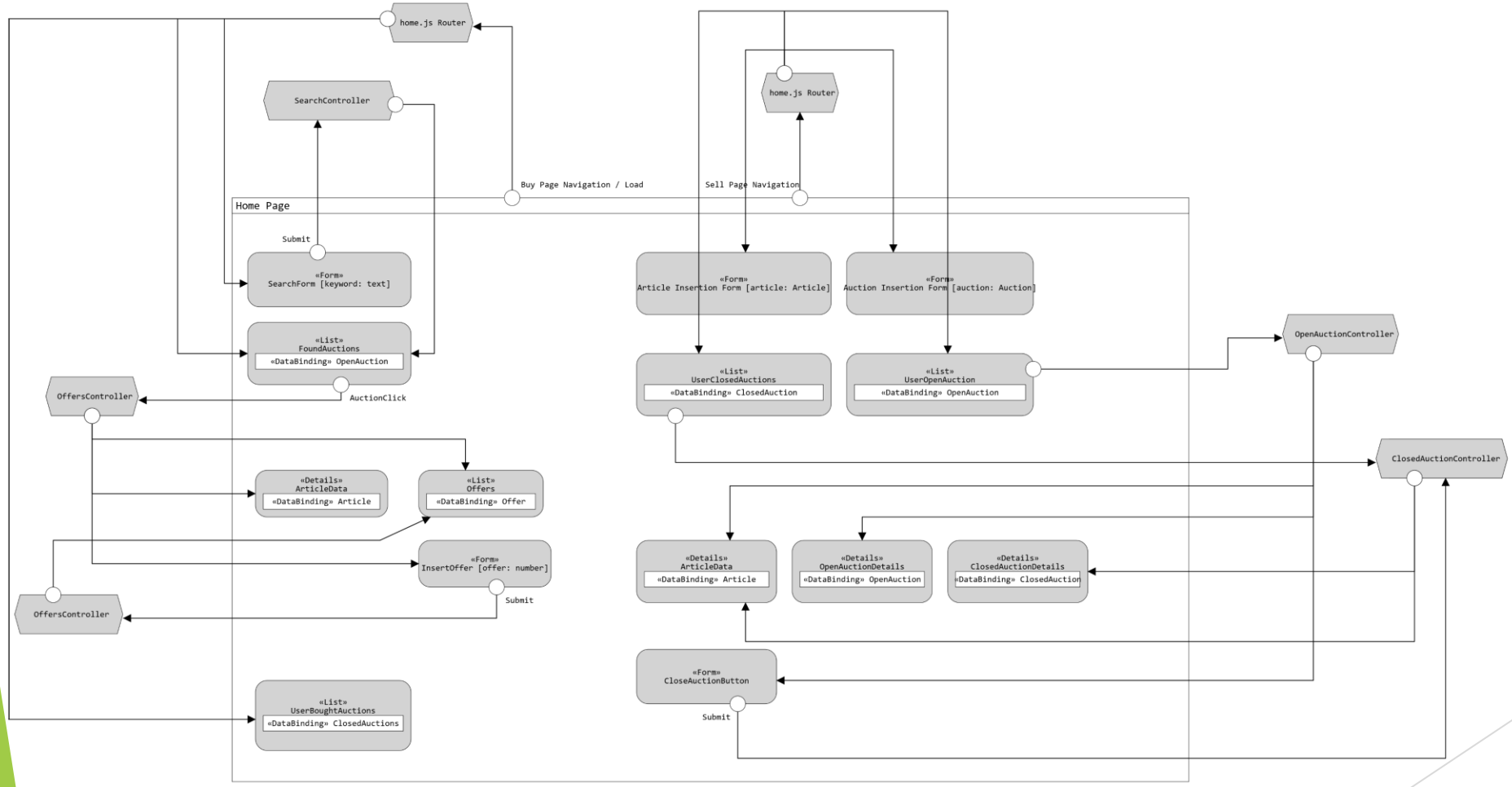
Design Applicativo (IFML - RIA)

Ultima interazione dell'utente: inserimento di un'asta



Design Applicativo (IFML - RIA)

Ultima interazione dell'utente: altri casi



HTMLPure – DAO, Servlet and Beans

DAOs:

- ArticleDao:
  - findAllArticles(int) : List<Article>
  - insertArticle(String, String, InputStream, double, int) : int
- AuctionDao:
  - findAuctions(int, boolean) : List<Auction>
  - findAuctionByIds(int, int) : ExtendedAuction
  - findUserBoughtAuctions(int) : List<ClosedAuction>
  - findOpenAuctionById(int) : OpenAuction
  - closeAuction(int, int) : int
  - findAuctionByWord(String) : List<Auction>
  - insertAuction(int, LocalDateTime, List<Integer>, int) : int
- LoginDao:
  - findUser(String, char[]) : User
- OffersDao
  - insertOffer(int, int, double, LocalDateTime) : InsertionResult

Beans:

- Article
- Auction
- ClosedAuction
- ExtendedAuction
- InsertionState
- LoginPageArgs
- Offer
- OffersPageArgs
- OpenAuction
- SellPageArgs
- User

Servlets:

- ArticleController
- AuctionController
- CloseAuctionController
- LoginController
- LogoutController
- OffersController

- AuctionDetailsPage
- BuyPage
- HomePage
- LoginPage
- OffersPage
- SellPage

RIA – DAO, Servlet and Beans

DAOs:

- ArticleDao:
  - findAllArticles(int) : List<Article>
  - insertArticle(String, String, InputStream, double, int) : int
- AuctionDao:
  - findAuctions(int, boolean) : List<Auction>
  - findAuctionByIds(int, int) : ExtendedAuction
  - findUserBoughtAuctions(int) : List<ClosedAuction>
  - findOpenAuctionById(int) : OpenAuction
  - closeAuction(int, int) : int
  - findAuctionByWord(String) : List<Auction>
  - insertAuction(int, LocalDateTime, List<Integer>, int) : int
- LoginDao:
  - findUser(String, char[]) : User
- OffersDao
  - insertOffer(int, int, double, LocalDateTime) : InsertionResult

Beans:

- Article
- Auction
- ClosedAuction
- ExtendedAuction
- InsertionSuccessful
- Offer
- OpenAuction
- ParsingError
- User

Servlets:

- ArticleController
- AuctionController
- ClosedAuctionController
- OpenAuctionController
- LoginController
- LogoutController
- OffersController
- SearchController

Views & view components

- router
  - set(Page, args?)
  - setById(pageId: string, args?)
  - get() : Page
  - isHomePage() : boolean
- buyPage : Page
- sellPage : Page
- auctionDetailsPage : Page
- offersPage : Page
- [Page]
  - create()
  - mount(args: URLSearchParams)
  - unmount()
- auctionRepository
  - getOpenAuctionById(id)
  - closeAuction(FormData)
  - getClosedAuction()
  - getOpenAuction()
  - getAuctionByIds(id)
  - insertAuction(FormData)
  - searchAuction(keyword)
  - getBoughtAuctions()
- articleRepository
  - findAllArticles()
  - insertArticle(FormData)
- offerRepository
  - insertOffer(FormData)

Controller / event handler

Ultima azione: Inserimento di un'asta

Client side		Server side	
Evento	**Controllore**	**Evento**	**Controllore**
LoginPage→loginForm→Submit	fetch()	POST(String, String)	findUser(String, char[])
HomePage→**load**	await router.doRoute()	GET()	findUserBoughtAuctions(int, true)
{Any Page}→**Buy Navigation**	await router.setById('buy') await router.triggerPageChange()	GET()	findUserBoughtAuctions(int, true)
BuyPage→searchForm→Submit	await router.setById('buy', keyword) await router.triggerPageChange()	GET(String)	findAuctionByWord(String)
BuyPage→foundAuctions→ click on auction	await router.setById('offers', auctionId) await router.triggerPageChange()	GET(int)	findOpenAuctionById(int)
OffersPage→insertOfferForm→ Submit	await page.mutateState()	POST(int, double)	insertOffer(int, int, double, LocalDateTime)
{Any Page}→**Sell Navigation**	await router.setById('sell') await router.triggerPageChange()	GET()	findClosedAuctions(int) findOpenAuctions(int) findArticles(int)
SellPage→ClosedAuctions→click on auction	await router.setById('ClosedAuction') await router.triggerPageChange()	GET(int)	findAuctionByIds(int, int)
SellPage→OpenAuctions→click on auction	await router.setById('OpenAuction') await router.triggerPageChange()	GET(int)	findAuctionByIds(int, int)
SellPage→ArticleInsertionForm→ Submit	await page.mutateState()	POST(String, String, InputStream, double)	insertArticle(String, String, InputStream, double, int)
SellPage→AuctionInsertionForm→ Submit	await page.mutateState()	POST(LocalDateTime, List<int>, int)	insertAuction(int, LocalDateTime, List<int>, int)

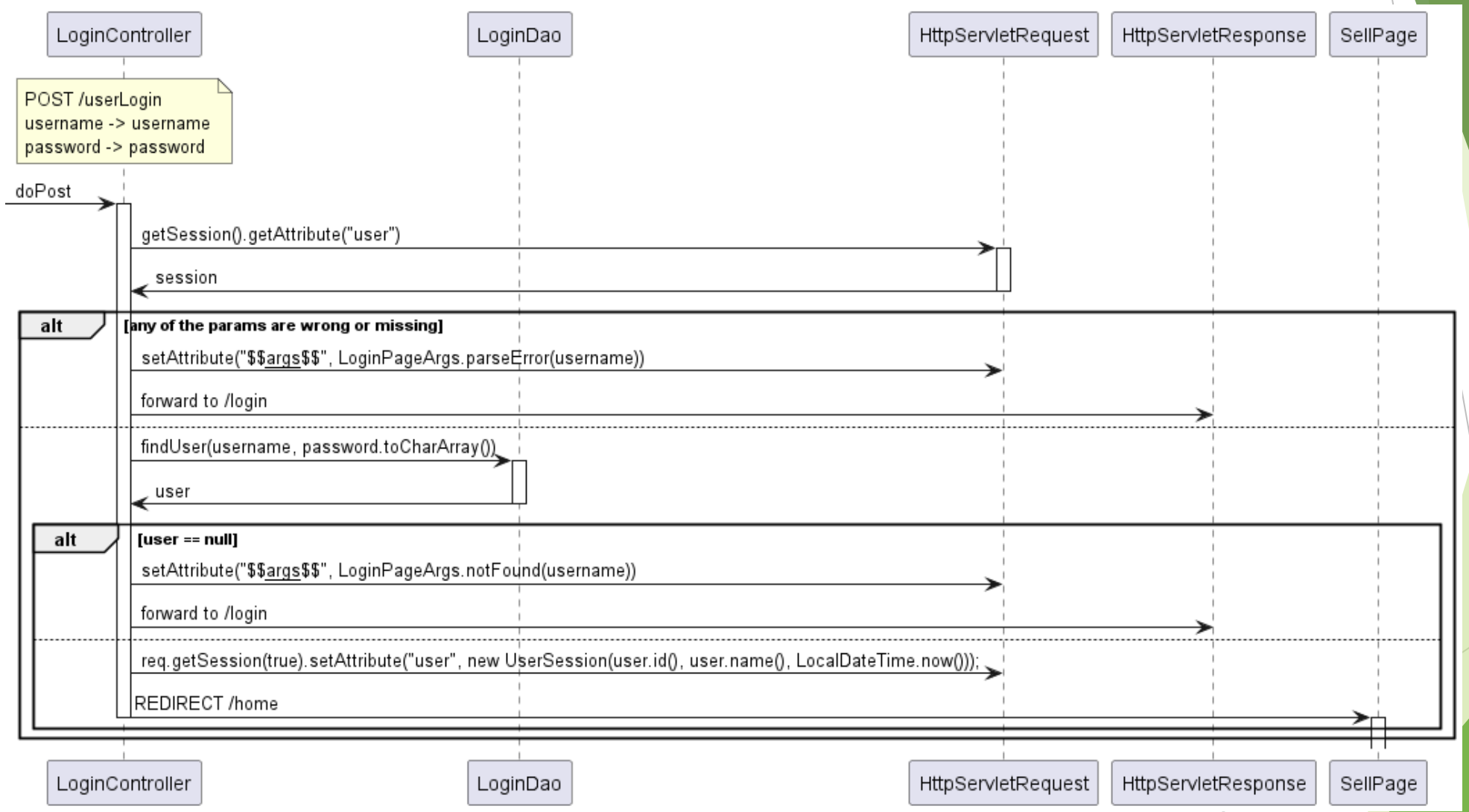
Controller / event handler

Ultima azione: Altri Casi

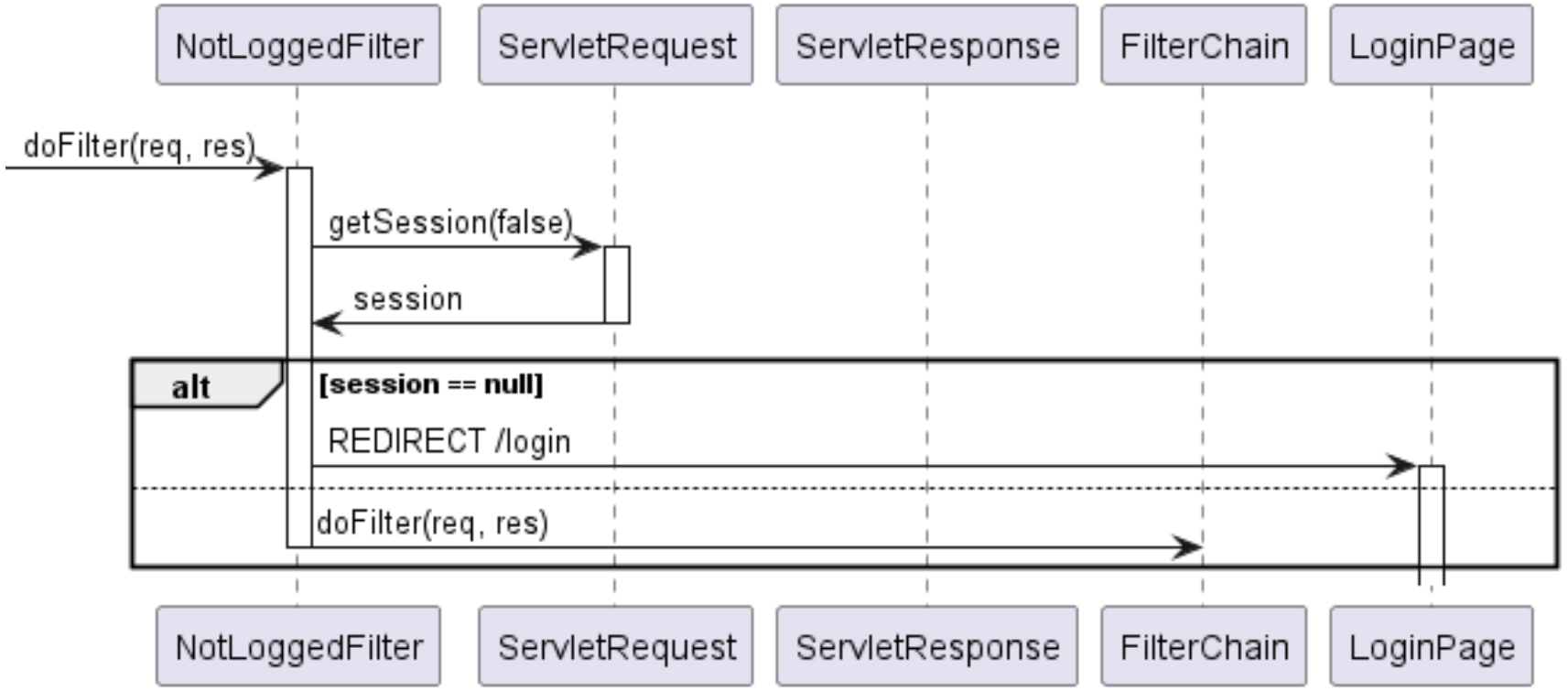
Client side		Server side	
Evento	**Controllore**	**Evento**	**Controllore**
LoginPage→loginForm→Submit	fetch()	POST(String, String)	findUser(String, char[])
HomePage→**load**	await router.doRoute()	GET()	findClosedAuctions(int) findOpenAuctions(int) findArticles(int)
{Any Page}→**Buy Navigation**	await router.setById('buy') await router.triggerPageChange()	GET()	findUserBoughtAuctions(int, true)
BuyPage→searchForm→Submit	await router.setById('buy', keyword) await router.triggerPageChange()	GET(String)	findAuctionByWord(String)
BuyPage→foundAuctions→ click on auction	await router.setById('offers', auctionId) await router.triggerPageChange()	GET(int)	findOpenAuctionById(int)
OffersPage→insertOfferForm→ Submit	await page.mutateState()	POST(int, double)	insertOffer(int, int, double, LocalDateTime)
{Any Page}→**Sell Navigation**	await router.setById('sell') await router.triggerPageChange()	GET()	findClosedAuctions(int) findOpenAuctions(int) findArticles(int)
SellPage→ClosedAuctions→click on auction	await router.setById('ClosedAuction') await router.triggerPageChange()	GET(int)	findAuctionByIds(int, int)
SellPage→OpenAuctions→click on auction	await router.setById('OpenAuction') await router.triggerPageChange()	GET(int)	findAuctionByIds(int, int)
SellPage→ArticleInsertionForm→ Submit	await page.mutateState()	POST(String, String, InputStream, double)	insertArticle(String, String, InputStream, double, int)
SellPage→AuctionInsertionForm→ Submit	await page.mutateState()	POST(LocalDateTime, List<int>, int)	insertAuction(int, LocalDateTime, List<int>, int)

Sequence diagram
HTMLPure

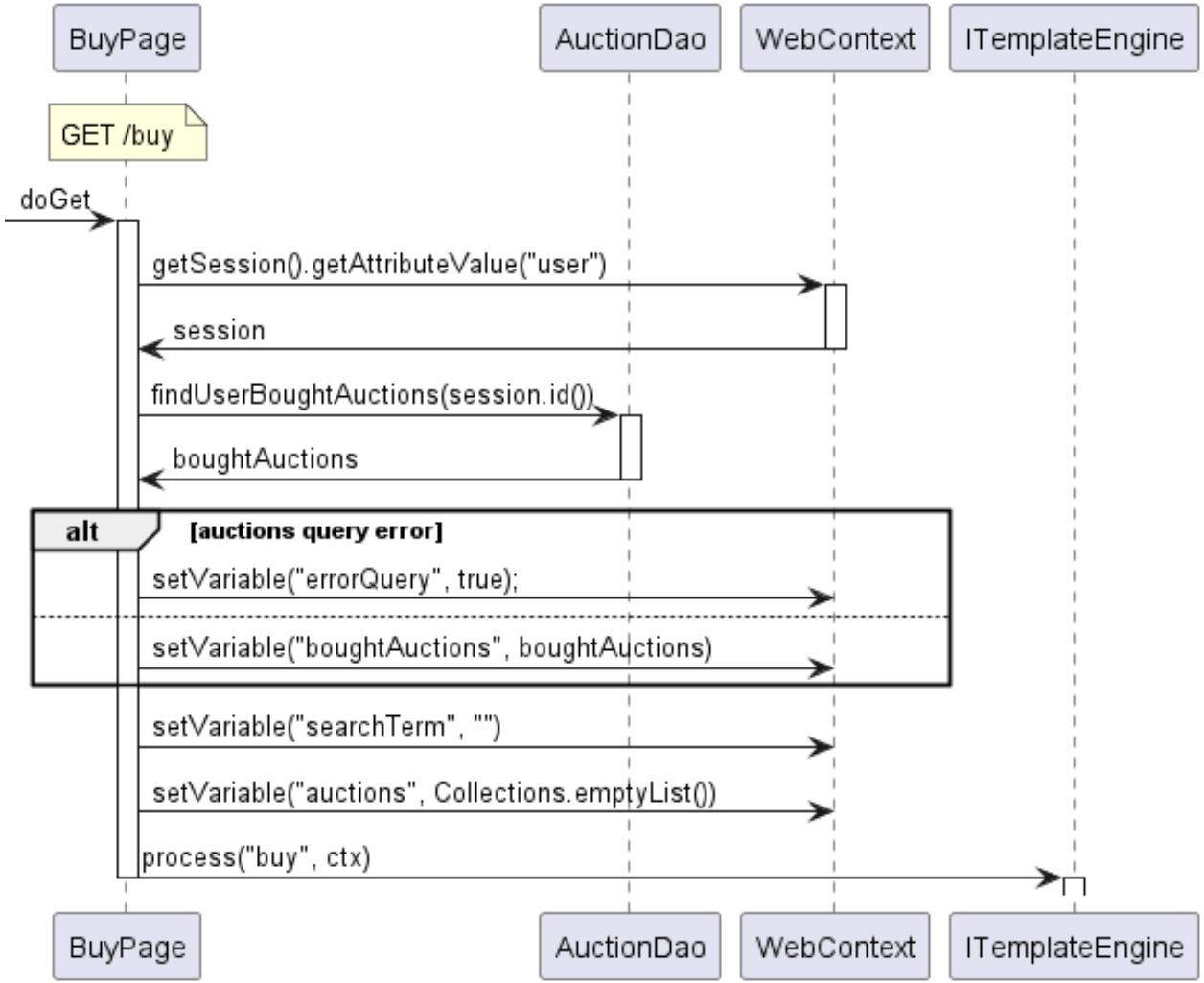
Sequence Diagram – Login (HTMLPure)



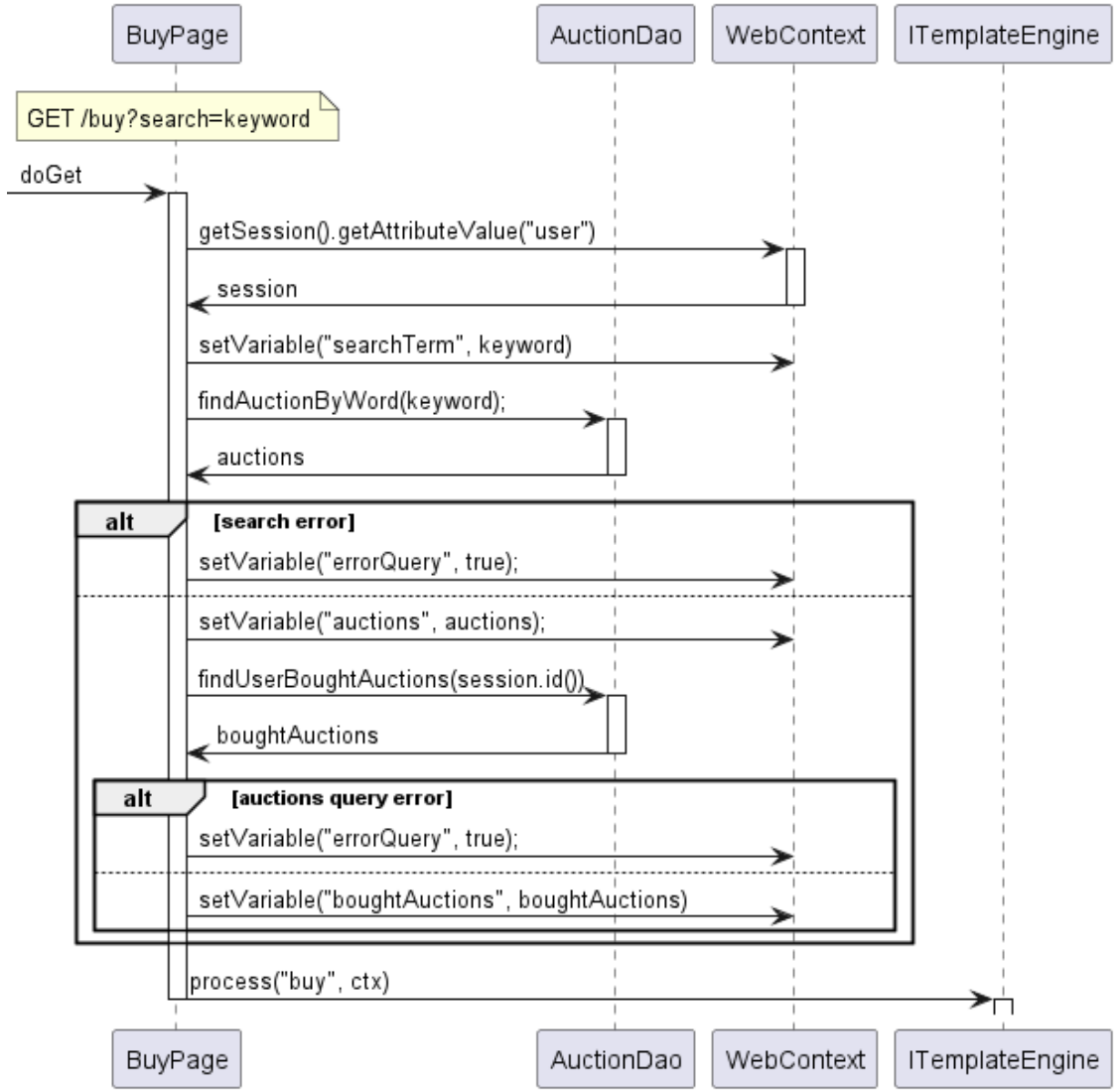
Sequence Diagram – NotLoggedFilter (HTMLPure)



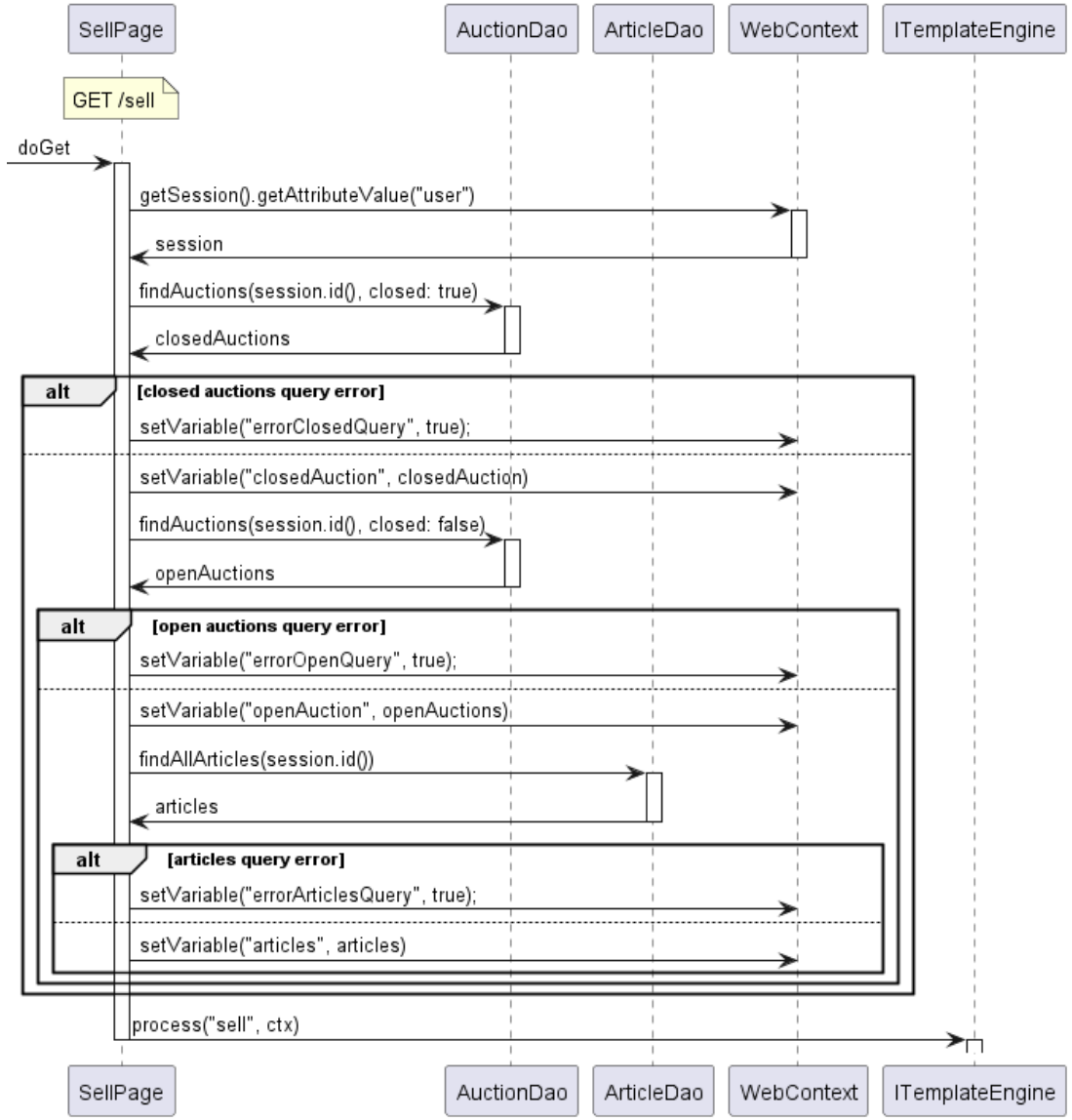
Sequence Diagram – BuyPage (HTMLPure)



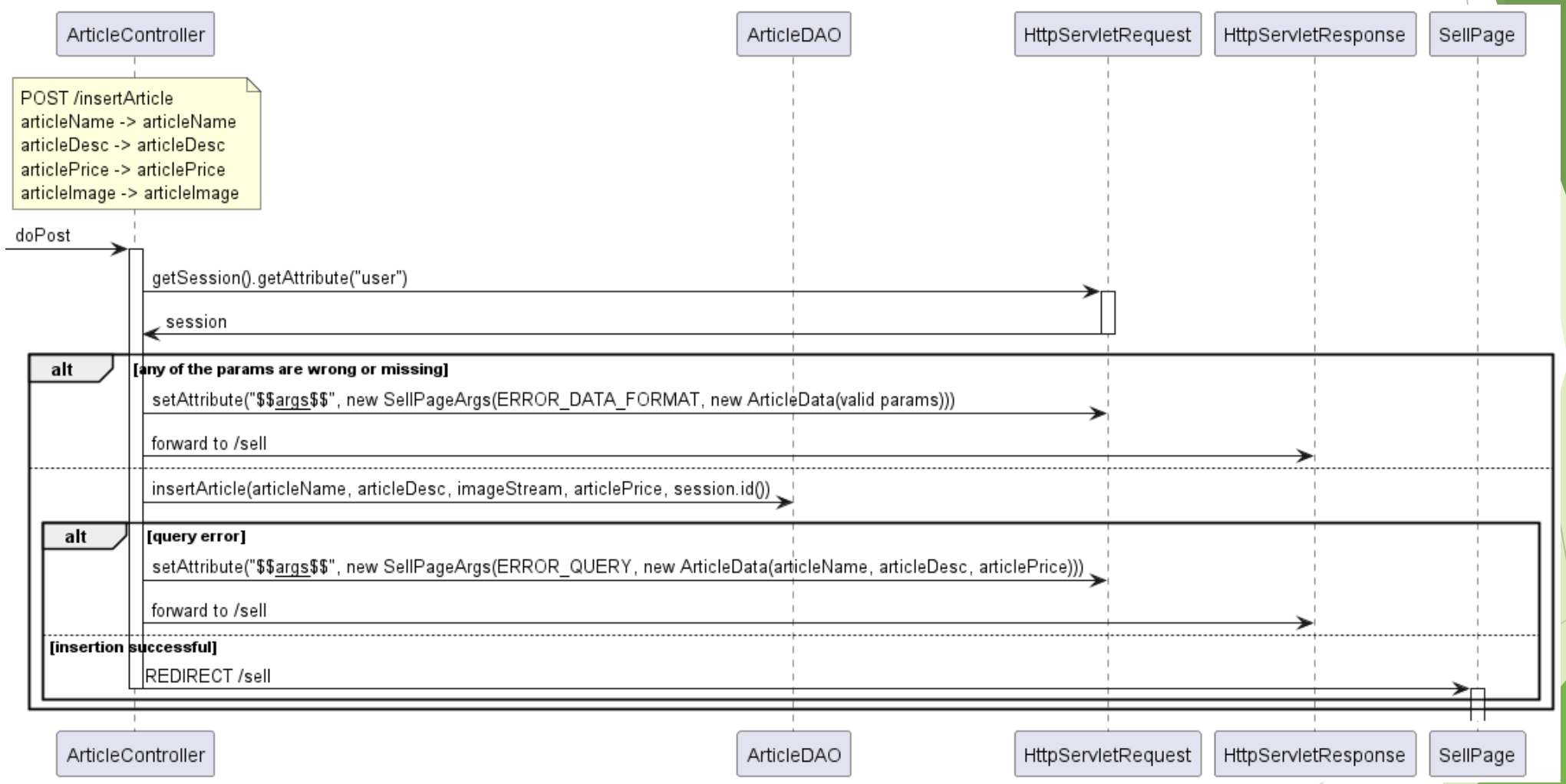
Sequence Diagram – Search (HTMLPure)



Sequence Diagram – SellPage (HTMLPure)

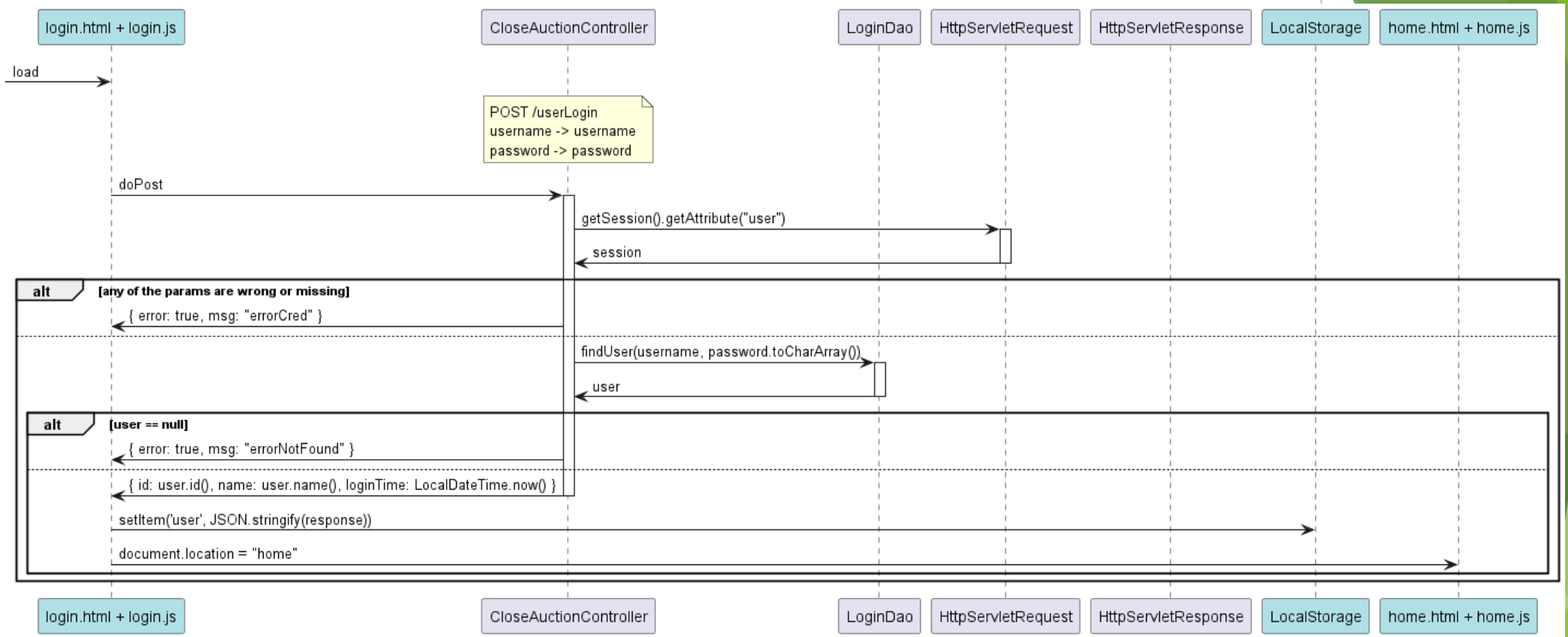


Sequence Diagram – Create Article (HTMLPure)

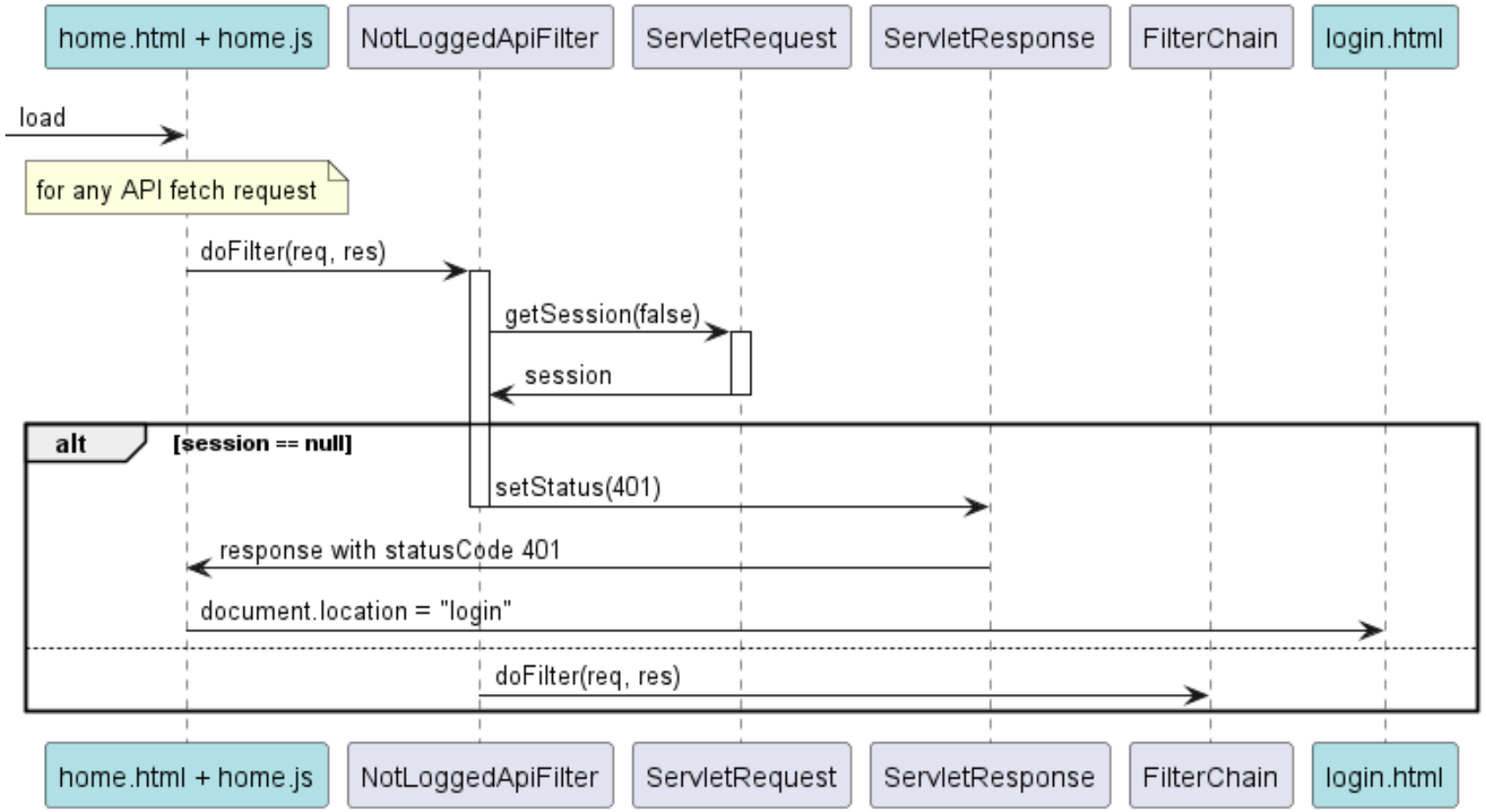


Sequence diagram
RIA

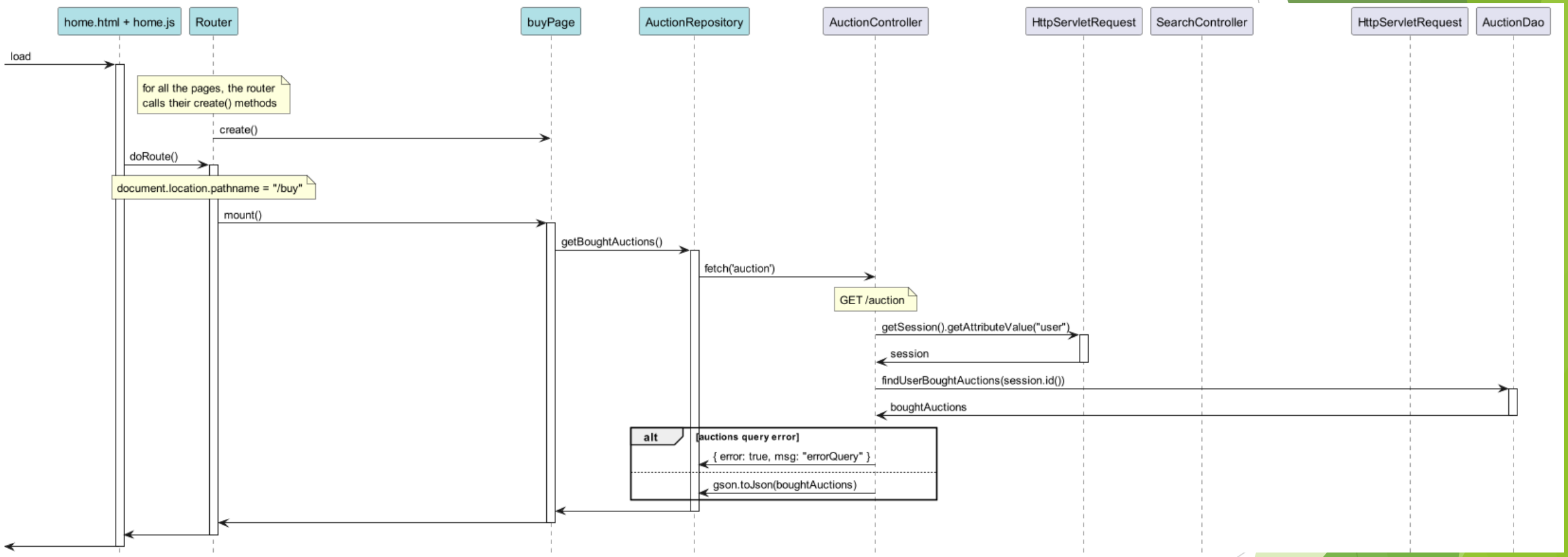
Sequence Diagram – Login (RIA)



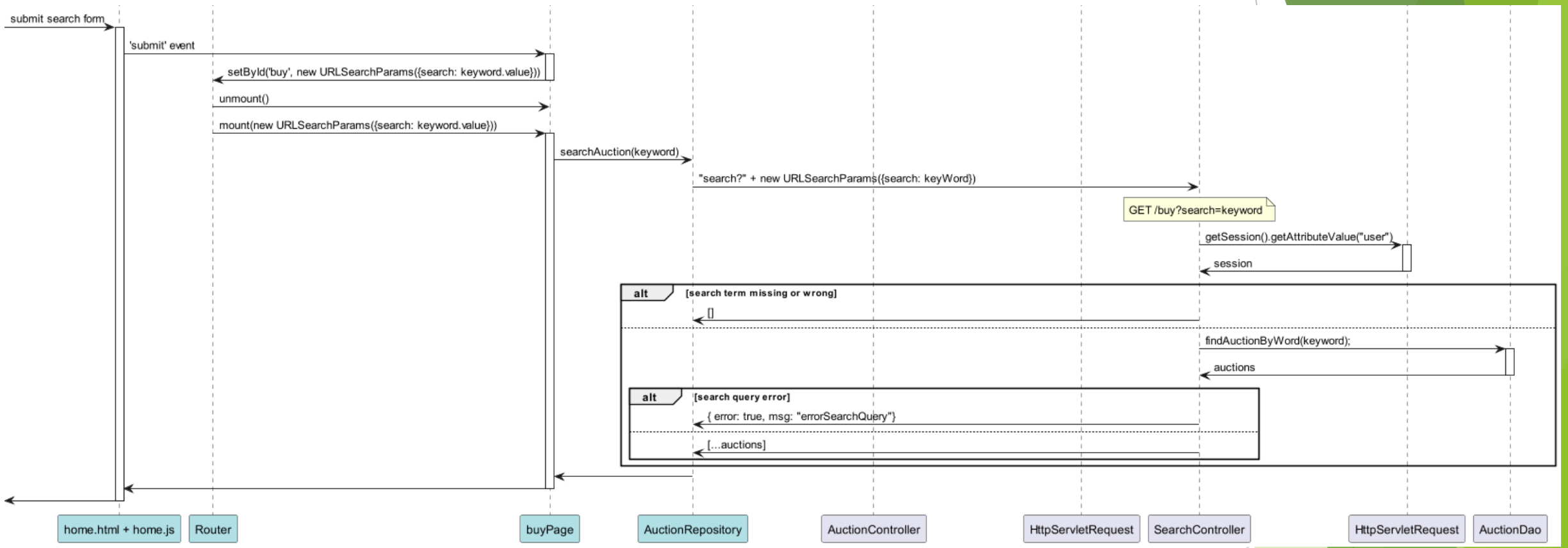
Sequence Diagram – LoginApiFilter (RIA)



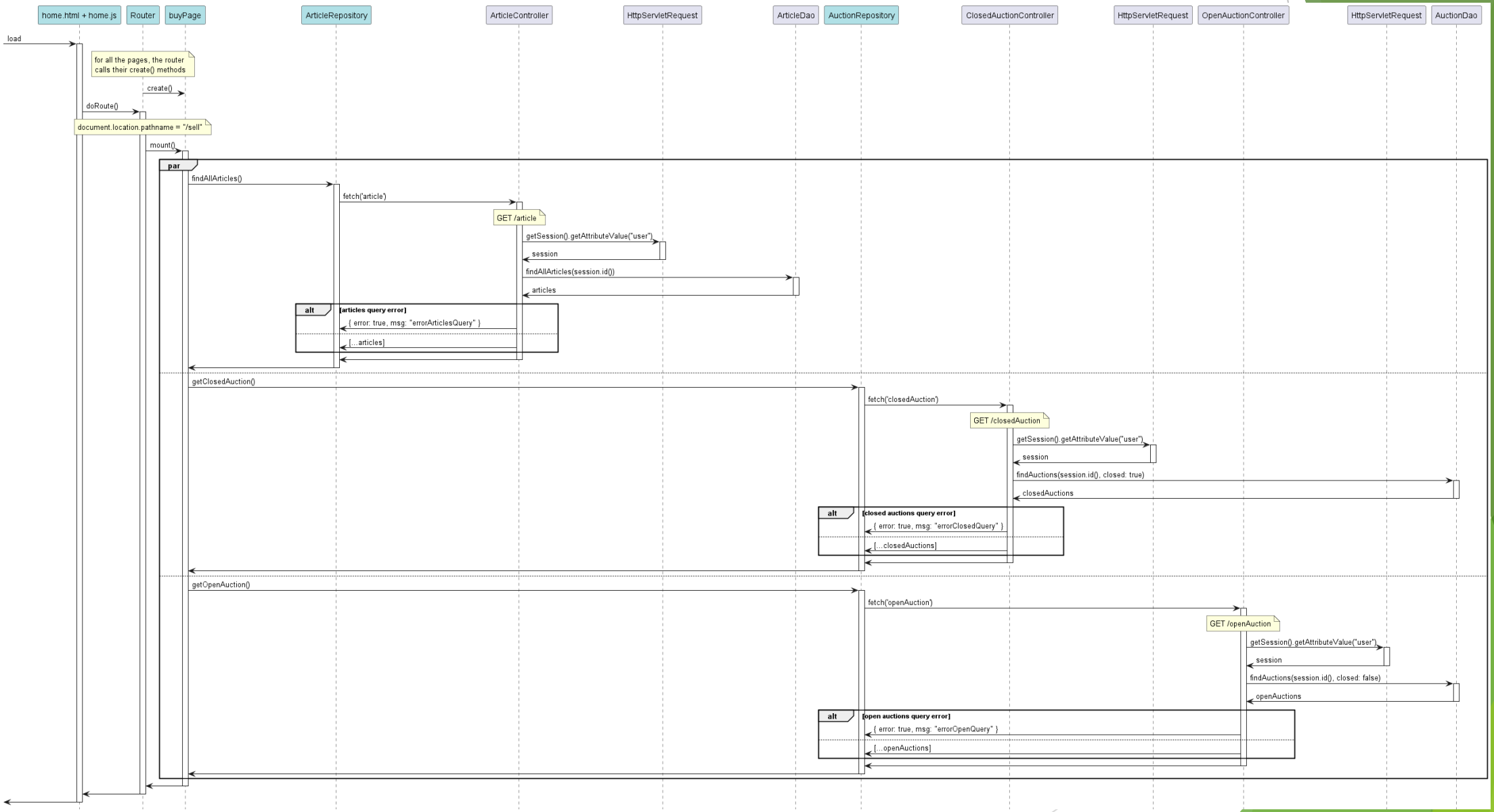
Sequence Diagram – BuyPage (RIA) 1/2



Sequence Diagram – BuyPage (RIA) 2/2



Sequence Diagram – SellPage (RIA) 1/2



Sequence Diagram – SellPage (RIA) 1/2