

Esercizio 5 – Gruppo 83

Davide Mornatta

Leonardo Panseri

Lea Zancani

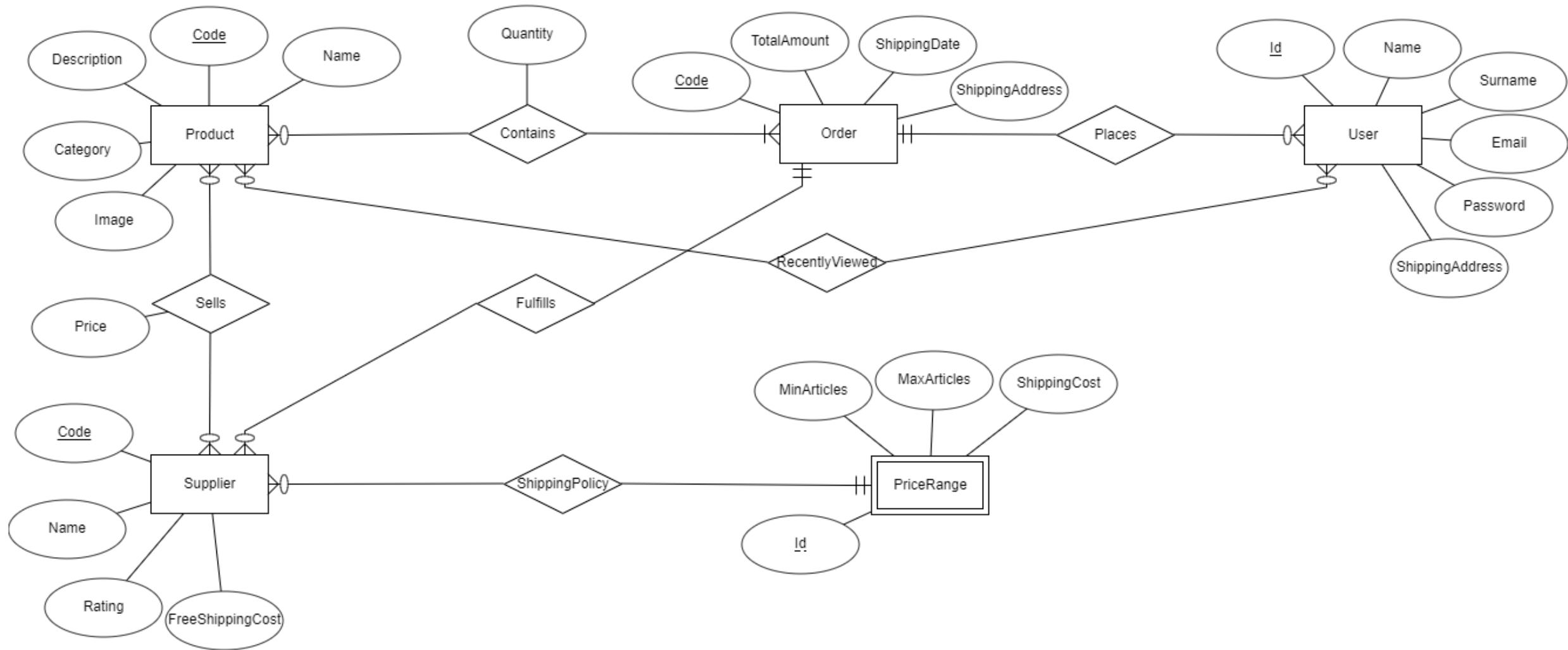
Carrello con più fornitori e ordine

Un'applicazione di commercio elettronico consente all'utente (acquirente) di visualizzare un catalogo di prodotti venduti da diversi fornitori, inserire prodotti in un carrello della spesa e creare un ordine di acquisto a partire dal contenuto del carrello. Un **prodotto** ha un **codice** (campo chiave), un **nome**, una **descrizione**, una **categoria merceologica** e una **foto**. Lo stesso **prodotto** (cioè codice prodotto) può essere **venduto da più fornitori a prezzi differenti**. Un **fornitore** ha un **codice**, un **nome**, una **valutazione** da 1 a 5 stelle e una **politica di spedizione**. Un **utente** ha un **nome**, un **cognome**, un'**e-mail**, una **password** e un **indirizzo di spedizione**. La politica di spedizione precisa il prezzo della spedizione in base al numero di articoli ordinati. Ogni **fornitore** è libero di definire **fasce di spesa**. Una fascia di spesa ha un **numero minimo**, un **numero massimo** e un **prezzo**. Ad esempio: da 1 a 3 articoli 15€, da 4 a 10 articoli 20€, oltre a 10 articoli, ecc. Oltre alla fascia di spesa, il fornitore può anche indicare un **importo in euro** oltre al quale la spedizione è gratuita. Se il totale supera la soglia per la gratuità della spedizione, la spedizione è gratuita indipendentemente dal numero di articoli.

Un **ordine** ha un **codice**, il **nome del fornitore**, l'**elenco dei prodotti**, un **valore totale** composto dalla somma del valore dei prodotti e delle spese di spedizione, una **data di spedizione** e l'**indirizzo di spedizione** dell'utente. I valori degli attributi di un ordine sono memorizzati esplicitamente nella base di dati indipendentemente dai dati del carrello.

Entities, **attributes**, **relationships**

Database design



Database schema

```
CREATE TABLE `user` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  `password` varchar(255) NOT NULL,
  `shipping_address` varchar(128) NOT NULL,
  PRIMARY KEY (`id`)
)
```

```
CREATE TABLE `product` (
  `code` int NOT NULL,
  `name` varchar(45) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  `category` varchar(45) DEFAULT NULL,
  `image` blob,
  PRIMARY KEY (`code`)
)
```

```
CREATE TABLE `order` (
  `code` int NOT NULL AUTO_INCREMENT,
  `total_amount` float NOT NULL,
  `shipping_date` date NOT NULL,
  `shipping_address` varchar(128) NOT NULL,
  `supplier_code` int NOT NULL,
  `user_id` int NOT NULL,
  PRIMARY KEY (`code`),
  KEY `supplier_code_idx` (`supplier_code`),
  KEY `user_id_idx` (`user_id`),
  CONSTRAINT `supplier_code` FOREIGN KEY
    (`supplier_code`) REFERENCES
    `supplier` (`code`)
    ON UPDATE CASCADE,
  CONSTRAINT `user_id` FOREIGN KEY (`user_id`)
    REFERENCES `user` (`id`)
    ON UPDATE CASCADE
)
```

```
CREATE TABLE `supplier` (
  `code` int NOT NULL,
  `name` varchar(45) NOT NULL,
  `rating` int DEFAULT NULL,
  `free_shipping_cost` float DEFAULT NULL,
  PRIMARY KEY (`code`)
)
```

```
CREATE TABLE `order_contains` (
  `order_code` int NOT NULL,
  `product_code` int NOT NULL,
  `quantity` int NOT NULL,
  PRIMARY KEY(`order_code`,`product_code`),
  KEY `product_code_idx` (`product_code`),
  CONSTRAINT `order_contains`
    FOREIGN KEY (`order_code`)
    REFERENCES `order` (`code`)
    ON UPDATE CASCADE,
  CONSTRAINT `product_code`
    FOREIGN KEY (`product_code`)
    REFERENCES `product` (`code`)
    ON UPDATE CASCADE
)
```

```
CREATE TABLE `recently_viewed` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `product_code` int NOT NULL,
  `time` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id_idx` (`user_id`),
  KEY `product_code_idx` (`product_code`),
  CONSTRAINT `product_code_fk`
    FOREIGN KEY (`product_code`)
    REFERENCES `product` (`code`),
  CONSTRAINT `user_id_fk`
    FOREIGN KEY (`user_id`)
    REFERENCES `user` (`id`)
)
```

```
CREATE TABLE `sells` (
  `product_code` int NOT NULL,
  `supplier_code` int NOT NULL,
  `price` float NOT NULL,
  PRIMARY KEY (`product_code`,`supplier_code`),
  KEY `supplier_code_sells_idx` (`supplier_code`),
  CONSTRAINT `product_code_sells`
    FOREIGN KEY (`product_code`)
    REFERENCES `product` (`code`)
    ON UPDATE CASCADE,
  CONSTRAINT `supplier_code_sells`
    FOREIGN KEY (`supplier_code`)
    REFERENCES `supplier` (`code`)
    ON UPDATE CASCADE
)
```

```
CREATE TABLE `price_range` (
  `id` int NOT NULL AUTO_INCREMENT,
  `supplier_code` int NOT NULL,
  `min_articles` int NOT NULL,
  `max_articles` int NOT NULL,
  `shipping_cost` float NOT NULL,
  PRIMARY KEY (`id`),
  KEY `supplier_code_idx` (`supplier_code`),
  CONSTRAINT `supplier_code_pr`
    FOREIGN KEY (`supplier_code`)
    REFERENCES `supplier` (`code`)
    ON UPDATE CASCADE
)
```

Application requirements analysis

Dopo il **login**, l'utente accede a una pagina **HOME** che mostra (come tutte le altre pagine) **un menù** con i link **HOME**, **CARRELLO**, **ORDINI**, **un campo di ricerca** e **una lista degli ultimi cinque prodotti visualizzati dall'utente**. Se l'utente non ha visualizzato almeno cinque prodotti, la lista è completata con prodotti in offerta scelti a caso in una categoria di default. L'utente può **inserire una parola chiave di ricerca nel campo di input e premere INVIO**. A seguito dell'invio compare una pagina **RISULTATI** con prodotti che contengono la chiave di ricerca nel nome o nella descrizione. **L'elenco mostra solo il codice, il nome del prodotto e il prezzo minimo di vendita del prodotto da parte dei fornitori che lo vendono** (lo stesso prodotto può essere venduto da diversi fornitori a prezzi diversi e l'elenco mostra il minimo valore di tali prezzi). L'elenco è ordinato in modo crescente in base al prezzo minimo di vendita del prodotto da parte dei fornitori che lo offrono. L'utente **può selezionare mediante un click un elemento** dell'elenco e **visualizzare nella stessa pagina i dati completi** e l'elenco dei fornitori che lo vendono a vari prezzi (questa azione rende il prodotto "visualizzato"). Per ogni fornitore in tale elenco compaiono: nome, valutazione, prezzo unitario, fasce di spesa di spedizione, importo minimo della spedizione gratuita e il numero dei prodotti e valore totale dei prodotti di quel fornitore che l'utente ha già messo nel carrello. Accanto all'offerta di ciascun fornitore compare un **campo di input intero** (quantità) e un **bottone METTI NEL CARRELLO**. **L'inserimento nel carrello di una quantità maggiore di zero di prodotti comporta l'aggiornamento del contenuto del carrello e la visualizzazione della pagina CARRELLO**. Questa mostra **i prodotti inseriti, raggruppati per fornitore**. Per ogni fornitore nel carrello si vedono la lista dei prodotti, il prezzo totale dei prodotti e il prezzo della spedizione calcolato in base alla politica del fornitore. Per ogni fornitore compare un **bottone ORDINA**. **Premere il bottone comporta l'eliminazione dei prodotti del fornitore dal carrello e la creazione di un ordine corrispondente**. I valori degli attributi di un ordine sono memorizzati esplicitamente nella base di dati indipendentemente dai dati del carrello. In ogni momento l'utente può accedere tramite il **menu** alla pagina **HOME**, **ORDINI** e **CARRELLO**. La pagina **ORDINI** mostra **l'elenco ordinato per data decrescente degli ordini con tutti i dati associati**. L'applicazione **NON** salva il carrello nella base di dati ma solo gli ordini.

Pages (views), view components, events, actions

Pure HTML

Sommario

VISTE

- Login Page
- Pagina Home
- NavBar
- SearchResult
- Cart
- Orders

CONTROLLERS

- CheckLogin
- GoToHome
- GoToOrders
- GoToSearchResults
- GoToShoppingCart
- ProcessOrder
- UpdateCart
- Logout

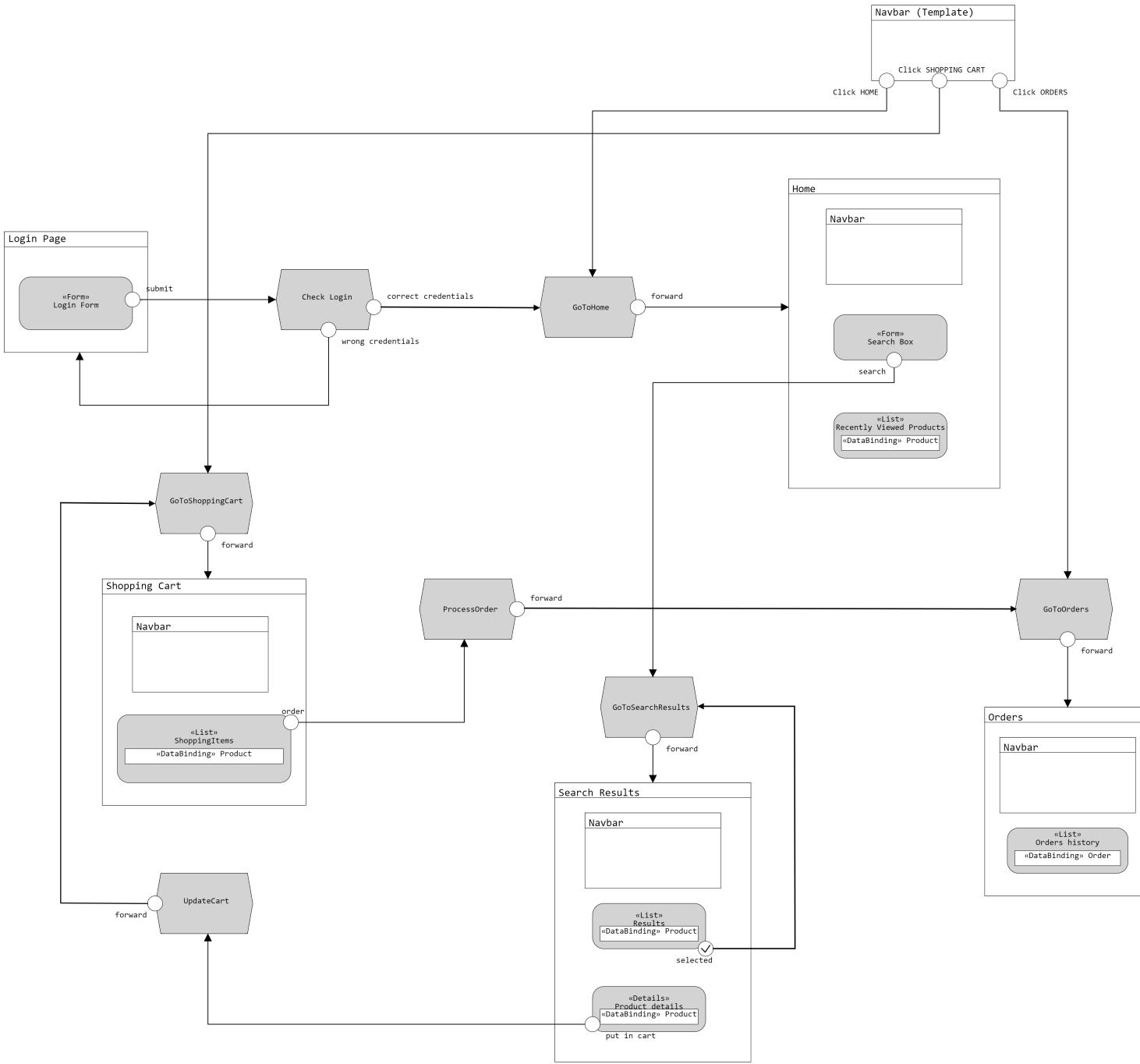
BEANS

- User
- Product
- Supplier
- Order
- PriceRange
- Cart
 - findProductQuantityFor(supplierCode)
 - findProductTotalFor(supplierCode)
 - findAllProducts()
 - findAllProductTotals()
 - getAllSupplierCodes()
 - findAllProductAndQuantitiesFor(supplierCode)
 - addProduct(supplierCode, productCode, quantity)
 - checkValidity()
 - getAllShippingCosts()

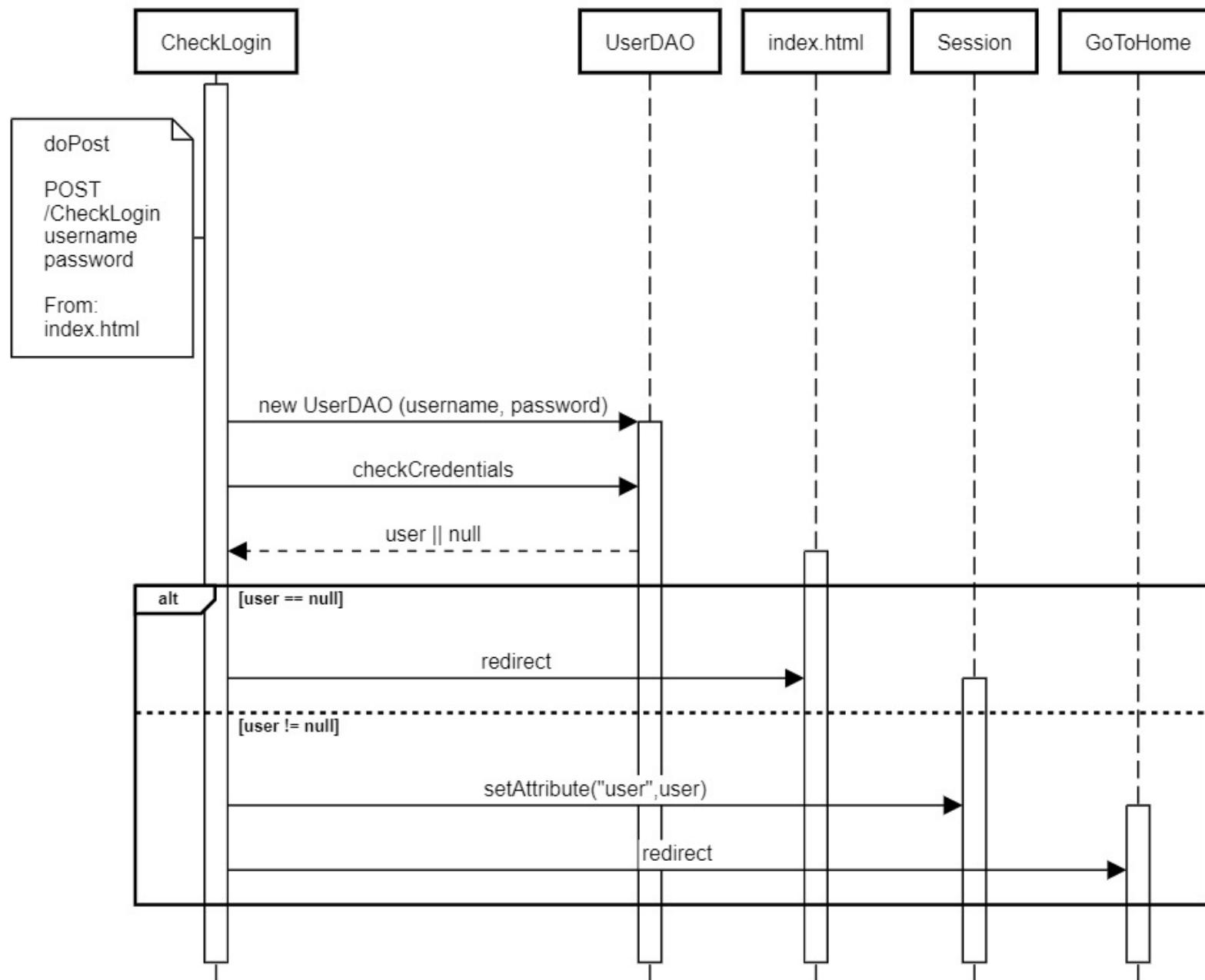
DAO

- OrderDAO
 - createOrder(totalAmount, shippingDate, shippingAddress, supplierCode, userId, productList)
 - findUserOrders(userId)
 - findAllProductsInOrder(orderCode)
- PriceRangeDAO
 - findPriceRangesForSupplier(supplierCode)
- ProductDAO
 - findProductByCode(productCode)
 - searchForProductsOrdered(searchQuery)
 - findAllProductsByCodes(productCodes)
 - isProductSoldBy(productCode, supplierCode)
 - getProductPriceFor(productCode, supplierCode)
 - findRandomProducts(quantity)
- SupplierDAO
 - findSupplierByCode(supplierCode)
 - findAllSuppliersFor(productCode)
 - findProductsTotalWithQuantities(supplierCode, productCodesAndQuantites)
- UserDAO
 - CheckCredentials(mail, password)
 - findLastFiveViewedBy(userId)
 - addViewToProductFrom(userId, productCode, timestamp)

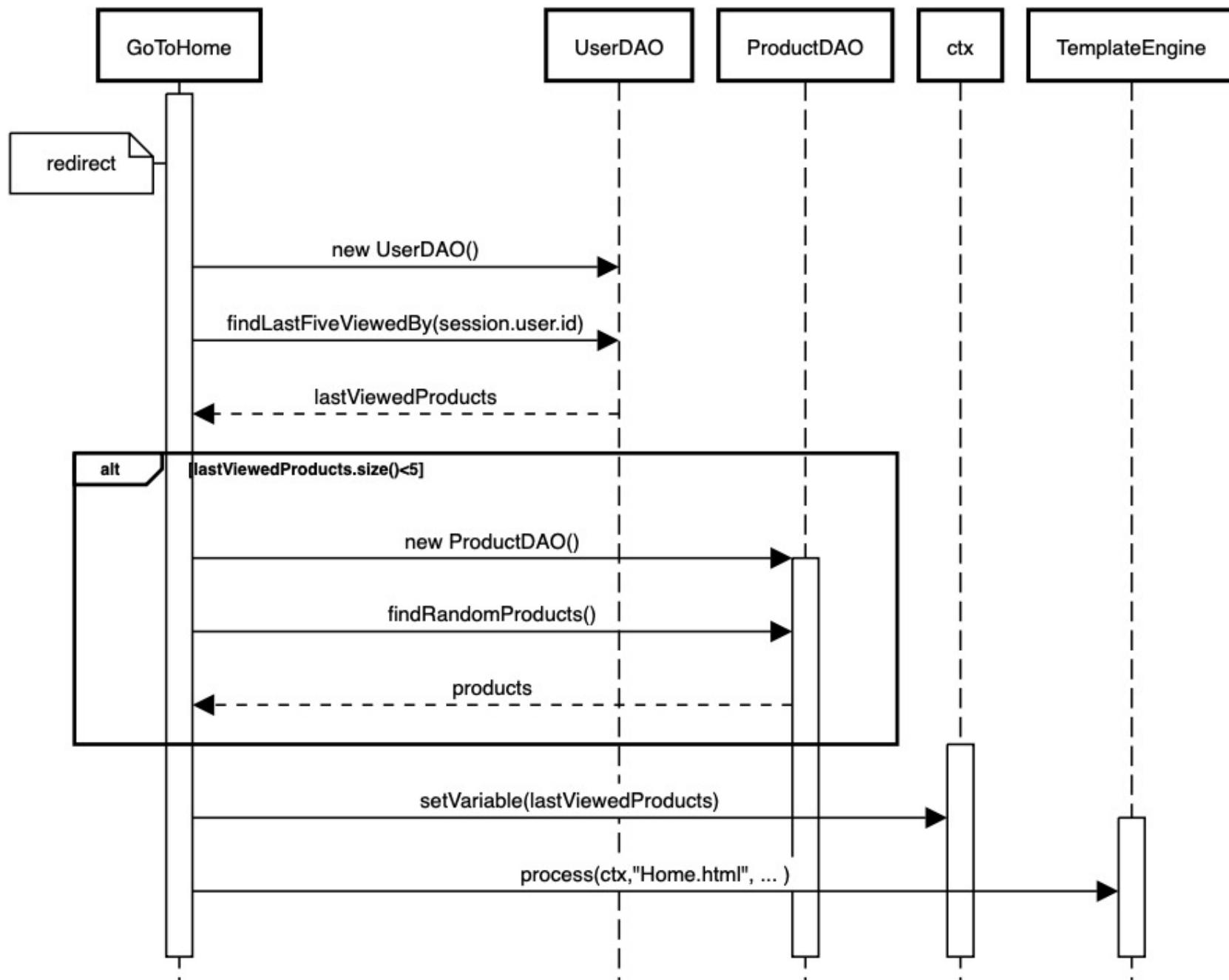
IFML



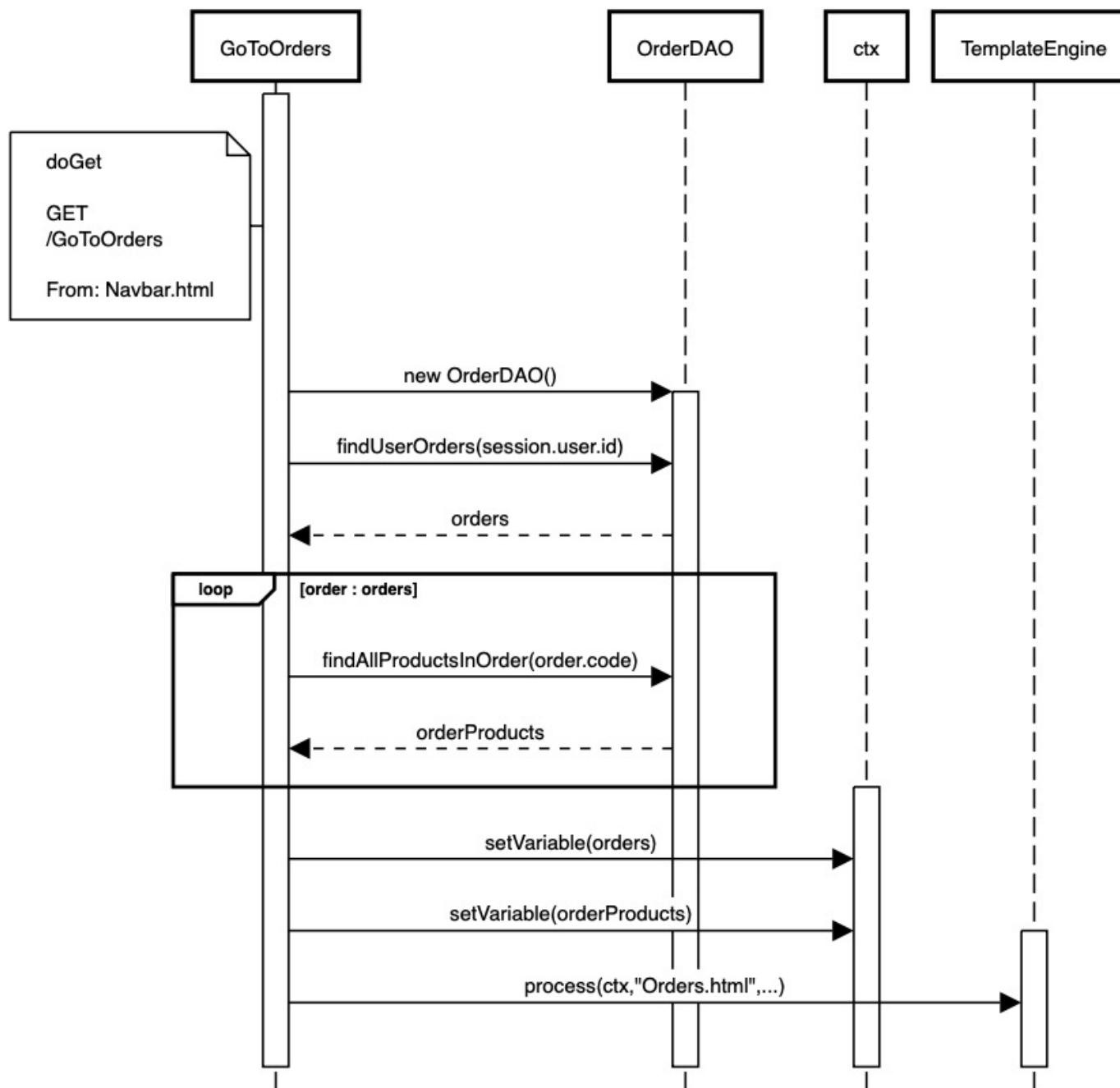
Login



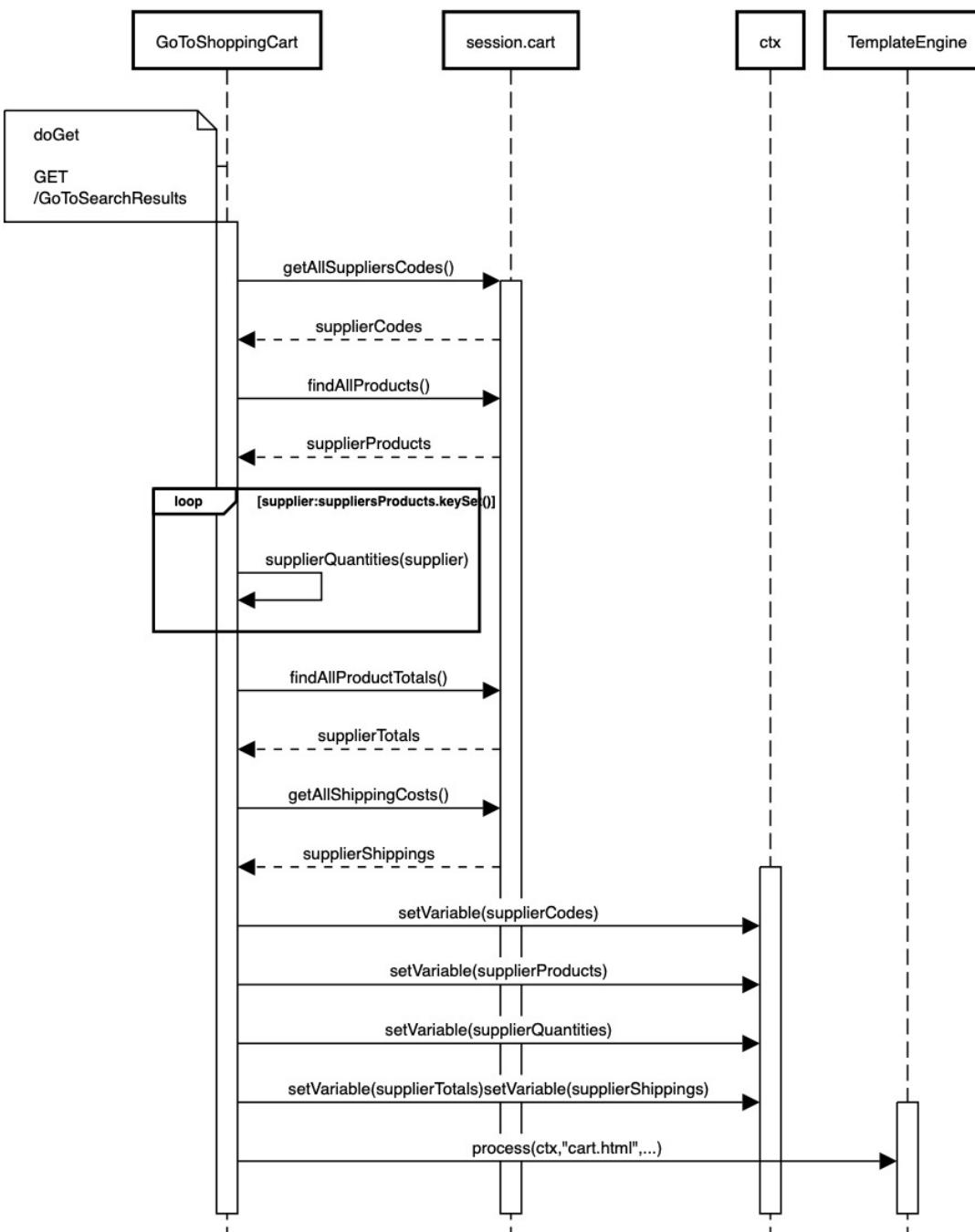
Go to Home

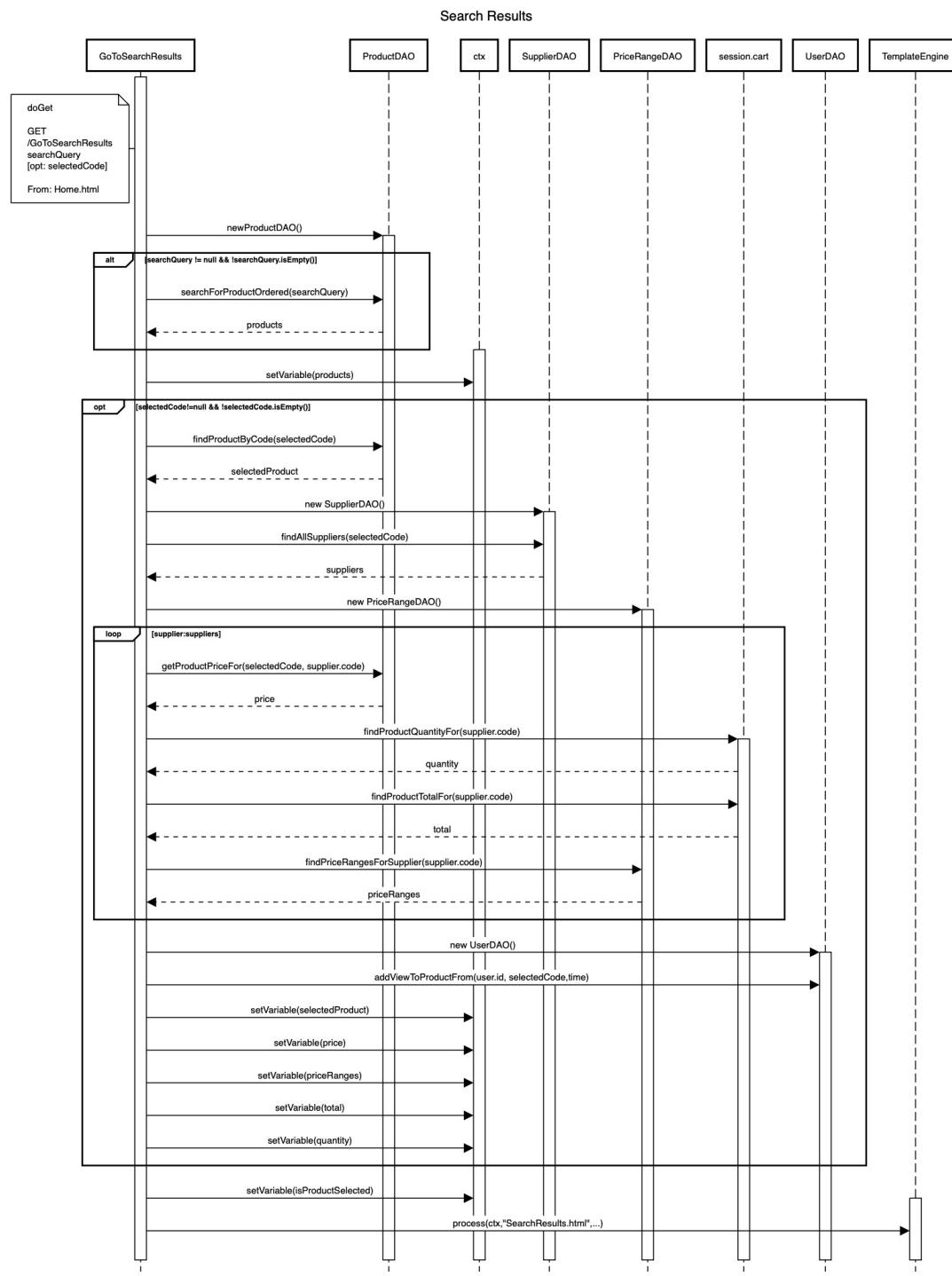


Go To Orders

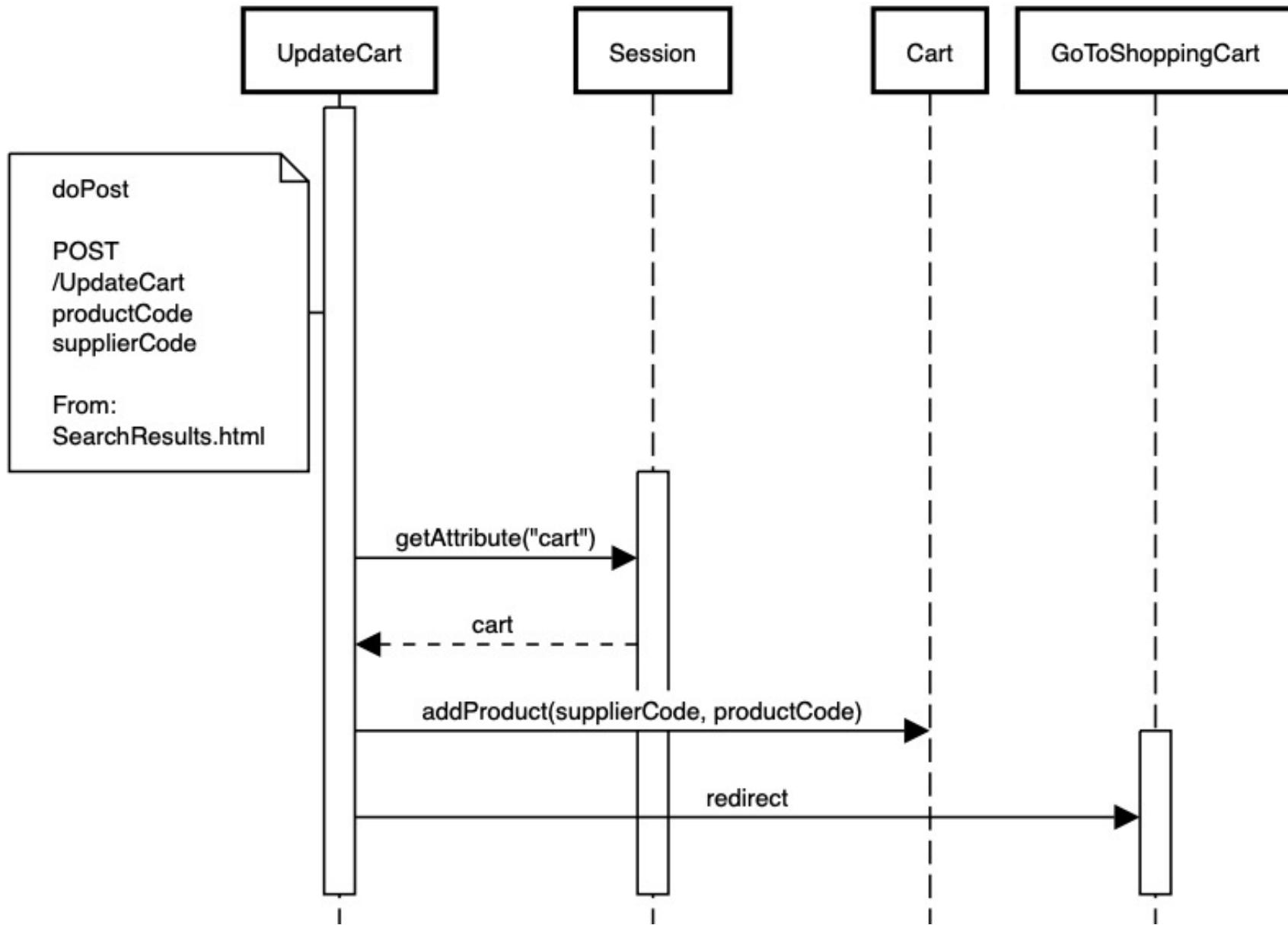


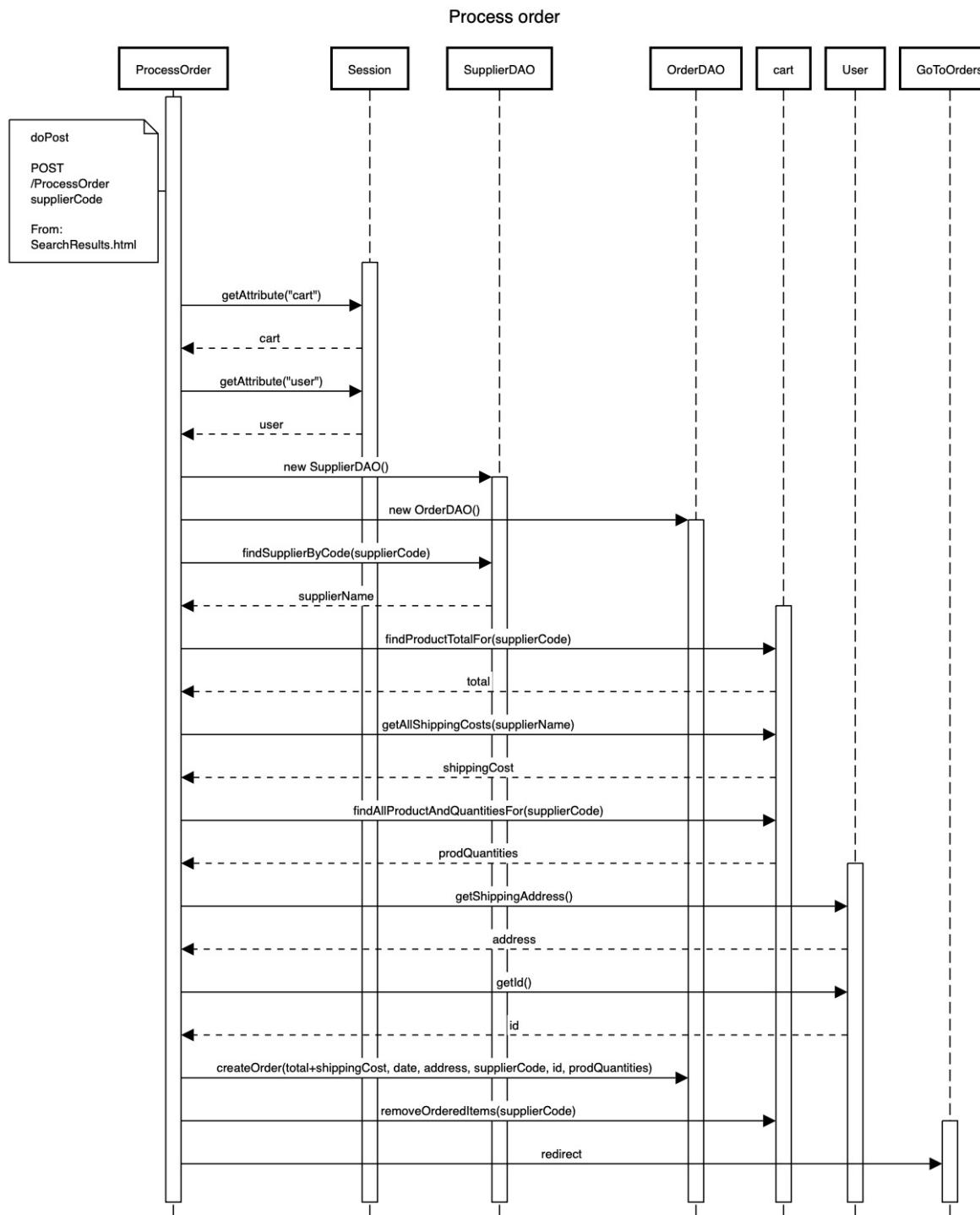
Go To Shopping Cart



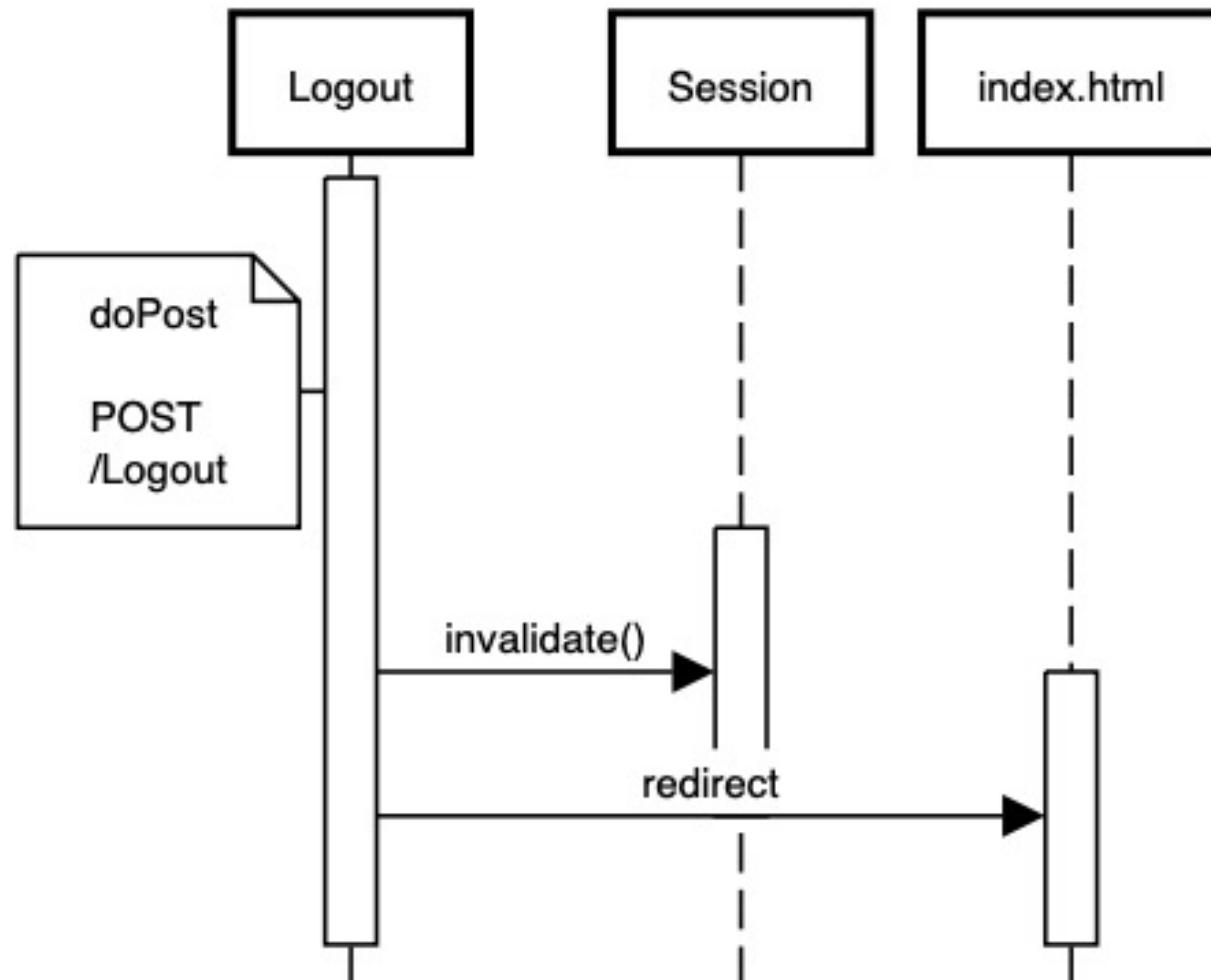


Update Cart



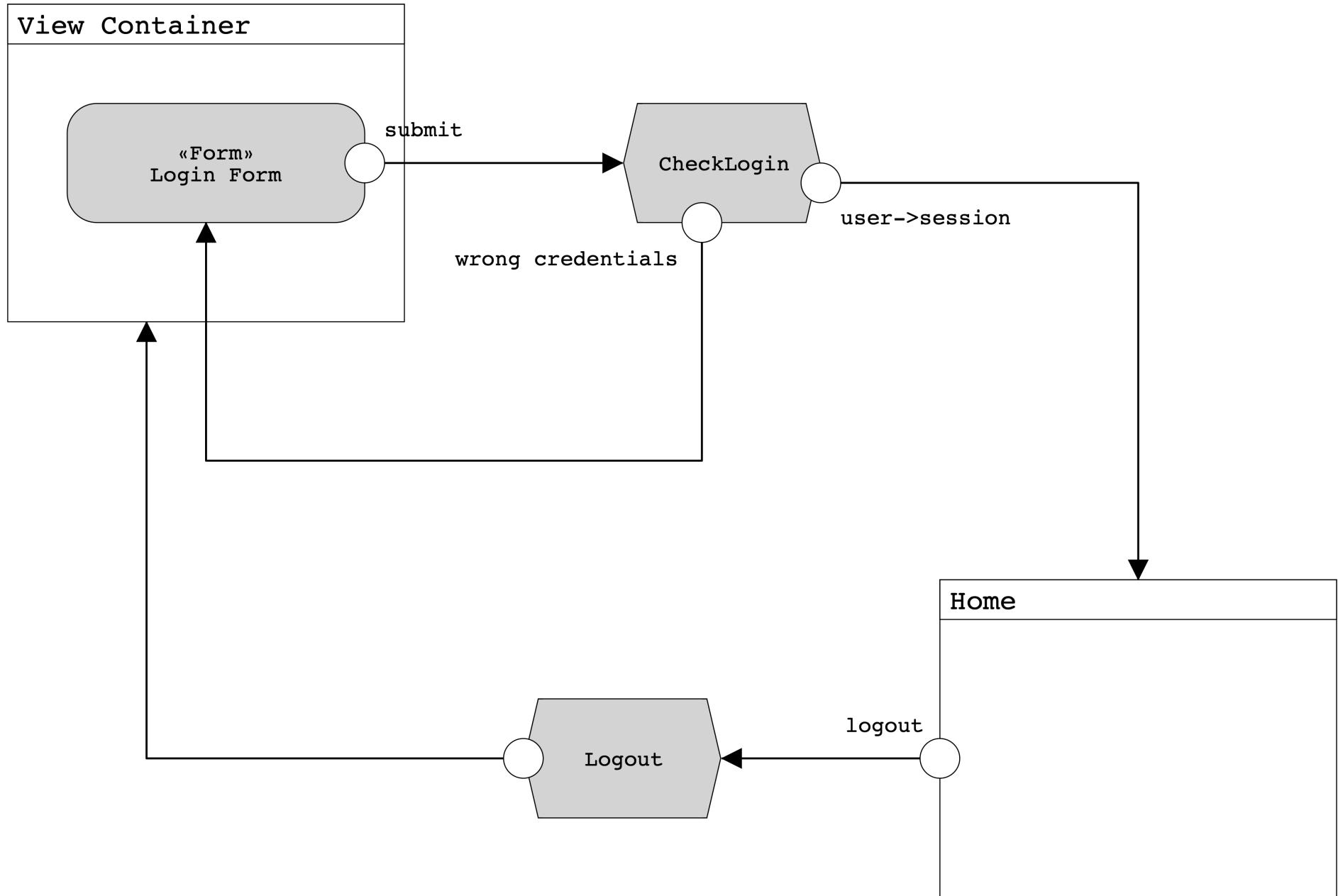


Logout

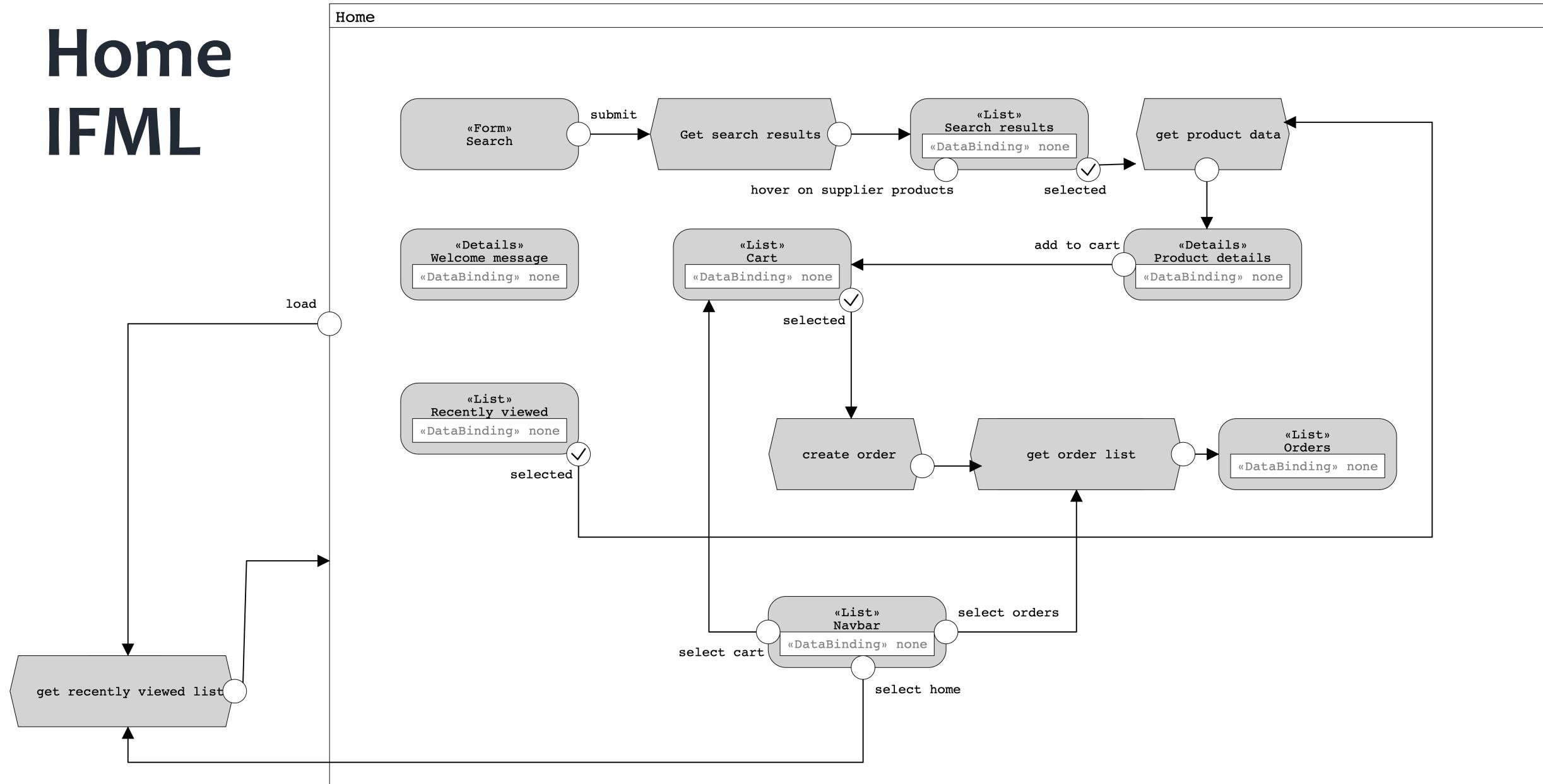


RIA

Login IFML



Home IFML



Additional requirements

- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- L'applicazione memorizza il contenuto del carrello a lato client.
- Nella pagina RISULTATI l'elenco dettagliato dei prodotti già nel carrello da parte di un fornitore compare mediante una finestra sovrapposta quando si passa con il mouse sopra il numero che indica quanti prodotti del medesimo fornitore sono già nel carrello.

Sommario

VISTE e COMPONENTI

- Pagina Home
- WelcomeMessage
- NavBar
- SearchBox
- RecentlyViewed
- SearchResult
- ProductDetails
- Cart
- OrderList

EVENTI e AZIONI

- Login
 - Verifica Credenziali
- Ricerca parola
 - Ricerca prodotti corrispondenti
- Click su prodotto visto/consigliato
 - Visualizzazione dettagli prodotto
- Click Navbar
 - Visualizzazione pagina corrispondente
- Click dettagli su ricerca
 - Visualizzazione dettagli prodotto
- Hover su numero prodotti in carrello
 - Visualizzazione carrello del fornitore
- Click inserimento nel carrello
 - Aggiunta prodotto al carrello
 - Visualizzazione carrello
- Click ordine
 - Aggiunta prodotto all'ordine
 - Rimozione prodotto dal carrello
 - Visualizzazione ordini
- Logout

Eventi & Azioni

Client side		Server side	
Evento	Azione	Evento	Azione
index -> login form -> submit	Controllo dati	POST username password	Controllo credenziali
Home page -> load	Aggiorna view con dati ultimi prodotti visualizzati	GET	Estrazione ultimi prodotti visualizzati o consigliati
Home -> elenco prodotti recenti -> seleziona prodotto	Aggiorna view con dettagli prodotto	GET product code	Estrazione dettagli prodotto
Home -> searchbox -> invio parola	Aggiorna view con prodotti corrispondenti	GET search query	Estrazione risultati
Home -> Search result -> selezione prodotto	Aggiorna view con dettagli prodotto	GET product code	Estrazione dettagli prodotto
Home-> Product details -> metti nel carrello	Aggiorna view con dati carrello		Aggiungi prodotti al carrello
Home -> Cart -> ordina	Aggiorna view con vista ordini	POST dati ordine	Crea ordine Rimuovi prodotti dal carrello

Client side		Server side	
Evento	Azione	Evento	Azione
Home -> Navbar -> seleziona link	Aggiorna view con vista corrispondente	GET ...	Estrai dati richiesti
Home -> Search results -> hover su prodotti nel carrello	Aggiorna view con finestra sovrapposta con dettagli		Estrai prodotti del fornitore dal carrello
Logout		GET	Terminazione sessione

SERVER SIDE

BEAN

- Order
- PriceRange
- Product
- Supplier
- User
- Cart
 - findProductTotalFor(supplierCode)
 - findProductQuantityFor(supplierCode)
 - findAllProducts()
 - getAllSupplierCodes()
 - findAllProductTotals()
 - checkValidity()
 - getAllShippingCosts()
 - findAllProductAndQuantitesFor(supplierCode)

CONTROLLERS

- CheckLogin
- CreateOrder
- GetCartPrices
- GetOrderList
- GetProductDetailsData
- GetRecentlyViewedList
- GetSearchResultsData
- Logout

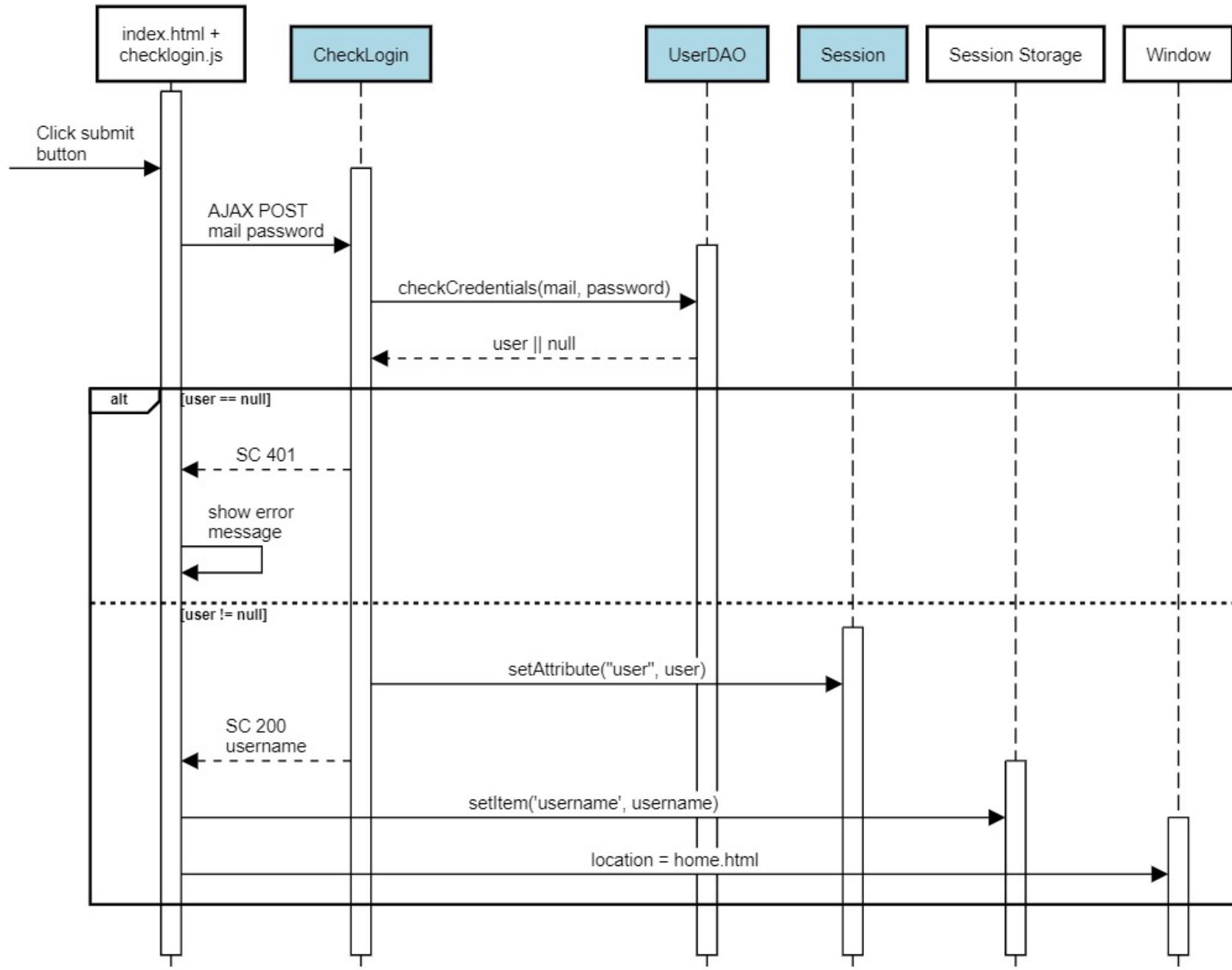
DAO

- **OrderDAO**
 - createOrder(totalAmount, shippingDate, shippingAddress, supplierCode, userId, productList)
 - findUserOrders(userId)
 - findAllProductsInOrder(orderCode)
- **PriceRangeDAO**
 - findPriceRangesForSupplier(supplierCode)
- **ProductDAO**
 - findProductByCode(productCode)
 - searchForProductsOrdered(searchQuery)
 - findAllProductsByCodes(productCodes)
 - isProductSoldBy(productCode, supplierCode)
 - getProductPriceFor(productCode, supplierCode)
 - findRandomProducts(quantity)
- **SupplierDAO**
 - findSupplierByCode(supplierCode)
 - findAllSuppliersFor(productCode)
 - findProductsTotalWithQuantities(supplierCode, productCodesAndQuantites)
- **UserDAO**
 - CheckCredentials(mail, password)
 - findLastFiveViewedBy(userId)
 - addViewToProductFrom(userId, productCode, timestamp)

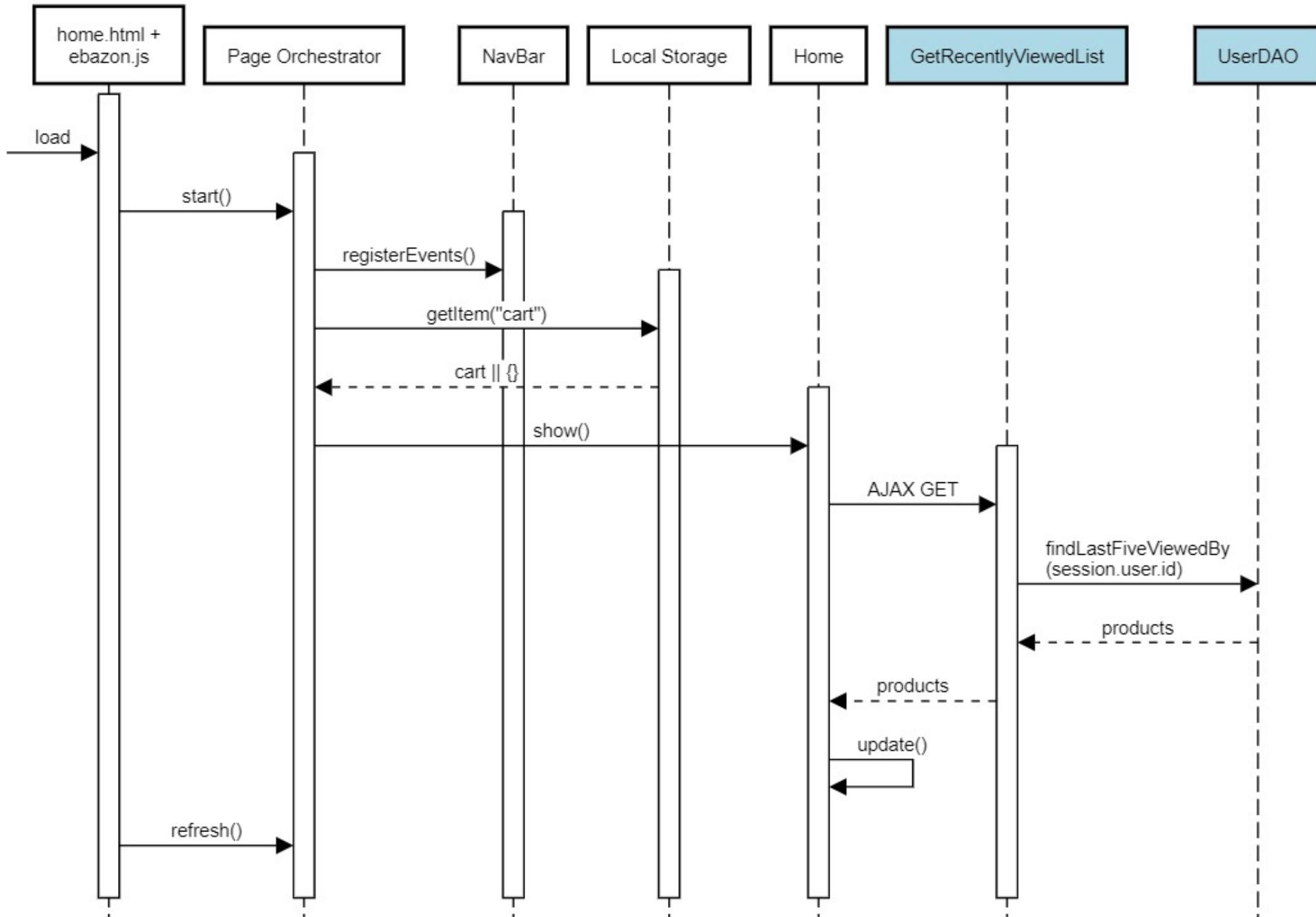
CLIENT SIDE

- **Index**
 - **LoginForm** : Gestione del submit e degli errori
- **Home**
 - **PageOrchestrator**
 - start() : inizializza i componenti di interfaccia registrando i gestori degli eventi
 - refresh() : orchestra il reperimento dei contenuti e la visualizzazione dei componenti
 - navigateTo() : visualizza i componenti della pagina richiesta, nascondendo gli altri
 - **NavBar**
 - registerEvents() : registra eventi di click per la navigazione
 - **Home**
 - show() : richiede al server i prodotti recentemente visualizzati
 - update() : riceve i dati dal server e aggiorna la lista dei prodotti
 - reset() : nasconde il contenuto della home
 - **SearchResults**
 - show() : richiede al server i prodotti corrispondenti alla ricerca e i dettagli del prodotto se richiesti
 - update() : riceve i dati dal server e aggiorna i risultati della ricerca e i dettagli del prodotto se richiesti
 - reset() : nasconde i risultati della ricerca e i dettagli del prodotto se presenti
 - createDetails() : crea il componente di visualizzazione dei dettagli del prodotto
 - **CartPage**
 - show() : richiede al server i dati relativi ai prodotti nel carrello
 - update() : riceve i dati dal server e aggiorna il carrello
 - reset() : nasconde il carrello
 - buildSupplierCart() : costruisce il componente del carrello relativo ad un supplier
 - **OrderPage**
 - show() : richiede al server gli ordini dell'utente
 - update() : riceve i dati dal server e aggiorna la lista degli ordini
 - reset() : nasconde la lista degli ordini

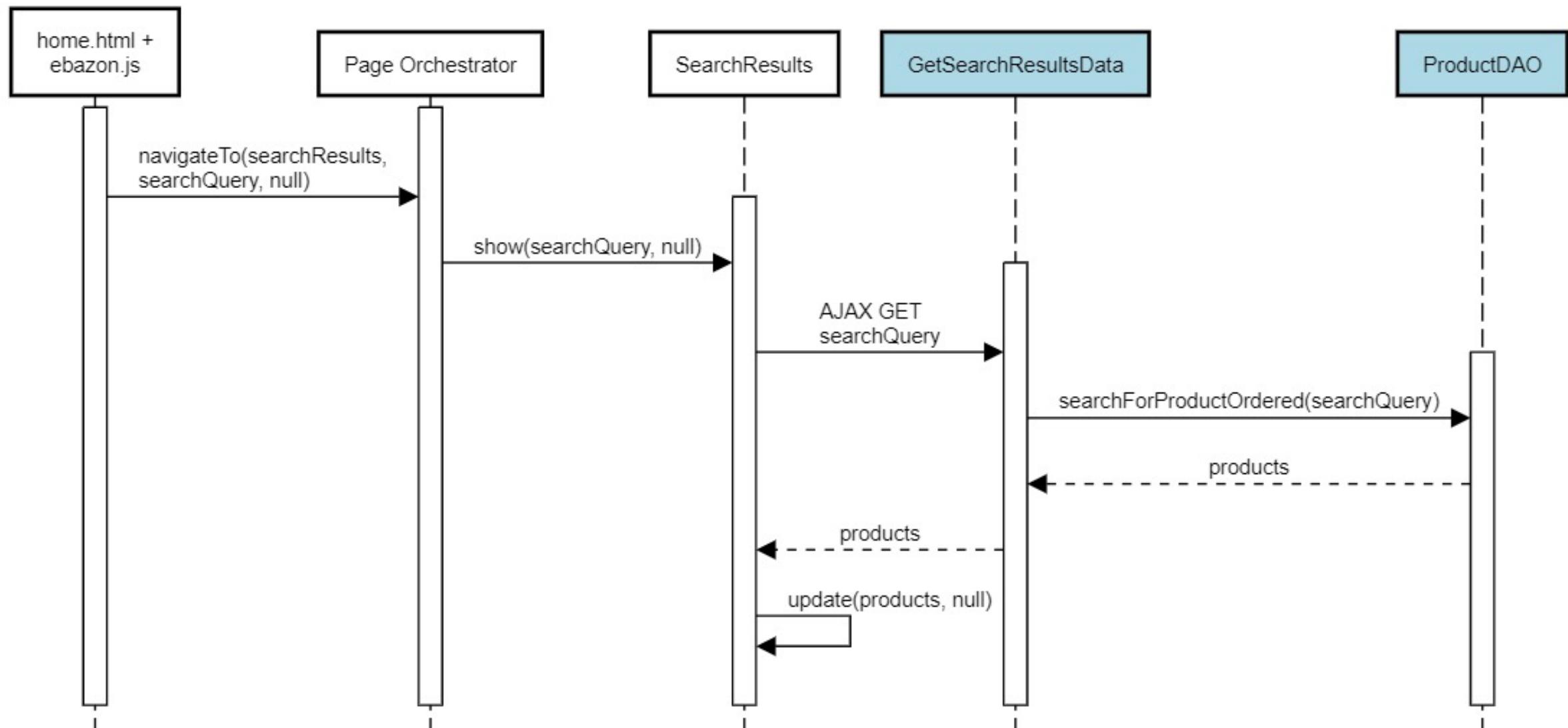
Login



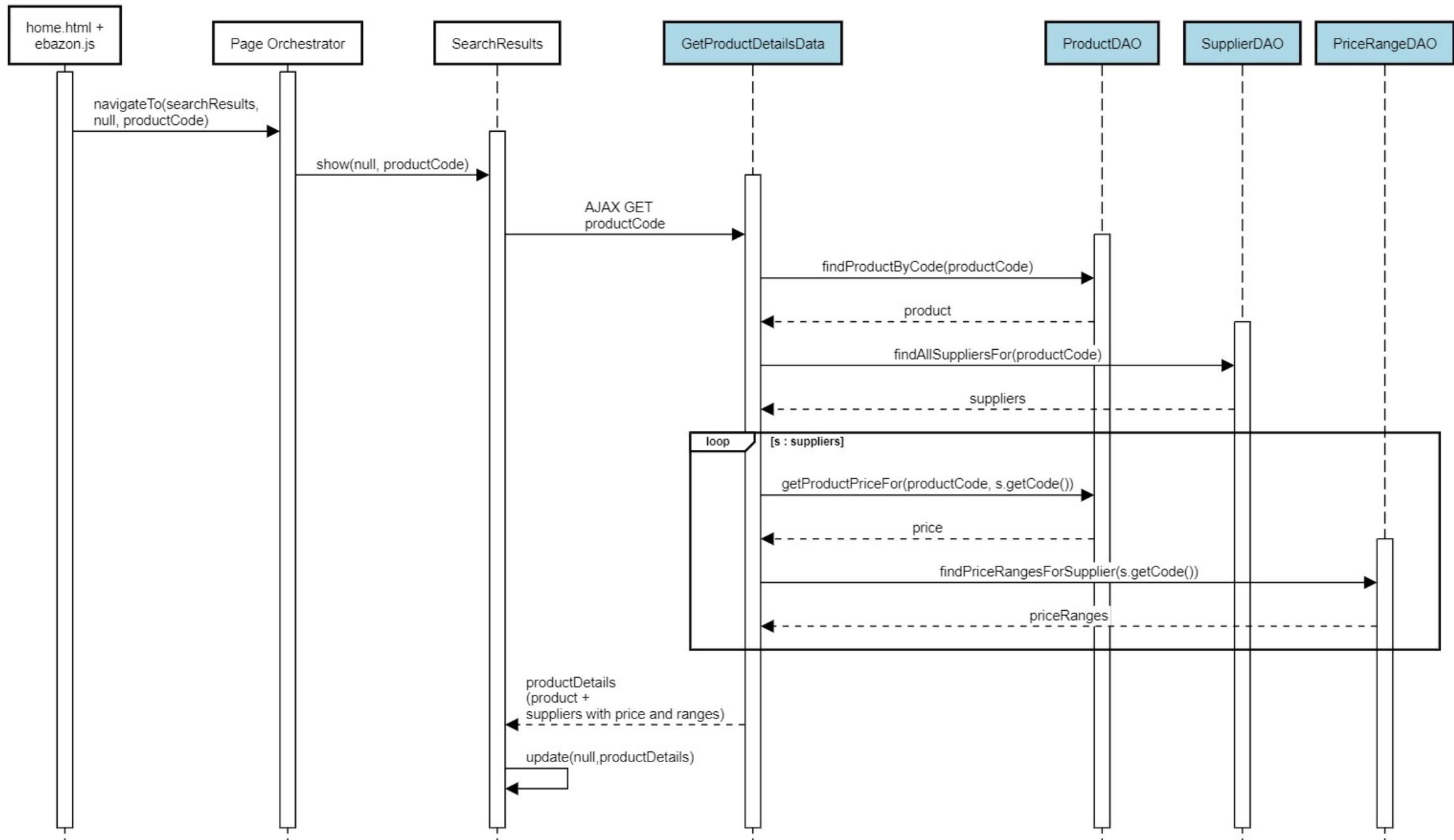
Load Home



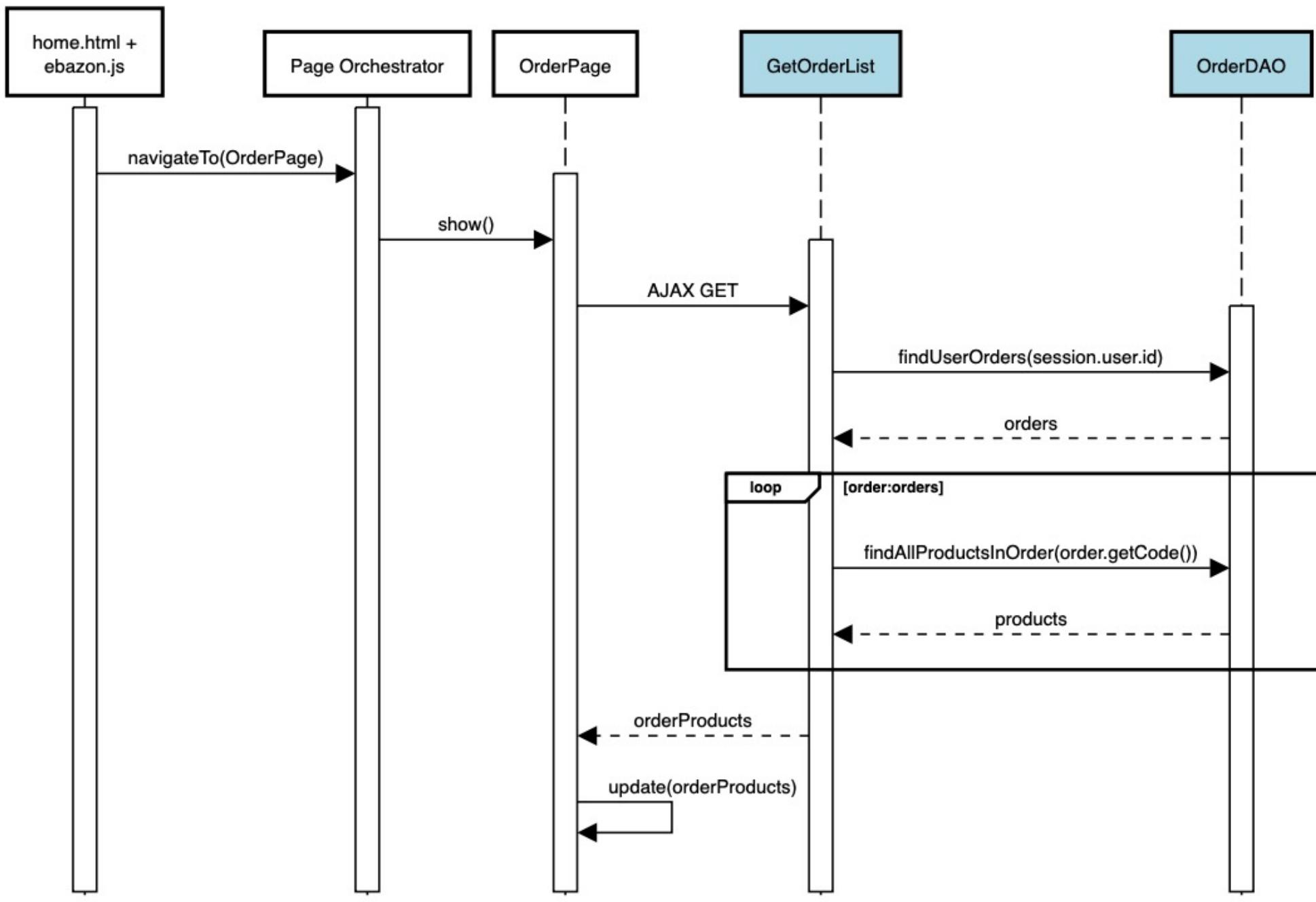
Search

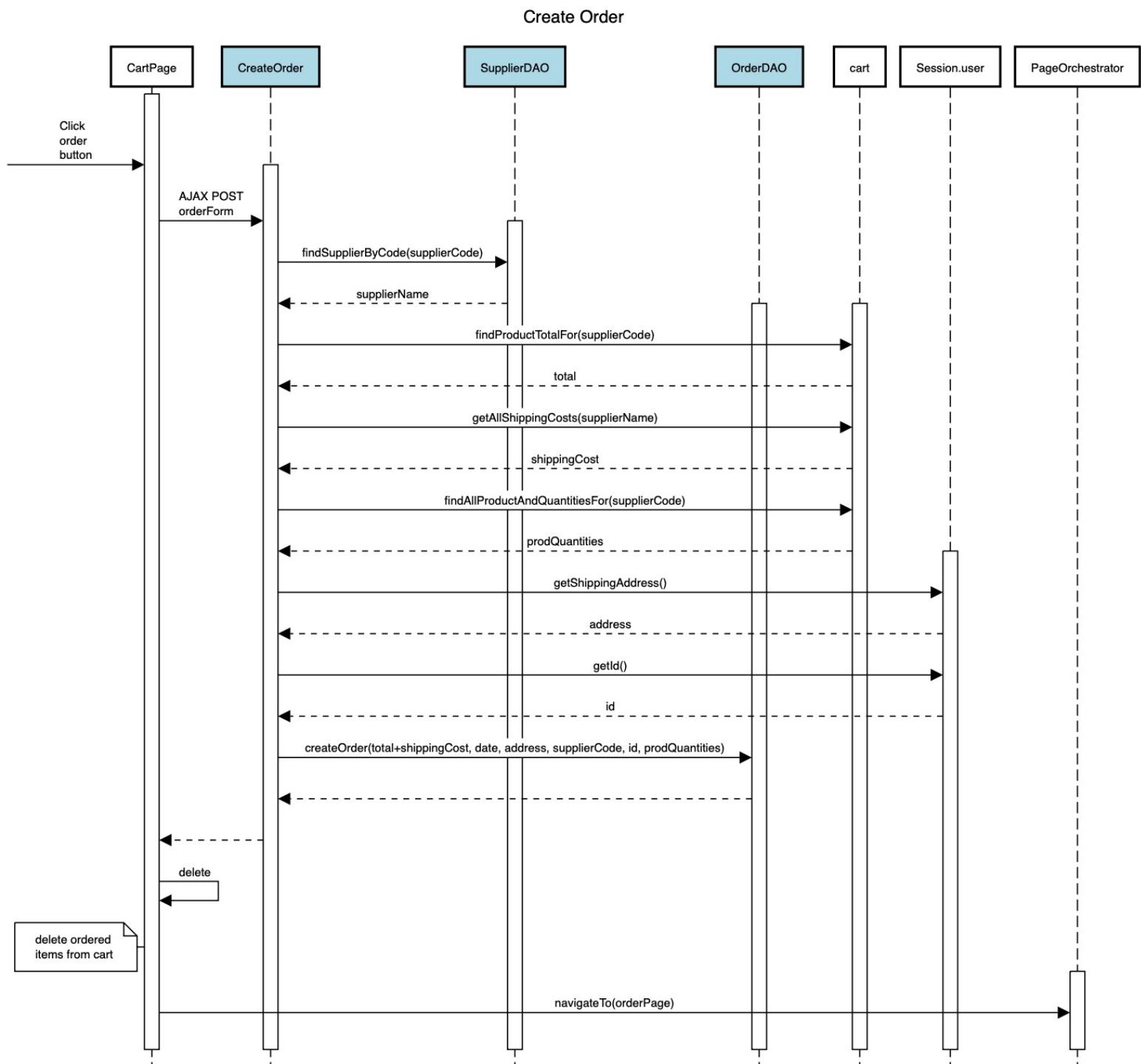


View Product Details



View order list





Logout

