Carrello con più fornitori

RIA version

Project TIW 2022/2023

Yang Hao Mao

Ken Chen

Gruppo 100

Specifications [RIA] (1/3)

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. 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).

Specifications [RIA] (2/3)

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. 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. In ogni momento l’utente può accedere tramite il menu alle pagine 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.

Specifications [RIA] (3/3)

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 prodoti del medesimo fornitore sono già nel carrello

Data Base Analysis [RIA]

Entities, attributes, relationships

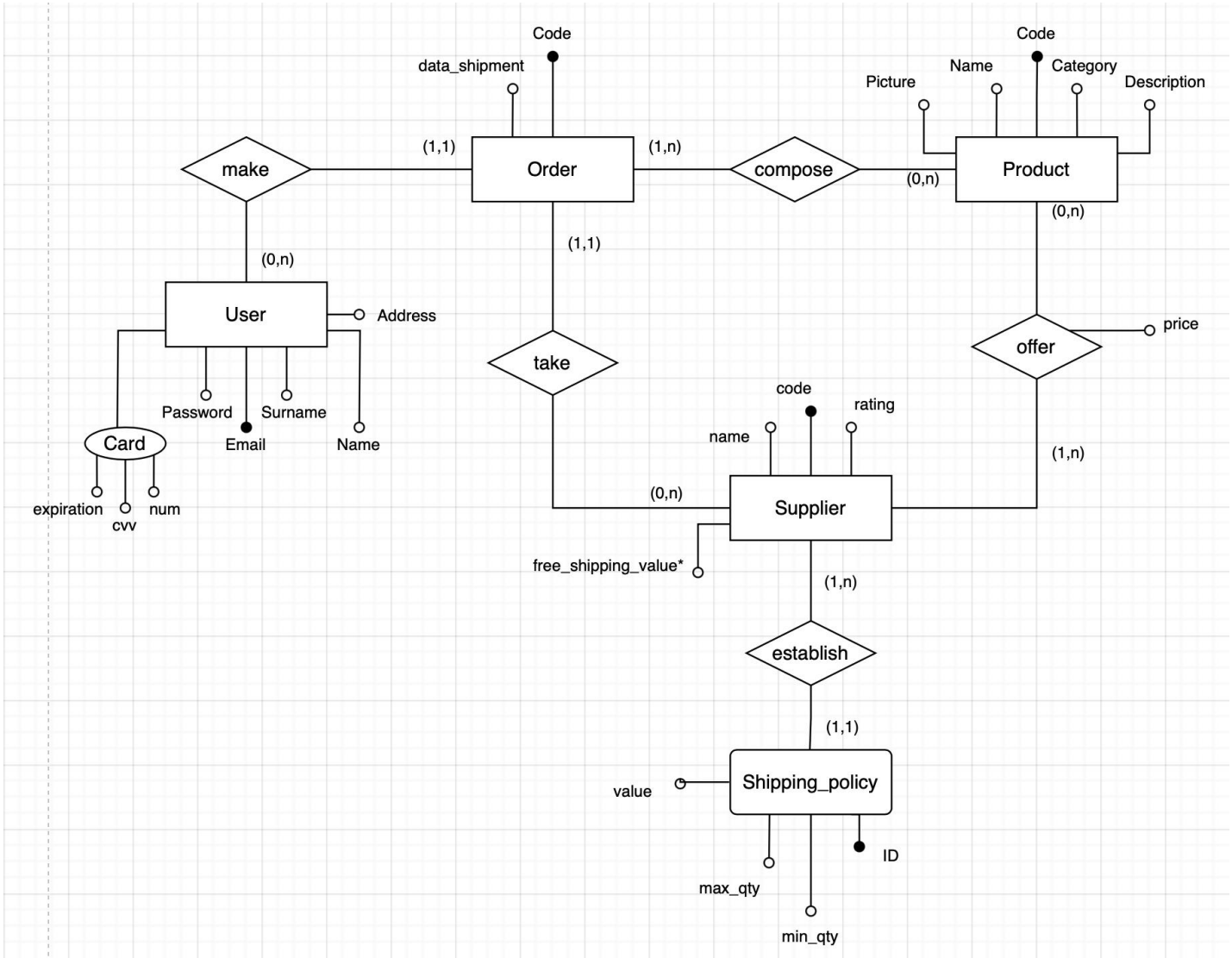
Un'applicazione di commercio elettronico consente all'**utente** (acquirente) di **visualizzare** un catalogo di **prodotti venduti** da diversi **fornitori**, inserire prodotti in un carrello della spesa e **creare** un **ordine** di acquisto a partire dal contenuto del carrello. Un prodotto ha un **codice** (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.

……..

……

Premere il bottone comporta l'eliminazione dei prodotti del fornitore dal carrello e la creazione di un ordine corrispondente. 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. In ogni momento l'utente può accedere tramite il menu alle pagine HOME, ORDINI e CARRELLO. La pagina ORDINI mostra l'elenco ordinato per data decrescente degli ordini con tutti i dati associati.

Data Base design



Data Base design

Logical scheme:

USER(**Email** , Name , Surname, Password, Address, Card_number, Card_expiration, Card_Cvv)

ORDER(**Code** , Code_Supplier , Data_Shipment)

PRODUCT(**Code** , Name , Category , Description , Photo)

compose(**Code_Order** , **Code_Product**)

SUPPLIER(**Code** , name, rating, free_shipping*)

offer(**Code_Product** , **Code_Supplier** ,price)

SHIPPING_POLICY(**ID** , **Code_Supplier** , max_qty, min_qty, value)

Foreign keys:

ORDER.Code_Supplier , offer.Code_Supplier, SHIPPING_POLICY.Code_Supplier->SUPPLIER.Code

compose.Code_Order-> Order.Code,

compose.Code_Product, offer.Code_Product-> PRODUCT.Code

Local Database schema

```
CREATE TABLE `user` (
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  `address` varchar(45) NOT NULL,
  `card_number` varchar(16) NOT NULL,
  `card_expiration` date NOT NULL,
  `card_cvv` int NOT NULL,
  PRIMARY KEY (`email`),
  UNIQUE KEY `email_UNIQUE` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `order` (
  `code` int NOT NULL AUTO_INCREMENT,
  `suplier_code` varchar(20) NOT NULL,
  `user_email` varchar(45) NOT NULL,
  `total_value` float NOT NULL,
  `date_of_shipment` date NOT NULL,
  PRIMARY KEY (`code`),
  KEY `user_email_idx` (`user_email`),
  KEY `supllier_code_idx` (`suplier_code`),
  CONSTRAINT `supllier_code` FOREIGN KEY (`suplier_code`) REFERENCES `supplier` (`code`),
  CONSTRAINT `user_email` FOREIGN KEY (`user_email`) REFERENCES `user` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=24 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Local Database schema

```
CREATE TABLE `order_composition` (
  `code_order` int NOT NULL,
  `code_product` varchar(20) NOT NULL,
  PRIMARY KEY (`code_product`,`code_order`),
  KEY `order_code_idx` (`code_order`),
  KEY `product_code_idx` (`code_product`),
  CONSTRAINT `order_code` FOREIGN KEY (`code_order`) REFERENCES `order` (`code`),
  CONSTRAINT `product_code` FOREIGN KEY (`code_product`) REFERENCES `product` (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `product` (
  `code` varchar(20) NOT NULL,
  `name` varchar(100) NOT NULL,
  `category` varchar(45) NOT NULL,
  `description` varchar(500) DEFAULT NULL,
  `photo` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`code`),
  UNIQUE KEY `code_UNIQUE` (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Local Database schema

```
CREATE TABLE `supplier` (
  `code` varchar(20) NOT NULL,
  `name` varchar(45) NOT NULL,
  `rating` int NOT NULL,
  `free_shipping` float DEFAULT NULL,
  PRIMARY KEY (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `offer` (
  `product_code` varchar(20) NOT NULL,
  `supplier_code` varchar(20) NOT NULL,
  `price` float NOT NULL,
  PRIMARY KEY (`product_code`,`supplier_code`),
  KEY `supplier_code_idx` (`supplier_code`),
  CONSTRAINT `product_id` FOREIGN KEY (`product_code`) REFERENCES `product` (`code`),
  CONSTRAINT `supplier_id` FOREIGN KEY (`supplier_code`) REFERENCES `supplier` (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `spending_range` (
  `id` varchar(20) NOT NULL,
  `supplier_code` varchar(20) NOT NULL,
  `min_qty` int NOT NULL,
  `max_qty` int NOT NULL,
  `price` float NOT NULL,
  PRIMARY KEY (`id`,`supplier_code`),
  KEY `supplier_code_idx` (`supplier_code`),
  CONSTRAINT `supplier_code` FOREIGN KEY (`supplier_code`) REFERENCES `supplier` (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Application Requirements Analysis [RIA] (1/3)

Pages (views), **view components**, **events**, **actions**

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).

Application Requirements Analysis [RIA] (2/3)

Pages (views), **view components**, **events**, **actions**

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**. 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. In ogni momento l'utente può accedere tramite il menu alle pagine 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

Application Requirements Analysis [RIA] (3/3)

Pages (views), **view components**, **events**, **actions**

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

Completamento delle specifiche

- L'applicazione gestisce anche la registrazione di un nuovo account, dove i dati vengono inseriti tramite un form e poi caricati nel data base.
- La pagina di default contiene il form di login visibile a tutti.
- Le pagine Home, Orders, Cart, Results presentano un menu, che contiene i link alle pagine Home, Orders, Cart e un pulsante per il Logout, il quale riporta l'utente alla pagina di default.
- La pagina Home contiene la lista degli ultimi 5 prodotti visualizzati dall'utente. Inoltre, la pagina contiene un form che permette la ricerca di prodotti tramite una parola chiave. La sottomissione del form avviene tramite un pulsante Search, con il quale viene aperta la pagina Results.
- La pagina Results contiene la lista dei prodotti trovati tramite la parola chiave. Per visualizzare un prodotto e, quindi, vederne i dettagli, si preme il pulsante Details. I dettagli sono visualizzabili nella stessa pagina.
- I dettagli, del prodotto visualizzato, mostrano una lista di fornitori che offrono il prodotto ad un prezzo differente, per ogni fornitore viene mostrato la lista delle fasce di spesa e un form che permette di scegliere la quantità di prodotto da inserire nel carrello. L'azione di inserimento avviene tramite un pulsante PutInCart, il quale porta alla pagina Cart. Inoltre, passando con il mouse sopra il valore della quantità di prodotti presenti nel carrello dello stesso fornitore, compare una finestra che elenca la lista dei prodotti nel carrello già presenti dello stesso fornitore
- La pagina Cart contiene la lista dei fornitori, dove per ogni fornitore è mostrato la lista di prodotti nel carrello del medesimo fornitore. Per ogni fornitore è possibile effettuare l'ordine tramite il click di un pulsante PlaceOrder. L'ordine sarà visibile nella pagina Orders cliccando il menu di navigazione.
- La pagina Orders contiene la lista degli ordini effettuati dall'utente.

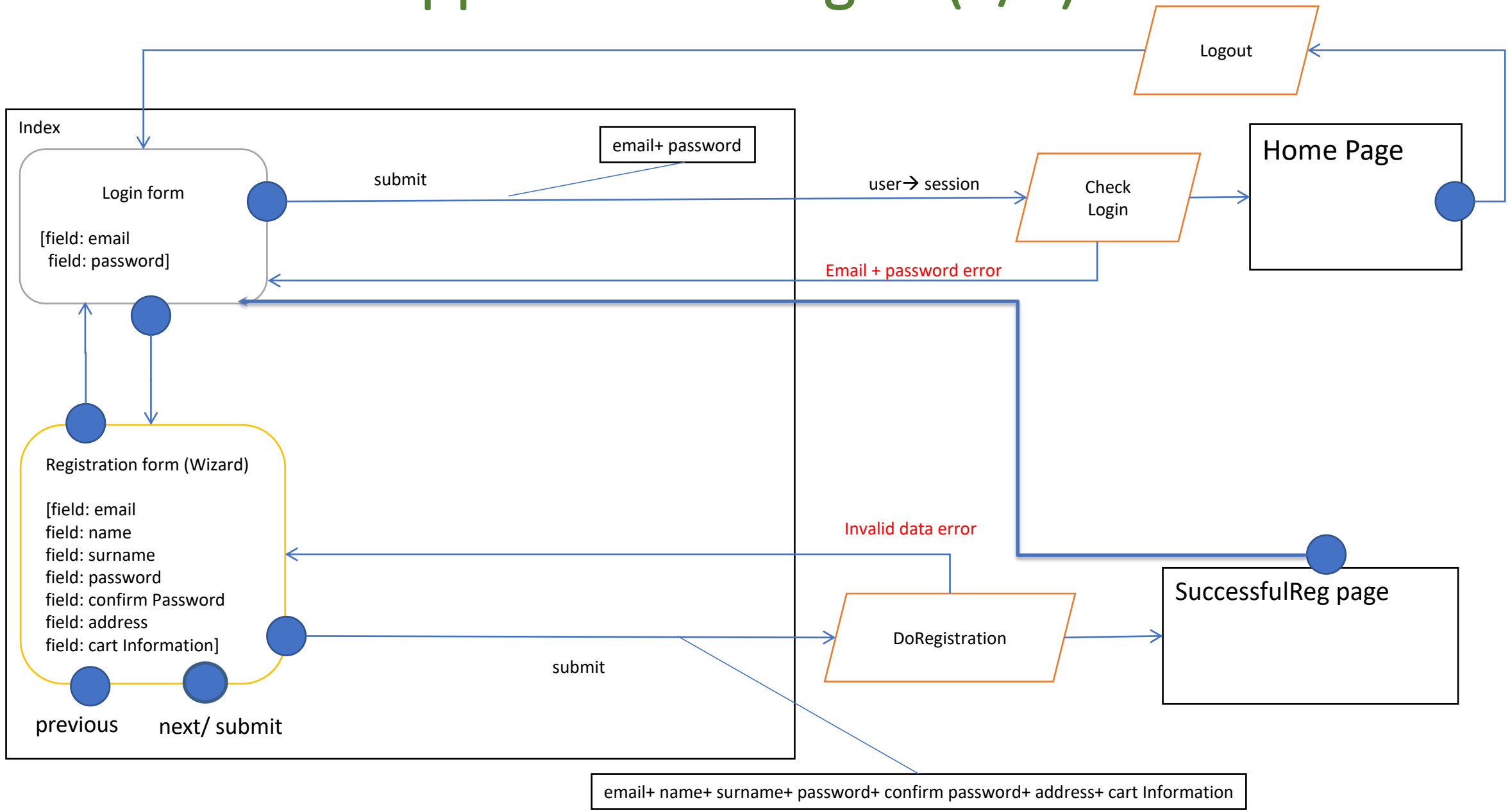
Server – Sides Components

- **Model Objects (Beans)**
  - User
  - Supplier
  - SupplierOffer
  - Offer
  - SpendingRange
  - Product
  - PreOrder
  - Order
  - ChronologyManager
- **Data Access Objects (Classes)**
  - **UserDAO**
    - existsAccount(email)
    - checkCredentials(email , password)
    - regiterAccount(email , surname, name, password…)
  - **SupplierDAO**
    - getProductSupplierList(product_code)
    - getSupplier(supplier_code)
  - **ShopDAO**
    - searchProduct(search)
    - getProduct(product_code)
    - getProductWithPrice(product_code)
    - getCategoryProducts(default_category)
    - getRandomCategory()
  - **OrderDAO**
    - uploadOrder(order)
    - getOrder(user)
    - getLastOrders(user)
  - **RangeDAO**
    - getRanges(supplier_code)
- **Controllers**
  - CheckLogin
  - ClickProduct
  - DoRegistration
  - GetProducts
  - GetLastSeen
  - PlaceOrder
  - ViewOrder
  - Logout
- **Utils (Classes)**
  - Chronology
  - PriceQuantity
  - UserCart
  - ConnectionManager
  - OrderPack

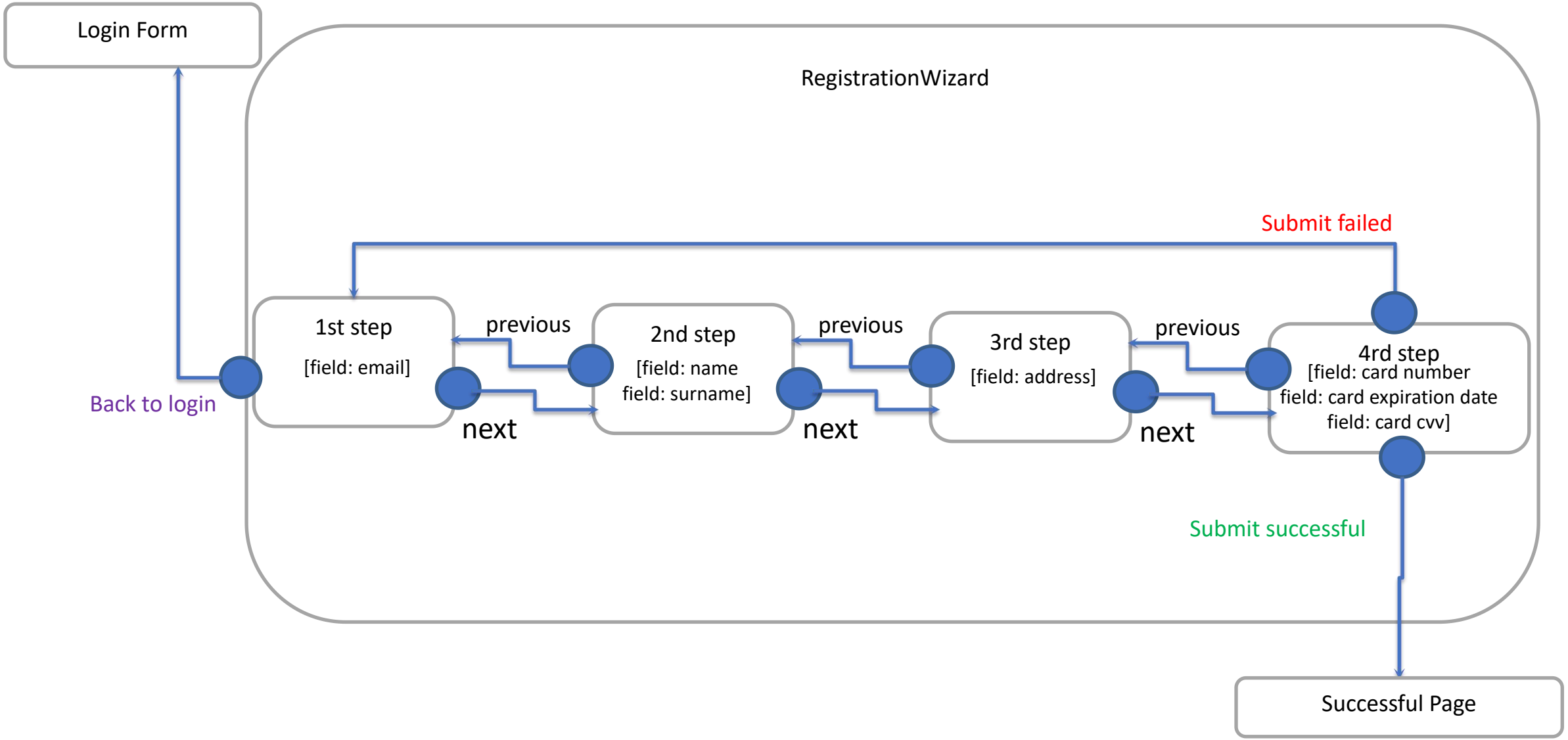
Client – Sides Components

- **Index (Login & Registration)**
  - Login form: manager submit and errors
  - Registration form: manager submit and errors
  - PageOrchestrator:
    - Start(): itilializes page components
    - Refresh(): refresh wizard page
  - RegisterWizard(id, errMsg):
    - registerEvents(): associates functions to the component to manage its events
    - Reset(): set initial conditions of visibility of modules
    - changeStep(): change the displayed form
- **Home**
  - PageOrchestrator:
    - Start(): initializes page components
    - Refresh(page): refresh all this.page to hidden, and show page
  - CurrenUser:
    - Show(): shows custom personal message according to user's email and name
  - Search:
    - Show(): go update()
    - Update(): set visible search bar
    - registerEvents(): associates functions to manage search events
  - MenuList:
    - show(): shows list of pages
    - update(): set different conditions of visibility of pages
  - HomePage:
    - Show(): shows the lastSeen list getting it from the server via an AJAX GET
    - Update(lastSeenList): builds the updated lastSeen Image and name section
    - Reset(): hides the Home page
  - ResultPage:
    - Show(result, resultToReport): go update()
    - Update(productList, keyword): builds the updated productList table, add EventListener to button **show**, so go SupplierFragment show(Object)
    - reset(): hides the Result page
  - SupplierFragment:
    - Show(Object[product, supplier etc..]): go update()
    - Update(Object[product, supplier etc..]): builds the updated supplierList table, so you can do following action:
      - Put In Cart: insert number of product for your interest supplier, so your will update the cart
      - Hand over the mouse on the quantity area section: you can see the details of cart for this supplier
      - Close button: hidden the supplier list of this product
    - Reset(): hides the Supplier Frament section
  - CartPage:
    - Show(): get all carts from cartManager, so go update()
    - Update(allCarts): builds the updated cartList Section, add EvenListener to button **placerOrder**, now will deleted cart from local saved
    - Reset(): hides the Cart page
  - OrderPage:
    - Show(orderList): shows the orderList list getting it from the server via an AJAX GET
    - Update(orderList): builds the updated orderList Image and name section
    - Reset(): hides the Order page

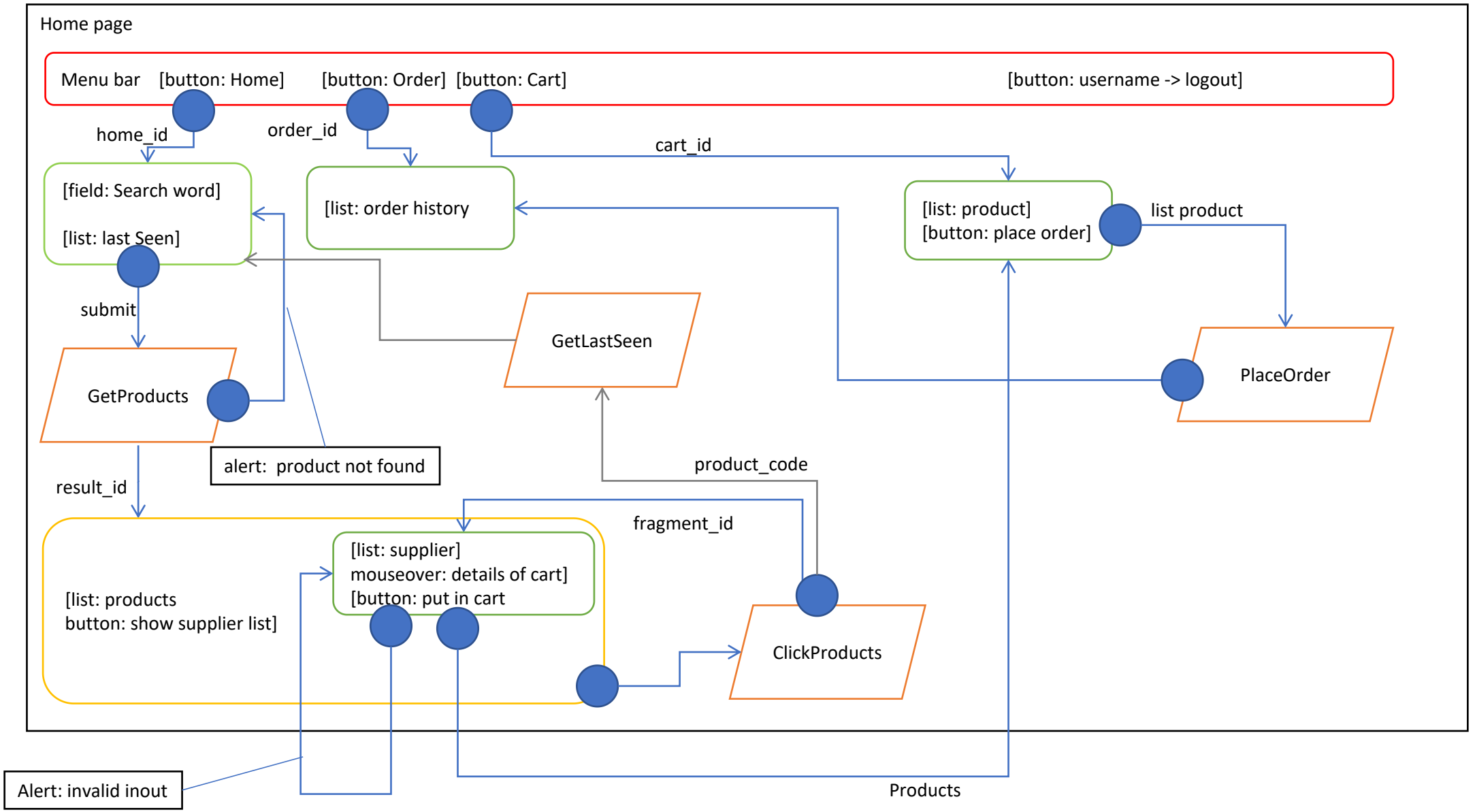
Application Design - (1/2)



Registration Wizard



Application Design - (2/2)



Eventi & azioni

NB: i controlli di validità dei dati (client e server side) e di autorizzazione (server side) all'accesso sono previsti per tutti gli eventi che li richiedono e non sono riportati nella tabella per brevità

Client side		Server side	
Evento	**Azione**	**Evento**	**Azione**
index → login form → submit	Controllo dati	POST (email password)	Controllo credenziali
index → registration form→registrationWizard	Cambio form	-	-
RegistrationWizard → next.previous	Controllo dati, cambio modulo del wizard, svuotamento form	-	-
RegistrationWizard → submit	Controllo dati	POST (dati user)	Inserimento user Cambio stato
RegistrationWizard → backToLogin	Cambio form	-	-
SuccesfulReg→backToLogin	Cambio form	-	-

Eventi & azioni

NB: i controlli di validità dei dati (client e server side) e di autorizzazione (server side) all'accesso sono previsti per tutti gli eventi che li richiedono e non sono riportati nella tabella per brevità

Client side		Server side	
Evento	**Azione**	**Evento**	**Azione**
Home page→ load	Aggiorna view con lista degli ultimi visualizzati	GET (lastSeen List)	Estrazione prodotti visualizzati (o casuali)
Home page → search→ click	Ajax post con oggetto JSON contenente lista dei prodotti	POST (parola chiave)	Estrazione lista dei prodotti che contono la parola chiave
Home/Order/Result/Cart page→ Order page→ click intestazione	Aggiornamento view con lista degli ordini	GET (order List)	Estrazione ordini effettuati precedentemente
Home/Order/Result/Cart page→ Cart page→ click intestazione	Aggiornamento view con lista dei prodotti suddivisi per fornitori	-	-
Home/Order/Result/Cart page→ Home page→ click intestazione	Aggiorna view con lista degli ultimi visualizzati	GET (lastSeen List)	Estrazione prodotti visualizzati
Logout		GET	Terminazione della sessione

Eventi & azioni

NB: i controlli di validità dei dati (client e server side) e di autorizzazione (server side) all'accesso sono previsti per tutti gli eventi che li richiedono e non sono riportati nella tabella per brevità

Client side		Server side	
Evento	**Azione**	**Evento**	**Azione**
Result page → elenco prodotti→ click bottone show	Aggiorna view con lista dei fornitori del prodotto cliccato	-	-
Result page → elenco prodotti→ click bottone close	Rimozione view della lista dei fornitori del prodotto cliccato	-	-
… → Fragmente fornitore → casella della quantità da inserire → keyboard numero→ click bottone Put Cart	Aggiornamento view con lista dei prodotti suddivisi per fornitori	-	-
… → Fragmente fornitore → area che indica il numero quantità del prodotto nel cart di tale fornitore → mouse over	Aggiornamento view con lista dettagliata del prodotto di tale fornitore, e il dettaglio di tale carello	-	-
CartPage → sezione del fornitore → click bottone Place Order	Rimozione sezione del carello di questo fornitore Ajax post	POST (cart)	Completamento dell'acquisto Upload sul data base

Controller / event handler

makeCall indica una funzione che fa una chiamata asincrona al server

Client side		Server side	
Evento	**Controllore**	**Evento**	**Controllore**
index → login form → submit	Function makeCall	POST email password	CheckLogin (servlet)
index → registration form→click intestazione	Function RegistrationForm.show	-	-
-	-	-	-
RegistationWizard → load	Function PageOrchestrator...	-	-
RegistrationWizard → next.previous	Function changeStep	-	-
RegistrationWizard → submit	Function makeCall	POST (dati user)	DoRegistration (servlet)

Controller / event handler

makeCall indica una funzione che fa una chiamata asincrona al server

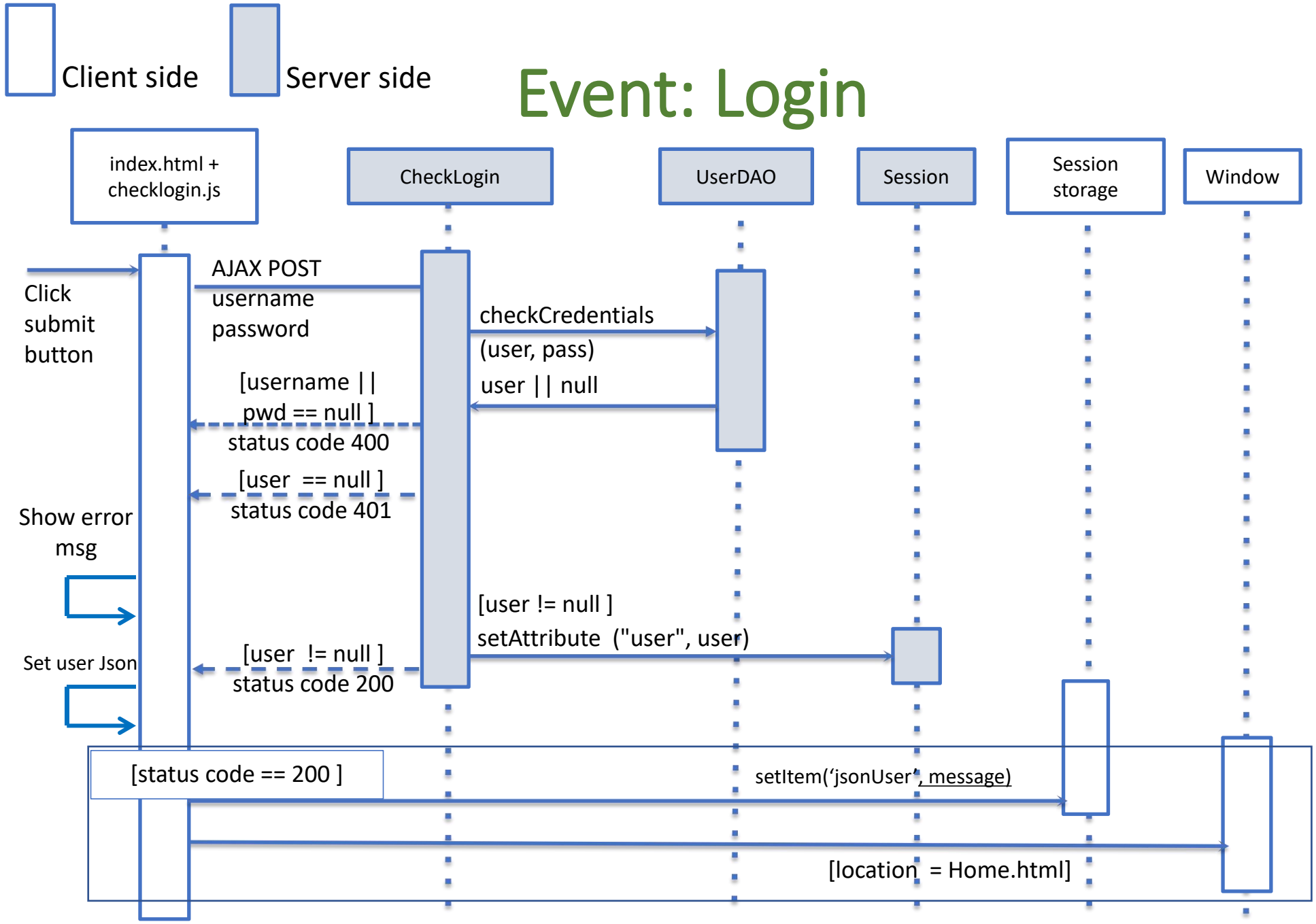
Client side		Server side	
Evento	**Controllore**	**Evento**	**Controllore**
Home page→ load	Funzione PageOrchestrator... Funzione homePage.show() Funzione menuList.show() Funzione search.show()	GET (lastSeen List)	GetLastSeen (servlet)
Home page → search→ click	Funzione resultPage.show() HTML script (printDiv)	POST (parola chiave)	GetProduct (servlet)
Home/Order/Result/Cart page→ Order page→ click intestazione	Funzione ordersPage.show() HTML script (printDiv)	GET (order List)	GetProduct (servlet)
Home/Order/Result/Cart page→ Cart page→ click intestazione	Funzione cartPage.show() HTML script (printDiv)	-	-
Home/Order/Result/Cart page→ Home page→ click intestazione	Funzione homePage.show() Funzione search.show()	GET (lastSeen List)	GetLastSeen (servlet)
Logout		GET	Logout (servlet)

Controller / event handler

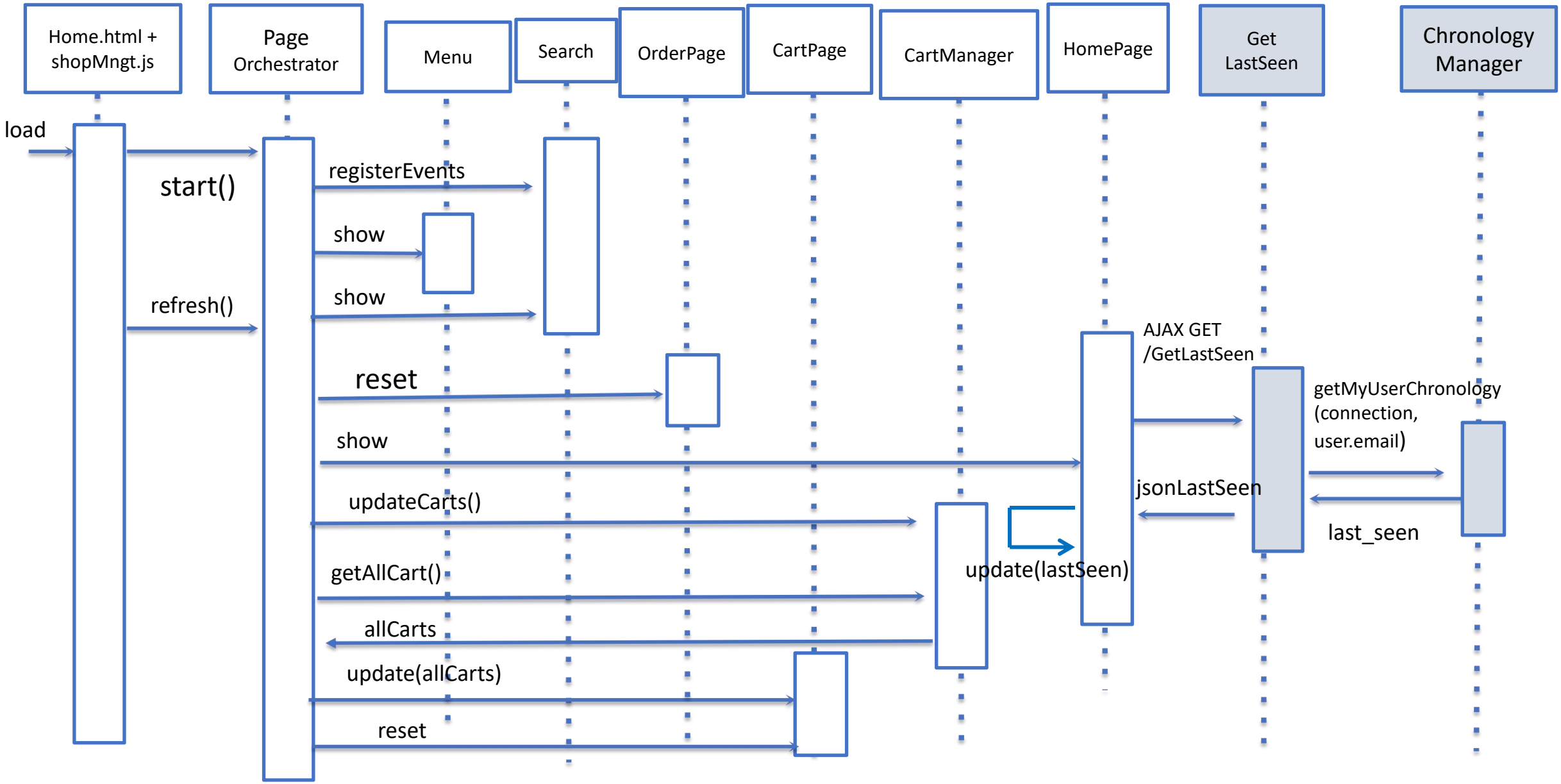
makeCall indica una funzione che fa una chiamata asincrona al server

Client side		Server side	
Evento	**Controllore**	**Evento**	**Controllore**
Result page → elenco prodotti→ click bottone show	Event handler bottone mostra lista del fornitore Funzione Fragment.show() HTML script (printDiv)	-	-
Result page → elenco prodotti→ click bottone close	Event handler bottone nasconde	-	-
… → Fragmente fornitore → casella della quantità da inserire → keyboard numero→ click bottone Put Cart	Funzione CartPage.show() HTML script (printDiv)	-	-
… → Fragmente fornitore → area che indica il numero quantità del prodotto nel cart di tale fornitore → mouse over	Event handler mouse del dettaglio del carello del fornitore	-	-
CartPage → sezione del fornitore → click bottone Place Order	Funzione OrderPage.show() HTML script (printDiv)	POST (cart)	PlaceOrder (servlet)

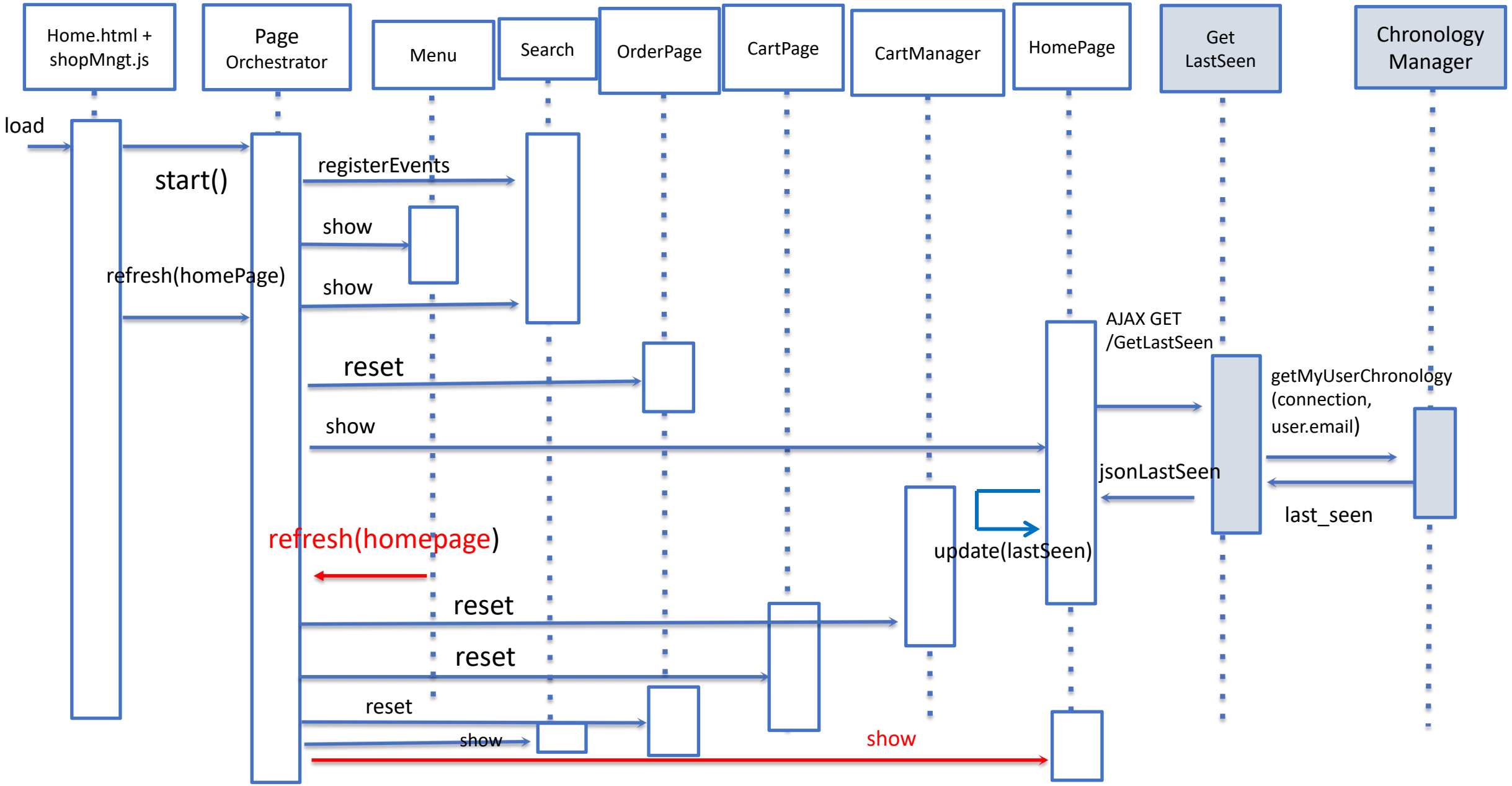
Sequence diagrams


Client side
Server side
Event: Login
index.html + checklogin.js
CheckLogin
UserDAO
Session
Session storage
Window
Click submit button
AJAX POST
username
password
checkCredentials
(user, pass)
user || null
[username || pwd == null]
status code 400
[user == null]
status code 401
Show error msg
[user != null]
setAttribute ("user", user)
Set user Json
[user != null]
status code 200
[status code == 200]
setItem('jsonUser', message)
[location = Home.html]

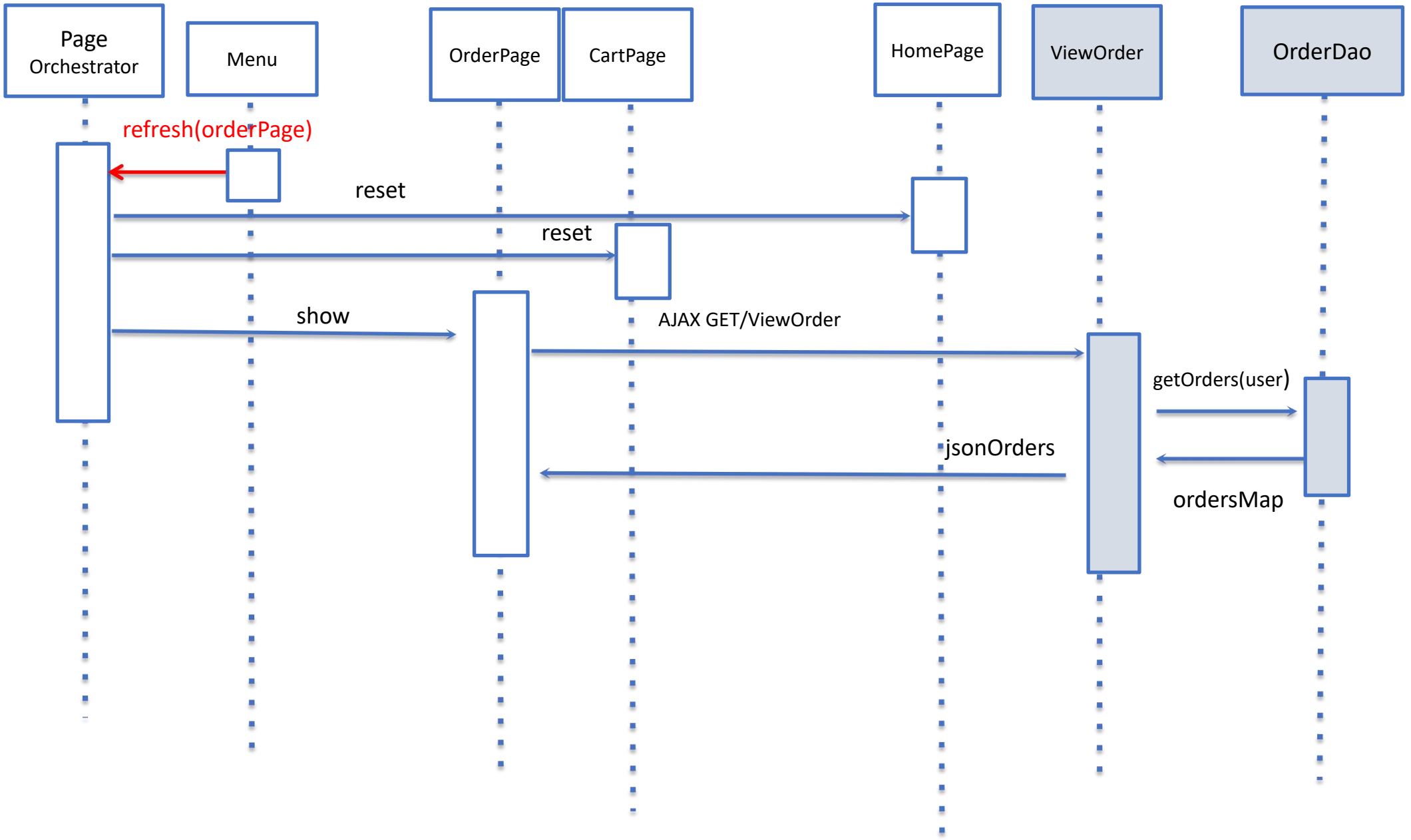
Event: Loading Home Page



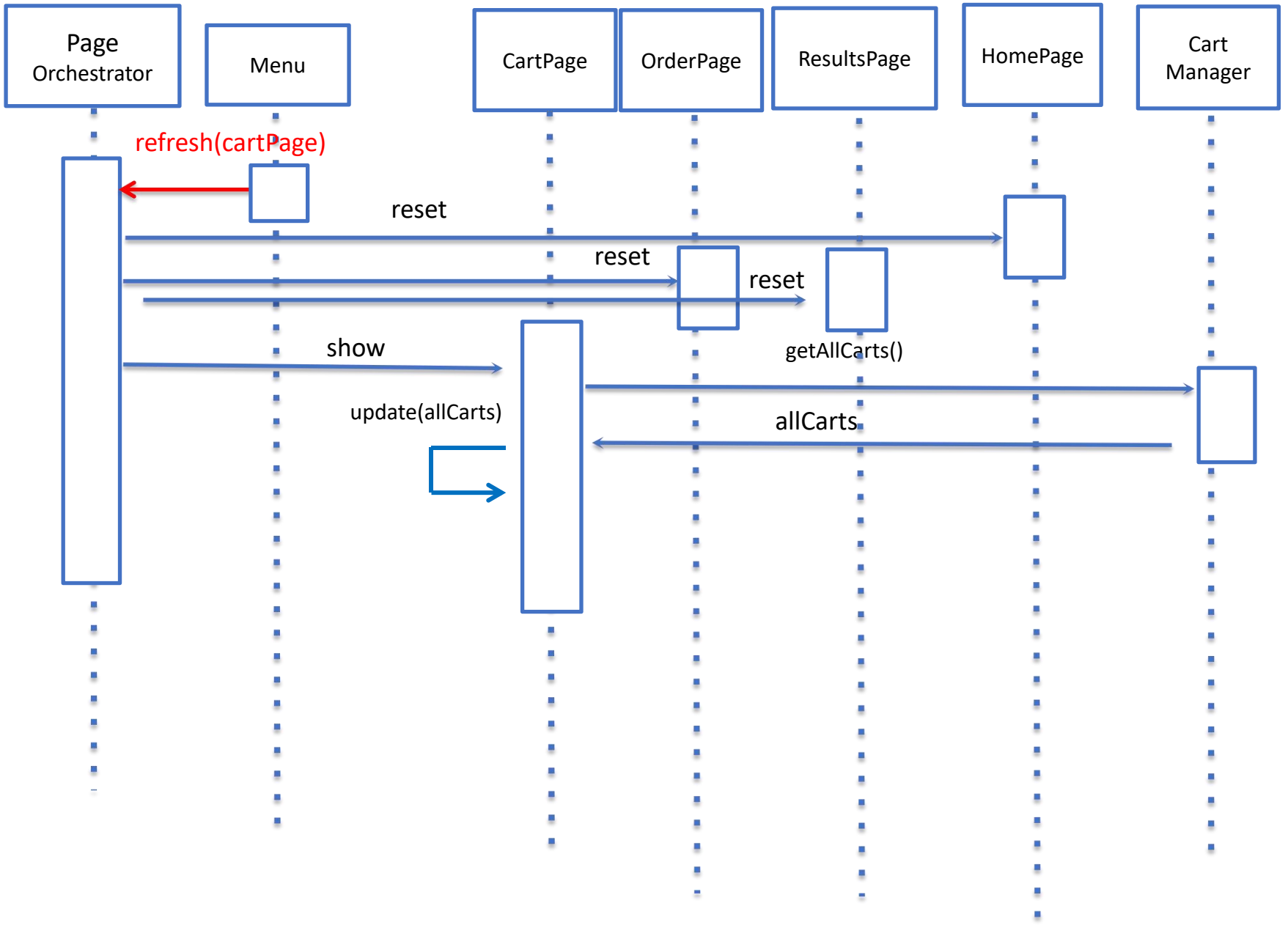
Event: Click HomePage



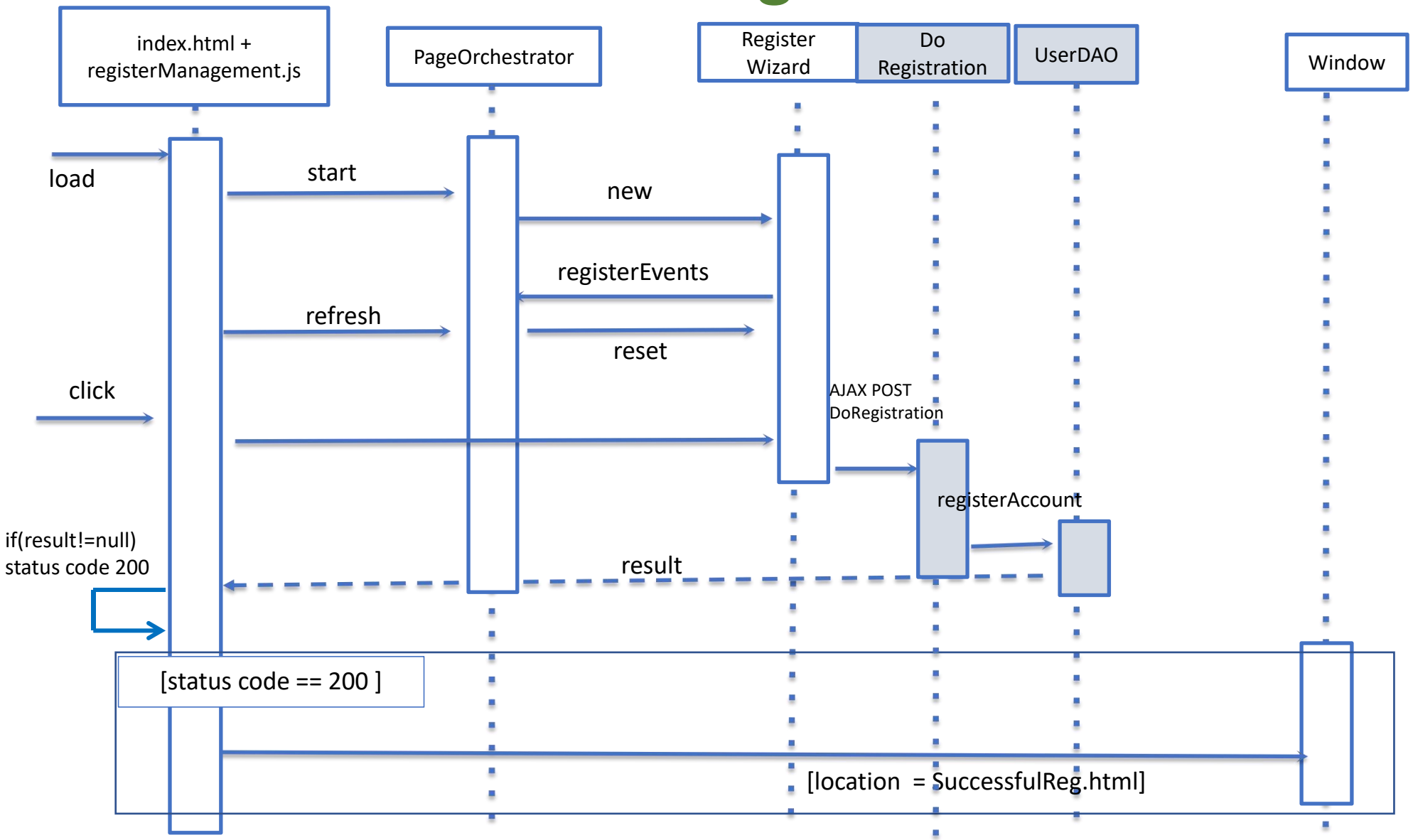
Event: Click OrdersPage



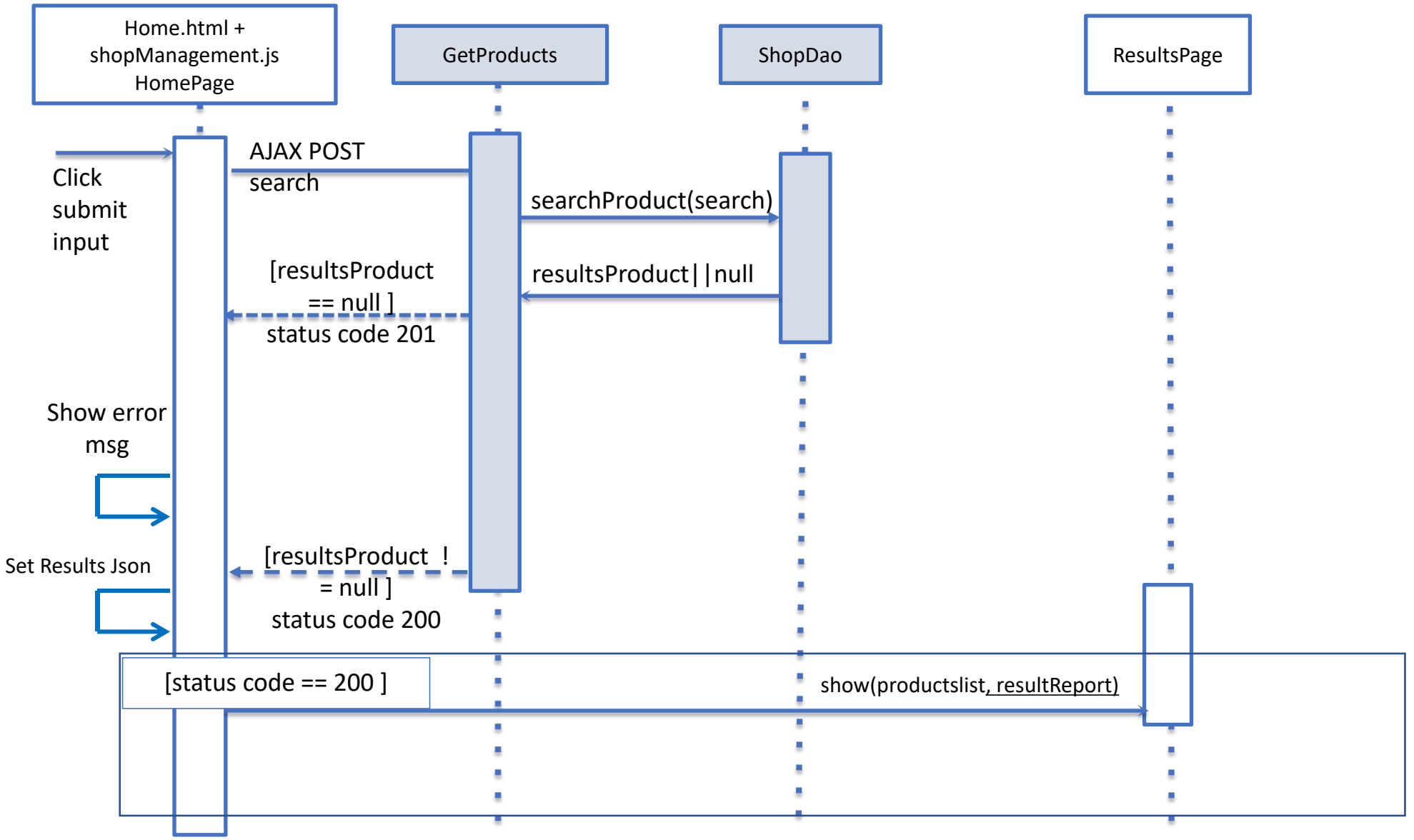
Event: Click CartPage



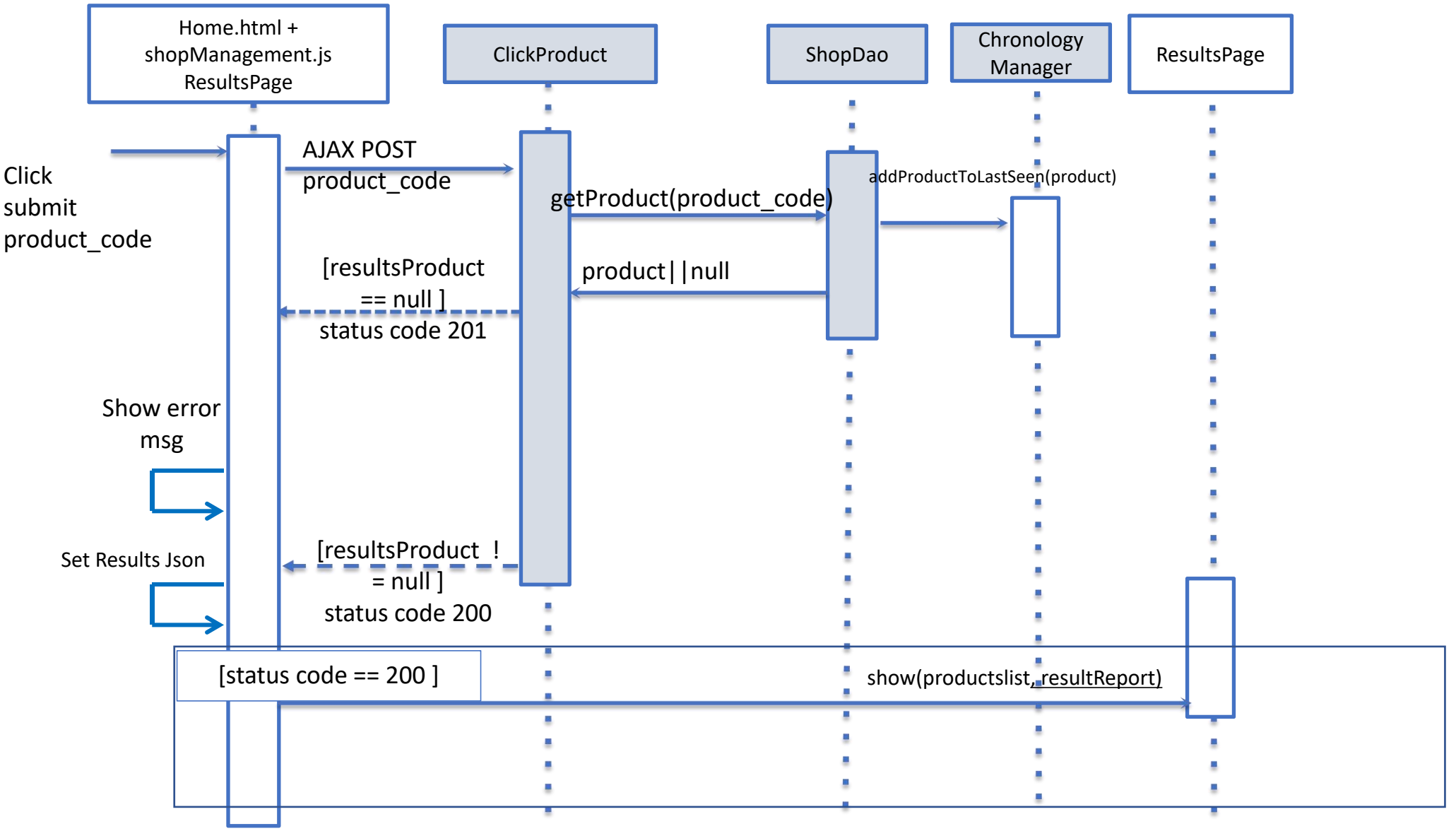
Event: Registration



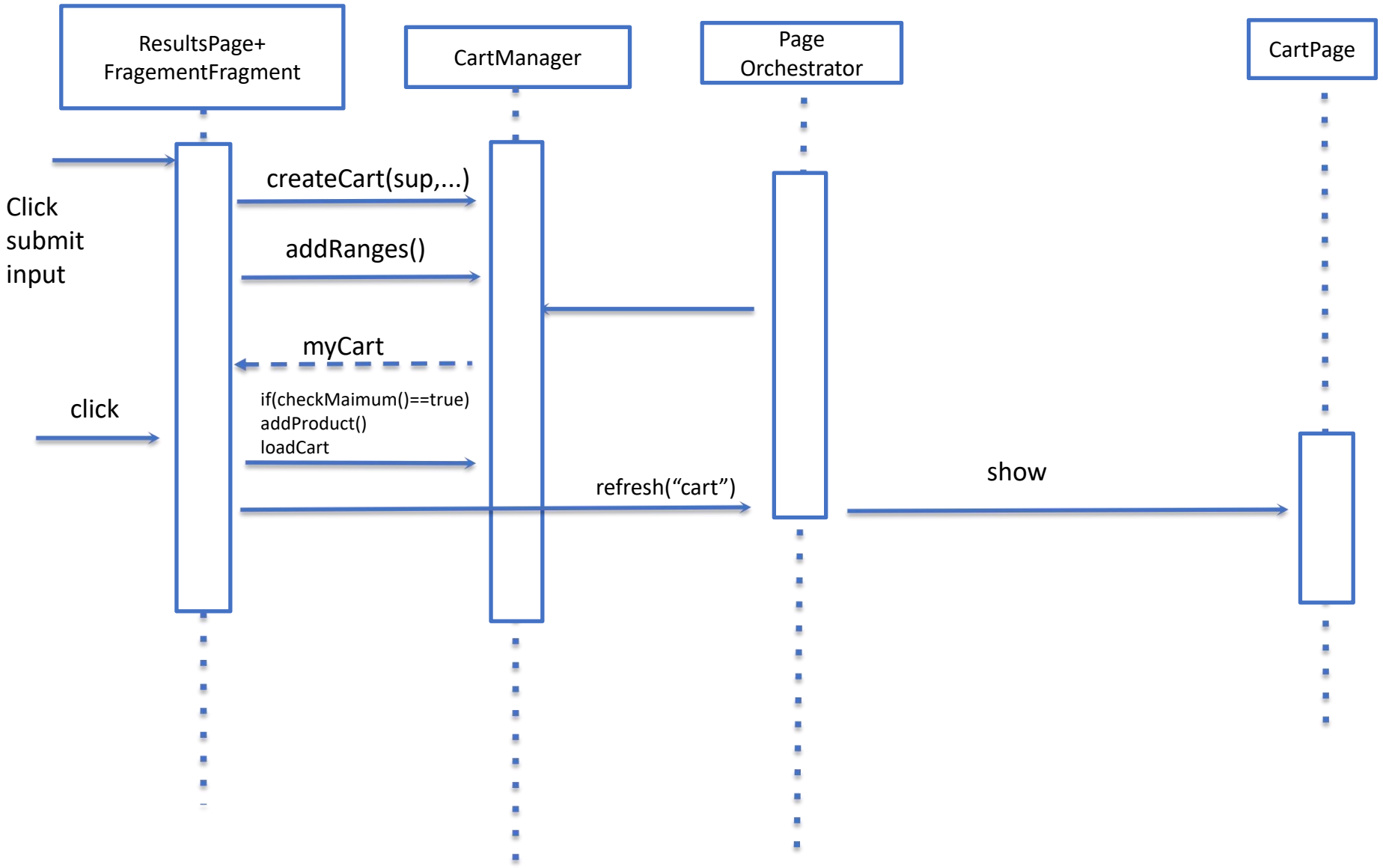
Event: Search Products



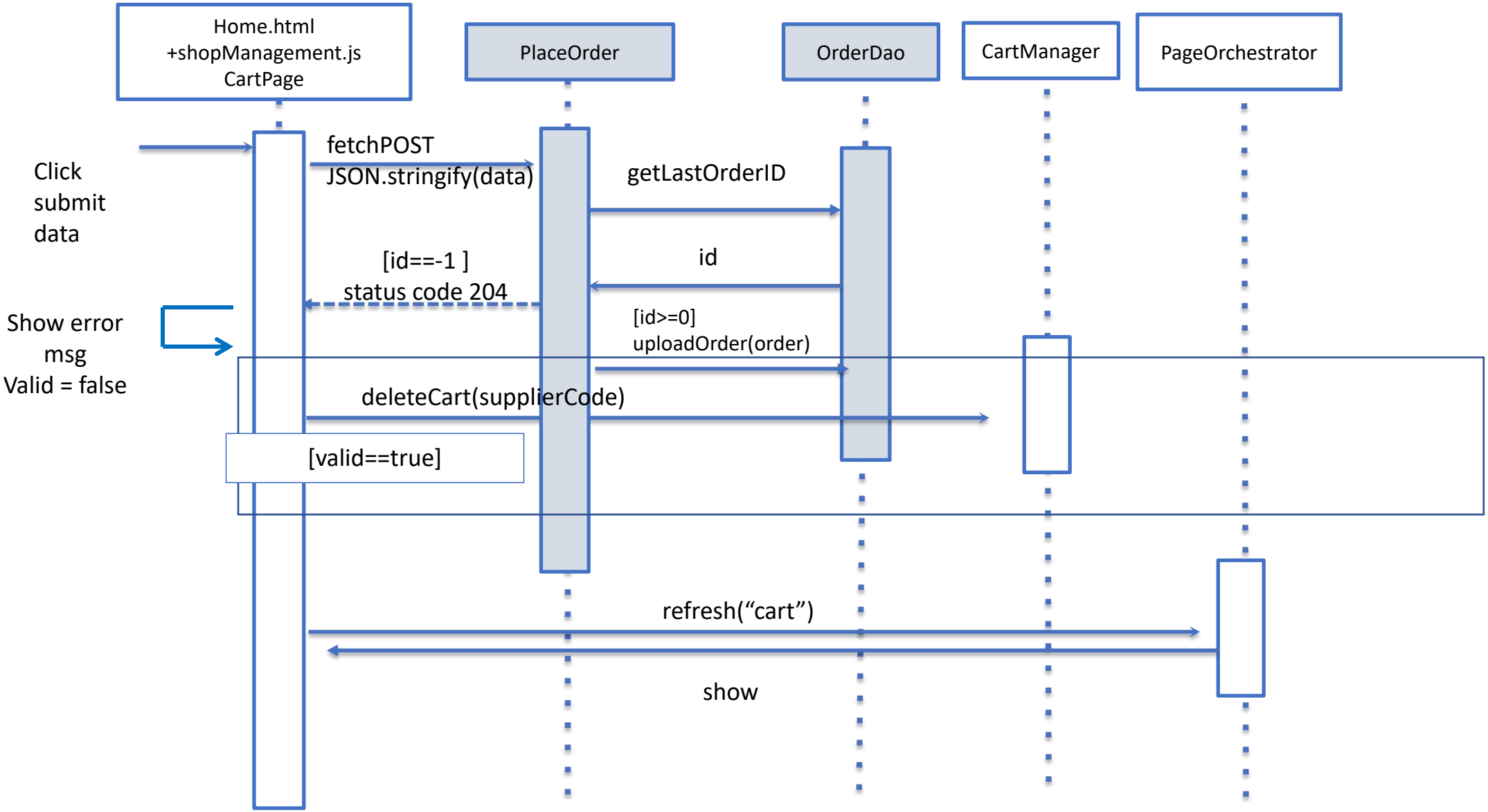
Event: Click Product



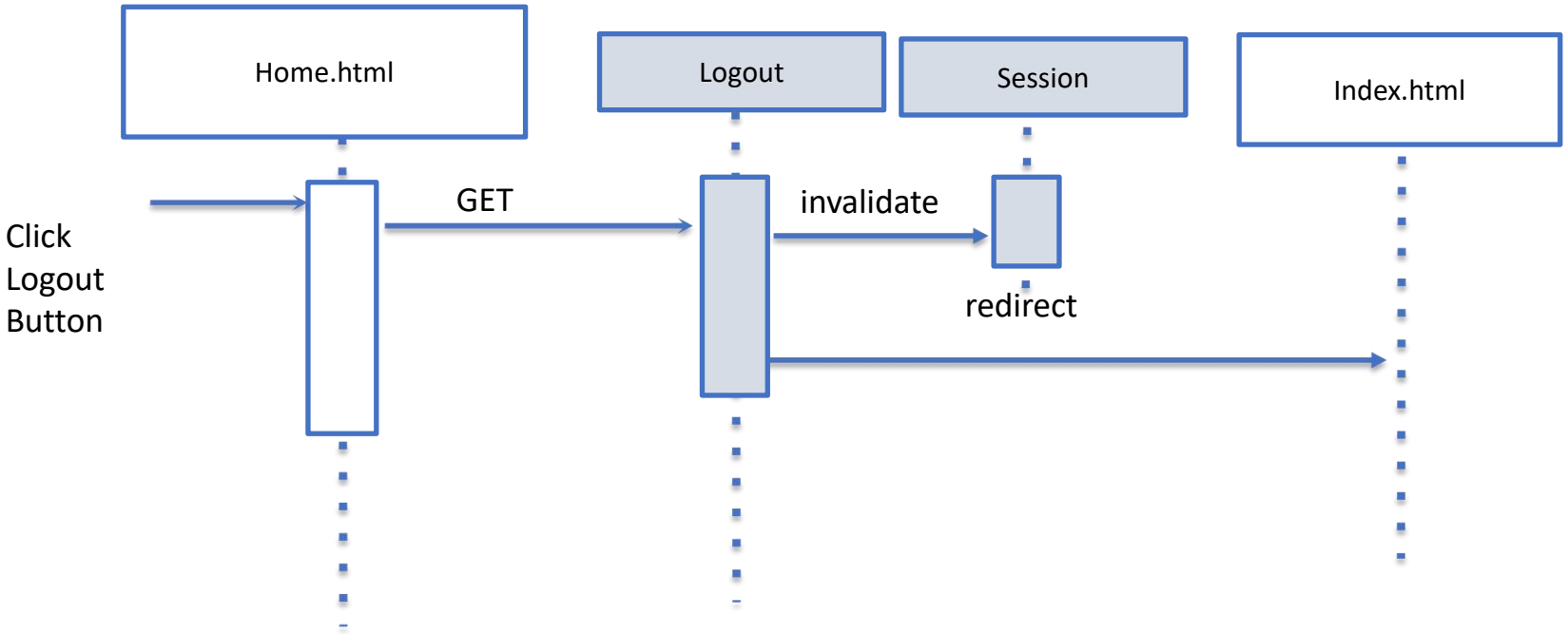
Event: Put In Cart



Event: Place Order



Logout


Home.html
Logout
Session
Index.html
Click
Logout
Button
GET
invalidate
redirect