# Esercizio 5:
# carrello con più fornitori e ordine
# Progettazione

Web technologies

Fabrizio Sordetti

# Es. 5: 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. 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. 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.

# Analisi dei dati

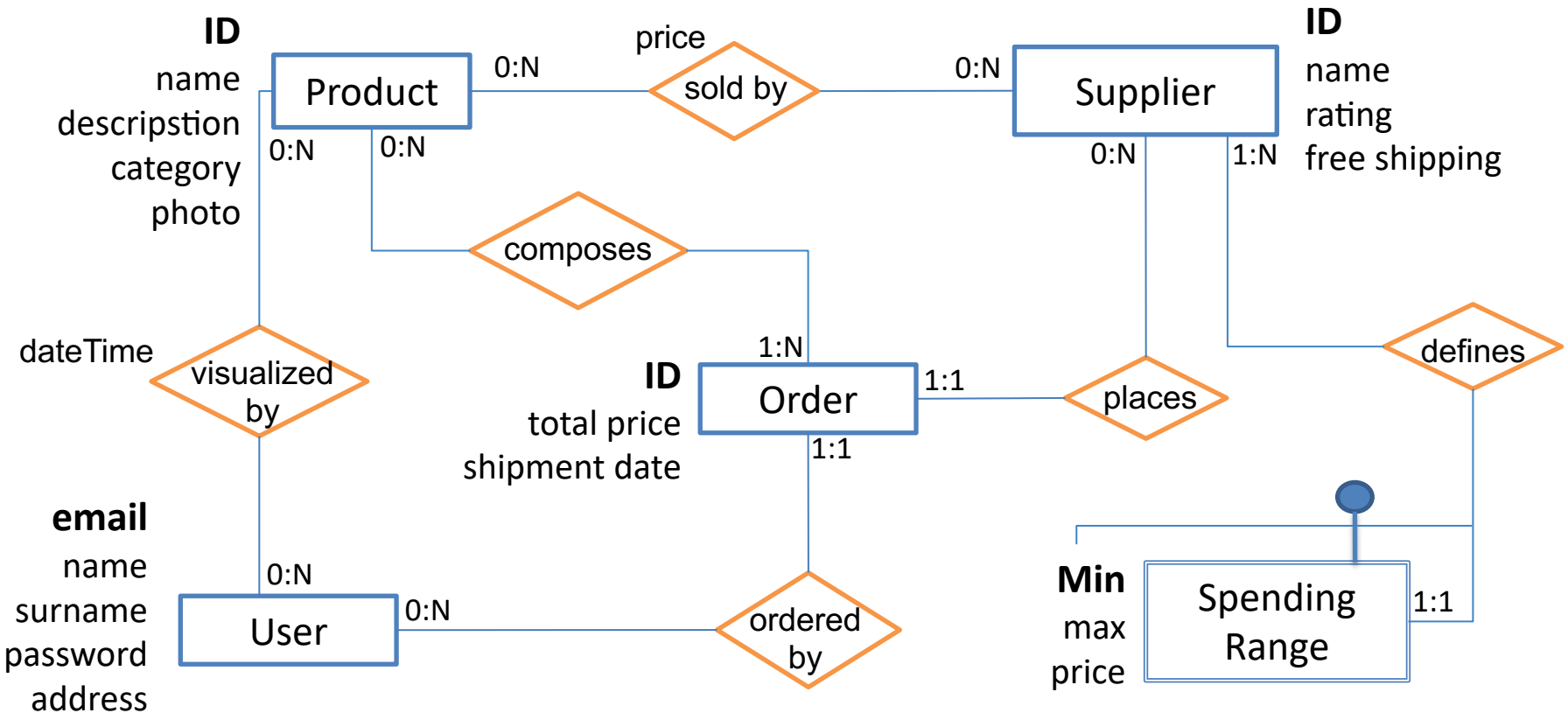
Un'applicazione di commercio elettronico consente all'utente (acquirente) di visualizzare un catalogo di prodotti venduti da diversi fornitori, inserire prodotti in un carrello della spesa e creare un ordine di acquisto a partire dal contenuto del carrello. Un prodotto ha un codice (campo chiave), un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

**Entities, attributes, relationships**

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. 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.

## Entities, attributes, relationships

# Database design



# Local database schema

```
CREATE TABLE `product` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` varchar(45) NOT NULL,
`description` varchar(45),
`category` varchar(45) NOT NULL
DEFAULT 'OFFER',
`photo` LONGBLOB NOT NULL,
PRIMARY KEY (`id`))
CREATE TABLE `user` (
`email` varchar(45) NOT NULL,
`name` varchar(45) NOT NULL,
`surname` varchar(45) NOT NULL,
`password` varchar(45) NOT NULL,
`address` varchar(45),
PRIMARY KEY (`email`))
```

```
CREATE TABLE `supplier` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` varchar(45) NOT NULL,
`rating` int(1) DEFAULT '0',
`freeShipping` decimal(19,4),
PRIMARY KEY (`id`))

CREATE TABLE `spendingRange` (
`min` int(11) NOT NULL,
`max` int(11),
`price` decimal(19,4) NOT NULL,
`idSupp` int(11) NOT NULL,
PRIMARY KEY (`min`, `idSupp`),
CONSTRAINT `suppDefines`
FOREIGN KEY (`idSupp`)
REFERENCES `supplier` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE)
```

# Local database schema

```
CREATE TABLE `order` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`price` decimal(19,4) NOT_NULL,
`shipment` date NOT NULL,
`idSupp` int(11) NOT NULL,
`emailUser` varchar(45) NOT NULL,
PRIMARY KEY (`id`),
CONSTRAINT `placedBy`
FOREIGN KEY (`idSupp`)
REFERENCES `supplier` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `orderedBy`
FOREIGN KEY (`emailUser`)
REFERENCES `user` (`email`)
ON DELETE CASCADE ON UPDATE CASCADE)
```

```
CREATE TABLE `composition` (
`idOrd` int(11) NOT NULL,
`idProd` int(11) NOT NULL,
`numProd` int(11) NOT NULL,
PRIMARY KEY (`idProd`, `idOrd`),
CONSTRAINT `prodComposes`
FOREIGN KEY (`idProd`)
REFERENCES `product` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `composesOrder`
FOREIGN KEY (`idOrd`)
REFERENCES `order` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE))
```

# Local database schema

```
CREATE TABLE `soldBy` (
`idProd` int(11) NOT NULL,
`idSupp` int(11) NOT NULL,
`price` decimal(19,4) NOT NULL,
PRIMARY KEY (`idProd`, `idSupp`),
CONSTRAINT `prodSoldBy`
FOREIGN KEY (`idProd`)
REFERENCES `product` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `soldBySupp`
FOREIGN KEY (`idSupp`)
REFERENCES `supplier` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE)
```

```
CREATE TABLE `visualization` (
`idProd` int(11) NOT NULL,
`emailUser` varchar(45) NOT NULL,
`dateTime` DATETIME NOT NULL
DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (`idProd`, `emailUser`),
CONSTRAINT `prodVis`
FOREIGN KEY (`idProd`)
REFERENCES `product` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `visByUser`
FOREIGN KEY (`emailUser`)
REFERENCES `user` (`email`)
ON DELETE CASCADE ON UPDATE CASCADE)
```

# Application requirements analysis

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.

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

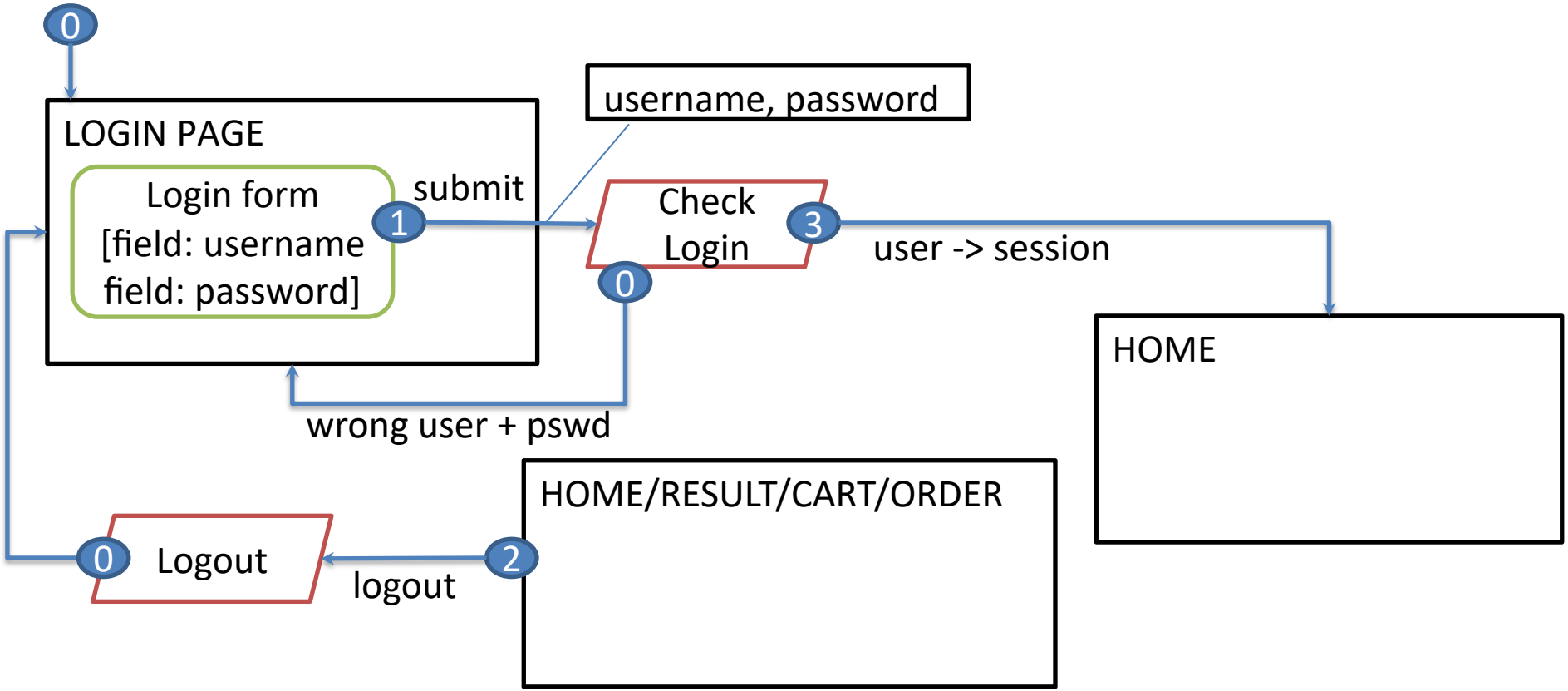
L’utente può inserire una parola chiave di ricerca nel campo di input e premere INVIO. A seguito dell’invio compare una pagina RISULTATI con prodotti che contengono la chiave di ricerca nel nome o nella descrizione. L’elenco mostra solo il codice, il nome del prodotto e il prezzo minimo di vendita del 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. 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.

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

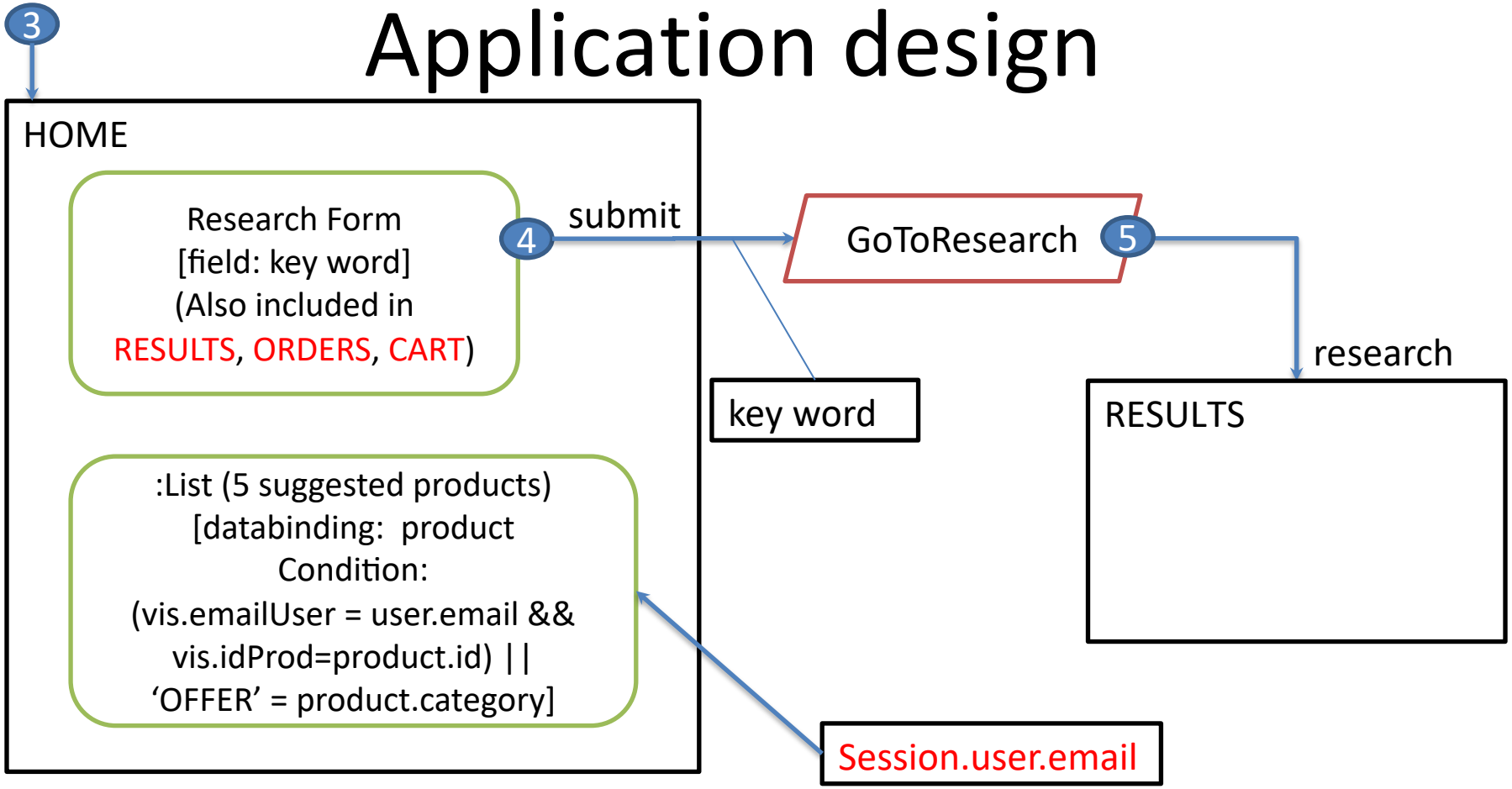
# Completamento delle specifiche

- Esiste una pagina di default (LOGIN PAGE) contenente la form di login.
- La valutazione dei fornitori è rappresentate da un numero intero e l'indirizzo dell'utente è rappresentato con una stringa per semplicità.
- Deve essere possibile fare logout e tornare alla pagina di login.
- Conseguentemente alla creazione di un ordine l'utente viene rediretto sulla pagina ORDINI.
- In tutte le pagine (a meno della LOGIN PAGE) è presente il menù e il campo di ricerca.
- Nella pagina RISULTATI anche prima della selezione del prodotto vengono mostrare le immagini di tutti i prodotti.

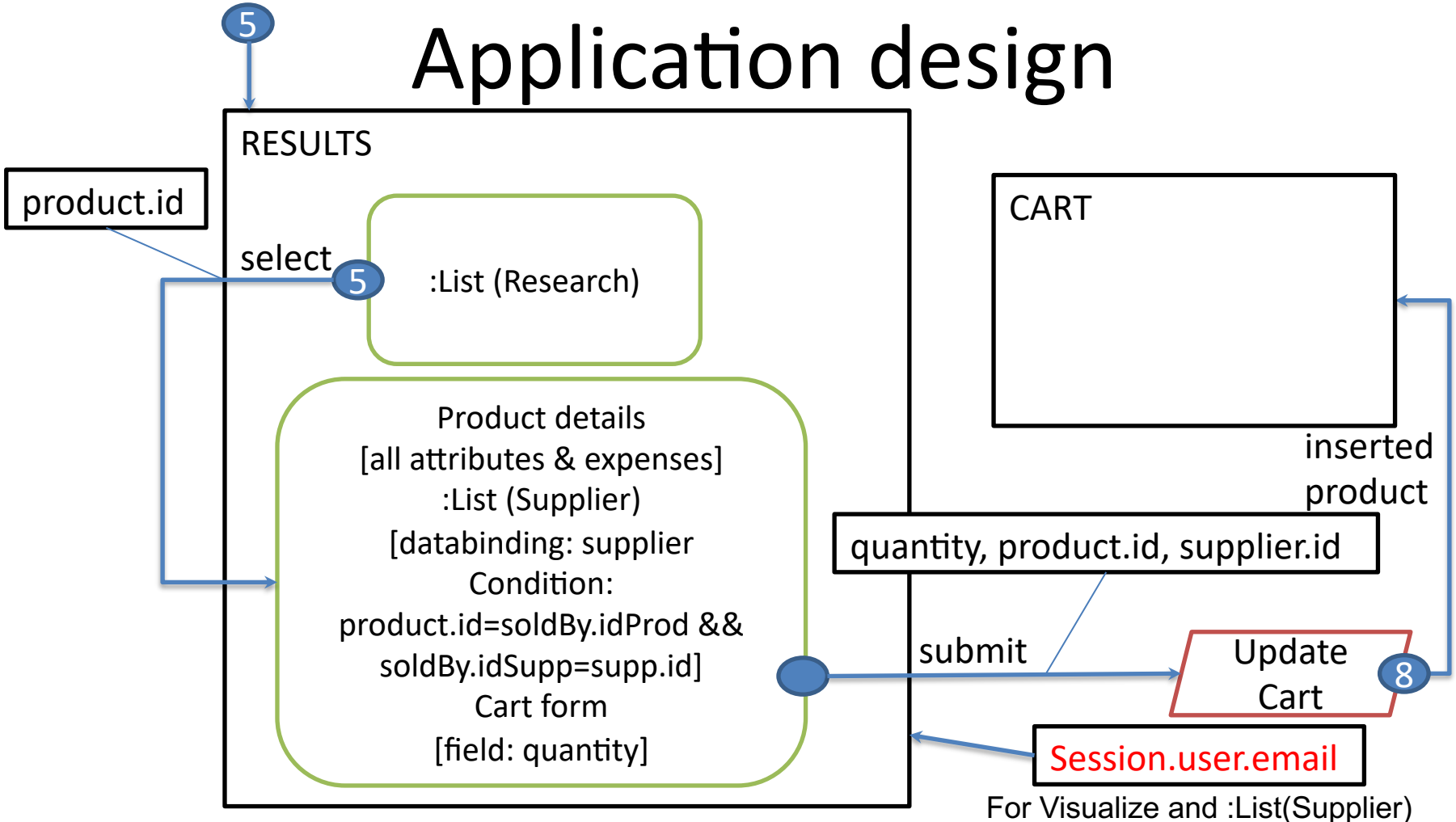
# Application design



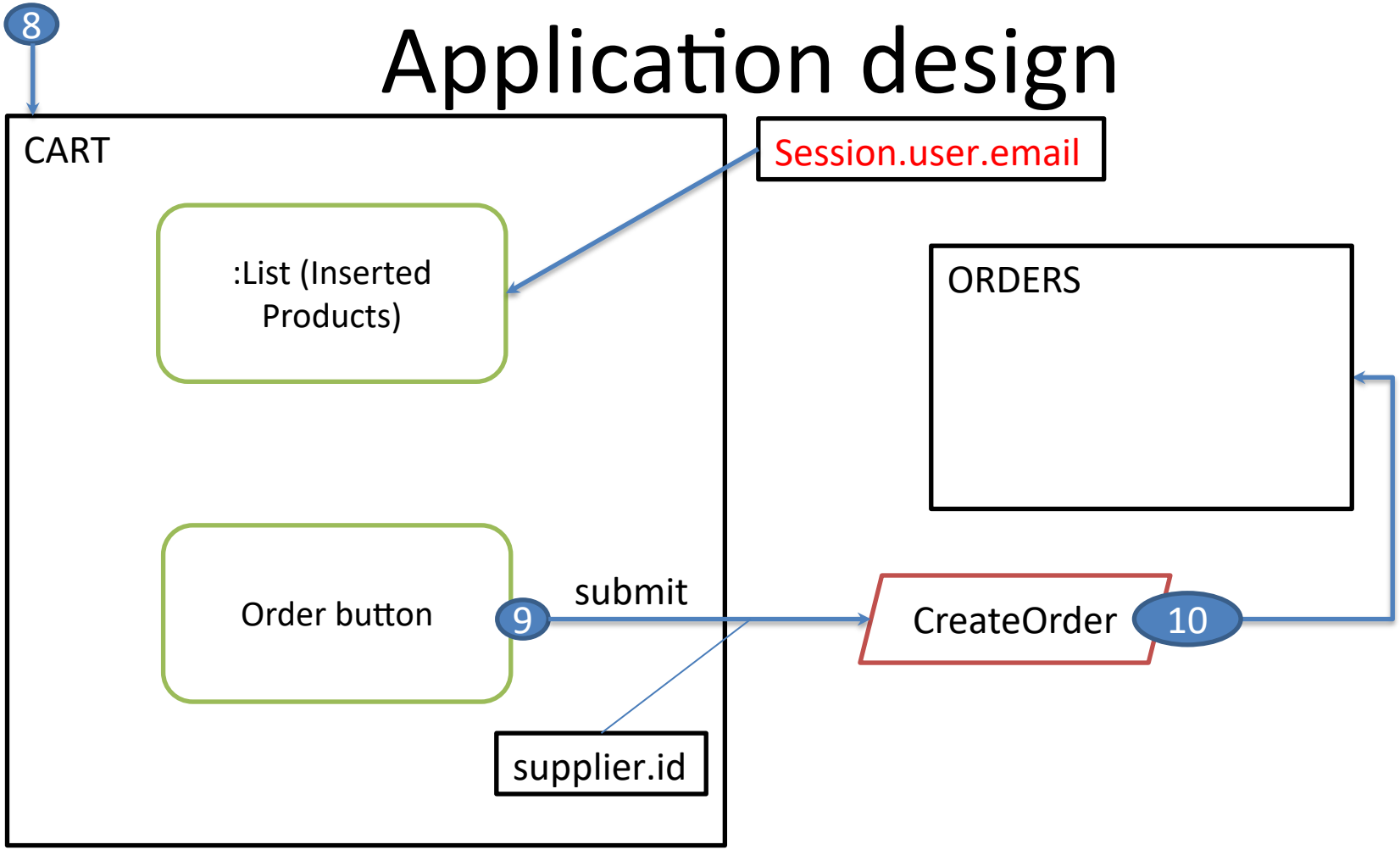
# Application design



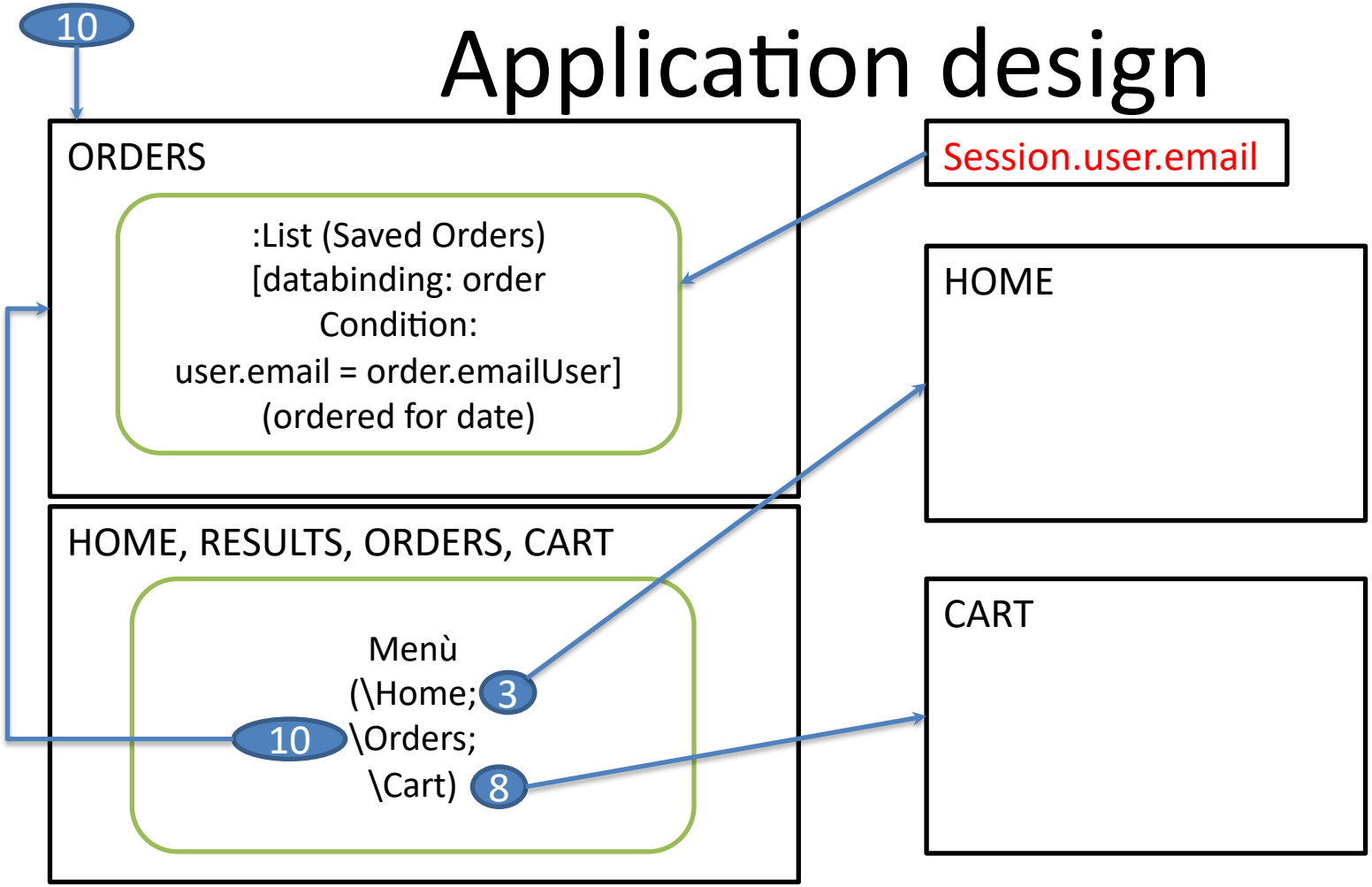
# Application design



# Application design



# Application design



# Events and Actions

- Events
  0. Accedere alla pagina iniziale di Login
  1. Login
  2. Logout
  3. Accedere alla Home (dal login e dal menù)
  4. Cercare Prodotto
  5. Accedere alla pagina dei risultati
     1. Ricercando un prodotto dalla Home
     2. Selezionando un prodotto per vederlo nel dettaglio
  6. Visualizzare un prodotto
  7. Inserire nel carrello una quantità di prodotti
  8. Accedere al carrello (dai risultati e dal menù)
  9. Effettuare un ordine
  10. Accedere alla pagina degli ordini (dal carrello e dal menù)

- Actions
  1. Verificare credenziali
  2. Effettuare Logout
  3. Preparare contenuto Home
  4. Trovare prodotti ricercati
  5. Preparare contenuto pagina dei risultati
     1. Prepara Lista Ricerca
     2. Prepara Dettaglio prodotto e Lista fornitori
  6. Rendere il prodotto visualizzato
  7. Aggiornare contenuto del carrello
  8. Preparare contenuto carrello
  9. Creare un ordine e aggiornare contenuto carrello (eliminare prodotti del fornitore)
  10. Preparare contenuto pagina degli oridni

# Components

- Data Access Objects (Classes)
  - ProductDAO
    - fiveProductsForUser(userEmail)
    - findProductsByKeyWord(keyWord)
    - findProductsByIDs(productID)
  - UserDAO
    - checkCredentials(email, pwd)
  - SupplierDAO
    - findSuppliersByProduct(productID)
    - findSupplierByID(supplierID)
  - SpendingRangeDAO
    - findSpendingRangesBySupplier(suppID)
  - OrderDAO
    - createOrder(userEmail, suppID, productsIDs)
    - findOrdersForUser(userEmail)
  - CompositionDAO
    - createComposition(orderID, productsIDs)
  - SoldByDAO
    - sumOfPrices(productsIDs, suppdID)
  - VisualizationDAO
    - productsVisualizedBy(userEmail)
    - visualize(product, userEmail)

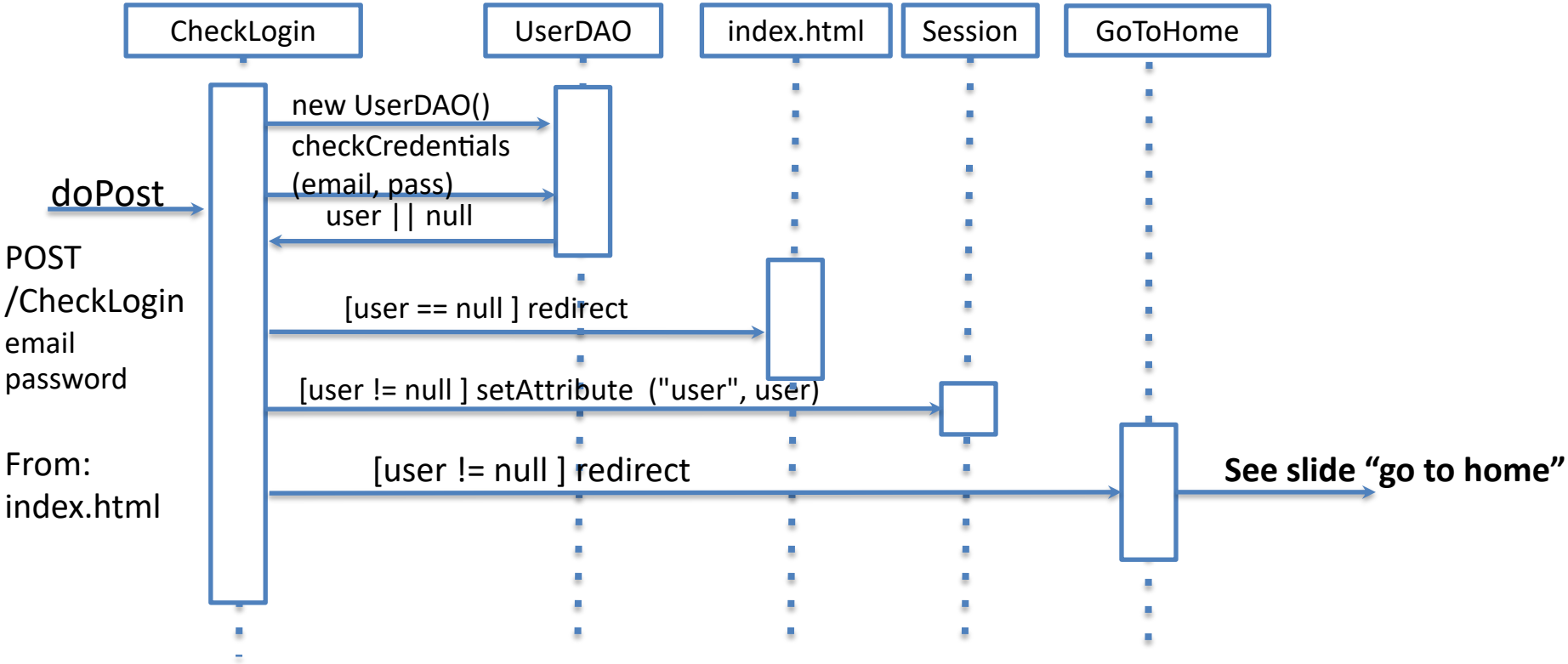
- Model Objects (Beans)
  - Product
  - User
  - Supplier
  - SpendingRange
  - Order

# Components

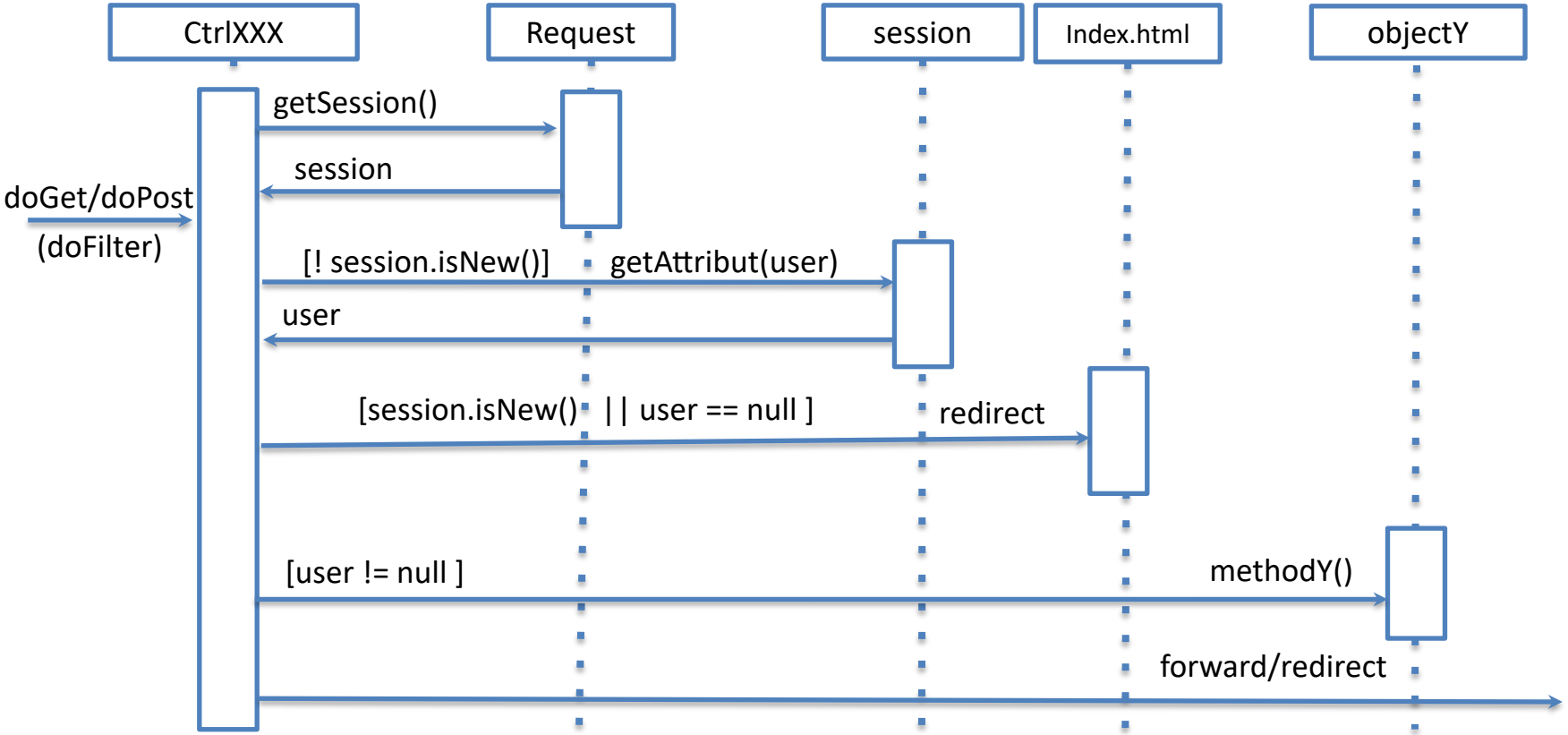
- Views (Templates) & components
  - Login
    - Login form
  - Home
    - Menù
    - Research Form (included in Menù)
    - 5 Suggested Products List (with links to search each one)
  - Results
    - Menù (Research Form included)
    - Researched Products List
    - Selected Product Details
    - Selected Product Supplier List
    - Selected Product Cart Form
  - Cart
    - Menù (Research Form included)
    - Inserted Products List (ordered by Suppliers)
    - Order Button (for each Supplier)
  - Orders
    - Menù (Research Form included)
    - SavedOrders List

- Controllers (Servlets)
  - CheckLogin
  - GoToResearch
  - UpdateCart
  - CreateOrder
  - GoToHome 3
  - GoToCart 5
  - GoToOrders 10

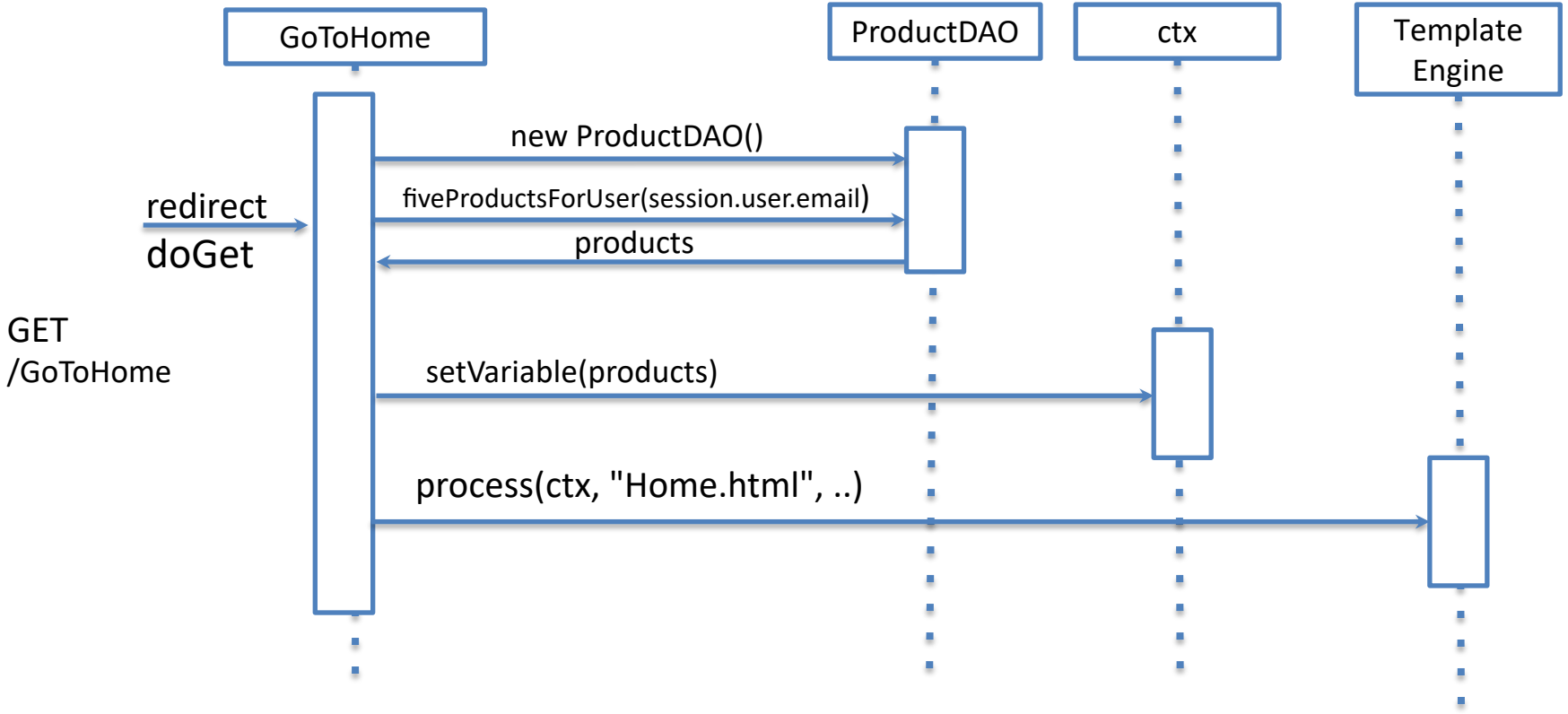
# Event: login


CheckLogin
UserDAO
index.html
Session
GoToHome
doPost
POST
/CheckLogin
email
password
From:
index.html
new UserDAO()
checkCredentials
(email, pass)
user || null
[user == null ] redirect
[user != null ] setAttribute ("user", user)
[user != null ] redirect
See slide “go to home”

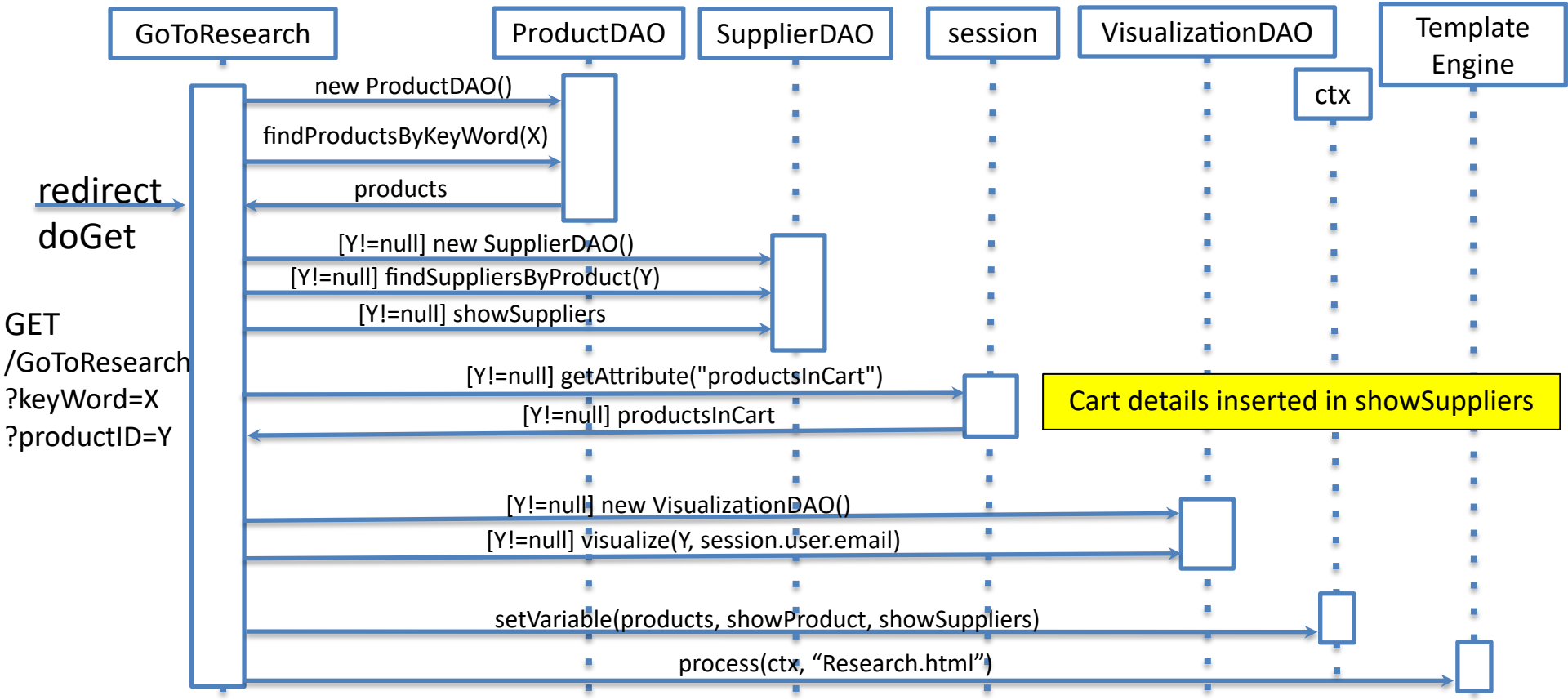
# Checking if the user is logged in with filter



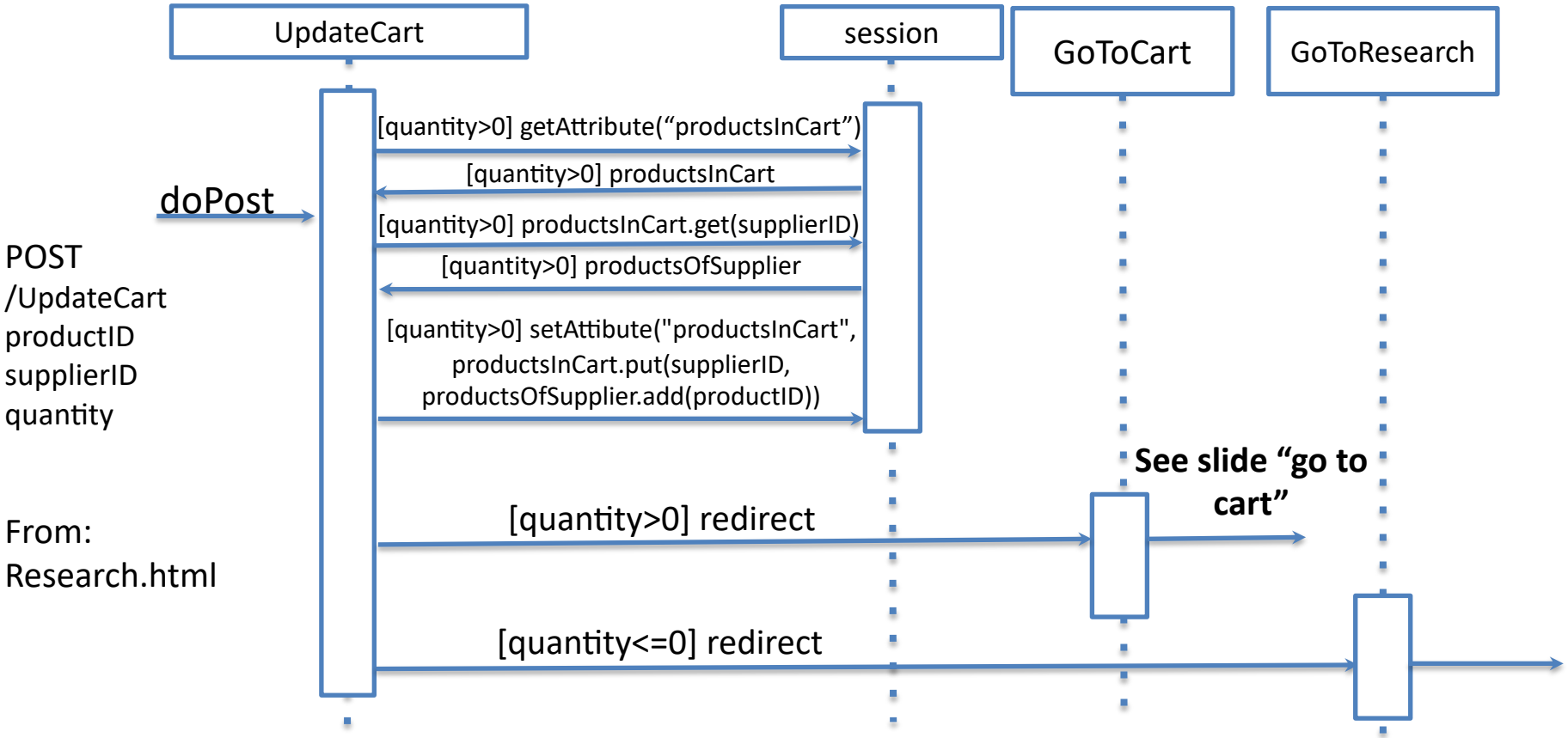
# Event: go to home



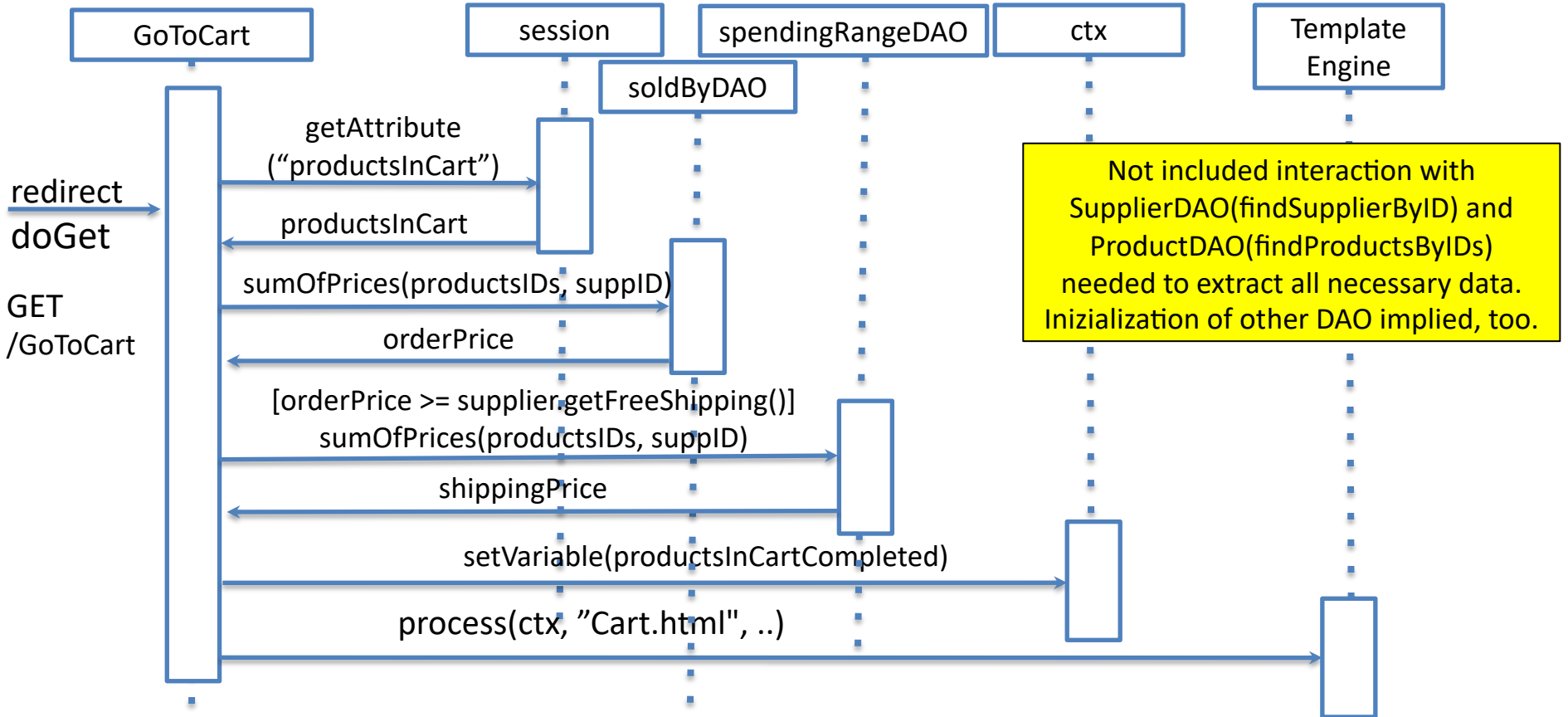
# Event: go to research



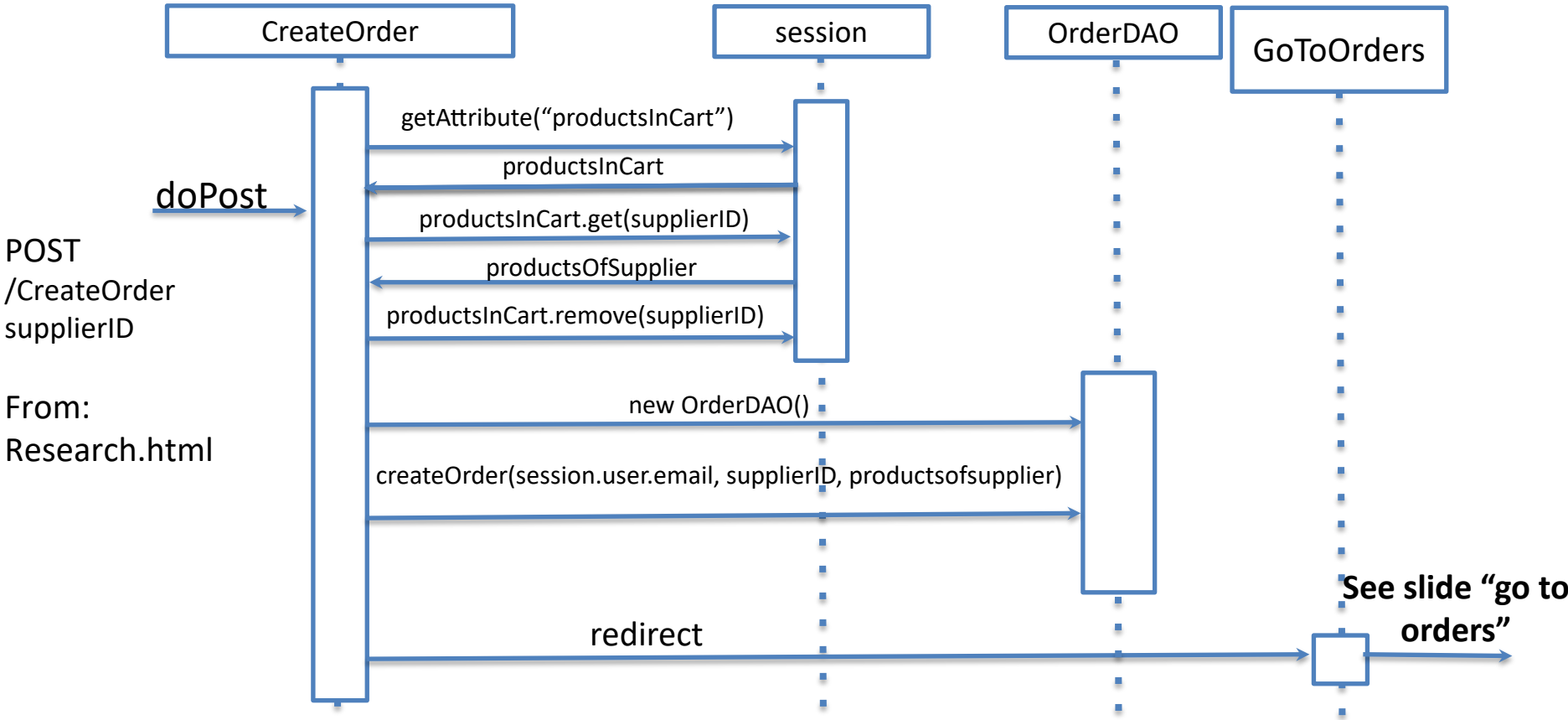
# Event: add product to cart



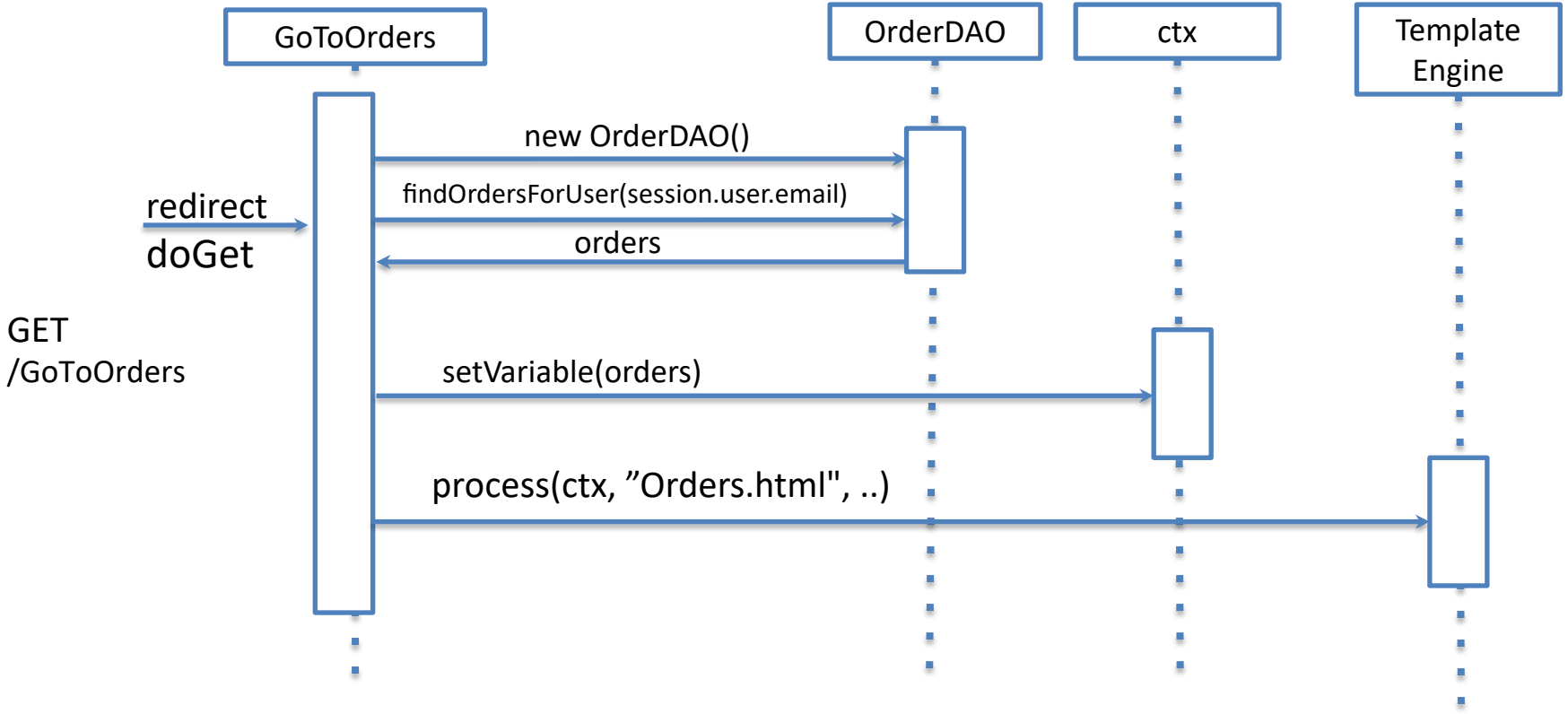
# Event: go to cart



# Event: create order



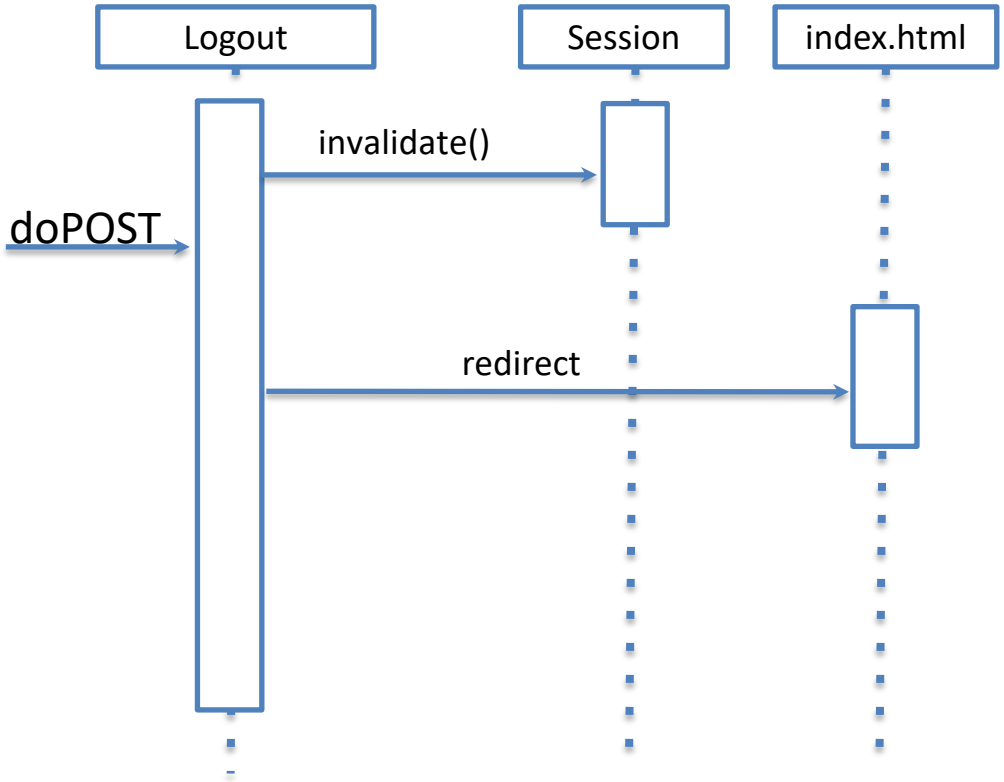
# Event: go to orders



# Event: logout

POST
/Logout

From:
Home
Research
Cart
Order



# Versione con JavaScript

Si realizzi un'applicazione client server web che estende e/o modifica le specifiche precedenti come segue:

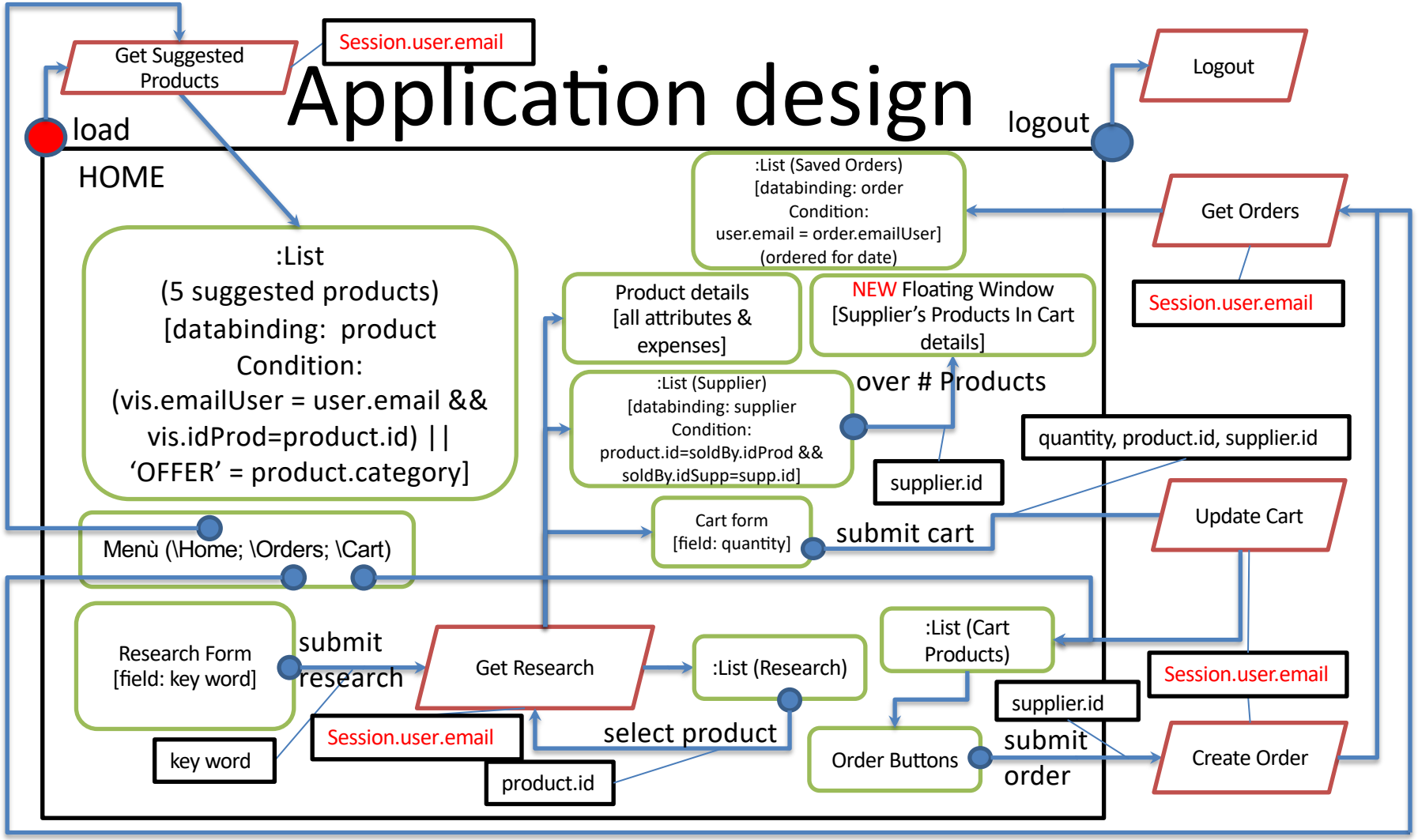
- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- L'applicazione 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.

# Application requirements analysis

Si realizzi un'applicazione client server web che estende e/o modifica le specifiche precedenti come segue:

- 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.

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


Application design
Get Suggested Products
Session.user.email
load
HOME
logout
Logout
:List (Saved Orders)
[databinding: order
Condition:
user.email = order.emailUser]
(ordered for date)
Get Orders
Session.user.email
:List
(5 suggested products)
[databinding: product
Condition:
(vis.emailUser = user.email &&
vis.idProd=product.id) ||
'OFFER' = product.category]
Product details
[all attributes & expenses]
NEW Floating Window
[Supplier's Products In Cart details]
:List (Supplier)
[databinding: supplier
Condition:
product.id=soldBy.idProd &&
soldBy.idSupp=supp.id]
over # Products
supplier.id
quantity, product.id, supplier.id
Cart form
[field: quantity]
submit cart
Update Cart
Menù (\Home; \Orders; \Cart)
Research Form
[field: key word]
submit research
Get Research
:List (Research)
:List (Cart Products)
Session.user.email
supplier.id
key word
Session.user.email
select product
product.id
Order Buttons
submit order
Create Order

# Server side: DAO & model objects

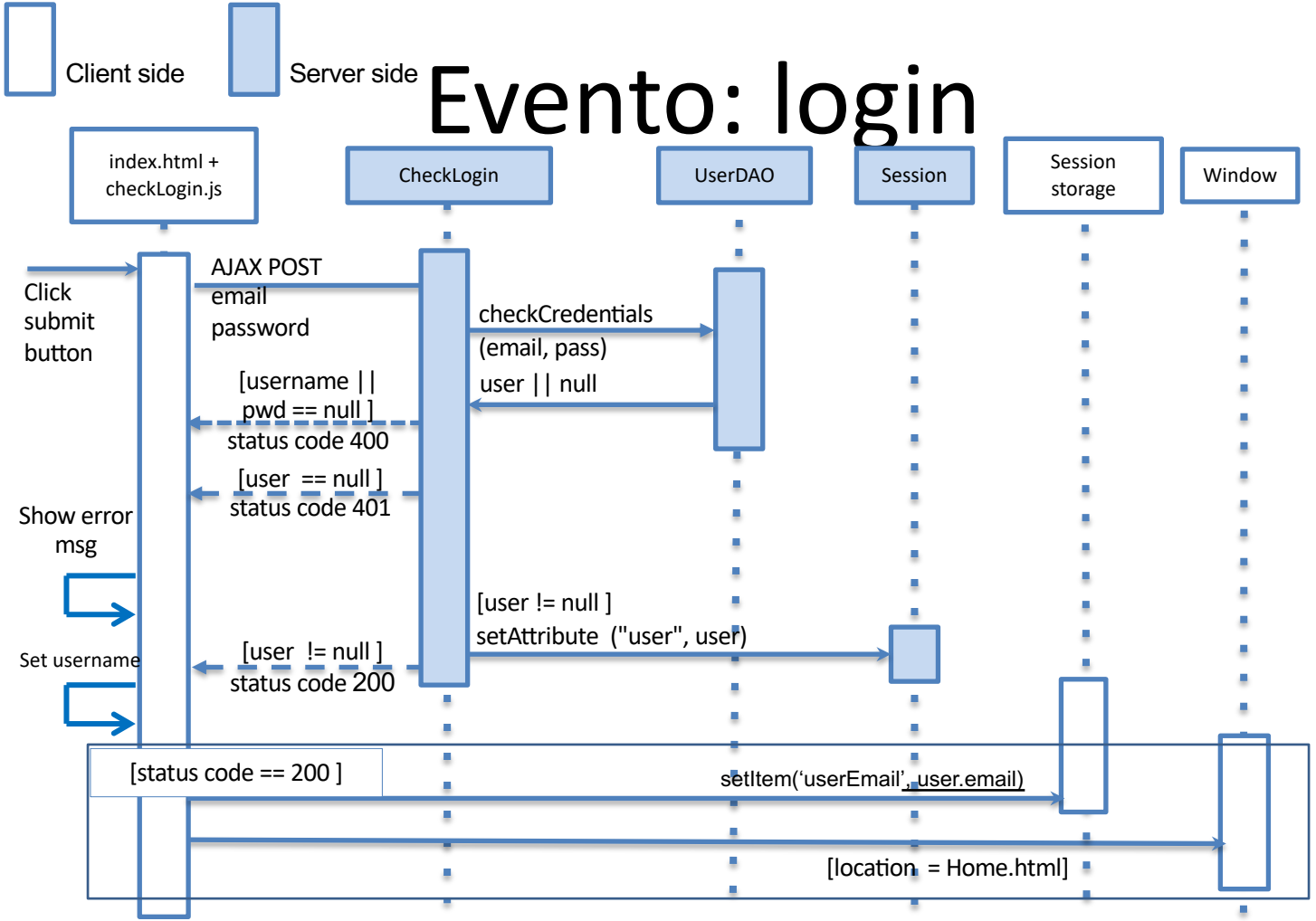
- Controllers
  - CheckLogin
  - Get Suggested Products
  - Get Research
  - Update Cart
  - Create Order
  - Get Orders
  - Logout
- Model objects (Beans)
  - Product
  - User
  - Supplier
  - SpendingRange
  - Order

- Data Access Objects (Classes)
  - ProductDAO
    - fiveProductsForUser(userEmail)
    - findProductsByKeyWord(keyWord)
    - findProductsByIDs(productID)
  - UserDAO
    - checkCredentials(email, pwd)
  - SupplierDAO
    - findSuppliersByProduct(productID)
    - findSupplierByID(supplierID)
  - SpendingRangeDAO
    - findSpendingRangesBySupplier(suppID)
  - OrderDAO
    - createOrder(userEmail, suppID, productsIDs)
    - findOrdersForUser(userEmail)
  - CompositionDAO
    - createComposition(orderID, productsIDs)
  - SoldByDAO
    - sumOfPrices(productsIDs, suppdID)
  - VisualizationDAO
    - productsVisualizedBy(userEmail)
    - visualize(product, userEmail)

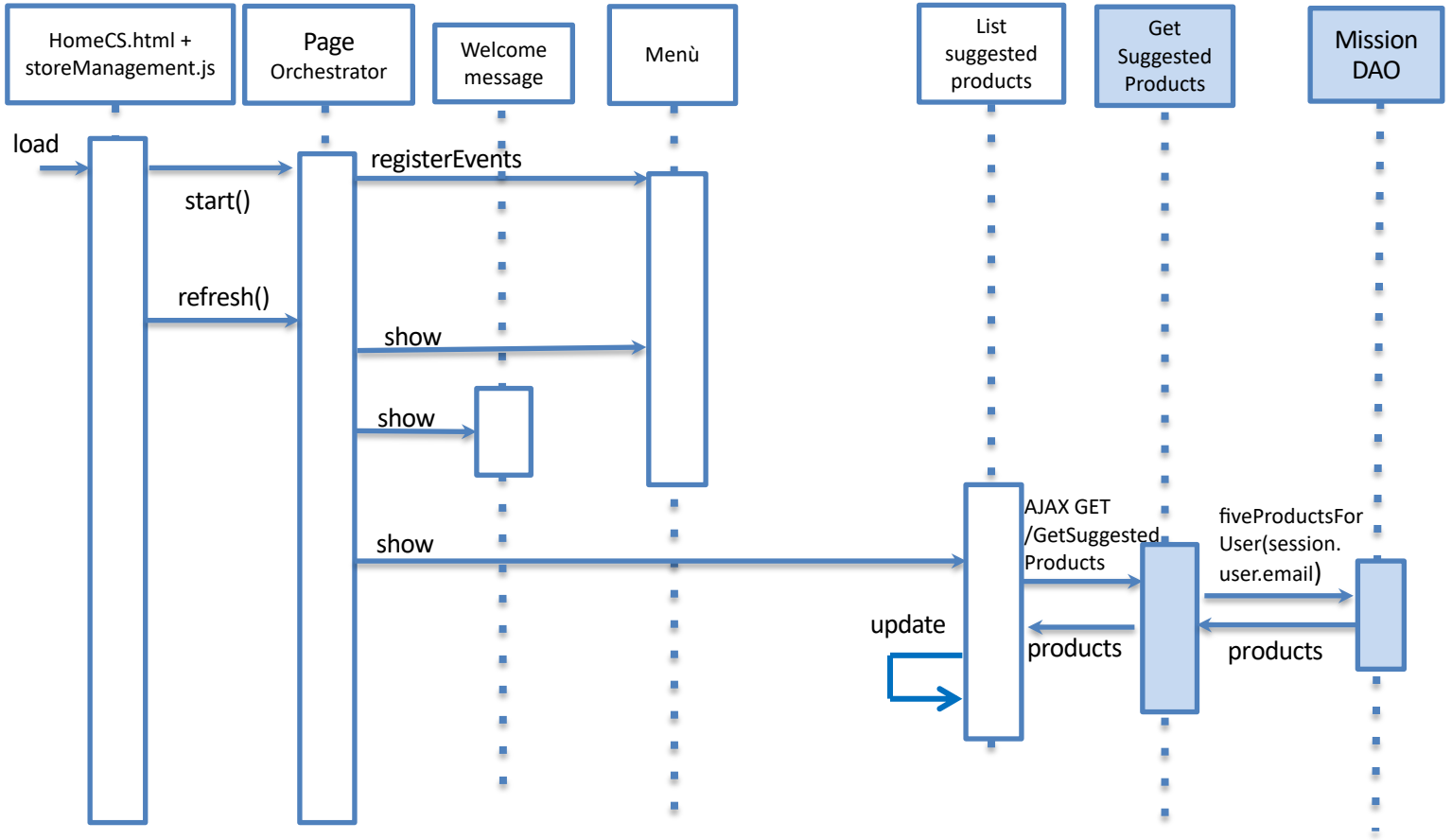
# Client side: view & view component

- Index
  - Login Form
    - Gestione del submit e degli errori
- Home
  - List suggested products
    - show(): richiede al server i dati dell'elenco dei 5 prodotti da suggerrire all'utente
    - update(): riceve dati server e aggiorna la lista
  - Menù: un unico comonent con sotto componenti per menù di navigazione e research form
    - registerEvents(): associa al componente le funzioni per gestirne gli eventi
    - show(): mostra il menù con la pagina corrente evidenziata e il research form
  - List Research
    - show(): riceve dati server e mostra i dati dei prodotti inerenti alla ricerca
    - update(): evidenzia il prodotto attualmente selezionato
  - Product Details: unico componente con sotto componenti per dati del prodotto, dati dei fornitori e bottone del carrello
    - show(): richiede al server i dettagli del prodotto
    - update(): riceve dati server e aggiorna i dettagli
  - Floating Window
    - show(): recupera i dati del carrello lato client
    - update(): riceve i dati e mostra la finestra sovrapposta
  - List Cart: un unico comonent con sotto componenti per dati dei prodotti nel carrello e bottone ordina
    - show(): recupera i dati del carrello lato client
    - update(): aggiorna la lista dei prodotti nel carrello
  - List Orders
    - show(): richiede al server i dati dell'elenco dei prodotti inerenti alla ricerca
    - update(): riceve dati server e aggiorna la lista
- Page Orchestrator
  - start(): crea e inizializza i componenti dell'interfaccia registrando i gestori degli eventi (mapping)
  - refresh() orchestra il reperimento dei contenuti e la visualizzazione dei componenti

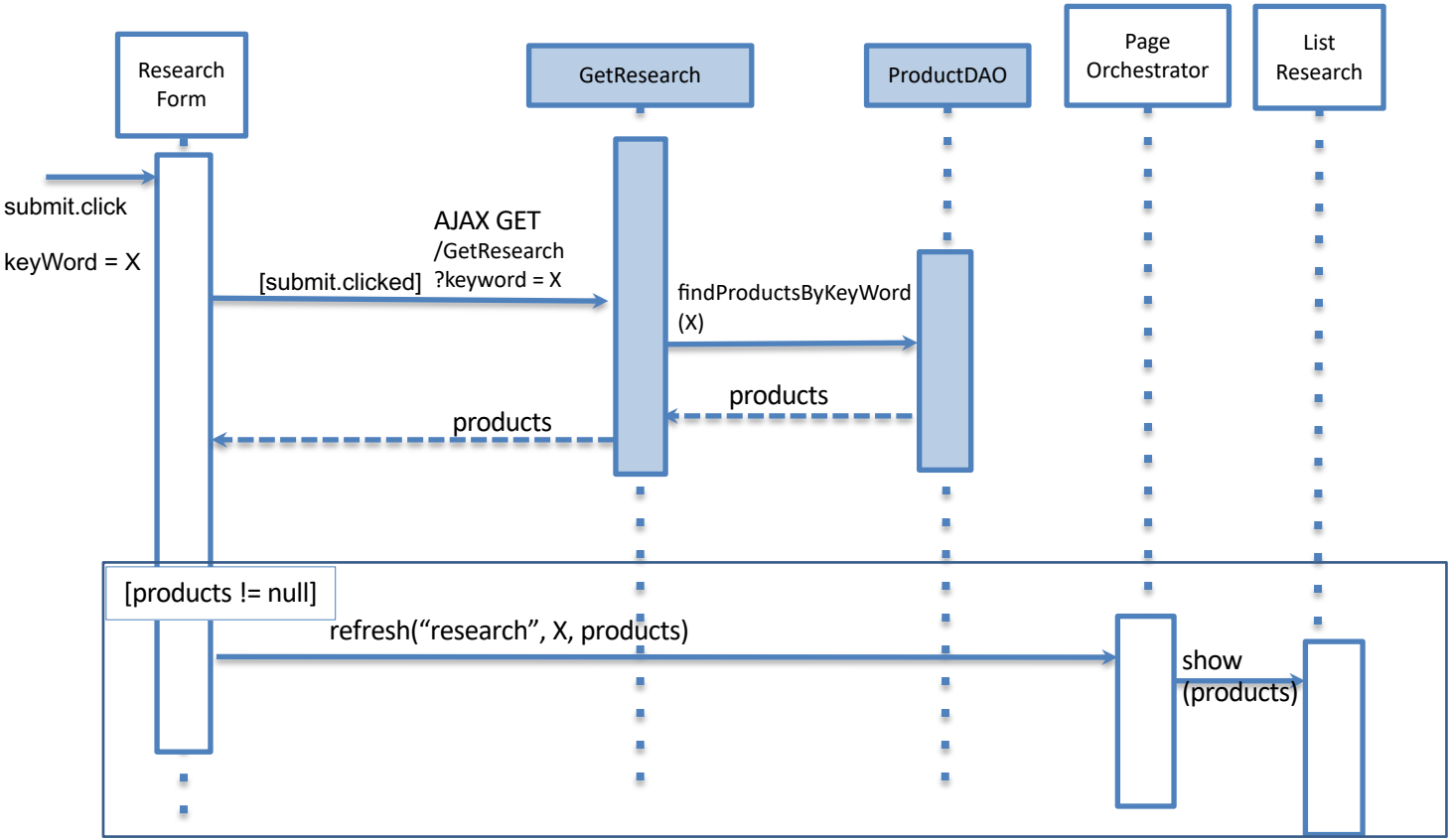
# Evento: login



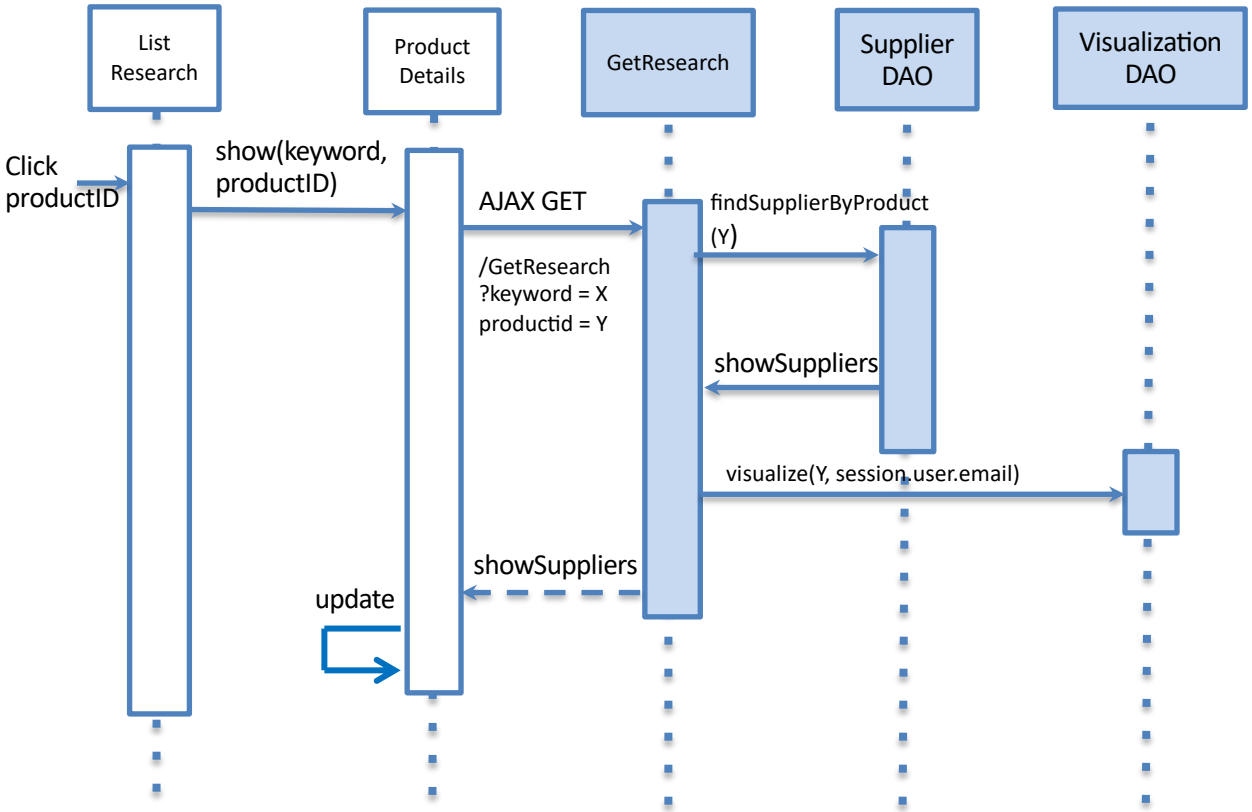
# Evento: caricamento Home page



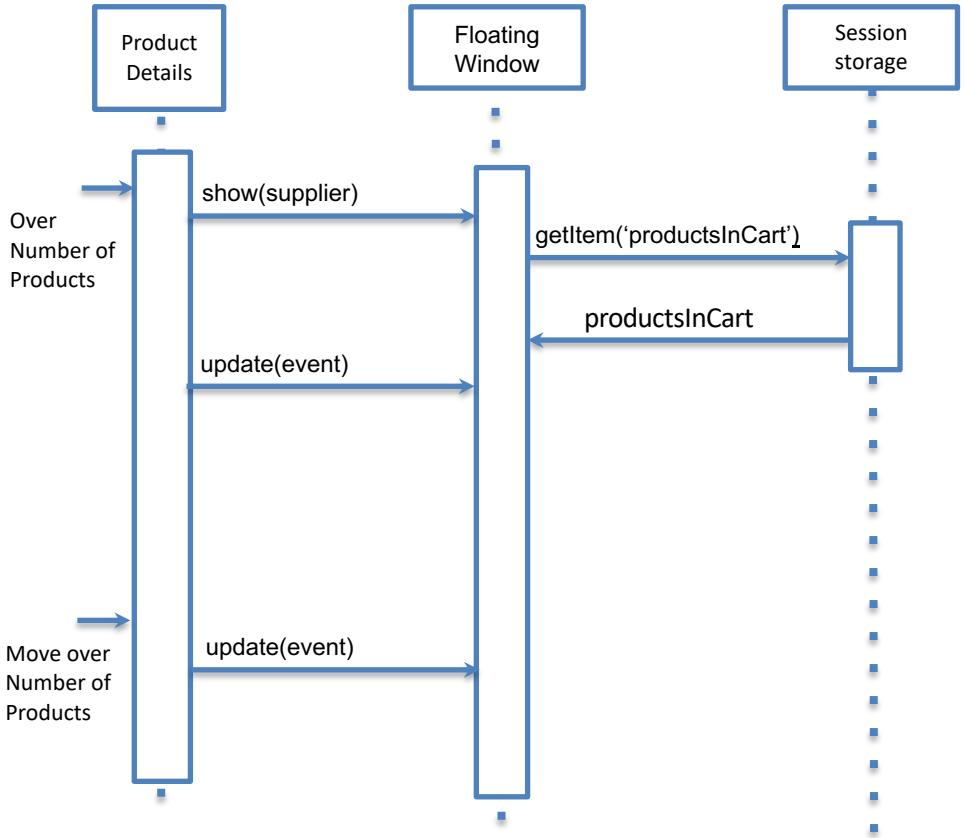
# Evento: effettua ricerca



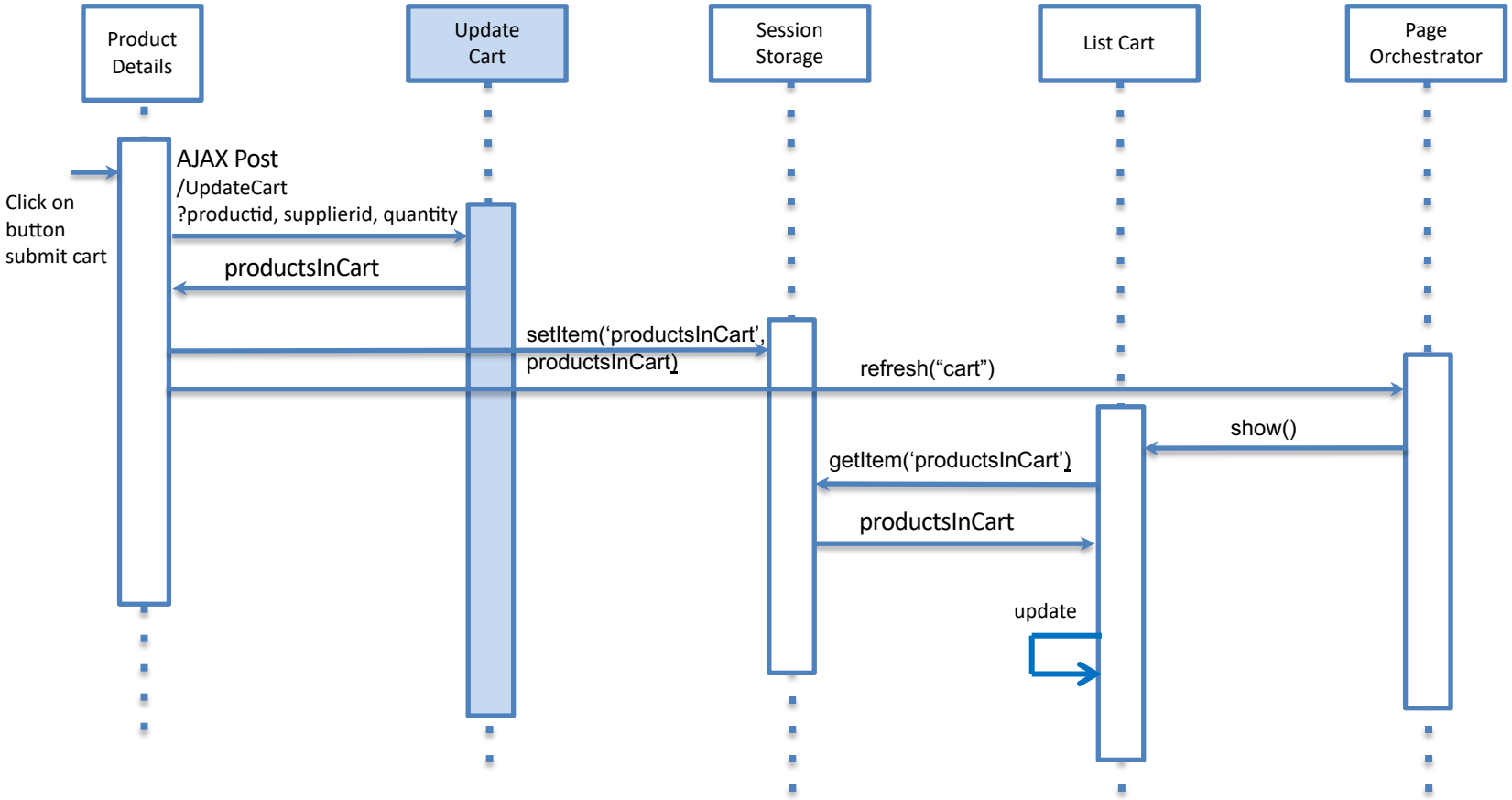
# Evento: selezione di un prodotto



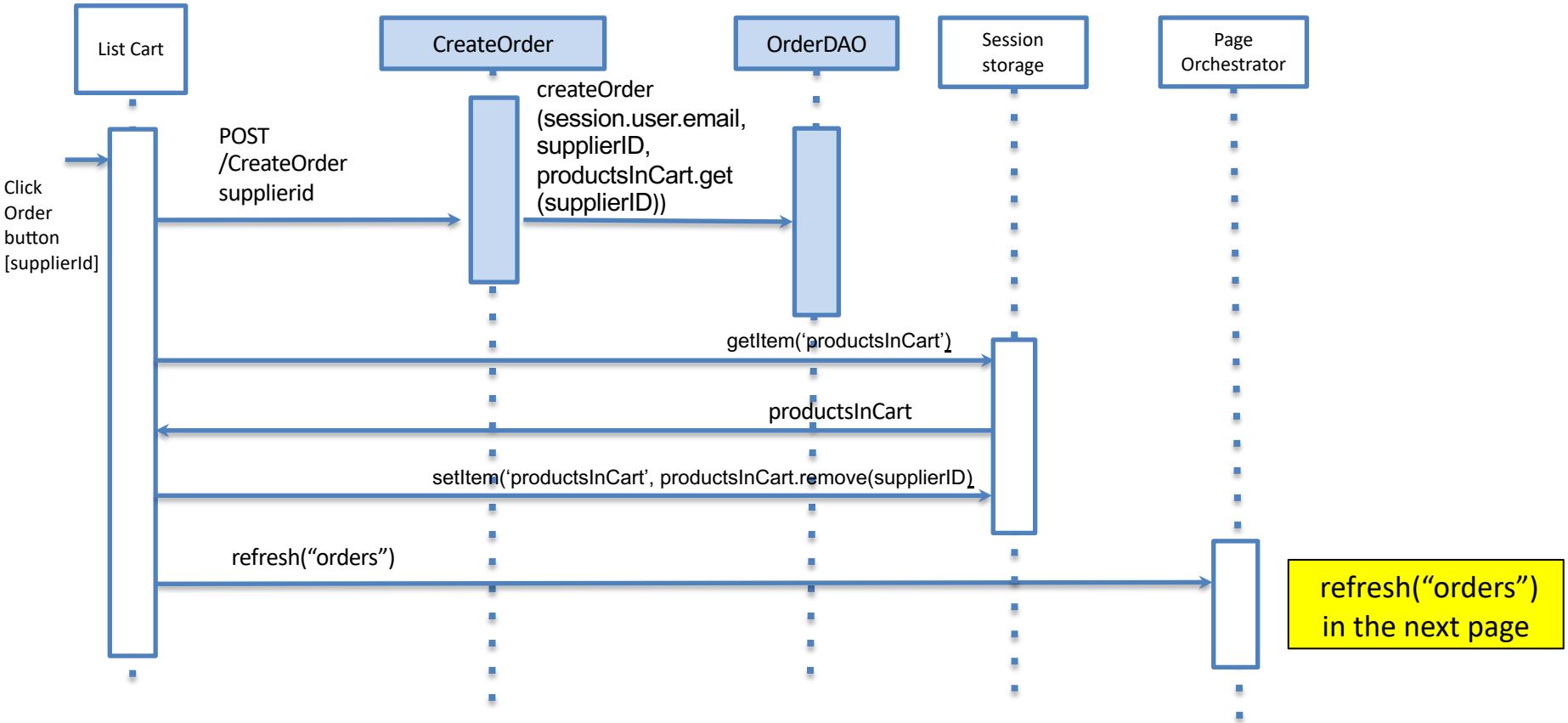
# Evento: mostro finestra sovrapposta



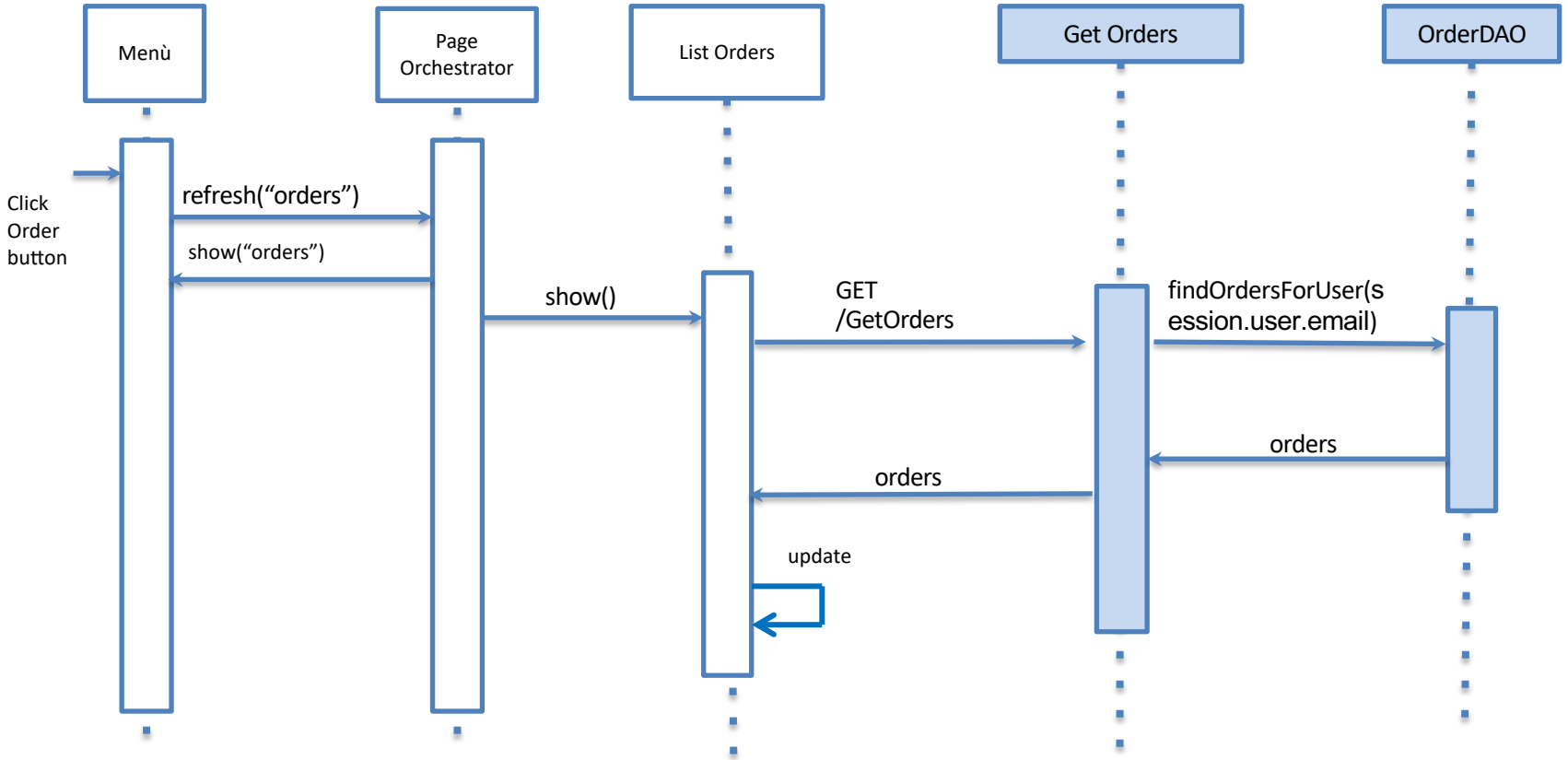
# Evento: inserimento nel carrello



# Event: order creation



# Event: click on menù button



# Evento: logout