Web Technologies

Esercizio 2 – Versione RIA

Gruppo 2

Negrini Samuele – Matr: 866797 – CP: 10539341

Plenzich Aldo - Matr: 868058 – CP: 10538310

Sala Mattia - Matr: 869766 – CP: 10540582

Gestione Preventivi

Un’applicazione web consente la gestione di richieste di preventivi per prodotti personalizzati. Un **preventivo è associato a un prodotto**, al **cliente** che l’ha richiesto e all’**impiegato** che l’ha gestito. **Il preventivo comprende una o più opzioni per il prodotto a cui è associato**, che devono essere tra quelle disponibili per il prodotto. Un **prodotto** ha un **codice**, **un’immagine** e un **nome**. Un’**opzione** ha un **codice**, un **tipo** (“normale”, “in offerta”) e un **nome**. Un **preventivo** ha un **prezzo**, **definito dall’impiegato**. Quando l’**utente** (cliente o impiegato) accede all’applicazione, appare una LOGIN PAGE, mediante la quale l’utente si autentica con **username** e **password**. Quando un cliente fa login, accede a una pagina HOME PAGE CLIENTE che contiene una form per creare un preventivo e l’elenco dei preventivi creati dal cliente. Mediante la form l’utente per prima cosa sceglie il prodotto; scelto il prodotto, la form mostra le opzioni di quel prodotto.

Entities Attributes Relationships

Gestione Preventivi

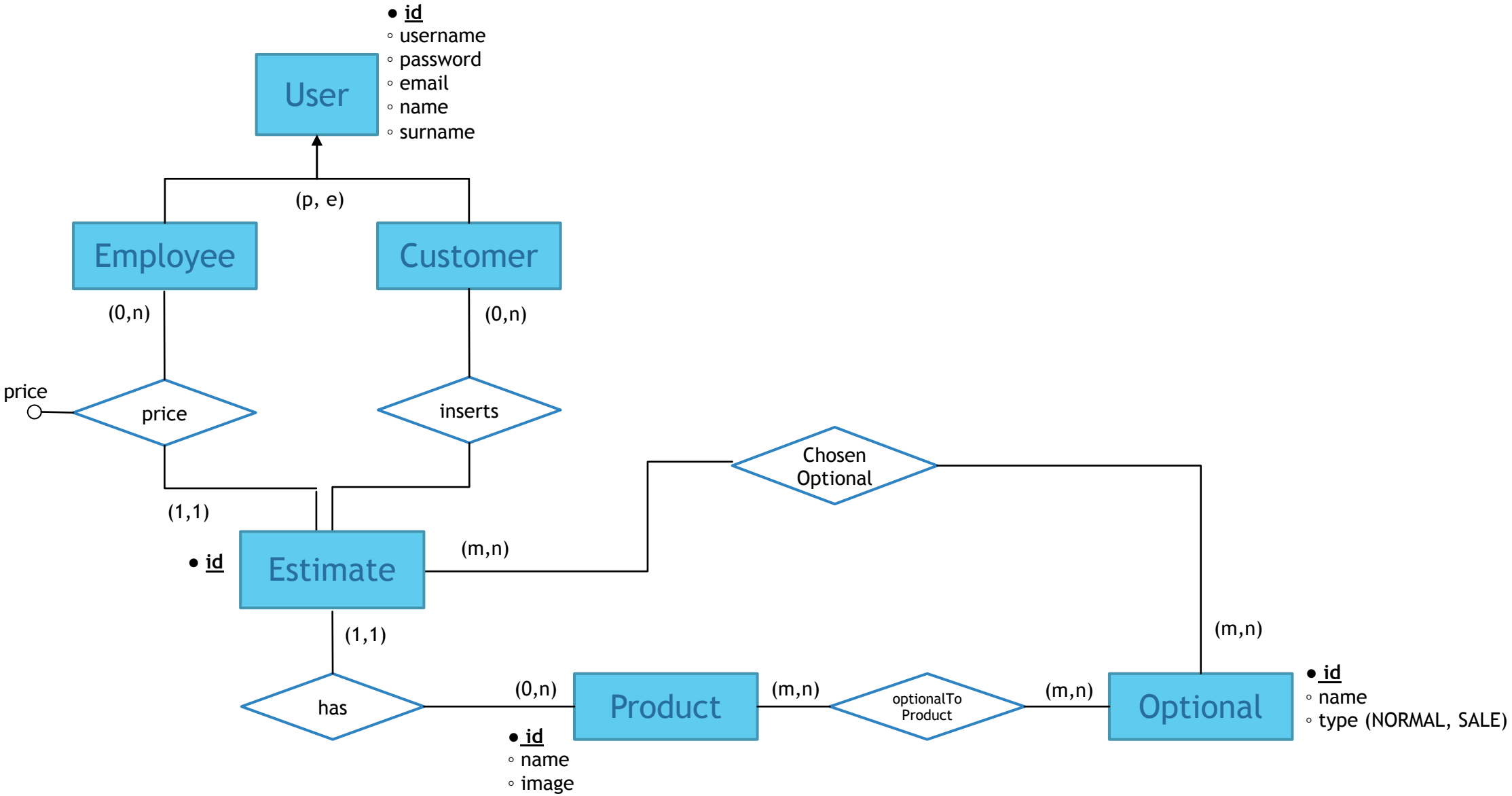
L’utente sceglie le opzioni (almeno una) e conferma l’invio del preventivo mediante il bottone INVIA PREVENTIVO. Quando un impiegato fa login, accede a una pagina HOME PAGE IMPIEGATO che contiene l’elenco dei preventivi gestiti da lui in precedenza e quello dei preventivi non ancora associati a nessun impiegato. Quando l’impiegato seleziona un elemento dall’elenco dei preventivi non ancora associati a nessuno, compare una pagina PREZZA PREVENTIVO che mostra i dati del cliente (username) e del preventivo e una form per inserire il prezzo del preventivo. Quando l’impiegato inserisce il prezzo e invia i dati con il bottone INVIA PREZZO, compare di nuovo la pagina HOME PAGE IMPIEGATO con gli elenchi dei preventivi aggiornati.

Specifiche Aggiuntive

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

- L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password", anche a lato client. La registrazione controlla l'unicità dello username.
- Dopo il login, l'intera applicazione è realizzata con un'unica pagina per ciascuno dei ruoli: una pagina singola per il ruolo di cliente e una pagina singola per il ruolo di impiegato.
- 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.
- Nella pagina del cliente, la scelta del prodotto comporta la successiva visualizzazione delle opzioni senza produrre un'ulteriore chiamata al server. L'invio del preventivo da parte del cliente deve produrre la verifica dei dati anche a lato client (almeno un'opzione scelta).
- Nella pagina dell'impiegato, il controllo del prezzo (non nullo e maggiore di zero) deve essere fatto anche a lato client.
- Eventuali errori a lato server devono essere segnalati mediante un messaggio di allerta all'interno della pagina del cliente o dell'impiegato.

Database design



Local database schema

```
CREATE TABLE IF NOT EXISTS `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `role` enum('customer','employee') NOT NULL DEFAULT 'customer',
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`)
)
```

```
CREATE TABLE IF NOT EXISTS `product` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

```
CREATE TABLE IF NOT EXISTS `optionaltoproduct` (
  `prdid` int(11) NOT NULL,
  `optid` int(11) NOT NULL,
  PRIMARY KEY (`prdid`,`optid`),
  KEY `FK_optionaltoproduct_optional` (`optid`),

 CONSTRAINT `FK_optionaltoproduct_optional`
  FOREIGN KEY (`optid`) REFERENCES `optional`
  (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  CONSTRAINT `FK_optionaltoproduct_product`
  FOREIGN KEY (`prdid`) REFERENCES `product`
  (`id`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

```
CREATE TABLE IF NOT EXISTS `optional` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `type` enum('NORMAL','SALE') NOT NULL DEFAULT 'NORMAL',
  PRIMARY KEY (`id`)
)
```

Local database schema

```
CREATE TABLE IF NOT EXISTS `estimate` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `usrid` int(11) NOT NULL DEFAULT '0',
  `prdid` int(11) NOT NULL,
  `empid` int(11) DEFAULT NULL,
  `price` double DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK__user` (`usrid`),
  KEY `FK_estimate_product` (`prdid`),
  KEY `FK_order_user` (`empid`),

  CONSTRAINT `FK_estimate_product`
  FOREIGN KEY (`prdid`)  REFERENCES `product` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE,

  CONSTRAINT `FK_order_user`
  FOREIGN KEY (`empid`) REFERENCES `user` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE
)
```

```
CREATE TABLE IF NOT EXISTS `chosenoptional` (
  `estid` int(11) NOT NULL,
  `optid` int(11) NOT NULL,
  PRIMARY KEY (`estid`,`optid`),
  KEY `FK_chosenoptional_optionaltoproduct` (`optid`)

CONSTRAINT `FK_chosenoptional_estimate`
  FOREIGN KEY  (`estid`) REFERENCES `estimate` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE,

  CONSTRAINT `FK_chosenoptional_optional`
  FOREIGN KEY (`optid`) REFERENCES `optional` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE
)
```

Gestione Preventivi

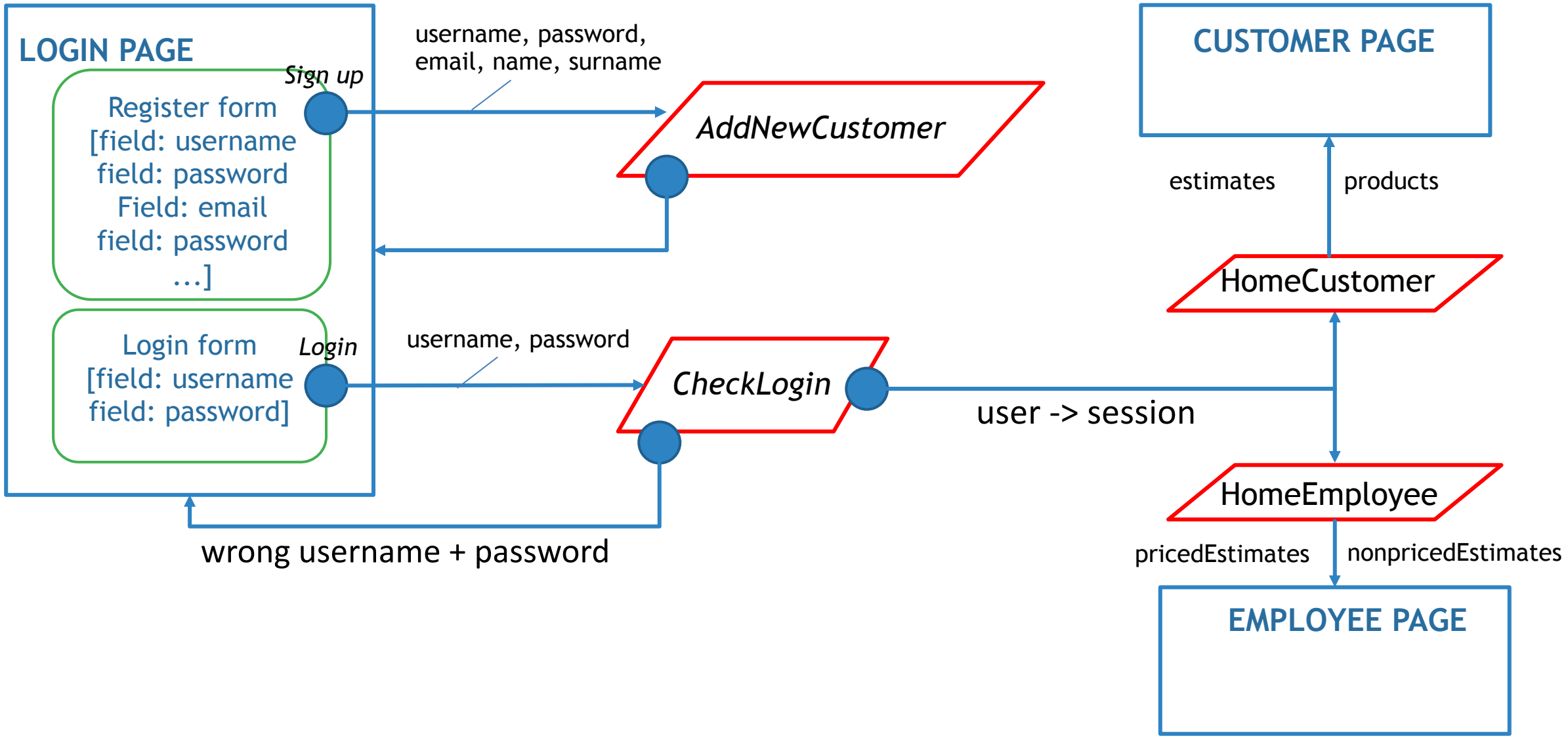
Un'applicazione web consente la gestione di richieste di preventivi per prodotti personalizzati. Un preventivo è associato a un prodotto, al cliente che l'ha richiesto e all'impiegato che l'ha gestito. Il preventivo comprende una o più opzioni per il prodotto a cui è associato, che devono essere tra quelle disponibili per il prodotto. Un prodotto ha un codice, un'immagine e un nome. Un'opzione ha un codice, un tipo ("normale", "in offerta") e un nome. Un preventivo ha un prezzo, definito dall'impiegato. Quando l'utente (cliente o impiegato) accede all'applicazione, appare una LOGIN PAGE, mediante la quale l'utente si autentica con username e password. Quando un cliente fa login, accede a una pagina HOME PAGE CLIENTE che contiene una form per creare un preventivo e l'elenco dei preventivi creati dal cliente. Mediante la form l'utente per prima cosa sceglie il prodotto; scelto il prodotto, la form mostra le opzioni di quel prodotto. L'utente sceglie le opzioni (almeno una) e conferma l'invio del preventivo mediante il bottone INVIA PREVENTIVO. Quando un impiegato fa login, accede a una pagina HOME PAGE IMPIEGATO che contiene l'elenco dei preventivi gestiti da lui in precedenza e quello dei preventivi non ancora associati a nessun impiegato. Quando l'impiegato seleziona un elemento dall'elenco dei preventivi non ancora associati a nessuno, compare una pagina PREZZA PREVENTIVO che mostra i dati del cliente (username) e del preventivo e una form per inserire il prezzo del preventivo. Quando l'impiegato inserisce il prezzo e invia i dati con il bottone INVIA PREZZO, compare di nuovo la pagina HOME PAGE IMPIEGATO con gli elenchi dei preventivi aggiornati.

Pages (View) View Components Events Actions

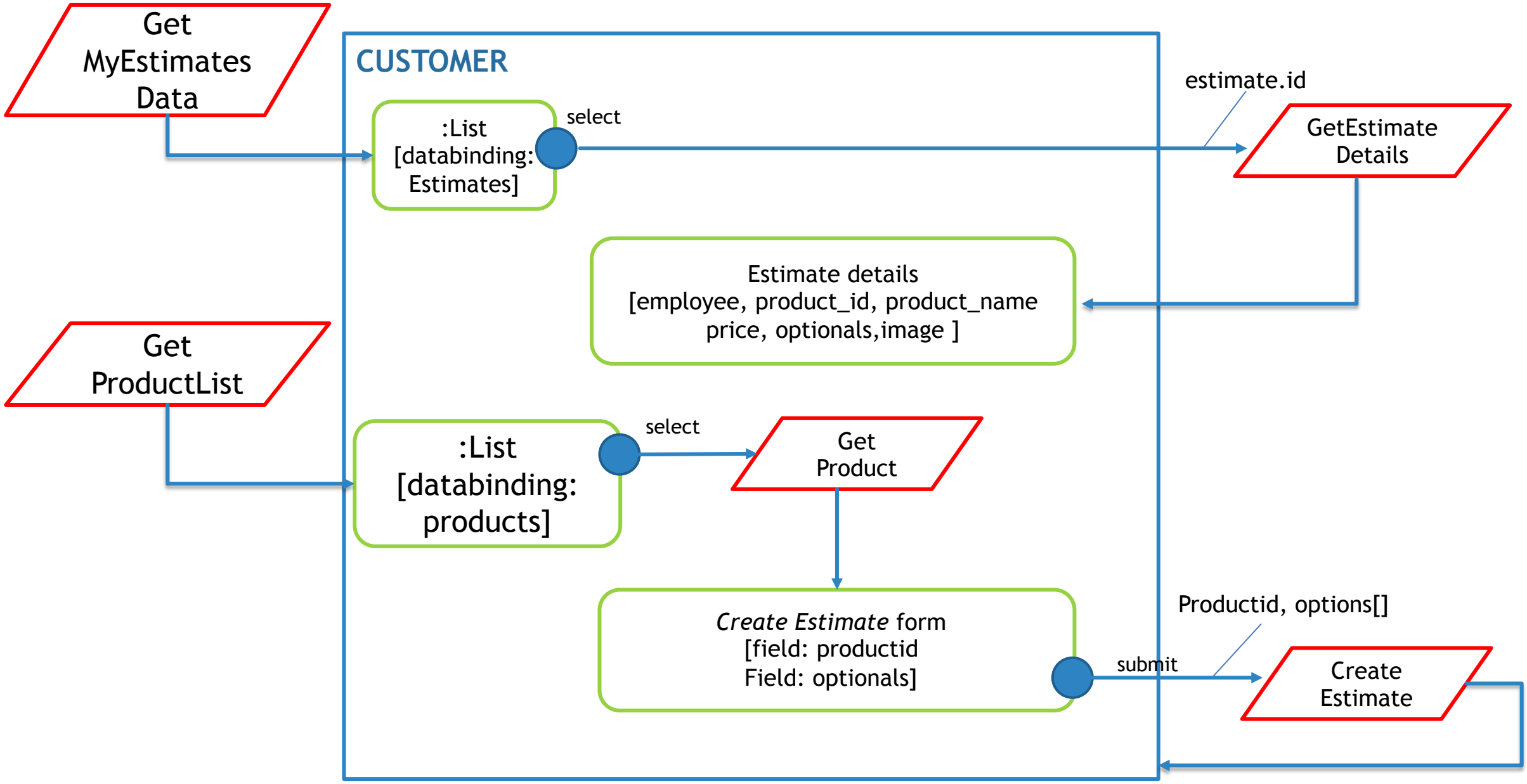
Assunzioni Progettuali:

- Il form di registrazione consente di creare nuovi utenti solo di tipo CLIENTE.
- L'account di tipo IMPIEGATO è riservato a chi lavora per l'azienda.
  La registrazione di un account di tipo IMPIEGATO non è accessibile tramite pagina web pubblica.
- Quando l'utente seleziona un preventivo o un prodotto, vengono mostrati i dettagli.

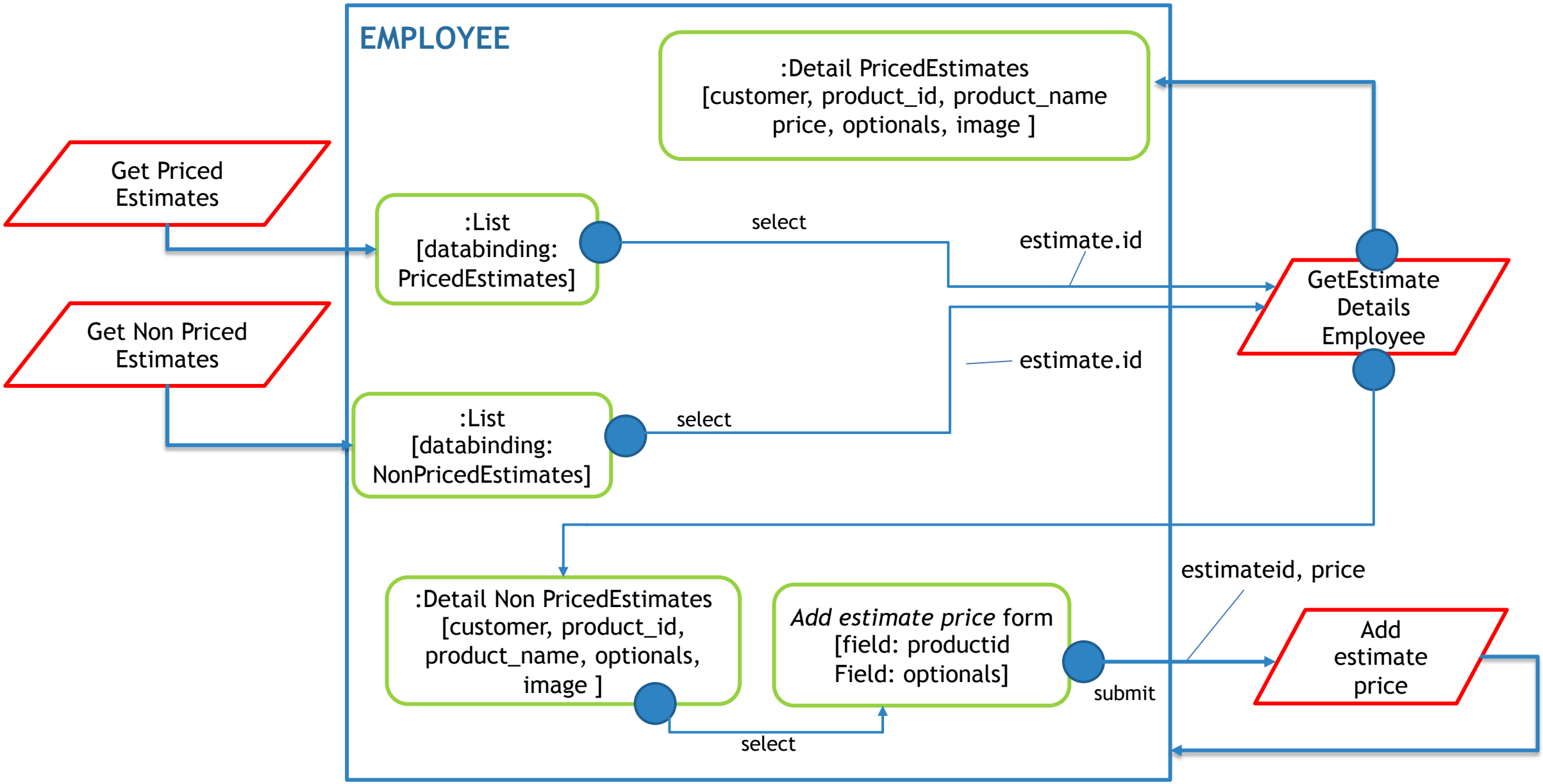
Application design - login



Application design - Customer



Application design - Employee



Components

Model Objects (Beans):

- Estimate
- Optional
- OptionalType
- Product
- User

Data Access Objects (Classes)

- EstimateDAO
- OptionalDAO
- ProductDAO
- UserDAO

Filters

- AuthChecker
- CustomerChecker
- EmployeeChecker

Controllers (Servlets):

- AddEstimate
- AddEstimatePrice
- AddNewCustomer
- CheckLogin
- GetEstimateDetails
- GetEstimateDetailsEmployee
- GetMyEstimatesData
- GetNonPricedEstimates
- GetPricedEstimates
- GetProductList
- Logout

Views (Templates)

- Login
- HomeCustomer
- HomeEmployee

Eventi & azioni

CLIENT SIDE		SERVER SIDE	
EVENTO	**AZIONE**	**EVENTO**	**AZIONE**
Index → login form → submit	Controlla formato dati	POST username, password	Controllo credenziali
HomeCustomer → load	Aggiorna view dei prodotti disponibili	GET (no param value)	Invio lista prodotti
HomeCustomer →load	Aggiorna view dei miei preventivi	GET (no param value)	Invia la lista dei miei preventivi
HomeCustomer→Elenco prodotti → seleziona prodotto	Aggiorna view opzioni prodotto	-	-
HomeCustomer→Prodotto selezionato→insert	Controlla optionals > 0, aggiorna view miei preventivi	POST(product id, optionals[])	Aggiunge preventivo, invia id preventivo aggiunto
HomeCustomer/HomeEmployee →Elenco preventivi → seleziona preventivo	Aggiorna view con dettagli preventivo	GET(estimate id)	Invio informazioni preventivo
HomeEmployee → Preventivi non prezzati → seleziona preventivo	Aggiorna view con dettagli preventivo non prezzato e input	GET(estimate id)	Invio informazioni preventivo
HomeEmployee → preventivo da prezzare → insert	Controllo prezzo non negativo, aggiorno view preventivi prezzati/non prezzati	POST(estimate id, price)	Controllo prezzo non negativo e lo aggiunge al preventivo
HomeEmployee → load	Aggiorna view dei preventivi prezzati	GET(no param value)	Invia lista preventivi prezzati da me
HomeEmployee → load	Aggiorna view dei preventivi non prezzati	GET(no param value)	Invia lista dei preventivi non prezzati
* →logout	Rimuove username da sessionstorage	GET(no param value)	Invalida la sessione

Controller / event handler

CLIENT SIDE		SERVER SIDE	
EVENTO	**CONTROLLORE**	**EVENTO**	**CONTROLLORE**
Index → login form → submit	Function MakeCall	POST username, password	CheckLogin (Servlet)
HomeCustomer → load	Function CustomerEstimateList.show	GET (no param value)	GetMyEstimateData (Servlet)
HomeCustomer →load	Function productlist.show	GET (no param value)	GetProductList (Servlet)
HomeCustomer→Elenco prodotti → seleziona prodotto	ProductDetails.show	-	-
HomeCustomer→Prodotto selezionato→insert	Function MakeCall	POST(product id, optionals[])	AddEstimate (Servlet)
HomeCustomer→Elenco preventivi → seleziona preventivo	Function EstimateDetails.show	GET(estimate id)	GetEstimateDetails (Servlet)
HomeEmployee → Preventivi non prezzati → seleziona preventivo	Function NonPricedEstimateDetails.show	GET(estimate id)	GetEstimateDetailsEmployee (Servlet)
HomeEmployee → i miei preventivi→ seleziona preventivo	Function PricedEstimateDetails.show	GET(estimate id)	GetEstimateDetailsEmployee (Servlet)
HomeEmployee → preventivo da prezzare → insert	Function MakeCall	POST(estimate id, price)	AddEstimatePrice (Servlet)
HomeEmployee → load	PricedEstimateList.show	GET(no param value)	GetPricedEstimates (Servlet)
HomeEmployee → load	NonPricedEstimateList.show	GET(no param value)	GetNonPricedEstimatesList (Servlet)
* →logout	-	Get(no param value)	Logout (Servlet)

Server side: DAO

EstimateDAO

- **createEstimate**(int customerId, int productId, String[] optionalsId)
- **addEstimatePrice**(int employeeId, int estimateId, float price)
- **findEstimateByCustomer**(int customerId)
- **findEstimatesByEmployee**(int employee)
- **findNonPricedEstimates()**
- **findEstimateById**(int estimateId)
- **changeEstimatePrice**(int estimateId, int employeeId, float price)
- **findEstimateByIdAndCustomer**(int estimateId, int customerId)

OptionalDAO

- **findAvailableOptionalsByProduct**(int productId)
- **findChosenOptionalsByEstimate**(int estimateId)

ProductDAO

- **findProducts**()
- **findProductsById**(int productId)
- **findPricedProductByEmployee**(int employeeId)
- **findNonPricedProducts**()
- **findDefaultProduct**()
- **findProductByEstimate**(int estimateId)

UserDAO

- **checkCredentials**(String usr, String psw)
- **findUserById**(int userId)
- **findCustomerByEstimate**(int estimateId)
- **findEmployeeByEstimate**(int estimateId)
- **userExists**(String username, String email)
- **addUser**(String username, String email, String password, String name, String surname)

Server side: Model Objects (BEAN)

Estimate

Optional

Optional Type

Product

User

Client side: View & View Component

Login

- **Login Form**

  -Effattua submit per il login di un utente (impiegato/cliente) già registrato.

- **Registration Form**

  -Effettua la submit dei dati di registrazione di un nuovo cliente.

Home Customer

- **CustomerEstimateList**

`-show():` richiede al server i dati dell'elenco dei preventivi del cliente.
`-update():` aggiorna la lista con i dati ricevuti dal server.
`-reset():` imposta le condizioni di iniziali visibilità dei vari sotto-componenti.
`-autoclick()`: focus sul preventivo in cima alla lista, evidenziandolo e mostrandone i dettagli nella Estimate Details.

- **Estimate Details**

`-show():` richiede al server dettagli del preventivo cliccato.
`-update():` aggiorna la tabella e l'immagine che mostrano i dettagli del preventivo.
`-reset():` imposta le condizioni di iniziali visibilità.

- **Product List**

`-show():` richiede al server la lista dei prodotti.
`-update():` aggiorna i dati e ascolta l'evento sul click del prodotto.
`-reset():` imposta le condizioni iniziali di visibilità.
`-autoclick():` focus sul primo prodotto della lista ricevuta al caricamento della pagina.

- **Product Details**

`-registerevents():` invio al server i dati riguardanti prodotto e opzioni selezionate.
`-show():` richiama update().
`-update():` aggiorna le opzioni relative a un determinato prodotto selezionato.

Client side: View & View Component

Home Employee

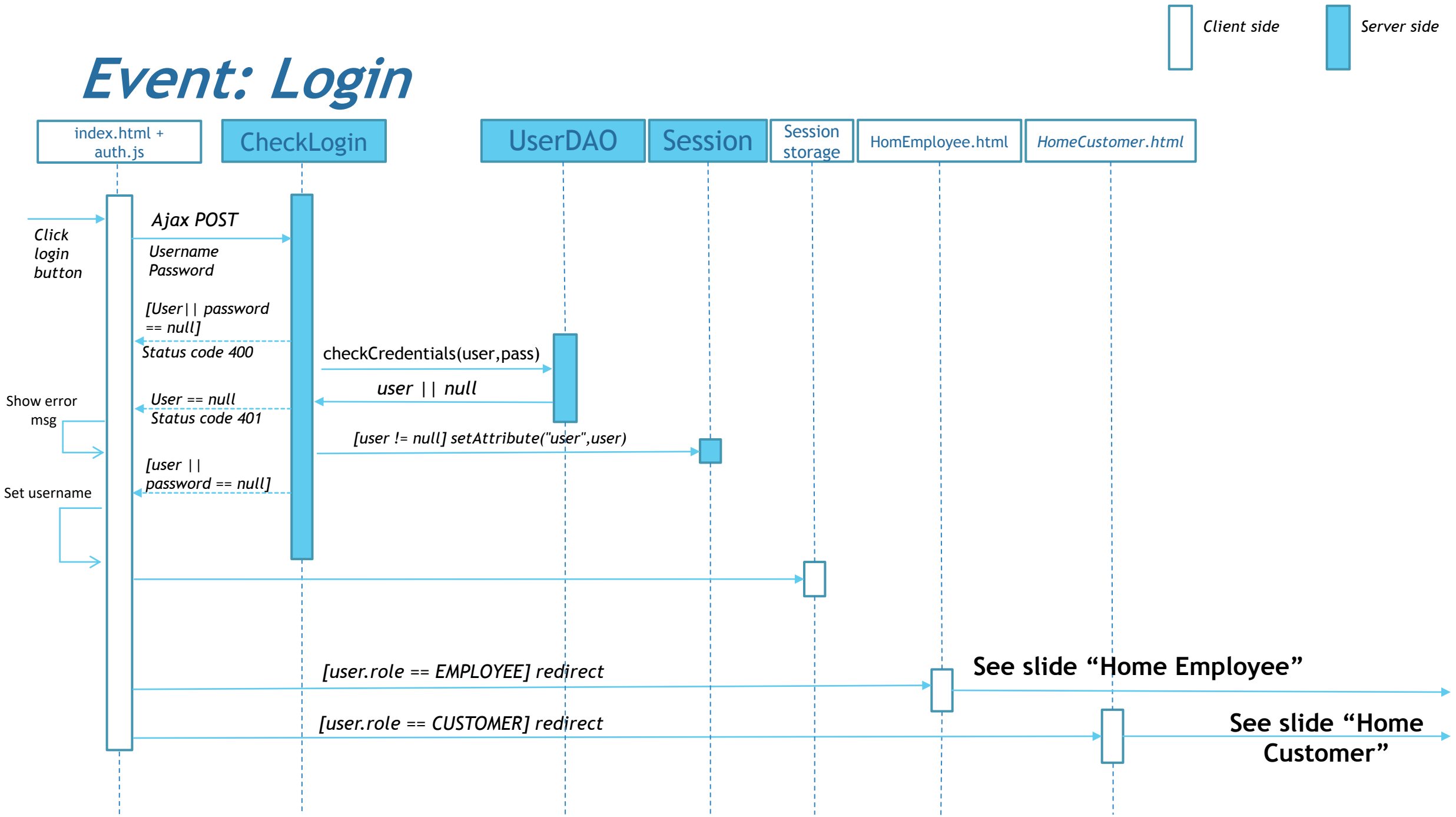
- **PricedEstimatesList**
  - `reset()`: nasconde il componente contenitore della lista dei preventivi prezzati dall'impiegato corrente.
  - `show()`: richiede al server la lista dei preventivi prezzati dall'impiegato corrente.
  - `update()`: aggiorna la lista inserendo i preventivi inviati dal server.
  - `autoclick()`: focus sul primo preventivo prezzato della lista ricevuta al caricamento della pagina.

- **EstimateDetails**
  - `show()`: richiede al server i dettagli del preventivo prezzato selezionato.
  - `reset()`: re-imposta i dettagli del preventivo di default.
  - `update()`: inserisce e mostra i dettagli del preventivo selezionato.

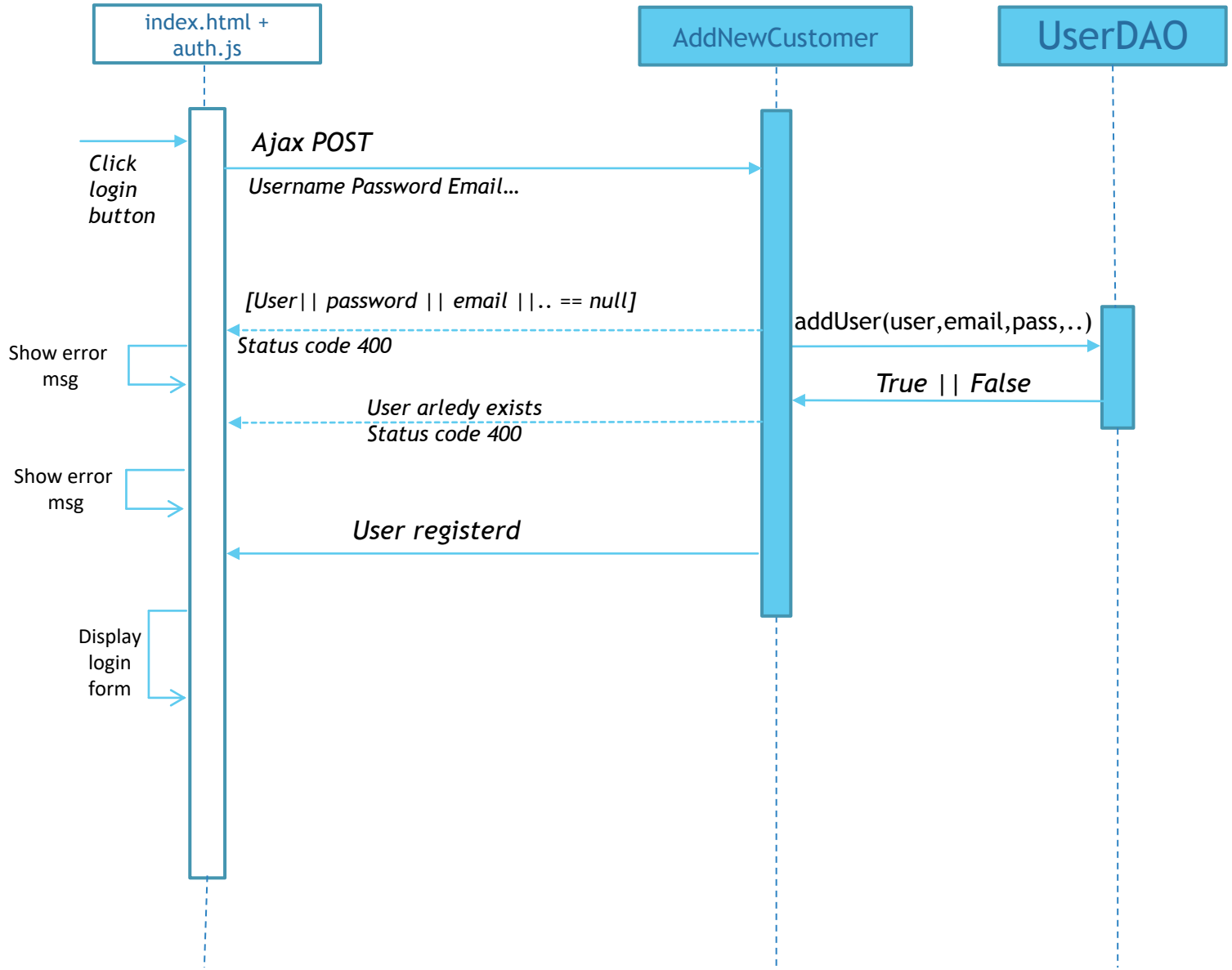
- **NonPricedEstimatesList**
  - `reset()`: nasconde il componente contenitore della lista dei preventivi non ancora prezzati.
  - `show()`: richiede al server la lista dei preventivi non ancora prezzati.
  - `update()`: aggiorna la lista inserendo i preventivi non prezzati, inviati dal server.
  - `autoclick()`: focus sul primo preventivo non prezzato della lista ricevuta al caricamento della pagina.

- **NonPricedEstimateDetails**
  - `registerEvents()`: effettua la POST al server del prezzo inserito dall'impiegato.
  - `reset()`: resetta il form per l'inserimento del prezzo alla condizioni iniziali.
  - `show()`: richiede al server i dettagli del preventivo correntemente selezionato.
  - `update()`: aggiorna il contenuto dei dettagli del preventivo selezionato.

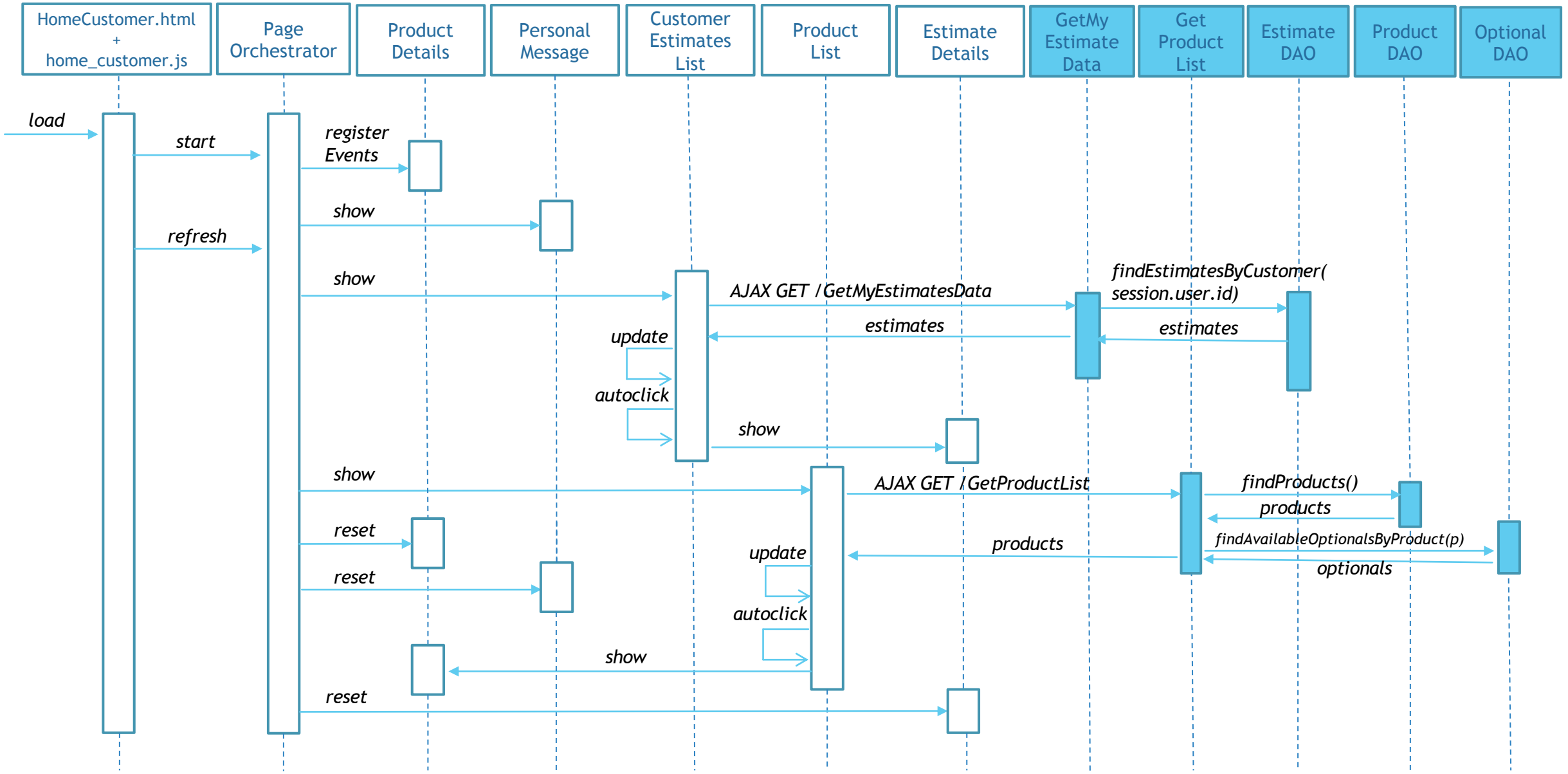
Event: Login


Client side
Server side
index.html + auth.js
CheckLogin
UserDAO
Session
Session storage
HomEmployee.html
HomeCustomer.html
Click login button
Ajax POST
Username
Password
[User|| password == null]
Status code 400
checkCredentials(user,pass)
user || null
Show error msg
User == null
Status code 401
[user != null] setAttribute("user",user)
[user || password == null]
Set username
[user.role == EMPLOYEE] redirect
See slide “Home Employee”
[user.role == CUSTOMER] redirect
See slide “Home Customer”

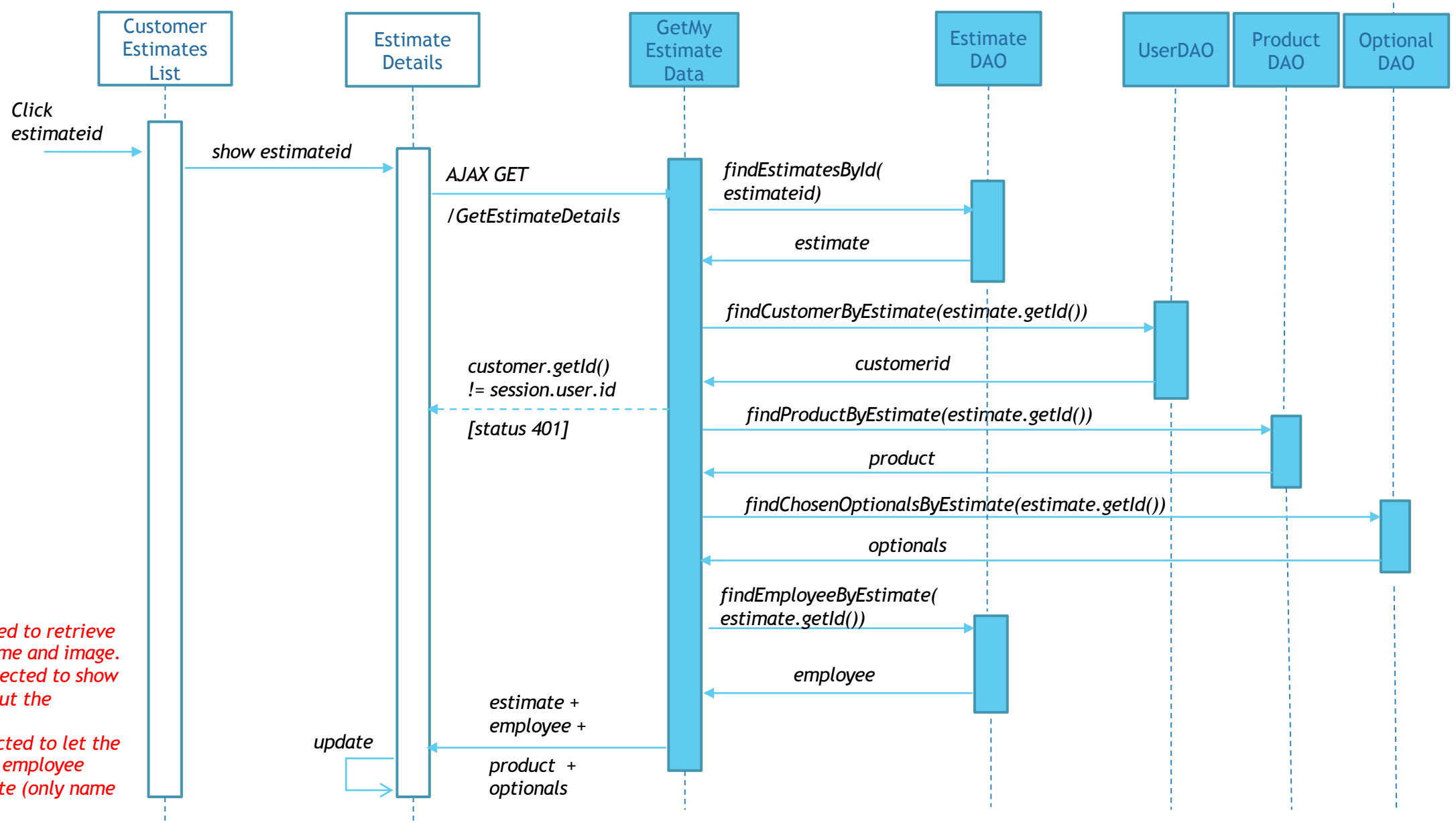
Event: Register



Event: Home Customer



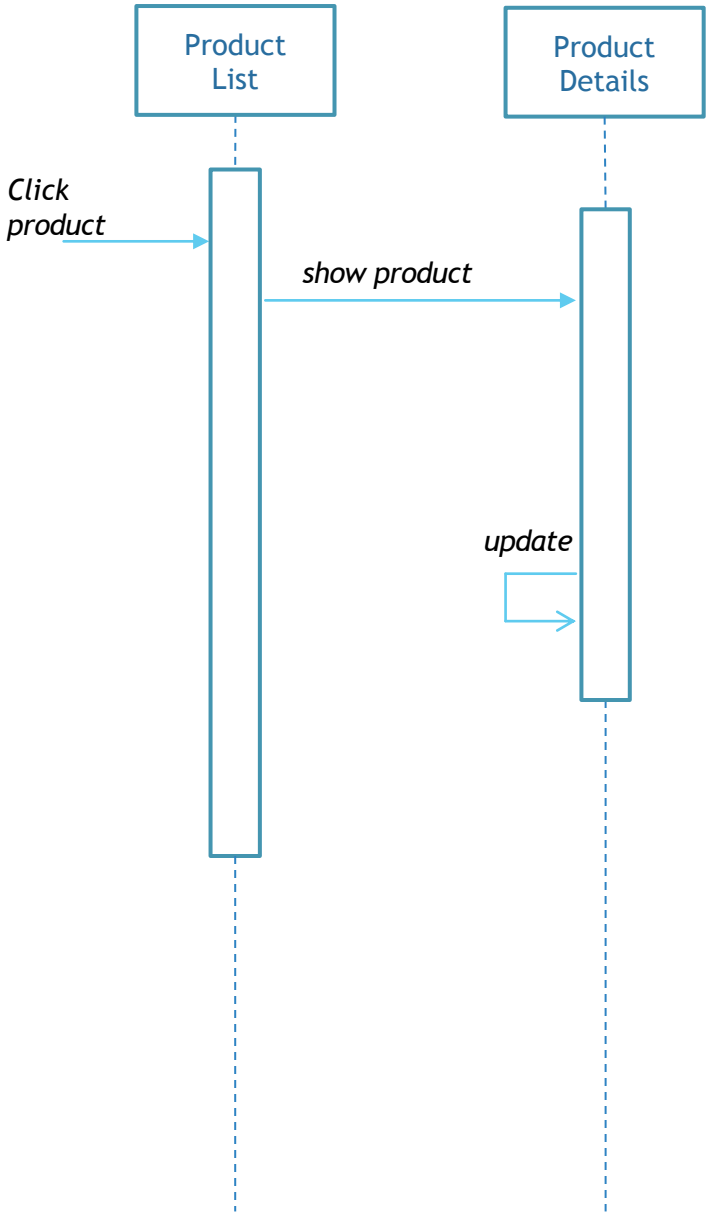
Event: Get Estimate Details [from Home Customer]



NOTE

- Product is selected to retrieve the product's name and image.
- Optionals are selected to show more details about the estimate.
- Employee is selected to let the user know which employee managed his quote (only name & surname).

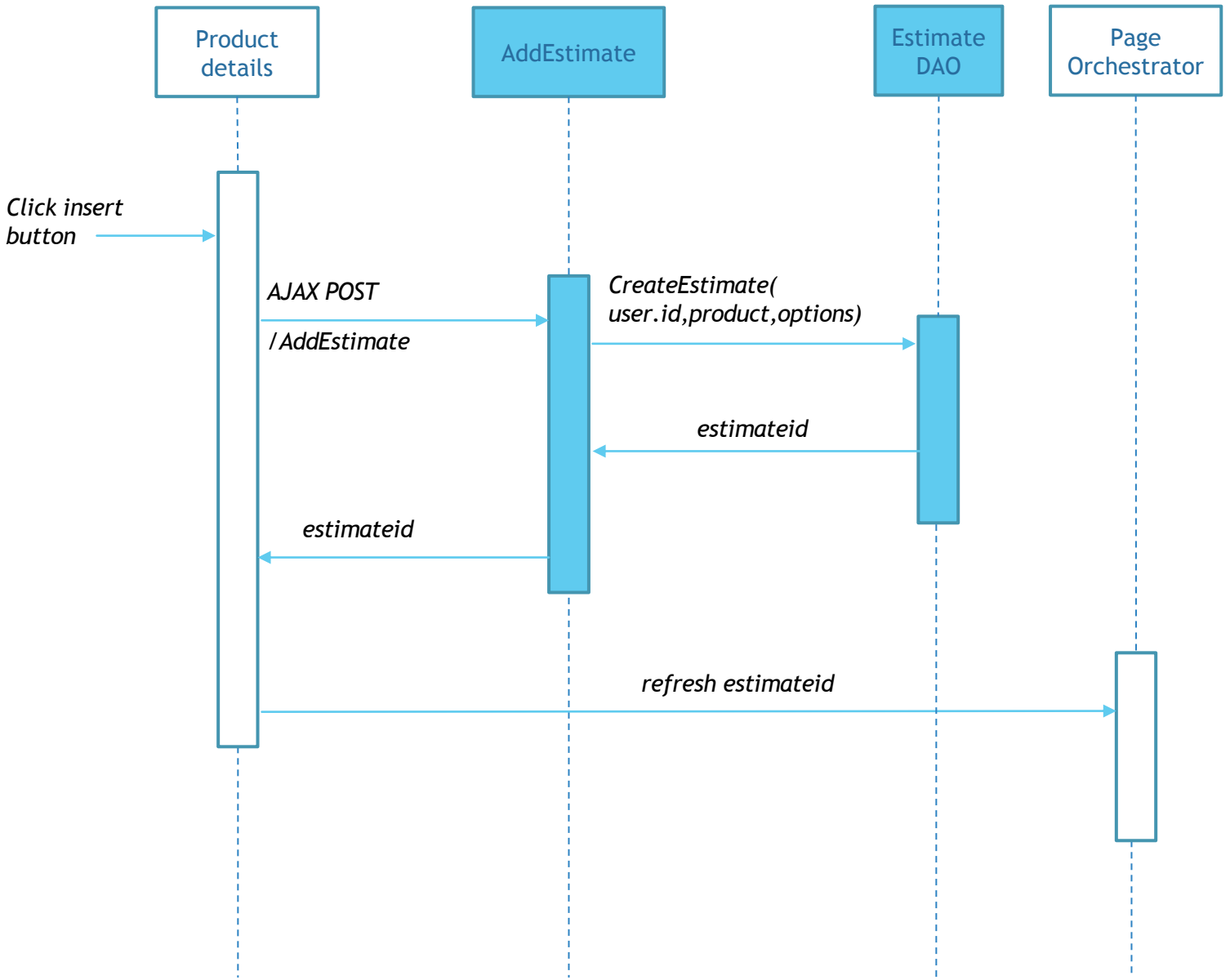
Event: Show Product Details [from Home Customer]



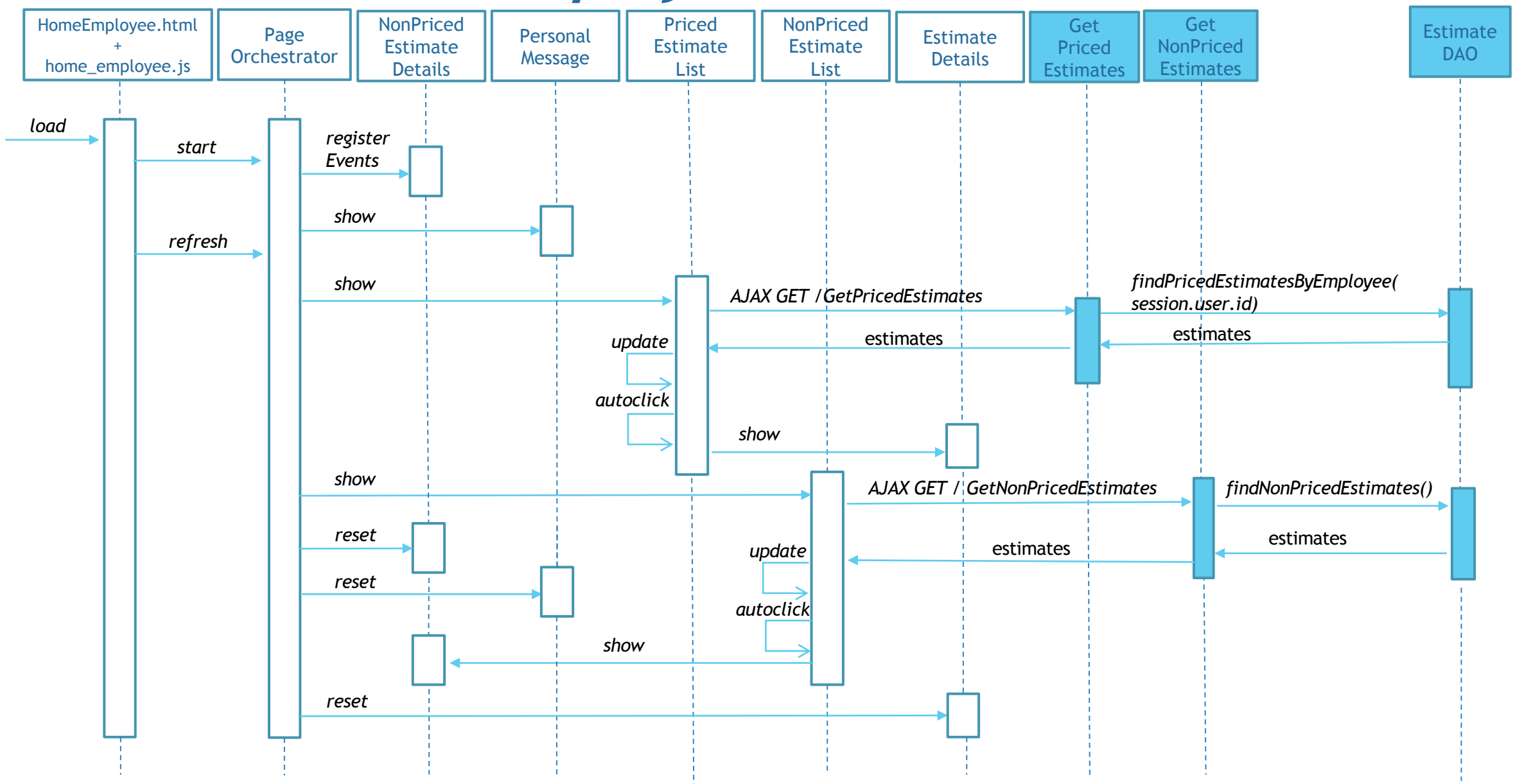
NOTE

Optionals are shown without making a new request to the server. Their data is already contained inside the product itself.

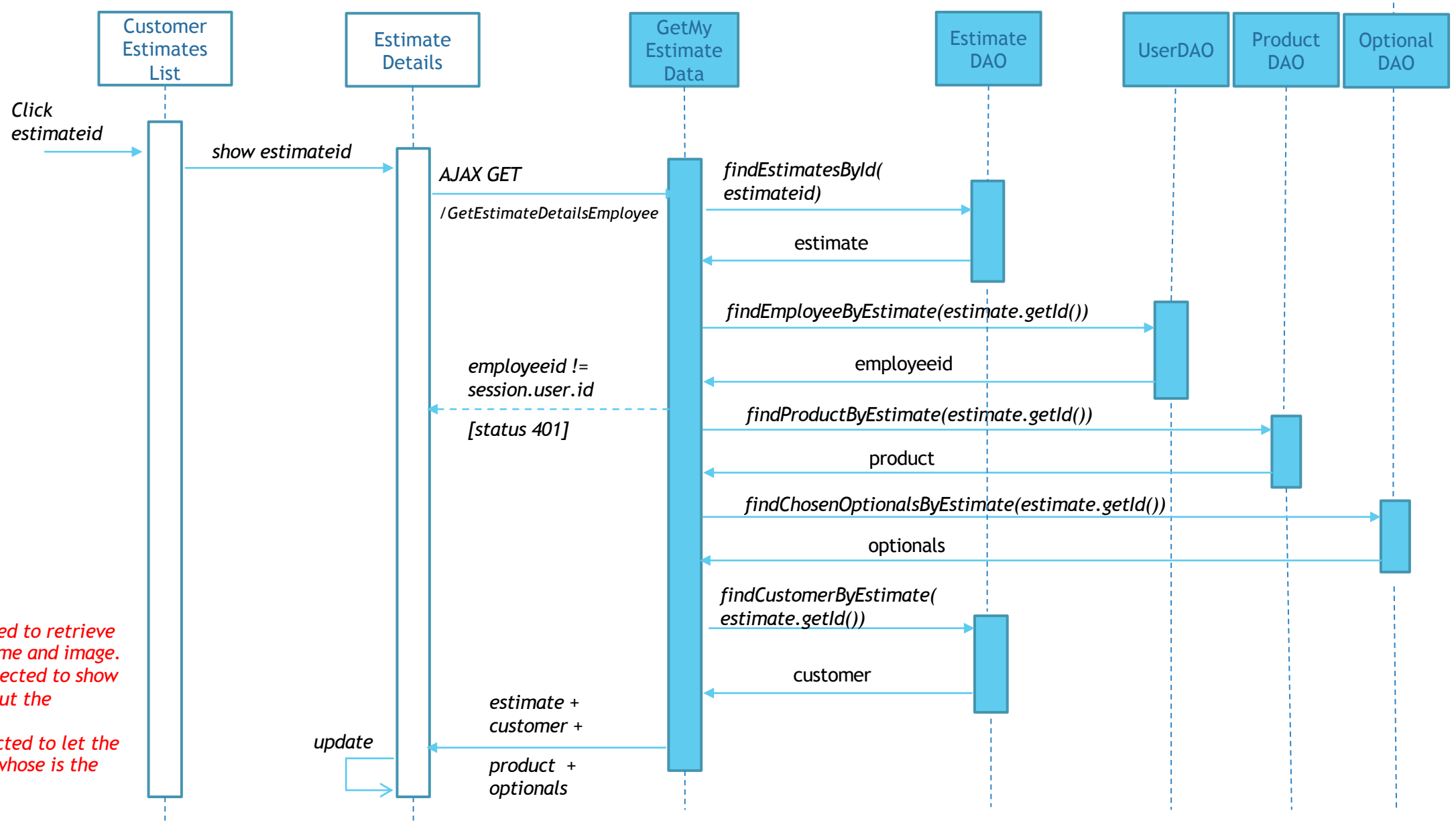
Event: Add new estimate [from Home Customer]



Event: Home Employee



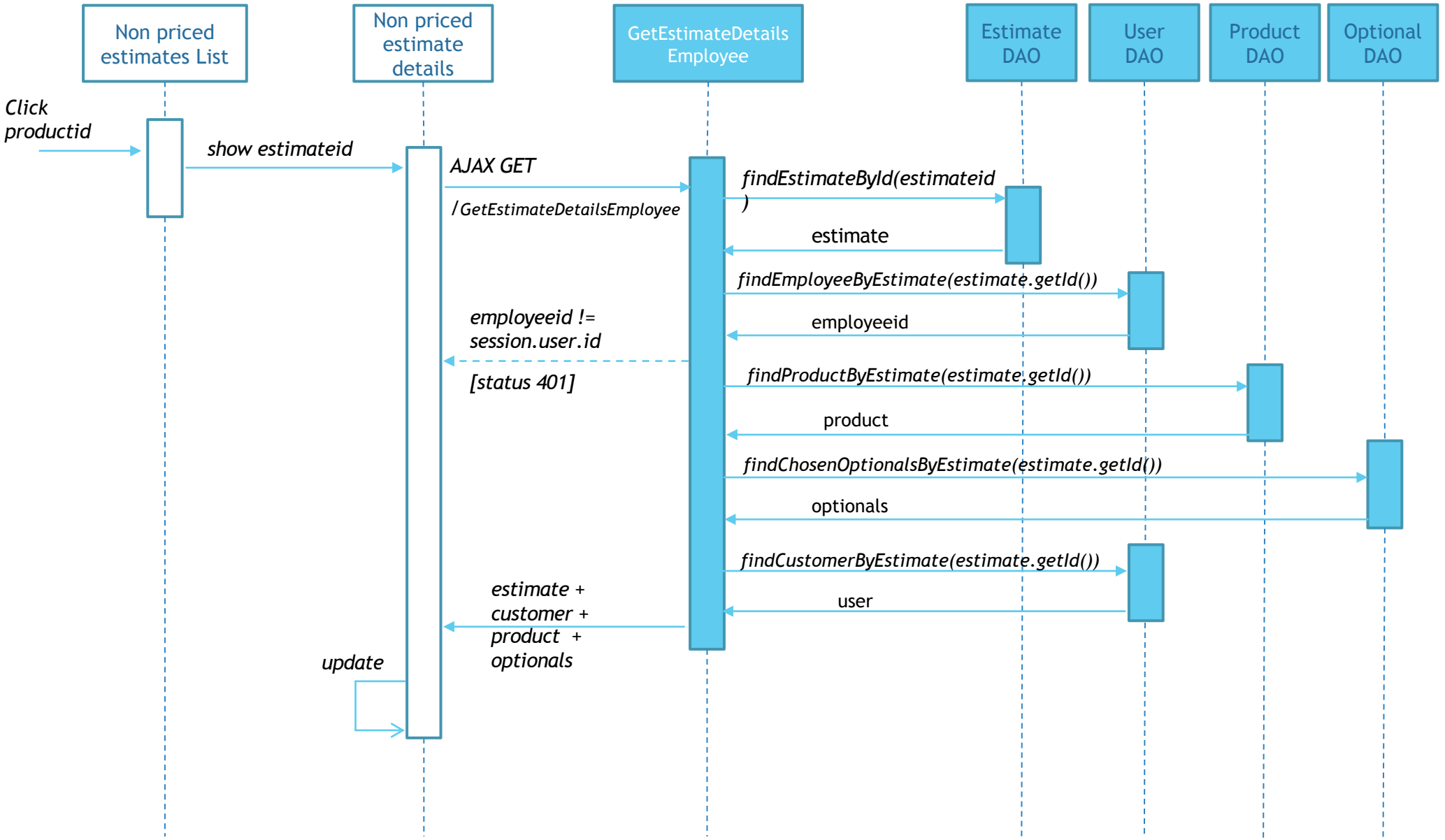
Event: Get Priced Estimate Details [from Home Employee]



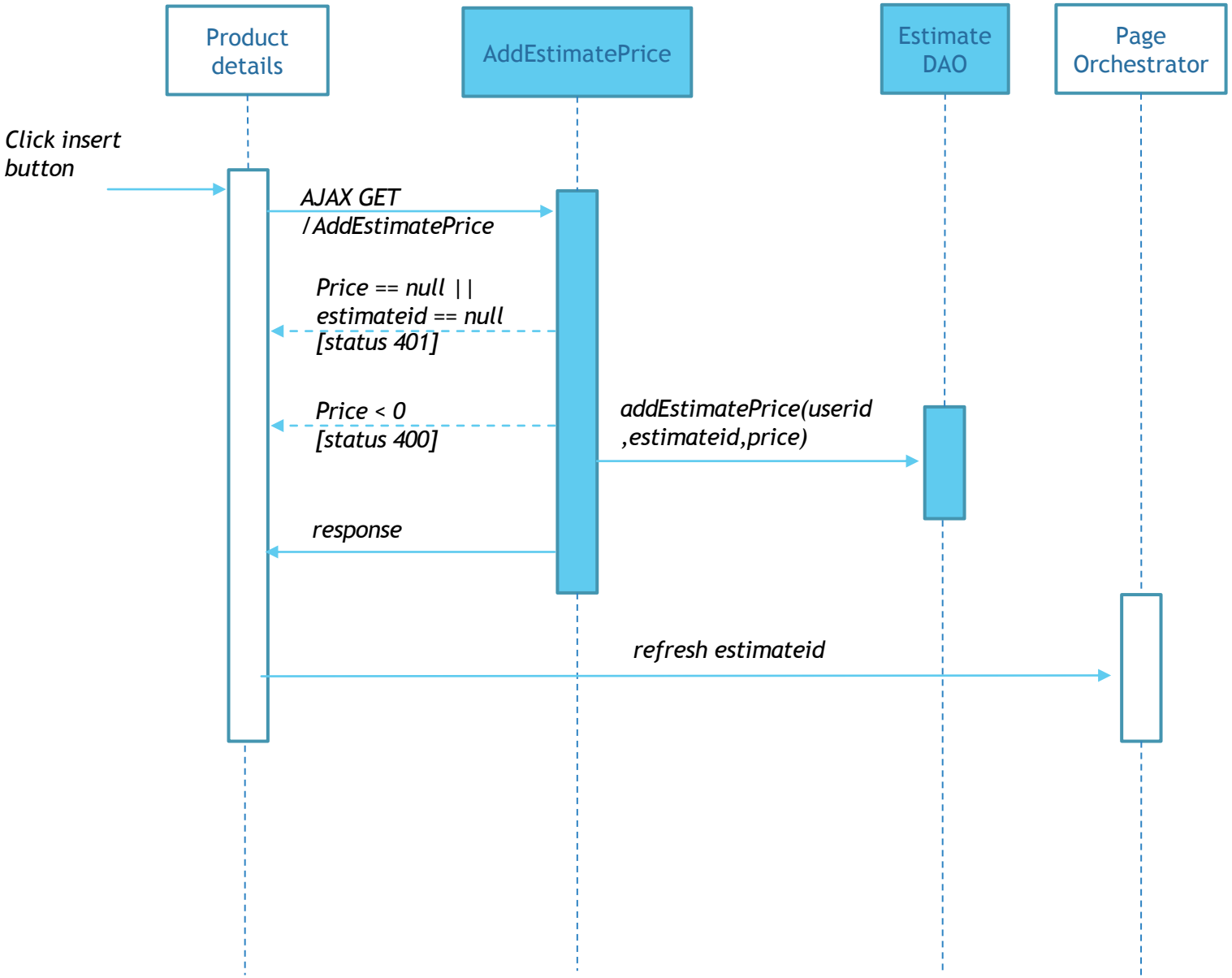
NOTE

- *Product is selected to retrieve the product's name and image.*
- *Optionals are selected to show more details about the estimate.*
- *Customer is selected to let the employee know whose is the estimate for.*

Event: Get Non Priced Estimate Details [from Home Employee]



Event: Add estimate price [from Home Employee]



Event: Logout