Customizable store: price quotes management

Exam exercise - Web Technologies

Wu Yingying - 23/06/2020

Specifications

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.

Specifications

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.

Specifications

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.

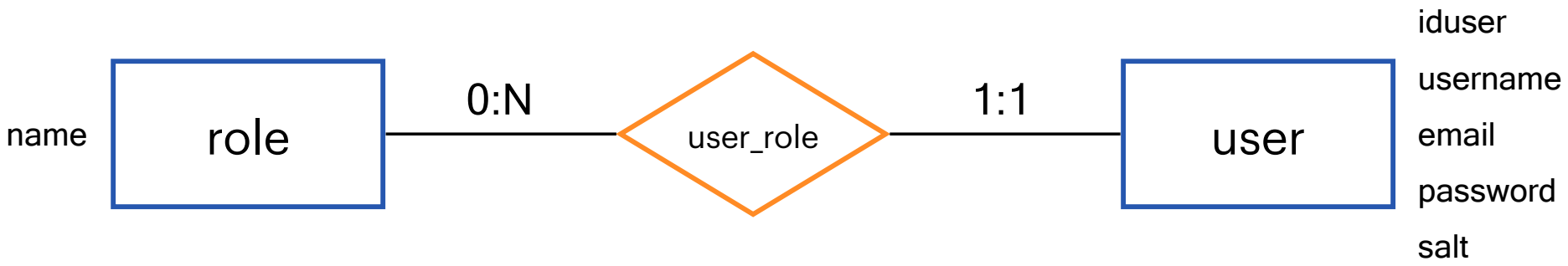
Data requirements analysis

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.

Entities, attributes, relationships

Database design

```
CREATE SCHEMA `customizable_store` DEFAULT CHARACTER SET utf8 ;
```

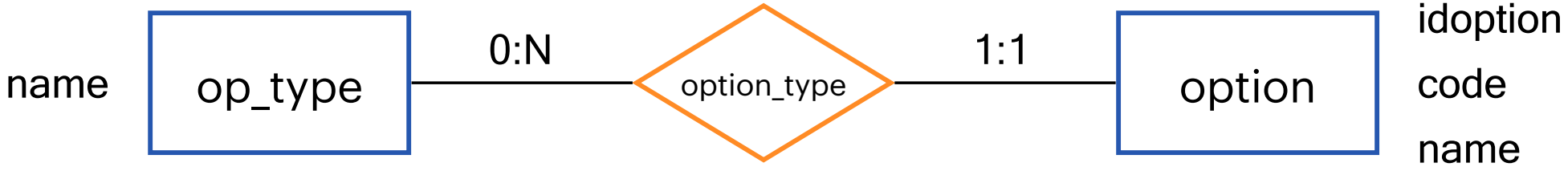



```
CREATE TABLE `customizable_store`.`role`(
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`name`)
);
```

`role`.`name`
“CLIENT”
“EMPLOYEE”

```
CREATE TABLE `customizable_store`.`user` (
  `iduser` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL UNIQUE,
  `email` VARCHAR(255) NULL,
  `password` BINARY(128) NOT NULL,
  `salt` BINARY(8) NOT NULL,
  `role` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`iduser`),
  CONSTRAINT `user_role`
    FOREIGN KEY (`role`)
    REFERENCES `customizable_store`.`role` (`name`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

Database design




```
CREATE TABLE `customizable_store`.`op_type`(
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`name`)
);
```

`op_type`.`name`
“NORMAL”
“ON_SALE”

```
CREATE TABLE `customizable_store`.`option` (
  `idoption` INT NOT NULL AUTO_INCREMENT,
  `code` VARCHAR(45) NOT NULL UNIQUE,
  `name` VARCHAR(255) NOT NULL,
  `type` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idoption`),
  CONSTRAINT `option_type`
    FOREIGN KEY (`type`)
    REFERENCES `customizable_store`.`op_type` (`name`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

Database design

product

idproduct

code

image

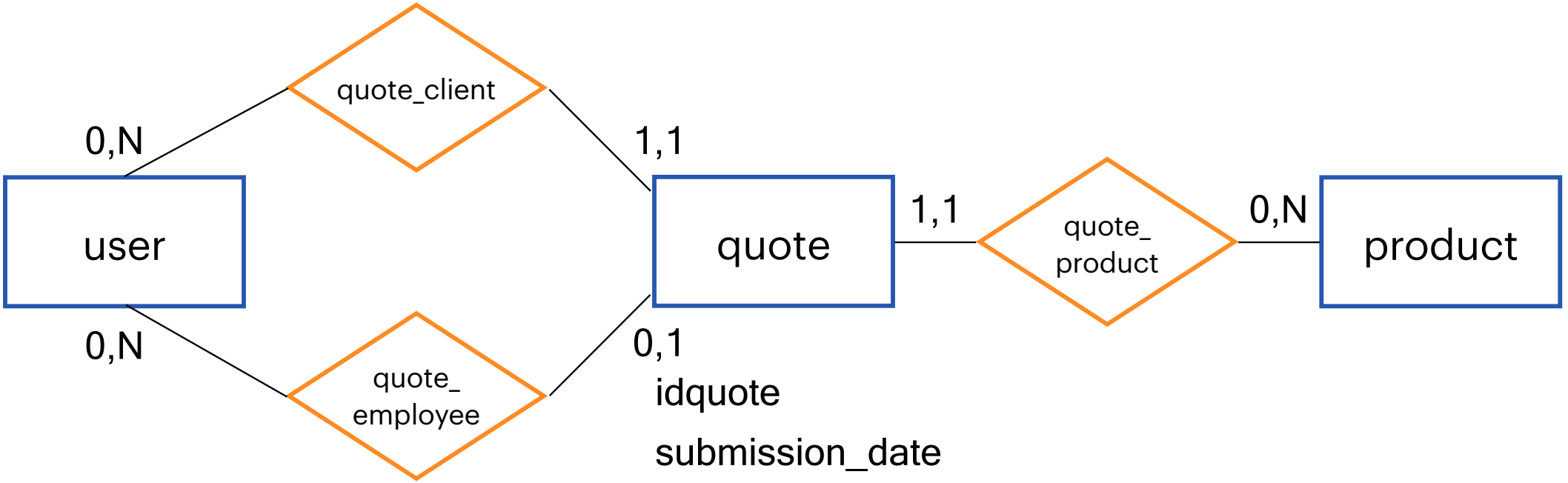
name

```
CREATE TABLE
`customizable_store`.`product`(
  `idproduct` INT NOT NULL AUTO_INCREMENT,
  `code` VARCHAR(45) NOT NULL,
  `image` MEDIUMBLOB NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`idproduct`)
);
```

Database design


```
CREATE TABLE `customizable_store`.`product_option` (
  `product_id` INT NOT NULL,
  `option_id` INT NOT NULL,
  PRIMARY KEY (`product_id`, `option_id`),
  CONSTRAINT `prod_prod_option`
    FOREIGN KEY (`product_id`)
    REFERENCES `customizable_store`.`product` (`idproduct`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `opt_prod_option`
    FOREIGN KEY (`option_id`)
    REFERENCES `customizable_store`.`option` (`idoption`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Database design



Database design

```
CREATE TABLE
`customizable_store`.`quote` (
  `idquote` INT NOT NULL,
  `submission_date` DATETIME NOT NULL,
  `client` INT NOT NULL,
  `employee` INT NULL,
  `product` INT NOT NULL,
  `price` DECIMAL(10,2) UNSIGNED NULL,
  PRIMARY KEY (`idquote`),
  CONSTRAINT `quote_client`
    FOREIGN KEY (`client`)
    REFERENCES
`customizable_store`.`user` (`iduser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `quote_employee`
    FOREIGN KEY (`employee`)
    REFERENCES
`customizable_store`.`user` (`iduser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `quote_product`
    FOREIGN KEY (`product`)
    REFERENCES
`customizable_store`.`product`
(`idproduct`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Database design


```
CREATE TABLE `customizable_store`.`quote_option` (
  `quote_id` INT NOT NULL,
  `option_id` INT NOT NULL,
  PRIMARY KEY (`quote_id`, `option_id`),
  CONSTRAINT `quote_qt_opt`
    FOREIGN KEY (`quote_id`)
    REFERENCES `customizable_store`.`quote` (`idquote`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `option_qt_opt`
    FOREIGN KEY (`option_id`)
    REFERENCES `customizable_store`.`option` (`idoption`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Database design

Trigger
before insert
on quote_option

```
DELIMITER //
CREATE TRIGGER validate_option
BEFORE INSERT ON customizable_store.quote_option
FOR EACH ROW
BEGIN
  DECLARE prod_id INT;
  SELECT q.product INTO prod_id
  FROM customizable_store.quote q
  WHERE q.idquote = NEW.quote_id;
  IF (prod_id, NEW.option_id) NOT IN (
    SELECT *
    FROM customizable_store.product_option po
    WHERE (po.product_id = prod_id)
  ) THEN
  SIGNAL SQLSTATE '45000'
  SET MESSAGE_TEXT = 'Invalid option for the product.';
  END IF;
END //
DELIMITER ;
```

Application requirements analysis

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.

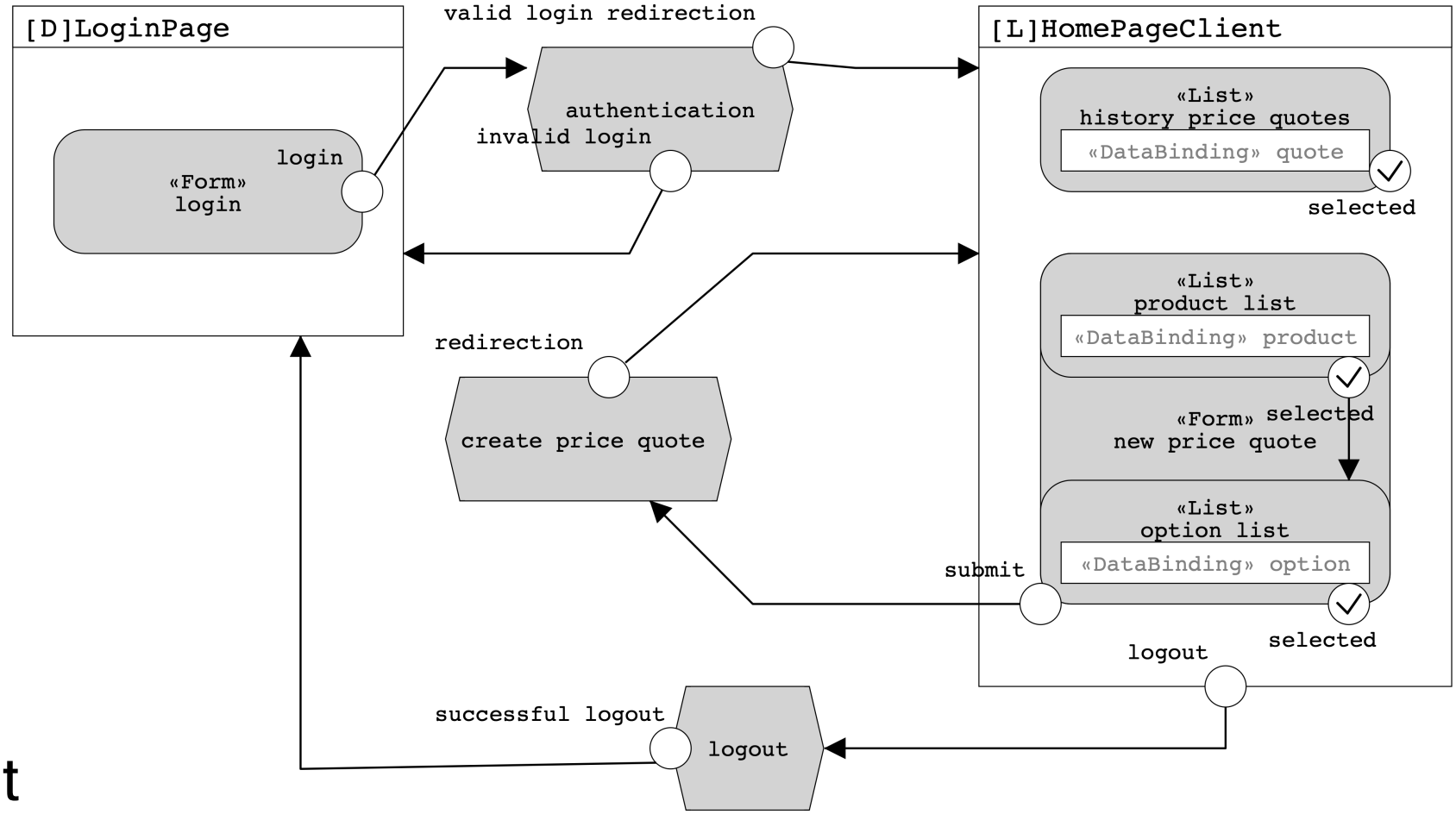
Pages (view), view components, events, actions

Application requirements analysis

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 [inserimento prezzo], compare di nuovo la pagina HOME PAGE IMPIEGATO con gli elenchi dei preventivi aggiornati.

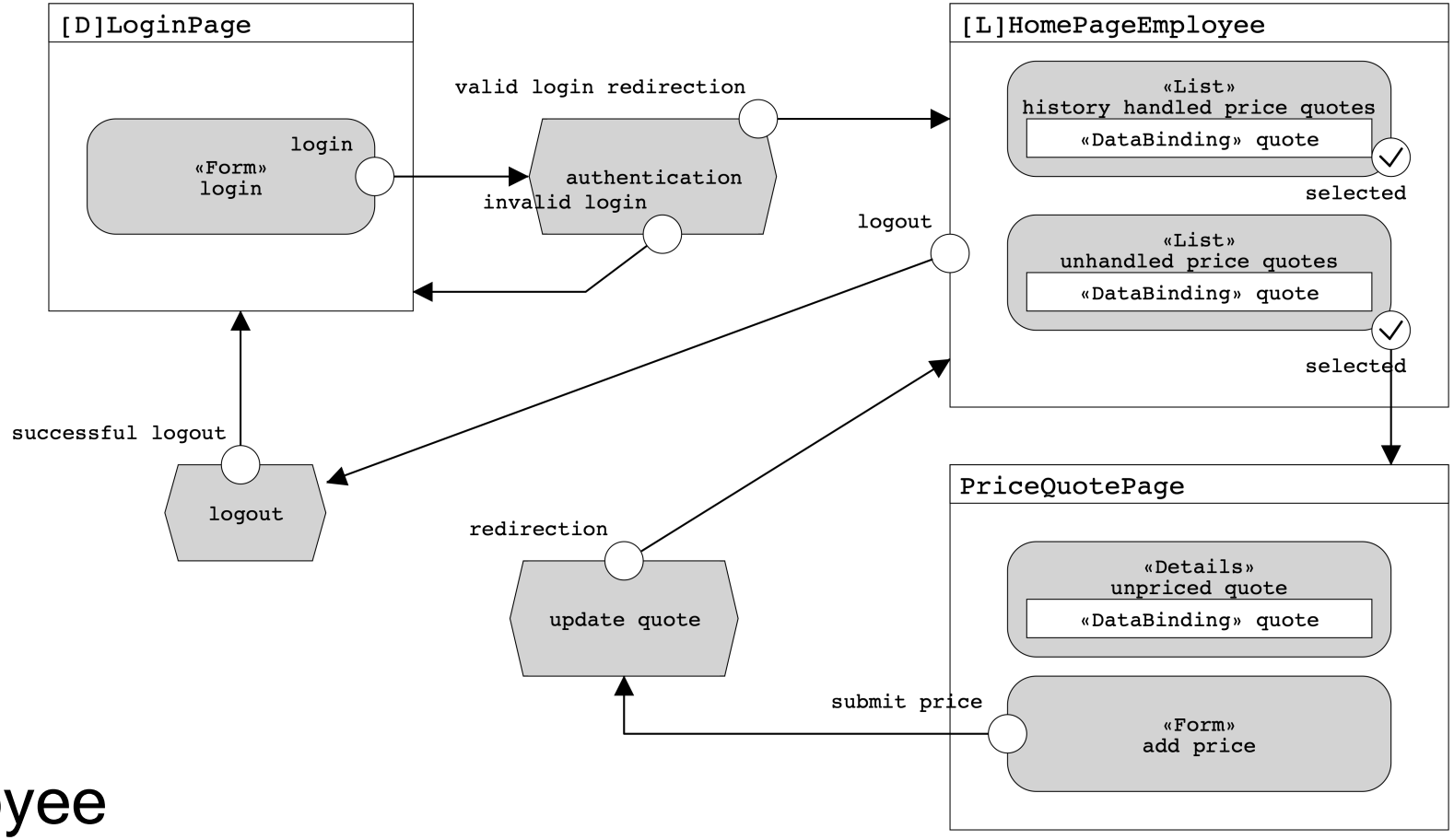
Pages (view), view components, events, actions

Application design (IFML)



Client

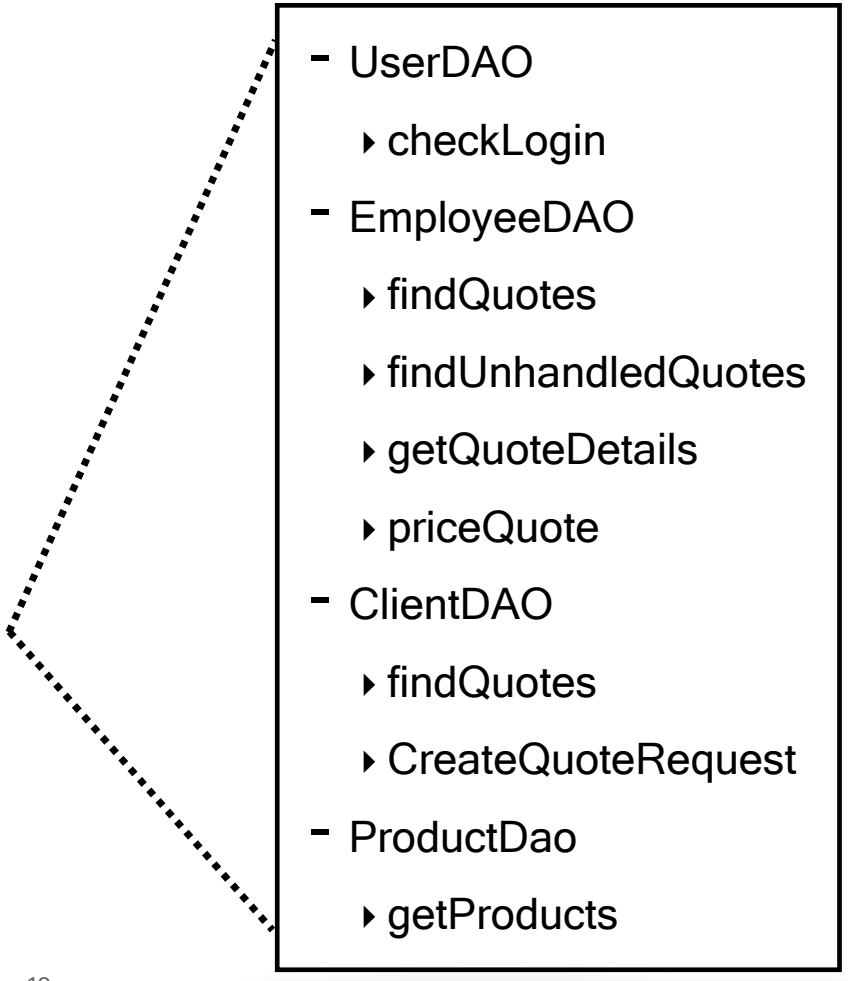
Application design (IFML)



Employee

Components

- Model objects (Beans)
  - User
  - Option
  - Product
  - Quote
- Data Access Objects (DAO)



Components

- Controllers (servlets)
  - CheckLogin
  - Logout
  - GoToClientHome
  - CreateQuote
  - GoToEmployeeHome
  - PriceQuote
- Views (Templates)
  - Login (all): Login form
  - HomeClient: Quote list, new quote form
  - HomeEmployee: Quote list, unhandled quote list
  - PriceQuote: Quote details, price quote form

Event: login

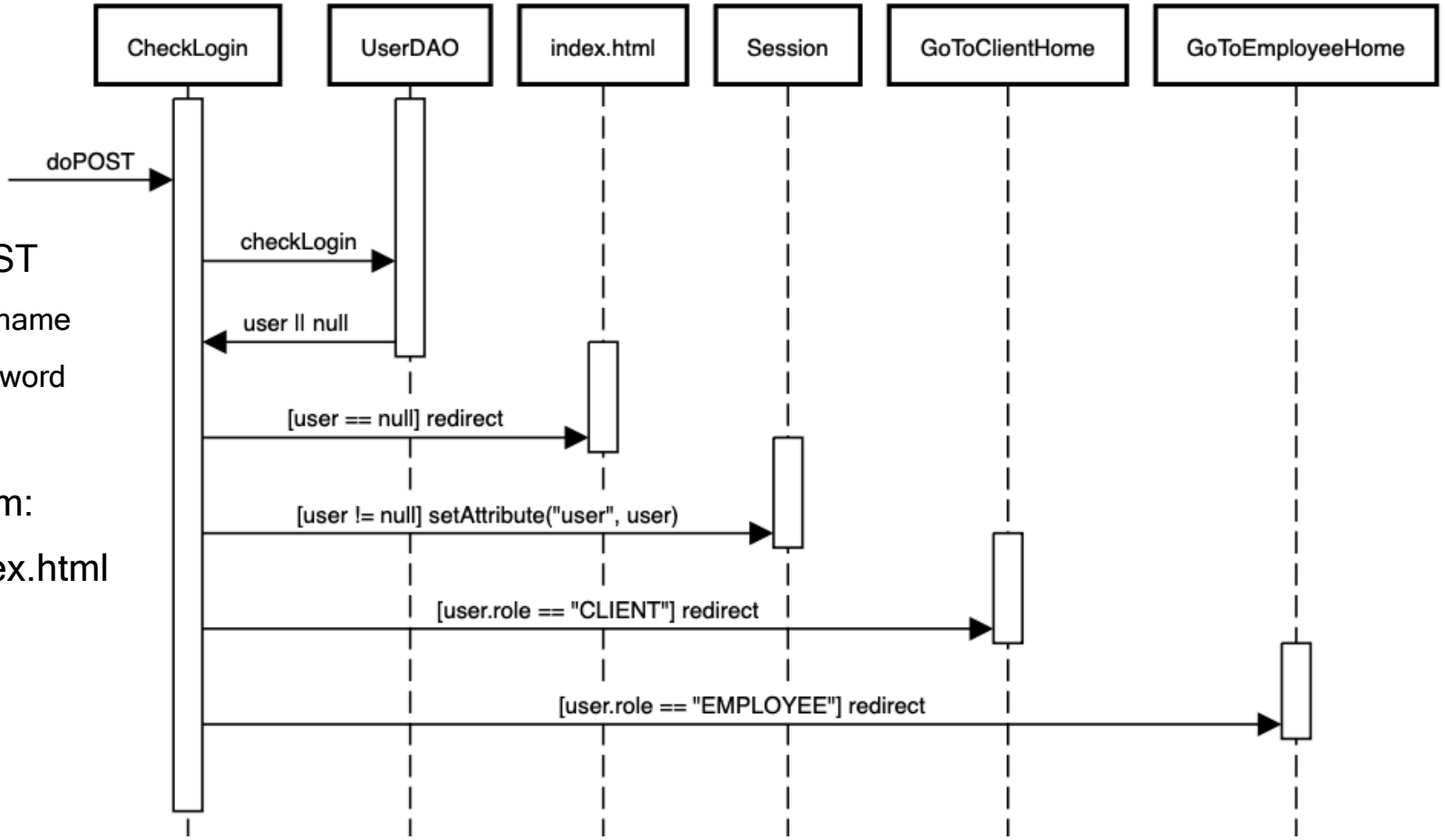
POST

username

password

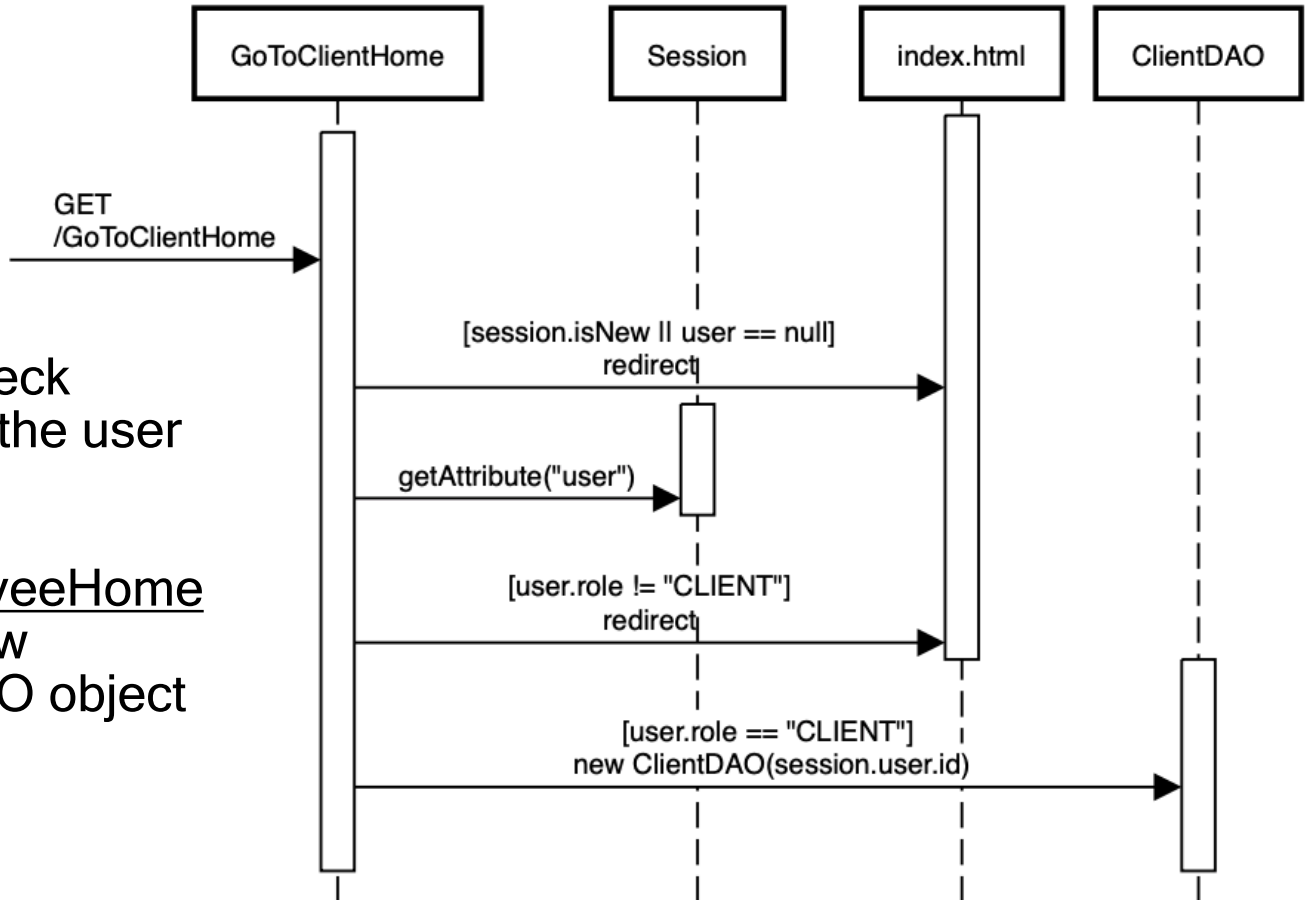
From:

index.html

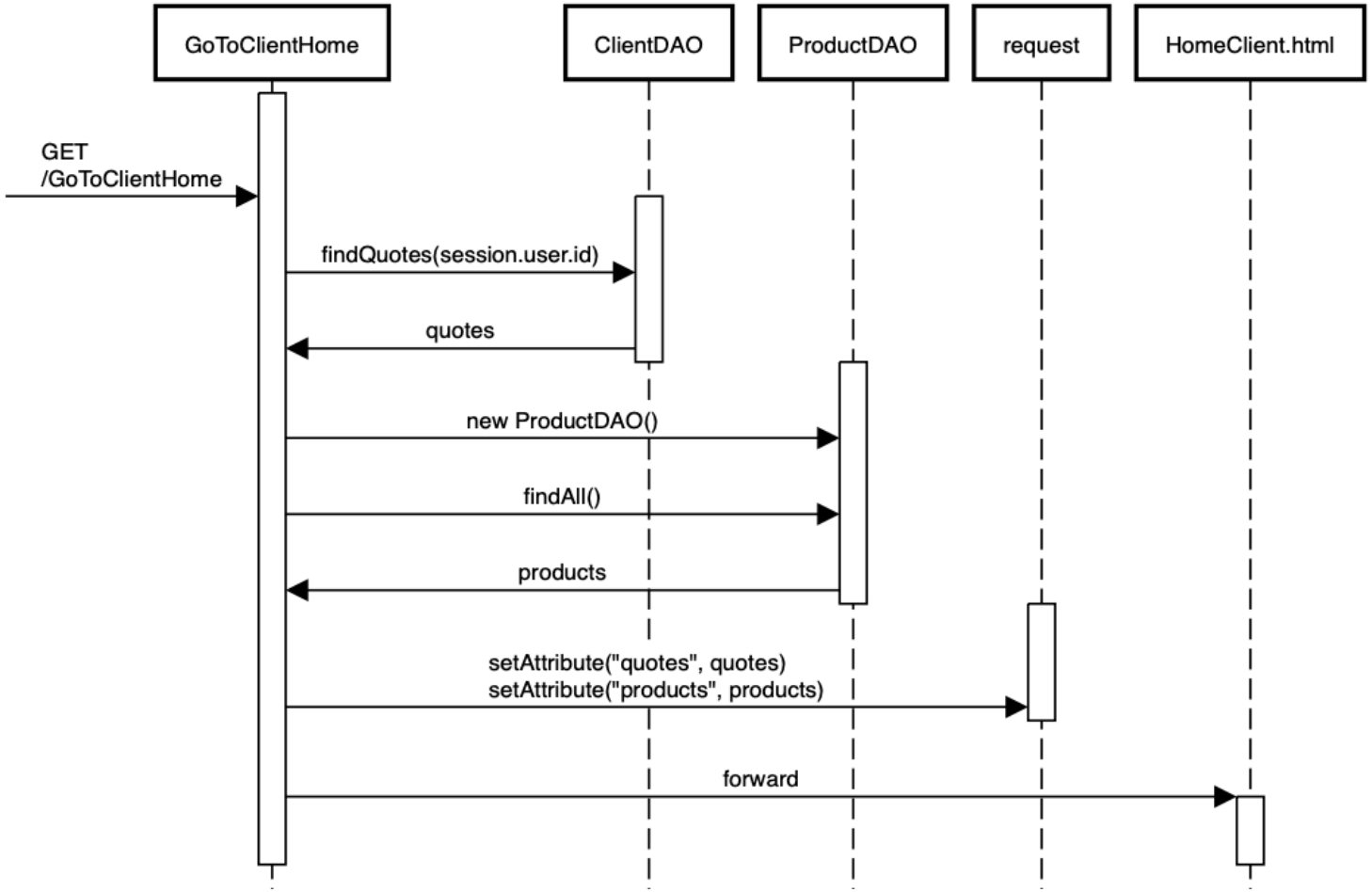


Checking access rights

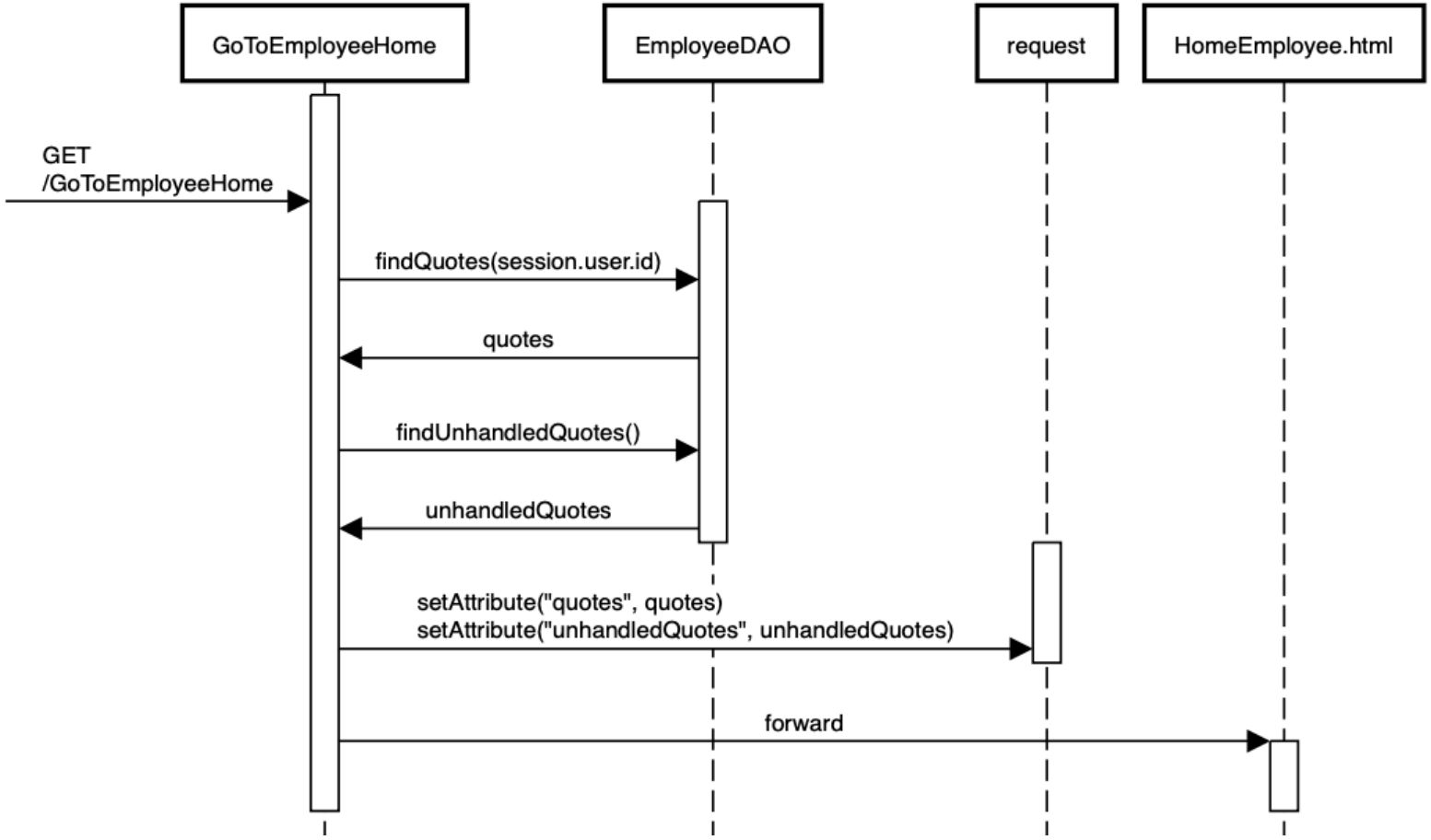
The same check occurs when the user requests GET /GoToEmployeeHome creating a new EmployeeDAO object instance.



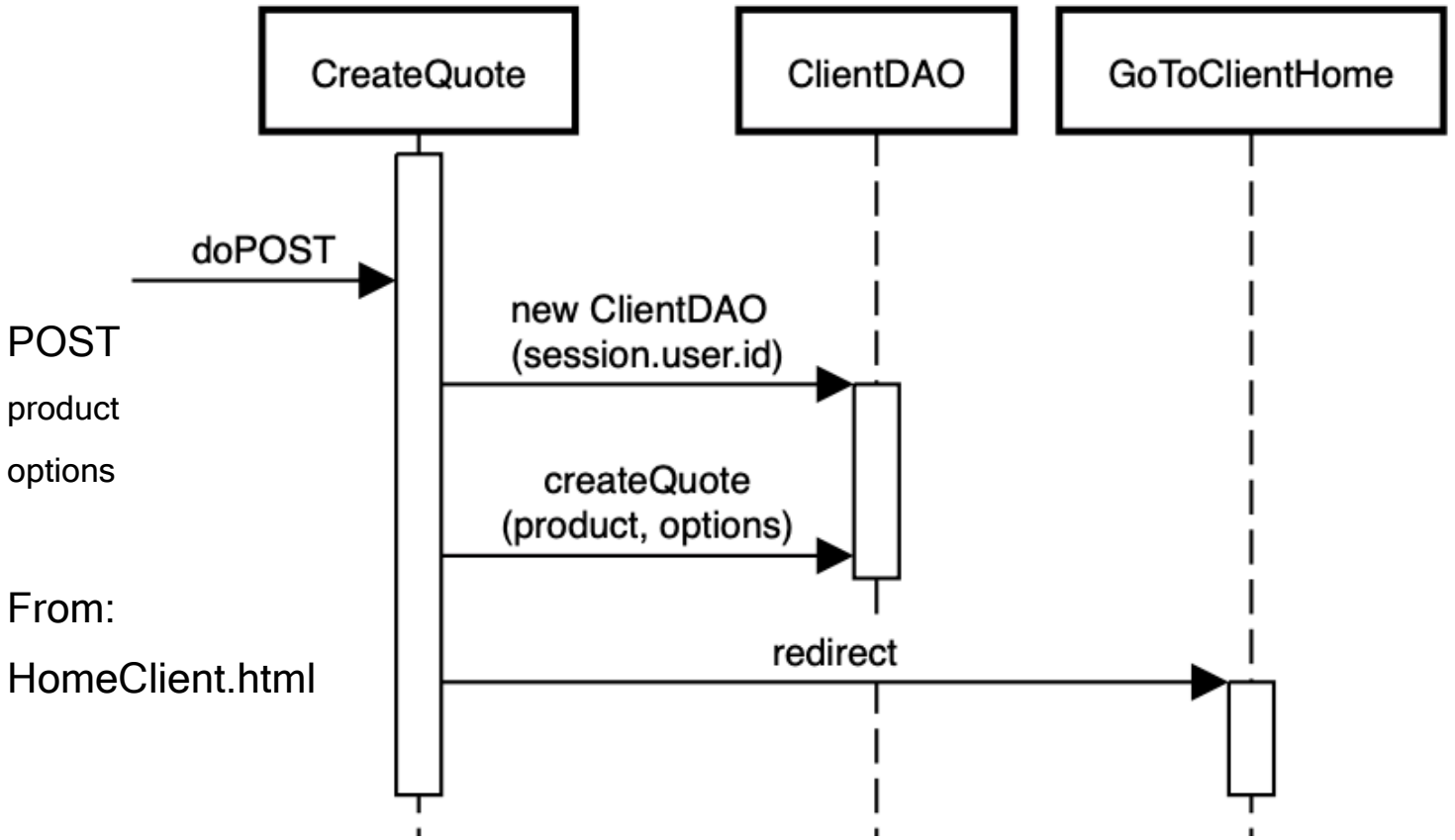
GoToClientHome



GoToEmployeeHome



Event: create quote



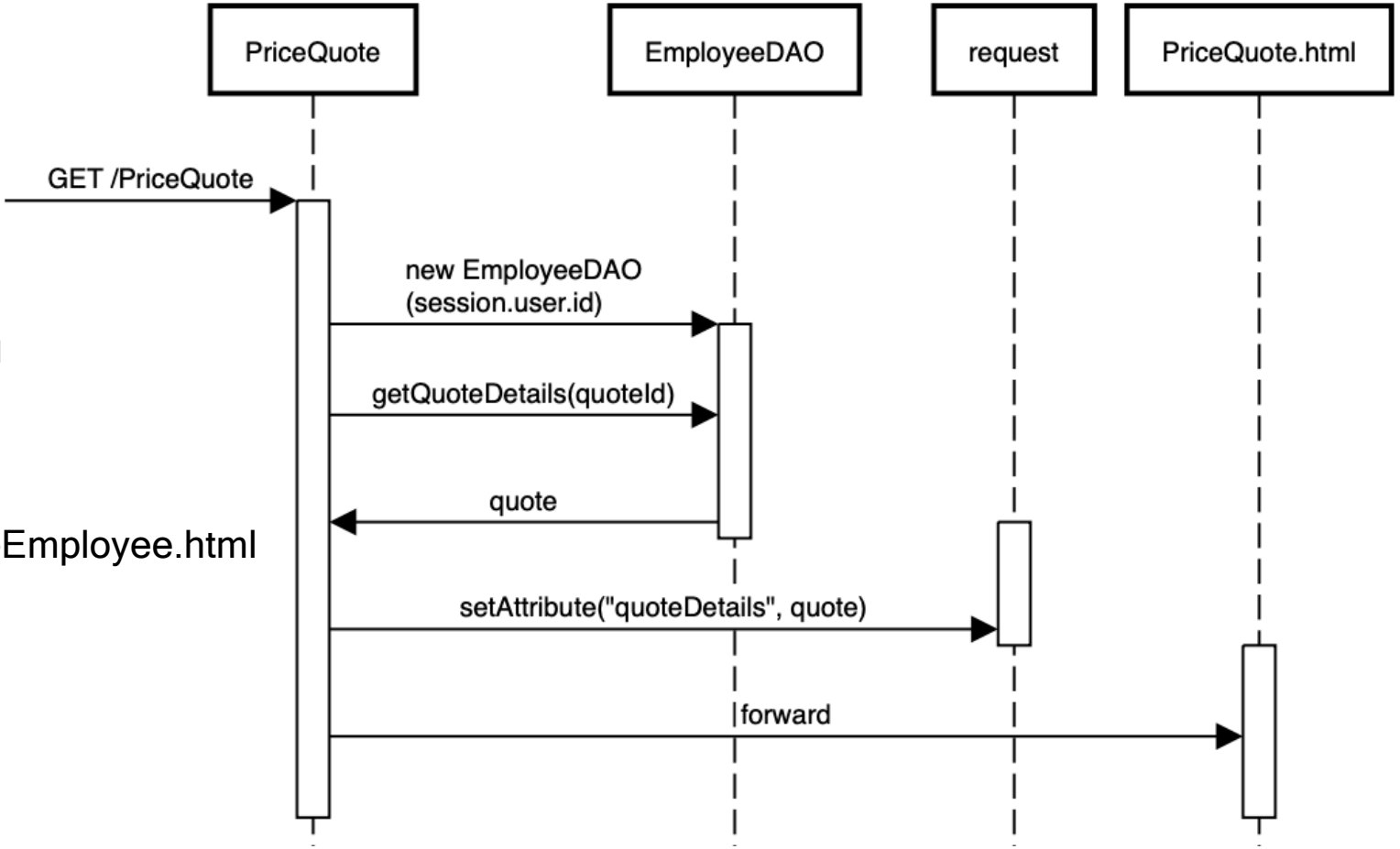
Event: price quote

GET

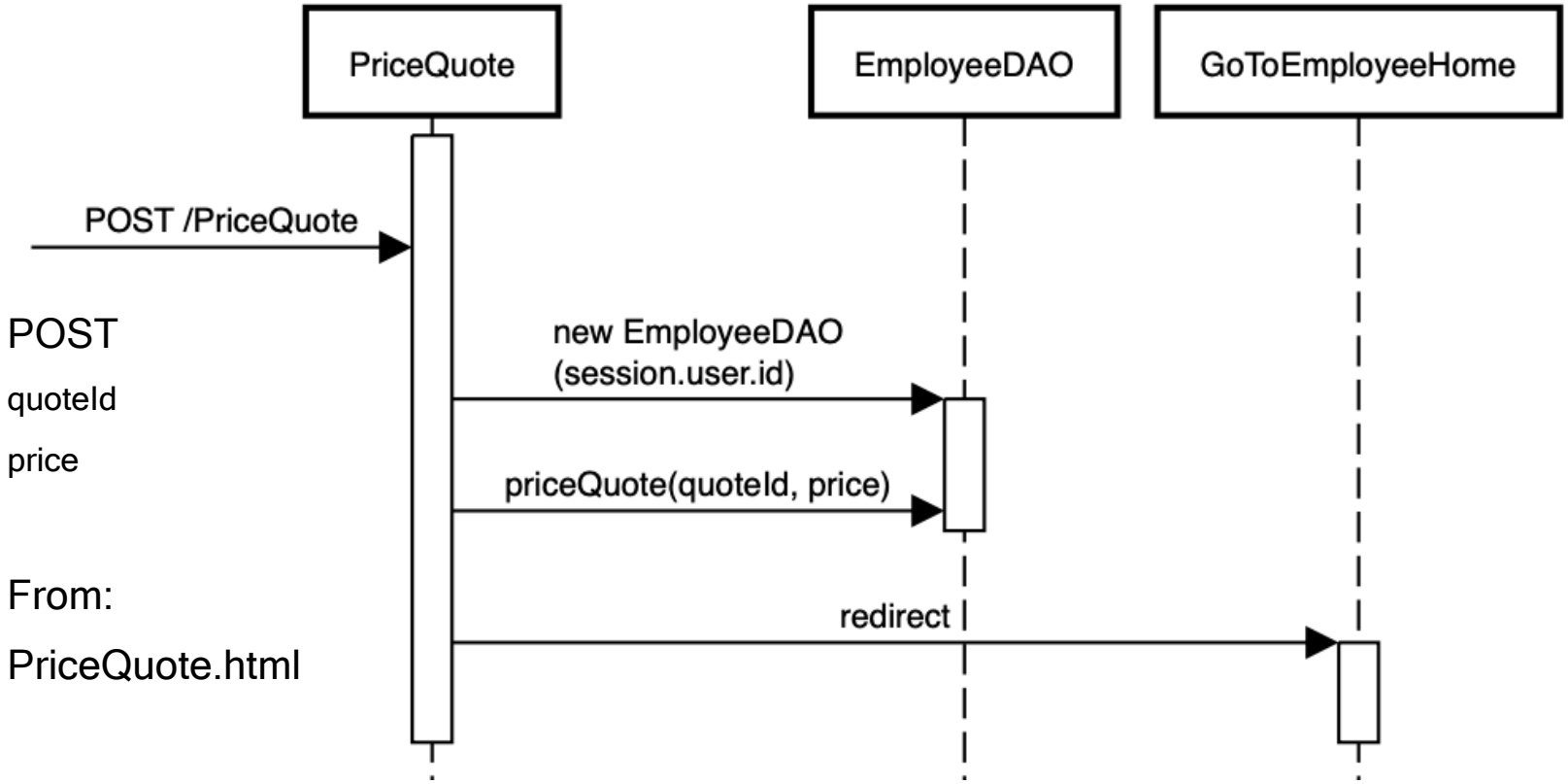
quoteId

From:

HomeEmployee.html



Event: price quote



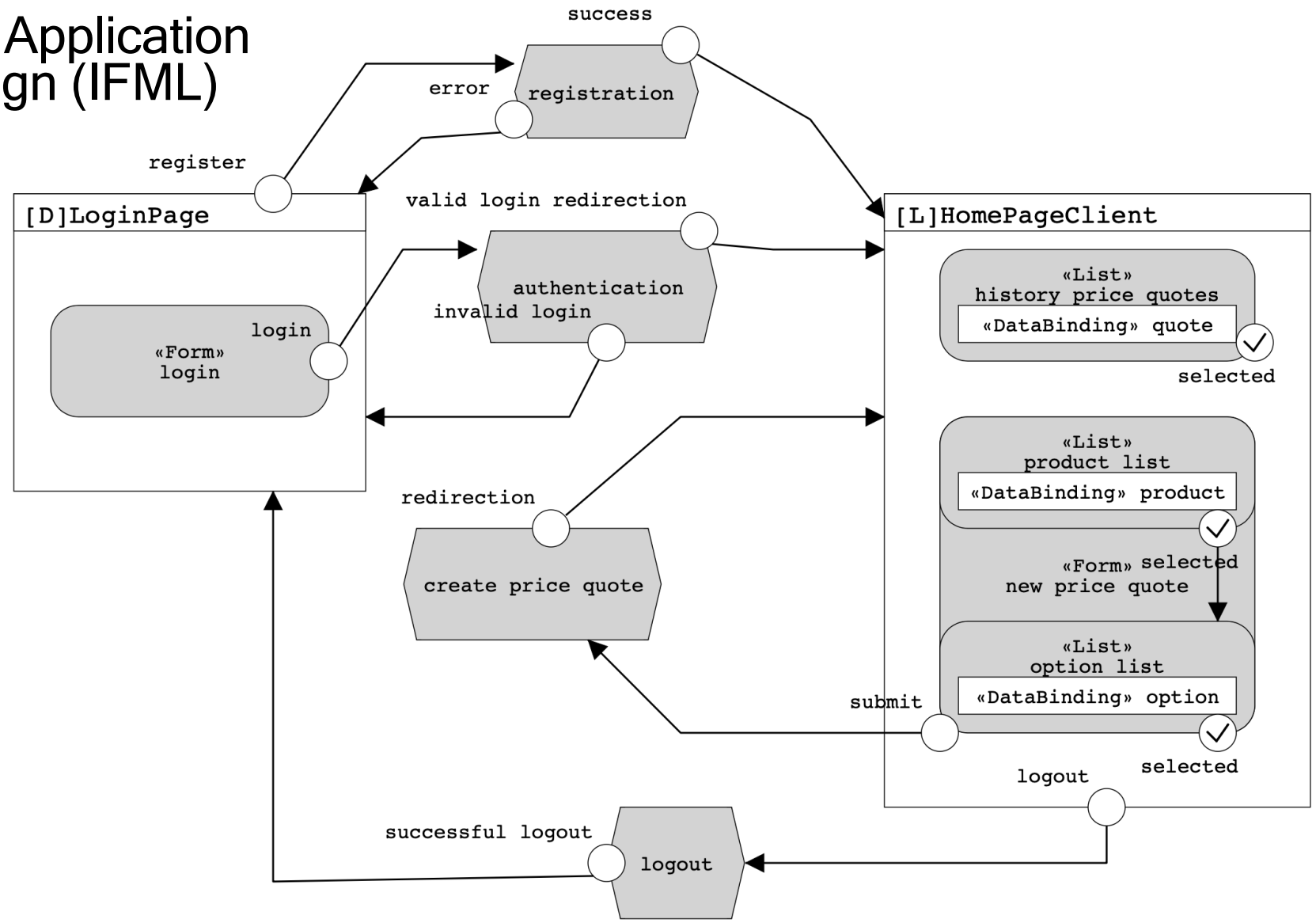
RIA - Components

- Controllers (servlets)
  - CheckLogin
  - Logout
  - GetQuotes (Client)
  - GetProducts (Client)
  - CreateQuote (Client)
  - GetQuotes (Employee)
  - GetUnhandledQuotes (Employee)
  - PriceQuote (Employee)
- Views (HTML files)
  - Index.html (all): Login form / Registration form
  - HomeClient.html: Quote list, product list, new quote form
  - HomeEmployee.html: Quote list, unhandled quote list, price quote form

RIA - Scripts

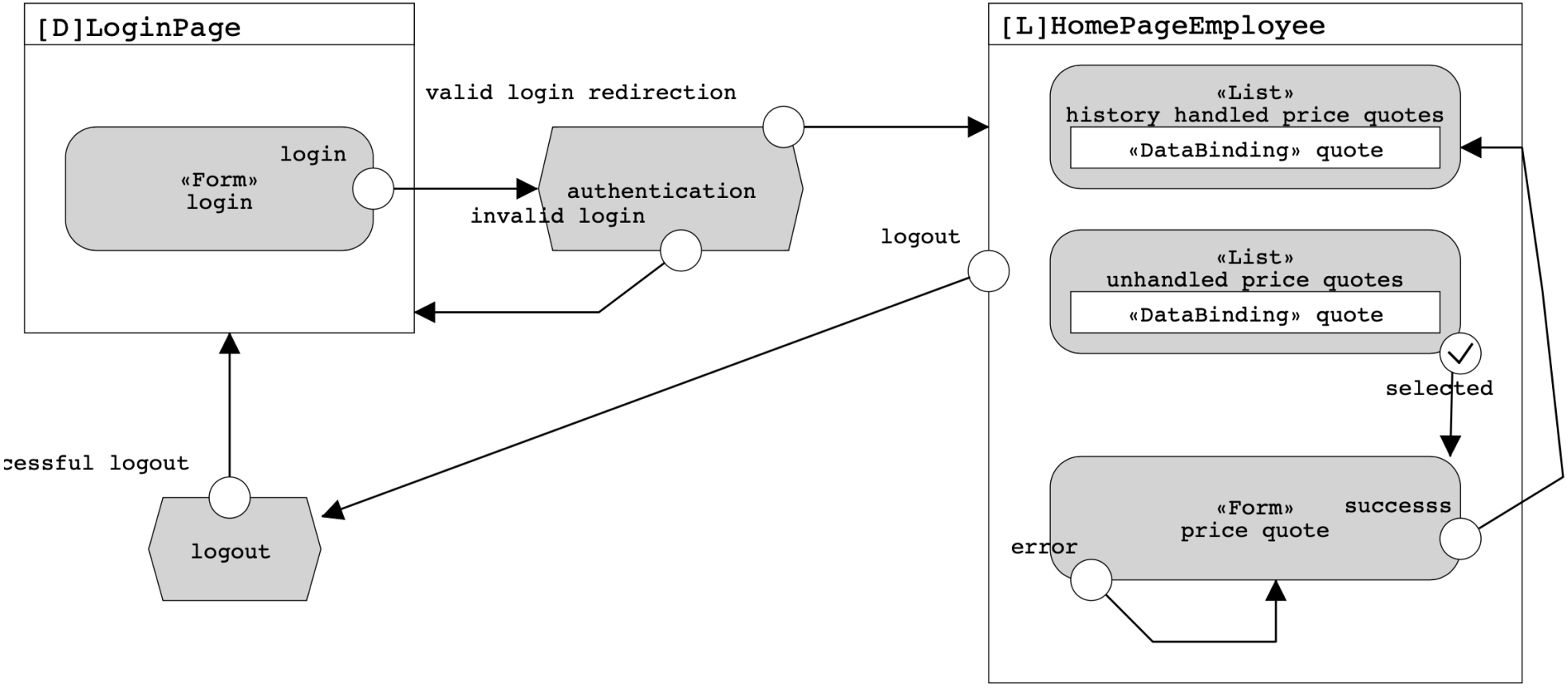
- user-util.js
  - Toggle registration form
  - Submit login
  - Submit registration
- quote.js
  - Build quote card
  - On price quote button click handler
  - On price quote form submit handler
- client-home.js (type = “module”)
  - Check session
  - Fetch quote list and product list
  - Build HTML (quotes and products)
  - On select product handler (show options)
  - On submit quote request handler
- employee-home.js (type = “module”)
  - Check session
  - Fetch quote list and unhandled quote list
  - Build HTML (quotes and unhandled quotes)
  - On select product handler (show options)
  - On submit quote request handler

RIA - Application design (IFML)



Client

RIA - Application design (IFML)



Employee