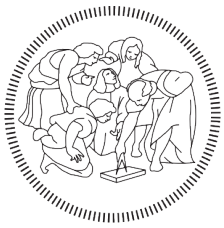
TIW - Gestione preventivi



POLITECNICO
MILANO 1863

Politecnico di Milano
Anno Accademico 2021/2022

Prof. Piero Fraternali

Riccardo Inghilleri (Codice Persona 10713236 - Matricola 937011)
Manuela Merlo (Codice Persona 10670533 - Matricola 936925)

Indice

1 Gestione Preventivi - Pure HTML

1.1 Analisi dei dati per il database

Un'applicazione web consente la gestione di richieste di preventivi per prodotti personalizzati. L'applicazione supporta registrazione e login di **clienti** e **impiegati** mediante una pagina pubblica con opportune form. La registrazione controlla l'unicità dello **username**. 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. Selezionando uno dei preventivi il cliente ne visualizza i dettagli. Mediante la form di creazione di un preventivo 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 effettua il 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. Il prezzo definito dall'impiegato risulta visibile al cliente quando questi accede all'elenco dei propri preventivi e visualizza i dettagli del preventivo. La pagina PREZZA PREVENTIVO contiene anche un collegamento per tornare alla HOME PAGE IMPIEGATO. L'applicazione consente il logout dell'utente.

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

1.2 Database Design

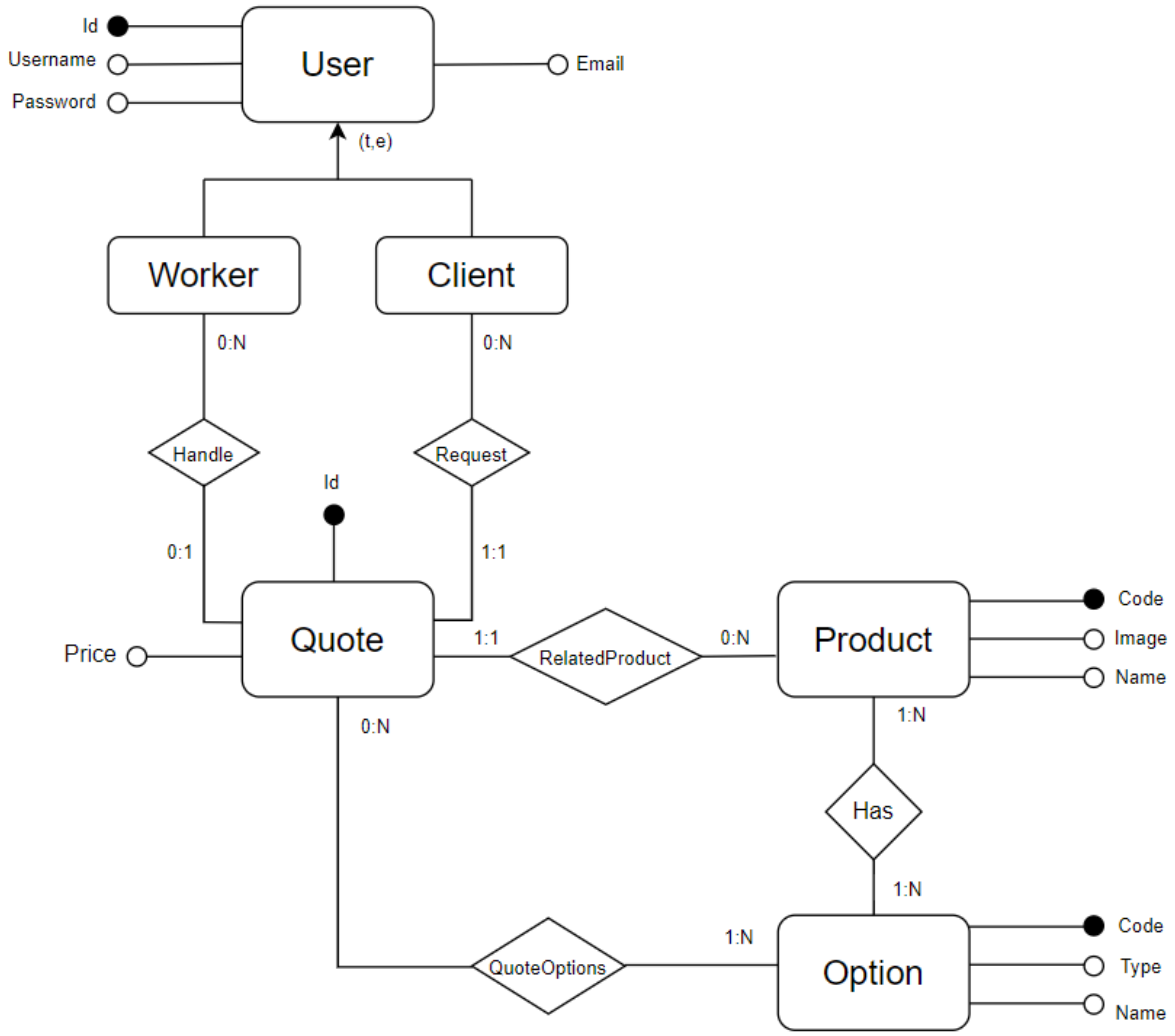


Figura 1: Database Desing

1.3 Local Database schema

```
CREATE TABLE `user` (
`id` int AUTO_INCREMENT,
`username` varchar(45),
`email` varchar(255),
`password` varchar(45) NOT NULL,
`role` varchar(45) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE(`email`),
CONSTRAINT `UC_UsernameRole` UNIQUE (`username`,`role`),
CONSTRAINT `NN_UsernameORRole`
CHECK (`username` is not null OR `email` is not null))

CREATE TABLE `product` (
```

```
`code` int AUTO_INCREMENT,
`image` varchar(45) NOT NULL,
`name` varchar(45) NOT NULL,
PRIMARY KEY (`code`))


CREATE TABLE `option` (
`code` int AUTO_INCREMENT,
`type` varchar(45) NOT NULL,
`name` varchar(45) NOT NULL,
PRIMARY KEY (`code`))


CREATE TABLE `quote` (
`id` int auto_increment,
`clientId` int NOT NULL,
`workerId` int,
`productCode` int NOT NULL,
`price` int default null,
PRIMARY KEY (`id`),
CONSTRAINT `asstoclient` FOREIGN KEY (`clientId`)
REFERENCES `client` (`id`) ON DELETE CASCADE,
CONSTRAINT `asstoworker` FOREIGN KEY (`workerId`)
REFERENCES `worker` (`id`) ON DELETE CASCADE,
CONSTRAINT `asstoproduct` FOREIGN KEY (`productCode`)
REFERENCES `product` (`code`) ON DELETE CASCADE)


CREATE TABLE `productoptions`(
`productCode` int,
`optionCode` int,
PRIMARY KEY (`productCode`,`optionCode`),
CONSTRAINT `relwithproduct` FOREIGN KEY (`productCode`)
REFERENCES `product` (`code`) ON DELETE CASCADE,
CONSTRAINT `relwithoption` FOREIGN KEY (`optionCode`)
REFERENCES `option` (`code`) ON DELETE CASCADE)


CREATE TABLE `quoteoptions`(
`quoteId` int NOT NULL,
`optionCode` int NOT NULL,
PRIMARY KEY (`quoteId`,`optionCode`),
CONSTRAINT `relwithquote` FOREIGN KEY (`quoteId`)
REFERENCES `quote` (`id`) ON DELETE CASCADE,
CONSTRAINT `selectedoption` FOREIGN KEY (`optionCode`)
REFERENCES `option` (`code`) ON DELETE CASCADE)
```

1.4 Analisi dei dati per i requisiti dell'applicazione

Un'applicazione web consente la gestione di richieste di preventivi per prodotti personalizzati. L'applicazione supporta **registrazione** e **login** di clienti e impiegati mediante una pagina pubblica con opportune **form**. La registrazione controlla l'unicità dello username. 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. **Selezionando uno dei preventivi** il cliente ne **visualizza i dettagli**. Mediante la form di creazione di un preventivo 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 effettua il 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. Il prezzo definito dall'impiegato risulta visibile al cliente quando questi **accede all'elenco dei propri preventivi** e visualizza i dettagli del preventivo. La pagina PREZZA PREVENTIVO contiene anche un collegamento per tornare alla HOME PAGE IMPIEGATO. L'applicazione consente il **logout** dell'utente.

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

1.5 Completamento delle specifiche

- La pagina di default contiene la form di login;
- Lo username e la password non possono essere nulli;
- Un client e un worker possono avere lo stesso username, due client o due worker invece no;
- Il prezzo non può essere negativo;
- É permesso al client di richiedere preventivi dello stesso prodotto con le stesse opzioni;
- Nel caso di riapertura del sito da parte di un utente che precedentemente non ha effettuato il logout, non viene richiesta una nuova autenticazione e viene aperta direttamente la sua home page.

- Quando un client vuole vedere i dettagli di un preventivo, viene reindirizzato a un'altra pagina.

1.6 Application Desing

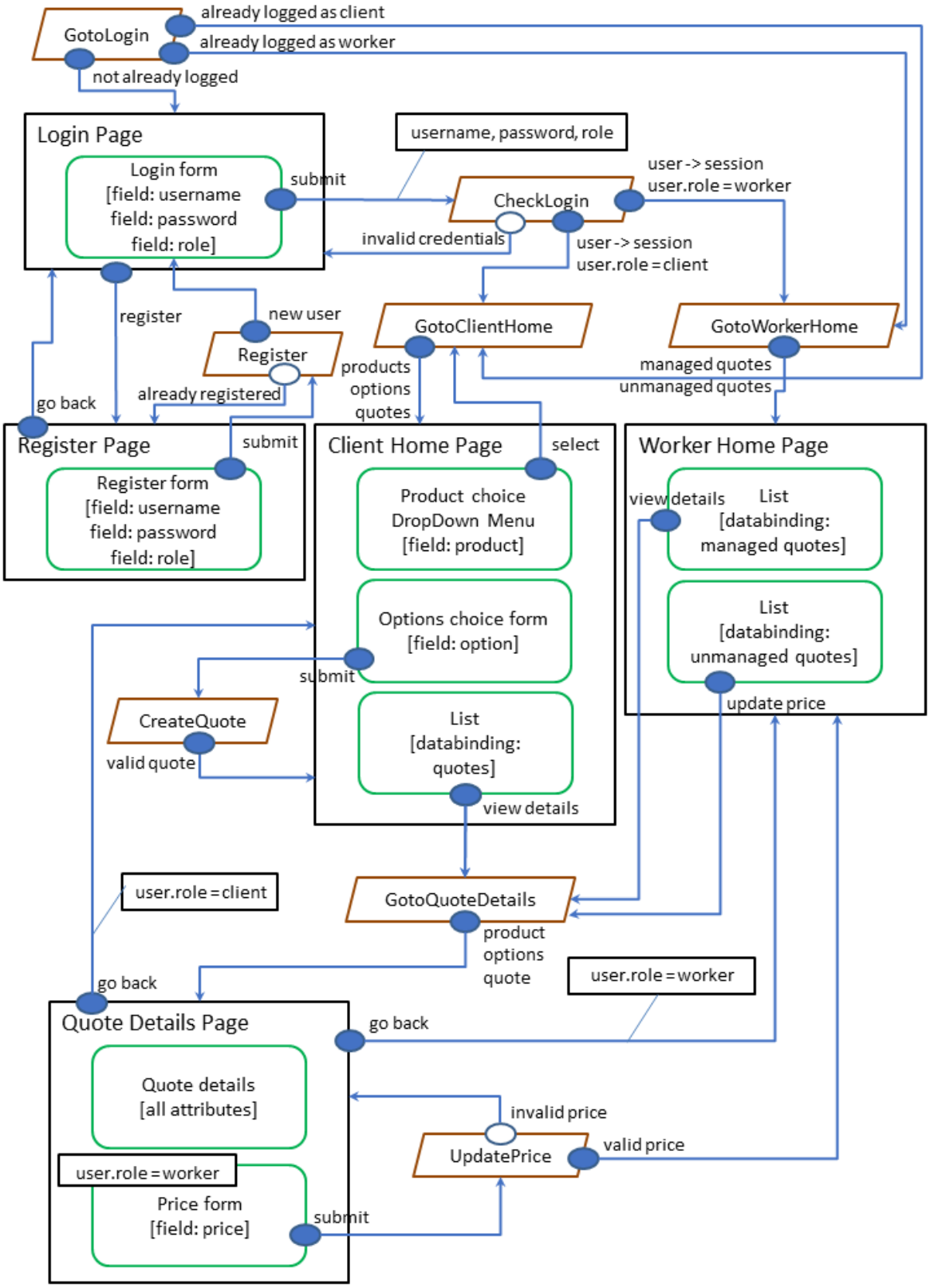


Figura 2: IFML diagram

1.7 Components

- **Model Objects (Beans)**
  - Option
  - Product
  - Quote
  - User
- **Data Access Objects (Classes)**
  - **OptionDAO**
    - boolean hasOptionByCode(int productCode, int optionCode);
    - List<Option> findOptionsByProductCode(int productCode);
    - List<Option> findOptionsByQuoteId(int quoteId);
    - void insertOption(int quoteId, int optionCode);
  - **ProductDAO**
    - List<Product> findAllProducts();
    - Product findProductByCode(int code);
  - **QuoteDAO**
    - List<Quote> findQuotesByUserId(int userId, String role);
    - Quote findQuoteById(int quoteId);
    - List<Quote> findUnmanagedQuotes();
    - int insertQuote(int clientId, int productCode);
    - void updateQuote(int quoteId, int workerId, int price);
  - **UserDAO**
    - User findUser(String username, String password, String role);
    - User findUser(String username, String role);
    - User findClientById(int clientId);
    - void registerUser(String username, String password, String role);
- **Controllers (Servlets)**
  - GotoLogin
  - CheckLogin
  - Register
  - GotoClientHome
  - GotoWorkerHome
  - CreateQuote
  - GotoQuoteDetails
  - UpdatePrice
  - Logout

- **Filters**
  - SessionChecker
  - ClientChecker
  - WorkerChecker
- **Views (Templates)**
  - Login.html
  - Register.html
  - ClientHome.html
  - WorkerHome.html
  - QuoteDetails.html

1.8 Events

I filtri utilizzati sono i seguenti:

SessionChecker:

- **Controllo:** session.isNew() || session.getAttribute("currentUser") == null
- **Azione:** redirect alla Login Page
- **Quando:** prima di ogni Servlet

ClientChecker: controlla che lo user corrente sia un Client

- **Controllo:** !user.getRole().equals("client")
- **Azione:** redirect alla Login Page
- **Quando:** prima di /GotoClientHome e /CreateQuote

WorkerChecker: controlla che lo user corrente sia un Worker

- **Controllo:** !user.getRole().equals("worker")
- **Azione:** redirect alla Login Page
- **Quando:** prima di /GotoWorkerHome e /UpdatePrice

1.8.1 Go to Login

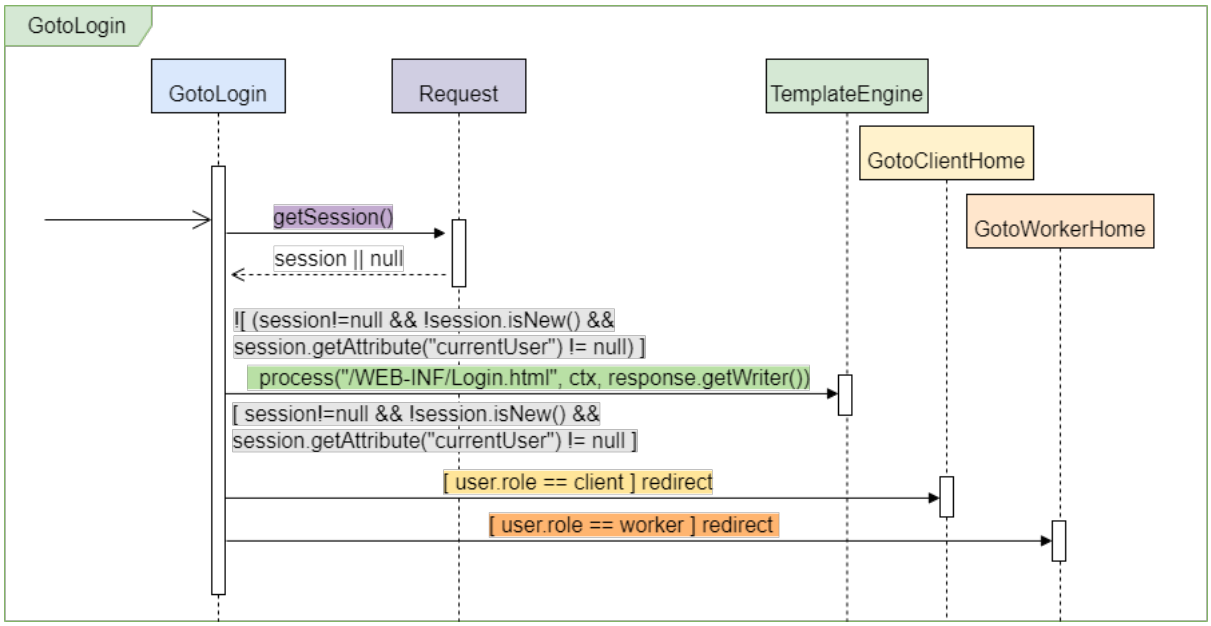


Figura 3: Event - Go to Login

1.8.2 Login

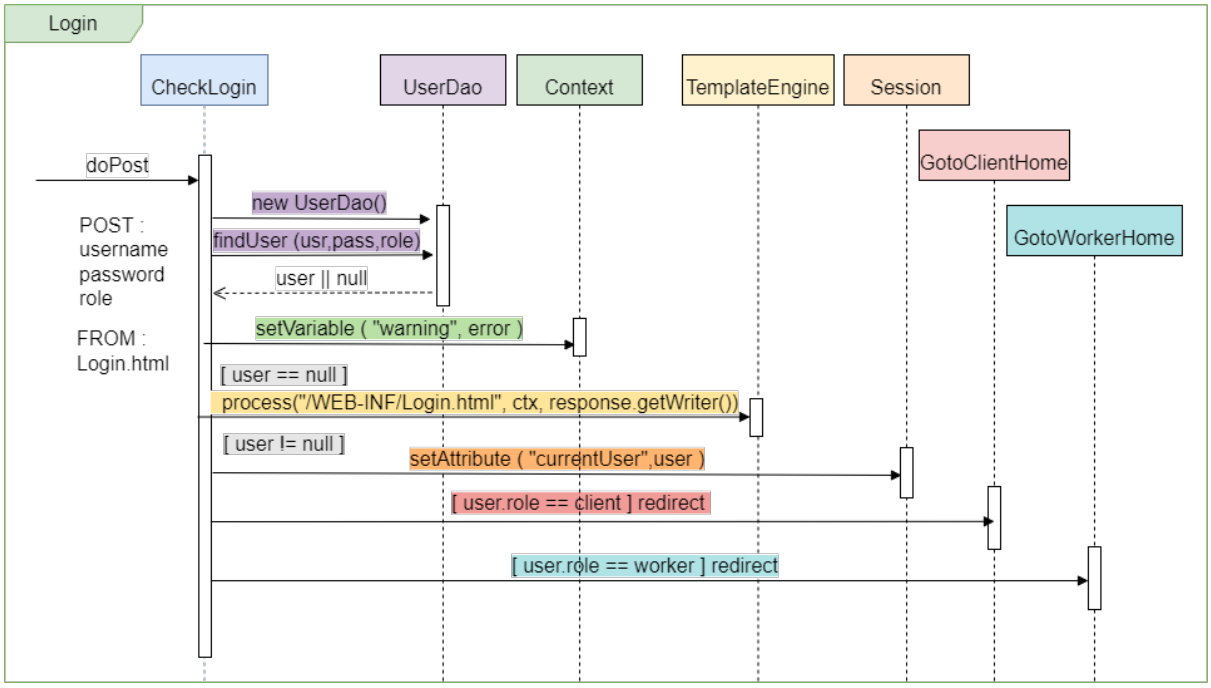


Figura 4: Event - Login

Controlli:
HTML:
Lo username e la password sono obbligatori.

Server:
Viene controllato che i parametri della Form non sia nulli, vuoti o invalidi:

```
(username = null || role = null ||
(!role.equalsIgnoreCase("client") && !role.equalsIgnoreCase("worker"))
|| password = null ||
username.isEmpty() || password.isEmpty())
```

1.8.3 Register

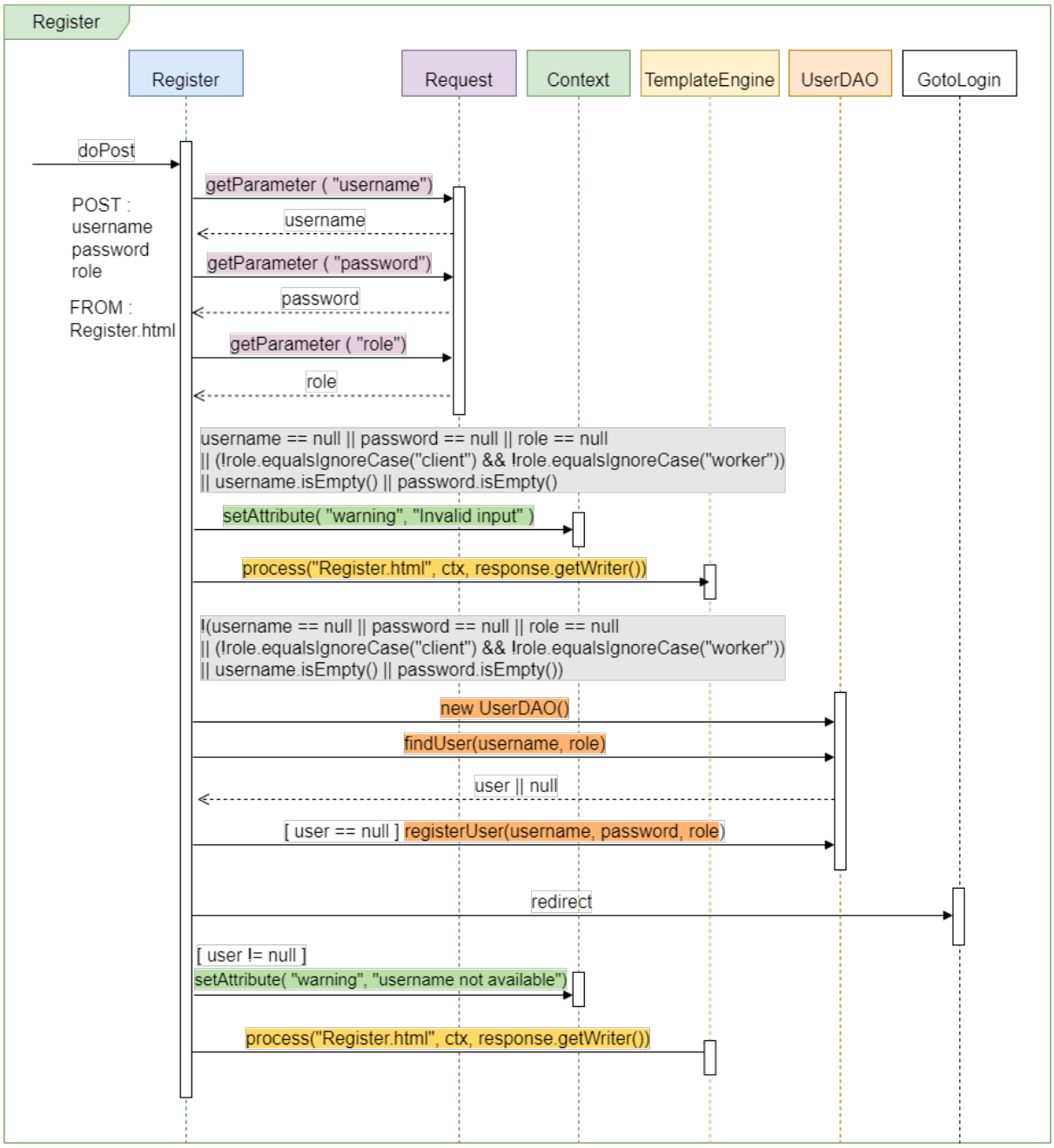


Figura 5: Event - Register

Controlli:
Server: Viene controllato che :

1. I parametri della form non siano nulli o vuoti (username - password - ruolo).
2. Che all'interno del DB non esista un altro utente con lo stesso ruolo, avente lo stesso username.

1.8.4 GotoClientHome

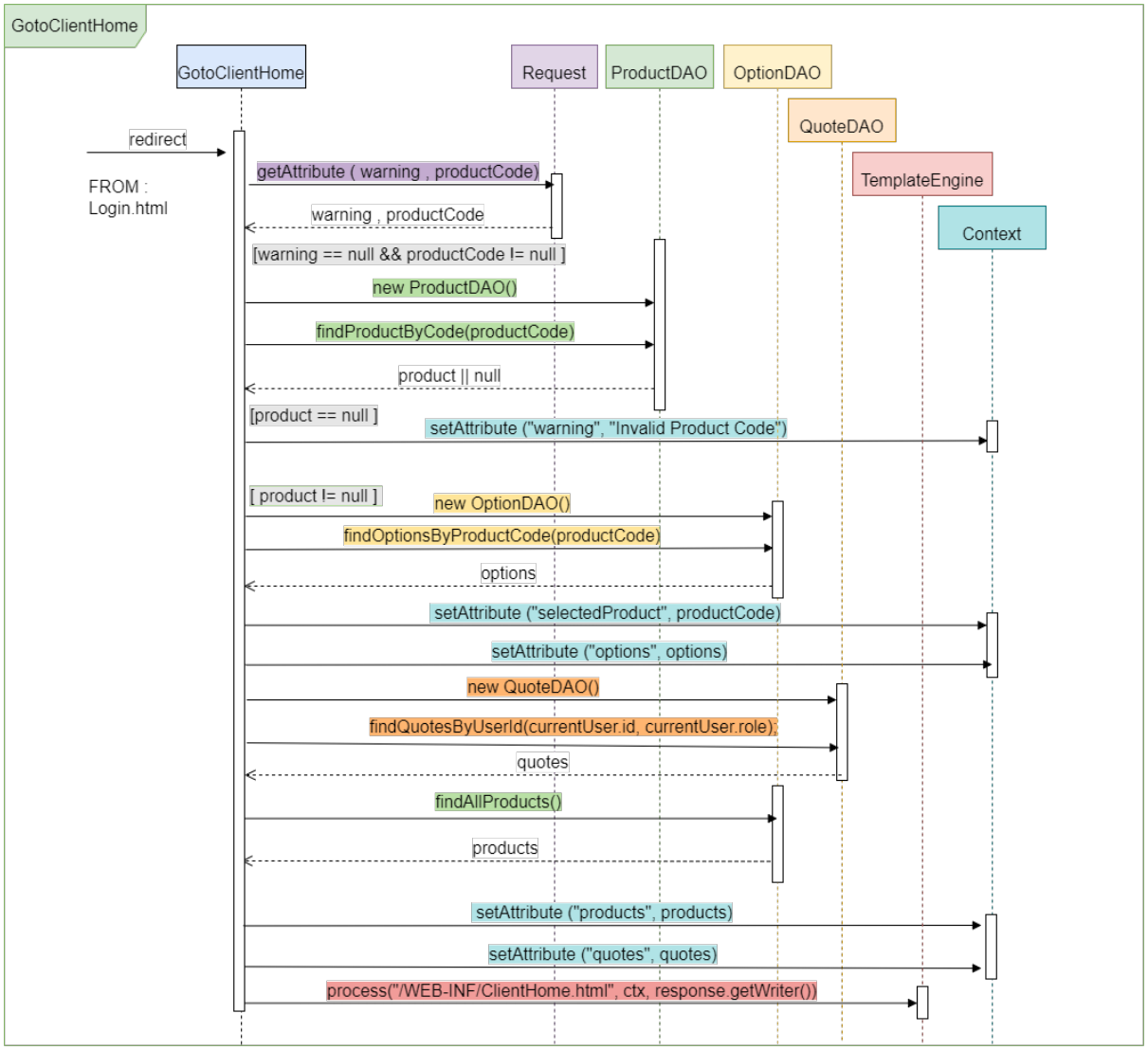


Figura 6: Event - Go to Client Home

Controlli:
Server: Al ricaricamento della pagina, dopo la selezione del prodotto per la richiesta di un nuovo preventivo, viene controllato che :

1. L'id del prodotto, se selezionato, sia in un formato numerico e corrisponda ad un id effetivamente presente nel DB.

1.8.5 GotoWorkerHome

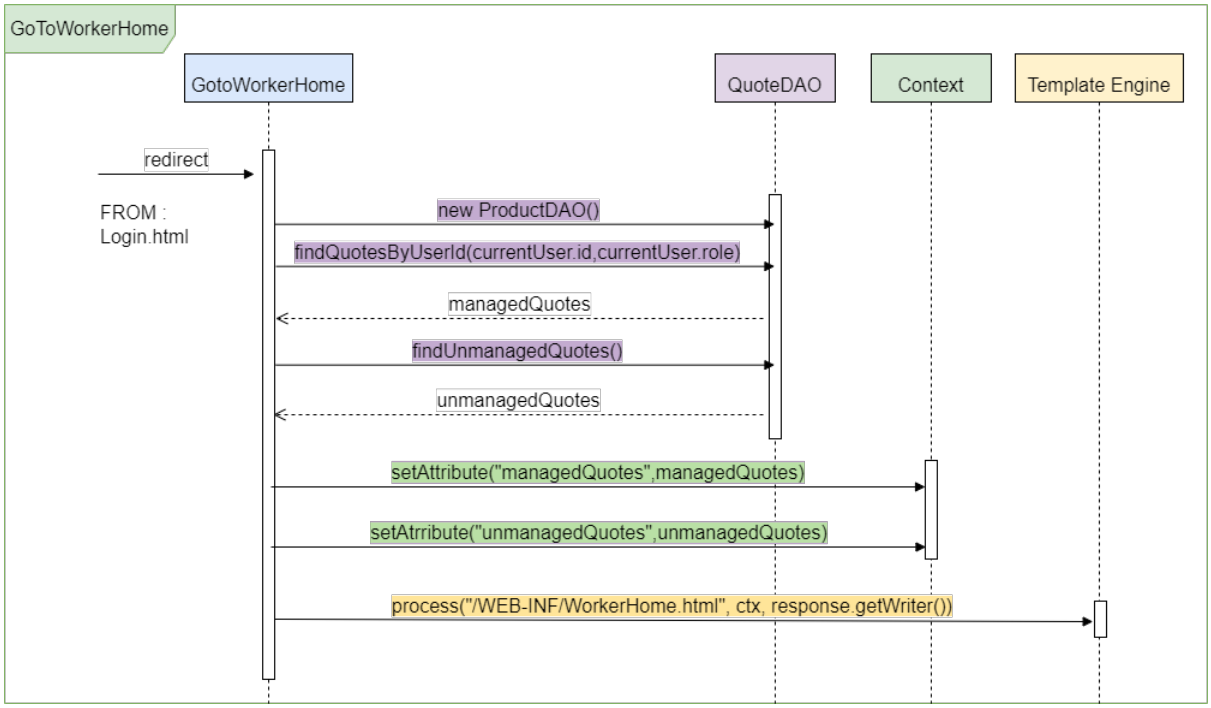


Figura 7: Event - Go to Worker Home

Controlli:
Server: Dato che la raccolta dei dati necessari per la visualizzazione della pagina non dipendono da nessun parametro di input, ma solo dall'id dello user presente nella sessione, non vengono effettuati controlli aggiuntivi a quelli già fatti dai filtri ovvero: che un utente sia effettivamente loggato e che quest'ultimo sia effetivamente un worker.

1.8.6 GotoQuoteDetails

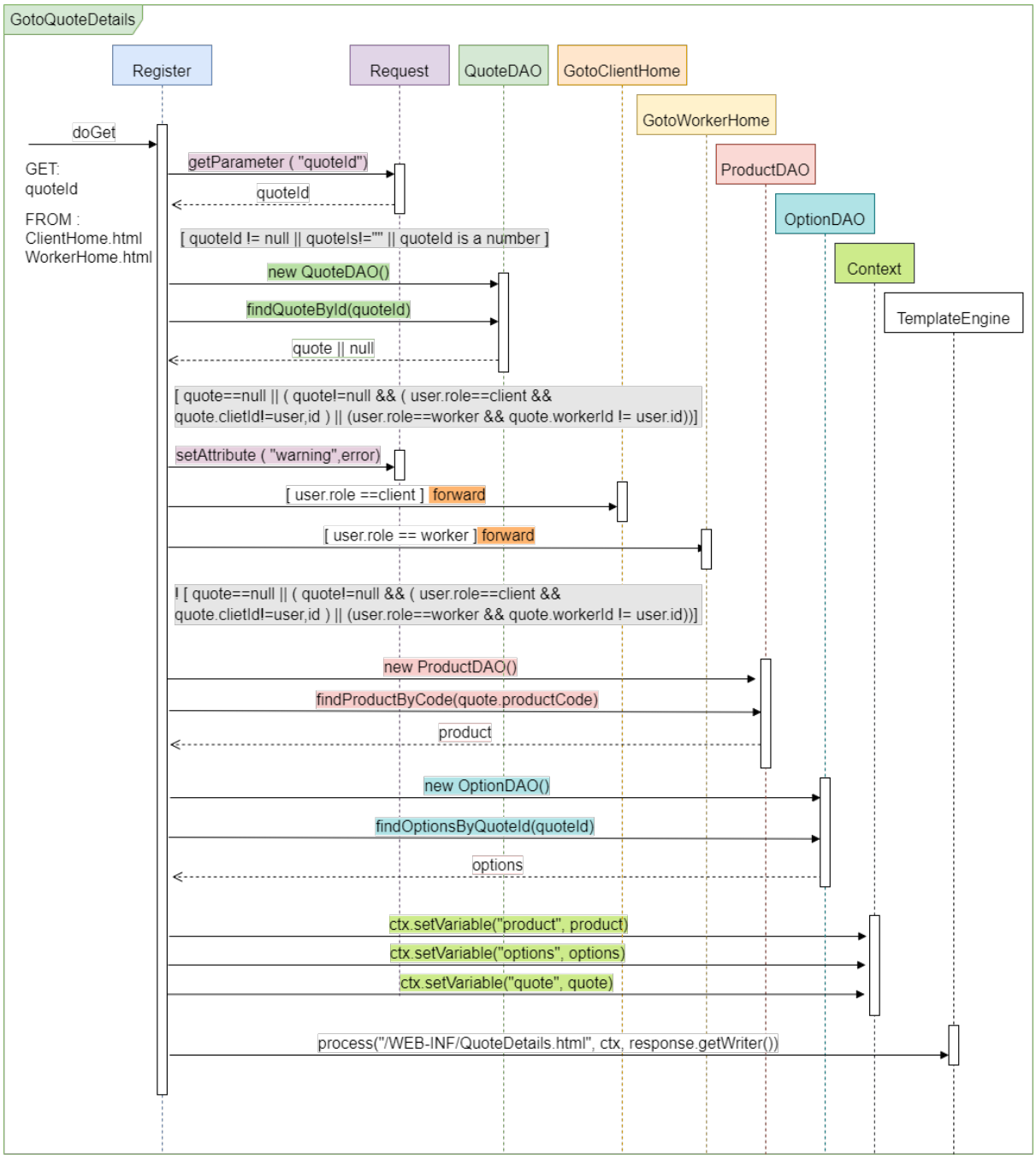


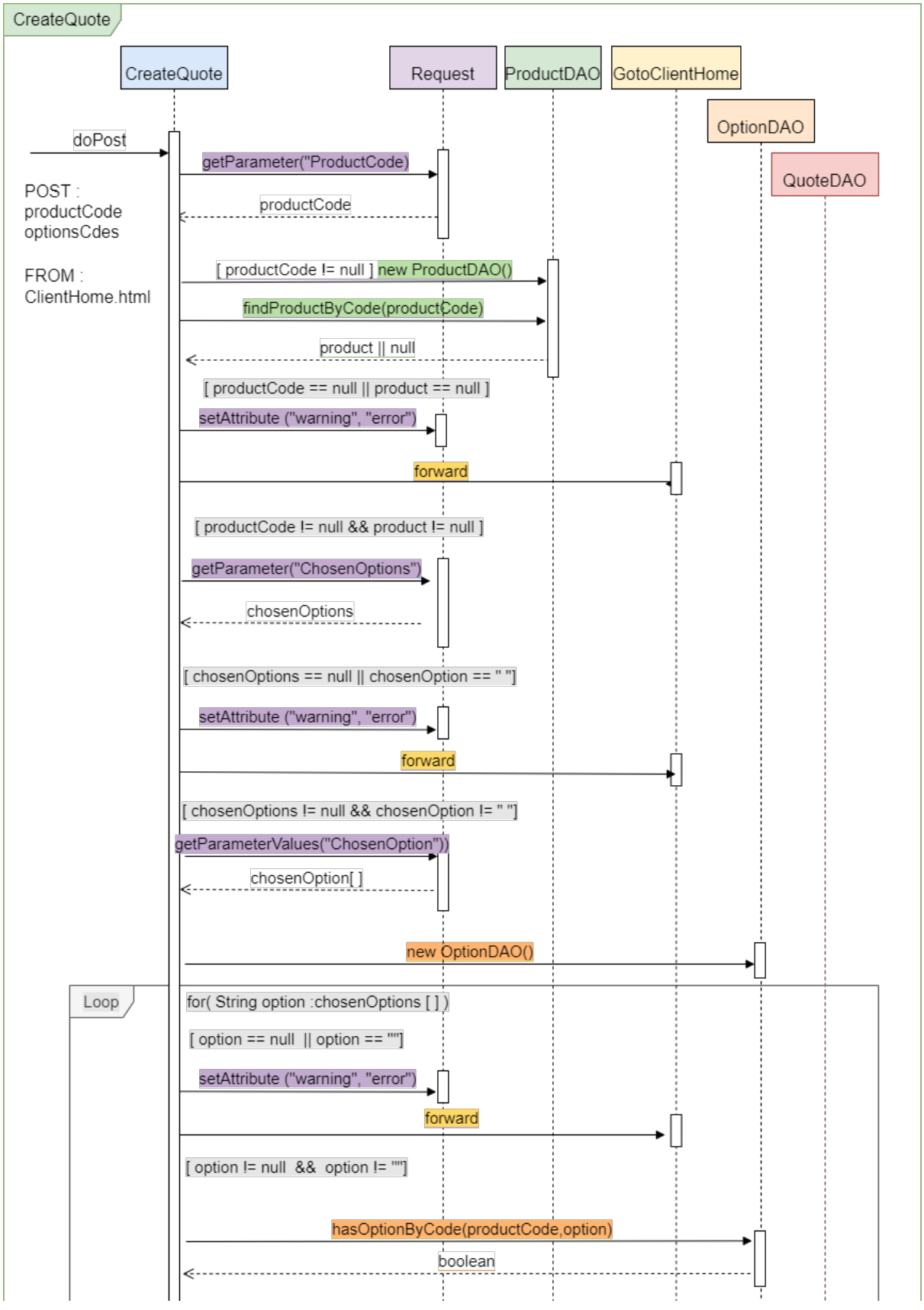
Figura 8: Event - Go to Quote Details

Controlli:
Server: Viene controllato che :

1. L'id del prodotto non sia nullo, vuoto, sia in un formato numerico e che corrisponda ad un id effetivamente presente nel DB.

2. Che lo user presente nella sessione abbia l'autorizzazione per visualizzare i dettagli del preventivo.

1.8.7 CreateQuote



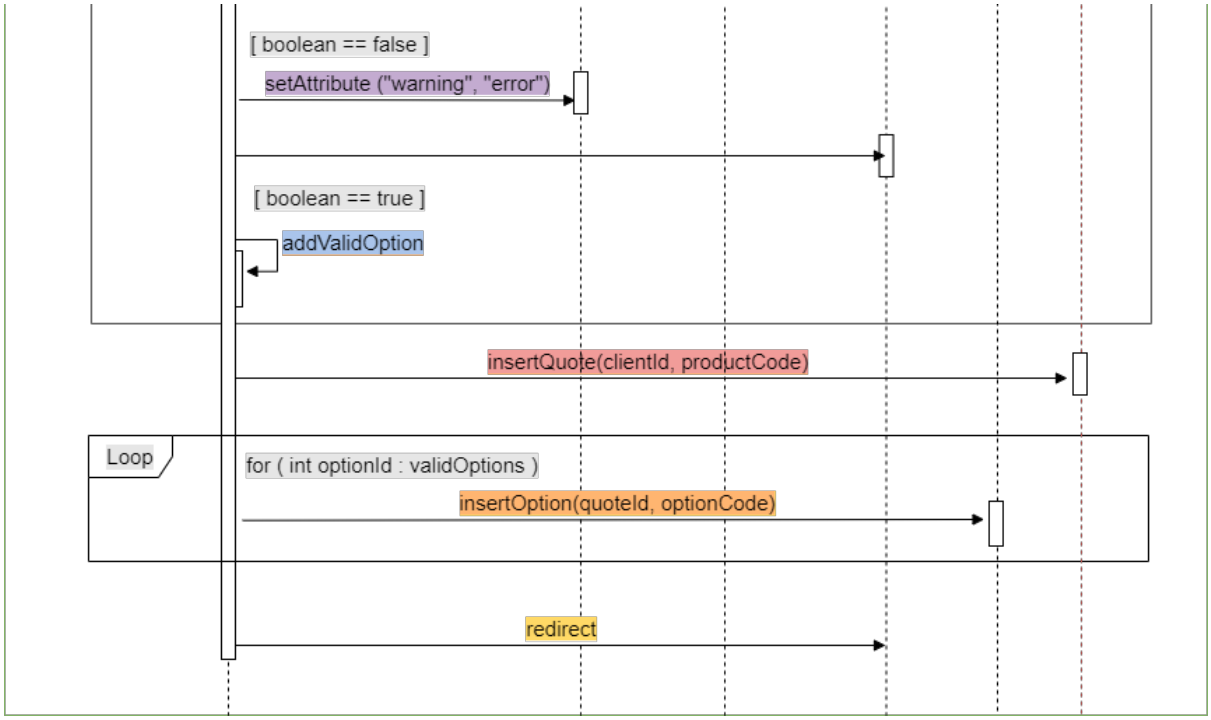


Figura 9: Event - Create Quote

Controlli:
HTML:
La scelta di almeno un'opzione per la creazione del preventivo è obbligatoria.
Server:
Viene controllato che :

1. L'id del prodotto non sia nullo, vuoto, sia in un formato numerico e che corrisponda ad un id effetivamente presente nel DB.

2. Il parametro `chosenOption` non sia nullo o vuoto. Inoltre viene controllato che ogni elemento dell'array non sia nullo, vuoto , sia in formato numerico e corrisponda effetivamente ad una opzione disponibile per il prodotto selezionato.

1.8.8 UpdatePrice

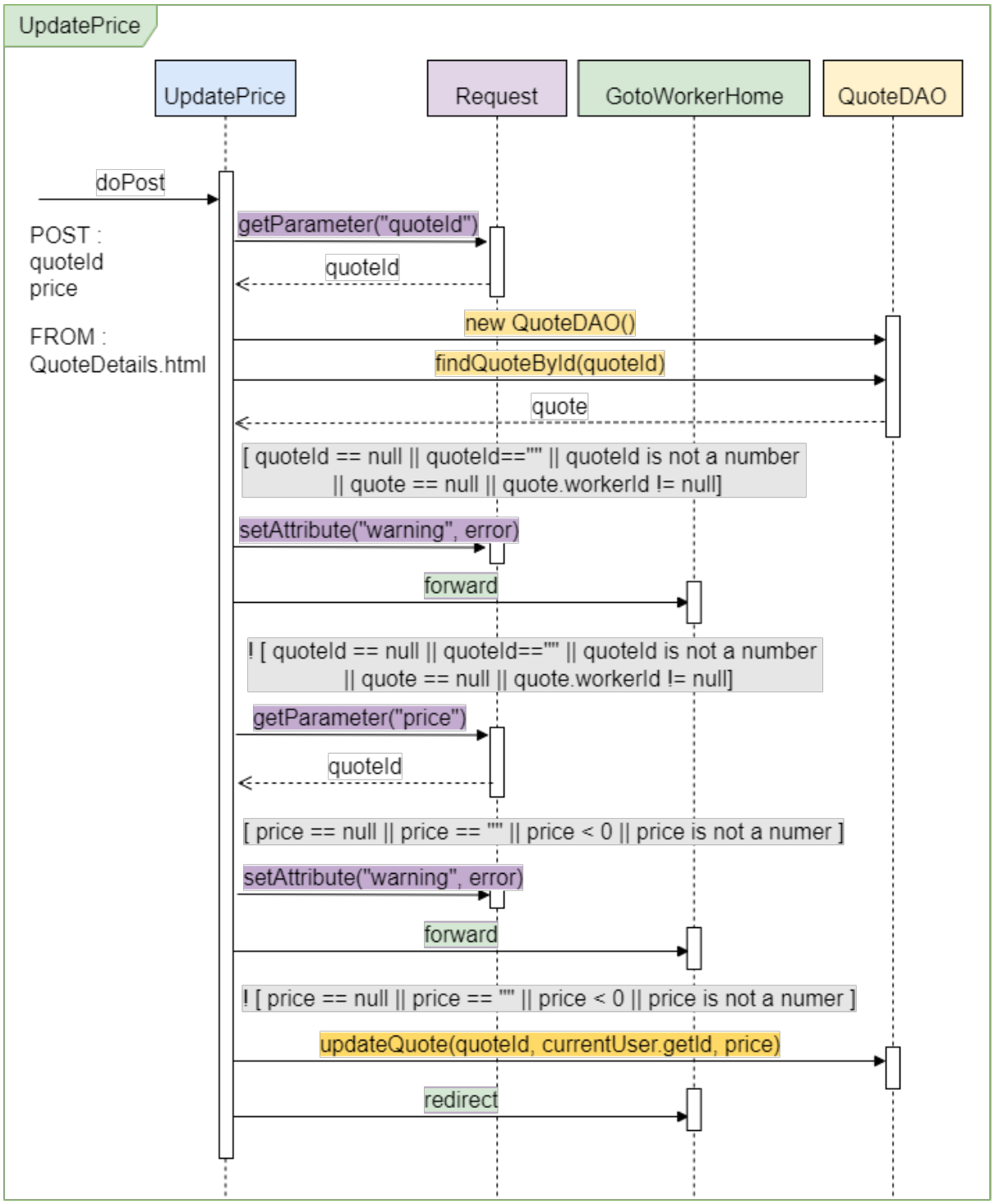


Figura 10: Event - Update Price

Controlli:
Server: Viene controllato che :

1. L'id del prodotto non sia nullo, vuoto, sia in un formato numerico e che corrisponda ad un id effetivamente presente nel DB.

2. Che il preventivo non sia già stato prezzato da un altro worker.
3. Che il prezzo inserito non sia nullo,vuoto, sia in un formato numerico e maggiore di 0.

2 Gestione Preventivi - RIA

2.1 Analisi dei dati per il database

Si realizzi un'applicazione client server web che modifica le specifiche precedenti (subsection 1.1) come segue:

- La registrazione controlla la validità sintattica dell'indirizzo di **email** e l'uguaglianza tra i campi "password" e "ripeti password", anche a lato client.
- 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.

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

2.2 Database Design

2.3 Local Database schema

2.4 Analisi dei dati per i requisiti dell'applicazione

Si realizzi un'applicazione client server web che modifica le specifiche precedenti (subsection 1.4) come segue:

- La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra **i campi "password" e "ripeti password"**, anche a lato client.
- 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.

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

2.5 Completamento delle specifiche

2.6 Sommario: Viste e componenti

- **Login Page**
  - **Login Form**
    - Email Input
    - Password Input
    - Login Button
    - Register Now Anchor
  - **Register Form**
    - Username Input
    - Email Input
    - Password Input
    - Repeat Password Input
    - Login Button
    - Go to Login Anchor
- **Client Page**
  - **CreateQuoteForm**
    - Products List
    - Options List
    - Request Quote Button
  - **QuotesList**
    - Quote Details
  - **Logout Button**
- **Worker Page**
  - **Managed Quotes List**

    - Quote Details
  - **Unmanaged Quotes List**
    - Quote Details
    - Price Input
    - Update Price Button
  - **Logout Button**

2.7 Application Desing

2.8 Eventi e azioni

Indirizzo di memoria	Valore binario	Valore decimale	Commento
0	00000011	3	Numero di parole in ingresso and and and a
1	01110000	112	Prima parola da codificare
2	10100100	164	Seconda parola da codificare
3	00101101	45	Terza parola da codificare
[...]			
1000	00111001	57	Codifica 1 della prima parola
1001	10110000	176	Codifica 2 della prima parola
1002	11010001	209	Codifica 1 della seconda parola
1003	11110111	247	Codifica 2 della seconda parola
1004	00001101	13	Codifica 1 della terza parola
1005	00101000	40	Codifica 2 della terza parola

Tabella 1: Descrizione della memoria

2.9 Server Side: DAO & Model Object

- **Model Objects (Beans)**
  - Option
  - Product
  - Quote
  - User
- **Data Access Objects (Classes)**
  - **OptionDAO**
    - boolean hasOptionByCode(int productCode, int optionCode);
    - List<Option> findOptionsByProductCode(int productCode);
    - List<Option> findOptionsByQuoteId(int quoteId);
    - void insertOption(int quoteId, int optionCode);
  - **ProductDAO**

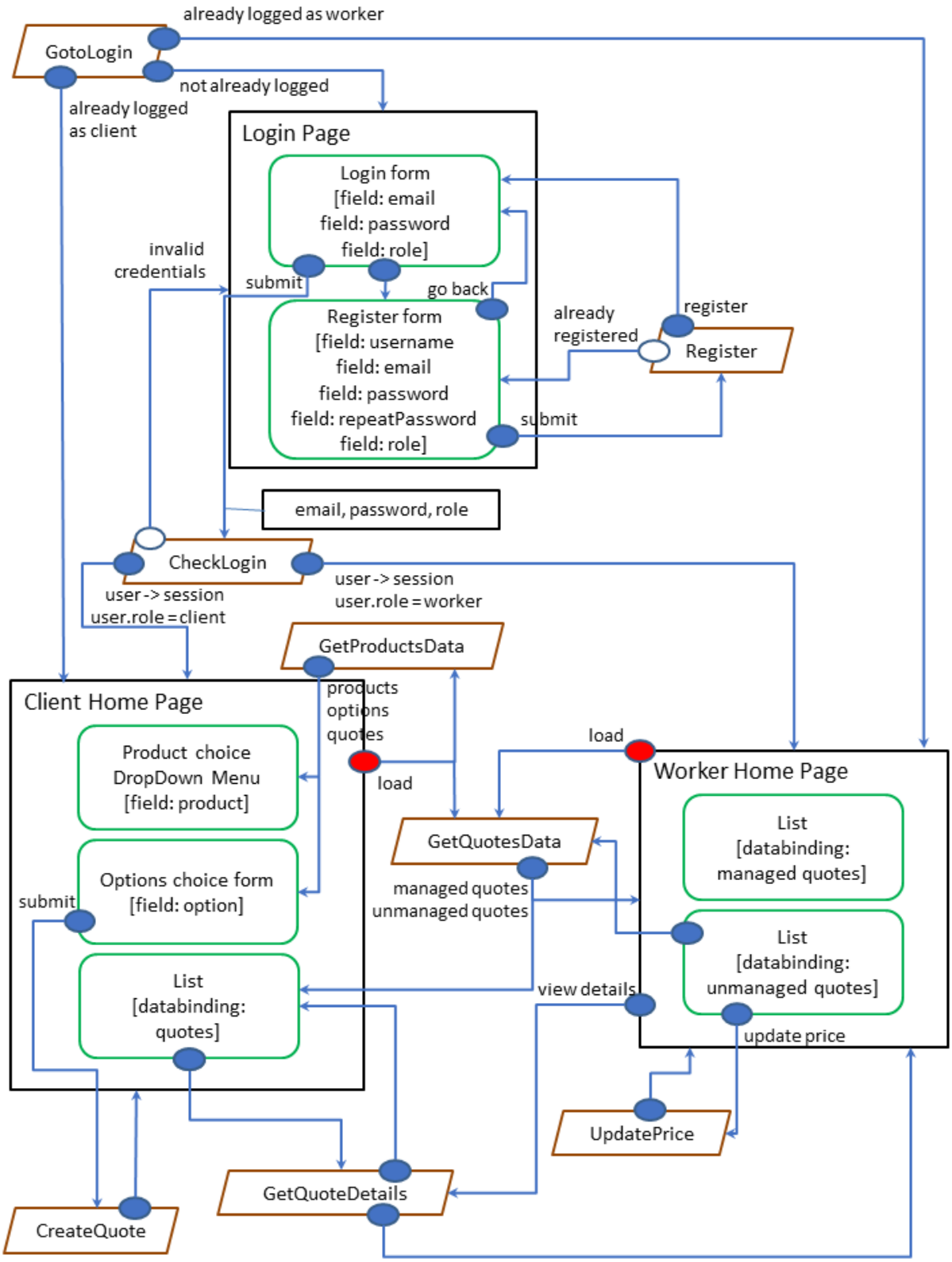


Figura 11: Event - IFML Diagram

* List<Product> findAllProducts();
* Product findProductByCode(int code);

  - **QuoteDAO**
    * List<Quote> findQuotesByUserId(int userId, String role);
    * Quote findQuoteById(int quoteId);
    * List<Quote> findUnmanagedQuotes();
    * int insertQuote(int clientId, int productCode);
    * void updateQuote(int quoteId, int workerId, int price);
  - **UserDAO**
    * User findUser(String email, String password, String role);
    * User findUserToRegister(String username, String email, String role);
    * User findClientById(int clientId);
    * void registerUser(String username, String password, String role);

- **Controllers (Servlets)**
  - GotoLogin
  - CheckLogin
  - Register
  - GetProductsData
  - GetQuotesData
  - CreateQuote
  - GetQuoteDetails
  - UpdatePrice
  - Logout

- **Filters**
  - SessionChecker
  - ClientChecker
  - WorkerChecker

- **Views (Templates)**
  - Login.html
  - ClientHome.html
  - WorkerHome.html

2.10 Client Side: view & view components

- **LoginManagement.js**
  - returnToLogin(): modifica gli elementi della form per passare dalla Register form alla Login form;
  - checkEmail(email): verifica validità email;
  - loginButton.addEventListener

  - openRegister.addEventListener

- **PageHandler.js**
  - **OptionsList**
    * reset(): rende invisibile le opzioni;
    * clear(): toglie dalla select tutte le options;
    * show(): visualizza le opzioni relative al prodotto selezionato;
    * createQuote(): verifica la validità degli input della form - registra il preventivo;
    * checkChosenOptions (): verifica validità opzioni scelte;
  - **CreateQuoteForm**
    * getDropDownButton(): restituisce il bottone che gestisce la scelta dei prodotti;
    * show(): richiede al server i prodotti disponibili;
    * update(): aggiorna la lista dei prodotti da visualizzare;
  - **QuoteList**
    * show(): richiede al server la lista dei preventivi richiesti dal client/gestiti dal Worker/non gestiti;
    * update(): aggiorna la lista dei preventivi;
    * addDetails(): richiede al server i dettagli del preventivo selezionato;
    * updateDetails(): visualizza i dettagli del preventivo;
    * updatePrice(): verifica la validità dell'input - aggiorna il prezzo del preventivo;
    * checkCorrectPrice(): verifica l'input - mostra un warning in caso di errore;
    * clear(): rimuove tutti i preventivi dalla lista;
  - **PageHandler**
    * start(): crea e inizializza i componenti dell'interfaccia;

2.11 Events

2.11.1 Login

2.11.2 Register

2.11.3 View Options

2.11.4 Create Quote

2.11.5 Update Price

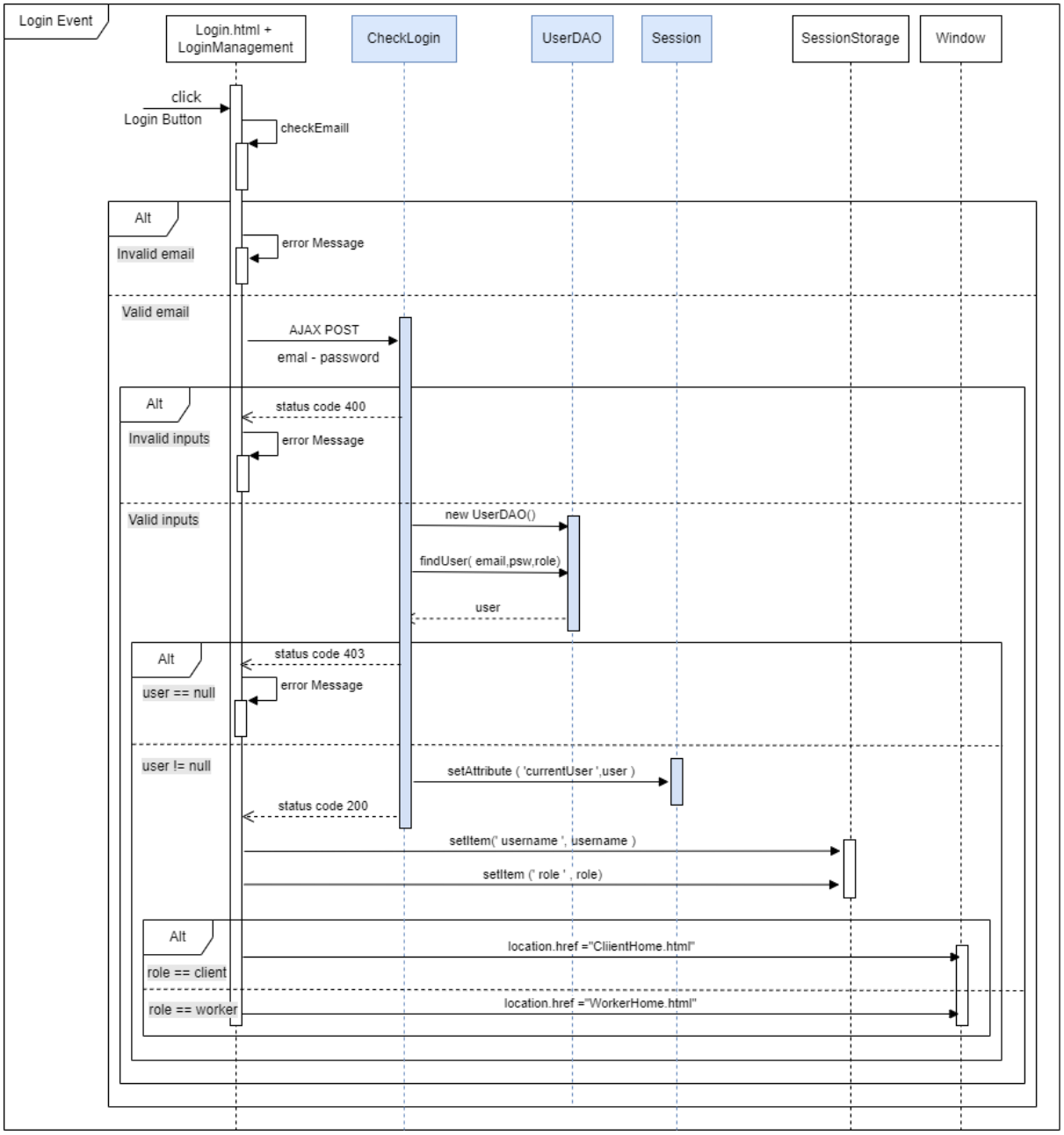


Figura 12: Event - Login

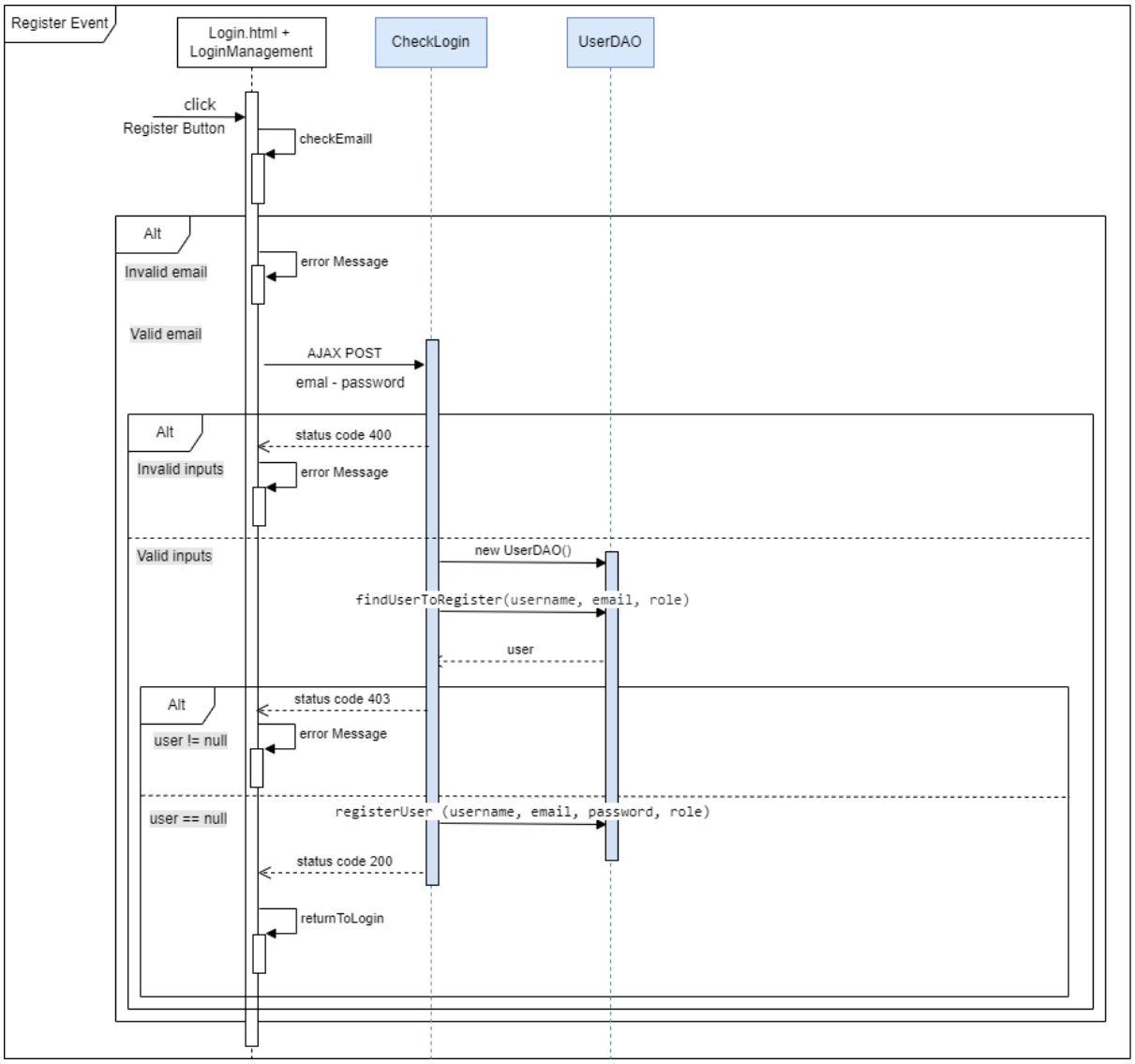


Figura 13: Event - Register

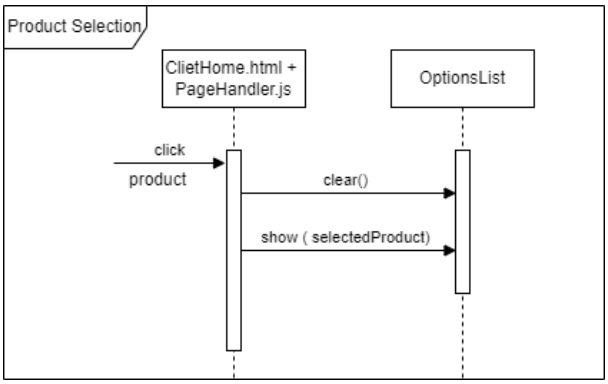


Figura 14: Event - View Options

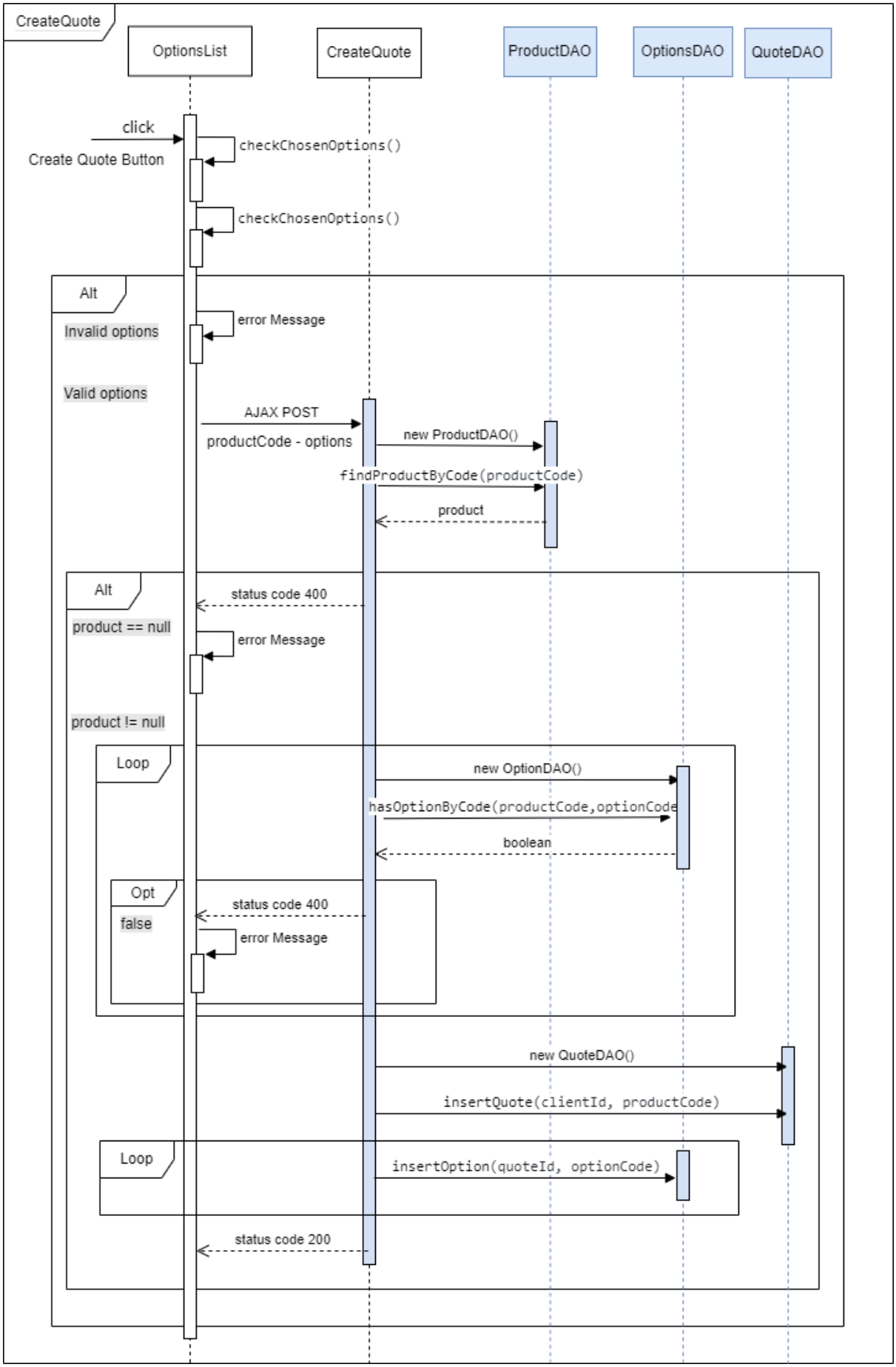


Figura 15: Event - Create Quote

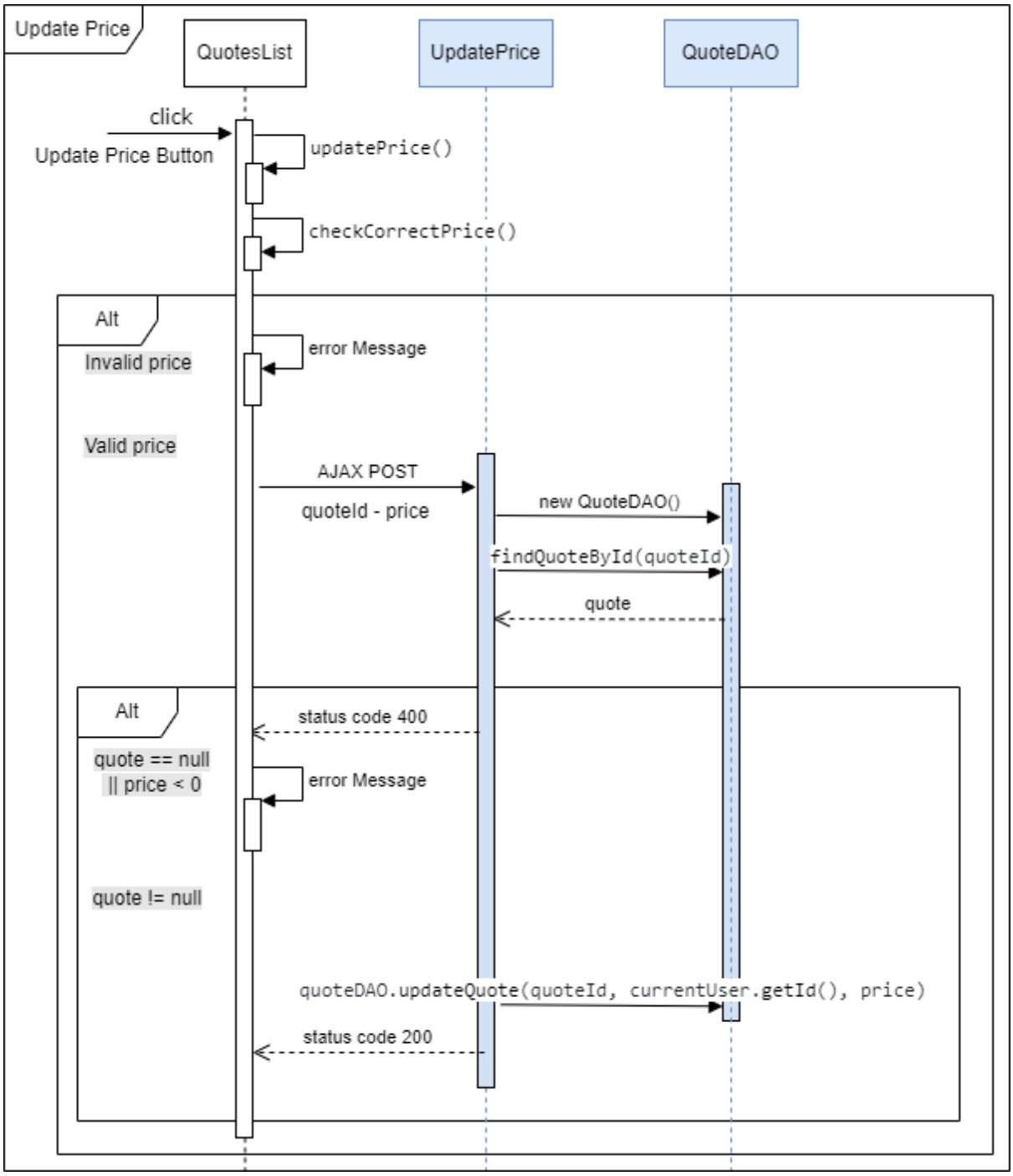


Figura 16: Event - Update Price