Gestione preventivi: versione pure HTML

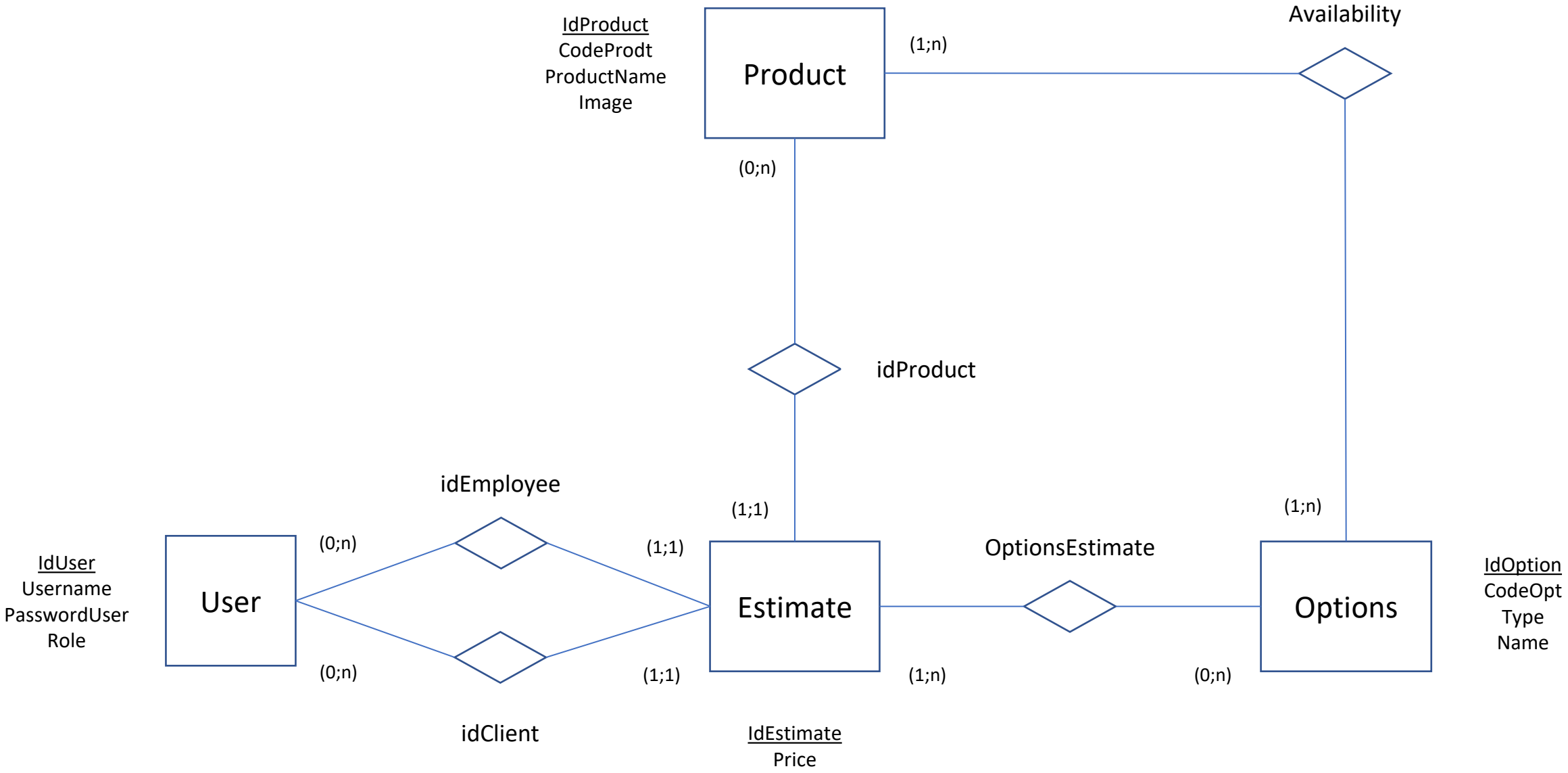
Andrea Lazzarini, Luca Muroni – Gruppo 33

Analisi dei dati

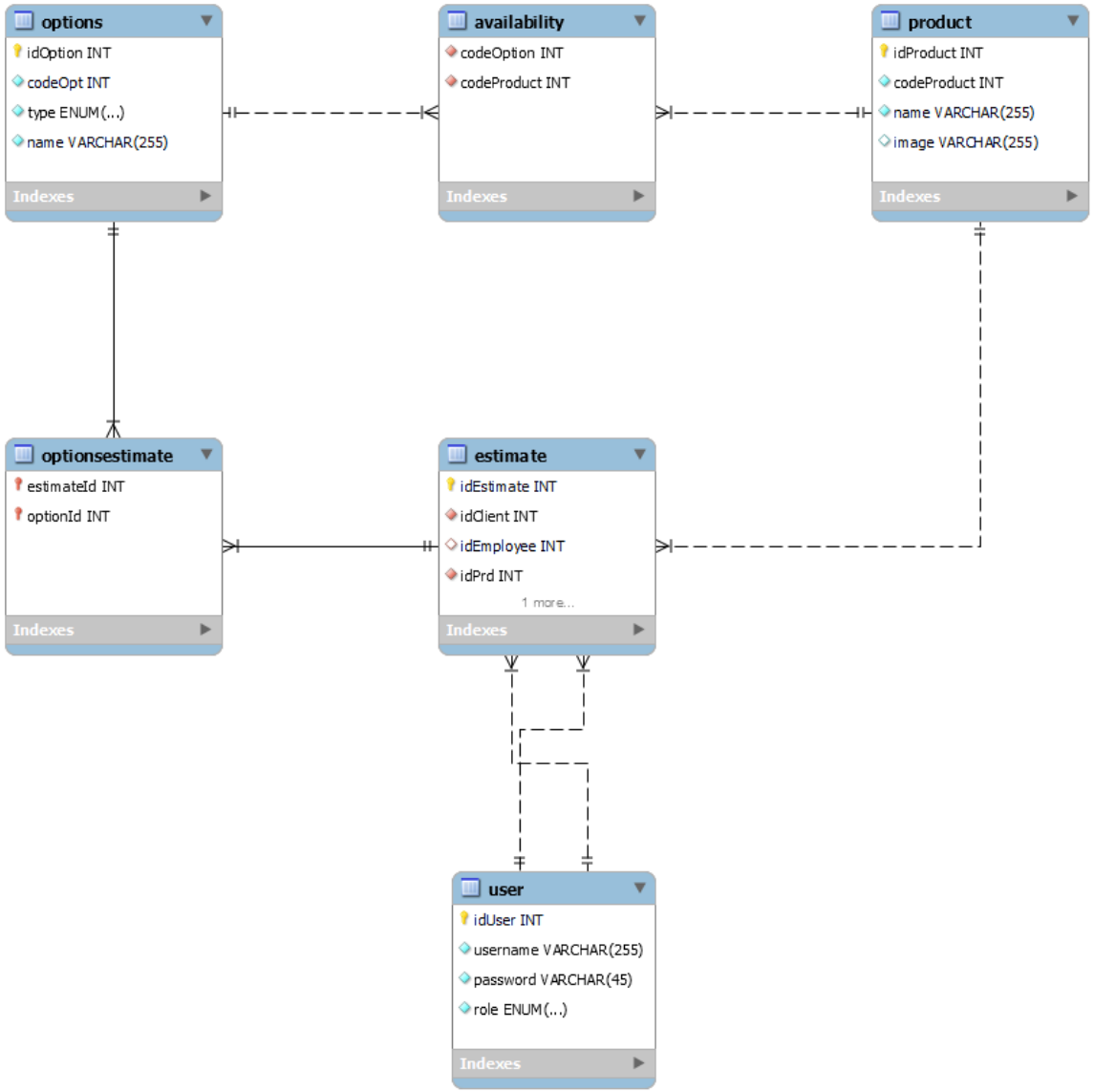
Un’applicazione web consente la gestione di richieste di preventivi per prodotti personalizzati. L’applicazione supporta registrazione e login di clienti e impiegati mediante una pagina pubblica con opportune form. La registrazione controlla l’unicità dello username. Un preventivo è associato a un prodotto, 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

Database Design



Diagram


options
idOption INT
codeOpt INT
type ENUM(...)
name VARCHAR(255)
Indexes
availability
codeOption INT
codeProduct INT
Indexes
product
idProduct INT
codeProduct INT
name VARCHAR(255)
image VARCHAR(255)
Indexes
optionsestimate
estimateId INT
optionId INT
Indexes
estimate
idEstimate INT
idClient INT
idEmployee INT
idPrd INT
1 more...
Indexes
user
idUser INT
username VARCHAR(255)
password VARCHAR(45)
role ENUM(...)
Indexes

Local Database Schema

```
CREATE TABLE `options` (

`idOption` int NOT NULL AUTO_INCREMENT,
`codeOpt` int NOT NULL UNIQUE,
`type` enum("normal", "on_sale") NOT NULL,
`name` varchar(255) NOT NULL,
PRIMARY KEY (`idOption`)

)
```

```
CREATE TABLE `product` (

`idProduct` int NOT NULL AUTO_INCREMENT,
`codeProduct` int NOT NULL UNIQUE,
`name` varchar(255) NOT NULL,
`image` varchar(255),
PRIMARY KEY (`idProduct`)

)
```

```
CREATE TABLE `user` (

`idUser` int NOT NULL AUTO_INCREMENT,
`username` varchar(255) NOT NULL UNIQUE,
`password` varchar(45) NOT NULL,
`role` enum('employee', 'client') NOT NULL,
PRIMARY KEY (`idUser`)

)
```

```
CREATE TABLE `estimate` (

`idEstimate` int NOT NULL AUTO_INCREMENT,
`idClient` int NOT NULL,
`idEmployee` int,
`idPrd` int NOT NULL,
`price` float DEFAULT -1,
PRIMARY KEY (`idEstimate`),
KEY `client` (`idClient`),  CONSTRAINT `client` FOREIGN KEY (`idClient`)
REFERENCES `user` (`idUser`) ON DELETE NO ACTION ON UPDATE
CASCADE,
KEY `employee` (`idEmployee`),  CONSTRAINT `employee` FOREIGN KEY
(`idEmployee`) REFERENCES `user` (`idUser`) ON DELETE NO ACTION ON
UPDATE CASCADE,
KEY `prd` (`idPrd`),  CONSTRAINT `prd` FOREIGN KEY (`idPrd`) REFERENCES
`product` (`idProduct`) ON DELETE NO ACTION ON UPDATE CASCADE

)
```

Local Database Schema

```
CREATE TABLE `availability` (

`codeOption` int NOT NULL,
`codeProduct` int NOT NULL,
PRIMARY KEY (`codeOption`,`codeProduct`),
KEY `opt` (`codeOption`),  CONSTRAINT `opt` FOREIGN KEY (`codeOption`)
REFERENCES `options` (`idOption`) ON DELETE NO ACTION ON UPDATE CASCADE,
KEY `prod` (`codeProduct`),  CONSTRAINT `prod` FOREIGN KEY (`codeProduct`)
REFERENCES `product` (`idProduct`) ON DELETE NO ACTION ON UPDATE CASCADE

)
```

```
CREATE TABLE `product` (

`idProduct` int NOT NULL AUTO_INCREMENT,
`codeProduct` int NOT NULL UNIQUE,
`name` varchar(255) NOT NULL,
`image` varchar(255),
PRIMARY KEY (`idProduct`)

)
```

Application requirements analysis

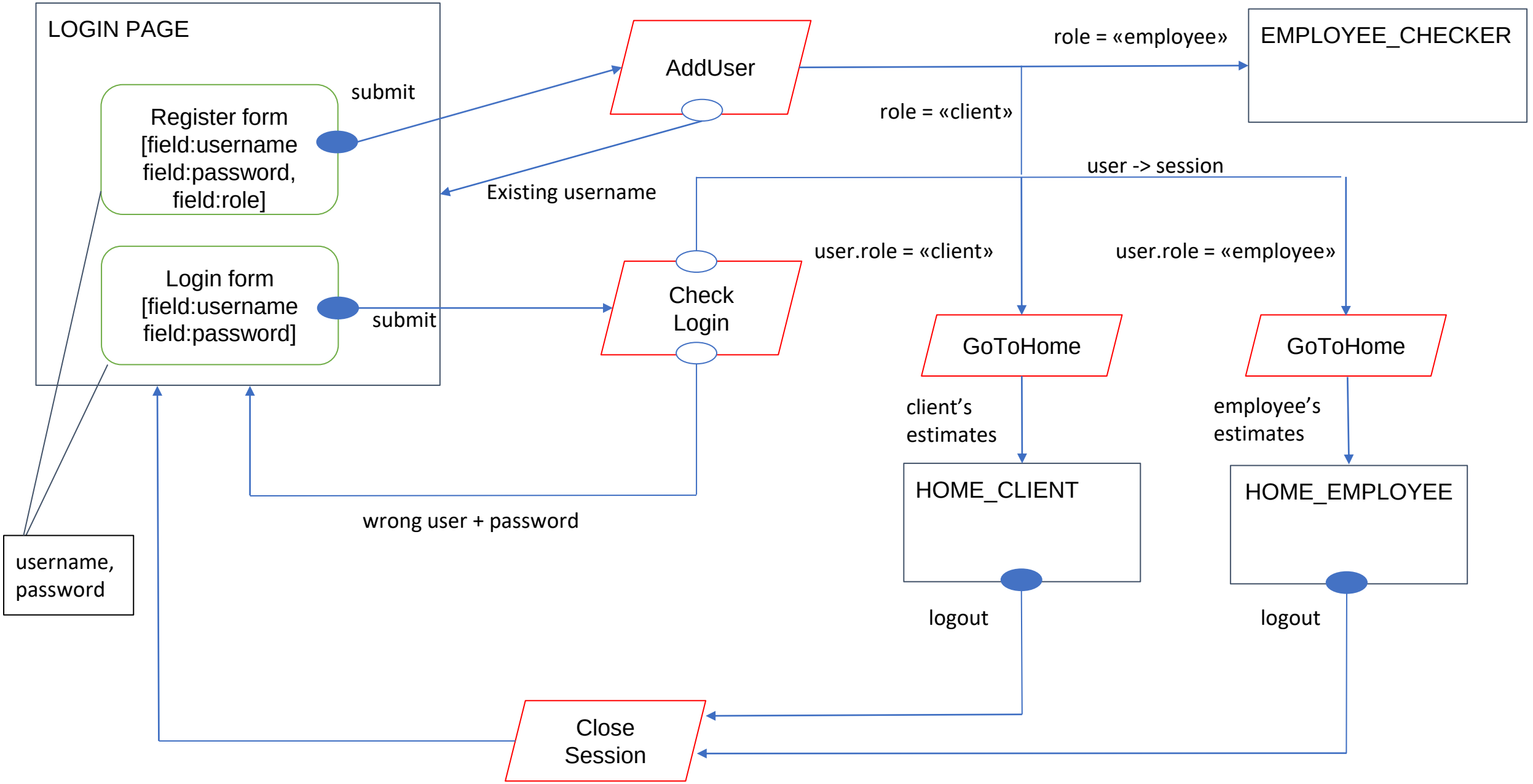
Un’applicazione web consente la gestione di richieste di preventivi per prodotti personalizzati. L’applicazione supporta registrazione e login di clienti e impiegati mediante una pagina pubblica con opportune form. La registrazione controlla l’unicità dello username. Un preventivo è associato a un prodotto, 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), view components, events, actions

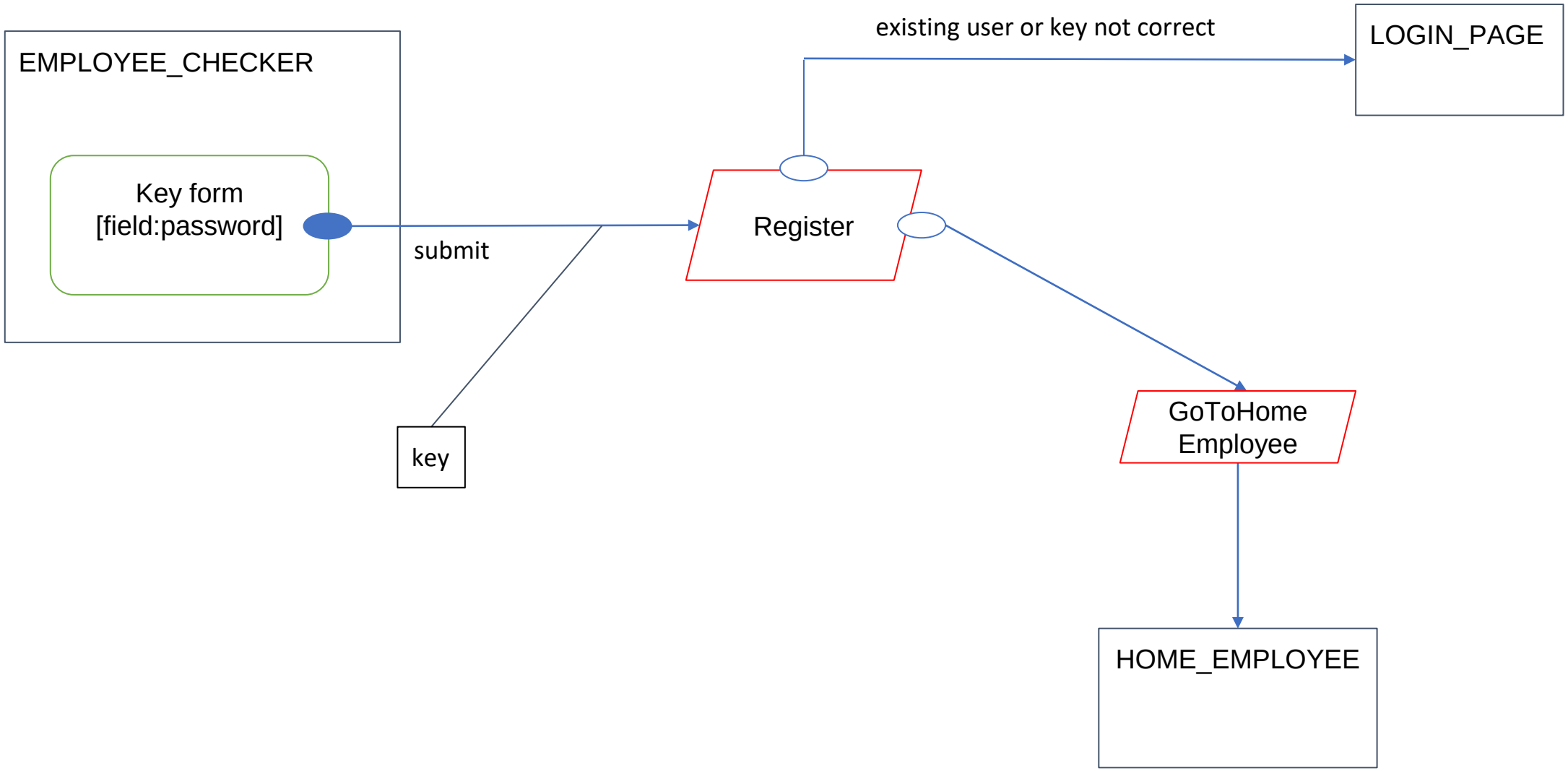
Completamento delle specifiche

- Lo username deve essere una mail
- La pagina LOGIN PAGE contiene anche la form di registrazione per l'utente
- Un utente che vuole registrarsi come employee deve prima passare per una pagina CONTROLLA PASSWORD, la quale richiede, attraverso una form, all'utente di inserire una password segreta nota solo a chi effettivamente può registrarsi come employee
- Un preventivo non può avere un prezzo negativo o uguale a 0
- A seguito del login, riconoscere l'utente e mostrare un messaggio di saluto con il suo nome
- Nella form di creazione di un preventivo, dopo la scelta del prodotto viene visualizzata anche la sua immagine durante la scelta delle opzioni
- I dettagli dei preventivi vengono visualizzati in una pagina ESTIMATE_DETAILS

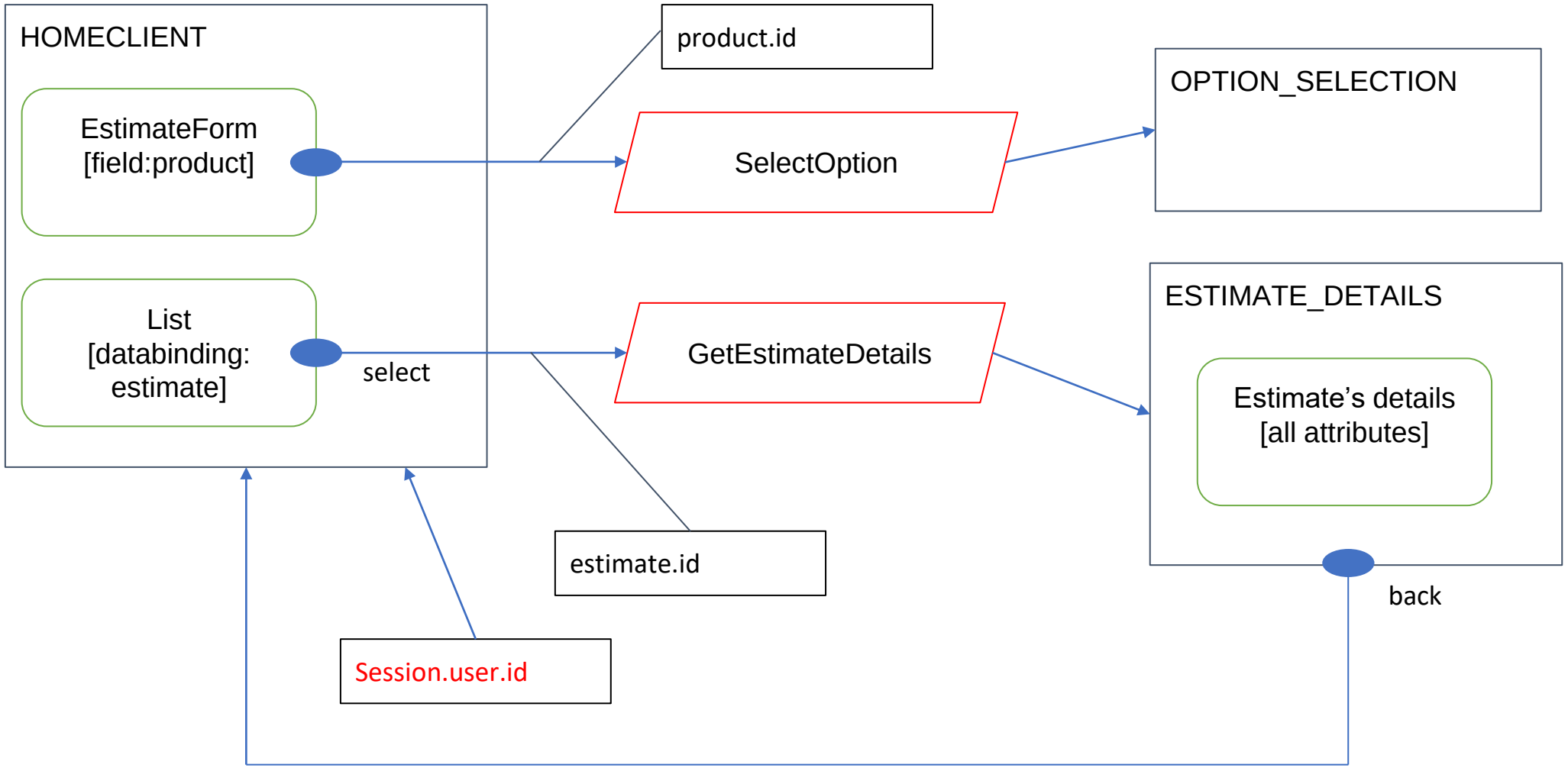
Application design



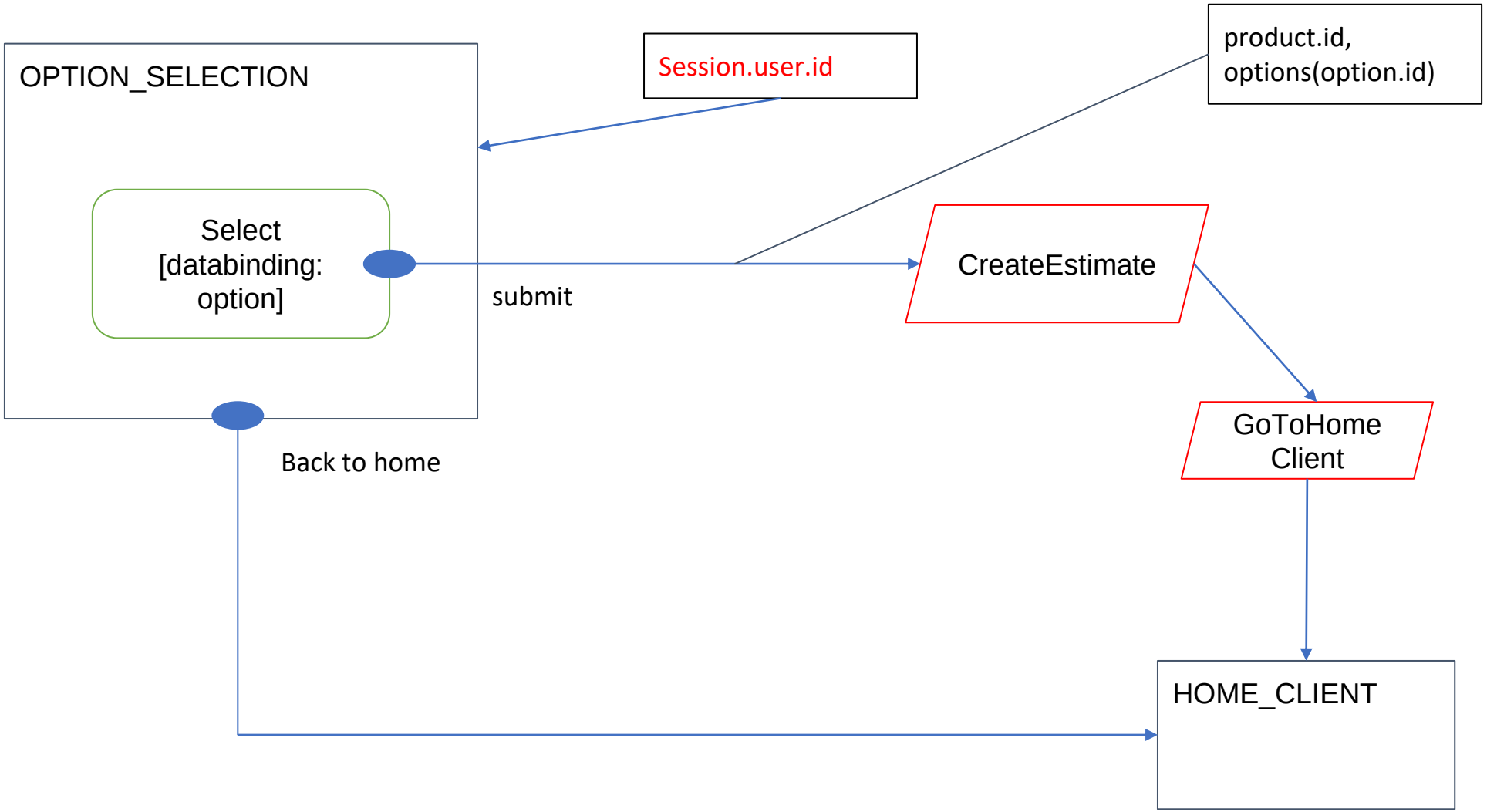
Application design



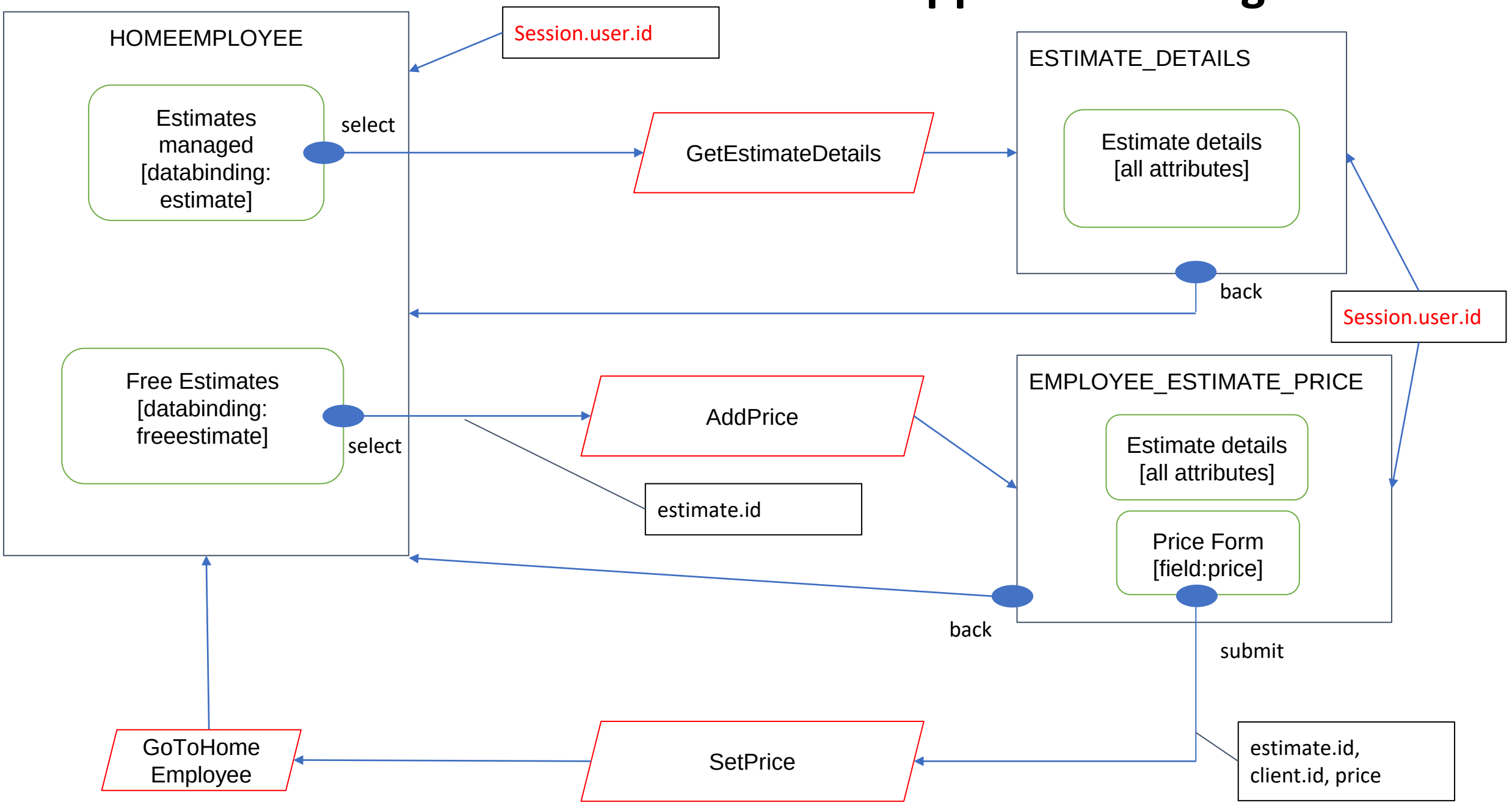
Application design



Application design



Application design



Components

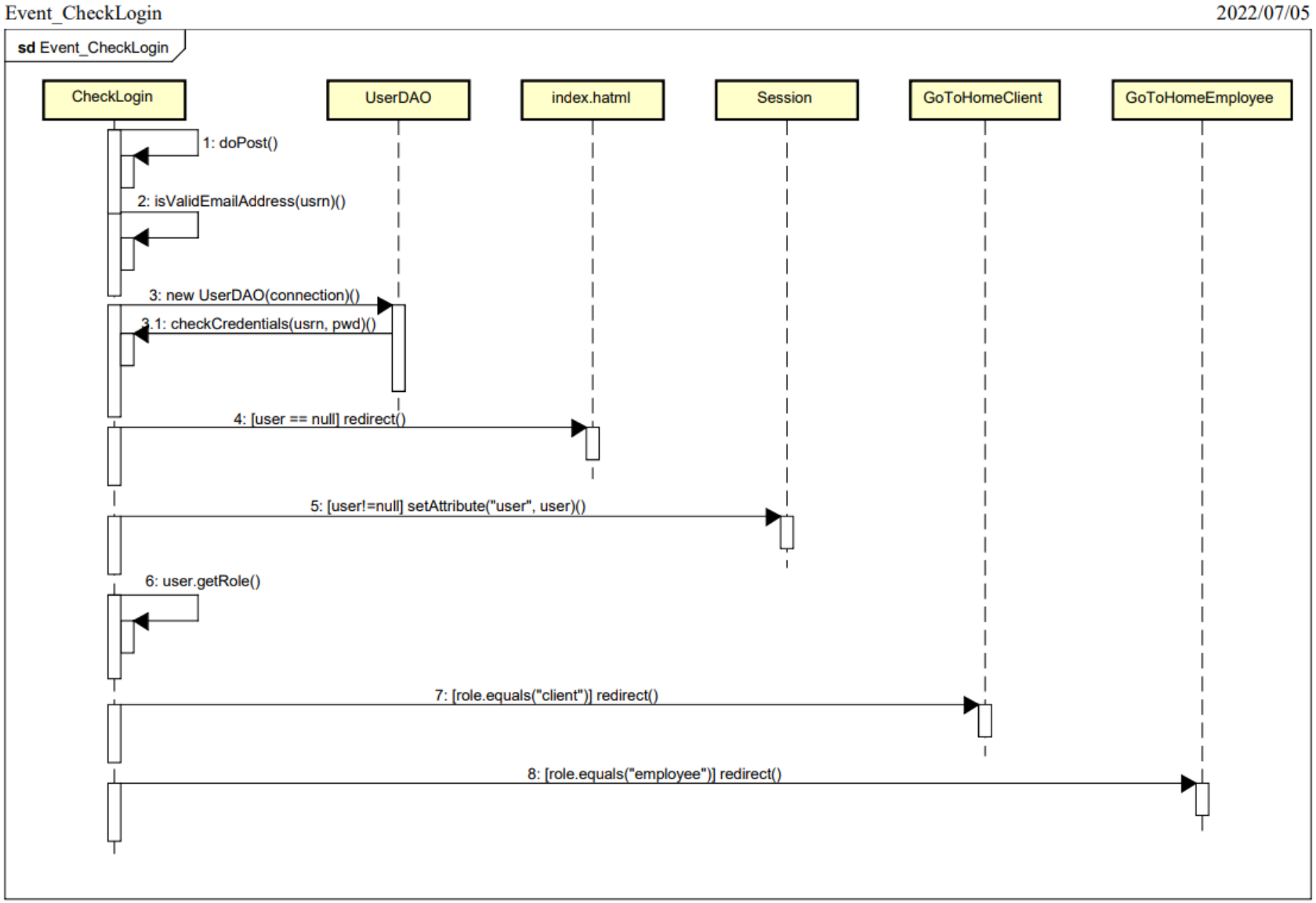
- Model objects (Beans)
  - User
  - Option
  - Estimate
  - Product
- Data Access Objects (Classes)
  - EstimateDAO
    - findEstimateByClientId(clientId)
    - findEstimateById(estimateId)
    - findProductNameByEstimate(estimateId)
    - addPriceEmployee(employeeId, price, estimateId)
    - findFreeEstimates()
    - findEstimatesByEmployeeId(employeeId)
    - createEstimate(productId, clientId)
    - searchLastEstimate(clientId)
  - AvailabiltityDAO
    - findProductOption(idProd)
  - OptionsEstimateDAO
    - addOptionsToEstimate(idEst, optionsId[])
    - findEstimateOptions(idEst)
- Data Access Objects (Classes)
  - UserDAO
    - checkCredentials(usrn, pwd)
    - checkUsername(usrn)
    - getUsername(idUser)
    - addCredentials(usrn, pwd, role)
    - getUsers(role)
  - ProductDAO
    - getAllProducts()
    - findProductById(prodId)
    - getName(idProduct)

Components

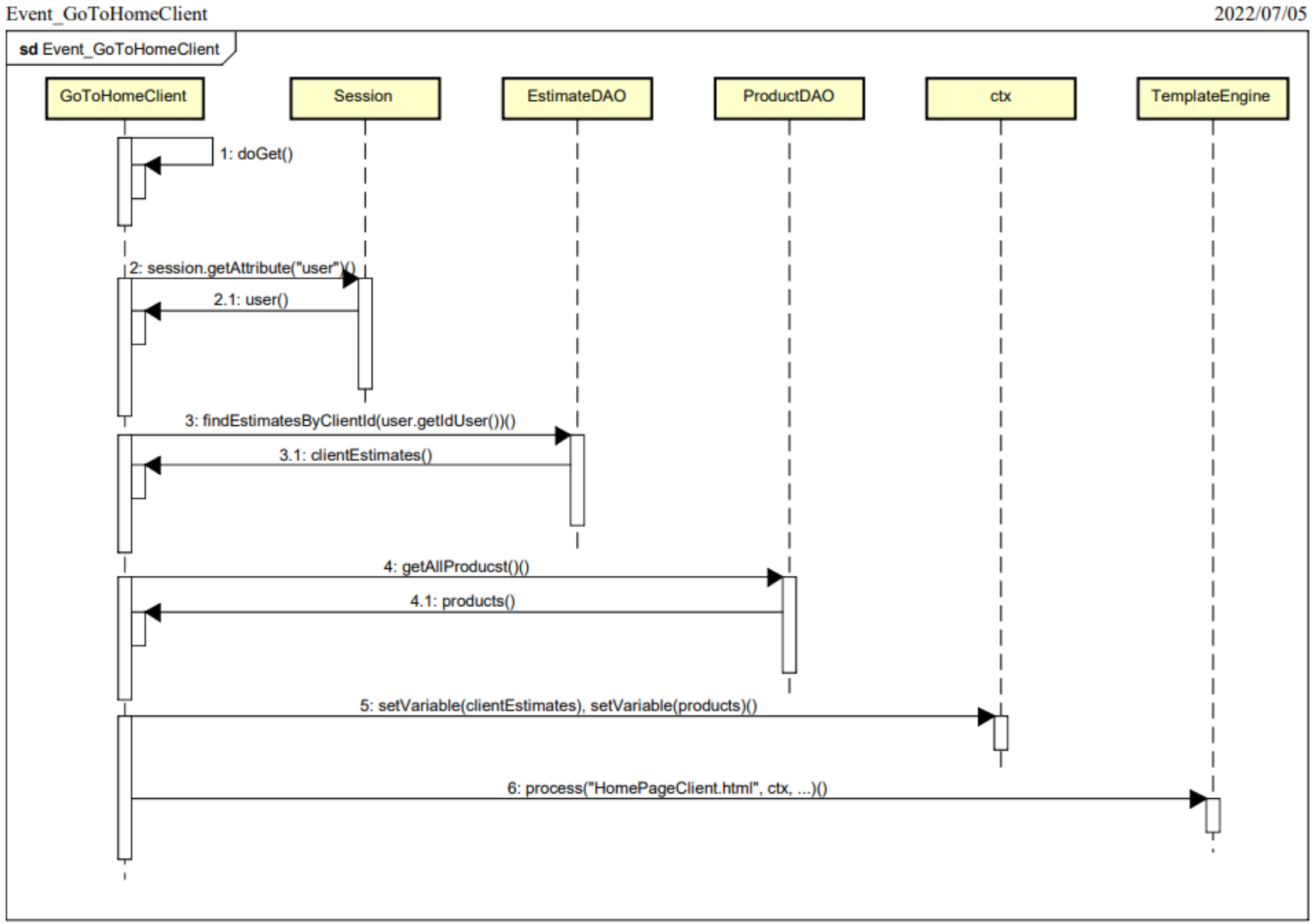
- Controllers (servlets)
  - CheckLogin
  - GoToHomeClient
  - GoToHomeEmployee
  - GetEstimateDetails
  - AddPrice
  - AddUser
  - CreateEstimate
  - Register
  - SelectOption
  - SetPrice
  - Logout

- Views (Templates)
  - EmployeeCecker
  - EmployeeEstimatePrice
  - EstimateDetails
  - HomePageClient
  - HomePageEmployee
  - OptionSelection

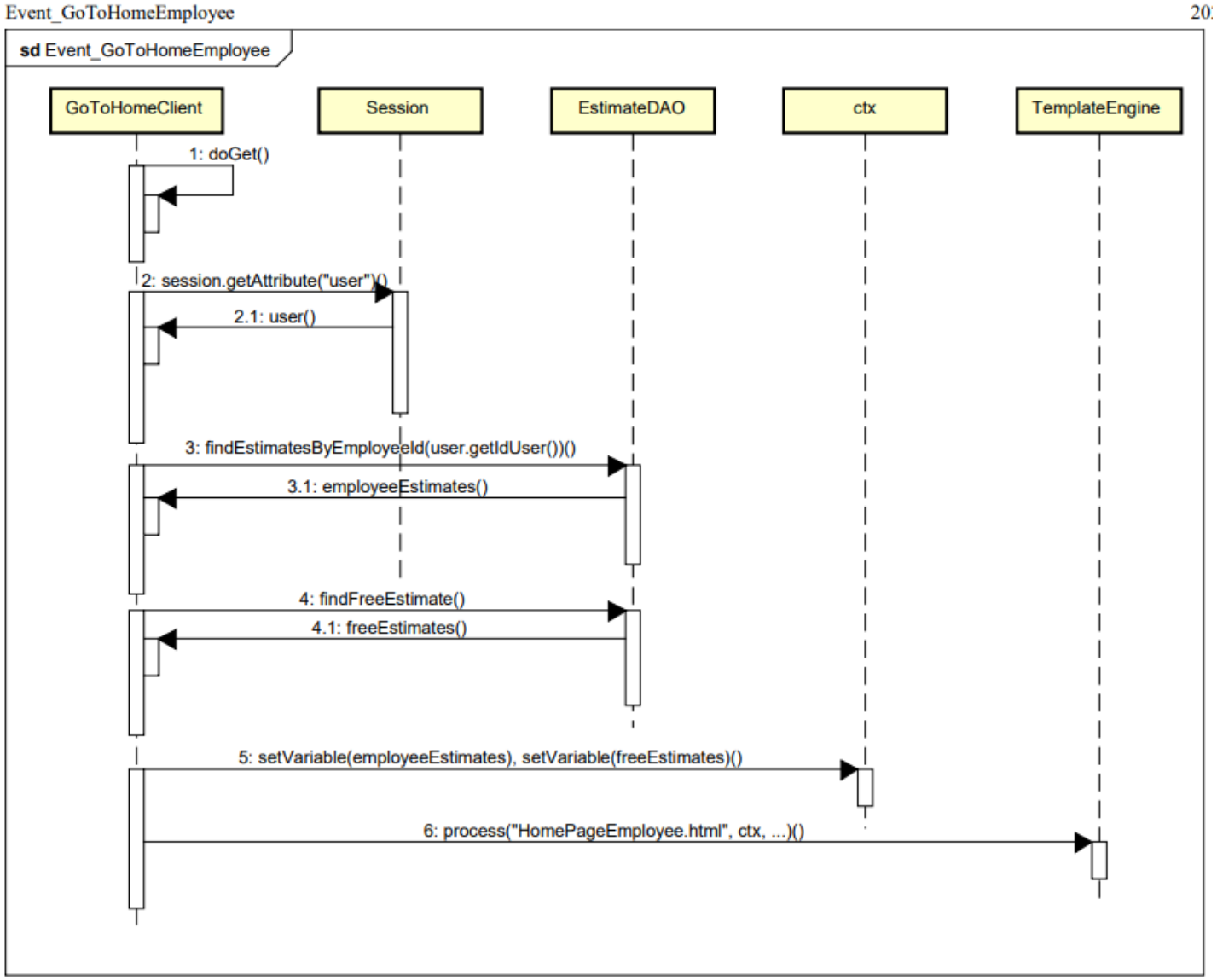
Event: login


Event_CheckLogin
2022/07/05
sd Event_CheckLogin
CheckLogin
UserDAO
index.hatml
Session
GoToHomeClient
GoToHomeEmployee
1: doPost()
2: isValidEmailAddress(usrn)()
3: new UserDAO(connection)()
3.1: checkCredentials(usrn, pwd)()
4: [user == null] redirect()
5: [user!=null] setAttribute("user", user)()
6: user.getRole()
7: [role.equals("client")] redirect()
8: [role.equals("employee")] redirect()

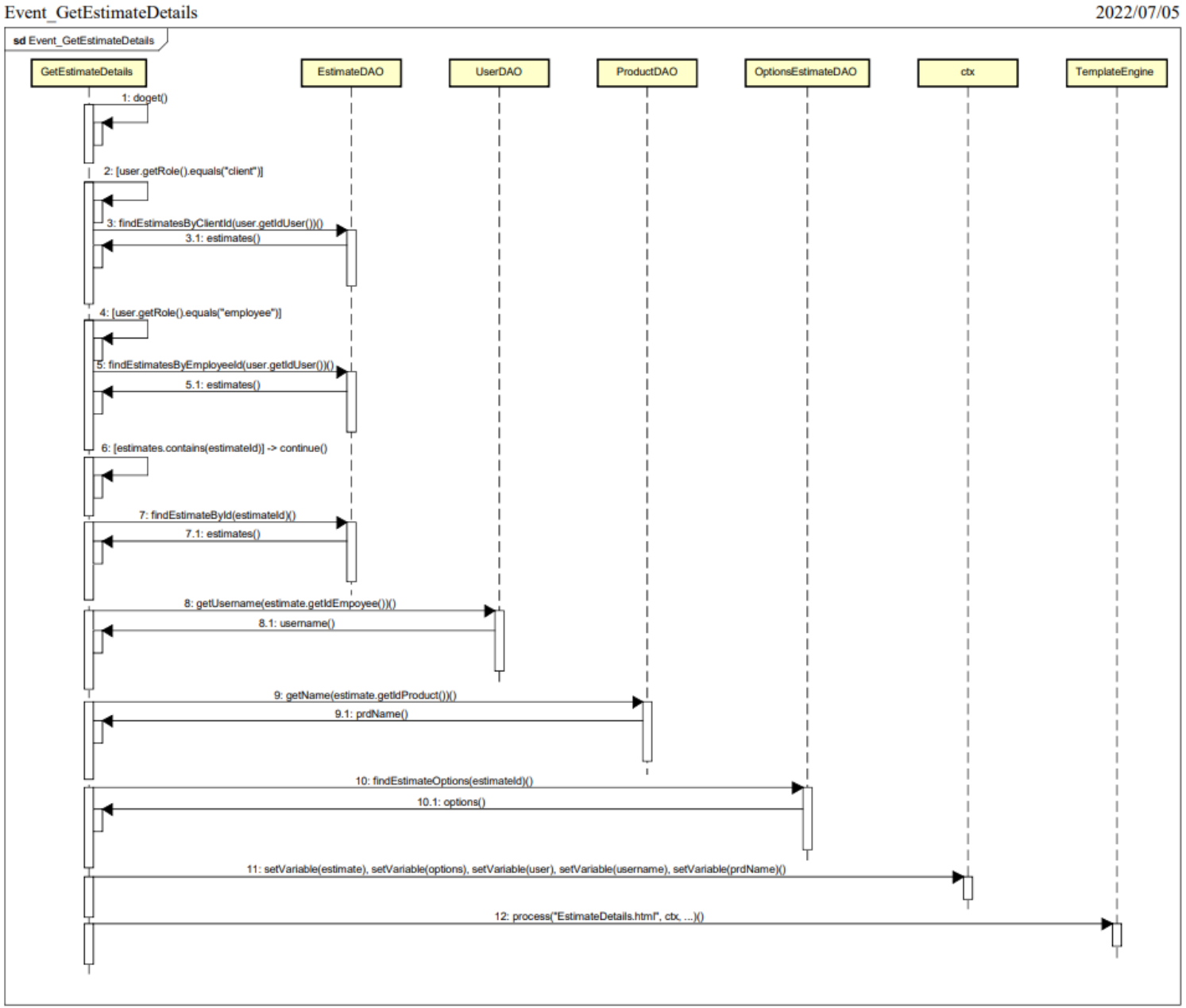
Event: GoToHomeClient



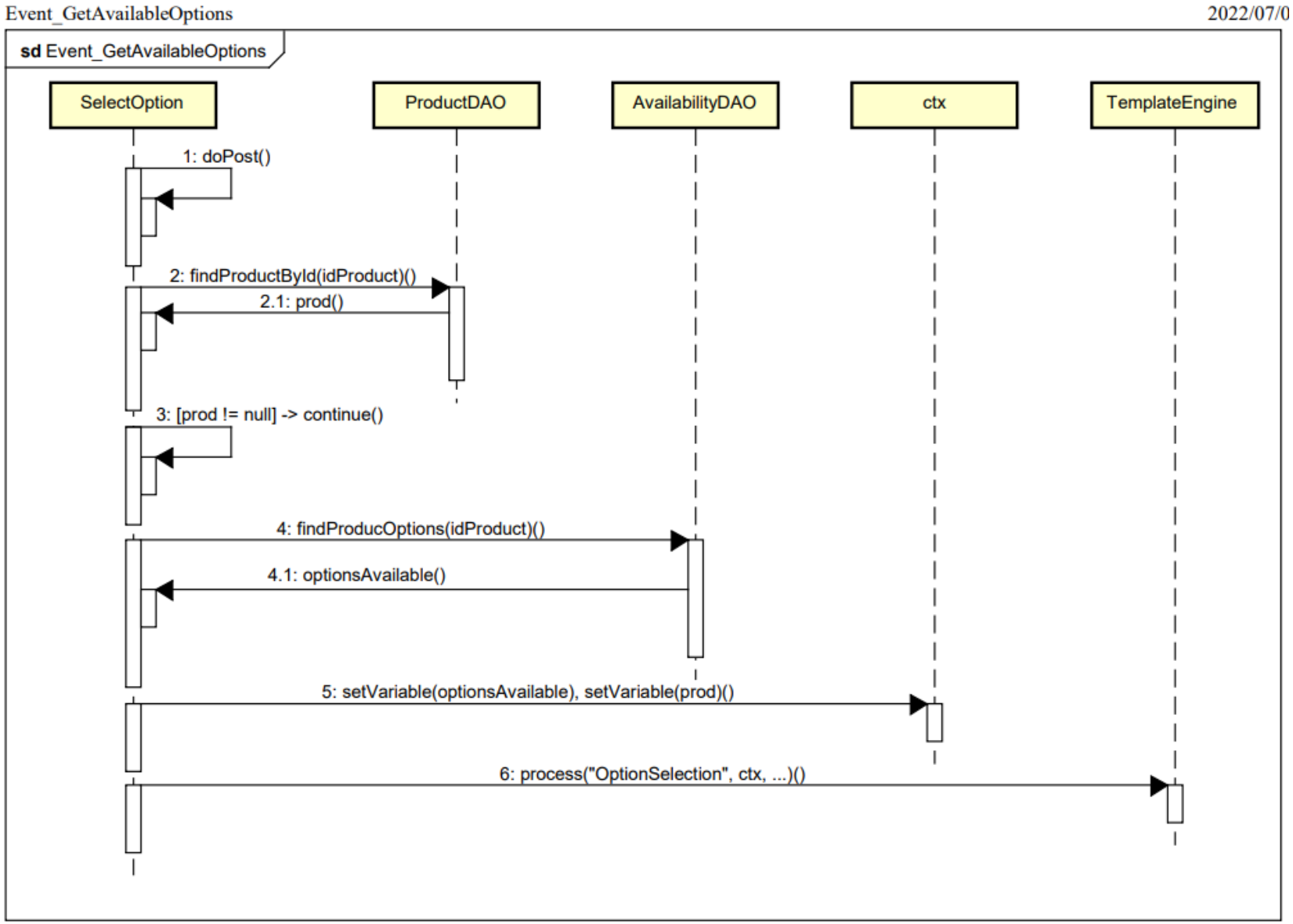
Event: GoToHomeEmployee



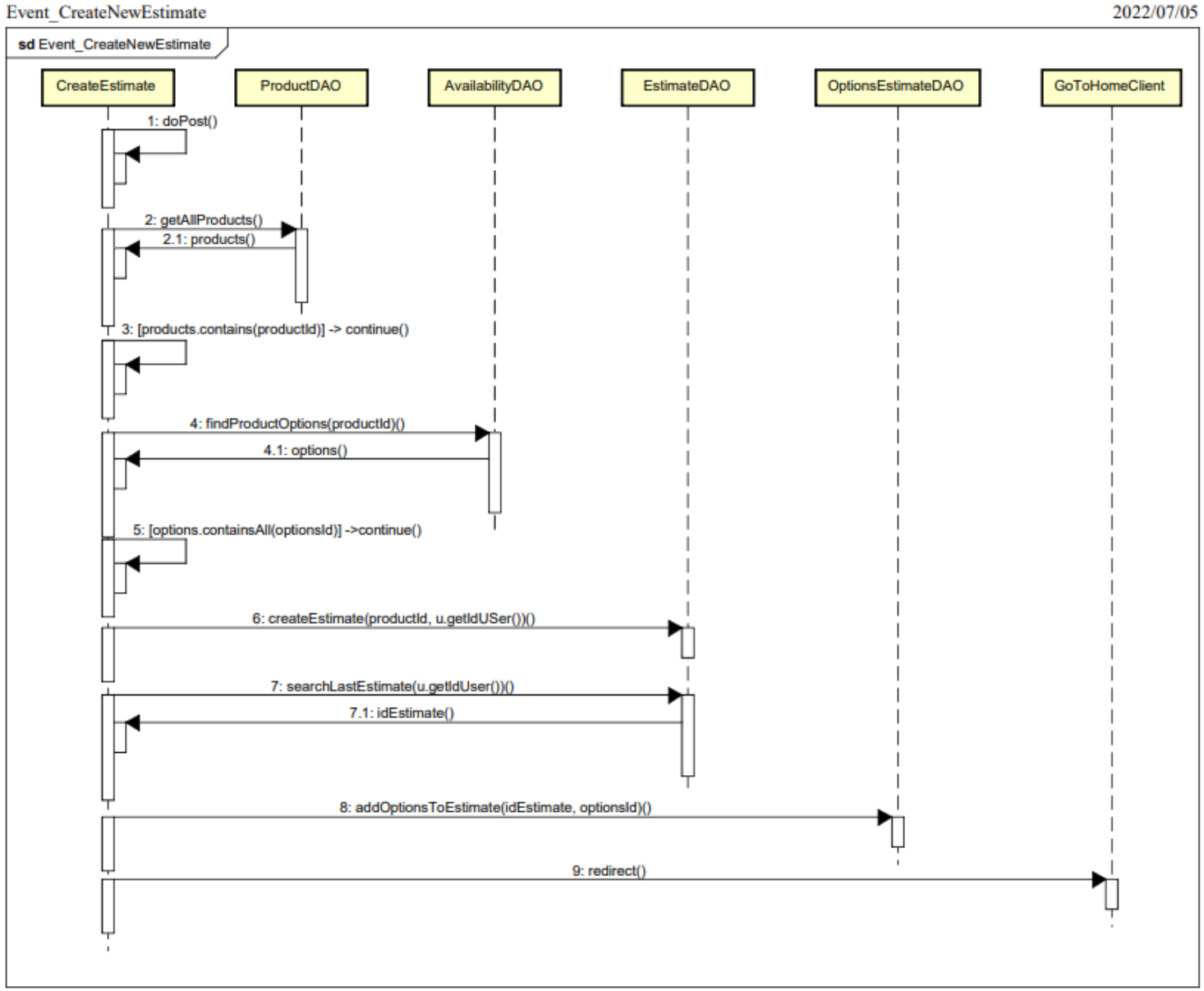
Event: GetEstimateDetails



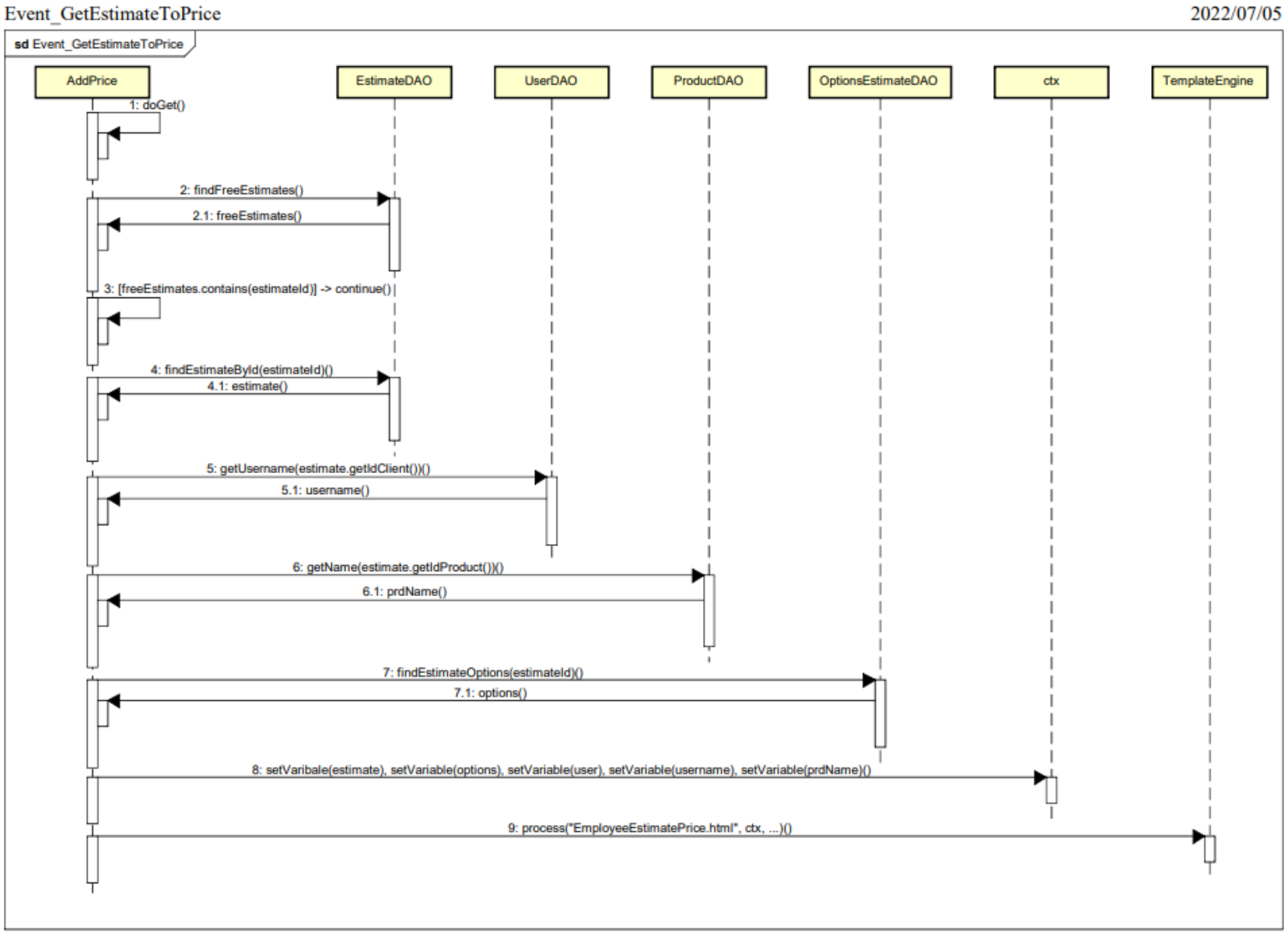
Event: GetAvailableOptions



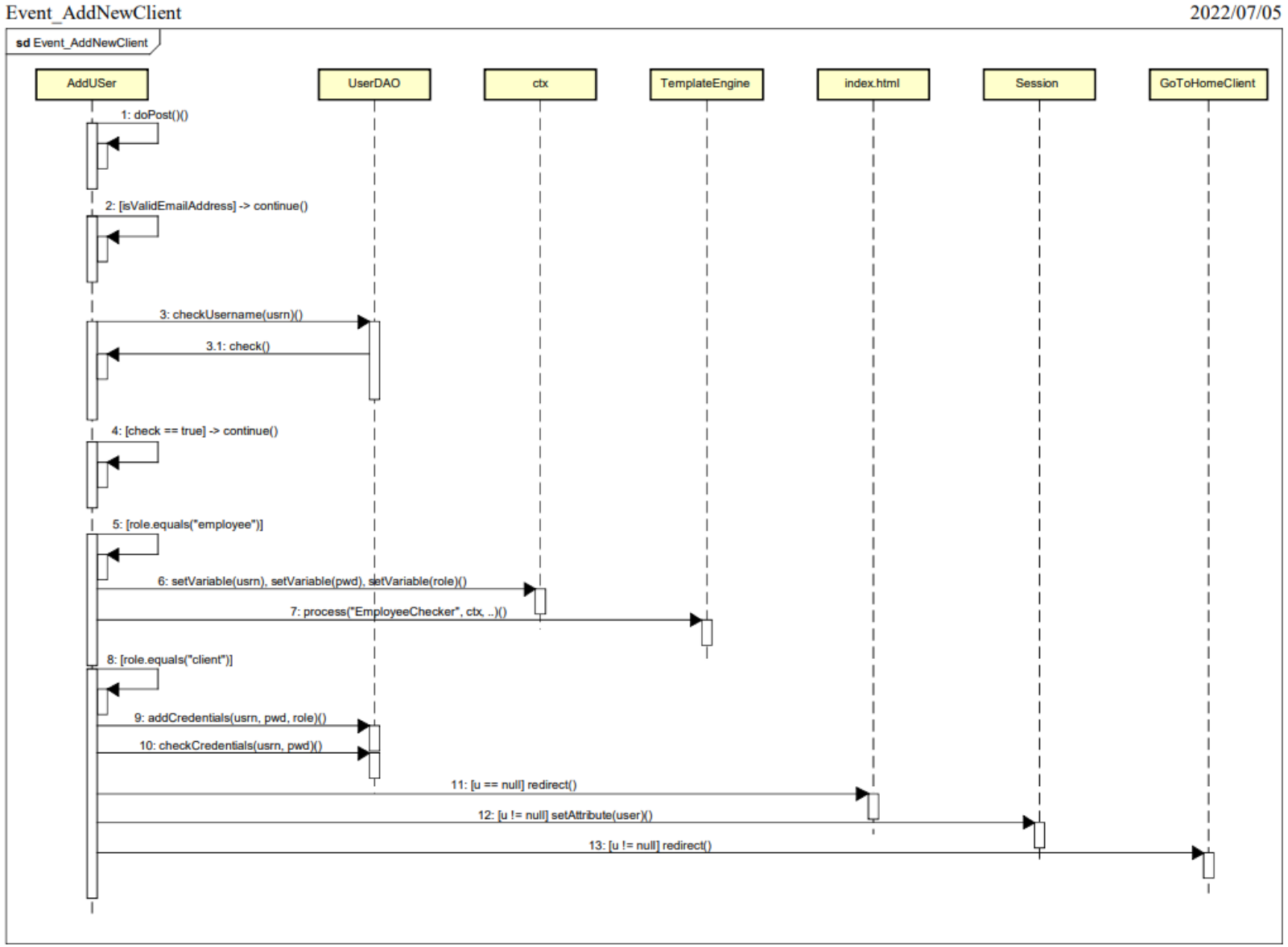
Event: CreateEstimate



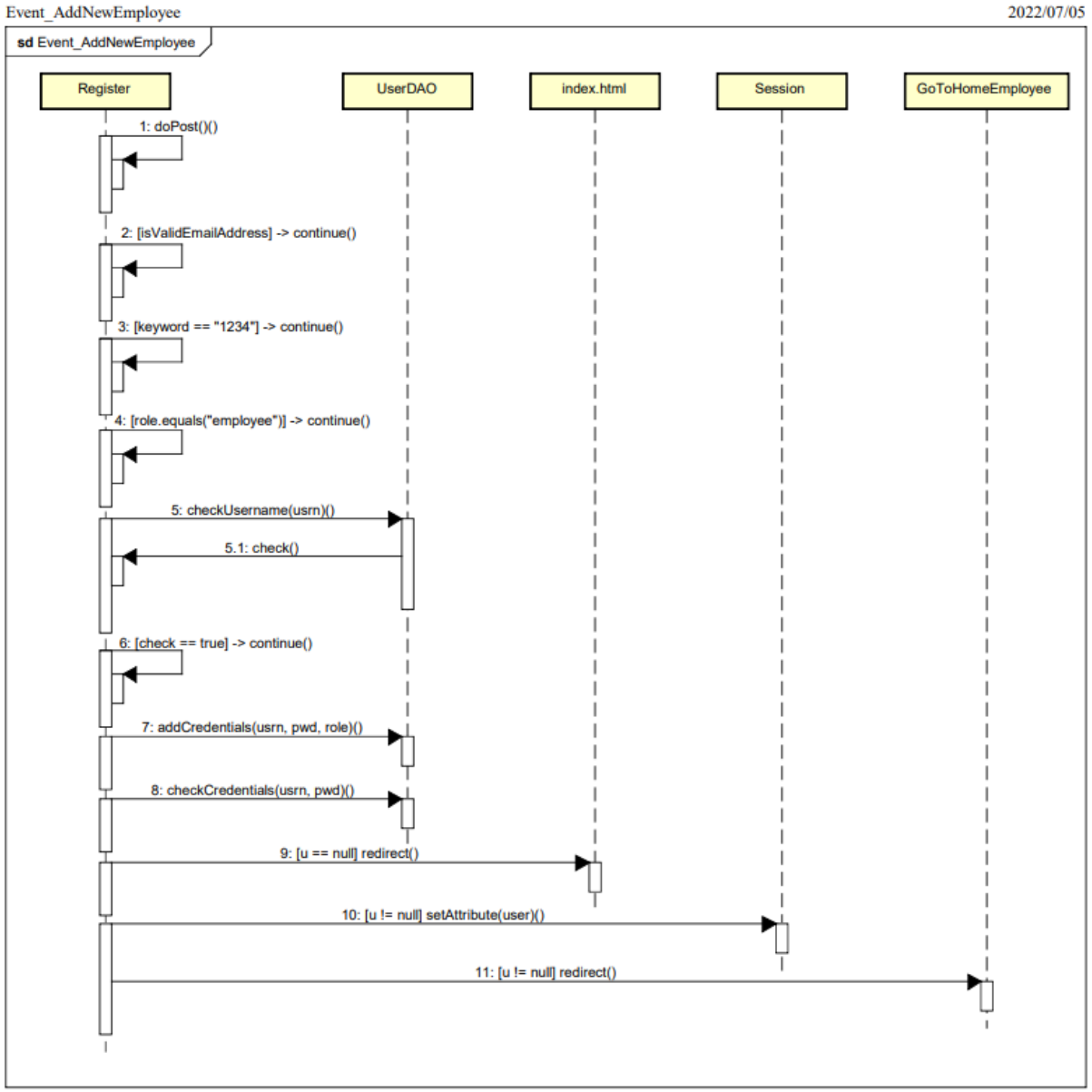
Event: GetEstimatToPrice



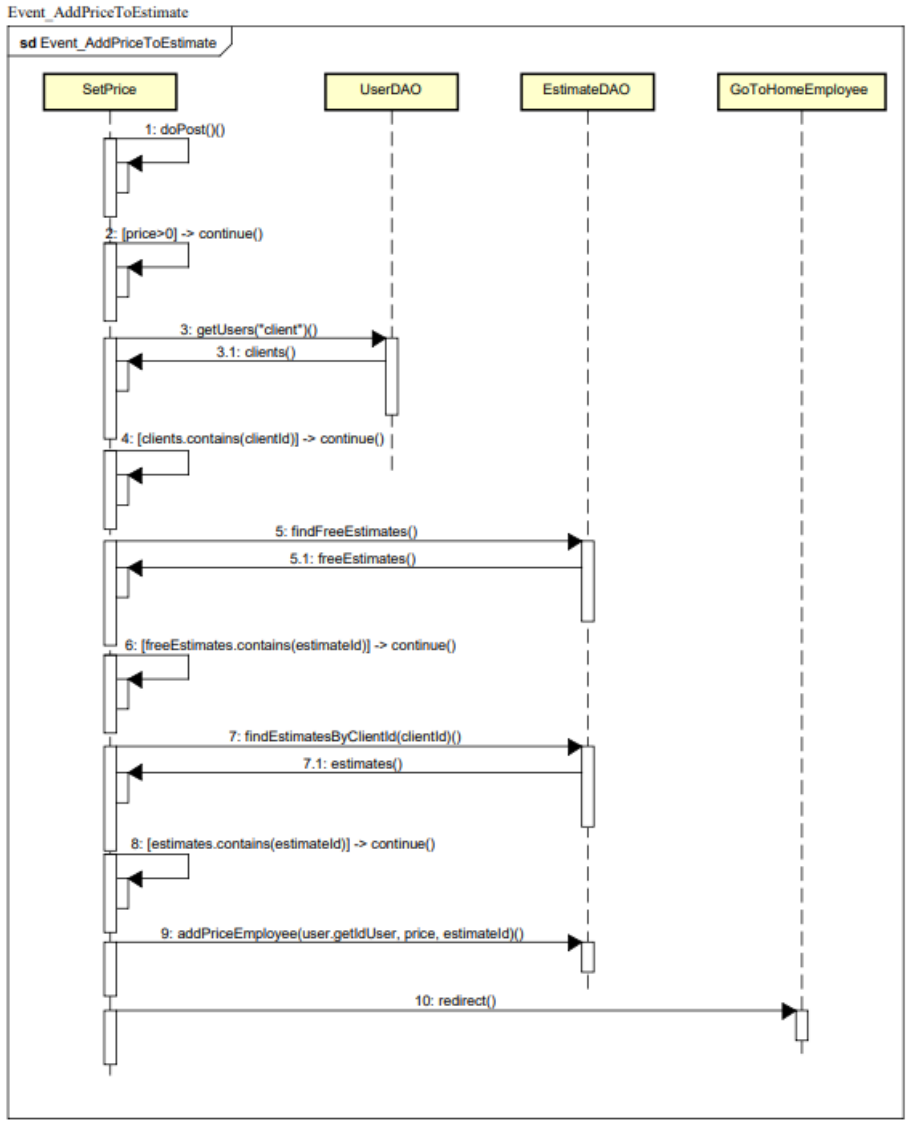
Event: AddNewClient



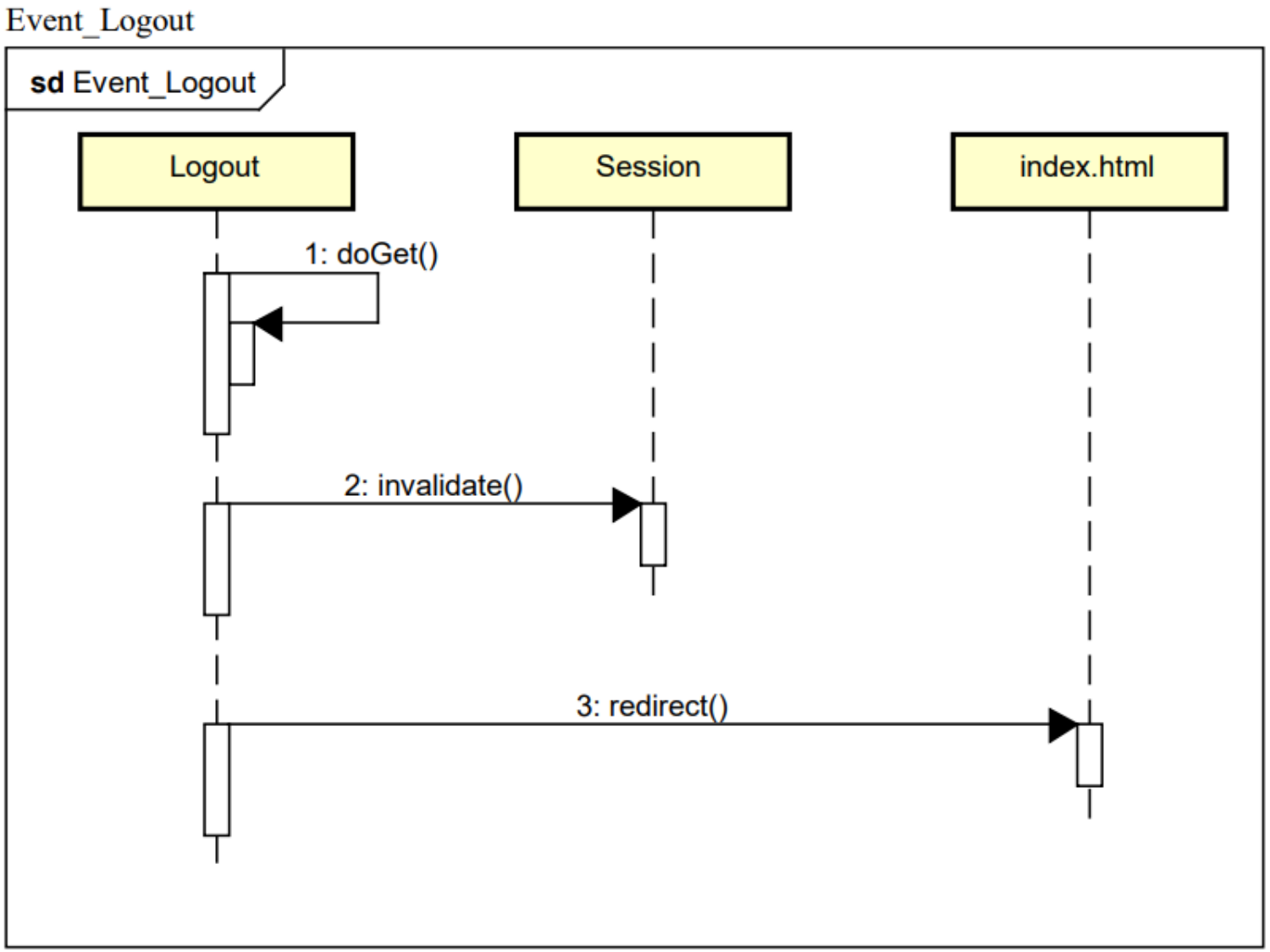
Event: AddNewEmployee



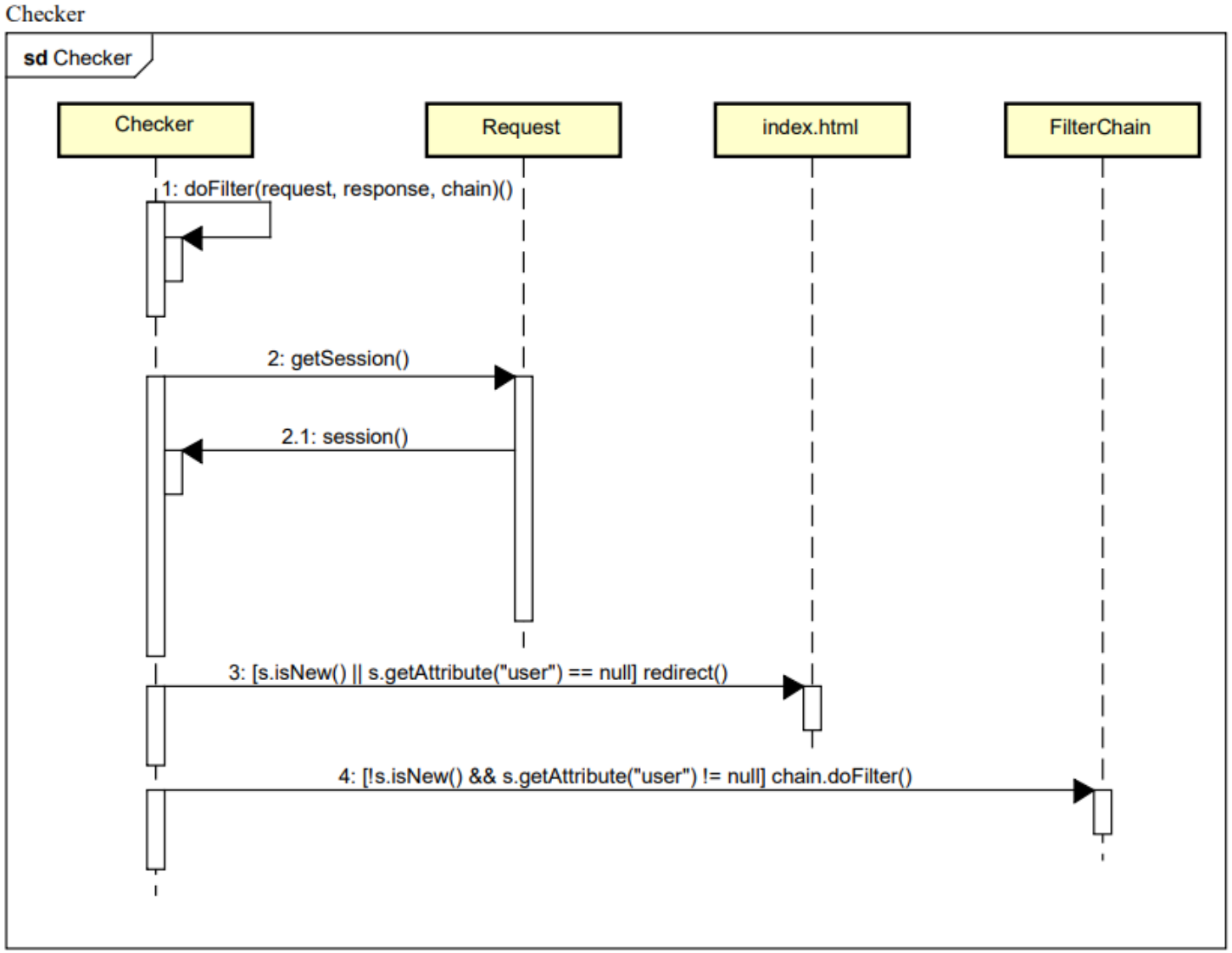
Event: AddPrice



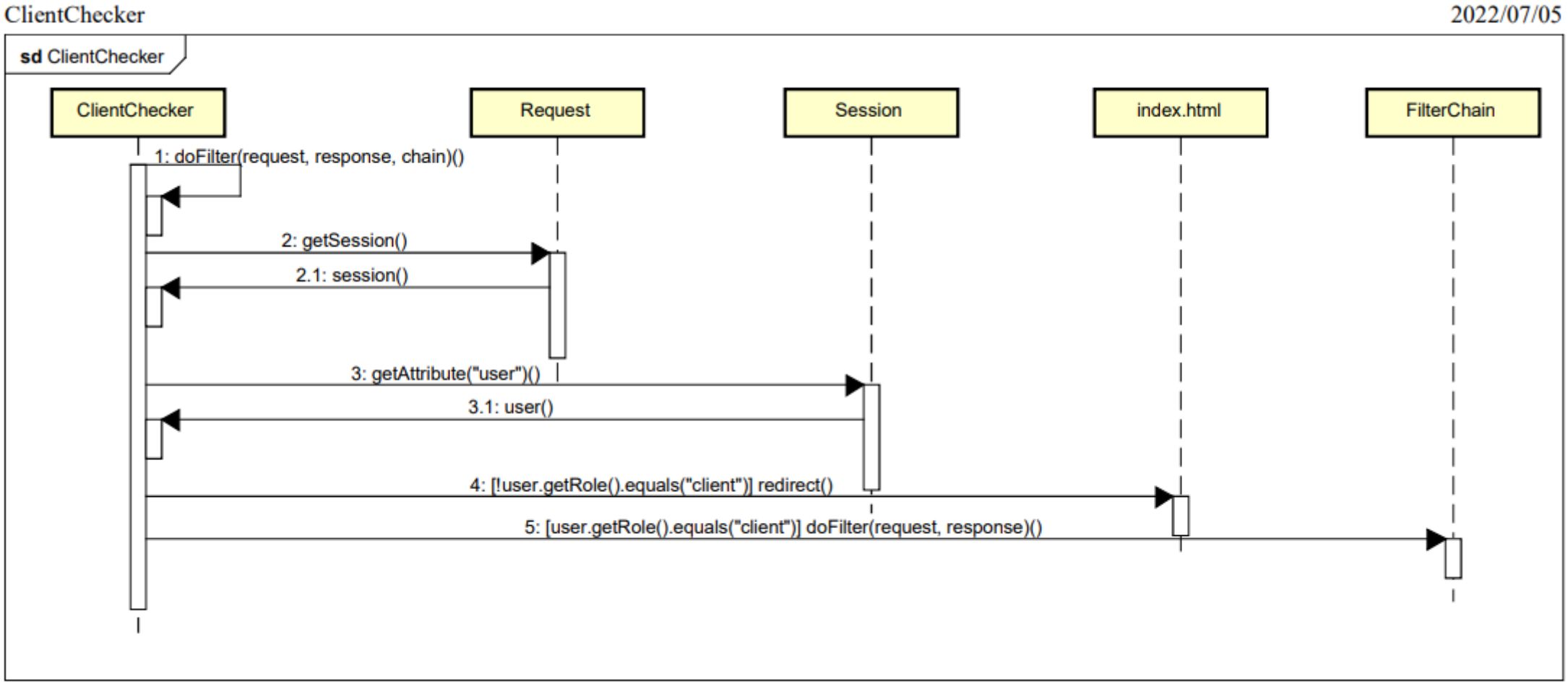
Event: Logout



Checker


Checker
sd Checker
Checker
Request
index.html
FilterChain
1: doFilter(request, response, chain)()
2: getSession()
2.1: session()
3: [s.isNew() || s.getAttribute("user") == null] redirect()
4: [!s.isNew() && s.getAttribute("user") != null] chain.doFilter()

Client Checker



Employee Checker