POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Progetto TIW versione RIA

Progetto di TIW
Ingegneria Informatica

Studente: **Yanfeng Li**

Codice Persona: 10650552
Prof: Prof. Fraternali Piero, Prof. Federico Milani
Anno Accademico: 2021-2022

Contents

1 | Introduction

1.1. Task

Ex2: Quote Management

An web application allows managing requests for quotes for personalized products. The application supports registration and login of both clients and employees through a public page with appropriate forms. The registration checks the uniqueness of the username. A quote is associated with a product, the client who requested it, and the employee who handled it. The quote includes one or more options for the associated product, which must be among those available for the product. A product has a code, an image, and a name. An option has a code, a type ("normal", "on sale"), and a name. A quote has a price, defined by the employee. When the user (client or employee) accesses the application, a LOGIN PAGE appears, through which the user authenticates with a username and password. When a client logs in, they access a CLIENT HOME PAGE containing a form to create a quote and a list of quotes created by the client. By selecting one of the quotes, the client can view its details. Through the quote creation form, the user first chooses the product; once the product is chosen, the form displays the options for that product. The user selects the options (at least one) and confirms the submission of the quote by clicking the SEND QUOTE button. When an employee logs in, they access an EMPLOYEE HOME PAGE containing the list of quotes they have previously managed and those not yet associated with any employee. When the employee selects an item from the list of quotes not yet associated with anyone, a QUOTE PRICING page appears showing the client's data (username) and the quote's data, along with a form to enter the quote's price. When the employee enters the price and submits the data with the SEND PRICE button, the EMPLOYEE HOME PAGE reappears with the updated lists of quotes. The price defined by the employee is visible to the client when they access their list of quotes and view the quote details. The QUOTE PRICING page also contains a link to return to the EMPLOYEE HOME PAGE. The application allows user logout.

Now, let's modify the previous specifications for a client-server web application as follows:

Registration checks the syntactic validity of the email address and equality between the "password" and "repeat password" fields, also on the client side. After login, the entire application is implemented with a single page for each role: a single page for the client role and a single page for the employee role. Every user interaction is handled without fully reloading the page but involves asynchronous server invocation and possible modification of the content to be updated following the event. In the client page, choosing the product results in the subsequent display of options without making an additional server call. Sending the quote by the client must also perform data verification on the client side (at least one option chosen). In the employee page, price control (not null and greater than zero) must also be done on the client side. Any server-side errors must be reported through an alert message within the client or employee page.

2 | Database

2.1. Textual Analysis

1. Entities
2. Attributes
3. Relationships

An web application allows managing requests for quotes for personalized products. The application supports registration and login of clients and employees through a public page with appropriate form. Registration checks the uniqueness of the username. A quote is associated with a product, the client who requested it, and the employee who handled it. The quote includes one or more options for the associated product, which must be among those available for the product. A product has a code, an image, and a name. An option has a code, a type ("normal", "on sale"), and a name. A quote has a price, defined by the employee. When the user (client or employee) accesses the application, a LOGIN PAGE appears, through which the user authenticates with username and password. When a client logs in, they access a HOME PAGE CLIENT containing a form to create a quote and the list of quotes created by the client. By selecting one of the quotes, the client can view its details. Through the quote creation form, the user first chooses the product; once the product is chosen, the form displays the options for that product. The user selects the options (at least one) and confirms the submission of the quote by clicking the SEND QUOTE button. When an employee logs in, they access a page HOME PAGE EMPLOYEE containing the list of quotes managed by them previously and those quotes not yet associated with any employee. When the employee selects an item from the list of quotes not yet associated with anyone, a page QUOTE PRICING appears showing the client's data (username) and the quote's data, along with a form to enter the quote's price. When the employee enters the price and submits the data with the SEND PRICE button, the HOME PAGE EMPLOYEE page reappears with the updated lists of quotes. The price defined by the employee is visible to the client when they access their list of quotes and view the details of the quote. The QUOTE PRICING page also contains a link to return to the HOME PAGE EMPLOYEE. The application allows user logout.

2.2. Database Design

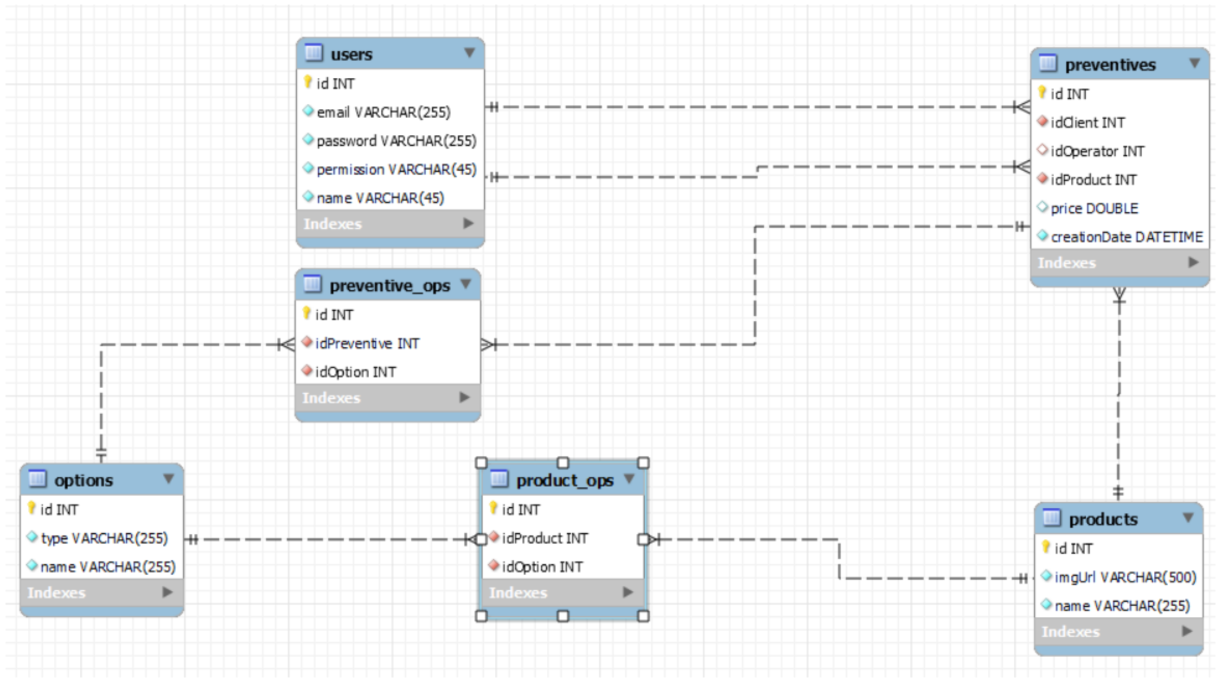
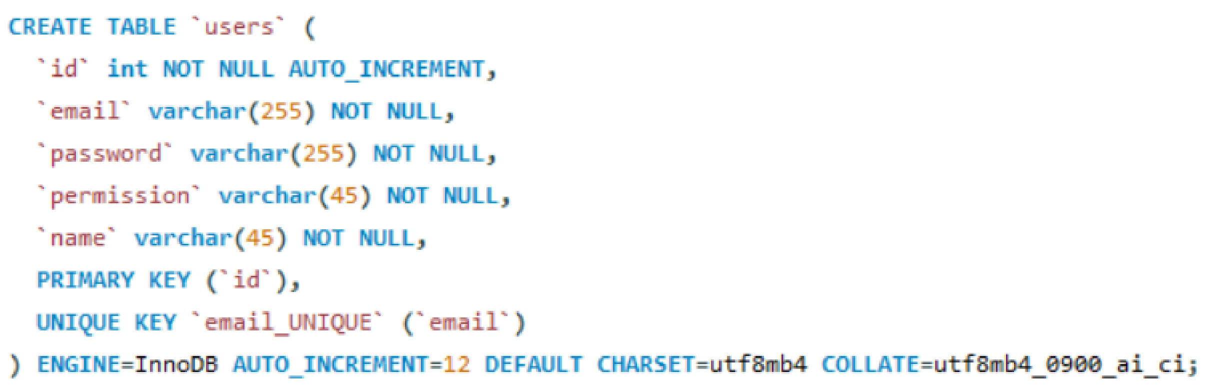


Figure 2.1: Diagramma Entita relazionale

2.3. Database Schema

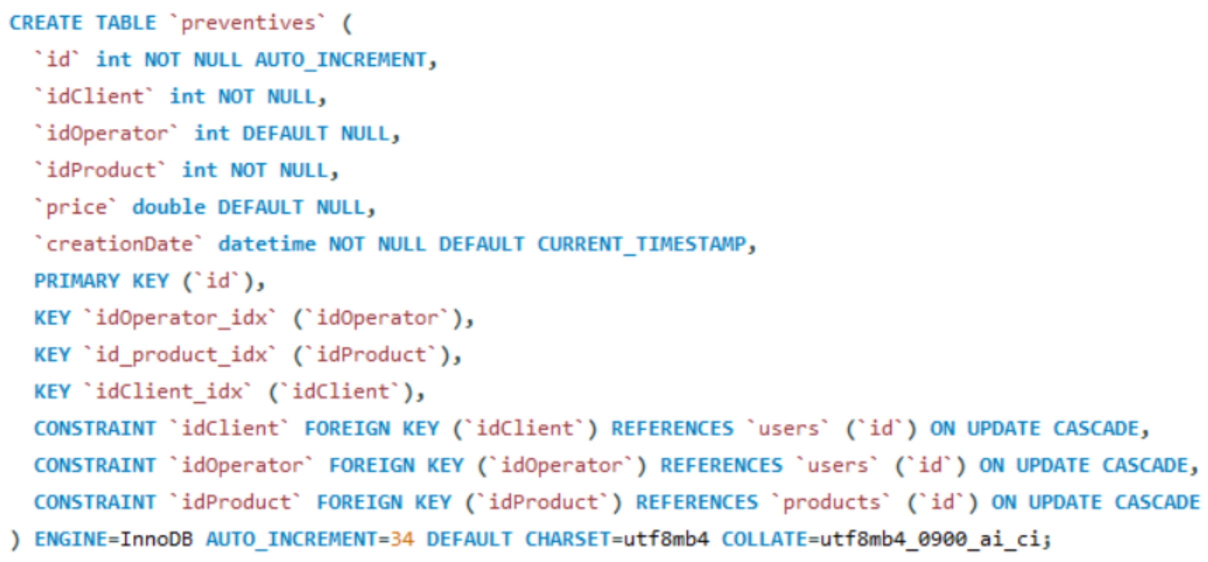
2.3.1. Users



```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `permission` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email_UNIQUE` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 2.2: Users

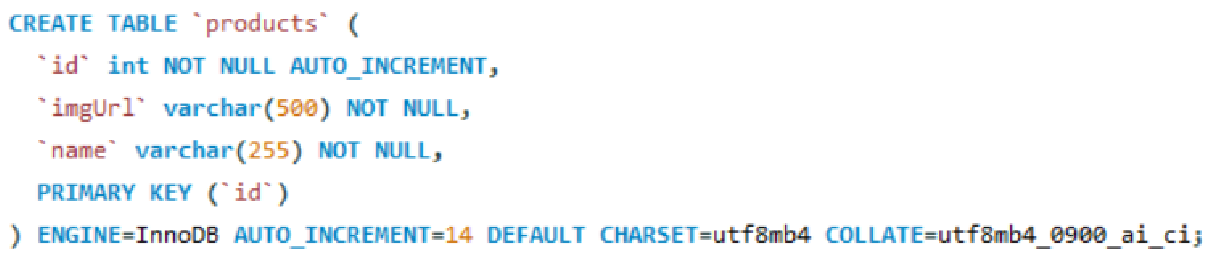
2.3.2. Preventives



```
CREATE TABLE `preventives` (
  `id` int NOT NULL AUTO_INCREMENT,
  `idClient` int NOT NULL,
  `idOperator` int DEFAULT NULL,
  `idProduct` int NOT NULL,
  `price` double DEFAULT NULL,
  `creationDate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `idOperator_idx` (`idOperator`),
  KEY `id_product_idx` (`idProduct`),
  KEY `idClient_idx` (`idClient`),
  CONSTRAINT `idClient` FOREIGN KEY (`idClient`) REFERENCES `users` (`id`) ON UPDATE CASCADE,
  CONSTRAINT `idOperator` FOREIGN KEY (`idOperator`) REFERENCES `users` (`id`) ON UPDATE CASCADE,
  CONSTRAINT `idProduct` FOREIGN KEY (`idProduct`) REFERENCES `products` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=34 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 2.3: Preventives

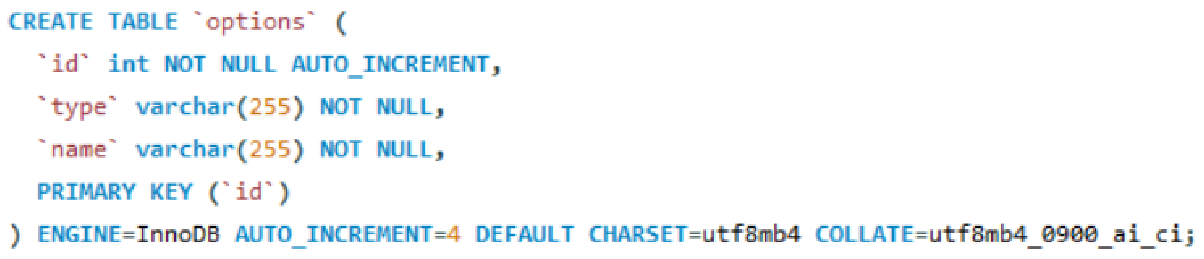
2.3.3. Products



```
CREATE TABLE `products` (
  `id` int NOT NULL AUTO_INCREMENT,
  `imgUrl` varchar(500) NOT NULL,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 2.4: Products

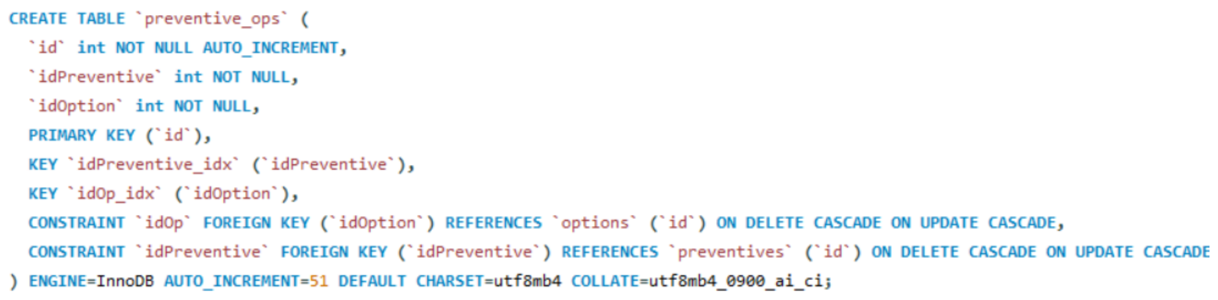
2.3.4. Options



```
CREATE TABLE `options` (
  `id` int NOT NULL AUTO_INCREMENT,
  `type` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 2.5: Options

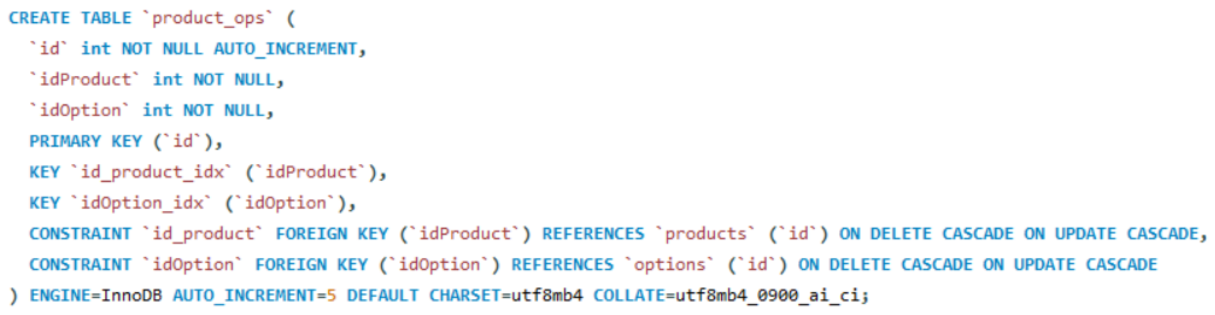
2.3.5. Preventive_ops



```
CREATE TABLE `preventive_ops` (
  `id` int NOT NULL AUTO_INCREMENT,
  `idPreventive` int NOT NULL,
  `idOption` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idPreventive_idx` (`idPreventive`),
  KEY `idOp_idx` (`idOption`),
  CONSTRAINT `idOp` FOREIGN KEY (`idOption`) REFERENCES `options` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `idPreventive` FOREIGN KEY (`idPreventive`) REFERENCES `preventives` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=51 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 2.6: Preventive_ops

2.3.6. Product_ops



```
CREATE TABLE `product_ops` (
  `id` int NOT NULL AUTO_INCREMENT,
  `idProduct` int NOT NULL,
  `idOption` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `id_product_idx` (`idProduct`),
  KEY `idOption_idx` (`idOption`),
  CONSTRAINT `id_product` FOREIGN KEY (`idProduct`) REFERENCES `products` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `idOption` FOREIGN KEY (`idOption`) REFERENCES `options` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 2.7: Product_ops

3 | Application

3.1. Requirements Analysis

1. Pages (view)
2. view components
3. events
4. actions

A client-server web application is to be developed that modifies the previous specifications as follows: The application supports registration and login through a public page with appropriate forms. Registration checks the syntactic validity of the email address and the equality between the "password" and "repeat password" fields, also on the client side. After login, the entire application is implemented with a single page for each of the roles: a single page for the client role and a single page for the employee role. Every user interaction is handled without fully reloading the page, but it produces asynchronous server invocation and potential modification of the content to be updated following the event. On the client page, product selection results in the subsequent display of options without making an additional server call. Submitting a quote by the client must also perform data verification on the client side (at least one option chosen). On the employee page, price control (not null and greater than zero) must also be done on the client side. Any server-side errors must be reported through an alert message within the client or employee page.

4 | Components from Server_side

4.1. Model Objects (Beans)

- User
- Preventive
- Product
- Option

4.2. Data Access Objects (DAO)

- UserDAO
  - findUserByCredentials(email, password) : User
  - signupUser(name, email, password, permission) : void
  - getUserByEmail(email) : User
- ClientDAO
  - getPreventivesByUserId() : List<Preventive>
  - sendPreventive(productId, options) : void
- AdminDAO
  - getAdminPreventives() : List<Preventive>
  - getAllAvailablePreventive() : List<Preventive>
  - sendPreventive(preventiveId, price) : void
- PreventiveDAO
  - getPreventiveById() : Preventive
  - getOptionsById() : List<Option>

- ProductDao
  - getAllProduct() : List<Product>
  - getProductById(productId) : Product
  - getProductOptions(productId) : List<Option>

4.3. Filters

- CheckAdmin
- CheckClient
- CheckNotLoggedUser
- CheckLoggedUser

4.4. Controllers (servlets) [access rights]

- Login [Not Logged User]
- Logout [ALL]
- Signup [Not Logged User]
- SendPreventivo [User, Client]
- SendPrezzo [User, Admin]
- GetClientPreventivoList [User, Client]
- GetProductList [User]
- GetAdminAvailableList [User, Admin]
- GetAdminPreventivoList [User, Admin]
- GetPreventivoPrezza [User, Admin]
- GetPreventivoDetails [User]

4.5. Views (Templates)

- index.html
- homeClient.html

- homeAdmin.html

5 | Components from Client_Side

5.1. Index

- Login form
  - Manages Data Submission and Error Handling
- Signup form
  - Manages Data Submission and Error Handling

5.2. Home_client

- PageOrchestrator
  - start() : Page Component Initialization
  - refresh() : Updating Component Contents
  - hide() : Hiding Components
  - handleError(error): Displaying Error Messages
- UserInfo
  - show() : To load user data, make it visible, and enable buttons for logout and back
- PreventivoDetail
  - show(): To download the data of a quote using an ID
  - update(): To load the data and make it visible
  - hide()
- PreventivoList
  - show(): To download a list of quotes

  - update(): To construct a list of components, load data, and make it visible ()

- ProductList
  - show(): To download a list of products
  - update(): To construct a list of components, load data, and make it visible
  - hide()
- ErrorDisplay
  - show(errorContent)
  - hide()

5.3. Home_admin

- PageOrchestrator
  - start() : Page Component Initialization
  - refresh() : Updating Component Contents
  - hide() : Hiding Components
  - handleError(error): Displaying Error Messages
- UserInfo
  - show() : To load user data, make it visible, and enable buttons for logout and back
- PreventivoDetail
  - show(): To download the data of a quote using an ID
  - update(): To load the data and make it visible
  - hide()
- PreventivoList
  - show(): To download a list of quotes
  - update(): To construct a list of components, load data, and make it visible ()
- AvailableList
  - show(): To download a list of quotes

  - update(): To construct a list of components, load data, and make it visible
  - hide()

- prezzaPreventivo
  - show(): To download the data of a quote by providing its ID
  - update(): To load the data and make it visible
  - hide()
- ErrorDisplay
  - show(errorContent)
  - hide()

6 | Design Applicativo(IFML)

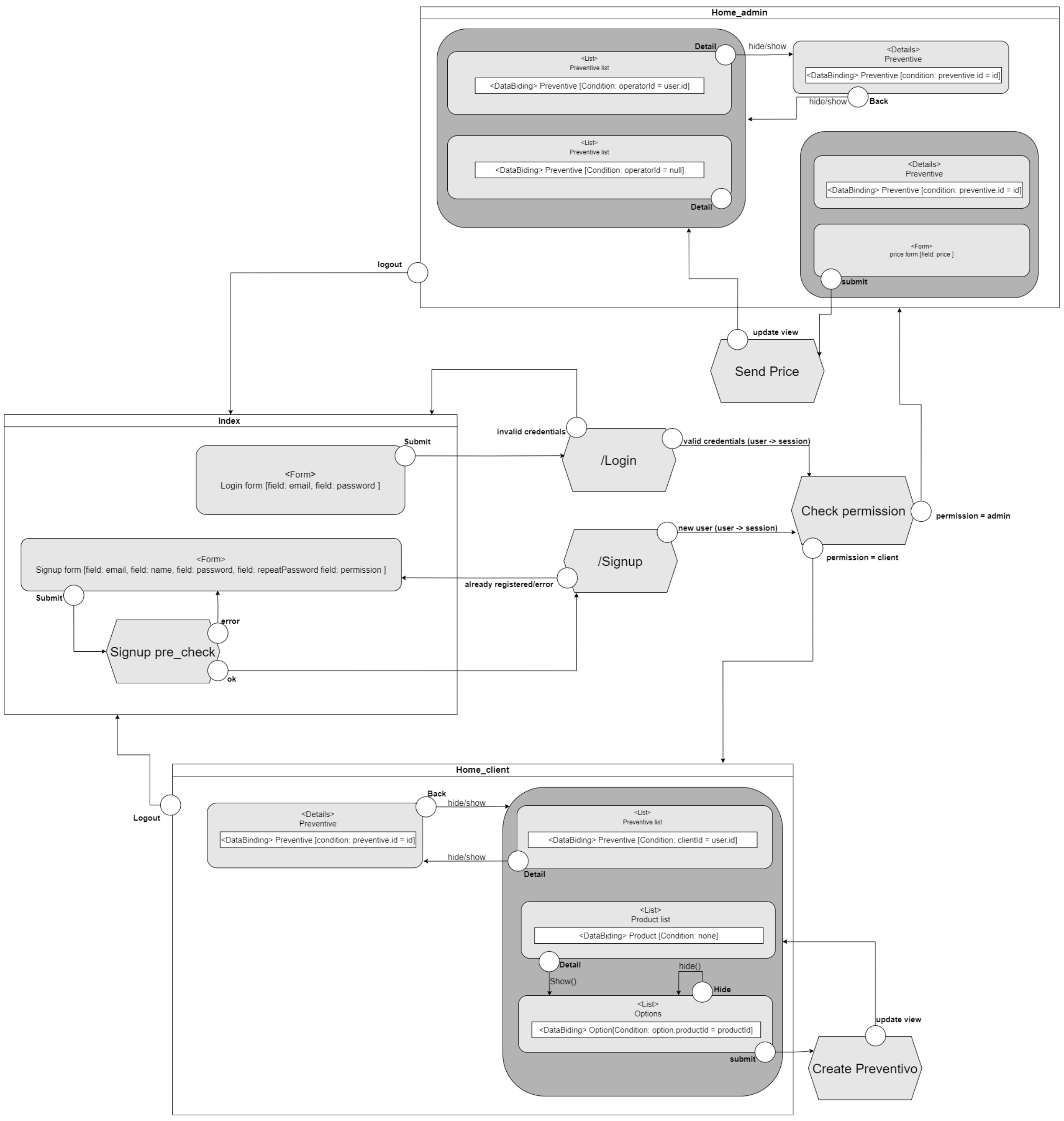

Home_admin
Detail
<List>
Preventive list
<DataBiding> Preventive [Condition: operatorId = user.id]
hide/show
<Details>
Preventive
<DataBinding> Preventive [condition: preventive.id = id]
hide/show
Back
<List>
Preventive list
<DataBiding> Preventive [Condition: operatorId = null]
Detail
<Details>
Preventive
<DataBinding> Preventive [condition: preventive.id = id]
<Form>
price form [field: price]
submit
logout
update view
Send Price
Index
Submit
<Form>
Login form [field: email, field: password]
invalid credentials
/Login
valid credentials (user -> session)
Check permission
permission = admin
new user (user -> session)
permission = client
<Form>
Signup form [field: email, field: name, field: password, field: repeatPassword field: permission]
/Signup
already registered/error
Submit
error
Signup pre_check
ok
Home_client
Logout
Back
hide/show
<Details>
Preventive
<DataBinding> Preventive [condition: preventive.id = id]
<List>
Preventive list
<DataBiding> Preventive [Condition: clientId = user.id]
hide/show
Detail
<List>
Product list
<DataBiding> Product [Condition: none]
Detail
Show()
hide()
Hide
<List>
Options
<DataBiding> Option[Condition: option.productId = productId]
submit
update view
Create Preventivo

Figure 6.1: Design Applicativo(IFML)

7 | Sequence Diagrams of Interactions

A few of Sequence Diagrans of Interactions are provided here.

7.1. CheckLoggedUser

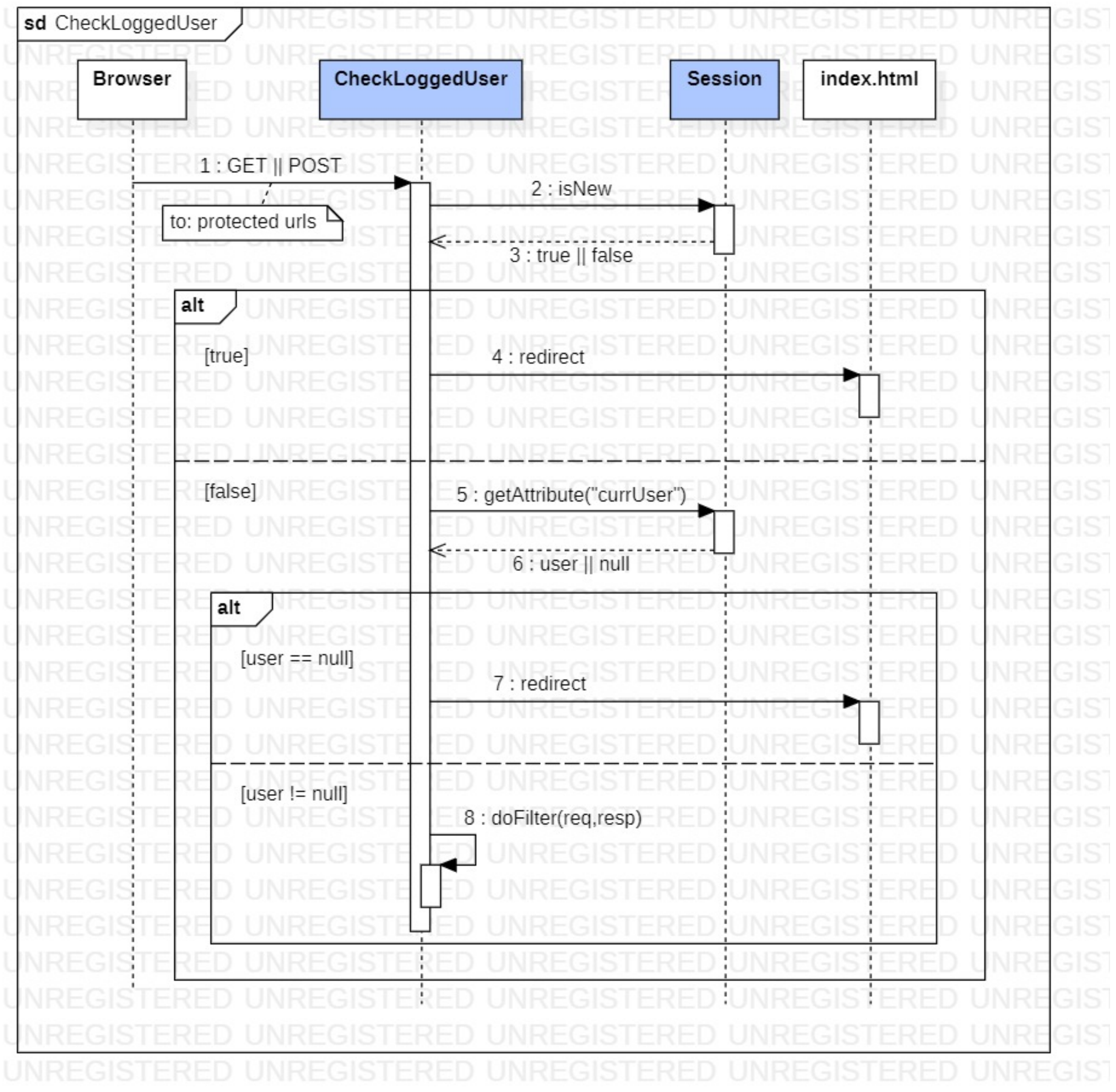


Figure 7.1: CheckLoggedUser

7.2. NoCacher

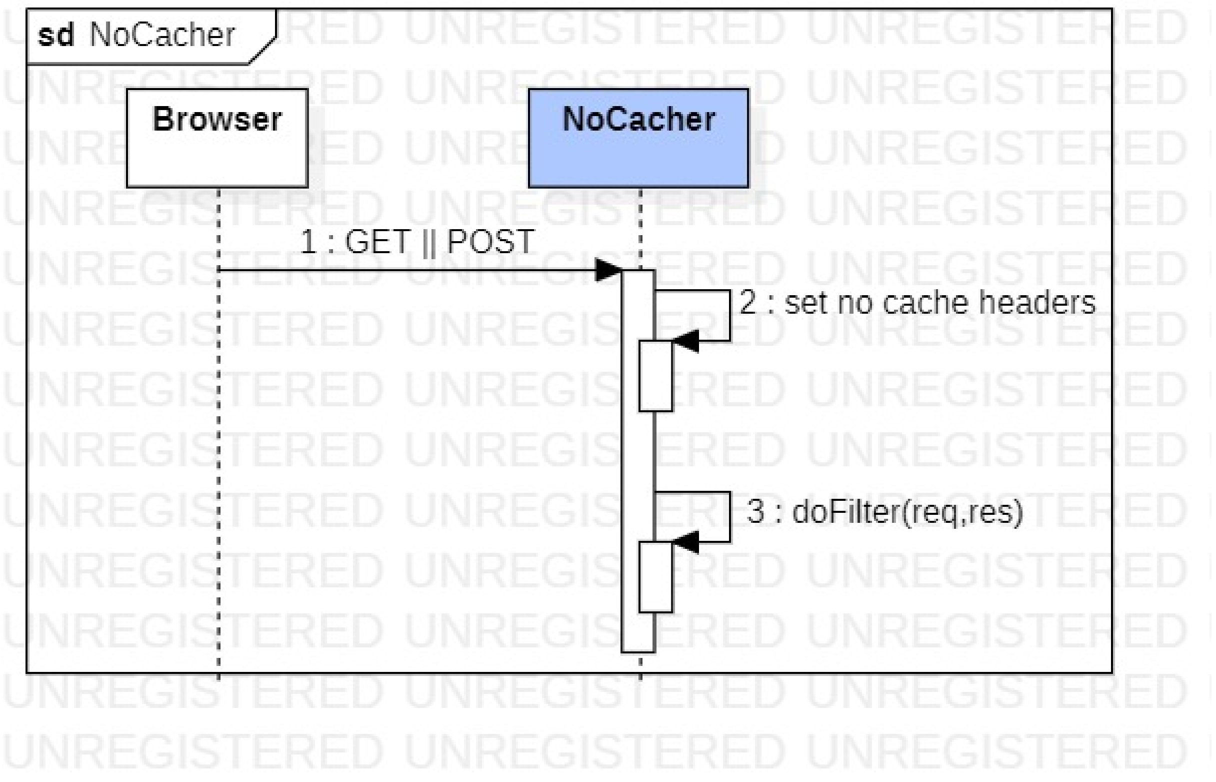


Figure 7.2: NoCacher

7.3. Login

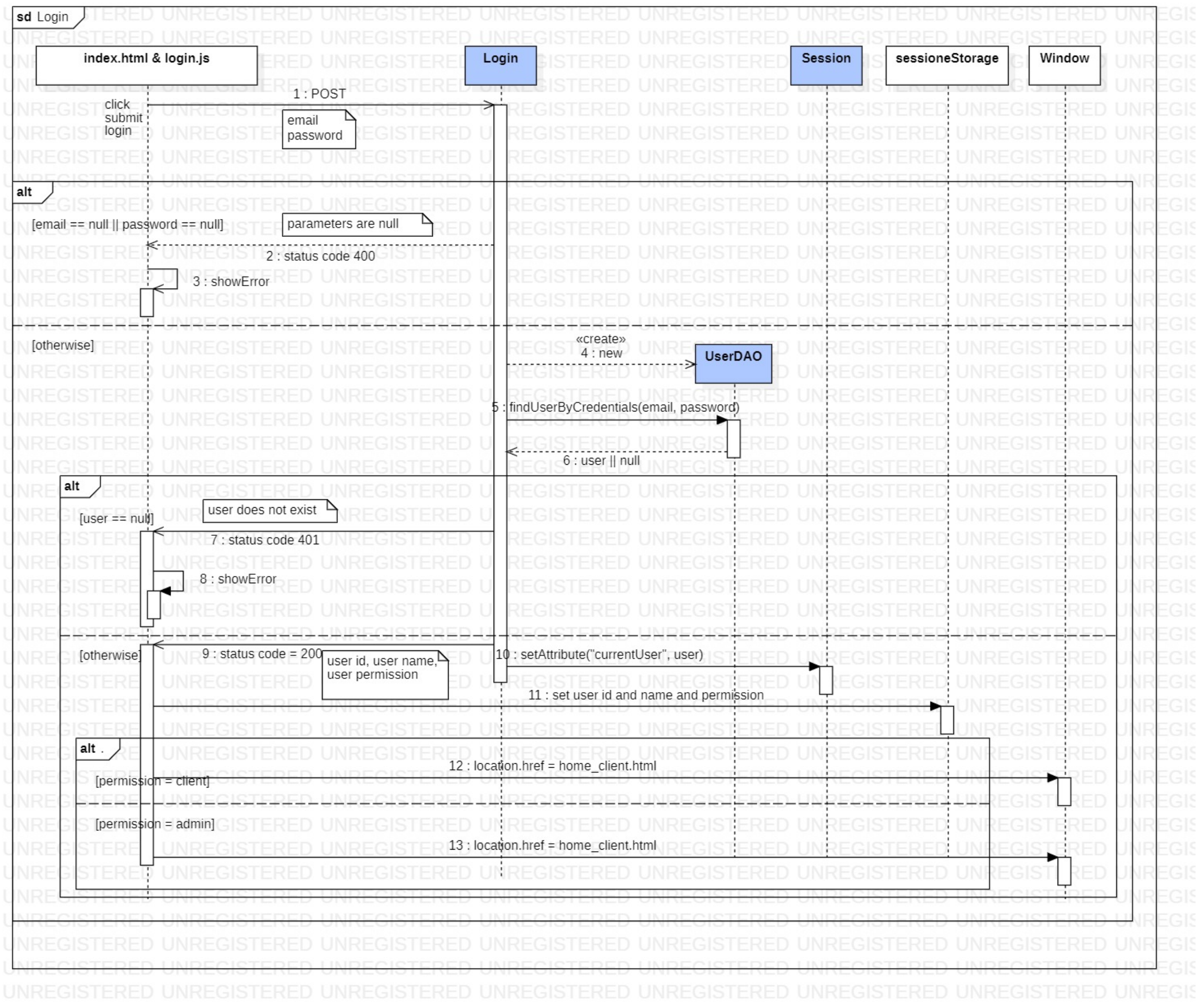


Figure 7.3: Login

7.4. Logout

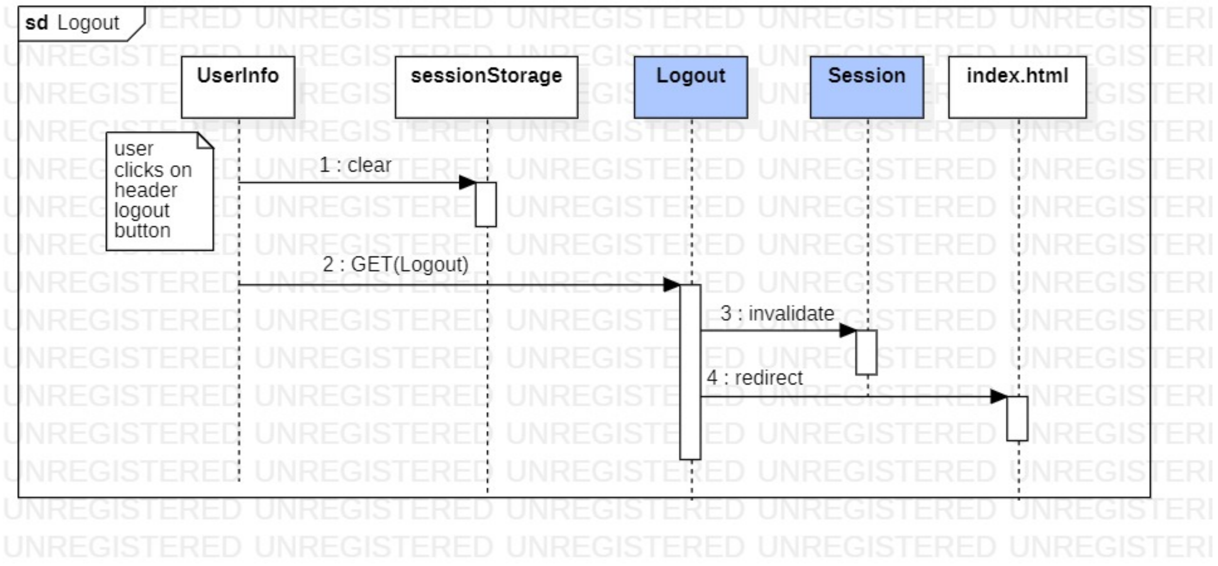


Figure 7.4: Logout

7.5. Signup

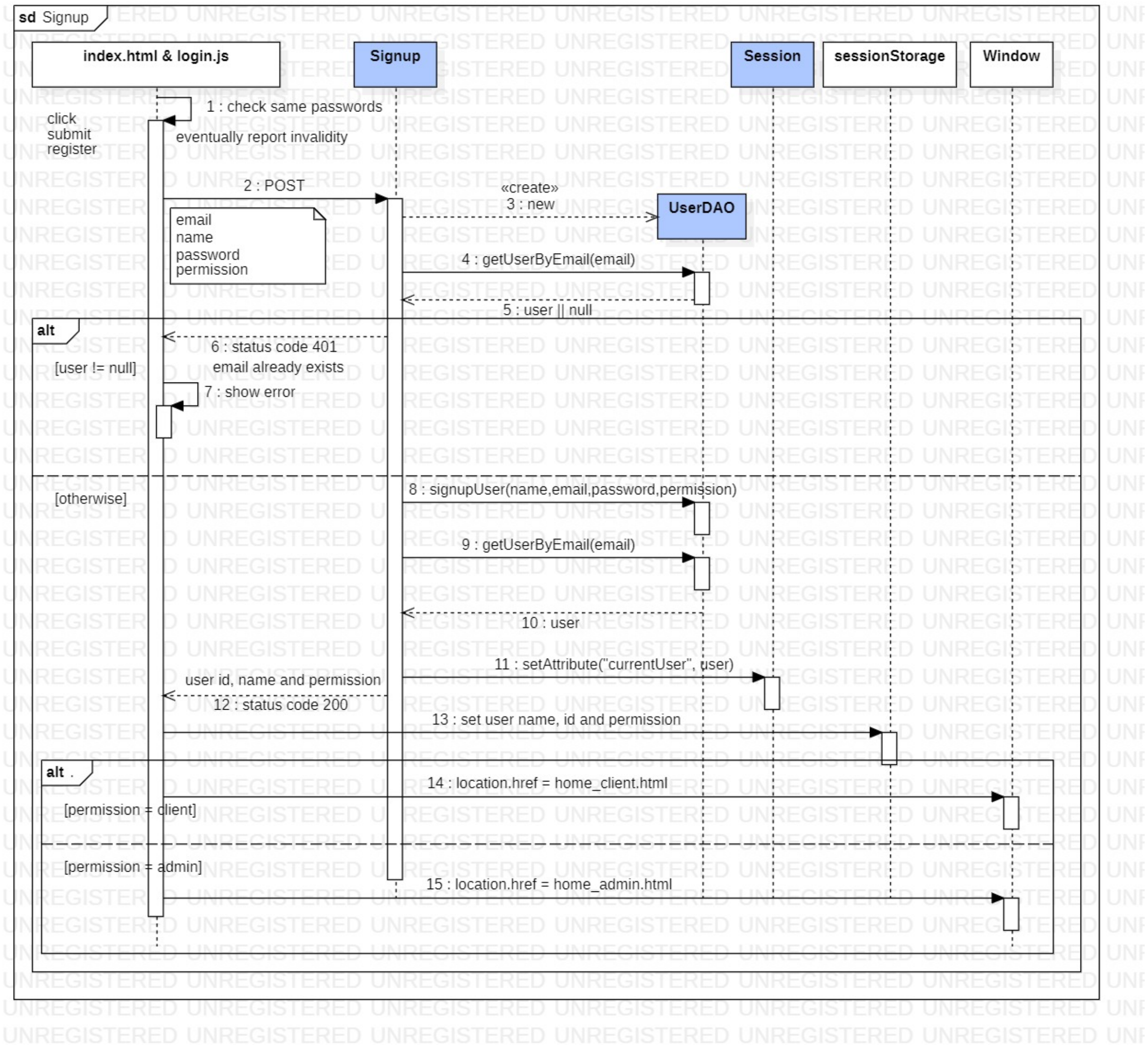


Figure 7.5: Signup

7.6. Home client page load

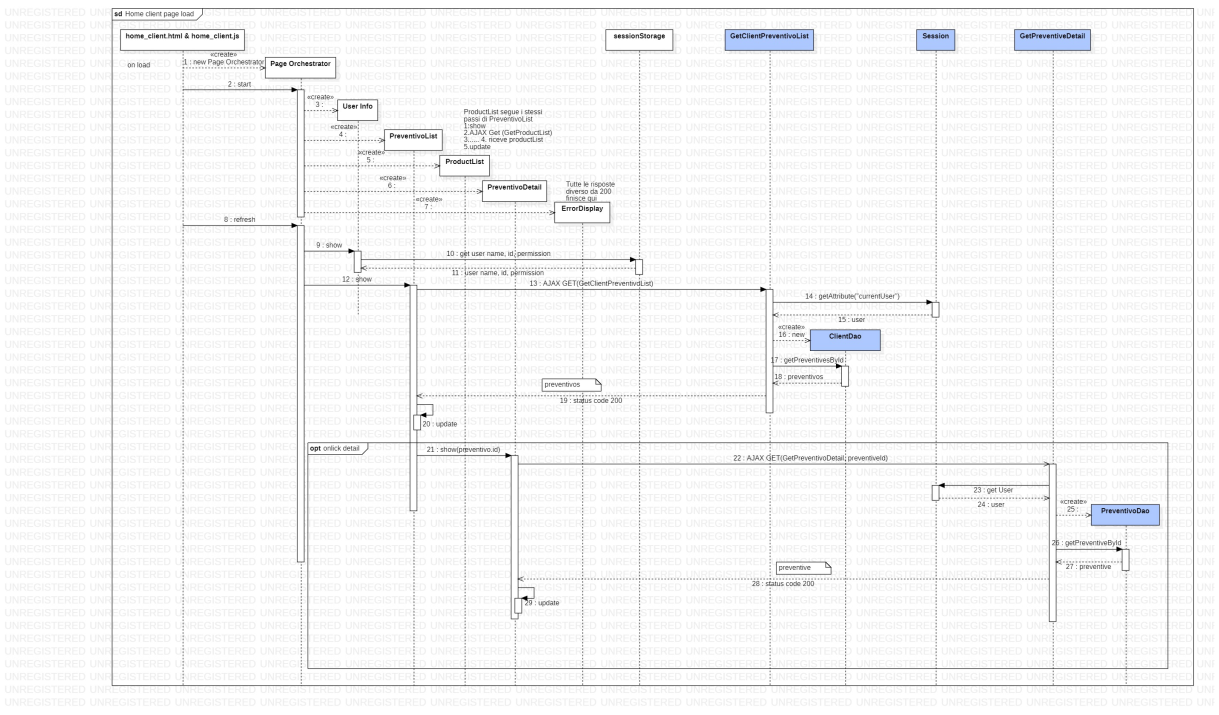


Figure 7.6: Home client page load