POLITECNICO
MILANO 1863

Progetto TIW - Traccia 3

A.A. 2022/2023
Prof. Piero Fraternali

Andrea Sgobbi
Numero Matricola: 957092
Codice Persona: 10726194

Maggio 2023

Contents

1 Pure HTML

1.1 Specifica

Un'applicazione permette all'utente (ad esempio il responsabile dei servizi ambientali di una regione) di gestire una collezione di immagini satellitari e una tassonomia di classificazione utile per etichettare immagini allo scopo di consentire la ricerca per categoria.

Dopo il login, l'utente accede a una pagina HOME in cui compare un albero gerarchico di categorie. Le categorie non dipendono dall'utente e sono in comune tra tutti gli utenti. L'utente può inserire una nuova categoria nell'albero. Per fare ciò usa una form nella pagina HOME in cui specifica il nome della nuova categoria e sceglie la categoria padre.

L'invio della nuova categoria comporta l'aggiornamento dell'albero: la nuova categoria è appesa alla categoria padre come ultimo sottoelemento. Alla nuova categoria viene assegnato un codice numerico che ne riflette la posizione (ad esempio, la nuova categoria "Amianto in tubi", figlia della categoria "9111 Amianto" assume il codice 91113). Dopo la creazione di una categoria, la pagina HOME mostra l'albero aggiornato.

Per velocizzare la costruzione della tassonomia l'utente può copiare un intero sottoalbero in una data posizione: per fare ciò clicca sul link "copia" associato alla categoria radice del sottoalbero da copiare. A seguito di tale azione l'applicazione mostra, sempre nella HOME page, l'albero con evidenziato il sottoalbero da copiare: tutte le altre categorie hanno un link "copia qui". La selezione di un link "copia qui" comporta l'inserimento di una copia del sottoalbero come ultimo figlio della categoria destinazione. Le modifiche effettuate da un utente e salvate nella base di dati diventano visibili agli altri utenti.

Per semplicità si ipotizzi che per ogni categoria il numero massimo di sottocategorie sia 9, numerate da 1 a 9. In questo caso l'operazione di copia deve controllare che lo spostamento non determini un numero di sottocategorie superiore a 9. Si preveda anche un link "copia qui" non associato a un nodo della tassonomia che permette di copiare un sotto-albero al primo livello della tassonomia (se non esistono già 9 nodi al primo livello della tassonomia).

PAGES/VIEWS - VIEW COMPONENTS - EVENTS - ACTIONS

1.2 Struttura DB

Per entrambe le versioni dell'applicazione é stato usato il medesimo schema all'interno del DB, riportato qui. A livello di implementazione si ritiene importante evidenziare la traduzione della relazione Padre-Figlio nella tabella **Category** con un semplice campo *parentID*.

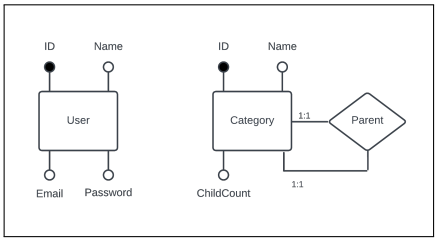

Figure 1: Diagramma Entità-Relazione

```sql
CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email_UNIQUE` (`email`)
)

CREATE TABLE `category` (
  `id` int(11) NOT NULL,
  `name` varchar(45) NOT NULL,
  `parentID` int(11) DEFAULT NULL,
  `childCount` int(11) NOT NULL,
  PRIMARY KEY (`id`)
```

1.3 Componenti

- **Views**
  - login.html (include la registrazione)
  - header.html (incluso se loggati)
  - home.html
  - error.html
- **DAO**
  - UserDAO
    - registerUser
    - getUserByEmail
    - checkCredentials
  - CategoryDAO
    - getChildCount
    - getCategoryTree
    - insertCategory
    - insertSubtree
- **Beans**
  - User
  - Category
- **Filters**
  - CheckLoggedIn
  - CheckLoggedOut
- **Controllers[privileges]**
  - Login[not logged]
  - Register[not logged]
  - Logout[logged]
  - GoToLogin[not logged]
  - GoToHome[logged]
  - CreateCategory[logged]
  - CopyCategory[not logged]

1.4 Application Design in IFML

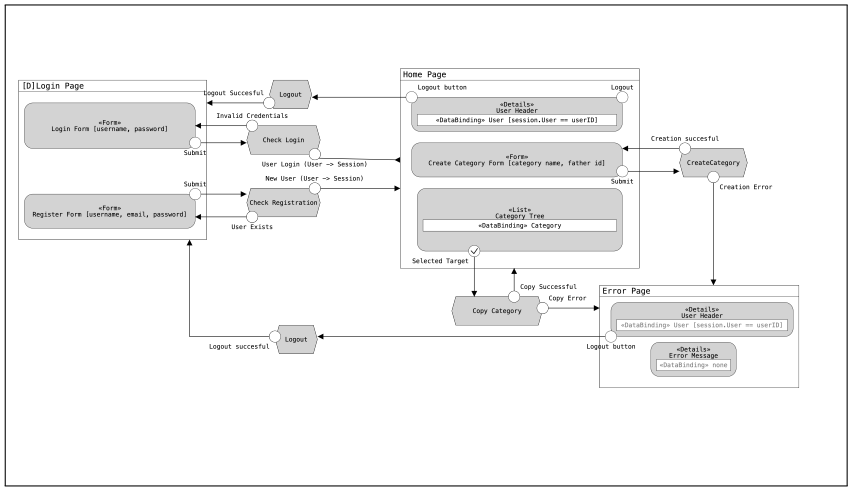

Figure 2: Diagramma IFML della versione HTML

1.5 Sequence Diagrams

Tutte le servlet svolgono controlli sulla correttezza dell'input. Per semplicità, essi sono mostrati per le servlet **Login** (5) e **Register** (6) e omessi nei diagrammi successivi, in quanto svolti nello stesso modo. Si é scelto di mostrare messaggi di *warning* per errori generabili senza modifiche all'interfaccia utente (ad esempio una email già utilizzata), mentre di mostrare la pagina *error.html* per errori riproducibili solo modificando richieste o html.

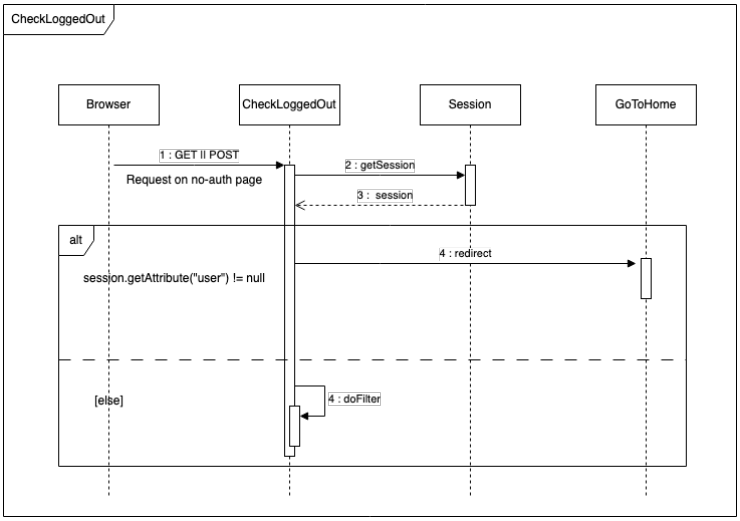

Figure 3: Interazione con il filtro CheckLoggedOut

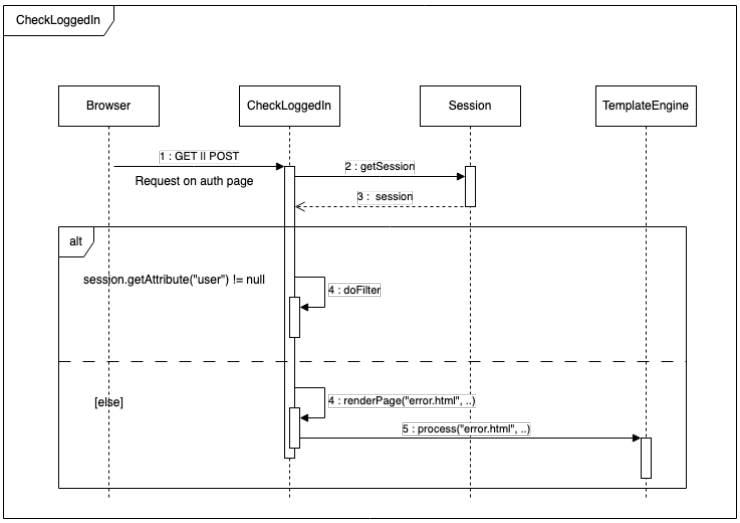

Figure 4: Interazione con il filtro CheckLoggedIn

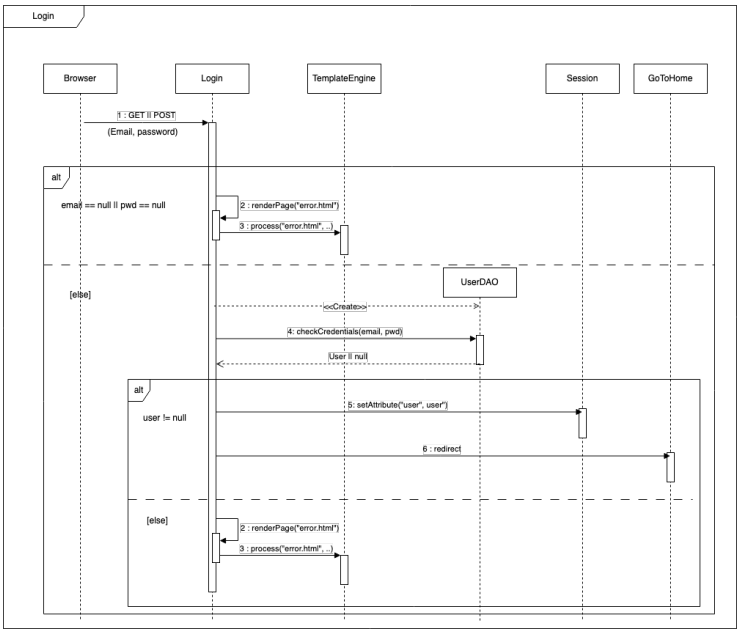

Figure 5: Interazione con la servlet Login

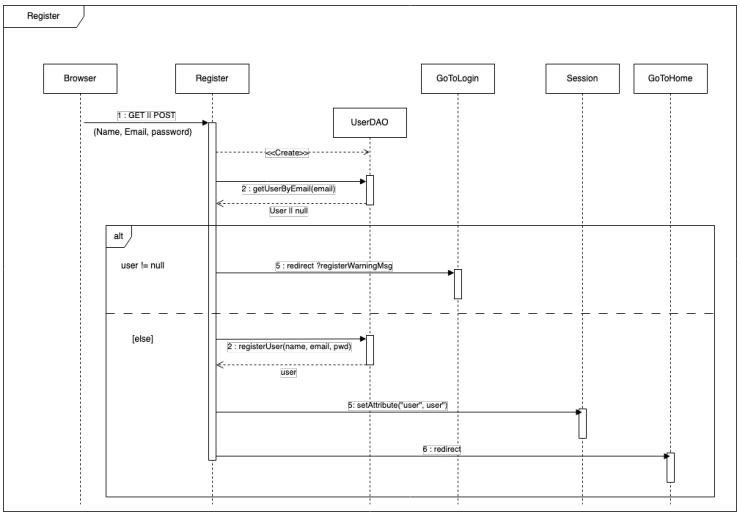

Figure 6: Interazione con la servlet Register

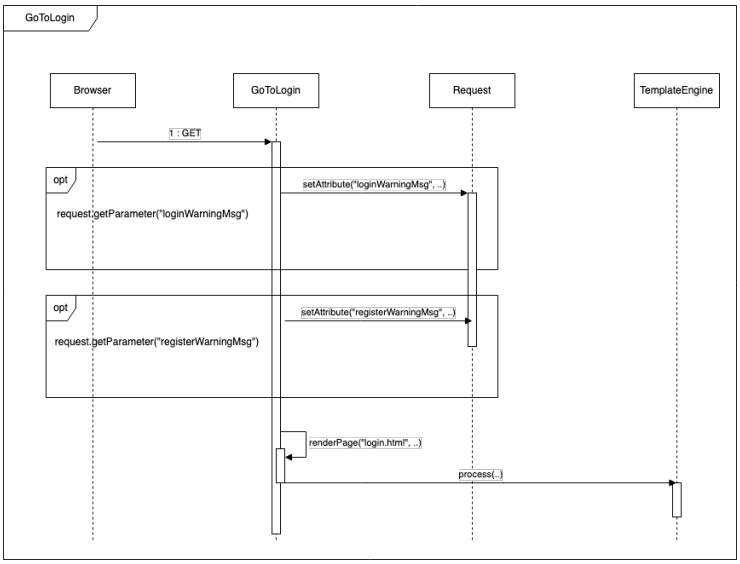

Figure 7: Interazione con la servlet GoToLogin

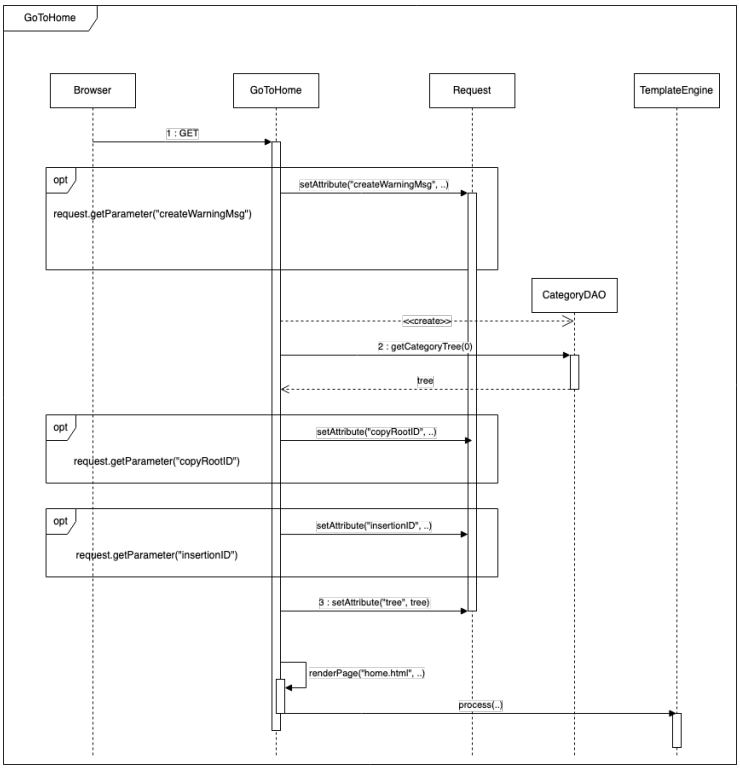

Figure 8: Interazione con la servlet GoToHome

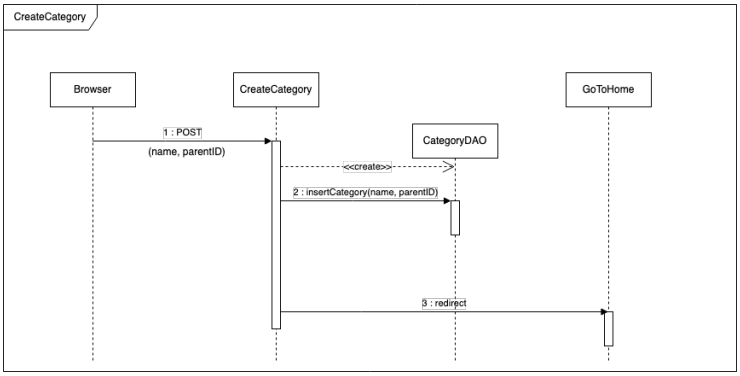

Figure 9: Interazione con la servlet CreateCategory

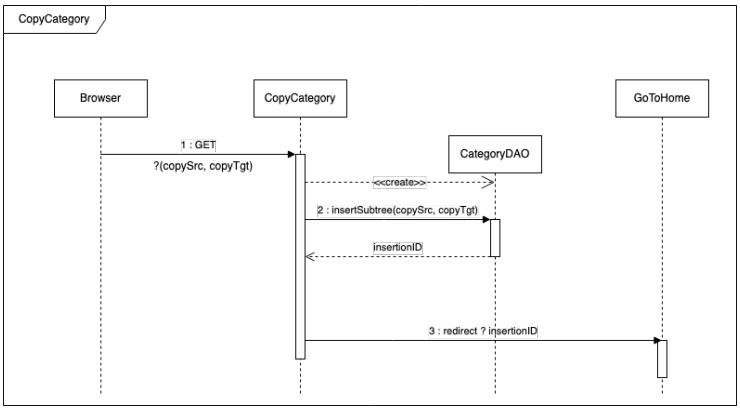

Figure 10: Interazione con la servlet CopyCategory

2 One-page RIA

2.1 Specifica aggiuntiva

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

- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina.
- 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.
- La funzione di copia di un sottoalbero è realizzata mediante drag&drop. A seguito del drop della radice del sottoalbero da copiare compare una finestra di dialogo con cui l'utente può confermare o cancellare la copia. La conferma produce l'aggiornamento solo a lato client dell'albero. La cancellazione riconduce allo stato precedente al drag&drop. A seguito della conferma compare un bottone SALVA che consente il salvataggio a lato server della tassonomia modificata.
- L'utente può cliccare sul nome di una categoria. A seguito di tale evento compare al posto del nome un campo di input contente la stringa del nome modificabile. L'evento di perdita del focus del campo di input produce il salvataggio nel database del nome modificato della categoria.

PAGES/VIEWS - VIEW COMPONENTS - EVENTS - ACTIONS

2.1.1 Dettagli di Implementazione

La funzionalità di copia *"locale"* del sottoalbero é stata gestita nel seguente modo:

- Una volta confermata la copia, il sottoalbero aggiunto viene mostrato localmente, evidenziato di rosso, ma senza la possibilità di interagirvi (copiarlo, creare figli di suoi nodi, cambiarne i nomi)
- Se viene salvato, sarà effettivamente aggiunto come parte interattiva.
- Se si prova a copiare un diverso sottoalbero, la conferma della copia porterà alla rimozione del precedente sottoalbero provvisorio.

2.2 Componenti Server-Side

- **Views**
  - login.html (include la registrazione)
  - home.html
- **DAO**
  - UserDAO
    - registerUser
    - getUserByEmail
    - checkCredentials
  - CategoryDAO
    - getChildCount
    - getCategoryTree
    - insertCategory
    - renameCategory
    - insertSubtree
- **Beans (con pacchetti immutabili)**
  - User/UserPacket
  - Category/CategoryPacket
- **Filters**
  - CheckLoggedIn
  - CheckLoggedOut
  - NoCache
- **Controllers[privileges]**
  - Login[not logged]
  - Register[not logged]
  - Logout[logged]
  - GetTaxonomy[logged]
  - CreateCategory[logged]
  - RenameCategory[logged]
  - CopyCategory[not logged]

2.3 Componenti Client-Side

- **Forms**
  Notificano asincronamente il server e riportano eventuali errori.
  - Login (login.html)
  - Register (login.html)
  - Create Category (home.html)
- **PageOrchestrator**
  Riunisce i componenti della home page, inizializzandoli ad ogni refresh.
  - **PageOrchestrator()**: inizializza componenti.
  - **refresh()**: mostra le componenti.
- **UserData**
  - **UserData()**: recupera dati da local storage.
  - **show()**: mostra i dati specifici all'utente.
- **Taxonomy**
  - **Taxonomy()**: inizializza componenti interattivi.
  - **show()**: recupera la tassonomia dal server.
  - **update()**: aggiorna la pagina con la tassonomia ricevuta.
  - **execute_copy()**: esegue una copia locale di un sottoalbero.

2.4 Eventi e Azioni

Client Side		Server Side	
Evento	**Azione**	**Evento**	**Azione**
login.html → login form → submit	Controllo Dati	POST email,pwd	Controllo Credenziali
login.html → register form → submit	Controllo Dati	POST email,name,pwd	Controllo Credenziali e aggiunta nuovo User
home.html → create form → submit	Controllo Dati e aggiorna view	POST name,parentId	Inserimento categoria
home.html → load	Aggiorna view con tassonomia	GET	Estrazione tassonomia
home.html → click su una categoria	Mostra box per rinominare	-	-
home.html → perdita focus del box per rinominare	Aggiorna view con nuovo nome	POST id,name	Modifica nome categoria
home.html → drag&drop	Chiedi conferma copia	-	-
home.html → conferma	Aggiorna view con copia locale	-	-
home.html → cancella	-	-	-
home.html → salva	Aggiorna tassonomia	POST src,tgt	Effettua copia
home.html → logout	-	GET	Termina sessione

2.5 Controller e Event Handler

Client Side		Server Side	
Evento	**Controllore**	**Evento**	**Controllore**
login.html → login form → submit	Function makeCall	POST email,pwd	Login(servlet)
login.html → register form → submit	Function makeCall	POST email,name,pwd	Register(servlet)
home.html → create form → submit	Function makeCall e Taxonomy.show()	POST name,parentId	CreateCategory(servlet)
home.html → load	PageOrchestrator.refresh()	GET	GetTaxonomy(servlet)
home.html → click su una categoria	Click listener di categoria	-	-
home.html → perdita focus del box per rinominare	Function makeCall	POST id,name	RenameCategory(servlet)
home.html → drag&drop	Ondrop listener di categoria	-	-
home.html → conferma	Taxonomy.execute_copy()	-	-
home.html → cancella	Click listener di Cancel	-	-
home.html → salva	Function makeCall	POST src,tgt	CopyCategory(servlet)
home.html → logout	Click listener di Logout	GET	Logout(servlet)

2.6 Application Design in IFML

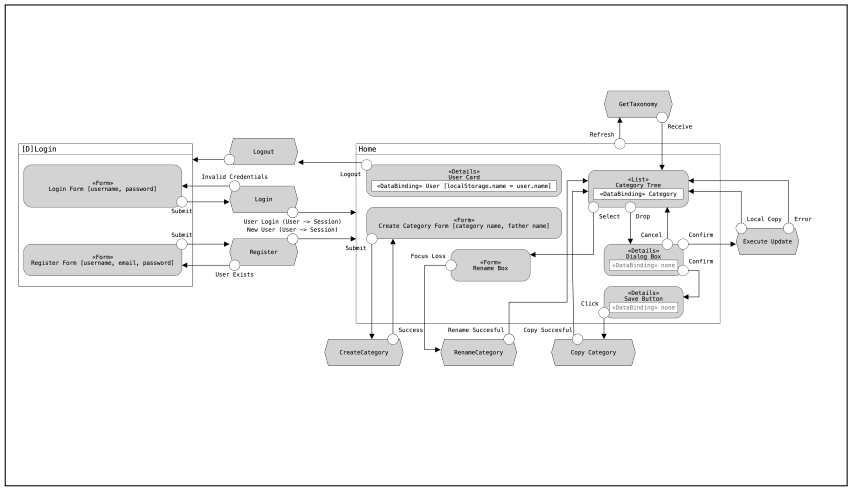

Figure 11: Diagramma IFML della versione RIA

2.7 Sequence Diagrams

La gestione degli errori è svolta in modo differente dalla versione HTML: tutte le chiamate di *makeCall()* hanno a disposizione una *errorDiv* (normalmente nascosta) in cui scrivere messaggi in caso di risposte di errore.
Come nel caso precedente, queste interazioni sono mostrate per la servlet **Login** (12) ed omesse nei diagrammi successivi.
Sono inoltre escluse le descrizioni dei filtri CheckLoggedIn e CheckLoggedOut, il cui funzionamento è rimasto inalterato, e del filtro NoCache, che aggiunge header per evitare caching client-side.
Nei diagrammi, le lifeline blu indicano componenti Server-Side.

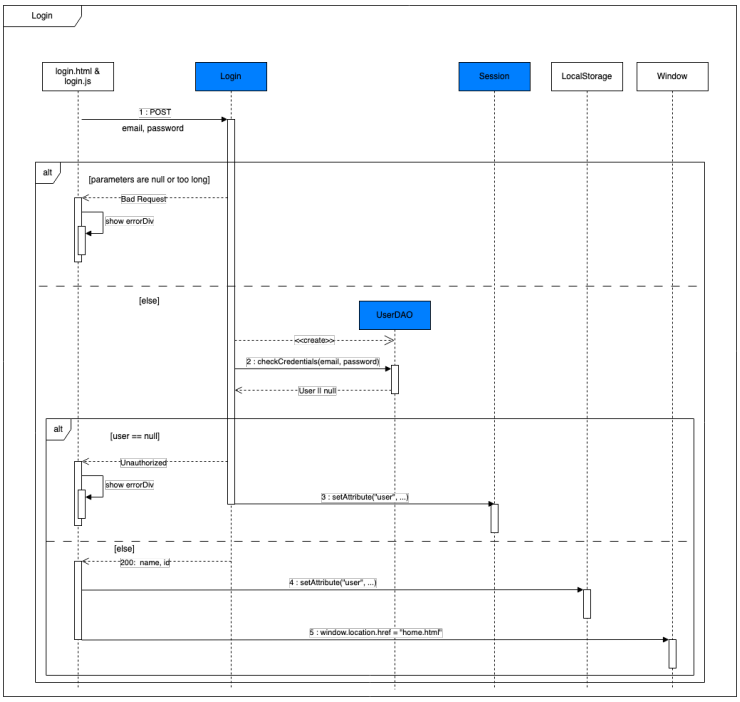

Figure 12: Interazione con la servlet Login

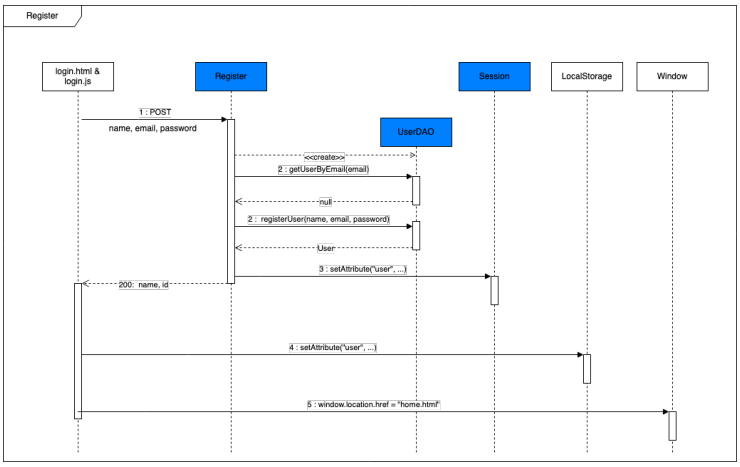

Figure 13: Interazione con la servlet Register

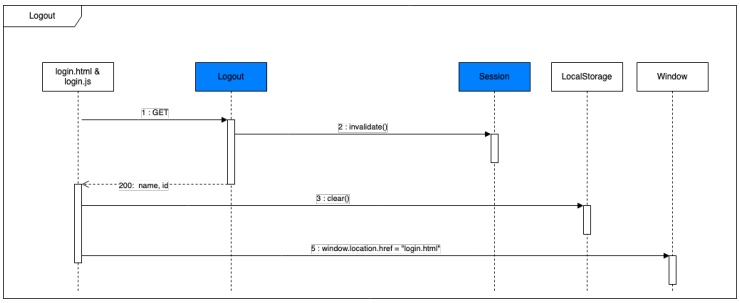

Figure 14: Interazione con la servlet Logout

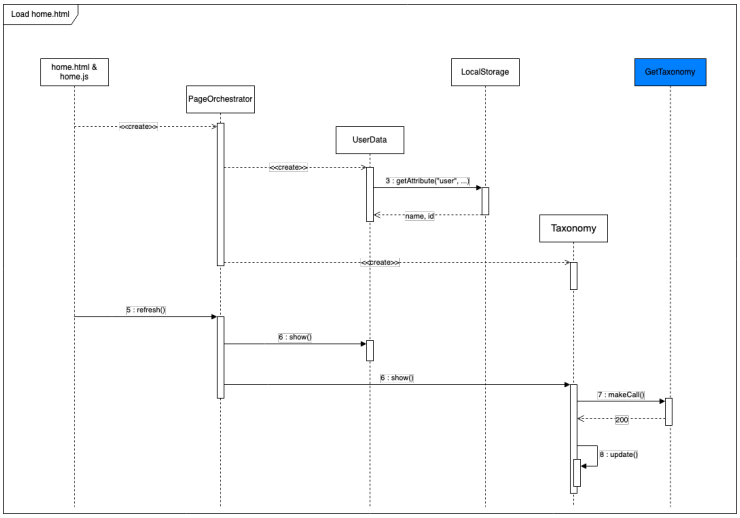

Figure 15: Interazioni a caricamento della Home

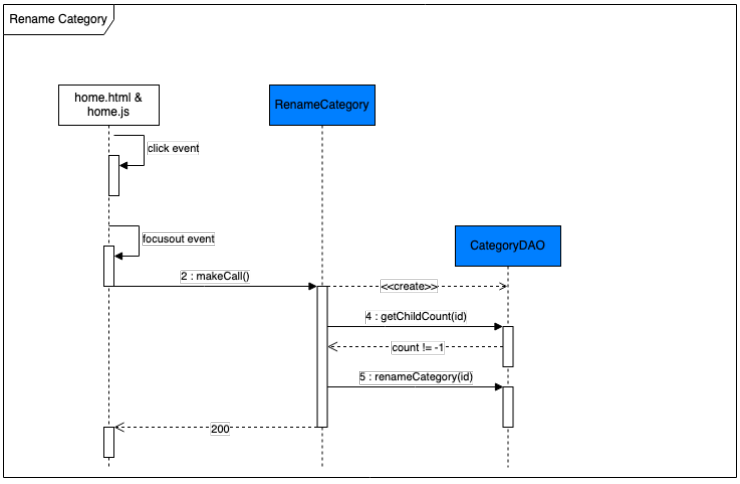

Figure 16: Interazione con la servlet RenameCategory

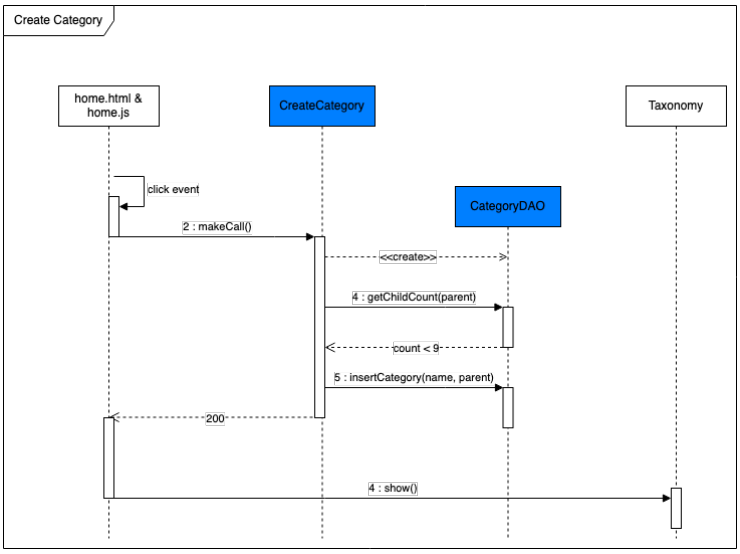

Figure 17: Interazione con la servlet CreateCategory

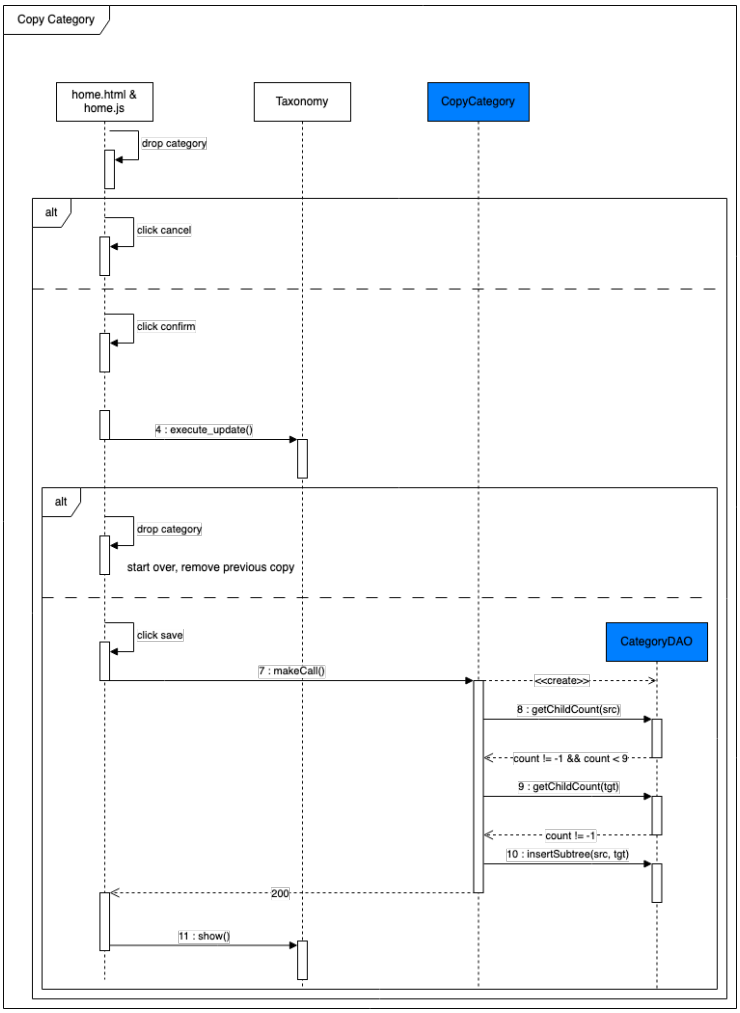

Figure 18: Interazione con la servlet CopyCategory