

Documentazione progetto

Tecnologie Informatiche per il Web

Gruppo 117

1 - Specifica iniziale

Versione RIA

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. Un esempio di un ramo dell'albero è il seguente:

9 Materiali solidi >> copia
91 Materiali inerti >> copia
911 Inerti da edilizia >> copia
9111 Amianto >> copia
91111 Amianto in lastre >> copia
91112 Amianto in frammenti >> copia
9112 Materiali cementizi >> copia
912 Inerti ceramici >> copia
9121 Piastrelle >> copia
9122 Sanitari >> copia
92 Materiali ferrosi >> copia

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". Ad esempio, a seguito del click sul link "copia" associato alla categoria "9111 Amianto" l'applicazione visualizza l'albero come segue.

9 Materiali solidi >> copia qui
91 Materiali inerti >> copia qui
911 Inerti da edilizia >> copia qui
9111 Amianto
91111 Amianto in lastre
91112 Amianto in frammenti
9112 Materiali cementizi >> copia qui
912 Inerti ceramici >> copia qui
9121 Piastrelle >> copia qui

9122 Sanitari >> copia qui
92 Materiali ferrosi >> copia qui

La selezione di un link “copia qui” comporta l’inserimento di una copia del sottoalbero come ultimo figlio della categoria destinazione. Ad esempio, la selezione del link “copia qui” della categoria “9 Materiali solidi” comporta la seguente modifica dell’albero:

9 Materiali solidi>> copia
91 Materiali inerti>> copia
911 Inerti da edilizia >> copia
9111 Amianto
91111 Amianto in lastre
91112 Amianto in frammenti
9112 Materiali cementizi >> copia
912 Inerti ceramici >> copia
9121 Piastrelle >> copia
9122 Sanitari >> copia
92 Materiali ferrosi >> copia
93 Amianto >> copia
931 Amianto in lastre >> copia
932 Amianto in frammenti >> copia

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 sottoalbero al primo livello della tassonomia (se non esistono già 9 nodi al primo livello della tassonomia).

Versione con JavaScript

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 contenente 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.

2 - Diagramma ER

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. Un esempio di un ramo dell'albero è il seguente:

[...]

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).

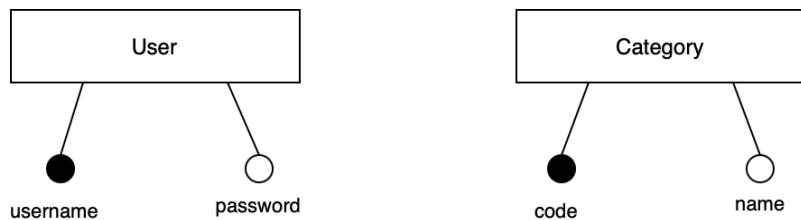
[...]

Dove si sono adottate le seguenti convenzioni: **entities**, **relationships** & **attributes**.

A partire dal testo sono state fatte le seguenti considerazioni:

1. Gli attributi "*categoria padre*" e "*codice numerico*" possono essere compressi in unico attributo stringa contenente, di fatto, la sequenza numerica identificante il percorso completo all'interno dell'albero gerarchico fra radice e nodo della categoria.
2. La ricerca di un intero sotto-albero avviene semplicemente con delle query che ricerchino i le tuple con codici contenenti al loro interno il codice della radice del sotto-albero richiesto concatenata ad una qualunque altra sequenza numerica.
3. Effettuando una query ordinata lessicograficamente rispetto al codice produce automaticamente una lista di categorie strutturate ad albero (analogia ad una visita DFS)
4. La creazione di una copia di un sotto-albero sotto una nuova radice avviene semplicemente clonando tutte le tuple del sotto-albero interessato e sostituendo la porzione di codice contenente la radice iniziale con la nuova radice concatenata una nuova cifra (che è pari al numero di figli diretti già presenti sotto la nuova radice incrementato di 1. È necessario verificare non si superi il limite di 9 figli diretti per nodo).
5. Poiché un'operazione di copia di sotto-albero viene salvata nella base di dati solo al click di un opportuno bottone di salvataggio, onde evitare conflitti e/o inconsistenze fra richieste (es.: , dall'inizio di una sequenza di copie di sotto-alberi fino al salvataggio / cancellazione le funzionalità di rinominazione di una categoria e creazione di nuove categorie devono essere **sospese** localmente.

Ottieniamo, a questo punto, il seguente diagramma ER:



Non sono presenti relazioni fra le due entità per il fatto che **tutti** gli utenti hanno accesso a tutte le categorie. Non esistendo, pertanto, un concetto di “privilegi” / “permessi”, le due entità sono completamente indipendenti l’una dall’altra.

In fase di sviluppo è stato creato uno schema dedicato “tiw” per la base di dati, con le due tabelle “user” e “category” al suo interno. Il progetto logico è, pertanto, semplicemente:

```
-- DB reset (if schema is already existing):
DROP SCHEMA IF EXISTS tiw;

-- Create main schema for project
CREATE SCHEMA IF NOT EXISTS tiw;

-- Create tables
CREATE TABLE IF NOT EXISTS tiw.`user` (
    `username`  VARCHAR(64)      PRIMARY KEY,
    `password`  VARCHAR(64)      NOT NULL
);

CREATE TABLE IF NOT EXISTS tiw.category (
    `code`       VARCHAR(64)      PRIMARY KEY,
    `name`       VARCHAR(64)      NOT NULL
);
```

3 - Diagrammi IFML

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. Un esempio di un ramo dell'albero è il seguente:

9 Materiali solidi >> copia
91 Materiali inerti >> copia
911 Inerti da edilizia >> copia
9111 Amianto >> copia
91111 Amianto in lastre >> copia
91112 Amianto in frammenti >> copia
9112 Materiali cementizi >> copia
912 Inerti ceramici >> copia
9121 Piastrelle >> copia
9122 Sanitari >> copia
92 Materiali ferrosi >> copia

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". Ad esempio, a seguito del click sul link "copia" associato alla categoria "9111 Amianto" l'applicazione visualizza l'albero come segue.

[...]

La selezione di un link "copia qui" comporta l'inserimento di una copia del sottoalbero come ultimo figlio della categoria destinazione. Ad esempio, la selezione del link "copia qui" della categoria "9 Materiali solidi" comporta la seguente modifica dell'albero:

[...]

~~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 sottoalbero al primo livello della tassonomia (se non esistono già 9 nodi al primo livello della tassonomia).~~

Versone con JavaScript

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

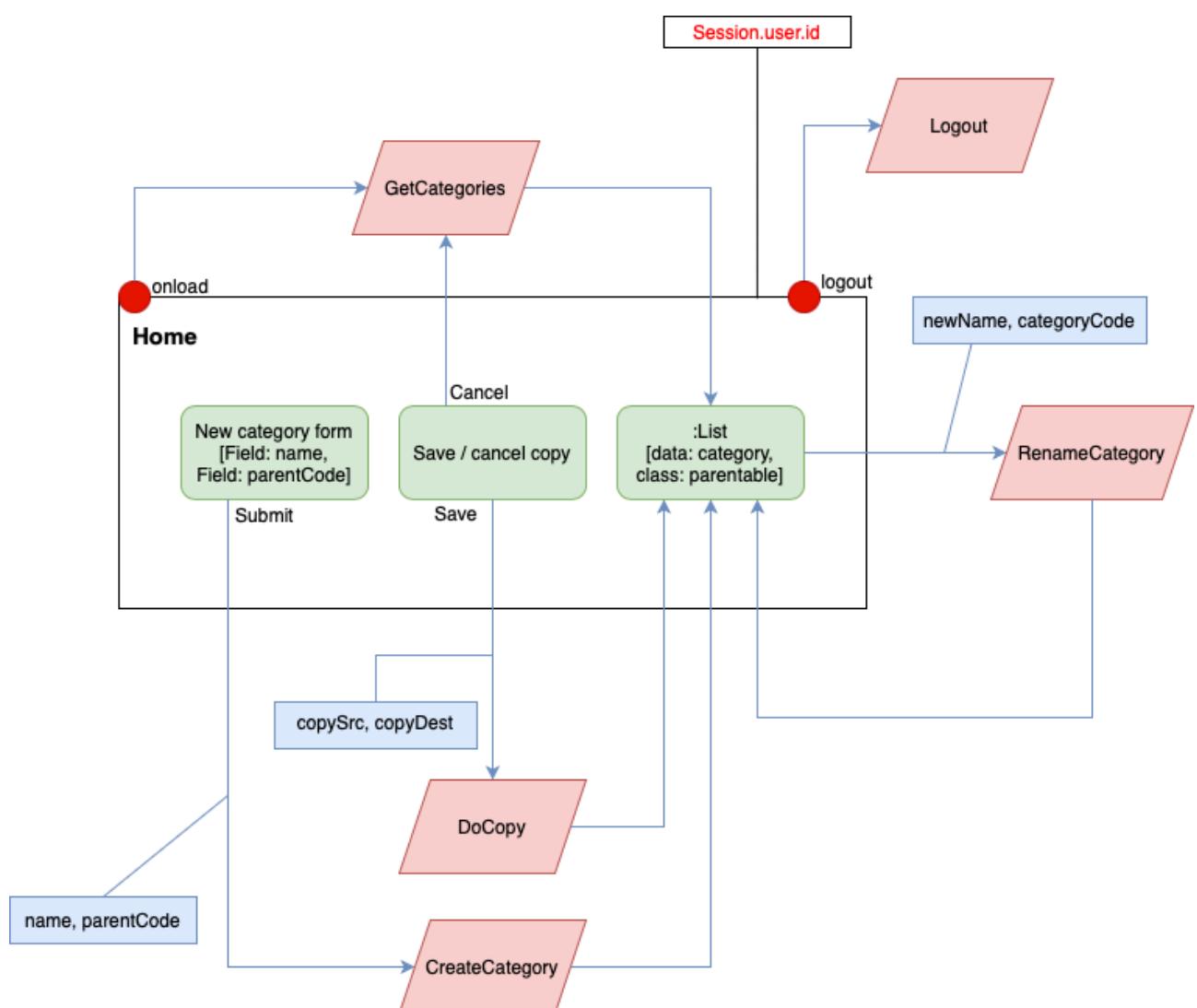
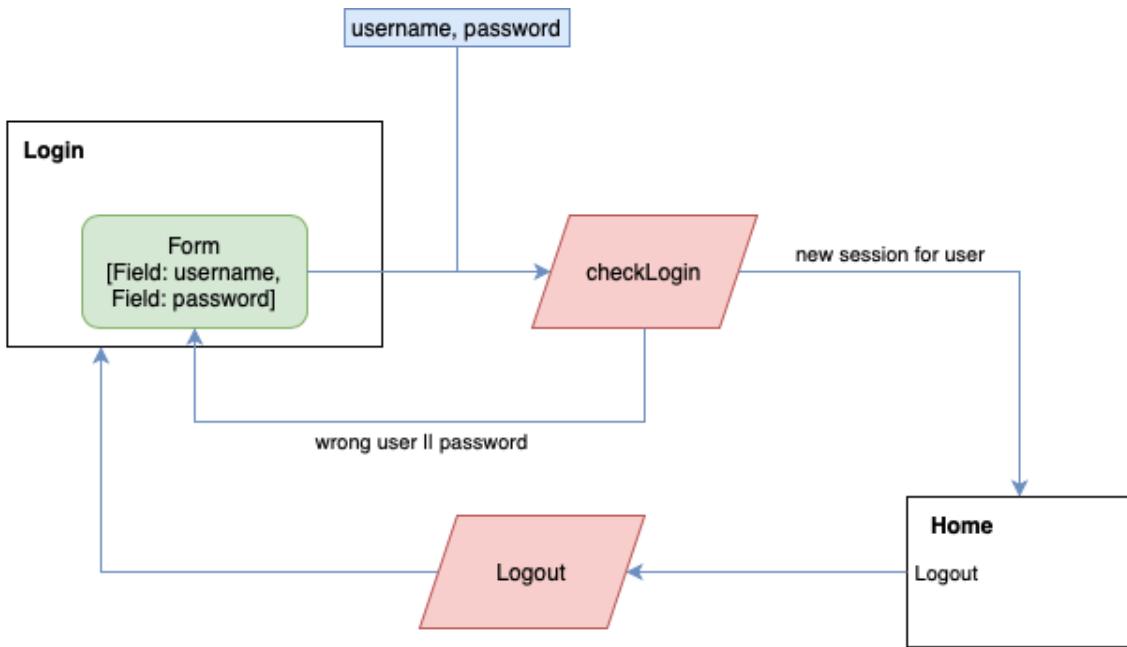
- Dopo il login dell'utente, l'intera applicazione è realizzata con **un'unica pagina**.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- La funzione 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.

Dove si sono adottate le seguenti convenzioni: **Views**, **View components**, **Actions** & **Events**.

Dal testo si possono inferire le seguenti informazioni **aggiuntive**:

1. È presente anche una view per il login dell'utente.
2. L'operazione di copia di un sotto-albero oltre a mostrare un tasto salva mostra anche un tasto di cancellazione dell'operazione.
3. Durante la rinominazione di una categoria se il nuovo nome coincide con quello precedente si evita di inviare alcuna richiesta al server. Di più, se il campo di input viene lasciato vuoto, alla perdita del focus l'operazione di rinominazione non viene effettuata.
4. Nella pagina HOME dovrà essere possibile effettuare il logout.
5. Durante la creazione di una nuova categoria, la categoria padre che viene selezionata nel Form deve esistere. Inoltre, non deve avere già 9 figli diretti.

Otteniamo, a questo punto, i seguenti diagrammi IFML:



4 - Events & Actions

| Client side | | Server side | |
|--------------------------|------------------------|--------------------|-------------------------|
| Event | Action | Event | Actions |
| Index | - | - | - |
| Login form > submit | Validate form | POST uname, pass | Create new session |
| /home.html > onload | Update view with data | GET | Read full category tree |
| Create category > submit | Validate form | POST name, parent | Create new category |
| Rename category | Check !empty && !same | POST newName, code | Update category name |
| Copy subtree (drag&drop) | Check if valid | - | - |
| Cancel copied subtree | Revert view data | GET categories | Read full category tree |
| Save copied subtree | Request copy to server | POST src, dest | Create subtree copy |
| Logout | Clear localStorage | GET | Session termination |

| Client side | | Server side | |
|--------------------------|--------------------------|--------------------|--------------------------|
| Event | Controller | Event | Controller |
| Index | - | - | Forward index.html |
| Login form > submit | Fetch | POST uname, pass | CheckLogin (servlet) |
| /home.html > onload | Function fetchCategories | GET categories | GetCategories (servlet) |
| Create category > submit | Fetch | POST name, parent | CreateCategory (servlet) |
| Rename category | Function renameCategory | POST newName, code | RenameCategory (servlet) |
| Copy subtree (drag&drop) | Event handling function | - | - |
| Cancel copied subtrees | Fetch | GET categories | GetCategories (servlet) |
| Save copied subtrees | Fetch | POST src, dest | DoCopy (servlet) |
| Logout | Redirect to "/logout" | GET | Logout (servlet) |

Eccetto la richiesta dell'indice (indirizzo "/"), per tutte le richieste lato server viene verificato se è presente una sessione attiva per il client. Inoltre, qualora necessario, tutti i dati da trasmettere al server vengono verificati sia lato client che lato server.

Questi due aspetti sono stati omessi nella tabella per ordine e semplicità.

5 - Server side: *DAO & Model Objects*

- **Servlets:**

1. CheckLogin
2. CreateCategory
3. DoCopy
4. GetCategories
5. Logout
6. RenameCategory

- **Beans:**

1. User
2. Category

- **DAO:**

1. UserDAO
 1. checkCredentials(username, password)
2. CategoryDAO
 1. getFullCategoryTree()
 2. createNewCategory(name, parent)
 3. renameCategory(code, newName)
 4. copySubtree(subtreeRootCode, newParentCode)
 5. getNumberOfDirectChildren(code)
 6. doesCategoryExist(code)

- **Filters:**

3. LoginChecker: impedisce di effettuare richieste se privi di sessione valida
4. NoCacher: impedisce il caching del file statico home.html

6 - Client side: *Views & View Components*

- **Index:**

1. **Login form**

1. Submit event: I dati inseriti sono controllati e successivamente inviati al server.

- **Home:**

1. **Greeter message**

2. **CreateCategory form**

1. Submit event: i dati inseriti sono controllati e successivamente inviati al server.

3. **Category-Tree:** tassonomia delle categorie

1. Category "*click*" event handler: funzione permette di rinominare una categoria.

2. Category input "*blur*" event handler: funzione che memorizza il nuovo nome assegnato ad una categoria nella base dati.

3. "*dragstart*" event handler: funzione invocata all'inizio di un evento di trascinamento di un sotto-albero.

4. "*dragover*" / "*dragleave*" event handler: funzione incaricate di evidenziare i sotto-alberi sotto al cursore durante il trascinamento di un sotto-albero, e duale per quando il cursore lascia il sotto-albero.

5. "*drop*" event handler:

6. Save "*click*" event handler: la copia di sotto-albero locale viene memorizzata nella base dati.

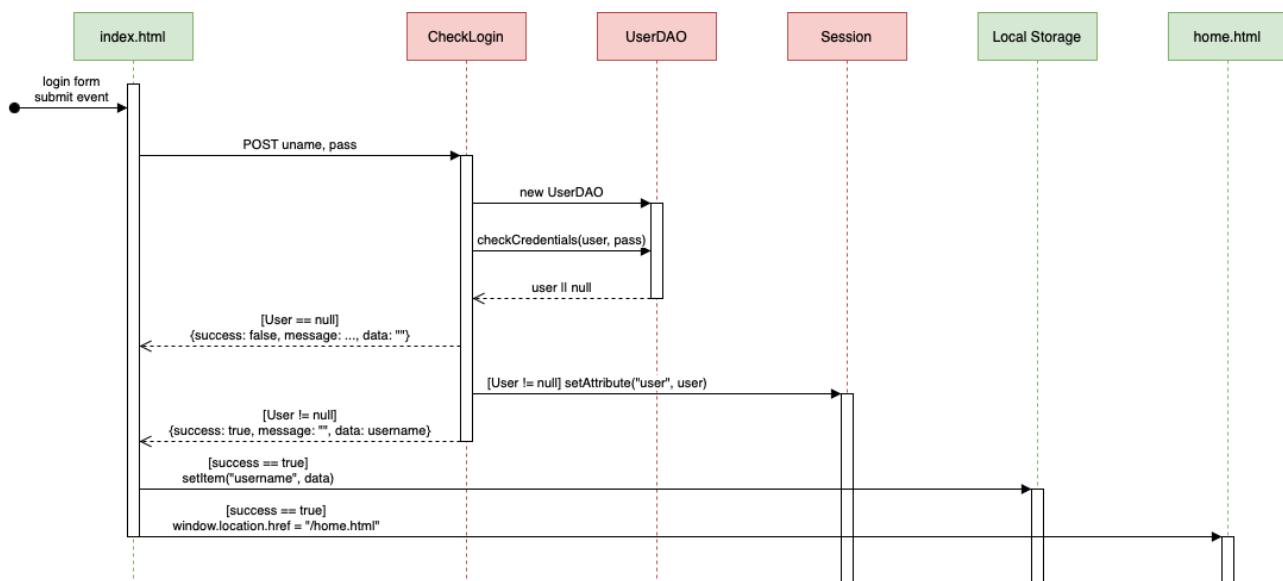
7. Cancel "*click*" event handler: la view viene riportata allo stato precedente al drag&drop.

4. **Logout button**

7 - Sequence diagrams

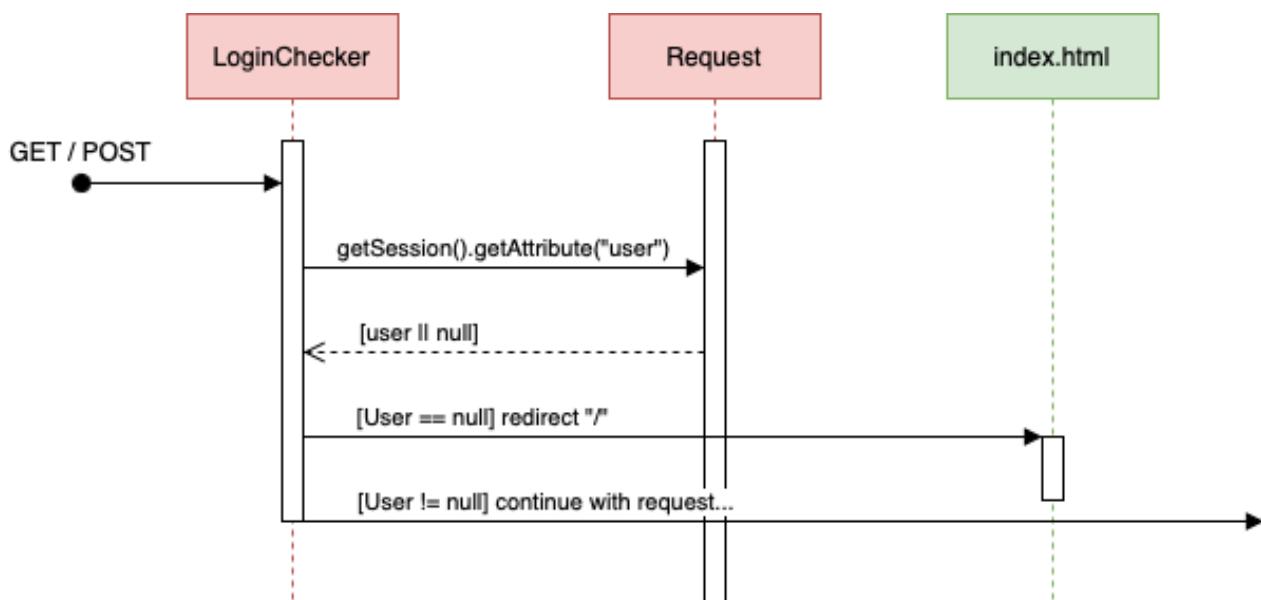
Nei diagrammi che seguono abbiamo adottato la seguente convenzione: **Server Side** e **Client Side**.

7.1 - Login event



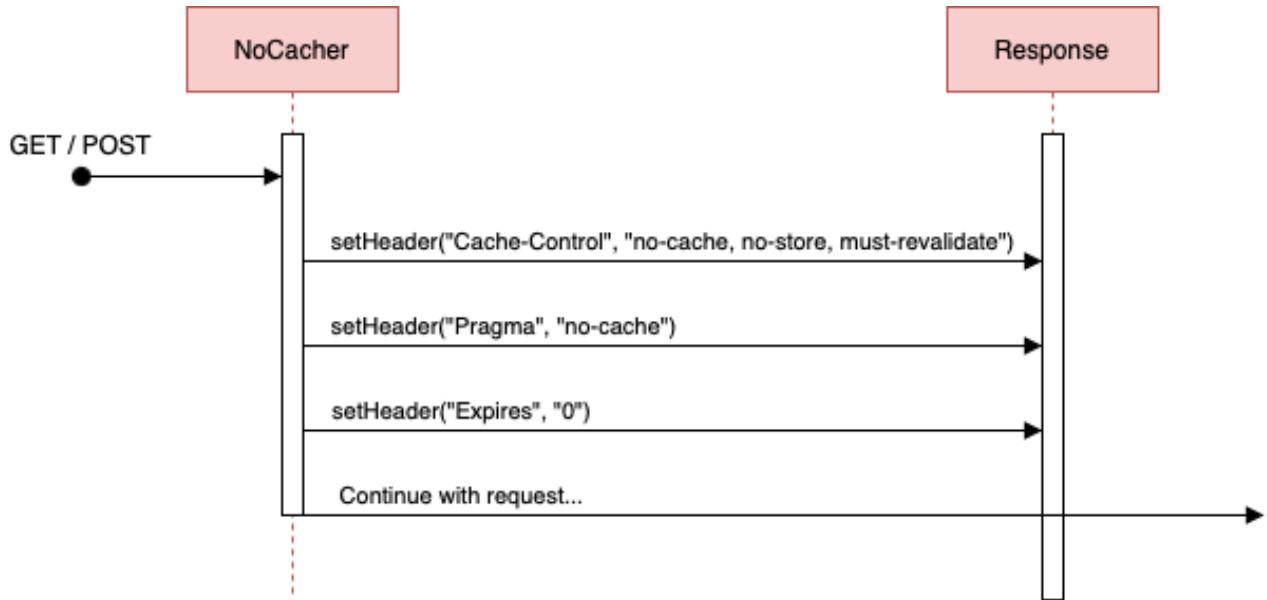
7.2.1 - Filters/LoginChecker

Qualunque richiesta, eccetto l'indice ("`/`"), passa sempre attraverso il filtro LoginChecker. Esso effettuerà le seguenti operazioni:



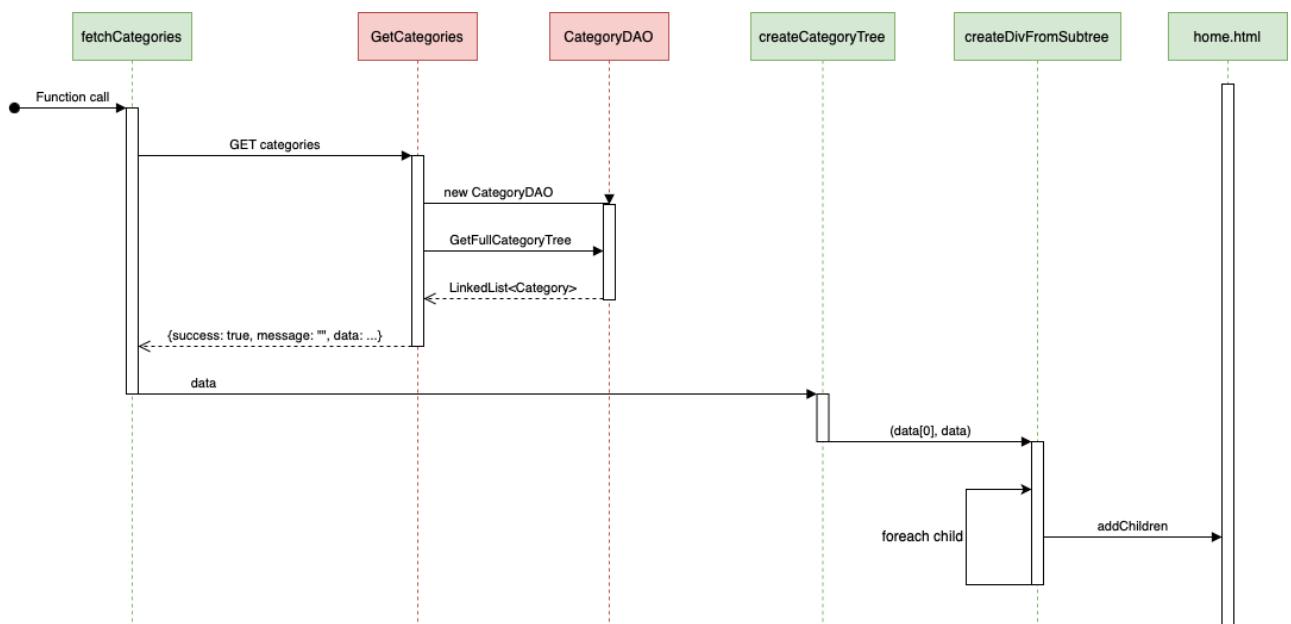
7.2.2 - Filters/NoCacher

Ogniqualvolta viene richiesto “/home.html” si passa attraverso il seguente filtro:

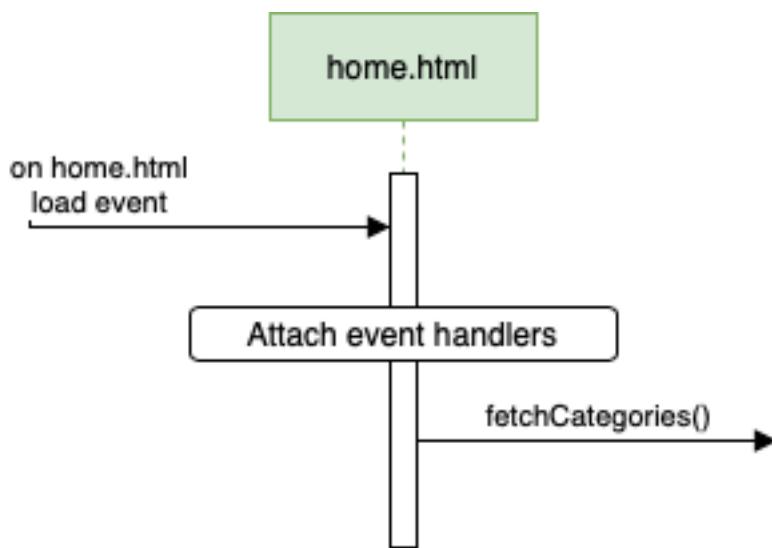


7.3 - Get all categories

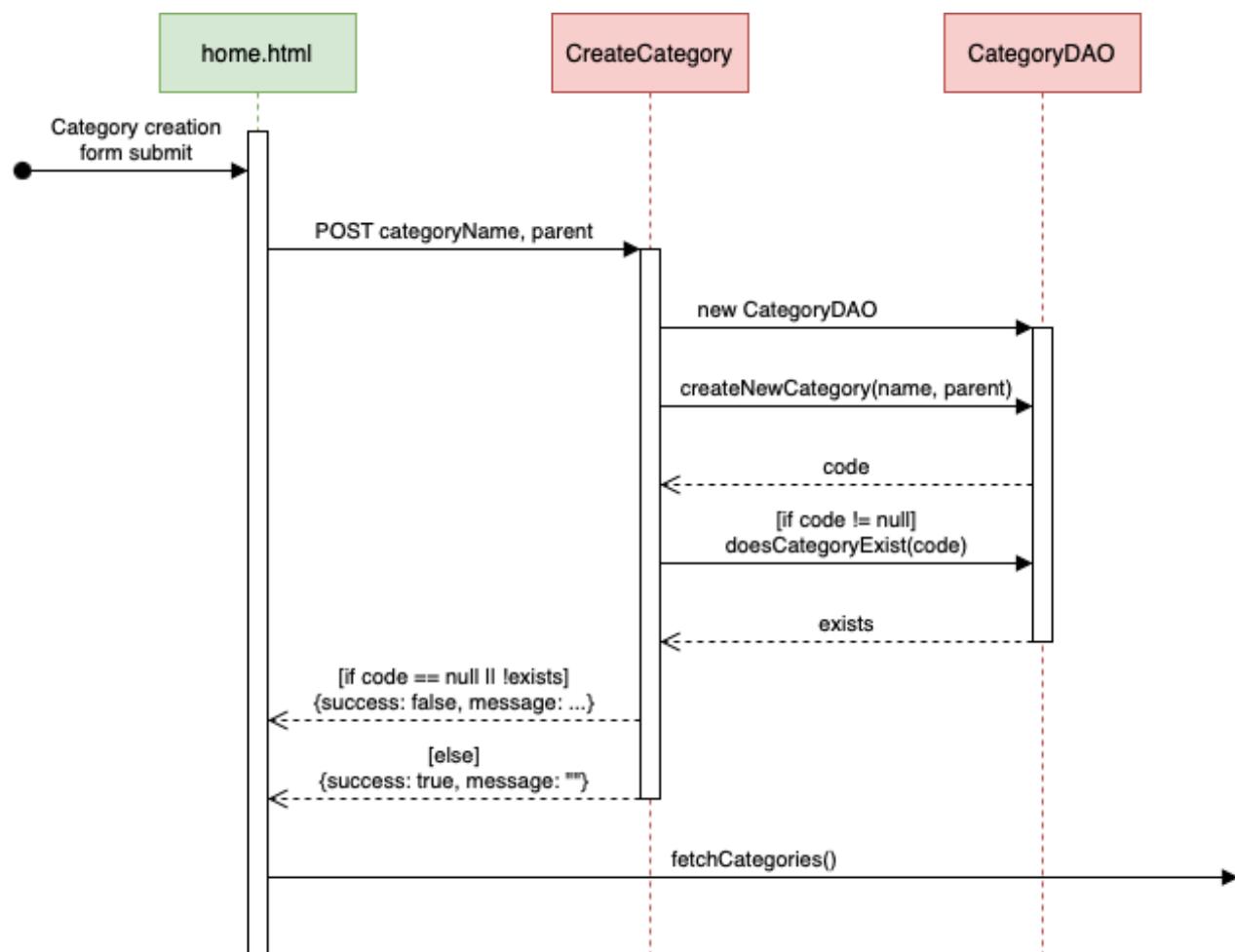
Il seguente sequence diagram mostra tutte le operazioni svolte al fine di scaricare l'albero di categorie attualmente memorizzato nella base dati. Quest'operazione è molto frequente e viene richiamata in più sequence diagrams successivi



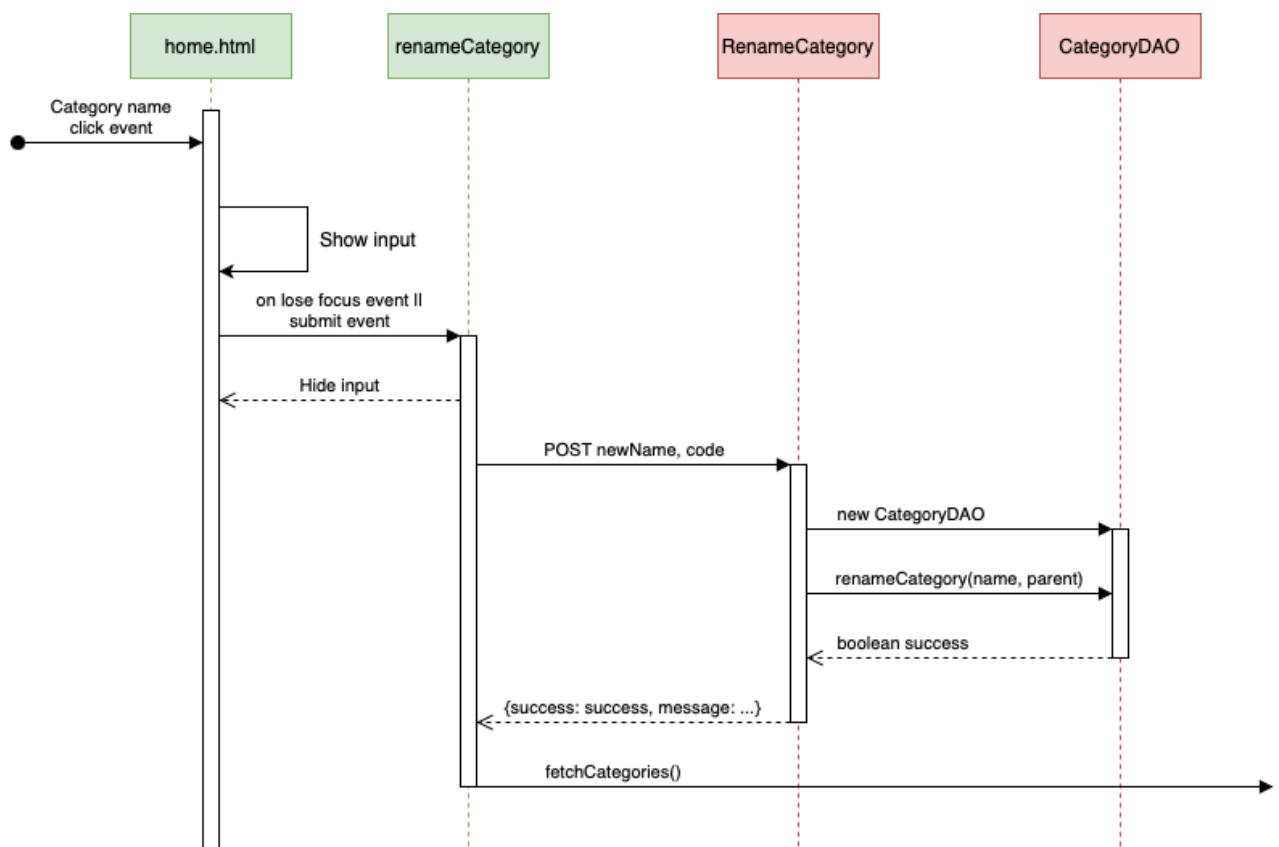
7.4 - home.html *onload* event



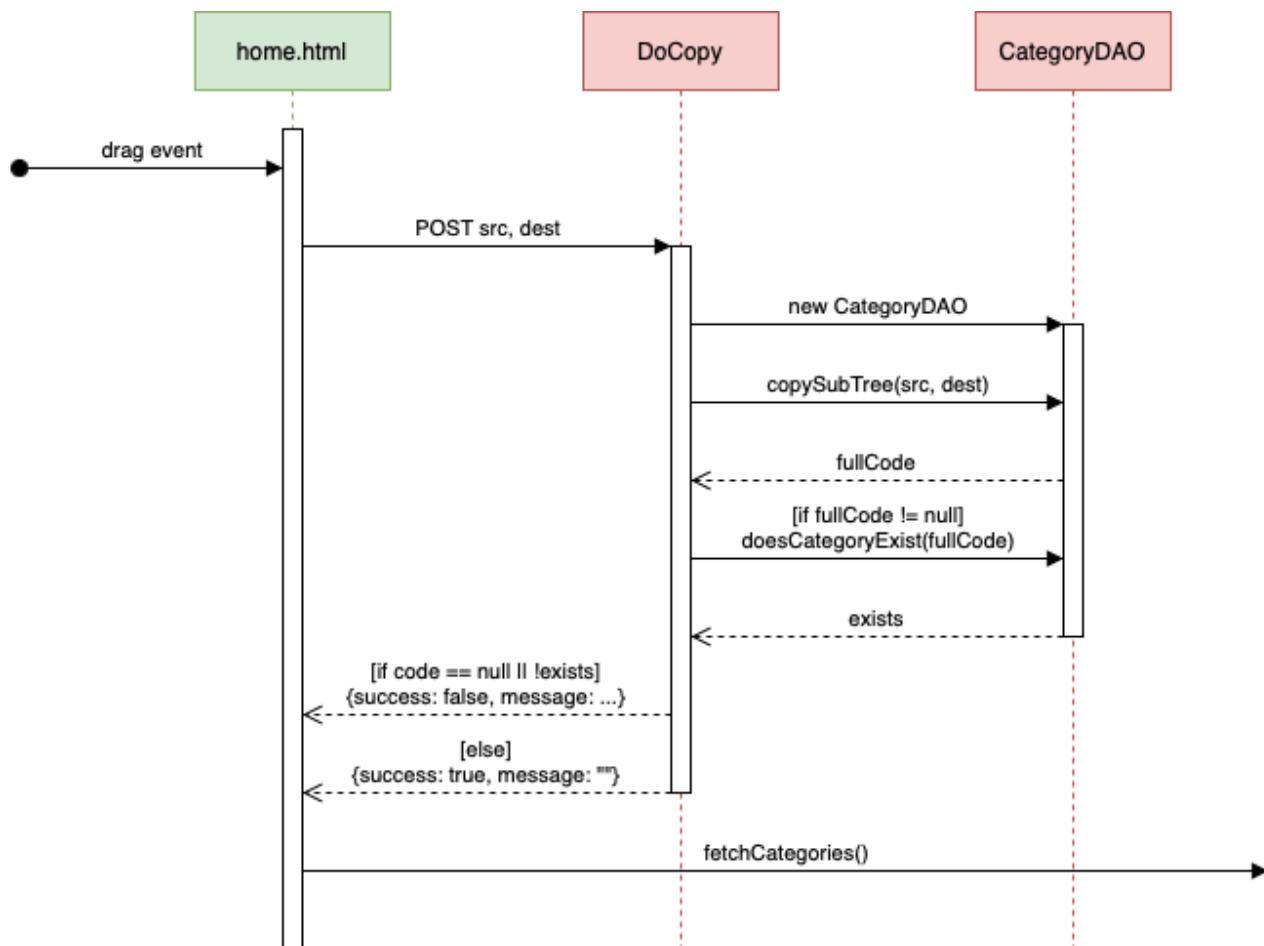
7.5 - Create new category



7.6 - Rename category



7.7- Sub-tree copy



7.8 - Logout

