

Contents

| | | |
|----------|---|----------|
| 1 | Tecnologie Informatiche per il Web - Documentazione Progetto | 2 |
| 1.1 | Introduzione | 2 |
| 1.2 | <i>Pure-HTML</i> | 3 |
| 1.2.1 | Specifica | 3 |
| 1.2.2 | Analisi della specifica | 4 |
| 1.2.3 | Analisi dei requisiti dell'applicazione | 6 |
| 1.2.4 | Design dell'applicazione | 7 |
| 1.2.5 | <i>Sequence Diagrams</i> : Evento Login | 9 |
| 1.2.6 | <i>Sequence Diagrams</i> : Evento GoToHome | 10 |
| 1.2.7 | <i>Sequence Diagrams</i> : Evento CreateCategory | 11 |
| 1.2.8 | <i>Sequence Diagrams</i> : Evento CopyCategory | 12 |
| 1.2.9 | <i>Sequence Diagrams</i> : Evento InsertCopiedCategory | 13 |
| 1.2.10 | <i>Sequence Diagrams</i> : Evento Logout | 14 |
| 1.3 | <i>Rich Internet Application</i> | 15 |
| 1.3.1 | Completamento della specifica | 15 |
| 1.3.2 | Design dell'applicazione | 16 |
| 1.3.3 | Eventi ed Azioni | 17 |
| 1.3.4 | Controller - <i>Event Handler</i> | 18 |
| 1.3.5 | <i>Event</i> e <i>View Dynamics</i> | 19 |
| 1.3.6 | <i>View</i> e <i>View Components</i> - client side | 21 |
| 1.3.7 | <i>Sequence Diagrams</i> : | 21 |

1 Tecnologie Informatiche per il Web - Documentazione Progetto

1.1 Introduzione

Documentazione relativa al progetto finale del corso “**Tecnologie Informatiche per il Web**”, professore Fraternali Piero.

Studenti De Ciechi Samuele e Deidier Simone, gruppo 9.

Questo file contiene la documentazione relativa sia al progetto *Pure-HTML* sia al progetto *Rich Internet Application*.

Anno Accademico 2022/2023

1.2 *Pure-HTML*

1.2.1 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. 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.

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*).

1.2.2 Analisi della specifica

1.2.2.1 Analisi del testo per la creazione del database Un'applicazione permette all'**utente** 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 (\Rightarrow *username* e *password*), 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. . .

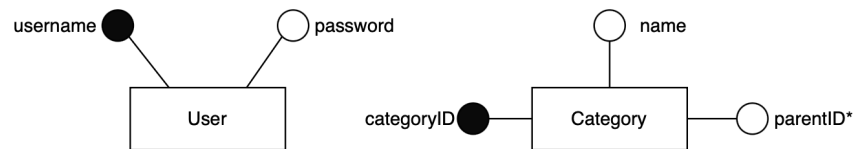


Figure 1: *Diagramma E-R del database utilizzato*

1.2.2.2 Creazione del database in MySQLWorkbench

```
CREATE TABLE Category (  
    categoryID bigint NOT NULL,  
    name varchar(45) NOT NULL,  
    parentID bigint DEFAULT NULL,  
    PRIMARY KEY (categoryID)  
)
```

```
CREATE TABLE User (  
    username varchar(45) NOT NULL,  
    password varchar(45) NOT NULL,  
    PRIMARY KEY (username)  
)
```

1.2.2.3 Note relative alla progettazione del database Abbiamo deciso di utilizzare *bigint* per *categoryID* e *parentID* così da permettere una maggiore libertà agli utenti nel creare diversi annidamenti di categorie, rispetto alla quantità di sottocategorie che avrebbero potuto creare con un semplice *int*.

Abbiamo inoltre scelto di non mettere una tabella intermedia fra le due categorie. Questo è dovuto al fatto che tutti gli utenti possono vedere tutte le categorie e quindi non c'è motivo di tenere traccia di chi crea una categoria o di dare visibilità ad alcune categorie a solo alcuni utenti.

1.2.3 Analisi dei requisiti dell'applicazione

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. 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.*

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*): pagina di login, pagina HOME.

| View components | Events | Action |
|------------------|----------------------------|----------------|
| Grassetto | Grassetto e corsivo | Corsivo |

1.2.4 Design dell'applicazione

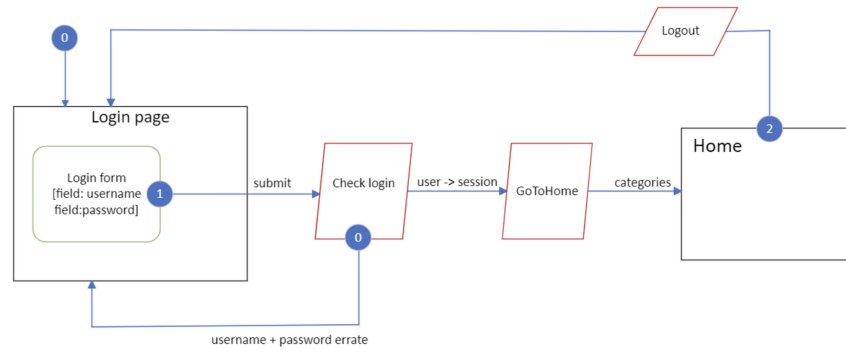


Figure 2: *Diagramma IFML del login.*

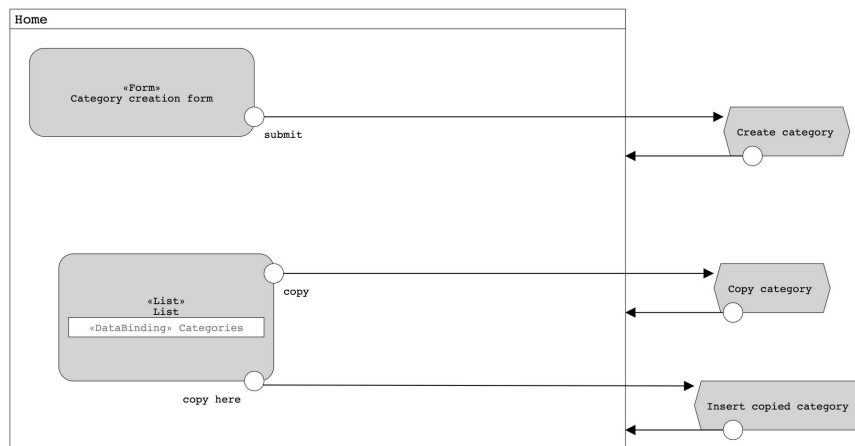


Figure 3: *Diagramma IFML della pagina HOME.*

1.2.4.1 Componenti

- Model Objects (*Beans*):
 - User
 - Category
- Data Access Objects (*Classes*):
 - UserDao
 - * checkLogin(username, password)
 - * registerNewUser(username, password)
 - CategoriesDao
 - * findAllCategories()
 - * createCategory(name, parentID)
 - * findLastChildrenID(categoryList, parentID)
 - * checkExistingCategoryFromID(categoryID)
 - * findSubCategories(categoryID)
 - * toCopyList(categoryID)
 - * insertCopiedCategory(categoryID, parentID)
 - * addCategoryInDatabase(newID, name, newParentID)
 - * orderCategoriesList(unorderedList, parentID)
- Controllers (*Servlets*):
 - GoToHome
 - CheckLogin
 - Logout
 - CreateCategory
 - CopyCategory
 - InsertCopiedCategory
- Filters:
 - NoCacheFilter
 - UserChecker
- View (*Templates*):
 - Login page (*index.html*)
 - Home

1.2.5 Sequence Diagrams: Evento Login

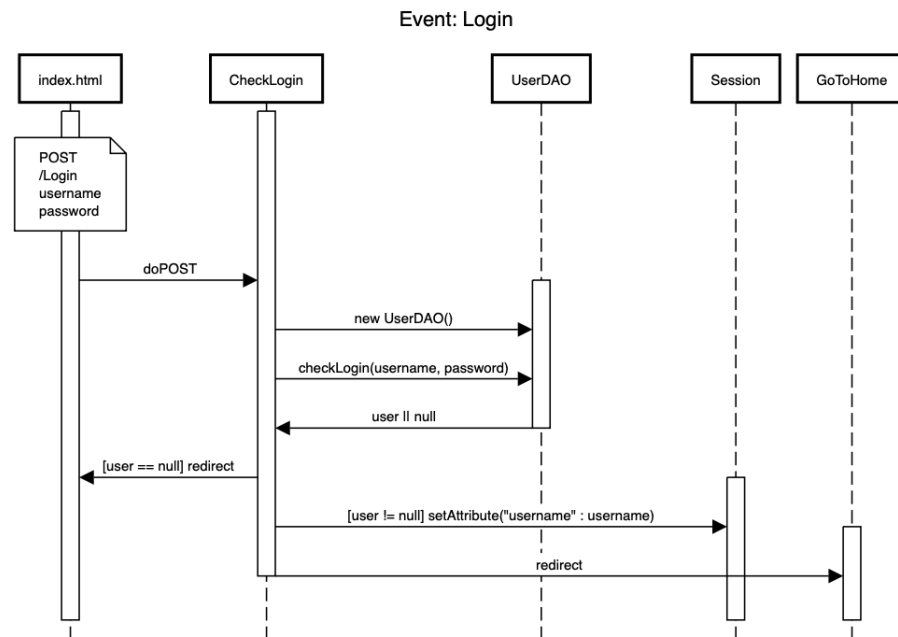


Figure 4: Sequence diagram dell'evento Login.

1.2.6 Sequence Diagrams: Evento GoToHome

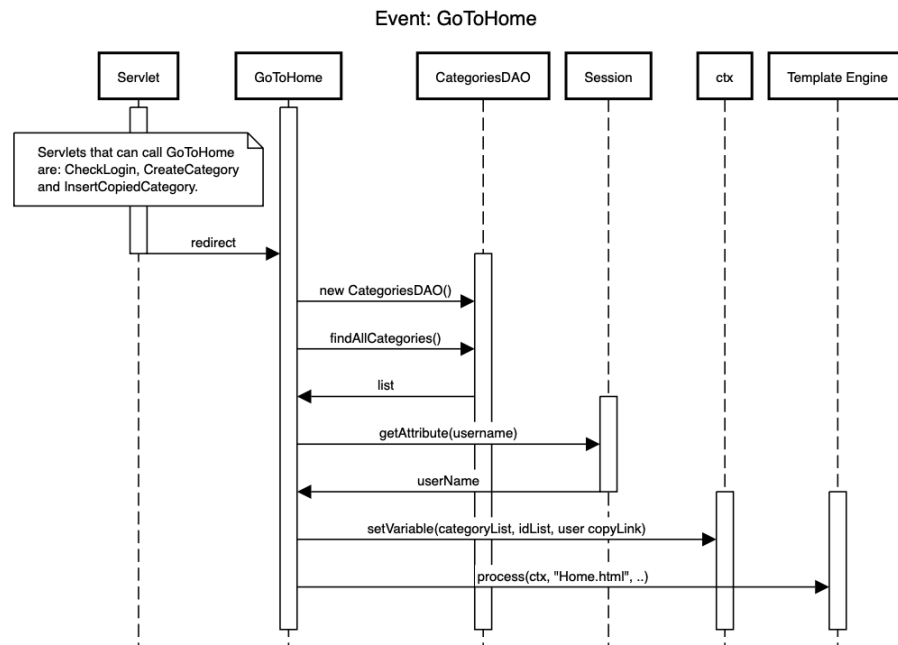


Figure 5: Sequence diagram dell'evento GoToHome.

1.2.7 Sequence Diagrams: Evento CreateCategory

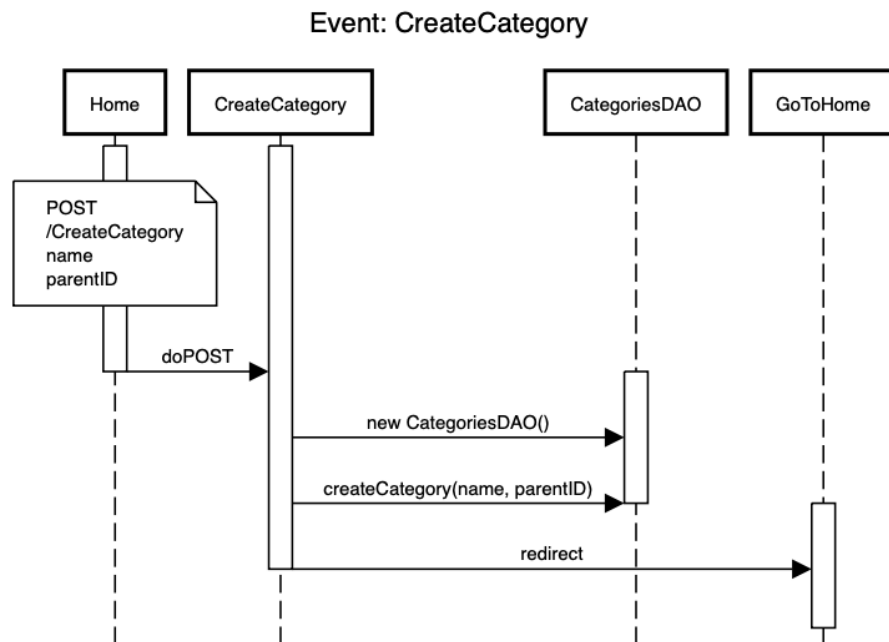


Figure 6: *Sequence diagram dell'evento CreateCategory.*

1.2.8 Sequence Diagrams: Evento CopyCategory

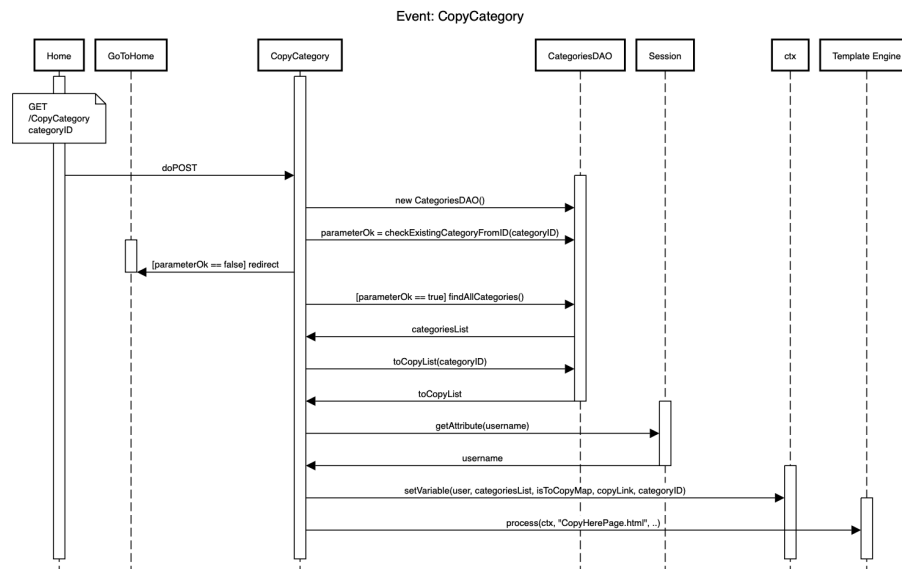


Figure 7: Sequence diagram dell'evento CopyCategory.

1.2.9 Sequence Diagrams: Evento InsertCopiedCategory

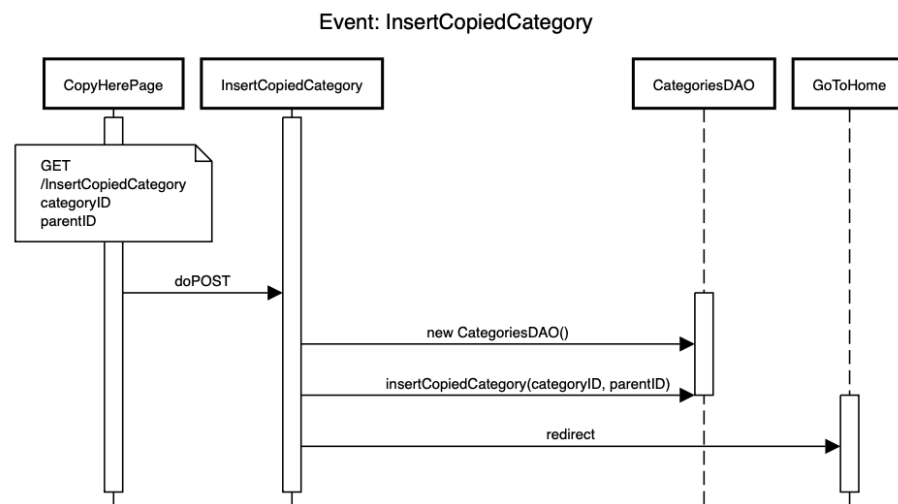


Figure 8: Sequence diagram dell'evento *InsertCopiedCategory*.

1.2.10 Sequence Diagrams: Evento Logout

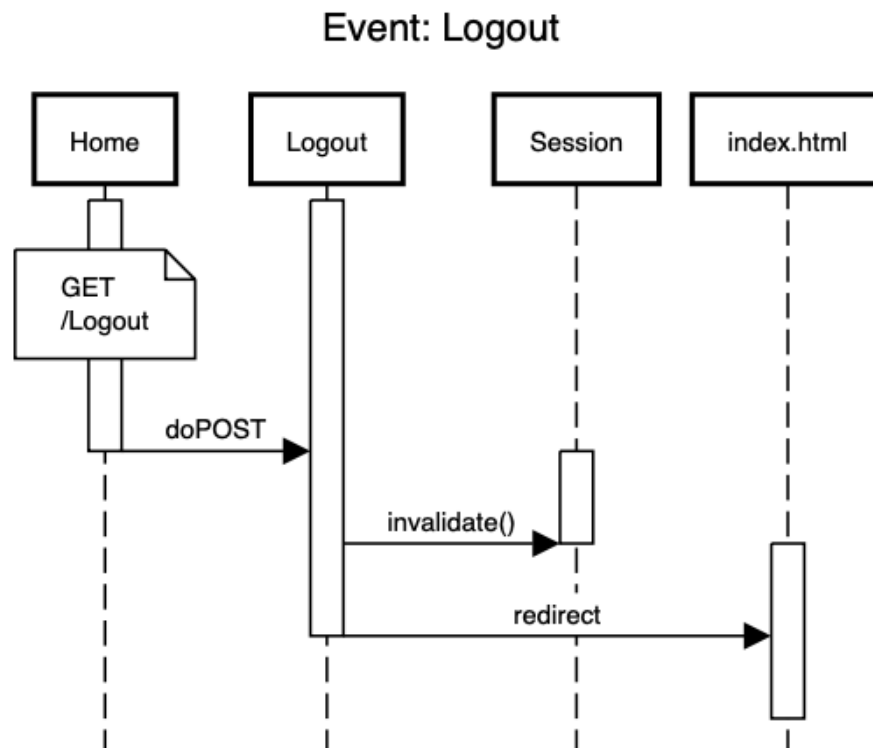


Figure 9: Sequence diagram dell'evento Logout.

1.3 *Rich Internet Application*

1.3.1 Completamento della specifica

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.

1.3.2 Design dell'applicazione

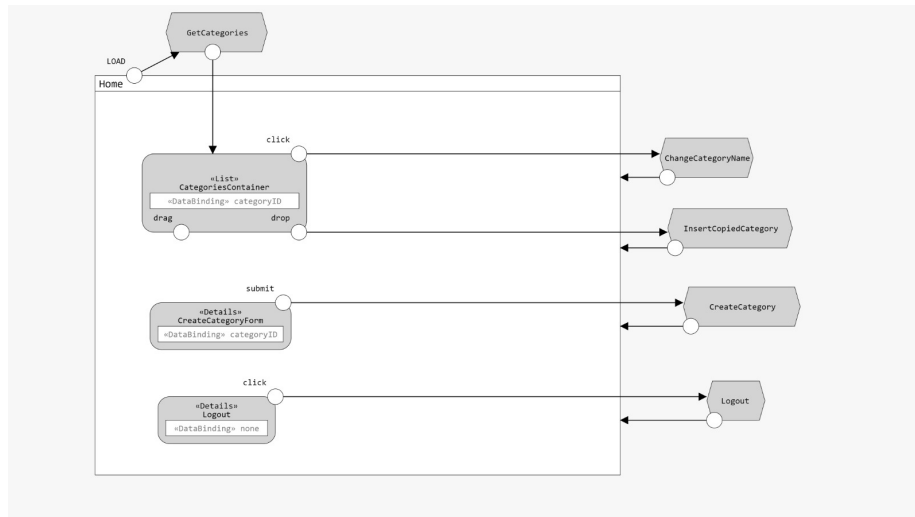


Figure 10: *Diagramma IFML della pagina HOME - versione RIA.*

1.3.3 Eventi ed Azioni

| Client side | | Server side | |
|---|--|-------------------------------|--|
| Evento | Azione | Evento | Azione |
| index → login form → submit | Controllo dati | POST username password | Controllo credenziali |
| Index → registration form → submit | Controllo dati | POST username password | Controllo credenziali |
| Home page → load | Aggiorna view con dati categorie | GET (nessun parametro) | Estrazione dati di tutte le categorie |
| Home page → elenco categorie → click su una categoria | Nuovo elemento per input utente. Controllo validità dati | POST changedID newName | Cambio del valore "name" della categoria con categoryID "changedID" nel database |
| Home page → elenco categorie → drag su una categoria | Calcolo categorie da trascinare, ora con colore rosso. Aggiunta di nuovi event listener su categorie non trascinate | - | - |
| Home page → elenco categorie → drop su una categoria | Calcolo nuovi ID per le categorie. Lista aggiornata mostrata lato server. Form e logout nascosti, nuovo bottone per inviare cambiamenti al database | - | - |
| Confirm button → click | Controllo dati | POST jsonData | Inserimento nuove categorie nel database |
| Create Category form → submit | Controllo dati | POST categoryID name parentID | Inserimento nuova categoria nel database |
| Logout → click | - | GET | Terminazione della sessione |

Figure 11: *Tabella degli Eventi e delle Azioni.*

1.3.4 Controller - *Event Handler*

| Client side | | Server side | |
|---|-------------------------------------|-------------------------------|--------------------------------|
| Evento | Controllore | Evento | Controllore |
| index → login form → submit | Function makeCall | POST username password | CheckLogin (servlet) |
| Index → registration form → submit | Function makeCall | POST username password | RegisterNewUser(servlet) |
| Home page → load | Function PageOrchestrator | GET (nessun parametro) | GetCategories (servlet) |
| Home page → elenco categorie → click su una categoria | Function makeCallReady | POST changedID newName | ChangeCategoryName (servlet) |
| Home page → elenco categorie → drag su una categoria | Function event listener 'dragstart' | - | - |
| Home page → elenco categorie → drop su una categoria | Function showTemporarylist | - | - |
| Confirm button → click | Function makeCallReady | POST jsonData | InsertCopiedCategory (servlet) |
| Create Category form → submit | Function makeCall | POST categoryID name parentID | CreateCategory (servlet) |
| Logout | Function makeCall | GET | Logout (servlet) |

Figure 12: *Tabella del Controller - Event Handler.*

1.3.5 Event & View Dynamics

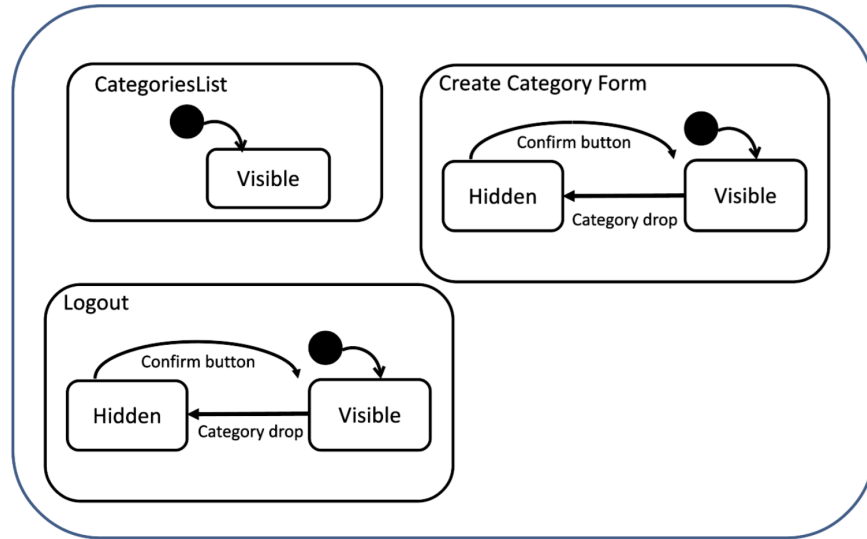


Figure 13: *Event & View Dynamics.*

1.3.5.1 DAO e *Model Objects* - server side

- Controllers (*Servlets*):
 - CheckLogin
 - Logout
 - CreateCategory
 - InsertCopiedCategory
 - GetCategories
 - RegisterNewUser
 - ChangeCategoryName
- Filters:
 - NoCacheFilter
 - UserChecker
- Model Objects (*Beans*):
 - User
 - Category
- Data Access Objects (*Classes*):
 - UserDao
 - * checkLogin(username, password)
 - * registerNewUser(username, password)
 - CategoriesDao
 - * findAllCategories()
 - * createCategory(name, parentID)
 - * findLastChildrenID(categoryList, parentID)
 - * insertCopiedCategory(categoryID, parentID)
 - * addCategoryInDatabase(newID, name, newParentID)
 - * orderCategoriesList(unorderedList, parentID)
 - * changeCategoryName(categoryID, newName)

1.3.6 *View e View Components* - client side

TODO (??)

1.3.7 *Sequence Diagrams*:

TODO