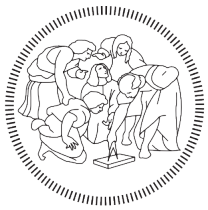


POLITECNICO
MILANO 1863

Progetto TIW

Traccia n. 3

Michele Cavicchioli 10706553

Anno accademico 2022/2023

Contents

1 Traccia e Analisi

1.1 Versione pureHTML

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 codicenumerico 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

Entity	View
Attribute	View component
Relationship	Event
	Action

Table 1: Legenda

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 sotto-albero al primo livello della tassonomia (se non esistono già 9 nodi al primo livello della tassonomia)

1.2 Versione Javascript

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

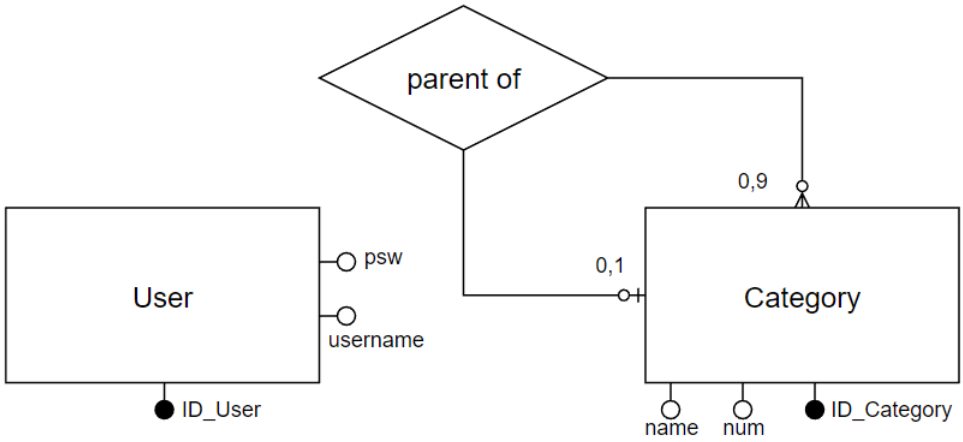
- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- La funzione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

2 Database design

2.1 Diagramma ER

Di seguito si può trovare il diagramma ER del database relativo all'applicazione.
Dal diagramma si nota che una categoria può anche non avere un padre. Questa scelta è motivata

dalla volontà di utilizzare una categoria come *radice* dell'albero che di conseguenza non può avere un genitore.



2.2 Local schema

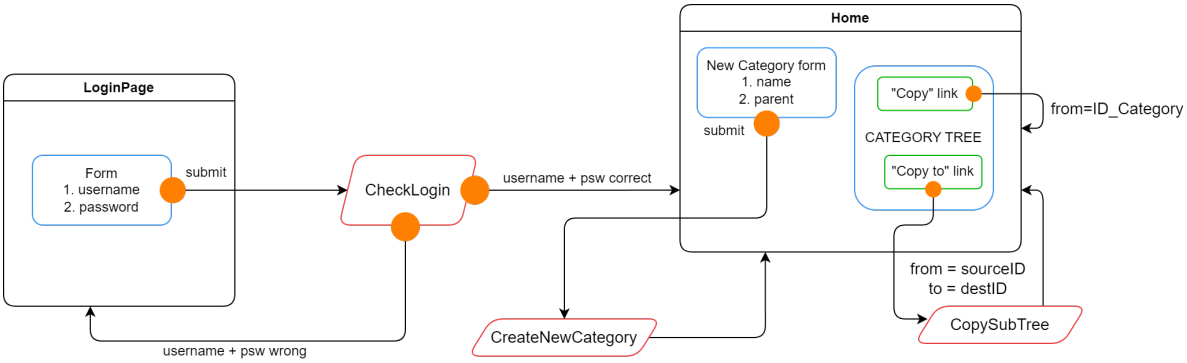
In questa sezione si può trovare il local schema del database. Questo comprende l'inserimento nel database della *root* dell'albero. L'obbligatorietà della presenza di un padre per tutte le categorie non *root* verrà garantita dalla implementazione del server. La scelta di rappresentare il numero identificativo di ciascuna categoria come una stringa è relativo alla dimensione che l'albero può arrivare ad avere; in questo modo dichiarando *num* come una stringa di massimo 20 caratteri, il nostro albero potrà rappresentare al massimo 20 layer di categorie, il che è ragionovele visto lo scopo del database.

```
create database TIW_Project;
use TIW_Project;

create table User(
    ID_User int AUTO_INCREMENT PRIMARY KEY,
    username varchar(50) not null,
    psw varchar(50) not null
);

create table Category(
    ID_Category bigint AUTO_INCREMENT PRIMARY KEY,
    name varchar(100) not null,
    num varchar(20) not null,
    parent bigint,
    foreign key (parent) references Category(ID_Category)
        on delete cascade on update cascade
);


INSERT INTO Category(name,num,parent) VALUES ("root","0",NULL);
INSERT INTO User(username,psw) VALUES ("Michele","12345");
```

3 Progetto pureHTML

3.1 Application design

Di seguito si trova il diagramma IFML dell'intero progetto.



3.2 Componenti

3.2.1 Controllers - Servlets

- LoginPage
- GoToHome
- GoTo_CopyToHome
- CreateNewCategory
- CopySubTree

3.2.2 Beans

- User
- Category

3.2.3 Views, Components

- LoginPage - *" "*
  - Sign-in form
- Home - *"/GoToHome"*
  - Categories tree
    - *copy/copy_to* links
  - New category form

3.2.4 DAOs - Data Access Objects

- UserDAO

  + checkCredentials

- CategoryDAO

  + getTopCategories()
  + getAllCategories()
  + getCategoryFromID(long ID)
  + getCategoryFromNum(String num)
  + createCategory(String name, long parent)
  + copySubTree(long sourceID, long destID)
  + isCopyPossible(long from, long to)
  - insertNewSubTree(Category curr, Category parent)
  - getParentIDs(Category root)
  - getDirectChildrenOf(long ID)
  - insertNewSubTree(Category curr, Category parent)

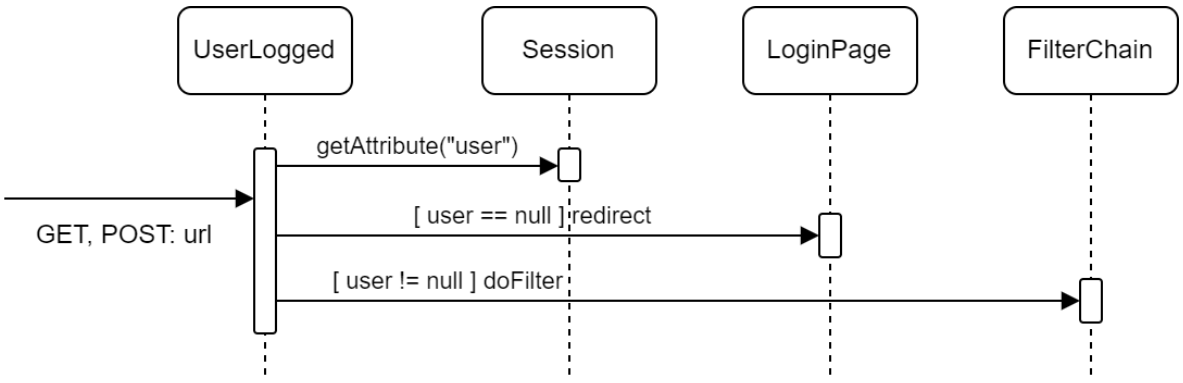
3.2.5 Filters

- UserLogged: filter used to check if there is a username saved in the current session →if the user is logged

3.3 Sequence diagram

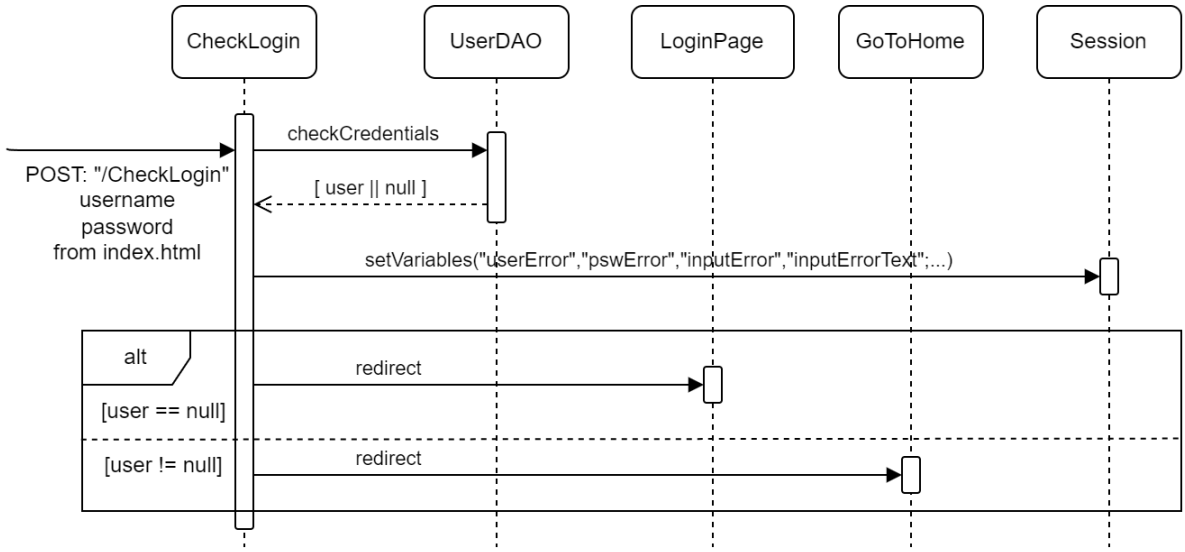
3.3.1 UserLogged

Di seguito si descrive il processo che il filtro *UserLogged* esegue prima di ogni HTTP request che vada a richiedere risorse *protette*.



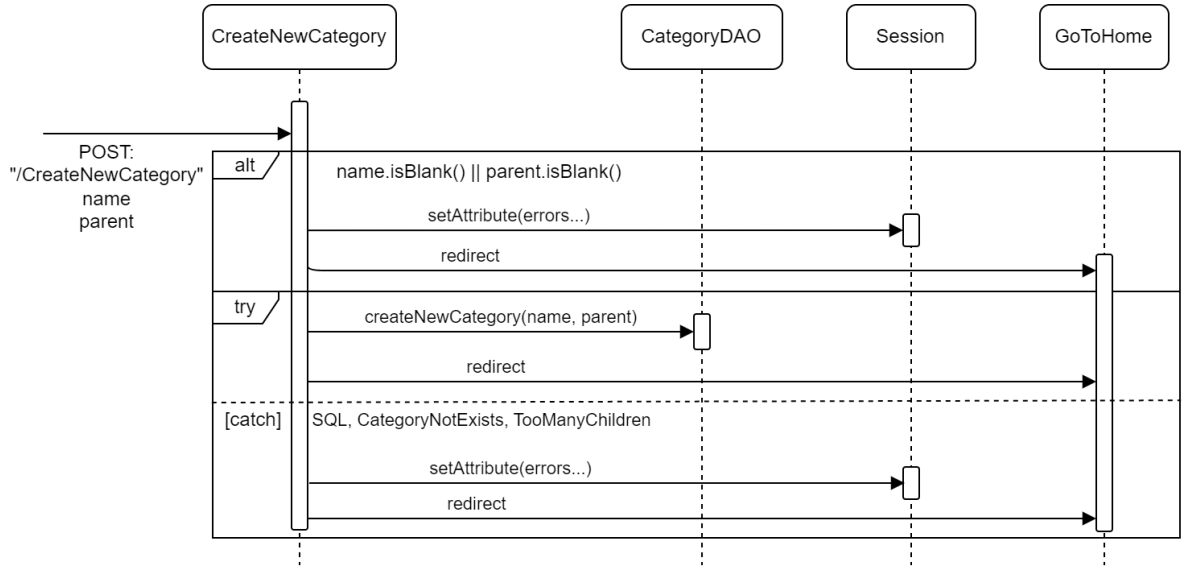
3.3.2 CheckLogin

Sotto si può trovare il sequence diagram che descrive il processo di login di un utente.



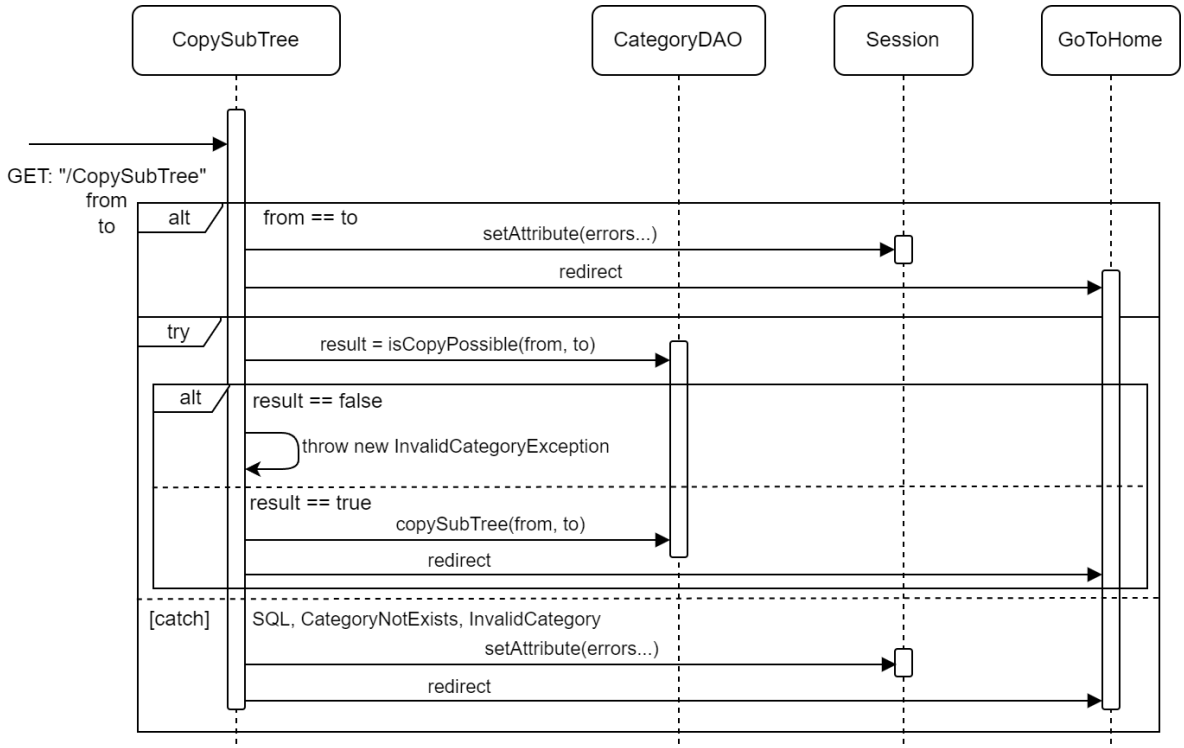
3.3.3 CreateNewCategory

Di seguito si può vedere il sequence diagram relativo al processo di creazione di una singola categoria.



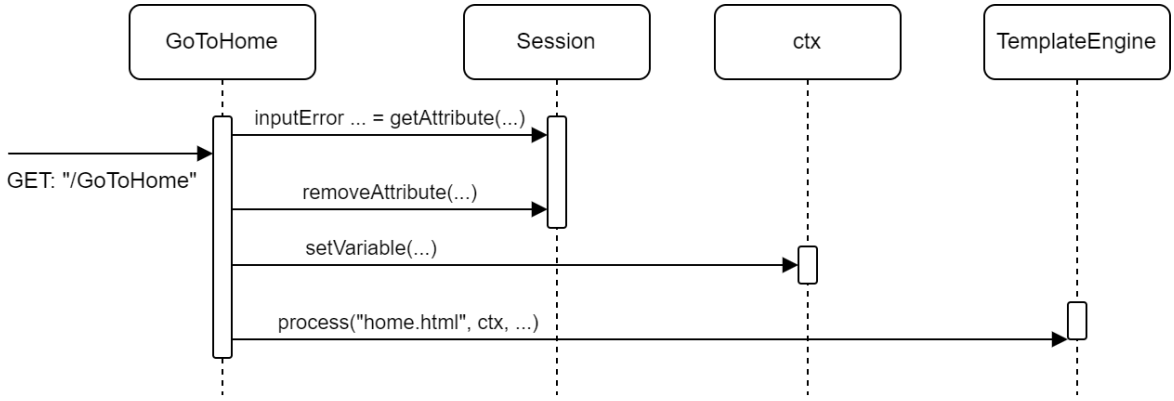
3.3.4 CopySubTree

Il prossimo sequence diagram descrive il processo di copiatura di un sottoalbero in un altro.



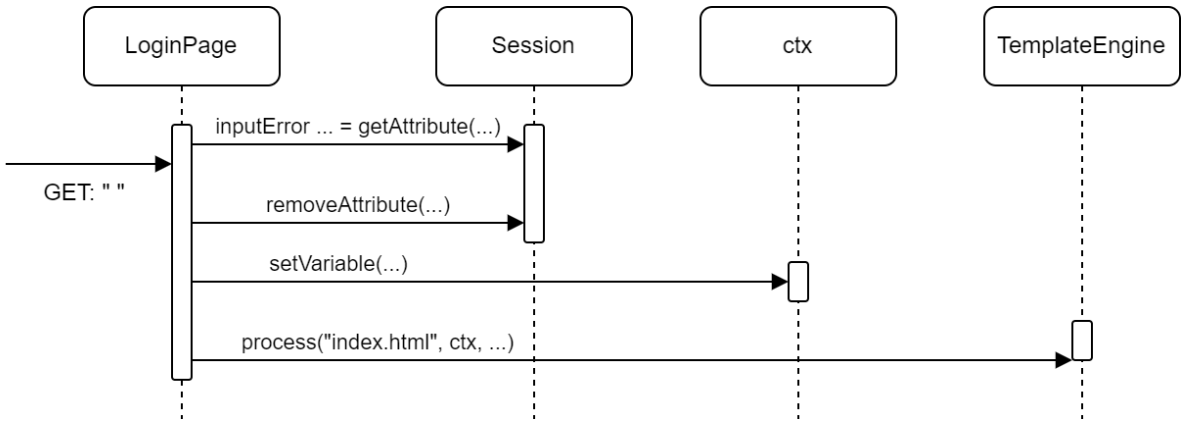
3.3.5 GoToHome

Il prossimo sequence diagram descrive il processo con cui il server restituisce la home page se richiesta.



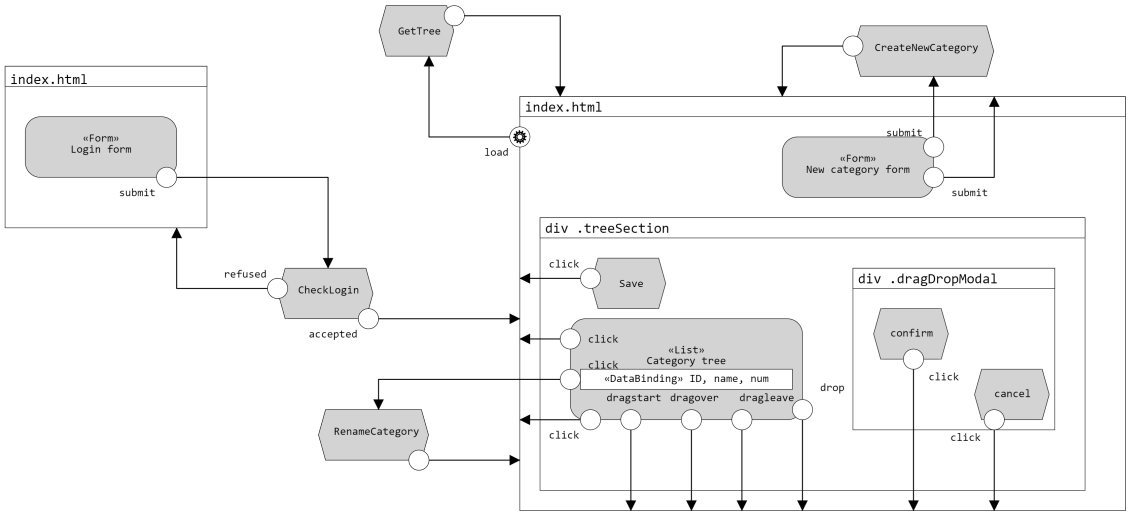
3.3.6 GoToLoginPage

Il prossimo sequence diagram descrive il processo con cui il server restituisce la login page se richiesta. Questo url è l'unico nel server a non essere coperto dal filtro *UserLogged* perchè deve essere aperto al *pubblico*.



4 Progetto javascriptVersion

4.1 Application design



4.2 Componenti

4.2.1 Controllers - Servlets

- CheckLogin
- GetTree
- ApplyChanges
- CreateNewCategory
- RenameCategory

4.2.2 Beans

- User
- Category
- ClientCategory: classe usata per rappresentare una categoria creata dinamicamente dal client.
- DataToCheck: classe con main goal quello di contenere tutti i dati necessari per effettuare i controlli di sicurezza prima di approvare una richieste di modifica alla base di dati.

4.2.3 Views, Components

- LoginPage - index.html
  - Sign-in form
- Home - home.html
  - Categories tree
    - save button
  - New category form
  - drag & drop Modal
    - confirm button
    - cancel button

4.2.4 DAOs - Data Access Objects

- UserDAO

  + checkCredentials

- CategoryDAO

  + getCategoryFromID(long ID)
  + getCategoryFromNum(String num)
  + createCategory(String name, long parent)
  + copySubTree(Category source, Category destination)
  + isCopyPossible(long from, long to)
  + renameCategoryById(long ID, String newName)
  + areTreeEqual(ArrayList<Category>userTree)
  + areOptionsOk(ArrayList<Category>option)
  - insertNewSubTree(Category curr, Category parent)
  - getDirectChildrenOf(long ID)
  - categoryEquals(Category x, Category y)
  - createListId(ArrayList<Category> categories, ArrayList<Long> ids, ArrayList <String> nums)
  - createListIdFromOptions(ArrayList<Category> options, ArrayList<Long> ids, ArrayList<String> nums)
  - checkSubTree(ClientCategory curr, Set<String> nums)
  - getParentNums(ClientCategory root)

4.2.5 Filters

- UserLogged: filter used to check if there is a username saved in the current session →if the user is logged

4.2.6 Javascript files

- loginManagement.js
- homeManagement.js
- utils.js

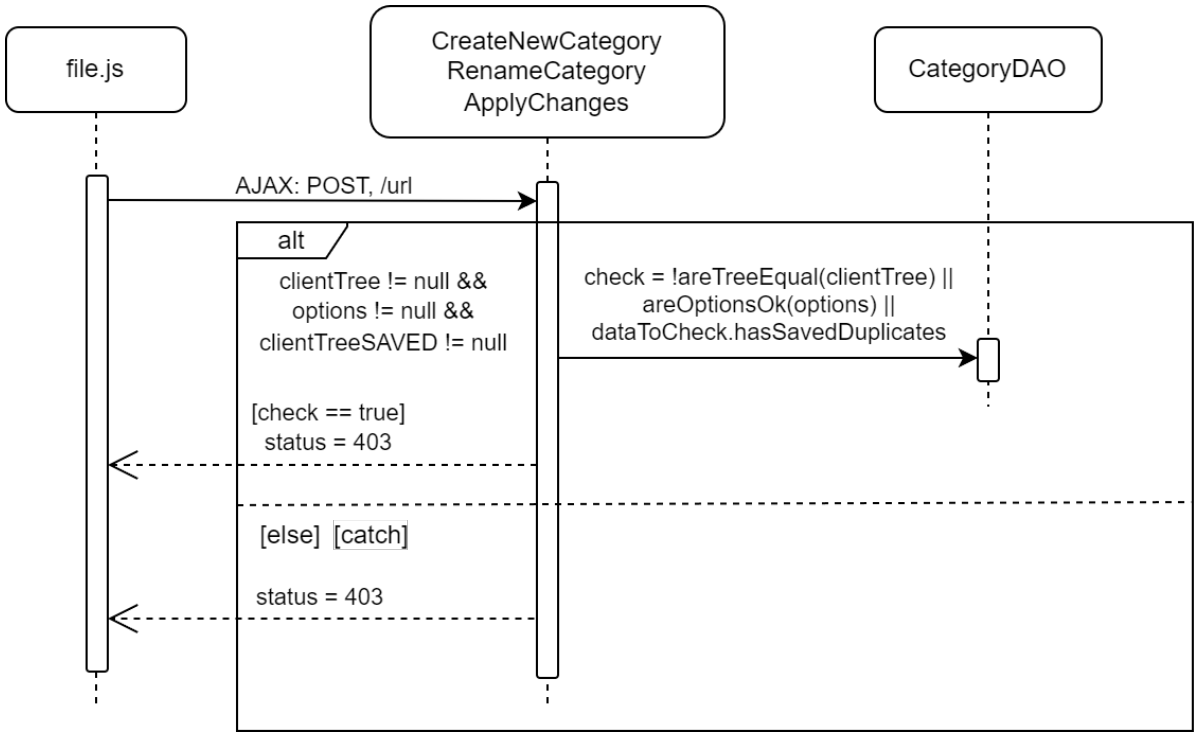
4.2.7 Eventi

Client side		Server side	
Event	Handle	Event	Controller
Login page - index.html			
index.html → login form → submit	makeCall(. . .)	POST(username, psw)	CheckLogin
Home page - home.html			
PageManager			
window → load	checkUserLogged		
window → click	resetError	-	-
CategoryTree			
saveButton → click	handleSaveButtonClick	newCategories + renamedCategories	ApplyChanges
.treeElementContent → dragstart	dragStart	-	-
.treeElementContent → dragleave	dragLeave	-	-
.treeElementContent → dragend	dragEnd	-	-
.treeElementContent → dragover	dragOver	-	-
.treeElementContent → drop	drop	-	-
.treeElement >*name* → click	renameCategoryLocally	-	-
.treeElement >*name* → blur	renameCategoryLocally > addToRenamed	-	-
.treeElement >*name* → click	renameCategoryOnline	-	-
.treeElement >*name* → blur	renameCategoryOnline >sendServer	POST(renamedCategory)	RenameCategory
.dragDropModal >confirm → click	confirmCallback	-	-
.dragDropModal >cancel→ click	cancelCallback	-	-
NewCategoryForm			
#newCategoryForm → submit	insertNewCategoryOnline	POST(newCategory)	CreateNewCategory
#newCategoryForm → submit	insertNewCategoryLocal	-	-

4.3 Sequence diagram

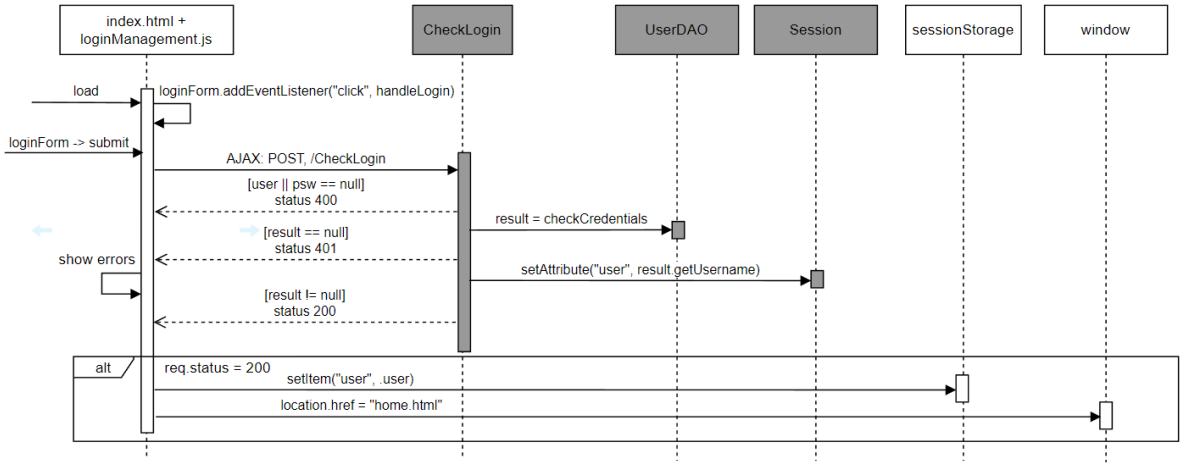
4.3.1 DataToCheck

Il sequence diagram qui sotto rappresenta una generalizzazione di quello che avviene all'interno di una servlet che si occupa di modificare la base di dati (*CreateNewCategory, RenameCategory, ApplyChanges*). Quello che si fa sono dei semplici controlli per verificare che il client non abbia manipolato l'albero delle categorie, la lista dei genitori nel form e che non abbia alterato il javascript in un modo che possa causare danni alla base di dati. Si noti che i alcune servlet possono richiedere più dati (*ApplyChanges* richiede anche il *clientTreeSAVED* mentre le altre due servlet no).



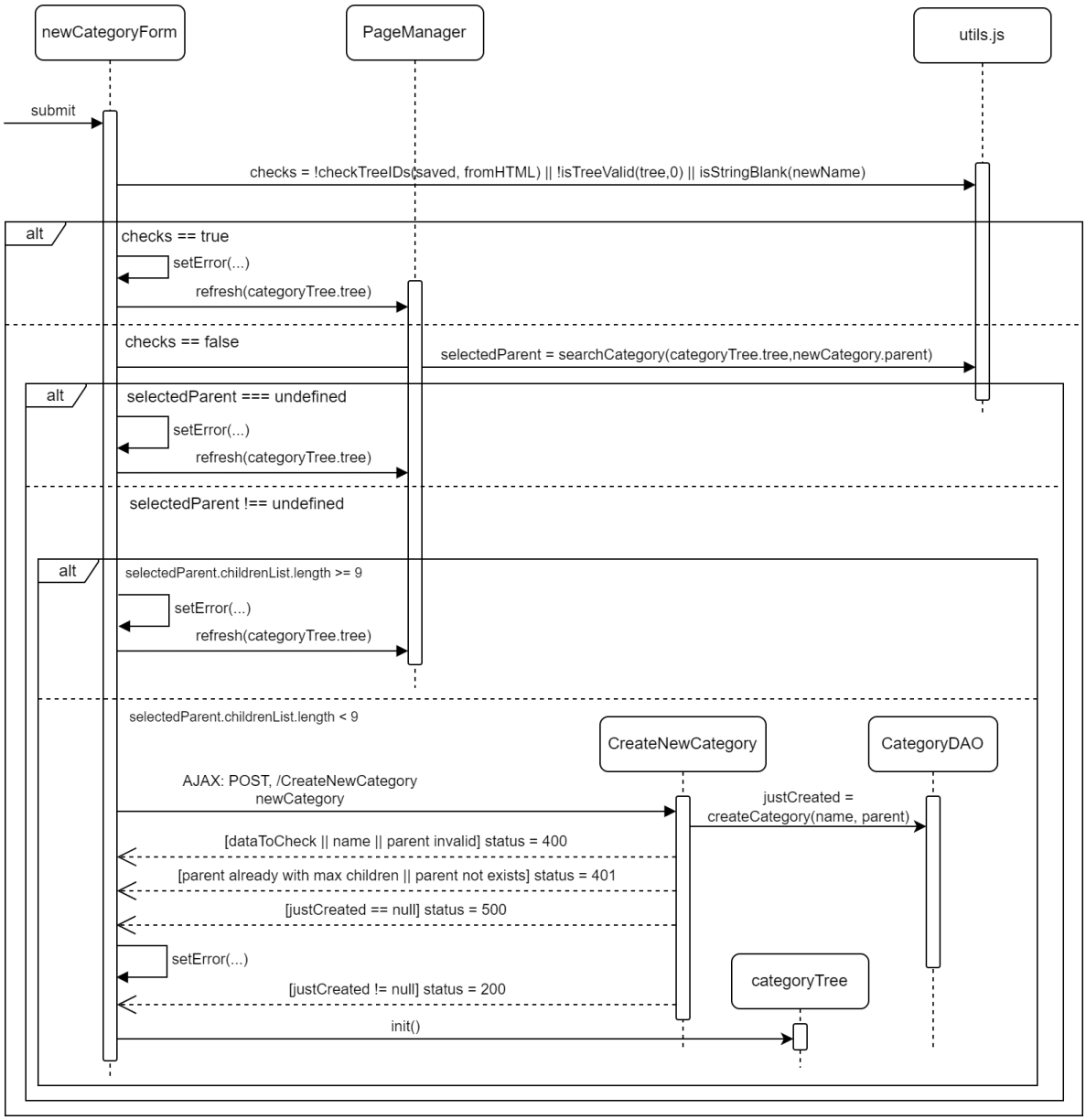
4.3.2 Login form → load, submit

Il seguente sequence diagram rappresenta il comportamento dell'applicazione quando viene caricata la pagina *index.html* e/o un utente usa il form di login per richiedere l'accesso.



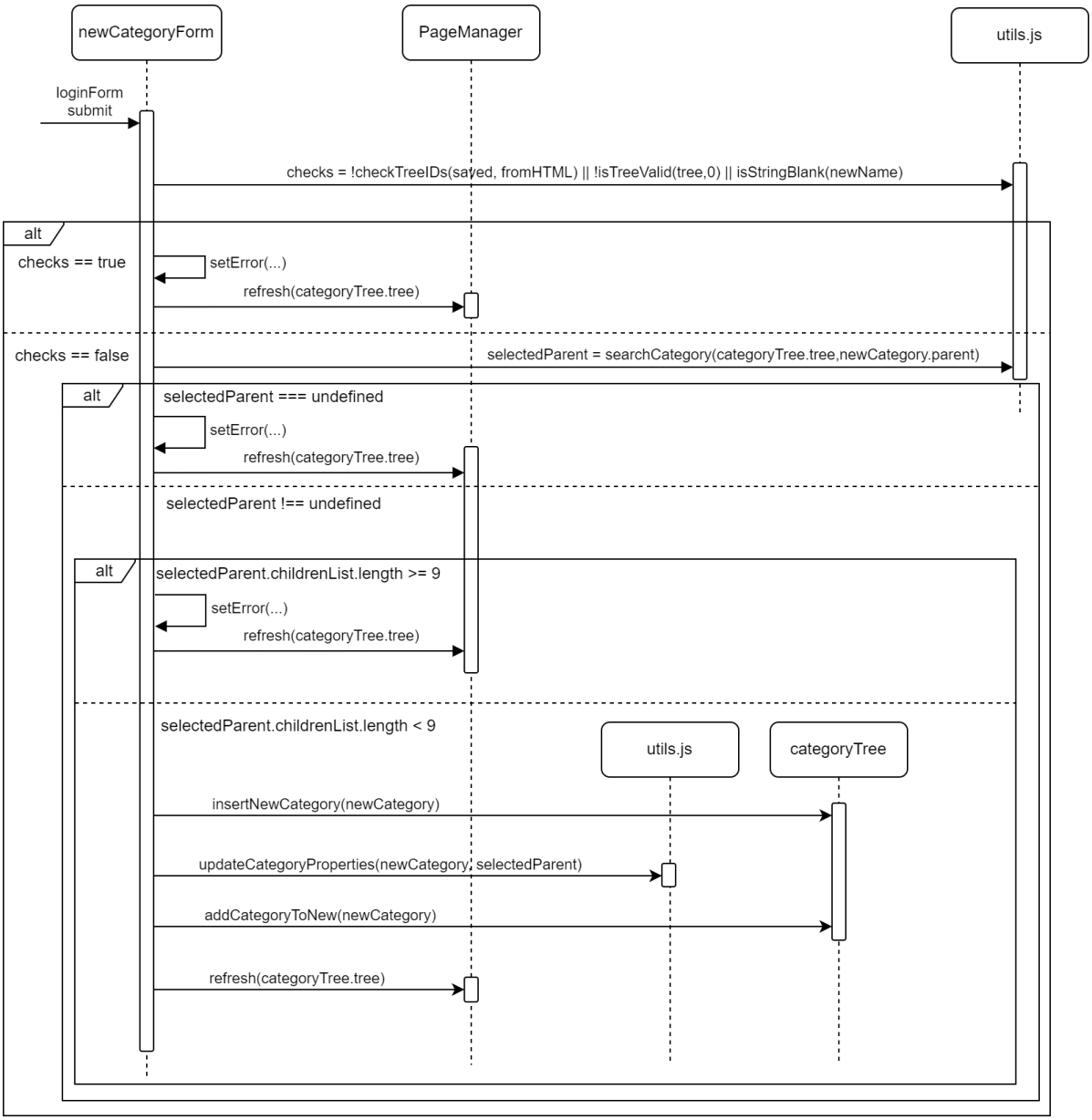
4.3.3 #newCategoryForm → submit (ONLINE)

Rappresentazione tramite sequence diagram del processo di gestione della richiesta di inserimento di una nuova categoria direttamente sul server.



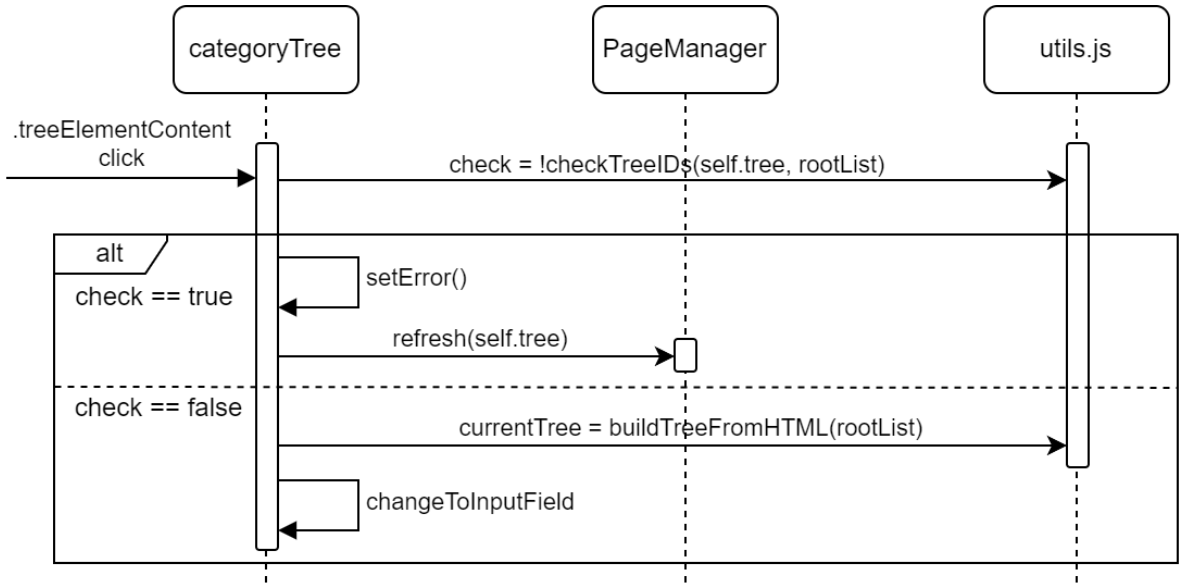
4.3.4 #newCategoryForm → submit (LOCALE)

Rappresentazione tramite sequence diagram del processo di gestione della richiesta di inserimento di una nuova categoria nell'albero locale.



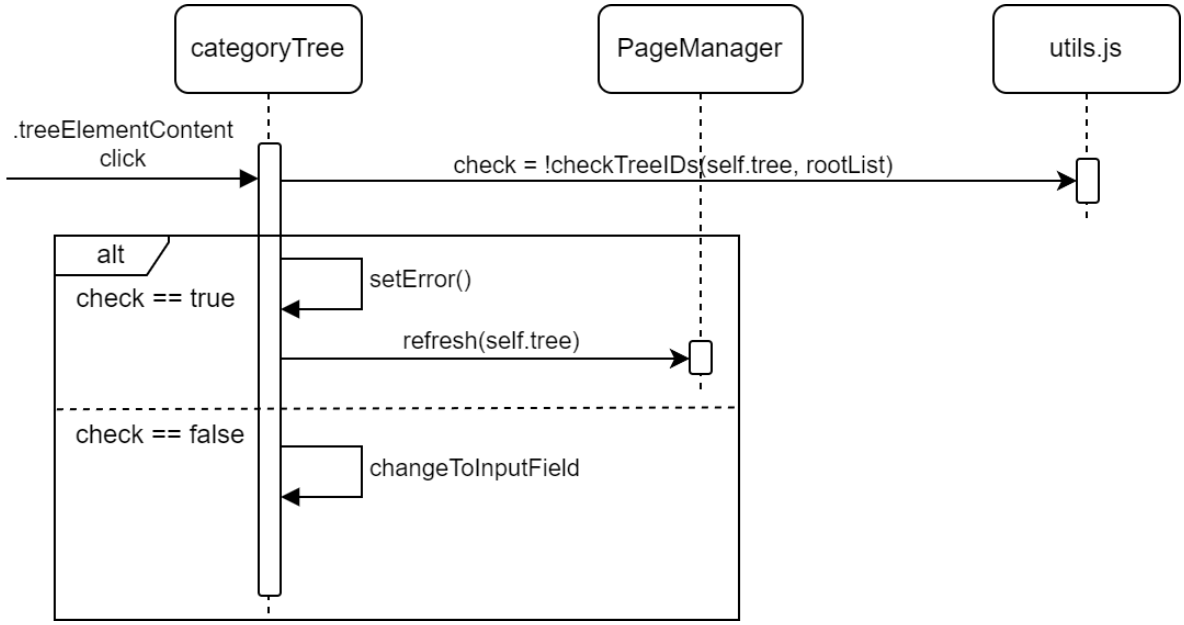
4.3.5 .treeElement → click (ONLINE)

Gestione del click di un elemento dell'albero delle categorie per rinominare l'elemento direttamente sul server.



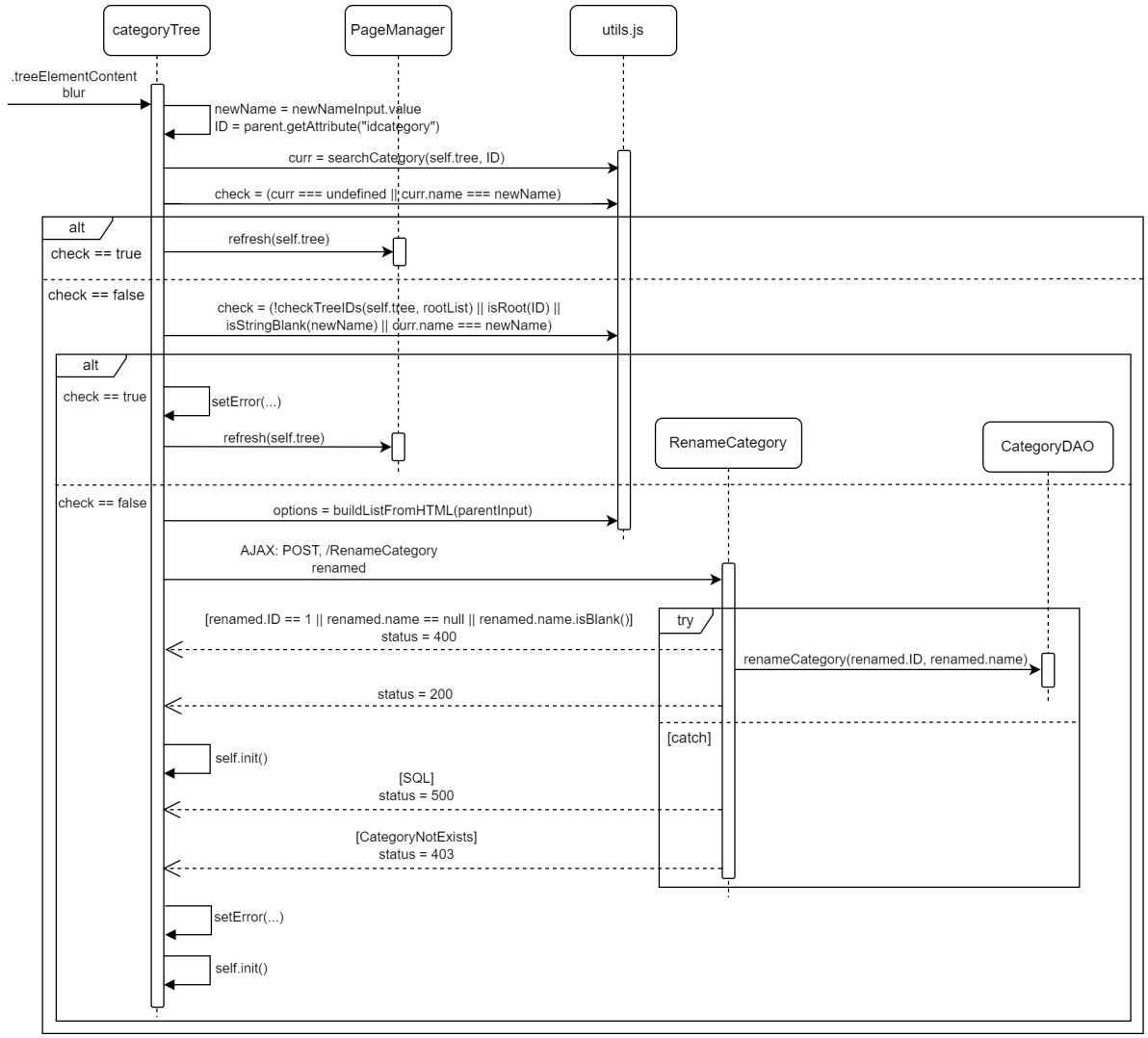
4.3.6 .treeElement → click (LOCALE)

Gestione del click di un elemento dell'albero delle categorie per rinominare l'elemento sull'albero locale.



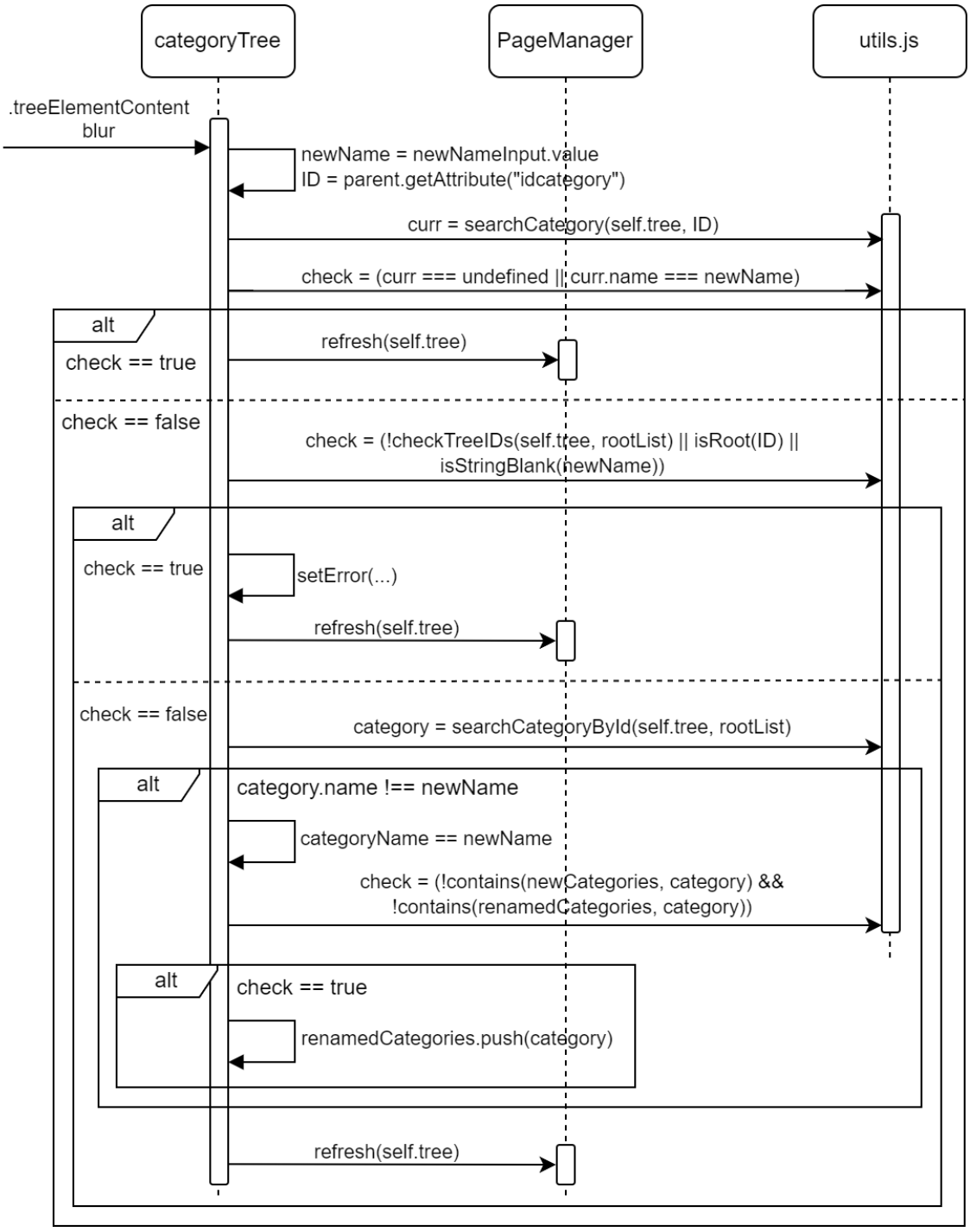
4.3.7 .treeElement → blur (ONLINE)

Gestione dell'evento di perdita del focus dall'input field usato dall'utente per rinominare l'elemento (rinomina istantanea sul server).

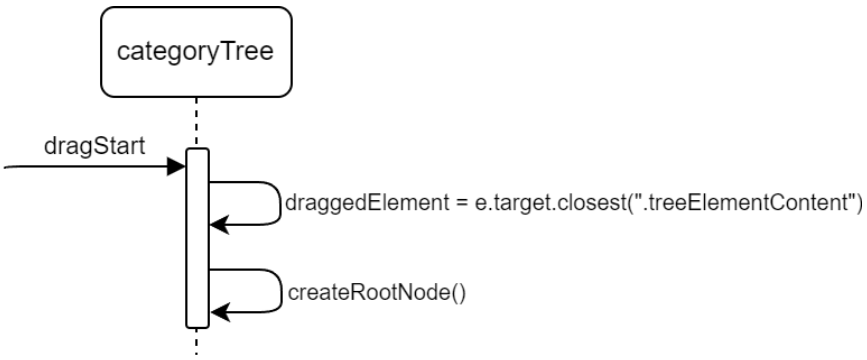


4.3.8 .treeElement → blur (LOCALE)

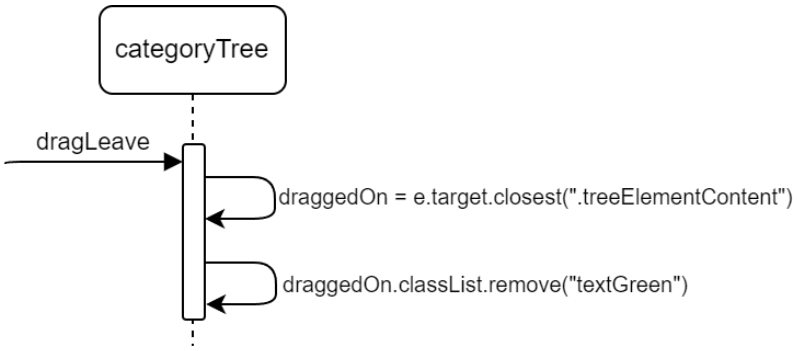
Gestione dell'evento di perdita del focus dall'input field usato dall'utente per rinominare l'elemento (rinomina solo sull'albero locale).



4.3.9 .treeElementContent → dragStart

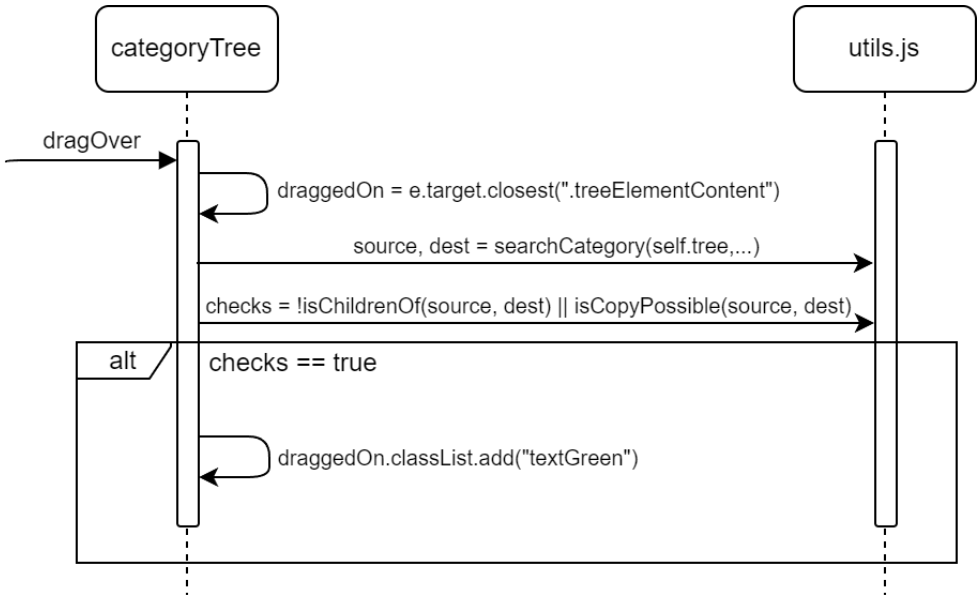


4.3.10 .treeElementContent → dragLeave



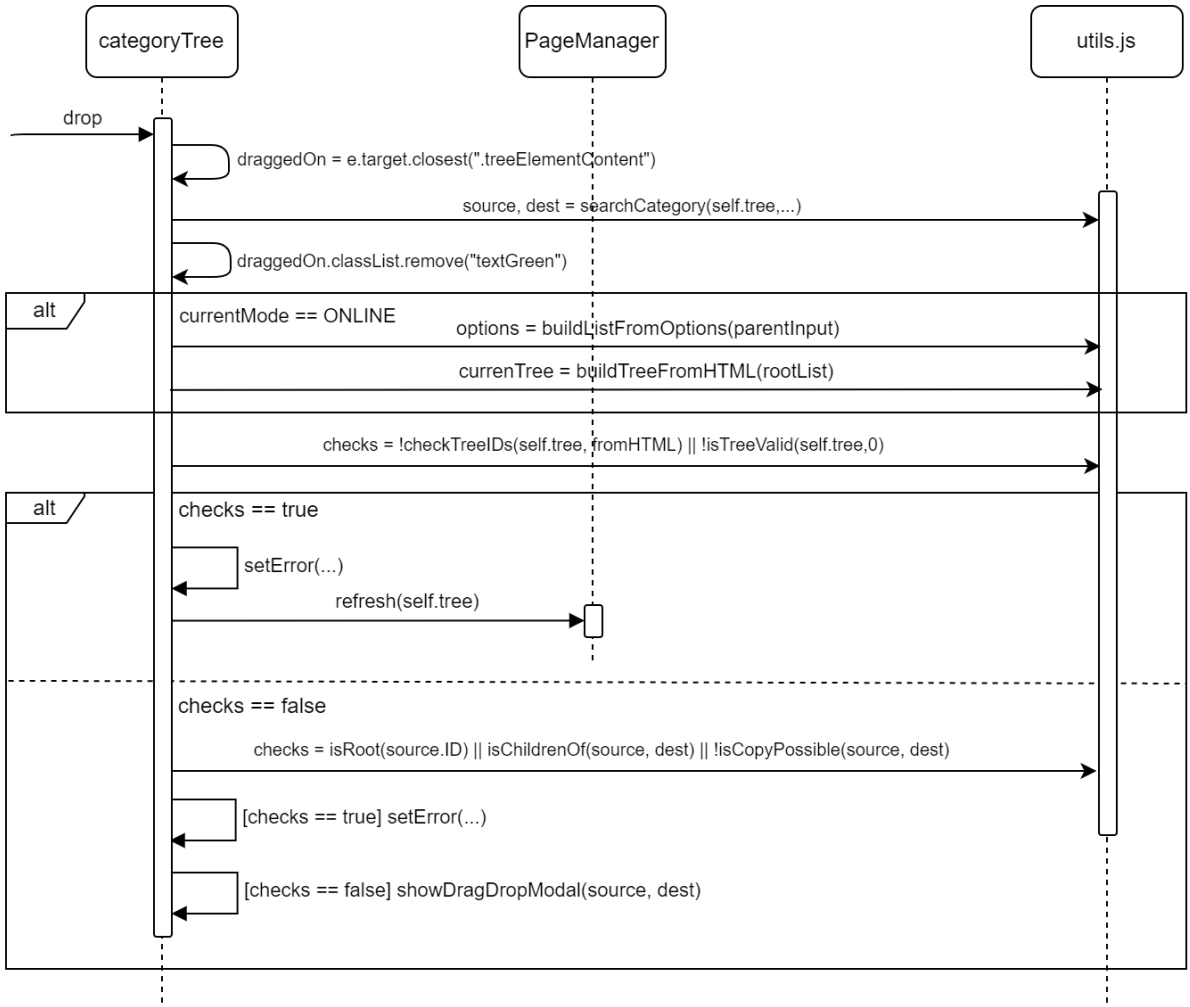
4.3.11 .treeElementContent → dragOver

Gestione dell'evento *dragover* generato quando un elemento *draggable* entra in uno spazio usato da un altro *HTMLElement*.



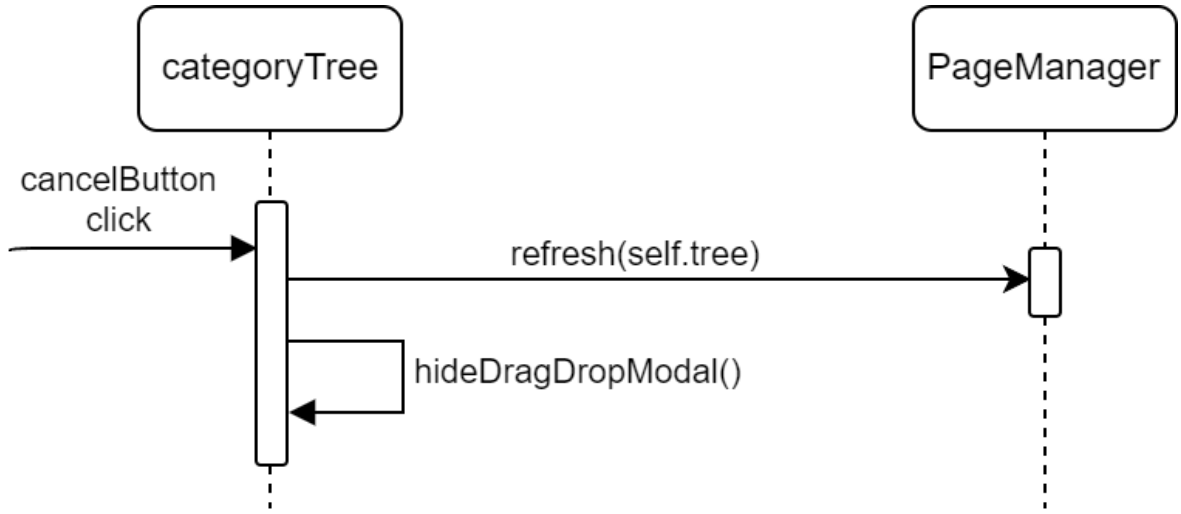
4.3.12 .treeElementContent → drop

Gestione del *drop* di un elemento su un altro.



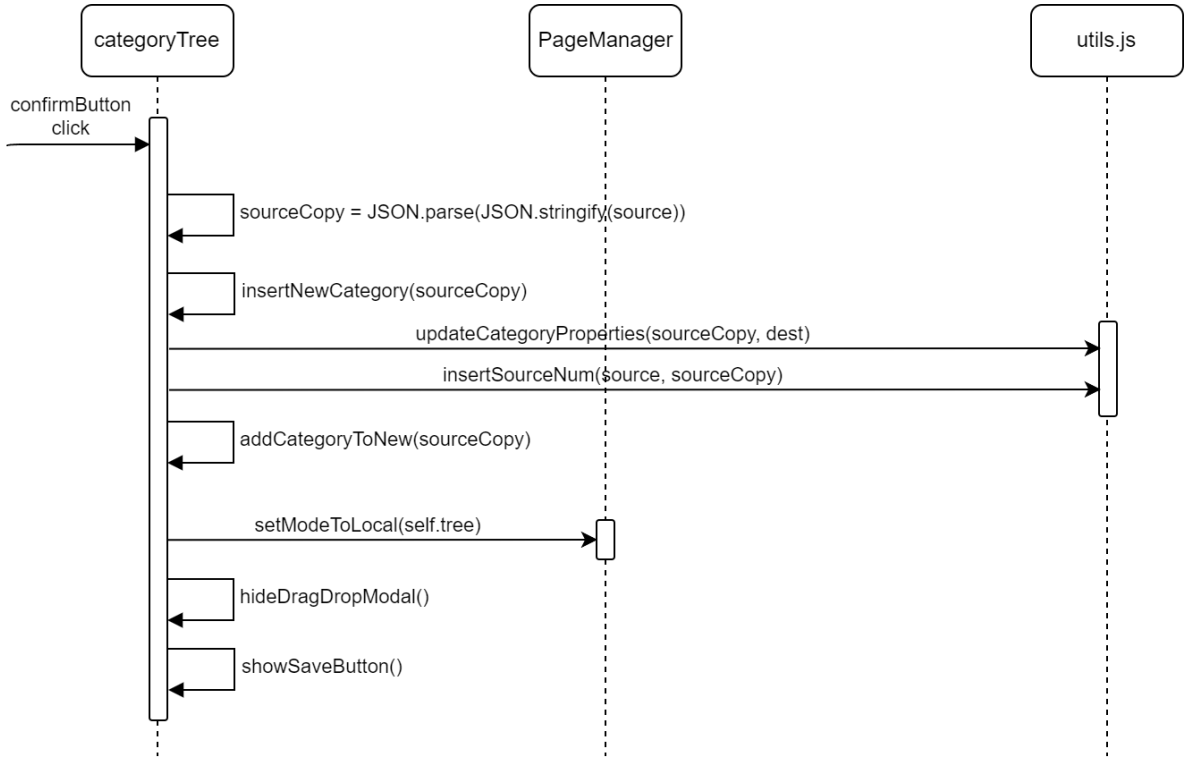
4.3.13 .cancel → click

Come viene gestito il *click* del bottone per la cancellazione del precendete drag & drop.



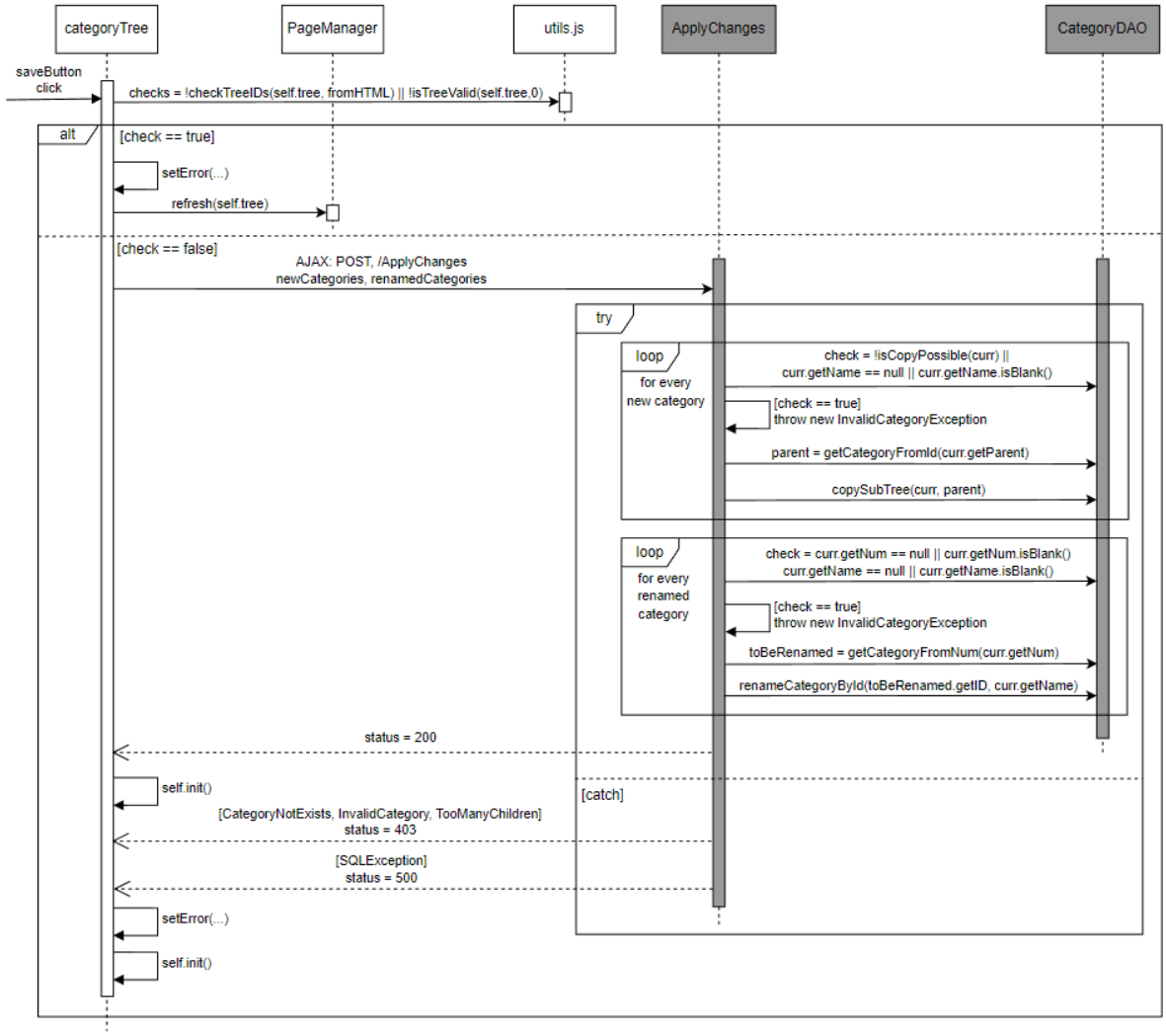
4.3.14 .confirm → click

Processo eseguito al *click* del bottone di conferma del drag & drop appena eseguito. Se è stato il primo drag & drop allora passiamo dalla modalità *ONLINE* a quella *LOCALE* (si può modificare l'albero come si vuole; per salvare le modifiche cliccare il bottone di salvataggio).



4.3.15 .saveButton → click

Il seguente sequence diagram rappresenta il processo di salvataggio delle modifiche eseguite dall'utente.



4.4 Note

- Per la gestione dei dati per lo scambio di informazioni tra client e server ho deciso di usare i json. Per poterli usare nelle servlet ho fatto uso della libreria *gson* e ho creato una classe privata per ogni servlet che abbia bisogno di ricevere dati dal client, in modo tale da poter trasformare il json ad un oggetto Java.

- Dopo un primo drag & drop la traccia non specificava quali potessero essere le possibili azioni di un utente, se potesse effettuare altri drag & drop in serie, oppure rinominare una categoria o usare il form per crearne una singola. La mia interpretazione è stata quella di non bloccare nessuna azione all'utente; infatti dopo che viene eseguita una prima copia tramite drag & drop l'applicazione entra in una modalità *LOCALE* dove tutte le modifiche eseguite sull'albero avvengono in locale, l'albero verrà aggiornato anche sul server solo al momento del saltaggio. Prima di un drag & drop il comportamente dell'applicazione non cambia, rinomina e creazione di una categoria vengono eseguite sia sul server che sul client allo stesso momento.