# ES3: catalogazione di immagini

Versione RIA

Palazzoli Matteo

# Catalogazione di immagini

Un’applicazione permette all’utente (ad esempio il curatore di un catalogo online di immagini) di gestire una tassonomia di classificazione utile per etichettare immagini allo scopo di consentire la ricerca in base alla 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. Le categorie hanno nomi distinti. 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. Per semplicità si ipotizzi che per ogni categoria il numero massimo di sottocategorie sia 9, numerate da 1 a 9. Dopo la creazione di una categoria, la pagina HOME mostra l’albero aggiornato. L’utente può spostare di posizione una categoria: per fare ciò clicca sul link “sposta” associato alla categoria da spostare. A seguito di tale azione l’applicazione mostra, sempre nella HOME page, l’albero con evidenziato il sotto albero attestato sulla categoria da spostare: tutte le altre categorie hanno un link “sposta qui”. La selezione di un link “sposta qui” comporta l’inserimento della categoria da spostare come ultimo figlio della categoria destinazione. Le modifiche effettuate da un utente e salvate nella base di dati diventano visibili agli altri utenti. Lo spostamento potrebbe creare un vuoto nella numerazione delle categorie figlie dello stesso padre. La funzione di ricalcolo della numerazione delle categorie figlie dello stesso padre di quella spostata non è richiesta ma è lasciata come opzionale.

# Catalogazione di immagini: RIA

- 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 spostamento di una categoria è realizzata mediante drag & drop.
- A seguito del drop della categoria da spostare compare una finestra di dialogo con cui l'utente può confermare o cancellare lo spostamento. La conferma produce l'aggiornamento a lato client dell'albero.
- L'utente realizza spostamenti anche multipli a lato client. A seguito del primo spostamento compare un bottone SALVA la cui pressione provoca l'invio al server dell'elenco degli spostamenti realizzati (NON dell'intero albero). L'invio degli spostamenti produce l'aggiornamento dell'albero nella base dei dati e la comparsa di un messaggio di conferma dell'avvenuto salvataggio.

# Data analysis

Un'applicazione permette all'**utente** (ad esempio il curatore di un catalogo online di immagini) di gestire una tassonomia di classificazione utile per etichettare immagini allo scopo di consentire la ricerca in base alla **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. Le categorie hanno **nomi distinti**. 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. Per semplicità si ipotizzi che per ogni categoria il numero massimo di sottocategorie sia 9, numerate da 1 a 9. Dopo la creazione di una categoria, la pagina HOME mostra l'albero aggiornato. L'utente può spostare di posizione una categoria: per fare ciò clicca sul link "sposta" associato alla categoria da spostare. A seguito di tale azione l'applicazione mostra, sempre nella HOME page, l'albero con evidenziato il sotto albero attestato sulla categoria da spostare: tutte le altre categorie hanno un link "sposta qui". La selezione di un link "sposta qui" comporta l'inserimento della categoria da spostare come ultimo figlio della categoria destinazione. Le modifiche effettuate da un utente e salvate nella base di dati diventano visibili agli altri utenti. Lo spostamento potrebbe creare un vuoto nella numerazione delle categorie figlie dello stesso padre. La funzione di ricalcolo della numerazione delle categorie figlie dello stesso padre di quella spostata non è richiesta ma è lasciata come opzionale.

## **Entities**, **attributes**, **relationships**

# Database design & schema

**username**
password
name
surname

user

```
CREATE TABLE user (
Username VARCHAR(30) PRIMARY KEY,
Password VARCHAR(30) NOT NULL,
Name VARCHAR(30) NOT NULL,
Password VARCHAR(30) NOT NULL
);
```

**Name**
ID

category

```
CREATE TABLE category (
Name VARCHAR(30) PRIMARY KEY,
ID VARCHAR(30) NOT NULL
);
```

# Application requirements analysis (pure html version)

Un'applicazione permette all'utente (ad esempio il curatore di un catalogo online di immagini) di gestire una tassonomia di classificazione utile per etichettare immagini allo scopo di consentire la ricerca in base alla 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. Le categorie hanno nomi distinti. 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. Per semplicità si ipotizzi che per ogni categoria il numero massimo di sottocategorie sia 9, numerate da 1 a 9. Dopo la creazione di una categoria, la pagina HOME mostra l'albero aggiornato. L'utente può spostare di posizione una categoria: per fare ciò **clicca sul link "sposta"** associato alla categoria da spostare. A seguito di tale azione l'applicazione mostra, sempre nella HOME page, l'albero con evidenziato il sotto albero attestato sulla categoria da spostare: tutte le altre categorie hanno **un link "sposta qui"**. La **selezione di un link "sposta qui"** comporta l'**inserimento della categoria da spostare** come ultimo figlio della categoria destinazione. Le modifiche effettuate da un utente e salvate nella base di dati diventano visibili agli altri utenti. Lo spostamento potrebbe creare un vuoto nella numerazione delle categorie figlie dello stesso padre. La funzione di ricalcolo della numerazione delle categorie figlie dello stesso padre di quella spostata non è richiesta ma è lasciata come opzionale.

**Pages (views)**, **view components**, **events**, **actions**

# Application requirements analysis (RIA version)

- 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 spostamento di una categoria è realizzata mediante **drag** & **drop**.
- A seguito del drop della categoria da spostare compare una finestra di dialogo con cui l'utente può **confermare** o **cancellare** lo spostamento. La conferma produce l'aggiornamento a lato client dell'albero.
- L'utente realizza spostamenti anche multipli a lato client. A seguito del primo spostamento compare un bottone SALVA la cui **pressione** provoca l'**invio al server dell'elenco** degli spostamenti realizzati (NON dell'intero albero). L'invio degli spostamenti produce **l'aggiornamento dell'albero** nella base dei dati e la comparsa di un **messaggio di conferma** dell'avvenuto salvataggio.

# Completion

- La **pagina di default** contiene la **form di login.**
- I dati di creazione di una categoria (nome e padre) sono obbligatori.
- Anche gli ID delle categorie sono unici.
- Non è possibile creare una categoria con un padre non presente.
- Non è possibile spostare una categoria sotto un proprio figlio.
- È possibile creare una nuova categoria principale (sotto il nodo “Root”).
- È possibile spostare una categoria come principale (sotto il nodo “Root”).

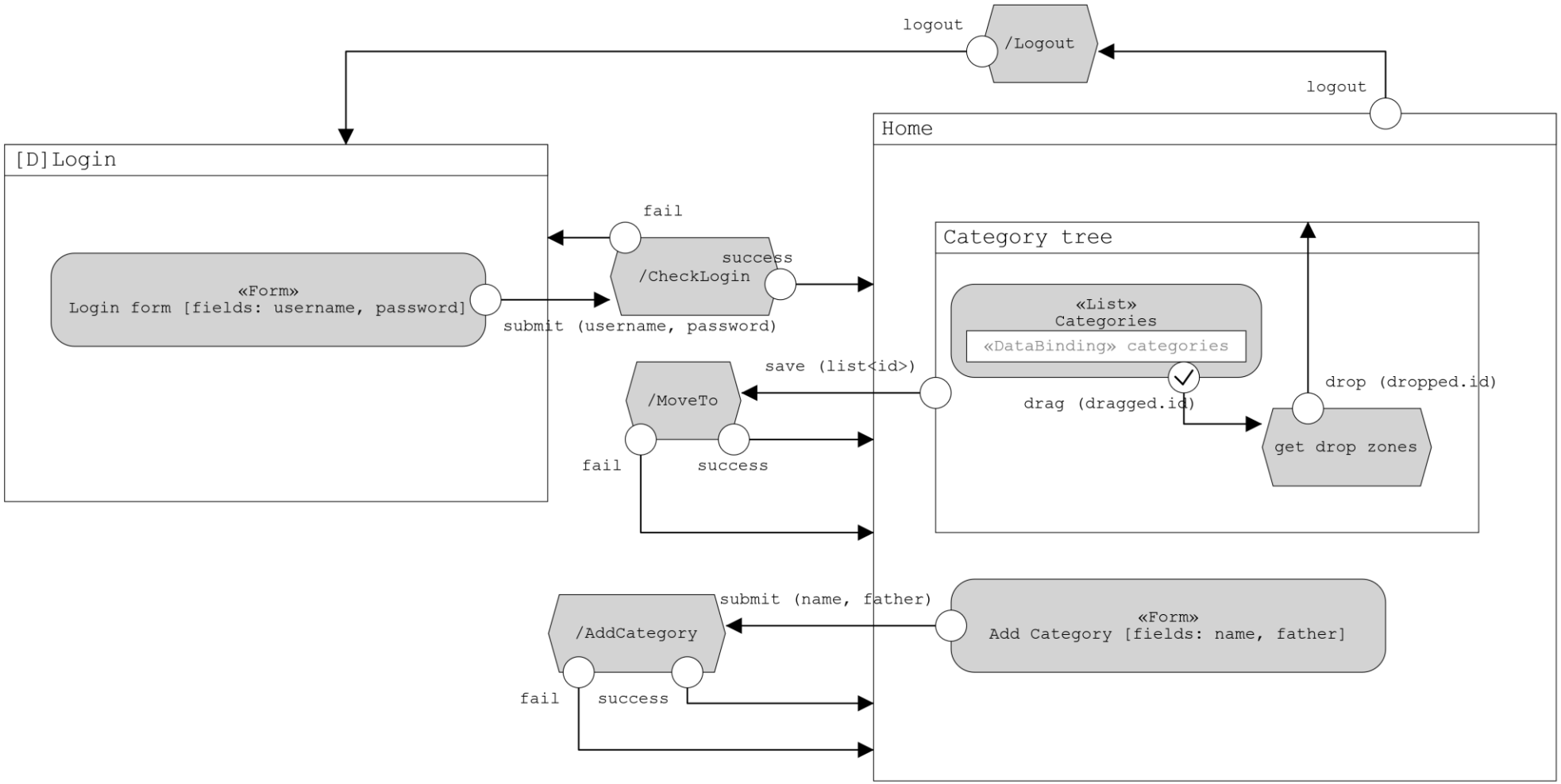
# Further additions

- A seguito del login, riconoscere l'utente e mostrare un messaggio di saluto con il suo nome;
- Anche a seguito di un'azione non valida mostrare un messaggio con l'esito;
- Permettere il logout;

# Non-functional requirements

- La validità dell'input deve essere controllata a lato client, oltre che a lato server
- L'autenticazione e autorizzazione devono avvenire in modo sicuro
  - Verifica delle credenziali nel database
  - Mantenimento dello stato di autorizzazione mediante la sessione delle servlet.

# Application design



# Server side components

- Model objects (Beans)
  - User
  - Category
- Data Access Objects (Classes)
  - UserDAO
    - checkCredentials(username, password)
  - CategoryDAO
    - getTree()
    - createCategory(name, father)
    - updateCategory(fromId, toId)

- Controllers (servlets)
  - CheckLogin
  - GoToHome
  - AddCategory
  - MoveTo
  - Logout
- The database connection is created by controllers in the `init()` method and passed to the DAO
- The session controls are made by a WebFilter

# Client side components

- Login
  - Login form (manages submit and errors)
- Home
  - Category Tree: manages the tree of categories and the drag-and-drop actions, with submit button.
  - Add Category Form: manages submit and errors.

Scripts:

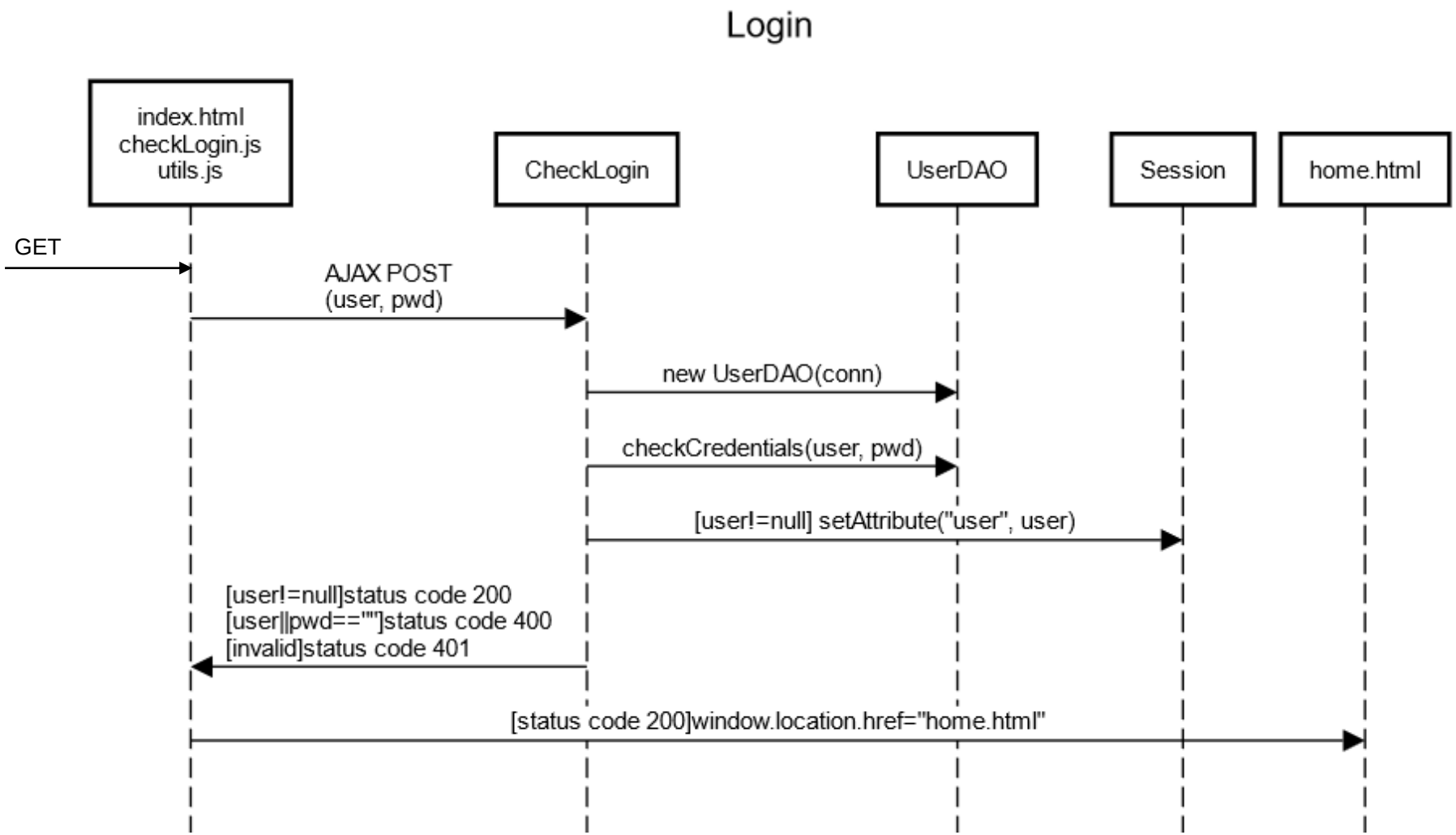
- mainController.js: initializes the page, adds the eventListeners
- addCat.js: handles the category addition
- categoryManagement.js: handles the elaboration of the tree (preview after a move action)
- getTree.js: retrieves the tree from the server
- Utils.js: AJAX makeCall function

# Events & actions

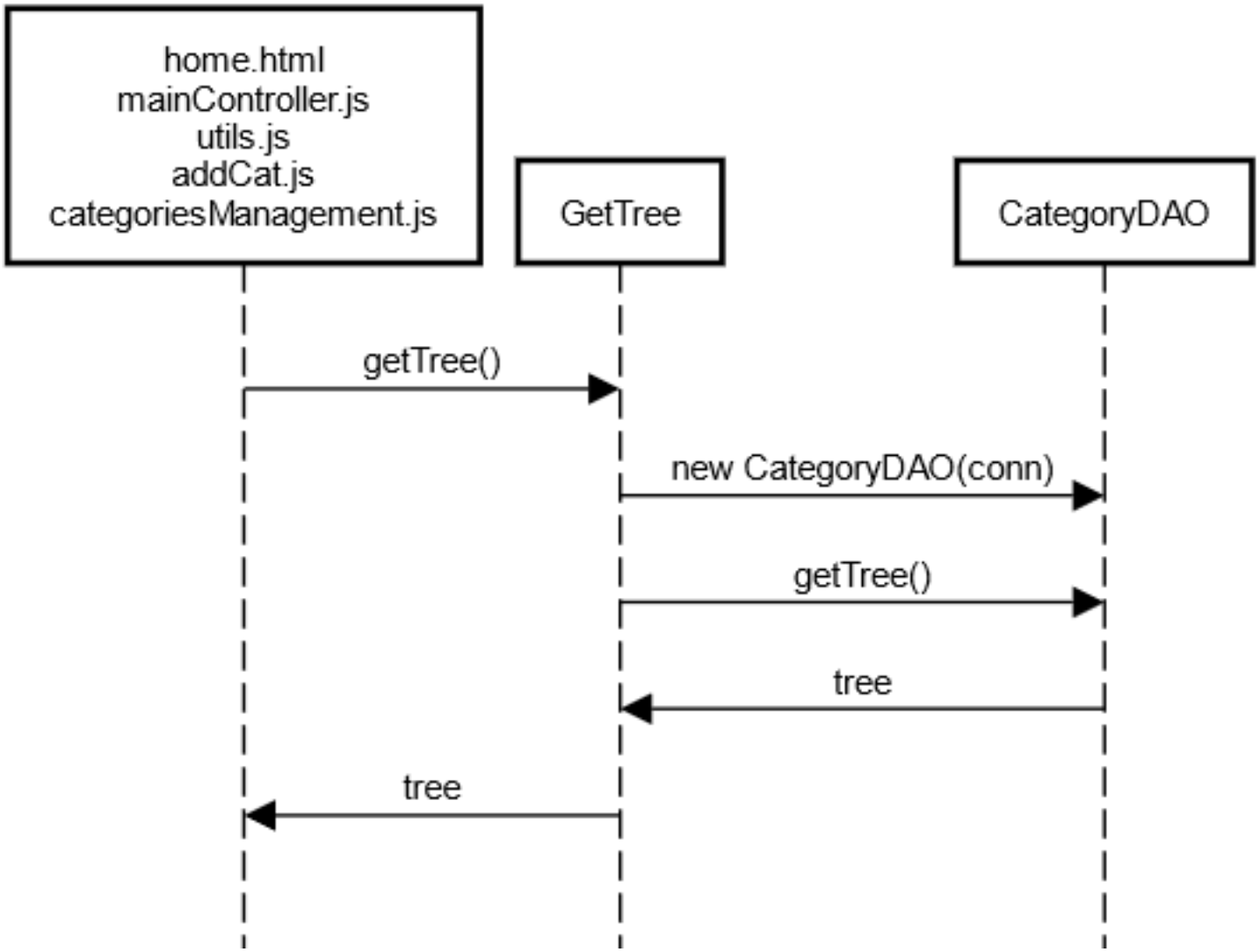
| Client side | | Server side | |
|---|---|---|---|
| **Evento** | **Azione** | **Evento** | **Azione** |
| index → login form → submit | Controllo dati | POST (username, password) | Controllo credenziali |
| home → category tree →drag | Identificazione categorie per il drop | - | - |
| Home → category tree → drop | Comparsa finestra di dialogo, aggiornamento, comparsa pulsante Save | - | - |
| Home → save | Aggiornamento stato | POST (idList) | Controllo, cambio ordine categorie |
| Home → Add Category form → submit | Controllo dati | POST (name, father) | Controllo, aggiunta della categoria |
| Logout | - | GET | Terminazione della sessione |

# Controller / event handler

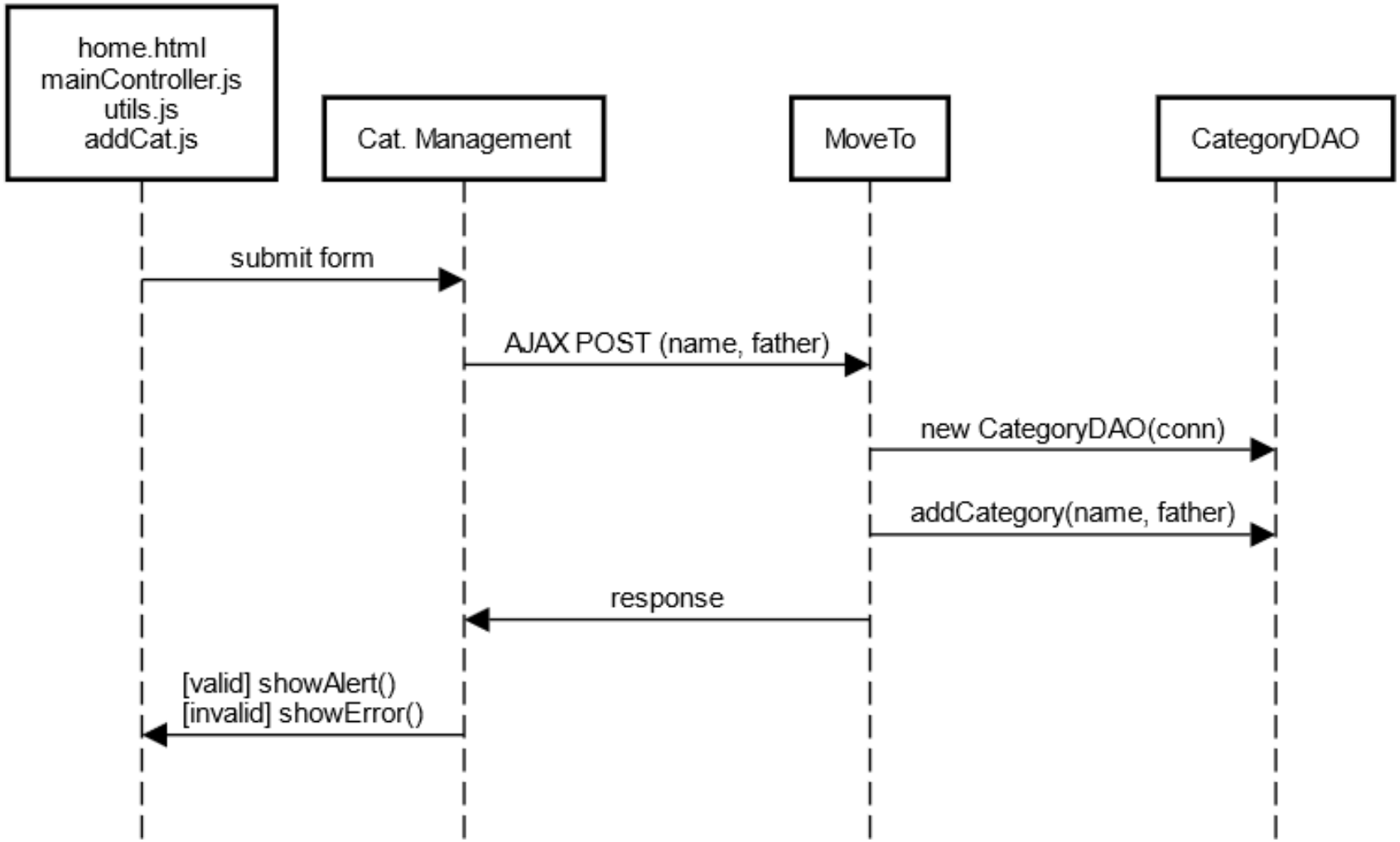
| Client side | | Server side | |
|---|---|---|---|
| **Evento** | **Controllore** | **Evento** | **Controllore** |
| index → login form → submit | Function makeCall | POST username password | CheckLogin (servlet) |
| Home → load | Function init | GET (no parameters) | GetTree (servlet) |
| Home → category tree → drag | Function drag | - | - |
| Home → category tree → drop | Function drop | - | - |
| Home → category tree → save | Function makeCall | POST (name, father) | MoveTo (servlet) |
| Home → add category form → submit | Function addCategory | POST (name, father) | AddCategory (servlet) |
| Logout | - | GET (no parameters) | Logout (servlet) |


Login
index.html
checkLogin.js
utils.js
CheckLogin
UserDAO
Session
home.html
GET
AJAX POST
(user, pwd)
new UserDAO(conn)
checkCredentials(user, pwd)
[user!=null] setAttribute("user", user)
[user!=null]status code 200
[user||pwd==""]status code 400
[invalid]status code 401
[status code 200]window.location.href="home.html"

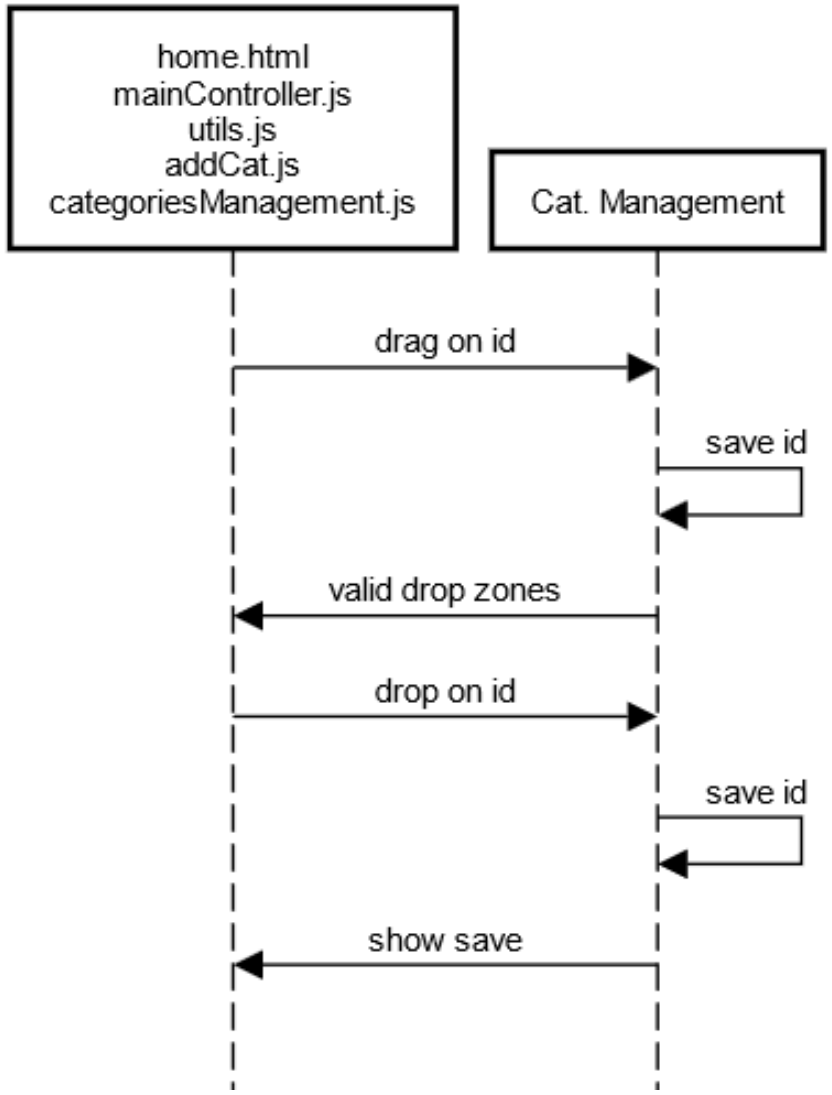
# Init Home


home.html
mainController.js
utils.js
addCat.js
categoriesManagement.js
GetTree
CategoryDAO
getTree()
new CategoryDAO(conn)
getTree()
tree
tree

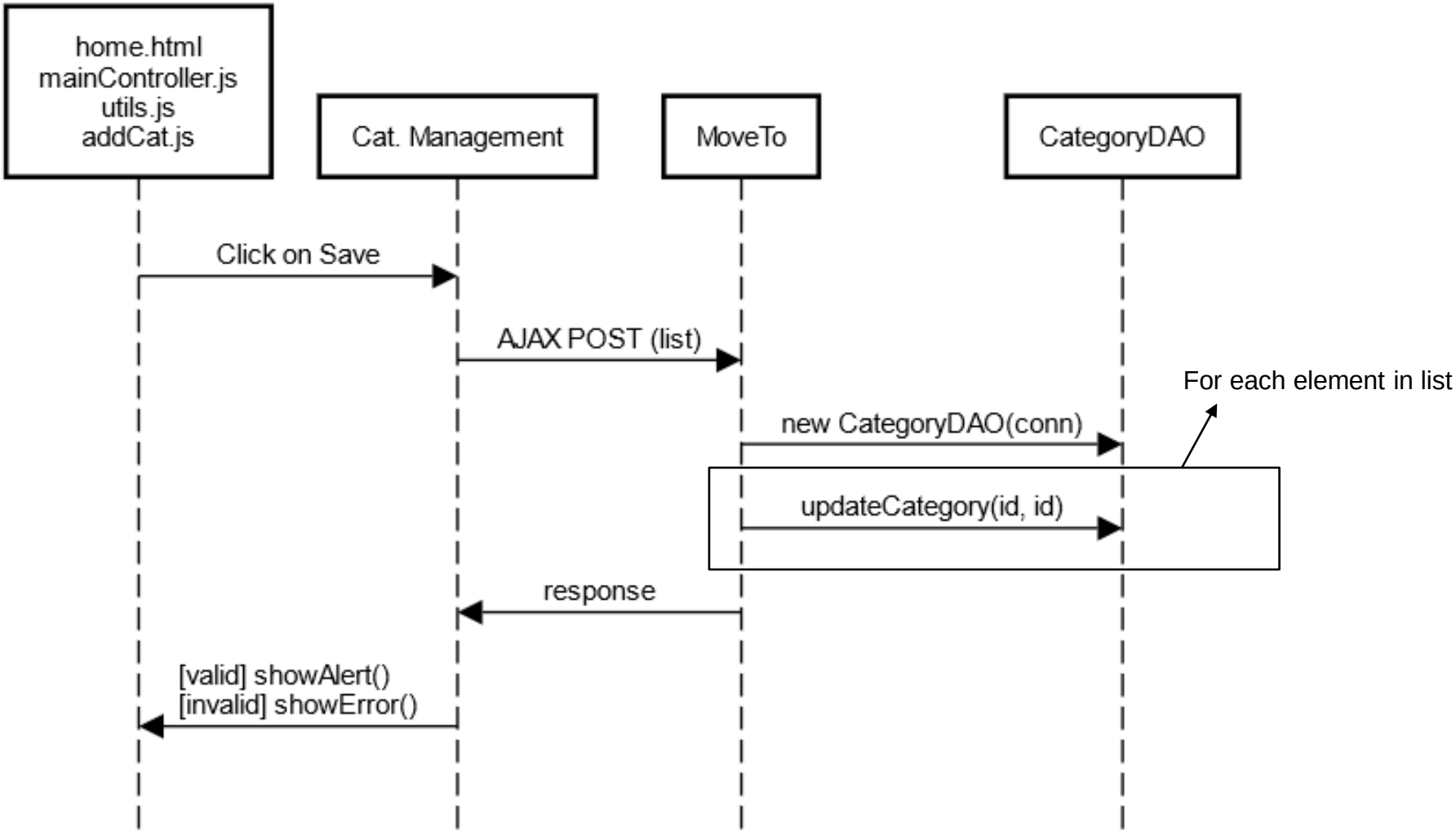
# Add category



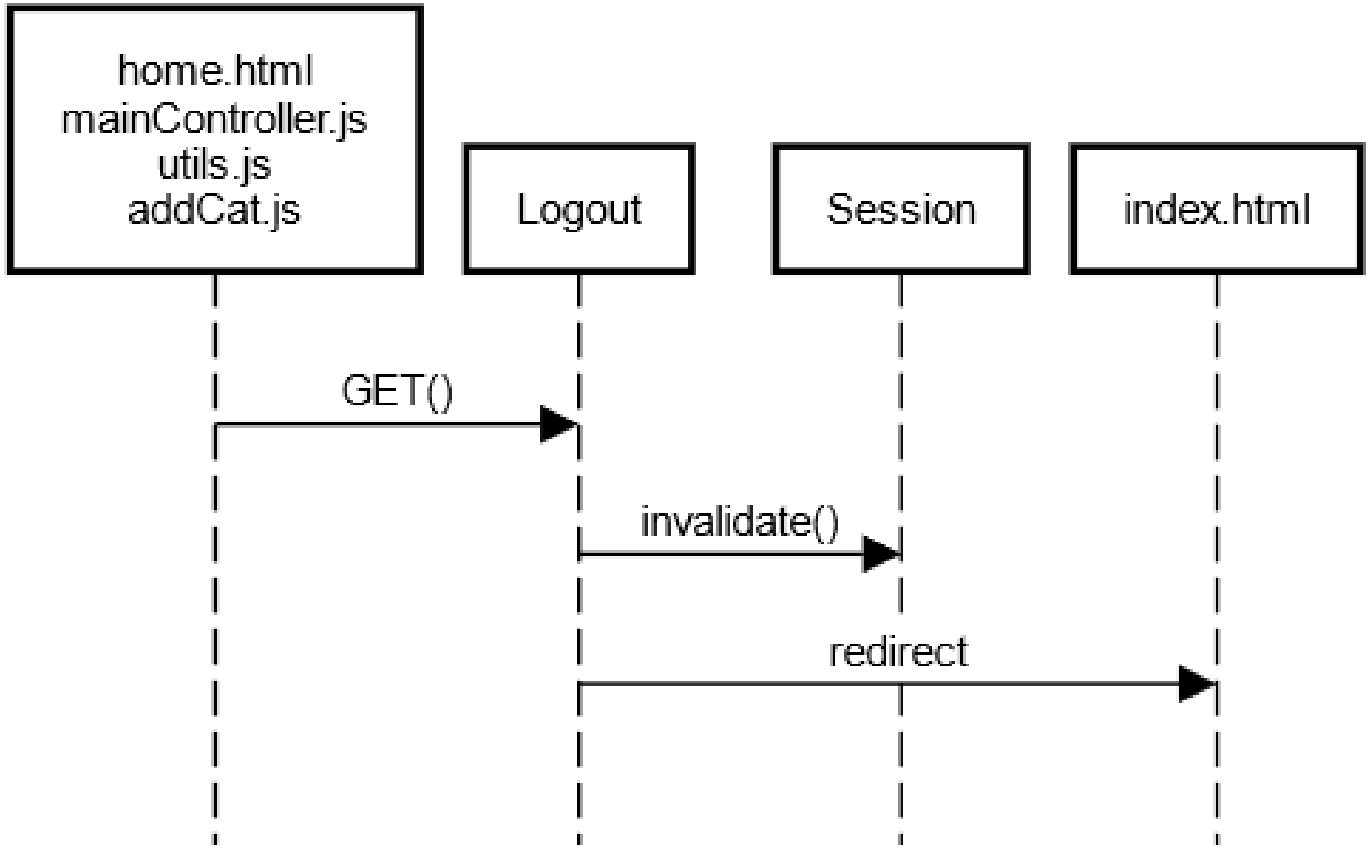
# Moving categories



# Saving moves



# Logout


home.html
mainController.js
utils.js
addCat.js
Logout
Session
index.html
GET()
invalidate()
redirect