30 OCTOBER 2023

PROGETTO DI TECNOLOGIE INFORMATICHE PER IL WEB

POLITECNICO DI MILANO

MATTEO BRISCINI
MATRICOLA: 960708
Matteo.briscini@mail.polimi.it

INDICE

1 REQUIREMENTS

1.1 Specification

An application allows the user (for example the environmental services manager of a region) to manage a collection of satellite images and a classification taxonomy useful for labeling images in order to allow searching by category. After logging in, the user accesses a HOME page where a hierarchical tree of categories appears. The categories do not depend on the user and are shared among all users. An example of a tree branch is as follows:

- 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

L The user can insert a new category into the tree. To do this, use a form on the HOME page where you specify the name of the new category and choose the parent category. Sending the new category involves updating the tree: the new category is attached to the parent category as the last sub element. The new category is assigned a numerical code that reflects its position (for example, the new category "Asbestos in pipes", daughter of the category "9111 Asbestos" takes on the code 91113). After creating a category, the HOME page shows the updated tree. To speed up the construction of the taxonomy, the user can copy an entire subtree in a given position: to do this click on the "copy" link associated with the root category of the subtree to be copied. Following this action, the application shows, again on the HOME page, the tree with the subtree to copy highlighted: all the other categories have a "copy here" link. For example, following a click on the “copy” link associated with the “9111 Asbestos” the application displays the tree as follows.

- 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

Selecting a “copy here” link causes a copy of the subtree to be inserted as the last child of the destination category. For example, selecting the “copy here” link of the “9 Solid Materials” category involves the following modification of the tree:

- 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
- 93 Amianto >>copia
- 931 Amianto in lastre >>copia
- 932 Amianto in frammenti >>copia

Changes made by a user and saved in the database become visible to other users. For simplicity, assume that for each category the maximum number of subcategories is 9, numbered from 1 to 9. In this case the copy operation must check that the move does not cause a number of subcategories greater than 9. Also provide a link “copy here” not associated with a taxonomy node which allows you to copy a sub-tree to the first level of the taxonomy (if 9 nodes do not already exist at the first level of the taxonomy).

1.2 Addition to the specification (HTML version)

- Added “cut” and “delete” functions.
- Added a search bar for multiple parameters.
- Through the DB it is possible to define a level for the user (0= guest, 1 = admin), the user interface changes depending on the user's level, showing or hiding buttons for functions for which he or she has authorization or not.

1.3 Extension to the specification for the JavaScript version

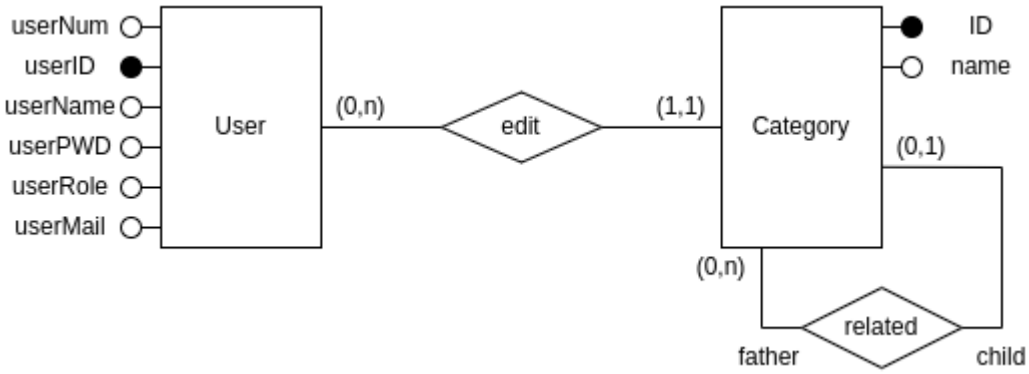
Create a web server client application that extends and/or modifies the previous specifications as follows:

- After the user logs in, the entire application is created with a single page.
- Each user interaction is managed without completely reloading the page, but produces the asynchronous invocation of the server and the possible modification of the content to be updated following the event.
- The function of copying a subtree is achieved by drag & drop. Following the drop of the root of the subtree to be copied, a dialog box appears with which the user can confirm or cancel the copy. Confirmation produces the update only on the client side of the tree. Deletion takes you back to the state before the drag & drop. Following confirmation, a SAVE button appears which allows the modified taxonomy to be saved on the server side.
- The user can click on the name of a category. Following this event, an input field containing the editable name string appears instead of the name. The event of loss of focus of the input field causes the modified name of the category to be saved in the database.

2 DATABASE DESIGN

2.1 ER schema

From the analysis of the specifications, the following relational scheme was obtained:



Using the userID as the primary key in the User table allows you to rename the user's account.
The taxonomy tree is stored via the parent-child self-relationship; a father can have multiple children, but the case of multiple inheritance is not foreseen.

The edit relationship allows you to verify, at DB level, that the user who created a new category has a userRole of sufficient level to perform this operation (1 or higher).

2.2 Code for the creation of SQL tables

2.2.1 User

The code for creating the User table is shown below:

```
User CREATE TABLE IF NOT EXIST `usersCredentials` (
            `userNum` int NOT NULL AUTO_INCREMENT,
            `userID` varchar(45) NOT NULL,
            `username` varchar(45) NOT NULL,
            `userPWD` varchar(45) NOT NULL,
            `userRole` int NOT NULL,
            `userMail` varchar(45) NOT NULL,
            PRIMARY KEY (`userID`),
            UNIQUE KEY `userID_UNIQUE` (`userID`),
            UNIQUE KEY `username_UNIQUE` (`username`),
            UNIQUE KEY `userMail_UNIQUE` (`userMail`),
            UNIQUE KEY `userNum_UNIQUE` (`userNum`)
)
```

2.2.2 Category

The code for creating the User table is shown below:

```
CREATE TABLE IF NOT EXIST `categories` (
            `ID` bigint NOT NULL,
            `name` varchar(45) NOT NULL,
            `fatherID`bigint DEFAULT NULL,
            `lastModifier` varchar(45) NOT NULL,
            PRIMARY KEY (`ID`),
            UNIQUE KEY `ID_UNIQUE` (`ID`),
            KEY `lastModifier_idx` (`lastModifier`),
             CONSTRAINT `creatorKey` FOREIGN KEY (`lastModifier`) REFERENCES `usersCredentials`
(`userID`)
)
```

2.3 Triggers

In addition to the constraints on individual attributes and those due to the presence of foreign keys, some triggers have been designed (of the AFTER INSERT, AFTER UPDATE or AFTER DELETE type).

The triggers, shown below, execute queries to verify that inconsistent elements or elements that do not respect the defined constraints are not saved in the DB. If inconsistent elements are found, the triggers throw an SQL exception.

2.3.1 Not allowed user

The trigger, shown below, verifies when adding a new category that the user carrying out the operation has the necessary permissions. This filter is also identically duplicated in the AFTER UPDATE case study (not reported for summary).

```
DELIMITER //
CREATE TRIGGER add_new_category_by_not_allowed_user
AFTER INSERT ON categories FOR EACH ROW BEGIN
IF EXISTS(
            SELECT * FROM categories JOIN usersCredentials ON lastModifier = userID
            WHERE userRole<1
)THEN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'you do not have autorization for this
operation';
END IF;
END//
DELIMITER ;
```

2.3.2 related relationship consistency (AFTER INSERT)

This trigger verifies, after the insertion of a new category, that all the "fatherID" numeric codes relate to categories actually saved in the database.

```
DELIMITER //
CREATE TRIGGER add_new_categories_with_not_existing_father
AFTER INSERT ON categories FOR EACH ROW BEGIN
IF EXISTS(
                        SELECT * FROM categories
                        WHERE fatherID != 0 AND fatherID NOT IN (SELECT ID FROM categories)
)THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'it is not possible to add a child to non
existent father';
END IF;
END//
DELIMITER;
```

2.3.3 related relationship consistency (AFTER UPDATE)

This trigger verifies, after updating an existing category, that all the "fatherID" numeric codes relate to categories actually saved in the database.

```
DELIMITER //
CREATE TRIGGER add_new_categories_with_not_existing_father_after_update
AFTER UPDATE ON categories FOR EACH ROW BEGIN
IF EXISTS(
                        SELECT * FROM categories
                        WHERE (@trigger_disable IS NULL OR @trigger_disable!=1) AND fatherID != 0
AND fatherID NOT IN (SELECT ID FROM categories)
)THEN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'it is not possible to add a child to non existent
father';
END IF;
END//
DELIMITER ;
```

NOTE: *trigger_disable* is a session variable, when multiple update or delete occur is set to 1 to disable triggers. The triggers will be rehabilitated before the last query to verify the database status.

2.3.4 related relationship consistency (AFTER DELETE)

This trigger verifies, after deleting an existing category, that all the "fatherID" numerical codes relate to categories actually saved in the database.

```
DELIMITER //
CREATE TRIGGER add_new_categories_with_not_existing_father_after_delete
AFTER DELETE ON categories FOR EACH ROW BEGIN
IF EXISTS(
                        SELECT * FROM categories
                        WHERE (@trigger_disable IS NULL OR @trigger_disable!=1)AND fatherID != 0
AND fatherID NOT IN (SELECT ID FROM categories)
)THEN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'non puoi rimuovere una categoria se prima non
rimuovi i suoi figli';
END IF;
END//
DELIMITER ;
```

2.4 Significative query

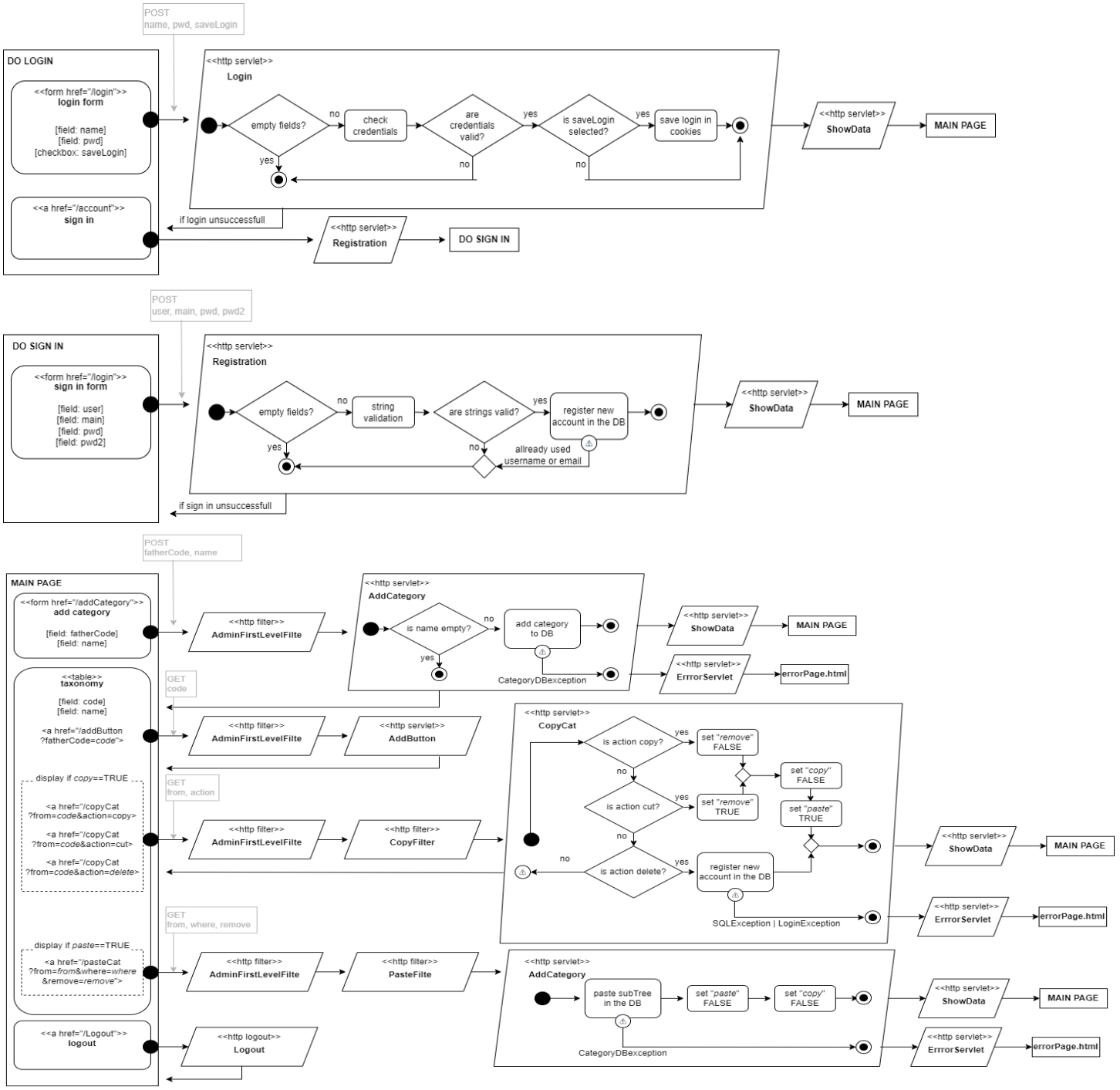
Below is the query to select all categories dominated by another category N

This query is used in multiple methods, useful for implementing functions such as "Paste", "Search" and "Select" (even if partially modified to meet different implementation needs).

```
SELECT  * FROM categories AS t, lastSon AS l
WHERE t.ID IN (
        WITH RECURSIVE subTree (fatherID, ID) AS (
                SELECT fatherID, ID FROM categories
                WHERE fatherID = N
                UNION ALL
                SELECT c.fatherID, c.ID FROM (subTree AS f) JOIN (categories AS c) ON f.ID =
c.fatherID
        )
        SELECT c.ID FROM subTree AS f JOIN categories AS c ON f.ID = c.ID
        UNION ALL
        SELECT ID from categories WHERE ID= N
)
```

3 PURE HTML APP DESIGN

3.1 Servlet structures



3.2 Access verification

All servlets (excluding Login, Logout and Registration) are preceded by a filter type servlet, called "Login filter". This checks for the presence of a User attribute in the session, confirming the success of the previous login action.

The User type attribute is also used to store user data, used when performing queries that act on the "Categories" table, thus allowing the operation of the "add_new_category_by_not_allowed_user" SQL filter”.

4 HTML VERSION COMPONENTS

4.1 Interaction with the DB

To manage connections with the database, a "ConnectionsHandler" class has been implemented, which allows you to save active connections, avoiding opening a new connection for each access to the DB.

Two methods are implemented in the class:

- takeConnection: which allows DAOs to use one of the available connections or create a new one if there are none;
- releaseConnection: once the DAO has finished using the connection with the DB it returns it, making it available for future use.

On the database, the lifespan of a connection has been reduced to one hour.

When a DAO requests access to a connection, if there are any expired connections, they are removed from the ConnectionsHandler.

4.2 Data beams

- Category
- UpdateCategory
- User

4.3 Funzionalità DAO

4.3.1 CategoriesDAO

- getCategories(from [optional], context)
- addCategory(fatherCode, name, creator, context)
- removeSubTree(fatherCode, context)
- pasteSubTree(from, where, creator, remove, context)
- addNewHead(name, creator, context)
- searchCategory(search, context)

4.3.2 UserDAO

- checkCredentials(pwd, user, context)
- checkUserID(userID, user, context)
- getUserRole(userID, context)
- signInMandatoryData(user, email, pwd, userRole, context)

4.4 Servlets

4.4.1 Controls

- AddButton
- AddCategory
- CopySubTree
- Login
- Logout

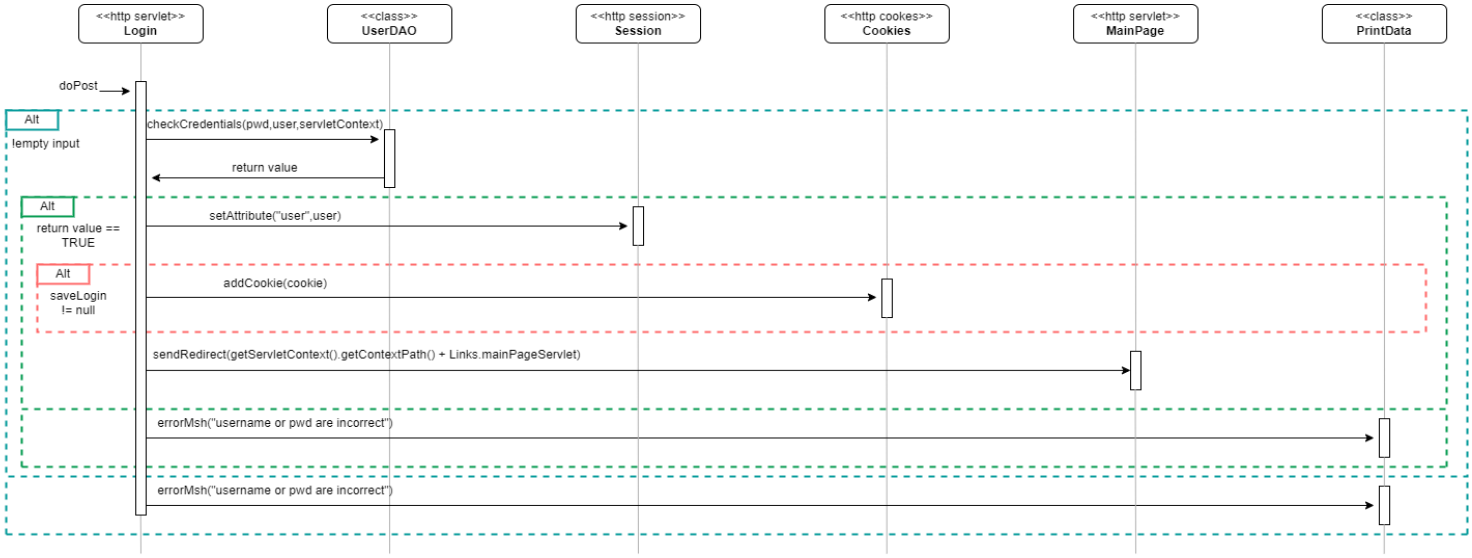
- PasteSubTree
- Registration
- SearchCategory
- ShowData
- ErrorServlet

4.4.2 Filters

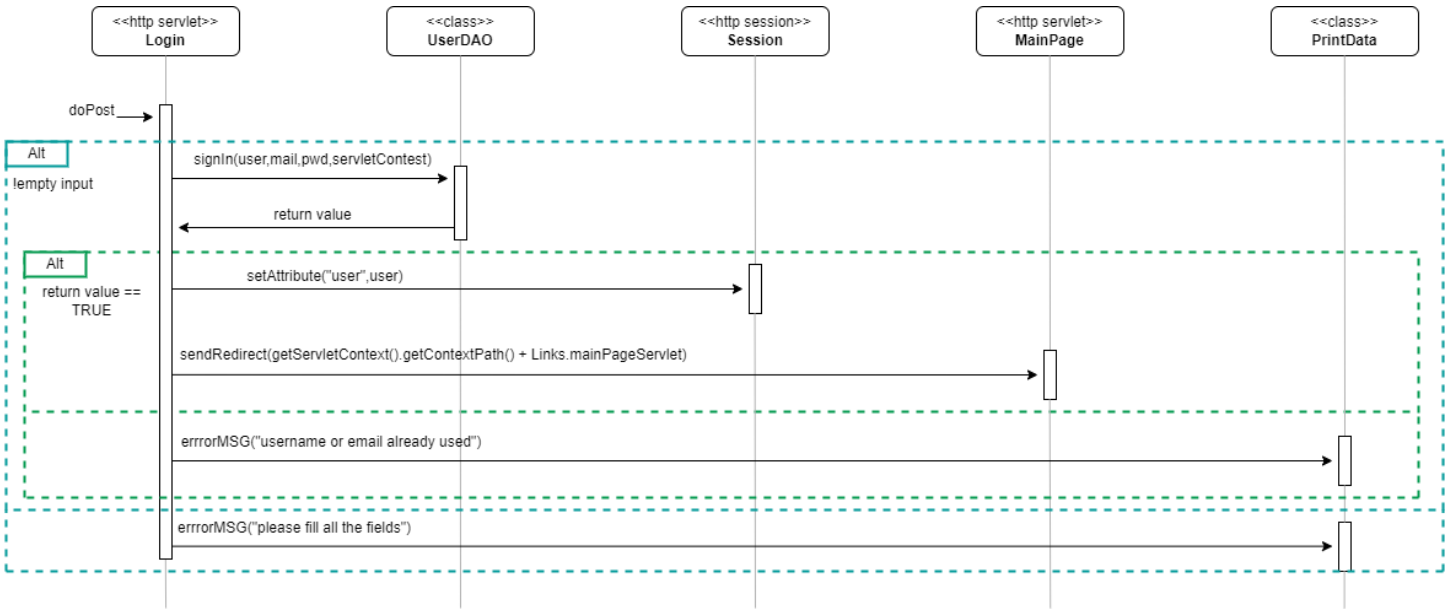
- AdminFirstLevelFilter
- CopyFilter
- LoginFilter
- PasteFilter

5 SEQUENCE DIAGRAM FOR HTML VERSION

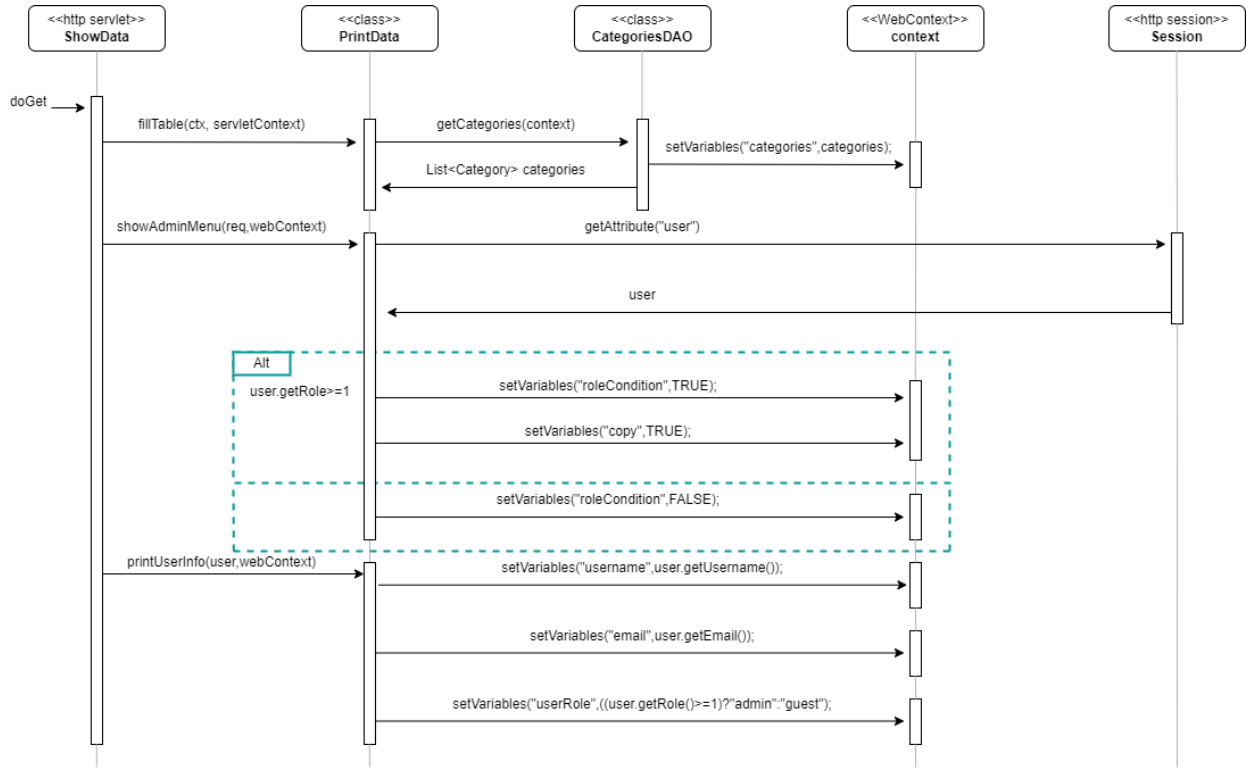
5.0.1 A user performs the login.



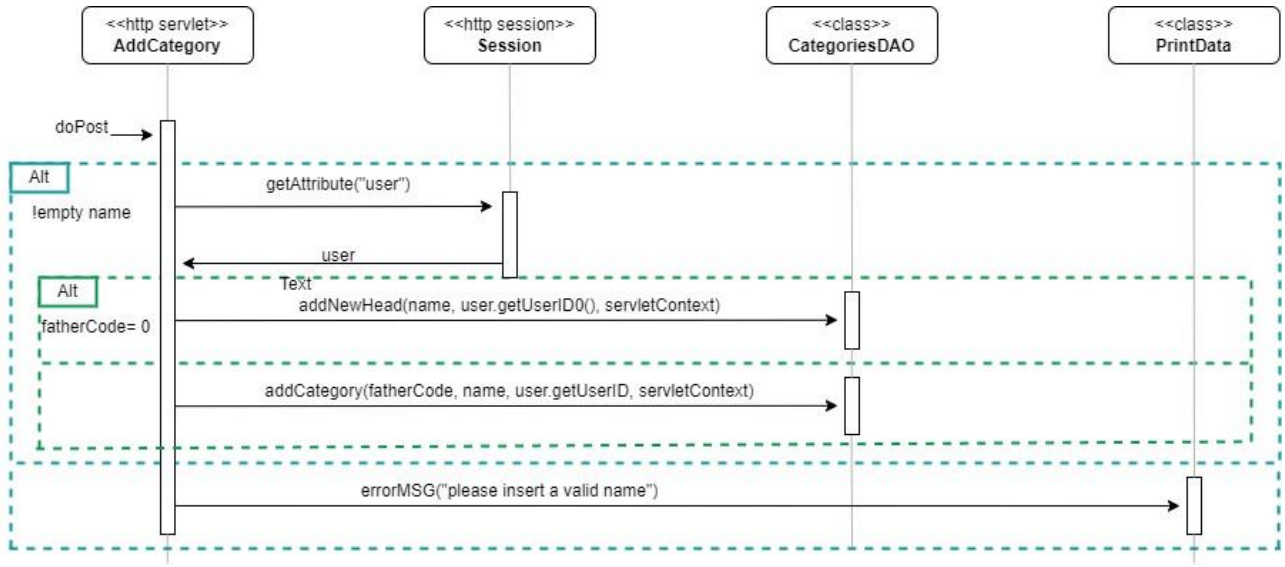
5.0.2 A user performs the sign-in



5.0.3 A user asks for the main page.

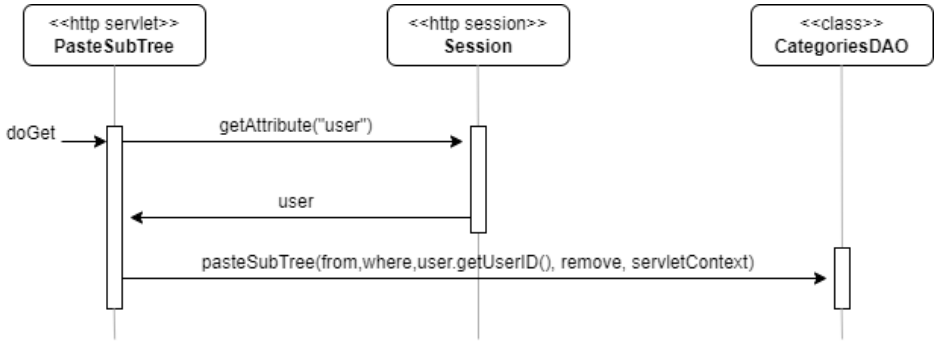


5.0.4 A user adds a new category.

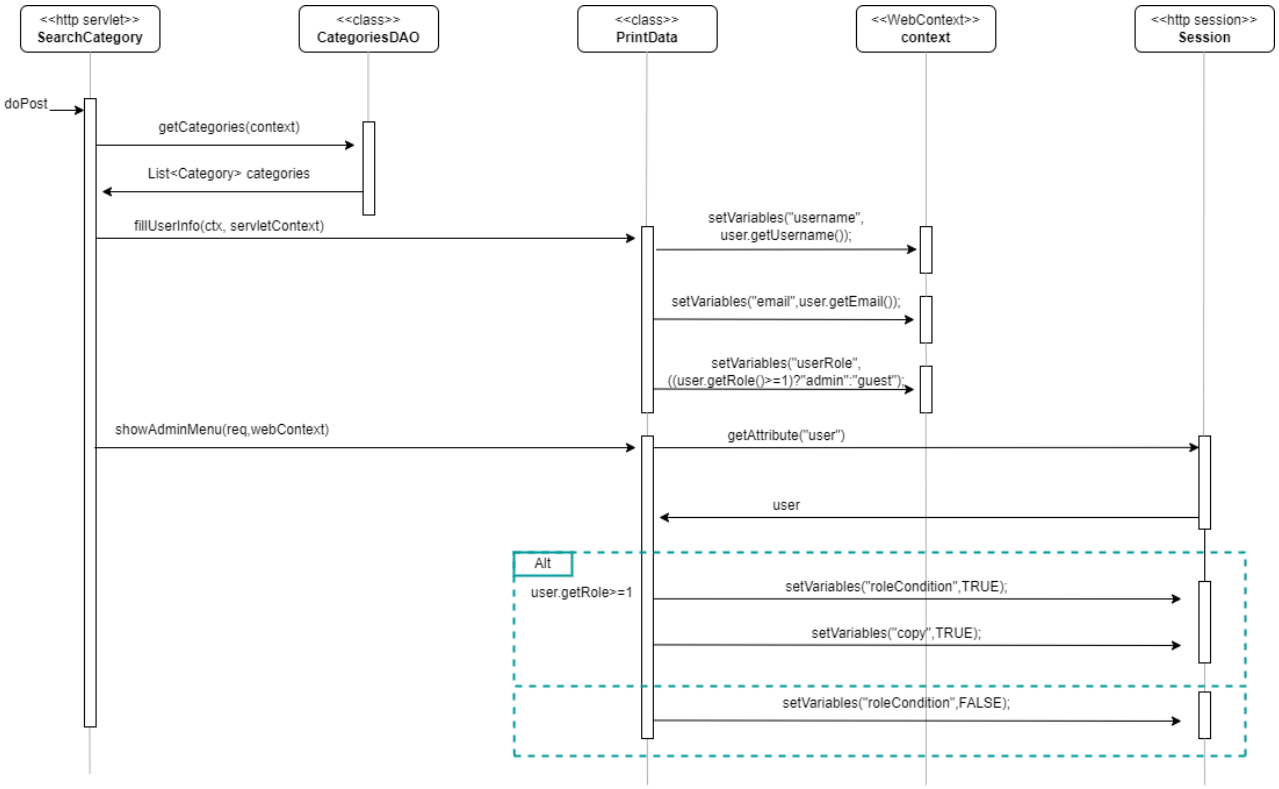


5.0.5 A user applies a new modification to the taxonomy.

Through this action, a user saves his changes (copy-paste or cut-paste) in the database.

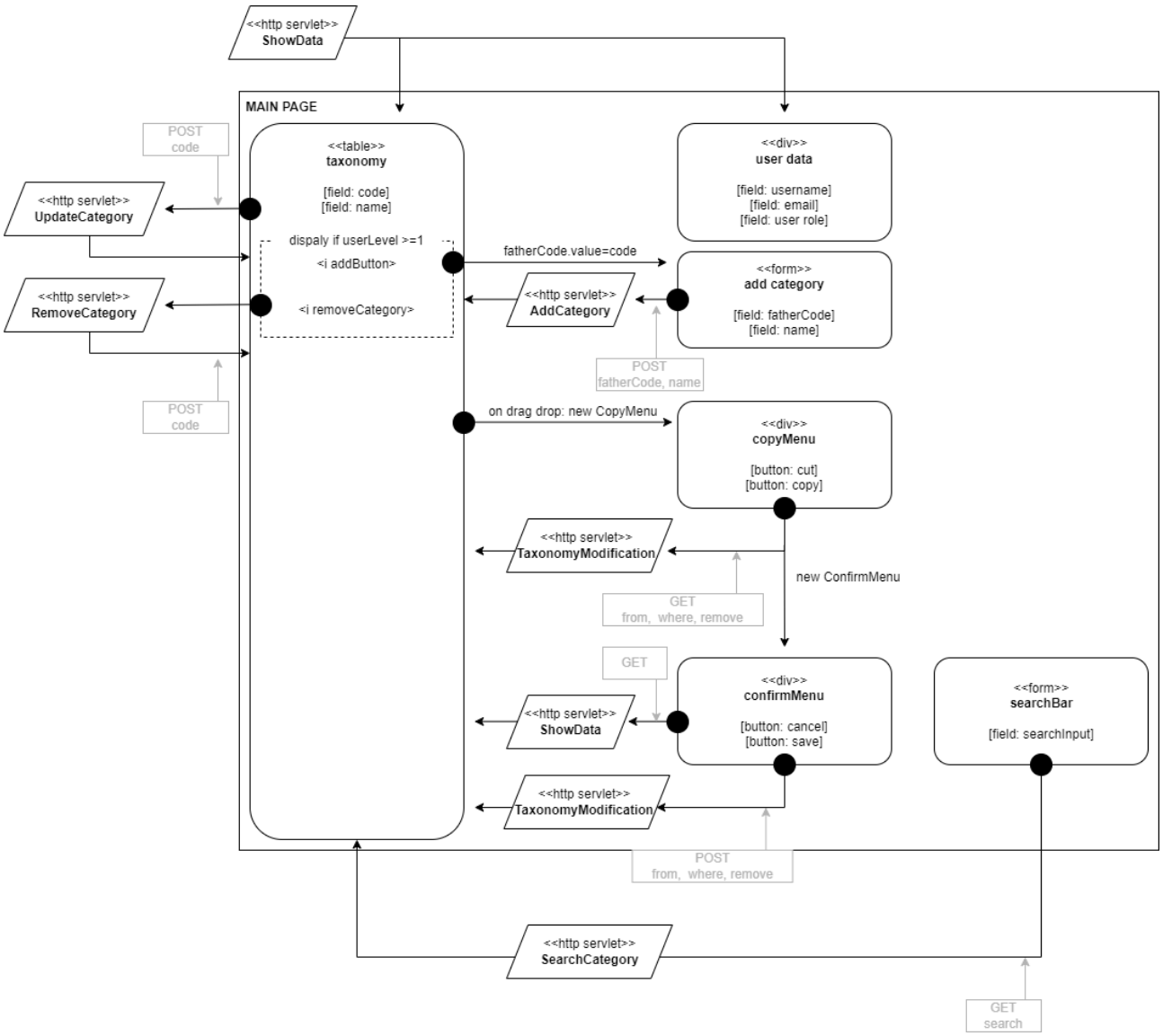


5.0.6 A user search for one or more categories.



6 JAVASCRIPT VERSION APP DESIGN

The specifications require, for the javaScript version, that the site is structured into a single page after the login (or registration) action. Additionally, all updates to the single page must occur dynamically without completely reloading the web page.



NOTE: as specifications require, after copy action, a local taxonomy is provided to the user. So, the user can save that taxonomy making it available for all the users.

however, as the taxonomy update computation can be a very high resource-demanding process, it is handled by the server.

7 JAVASCRIPT VERSION COMPONENTS

7.1 Interaction with the DB

For this section, refer to what was said for the pure-HTML version since no changes have been made to these components.

7.2 Servlets

7.2.1 Controls

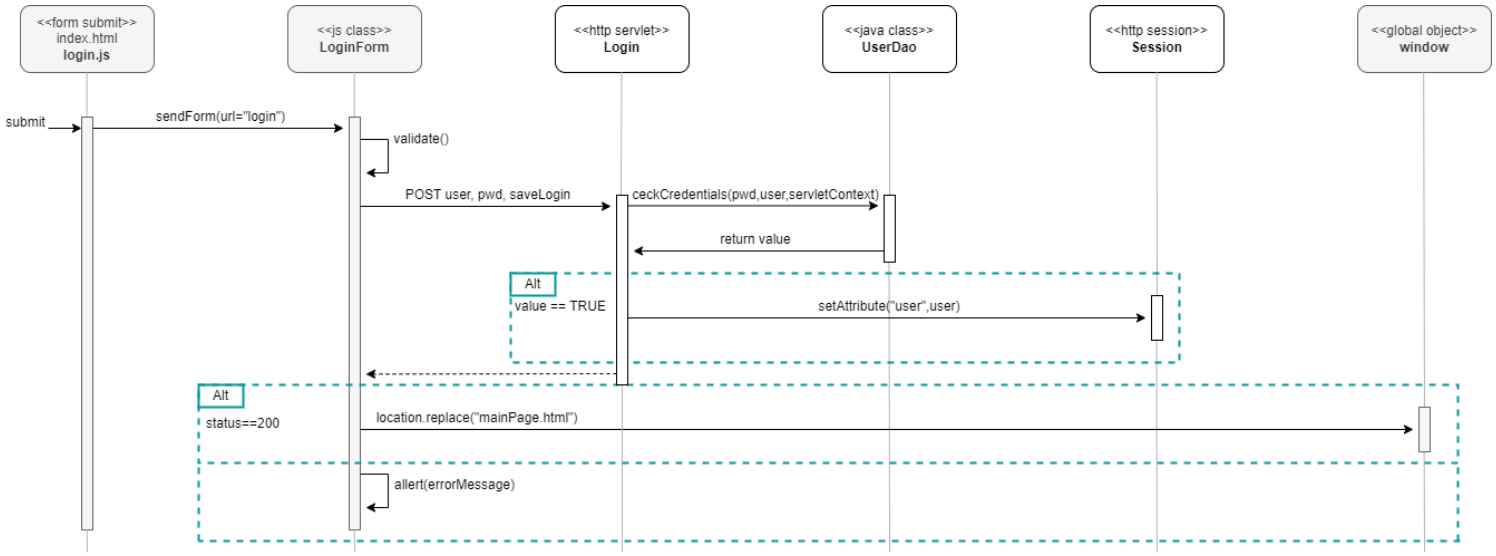
- AddCategory
- Login
- LoginCookies
- Logout
- RemoveCategory
- SearchCategory
- ShowData
- SignIn
- TaxonomyModification
- UpdateCategory

7.2.2 Filters

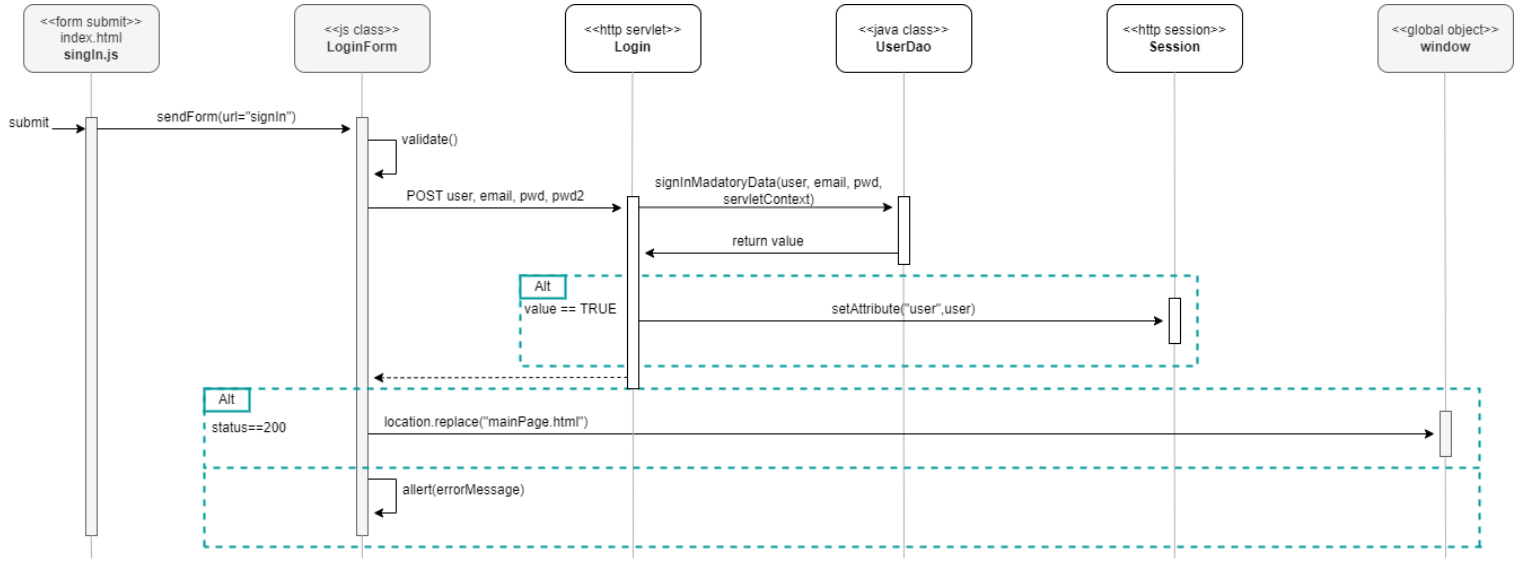
- AdminFistLevlFilter
- LoginFilter
- PasteFilter

8 SEQUENCE DIAGRAM FOR JAVASCRIPT VERSION

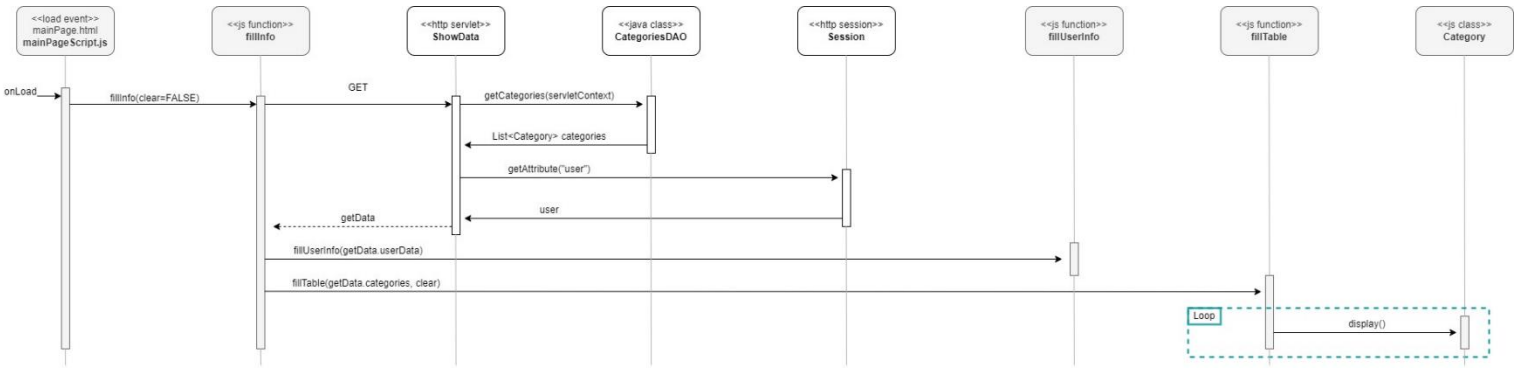
8.0.1 A user performs the login.



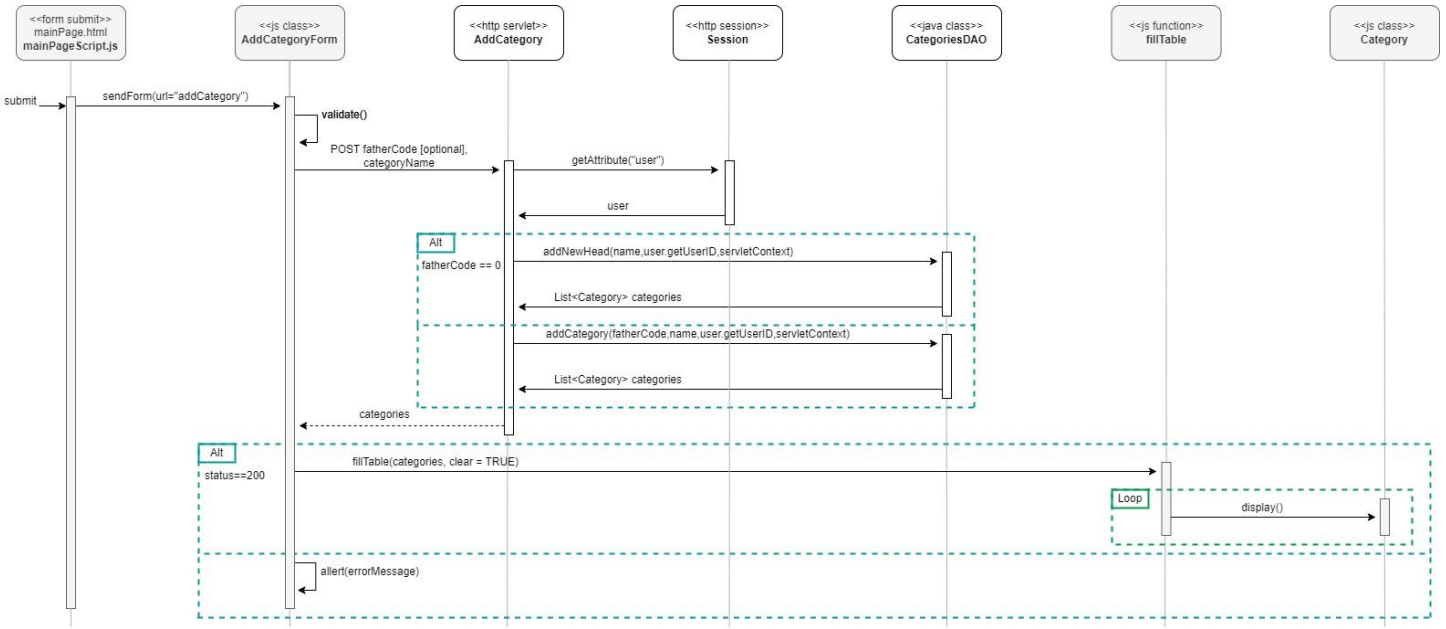
8.0.2 A user performs the sign-in.



8.0.3 A user asks for the main page.



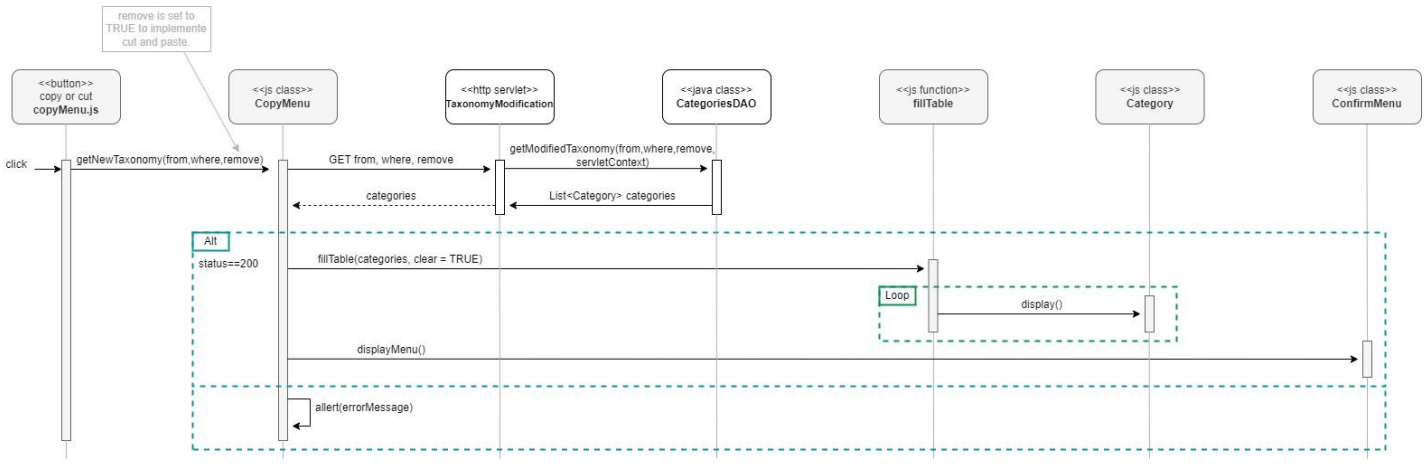
8.0.4 A user adds a new category.



8.0.5 A user asks for a local modified version of the taxonomy.

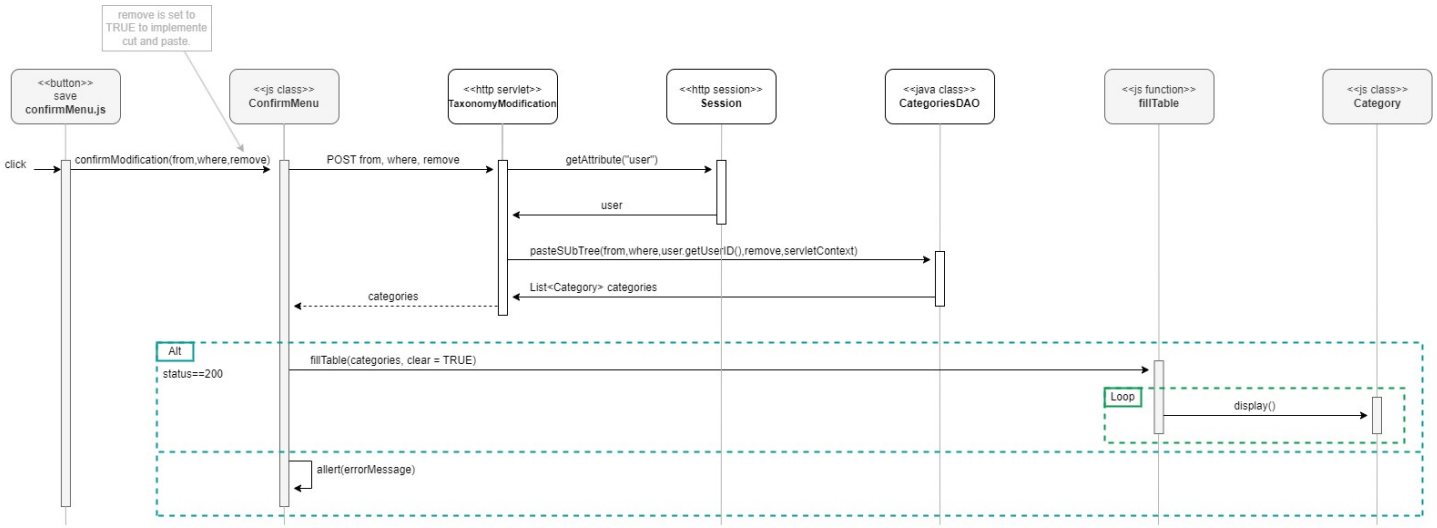
Following a copy-paste (or cut-paste) operation, the user sees the result of his changes in a local version of the taxonomy. The latter is however processed on the server side (see note CHAP.6 for the reasons for this choice).

> **NOTE:** the local version of the taxonomy is only visible to the current user.



8.0.6 A user saves a modified version of the taxonomy.

After receiving the local version of the modified taxonomy, the user can decide to save it in the DB, thus making the change visible to the rest of the users.



8.0.7 A user updates a category name.