Gestore File RIA

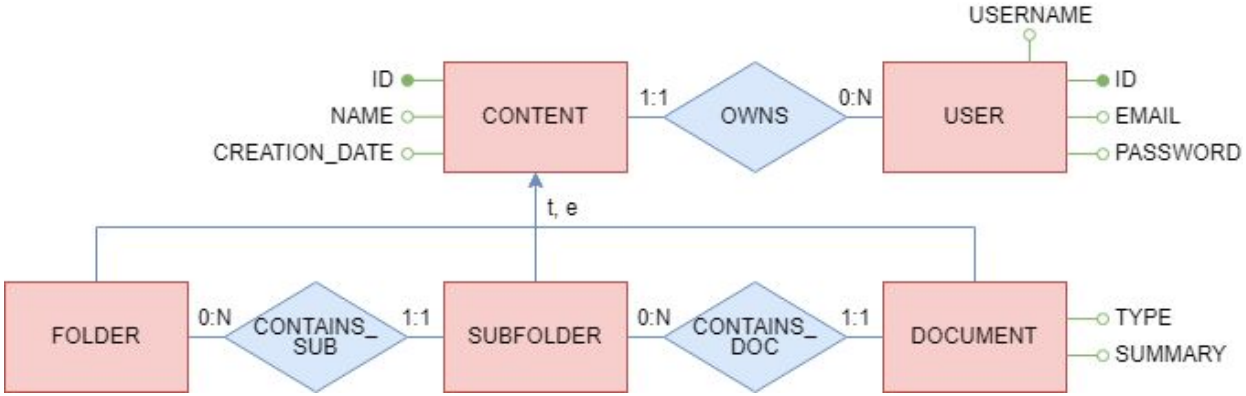
(Progetto 3) TIW 2021-2022

Data requirements analysis

Un'applicazione web consente la gestione di cartelle, sottocartelle e documenti online. L'applicazione supporta registrazione e login di utenti mediante una pagina pubblica con opportune form. La registrazione controlla l'unicità dello username. Una cartella ha un proprietario, un nome e una data di creazione e può contenere (solo) sottocartelle. Una sottocartella può contenere (solo) dei documenti. Un documento ha un proprietario, nome, una data di creazione, un sommario e un tipo. Quando l'utente accede all'applicazione appare una HOME PAGE che contiene un albero delle proprie cartelle e delle sottocartelle.
Nell'HOME PAGE l'utente può selezionare una sottocartella e accedere a una pagina DOCUMENTI che mostra l'elenco dei documenti di una sottocartella. Ogni documento in elenco ha due link: accedi e sposta. Quando l'utente seleziona il link accedi, appare una pagina DOCUMENTO che mostra tutti i dati del documento selezionato. Quando l'utente seleziona il link sposta, appare la HOME PAGE con l'albero delle cartelle e delle sottocartelle; in questo caso la pagina mostra il messaggio "Stai spostando il documento X dalla sottocartella Y. Scegli la sottocartella di destinazione", la sottocartella a cui appartiene il documento da spostare NON è selezionabile e il suo nome è evidenziato (per esempio con un colore diverso). Quando l'utente seleziona la sottocartella di destinazione, il documento è spostato dalla sottocartella di origine a quella di destinazione e appare la pagina DOCUMENTI che mostra il contenuto aggiornato della sottocartella di destinazione. Ogni pagina, tranne la HOME PAGE, contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente. Una pagina GESTIONE CONTENUTI raggiungibile dalla HOME PAGE permette all'utente di creare una cartella, una sottocartella di una cartella esistente e un documento all'interno di una sottocartella.

Entities, attributes, relationships

Database design



N.B. la relazione di ownership verrà salvata solo per la cartella dato che non si gestirà la delega della creazione di contenuti ad altri utenti. Se una sottocartella o una cartella sono in una cartella avranno lo stesso owner della stessa da design.

SCHEMA LOGICO
USER(id, username, email, password)
FOLDER(id, name, creation_date, owner_id)
SUBFOLDER(id, name, creation_date, parent_folder_id)
DOCUMENT(id, name, creation_date, parent_folder_id, summary, type)

FOLDER.owner_id -> USER.id
SUBFOLDER.parent_folder_id -> FOLDER.id
DOCUMENT.parent_folder_id -> SUBFOLDER.id

```
CREATE TABLE `user` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `username` VARCHAR(30) NOT NULL UNIQUE,
    `email` VARCHAR(320) NOT NULL UNIQUE,
    `password` BINARY(32) NOT NULL,
    PRIMARY KEY(`id`)
);

CREATE TABLE `folder` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(30) NOT NULL,
    `creation_date` DATE NOT NULL,
    `owner_id` INT NOT NULL,
    PRIMARY KEY(`id`),
    CONSTRAINT `owner_id_folder` FOREIGN KEY
(`owner_id`) REFERENCES `user` (`id`) ON DELETE
CASCADE ON UPDATE NO ACTION
);
```

```
CREATE TABLE `subfolder` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(30) NOT NULL,
    `creation_date` DATE NOT NULL,
    `parent_folder_id` INT NOT NULL,
    PRIMARY KEY(`id`),
    CONSTRAINT `main_folder` FOREIGN KEY
(`parent_folder_id`) REFERENCES `folder` (`id`) ON
DELETE CASCADE ON UPDATE NO ACTION
);

CREATE TABLE `document` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(30) NOT NULL,
    `creation_date` DATE NOT NULL,
    `parent_folder_id` INT NOT NULL,
    `summary` TEXT,
    `type` VARCHAR(3),
    PRIMARY KEY(`id`),
    CONSTRAINT `sub_folder` FOREIGN KEY
(`parent_folder_id`) REFERENCES `subfolder` (`id`) ON
DELETE CASCADE ON UPDATE NO ACTION,
);
```

Application requirements analysis Pure HTML

Un'applicazione web consente la gestione di cartelle, sottocartelle e documenti online. L'applicazione supporta registrazione e login di utenti mediante una pagina pubblica con opportune form. La registrazione controlla l'unicità dello username. Una cartella ha un proprietario, un nome e una data di creazione e può contenere (solo) sottocartelle. Una sottocartella può contenere (solo) dei documenti. Un documento ha un proprietario, nome, una data di creazione, un sommario e un tipo. Quando l'utente accede all'applicazione appare una HOME PAGE che contiene un albero delle proprie cartelle e delle sottocartelle.
Nell'HOME PAGE l'utente può selezionare una sottocartella e accedere a una pagina DOCUMENTI che mostra l'elenco dei documenti di una sottocartella. Ogni documento in elenco ha due link: accedi e sposta. Quando l'utente seleziona il link accedi, appare una pagina DOCUMENTO che mostra tutti i dati del documento selezionato. Quando l'utente seleziona il link sposta, appare la HOME PAGE con l'albero delle cartelle e delle sottocartelle; in questo caso la pagina mostra il messaggio "Stai spostando il documento X dalla sottocartella Y. Scegli la sottocartella di destinazione", la sottocartella a cui appartiene il documento da spostare NON è selezionabile e il suo nome è evidenziato (per esempio con un colore diverso). Quando l'utente seleziona la sottocartella di destinazione, il documento è spostato dalla sottocartella di origine a quella di destinazione e appare la pagina DOCUMENTI che mostra il contenuto aggiornato della sottocartella di destinazione. Ogni pagina, tranne la HOME PAGE, contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente. Una pagina GESTIONE CONTENUTI raggiungibile dalla HOME PAGE permette all'utente di creare una cartella, una sottocartella di una cartella esistente e un documento all'interno di una sottocartella.

Pages (views), view components, events, actions

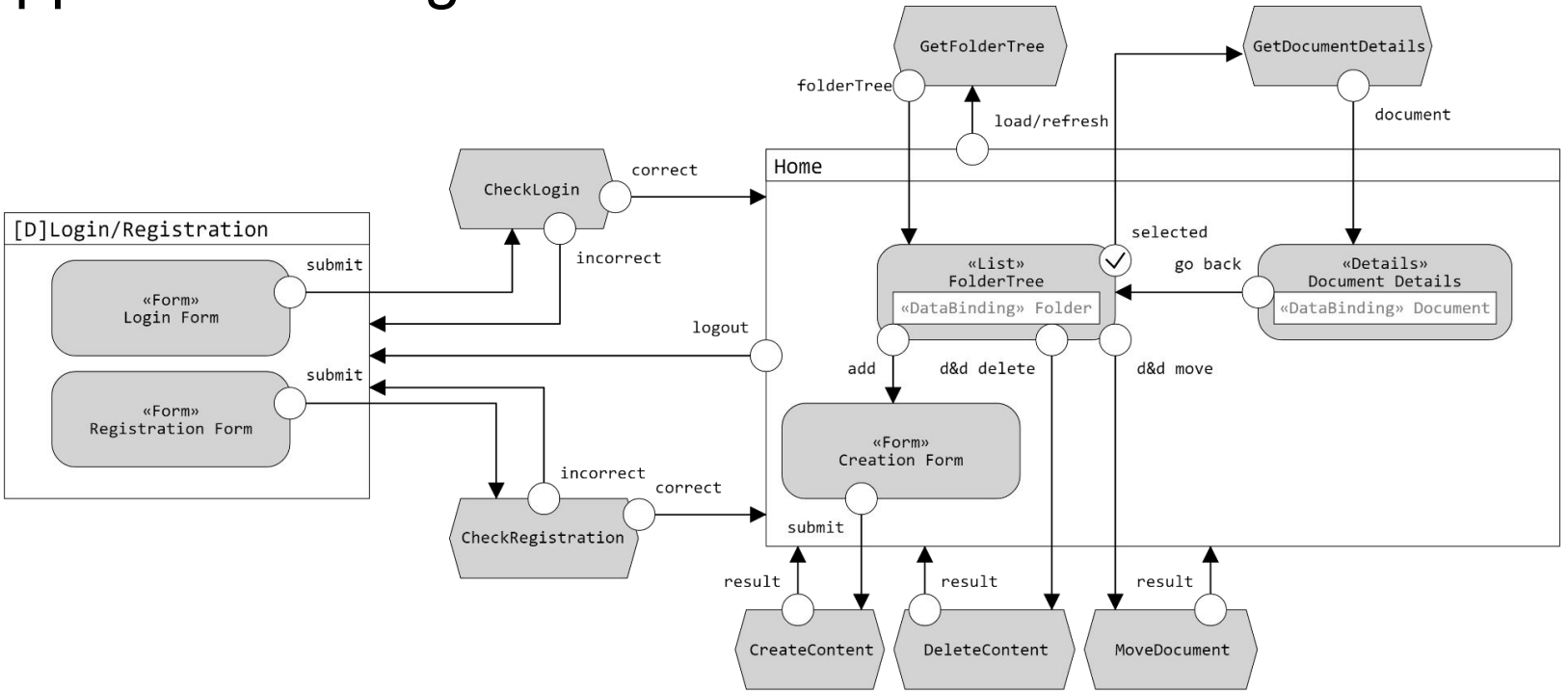
Modifiche RIA

- La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password", anche a lato client.
- Dopo il login, 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.
- Errori a lato server devono essere segnalati mediante un messaggio di allerta all'interno della pagina.
- La funzione di spostamento di un documento è realizzata mediante drag and drop.
- La funzione di creazione di una sottocartella è realizzata nella pagina HOME mediante un bottone AGGIUNGI SOTTOCARTELLA posto di fianco ad ogni cartella. La pressione del bottone fa apparire un campo di input per l'inserimento del nome della sottocartella.
- La funzione di creazione di un documento è realizzata nella pagina HOME mediante un bottone AGGIUNGI DOCUMENTO posto di fianco ad ogni sottocartella. La pressione del bottone fa apparire una form di input per l'inserimento dei dati del documento.
- Si aggiunge una cartella denominata "cestino". Il drag & drop di un documento o di una cartella nel cestino comporta la cancellazione. Prima di inviare il comando di cancellazione al server l'utente vede una finestra modale di conferma e può decidere se annullare l'operazione o procedere.

Completamento delle specifiche

- Quando l'utente accede all'applicazione per la prima volta gli viene mostrata la pagina di login e registrazione. La registrazione viene fatta anche con l'email e si controlla l'unicità anche di quello, per evitare che un utente crei più account collegati ad una sola email.
- Nella HOME PAGE ci sarà un collegamento per tornare alla pagina precedente solo se si sta spostando un documento. Cliccare su questo collegamento effettivamente annullerà lo spostamento del documento.
- Il comportamento del bottone "visualizza dettagli" verrà sostituito con il click sul nome del file.
- Il "tipo" di un documento viene considerato congruente alla sua estensione e per semplicità verrà imposto un limite massimo di 3 caratteri di lunghezza.

Application design



Server-Side Components

- Model Objects (Beans)
  - Content
  - User
  - Folder
  - Subfolder
  - Document
- Data Access Objects (Classes)
  - UserDAO
    - checkCredentials
    - registerUser
    - areSignUpInfoTakes
  - FolderDAO
    - getFoldersOfUser
    - getFolderById
    - deleteFolder
  - SubfolderDAO
    - getSubfoldersInsideFolder
    - getSubfolderById
    - createSubfolder
    - deleteSubfolder
  - DocumentDAO
    - getDocumentInsideSubfolder
    - getDocumentById
    - moveDocumentToSubfolder
    - createDocument
    - deleteDocument
- Controllers (Servlets)
  - CheckLogin
  - RegisterUser
  - GetFolderTree
  - MoveDocument
  - GetDocumentDetails
  - CreateContent
  - DeleteContent
  - Logout
- Views (Templates)
  - Login/Registration
  - Home
- Filters
  - BlockUnknownUsers
  - ForwardLoggedUsers

Client-Side Components

- Home
  - PageOrchestrator
    - **start**: inizializza i componenti
    - **show**: mostra i componenti di default della pagina
  - Header
    - **start**: inizializza gli eventListener
    - **show(state)**: mostra i bottoni in base allo stato
    - **handleModalInteraction(confirmed)**: gestisce la conferma o la chiusura del modal di cancellazione
    - **hideBin**: nasconde il cestino
  - Modal
    - **start**: inizializza gli eventListener
    - **show**: mostra il modalBox
  - FolderTree
    - **show(shouldFetch)**: mostra l'albero, se *shouldFetch* richiede al server i dati
    - **hide**: nasconde l'albero
  - DocumentDetails
    - **show(documentId)**: chiede al server e mostra i dettagli di un documento
    - **hide**: nasconde la finestra
- Login/Registration
  - Login form
    - gestisce submit ed errori
  - Registration form
    - gestisce submit ed errori

Eventi & Azioni

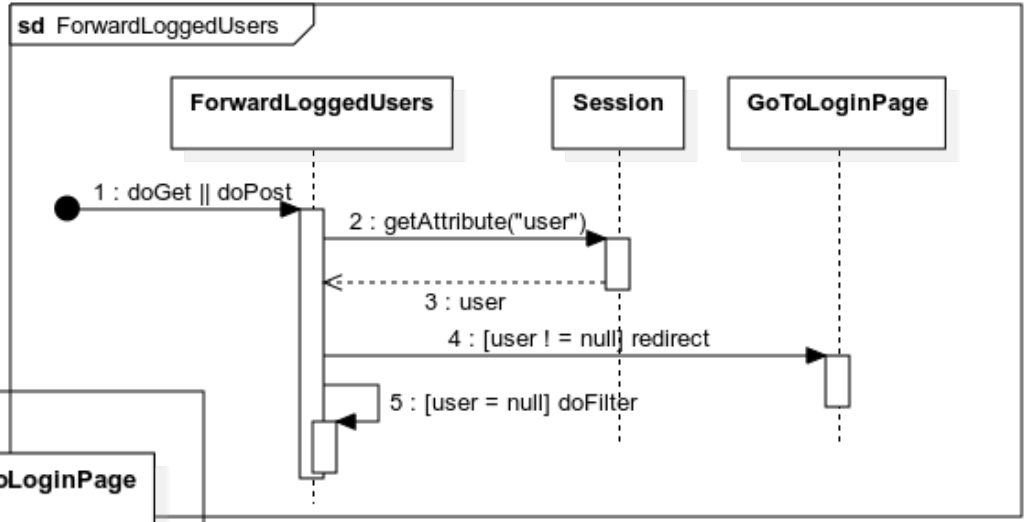
Client side		Server side	
Evento	Azione	Evento	Azione
login → login (registration) form → submit	controllo validità dati	POST (credentials)	controllo validità dati
home → load	aggiorna view con dati	GET ()	estrazione albero cartelle
home → folderTree → click su documento	aggiorna view con dati	GET (documentId)	estrazione dati documento
home → folderTree → click su addButton	mostra form creazione	-	-
home → folderTree → form creazione → submit	controllo validità dati	POST (content)	crea contenuto
home → folderTree → drag&drop content	aggiorna view	POST (documentId, subfodlerId)	sposta documento
home → header →click GoBack	mostra folderTree	-	-
home → header → drop su Bin	mostra modal	-	-
home → modal → click	aggiorna view con dati	POST (contentId, type)	cancella contenuto
home → header → click su Logout	cancella StorageSession	GET ()	cancella Session

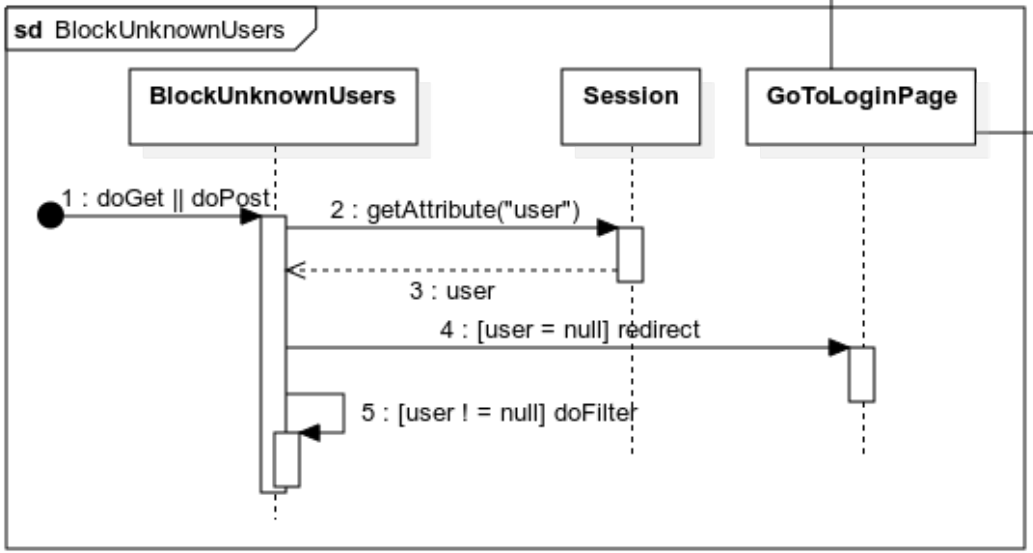
Controller & EventHandler

Client side		Server side	
Evento	Controllore	Evento	Controllore
login → login (registration) form → submit	makeCall	POST (credentials)	CheckLogin (RegisterUser)
home → load	PageOrchestrator.show	GET ()	GetFolderTree
home → folderTree → click su documento	DocumentDetails.show	GET (documentId)	GetDocumentDetails
home → folderTree → form creazione → submit	makeCall	POST (content)	CreateContent
home → folderTree → drag&drop content	makeCall	POST (documentId, subfodlerId)	MoveDocument
home → header →click GoBack	FolderTree.show(false)	-	-
home → header → drop su Bin	Modal.show	-	-
home → modal → click	Header.hadleModalInteraction	POST (contentId, type)	DeleteContent
home → header → click su Logout	makeCall	GET ()	Logout

Filters

ForwardLoggedUsers è un filtro messo alla servlet GoToLoginPage, CheckLogin e RegisterUser per evitare che utenti già loggati si loggino di nuovo o registrino altri utenti e che invece vadano direttamente alla loro home page.





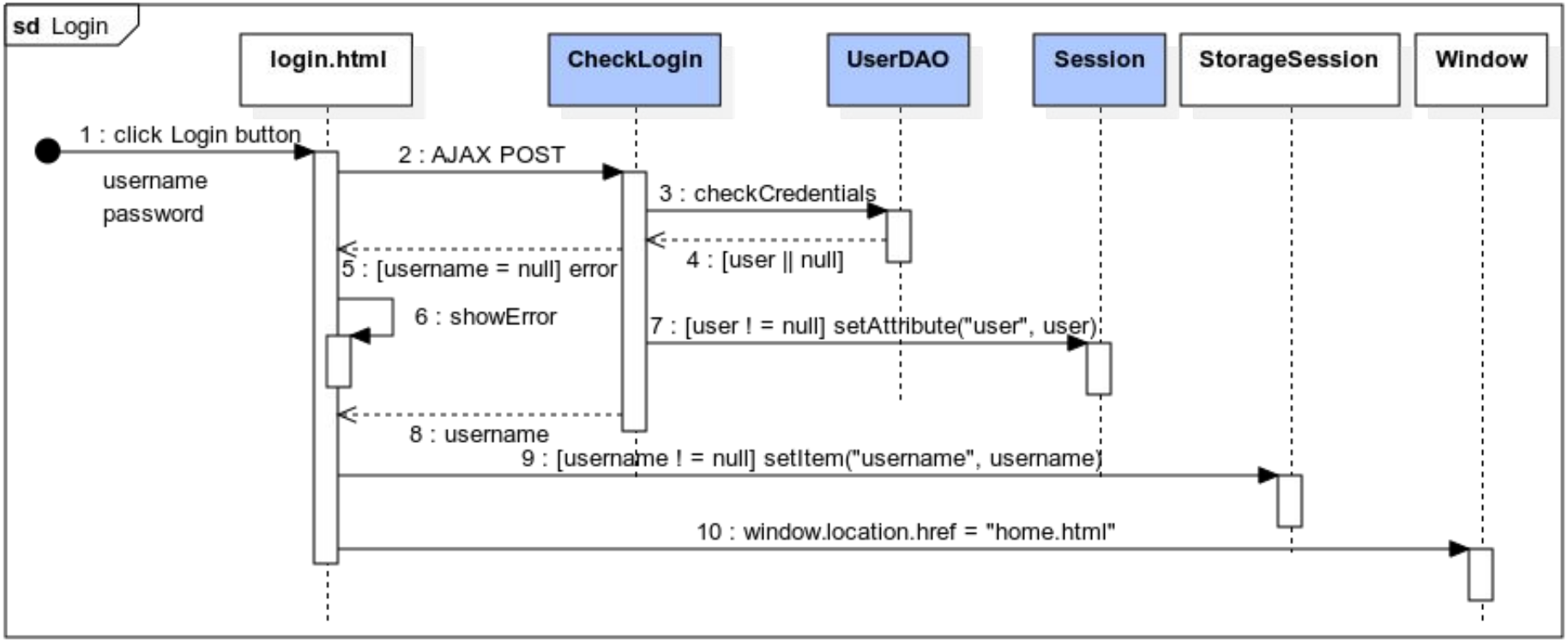
BlockUnknownUsers è un filtro messo a tutte le altre pagine che non riguardano azioni di logging. Impedisce ad utenti non autenticati di interagire con i contenuti.

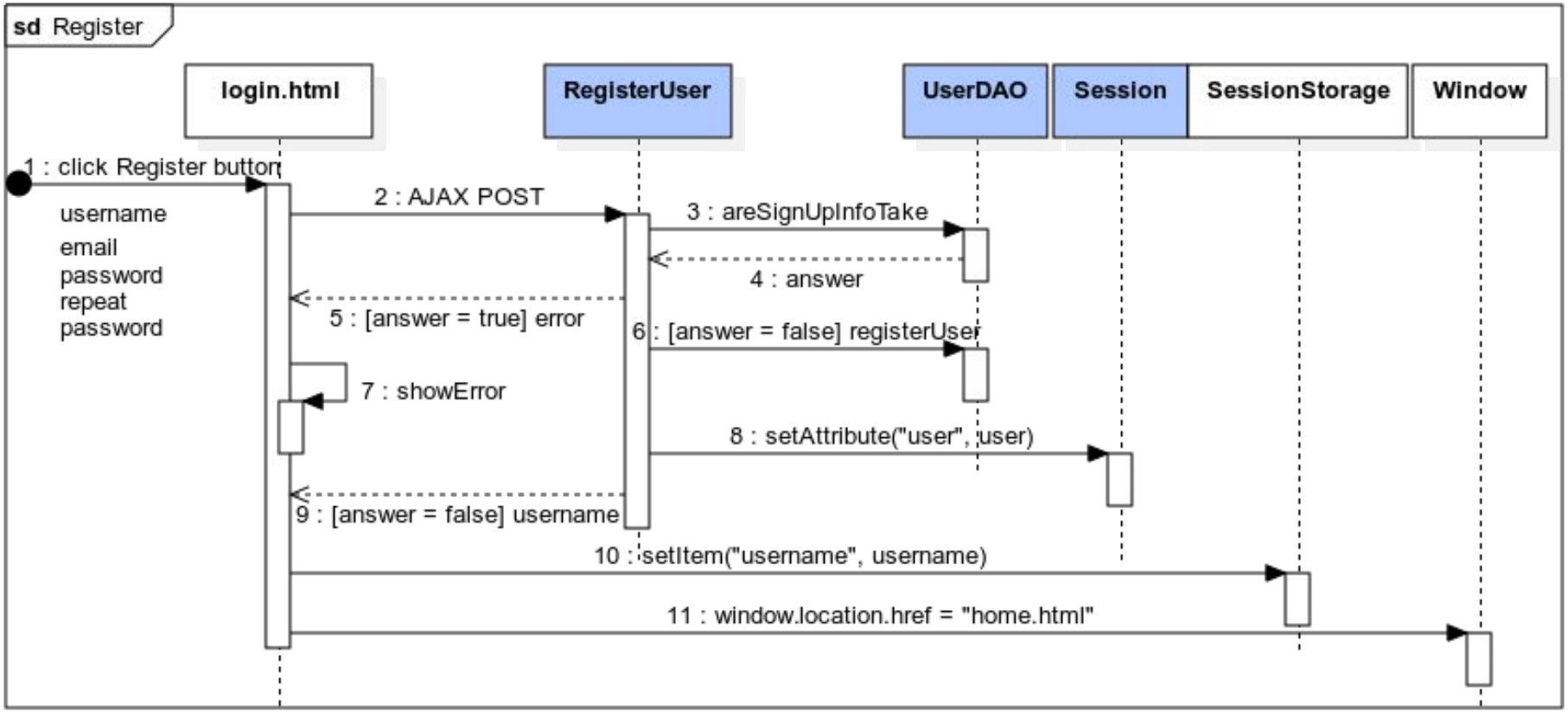
Events

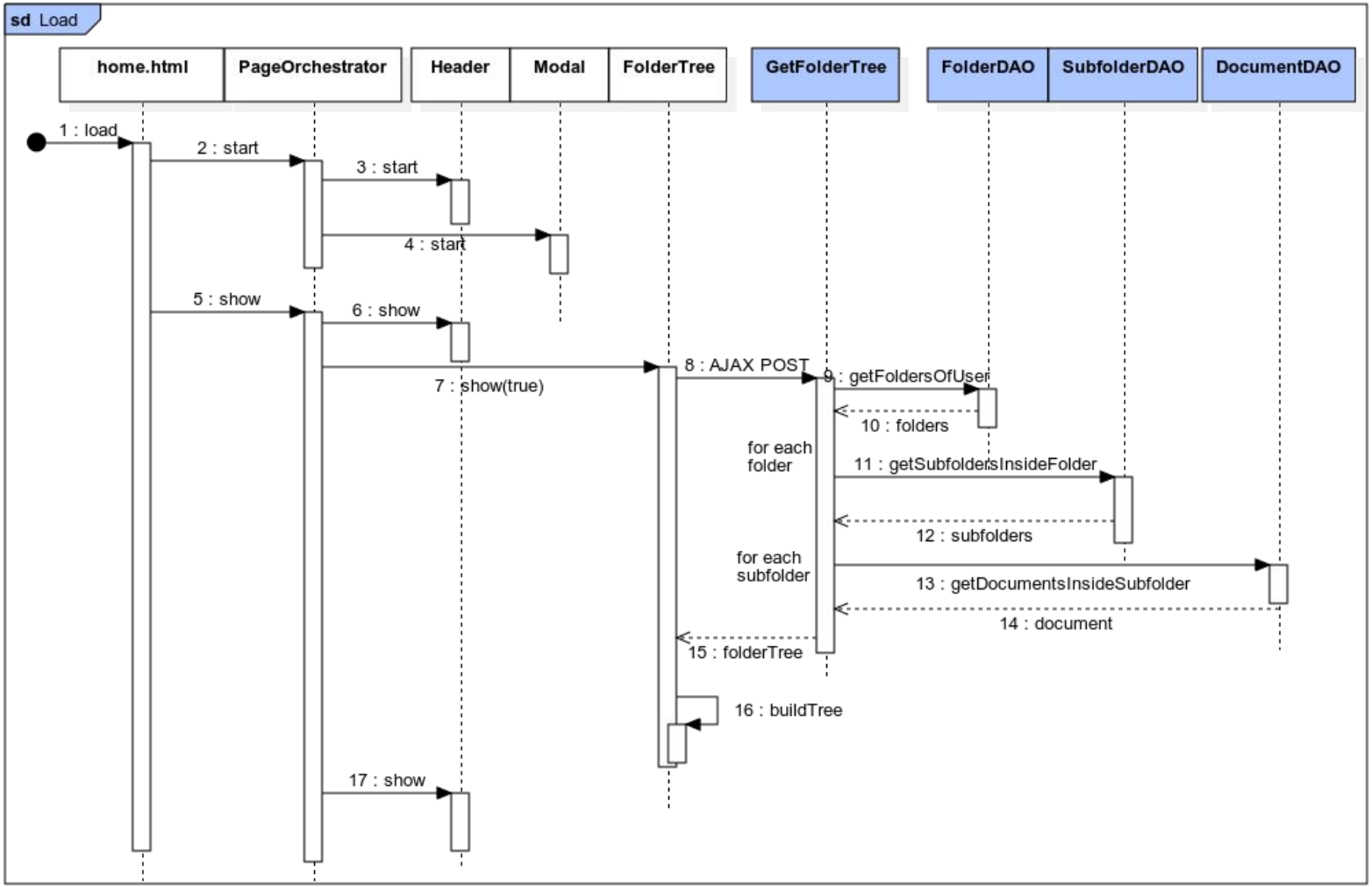
L’evento di move non è illustrati perchè molto simile a quello di delete.

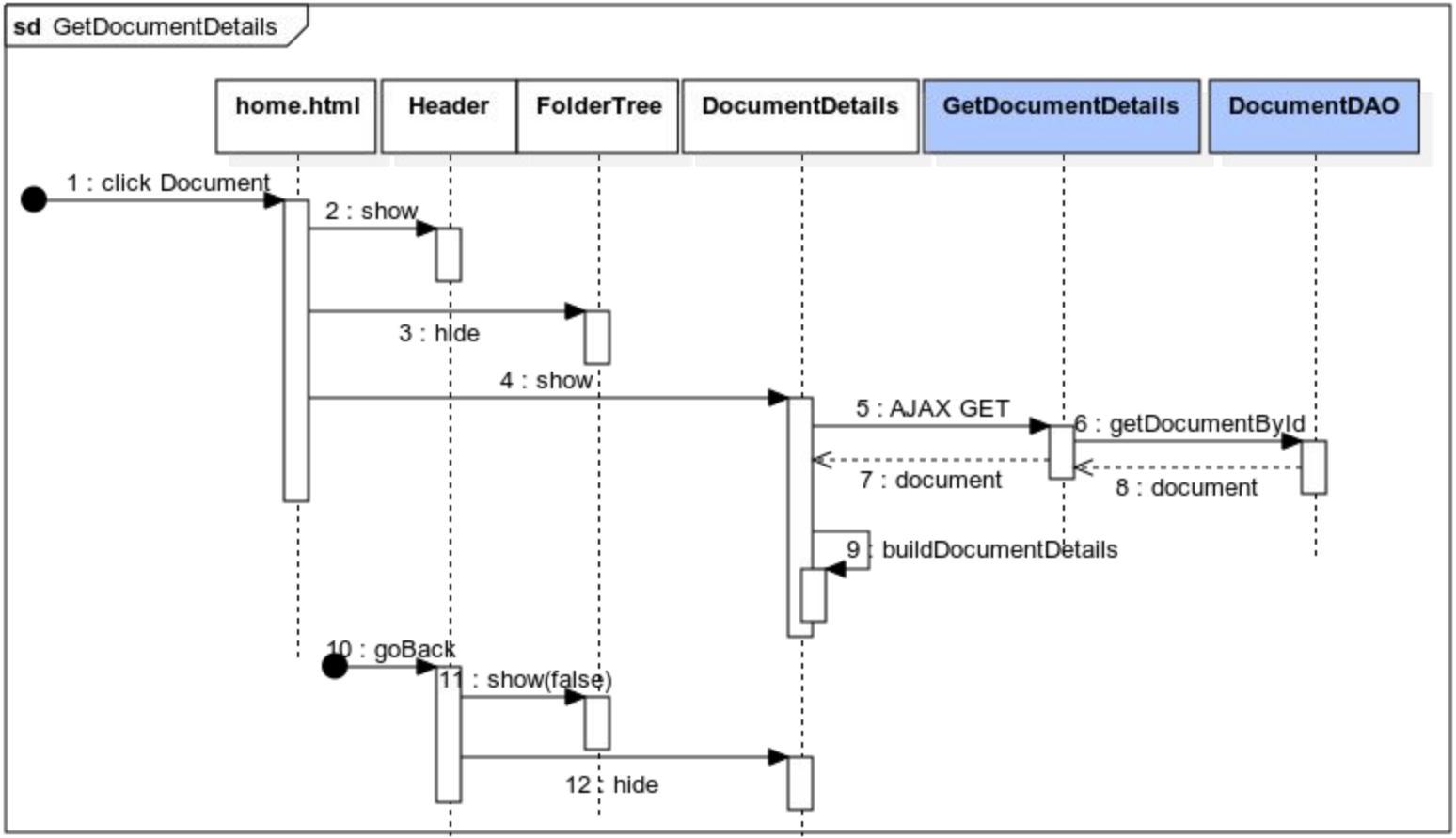
N.B. Ad ogni evento viene controllata la non nullità dei parametri, la loro correttezza sintattica e logica (contenuto di un altro utente ad esempio) ed altri controlli lato server, oltre ai controlli a lato client. Tutti questi risultano ad una risposta con il corrispettivo ErrorCode e ErrorMessage. Questi controlli e le rispettive risposte non vengono riportati nei Sequence Diagram per evitare ripetizioni e contenere le dimensioni degli stessi. Ogni errore è sempre mostrato con un alert.

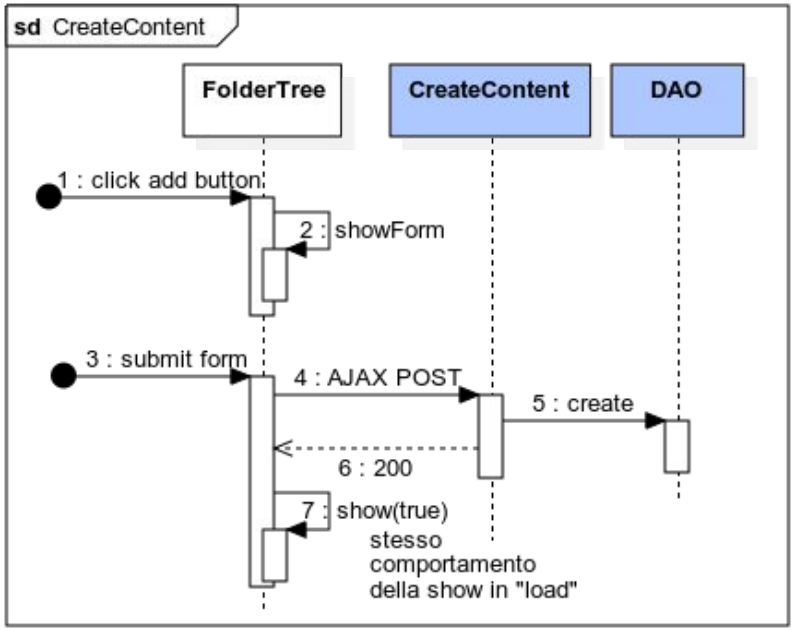
N.B. Le entità segnate in blu sono sul server, mentre le altre sono sul client.

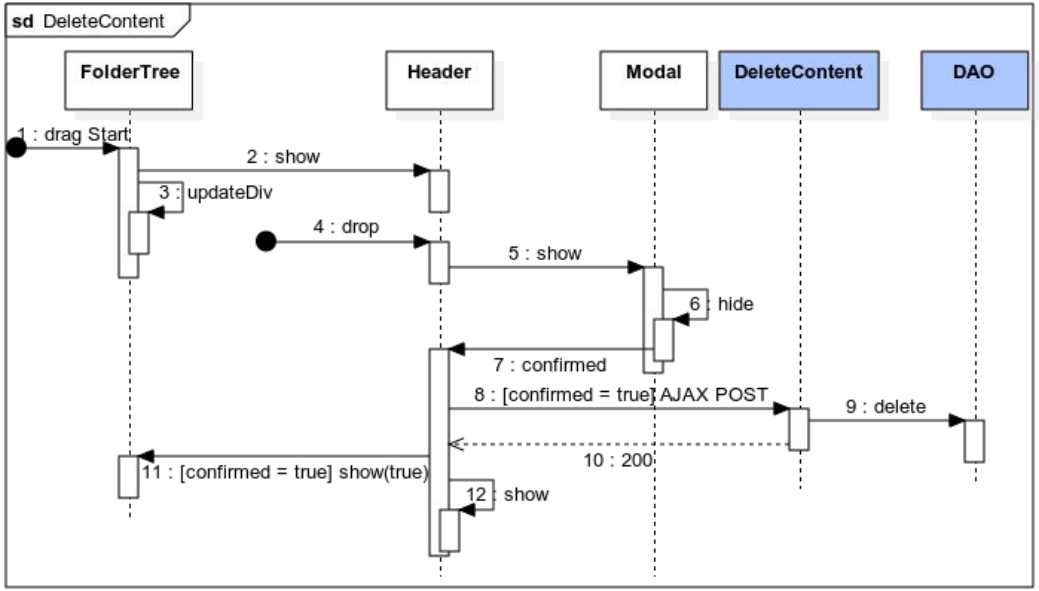

sd Login
login.html
CheckLogin
UserDAO
Session
StorageSession
Window
1 : click Login button
username
password
2 : AJAX POST
3 : checkCredentials
4 : [user || null]
5 : [username = null] error
6 : showError
7 : [user ! = null] setAttribute("user", user)
8 : username
9 : [username ! = null] setItem("username", username)
10 : window.location.href = "home.html"

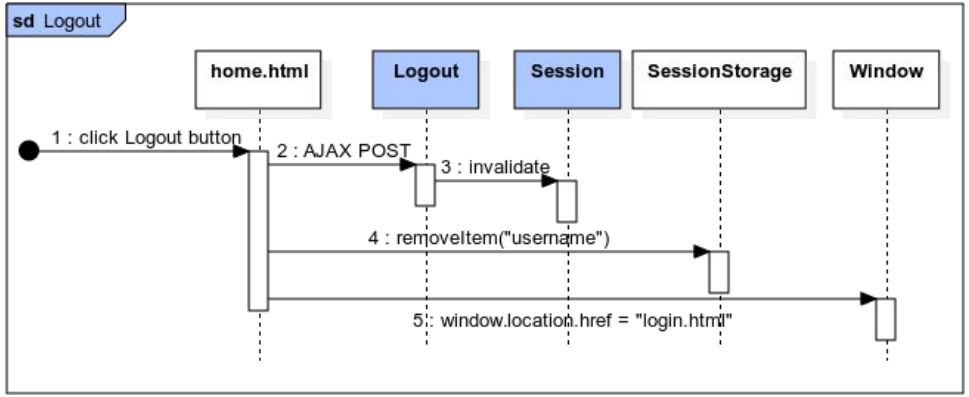

sd Register
login.html
RegisterUser
UserDAO
Session
SessionStorage
Window
1 : click Register button
username
email
password
repeat
password
2 : AJAX POST
3 : areSignUpInfoTake
4 : answer
5 : [answer = true] error
6 : [answer = false] registerUser
7 : showError
8 : setAttribute("user", user)
9 : [answer = false] username
10 : setItem("username", username)
11 : window.location.href = "home.html"


sd Load
home.html
PageOrchestrator
Header
Modal
FolderTree
GetFolderTree
FolderDAO
SubfolderDAO
DocumentDAO
1 : load
2 : start
3 : start
4 : start
5 : show
6 : show
7 : show(true)
8 : AJAX POST
9 : getFoldersOfUser
10 : folders
for each folder
11 : getSubfoldersInsideFolder
12 : subfolders
for each subfolder
13 : getDocumentsInsideSubfolder
14 : document
15 : folderTree
16 : buildTree
17 : show


sd GetDocumentDetails
home.html
Header
FolderTree
DocumentDetails
GetDocumentDetails
DocumentDAO
1 : click Document
2 : show
3 : hide
4 : show
5 : AJAX GET
6 : getDocumentById
7 : document
8 : document
9 : buildDocumentDetails
10 : goBack
11 : show(false)
12 : hide


sd CreateContent
FolderTree
CreateContent
DAO
1 : click add button
2 : showForm
3 : submit form
4 : AJAX POST
5 : create
6 : 200
7 : show(true)
stesso
comportamento
della show in "load"


sd DeleteContent
FolderTree
Header
Modal
DeleteContent
DAO
1 : drag Start
2 : show
3 : updateDiv
4 : drop
5 : show
6 : hide
7 : confirmed
8 : [confirmed = true] AJAX POST
9 : delete
10 : 200
11 : [confirmed = true] show(true)
12 : show


sd Logout
home.html
Logout
Session
SessionStorage
Window
1 : click Logout button
2 : AJAX POST
3 : invalidate
4 : removeItem("username")
5 : window.location.href = "login.html"