Esercizio 3 - RIA

Luca Maestri - Virginia Longo

Esercizio 3: gestione documenti (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.

Entità, **attributi**, **relazioni**

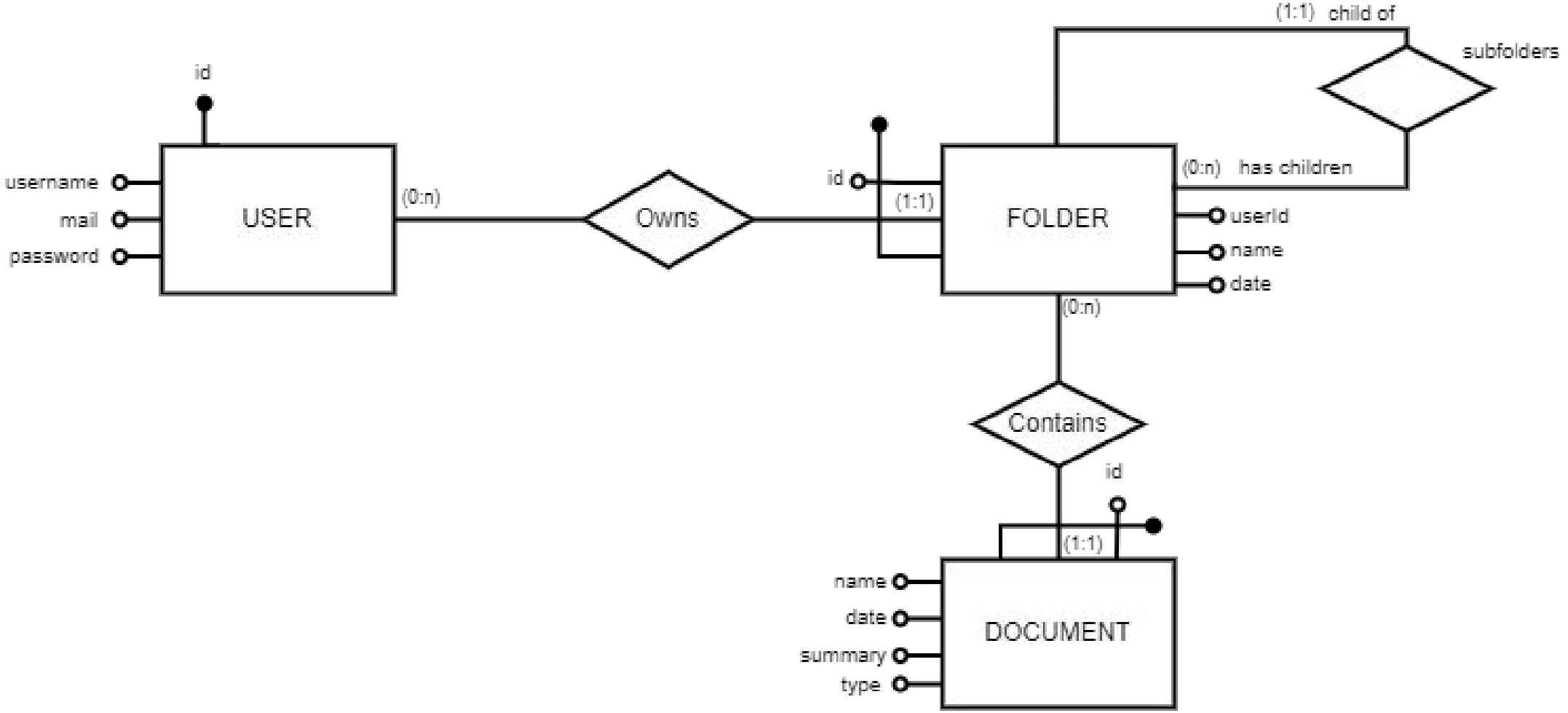
Esercizio 3: gestione documenti (RIA)

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

- 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.

Entità, **attributi**, **relazioni**

Database Design



Database Schema (1/2)

```
CREATE TABLE `user` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `pwd` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB
```

```
CREATE TABLE `folder` (
  `id` int NOT NULL AUTO_INCREMENT,
  `userId` int DEFAULT NULL,
  `name` varchar(45) DEFAULT NULL,
  `date` date DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `userId_idx` (`userId`),
  CONSTRAINT `userId` FOREIGN KEY (`userId`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB
```

Database Schema (2/2)

```
CREATE TABLE `subfolder` (
  `idsubfolder` int NOT NULL,
  `idFather` int DEFAULT NULL,
  PRIMARY KEY (`idsubfolder`),
  KEY `idFather_idx` (`idFather`),
  CONSTRAINT `idFather` FOREIGN KEY (`idFather`) REFERENCES `folder` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `idSubFolder` FOREIGN KEY (`idsubfolder`) REFERENCES `folder` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB
```

```
CREATE TABLE `document` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `summary` varchar(160) DEFAULT NULL,
  `type` varchar(45) DEFAULT NULL,
  `folderId` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `folderId_idx` (`folderId`),
  CONSTRAINT `folderId` FOREIGN KEY (`folderId`) REFERENCES `folder` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB
```

Analisi Requisiti Applicazione

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

- 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.

Pagine (viste), **componenti vista**, **eventi** , **azioni**.

Completamento delle Specifiche

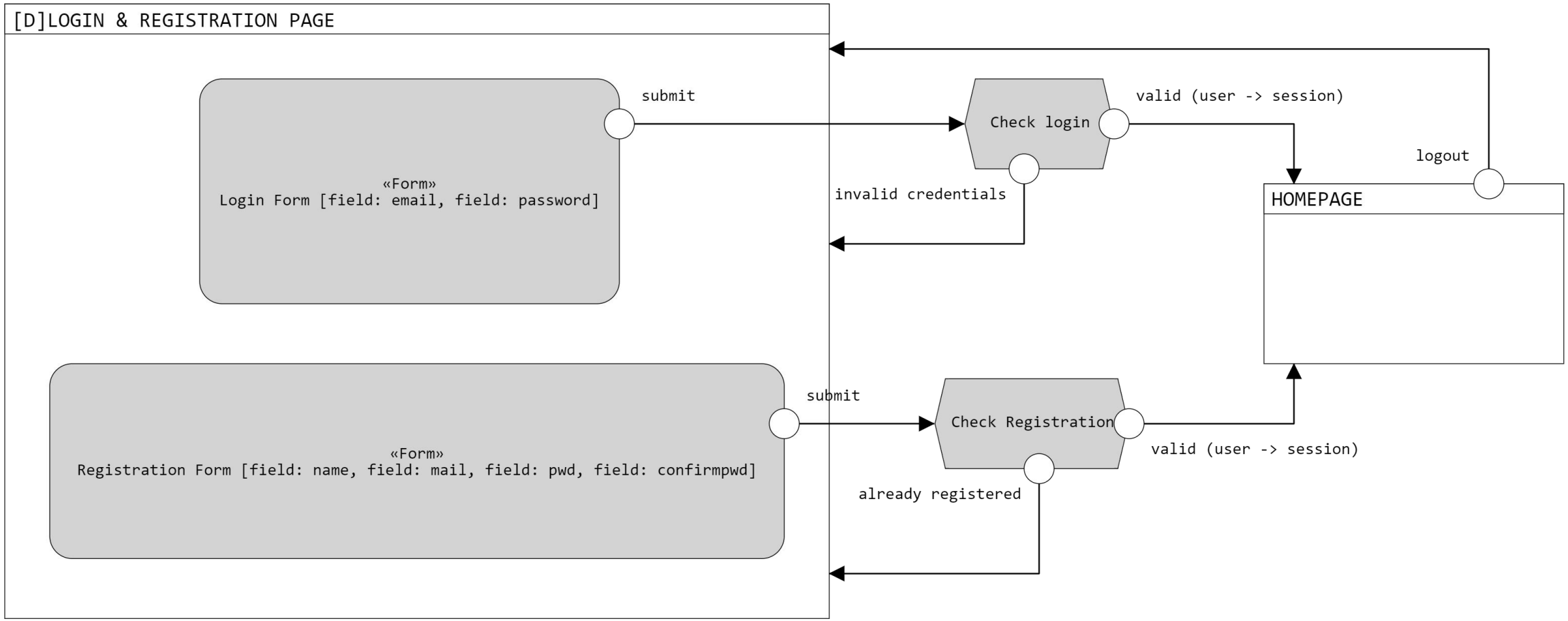
- Il controllo di unicità è fatto sia sullo username che sulla mail inserite dall'utente al momento della registrazione.
- Al caricamento della Homepage, l'utente visualizza unicamente le cartelle presenti nel suo Drive, con tutte le restanti finestre collassate.
- L'atto di trascinamento di un documento fa apparire le sottocartelle in cui è possibile spostarlo e il cestino. La sottocartella di provenienza non compare tra queste.
- Le informazioni del documento e delle cartelle sono visibili direttamente all'interno delle loro card.
- Possono essere eliminati documenti e cartelle. L'eliminazione di una cartella produce un effetto a cascata sulle sottocartelle e i documenti in essa contenuti.
- Non si possono avere cartelle, sottocartelle o documenti omonimi.

Sommario

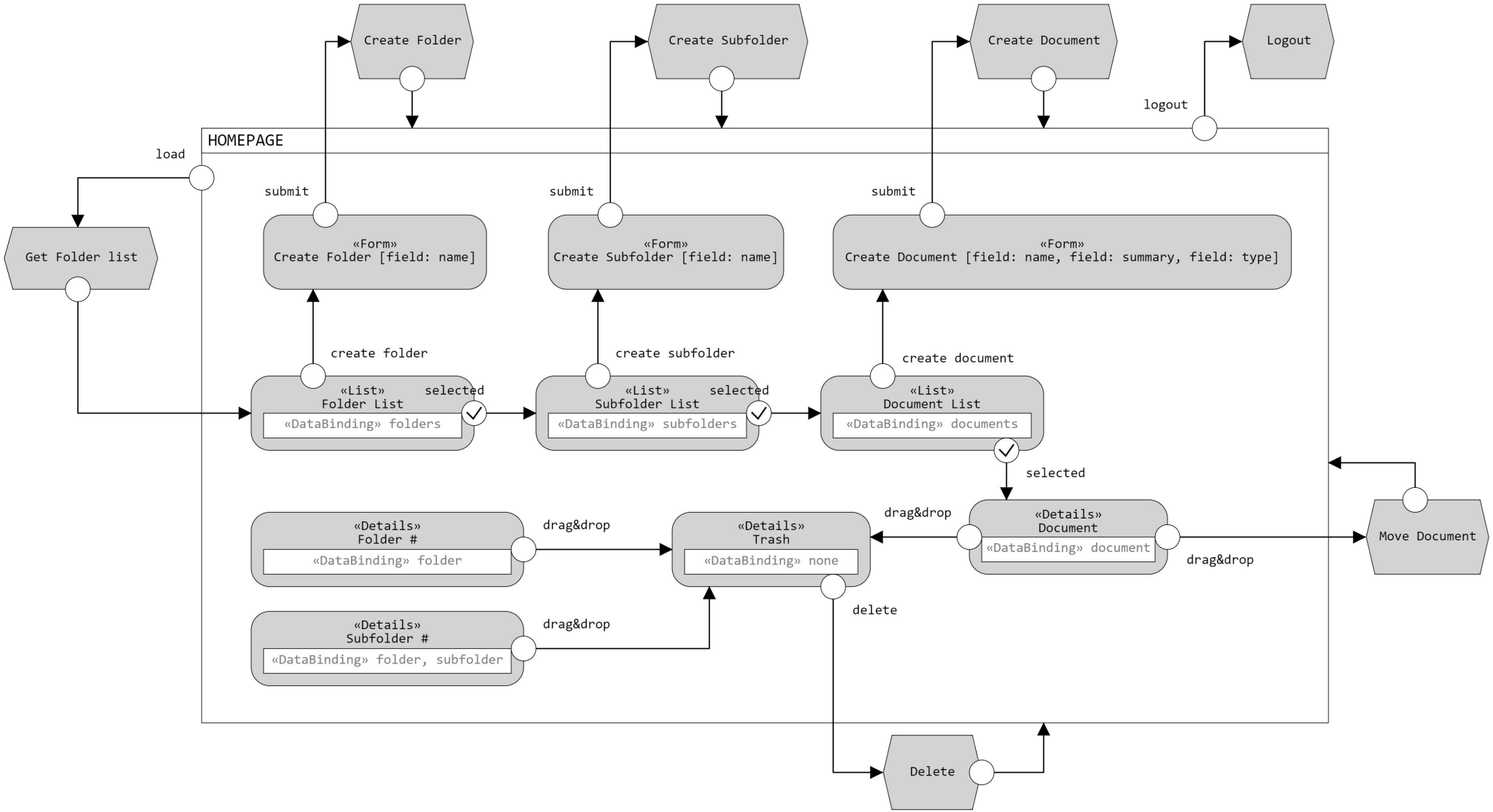
- **Viste e Componenti**
  - **Pagina Home**
    - Lista Folders (+ dettagli di ogni Folder)
    - Bottone “Create Folder”
      - Form Create Folder
    - Bottone Open (Folder)
    - Lista Subfolders (+dettagli di ogni Subfolder)
    - Bottone “Create Subfolder”
      - Form Create Subfolder
    - Bottone Open (Subfolder)
    - Lista Documenti (+dettagli di ogni Documento)
    - Bottone “Create Document”
      - Form Create Document
    - Bottone Open (document)
      - Dettagli documento
    - Cestino
    - Header
      - Bottone Logout

- **Eventi & azioni**
  - Login
    - Check login
  - Click su “Create Folder” (invio form)
    - Creazione folder
  - Click su “Create Subfolder” (invio form)
    - Creazione subfolder
  - Click su “Create Document” (invio form)
    - Creazione document
  - Drag & Drop nel cestino
    - Cancella l'elemento
  - Drag & Drop in un'altro subfolder
    - Sposta documento
    - Aggiorna stato documento e subfolder
  - Logout
    - logout

Design Applicativo (IFML)



Design Applicativo (IFML)



Events & Actions

Client side		Server side	
Evento	**Azione**	**Evento**	**Azione**
login.html->login form->submit	Data check	POST(email,password)	Data check
login.html->register form->submit	Data check	POST(credentials)	Data check
home.html->load	Update view->user info & folder list	GET no param	Retrieves data from db and converts it to Json
home.html->folders list-> folder selection (click open)	Update view->displays subfolders list	GET(folderId)	Retrieves data from db and converts it to Json
home.html->hide/show buttons	Shows/Hides the corresponding forms	-	-
home.html-> create folder-> form	Data check	POST(folder name)	Updates folder list and db
home.html->subfolders list-> subfolder selection (open)	Update view->displays documents list	GET(idsubfolder)	Retrieves data from db and converts it to Json
home.html->create subfolder->form	Data check	POST(subfolder name)	Updates subfolder list and db

Events & Actions

Client side		Server side	
Evento	**Azione**	**Evento**	**Azione**
home.html->create document-> form	Data check	POST(name, summary, type)	Uodates document list and db
home.html->document-> drag&drop on subfolder	Process & updates view	POST(idsubfolder)	Updates document and folder status
home.html->document-> drag&drop on trash	Process & updates view	POST(documentId)	Deletes document and updates db
home.html->subfolder-> drag&drop on trash	Process & updates view	POST(folderId)	Deletes subfolder and content and updates db
home.html->folder-> drag&drop on trash	Process & update view	POST(folderId)	Deletes folder and its content and updates db
home.html->logout	-	GET no param	Terminates session

Controller & Event Handlers

Client side		Server side	
Evento	**Controller**	**Evento**	**Controller**
login -> login form ->submit	loginManagement -> makeCall	POST(email password)	Login (servlet)
login->register form->submit	loginManagement ->makeCall	POST(username, email, pwd)	Register (servlet)
homepage -> load	function PageOrchestrator ->UserInfo ->FolderList	GET no param	GetFolders (servlet)
homepage->folders list-> select folder	function FolderList ->SubfolderList	GET (folderid)	GetSubFolders (servlet)
homepage->create folder form-> submit	function FolderList	POST(folder name)	CreateFolder (servlet)
homepage->subfolders list-> select subfolder	function SubfolderList ->DocumentList	GET(idsubfolder)	GetDocuments (servlet)
homepage->create subfolder form-> submit	function SubfolderList	POST(subfolder name)	CreateSubFolder (servlet)

Controller & Event Handlers

Client side		Server side	
Evento	**Controller**	**Evento**	**Controller**
homepage->create document form-> submit	function DocumentList	POST(name,summary,type)	CreateDocument (servlet)
homepage->documents list-> move document	function DocumentList -> SubFolderList	POST(folderId,documentId)	MoveDocument (servlet)
homepage->document list-> delete document	function drop_document	POST(documentId)	DeleteDocument (servlet)
homepage-> (sub)folder list -> delete (sub)folder	function drop_folder	POST(userId,folderId)	DeleteFolder (servlet)
logout	-	GET no param	Logout (servlet)

Server-Side Components

- **Data Access Objects (DAO)**
  - UserDAO
    - findUser(email, password) : user
    - registerUser(name, email, pwd)
    - getUserByEmail(email) : user
    - getUserById(id) : user
  - FolderDAO
    - findAllFolders() : List<Folder>
    - findAllFoldersById(userId) : List<Folder>
    - findFolderById(id) : Folder
    - findTopFolderAndSubfoldersById(id) : List<Folder>
    - findTopFolderAndSubfolders() : List <Folder>
    - findSubparts(folder)
    - createFolder(userId, name, date)
    - createSubFolder(userId, name, date, idFather)
    - findSubfoldersById(userId) : List<Folder>
    - findSubfoldersByFolderId(folderId): List<Folder>
    - deleteFolder(folderId)
  - DocumentDAO
    - getDocumentsByIdFolder(folderId) : List<Document>
    - getDocumentById(id) : Document
    - createDocument(name, date, summary, type, folderId)
    - updateDocument(id, folderId)
    - deleteDocument(id)
- **Model Objects (Beans)**
  - User
  - Folder
  - Document
- **Filters (Beans)**
  - CheckLoggedUser
  - CheckNotLoggedUser
- **Controllers (Servlets) [access rights]**
  - Login
  - Logout
  - Register
  - GetDocuments
  - GetFolders
  - GetSubFolder
  - GetSubFolders
  - CreateFolder
  - CreateSubFolder
  - CreateDocument
  - MoveDocument
  - DeleteDocument
  - DeleteFolder
- **View (Templates)**
  - login.html
  - home.html

Client-Side Components

Home

- PageOrchestrator
  - **start()**: initializes the components
  - **refresh()**: displays initial content
- UserInfo
  - **show()**
- FolderList
  - **show()**: displays folders
  - **update()**: updates view with the results
- SubfolderList
  - **showCreate()**: shows the Create button
  - **hideCreate()**: hides the Create button
  - **show(accountId)**: displays subfolders
  - **hide()**: hides subfolders
  - **update(folder, subfolders, errmex)**: updates view
- DocumentList
  - **showCreate()**: shows Create button
  - **hideCreate()**: hides Create button
  - **show(accountId)**: displays documents
  - **hide()**: hides documents
  - **update(folder, documents, errmex)**: updates view
- SubFolderList (only shown for drag&drop)
  - **showCreate()**: shows the Create button
  - **hideCreate()**: hides the Create button
  - **show(accountId)**: displays subfolders
  - **hide()**: hides subfolders
  - **update(subfolders, errmex)**: updates view
- handleDragOver
- handleDragLeave
- handleDragStart
- handleDragEnd
- drop_folder(event)
- drop_document(event)

Login (& Registration)

- checks credentials
- sendToServer

Sequence diagrams

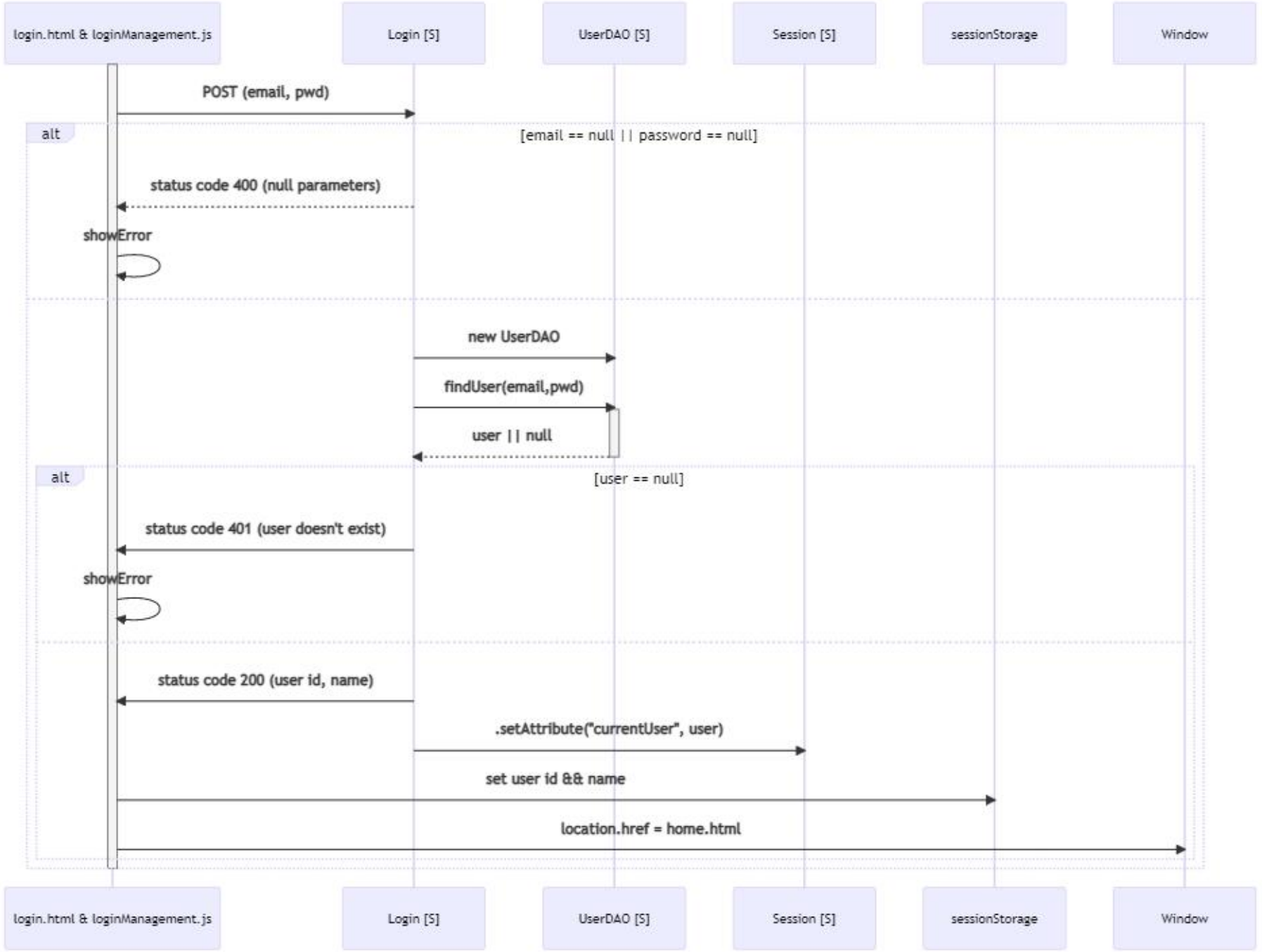
I seguenti diagrammi vogliono descrivere quanto più coerentemente possibile i principali eventi dell'applicazione web realizzata.

Alcuni dettagli sono talvolta omessi per mantenere i diagrammi chiari e puliti.

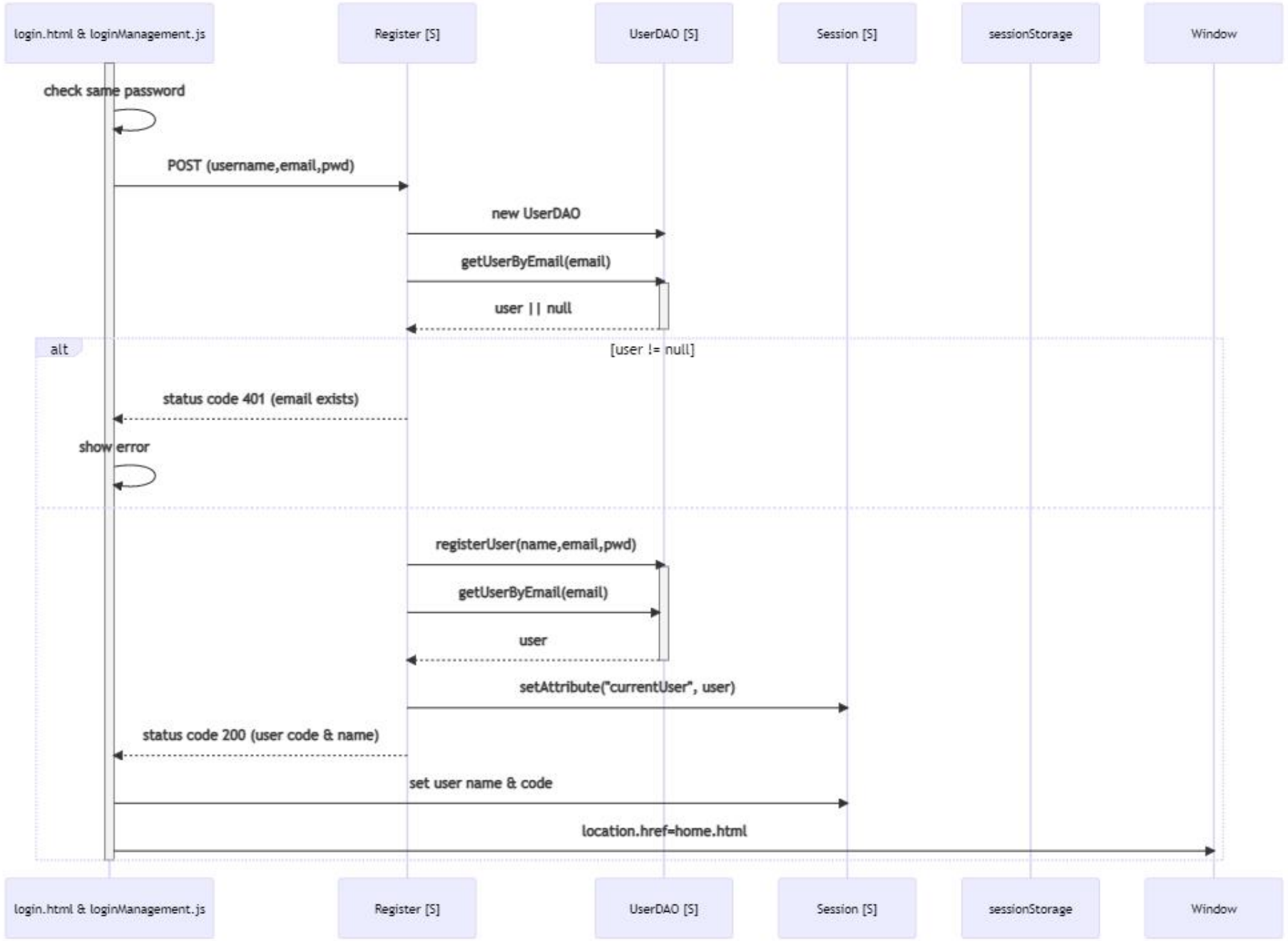
Ogni errore interno del server o di accesso e interazione col database è sottointeso nei diagrammi ma sempre trattato in pratica con la visualizzazione di una pagina o di un messaggio d'errore.

Gli elementi lato Server sono indicati con [S] e tutte le POST e GET rappresentate sono AJAX.

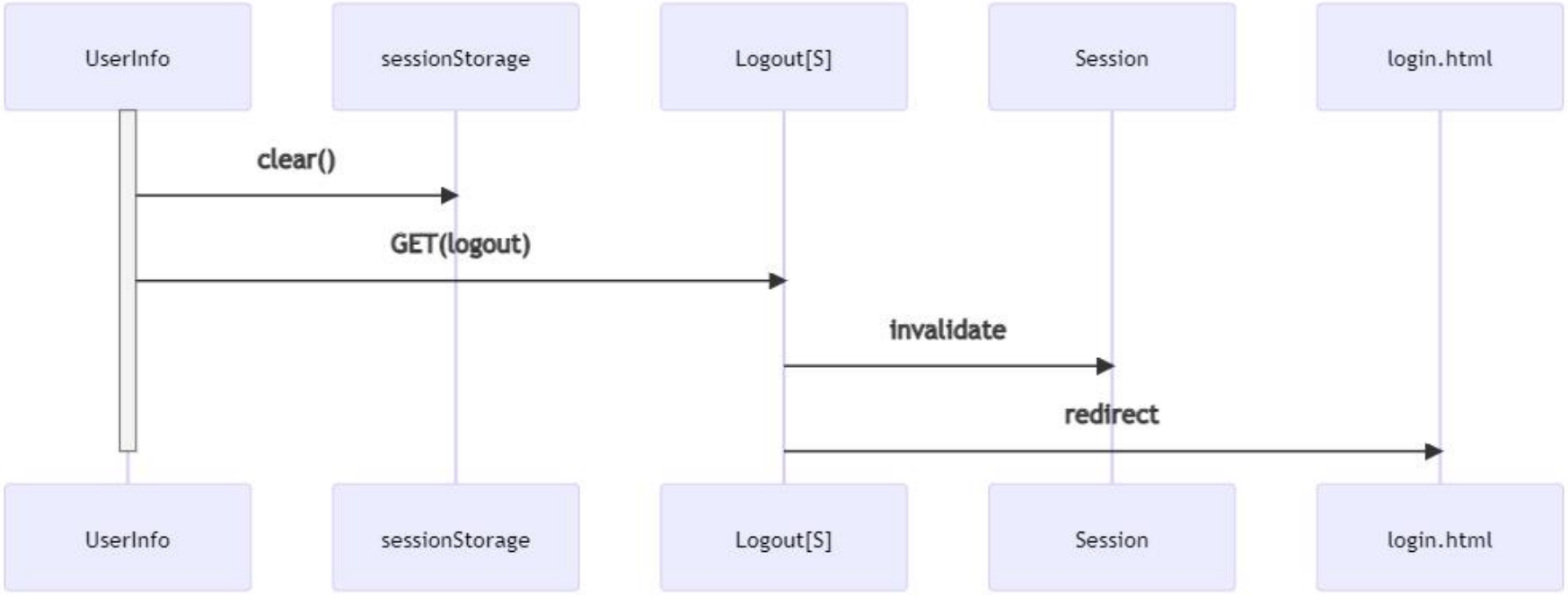
Login


login.html & loginManagement.js
Login [S]
UserDAO [S]
Session [S]
sessionStorage
Window
POST (email, pwd)
alt
[email == null || password == null]
status code 400 (null parameters)
showError
new UserDAO
findUser(email,pwd)
user || null
alt
[user == null]
status code 401 (user doesn't exist)
showError
status code 200 (user id, name)
.setAttribute("currentUser", user)
set user id && name
location.href = home.html
login.html & loginManagement.js
Login [S]
UserDAO [S]
Session [S]
sessionStorage
Window

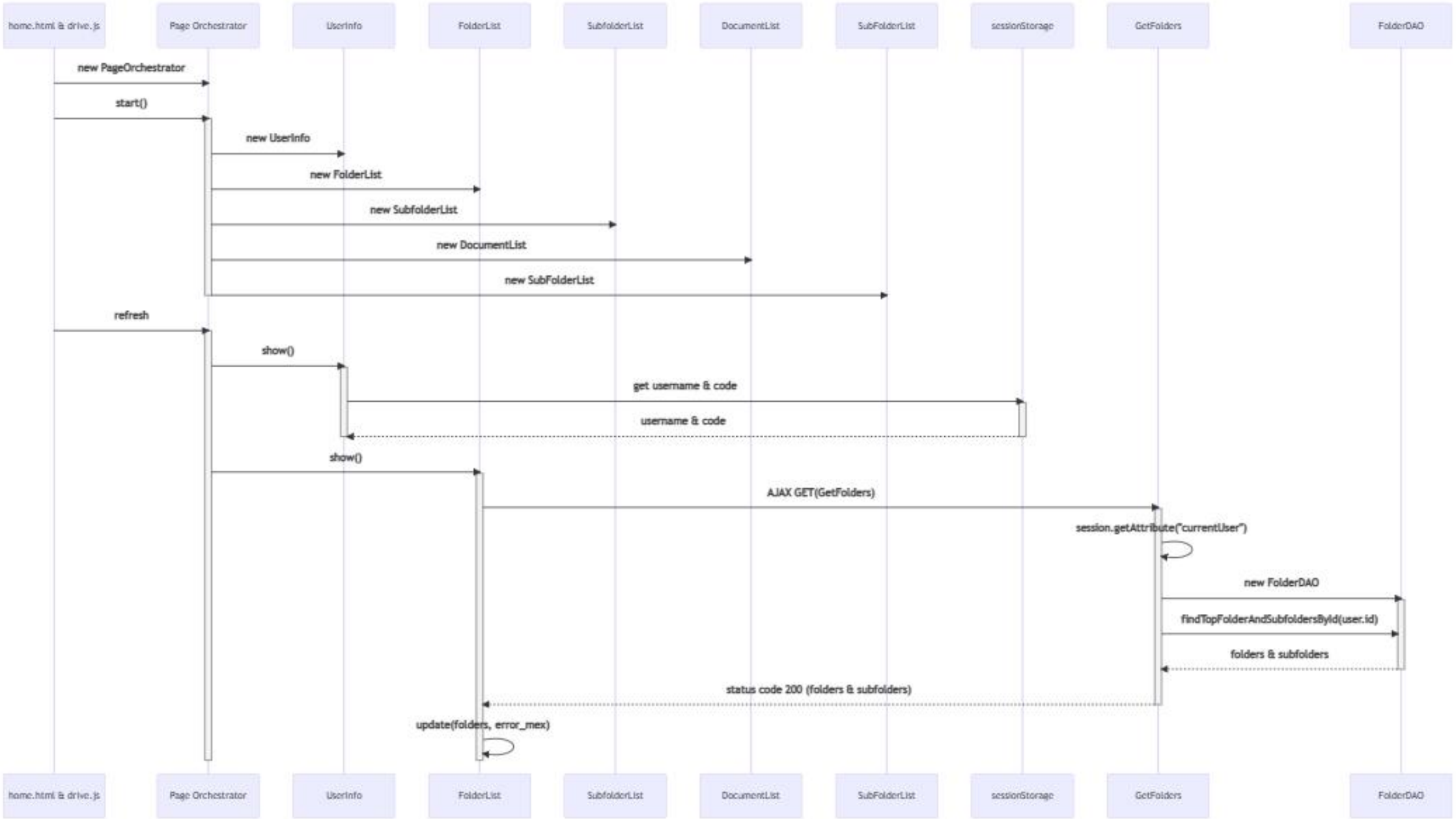
Registration



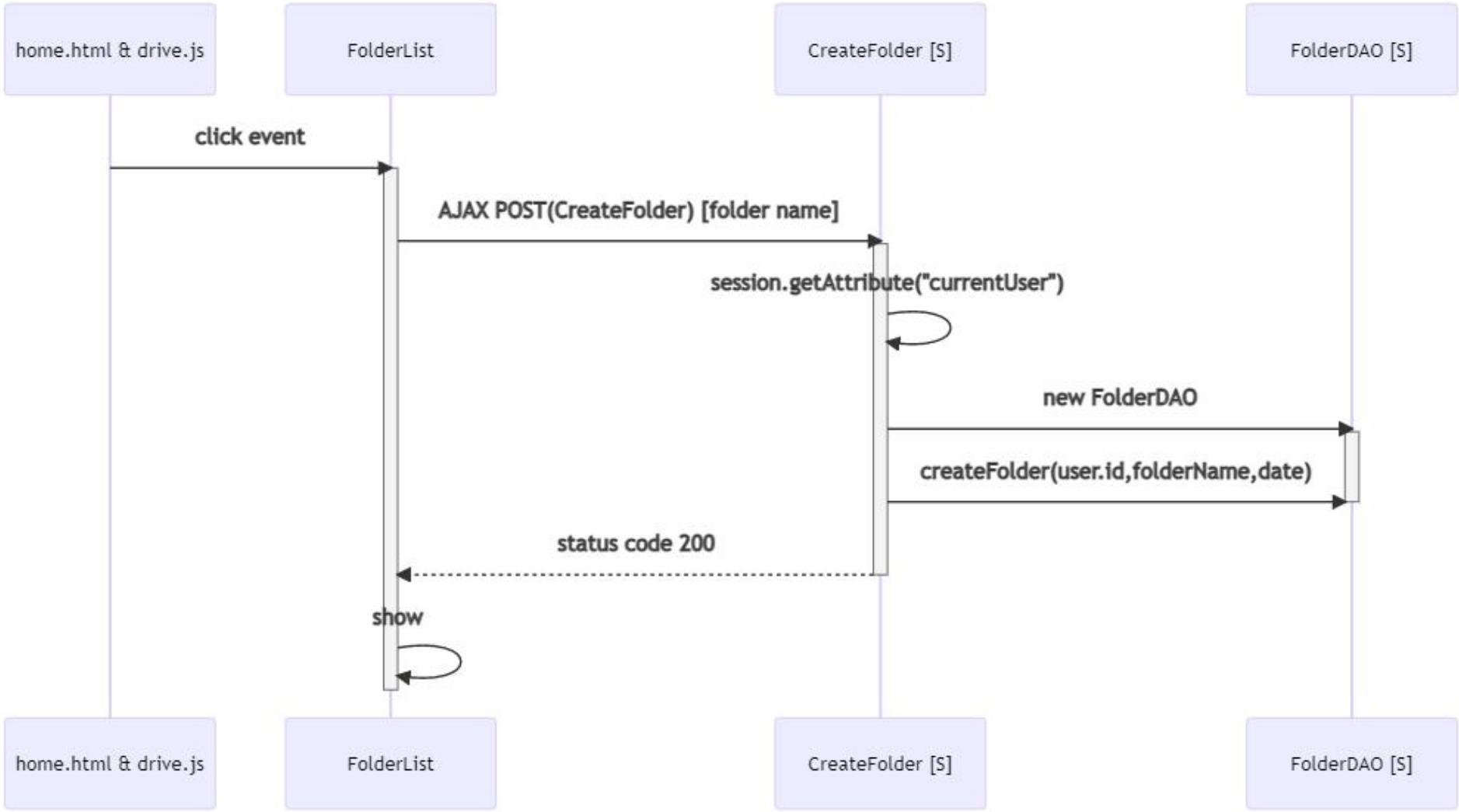
Logout


UserInfo
sessionStorage
Logout[S]
Session
login.html
clear()
GET(logout)
invalidate
redirect
UserInfo
sessionStorage
Logout[S]
Session
login.html

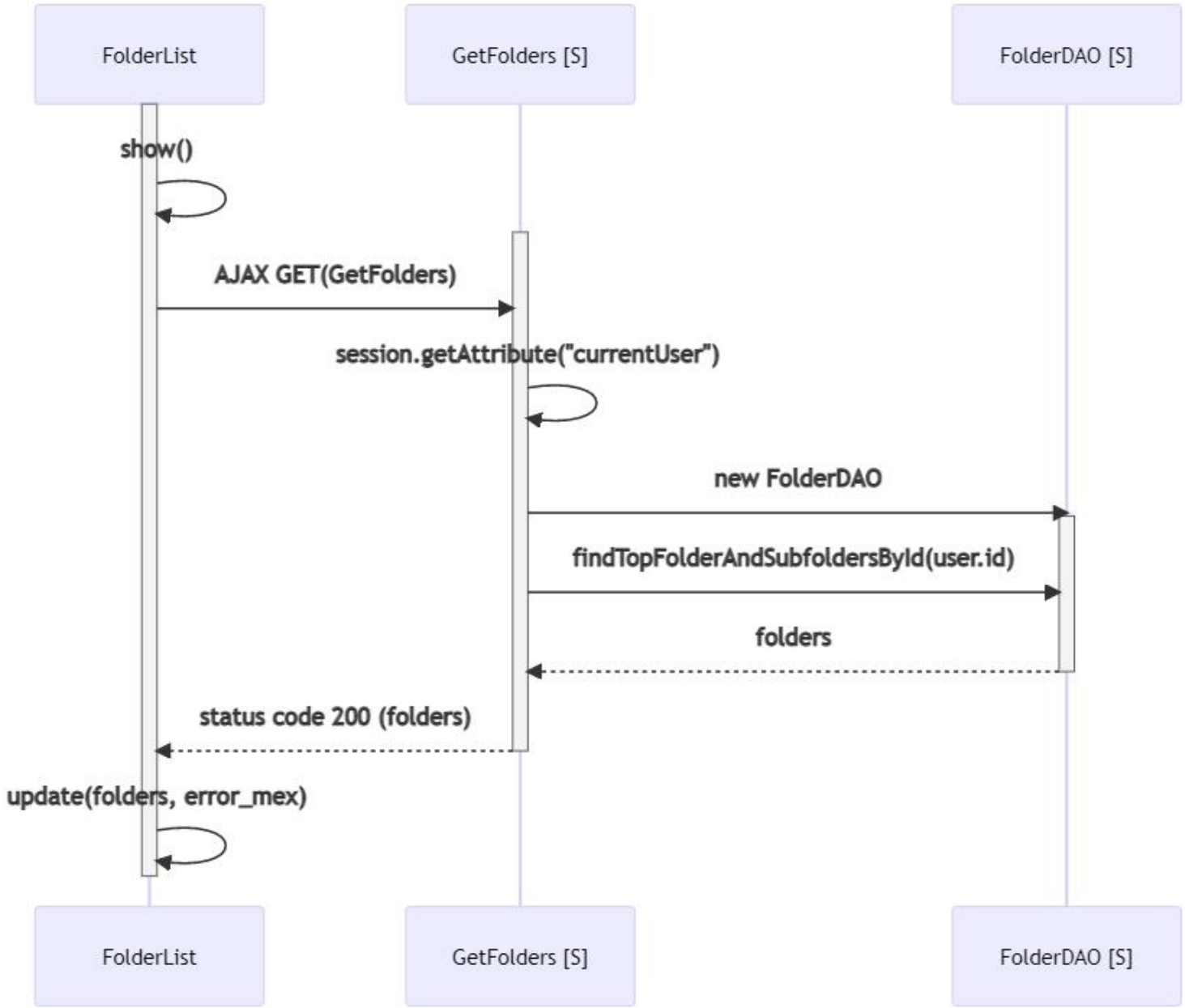
Load


home.html & drive.js
Page Orchestrator
UserInfo
FolderList
SubfolderList
DocumentList
SubFolderList
sessionStorage
GetFolders
FolderDAO
new PageOrchestrator
start()
new UserInfo
new FolderList
new SubfolderList
new DocumentList
new SubFolderList
refresh
show()
get username & code
username & code
show()
AJAX GET(GetFolders)
session.getAttribute("currentUser")
new FolderDAO
findTopFolderAndSubfoldersById(user.id)
folders & subfolders
status code 200 (folders & subfolders)
update(folders, error_mex)

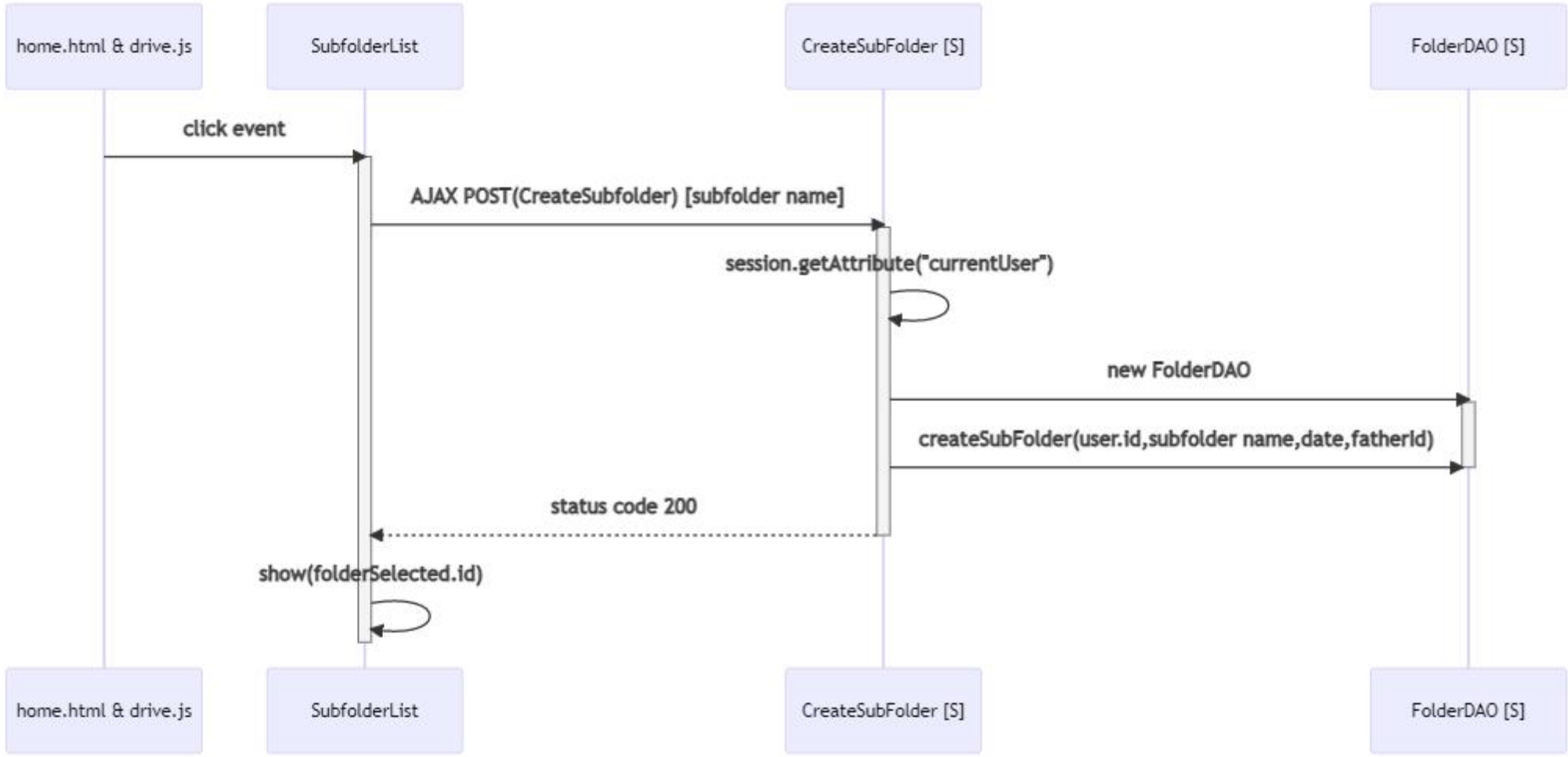
Create Folder



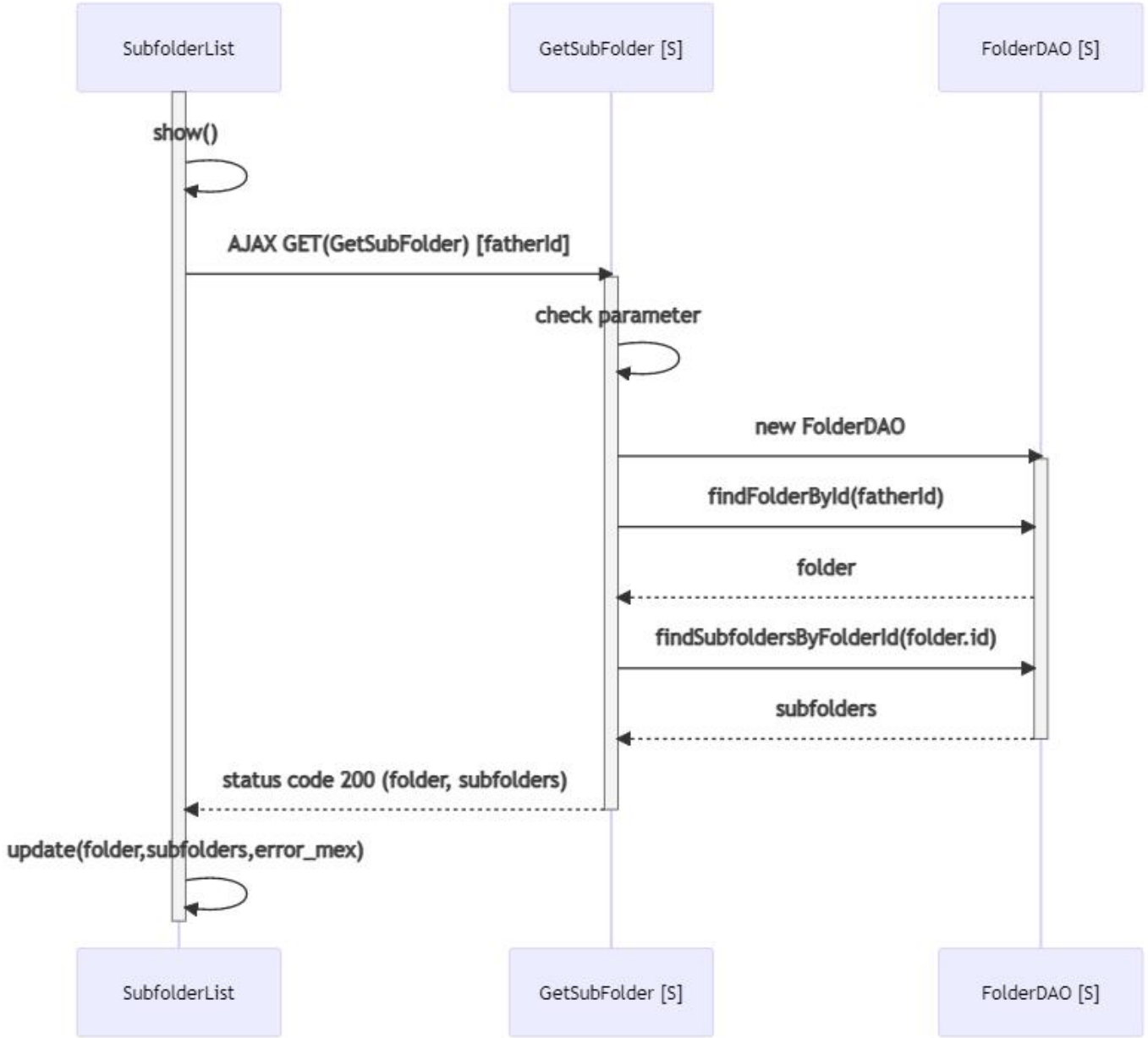
FolderList show



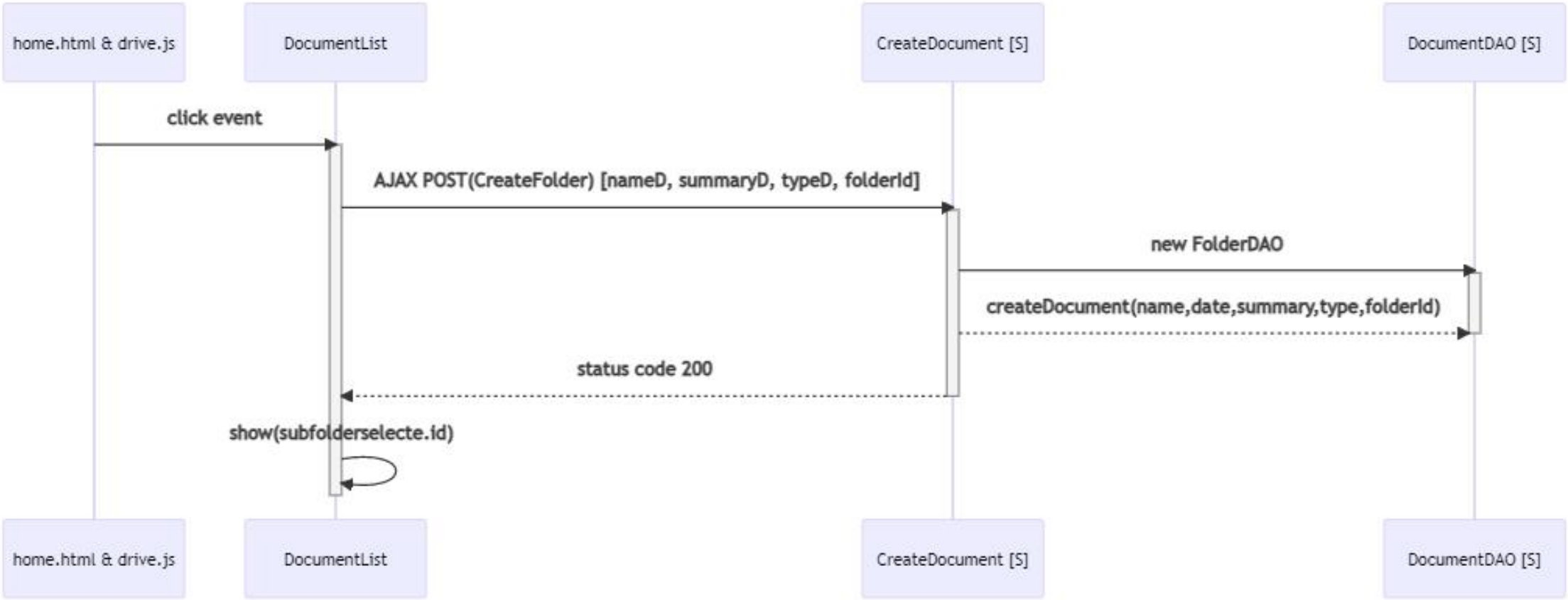
Create Subfolder



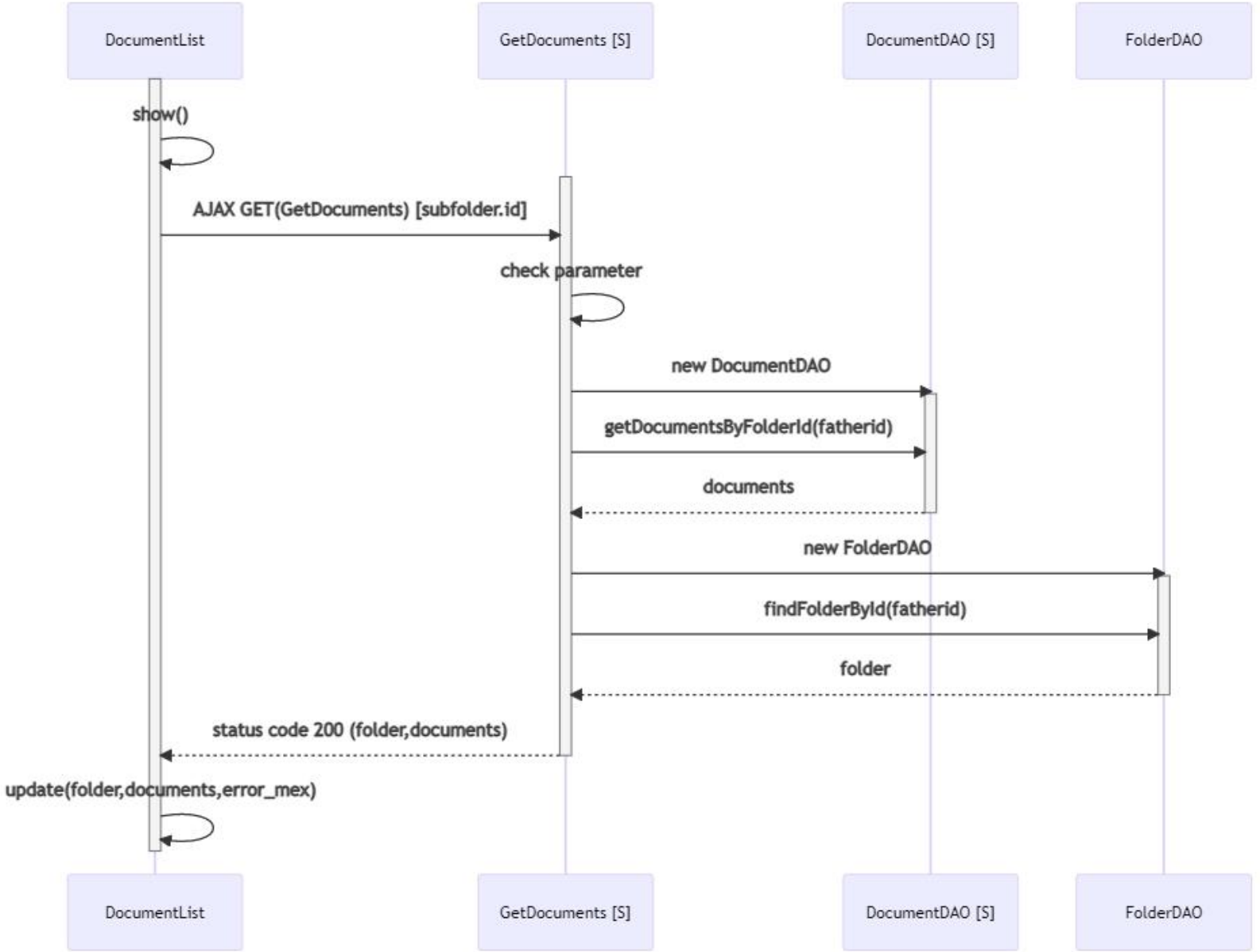
SubfolderList show



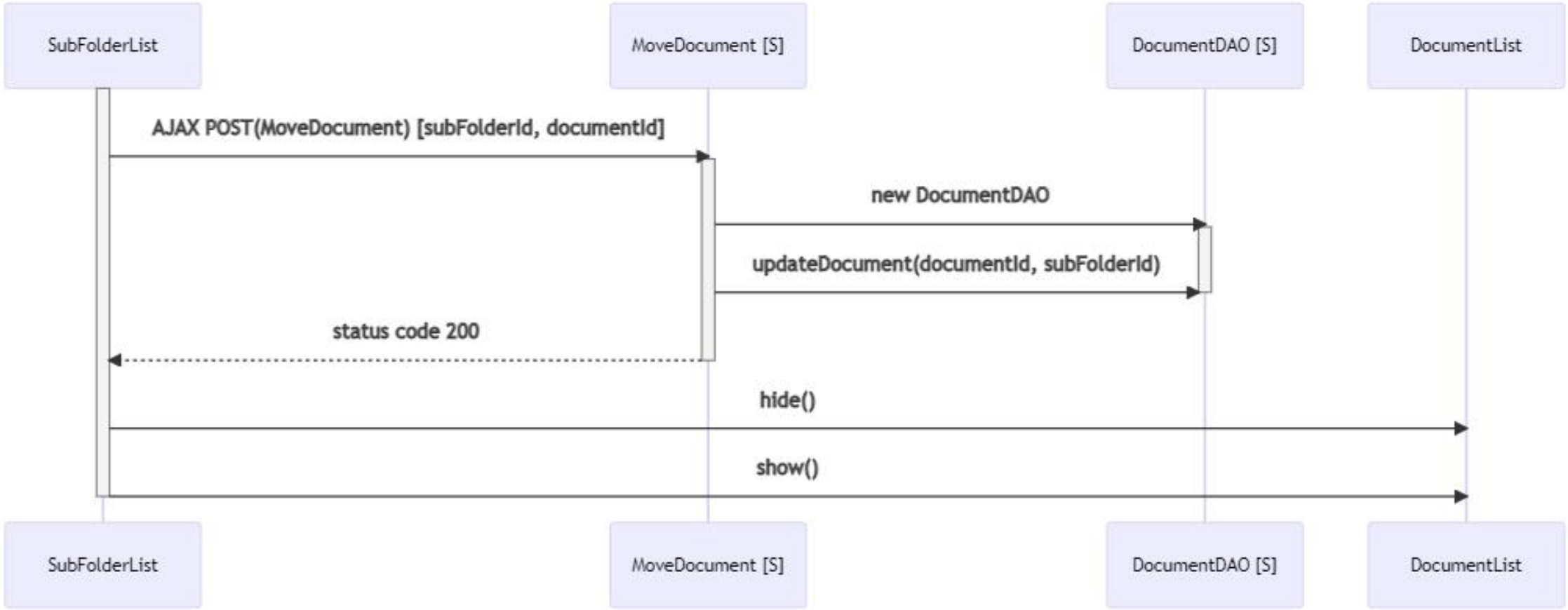
Create Document



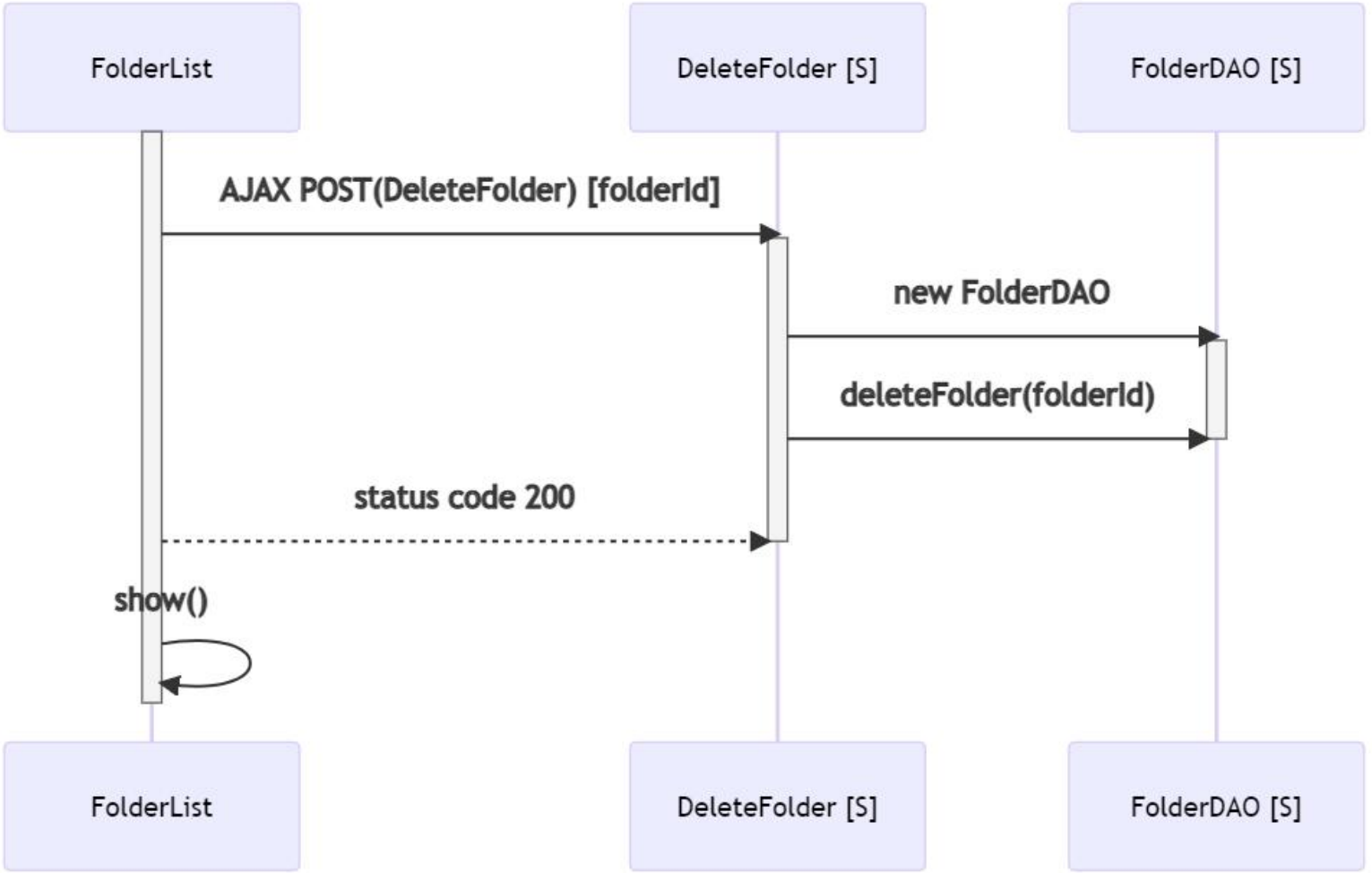
DocumentList show



Move Document



Delete Folder



Delete Document