PROGETTO TECNOLOGIE INFORMATICHE PER IL WEB

Galleria di Immagini – RIA

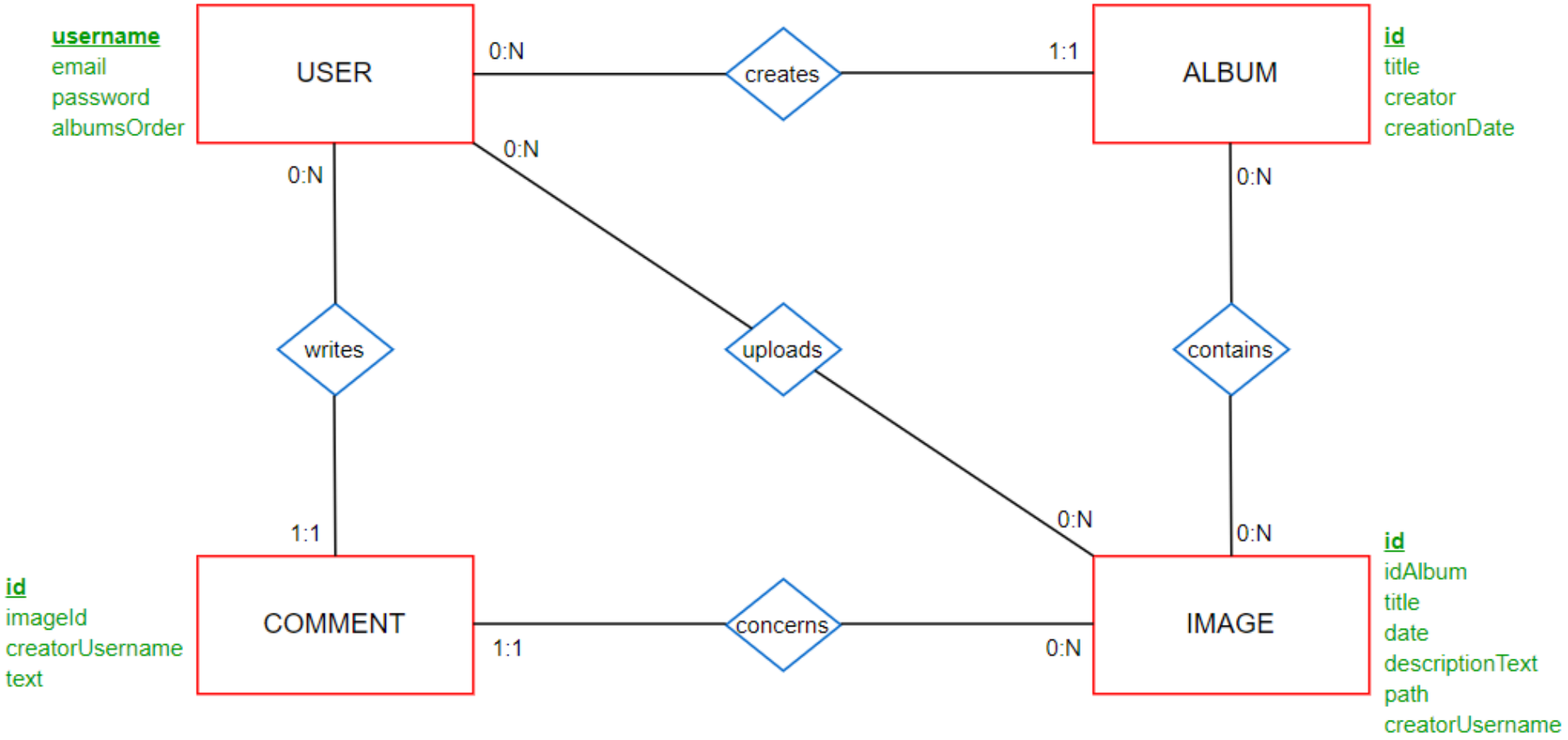
Federica Di Filippo - 10666561

DATA ANALYSIS

Un’applicazione web consente la gestione di una galleria d’immagini. L’applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell’indirizzo di email e l’uguaglianza tra i campi “password” e “ripeti password”. La registrazione controlla l’unicità dello username. Ogni immagine è memorizzata come file nel file system del server su cui l’applicazione è rilasciata. Inoltre nella base di dati sono memorizzati i seguenti attributi: un titolo, una data, un testo descrittivo e il percorso del file dell’immagine nel file system del server. Le immagini sono associate all’utente che le carica. L’utente può creare album e associare a questi le proprie immagini. Un album ha un titolo, il creatore e la data di creazione. Le immagini sono associate a uno o più commenti inseriti dagli utenti (dal proprietario o da altri utenti). Un commento ha un testo e il nome dell’utente che lo ha creato. Quando l’utente accede all’HOME PAGE, questa presenta l’elenco degli album che ha creato e l’elenco degli album creati da altri utenti. Entrambi gli elenchi sono ordinati per data di creazione decrescente.

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

DATABASE DESIGN



APPLICATION REQUIREMENTS ANALYSIS

Un'applicazione web consente la gestione di una galleria d'immagini. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username. Ogni immagine è memorizzata come file nel file system del server su cui l'applicazione è rilasciata. Inoltre nella base di dati sono memorizzati i seguenti attributi: un titolo, una data, un testo descrittivo e il percorso del file dell'immagine nel file system del server. Le immagini sono associate all'utente che le carica. L'utente può creare album e associare a questi le proprie immagini. Un album ha un titolo, il creatore e la data di creazione. Le immagini sono associate a uno o più commenti inseriti dagli utenti (dal proprietario o da altri utenti). Un commento ha un testo e il nome dell'utente che lo ha creato. Quando l'utente accede all'HOME PAGE, questa presenta l'elenco degli album che ha creato e l'elenco degli album creati da altri utenti. Entrambi gli elenchi sono ordinati per data di creazione decrescente. Quando l'utente clicca su un album che appare negli elenchi della HOME PAGE, appare la pagina ALBUM PAGE che contiene inizialmente una tabella di una riga e cinque colonne. Ogni cella contiene una miniatura (thumbnail) e il titolo dell'immagine. Le miniature sono ordinate da sinistra a destra per data decrescente.

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

APPLICATION REQUIREMENTS ANALYSIS

Se l'album contiene più di cinque immagini, sono disponibili comandi per vedere il precedente e successivo insieme di cinque immagini. Se la pagina ALBUM PAGE mostra il primo blocco d'immagini e ne esistono altre successive nell'ordinamento, compare a destra della riga il bottone SUCCESSIVE, che permette di vedere le successive cinque immagini. Se la pagina ALBUM PAGE mostra l'ultimo blocco d'immagini e ne esistono altre precedenti nell'ordinamento, compare a sinistra della riga il bottone PRECEDENTI, che permette di vedere le cinque immagini precedenti. Se la pagina ALBUM PAGE mostra un blocco d'immagini e ne esistono altre precedenti e successive nell'ordinamento, compare a destra della riga il bottone SUCCESSIVE, che permette di vedere le successive cinque immagini, e a sinistra il bottone PRECEDENTI, che permette di vedere le cinque immagini precedenti. Quando l'utente seleziona una miniatura, la pagina ALBUM PAGE mostra tutti i dati dell'immagine scelta, tra cui la stessa immagine a grandezza naturale e i commenti eventualmente presenti. La pagina mostra anche una form per aggiungere un commento. L'invio del commento con un bottone INVIA ripresenta la pagina ALBUM PAGE, con tutti i dati aggiornati della stessa immagine. La pagina ALBUM PAGE contiene anche un collegamento per tornare all'HOME PAGE. L'applicazione consente il logout dell'utente.

Pages (views), view components, events, actions

APPLICATION REQUIREMENTS ANALYSIS

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 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.
- L'evento di visualizzazione del blocco precedente/successivo d'immagini di un album è gestito a lato client senza generare una richiesta al server.

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

APPLICATION REQUIREMENTS ANALYSIS

- Quando l'utente passa con il mouse su una miniatura, l'applicazione mostra una finestra modale con tutte le informazioni dell'immagine, tra cui la stessa a grandezza naturale, i commenti eventualmente presenti e la form per inserire un commento.
- L'applicazione controlla anche a lato client che non si invii un commento vuoto.
- Errori a lato server devono essere segnalati mediante un messaggio di allerta all'interno della pagina.
- Si deve consentire all'utente di riordinare l'elenco dei propri album con un criterio diverso da quello di default (data decrescente). L'utente trascina il titolo di un album nell'elenco e lo colloca in una posizione diversa per realizzare l'ordinamento che desidera, senza invocare il server. Quando l'utente ha raggiunto l'ordinamento desiderato, usa un bottone "salva ordinamento", per memorizzare la sequenza sul server. Ai successivi accessi, l'ordinamento personalizzato è usato al posto di quello di default.

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

LOCAL DATABASE SCHEMA

```
CREATE TABLE `album` (
`id` int NOT NULL AUTO_INCREMENT,
`title` varchar(45) NOT NULL,
`creator` varchar(45) NOT NULL,
 `creationDate` date NOT NULL,  PRIMARY KEY (`id`))
```

```
CREATE TABLE `image` (
`id` int NOT NULL AUTO_INCREMENT,
`idAlbum` int NOT NULL,
`title` varchar(45) NOT NULL,
`date` date NOT NULL,
`descriptionText` varchar(100) NOT NULL,
`path` varchar(200) NOT NULL,
`creatorUsername` varchar(45) NOT NULL,  PRIMARY KEY (`id`))
```

```
CREATE TABLE `comment` (
`id` int NOT NULL AUTO_INCREMENT,  `imageId` int
NOT NULL,
 `creatorUsername` varchar(45) NOT NULL,  `text`
varchar(400) NOT NULL,  PRIMARY KEY (`id`))
```

```
CREATE TABLE `user` (
`username` varchar(45) NOT NULL,
`email` varchar(45) NOT NULL,
`password` varchar(45) NOT NULL,
`albumsOrder` varchar(45) DEFAULT NULL,  PRIMARY KEY
(`username`))
```

COMPONENTS: SERVER SIDE

- Model objects (Beans)
  -User
  -Album
  -Image
  -Comment

- Data Access Objects (Classes)
  -UserDAO
  -AlbumDAO
  -ImageDAO
  -CommentDAO

- Controllers

-AddComment

-AddImageToAlbum

-CheckLogin

-CreateAlbum

-GetAlbumPage

-GetHomePage

-GetImageComments

-Logout

-SaveNewAlbumsOrder

-SignUp

- Utils

-ConnectionHandler

COMPONENTS: CLIENT SIDE

INDEX

- LoginForm
- SignUpForm

HOME

- AlbumList:

  -reset: hide containers

  -registerEvents: add listeners to forms and for drag and drop

  -getAlbumList: returns all albums

  -update: divide and show user's and other users' albums

  -getDragAfterElement: gets closest position to drop element when dragged

  -getAlbumsOrder: returns current albums order

  -saveNewOrder: saves new albums'order in database

- SelectedAlbum:

  -reset: hides buttons and alerts

  -registerEvents: adds listener to addImageForm

  -show: shows the selected album

  -update: fills images table with images and descriptions

  -checkPreviousNext: checks if there are a previous or a next page

  -previousPage: gets the previous page

  -nextPage: gets the next page

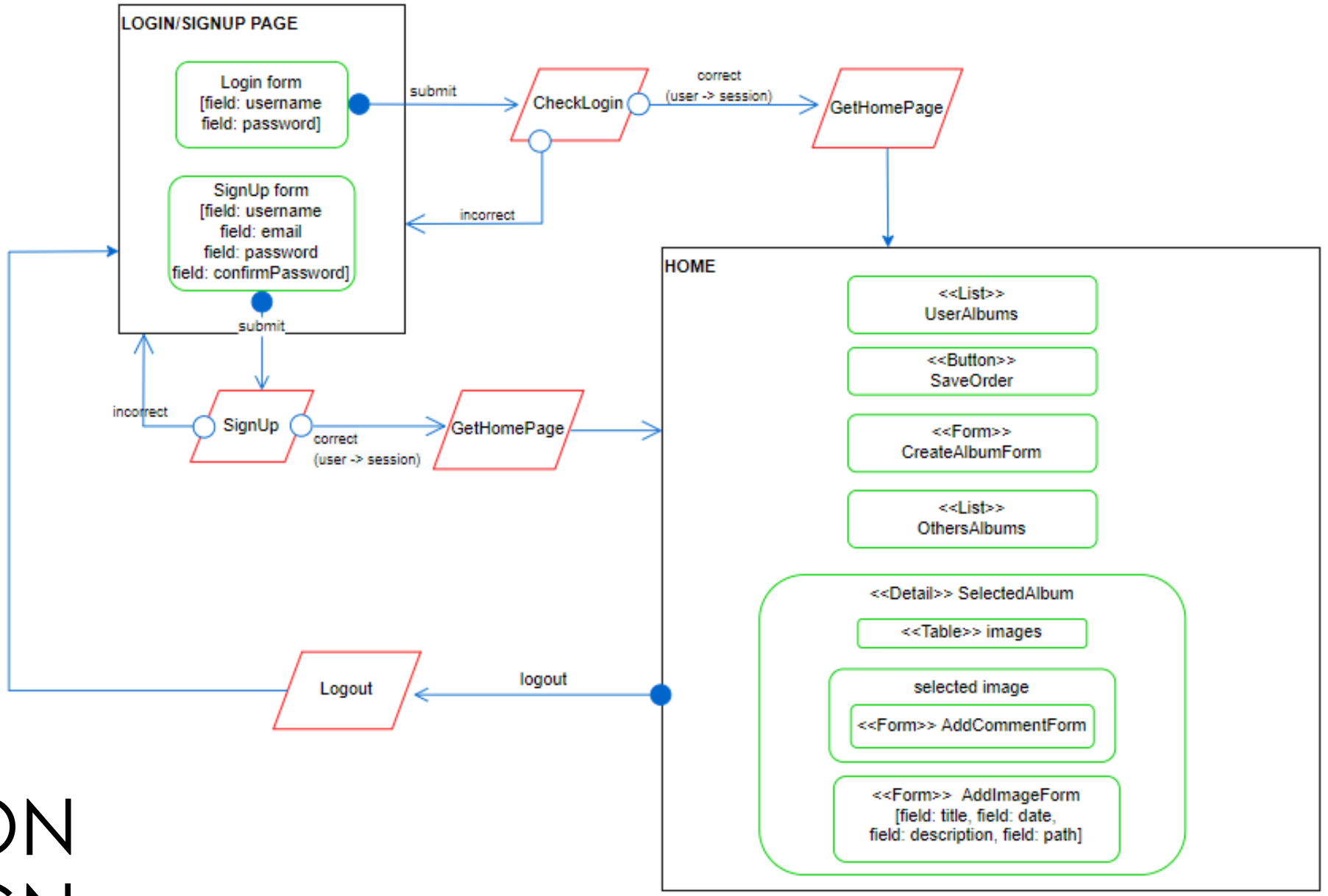
COMPONENTS: CLIENT SIDE

- SelectedImage
  -reset: hides the image container
  -update: shows image with description
  -openModal: opens the modal window when the mouse is over the image
  -closeModal: closes the modal window when « X » button is pressed

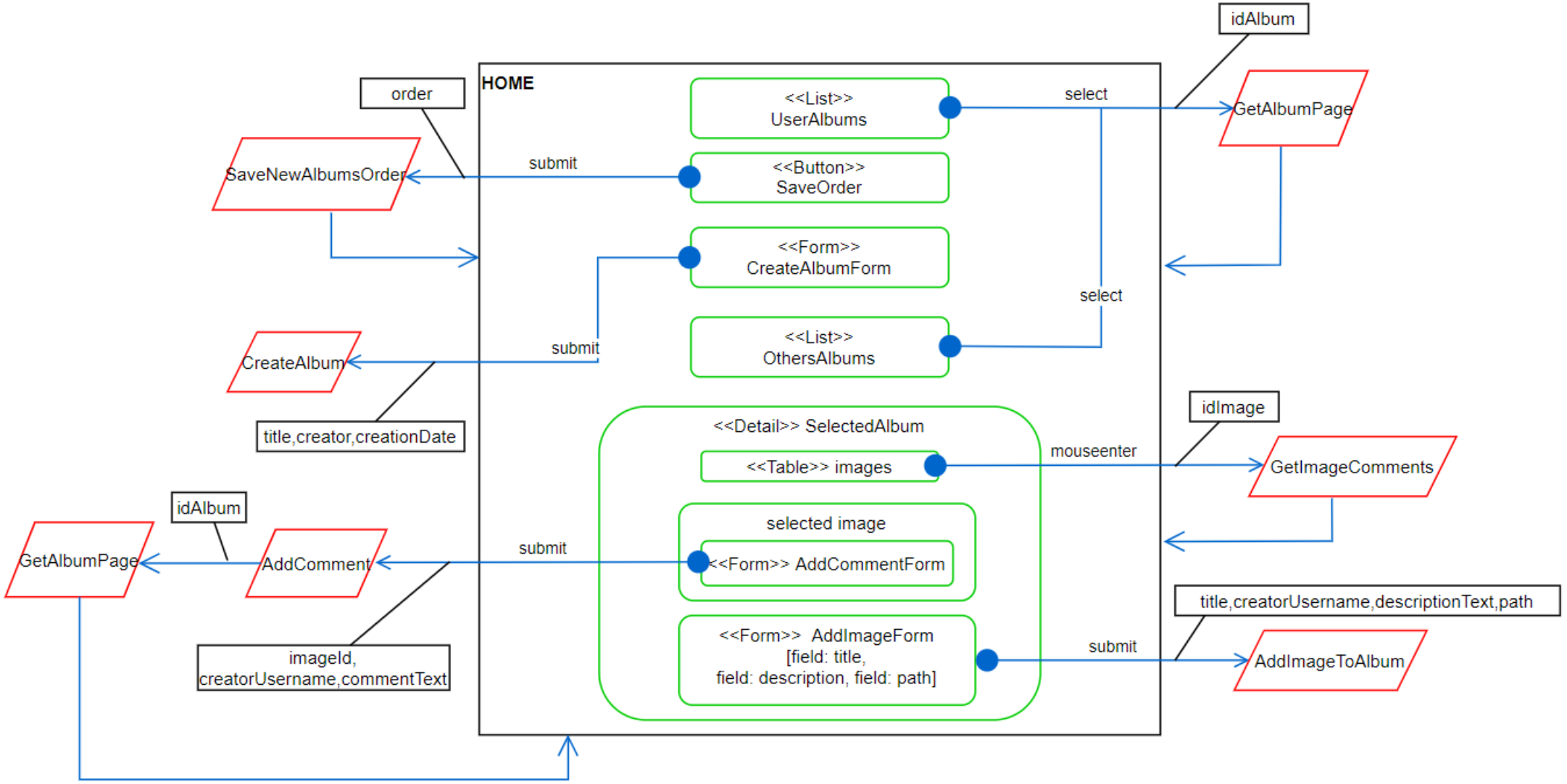
- PageOrchestrator:
  -start: initializes all components
  -refresh: reset all components and load the page

- Comment Section:
  -reset: hides comments container
  -show: gets image's comment from database
  -update: shows comments and AddCommentForm
  -registerEvents: adds listener to AddCommentForm

APPLICATION DESIGN



APPLICATION DESIGN



EVENTS AND ACTIONS

CLIENT SIDE		*SERVER SIDE*	
EVENT	**ACTION**	**EVENT**	**ACTION**
Index.html ⇒ login form ⇒ submit	Data check	POST Username, password	Credentials check
Index.html ⇒ signup form ⇒ submit	Data check	POST username,email, password, confirmPsw	Credentials check, insert user into database
Home.html ⇒ load	Update albums view	GET (no param value)	All albums extraction
Home ⇒ album list ⇒ select album	Update view with album images	GETet id album	Album images extraction
Home ⇒ album list ⇒save order	Data check	POST new order (string)	Save new albums order in database
Home ⇒ selected album ⇒ select image	Update view with image informations (open modal window)	GET idImage	Images comment extraction from database
Home ⇒ addImageForm ⇒ submit	Data check	POST title, description …	Insert image into database

EVENTS AND ACTIONS

CLIENT SIDE		*SERVER SIDE*	
EVENT	**ACTION**	**EVENT**	**ACTION**
Home ⇒ create AlbumForm ⇒ submit	Data check	POST title, date	Insert album into database
Home ⇒ selected Image ⇒ addCommentForm	Data check	POST text, creator, imageId	Insert comment into database
Home ⇒ logout		GET	Close session and redirect to login.html

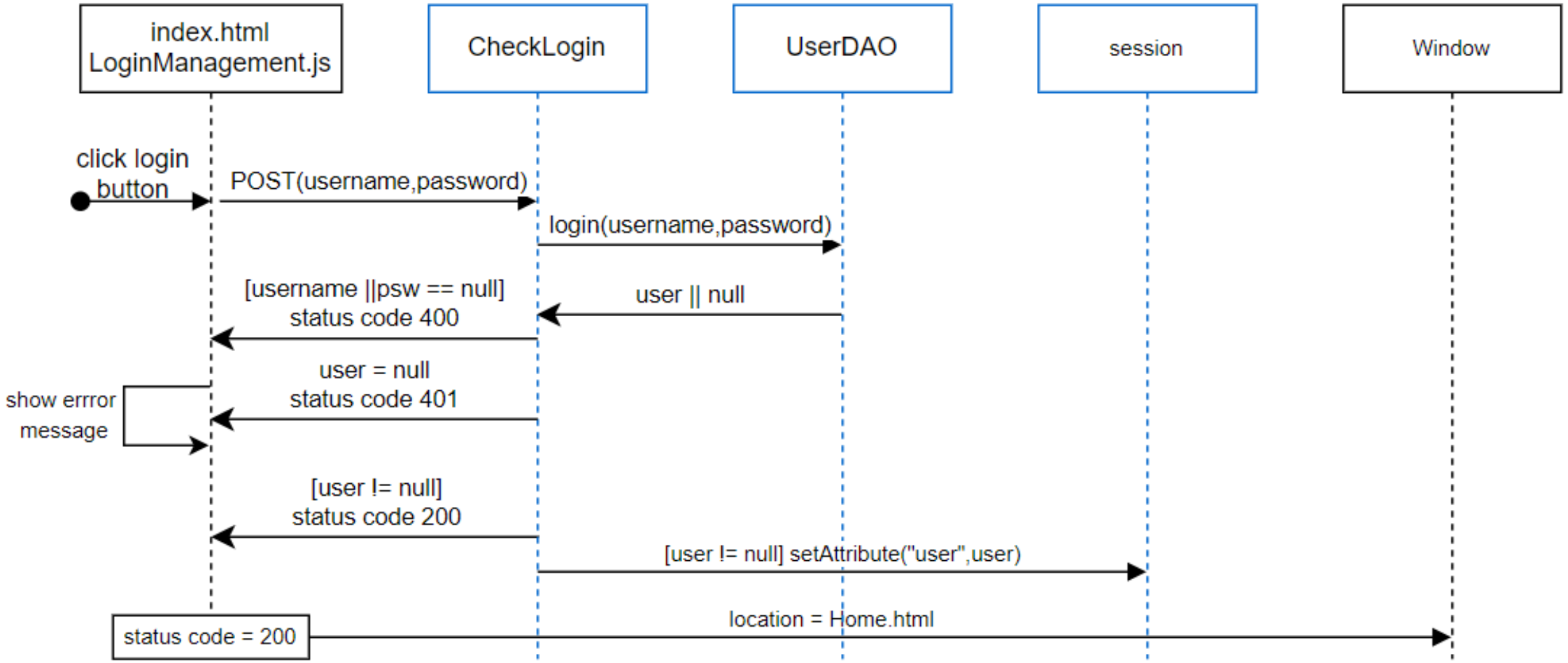
CONTROLLER/EVENT HANDLER

CLIENT SIDE		*SERVER SIDE*	
EVENT	**CONTROLLER**	**EVENT**	**CONTROLLER**
Index.html ⇒ login form ⇒ submit	Function MakeCall	POST Username, password	CheckLogin(servlet)
Index.html ⇒ signup form ⇒ submit	Function MakeCall	POST username,email, password, confirmPsw	SignUp(servlet)
Home.html ⇒ load	Function PageOrchestrator…	GET (no param value)	GetHomePage(servlet)
Home ⇒ album list ⇒ select album	Function show	GETet id album	GetAlbumPage(servlet)
Home ⇒ album list ⇒save order	Function MakeCall	POST new order (string)	SaveNewAlbumsOrder(servlet)
Home ⇒ selected album ⇒ select image	Function openModal,update Function MakeCall	GET idImage	GetImageComments(servlet)
Home ⇒ addImageForm ⇒ submit	Function MakeCall	POST title, description ...	AddImageToAlbum(servlet)

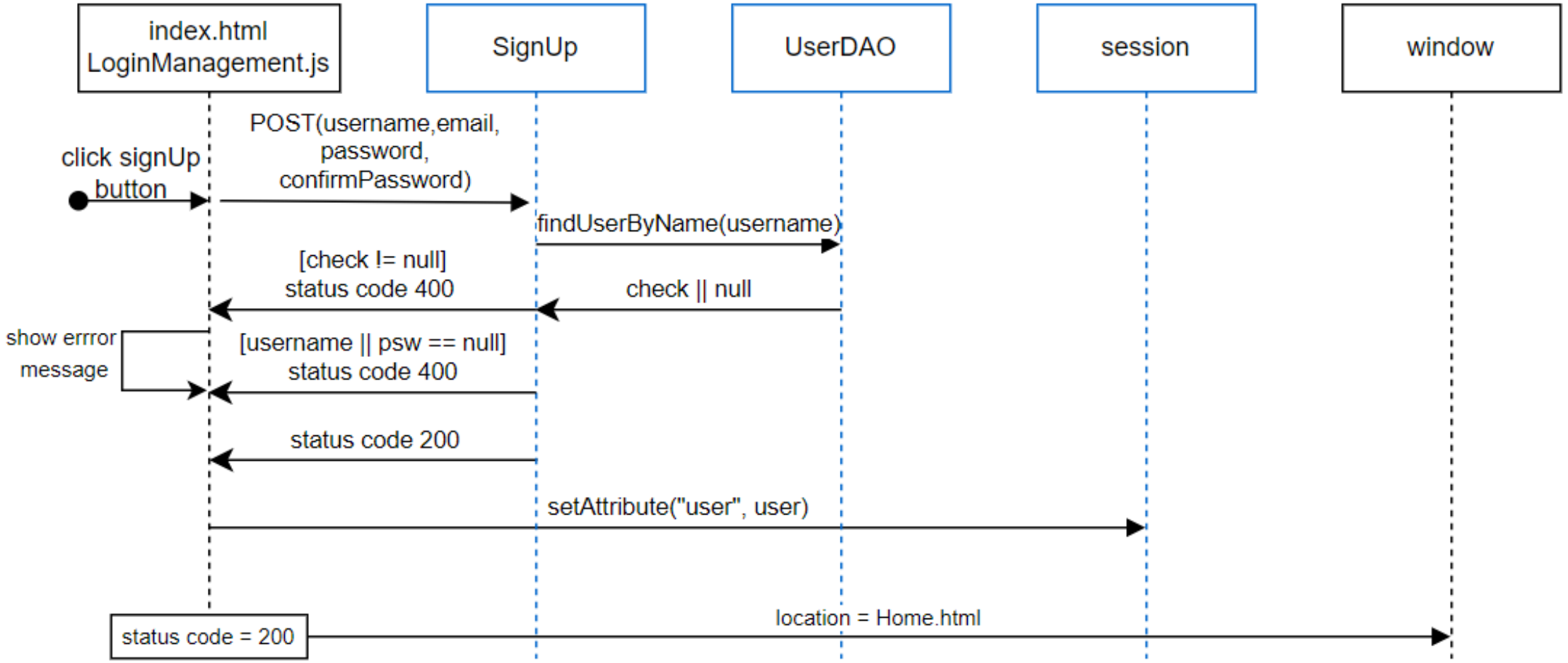
CONTROLLER/EVENT HANDLER

CLIENT SIDE		*SERVER SIDE*	
EVENT	**CONTROLLER**	**EVENT**	**CONTROLLER**
Home ⇒ create AlbumForm ⇒ submit	Function MakeCall	POST title, date	CreateAlbum(servlet)
Home ⇒ selected Image ⇒ addCommentForm	Function MakeCall	POST text, creator, imageId	AddComment(servlet)
Home ⇒ logout		GET	Logour(servlet)

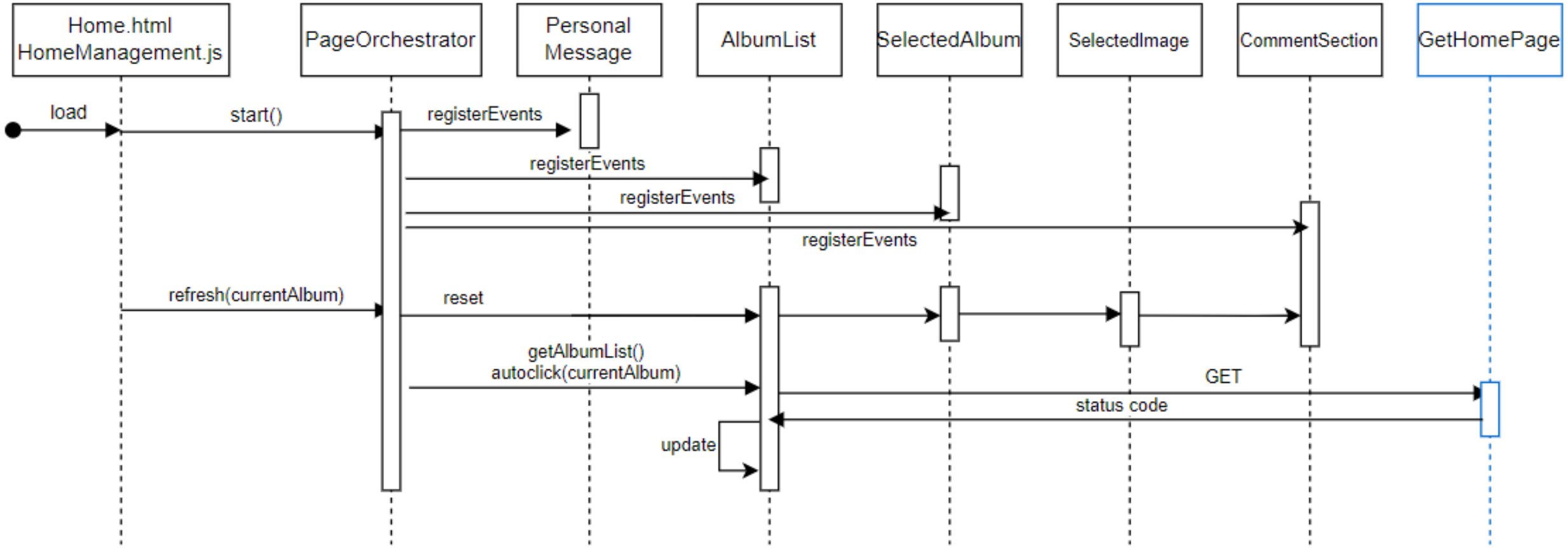
EVENT: LOGIN


index.html
LoginManagement.js
CheckLogin
UserDAO
session
Window
click login button
POST(username,password)
login(username,password)
[username ||psw == null]
status code 400
user || null
user = null
status code 401
show errror message
[user != null]
status code 200
[user != null] setAttribute("user",user)
status code = 200
location = Home.html

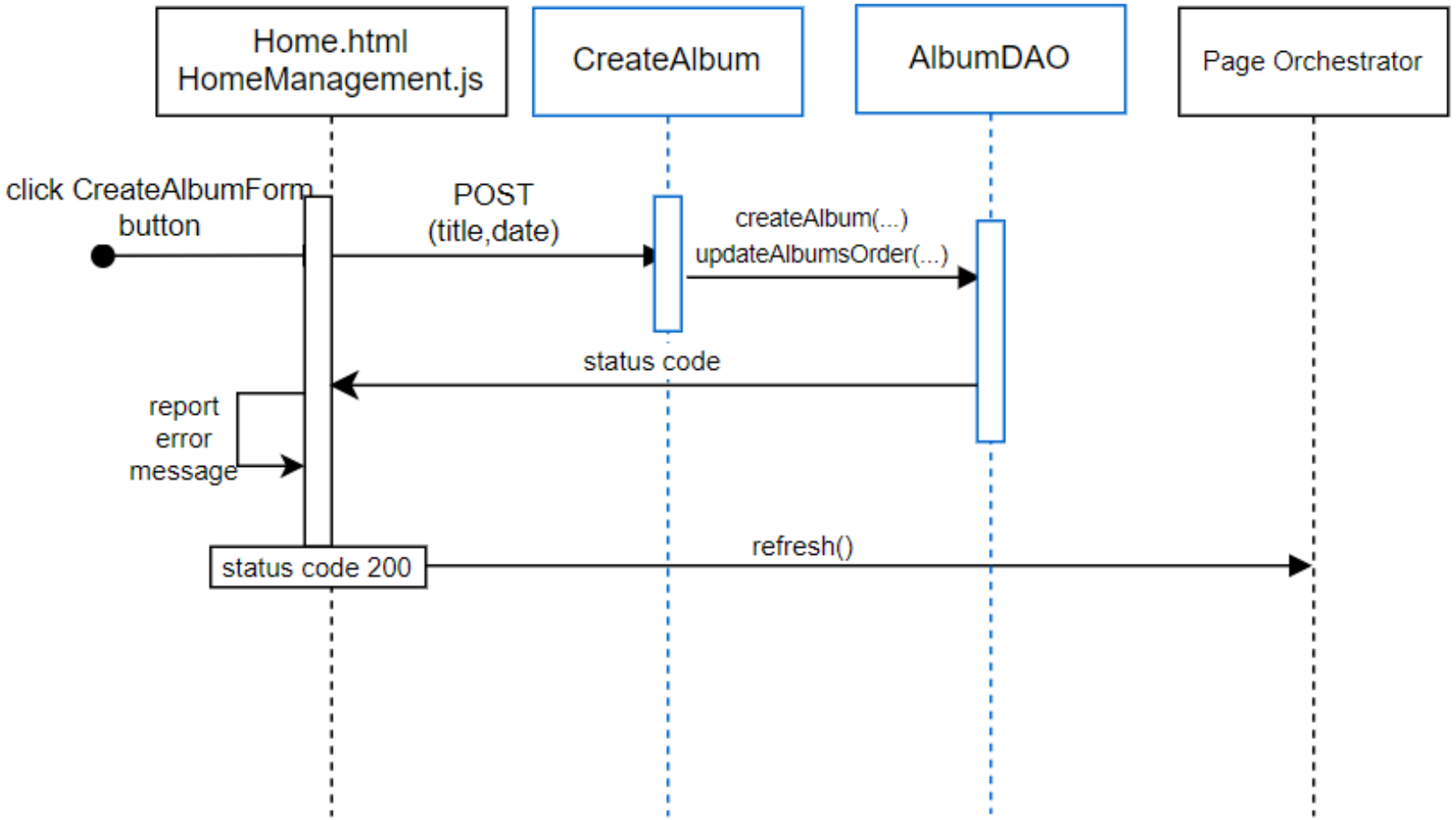
EVENT: SIGNUP



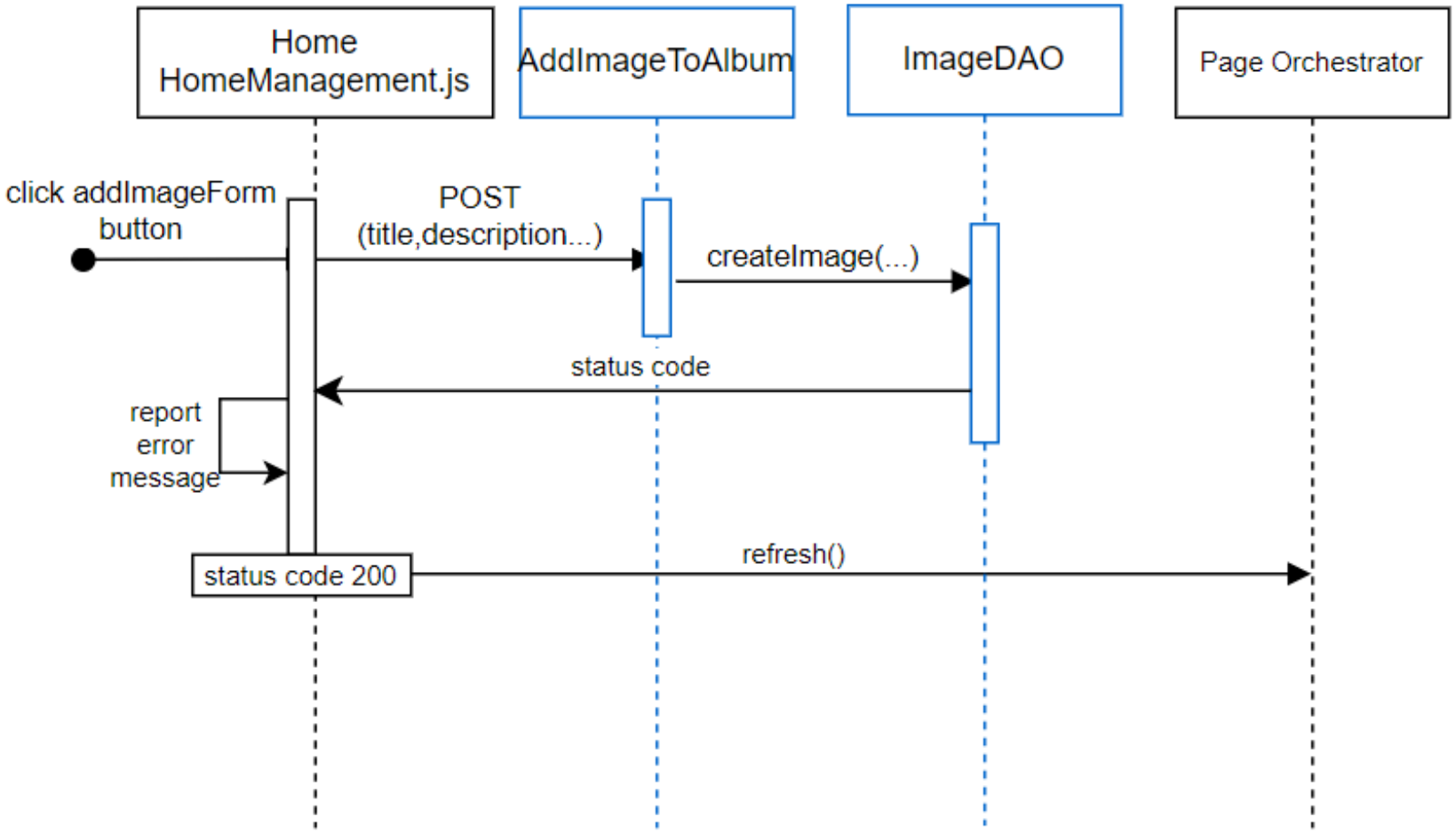
EVENT: GETHOMEPAGE



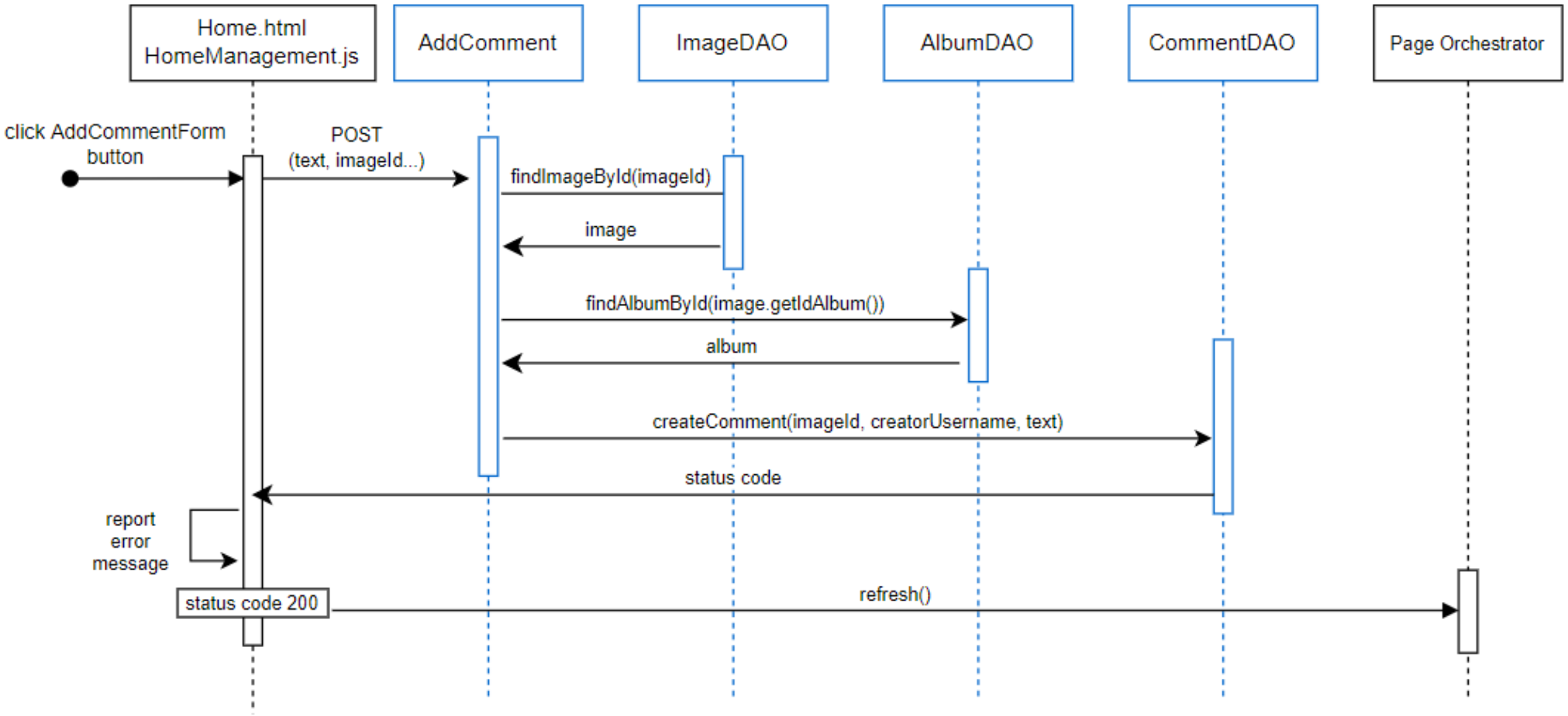
EVENT: CREATE ALBUM



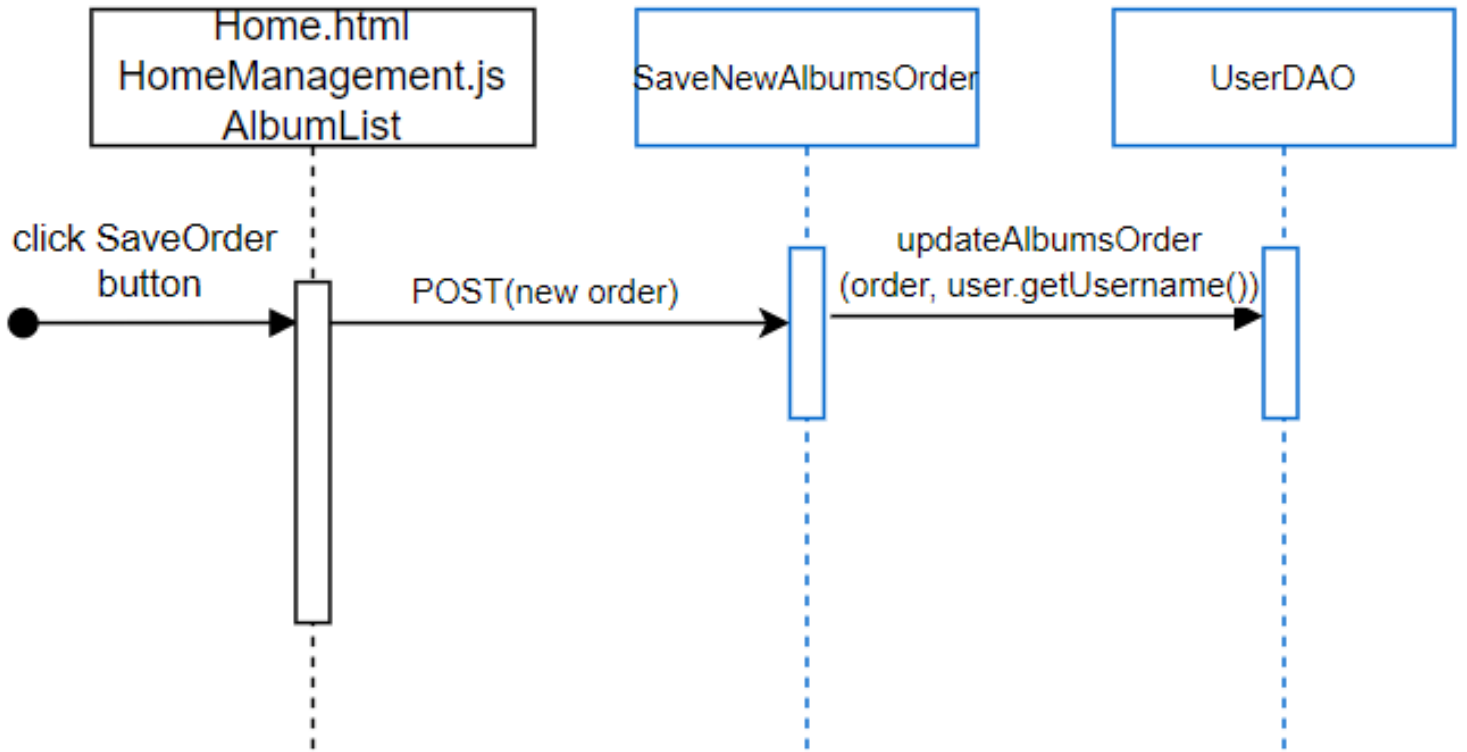
EVENT: ADD IMAGE TO ALBUM



EVENT: ADD COMMENT



EVENT: SAVE NEW ALBUMS ORDER



EVENT: LOGOUT