

Specifiche di Progetto: galleria di immagini

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**.

- Entities
- Attributes
- Relationships

Specifiche di Progetto

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.

• **Pages/views**

• **View**

Ogni immagine è memorizzata come file nel file system del server su cui l'applicazione è rilasciata.

Components

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.

• **Events**

• **Actions**

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.

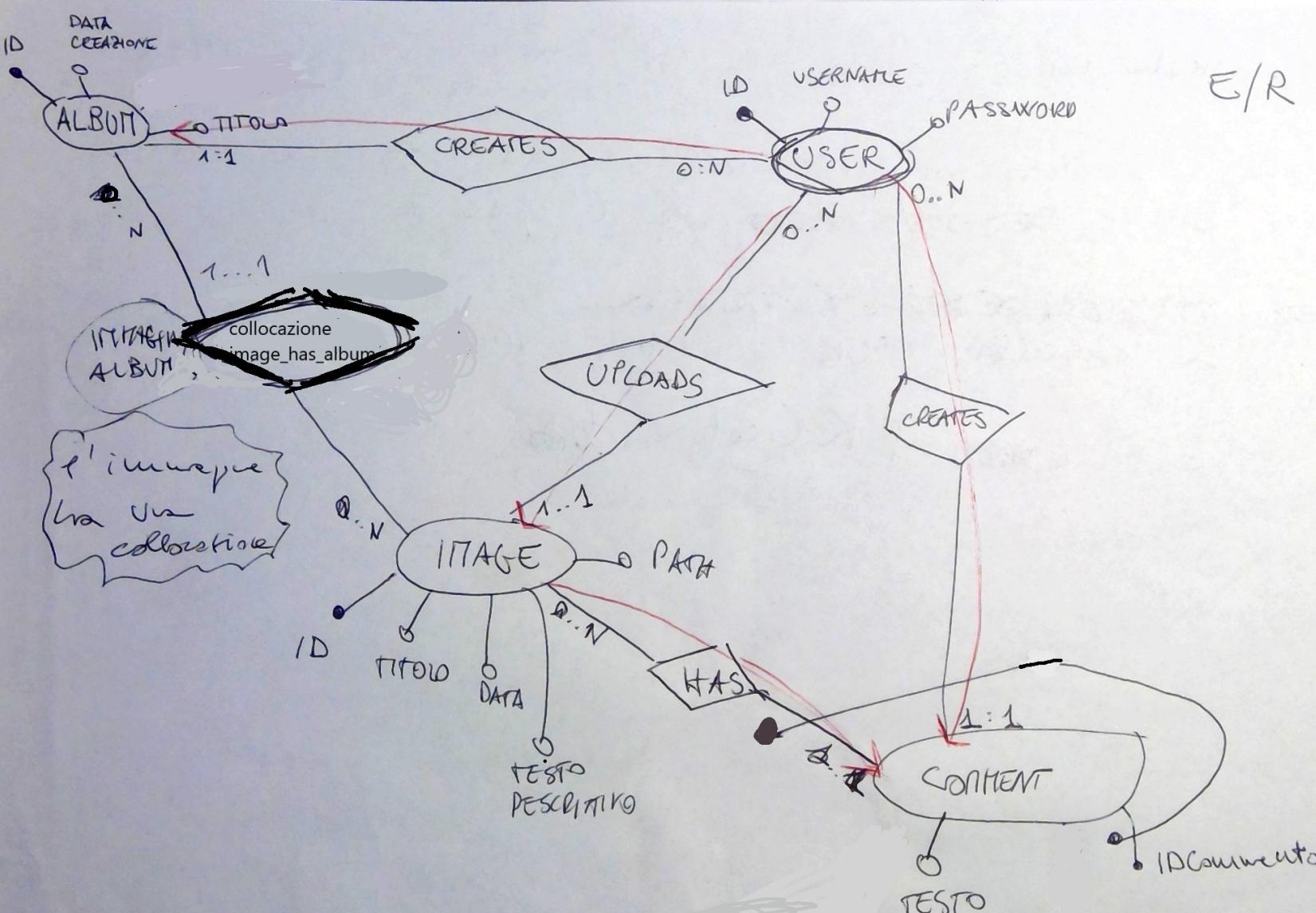
Le immagini sono associate a uno o più commenti inseriti dagli utenti (dal proprietario o da altri utenti). Un commento ha un testo e il nome dell'utente che lo ha creato.

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. 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.

- Pages(views)
- View Components
- Events
- Actions

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.

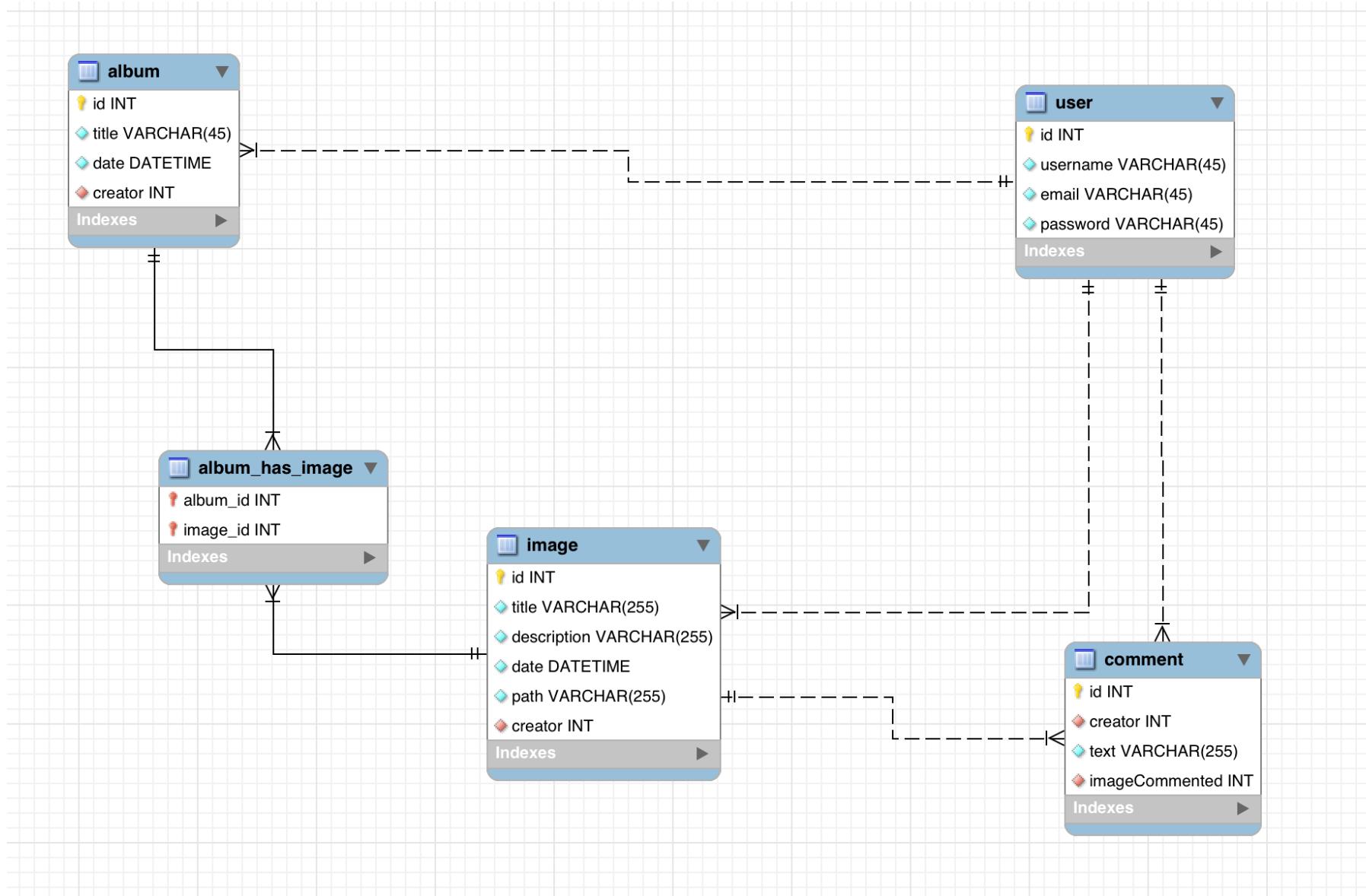
Schema Entità Relazione



E/R

- Un commento viene considerato entità debole siccome non può esistere senza un' immagine ed un utente creatore a cui è associato (nonostante ciò possiede id autoincrementale, why not).
- La relazione molti a molti tra immagine ed album (un'immagine può essere presente in più album, un album possiede più immagini) è stata gestita con una tabella intermedia **album_has_image**.
- Le altre relazioni sono state dedotte dalle azioni che l'utente può effettuare sul modello dati (creazione di un album, caricamento di un'immagine, creazione di un commento).

Design del modello dati nel workbench

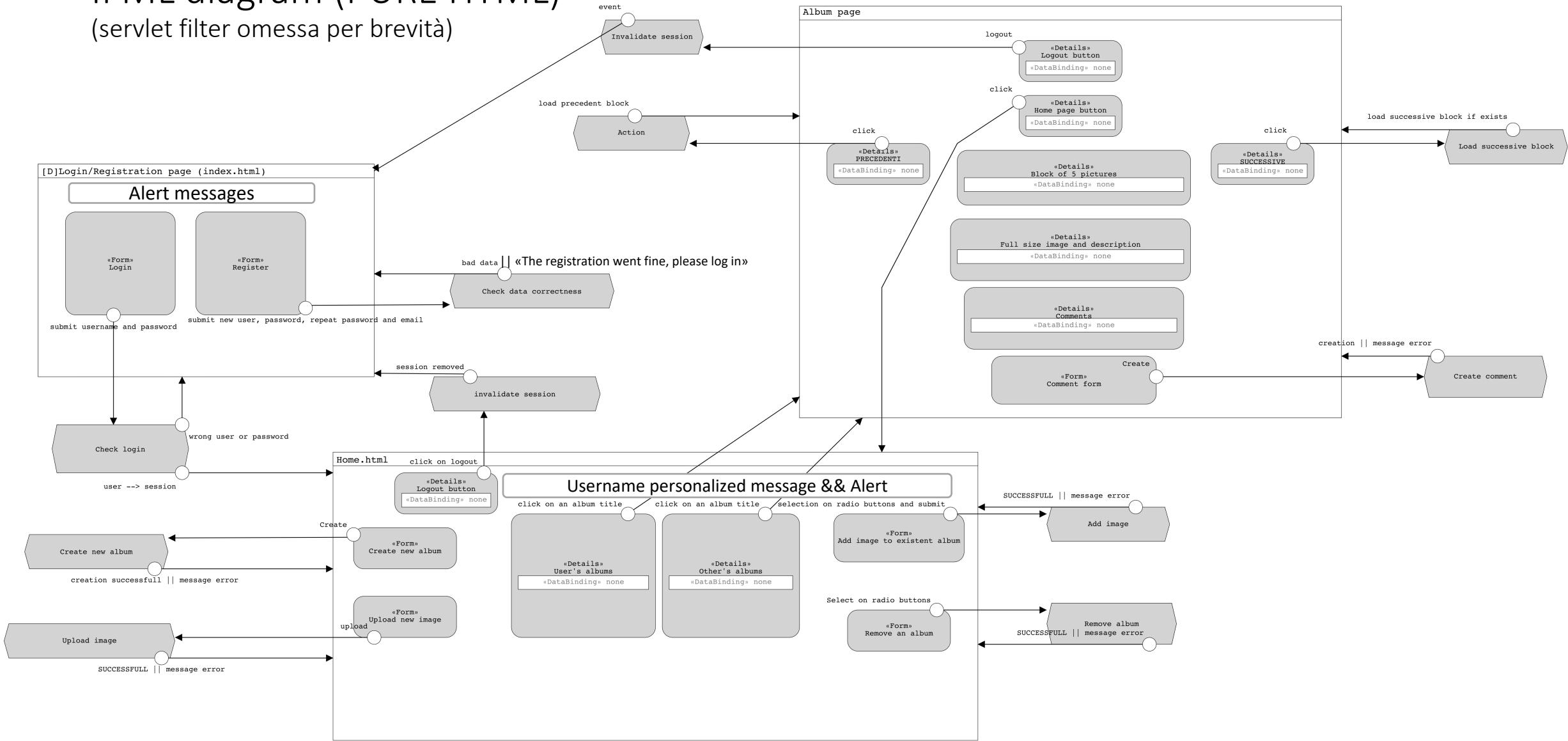


Completamento delle specifiche

- La home page, oltre a possedere la lista degli album dell'utente e la lista degli album degli altri utenti, deve possedere delle form per l'**upload** di immagini, la **creazione** e **rimozione** di un album, l'**aggiunta** di un'immagine esistente ad un album (*essenziale per una completa dimostrazione dell'applicazione*).
- Nella versione ria sarà necessaria un'ulteriore tabella **order** per salvare la path di un file per ciascun utente contenente l'ordine customizzato dall'utente dei propri album.

IFML diagram (PURE HTML)

(servlet filter omessa per brevità)



Components

- **Model objects (Beans)**
 - Album
 - AlbumHasImage
 - Comment
 - Image
 - User
- **Data Access Objects (Classes)**
 - **AlbumDAO**
 - findAlbumsListCreatedBy(int creatorId)
 - findAlbumsListNotCreatedBy(int creatorId)
 - findAlbumById(int albumId)
 - removeAlbum(int albumId)
 - createAlbum(String title, Date date, int creatorid)
 - existAlbumWith(String title, int creator)
 - **AlbumHasImageDAO**
 - findImagesOfAlbum(int albumId)
 - addImageToAlbum(int imageId, int albumId)
 - removeImageFromAlbum(int imageId, int albumId)
 - removeAllImageOfAlbum(int albumId)
 - exist(int albumId, int imageId)
 - **CommentDAO**
 - findCommentsByImage(int imageId)
 - addComment(int creatorId, String text, int imageCommented)
- **ImageDAO**
 - findAllImagesOfAlbum(int albumId)
 - findAllImagesCreatedBy(int creatorId)
 - findImageById(int imageId)
 - createImage(String title, String description, String path, Date date, int creatorid)
 - exist(int imageId)
- **UserDAO**
 - checkCredentials(String identifier, String password)
 - register(String username, String email, String password)
 - findUsernameById(int userId)
 - existUserWithNicknameEqualsTo(String username)

Components

- Controllers (servlets)
 - CheckLogin + GoToHomePage
 - AddComment
 - AddImageToAlbum
 - GoToAlbum
 - Logout
 - Registration
 - RemoveAlbum
 - UploadNewImage
- Filters
 - LoginChecker
- Views (Templates)
 - Login
 - Home
 - Album page

Alcuni screenshot dimostrativi (Versione HTML)

The screenshot shows a web browser window with the URL <http://localhost:8080/ImageGallery/>. The browser's toolbar includes icons for back, forward, refresh, and home, along with a search bar and a tab for the current page.

The main content area displays the "Image Gallery" logo and tagline: "A place to store your images and share your albums". Below this, there are two separate login forms side-by-side.

Login Form (Left):

- Label: Username or email:
Value: Pippo
- Label: Password:
Value: pippo98
- Button: Login

Register Form (Right):

- Label: Insert username:
Value: Pluto
- Label: Insert email:
Value: pluto@gmail.com
- Label: Insert password:
Value: Pluto99
- Label: Repeat password:
Value: Pluto99
- Button: Register

Home page (personalizzata per ciascun utente)

[Logout](#)

Pippo's Home

Create new album

Title:

Date:

 gg / mm / aaaa

Your albums

[Date: 2022-06-05](#)

[Title: Macchine](#)

[Date: 1998-10-10](#)

[Title: Motori](#)

The albums of other users

[Author: Maria](#)

[Date: 2022-05-10](#)

[Title: Barca](#)

[Author: Maria](#)

[Date: 2022-05-02](#)

[Title: Ciao](#)

[Author: Pluto](#)

[Date: 1998-10-10](#)

[Title: Sport](#)

Add image to an existent album

Your images

- Panda
- 500
- Ypsilon
- Renegade
- 500X
- Yaris
- myBPMN

Your albums

- Macchine
- Motori

Upload new image

Title:

Description:

Date:

 gg / mm / aaaa

Nessun file selezionato

Remove Album

Choose the album to remove

- Macchine
- Motori

Album page

[Logout](#)

[Home page](#)

Motori

[Panda](#) [500](#) [Ypsilon](#) [Renegade](#) [500X](#)

[SUCCESSIONE](#)

Panda



1999-01-01

Descrizione

Comments

Pippo
Wow

Pippo
Ciao

Pippo
Pizza

← → ⌂ ⌃ ⌄ ⓘ http://localhost:8080/ImageGallery/GoToAlbum?album=1&image=1

serv. online Posta - Fernando... WeBeep Piazza Library Genesis appunti Polinformatici | Pol...

» Altri Preferiti



1999-01-01

Descrizione

Comments

Pippo
Wow

Pippo
Ciao

Pippo
Pizza

Pippo
Prova di scrittura commento

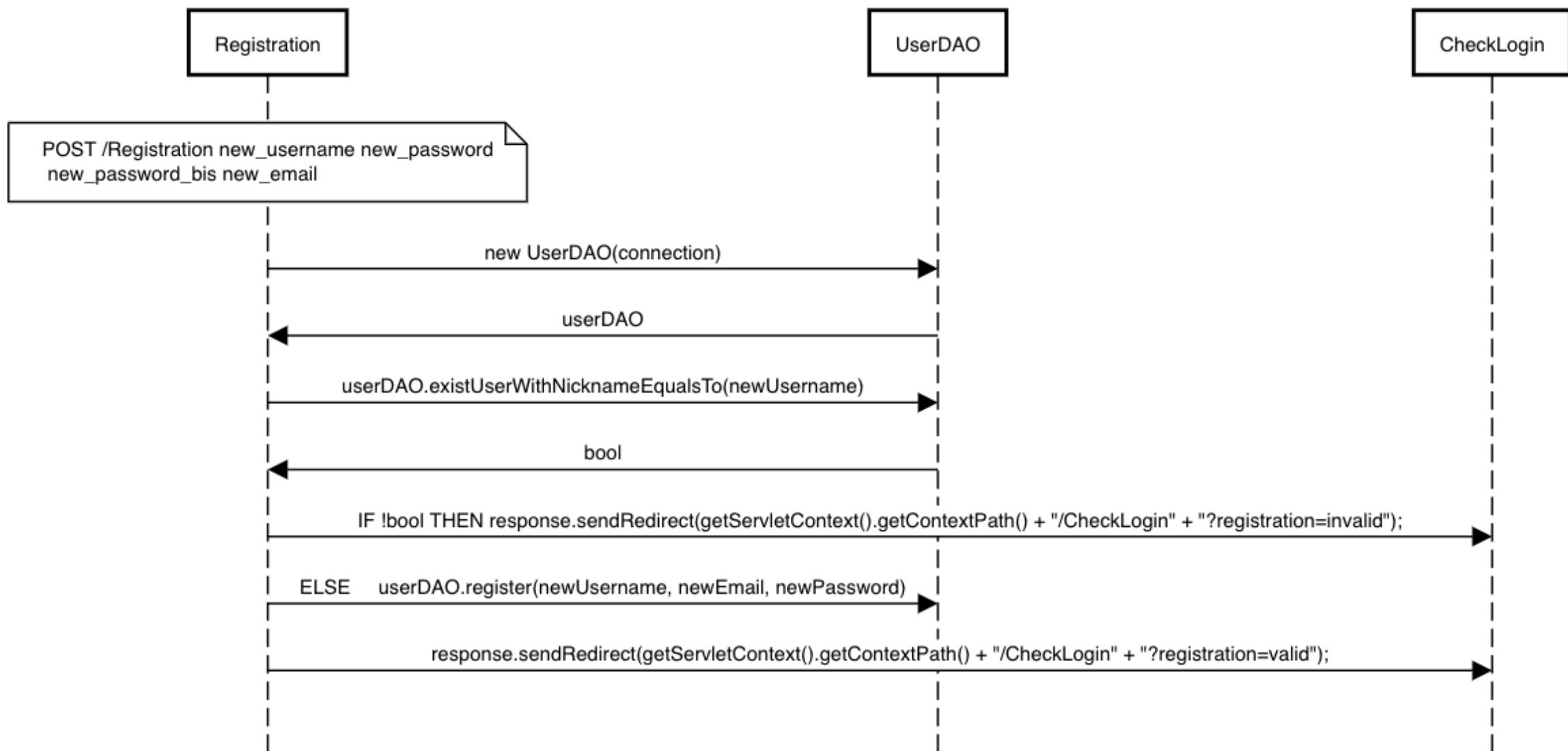
Pippo
FIAT made in italy!!!

Create new comment:

Your comment...

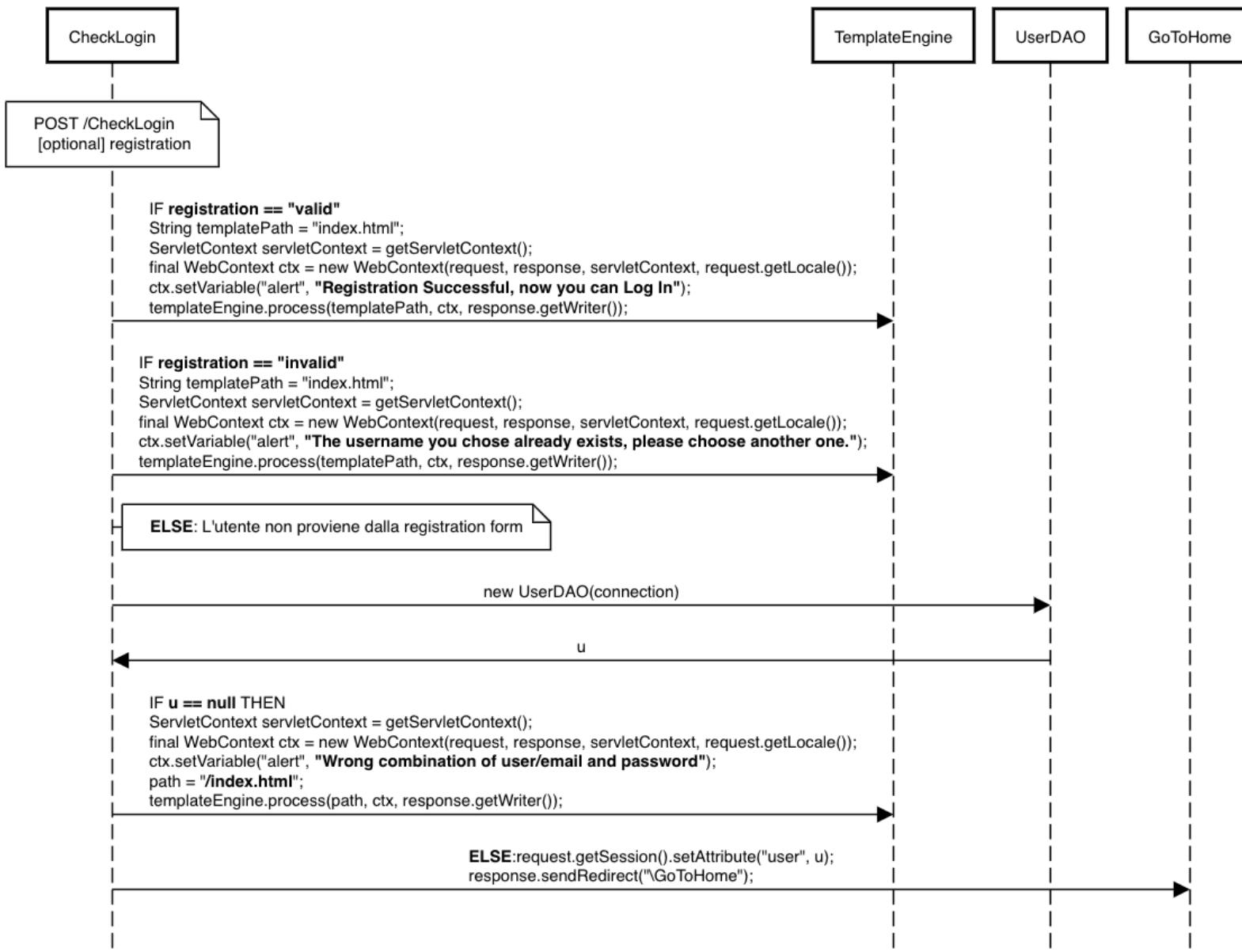
Sequence Diagrams

Registration



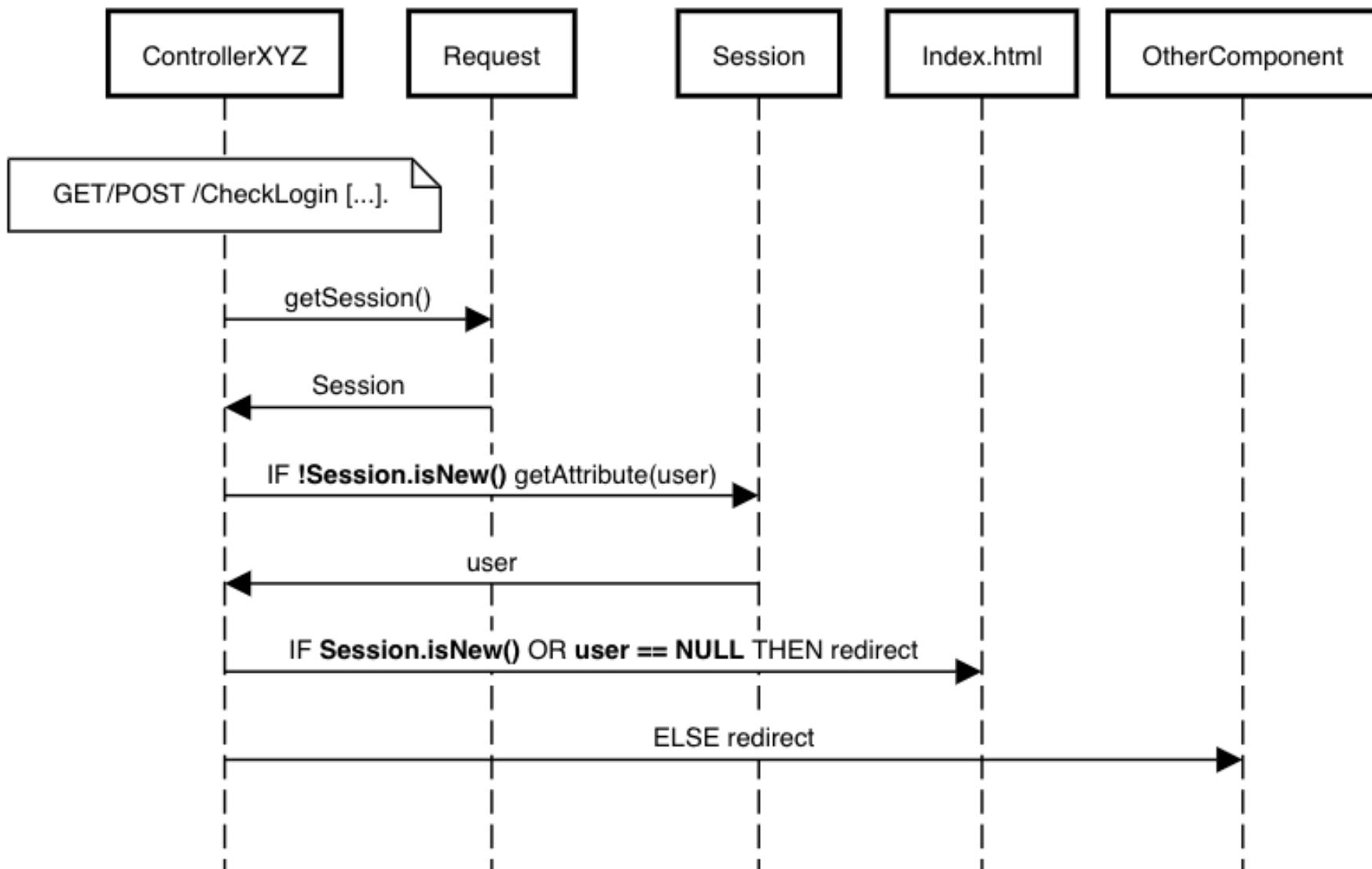
Sequence Diagrams

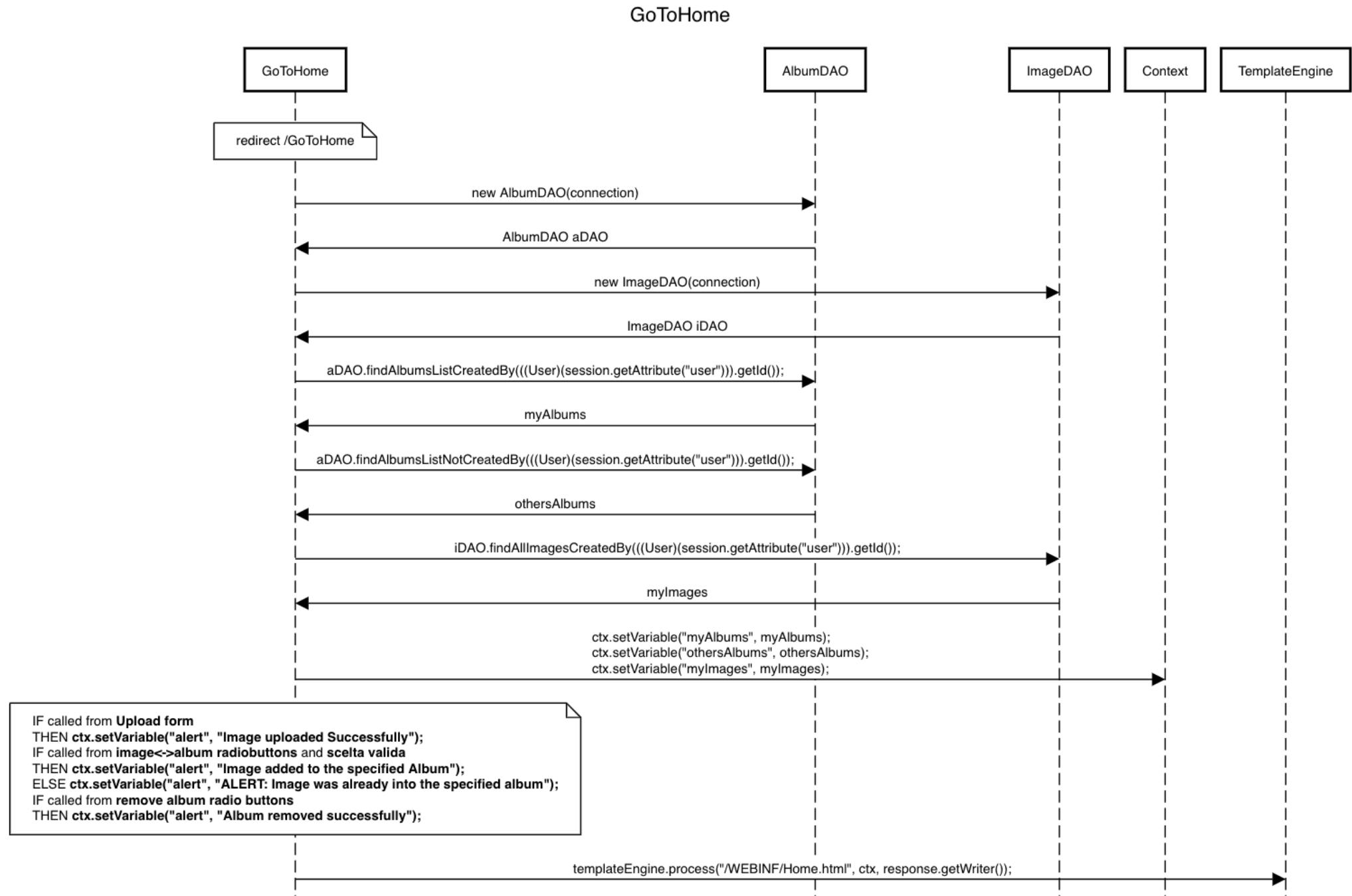
CheckLogin

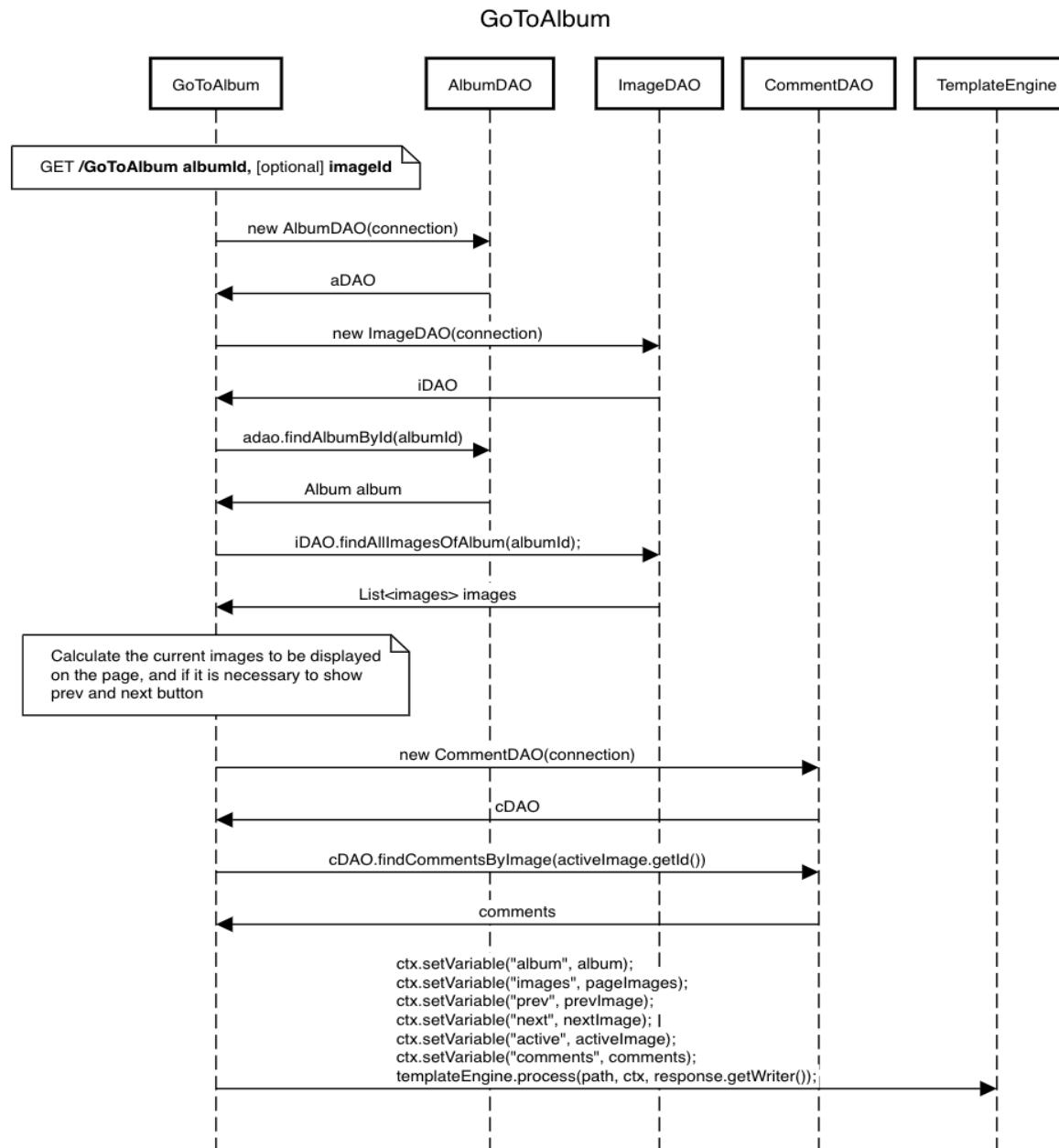


Sequence Diagrams

LoginChecker (servlet filter)

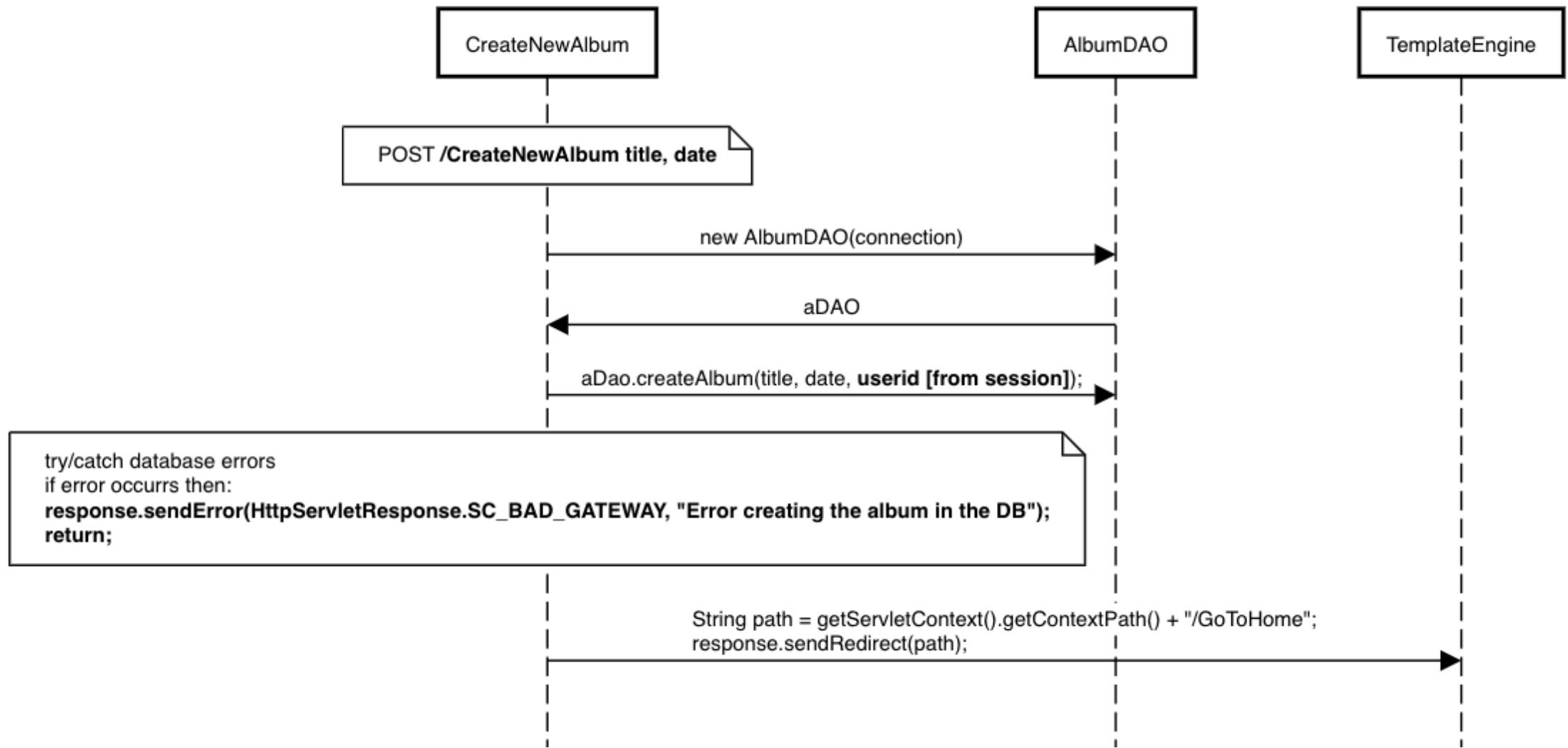






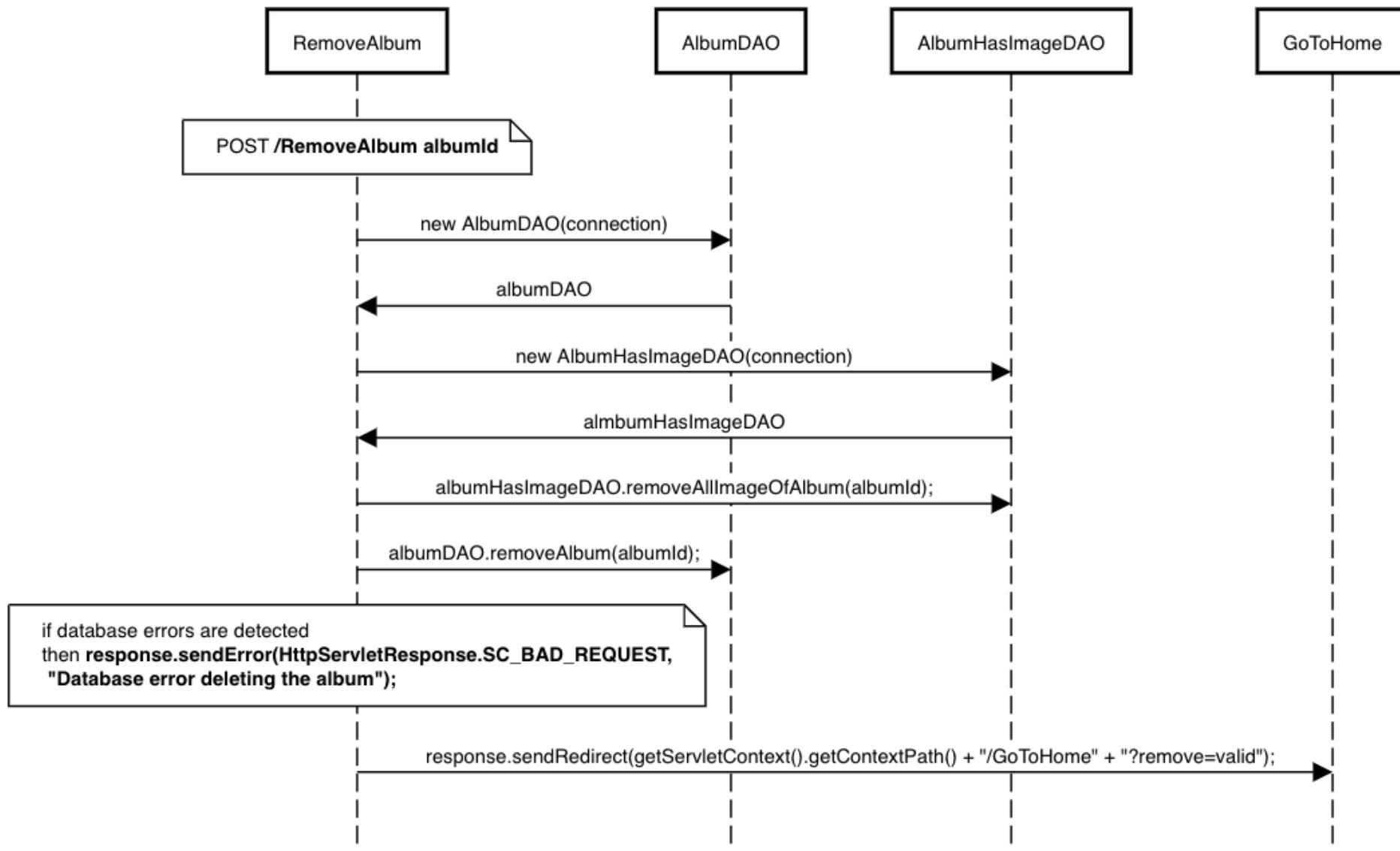
Sequence Diagrams

CreateNewAlbum



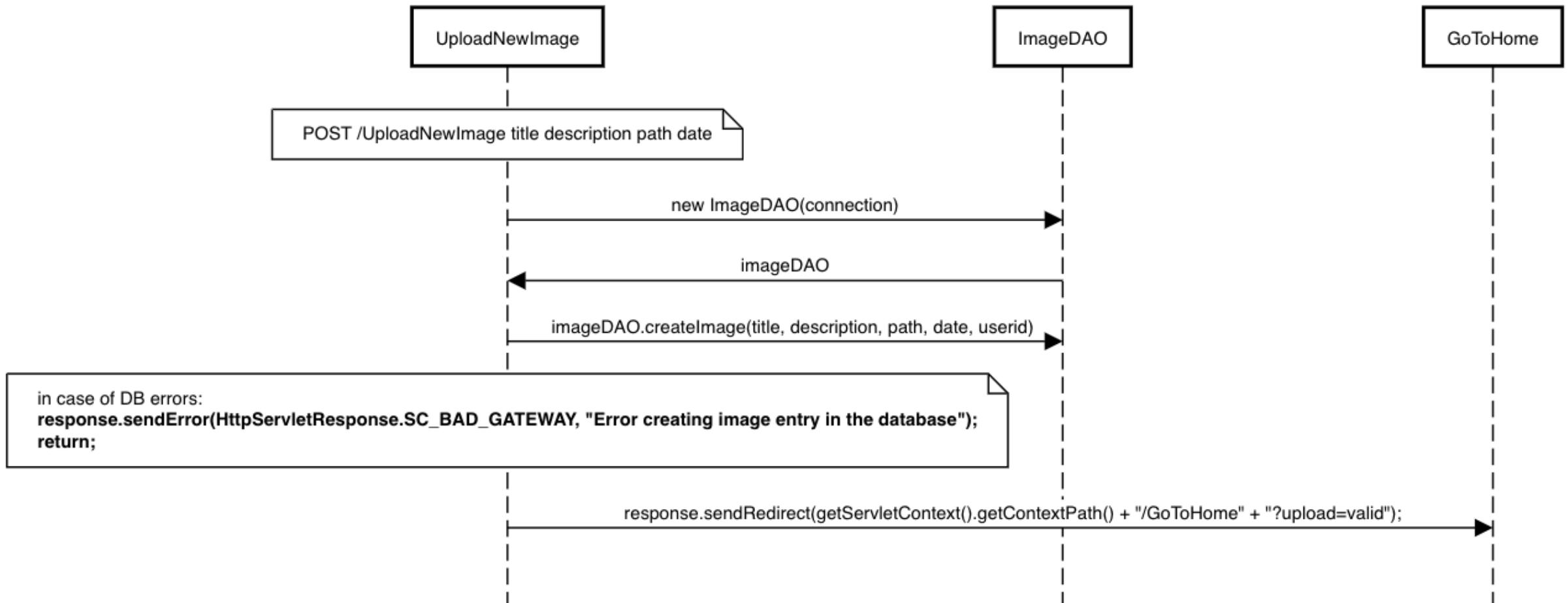
Sequence Diagrams

RemoveAlbum



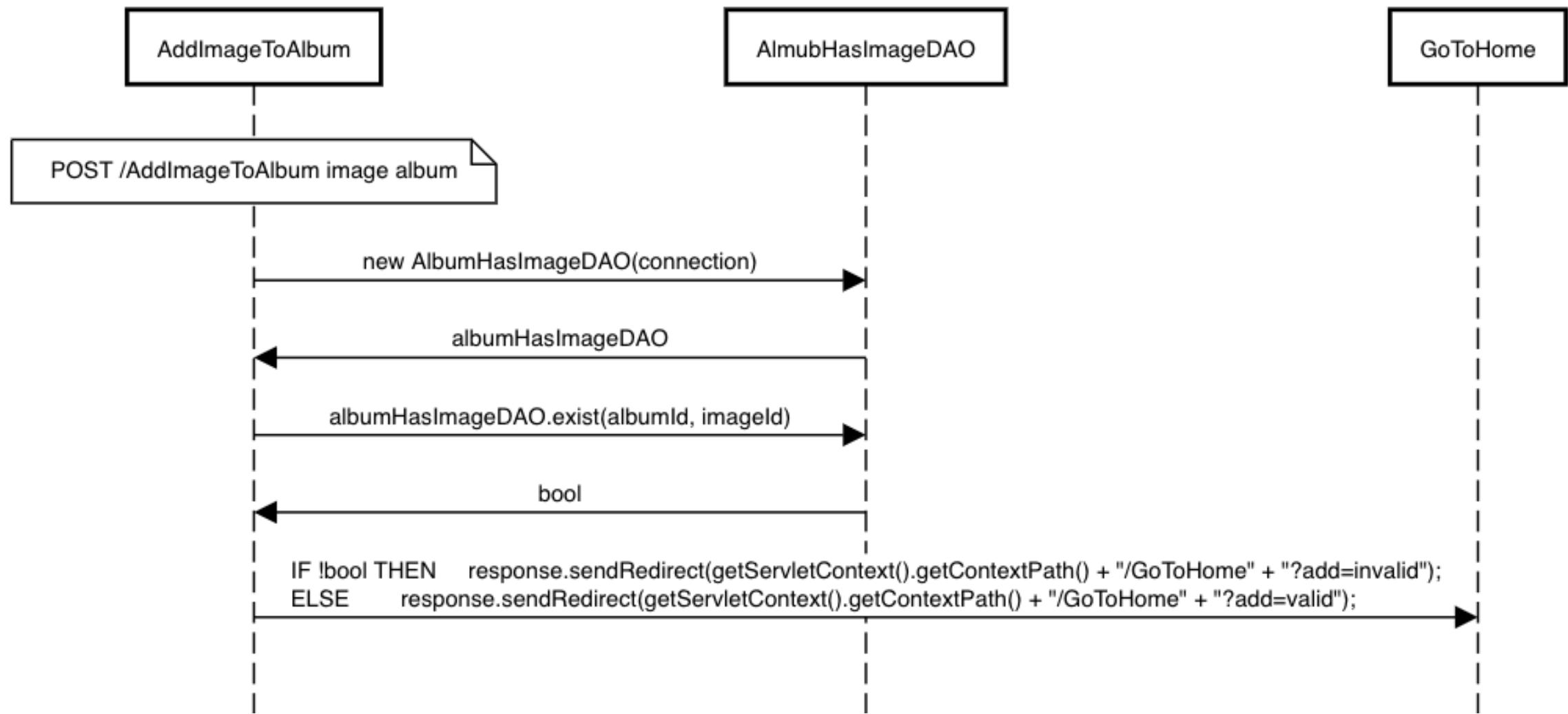
Sequence Diagrams

UploadNewImage



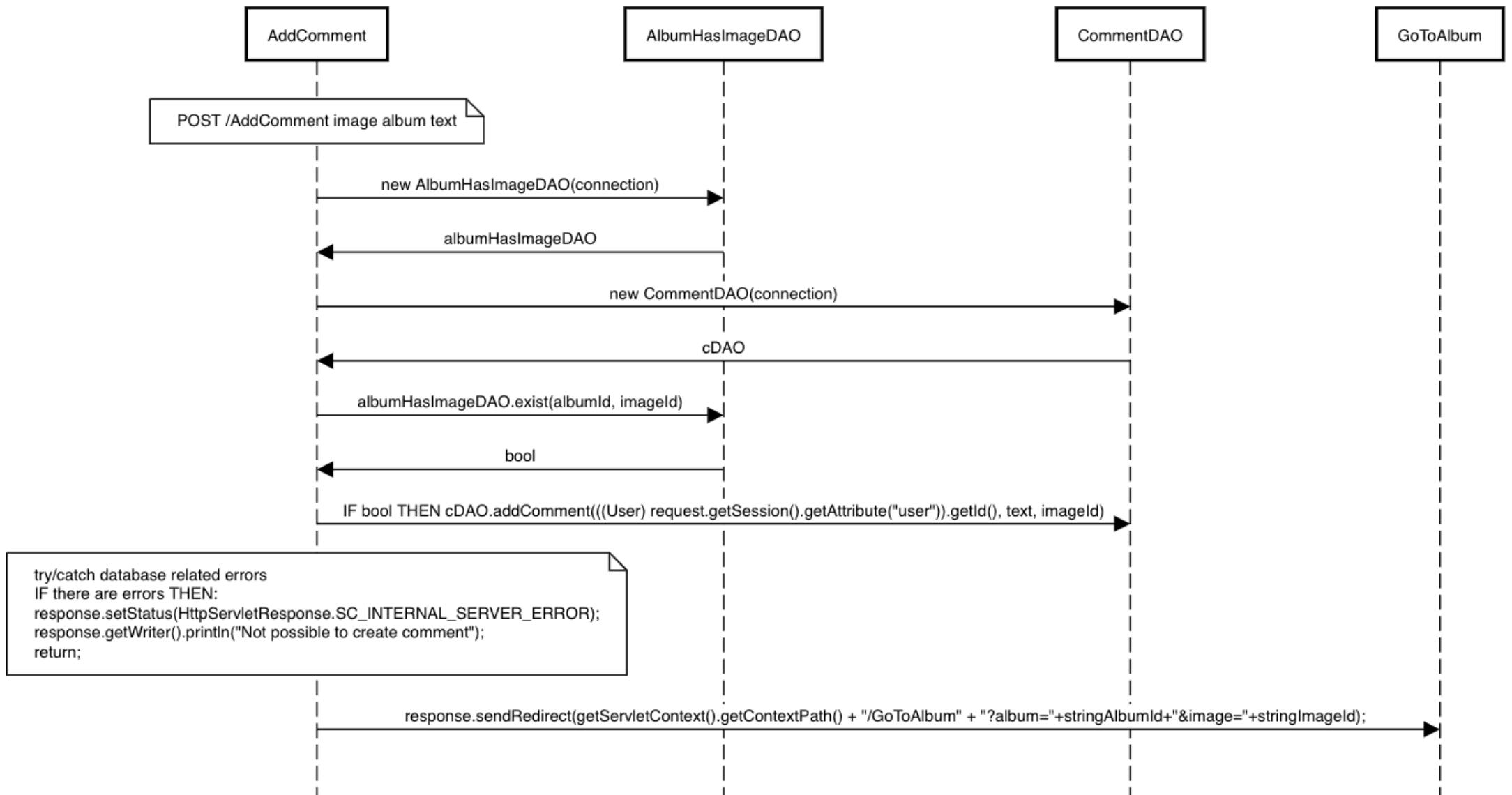
Sequence Diagrams

AddImageToAlbum



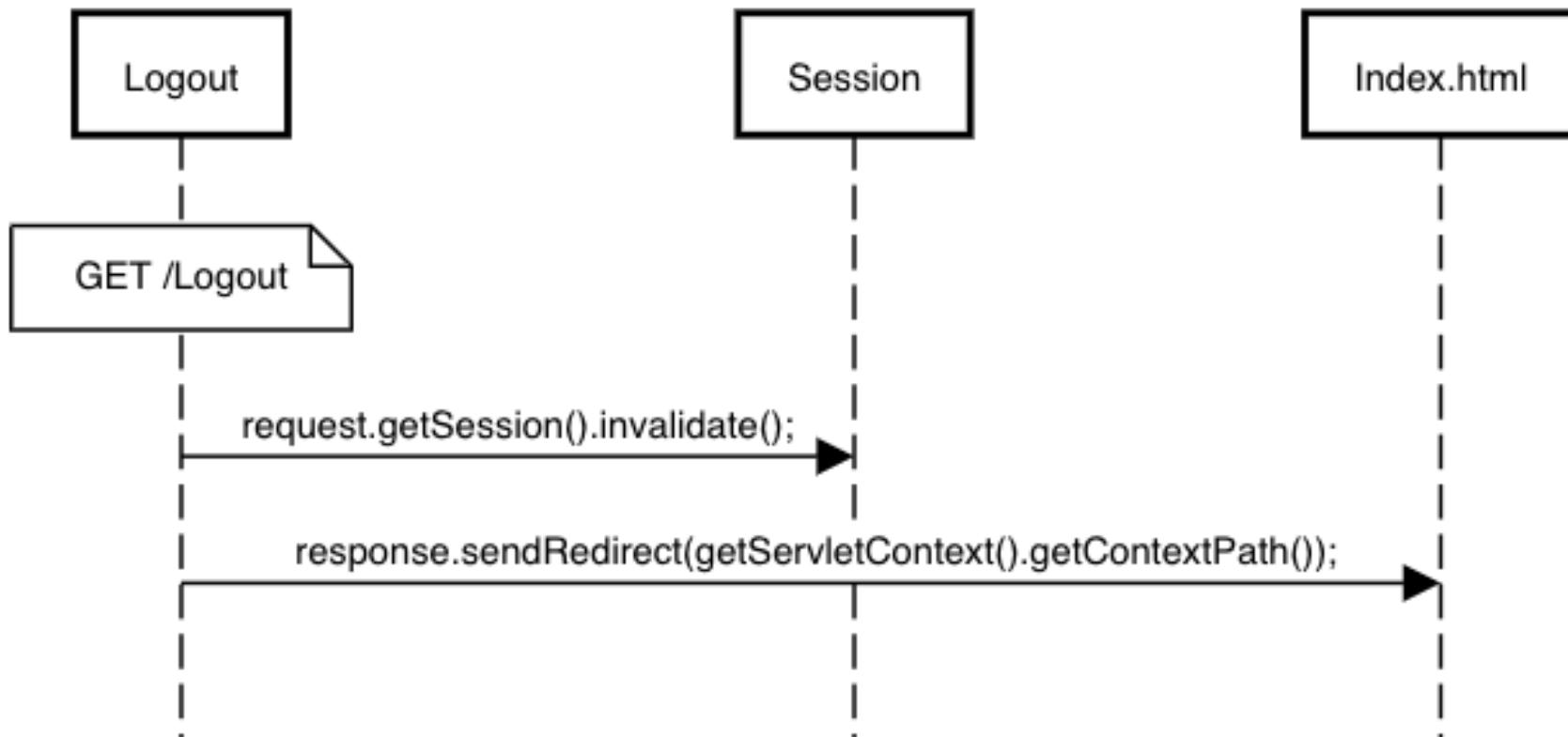
Sequence Diagrams

AddComment



Sequence Diagrams

Logout



Requisiti Versione RIA

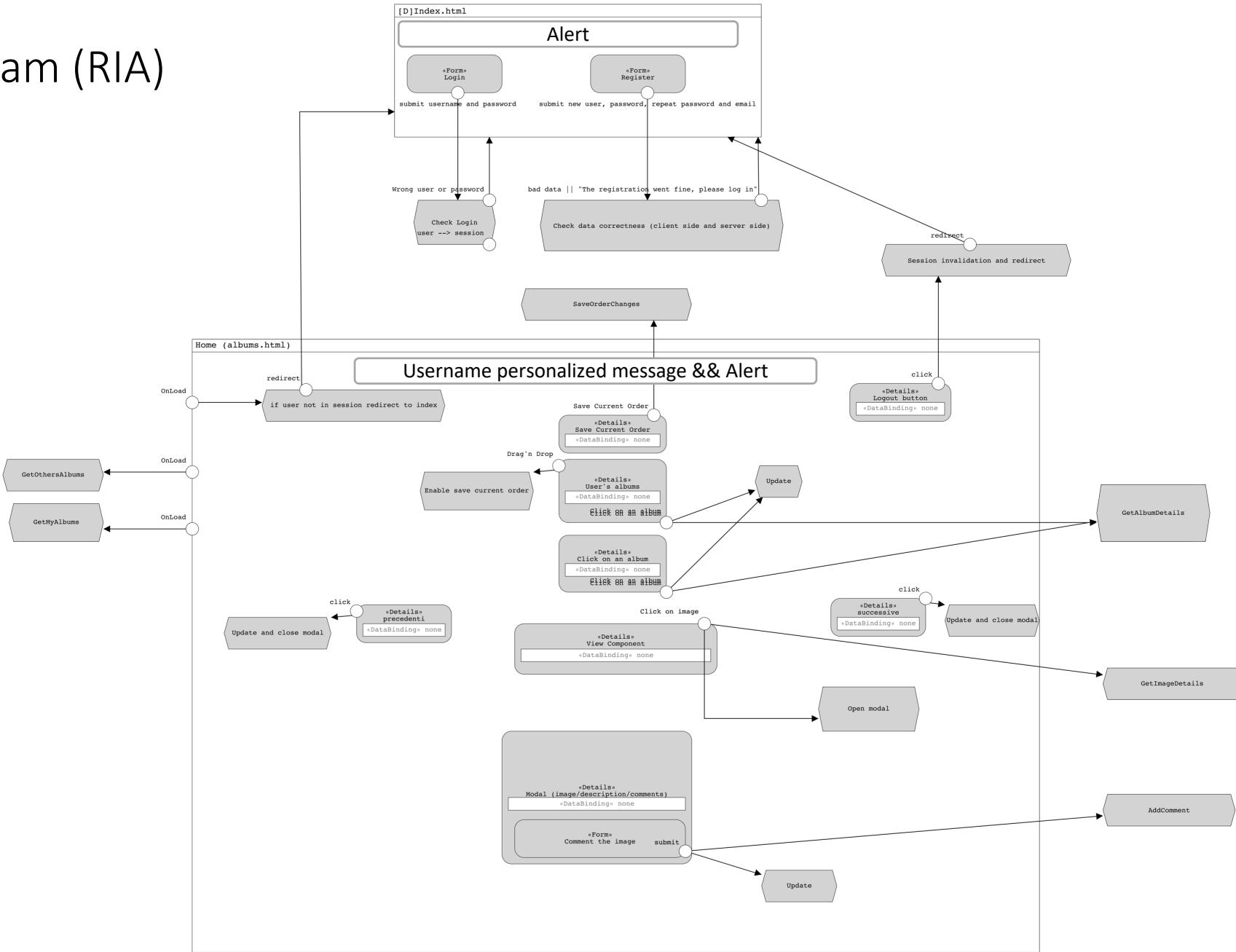
Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- 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 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.
- 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 (**drag and drop**) 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.

Riuso del codice

- Stesso database, aggiunta la tabella order per memorizzare dei path di file dove verranno salvati gli ordinamenti personalizzati degli utenti.
- Stessi DAO, aggiunto OrderDAO.
- Molte delle servlet controllers riadattate facilmente rimuovendo i riferimenti a thymeleaf e inviando solo dati formattati in json.
- Stessa pagina di login/registration riadattata semplicemente con uno script javascript (si trova in /js/index.js).

IFML diagram (RIA)



Update == local view update

Componenti del codice diverse rispetto alla versione HTML

- **Model objects (Beans)**
 - Order
- **Data Access Objects (Classes)**
 - **OrderDAO**
 - addOrder(int user_id, String filepath)
 - getOrder(int user_id)

Controllers

- GetAlbumsDetails
- GetImageDetails
- GetMyAlbums
- GetOtherAlbums
- SaveOrderChanges
- AddComment
- CheckLogin, Logout, Registration

Alcuni screenshot dimostrativi (Versione RIA)

(la login page è visualmente identica alla versione HTML, pertanto viene omessa)

Screenshot of the "Your Albums" section of the application. The title "Pippo 's Home" is at the top. Below it, a heading "Your Albums:" is followed by a "Logout" link. A button labeled "Save Current Order" is present. A note says "You can freely drag and drop the albums in the stack to any position".

2022 Jun 05	Macchine
1998 Oct 10	Motori

Below this, a heading "Other's Albums:" is shown:

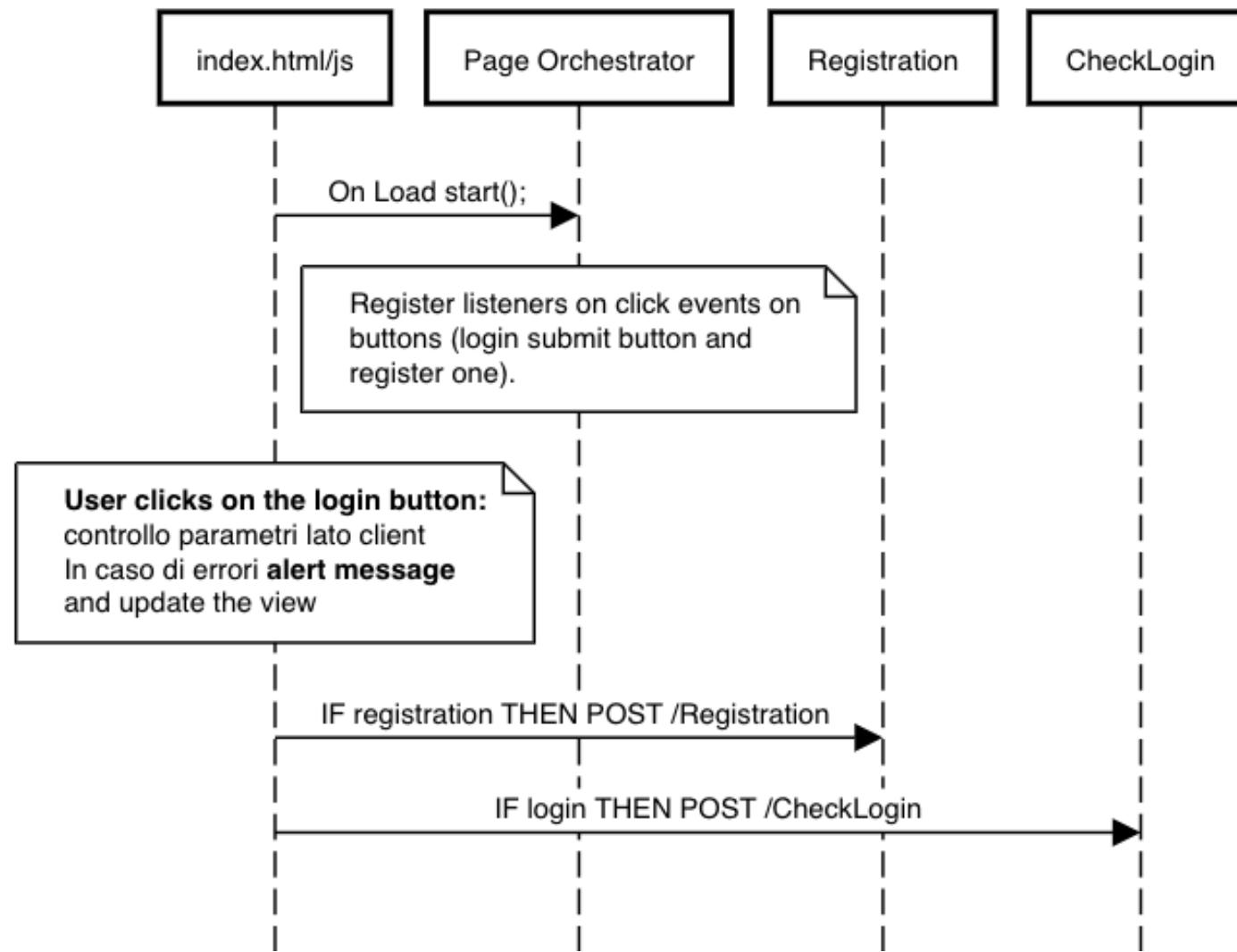
2022 May 10	Barca
2022 May 02	Ciao
1998 Oct 10	Sport

Screenshot of the "500" album view. At the top, there is a grid of five car models: Panda, 500, Ypsilon, Renegade, and 500X. Below this, a large image of a yellow Fiat 500 is displayed. To the right of the image, there is a "Descrizione" (Description) section with a "Comment" input field and a "Submit" button. Below this, several comments are listed in grey boxes:

- Pippo
- Questa macchina è fantastica
- Pippo
- Troppo gialla!!

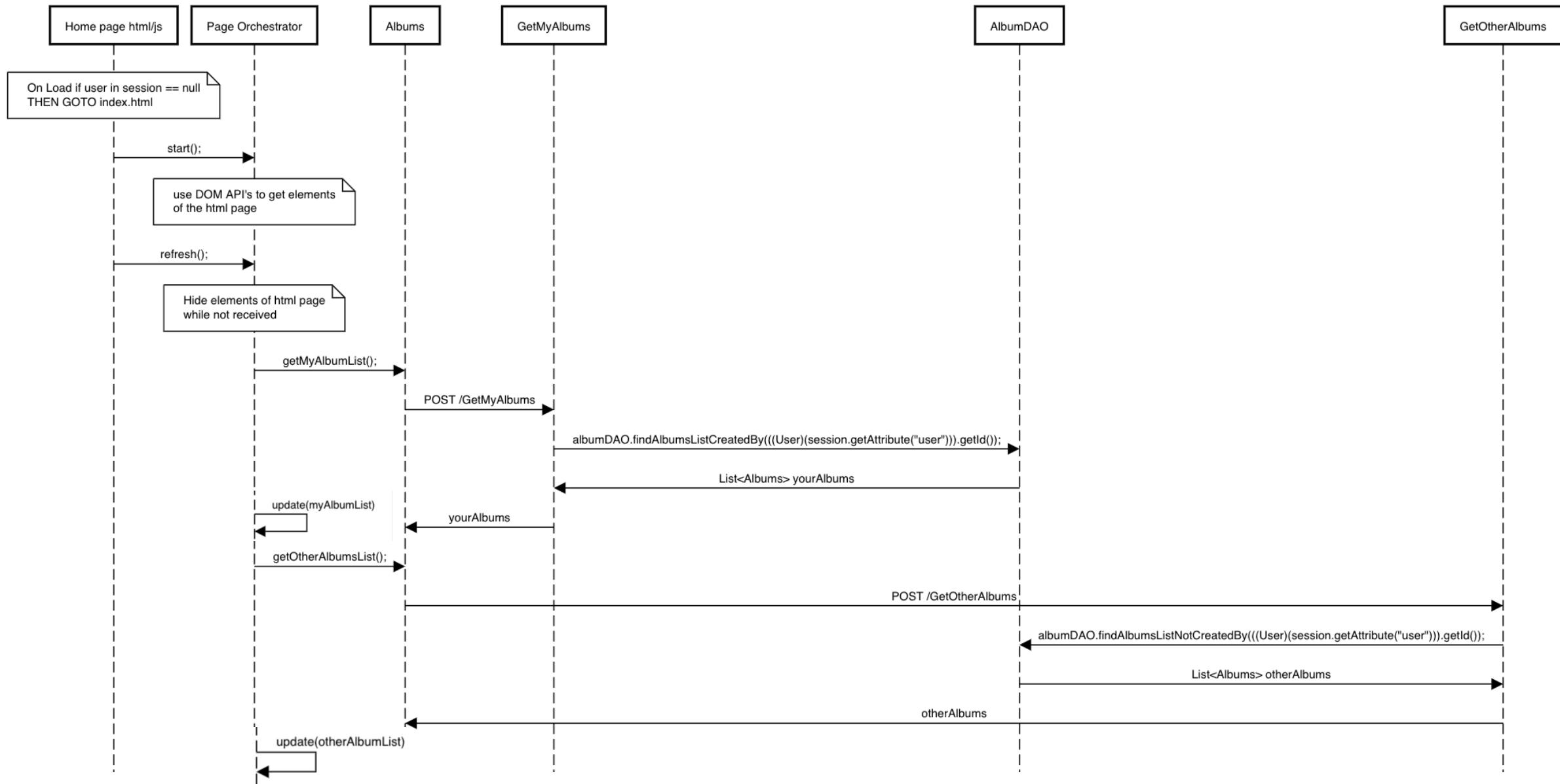
Sequence Diagrams

Login/Registration



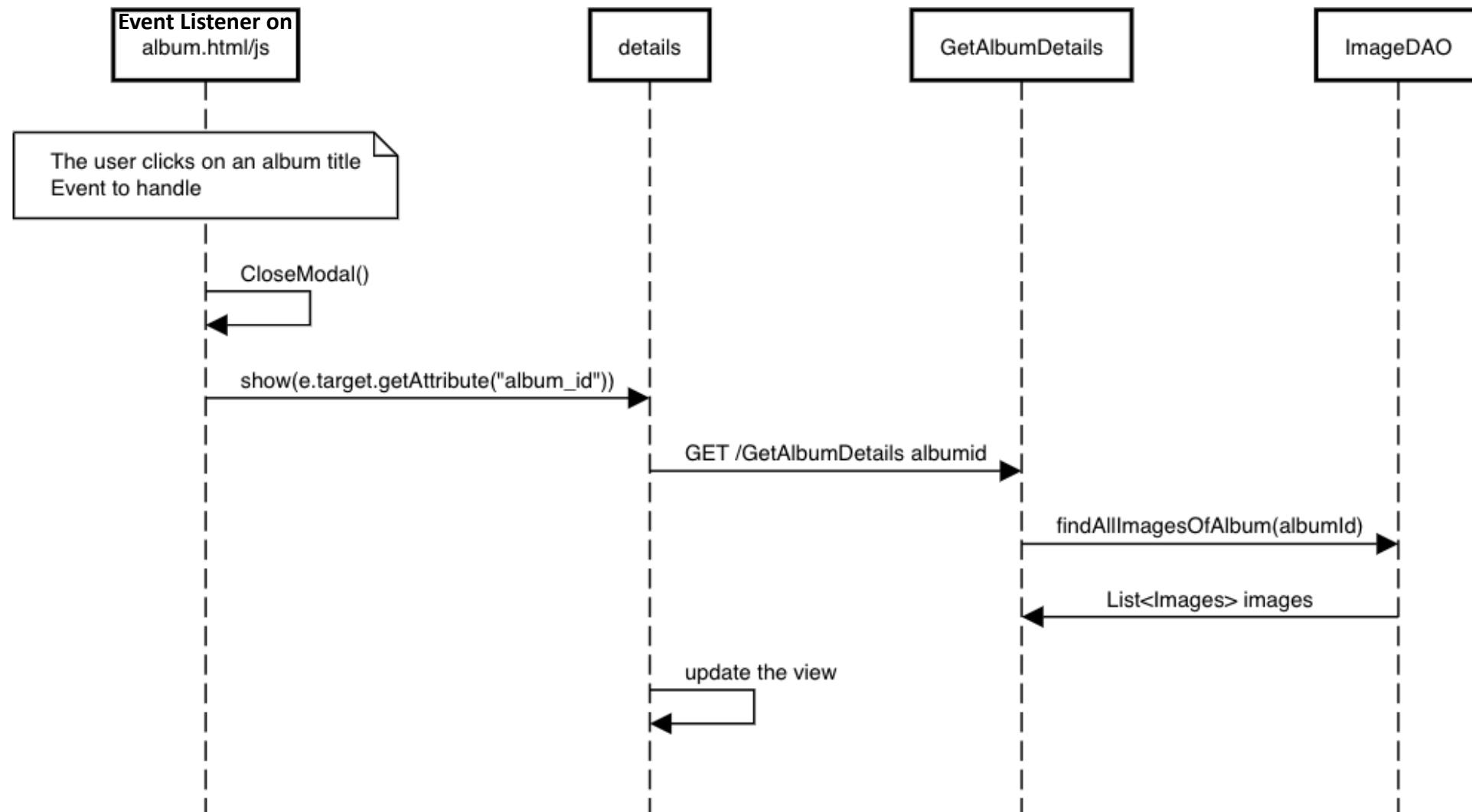
Sequence Diagrams

Home page Loading



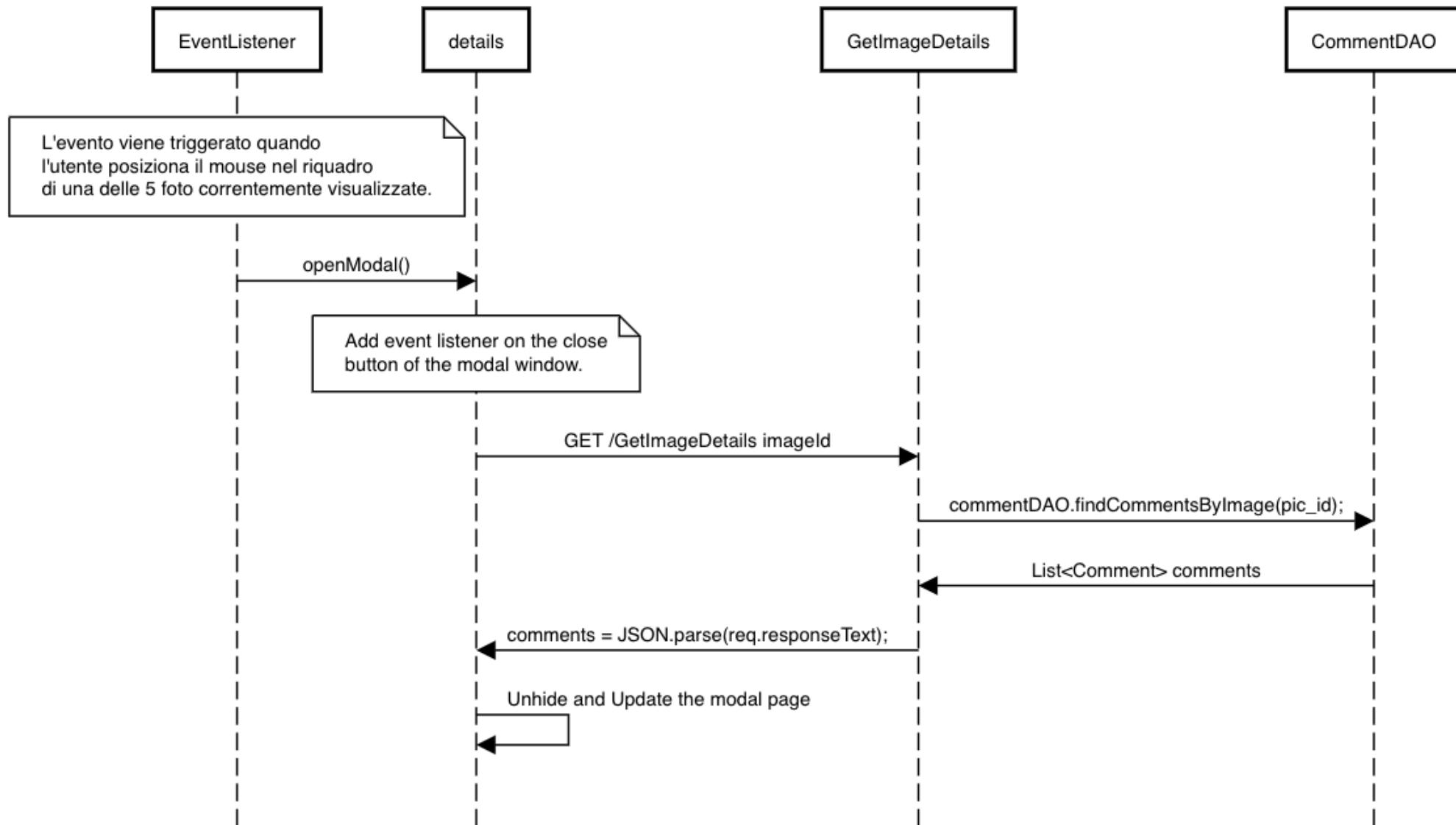
Sequence Diagrams

Album selection



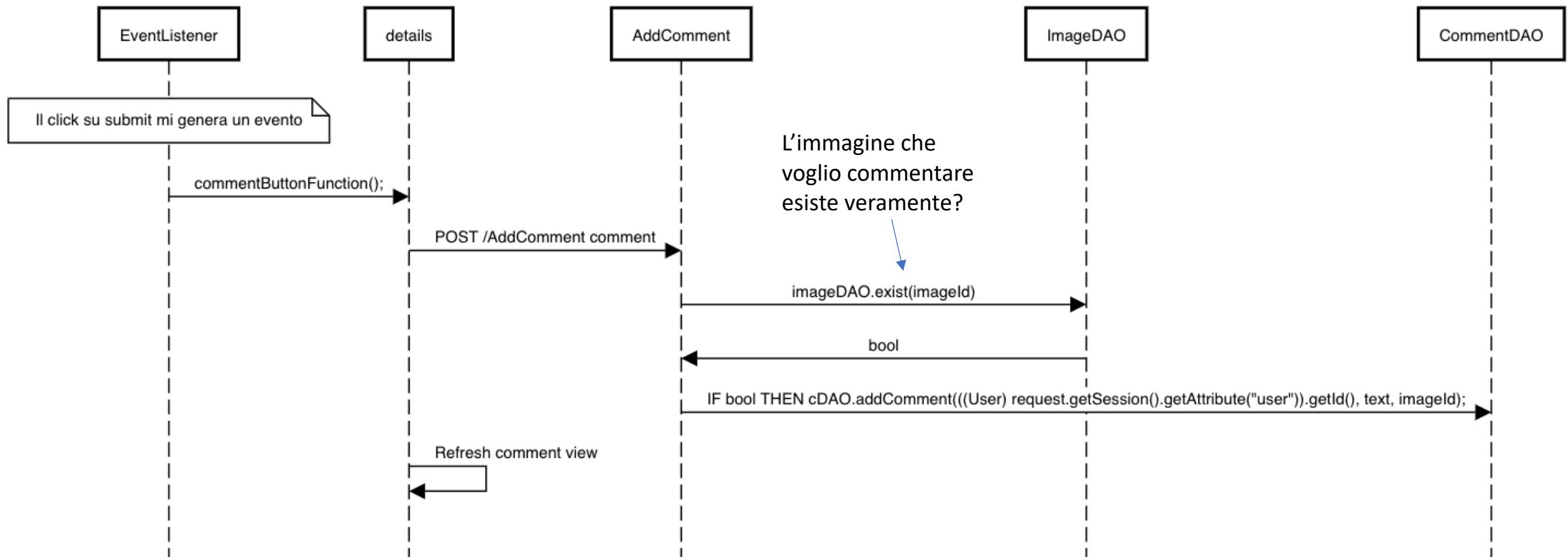
Sequence Diagrams

Mouse over image Event



Sequence Diagrams

Comment creation



L'error handling con i messaggi di allerta/pagine di errore è stato omesso dai sequence diagrams per concisione, ma è presente nel codice.