Tecnologie informatiche per il web

Progetto 1 - Galleria Immagini

Moretti Lorenzo - 10671362
Mormile Matteo - 10666565

17 Giugno, 2022

Indice

1 Versione PureHTML

1.1 Progettazione della base di dati

1.1.1 Analisi del testo

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`

1.1.2 Completamento delle specifiche della base di dati

- Si è scelto di dare la possibilità all'utente di poter caricare la **stessa foto in più album.**
- Per poter rendere la base di dati univoca per il progetto RIA e PureHTML si è gia introdotto l'attributo **sorting** che servirà successivamente per la gestione dell'ordinamento personalizzato dei proprio albums

1.1.3 Diagramma E/R della base di dati

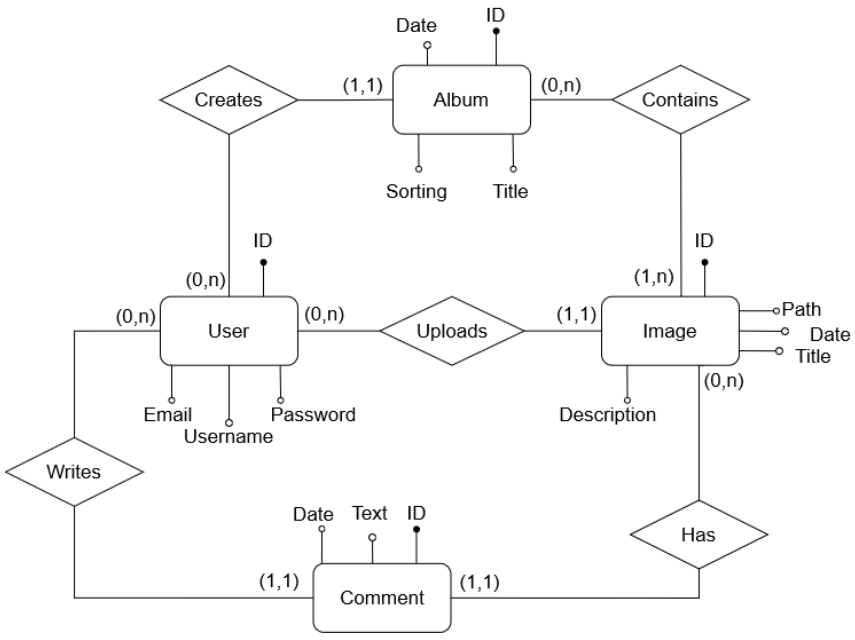

Figura 1: Schema del datatabase

1.1.4 Schema logico della base di dati

CREATE TABLE ‘user‘ (
‘id‘ int NOT NULL PRIMARY KEY AUTO_INCREMENT,
‘ username‘ varchar(255) NOT NULL,
‘password‘ varchar(255) NOT NULL,
‘email‘ varchar(255) NOT NULL)

CREATE TABLE ‘image‘ (
‘id‘ int NOT NULL PRIMARY KEY AUTO_INCREMENT,
‘path‘ varchar(255) NOT NULL,
‘description‘ varchar(255) NOT NULL,
‘title‘ varchar(255) NOT NULL,
‘date‘ datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
‘user‘ int NOT NULL,
CONSTRAINT ‘fk_user_image‘ FOREIGN KEY (‘user‘) REFERENCES ‘user‘ (‘id‘) ON DELETE CASCADE)

CREATE TABLE ‘comment‘ (
‘id‘ int NOT NULL PRIMARY KEY AUTO_INCREMENT,
‘text‘ varchar(255) NOT NULL,
‘date‘ datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
‘image‘ int NOT NULL,
‘user‘ int NOT NULL,
CONSTRAINT ‘fk_image_comment‘ FOREIGN KEY (‘image‘) REFERENCES ‘image‘ (‘id‘) ON DELETE CASCADE,
CONSTRAINT ‘fk_user_comment‘ FOREIGN KEY (‘user‘) REFERENCES ‘user‘ (‘id‘) ON DELETE CASCADE)

CREATE TABLE ‘album‘ (
‘id‘ int NOT NULL PRIMARY KEY AUTO_INCREMENT,
‘title‘ varchar(255) NOT NULL,
‘date‘ datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
‘user‘ int NOT NULL,
‘sorting‘ int,
CONSTRAINT ‘fk_user_album‘ FOREIGN KEY (‘user‘) REFERENCES ‘user‘ (‘id‘) ON DELETE CASCADE)

CREATE TABLE ‘albumimages‘ (
‘id‘ int NOT NULL PRIMARY KEY AUTO_INCREMENT,
‘image‘ int NOT NULL, ‘album‘ int NOT NULL,
CONSTRAINT ‘fk_image‘ FOREIGN KEY (‘image‘) REFERENCES ‘image‘ (‘id‘) ON DELETE CASCADE,
CONSTRAINT ‘fk_album‘ FOREIGN KEY (‘album‘) REFERENCES ‘album‘ (‘id‘) ON DELETE CASCADE)

1.2 Definizione dei requisiti applicativi

1.2.1 Analisi del testo

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. 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`

1.2.2 Completamento delle specifiche applicative

- Creazione della filter **LoggedChecker** per il controllo di accesso autorizzato alle pagine che necessitano l'autenticazione
- Creazione di una pagina di supporto **error.html** per la visualizzazione dei messaggi di errore generati dal server
- Utilizzo della servlet **GetImage** per la visualizzazione delle immagini memorizzate al di fuori del deploy path.

- Creazione delle servlets **CreateAlbum e CreateImage** per la creazione di album e il carimento di immagini.

1.2.3 Application design

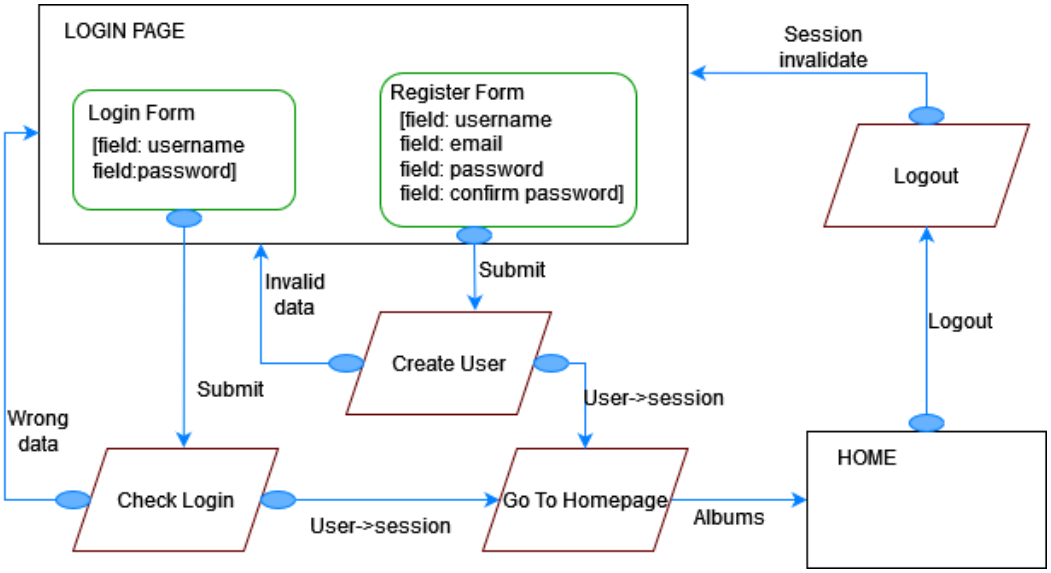

Figura 2: Application design della pagina di login

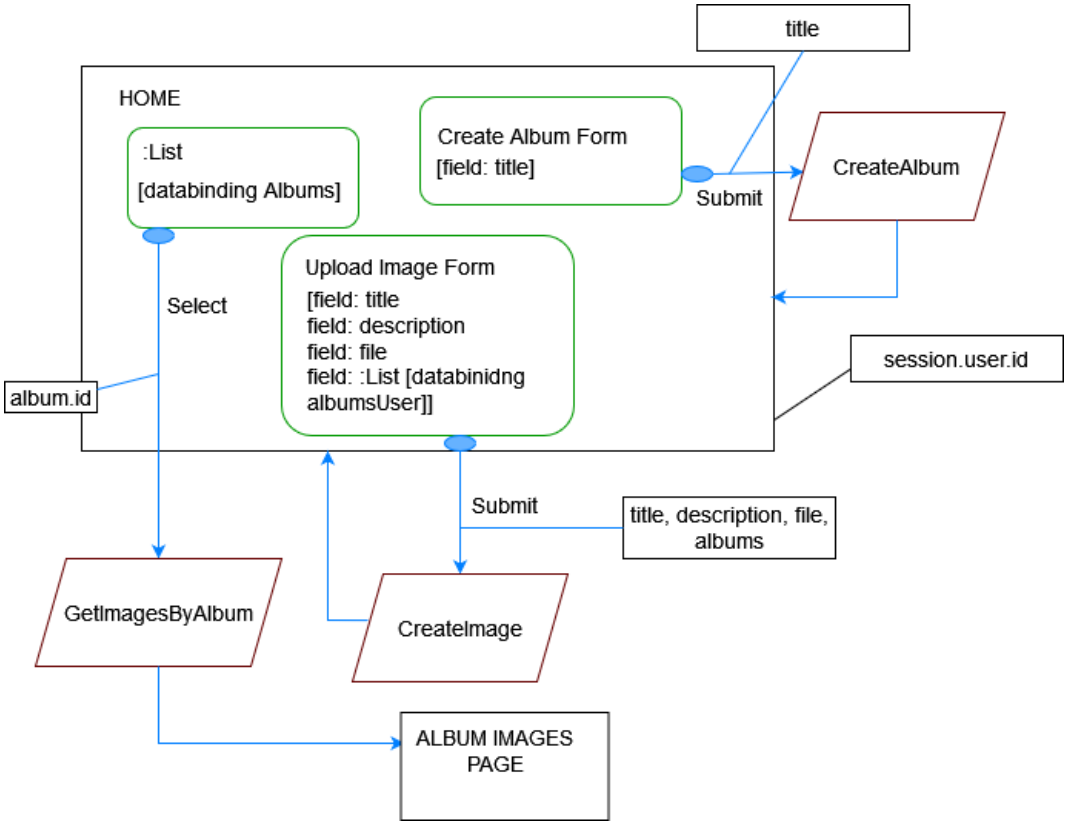

Figura 3: Application design della home page

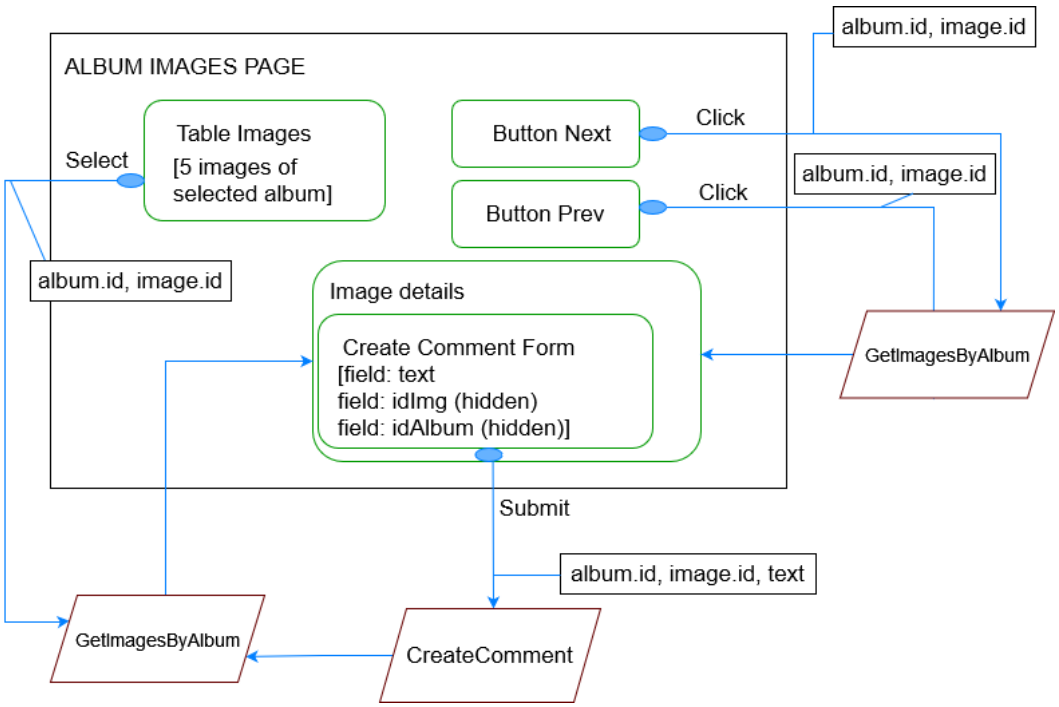

Figura 4: Application design della pagina album

1.2.4 Componenti del progetto

Model Object (Beans)

- Album
- Comment
- Image
- User

Data Access Object (Classes)

AlbumDAO
- getAlbumByID(int id)
- getAlbumsByUserID(int id)
- getAlbumsByNotUserID(int id)
- createAlbum(String title, int idUser)

AlbumImagesDAO
- addImageToAlbum(int imgId, int albumId)
- checkImageInAlbum(int imgId,int albumId)

CommentDAO
- getCommentsFromImages(int idImg)
- createComment(int idImg, String comment, int idUser)

ImageDAO
- getImagesFromAlbum(int idAlbum)
- getImageByID(int id)
- createImage(String path, String description,String title,int idUser)

UserDAO
- checkLogin(String username, String email, String password)
- isMailAvailable(String mail)
- isUsernameAvailable(String username)
- registerUser(String username, String email, String password)

Controllers (Servlets)

- CheckLogin
- CreateAlbum
- CreateComment
- CreateImage
- CreateUser
- GetImagesByAlbum
- GoToHomePage
- Logout
- GetImage*

Views (Templates)

- Index
- Home
- Album_images

1.3 Sequence diagram

1.3.1 Evento di login

Il **submit del form di login** genera una chiamata POST alla servlet **CheckLogin** che dopo aver controllato le credenziali tramite interrogazione al database, salva in sessione lo username e reindirizza alla servlet **GoToHomePage.**

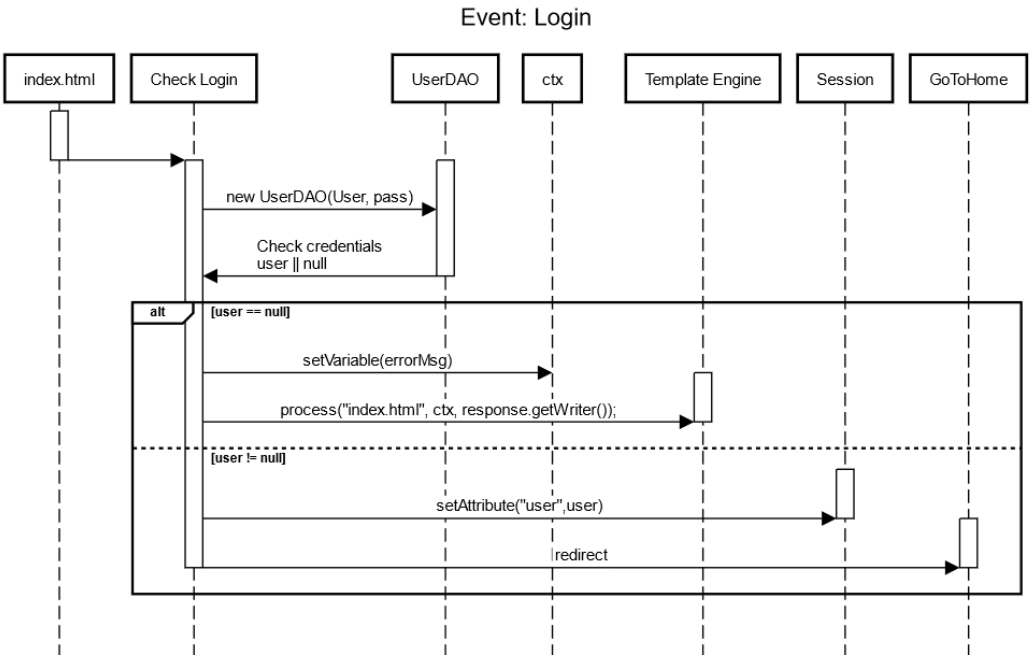

Figura 5: Sequence diagram evento login

1.3.2 Visualizza HomePage

Il metodo GET della servlet **GoToHomePage** si occupa di recuperare dal database gli albums dell'utente e degli altri per poi passarli al template engine che restituirà all'utente la pagina home.html

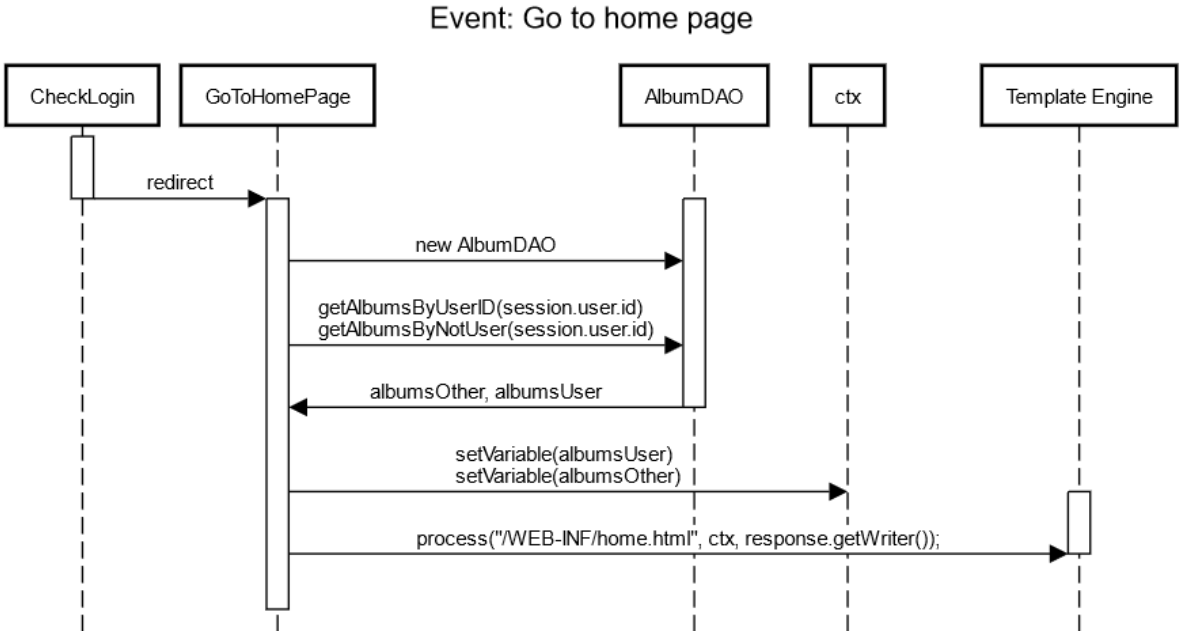

Figura 6: Sequence diagram del caricamento della homepage

1.3.3 Creazione album

Il **submit del form di creazione di un album** genera una chiamata POST alla servlet **CreateAlbum** che dopo aver controllato la validità del titolo, aggiunge al database il nuovo album e ritorna in homepage.

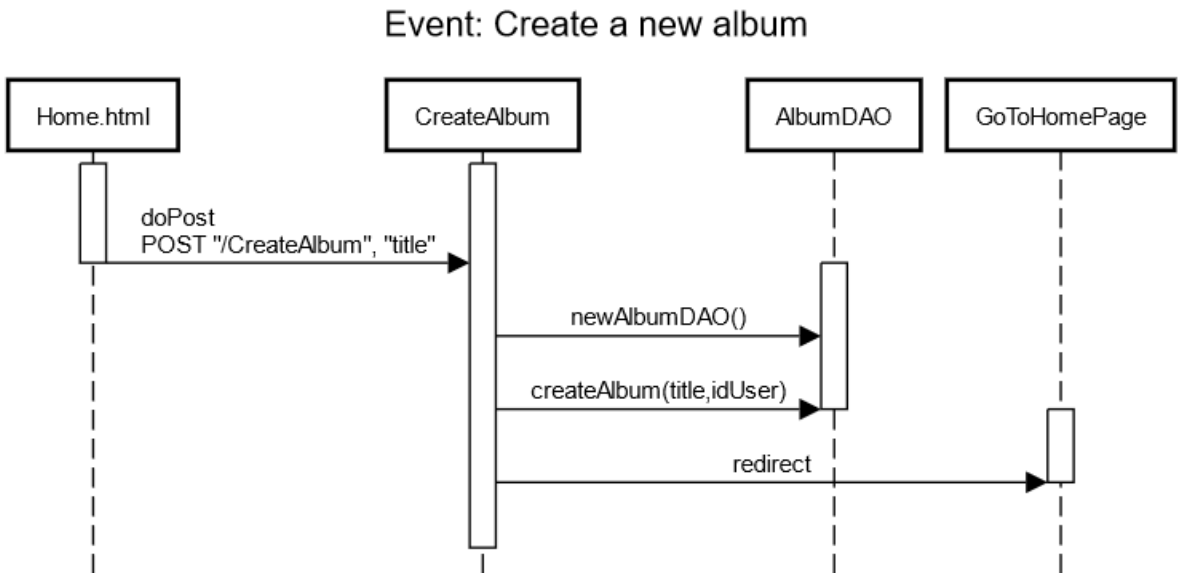

Figura 7: Sequence diagram della creazione di un album

1.3.4 Caricamento immagine

Il **submit del form di caricamento di una immagine** genera una chiamata POST alla servlet **CreateImage** che controlla la validità del titolo, della descrizione e del file. Controlla poi per gli album selezionati che appartengano correttamente all'utente. In caso positivo, crea l'immagine nel database, copia in locale l'immagine e aggiunge le relazioni tra immagine e albums selezionati. Reindirizza alla servlet **GoToHomePage** per mostrare la home

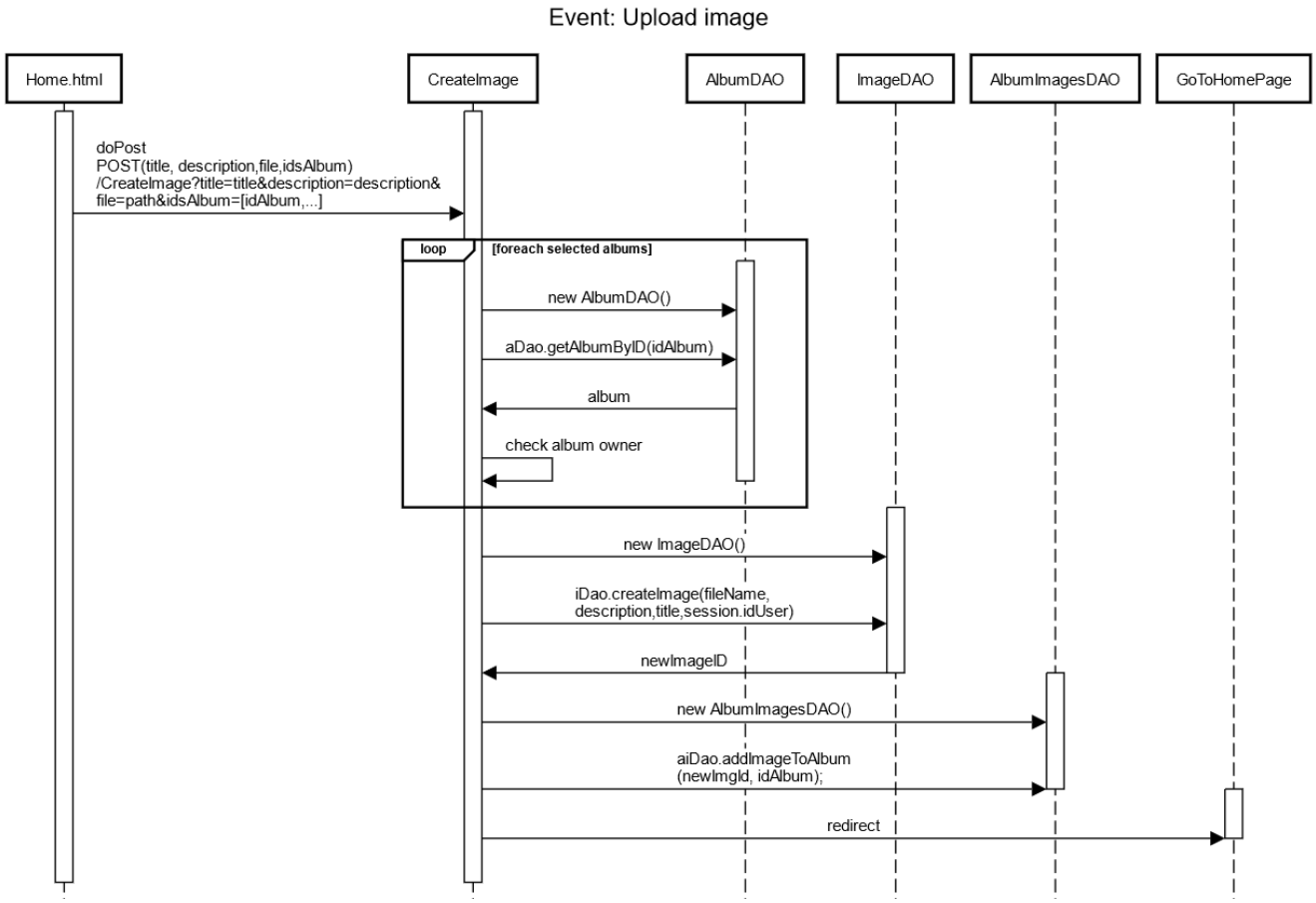

Figura 8: Sequence diagram del caricamento di una foto

1.3.5 Mostra immagini di un album

Il **click sul bottone show di un album**, effettua una chiamata GET alla servlet **GetImagesByAlbum** che si occupa di recuperare l'elenco di immagini dalla base di dati, prendere le prime 5 (o il numero massimo se inferiore) e mostrare la prima con i relativi dettagli. Si occupa anche di configurare i pulsanti per lo scorrimento della lista. Viene poi chiamato il Template Engine per la visulizzazione della pagina album_images.html

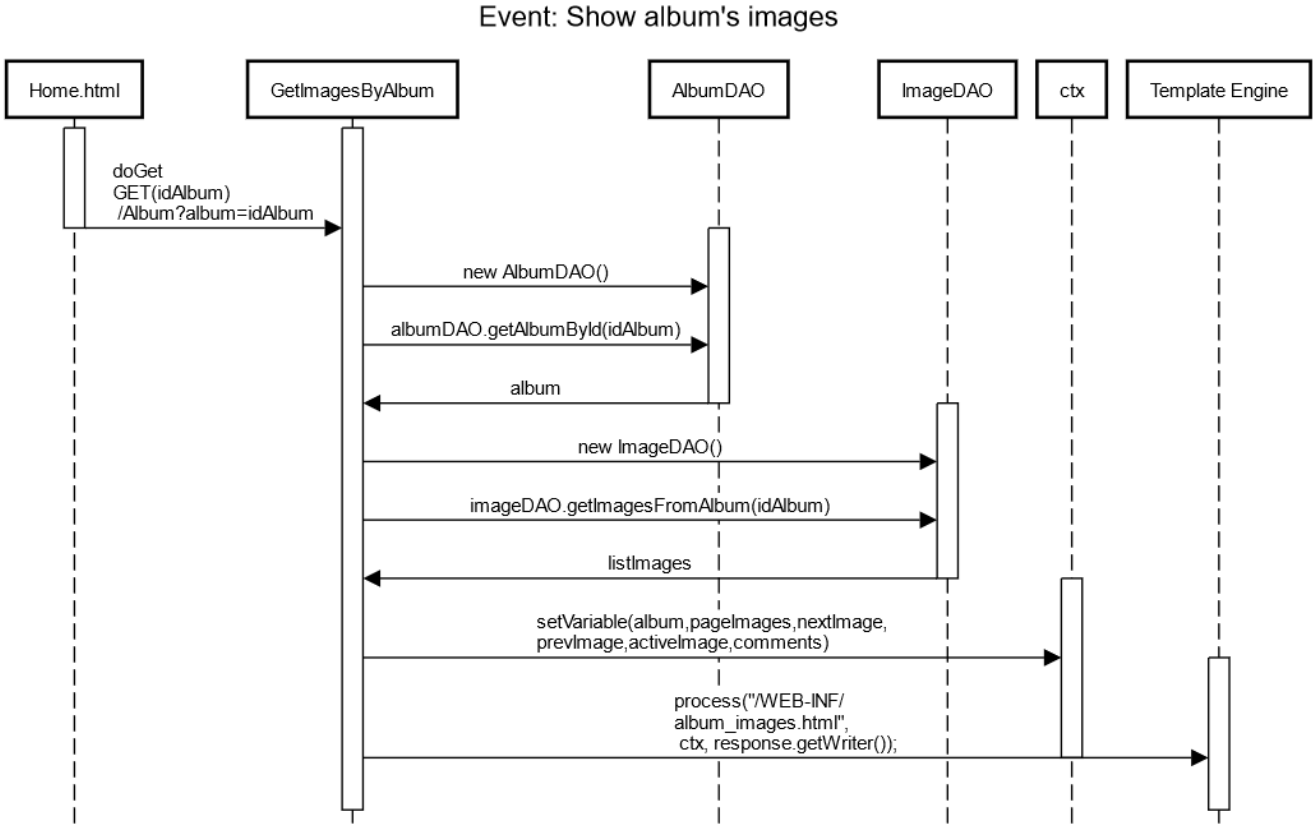

Figura 9: Sequence diagram della visualizzazione di un album

1.3.6 Mostra dettagli immagine di un album

Il **click sulla miniatura** di una immagine, effettua una chiamata GET alla servlet **GetImagesByAlbum** che si occupa di recuperare l'elenco di immagini dalla base di dati e prendere il blocco di 5 immagini (o il numero massimo se inferiore) di cui fa parte quella selezionata e mostrare la selezionata con i relativi dettagli. Si occupa anche di configurare i pulsanti per lo scorrimento della lista. Viene poi chiamato il Template Engine per la visulizzazione della pagina album_images.html

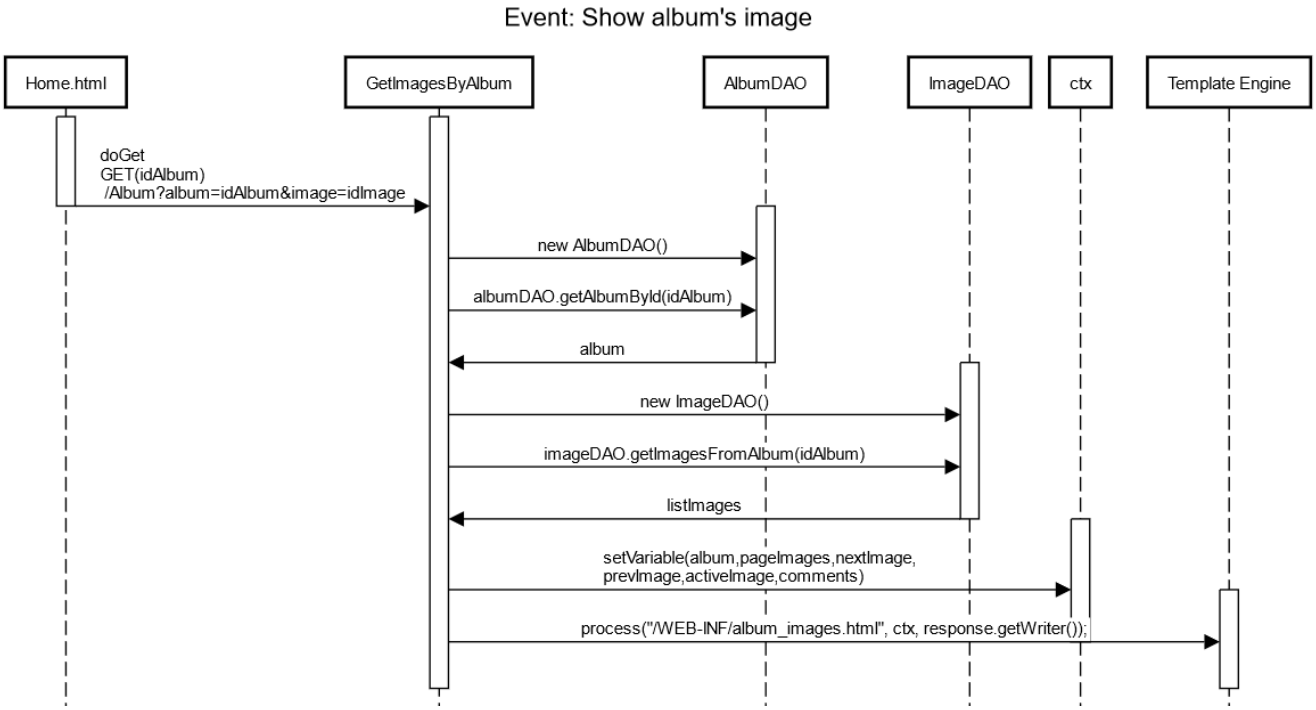

Figura 10: Sequence diagram della visualizzazione di una immagine

1.3.7 Mostra blocco di immagini successivo/precedente

Il **click sulle frecce di navigazione**, effettua una chiamata GET alla servlet **GetImagesByAlbum** che si occupa di recuperare l'elenco di immagini dalla base di dati e prendere il blocco di 5 immagini (o il numero massimo se inferiore) di cui fa parte quella collegata alla freccia (la prima successiva al blocco visualizzato o l'ultima del blocco precedente) e mostrare la selezionata con i relativi dettagli. Si occupa anche di configurare i pulsanti per lo scorrimento della lista. Viene poi chiamato il Template Engine per la visulizzazione della pagina album_images.html
Vedi Figura 10 per funzionamento (analogo alla visualizzazione di una immagine)

1.3.8 Creazione di un commento

Il **submit del form di creazione di un commento** genera una chiamata POST alla servlet **CreateComment** che dopo aver controllato la validità del testo, crea nel database il nuovo commento e reindirizza alla servlet **GetImagesByAlbum** passandole l'id dell'album e dell'immagine a cui è stato aggiunto il commento (valori nascosti nella form).

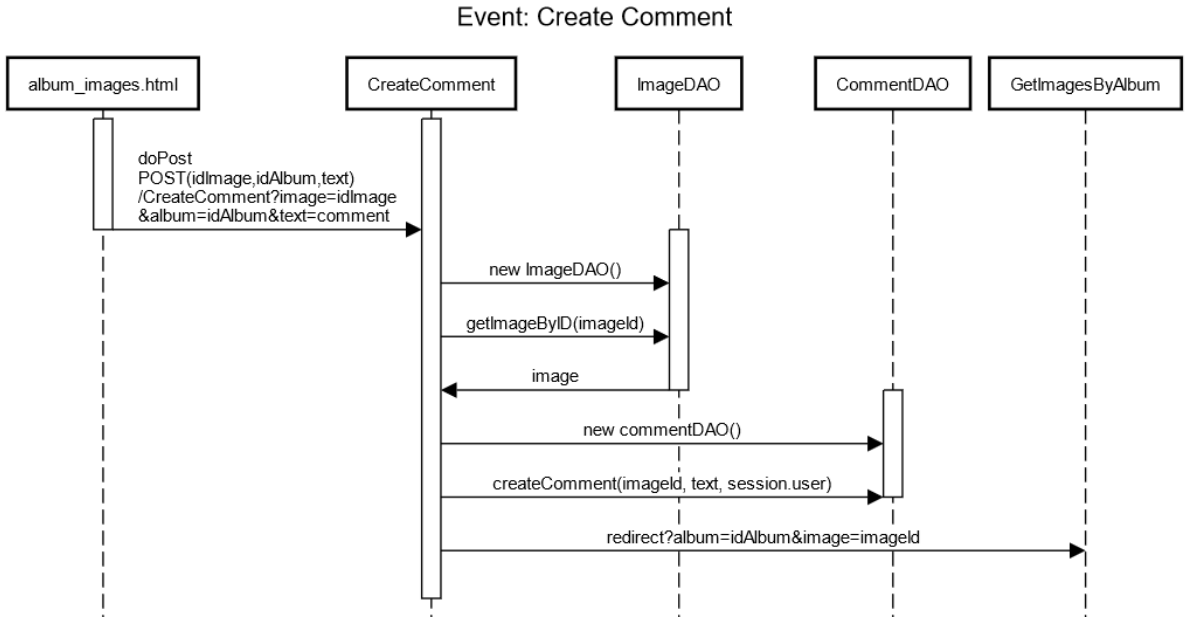

Figura 11: Sequence diagram della creazione di un commento

1.3.9 Filter Login

La filter **LoggedChecker** si occupa di verificare che l'utente sia correttamente autenticato, in caso affermativo continua con l'esecuzione delle servlet a cui è associata altrimenti reindirizza alla pagina di login

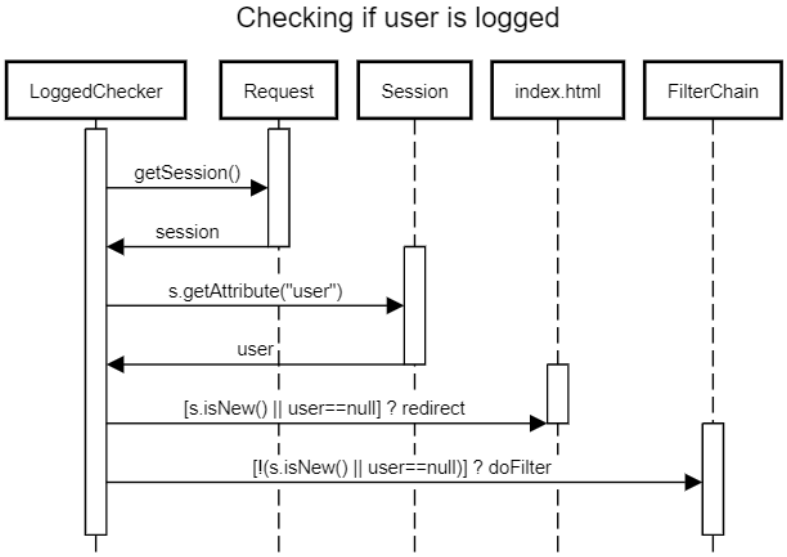

Figura 12: Sequence diagram della filter LoggedChecker