Documentazione Progetto TIW

Matteo Pancini[1] - Samuele Scherini[2]

25 giugno 2022

[1]Cod. Persona: 10656944, Matricola: 932095
[2]Cod. Persona: 10674683, Matricola: 933526

Indice

Parte I

Versione PureHTML

Capitolo 1

Analisi della specifica

1.1 Analisi dei dati

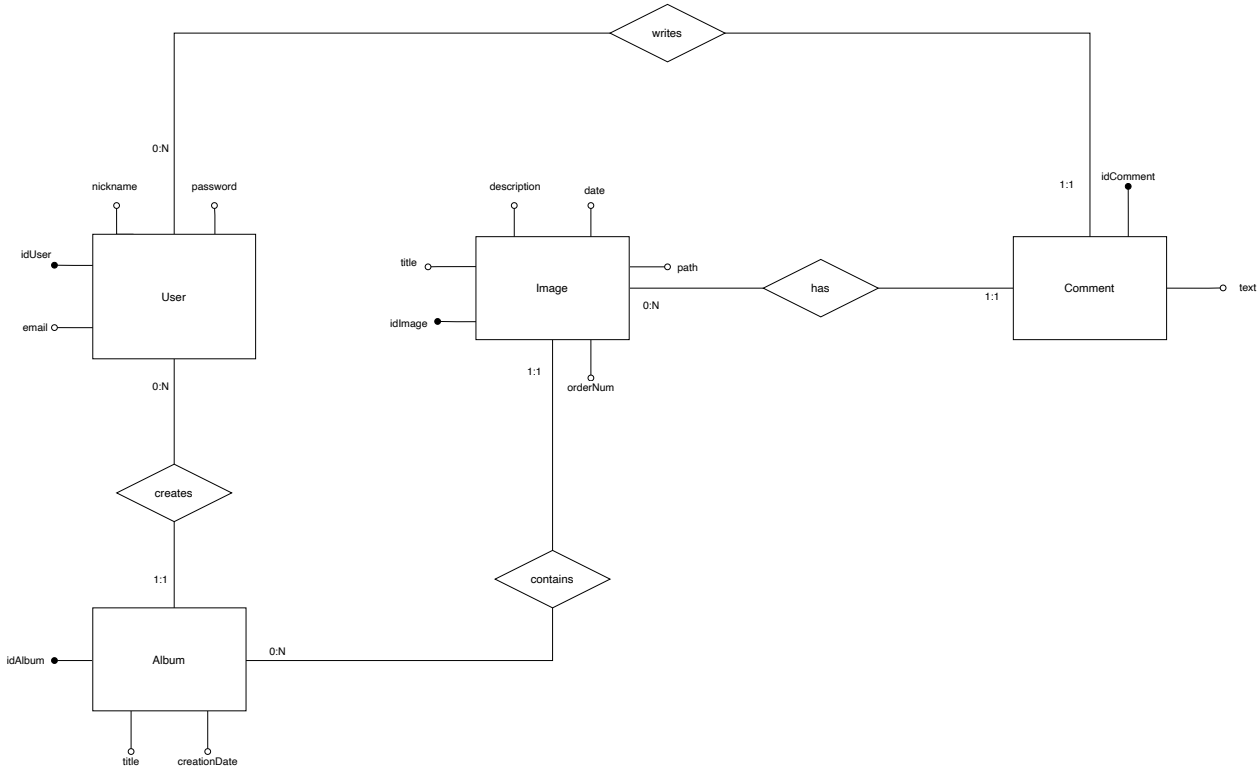
In questa sezione si è svolta l'analisi dei dati del progetto, sottolineando con opportuni colori i contenuti che realizzano il database.

Un'applicazione web consente la gestione di una galleria d'immagini. L'applicazione supporta registrazione (utente) 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.

Legenda:

- entità
- attributi
- relazioni

1.2 Diagramma ER




```
USER (idUser, email, nickname, password)
IMAGE (idImage, idUser, idAlbum description, date, path, orderNum)
ALBUM (idAlbum, idUser, title, creationDate)
COMMENT (idComment, idImage, idUser, text)

IMAGE.idUser -> ALBUM.idUser
IMAGE.idAlbum -> ALBUM.idAlbum
ALBUM.idUser -> USER.idUser
COMMENT.(idImage, idUser) -> IMAGE.(idImage, idUser)
```

Schema del database:

```
CREATE TABLE `user` (
        `iduser` int NOT NULL AUTO_INCREMENT,
        `email` varchar(45) NOT NULL,
        `username` varchar(45) NOT NULL,     `
        password` varchar(45) NOT NULL,
       PRIMARY KEY (`iduser`),
       UNIQUE KEY `nickname_UNIQUE` (`username`)
)

CREATE TABLE `album` (
        `idAlbum` int NOT NULL AUTO_INCREMENT,
        `idUser` int NOT NULL,
        `title` varchar(45) NOT NULL,
        `creationDate` datetime NOT NULL,
       PRIMARY KEY (`idAlbum`),
       CONSTRAINT `userFromUser` FOREIGN KEY (`idUser`) REFERENCES `user` (`iduser
           `) ON UPDATE CASCADE ON DELETE CASCADE
)
```

```
CREATE TABLE `image` (
        `idimage` int NOT NULL AUTO_INCREMENT,
        `idUser` int NOT NULL,
        `idAlbum` int NOT NULL,
        `description` varchar(100) DEFAULT NULL,
        `date` datetime DEFAULT NULL,
        `path` varchar(255) DEFAULT NULL,
        `orderNum` int NOT NULL,
        PRIMARY KEY (`idimage`),
        CONSTRAINT `albumFromAlbum` FOREIGN KEY (`idAlbum`) REFERENCES `album` (`
            idAlbum`) ON UPDATE CASCADE ON DELETE CASCADE,
        CONSTRAINT `userFromAlbum` FOREIGN KEY (`idUser`) REFERENCES `user` (`
            idAlbum`) ON UPDATE CASCADE ON DELETE CASCADE
)

CREATE TABLE `comment` (
        `idComment` int NOT NULL AUTO_INCREMENT,
        `idUser` int NOT NULL,
        `idImage` int NOT NULL,
        `text` varchar(255) NOT NULL,
        PRIMARY KEY (`idComment`),
        CONSTRAINT `imageFromImage` FOREIGN KEY (`idImage`) REFERENCES `image` (`
            idimage`) ON UPDATE CASCADE ON DELETE CASCADE,
        CONSTRAINT `userFromImage` FOREIGN KEY (`idUser`) REFERENCES `image` (`
            idUser`) ON UPDATE CASCADE ON DELETE CASCADE
)
```

1.3 Analisi dei requisti

In questa sezione si è svolta l'analisi dei requisiti del progetto, sottolineando con opportuni colori i contenuti web che realizzano l'applicazione.

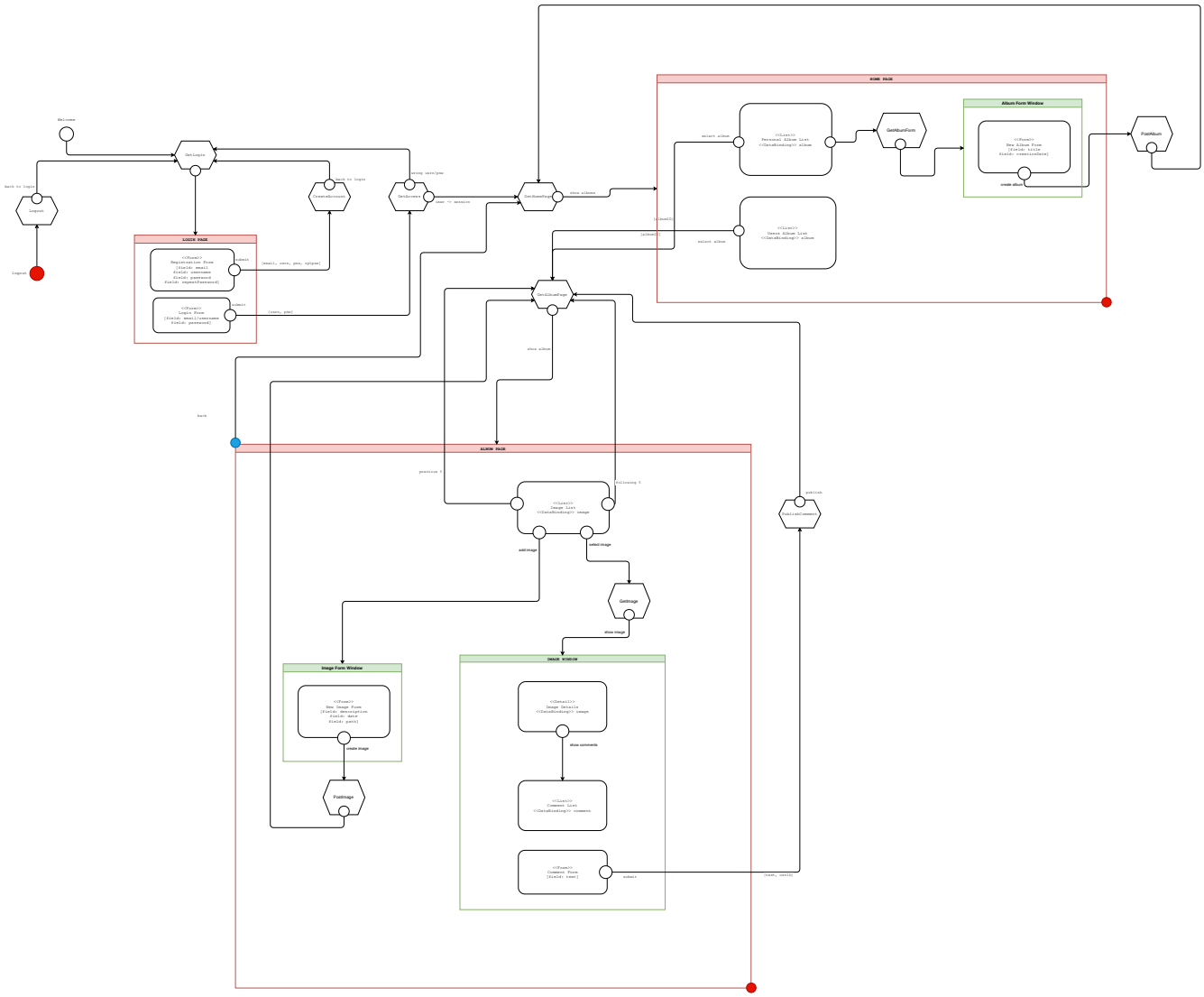
Quando l'utente accede all'HOME PAGE, questa presenta l'elenco degli album che ha creato e l'elenco degli album creati da altri utenti. Entrambi gli elenchi sono ordinati per data di creazione decrescente. Quando l'utente clicca su un album che appare negli elenchi della HOME PAGE, appare la pagina ALBUM 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.

Legenda:

- pagine
- componenti visivi
- eventi
- azioni

Capitolo 2

Flow Diagram



Capitolo 3

Componenti

Si elencano di seguito i componenti del progetto.

Model Objects (Beans)

- User
- Album
- Image
- Comment

Data Access Object (DAO)

- User DAO
  - `checkLoginCredentials(String username, String psw)`
  - `findAllUsernames()`
  - `createUser(String email, String username, String psw)`
  - `getIdFromUsername(String username)`
- Album DAO
  - `findUserAlbums(int idUser)`
  - `findOtherAlbums(int idUser)`
  - `createNewAlbum(int idUser, String title)`
- Image DAO
  - `findAllAlbumImages(int idAlbum)`
  - `createNewImage(int idUser, int albumId, String imageTitle, String description, String imagePath)`
- Comment DAO
  - `findAllComments(int idImage, int idAlbum)`
  - `createNewComment(int idImage, int idAlbum, int idUser, String text)`

Controllers (Servlet)

- `CheckLogin`

- `CreateAccount`
- `GetAlbumPage`
- `GetHomePage`
- `GetImage`
- `Logout`
- `PostAlbum`
- `PostImage`
- `ShowImageDetails`

Views (Templates)

- index
- register
- home
- newalbumform
- albumpage
- newimageform

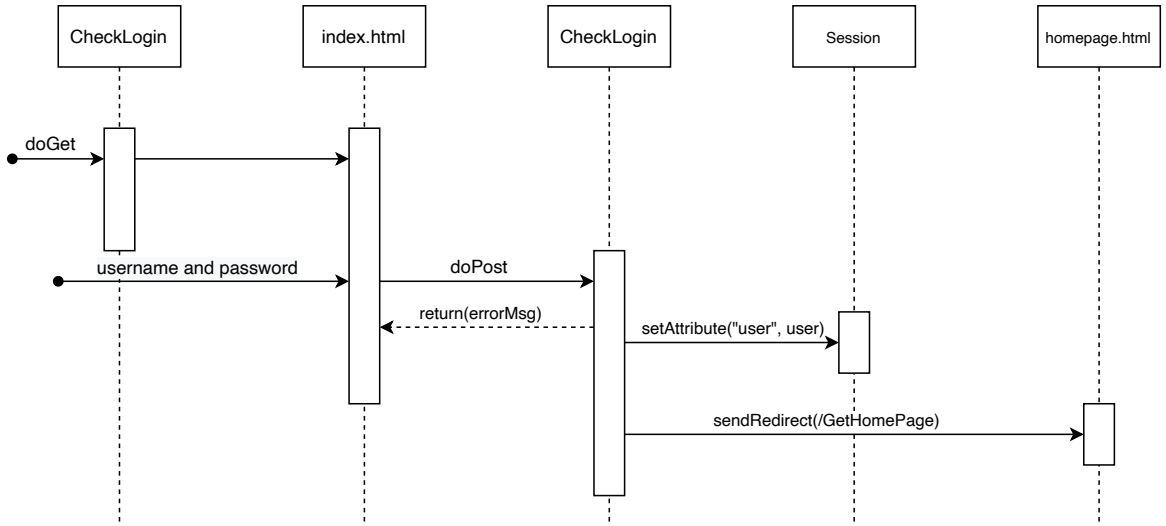
Filters

- `UserChecker`

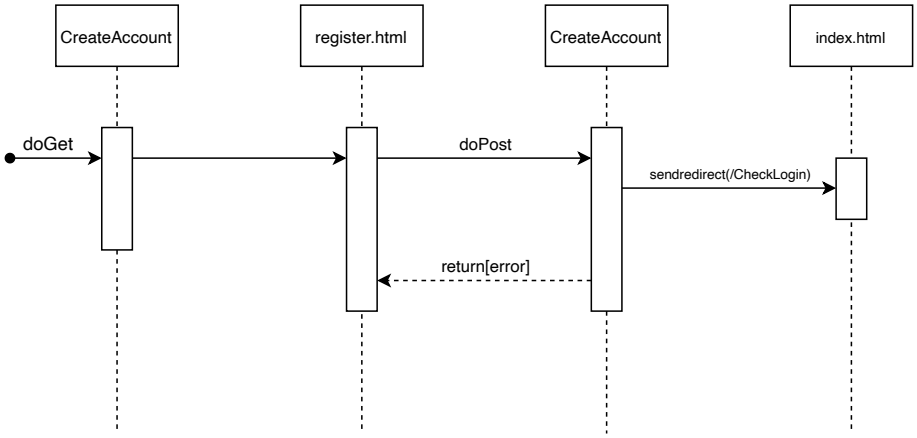
Capitolo 4

Sequence diagrams

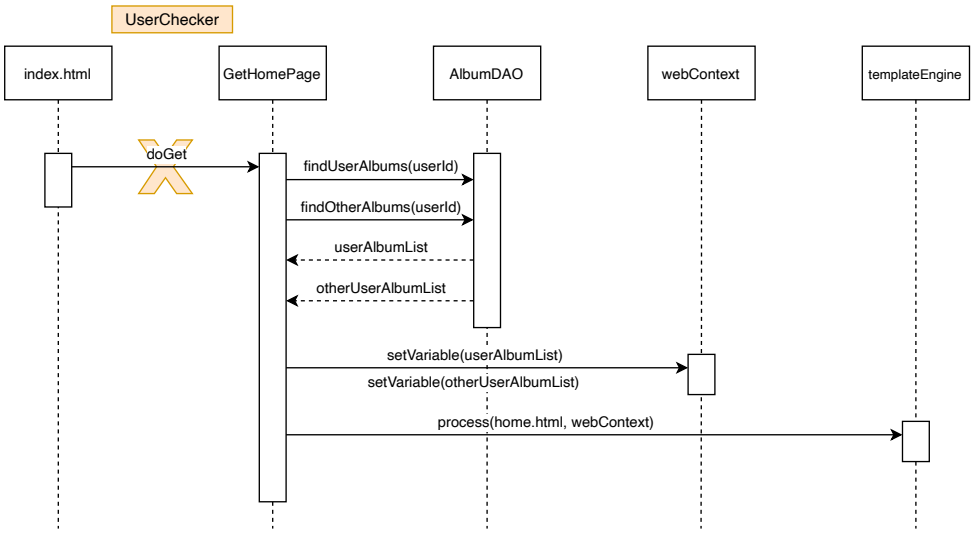
- Login



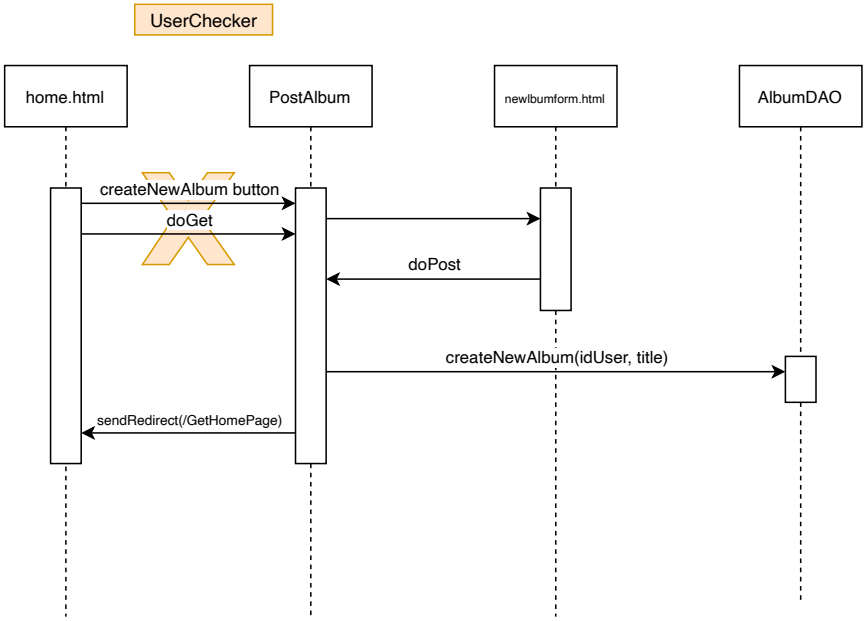
- Register



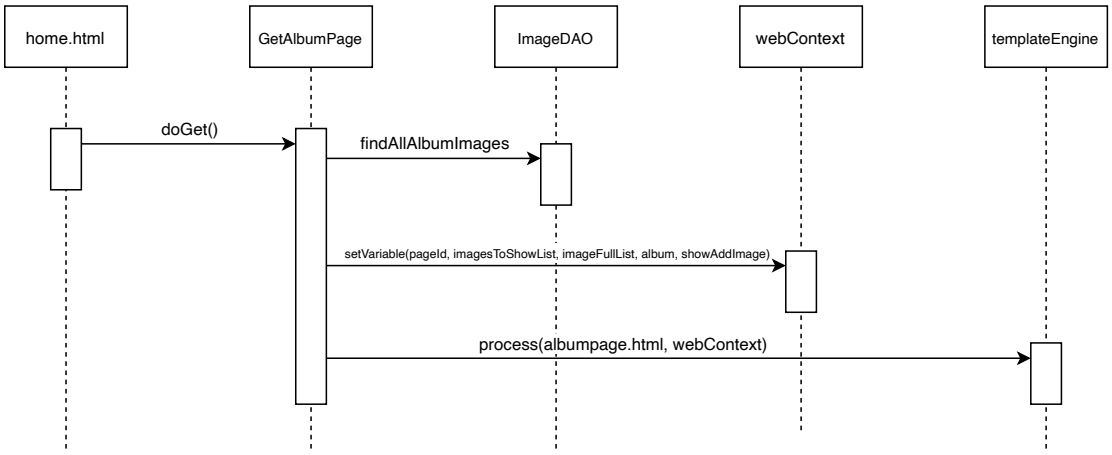
- Go to homepage



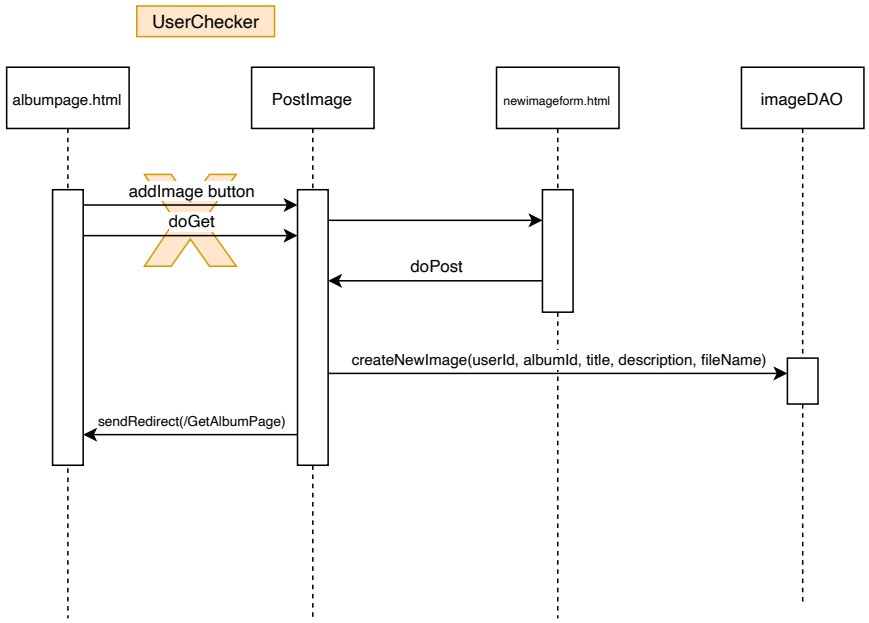
- Create album



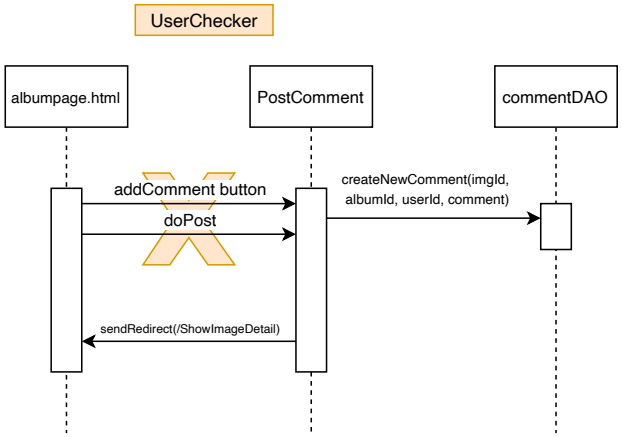
- Go to albumpage



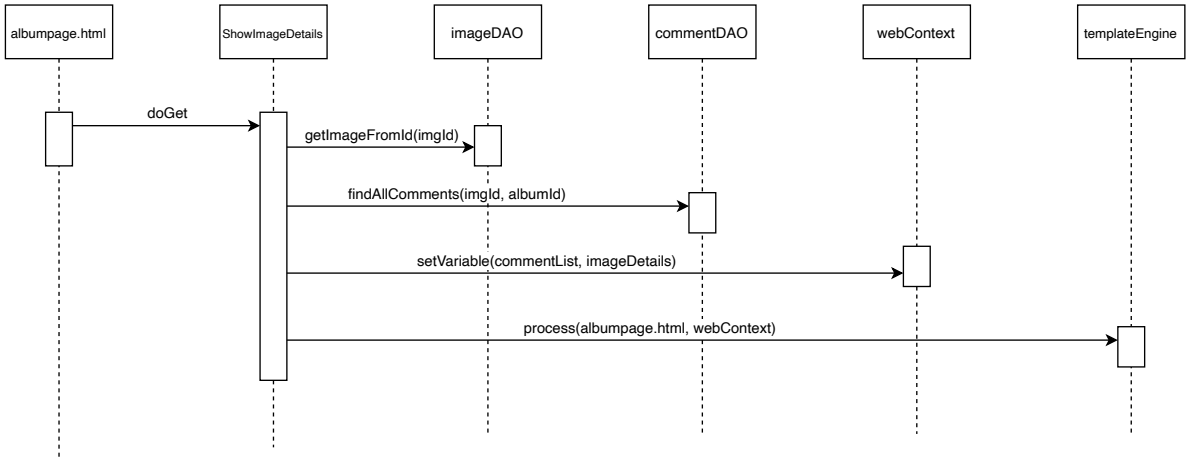
- Add image



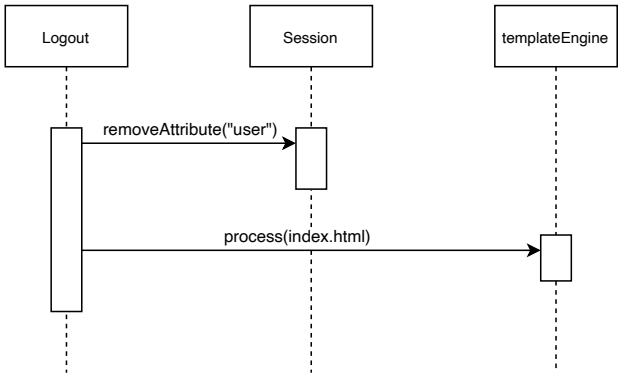
- Add comment



- Show image details



- Logout


Logout
Session
templateEngine
removeAttribute("user")
process(index.html)

Parte II

RIA

Capitolo 5

Analisi della specifica

5.1 Analisi completamento dei requisiti

In questa sezione si è svolta l'analisi del completamento dei requisiti del progetto, sottolineando con opportuni colori i contenuti web che realizzano l'applicazione.

- 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 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.

Legenda:

- pagine
- componenti visivi
- eventi
- azioni

5.2 Analisi eventi & azioni

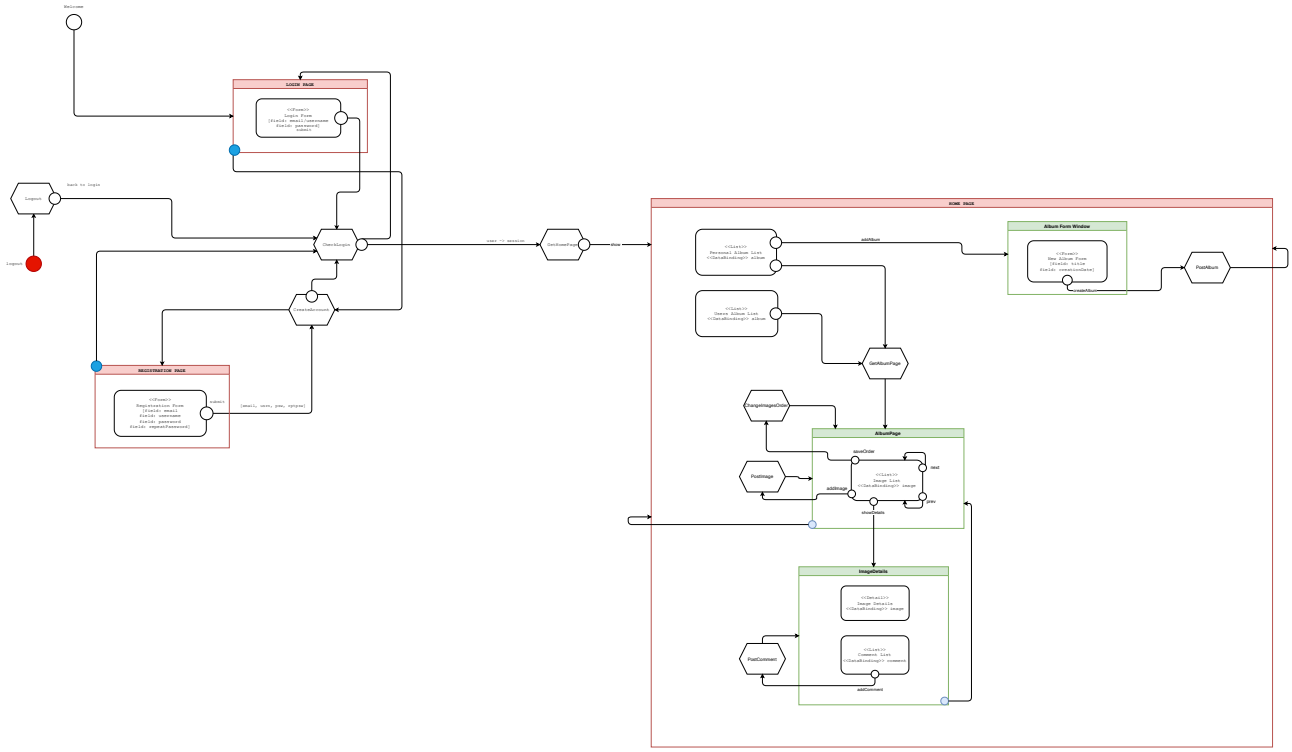
Evento	Azione
index → login form → submit	controllo dati
registration → registration form → submit	controllo dati
homepage → load	aggiorna view con dati
newalbumform → submit	controllo dati album
newimageform → submit	controllo dati image
newimagecomment → submit	controllo dati comment
albumpage → next/prev	cambio dati elenco nella view album
albumpage → save order	cambio ordinamento elenco

5.3 Analisi eventi & controller

Evento	Controller
index → login form → submit	makeCall
homepage → load	PageOrchestrator
newalbumform → submit	makeCall
newimageform → submit	makeCall
newimagecomment → submit	makeCall
albumpage → next/prev	changeImages
homepage → onClick/onMouseHover	makeCall

Capitolo 6

Flow Diagram



Capitolo 7

Componenti

Si elencano di seguito i componenti del progetto.

Model Objects (Beans)

- User
- Album
- Image
- Comment

Data Access Object (DAO)

- User DAO
  - `checkLoginCredentials(String username, String psw)`
  - `findAllUsernames()`
  - `createUser(String email, String username, String psw)`
  - `getIdFromUsername(String username)`
- Album DAO
  - `findUserAlbums(int idUser)`
  - `findOtherAlbums(int idUser)`
  - `createNewAlbum(int idUser, String title)`
- Image DAO
  - `findAllAlbumImages(int idAlbum)`
  - `getImageFromId(int imageId)`
  - `createNewImage(int idUser, int albumId, String imageTitle, String description, String imagePath)`
  - `getImagesPositionForOrdering(int albumId, String minId, String maxId)`
  - `updateOrder(int idImage, int orderNum)`
- Comment DAO
  - `findAllComments(int idImage, int idAlbum)`
  - `createNewComment(int idImage, int idAlbum, int idUser, String text)`

Controllers (Servlet)

- `ChangeImagesOrder`
- `CheckLogin`
- `CreateAccount`
- `GetAlbumPage`
- `GetComments`
- `GetImage`
- `GetOtherAlbums`
- `GetPersonalAlbums`
- `Logout`
- `PostAlbum`
- `PostImage`
- `ShowImageDetails`

Views (Templates)

- index
- register
- home
- create album form (modal window)
- add image form (modal window)
- image details (modal window)
- album details

Filters

- `UserChecker`

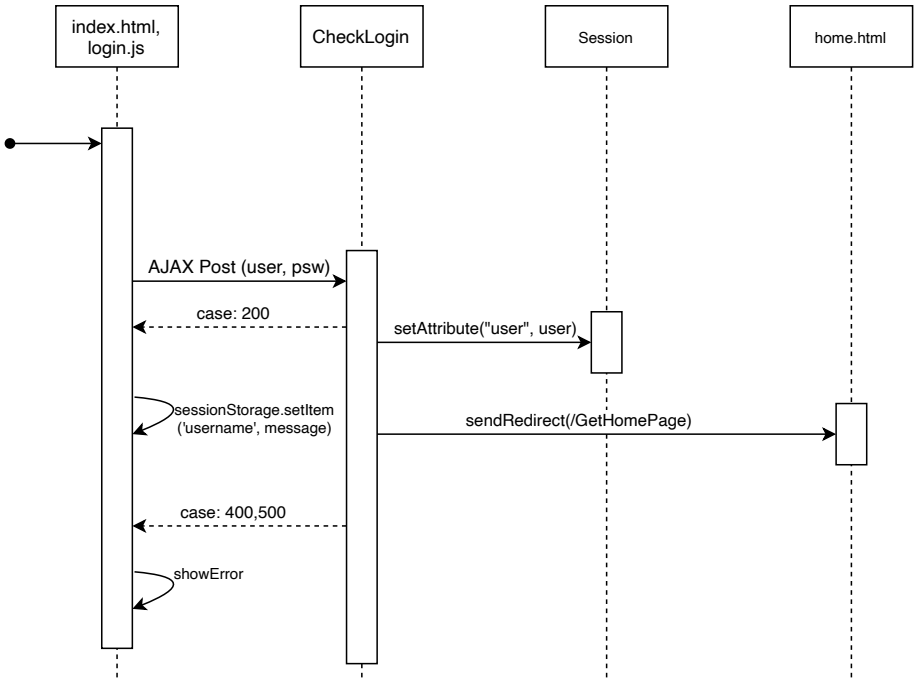
Scripts

- login.js
- register.js
- util.js
- modal.js
- home.js
- tableSorting.js

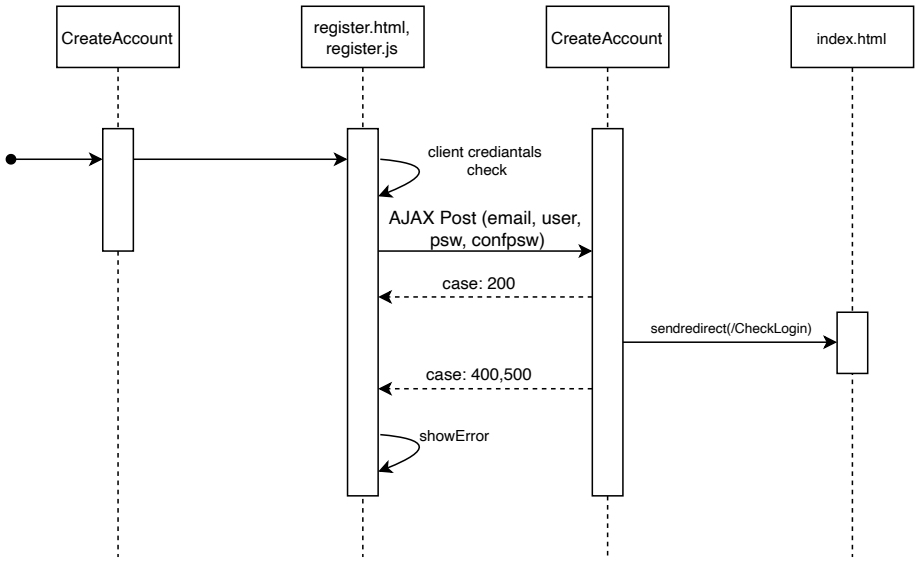
Capitolo 8

Sequence Diagrams

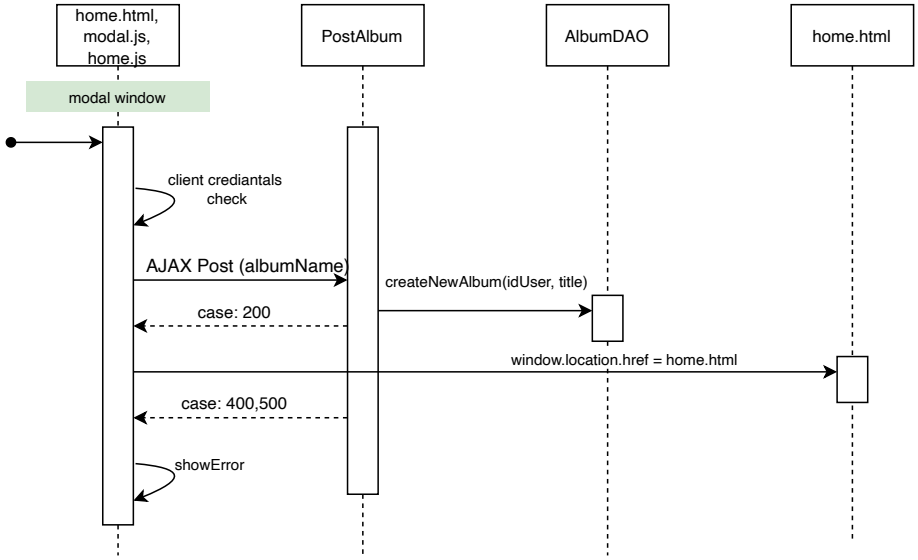
- Login



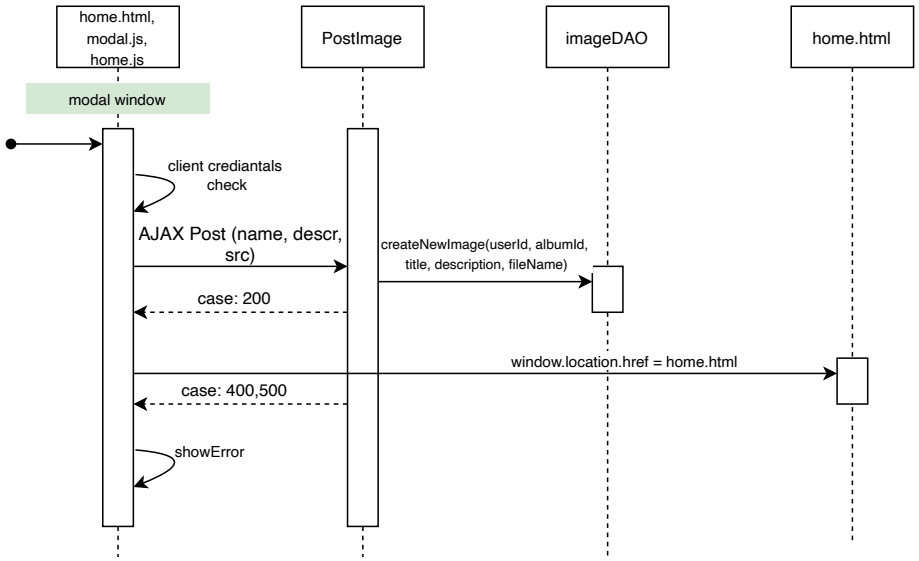
- Register



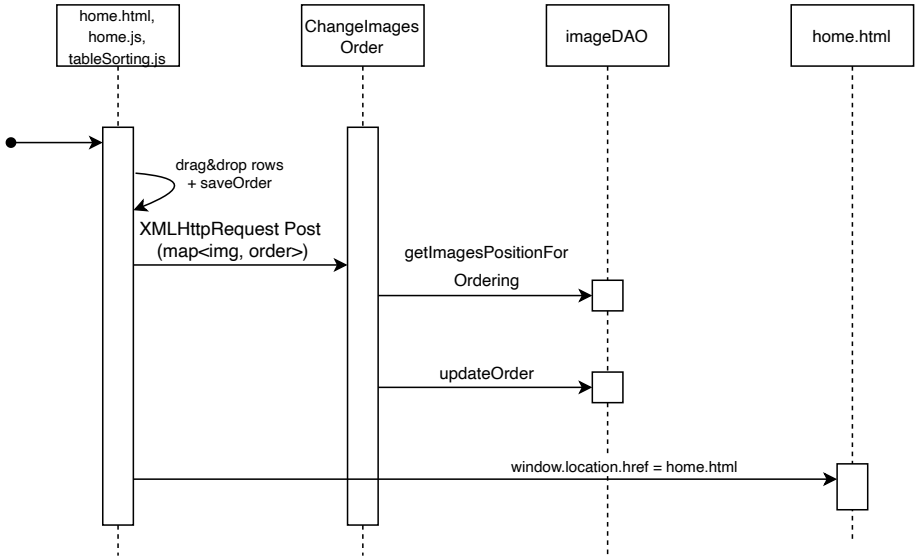
- Create album



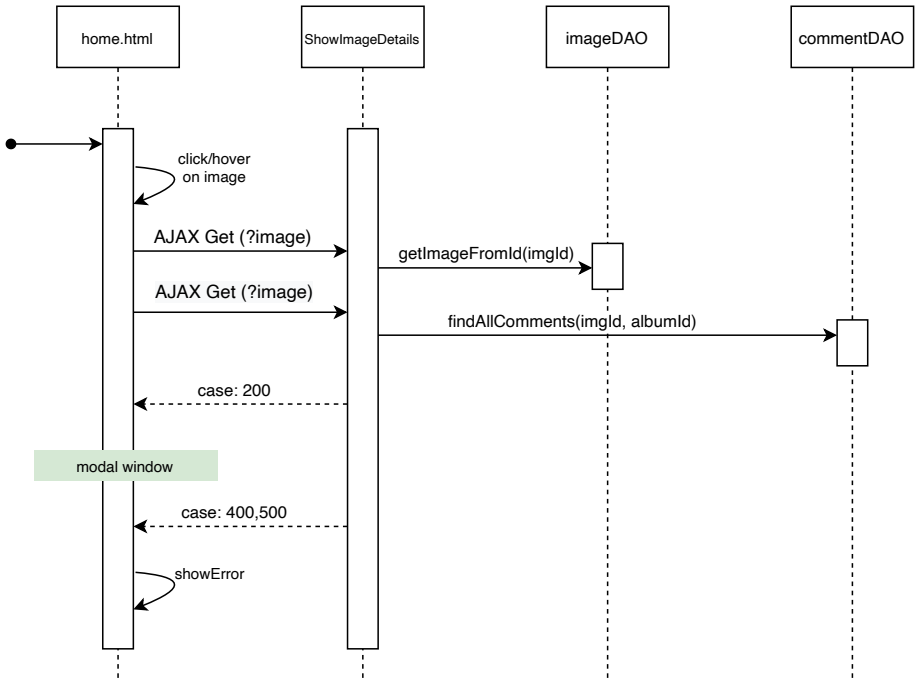
- Add image



- Change images order



- Show image details


home.html
ShowImageDetails
imageDAO
commentDAO
click/hover
on image
AJAX Get (?image)
getImageFromId(imgId)
AJAX Get (?image)
findAllComments(imgId, albumId)
case: 200
modal window
case: 400,500
showError

Capitolo 9

Scelte di progettazione e design

Durante lo svolgimento del progetto si sono effettuate alcune scelte di progettazione e design, disconstandosi (seppur lievemente) dalle specifiche.

1. Per rendere l'applicativo il più coerente possibile in tutte le sue parti dal punto di vista dell'interfaccia, si è deciso di sviluppare anche la tabella delle immagini degli album per riga (e non per colonna come seggerito da specifica), perdendo così l'effetto «carosello», ma garantendo un'interfaccia e un design omogeneo in ogni sua componente.

2. Si è preferito aprire la finestra modale (dei dettagli delle immagini) tramite un click dell'utente sull'immagine piuttosto che un «mouse over», per evitare aperture involontarie da parte dell'utente durante l'uso dell'applicativo.

3. È stata aggiunta la possibilità di caricare autonomamente immagini, in modo tale da rendere completa e realistica l'applicazione.