

Documentazione Progetto Object Orientation



UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Cristian Proroga [N86005629]

Maria Grazia Toppi [N86005183]

Pasquale Junior Montò [N86005762]

Anno accademico 2024/2025

Link repository: https://github.com/pakyjr/ToDo_Maven

Indice

- I. Introduzione - Traccia del Progetto**
- II. Capitolo 1 - Analisi dei requisiti**
 - 1.1 Utente**
 - 1.2 Board**
 - 1.3 ToDo**
 - 1.4 Relazioni**
- III. Capitolo 2 – Diagramma UML delle classi del dominio del progetto**
- IV. Capitolo 3 – Diagramma UML di dettaglio delle classi nell’implementazione**
- V. Capitolo 4 – Sequence Diagram di funzionalità a scelta**
 - 4.1 Login (username, plainPassword)**
- VI. Capitolo 5 – Manuale della GUI**
 - 5.1 Interfaccia login**
 - 5.2 Interfaccia registrazione**
 - 5.3 Interfaccia Personal Area**
 - 5.4 Interfaccia dei ToDo**
- VII. Capitolo 6 – Gestione degli errori**
 - 6.1 Schermata Login e Registrazione**
 - 6.2 Schermata ToDo**

Introduzione: Traccia del Progetto

1. Obiettivo del Sistema

Il sistema sviluppato per la gestione delle attività personali (ToDo), denominato *Galatea*, si ispira al modello di interazione offerto da *Trello*, ponendosi l'obiettivo di offrire un'interfaccia semplice, intuitiva e modulare per l'organizzazione dei compiti. Il progetto è stato realizzato seguendo i principi dell'Object Orientation, e suddiviso logicamente in tre livelli principali: interfaccia utente (GUI), logica applicativa (Controller) e accesso ai dati (DAO).

2. Accesso

L'accesso all'applicazione avviene tramite autenticazione con credenziali personali (login e password).

3. Gestione delle Board

La classe Board contiene gli attributi *boardName*, *description*, *color*, *owner* e *todoList*. L'attributo *owner* consente di distinguere tra chi ha creato una board e chi la visualizza in seguito a una condivisione. La gestione dei ToDo all'interno della board è strutturata in modo simile alla gestione delle board da parte dell'utente, con metodi dedicati all'ordinamento (*sortDueDate*) e alla ricerca (*searchTitle*) dei ToDo.

4. Gestione dei To Do

Ogni oggetto ToDo è caratterizzato da attributi obbligatori (come *title*, *status*, *owner*) e facoltativi (tra cui *dueDate*, *url*, *color*, *image*). Il campo *done* è rappresentato da un valore booleano, inizialmente impostato a false, e può essere modificato dall'utente tramite il metodo *toggle*. L'attributo *activityList* contiene un insieme di attività da completare affinché il ToDo risulti terminato; il metodo *checkValidActivity* consente di verificarne lo stato. Inoltre, ogni ToDo è identificato in modo univoco tramite un *UUID*, utile per la gestione della condivisione e cancellazione.

Capitolo 1 - Analisi dei requisiti

Per la realizzazione di un applicativo Java utile alla gestione dei To Do, in accordo ai requirements sono state individuate le seguenti classi: Utente, Board e ToDo.

1.1 Utente

- **Attributi:**

- UUID id: identificativo univoco dell'utente;
- String username: nome utente;
- String hashedPassword: password cifrata;
- ArrayList<Board> boardList: lista delle bacheche associate all'utente.

- **Metodi:**

- checkPassword(String): verifica la correttezza della password;
- addBoard(BoardName, String): crea e aggiunge una nuova bacheca;
- addBoard(Board): aggiunge una bacheca esistente;
- clearBoards(): rimuove tutte le bacheche;
- fillBoard(String): aggiunge bacheche predefinite se mancanti;
- deleteBoard(BoardName): elimina una bacheca;
- getId(), getUsername(), getHashedPassword(), getBoardList(): metodi getter;
- getBoard(BoardName): restituisce una bacheca specifica;
- moveToDoToAnotherBoard(BoardName, BoardName, int): sposta un ToDo tra bacheche.

- **Descrizione:**

Rappresenta un utente registrato nel sistema, con la possibilità di gestire bacheche personali contenenti attività (ToDo); gestisce l'autenticazione tramite password cifrata e consente operazioni CRUD sulle bacheche.

1.2 Board

- **Attributi:**

- int id: identificativo della bacheca;

- BoardName name: nome della bacheca;
 - String owner: creatore della bacheca;
 - String color: colore associato;
 - List<ToDo> todoList: lista di attività (ToDo).
- **Metodi:**
 - addTodo(String), addTodo(String, String): aggiunge un nuovo ToDo;
 - addExistingTodo(ToDo): aggiunge un ToDo esistente;
 - removeToDo(ToDo): rimuove un ToDo e aggiorna le posizioni;
 - getToDoList(), getName(), getOwner(), getColor(), getId(): metodi getter;
 - setColor(String), setId(int): metodi setter;
 - equals(Object), hashCode(): confronta bacheche per uguaglianza.
- **Descrizione:**

Rappresenta una bacheca di attività associata a un utente, e gestisce l'aggiunta, la rimozione e la visualizzazione dei ToDo, assicurando unicità e mantenendo l'ordine.

1.3 ToDo

- **Attributi:**
 - UUID id: identificativo univoco;
 - String title: titolo del ToDo;
 - String description: descrizione;
 - String status: stato dell'attività;
 - LocalDate dueDate, createdAt: data di scadenza e di creazione;
 - int position: posizione nella lista;
 - String owner: creatore del ToDo;
 - String url, color, image: metadati grafici e link utili;
 - Map<String, Boolean> activityList: sotto-attività;
 - Set<User> sharedUsers: utenti con cui è condiviso.
- **Metodi:**
 - Getter e setter per tutti gli attributi;

- `addActivity(String)`, `deleteActivity(String)`: gestisce sotto-attività;
- `addSharedUser(User)`, `removeSharedUser(String)`, `clearUsers()`: gestisce utenti condivisi;
- `equals(Object)`, `hashCode()`: confronta `ToDo` per ID.

- **Descrizione:**

Rappresenta una singola attività nella bacheca, con dettagli personalizzabili, sotto-attività, e possibilità di condivisione con altri utenti.

1.4 Relazioni

- **Ha ($\text{User} \rightarrow \text{Board}$):**

"Ogni utente può avere 0 o più bacheche (boards)".

- **Implementazione dei metodi:**

```
saveBoard(Board board, UUID userId)
```

```
loadUserBoardsAndTodos(User user)
```

```
getBoardId(BoardName boardName, String username)
```

La relazione è realizzata tramite la colonna `user_id` nella tabella `boards`.

Un utente può possedere più board, ma ogni board è associata a un solo utente.

- **Ha ($\text{Board} \rightarrow \text{ToDo}$)**

"Ogni board può avere 0 o più `ToDo` associati".

- **Gestione dei metodi:**

```
saveToDo(ToDo toDo, int boardId)
```

```
updateToDo(ToDo toDo, int boardId)
```

```
updateToDoBoardId(String toDoId, int newBoardId)
```

```
loadUserBoardsAndTodos(User user)
```

Ogni `ToDo` è collegato a una board tramite `board_id`. Le board possono contenere più `ToDo`,

ma ogni ToDo appartiene a una sola board.

- **Ha (ToDo → Activity):**

"Ogni ToDo può avere 0 o più attività associate (checklist)";

Relazione realizzata nella tabella activities, dove todo_id è la chiave esterna:

- **Metodi coinvolti:**

```
saveActivities(String todoId, Map<String, Boolean> activities)
clearActivities(String todoId)
```

Parte di `loadUserBoardsAndTodos()` per il caricamento delle attività.

- **Condivide (ToDo → SharedToDo):**

"Un ToDo può essere condiviso con 0 o più utenti".

Rappresentazione dalla tabella shared_todos:

- **Metodi:**

```
shareToDo(String todoId, String sharedWithUsername)
removeToDoSharing(String todoId, String sharedWithUsername)
removeAllToDoSharing(String todoId)
getSharedUsernamesForToDo(String todoId)
```

La relazione è multi-a-molti (ToDo ↔ Utente) mediata dalla tabella shared_todos.

- **Ha (ToDo → Proprietario)**

"Ogni ToDo ha un proprietario (username)"

Il campo owner_username in todos identifica l'utente creatore del ToDo, e viene usato per distinguere tra ToDo propri e quelli condivisi.

- **Accesso (User → Shared ToDo)**

"Ogni utente può accedere a 0 o più ToDo condivisi con lui", implementato nella parte finale di `loadUserBoardsAndTodos(User user):`

Il DAO carica i ToDo condivisi per un determinato utente e li assegna alla board originale, se presente.

Capitolo 2 - Diagramma UML delle classi del dominio del problema

Di seguito forniamo un diagramma, realizzato secondo il formalismo UML, utilizzato per modellare il dominio del problema. Il diagramma delle classi rappresenta i principali concetti coinvolti nel sistema informativo per la gestione dei ToDo, evidenziando le entità, gli attributi rilevanti e le relazioni tra gli oggetti che compongono l'architettura logica dell'applicazione.

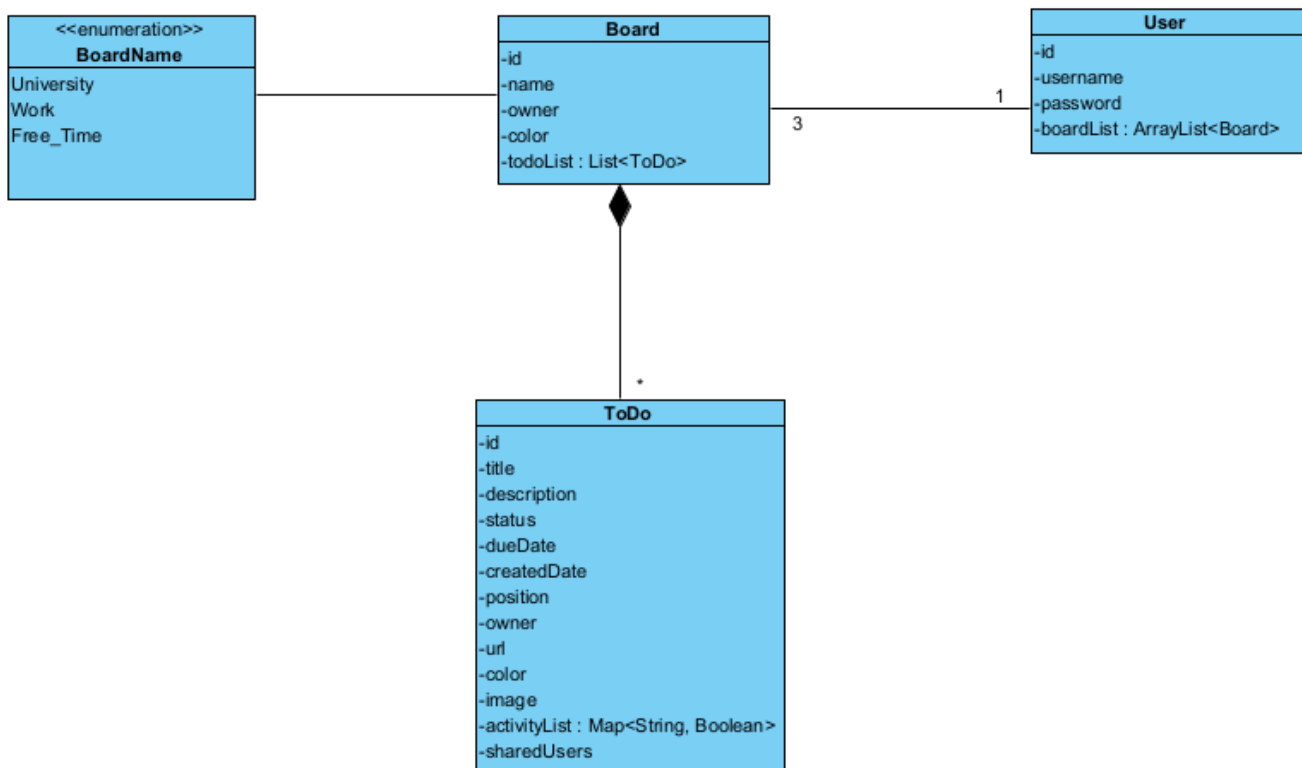


Figura 2.1

Capitolo 3 - Diagramma UML di dettaglio delle classi nel dominio della soluzione

Il seguente diagramma, costruito secondo il formalismo UML, rappresenta il dominio della soluzione e descrive l'architettura logica del sistema informativo così come implementato. In particolare, il diagramma evidenzia le principali classi software dell'applicazione, con i relativi attributi e metodi, nonché le relazioni statiche tra di esse (come associazioni, aggregazioni e dipendenze). Vengono inoltre riportati i legami con la struttura dei dati persistenti, offrendo una visione chiara della corrispondenza tra il modello concettuale e la sua realizzazione concreta nel codice. Questo modello è stato progettato seguendo i principi di separazione delle responsabilità e modularità, facilitando così la manutenibilità, l'estensibilità e il riutilizzo del software. In particolare, si è adottata un'architettura a livelli, che distingue logicamente tra interfaccia utente, logica di funzionamento e accesso ai dati.

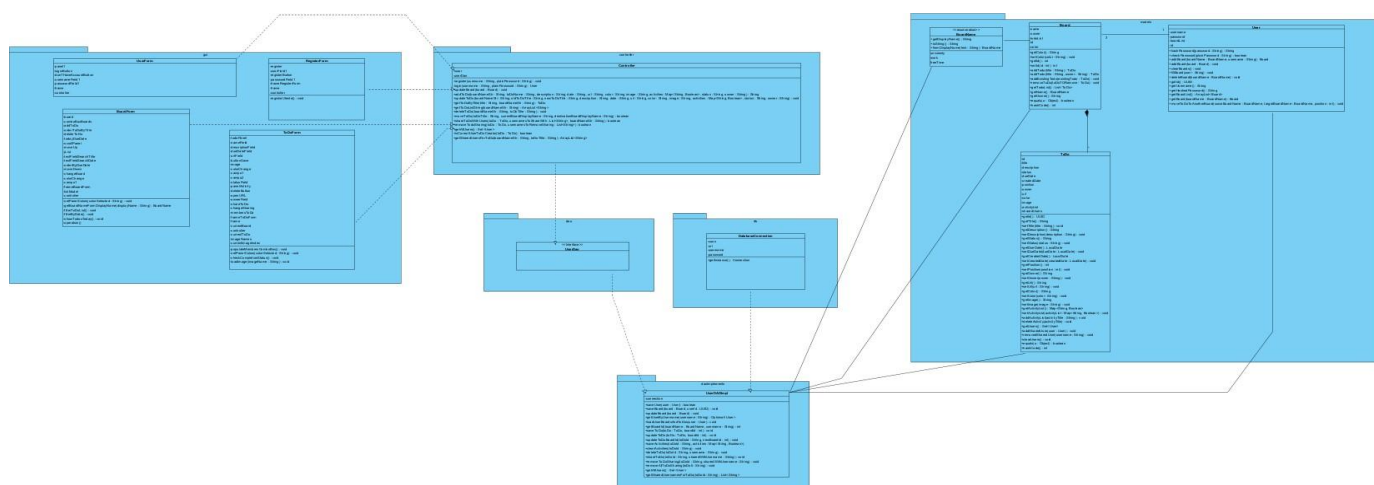


Figura 3.1

Capitolo 4 - Sequence diagram di funzionalità a scelta

Di seguito viene presentato un diagramma di sequenza UML, selezionato per illustrare in modo puntuale una delle principali funzionalità offerte dall'applicativo. Tali diagrammi descrivono l'interazione dinamica tra gli oggetti del sistema durante l'esecuzione di specifici casi d'uso, evidenziando l'ordine temporale dei messaggi scambiati e la responsabilità dei diversi componenti. L'obiettivo è fornire una rappresentazione chiara del comportamento del sistema in risposta a determinati eventi, mettendo in luce il flusso delle informazioni tra interfaccia utente, logica applicativa e livello di persistenza.

4.1 Login(username, plainPassword)

Il seguente diagramma di sequenza descrive l'interazione tra i componenti del sistema durante l'esecuzione del metodo `login(username, plainPassword)`.

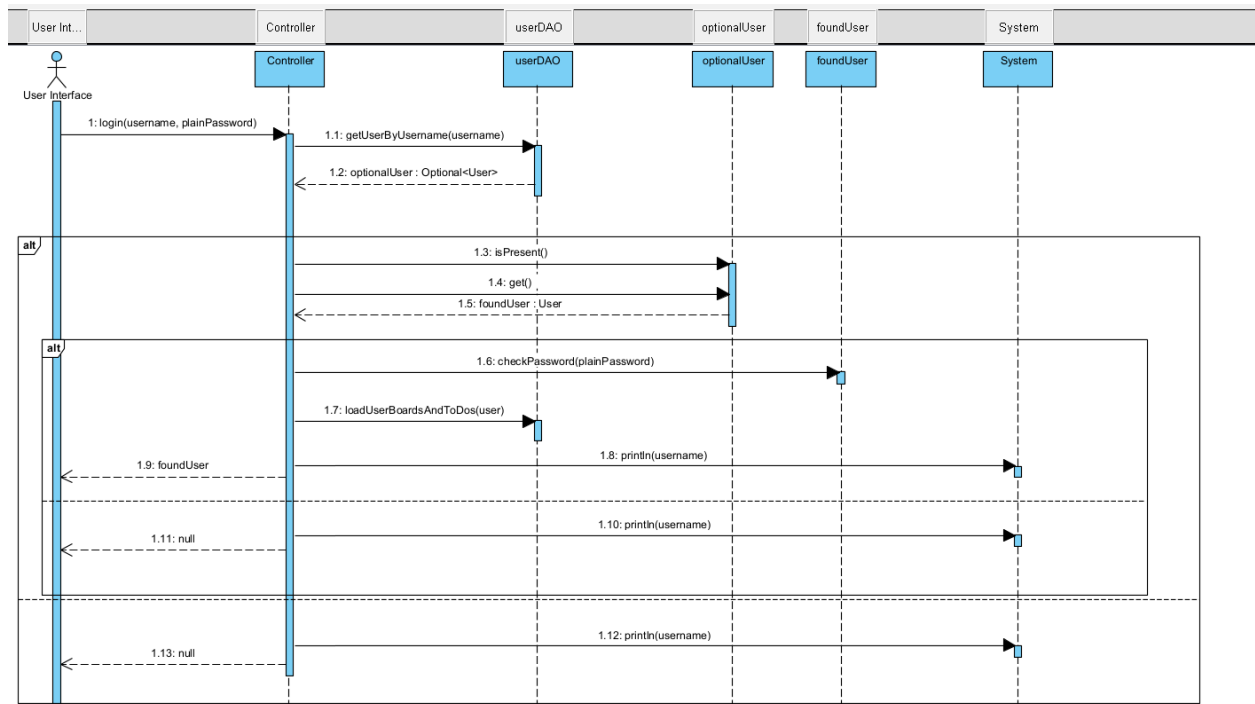


Figura 4.1

Capitolo 5 - Manuale della GUI

5.1 Interfaccia login

All'avvio dell'applicazione viene presentata una schermata di benvenuto, dalla quale l'utente può scegliere se procedere con il login o con la registrazione di un nuovo account. Questa interfaccia funge da punto di ingresso al sistema e garantisce una separazione chiara tra utenti già registrati e nuovi utenti.



Figura 5.1

5.2 Interfaccia registrazione

L'interfaccia di registrazione consente la creazione di un nuovo account utente mediante l'inserimento

obbligatorio delle seguenti informazioni: username, password.

È previsto un pulsante per tornare alla schermata di login in caso di errore o ripensamento, e il sistema verifica che tutti i campi siano compilati correttamente prima di consentire la registrazione, garantendo la consistenza dei dati utente.

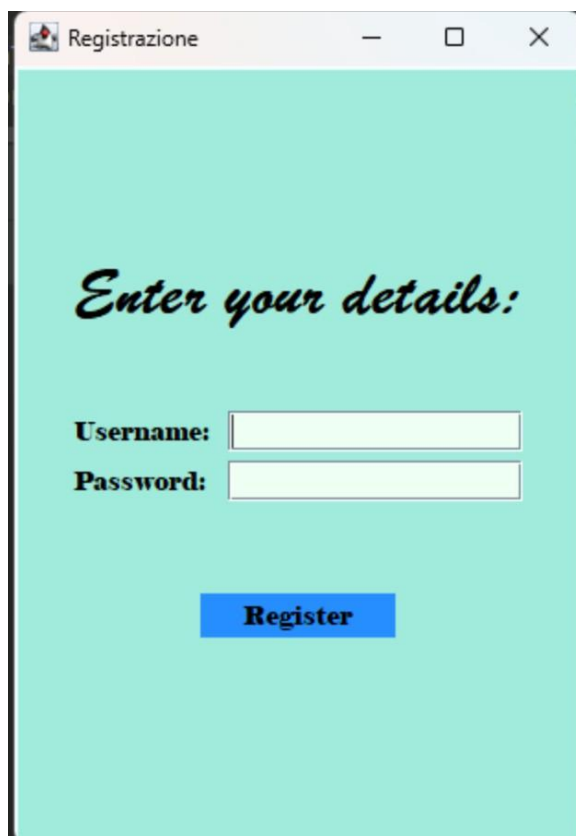
The image shows a screenshot of a web browser window titled "Registrazione". The window has a light blue background. In the center, the text "Enter your details:" is written in a black, cursive font. Below this text, there are two input fields. The first is labeled "Username:" and the second is labeled "Password:". Both labels are in a bold, black, sans-serif font. Below the input fields, there is a blue button with the word "Register" in white, bold, sans-serif font. The window has standard window controls (minimize, maximize, close) in the top right corner.

Figura 5.2

5.5 Interfaccia Personal Area

Dopo l'autenticazione, l'utente viene reindirizzato alla schermata Personal Area dalla quale è possibile gestire le proprie board, e i ToDo all'interno di esse.

Lateralmente, l'interfaccia permette all'utente di navigare tra le board predefinite, oltre che di scegliere il colore di ogni singola board, come da propria preferenza. Naturalmente, presenta anche le opzioni per l'aggiunta del ToDo e per la sua eliminazione, oltre che la possibilità di cambiarne la board di appartenenza, di cambiarne l'ordine in base al proprio desiderio (tramite le opzioni *MoveUp* e *MoveDown*), di ottenere un ordinamento alfabetico o in base alla data di scadenza.

Grande rilevanza hanno le opzioni di filtraggio dei ToDo ovviate dal *Today DueDate*, che permette la visualizzazione dei ToDo in scadenza nel giorno corrente, dal *Search by Title*, che permette la ricerca

del ToDo tramite nome, e dal *Search by DueDate*, che permette la ricerca del ToDo in base alla data di scadenza.

Seguitamente la creazione del ToDo, se la data di scadenza è passata e le attività non sono state completate, il nome del ToDo sulla schermata diventerà rosso. Ovviamente, se le attività sono state completate, questa situazione non si verifica.

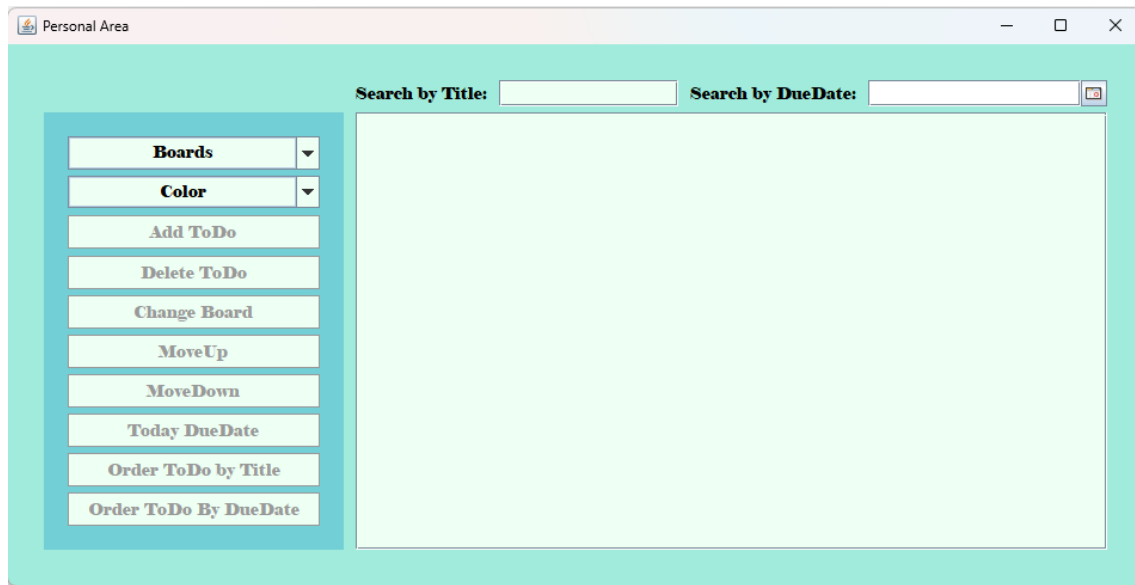


Figura 5.3

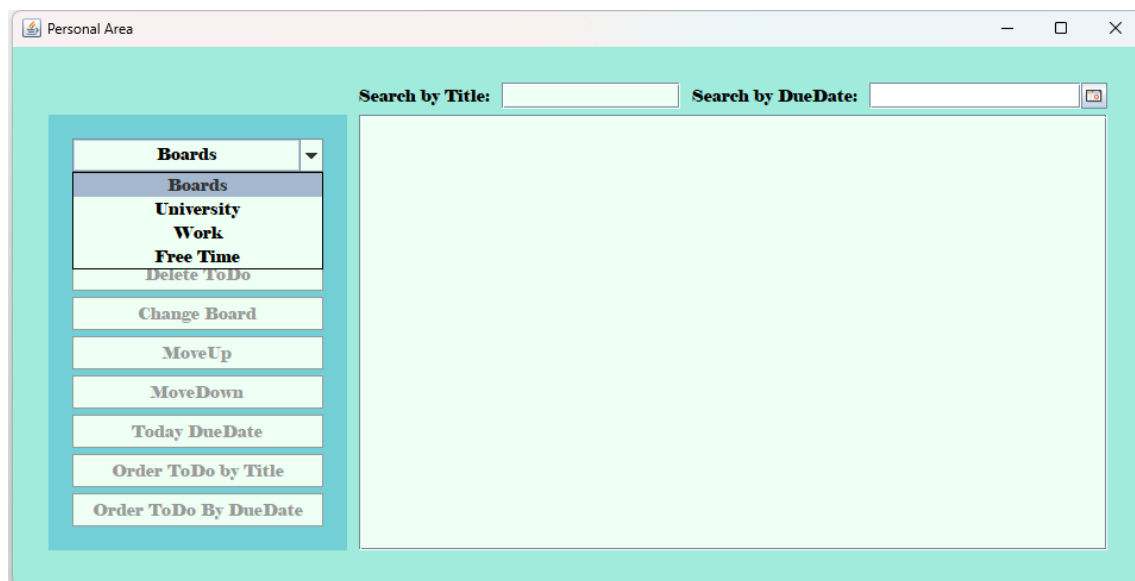


Figura 5.4

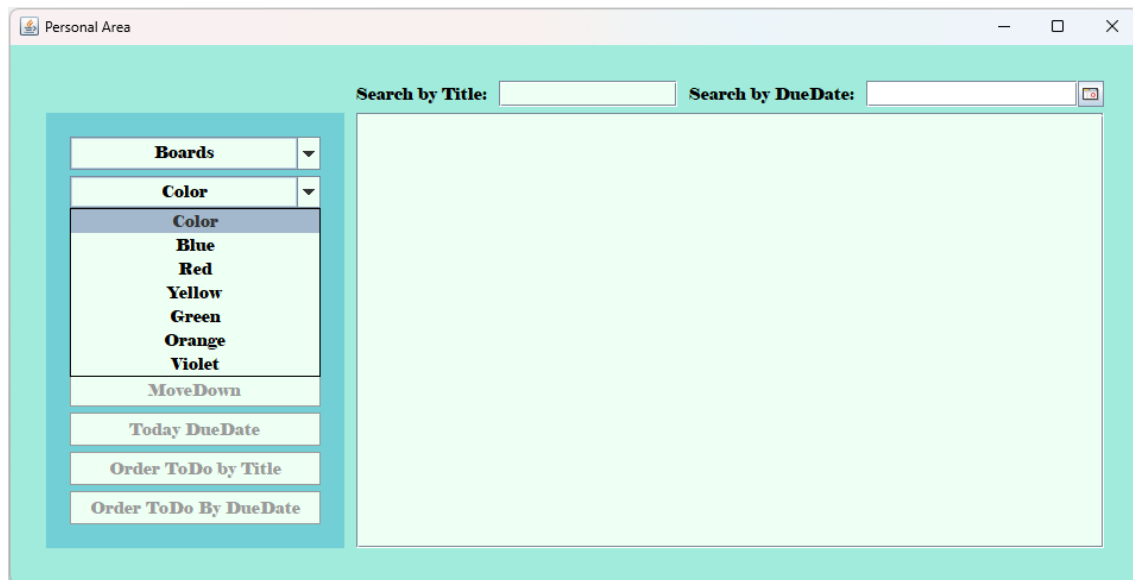


Figura 5.5

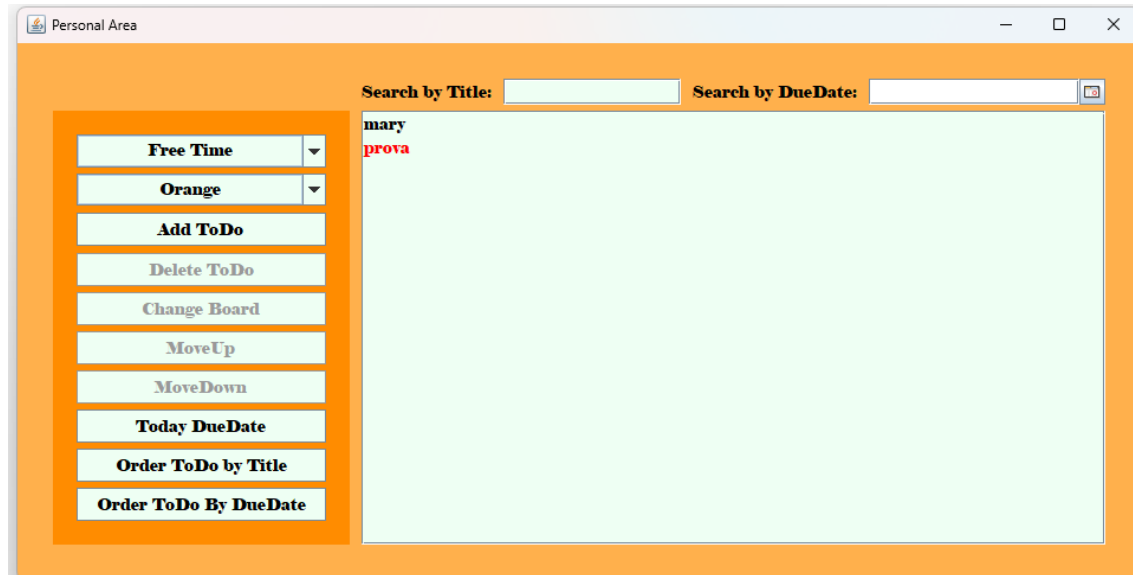


Figura 5.6

5.6 Interfaccia ToDo

La schermata “Edit ToDo” rappresenta l’interfaccia grafica dedicata alla modifica dettagliata di un’attività (ToDo) all’interno dell’applicazione. La finestra si presenta con un layout a due colonne: in

alto sinistra è presente un riquadro che raccoglie tutti i principali campi editabili dell'attività, mentre sulla destra sono disponibili i controlli relativi alle funzionalità aggiuntive.

Una volta editati i campi obbligatori del titolo, della descrizione e della data di scadenza, è possibile inserire anche un URL a scelta, che verrà poi aperto tramite l'opzione *open url*.

Cliccando sulla schermata bianca è possibile aggiungere attività nominandole a piacere, e il loro completamento o meno andrà ad influenzare lo status del ToDo, il quale in assenza di attività presenterà la dicitura "Not Started", con attività la dicitura sarà "Incomplete", e con attività complete la dicitura sarà appunto "Complete". Inoltre, sussiste anche la possibilità di cancellare le attività, utilizzando sulla destra l'opzione *delete activity*.

Relativamente le funzionalità aggiuntive, la schermata consta anche in un'immagine decorativa, modificabile tramite clic secondo la volontà dell'utente, oltre che di opzioni relative al cambio del colore della schermata del ToDo (colori primari e secondari, come per l'opzione delle board), e opzioni relative alla condivisione del ToDo con altri utenti registrati sulla piattaforma, nonché sulla modifica della condivisione e sulla visualizzazione di tutti gli utenti con il quale il ToDo è stato condiviso. Tuttavia, la condivisione del ToDo può avvenire solo in un secondo momento, ovvero dopo la seconda apertura del ToDo seguitamente al suo salvataggio tramite l'opzione *save*; la visualizzazione del ToDo da parte di un utente non creatore del ToDo sarà differente, in quanto non solo non gli sarà permesso di modificare alcun campo, ma sia la schermata che mostra le attività che l'immagine risulteranno grigie, e la parte del ToDo che dovrebbe contenere il nome del creatore del ToDo conterrà appunto quello, e non quello dell'utente che ha effettuato l'accesso al programma.

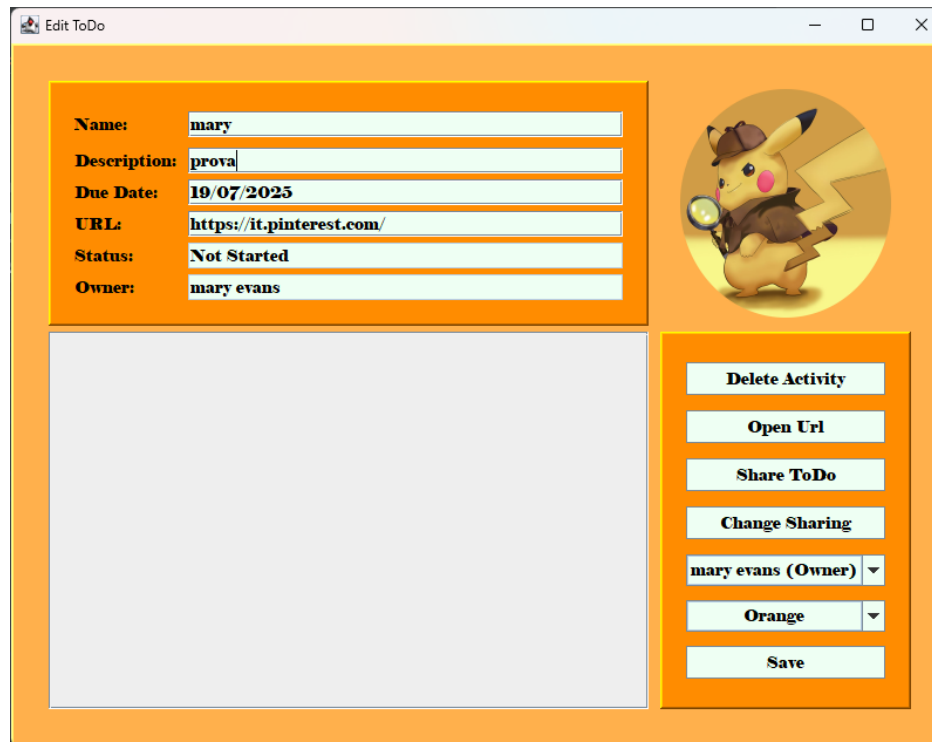


Figura 5.7

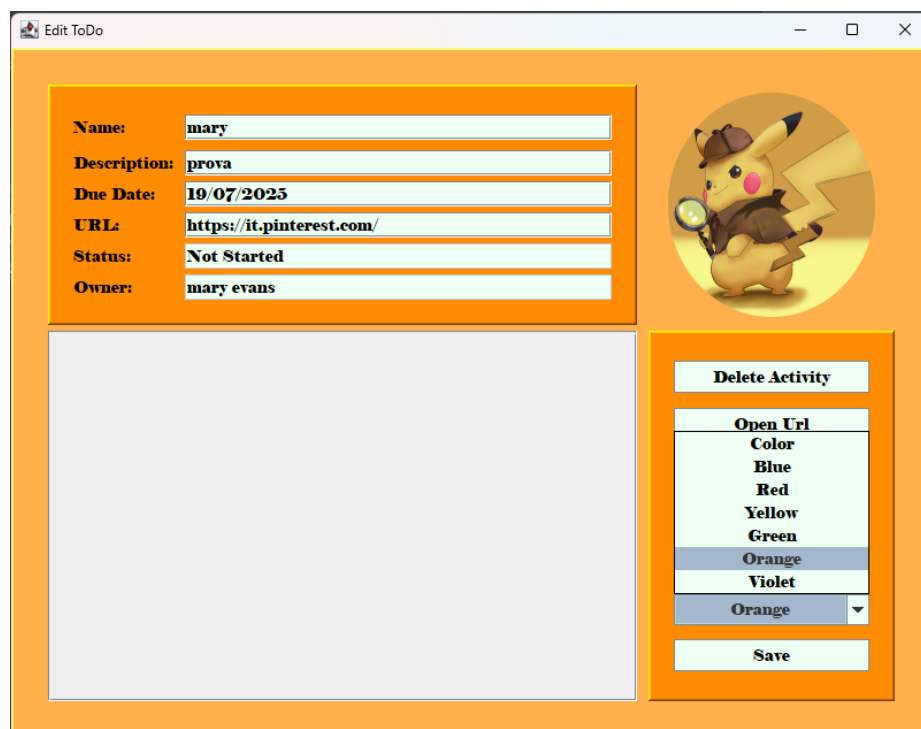


Figura 5.8

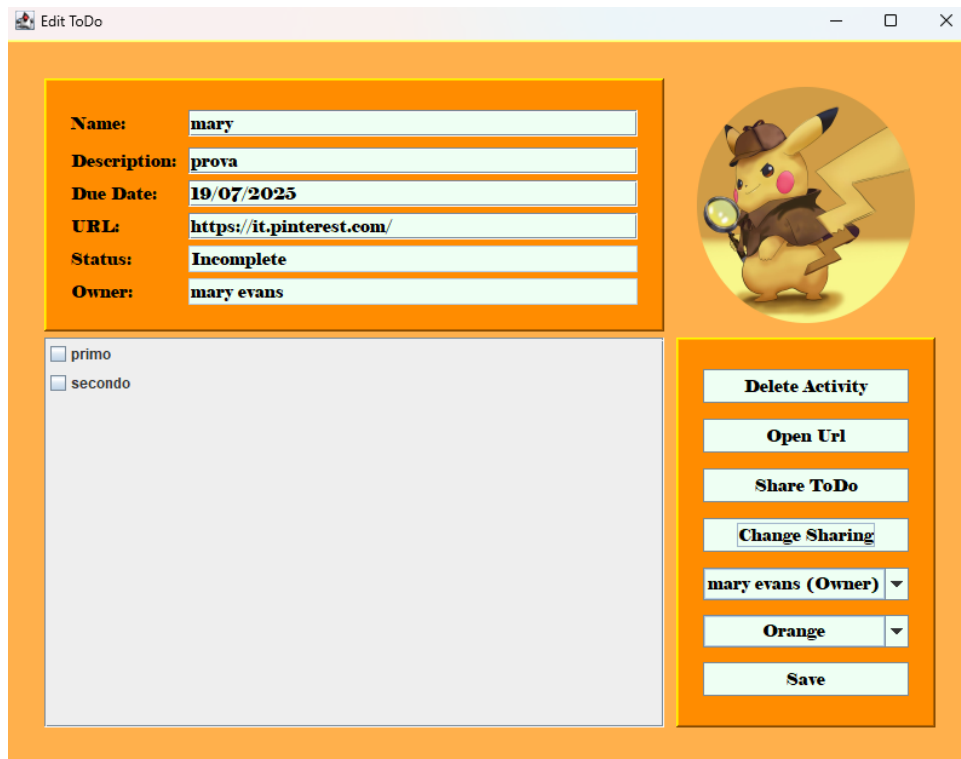


Figura 5.9

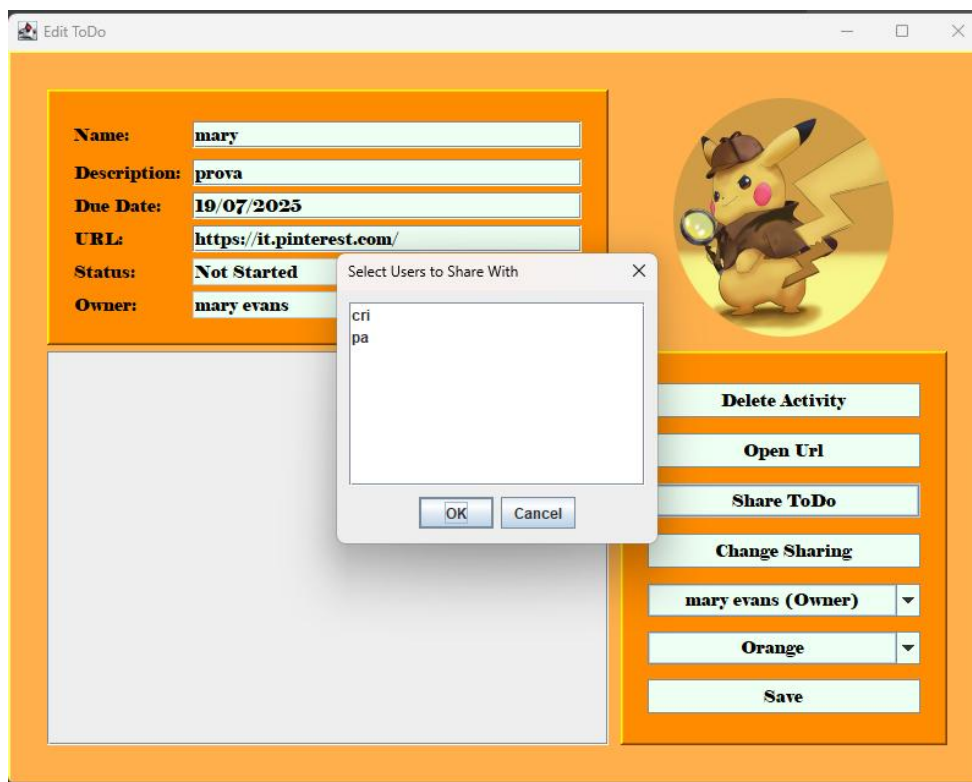


Figura 5.10

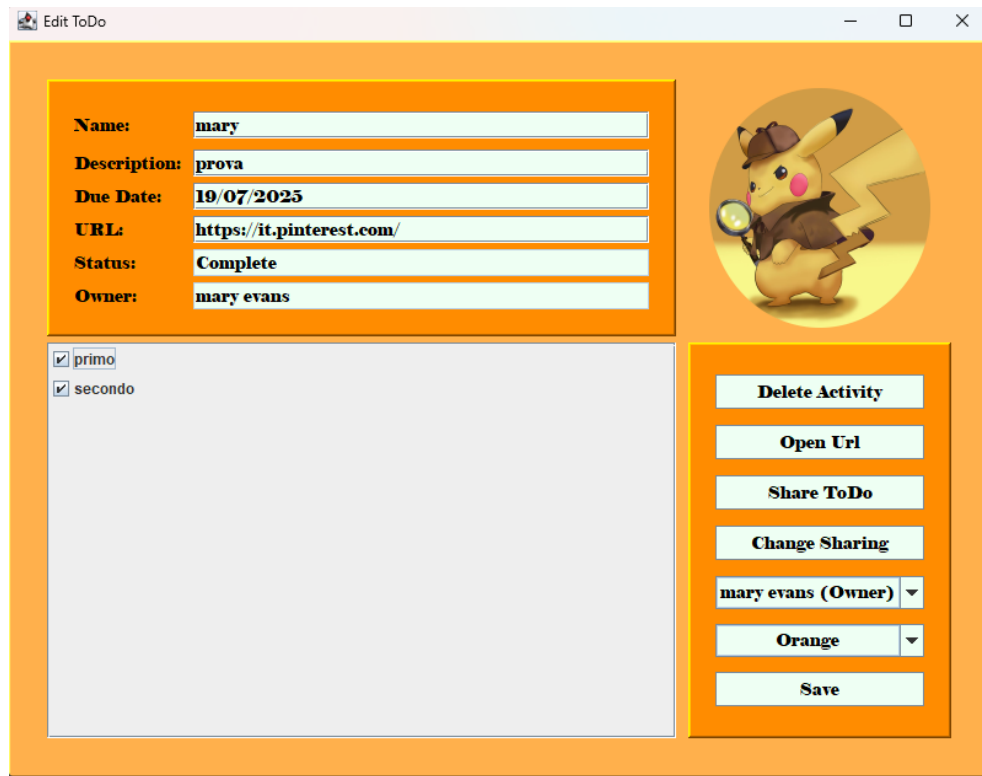


Figura 5.11

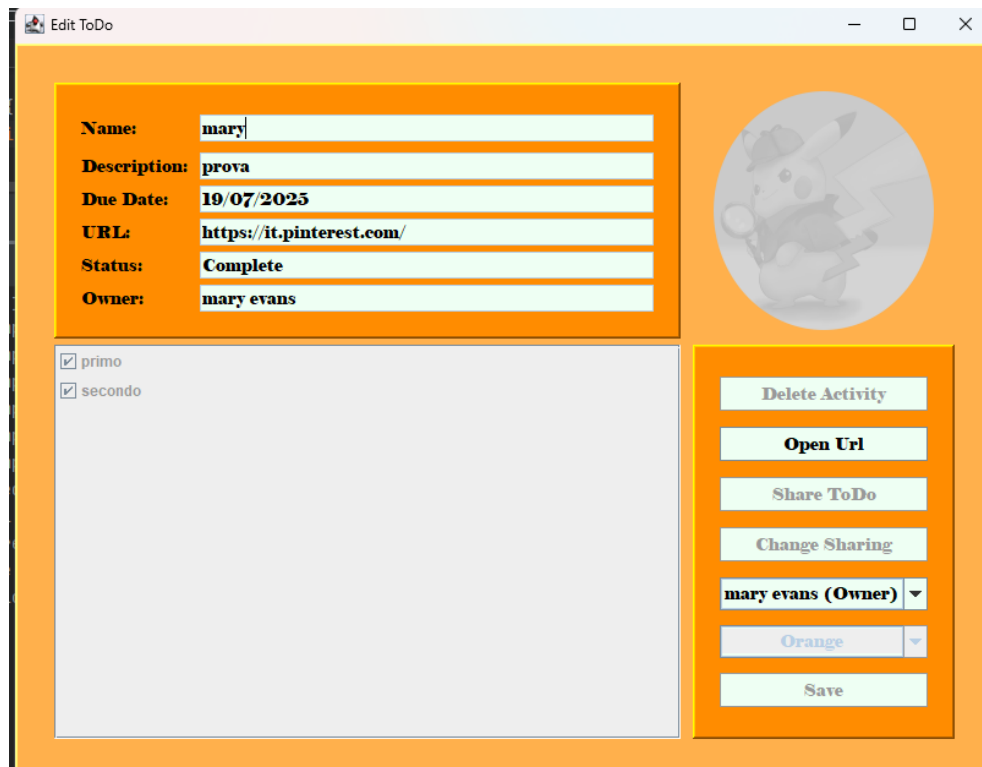


Figura 5.12

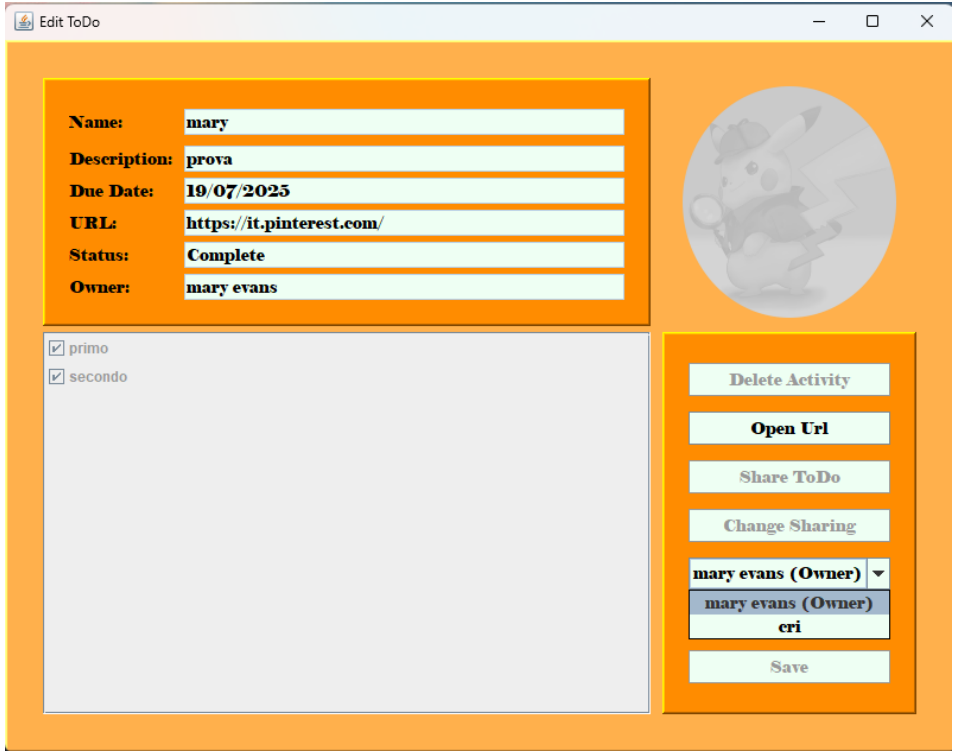


Figura 5.13

Capitolo 6 - Gestioni degli errori

Al fine di garantire l'affidabilità e la coerenza dei dati inseriti, l'applicazione implementa un sistema di validazione preventiva sui campi di input presenti in ogni schermata, fornendo messaggi di errore informativi e contestuali nel caso in cui l'utente inserisca valori non conformi ai vincoli richiesti. Questa logica di controllo consente di intercettare eventuali anomalie in fase di input, evitando comportamenti imprevedibili o compromissioni del flusso applicativo. I messaggi vengono mostrati in tempo reale mediante finestre modali, contribuendo a migliorare l'usabilità e la robustezza del sistema. Di seguito alcuni esempi esplicativi in relazione alle diverse interfacce.

6.1 Schermata Login e Registrazione

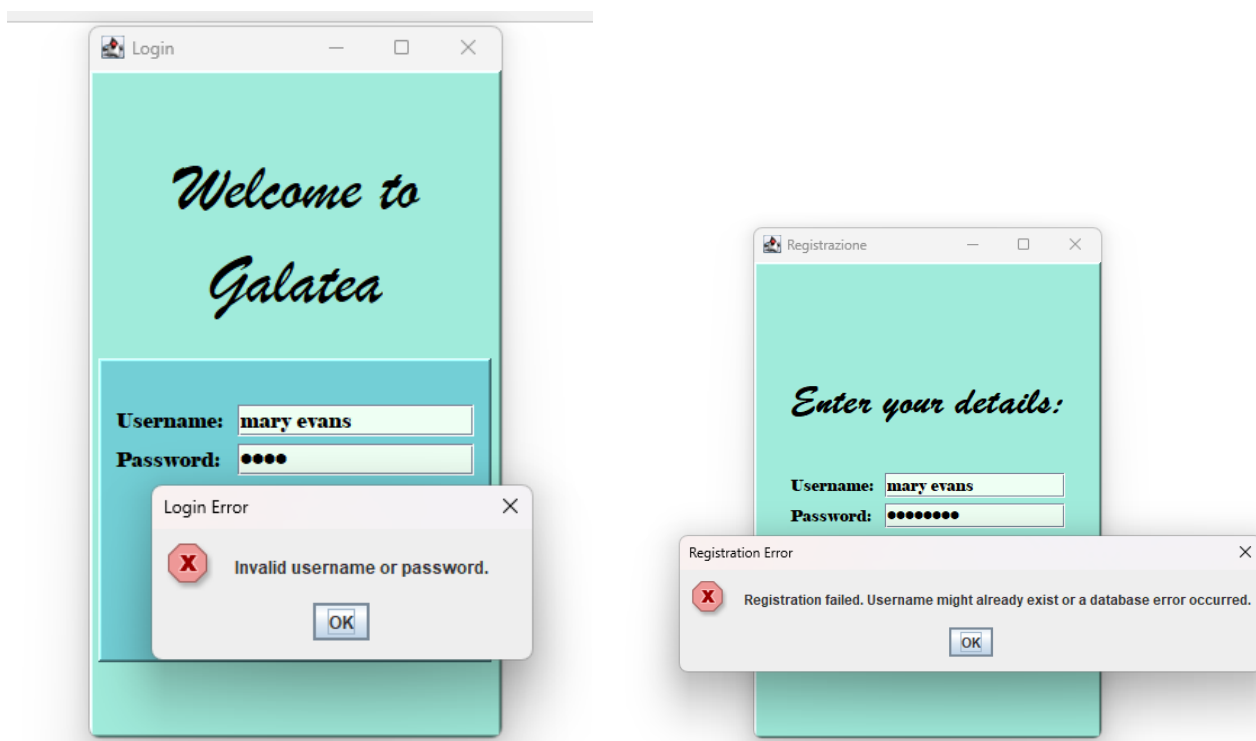


Figura 6.1 e 6.2

Nelle figure soprastanti vengono mostrati gli errori relativamente alle interfacce del login e della registrazione. Nel caso di un login con password sbagliata che faccia riferimento a un utente già registrato, oppure a un login di un utente non registrato, viene mostrata la relativa finestra che rende

non valido l'accesso alla schermata delle board; relativamente la registrazione di un utente, invece, essendo implementato un database che salva i differenti utenti registrati, nel caso di un tentativo di registrazione con uno username identico già presente il sistema emette un errore tramite la finestra mostrata.

6.2 Schermata ToDo

Nella schermata relativa al ToDo, invece, gli avvisi mostrati dal sistema concernono il tentativo di salvataggio prima di aver compilato i campi obbligatori, l'immissione di un formato data non conforme, la mancata selezione di un utente (seguita tuttavia da un tentativo di conferma) durante l'utilizzo delle opzioni *share ToDo* e *change sharing*, e infine il tentativo di modifica da parte di un utente con cui è stato condiviso un ToDo del ToDo stesso, pur senza esserne il proprietario.

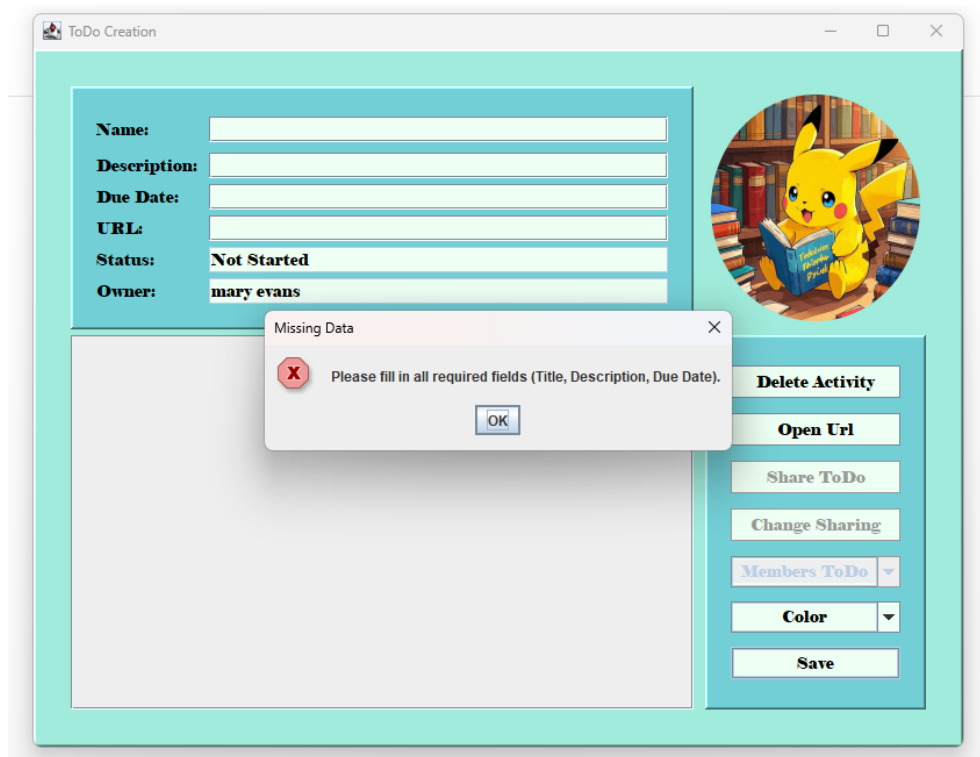


Figura 6.3

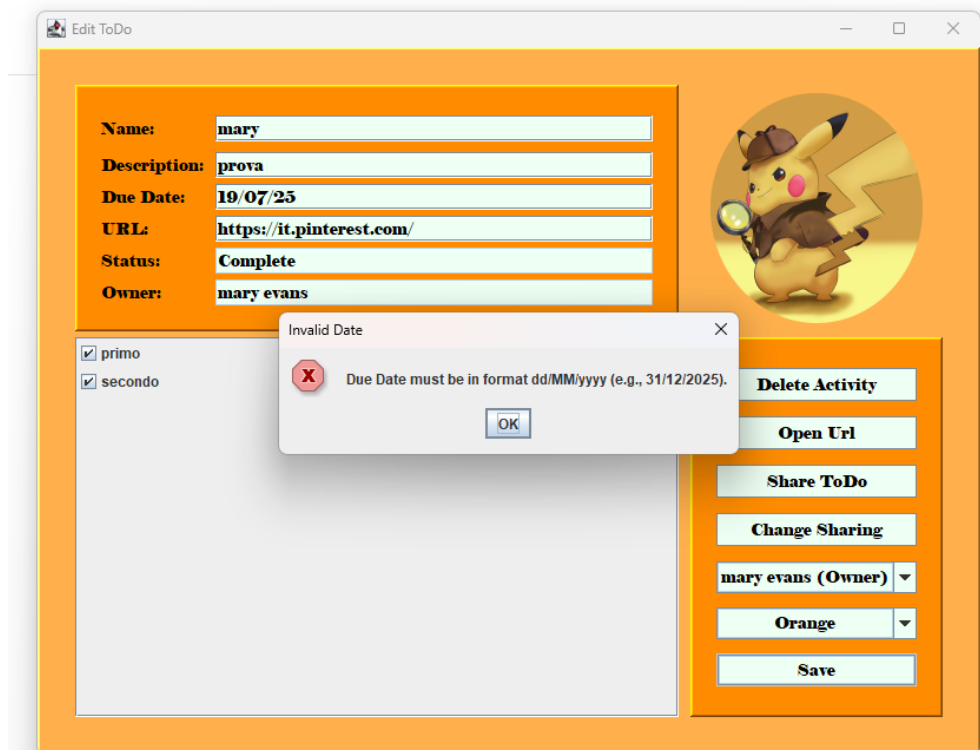


Figura 6.4

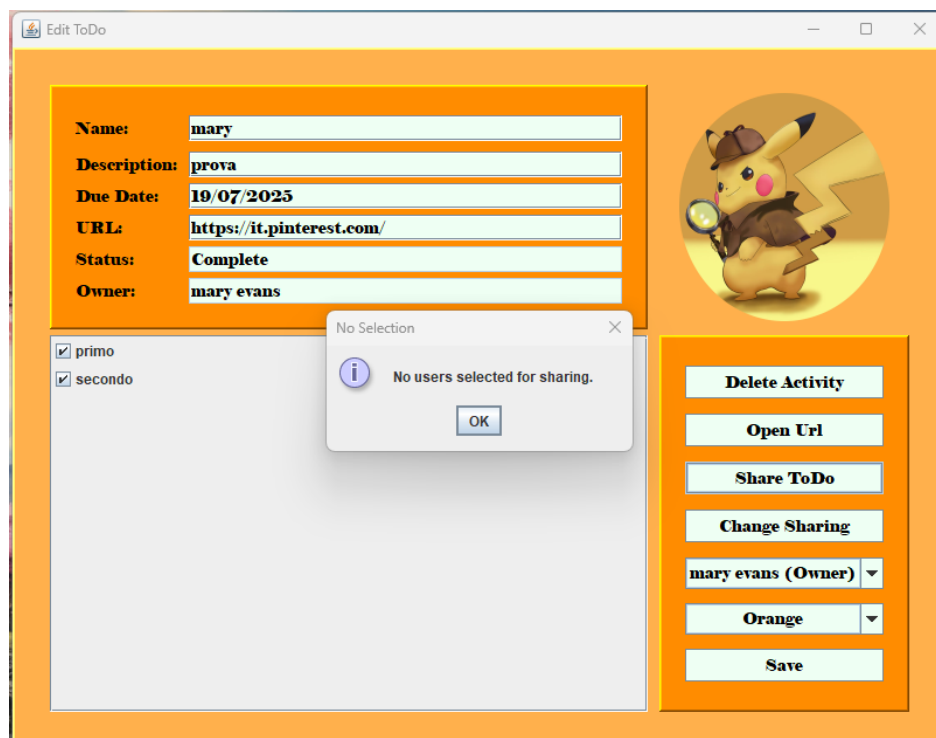


Figura 6.5

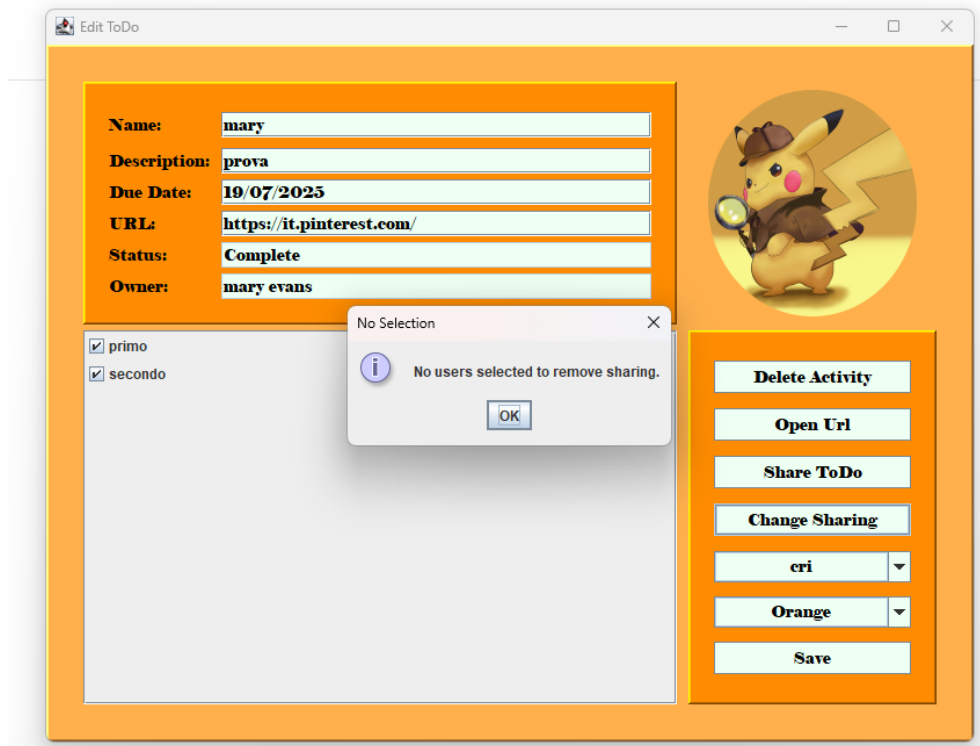


Figura 6.6

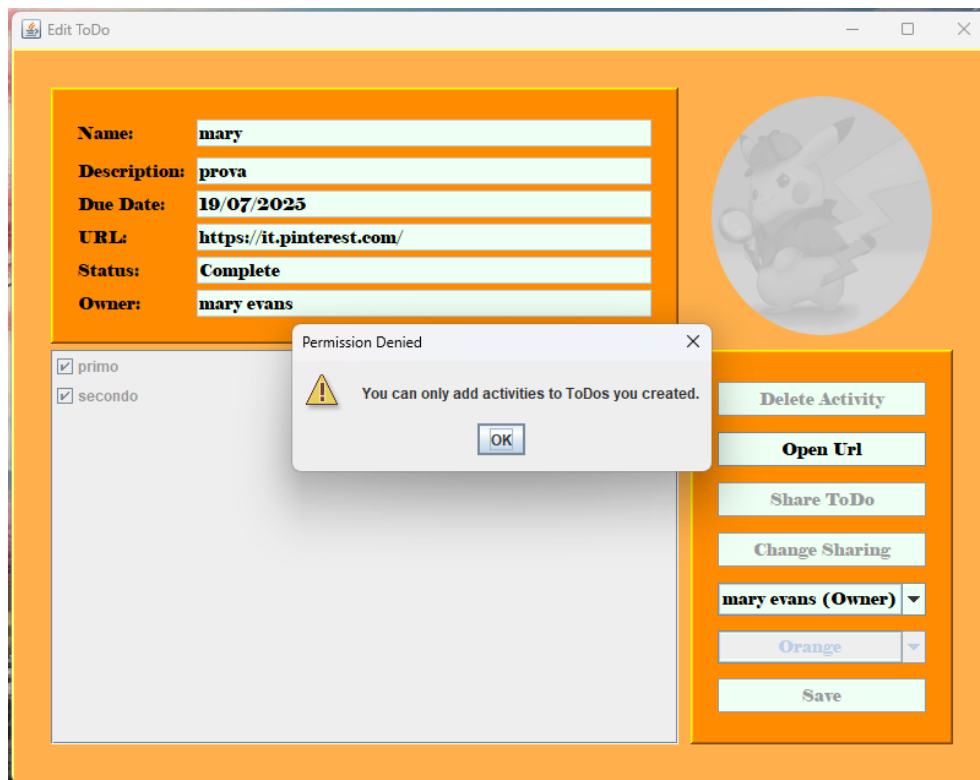


Figura 6.7