Gestore ToDo

Obiettivo del Progetto

L'applicativo è progettato per consentire a più utenti di gestire in locale le proprie attività, dette ToDo, organizzate in bacheche. Il sistema prevede meccanismi per la creazione, modifica, rimozione e condivisione dei ToDo, nonché per la gestione dell'ordinamento all'interno delle bacheche e la creazione, modifica e rimozione delle bacheche stesse.

Classi Principali

Ogni classe ha un ruolo ben definito:

- **Utente** si occupa della gestione dell'account e delle bacheche.
- **Bacheca** organizza e gestisce l'ordinamento dei ToDo.
- **ToDo** racchiude i dati dell'attività e le logiche di spostamento e condivisione.

Utente

Responsabilità:

Rappresenta l'utente del sistema. Contiene dati di autenticazione (username, password) e lo stato del login (booleano). L'utente gestisce inoltre la propria collezione di bacheche.

Attributi Chiave:

- `username` (immutabile): garantisce l'identità in modo univoco.
- `password`: usata per l'autenticazione, non posta come final per una possibile implementazione di recupero e cambio password.
- `stato`: indica se l'utente è loggato. Serve per poter garantire l'uso di funzioni attuabili solo se è un preciso utente a chiamarle (come ToDo.condividi).
- `bacheche`: lista (implementata con ArrayList) delle bacheche create dall'utente.

Metodi Principali:

`login`, `logout`, `creaBacheca`, `eliminaBacheca`.

Scelte Progettuali:

L'uso dell'attributo final per il nome utente impone l'immutabilità dell'identificatore, mentre la gestione delle bacheche mediante una lista permette di mantenere l'ordinamento definito dall'utente.

Bacheca

Responsabilità:

Rappresenta una bacheca in cui vengono raggruppati i ToDo. È direttamente associata a un utente (proprietario) e contiene i ToDo organizzati in una lista ordinata.

Attributi Chiave:

- `titolo`: indica il nome della bacheca, è associato a un enum (TitoloBacheca) per limitare i valori ammessi: ciò limita anche il numero di bacheche massime creabili, ossia 3.
- `descrizione`: fornisce informazioni supplementari sulla bacheca.
- `todos`: lista di ToDo attivi in questa bacheca.
- `utente`: riferimento all'utente proprietario della bacheca.

Metodi Principali:

- `creaToDo`: crea un nuovo ToDo e lo inserisce nella bacheca che chiama il metodo. Inoltre assegna l'utente originale e crea una lista di utenti inserendoci l'utente originale
- `creaToDoCondiviso`: crea un nuovo ToDo a partire da un ToDo preesistente, copiando gli attributi ma non cambiando utente originale né creando una nuova lista utenti; inoltre aggiunge il nuovo utente a quest'ultima lista
- `refreshPosizioni`: ricalcola l'attributo posizione per ogni ToDo presente nella lista todos, da attuare ogni qualvolta ci sia una modifica di posizione di un ToDo, una eliminazione di un ToDo o uno spostamento di ToDo tra bacheche.

Scelte Progettuali:

La scelta di utilizzare una lista (ArrayList) per i ToDo permette di gestire ordinamenti e posizioni. In fase di creazione della bacheca, la relazione diretta con l'utente facilita la verifica dei permessi e garantisce l'assegnazione corretta del ToDo al suo creatore. Inoltre, il metodo `refreshPosizioni` centralizza l'aggiornamento dell'ordinamento, evitando incoerenze quando si riorganizzano i ToDo.

ToDo

Responsabilità:

Rappresenta l'attività personale da svolgere. Un ToDo contiene informazioni come titolo, data di scadenza, url, descrizione, immagine, colore e stato (completato o meno). Inoltre, il ToDo è dotato di meccanismi per la gestione della posizione all'interno della bacheca e per la condivisione fra utenti.

Attributi Chiave:

- `titolo`, `scadenza`, `url`, `descrizione`, `immagine`, `colore`, `stato`: informazioni di base dell'attività.
- `posizione`: determina l'ordine del ToDo nella bacheca.
- `bacheca`: riferimento alla bacheca nella quale è contenuto.
- `utenteOriginale`: l'utente creatore del ToDo.
- `utenti`: collezione (HashSet) degli utenti con cui il ToDo viene condiviso.

Metodi Principali:

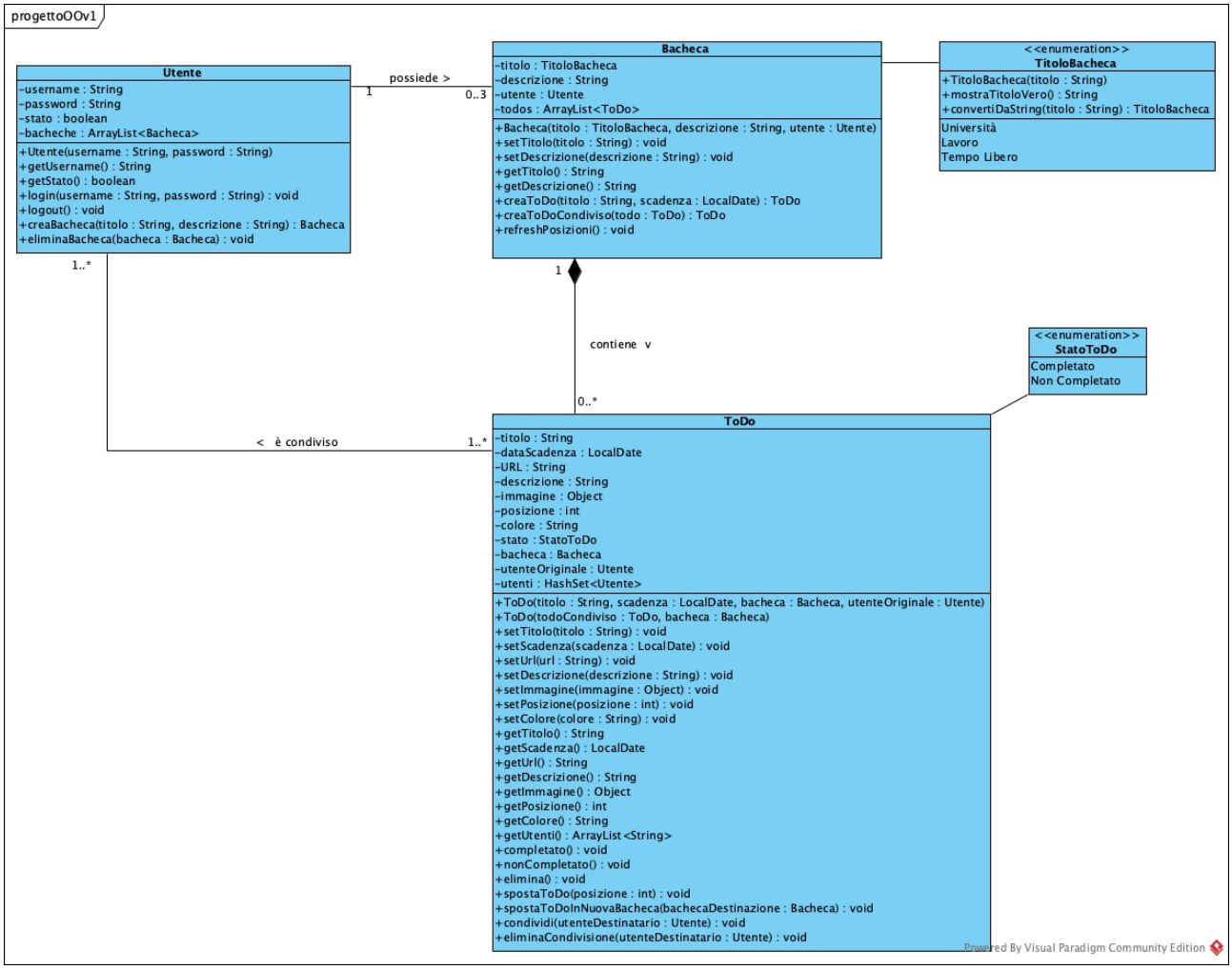
- Metodi setter e getter per le proprietà: nota di riguardo per il setter del titolo, che non è modificabile se il ToDo è condiviso, perché questo attributo viene usato come identificativo dal metodo eliminaCondivisione.
- Operazioni per spostare il ToDo all'interno della stessa bacheca (`spostaToDo`) o spostarlo in un'altra bacheca (`spostaToDoInNuovaBacheca`).
- Metodi per gestire lo stato di completamento (`completato` / `nonCompletato`).
- Metodi per la gestione della condivisione: `condividi`, `eliminaCondivisione`,.

Scelte Progettuali:

L'utilizzo di un Set per mantenere traccia degli utenti al quale il ToDo è condiviso garantisce l'assenza di ripetizioni. La presenza di un riferimento all'utente creatore (utenteOriginale) permette di stabilire i permessi per operazioni di condivisione. I ToDo condivisi sono volutamente duplicati per permettere personalizzazioni per utente: l'utente originale di un ToDo condiviso può anche eliminare il suo ToDo senza toccare quelli condivisi.

Limiti del modello

I limiti di questo primo prototipo sono la mancanza di un sistema di gestione per l'unicità degli username, di strumenti di ricerca e di ordinamento per scadenza giornaliera dei ToDo, di visualizzazione di ToDo scaduti e l'assenza totale di un'interfaccia grafica.



NB: sono presenti, soprattutto nella classe ToDo, metodi getter e setter che non sono utilizzati e potrebbero essere eliminati: li ho lasciati per scelta, per poter testare in questa fase del progetto il corretto funzionamento delle classi, e se necessario in futuro saranno tagliati.

Link GitHub

https://github.com/ubiestubi/GestoreToDo