Documentazione: modello di dominio e progettazione della soluzione

Fabio Pavone, Vincenzo Manna

1 Analisi del dominio del problema

Dall'analisi della specifica del problema si evince la necessità della creazione di un sistema informatico composto da un applicativo Java, dotato di GUI costruita con il framework Swing, ed in congiunzione con un DMBS PostgreSQL.
L'applicazione dovrà essere in grado permettere ai suoi utenti di organizzare le proprie attività in bacheche che conterranno 'ToDo', oggetti che descrivono le attività dell'utente.
Ogni utente del sistema potrà registrarsi[1] o fare login nel sistema e conseguentemente potrà accedere a funzioni come: creazione, modifica ed eliminazione di bacheche e ToDo, spostamento di ToDo fra bacheche ed all'interno della bacheca di appartenenza. L'utente potrà inoltre decidere di condividere ToDo con altri utenti, apparendo ad ogni utente che lo condivide.
L'utente potrà inoltre utilizzare funzioni di ricerca per mostrare: i ToDo in scadenza il giorno corrente, in una data specifica, ed effettuare una ricerca per titolo dei ToDo.

1.1 Analisi del modello dei dati

Dall'analisi della specifica del problema si può evincere la necessità della creazione di un modello dei dati che tiene traccia di utenti, bacheche, ToDo e della condivisione di ToDo con altri utenti.
Gli utenti del sistema saranno identificati dalle proprie credenziali di accesso al sistema e ad ogni utente sarà associata una lista di bacheche.
Le bacheche, appartenenti ognuna ad un utente, saranno caratterizzate da un titolo ed eventuale descrizione, oltre che ToDo dell'utente che conterranno.
I ToDo, appartenenti ognuno ad una bacheca, saranno caratterizzati da un titolo e di uno stato per segnarne il completamento ed opzionalmente: descrizione, link ad un URL correlato all'attività, immagine, data di scadenza, colore di sfondo. Ogni ToDo terrà poi eventualmente conto anche dalla lista di utenti con cui è convidiso.

[1] Un utente che si registra avrà inizialmente tre bacheche: *Università*, *Lavoro* e *Tempo Libero*

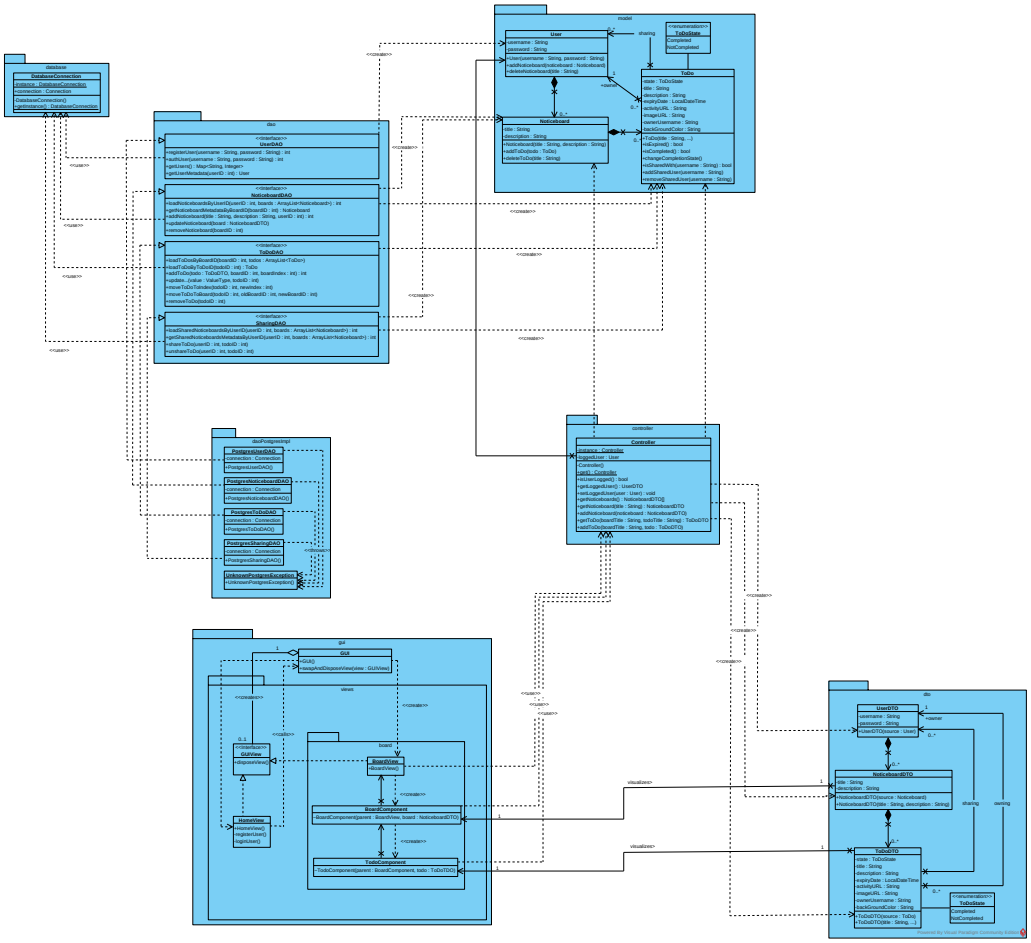
1.2 Analisi del modello dell'Interfaccia Grafica Utente

L'interfaccia grafica del sistema dovrà essere composta da un sistema che si occupa di controllare le viste del programma, generando inizialmente un form di registrazione/login, che permetterà all'utente di accedere al sistema. Conseguentemente all'accesso il sistema dovrà generare una vista principale che mostrerà le bacheche dell'utente ed i relativi ToDo e permetterà all'utente di interagire con essi.
La vista principale dovrà dunque svolgere le funzioni di visualizzazione e di gestione di bacheche e ToDo, oltre che attuare le funzioni di ricerca dei ToDo.
Ogni bacheca sarà rappresentata da una componente grafica che si occuperà di: visualizzare i ToDo contenuti nella bacheca; permettere all'utente di aggiungere altri ToDo; cambiare bacheca visualizzata.
Ogni ToDo sarà rappresentato da una componente grafica che si occuperà di visualizzarne gli attributi e permettendo all'utente di gestire il ToDo in modo da: variarne gli attributi; eliminarlo; modificare la lista di utenti che lo condividono; modificare la bacheca di appartenenza; modifiare la posizione in bacheca.

2 Modellazione UML

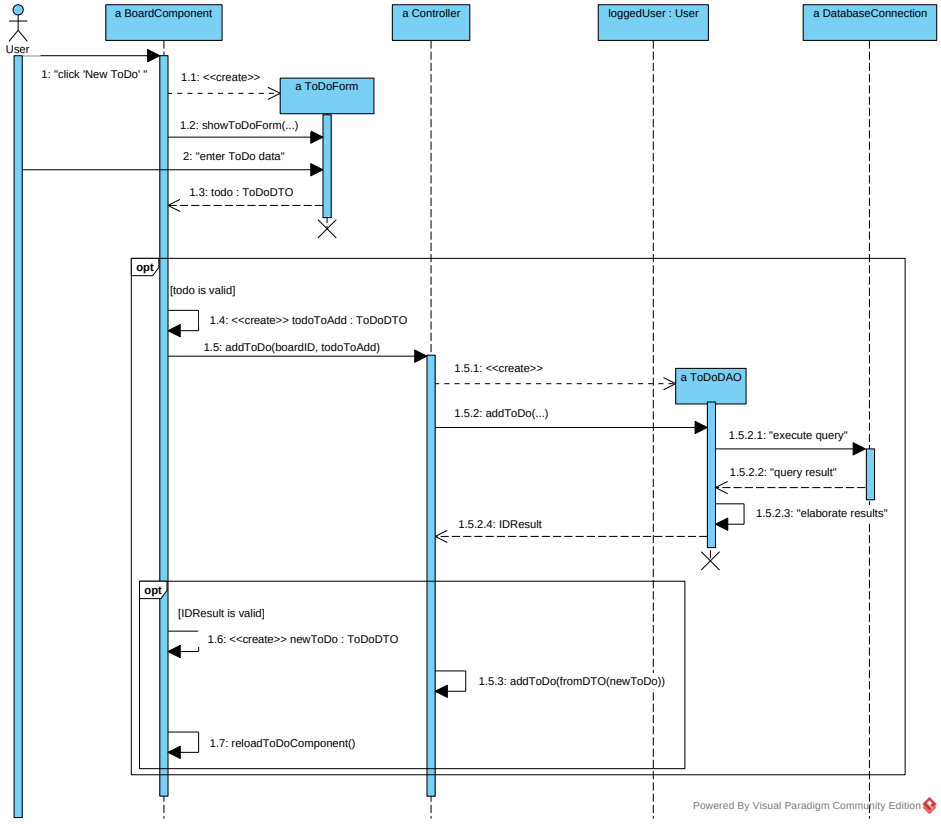
Di seguito sono presentati il Class Diagram del progetto di sistema e due Sequence Diagram che illustrano l'implementazione di alcune funzionalità del sistema.

2.1 Class Diagram del sistema



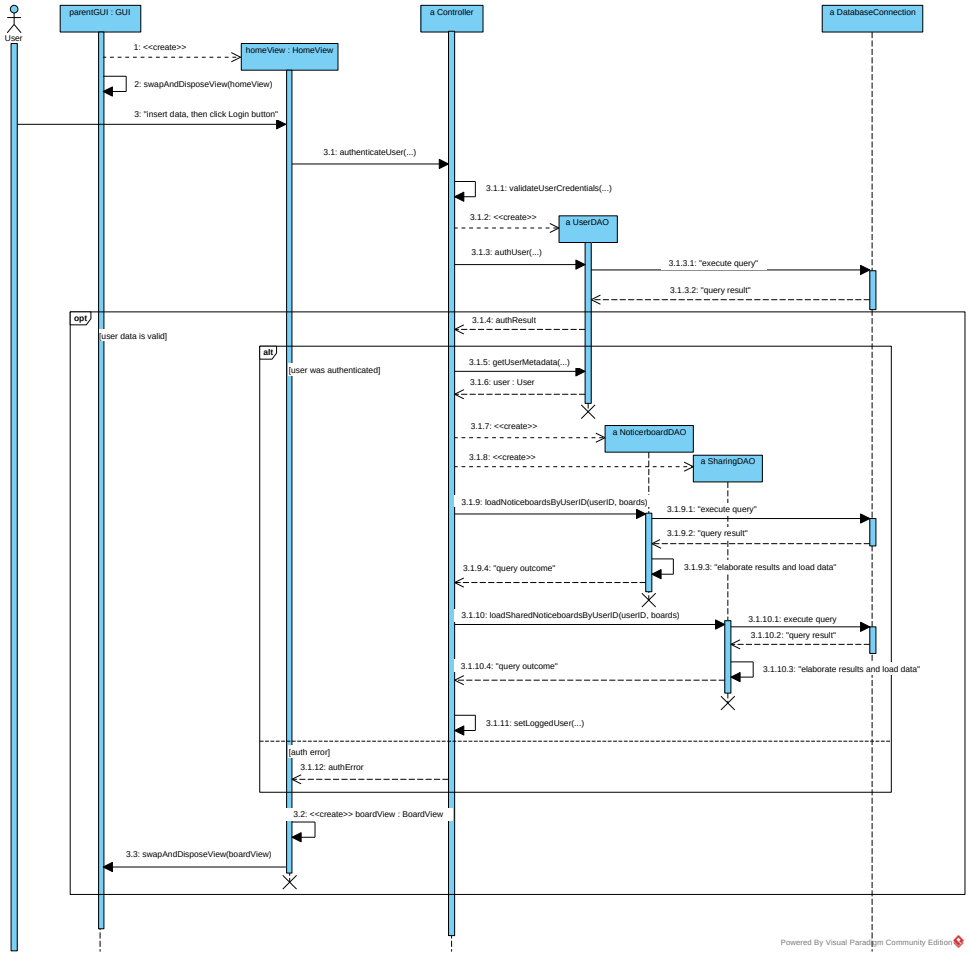
2.2 Sequence diagram - Aggiunta di un nuovo ToDo

Nel seguente Sequence Diagram sono descritti i principali scambi di messaggi fra entità del sistema allo scopo di illustrare la funzionalità di aggiunta di un nuovo ToDo in una bacheca.



2.3 Sequence diagram - Caricamento vista utente

Nel seguente Sequence Diagram sono descritti i principali scambi di messaggi fra entità del sistema allo scopo di illustrare il processo di autenticazione e caricamento delle bacheche e ToDo nella vista principale di un utente.



3 Progettazione del sistema

Per la progettazione del sistema si è adottato il paradigma ad oggetti del linguaggio **Java**, basandosi sul il modello architetturale **BCE + DAO (*Boundary-Control-Entity + Data Access Object*)**, che prevede la suddivisione del sistema in diverse componenti principali:

- **L'interfaccia utente (Boundary Layer)**, che si occupa di elaborare l'input utente inoltrando le richieste al Control Layer e di rappresentare i dati dell'applicazione.
- **Il Controller (Control Layer)**, che si occupa di processare le richieste dell'interfaccia utente in base gestendo la logica dell'applicazione, gestendo i dati del sistema tramite i Layer successivi.
- **Il Modello dei Dati (Entity Layer)**, che si occupa di gestire la rappresentazione interna dei dati del sistema.
- **Il Data Access Layer**, basata sulla libreria driver **JDBC** ed il Database Management System **PostgreSQL**, che si occupa di gestire la persistenza dei dati dell'applicazione.

3.1 Entity Layer - Package *model*

Il package *model* contiene le classi che rappresentano il modello dei dati del sistema, sviluppato nelle classi **ToDo, Noticeboard, User**.

3.1.1 ToDo

La classe ToDo rappresenta una attività.

Attributi[2]

ID	rappresenta l'ID unico del ToDo nel sistema.
state	rappresenta lo stato di completamento del ToDo.
title	il titolo assegnato al ToDo.
description	una breve descrizione del ToDo.
expiryDate	la data di scadenza entro il quale il ToDo è completabile.
activityURL	il link all'attività collegata al ToDo.
imageURL	rappresentante un link alla immagine del ToDo.
backgroundColor	rappresentante il colore di sfondo del ToDo.

Operazioni

- Operazioni per gestire gli attributi della classe.
- Operazioni di condivisione, che permettono di gestire la lista di utenti con cui è condiviso il ToDo.

[2]Gli attributi **evidenziati** sono obbligatori, quelli sottolineati sono *opzionali*.

3.1.2 Noticeboard

La classe Noticeboard rappresenta una collezione di ToDo.

Attributi

ID rappresenta l'ID unico della Noticeboard nel sistema.

title rappresenta il titolo della Noticeboard.

description rappresenta la descrizione della Noticeboard.

Operazioni

La classe avrà le seguenti operazioni:

- Operazioni per gestire gli attributi della classe.
- Operazioni per per gestire i ToDo.

3.1.3 User

La classe User rappresenta un utente del sistema.

Attributi

username rappresenta il nome utente.

password rappresenta la password dell'utente.

Operazioni

- Operazioni per gestire gli attributi della classe.
- Operazioni per gestire le Noticeboard dell'utente

3.2 Entity Layer - Package *dto*

Il package dto contiene le classi che rappresentano Data Transfer Object (oggetti che rappresentano immutabilmente altri oggetti) relativi agli oggetti del Data Model del sistema.
Le classi del package *dto* sono utilizzate per l'interscambio di informazioni tra i layer Boundary e Control.

3.2.1 ToDoDTO

La classe ToDoDTO rappresenta immutabilmente un ToDo nel data model.

Attributi

La classe ToDoDTO avrà gli stessi attributi della classe ToDo del data model ma, a differenza della classe del model, questi saranno immutabili.

Operazioni

La classe avrà le seguenti operazioni:

- Operazioni per la visualizzazione degli attributi della classe.
- Operazioni per visualizzare la lista di utenti con cui è condiviso il ToDo.

3.2.2 NoticeboardDTO

La classe **NoticeboardDTO** rappresenta immutabilmente una Noticeboard nel data model.

Attributi

La classe NoticeboardDTO avrà gli stessi attributi della classe Noticeboard del data model ma, a differenza della classe del model, questi saranno immutabili.

Operazioni

La classe conterrà operazioni per visualizzare gli attributi della classe.

3.2.3 UserDTO

La classe **UserDTO** rappresenta immutabilmente un User nel data model.

Attributi

La classe UserDTO avrà gli stessi attributi della classe User del data model ma, a differenza della classe del model, questi saranno immutabili.

Operazioni

La classe conterrà operazioni per visualizzare gli attributi della classe.

3.3 Control Layer - Package *controller*

Il package *controller* contiene la classe **Controller**, che rappresenta una interfaccia fra data model ed interfaccia grafica dell'applicazione.

3.3.1 Controller

La classe **Controller** si occupa di gestire le operazioni di un utente sulla GUI traducendole in operazioni verso il Data Model dell'applicazione e verso l'interfaccia di persistenza dati.

Attributi

loggedUser rappresentante l'utente autenticato nel sistema in un preciso istante.

Operazioni

La classe avrà le seguenti operazioni:

- Operazioni per la registrazione ed autenticazione degli utenti.
- Operazioni per gestire Noticeboard e ToDo.
- Operazioni per gestire lo spostamento di ToDo tra bacheche e la loro condivisione fra utenti.

3.4 Boundary Layer - Package *gui*

Il package *gui* contiene le classi che gestiscono l'interfaccia grafica del sistema, ossia l'insieme di viste, finestre di dialogo e componenti grafici che l'utente visualizzerà e con cui interagirà nel corso dell'uso dell'applicazione.

3.4.1 GUI

La classe **GUI** rappresenta l'interfaccia grafica dell'applicazione e si occupa di gestire le viste dell'applicazione, ossia l'insieme di finestre e dialoghi che l'utente vede e con cui interagisce.

Attributi

currentGUI tiene solo conto di quale GUIView sta attualmente venendo esposta.

Operazioni

La classe avrà una operazione per visualizzare una nuova vista e disfarsi della vista attuale.

3.5 Boundary Layer - Package *gui.views*

Il package *gui.views* contiene le interfacce e classi che rappresentano le viste dell'interfaccia grafica dell'applicazione.

3.5.1 GUIView

La interfaccia **GUIView** rappresenta una vista della GUI.

Attributi

L'interfaccia non ha attributi.

Operazioni

Le classi che implementeranno l'interfaccia dovranno contenere un'operazione per disfarsi della vista.

3.6 Boundary Layer - Package *gui.views.homeview*

3.6.1 HomeView

La classe **HomeView**, che implementa l'interfaccia **GUIView**, rappresenta l'interfaccia di dialogo per la registrazione ed autenticazione utente nell'applicazione e si occupa, interfacciandosi con la classe **Controller**, di gestire la validazione delle credenziali inserite dall'utente e successivamente di generare la vista principale dell'utente (BoardView).

Attributi

La classe non ha attributi.

Operazioni

La classe avrà le seguenti operazioni:

- Una operazione per validare le credenziali inserite dall'utente.
- Una operazione per registrare un utente.
- Una operazione per autenticare un utente.

La classe implementerà le seguenti operazioni di **GUIView**:

- Una operazione per disfarsi della vista.

3.7 Boundary Layer - Package *gui.views.boardview*

Il package *gui.views.boardview* contiene le classi che visualizzano i componenti del modello dei dati dell'applicazione.

3.7.1 BoardView

La classe **BoardView**, che implementa **GUIView**, rappresenta l'interfaccia principale dell'utente con l'applicazione, si occupa di gestire la visione dei dati dell'applicazione e di soddisfare le richieste dell'utente.

Attributi

La classe non ha attributi.

Operazioni

La classe avrà le seguenti operazioni:

- Operazioni per gestire le bacheche nella vista e la loro visualizzazione.
- Operazioni per gestire le funzionalità di ricerca dei ToDo.
- Operazioni per ricaricare i **BoardComponent** della componente.

La classe implementerà le seguenti operazioni di **GUIView**:

- Una operazione per disfarsi della vista.

3.7.2 BoardComponent

La classe **BoardComponent** è una componente grafica che rappresenta una **Noticeboard**, si occupa di gestire la visione della bacheca che rappresenta visualizzando i suoi **ToDo** (tramite l'uso di ToDoComponent). Inoltre gestisce la possibilità dell'utente di cambiare bacheca visualizzata ed aggiungere ToDo alla bacheca.

Attributi

parentBoardView rappresentante la **BoardView** che contiene la componente.

board rappresentante la **Noticeboard** a cui la componente grafica si riferisce.

Operazioni

La classe avrà le seguenti operazioni:

- Operazioni per ricaricare i **ToDoComponent** della componente.
- Operazioni per aggiungere nuovi **ToDo**.

3.7.3 ToDoComponent

La classe **ToDoComponent** è una componente grafica che rappresenta un **ToDo**, si occupa di gestire la visione del ToDo che rappresenta esponendone appropriatamente gli attributi presenti e dando la possibilità all'utente di gestire gli attributi del ToDo, modificare la sua posizione in una bacheca, lo spostamento fra bacheche ed infine di gestire la sua condivisione con altri utenti.

Attributi

parentBoardComponent rappresentante la **BoardComponent** che contiene la componente.

todo rappresentante il **ToDo** a cui la componente grafica si riferisce.

Operazioni

La classe avrà le seguenti operazioni:

- Operazioni per gestire gli attributi del ToDo.
- Operazioni per spostare il ToDo all'interno della bacheca.
- Operazioni per spostare il ToDo in un'altra bacheca.
- Operazioni per gestire la condivisione del ToDo con altri utenti.
- Operazioni per eliminare il ToDo.

3.8 Data Access Layer - Package *dao*

Il package *dao* contiene le interfacce che rappresentano il sistema di persistenza dati dell'applicazione, basato sul pattern DAO.
Le classi che implementano le sue interfacce sono usate dal Controller per gestire la persistenza dei dati del *model*.

3.8.1 ToDoDAO

L'interfaccia **ToDoDAO** è una interfaccia DAO che si occupa della gestione della gestione della persistenza dei **ToDo** nell'applicazione, si occupa di ciò dando la possibilità al Controller di gestire comunicazioni riguardanti ToDo, i loro attributi, posizione nella bacheca di appartenenza e spostamento fra bacheche, fra l'applicazione ed il meccanismo di persistenza dati desiderato.

Attributi

L'interfaccia non ha attributi.

Operazioni

Le classi che implementeranno l'interfaccia dovranno implementare:

- Operazioni per gestire i ToDo.
- Operazioni per gestire gli attributi dei ToDo.
- Operazioni per gestire la posizione dei ToDo nella bacheca di appartenenza e per spostare ToDo in un'altra bacheca.

3.8.2 NoticeboardDAO

L'interfaccia **NoticeboardDAO** è una interfaccia DAO che si occupa della gestione della gestione della persistenza delle **Noticeboard** nell'applicazione, si occupa di ciò dando la possibilità al Controller di gestire comunicazioni riguardanti le Noticeboard ed i loro attributi fra l'applicazione ed il meccanismo di persistenza dati desiderato.

Attributi

L'interfaccia non ha attributi.

Operazioni

Le classi che implementeranno l'interfaccia dovranno implementare:

- Operazioni per gestire le Noticeboard.
- Operazioni per gestire gli attributi delle Noticeboard.

3.8.3 UserDAO

L'interfaccia **UserDAO** è una interfaccia DAO che si occupa della gestione della gestione della persistenza degli **User** nell'applicazione, si occupa di ciò dando la possibilità al Controller di gestire comunicazioni riguardanti gli User fra l'applicazione ed il meccanismo di persistenza dati desiderato.

Attributi

L'interfaccia non ha attributi.

Operazioni

Le classi che implementeranno l'interfaccia dovranno implementare:

- Operazioni per registrare ed autenticare gli User.
- Operazioni per gestire gli User.

3.8.4 SharingDAO

L'interfaccia **SharingDAO** è una interfaccia DAO che si occupa della gestione della gestione della persistenza della condivisione di **ToDo** fra **User** nell'applicazione, si occupa di ciò dando la possibilità al Controller di gestire comunicazioni riguardanti lo sharing dei ToDo fra l'applicazione ed il meccanismo di persistenza dati desiderato.

Attributi

L'interfaccia non ha attributi.

Operazioni

Le classi che implementeranno l'interfaccia dovranno implementare:

- Operazioni per gestire la condivisione dei ToDo.

3.9 Data Access Layer - Package *database*

3.9.1 DatabaseConnection

La classe **DatabaseConnection** si occupa di gestire la connessione dell'applicazione ad un database PostgreSQL e dello scambio di informazioni delle query.

Attributi

connection rappresenta la connessione al database.

Operazioni

La classe avrà una operazione per gestire la connessione al Database.

3.10 Data Access Layer - Package *daopostgresimplementation*

Il package *daopostgresimplementation* contiene le classi che implementano le interfacce DAO descritte del package dao connessione con un DBMS PostgreSQL.

3.11 Altri package

3.11.1 Boundary Layer - Package *gui.components.forms*

Il package *gui.components.forms* classi di utilità per l'interfaccia grafica, come:

- La classe **ToDoForm**, una componente grafica che permette di visualizzare un form di dialogo per creare o modificare un ToDo inserendone gli attributi.
- La classe **NoticeboardForm**, una componente grafica che permette di visualizzare un form di dialogo per creare o modificare una Noticeboard inserendone gli attributi.

3.11.2 Boundary Layer - Package *gui.components*

Il package *gui.components* contiene classi di utilità per l'interfaccia grafica, come:

- La classe **ListComponent** è una componente grafica che permette di visualizzare una lista di stringhe.