|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Object  Design  Document  Progetto Fund.It   |  |  | | --- | --- | | Riferimento | Gruppo G13 | | Versione | 0.7 | | Scritto da | Luigi Crescenzo, Francesco Esposito, Sabato Genovese, Angelo Meo | |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autori |
| 04/01/2022 | 0.1 | Prima stesura | Sabato Genovese |
| 05/01/2022 | 0.2 | Scrittura [Object design goal](#_heading=h.35nkun2), [Object design trade-offs](#_heading=h.z337ya),  d[efinizioni, abbreviazioni e acronimi](#_heading=h.4i7ojhp) | Luigi Crescenzo,  Sabato Genovese |
| 06/01/2022 | 0.3 | Definizione dei class diagram per i vari sottosistemi individuati nel system design | Gruppo |
| 07/01/2022 | 0.4 | Definizione packages e class interfaces | Francesco Esposito,  Sabato Genovese |
| 09/01/2022 | 0.5 | Definizione dei design patterns da utilizzare | Gruppo |
| 15/01/2022 | 0.6 | Revisione del lavoro svolto | Gruppo |
| 02/02/2022 | 0.7 | Revisione e correzioni | Gruppo |

Sommario

[1 Introduzione 4](#_Toc95672094)

[1.1 Object design goals 4](#_Toc95672095)

[1.2 Object design trade-offs 5](#_Toc95672096)

[Spazio di memoria vs Tempi di risposta 5](#_Toc95672097)

[Funzionalità vs Testing 5](#_Toc95672098)

[Comprensibilità vs Tempo di sviluppo 5](#_Toc95672099)

[1.3 Linee guida per la documentazione dell’interfaccia 5](#_Toc95672100)

[1.4 Definizioni, abbreviazioni e acronimi 6](#_Toc95672101)

[1.5 Class Diagram 6](#_Toc95672102)

[Sottosistema Gestione Utente 6](#_Toc95672103)

[Sottosistema Gestione Categorie 7](#_Toc95672104)

[Sottosistema Gestione Segnalazioni 7](#_Toc95672105)

[Sottosistema Gestione FAQ 8](#_Toc95672106)

[Sottosistema Gestione Immagini 8](#_Toc95672107)

[Sottosistema Autenticazione 9](#_Toc95672108)

[Sottosistema Gestione Campagne 10](#_Toc95672109)

[Sottosistema Gestione Donazioni 11](#_Toc95672110)

[1.6 Riferimenti 12](#_Toc95672111)

[2 Packages 13](#_Toc95672112)

[3 Class Interfaces 15](#_Toc95672113)

[AutenticazioneService 15](#_Toc95672114)

[CampagnaService 17](#_Toc95672115)

[UtenteService 21](#_Toc95672116)

[CategoriaService 23](#_Toc95672117)

[DonazioniService 25](#_Toc95672118)

[FAQService 27](#_Toc95672119)

[SegnalazioniService 29](#_Toc95672120)

[ImmaginiService 31](#_Toc95672121)

[4 Design Patterns 32](#_Toc95672122)

[Proxy Pattern 32](#_Toc95672123)

[Singleton Pattern 33](#_Toc95672124)

[DAO Pattern 34](#_Toc95672125)

[5 Glossario 35](#_Toc95672126)

# 1 Introduzione

*Fund.It* si propone di aiutare il finanziamento di progetti attraverso il meccanismo del crowdfunding, al fine di finanziare progetti creativi e sociali.

Nella prima sezione del documento, saranno dedicati ai design goals e gli object design trade-offs e le linee guida per la fase di implementazione, riguardanti la nomenclatura, la documentazione e le convenzioni sui formati.

## Object design goals

**Riusabilità:**

Fund.it attraverso l’uso sapiente di design patterns fornirà un sistema software che garantirà un riutilizzo del codice.

**Robustezza:**

Il sistema deve garantire una buona robustezza reagendo a situazioni impreviste, controllando i possibili errori e gestendo le eccezioni.

**Incapsulamento:**

Il sistema garantisce la segretezza sui dettagli implementativi delle classi grazie all’utilizzo delle interfacce, rendendo possibile l’utilizzo di funzionalità offerte da diversi componenti o layer sotto forma di blackbox.

## Object design trade-offs

### Spazio di memoria vs Tempi di risposta

La creazione di oggetti costosi verrà effettuata in modo tale da ridurre lo spazio di memoria utilizzata. Per arrivare a ciò, i dati persistenti presenti sul database verranno trasferiti in maniera “pigra” (lazy) nel sistema. Ciò implica che saranno necessarie ulteriori operazioni di I/O da e verso il database per poter avere una visione completa di un oggetto che ha relazioni verso altri oggetti.

### Funzionalità vs Testing

Il sistema sarà sviluppato implementando tutte le funzionalità presenti nei requisiti funzionali a discapito della robustezza in quanto garantita solo per i requisiti con una priorità medio/alta.

### Comprensibilità vs Tempo di sviluppo

Le classi saranno scritte rispettando lo standard proposto da Sun per il linguaggio Java. In questo modo viene utilizzato uno stile unico di scrittura adottato da tutti gli sviluppatori del sistema, che andrà a migliorare la comprensibilità del codice attraverso l’utilizzo di commenti. Ciò facilita anche la manutenzione del codice delle classi. Per rispettare le regole dello standard c’è stato un aumento delle ore di sviluppo nella fase di implementazione.

## 1.3 Linee guida per la documentazione dell’interfaccia

Le linee guida includono una lista di regole che gli sviluppatori dovrebbero rispettare durante la progettazione delle interfacce. Per la costruzione dell’interfaccia abbiamo fatto riferimento alla convenzione Java nota come Sun Java Coding Conventions [Sun, 2009].

Di seguito una lista di link alle convenzioni usate per definire le linee guida:

• Java Sun: <https://checkstyle.sourceforge.io/sun_style.html>

• HTML: <https://www.w3schools.com/html/html5_syntax.asp>

## 1.4 Definizioni, abbreviazioni e acronimi

Vengono riportati di seguito alcune definizioni presenti nel documento:

**Package**: raggruppamento di classi, interfacce o file correlati;

**Design pattern**: template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità;

**Interfaccia**: insieme di firme dei metodi offerti dalla classe;

**Javadoc**: sistema di documentazione offerto da Java, che viene generato sotto forma di interfaccia in modo da rendere la documentazione accessibile e facilmente leggibile.

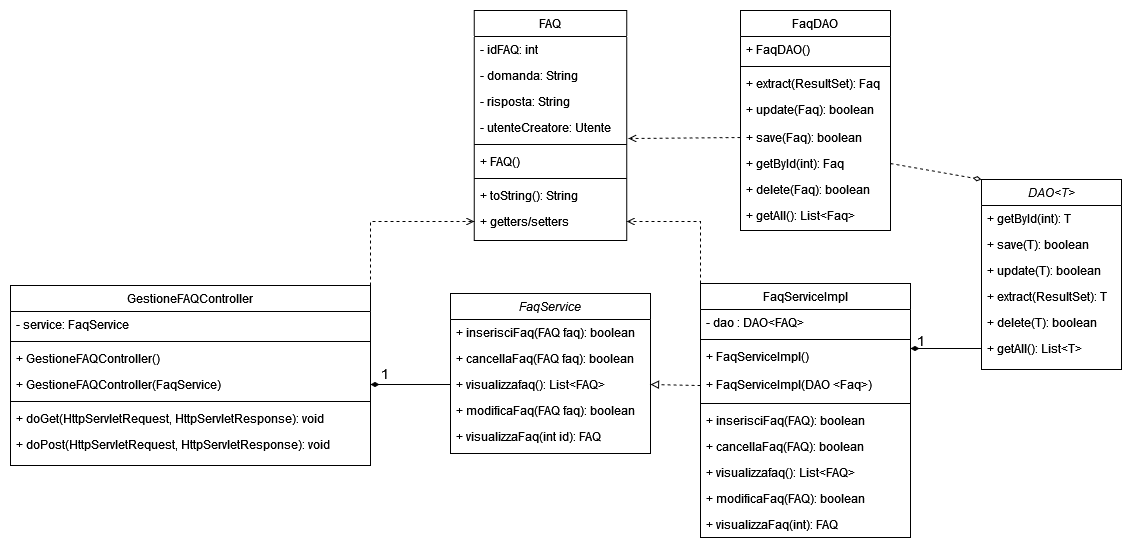
## 1.5 Class Diagram

### Sottosistema Gestione Utente

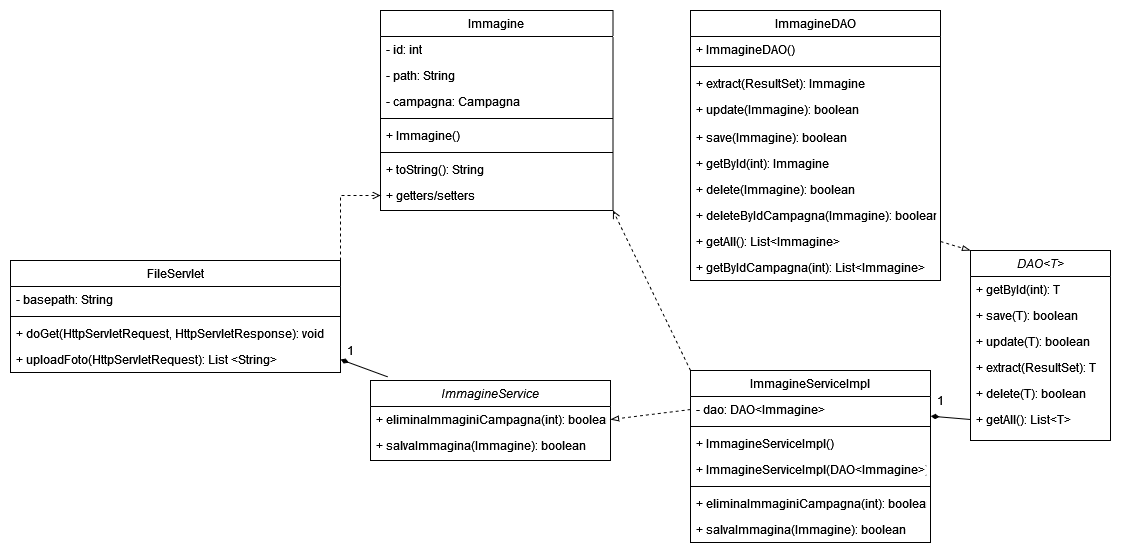
### Sottosistema Gestione Categorie

### Sottosistema Gestione Segnalazioni

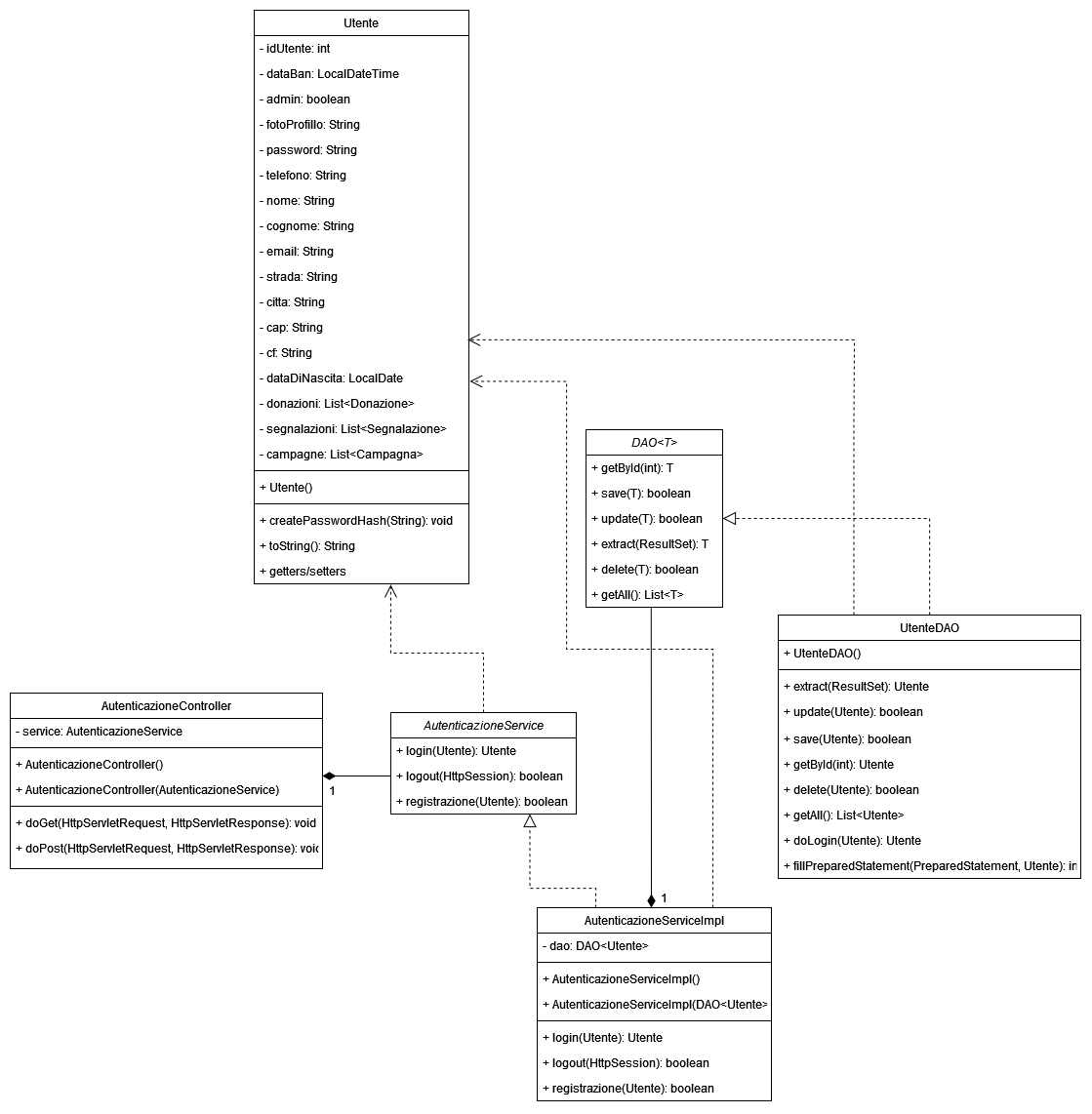
### Sottosistema Gestione FAQ



### Sottosistema Gestione Immagini



### Sottosistema Autenticazione



### Sottosistema Gestione Campagne

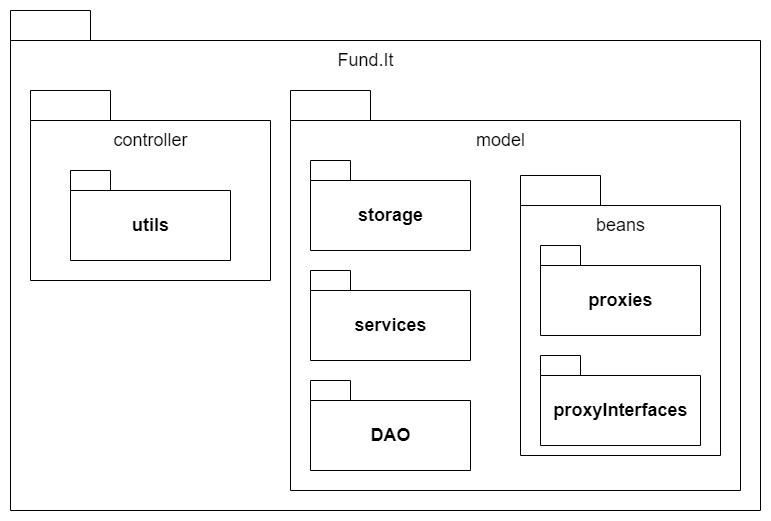
### Sottosistema Gestione Donazioni

## 1.6 Riferimenti

* Notazione per ENUM: <https://www.softwareideas.net/class-diagram-enum>
* Libri usati per la scelta dei design patterns: [Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides - Design Patterns - Elements of Reusable Object-Oriented Software (1994, Addison-Wesley)](https://www.oreilly.com/library/view/design-patterns-elements/0201633612/), [Object-Oriented Software Engineering: Using Uml, Patterns and Java: Using UML, Patterns and Java](https://www.amazon.it/Object-Oriented-Software-Engineering-Using-Patterns/dp/0130471100)

# 2 Packages

In questa sezione viene mostrata la struttura del sistema software Fund.It in package, in base a quanto definito nel documento di System Design. Tale suddivisione è dettata dalla struttura standard di Maven.



**Package Fund.It**

In questa sezione viene mostrata la struttura del package principale di Fund.It. La struttura è stata ottenuta da due principali scelte:

1. Creare un package separato per ogni sottosistema, uno destinato al layer controller*,* uno per il model ed uno per classi di utility.
2. Creare un package per i beans contenente le classi entity, uno per i DAO per l’accesso al DB, uno per i services e uno per lo storage che si occupa di fornire la connessione al DB.
3. Creare un package per i proxy contenente le classi che permettono di realizzare la lazy allocation di risorse e uno per le interfacce al fine di semplificare eventi di manutenzione evolutiva.

* **.idea**, contiene tutti le impostazioni specifiche per il progetto
* **deliverables**, contiene tutti i file che documentano il sistema software
* **src**, contiene tutti i file sorgente
  + **main**, contiene classi e risorse
    - **java**, contiene le classi Java relative alle componenti Controller e Model
    - **webapp**, contiene i file relativi alle componenti View
      * **css,** contiene gli Style Sheet CSS
      * **img,** contiene le immagini presenti sulla piattaforma
      * **js,** contiene i file JavaScript
      * **WEB-INF,** contiene i file .jsp
      * **index.jsp,** è il file che contiene la homepage della piattaforma
  + **test**, contiene classi e risorse per il testing
    - **java,** contiene le classi Java usate per il testing
* **target**, contiene tutti i file prodotti da Maven
* **pom.xml**, è un file che contiene informazioni sul progetto e dettagli di configurazione usati da Maven per effettuare il build del progetto.
* **README.md**

# 3 Class Interfaces

La documentazione completa è disponibile al sito: https://sabatogenovese.github.io/Fund.It/JavaDoc/

### AutenticazioneService

|  |  |
| --- | --- |
| **Nome classe** | AutenticazioneService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative all’autenticazione |
| **Metodi** | +login(Utente utente): Utente  +registrazione(Utente utente): boolean  +logout(HttpSession session): boolean |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | +login(Utente utente): Utente |
| **Descrizione** | Questo metodo permette il login di un utente registrato sulla piattaforma. |
| **Pre-condizione** | / |
| **Post-condizione** | **context:** AutenticazioneService:: login(Utente utente) |

|  |  |
| --- | --- |
| **Nome metodo** | +registrazione(Utente utente): boolean |
| **Descrizione** | Questo metodo permette la registrazione di un nuovo utente. |
| **Pre-condizione** | / |
| **Post-condizione** | **context:** AutenticazioneService:: registrazione(Utente utente)  **post:** UtenteDAO.save(utente) == true |

|  |  |
| --- | --- |
| **Nome metodo** | +logout(HttpSession session): boolean |
| **Descrizione** | Questo metodo permette il logout di un utente. |
| **Pre-condizione** | **context:** AutenticazioneService:: logout(HttpSession session)  **pre:** session.contains(“utente”) |
| **Post-condizione** | **context:** AutenticazioneService:: logout(HttpSession session)  **post:** not session.contains(“utente”) |

### CampagnaService

|  |  |
| --- | --- |
| **Nome classe** | CampagnaService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative alle campagne |
| **Metodi** | +ricercaCampagna(String text): List<Campagna>  +ricercaCampagnaperCategoria(String text): List<Campagna>  +creazioneCampagna(Campagna campagna): boolean  +modificaCampagna(Campagna campagna): boolean  +condividiCampagna(int idCampagna, HttpServletRequest request): Map<String, String>  +visualizzaCampagne(int size, int offset): Campagna  +trovaCampagna(int idCampagna): Campagna  +cancellaCampagna(Campagna campagna): boolean  +rimborsaDonazioni(Campagna campagna, CampagnaInterface proxy): boolean  +getAllCampagne(): List<Campagna>  +getActiveCampagne(): List<Campagna> |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | ricercaCampagna(String text): List<Campagna> |
| **Descrizione** | Questo metodo permette la ricerca di una campagna presente sulla piattaforma |
| **Pre-condizione** | / |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | ricercaCampagnaperCategoria(String text): List<Campagna> |
| **Descrizione** | Questo metodo permette la ricerca delle campagne sulla piattaforma in base alla categoria. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | creazioneCampagna(Campagna campagna): boolean |
| **Descrizione** | Questo metodo permette la creazione di una campagna sulla piattaforma. |
| **Pre-condizione** | **context:** CampagnaService:: creazioneCampagna(Campagna campagna)  **pre:** not getAll()Campagne.contains(campagna) && not campagna == null |
| **Post-condizione** | **context:** CampagnaService:: creazioneCampagna(Campagna campagna)  **post:** getAll()Campagne.contains(campagna) |

|  |  |
| --- | --- |
| **Nome metodo** | modificaCampagna(Campagna campagna): boolean |
| **Descrizione** | Questo metodo permette la modifica di una campagna presente sulla piattaforma. |
| **Pre-condizione** | **context:** CampagnaService:: creazioneCampagna(Campagna campagna)  **pre:** not campagna == null |
| **Post-condizione** | **context:** CampagnaService:: creazioneCampagna(Campagna campagna)  **post:** CampagnaDAO.update == true |

|  |  |
| --- | --- |
| **Nome metodo** | condividiCampagna(int idCampagna, HttpServletRequest request): Map<String, String> |
| **Descrizione** | Questo metodo permette la condivisione di una campagna     presente sulla piattaforma. |
| **Pre-condizione** | **context:** CampagnaService:: condividiCampagna(int idCampagna, HttpServletRequest request)  **pre:** not campagna == null |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaCampagne(int size, int offset): Campagna |
| **Descrizione** | Questo metodo permette la visualizzazione delle campagne presenti sulla piattaforma. |
| **Pre-condizione** | **context:** CampagnaService:: visualizzaCampagne(int size, int offset)  **pre:** size > 0 |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | trovaCampagna(int idCampagna): Campagna |
| **Descrizione** | Questo metodo cerca una campagna dato l’id. |
| **Pre-condizione** | **/** |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | cancellaCampagna(Campagna campagna): boolean |
| **Descrizione** | Questo metodo si occupa di cancellare una campagna dalla piattaforma |
| **Pre-condizione** | **context:** CampagnaService:: cancellaCampagna(Campagna campagna)  **pre:** not campagna == null |
| **Post-condizione** | **context:** CampagnaService:: cancellaCampagna(Campagna campagna)  **post:** not visualizzaCampagne.contains(campagna) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome metodo** | | rimborsaDonazioni(Campagna campagna, CampagnaInterface proxy): boolean | |
| **Descrizione** | | Questo metodo si occupa di rimborsare ai donatori eventuali donazioni di campagne cancellate. | |
| **Pre-condizione** | | **context:** CampagnaService:: rimborsaDonazioni(Campagna campagna, CampagnaInterface proxy)  **pre:** not campagna == null | |
| **Post-condizione** | | / | |
|  | **Nome metodo** | getAllCampagne(): List<Campagna> |
|  | **Descrizione** | Questo metodo si occupa di prendere tutte le campagne sulla piattaforma |
|  | **Pre-condizione** | / |
|  | **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | getActiveCampagne(): List<Campagna> |
| **Descrizione** | Questo metodo si occupa di prendere tutte le campagne attive sulla piattaforma |
| **Pre-condizione** | / |
| **Post-condizione** | / |

### UtenteService

|  |  |
| --- | --- |
| **Nome classe** | UtenteService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative agli utenti |
| **Metodi** | +visualizzaDashboardUtente(int id): Utente  +modificaProfilo(Utente utente): boolean  +visualizzaUtenti(Utente richiedente): Liste <Utente>  +promuoviDeclassaUtente(Utente richiedente, Utente soggetto): boolean  +sospensioneUtente(Utente utente): boolean |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaDashboardUtente(int id): Utente |
| **Descrizione** | Questo metodo si occupa di prelevare le informazioni di un utente della piattaforma. |
| **Pre-condizione** | / |
| **Post-condizione** | **context:** UtenteService:: visualizzaDashboardUtente(int id)  **post:** UtenteDAO.getById == true |

|  |  |
| --- | --- |
| **Nome metodo** | modificaProfilo(Utente utente): boolean |
| **Descrizione** | Questo metodo si occupa di modificare le informazioni di un utente della piattaforma. |
| **Pre-condizione** | **context:** UtenteService:: modificaProfilo(Utente utente)  **pre:** not utente == null |
| **Post-condizione** | **context:** UtenteService:: modificaProfilo(Utente utente)  **post:** UtenteDAO.update == true |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaUtenti(Utente richiedente): List<Utente> |
| **Descrizione** | Questo metodo si occupa di recuperare la lista degli utenti iscritti alla piattaforma. |
| **Pre-condizione** | **context:** UtenteService:: visualizzaUtenti(Utente richiedente)  **pre:** not richiedente== null&& richiedente.isAdmin == true |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | promuoviDeclassaUtente(Utente richiedente, Utente soggetto): boolean |
| **Descrizione** | Questo metodo si occupa di promuovere/declassare un utente. |
| **Pre-condizione** | **context:** UtenteService:: visualizzaUtenti(Utente richiedente, Utente soggetto)  **pre:** not richiedente== null&& richiedente.isAdmin == true && not soggetto== null |
| **Post-condizione** | **context:** UtenteService:: visualizzaUtenti(Utente richiedente, Utente soggetto)  **post:** UtenteDAO.update == true |

|  |  |
| --- | --- |
| **Nome metodo** | sospensioneUtente(Utente utente): boolean |
| **Descrizione** | Questo metodo si occupa di sospendere un utente. |
| **Pre-condizione** | **context:** UtenteService:: sospensioneUtente(Utente utente)  **pre:** not utente== null |
| **Post-condizione** | **context:** UtenteService:: sospensioneUtente(Utente utente)  **post:** UtenteDAO.update == true |

### CategoriaService

|  |  |
| --- | --- |
| **Nome classe** | CategoriaService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative alle categorie |
| **Metodi** | +inserisciCategoria(Categoria categoria): boolean  +modificaCategoria(Categoria categoria): boolean  +visualizzaCategorie: List<Categoria>  +visualizzaCategoria(Categoria): Categoria |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | inserisciCategoria(Categoria categoria): boolean |
| **Descrizione** | Questo metodo si occupa di aggiungere una categoria sulla piattaforma. |
| **Pre-condizione** | **context:** CategoriaService:: inserisciCategoria(Categoria)  **pre:** not categoria== null |
| **Post-condizione** | **context:** CategoriaService:: inserisciCategoria(Categoria)  **post:** visualizzaCategorie.contains(categoria) |

|  |  |
| --- | --- |
| **Nome metodo** | modificaCategoria(Categoria categoria): boolean |
| **Descrizione** | Questo metodo si occupa di modificare una categoria presente sulla piattaforma. |
| **Pre-condizione** | **context:** CategoriaService:: modificaCategoria(Categoria)  **pre:** not categoria== null |
| **Post-condizione** | **context:** CategoriaService:: modificaCategoria(Categoria)  **post:** CategoriaDAO.update == true |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaCategorie(): List<Categorie> |
| **Descrizione** | Questo metodo si occupa di recuperare le categorie presenti sulla piattaforma. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaCategoria(Categoria categoria): Categoria |
| **Descrizione** | Questo metodo si occupa di recuperare una categoria sulla piattaforma. |
| **Pre-condizione** | **context:** CategoriaService:: visualizzaCategoria(Categoria categoria)  **pre:** not categoria== null |
| **Post-condizione** | / |

### DonazioniService

|  |  |
| --- | --- |
| **Nome classe** | DonazioniService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative alle donazioni |
| **Metodi** | +effettuaDonazione(Donazione d): boolean  +visualizzaDonazioni(Utente): List<Donazione>  +commenta(Donazione d): boolean  +visualizzaDonazioni(): List<Donazione> |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | effettuaDonazione(Donazione d): boolean |
| **Descrizione** | Questo metodo si occupa di effettuare una donazione ad una campagna |
| **Pre-condizione** | **context:** DonazioniService:: effettuaDonazione(Donazione d)  **pre:** not d== null |
| **Post-condizione** | **context:** DonazioniService:: effettuaDonazione(Donazione d)  **post:** visualizzaDonazioni.contains(d) |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaDonazioni(Utente u): List<Donazione> |
| **Descrizione** | Questo metodo si occupa di recuperare le donazioni di un utente iscritto alla piattaforma. |
| **Pre-condizione** | **context:** DonazioniService:: visualizzaDonazioni(Utente u)  **pre:** not u== null |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | commenta(Donazione d): boolean |
| **Descrizione** | Questo metodo si occupa di recuperare le donazioni di un utente iscritto alla piattaforma. |
| **Pre-condizione** | **context:** DonazioniService:: commenta(Donazione d)  **pre:** not d== null |
| **Post-condizione** | **context:** DonazioniService:: commenta(Donazione d)  **post:**  DonazioneDAO.update== true |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaDonazioin(): List<Donazioni> |
| **Descrizione** | Questo metodo si occupa di recuperare le donazioni presenti sulla piattaforma. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

### FAQService

|  |  |
| --- | --- |
| **Nome classe** | FAQService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative alle FAQ |
| **Metodi** | +inserisciFAQ(FAQ faq): boolean  +cancellaFAQ(FAQ faq): boolean  +modificaFAQ(FAQ faq): boolean  +visualizzaFAQ(): List<FAQ>  +visualizzaFAQ(int idFAQ): FAQ |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | inserisciFAQ (FAQ faq): boolean |
| **Descrizione** | Questo metodo si occupa di inserire una FAQ sulla piattaforma. |
| **Pre-condizione** | **context:** FAQService:: inserisciFAQ(FAQ faq)  **pre:** not faq== null |
| **Post-condizione** | **context:** FAQService:: inserisciFAQ(FAQ faq)  **post:** visualizzaFAQ.contains(faq) |

|  |  |
| --- | --- |
| **Nome metodo** | cancellaFAQ (FAQ faq): boolean |
| **Descrizione** | Questo metodo si occupa di eliminare una FAQ dalla piattaforma. |
| **Pre-condizione** | **context:** FAQService:: cancellaFAQ(FAQ faq)  **pre:** not faq== null |
| **Post-condizione** | **context:** FAQService:: cancellaFAQ(FAQ faq)  **post:** FaqDAO.delete = true |

|  |  |
| --- | --- |
| **Nome metodo** | modificaFAQ (FAQ faq): boolean |
| **Descrizione** | Questo metodo si occupa di modificare una FAQ sulla piattaforma. |
| **Pre-condizione** | **context:** FAQService:: modificaFAQ(FAQ faq)  **pre:** not faq== null |
| **Post-condizione** | **context:** FAQService:: modificaFAQ(FAQ faq)  **post:** FaqDAO.update = true |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaFAQ (FAQ faq): List<FAQ> |
| **Descrizione** | Questo metodo si occupa di recuperare una FAQ sulla piattaforma. |
| **Pre-condizione** | **context:** FAQService:: visualizzaFAQ(FAQ faq)  **pre:** not faq== null |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | visualizzaFAQ (): List<FAQ> |
| **Descrizione** | Questo metodo si occupa di recuperare tutte le FAQ sulla piattaforma. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

### SegnalazioniService

|  |  |
| --- | --- |
| **Nome classe** | SegnalazioniService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative alle segnalazioni |
| **Metodi** | +trovaSegnalazioni(): List<Segnalazione>  +trovaSegnalazione(int idSegnalazione)  +risolviSegnalazione(int idSegnalazione, StatoSegnalazione stato);  + segnalaCampagna(Segnalazione segnalazione) |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | trovaSegnalazioni(): List<Segnalazione> |
| **Descrizione** | Questo metodo si occupa di recuperare tutte le segnalazioni sulla piattaforma. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | trovaSegnalazione(int idSegnalazione): List<Segnalazione> |
| **Descrizione** | Questo metodo si occupa di recuperare una segnalazione sulla piattaforma |
| **Pre-condizione** | / |
| **Post-condizione** | / |

|  |  |
| --- | --- |
| **Nome metodo** | risolviSegnalazione(int idSegnalazione, StatoSegnalazione stato): List<Segnalazione> |
| **Descrizione** | Questo metodo si occupa di recuperare una segnalazione sulla piattaforma |
| **Pre-condizione** | **context:** SegnalazioniService:: risolviSegnalazione(int idSegnalazione, StatoSegnalazione stato)  **pre:** not StatoSegnalazione== null |
| **Post-condizione** | **context:** SegnalazioniService:: risolviSegnalazione(int idSegnalazione, StatoSegnalazione stato)  **post:** SegnalazioneDAO.update == true |

|  |  |
| --- | --- |
| **Nome metodo** | segnalaCampagna(Segnalazione segnalazione): boolean |
| **Descrizione** | Questo metodo si occupa di recuperare una segnalazione sulla piattaforma |
| **Pre-condizione** | **context:** SegnalazioniService:: segnalaCampagna(Segnalazione segnalazione)  **pre:** not segnalazione== null |
| **Post-condizione** | **context:** SegnalazioniService:: segnalaCampagna(Segnalazione segnalazione)  **post:** SegnalazioneDAO.update == true |

### ImmaginiService

|  |  |
| --- | --- |
| **Nome classe** | ImmaginiService |
| **Descrizione** | Questa classe permette di gestire le operazioni relative alle immagini |
| **Metodi** | +salvaImmagine(Immagine immagine): boolean  +eliminaImmaginiCampagne(int idCampagna): boolean |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome metodo** | salvaImmagine(Immagine immagine): boolean |
| **Descrizione** | Questo metodo si occupa di recuperare una segnalazione sulla piattaforma |
| **Pre-condizione** | **context:** ImmagineService:: salvaImmagine(Immagine immagine)  **pre:** not immagine== null |
| **Post-condizione** | **context:** ImmagineService:: salvaImmagine(Immagine immagine)  **post:** ImmagineDAO.save == true |

|  |  |
| --- | --- |
| **Nome metodo** | eliminaImmaginiCampagna(Immagine immagine): boolean |
| **Descrizione** | Questo metodo si occupa di recuperare una segnalazione sulla piattaforma |
| **Pre-condizione** | **context:** ImmagineService:: eliminaImmaginiCampagna(Immagine immagine)  **pre:** not immagine== null |
| **Post-condizione** | / |

# 4 Design Patterns

Il sistema da noi proposto implementa tre design patterns: Singleton, Proxy e DAO.

## Proxy Pattern

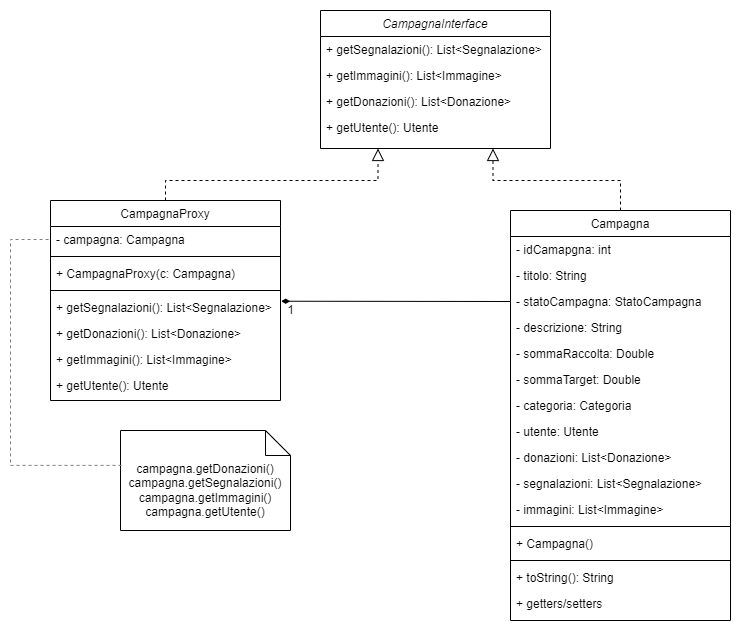
La motivazione per la quale si è scelto di utilizzare questo pattern è per ritardare l’inizializzazione dei campi di alcuni oggetti che sono costosi, in modo da portarli in memoria soltanto quando vi è la necessità di averli.

In questo modo possiamo creare oggetti costosi on demand, in modo da ottimizzare l’impatto che questi oggetti hanno in memoria.

La soluzione è quella di utilizzare una classe ***SubjectProxy***che implementa la stessa interfaccia di ***Subject***

Per ***Subject*** si intende qualsiasi classe che richiede una creazione ed inizializzazione costosa in termini di impatto sulla memoria ed eventuali operazioni di I/O.

Le classi che fungono da proxy per un particolare tipo di classe hanno come attributo un riferimento all’oggetto dello stesso tipo della classe di cui implementano l’interfaccia, in modo tale da oscurare l’utilizzatore sul tipo di oggetto (reale o proxy) con la quale sta interagendo.

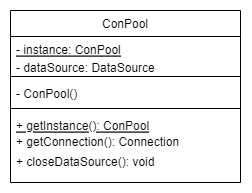
In particolare, gli oggetti delle classi proxy da noi implementate vengono definiti virtual proxy, ovvero un tipo particolare di proxy che creano e inizializzano oggetti di una classe della quale hanno un riferimento.

## Singleton Pattern

Questo pattern ci assicura che di una classe venga istanziato solo un oggetto, e di provvedere un punto globale di accesso ad esso.

In alcuni contesti è importante avere un’unica istanza di una classe con la quale altri oggetti possono comunicare.

Una soluzione per implementare questo pattern è quella di delegare la responsabilità della creazione della sola istanza di un tipo di classe alla classe stessa. La classe in questione deve anche offrire un modo per accedere all’istanza creata.

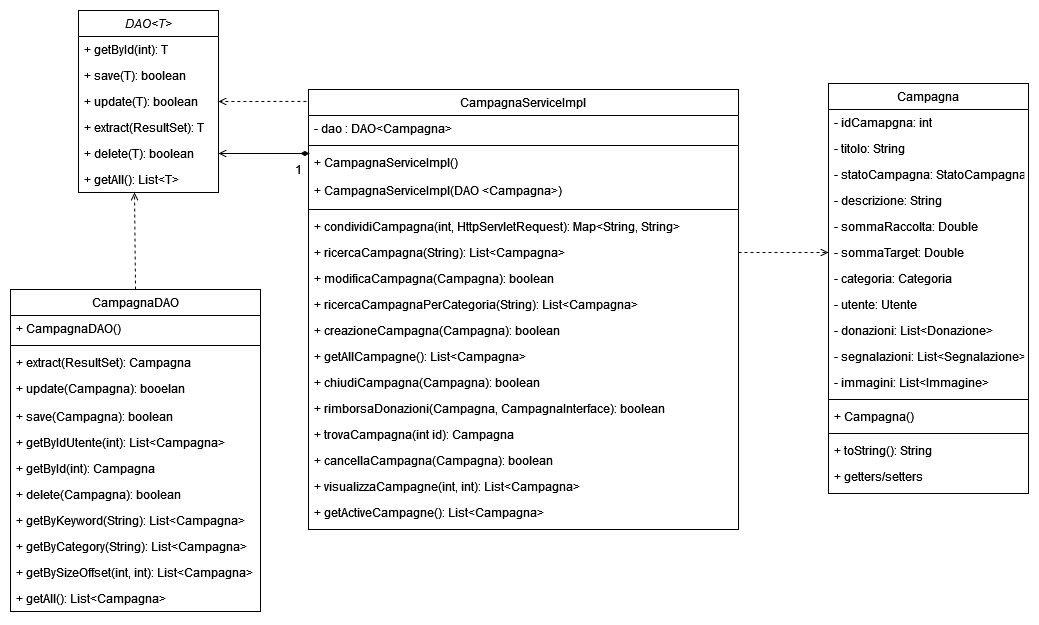


## DAO Pattern

Un DAO (Data Access Object) è un pattern che offre un’interfaccia astratta per alcuni tipi di database. Il DAO fornisce alcune operazioni specifiche sui dati senza esporre i dettagli del database. I DAO sono utilizzabili nella maggior parte dei linguaggi e la maggior parte dei software con bisogni di persistenza. Fund.It è una web application che punta a digitalizzare e gestire le raccolte fondi, ha bisogno pertanto di poter interagire con database in modo rapido e sicuro nonostante la moltitudine di query per gestire i dati persistenti.

Nel nostro sistema, le classi che hanno intenzione di eseguire le operazioni CRUD sulle classi entity dal database lo faranno tramite un’interfaccia unica (DAO) che espone dei metodi appositamente scritti per queste operazioni.

Qui è riportato un esempio dell’applicazione del pattern DAO utilizzato in Fund.It con alcune relazioni tra le altre componenti del nostro sistema software



# 5 Glossario

* Web Application: applicazione accessibile via Web per mezzo di una rete Internet.
* Web Server: applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un Web Browser.