# Object Design Document

# Shareboard

| | |
|---|---|
| Riferimento | ODD_G07 |
| Versione | 1.0 |
| Data | 10/03/2022 |
| Destinatario | Prof. Carmine Gravino |
| Presentato da | Antonio Romano (AR), Alessandro Saverio De Maio (AM), Carmine Leo (CL) |

# Revision History

| Data | Versione | Descrizione | Autori |
|---|---|---|---|
| 15/01/2022 | 0.1 | • Introduzione. | AR, AM, CL |
| 02/02/2022 | 0.2 | • Packages.<br>• Class Interfaces. | AR, AM, CL |
| 04/02/2022 | 0.3 | • Design Patterns | Antonio Romano |
| 07/02/2022 | 0.4 | • Glossario | AR, AM, CL |
| 10/03/2022 | 1.0 | • Revisione generale | Antonio Romano |

# Sommario

# Introduzione

## Object Design Trade-offs

- **Spazio vs Velocità:**
  Il sistema darà priorità alla velocità per fornire una buona esperienza utente a discapito della memoria, per questo motivo si utilizzeranno meccanismi di caching e di ridondanza.
- **Tempo di rilascio vs Funzionalità:**
  Se i tempi di rilascio di tutte le funzionalità non possono essere rispettati, saranno rilasciate soltanto le funzionalità che hanno la massima priorità in modo da rientrare nei tempi previsti.
- **Tempo di rilascio vs Qualità:**
  Per rispettare i tempi di rilascio in caso di ritardi, si concentreranno gli sforzi per ridurre il numero di bug nelle funzionalità ad alta priorità, successivamente al rilascio saranno sistemati tutti i restanti difetti del codice.

## Linee Guida per la Documentazione delle Interfacce

Di seguito sono elencate delle regole e convenzioni per la stesura del codice.

La conformità a tali convenzioni risulta vincolante per l'approvazione del codice nel master branch.

- Le pagine JSP devono essere prive di Scriptlet
- I file sorgente devono essere codificati in UTF-8
- I nomi delle classi in Java devono essere in PascalCase
- I nomi delle classi in Java devono essere al singolare
  - Ad eccezione di classi contenenti solo membri statici (Ad esempio common.DateUtils)
- I nomi degli attributi e dei metodi in Java devono essere in camelCase
- I nomi delle tabelle nel database devono essere in snake_case

La conformità a tali convenzioni non risulta vincolante per l'approvazione, ma è caldamente consigliata:

- Google Java Style Guide
- HTML

## Definizioni, acronimi e abbreviazioni

- **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 signature delle operazioni offerte dalla classe;
- **Interceptor:** metodo invocato prima (o dopo) l'esecuzione di un'azione in modo trasparente. Usato tipicamente nell'Aspect Oriented Programming

## Riferimenti

- Repository Github
- Generated Reports
  - Contiene Javadoc, report Jacoco, Maven Failsafe report, Maven Surefire report, Checkstyle report
- Elenco delle dipendenze

### Documenti

- System Design Document
- Object Design Document
- Test Plan

- Matrice di tracciabilità
- Manuale di installazione
- Manuale utente

### Bibliografia

Bruegge, B., & Dutoit, A. H. (2014). *Object-oriented software engineering.* Pearson.

Fowler, M. (2015). *Patterns of enterprise application architecture.* Addison-Wesley.

Gamma, E., Helm, R., & Johnson, R. (1998). *Design patterns.* Addison Wesley.

Oracle. (2022). *Core J2EE Patterns - Intercepting Filter.* Retrieved from Oracle: http://www.oracle.com/technetwork/java/interceptingfilter-142169.html

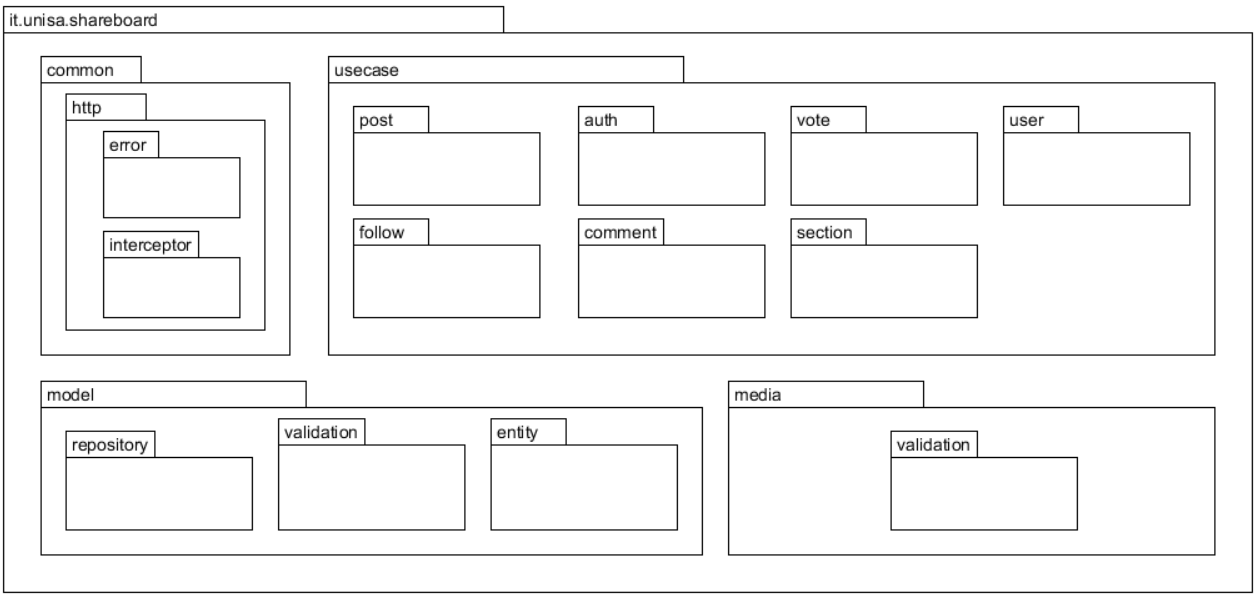
Sommerville, I. (2019). *Software engineering, tenth edition.* Pearson Education.

## Packages

Di seguito saranno mostrate le suddivisioni del sistema in package, in base a quanto definito nel documento di System Design e la struttura delle directory.

- **.github**
- **.idea**
- **maven_site**
- **sql**, contiene un dump del database
- **src**, contiene i file sorgente
    - **main**
        - **java**, contiene le classi Java relative alle Servlet, ai servizi e alla persistenza
        - **resources**, contiene i file xml relativi alla persistenza
        - **webapp**, contiene i file relativi alle componenti View
            - **CSS,** contiene i file CSS
            - **images**, contiene le immagini utilizzate dalla piattaforma
            - **js**, contiene i file JavaScript
            - **WEB-INF**, contiene le pagine JSP
    - **test**
        - **java**, contiene le classi necessarie per l'implementazione del testing
- **target**, contiene i file prodotti dal sistema di build Maven

Di seguito una panoramica dei package contenuti della cartella **src** dei file sorgente.




# Class interfaces

Per motivi di leggibilità, le interfacce disponibili potranno essere consultate tramite il Javadoc al link fornito di seguito, hostato tramite Github Pages.

**Shareboard Javadoc:** https://ra-c.github.io/shareboard_is/apidocs/index.html

## Authentication Service

| **Nome Classe** | AuthenticationService |
|---|---|
| **Descrizione** | Questa classe permette di gestire le funzioni relative all'autenticazione di un utente. |
| **Metodi** | +authenticate(String username, String password): Boolean<br>+getCurrentUser(): CurrentUser |
| **Invariante di classe** | / |

| **Nome Metodo** | +authenticate(String username, String password) |
|---|---|
| **Descrizione** | Questo metodo permette di autenticare un utente. |
| **Pre-condizione** | / |
| **Post-condizione** | **Context:** AuthenticationService:: authenticate(String username, String password)<br>**Post:** userExists(username) && passwordMatch(password) == true |

| **Nome Metodo** | +getCurrentUser() |
|---|---|
| **Descrizione** | Questo metodo restituisce l'utente attualmente loggato. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

## Ban Service

| **Nome Classe** | BanService |
|---|---|
| **Descrizione** | Questa classe permette di gestire le funzioni relative al ban per gli utenti. |
| **Metodi** | +addBan(Instant date, int userId): int<br>+removeBan(int banId): void<br>+retrieveUserBan(int userId): List<BanDTO> |
| **Invariante di classe** | / |

| **Nome Metodo** | + addBan(Instant date, int userId) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere un ban ad un utente esistente, specificando una data futura di fine ban e l'id relativo all'utente. |
| **Pre-condizione** | **Context:** BanService:: addBan(Instant date, int userId)<br>**Pre:** date != null && date.isAfter(currentDate) && userExists(userId) |
| **Post-condizione** | **Context:** BanService:: addBan(Instant date, int userId)<br>**Post:** userBanned(userId) == true |

| **Nome Metodo** | + removeBan(int banId) |
|---|---|
| **Descrizione** | Questo metodo permette di rimuovere il ban ad un utente. |
| **Pre-condizione** | **Context:** BanService:: removeBan(int banId)<br>**Pre:** banExists(banId) == true |
| **Post-condizione** | **Context:** BanService:: removeBan(int banId)<br>**Post:** banExists(banId) == false |

| **Nome Metodo** | + retrieveUserBan(int userId) |
|---|---|
| **Descrizione** | Questo metodo restituisce la lista dei ban relativi ad un utente. |
| **Pre-condizione** | **Context:** BanService:: retrieveUserBan (int userId)<br>**Pre:** userExists(userId) == true |
| **Post-condizione** | / |

## Comment Service

| **Nome Classe** | CommentService |
|---|---|
| **Descrizione** | Questa classe permette di gestire le funzionalità relative ai commenti. |
| **Metodi** | +getPostComments(int postId): Map<Integer, List<CommentDTO><br>+getReplies(int commentId): Map<Integer, List<CommentDTO><br>+getComment(int id): CommentDTO<br>+delete(int id): void<br>+editComment(int id, String text): void<br>+newComment(String text, int postId): int<br>+newCommentReply(String text, int parentCommentId): int |
| **Invariante di classe** | / |

| **Nome Metodo** | + getPostComments (int postId) |
|---|---|
| **Descrizione** | Questo metodo restituisce i commenti di un post. |
| **Pre-condizione** | **Context:** CommentService:: getPostComments(int postId)<br>**Pre:** PostExists(postId) == true |
| **Post-condizione** | / |

| **Nome Metodo** | + getReplies (int commentId) |
|---|---|
| **Descrizione** | Questo metodo restituisce le risposte ai commenti. |
| **Pre-condizione** | **Context:** CommentService:: getReplies(int commentId)<br>**Pre:** CommentExists(commentId) == true |
| **Post-condizione** | / |

| **Nome Metodo** | + getComment (int id) |
|---|---|
| **Descrizione** | Questo metodo restituisce un commentDTO derivato da un commento. |
| **Pre-condizione** | **Context:** CommentService:: getComment(int id)<br>**Pre:** CommentExists(id) == true |
| **Post-condizione** | / |

| **Nome Metodo** | + delete (int id) |
|---|---|
| **Descrizione** | Questo metodo elimina un commento. |
| **Pre-condizione** | **Context:** CommentService:: delete(int id)<br>**Pre:** CommentExists(id) == true |
| **Post-condizione** | **Context:** CommentService:: delete(int id)<br>**Post:** CommentExists(id) == false |

| **Nome Metodo** | + editComment (int id, String text) |
|---|---|
| **Descrizione** | Questo metodo permette di modificare un commento. |
| **Pre-condizione** | **Context:** CommentService:: editComment(int id, String text)<br>**Pre:** CommentExists(id) && notBlank(text) && numChar(text) <=65535 |
| **Post-condizione** | / |

| **Nome Metodo** | + newComment (String text, int postId) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere un commento ad un post. |
| **Pre-condizione** | **Context:** CommentService:: newComment(String text, int postId)<br>**Pre:** notBlank(text) && numChar(text) <=65535 && postExists(postId) |
| **Post-condizione** | **Context:** CommentService:: newComment(String text, int postId)<br>**Post:** CommentExists(newCommentId) == true |

| **Nome Metodo** | + newCommentReply (String text, int parentCommentId) |
|---|---|
| **Descrizione** | Questo metodo permette di rispondere ad un commento. |

| | |
|---|---|
| **Pre-condizione** | **Context:** CommentService:: newCommentReply(String text, int parentCommentId)<br>**Pre:** notBlank(text) && numChar(text) <=65535 && CommentExists(parentCommentId) |
| **Post-condizione** | **Context:** CommentService:: newCommentReply(String text, int parentCommentId)<br>**Post:** CommentExists(newCommentId) == true |

## Follow Service

| | |
|---|---|
| **Nome Classe** | FollowService |
| **Descrizione** | Questa classe permette di gestire le funzioni relative ai follow. |
| **Metodi** | +follow(int sectionId): void<br>+unfollow(int sectionId): void |
| **Invariante di classe** | / |

| | |
|---|---|
| **Nome Metodo** | +follow(int sectionId) |
| **Descrizione** | Questo metodo permette di aggiungere un follow ad una sezione. |
| **Pre-condizione** | **Context:** FollowService:: follow(int sectionId)<br>**Pre:** SectionExists(sectionId) == true |
| **Post-condizione** | **Context:** FollowService:: follow(int sectionId)<br>**Pre:** numFollow(sectionId) = @pre.numFollow(sectionId) + 1 |

| | |
|---|---|
| **Nome Metodo** | +unFollow(int sectionId) |
| **Descrizione** | Questo metodo permette di aggiungere un follow ad una sezione. |
| **Pre-condizione** | **Context:** FollowService:: unFollow(int sectionId)<br>**Pre:** SectionExists(sectionId) == true |
| **Post-condizione** | **Context:** FollowService:: unFollow(int sectionId)<br>**Pre:** numFollow(sectionId) = @pre.numFollow(sectionId) - 1 |

## PostService

| | |
|---|---|
| **Nome Classe** | PostService |
| **Descrizione** | Questa classe permette di gestire le funzioni relative ai post. |
| **Metodi** | +getPost(int id): PostPage<br>+loadPosts(PostSearchForm form): List<PostPage><br>+delete(int id): void<br>+newPost(String title, String body, String sectionName): int<br>+newPost(String title, BufferedInputStream content, String sectionName): int |
| **Invariante di classe** | / |

| | |
|---|---|
| **Nome Metodo** | +getPost(int id) |
| **Descrizione** | Questo metodo restituisce un post in formato DTO. |

| | |
|---|---|
| **Pre-condizione** | **Context:** PostService:: getPost(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | / |

| | |
|---|---|
| **Nome Metodo** | +loadPosts(PostSearchForm form) |
| **Descrizione** | Questo metodo permette di aggiungere un follow ad una sezione. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

| | |
|---|---|
| **Nome Metodo** | +delete(int id) |
| **Descrizione** | Questo metodo permette di eliminare un post. |
| **Pre-condizione** | **Context:** PostService:: delete(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | **Context:** PostService:: delete(int id)<br>**Post:** PostExists(id) == false |

| | |
|---|---|
| **Nome Metodo** | +newPost(String title, String body, String sectionName) |
| **Descrizione** | Questo metodo permette di aggiungere un nuovo post. |
| **Pre-condizione** | **Context:** PostService:: newPost(String title, String body, String sectionName)<br>**Pre:** notBlank(title) && numChar(body) <= 65535 && sectionName != null && SectionExists(sectionName) == true |
| **Post-condizione** | **Context:** PostService:: newPost(String title, String body, String sectionName)<br>**Post:** PostExists(newPostId) == true |

| | |
|---|---|
| **Nome Metodo** | +newPost(String title, BufferedInputStream content, String sectionName) |
| **Descrizione** | Questo metodo permette di aggiungere un nuovo post. |
| **Pre-condizione** | **Context:** PostService:: newPost(String title, BufferedInputStream content, String sectionName)<br>**Pre:** notBlank(title) && numChar(title) <= 255 && content != null && isImage(content) && sectionName != null && SectionExists(sectionName) == true |
| **Post-condizione** | **Context:** PostService:: newPost(String title, BufferedInputStream content, String sectionName)<br>**Post:** PostExists(newPostId) == true |

## Section Service

| | |
|---|---|
| **Nome Classe** | SectionService |
| **Descrizione** | Questa classe permette di gestire le funzioni relative alle sezioni. |
| **Metodi** | +delete(int id): void<br>+newSection(String name, String description, BufferedInputStream picture, BufferedInputStream banner): int<br>+showSections(): List<SectionPage> |

| | |
|---|---|
| | +getSectionsMap(): Map<Integer, SectionPage><br>+showSection(int id): SectionPage<br>+getSection(String sectionName): SectionPage<br>+getTopSections(): List<SectionPage><br>+getTrendingSections(): List<SectionPage> |
| **Invariante di classe** | / |

| **Nome Metodo** | +delete(int id) |
|---|---|
| **Descrizione** | Questo metodo elimina una sezione. |
| **Pre-condizione** | **Context:** SectionService:: delete(int id)<br>**Pre:** SectionExists(id) == true |
| **Post-condizione** | **Context:** SectionService:: delete(int id)<br>**Post:** SectionExists(id) == false |

| **Nome Metodo** | +newSection(String name, String description, BufferedInputStream picture, BufferedInputStream banner) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere una nuova sezione. |
| **Pre-condizione** | **Context:** SectionService:: newSection(String name, String description, BufferedInputStream picture, BufferedInputStream banner)<br>**Pre:** notBlank(name) && numChar(name) <= 50 && uniqueSection(name) && numChar(description) <= 255 && isImage(picture) && isImage(banner) |
| **Post-condizione** | **Context:** SectionService:: newSection(String name, String description, BufferedInputStream picture, BufferedInputStream banner)<br>**Post:** SectionExists(newSectionId) == true |

| **Nome Metodo** | +showSections() |
|---|---|
| **Descrizione** | Questo metodo restituisce la lista delle sezioni presenti. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

| **Nome Metodo** | +getSectionsMap() |
|---|---|
| **Descrizione** | Questo metodo restituisce una mappa di tutte le sezioni esistenti. |
| **Pre-condizione** | / |
| **Post-condizione** | / |

| **Nome Metodo** | +showSection(int id) |
|---|---|
| **Descrizione** | Questo metodo restituisce il DTO di una sezione. |
| **Pre-condizione** | **Context:** SectionService:: showSection(int id)<br>**Pre:** SectionExists(id) == true |
| **Post-condizione** | / |

| Nome Metodo | +getSection(String sectionName) |
|---|---|
| Descrizione | Questo metodo restituisce il DTO di una sezione dato il suo nome. |
| Pre-condizione | **Context:** SectionService:: getSection(String sectionName)<br>**Pre:** sectionName != null && SectionExists(sectionName) == true |
| Post-condizione | / |

| Nome Metodo | +getTopSection() |
|---|---|
| Descrizione | Questo metodo restituisce una lista di sezioni con più follow. |
| Pre-condizione | / |
| Post-condizione | / |

| Nome Metodo | +getTrendingSections() |
|---|---|
| Descrizione | Questo metodo restituisce una lista di sezioni con più follow negli ultimi 7 giorni. |
| Pre-condizione | / |
| Post-condizione | / |

User Service

| Nome Classe | UserService |
|---|---|
| Descrizione | Questa classe permette di gestire le funzioni relative agli utenti. |
| Metodi | +toggleAdmin(int id): void<br>+getUser(String name): UserProfile<br>+getUser(int id): UserProfile<br>+getUsernameById(int id): String<br>+showUsers(): List<UserProfile><br>+delete(int id): void<br>+edit(UserEditPage edit, int id): void<br>+newUser(String email, String username, String password): int |
| Invariante di classe | / |

| Nome Metodo | +toggleAdmin(int id) |
|---|---|
| Descrizione | Questo metodo inverte lo stato di admin di un utente dato il suo id. |
| Pre-condizione | **Context:**UserService:: toggleAdmin(int id)<br>**Pre:** UserExists(id) == true |
| Post-condizione | **Context:**UserService:: toggleAdmin(int id)<br>**Post:** isAdmin(id) != @pre.isAdmin(id) |

| Nome Metodo | +getUser(String name) |
|---|---|
| Descrizione | Questo metodo restituisce un DTO di un utente dato il suo nome. |
| Pre-condizione | **Context:**UserService:: getUser(String name)<br>**Pre:** notBlank(name) && UserExists(name) == true |
| Post-condizione | / |

| Nome Metodo | +getUser(int id) |
|---|---|
| Descrizione | Questo metodo restituisce un DTO di un utente dato il suo id. |
| Pre-condizione | **Context:**UserService:: getUser(String name)<br>**Pre:** UserExists(id) == true |
| Post-condizione | / |

| Nome Metodo | +getUsernameById(int id) |
|---|---|
| Descrizione | Questo metodo restituisce lo username di un utente dato il suo id. |
| Pre-condizione | **Context:**UserService:: getUsernameById(int id)<br>**Pre:** UserExists(id) == true |
| Post-condizione | / |

| Nome Metodo | +showUsers() |
|---|---|
| Descrizione | Questo metodo restituisce una lista di DTO relativa agli utenti registrati. |
| Pre-condizione | / |
| Post-condizione | / |

| Nome Metodo | +delete(int id) |
|---|---|
| Descrizione | Questo metodo permette di eliminare un utente dato il suo id. |
| Pre-condizione | **Context:**UserService:: delete(int id)<br>**Pre:** UserExists(id) == true |
| Post-condizione | **Context:**UserService:: delete(int id)<br>**Post:** UserExists(id) == false |

| Nome Metodo | +edit(UserEditPage edit, int id) |
|---|---|
| Descrizione | Questo metodo permette di modificare i dati di un utente. |
| Pre-condizione | **Context:**UserService:: edit(UserEditPage edit, int id)<br>**Pre:** isValid(edit) && UserExists(id) == true |
| Post-condizione | / |

| Nome Metodo | +newUser(String email, String username, String password) |
|---|---|
| Descrizione | Questo metodo permette di modificare i dati di un utente. |
| Pre-condizione | **Context:**UserService:: newUser(String email, String username, String password)<br>**Pre:** email != null && emailFormat(email) && uniqueEmail(email) && username != null && usernameFormat(username) && uniqueUsername(username) && password != null && passwordFormat(password) |
| Post-condizione | **Context:**UserService:: newUser(String email, String username, String password)<br>**Post:** UserExists(newUserId) == true |

## VoteService

| Nome Classe | VoteService |
|---|---|

| **Descrizione** | Questa classe permette di gestire le funzioni relative ai voti. |
|---|---|
| **Metodi** | +upvoteComment(int id): void<br>+downvoteComment(int id): void<br>+upvotePost(int id): void<br>+downvotePost(int id): void<br>+unvoteComment(int id): void<br>+unvotePost(int id): void<br>+votePost(int id, short vote): void<br>+voteComment(int id, short vote): void |
| **Invariante di classe** | / |

| **Nome Metodo** | + upvoteComment(int id) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere un voto positivo ad un commento. |
| **Pre-condizione** | **Context:**VoteService:: upvoteComment(int id)<br>**Pre:** CommentExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: upvoteComment(int id)<br>**Post:** numUpvote(id) = @pre.numUpvote(id) +1 |

| **Nome Metodo** | + downvoteComment(int id) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere un voto negativo ad un commento. |
| **Pre-condizione** | **Context:**VoteService:: downvoteComment(int id)<br>**Pre:** CommentExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: downvoteComment(int id)<br>**Post:** numDownvote(id) = @pre.numDownvote(id) -1 |

| **Nome Metodo** | + upvotePost(int id) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere un voto positivo ad un post. |
| **Pre-condizione** | **Context:**VoteService:: upvotePost(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: upvotePost(int id)<br>**Post:** numUpvote(id) = @pre.numUpvote(id) +1 |

| **Nome Metodo** | + downvotePost(int id) |
|---|---|
| **Descrizione** | Questo metodo permette di aggiungere un voto negativo ad un post. |
| **Pre-condizione** | **Context:**VoteService:: downvotePost(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: downvotePost(int id)<br>**Post:** numDownvote(id) = @pre.numDownvote(id) -1 |

| **Nome Metodo** | + unvoteComment(int id) |
|---|---|
| **Descrizione** | Questo metodo permette di rimuovere un voto ad un commento. |

| | |
|---|---|
| **Pre-condizione** | **Context:**VoteService:: unvoteComment(int id)<br>**Pre:** CommentExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: unvoteComment(int id)<br>**Post:** numVote(id) = @pre.numVote(id) -1 |

| | |
|---|---|
| **Nome Metodo** | + unvotePost(int id) |
| **Descrizione** | Questo metodo permette di rimuovere un voto ad un post. |
| **Pre-condizione** | **Context:**VoteService:: unvotePost(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: unvotePost(int id)<br>**Post:** numVote(id) = @pre.numVote(id) -1 |

| | |
|---|---|
| **Nome Metodo** | - votePost(int id) |
| **Descrizione** | Questo metodo permette di aggiungere un voto ad un post. |
| **Pre-condizione** | **Context:**VoteService:: votePost(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: votePost(int id)<br>**Post:** numVote(id) = @pre.numVote(id) +1 |

| | |
|---|---|
| **Nome Metodo** | - voteComment(int id) |
| **Descrizione** | Questo metodo permette di aggiungere un voto ad un comment. |
| **Pre-condizione** | **Context:**VoteService:: voteComment(int id)<br>**Pre:** PostExists(id) == true |
| **Post-condizione** | **Context:**VoteService:: voteComment(int id)<br>**Post:** numVote(id) = @pre.numVote(id) +1 |

# Design Patterns

In questa sezione sono descritti i design pattern utilizzati nello sviluppo dell'applicazione, fornendo per ognuno di questi una breve introduzione e le motivazioni dietro la scelta di tale utilizzo.

## Chain-of-responsibility pattern

Il Chain-of-responsibility è un pattern comportamentale che consente l'inoltro di una richiesta a molteplici gestori (*handler*). Alla ricezione di una richiesta, il gestore processa tale richiesta e decide infine se passarla al prossimo gestore nella catena. Tale meccanismo consente di disaccoppiare chi invia la richiesta dal gestore che la finalizza.

Alcune applicabilità del pattern sono le seguenti: (Gamma, Helm, & Johnson, 1998)

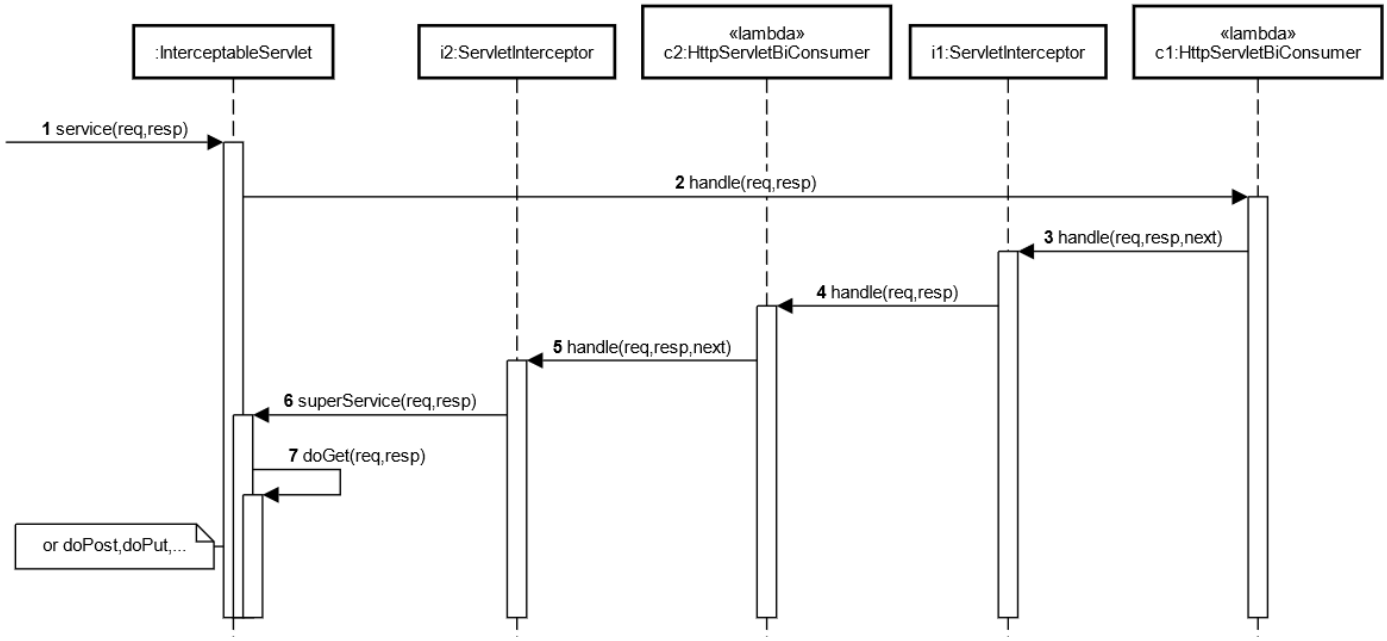
- Utilizzare il Chain-of-responsability pattern quando più di un oggetto potrebbe gestire la richiesta
- Utilizzare il Chain-of-responsability pattern quando si vuole inviare una richiesta a più oggetti senza specificare esplicitamente l'effettivo destinatario
- Utilizzare il Chain-of-responsability pattern quando è possibile specificare dinamicamente l'insieme di oggetti che può gestire la richiesta

Nell'ambito del progetto Shareboard, tale pattern è stato adoperato per la realizzazione degli interceptor per le Java Servlet. Tale meccanismo consente la definizione di appositi handler (`ServletInterceptor`) applicabili in modo dichiarativo ai singoli metodi delle Servlet per definire comportamenti aggiuntivi senza modificare il corpo dei metodi delle singole Servlet.

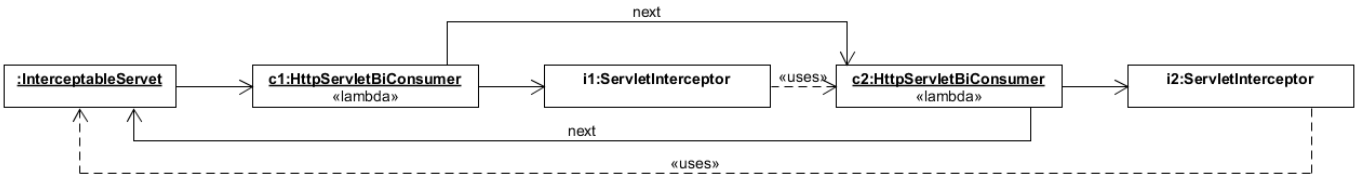
Tale meccanismo, molto simile al già presente meccanismo dei filtri Servlet (Oracle, 2022), presenta le seguenti differenze:

- L'applicazione degli interceptor è basata sulle annotazioni Java
  - È possibile applicare interceptor ai singoli metodi delle Servlet
  - È possibile specificare parametri alle annotazioni: questo consente l'applicazione di interceptor secondo eventuali preferenze particolari
- Ogni interceptor chiama direttamente l'interceptor successivo
  - I filtri Servlet utilizzano un oggetto centrale (`FilterChain`) che tiene traccia dell'andamento della catena e si occupa di invocare il filtro successivo

Il seguente sequence diagram mostra le interazioni tra una Servlet abilitata al meccanismo degli interceptor (ossia una Servlet che estende InterceptableServlet) e gli interceptor.



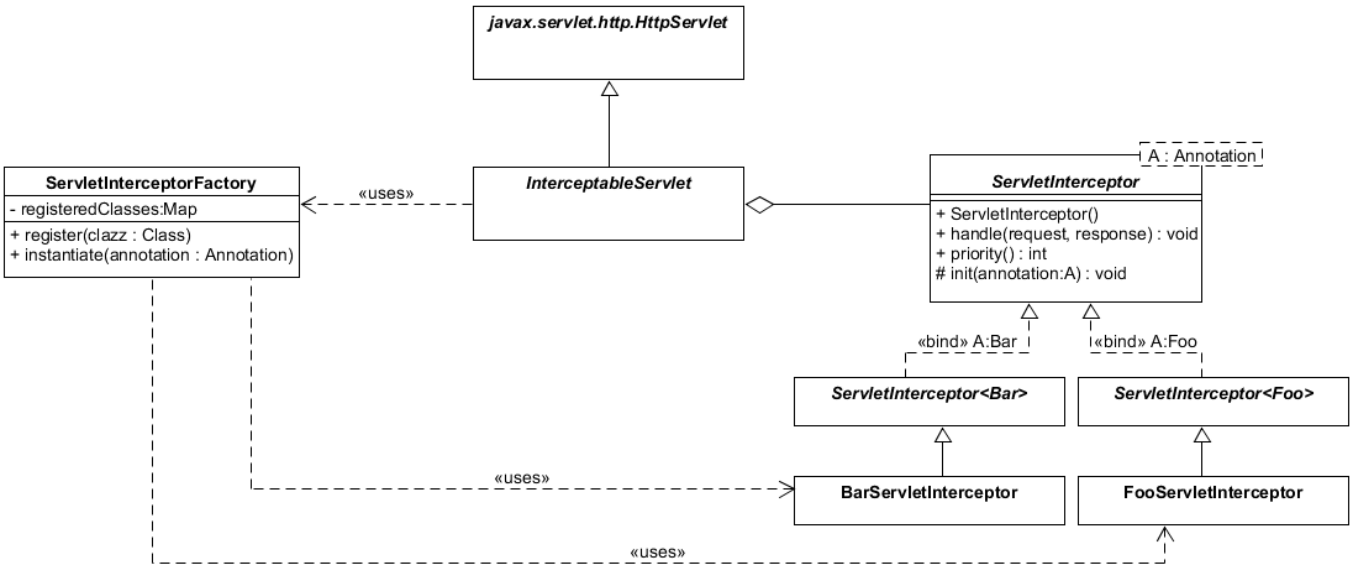

Il seguente object diagram mostra le relazioni tra gli oggetti partecipanti sopra.



# Factory method

Il Factory Method pattern consente la definizione di un'interfaccia per la creazione di una famiglia di oggetti. Grazie al Factory Method pattern è possibile istanziare oggetti senza conoscere la loro classe esatta.

Nel caso del progetto Shareboard, tale pattern è utilizzato per la realizzazione del meccanismo dei Servlet Interceptor: ad ogni Servlet Interceptor è associata una (e una sola) annotazione da poter applicare sulle firme dei metodi e sulle firme delle classi. La classe `ServletInterceptorFactory` costituisce una realizzazione del Factory Method Pattern e consente l'istanziazione e l'inizializzazione di specifiche sottoclassi di `ServletInterceptor` in base all'annotazione fornitagli come parametro.



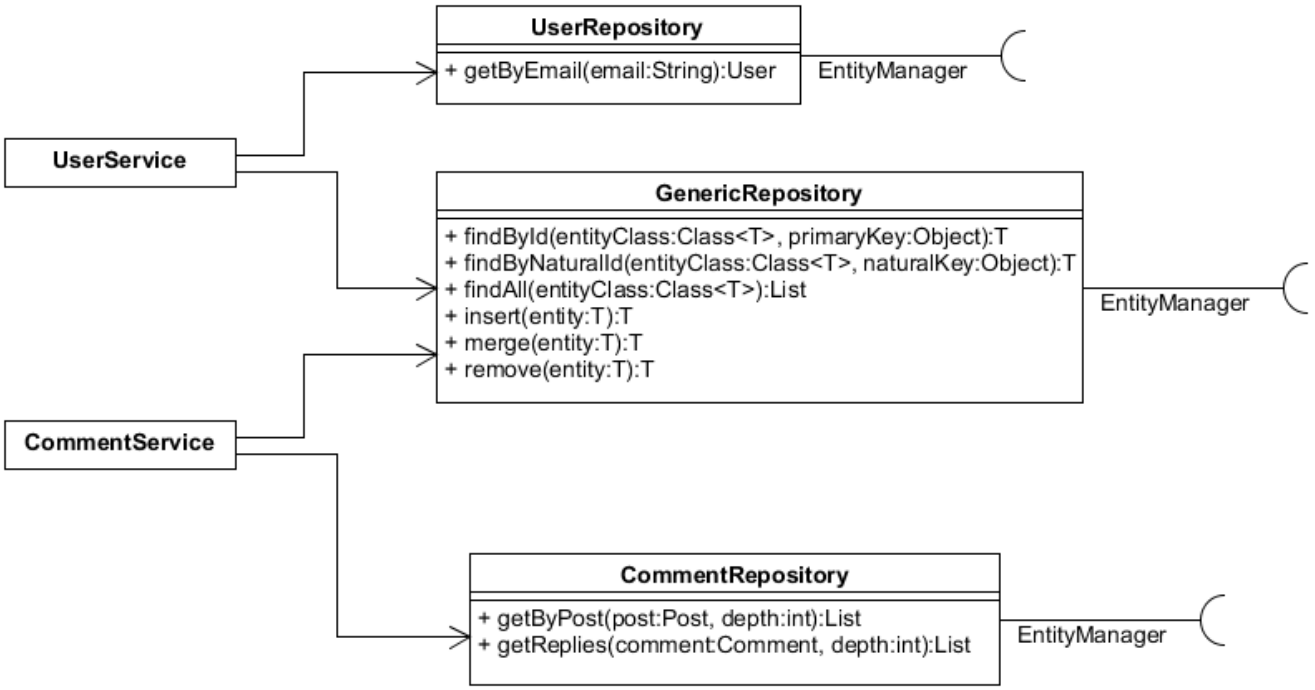
## Repository pattern

Un *Repository* (Fowler, 2015) funge da mediatore tra il layer della logica di business e il data layer, fornendo un'interfaccia tipica delle collezioni per l'accesso ad oggetti del dominio applicativo.

I client possono costruire query in modo dichiarativo utilizzando un linguaggio di dominio specifico da sottomettere al Repository per ottenere entità di loro interesse. Gli oggetti Repository incapsulano l'insieme di oggetti persistenti in un data store, abilitando le operazioni di ritrovo, modifica, cancellazione e inserimento ai client.

I repository portano a una maggiore separazione delle responsabilità tecniche. È possibile opzionalmente fornire diverse strategie di accesso alle entità persistenti (Vendor JPA differenti, diverso DBMS), mantenendo la stessa interfaccia pubblica.

Il seguente class diagram mostra la dipendenza di alcune classi del business logic layer rispetto agli oggetti repository. Da notare la relazione di dipendenza unidirezionale.



# Glossario

| Sigla/Termine | Definizione |
|---|---|
| **Account** | Rappresentazione dell'utente in formato digitale. |
| **Admin** | Amministratore della piattaforma |
| **Ban** | Divieto di interazione sulla piattaforma per un particolare utente |
| **Commento** | Messaggio di risposta ad un post o ad un altro commento |
| **Feed** | Insieme dei post appartenenti alle sezioni seguite da un particolare utente |
| **Follow** | Una istanza di relazione "segue" tra un utente e una sezione |
| **Mock-up** | Una rappresentazione visuale della User Interface, utile al committente per capire come il prodotto sarà fruibile all'utente. |
| **Piattaforma** | Base software o hardware su cui sono sviluppate o eseguite applicazioni. |
| **Post** | Contenuto condiviso da un utente in una determinata sezione, può contenere il titolo, il testo e anche un'immagine. |
| **Sezione** | Una categoria di argomenti creata da un amministratore, dove gli utenti possono condividere post coerenti in termini contenutistici alla categoria. |