Università degli Studi di Salerno
Corso di Ingegneria del Software

SwaGGed
Object Design Document
Versione 1.0

Data: 2/12/2024

Coordinatore del progetto

Nome	Matricola

Partecipanti:

Nome	Matricola
Choaib Goumri	0512118390
Mattia Gallucci	0512116893

Scritto da:	Choaib Goumri
	Mattia Gallucci

Revision History

Data	Versione	Descrizione	Autore
2/12/2024	1.0	Prima stesura del documento	Choaib Goumri, Gallucci Mattia

Indice

Sommario

1. INTRODUZIONE

1.1 Object design trade-offs

Comprensibilità vs Tempo

A causa dei tempi di sviluppo limitati, non sarà prioritario inserire commenti dettagliati in ogni metodo. Questa scelta mira a velocizzare l'implementazione del sistema, ma comporterà una maggiore complessità nella fase di testing e nelle eventuali modifiche future.

Riusabilità vs Costi

Il sistema sarà sviluppato senza l'utilizzo di librerie o componenti a pagamento, poiché il progetto non dispone di un budget economico.

Sicurezza vs Efficienza

Considerando le tempistiche ristrette, il sistema implementerà misure di sicurezza basate su **username e password crittografate**. Sarà garantito un controllo degli accessi mediante ruoli definiti, bilanciando sicurezza ed efficienza.

Incapsulamento vs Efficienza

L'architettura del sistema assicura la protezione dei dettagli implementativi delle classi. Gli attributi saranno manipolabili esclusivamente attraverso i metodi definiti, rispettando il principio di incapsulamento.

Trasparenza vs Efficienza

Per la gestione della persistenza dei dati, si utilizzerà un database relazionale. Grazie al supporto di un **DBMS**, sarà possibile operare in modo consistente e transazionale, accettando una minima perdita di efficienza. Le connessioni al database saranno gestite tramite un **Driver Manager**, che semplifica la sincronizzazione e l'interazione con il DBMS.

1.2 Interface documentiation guidelines

Per garantire un'elevata leggibilità e una buona manutenibilità del codice, gli sviluppatori devono seguire specifiche linee guida durante la scrittura del codice.

Stile del codice

Gli sviluppatori sono invitati ad aderire il più possibile alle convenzioni di stile del codice definite da Google Java. In particolare:

- **Indentazione:** utilizzare i **tab** invece degli spazi.
- **Formattazione:** mantenere uno stile conciso e facilmente comprensibile.

Convenzioni per i nomi

I nomi utilizzati nel codice devono essere:

- **Descrittivi:** chiari e indicativi del loro scopo.
- **Pronunciabili:** facilmente leggibili e comprensibili.
- **Di uso comune:** seguire termini standard e familiari.
- **Non abbreviati:** evitare abbreviazioni, tranne per variabili temporanee.
- **Validi:** contenere solo caratteri consentiti (A-Z, a-z, 0-9).

Classi

- I nomi delle classi devono seguire lo stile **Upper Camel Case (o Pascal Case)**, in cui la prima lettera di ogni parola è maiuscola.
- Non utilizzare acronimi o abbreviazioni nei nomi delle classi, a meno che l'abbreviazione sia più diffusa della forma estesa (ad esempio, *URL* o *HTML*).

Esempio: class User {}

Metodi e Funzioni

- I nomi dei metodi e delle funzioni devono seguire lo stile **Lower Camel Case (o Dromedary Case)**, in cui la prima lettera della prima parola è minuscola e la prima lettera di ogni parola successiva è maiuscola.
- Devono essere composti da un verbo o includere un verbo come prima parola.

Esempio: getName();

1.3 Definitions, acronyms, and abbreviations

Acronimo	Abbreviazione	Definizione
URL	Uniform Resource Locator	È' un indirizzo che specifica la posizione di una risorsa su Internet
HTML	HyperText Markup Language	È il linguaggio standard utilizzato per creare e strutturare le pagine web
DP	Design Pattern	Sono soluzioni riutilizzabili e consolidate per problemi comuni
DAO	Data Access Object	È un pattern di progettazione che fornisce un'interfaccia astratta per interagire con un database o altre sorgenti dati persistenti

1.4 References

- SDD

  Ci si riferisce al SDD quando si spiega l'organizzazione dei package, dato che quest'ultima è stata creata proprio a partire dalla suddivisione in subsystem.

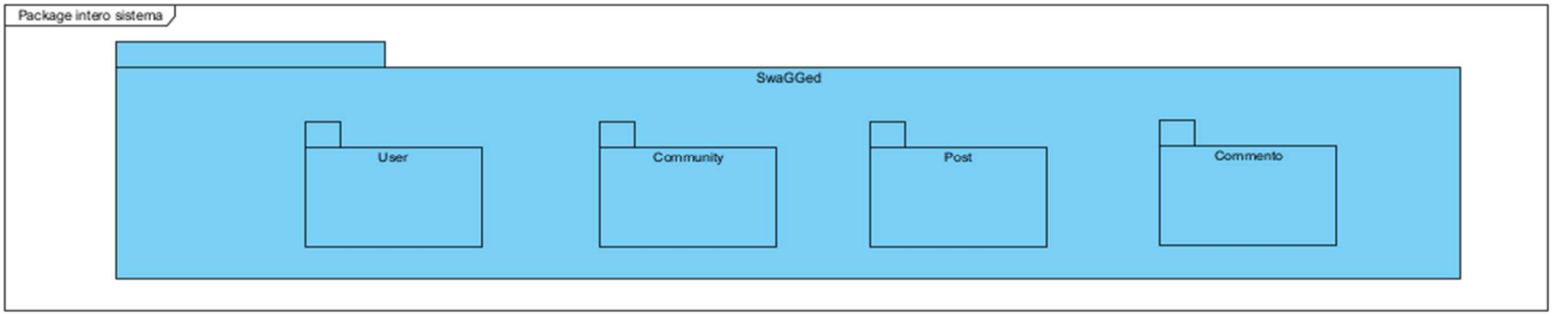
2. PACKAGES

In questa sezione viene illustrata la struttura del package principale di Swagged.

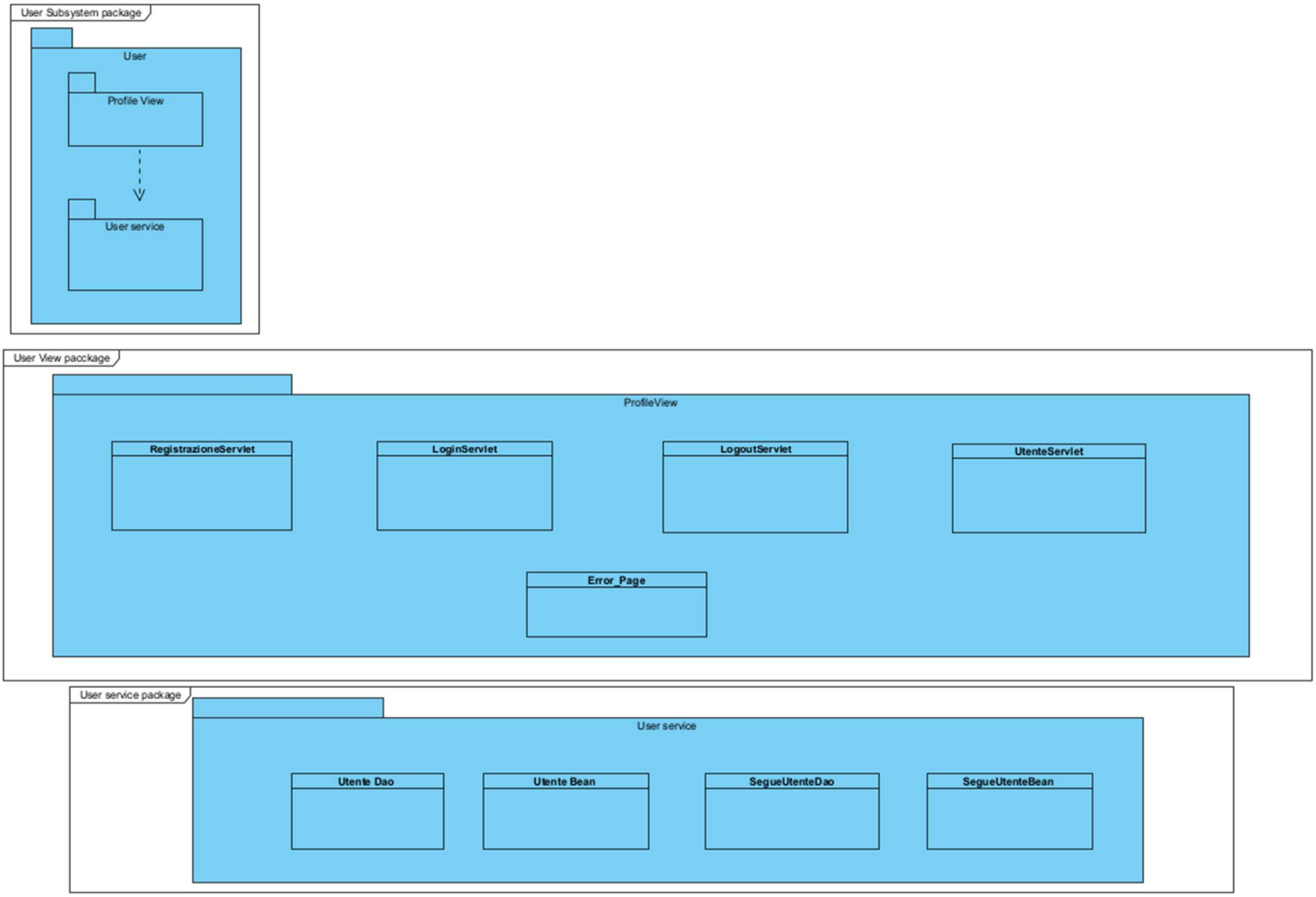
La progettazione generale si basa sulla suddivisione in partizioni e sottosistemi individuati durante la fase di System Design, come documentato nel **System Design Document (SDD)**.

La struttura adotta un'architettura a tre livelli, in cui ogni classe è assegnata a uno dei seguenti ruoli principali:

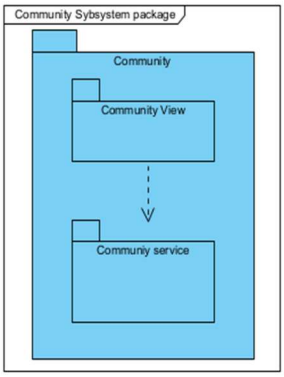
- **Presentazione**: gestione della visualizzazione e rappresentazione dei dati del modello.
- **Controllo**: supervisione delle attività di processamento interno del sistema.
- **Manipolazione dati**: gestione e trasformazione dei dati relativi all'Application Domain.

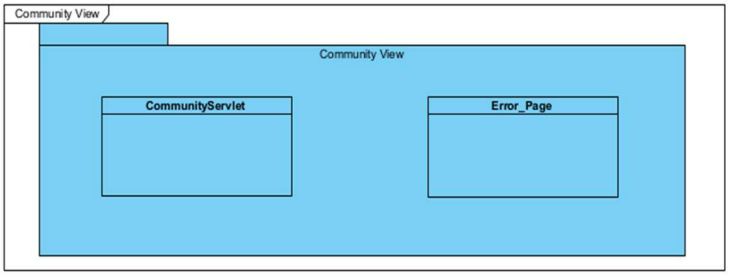


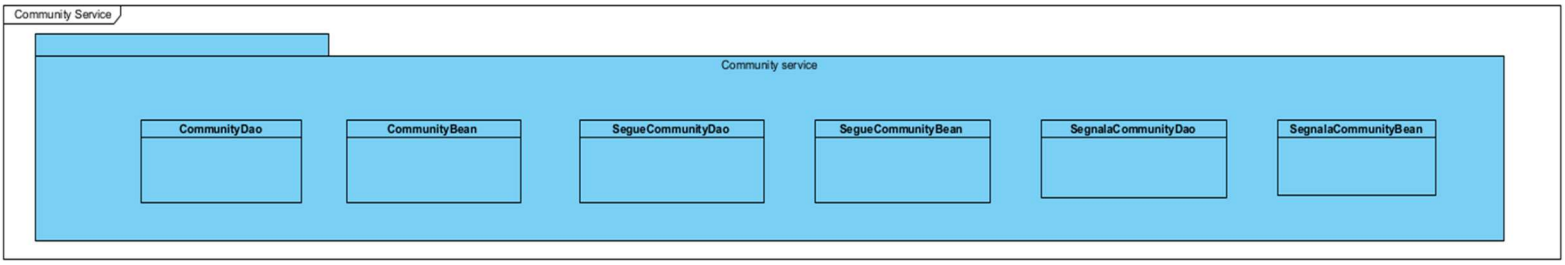
2.1 Partizione User



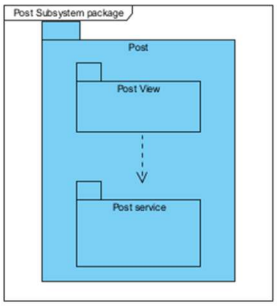
2.2 Partizione Community

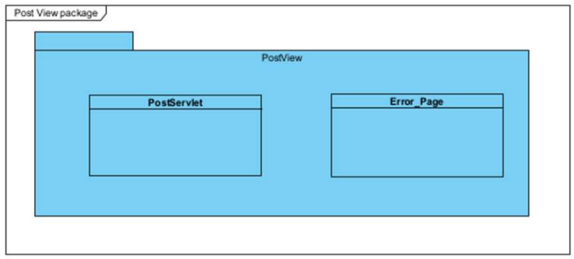


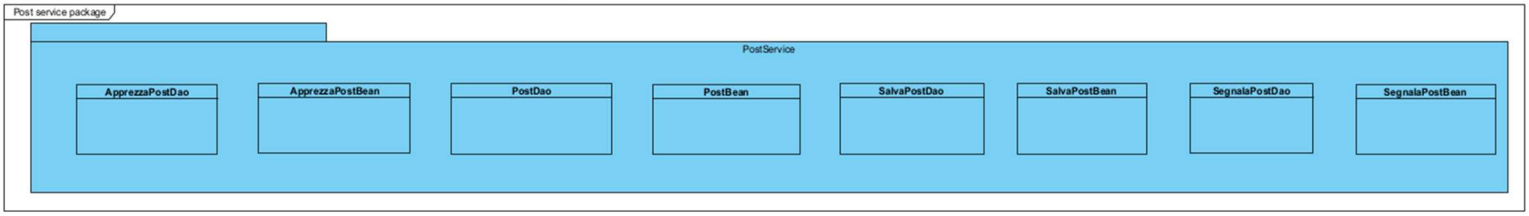




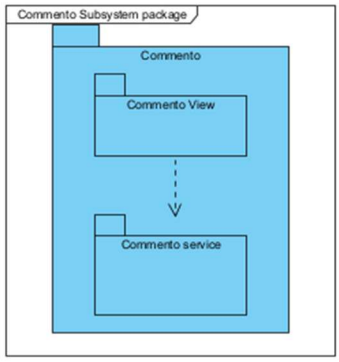
2.3 Partizione Post

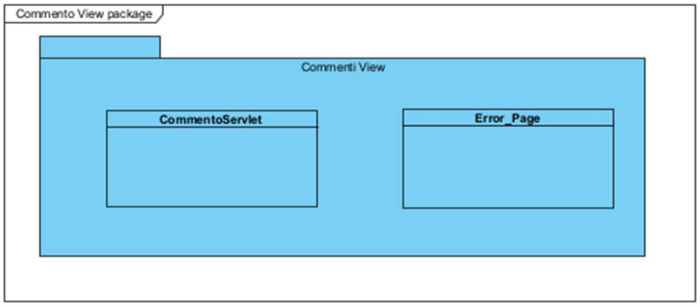


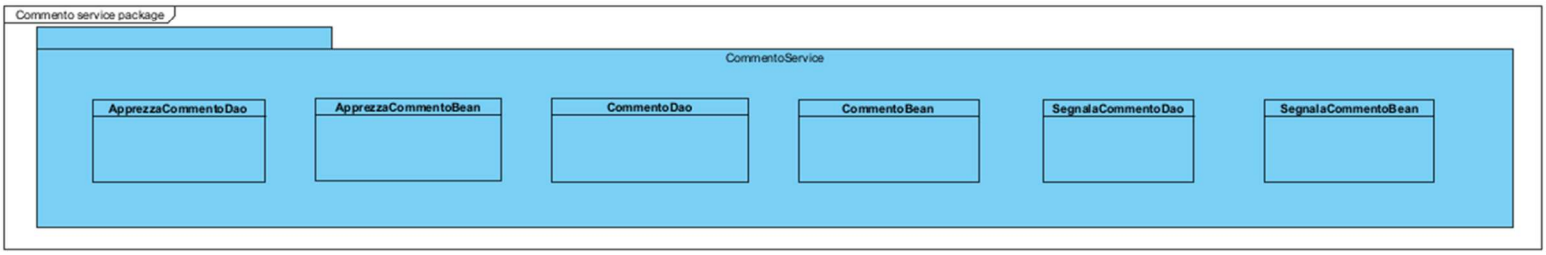




2.4 Partizione Commento







3. CLASS INTERFACES

3.1 User Service

Nome Classe	UtenteDAO
Descrizione	Questa classe consente di amministrare le operazioni relative agli account degli utenti gestori
Metodi	+doSave(bean: UtenteBean): void +doDelete(email: String): Boolean +doRetrieveByEmail(email: String): UtenteBean +doRetrieveByUsername(username: String): UtenteBean +doRetrieveAll(order: String): List<UtenteBean> +doUpdate(bean: UtenteBean, email: String): Boolean +doRetrieveByUsernameSubstring(substring: String): List<UtenteBean> +checkEmail(email: String): Boolean +checkUsername(username: String): Boolean
Invariante di classe	

Nome Metodo	+doSave(bean: UtenteBean): void
Descrizione	Questo metodo consente di inserire un nuovo utente nel database
Pre-condizione	**Context**: **Pre**: bean != null AND bean.email != null AND bean.email != "" AND bean.username != null AND bean.username != "" AND bean.password != nul AND bean.password != ""
Post-condizione	**Context**: **Post**: Database.utenti -> include(bean)

Nome Metodo	+doDelete(email: String): Boolean
Descrizione	Questo metodo consente di rimuovere un

	utente dal database
Pre-condizione	**Context**: **Pre**: email != null AND email != "" AND Database.utenti->exists(u | Utente.email = email)
Post-condizione	**Context**: **Post**: Database.utenti -> !include(utente)

Nome Metodo	+doRetrieveByEmail(email: String): UtenteBean
Descrizione	Questo metodo restituisce l'utente a cui è asscoiata l'email
Pre-condizione	**Context**: **Pre**: email != null AND email != "" AND Database.utenti->exists(u | Utente.email = email)
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByUsername(username: String): UtenteBean
Descrizione	Questo metodo restituisce l'utente a cui è asscoiato l'username
Pre-condizione	**Context**: **Pre**: username != null AND username != "" AND Database.utenti->exists(u | Utente.username = username)
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<UtenteBean>

Descrizione	Questo metodo restituisce tutti gli utenti presenti nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doUpdate(bean: UtenteBean, email: String): Boolean
Descrizione	Questo metodo aggiorna le informazioni di un utente nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByUsernameSubstring(substring: String): List<UtenteBean>
Descrizione	Questo metodo restituisce gli utenti che hanno nell'username la sottostringa
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+checkEmail(email: String): Boolean
Descrizione	Questo metodo verifica l'esistenza di utente nel database a cui è associata l'email
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+checkUsername(username: String): Boolean
Descrizione	Questo metodo verifica l'esistenza di un

	utente nel database a cui è associato l'username
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

3.2 Post Service

Nome Classe	PostDAO
Descrizione	Questa classe consente di amministrare le operazioni relative ai post
Metodi	+doSave(bean: PostBean): void +doRetrieveById(id: Integer): PostBean +doRetrieveAll(orderBy: String, descending: Boolean): List(PostBean) +doRetrieveByEmail(email: String): List(PostBean) +doRetrieveByTitleSubstring(substring: String): List(PostBean) +doDelete(id: Integer): void +doUpdate(bean: PostBean): Boolean +doRetrieveByCommunityId(communityId: Integer, orderBy: String, descending: Boolean): List(PostBean)
Invariante di classe	

Nome Metodo	+doSave(bean: PostBean): void
Descrizione	Questo metodo consente di inserire un nuovo post nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveById(id: Integer): PostBean
Descrizione	Questo metodo restituisce il post a cui è asscoiato l'id
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(orderBy: String, descending: Boolean):

	List(PostBean)
Descrizione	Questo metodo restituisce tutti i post presenti nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByEmail(email: String): List(PostBean
Descrizione	Questo metodo restituisce tutti i post a cui è associata l'email
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByTitleSubstring(substring: String): List(PostBean)
Descrizione	Questo metodo restituisce i post che hanno nel titolo la sottostringa
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(id: Integer): void
Descrizione	Questo metodo consente di rimuovere un utente dal database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doUpdate(bean: PostBean): Boolean
Descrizione	Questo metodo aggiorna le

	informazioni di un post nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByCommunityId(communityId: Integer, orderBy: String, descending: Boolean): List(PostBean)
Descrizione	Questo metodo restituisce tutti i post a cui è associato l'id della community
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

3.3 Commento Service

Nome Classe	CommentoDAO
Descrizione	Questa classe consente di amministrare le operazioni relative commenti
Metodi	+doSave(bean: CommentoBean): void +doDelete(id: Integer): Boolean +doRetrieveById(id: Integer): CommentoBean +doRetrieveAll(): List(CommentoBean) +doRetrieveByPostId(postId: Integer): List(CommentoBean) +doRetrieveByUtenteEmail(email: String): List(CommentoBean) +doUpdate(bean: CommentoBean): Boolean
Invariante di classe	

Nome Metodo	+doSave(bean: CommentoBean): void
Descrizione	Questo metodo consente di inserire un nuovo commento nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(id: Integer): Boolean
Descrizione	Questo metodo consente di rimuovere un commento dal database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveById(id: Integer): CommentoBean

Descrizione	Questo metodo restituisce il commento a cui è asscoiato l'id
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(): List(CommentoBean)
Descrizione	Questo metodo restituisce tutti i commenti presenti nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByPostId(postId: Integer): List(CommentoBean)
Descrizione	Questo metodo restituisce tutti i commenti a cui è associato l'id del post
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByUtenteEmail(email: String): List(CommentoBean)
Descrizione	Questo metodo restituisce tutti i commenti a cui è associata l'email
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doUpdate(bean: CommentoBean): Boolean

Descrizione	Questo metodo aggiorna le informazioni di un commento nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

3.4 Community Service

Nome Classe	CommunityDAO
Descrizione	Questa classe consente di amministrare le operazioni relative alle community
Metodi	+doSave(bean: CommunityBean): void +doDelete(nome: String): Boolean +doRetrieveByNome(nome: String): CommunityBean +doRetrieveAll(): List(CommunityBean) +doUpdate(bean: CommunityBean): Boolean +doRetrieveByEmail(email: String): List(CommunityBean) +doRetrieveByNameSubstring(substring: String): List(CommunityBean) +checkNome(nome: String): Boolean
Invariante di classe	

Nome Metodo	+doSave(bean: CommunityBean): void
Descrizione	Questo metodo consente di inserire una nuova community nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(nome: String): Boolean
Descrizione	Questo metodo consente di rimuovere una community dal database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByNome(nome: String): CommunityBean

Descrizione	Questo metodo restituisce la community a cui è asscoiato il nome
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(): List(CommunityBean)
Descrizione	Questo metodo restituisce tutte le community presenti nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doUpdate(bean: CommunityBean): Boolean
Descrizione	Questo metodo aggiorna le informazioni di una community nel database
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByEmail(key: String): List(CommunityBean)
Descrizione	Questo metodo restituisce tutte le community a cui è associata l'email
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByNameSubstring(substring: String): List(CommunityBean)

Descrizione	Questo metodo restituisce le community che hanno nel nome la sottostringa
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

Nome Metodo	+checkNome(nome: String): Boolean
Descrizione	Questo metodo verifica l'esistenza di una community nel database a cui è associato il nome
Pre-condizione	**Context**: **Pre**:
Post-condizione	**Context**: **Post**:

3.5 Interazione Service

Nome Classe	ApprezzaPostDAO
Descrizione	Questa classe consente di amministrare le operazioni relative agli apprezzamenti dei post
Metodi	+doSave(bean: ApprezzaPostBean): void +doDelete(utenteEmail: String, postId: int): boolean +doRetrieveByKey(utenteEmail: String, postId: int): ApprezzaPostBean +doRetrieveByEmail(utenteEmail: String): List<ApprezzaPostBean> +doRetrieveAll(order: String): List<ApprezzaPostBean> +doUpdate(bean: ApprezzaPostBean, utenteEmail: String, postId: int): boolean
Invariante di classe	

Nome Metodo	+doSave(bean: ApprezzaPostBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente apprezza post
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, postId: int): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, postId: int): ApprezzaPostBean

Descrizione	Questo metodo restituisce la relazione identificata dall'email e dall'id
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByEmail(utenteEmail: String): List<ApprezzaPostBean>
Descrizione	Questo metodo restituisce tutti i post apprezzati dall'utente a cui è associata l'email
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<ApprezzaPostBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doUpdate(bean: ApprezzaPostBean, utenteEmail: String, postId: int): boolean
Descrizione	Questo metodo aggiorna le informazioni della relazione nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	SalvaPostDAO
Descrizione	Questa classe consente di

	amministrare le operazioni relative ai post salvati
Metodi	+doSave(bean: SalvaPostBean): void +doDelete(utenteEmail: String, postId: int): boolean +doRetrieveByKey(utenteEmail: String, postId: int): SalvaPostBean +doRetrieveByEmail(utenteEmail: String): List<SalvaPostBean> +doRetrieveAll(order: String): List<SalvaPostBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SalvaPostBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente salva post
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, postId: int): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, postId: int): SalvaPostBean

Descrizione	Questo metodo restituisce la relazione identificata dall'email e dall'id
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByEmail(utenteEmail: String): List<SalvaPostBean>
Descrizione	Questo metodo restituisce tutti i post salvati dall'utente a cui è associata l'email
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<SalvaPostBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	SegnalaPostDAO
Descrizione	Questa classe consente di

	amministrare le operazioni relative ai post segnalati
Metodi	+doSave(bean: SegnalaPostBean): void +doDelete(utenteEmail: String, postId: int): boolean +doRetrieveByKey(utenteEmail: String, postId: int): SegnalaPostBean +doRetrieveAll(order: String): List<SegnalaPostBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SegnalaPostBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente segnala post
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, postId: int): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, postId: int): SegnalaPostBean
Descrizione	Questo metodo restituisce la relazione identificata dall'email e dall'id
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String):

	List<SegnalaPostBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	ApprezzaCommentoDAO
Descrizione	Questa classe consente di amministrare le operazioni relative agli apprezzamenti dei commenti
Metodi	+doSave(bean: ApprezzaCommentoBean): void +doDelete(utenteEmail: String, commentoId: int): boolean +doRetrieveByKey(utenteEmail: String, commentoId: int): ApprezzaCommentoBean +doRetrieveByEmail(utenteEmail: String): List<ApprezzaCommentoBean> +doRetrieveAll(order: String): List<ApprezzaCommentoBean> +doUpdate(bean: ApprezzaCommentoBean, utenteEmail: String, commentoId: int): boolean
Invariante di classe	

Nome Metodo	+doSave(bean: ApprezzaCommentoBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente apprezza commento
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, commentoId:

	int): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, commentoId: int): ApprezzaCommentoBean
Descrizione	Questo metodo restituisce la relazione identificata dall'email e dall'id
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByEmail(utenteEmail: String): List<ApprezzaCommentoBean>
Descrizione	Questo metodo restituisce tutti i commenti apprezzati dall'utente a cui è associata l'email
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<ApprezzaCommentoBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doUpdate(bean: ApprezzaCommentoBean, utenteEmail: String, commentoId: int):

	boolean
Descrizione	Questo metodo aggiorna le informazioni della relazione nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	SegnalaCommentoDAO
Descrizione	Questa classe consente di amministrare le operazioni relative ai commenti segnalati
Metodi	+doSave(bean: SegnalaCommentoBean): void +doDelete(utenteEmail: String, commentoId: int): boolean +doRetrieveByKey(utenteEmail: String, commentoId: int): SegnalaCommentoBean +doRetrieveAll(order: String): List<SegnalaCommentoBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SegnalaCommentoBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente segnala commento
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, commentoId: int): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, commentoId: int): SegnalaCommentoBean
Descrizione	Questo metodo restituisce la relazione identificata dall'email e dall'id
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<SegnalaCommentoBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	SegueCommunityDAO
Descrizione	Questa classe consente di amministrare le operazioni relative alle community seguite
Metodi	+doSave(bean: SegueCommunityBean): void +doDelete(utenteEmail: String, communityNome: String): boolean +doRetrieveByKey(utenteEmail: String, communityNome: String): SegueCommunityBean +doRetrieveByEmail(utenteEmail: String): List<SegueCommunityBean> +doRetrieveAll(order: String): List<SegueCommunityBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SegueCommunityBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente segue community
Pre-condizione	**Context**:

	Pre
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, communityNome: String): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, communityNome: String): SegueCommunityBean
Descrizione	Questo metodo restituisce la relazione identificata dall'email e dal nome
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByEmail(utenteEmail: String): List<SegueCommunityBean>
Descrizione	Questo metodo restituisce tutte le community seguite dall'utente a cui è associata l'email
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<SegueCommunityBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**:

	Pre
Post-condizione	**Context**: **Post**:

Nome Classe	SegnalaCommunityDAO
Descrizione	Questa classe consente di amministrare le operazioni relative alle community segnalate
Metodi	+doSave(bean: SegnalaCommunityBean): void +doDelete(utenteEmail: String, communityNome: String): boolean +doRetrieveByKey(utenteEmail: String, communityNome: String): SegnalaCommunityBean +doRetrieveAll(order: String): List<SegnalaCommunityBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SegnalaCommunityBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente segnala community
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(utenteEmail: String, communityNome: String): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(utenteEmail: String, communityNome: String): SegnalaCommunityBean

Descrizione	Questo metodo restituisce la relazione identificata dall'email e dal nome
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<SegnalaCommunityBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	SegueUtenteDAO
Descrizione	Questa classe consente di amministrare le operazioni relative agli utenti seguiti
Metodi	+doSave(bean: SegueUtenteBean): void +doDelete(seguaceEmail: String, seguitoEmail: String): boolean +doRetrieveByKey(seguaceEmail: String, seguitoEmail: String): SegueUtenteBean +doRetrieveAll(order: String): List<SegueUtenteBean> +doRetrieveBySeguace(seguaceEmail: String): List<SegueUtenteBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SegueUtenteBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente segue utente
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**:

	Post:

Nome Metodo	+doDelete(seguaceEmail: String, seguitoEmail: String): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(seguaceEmail: String, seguitoEmail: String): SegueUtenteBean
Descrizione	Questo metodo restituisce la relazione identificata dall'email dell'utente seguace e dall'email dell'utente seguito
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveAll(order: String): List<SegueUtenteBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveBySeguace(seguaceEmail: String): List<SegueUtenteBean>
Descrizione	Questo metodo restituisce tutti gli utenti seguiti dall'utente a cui è associata l'email
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Classe	SegnalaUtenteDAO
Descrizione	Questa classe consente di amministrare le operazioni relative agli utenti segnalati
Metodi	+doSave(bean: SegnalaUtenteBean): void +doDelete(segnalatoreEmail: String, segnalatoEmail: String): boolean +doRetrieveByKey(segnalatoreEmail: String, segnalatoEmail: String): SegnalaUtenteBean +doRetrieveAll(order: String): List<SegnalaUtenteBean>
Invariante di classe	

Nome Metodo	+doSave(bean: SegnalaUtenteBean): void
Descrizione	Questo metodo consente di inserire una nuova relazione utente segnala utente
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doDelete(segnalatoreEmail: String, segnalatoEmail: String): boolean
Descrizione	Questo metodo consente di rimuovere una relazione dal database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

Nome Metodo	+doRetrieveByKey(segnalatoreEmail: String, segnalatoEmail: String): SegnalaUtenteBean
Descrizione	Questo metodo restituisce la relazione identificata dall'email dell'utente segnalatore e dall'email dell'utente segnalato
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**:

	Post:

Nome Metodo	+doRetrieveAll(order: String): List<SegnalaUtenteBean>
Descrizione	Questo metodo restituisce tutte le relazioni presenti nel database
Pre-condizione	**Context**: **Pre**
Post-condizione	**Context**: **Post**:

4. DESIGN PATTERN

I seguenti **Design Patterns** sono stati adottati per l'implementazione:

- **DAO (Data Access Object)**
- **Façade**

DAO

Il pattern DAO è stato utilizzato per semplificare le operazioni di accesso e manipolazione dei dati nel database del sistema. Per ogni tipo di entità definita a livello applicativo, è stato sviluppato un DAO specifico, garantendo una gestione dei dati chiara e strutturata.

Façade

Il pattern Façade è stato impiegato per fornire un'interfaccia semplificata per interagire con le operazioni associate a un **Post**. In particolare, il Façade gestisce le operazioni di apprezzamento, commento, segnalazione e salvataggio di un post, evitando la necessità di invocazioni separate per ciascuna di esse. Questo approccio riduce la complessità e il rischio di utilizzi errati, migliorando la manutenibilità del sistema e l'accoppiamento tra le componenti.