SDD
System Design
GameTalk

Riferimento	NC8_SDD_ver.2.0
Versione	2.0
Data	02/12/2024
Destinatario	Prof.re Carmine Gravino Dott. Giammaria Giordano
Presentato da	NC8
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
29/11/2024	0.1	Prima stesura	Tutto il team
23/12/2024	0.2	Design Goals, Trade-offs, Components Diagram	Tutto il team
02/01/2025	0.3	Deployment Diagram, Modello Entità-Relazione	Alessio Sica Carmelo Cappiello
07/01/2025	0.4	Boundary Conditions Use Cases	Alessio Sica Carmelo Cappiello Antonino Zappile
06/03/2025	0.5	Matrice degli accessi	Alessio Sica Carmelo Cappiello
01/05/2025	0.6	Unit e System Test Cases	Tutto il team

Team members

Nome	Acronimo	E-Mail
Cappiello Carmelo	CC	c.cappiello2@studenti.unisa.it
Sica Alessio	SA	a.sica104@studenti.unisa.it
Zappile Antonino	ZA	a.zappile1@studenti.unisa.it
Ragusa Francesco	RF	f.ragusa@studenti.unisa.it

Sommario

1 Obiettivi di Design (Design Goals)

Nella presente sezione si andranno a presentare i Design Goals, ovvero le qualità sulle quali il sistema deve essere focalizzato, formalizzati esplicitamente così che qualsiasi importante decisione di design può essere fatta consistentemente seguendo lo stesso insieme di design goal.

Seguendo le linee guida del libro Bernd Bruegge – Object Oriented Software Engineering i design goal sono statti suddivisi nelle seguenti categorie:

- **Performance**: includono i requisiti di spazio e velocità imposti sul sistema.
- **Dependability**: determinano quanto sforzo deve essere speso per minimizzare i fallimenti del sistema (crash, falle di sicurezza) e le loro conseguenze.
- **Maintenance**: determina quanto sforzo è necessario per modificare il sistema dopo il suo rilascio.
- **End User**: includono qualità che sono desiderabili dal punto di vista dell'utente, ma che non sono state coperte dai criteri di Performance e Dependability.

Ciascun design goal è descritto da:

- **Rank**, che ne specifica un valore di priorità compreso tra 1 e 16 (1 massima e 16 minima).
- **ID Design Goal**, un identificatore univoco e un nome esplicativo.
- **Descrizione,** una descrizione del design goal.
- **Categoria**, ovvero la categoria di appartenenza del design goal.
- **RNF di origine**, ovvero il requisito non funzionale che lo ha generato.

Design goals

Legenda:

- **GIALLO: Carmelo Cappiello**
- **BLU Alessio Sica**
- **VIOLA SCURO Antonino Zappile**
- **VERDE Francesco Ragusa**

Rank	ID Design Goal	Descrizione	Categoria	RNF di origine
9	DG_1 Tempi di risposta	Il sistema dovrebbe garantire un tempo di risposta non superiore a 5 secondi.	Performance	RNF_P_1
7	DG_2 Navigazione concorrente	Il sistema non dovrebbe subire rallentamenti anche con più di 50 utenti collegati contemporaneamente	Performance	RNF_P_2
4	DG_3 Gestione errori	Il sistema dovrebbe essere in grado di resistere e segnalare la maggior parte degli errori mediante l'uso di error handlers e logging.	Dependability	RNF_A_1
5	DG_4 Disponibilità del sistema	Il Sistema dovrebbe garantire la massima disponibilità, con un limite di 48 ore all'anno di downtime, salvo aggiornamenti alla piattaforma.	Dependability	RNF_P_4
1	DG_5 Sicurezza dei dati	Il sistema dovrebbe garantire la sicurezza e riservatezza dei dati trattati, utilizzando protocolli sicuri di comunicazione come HTTPS e metodi di crittografia per la persistenza come l' hashing	Dependability	RNF_L_1
2	DG_6 Manutenibilità	Il sistema dovrebbe essere facilmente manutenibile ed estendibile.	Maintenance	RNF_S_1
3	DG_7 Estendibilità	Il sistema dovrebbe prestarsi facilmente all'aggiunta di nuove funzionalità date le elevate necessità dell'utenza.	Maintenance	RNF_S_2
8	DG_8	Il sistema dovrebbe risultare facile da utilizzare anche da utenti con limitata	End User	RNF_U_1

	Facilità d'Uso	conoscenza tecnica rispettando anche convenzioni sull' ombreggiatura degli elementi		RNF_U_2
10	DG_9 Facilità di lettura	I contenuti della piattaforma dovrebbero essere facili da leggere con un contrasto conforme alle linee guida W3C con un punteggio minimo di AA	End User	N/A
11	DG_10 Feedback esplicito	Ogni azione all'interno della piattaforma in seguito ad un'interazione dell'utente dovrebbe comunicare un chiaro feedback allo stesso	End User	RNF_U_3
6	DG_11 Disponibilità sul Web	Il sistema dovrebbe essere deployato su una piattaforma web per garantire l'accessibilità dello stesso sulla più vasta gamma di dispositivi compatibili con un browser supportato	End User	RNF_IM_1

Trade-off

Trade-off	Descrizione
Tempi di risposta vs sicurezza	Per garantire una sicurezza del sito si punta ad implementare sistemi che aumentino la stessa a discapito della velocità delle operazioni, le quali potrebbero impiegare fino a 5 secondi.
Spazio vs velocità	Per garantire un tempo di accesso basso puntiamo ad aumentare leggermente la ridondanza delle variabili
Tempi di rilascio vs funzionalità	I tempi di rilascio sono stringenti, si preferisce rilasciare il software con meno funzionalità, pur di rilasciarlo in tempo

1.1 Definizioni, acronimi, e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema:** un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal:** le qualità sulle quali il sistema deve essere focalizzato.
- **Dati Persistenti**: dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software**: studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **SDD:** System Design Document
- **RAD:** Requirements Analysis Document

1.2 Organizzazione del documento

Il presente documento di System Design consta di quattro sezioni:

Introduzione: Viene descritto in generale lo scopo del sistema, gli obiettivi di design che il sistema propone di raggiungere.

Architettura software proposta: Viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software e la gestione dei dati persistenti.

Glossario: Contiene la lista dei termini usati nel documento con annessa spiegazione.

2 Architettura del sistema proposto

2.1 Panoramica sulla sezione

Il sistema proposto è basato sullo stile architetturale Three Tier e implementato tramite l'utilizzo del servlet container Tomcat.

Per la parte di presentazione verranno usati HTML5, CSS3, Bootstrap 5 e JQuery.

Per la logica applicativa e quindi il back-end sarà utilizzato Java.

Per la gestione del database saranno usati:

- **Oggetti DAO** per interagire con il database.
- **PhpMyAdmin e MySQL**

2.2 Decomposizione in sottosistemi

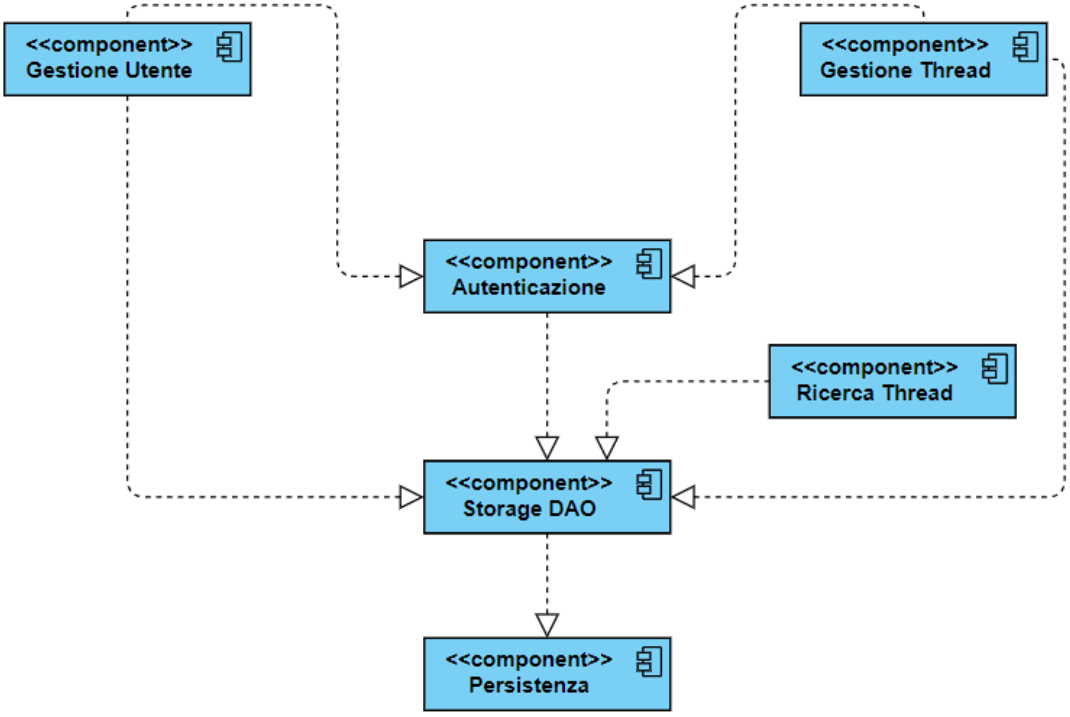
I sottosistemi individuati sono:

- **Autenticazione**: è responsabile delle funzionalità di login, logout ed eventuali filtri di accesso.
- **Gestione Utente**: si occupa di garantire le operazioni di registrazione e modifica account da parte dell'utente/moderatore e l'assegnazione di ban da parte del moderatore.

- **Gestione Thread**: responsabile della pubblicazione, modifica, valutazione e rimozione dei vari thread e commenti

- **Ricerca Thread**: si occupa di garantire la funzione di ricerca di un thread da parte di un utente generico
- **Storage DAO (Data Access Object)**: per offrire una ulteriore astrazione tra il layer di persistenza e gli altri sottosistemi.
- **Persistenza**: gestisce la persistenza dei dati su un database.
    - La persistenza dei dati sarà gestita da un COTS (Component Off The Shelf): un DBMS relazionale.

Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un component diagram UML:

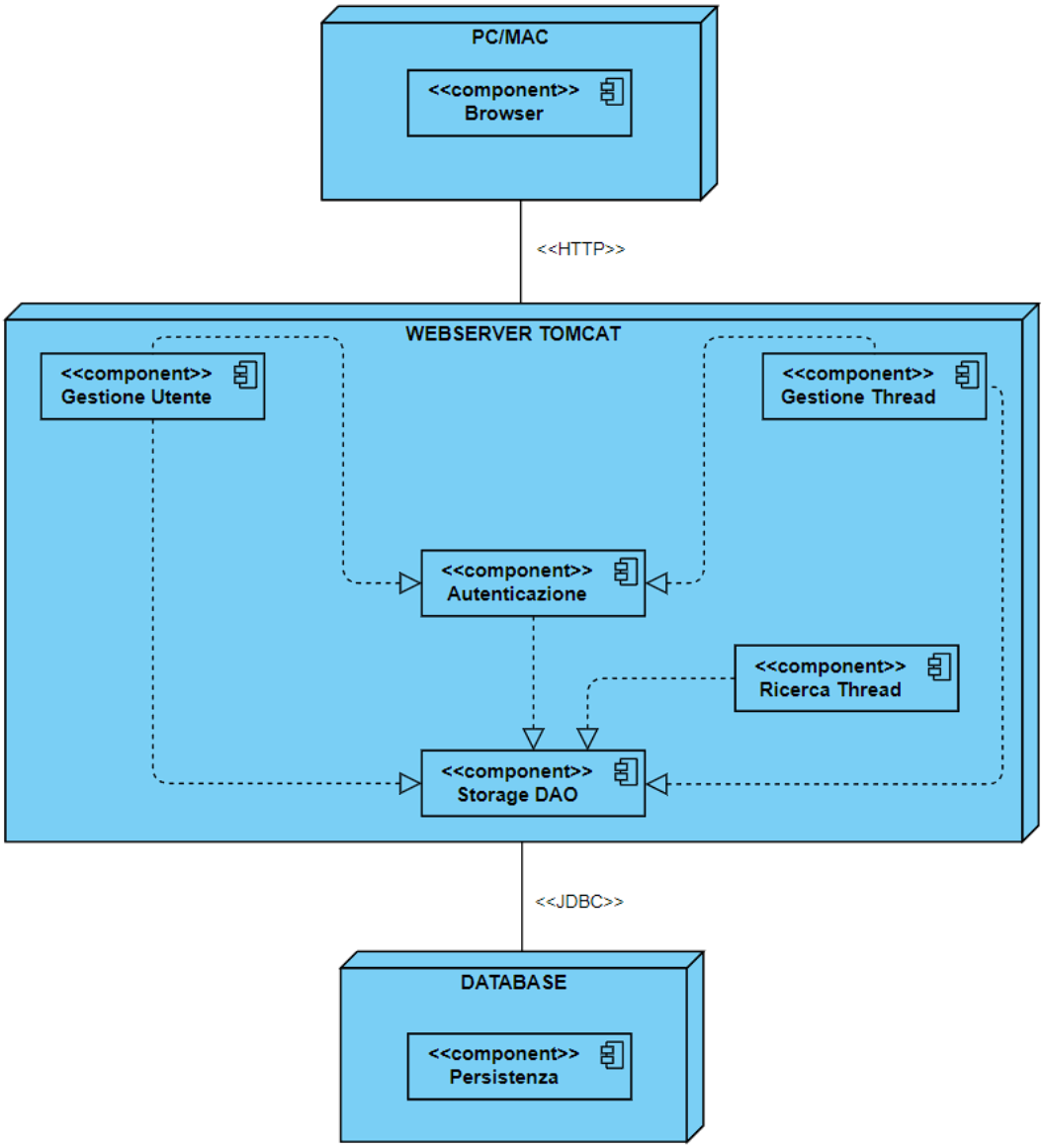


2.3 Mapping hardware/software

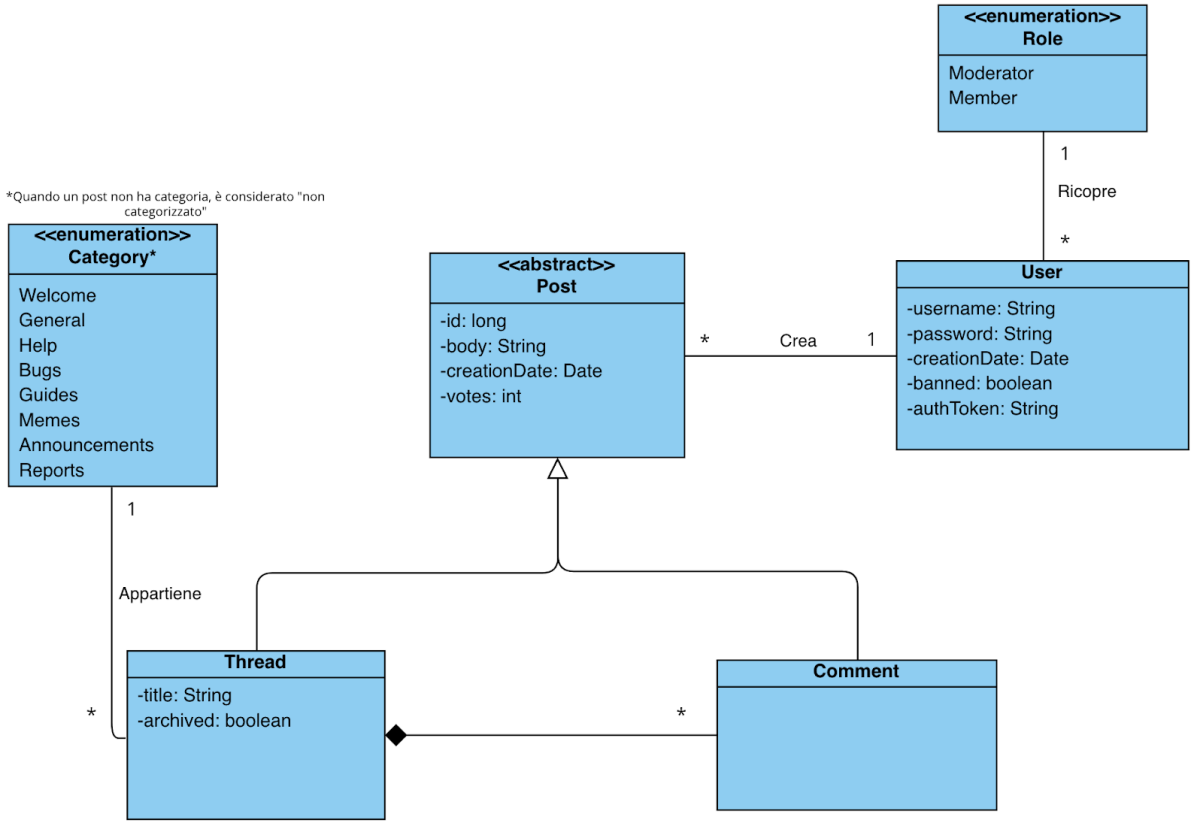
L' applicazione web che verrà sviluppata si basa su una piattaforma hardware costituita da un server che risponde alle richieste effettuate dai clients da una qualsiasi macchina con un browser e una connessione ad Internet.

Poiché il nostro sistema è una web application basata su un'architettura non distribuita, risiede su un solo nodo (quello del web server).

Di seguito un UML deployment diagram che descrive il mapping hardware/software:


PC/MAC
<<component>>
Browser
<<HTTP>>
WEBSERVER TOMCAT
<<component>>
Gestione Utente
<<component>>
Gestione Thread
<<component>>
Autenticazione
<<component>>
Ricerca Thread
<<component>>
Storage DAO
<<JDBC>>
DATABASE
<<component>>
Persistenza

Class diagram ristrutturato



Dizionario dei dati

Di seguito sono riportati gli attributi per ogni entità individuata.

Nome entità	**Utente**			
Descrizione	Contiene informazioni relative ad un utente registrato sul forum.			
Campo	**Tipo**	**Chiave**	**Null**	**Note**
username	VARCHAR(24)	PRIMARY KEY		Nome utente univoco.
password	CHAR(64)			Hash della password.
creation_date	DATE			Data di creazione dell'account.
banned	BOOLEAN			Indica se l'utente è bannato.

role	ENUM			Può essere `Moderator` o `Member`.
Auth_token	CHAR(64)			Token per login automatico/sessione sul sito web

Nome entità	**Thread**			
Descrizione	Contiene informazioni relative ad un thread del forum			
Campo	**Tipo**	**Chiave**	**Null**	**Note**
id	BIGINT	PRIMARY KEY		Identificativo unico del thread.
username	VARCHAR(24)	FOREIGN KEY		Identificativo dell'utente che ha creato il thread.
title	VARCHAR(100)			Titolo del thread.
body	TEXT			Testo del thread.
votes	INT			Indica la valutazione del thread
archived	BOOLEAN			Indica se il thread è archiviato.
category	VARCHAR(20)	FOREIGN KEY	x	Può essere Welcome, General, ecc.
creation_date	DATE			Data di creazione del thread.

Nome entità	**Commento**			
Descrizione	Contiene informazioni relative ad un commento sotto a un thread.			
Campo	**Tipo**	**Chiave**	**Null**	**Note**

id	BIGINT	PRIMARY KEY		Identificativo unico del commento.
thread_id	BIGINT	FOREIGN KEY		ID del thread associato.
username	VARCHAR(24)	FOREIGN KEY	X	ID dell'utente associato.
body	TEXT			Contenuto del commento.
votes	INT			Indica la valutazione del commento
creation_date	DATE			Data di creazione del commento.

2.4 Condizioni limite

Nel presente paragrafo verranno presentate le boundary conditions inerenti all'avvio, spegnimento e fallimento del sistema.

Avvio del sistema

Identificativo	**UCBC_1 – Avvio del Sistema**	**Data**	**07/01/2025**
		Versione	1.0
		Autori	Alessio Sica, Carmelo Cappiello
Descrizione	Lo UC permette l'avvio del sistema.		
Attore principale	Amministratore		
Attori secondari	NA		
Entry condition	L'Amministratore è autenticato nel servizio di hosting		
Exit condition On success	Il sistema viene avviato correttamente		
Exit condition On failure	Il sistema non viene avviato		
Flusso di eventi principale			
1	Amministratore	Esegue sul server il comando che avvia il sistema.	

2	Sistema	Mette a disposizione i suoi servizi e le sue funzionalità agli utenti.
3	Sistema	Il sistema notifica l'amministratore dell'avvio andato a buon fine.

Spegnimento del sistema

Identificativo		**UCBC_2 – Spegnimento del Sistema**	**Data**	**07/01/2025**
			Versione	1.0
			Autori	Alessio Sica, Carmelo Cappiello
Descrizione		Lo UC permette lo spegnimento del sistema		
Attore principale		Amministratore		
Attori secondari		NA		
Entry condition		L'Amministratore accede al servizio di hosting AND Il Sistema è stato precedentemente avviato AND Il Sistema non è stato ancora spento		
Exit condition **On success**		Il sistema viene spento correttamente		
Flusso di eventi principale				
1	Amministratore	Invia un segnale di spegnimento al Sistema tramite il pannello di hosting		
2	Sistema	Se tutte le richieste in corso sono state soddisfatte, spegne il sistema.		
3	Sistema	Notifica l'Amministratore dell'avvenuto spegnimento del sistema.		

Fallimento del sistema

Identificativo	**UCBC_3 – Fallimento del Sistema**	**Data**	**24/11/2020**
		Versione	1.0
		Autori	Alessio Sica, Carmelo Cappiello, Antonino Zappile
Descrizione	Lo UC definisce il comportamento del Sistema in caso di fallimento.		
Attore principale	Amministratore		
Attori secondari	NA		
Entry condition	Il Sistema viene terminato inaspettatamente		
Exit condition On success	Il Sistema viene riavviato correttamene		
Exit condition On failure	Il Sistema non viene riavviato		
Flusso di eventi principale			
1	Amministratore	**Include UCBC_1**	

2.5 Servizi dei sottosistemi

Servizi Autenticazione

Servizio	Descrizione	Interfaccia
Login	Permette di entrare con il proprio account.	AuthenticationService
Logout	Permette di uscire dal proprio account.	AuthenticationService
Login con token	Permette di autenticarsi con un token di autenticazione.	AuthenticationService

Servizi Utente

Servizio	Descrizione	Interfaccia
Crea utente	Permette di creare un account	UserService
Rimuovi utente	Permette di eliminare un account	UserService
Aggiorna password	Permette di aggiornare la password di un account	UserService
Aggiorna token	Permette di aggiornare il token di autenticazione dell'utente	UserService
Trova utente	Permette di individuare il profilo di un utente	UserService
Trova utenti	Permette di effettuare una ricerca di utenti in base all'username	UserService
Trova utenti banditi	Permette di trovare tutti gli utenti banditi dal sito	UserService
Bandisci utente	Permette di bandire un utente dal sito	UserService

Servizi Gestione Thread

Servizio	Descrizione	Interfaccia

Crea thread	Permette di creare un thread	ThreadService
Rimuovi thread	Permette di rimuovere un thread	ThreadService
Aggiorna thread	Permette di aggiornare un thread	ThreadService
Valuta thread	Permette di valutare un thread (upvote, downvote)	ThreadService
Aggiorna categoria thread	Permette di aggiornare la categoria di un thread	ThreadService
Archivia thread	Permette di archiviare un thread, rendendolo read-only	ThreadService
Crea commento	Permette di creare un commento	CommentService
Rimuovi commento	Permette di eliminare un commento	CommentService
Valuta commento	Permette di valutare un commento (upvote, downvote)	CommentService

Servizi Gestione Ricerca

Servizio	Descrizione	Interfaccia
Cerca thread per ID	Permette di cercare un thread per il suo identificativo (ID)	ThreadService
Cerca threads	Permette di cercare threads filtrando per titolo, categoria, pagina, range di data e ordinamento	ThreadService

Cerca threads di un utente	Permette di cercare tutti i threads di un utente attraverso il suo username	ThreadService
Cerca commento per ID	Permette di cercare un commento tramite il suo identificativo (ID)	CommentService
Cerca commenti per ID thread	Permette di cercare i commenti di un thread tramite il suo identificativo (ID)	CommentService
Conta commenti per ID thread	Permette di contare i commenti di un thread in particolare (usato per la paginazione)	CommentService

Servizi DAO

Servizio	Descrizione	Interfaccia
Recupera	Permette di recuperare un oggetto entità in base alla sua chiave primaria	DAO
Recupera tutti	Permette di recuperare tutti gli oggetti entità	DAO
Salva	Permette di salvare un oggetto entità nel database	DAO
Aggiorna	Permette di aggiornare un oggetto entità nel database	DAO
Rimuovi	Permette di eliminare un oggetto entità dal database	DAO

Matrice degli accessi (*= solo se è proprio)

Attori Oggetti	Membro	Moderatore (estensione permessi membro)
Autenticazione	Login Logout Login con token	
Gestione Thread	Crea thread Rimuovi thread* Archivia thread* Valuta thread Crea commento Valuta commento Rimuovi commento*	Archivia thread Rimuovi thread Aggiorna thread Rimuovi commento
Gestione Ricerca	Cerca thread	
Gestione Utente	Registrazione Modifica Password*	Bandisci/Sbandisci utente Trova utenti banditi Modifica password

3 Cenni di Object Design

Durante lo sviluppo del progetto abbiamo adottato dei **design pattern** per semplificare il processo. In particolare, abbiamo scelto:

- **Abstract Factory:** un pattern creazionale che permette di creare una famiglia di prodotti attraverso una singola factory, non conoscendo i dettagli implementativi né dei prodotti né della factory.
    - Abbiamo usato questo pattern per la creazione degli oggetti Service e DAO con lo scopo di facilitare l'iniezione delle loro dipendenze

- **Bridge pattern:** pattern strutturale che consiste nel separare l'astrazione dall'implementazione così che quest'ultima possa essere sostituita anche a runtime.
    - Abbiamo fatto uso di questo pattern in ogni layer dell'applicazione web, garantendo modularità e manutenibilità.

4 Testing

Comprende lo **unit testing**, ovvero il testing del singolo metodo, isolato dal resto del sistema, mockando le classi da cui dipende.

Per il **system testing** utilizziamo **Postman**, un potente tool che permette di simulare richieste GET/POST, al fine di provare una funzionalità del sistema nella sua interezza.

Utilizzeremo il concetto di **category partition** per evidenziare solo i casi di test necessari evitando ridondanze.

ThreadServiceImpl::**createThread**

Category partition

Categoria	Valori Validi	Valori non validi
username	Non nullo, non vuoto	null, Stringa vuota
title	Non nullo, non vuoto	null, Stringa vuota
body	Non nullo, vuoto	null
category	Oggetto valido	null
threadValidator	Dati validi	Errore di validazione (Validation Exception)
threadDAO	Salvataggio OK, torna long	SQL Exception o DAOException
Session.user	Not null	Null

Casi di test individuati per unit testing (ID = UTC _TS#)

ID	username	title	body	category	validator	DAO / DB	Esito atteso
1	valido	valido	valido	valido	OK	OK	Ritorna **long** corretto
2	**null**	valido	valido	valido	OK	SQLException	Validation Exception

3	valido	**null**	valido	valido	X	-	Validation Exception
4	valido	valido	**null**	valido	X	-	Validation Exception
5	valido	valido	valido	**null**	X	-	Validation Exception
6	valido	Non nullo	Non nullo	Non nullo	X	-	Validation Exception
7	valido	valido	valido	valido	OK	SQLException	Service Exception
8	valido	valido	valido	valido	OK	DAO Exception	Service Exception

Casi di test individuati per system design (STC_TS#)

ID	user	title	body	category	validator	DAO / DB	Esito atteso
1	**null**	**-**	**-**	**-**	**-**	**-**	**Non autorizzato**
2	valido	valido	valido	valido	OK	OK	Ritorna **long** corretto
3	valido	**null**	valido	valido	X	-	Validation Exception
4	valido	valido	**null**	valido	X	-	Validation Exception
5	valido	valido	valido	**null**	X	-	Validation Exception
6	valido	Non nullo	Non nullo	Non nullo	X	-	Validation Exception
7	valido	valido	valido	valido	OK	x	Service Exception (SQLException)
8	valido	valido	valido	valido	OK	x	Service Exception (DAOException)

ThreadServiceImpl::**updateThread**

Category partition

Categoria	Valori Validi	Valori non validi
id	> 0 (purché esista il thread)	<= 0
title	Non nullo, non vuoto	null, Stringa vuota
body	Non nullo, vuoto	null
category	Enum Category valido	null
archived	false	true
Session.user	Non nullo (utente loggato)	null

Casi di test individuati per unit testing (TID = UTC _TS_U_#)

TID	id	title	body	category	archived	DAO / DB	Esito atteso
1	valido	valido	valido	valido	false	OK	OK
2	Non valido	-	-	-	-	-	Illegal Argument Exception
3	valido	valido	valido	valido	true	-	Illegal Argument Exception
4	valido	**null**	valido	valido	false	-	Validation Exception
5	valido	valido	**null**	valido	false	-	Validation Exception
6	valido	valido	valido	**null**	false	-	Validation Exception
7	valido	Non nullo	Non nullo	Non nullo	false	-	Validation Exception
8	valido	valido	valido	valido	false	SQL Exception	Service Exception

9	valido	valido	valido	valido	false	DAO Exception	Service Exception

Casi di test individuati per system design (STC_TS_U_#)

TID	user	id	title	body	category	archived	DAO / DB	Esito atteso
1	**null**	**-**	**-**	**-**	**-**	**-**	**-**	**Non autorizzato**
2	valido	valido	valido	valido	valido	false	OK	OK
3	valido	Non valido	-	-	-	-	-	Illegal Argument Exception
4	valido	valido	valido	valido	valido	true	-	Illegal Argument Exception
5	valido	valido	**null**	valido	valido	false	-	Validation Exception
6	valido	valido	valido	**null**	valido	false	-	Validation Exception
7	valido	valido	valido	valido	**null**	false	-	Validation Exception
8	valido	valido	Non nullo	Non nullo	Non nullo	false	-	Validation Exception
9	valido	valido	valido	valido	valido	false	SQL Exception	Service Exception
10	valido	valido	valido	valido	valido	false	DAO Exception	Service Exception

ThreadServiceImpl::**rateComment**

Category partition

Categoria	Valori validi	Valori non validi
commentId	ID di un commento esistente (> 0)	≤ 0, commento non esistente
userName	Nome utente valido (esistente), stringa non nulla e non vuota	null, stringa vuota, utente non esistente
threadId	ID thread associato valido (> 0, thread non archiviato)	<= 0
vote	-1 (downvote), 0 (rimozione), 1 (upvote)	qualsiasi altro valore (es. -2, 2)
commentDAO	Funziona correttamente	Lancia `DAOException`, errore di connessione
userDAO	Funziona correttamente	Lancia `DAOException`, errore di connessione
threadDAO	Funziona correttamente	Lancia `DAOException`, errore di connessione

Casi di test individuati per unit testing (ID = TC_CS#)

ID	commentId	userName	threadId	vote	Comment DAO	User DAO	Thread DAO	Esito atteso
1	**null**	Non nullo	Non nullo	1	valido	valido	valido	`Service Exception` (commentId non valido)
2	<=0	Non nullo	Non nullo	-1	valido	valido	valido	`Service Exception` (commentId non valido)
3	Non nullo	**null**	Non nullo	1	valido	valido	valido	`Service Exception` (username null)

4	Non nullo	stringa vuota	Non nullo	1	valido	valido	valido	`Service Exception` (username vuoto)
5	Non nullo	Non nullo	<=0	**1**	valido	valido	valido	`Service Exception` (threadId non valido)
6	Non nullo	Non nullo	Non nullo	2	valido	valido	valido	`Illegal Argument Exception` (vote non valido)
7	Non nullo	Non nullo	Non nullo	-2	valido	valido	valido	`Illegal Argument Exception` (vote non valido)
8	Non nullo	Non nullo	Non nullo	0	valido	valido	valido	Voto rimosso
9	Non nullo	Non nullo	Non nullo	1	valido	valido	valido	`ServiceException` (thread archiviato)
10	Non nullo	Non nullo	Non nullo	1	null / errore DAO	valido	valido	`ServiceException` (errore commentDAO)
11	Non nullo	Non nullo	Non nullo	1	valido	null / errore DAO	valido	`ServiceException` (errore userDAO)
12	Non nullo	Non nullo	Non nullo	1	valido	valido	null / errore DAO	`ServiceException` (errore threadDAO)
13	Non nullo	Non nullo	Non nullo	1	valido	valido	valido	Voto positivo registrato

14	Non nullo	Non nullo	Non nullo	-1	valido	valido	valido	Voto negativo registrato

ThreadServiceImpl::**findThreadByUsername**

Category partition

Categoria	Valori validi	Valori non validi
username	Stringa valida	Stringa vuota, Null
page	> 0	<=0
pageSize	> 0	<= 0
order	Enum	Null
startDate	Data valida e <= endDate, Null	Data non valida, Data > endDate
endDate	Null, Data valida e >= startDate	Data non valida, Data < endDate

Casi di test individuati per unit testing (TC_TSFU#)

ID	username	page	pageSize	order	startDate	endDate	Esito Atteso
1	Valido	Valido	Valido	Valido	Valido	Valido	**Valido**
2	**""**	Valido	Valido	Valido	Valido	Valido	Errore: username vuoto
3	**null**	Valido	Valido	Valido	Valido	Valido	Errore: username null
4	Valido	**0**	**10**	Valido	Valido	Valido	Valido
5	Valido	**1**	**0**	Valido	Valido	Valido	Valido
6	Valido	Valido	Valido	**"newest"**	Valido	Valido	Valido
7	Valido	Valido	Valido	**"oldest"**	Valido	Valido	Valido
8	Valido	Valido	Valido	**null**	Valido	Valido	Valido

9	Valido	Valido	Valido	Valido	**"2024-12-31"**	**"2023-01-01"**	Valido
10	Valido	Valido	Valido	Valido	**Non Valido**	Valido	Errore: startDate non valida
11	Valido	Valido	Valido	Valido	**null**	Valido	Valido
12	Valido	Valido	Valido	Valido	Valido	**Non Valido**	Errore: endDate non valida
13	Valido	Valido	Valido	Valido	Valido	**null**	Valido
14	Valido	Valido	Valido	Valido	**"2023-01-01"**	**"2022-12-31"**	Valido

5 Glossario

Nella presente sezione sono raccolti le sigle o i termini del documento che necessitano di una definizione.

Sigla/Termine	Definizione
Gaming Forum	Nome dell'applicativo.
COTS	Commercial Off The Shelf, si riferisce a componenti hardware e software disponibili sul mercato per l'acquisto da parte di aziende di sviluppo interessate a utilizzarli nei loro progetti.
DAO	Data Access Object, che si occupa di fornire accesso ai dati persistenti.