Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software 2025/2026*
Prof. Carmine Gravino

ClubConnect

SDD
System Design Document
ClubConnect

Riferimento	NC11_SDD
Versione	1.0
Data	21 nov 2025
Destinatario	Prof. Carmine Gravino
Presentato da	NC11 Team
Approvato Da	

Revision History

Data	Versione	Descrizione	Autori
21/11/2025	0.1	Prima stesura: scopo del sistema, obiettivi di design e trade-off	D.R. G.R.
22/11/2025	0.2	Stesura capitolo 3, architettura del sistema proposto	D.R. G.R. S.R.
24/11/2025	0.3	Stesura matrice degli accessi	D.R. G.R.
29/11/2025	0.4	Diagramma architetturale aggiunto	A.Z.
30/11/2025	0.5	Aggiunto Design Pattern: Facade	D.R A.Z G.R
01/12/2025	0.6	Aggiunto Design Pattern: Proxy	D.R. A.Z. G.R. S.R.
05/01/2026	1.1	Revisione della Gestione dati Persistenti e dei Design Pattern a seguito dei feedback	D.R. A.Z. G.R. S.R.

Membri del Team

Nome	Acronimo	Info Contatto
Andrea Zeno	A.Z.	a.zeno1@studenti.unisa.it
Domenico Ricciardelli	D.R.	d.ricciardelli1@studenti.unisa.it
Salvatore Pio Ruggiero	S.R.	s.ruggiero43@studenti.unisa.it
Gerardo Russo	G.R.	g.russo274@studenti.unisa.it

SOMMARIO

1 Introduzione

1.1 Scopo del sistema

Il progetto ClubConnect consiste nello sviluppo di un sistema web-based per la gestione digitale di associazioni, club sportivi o culturali.
L'applicazione mira a semplificare le attività amministrative e organizzative, consentendo ai responsabili di gestire in modo efficiente i soci, gli eventi, le quote associative e le comunicazioni interne.

Il sistema offrirà un'interfaccia intuitiva e responsiva con funzionalità differenziate a seconda del ruolo dell'utente (amministratore o socio).
Obiettivi del progetto:

- Digitalizzare la gestione dei soci e delle loro iscrizioni.
- Automatizzare il rinnovo delle tessere e la registrazione dei pagamenti.
- Fornire un calendario eventi condiviso per i membri dell'associazione.
- Permettere l'invio di comunicazioni interne via email o tramite bacheca.
- Generare report e esportazioni (in PDF) su soci, eventi e pagamenti.
- Garantire sicurezza, affidabilità e facilità d'uso.

1.2 Obiettivi di Design

Rango	ID Design Goal	Descrizione	Categoria	RNF di origine
5	DG_1 Facilità d'uso	L'utente finale deve poter utilizzare facilmente il sistema	End user	RNF_8
1	DG_2 Sicurezza del Sistema	Il sistema deve garantire all'utente massima sicurezza per quanto riguarda i dati sensibili e le informazioni dell'account	Dependability	RNF_4, RNF_5, RNF_6

4	DG_3 Scalabilità	Il sistema deve essere in grado di poter sopportare un numero consistente di utenti	Performance	RNF_3
3	DG_4 Modularità	Il sistema deve garantire la modularità, in modo tale da facilitare futuri aggiornamenti e manutenzione	Maintenance	RNF_12
8	DG_5 Avviso minimo di manutenzione	Il sistema deve avvisare gli utenti del periodo di manutenzione in tempo utile	Dependability	RNF_11
7	DG_6 Compatibilità Multipiattaforma	Il sistema deve garantire la compatibilità su diversi browser	Dependability	RNF_7, RNF_10
2	DG_7 Tempo di Risposta	Il sistema deve essere sempre reattivo e veloce	Performance	RNF_1, RNF_2
6	DG_8 Accessibilità	Il sistema deve garantire l'accessibilità a ogni tipo di utente	End user	RNF_9

1.3 Trade-off

Trade-off	Descrizione
Tempo di rilascio vs Funzionalità	Se i tempi di rilascio sono stringenti, possono essere rilasciate meno funzionalità di quelle richieste, ma nei tempi giusti.
Tempo di rilascio vs Qualità	Se il sistema in fase di rilascio presenta bug critici, è preferibile posticipare la data di consegna per garantire la risoluzione dei problemi.
Modularità vs Overhead di gestione	I microservizi offrono scalabilità e separazione chiara delle funzionalità, ma richiedono infrastrutture più complesse. È stata privilegiata la modularità per facilitare le modifiche e la manutenzione
Manutenibilità vs Disponibilità	Si è scelto un approccio di manutenzione interruttiva piuttosto che un'architettura ad alta disponibilità.

1.4 Definizioni, Acronimi e abbreviazioni

- **NFR**: requisito non funzionale
- **DG**: Design Goal: Obiettivi di design per il sistema proposto
- **Dati Persistenti**: Dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **SDD**: System Design Document
- **DBMS**: DataBase Management System
- **Trade-off**: Scelte e compromessi tra design goals

1.5 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:

- RAD
- SOW
- SDD
- TP

- TCS
- TIR
- TSR

1.6 Panoramica

Questo documento è formato da 4 Sezioni:

- Introduzione: Viene descritto in generale lo scopo del sistema, gli obiettivi di design che il sistema propone di raggiungere.
- Architettura software corrente: Viene descritto lo stato attuale dell'architettura del software già presente.
- Architettura software proposta: Viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software, la gestione dei dati persistenti. Verranno poi presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l'intero sistema.
- Servizi dei sottosistemi: Vengono descritti i servizi offerti da ciascun sottosistema, specificando i compiti, le funzionalità principali e le interazioni con altri sottosistemi.

2 Architettura del sistema corrente

Attualmente la gestione di un club o un'associazione non si basa su un'architettura software che permetta di poter dare una gestione a 360 gradi della propria organizzazione. Solitamente il tutto viene effettuato con un insieme eterogeneo di software come fogli di calcolo, app di messaggistica o e-mail.
L'obiettivo di ClubConnect è quello di offrire una struttura centralizzata, implementando un hub tramite il quale è possibile in modo semplice e intuitivo gestire il proprio gruppo.

3 Architettura del sistema proposto

3.1 Panoramica

L'architettura scelta per il sistema è 3-Tier. Questa decisione deriva dalla semplicità di implementazione di ogni sottosistema che, in base alla responsabilità, può essere assegnato ad una delle tre categorie che il modello 3-tier identifica.
Il modello 3-tier, inoltre, garantisce una maggiore sicurezza sui dati, in quanto la separazione di responsabilità fa sì che il client non possa accedere direttamente al database.
Centralizzando la logica di business nel livello intermedio, l'architettura 3-tier promuove la riutilizzabilità. Diversi tipi di client possono interfacciarsi con lo stesso back-end, garantendo coerenza nei dati e nelle regole di business su tutte le piattaforme.
Inoltre, permette di separare la logica di business dal client,facilitando la modifica, l'aggiunta o la rimozione di funzionalità, senza intaccare l'esperienza utente.

3.2 Scomposizione in sottosistemi

- **Autenticazione**: è responsabile delle funzionalità di Login, Logout, Registrazione
- **Gestione Account**: si occupa di visualizzare l'area utente e della modifica dati dell'account, funzionalità admin se attive.
- **Gestione gruppo (club e associazioni)**: si occupa della creazione di gruppi, visualizzazione gruppi e modifica gruppi, generazione report per utente gestore.
- **Gestione eventi**: si occupa della creazione, rimozione e visualizzazione di eventi.
- **Gestione Comunicazione**: si occupa della creazione e visualizzazione delle comunicazioni agli utenti, quelle interne ai gruppi e quelle globali.
- **Gestione Pagamento**: si occupa dell'aggiunta metodo di pagamento, visualizzazione ed eventuale rimozione, inoltre gestisce i pagamenti della retta dei club.
- **Gestione Sistema**: si occupa della gestione globale del sistema, del ban di gruppi o utenti, o dell'attivazione/disattivazione della modalità manutenzione

- **Persistenza**:si occupa di gestire la persistenza dei dati con un database, usando JDBC.

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

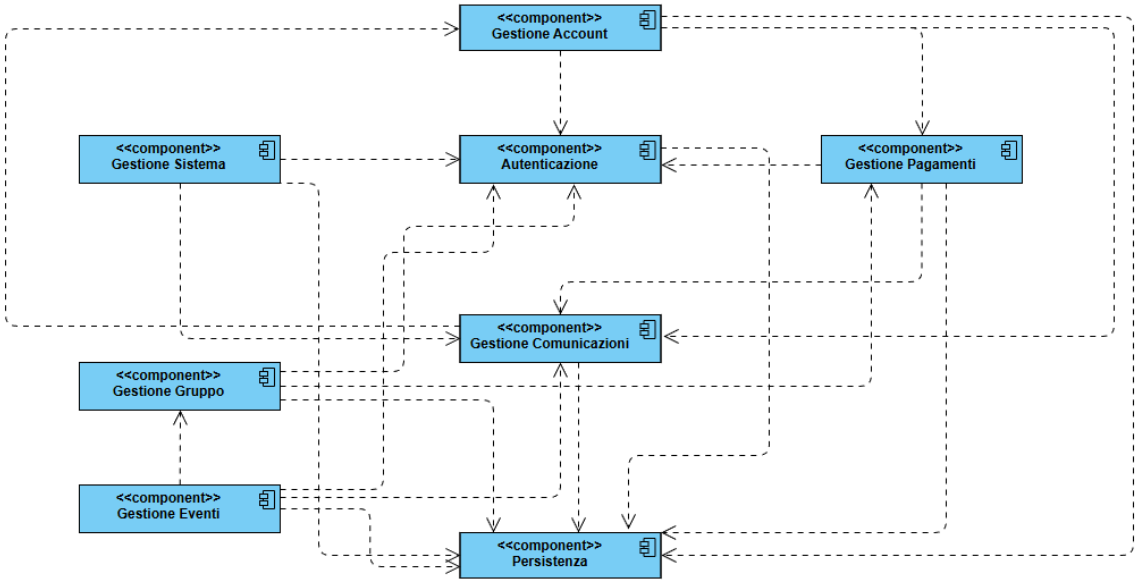
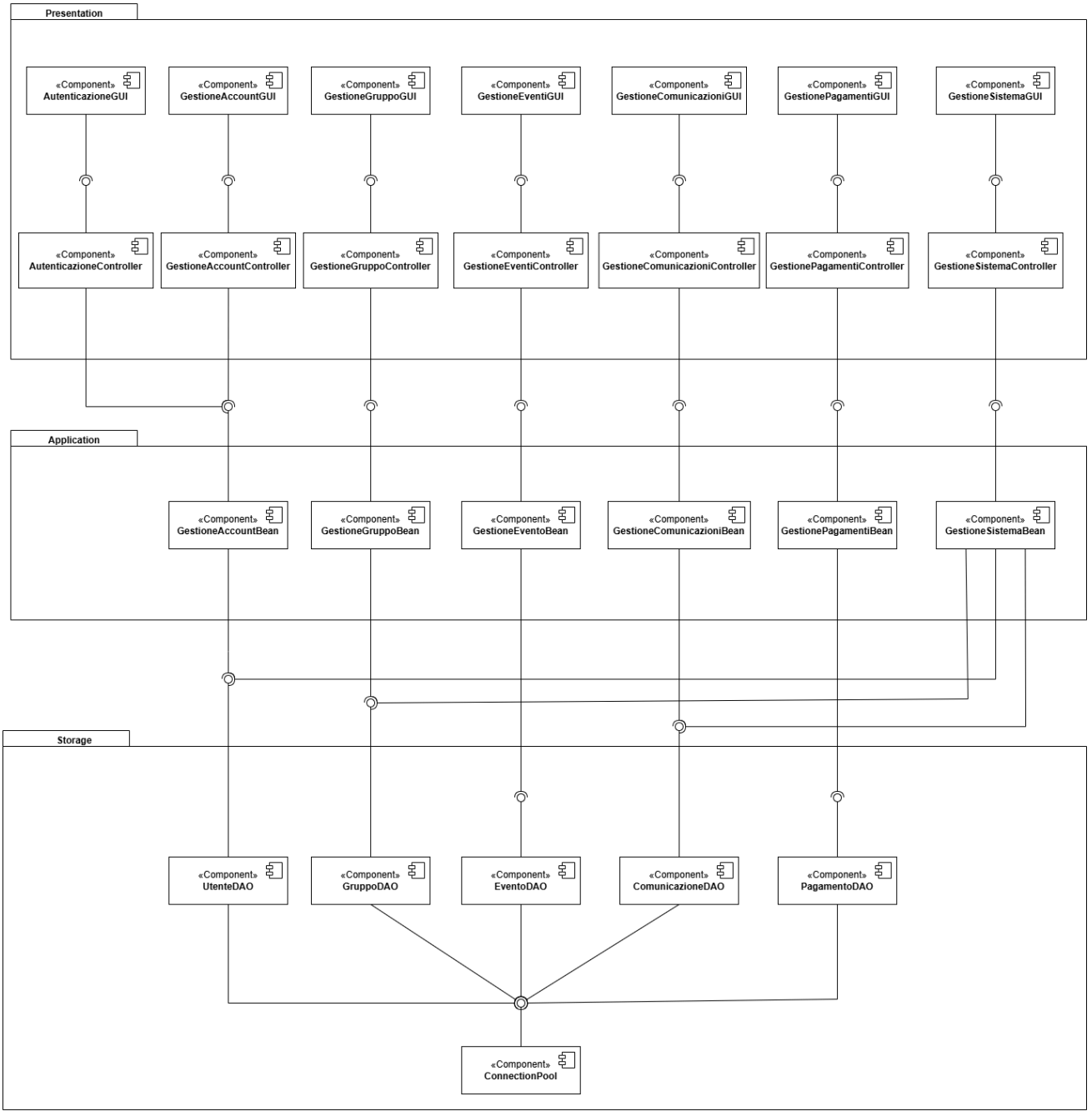


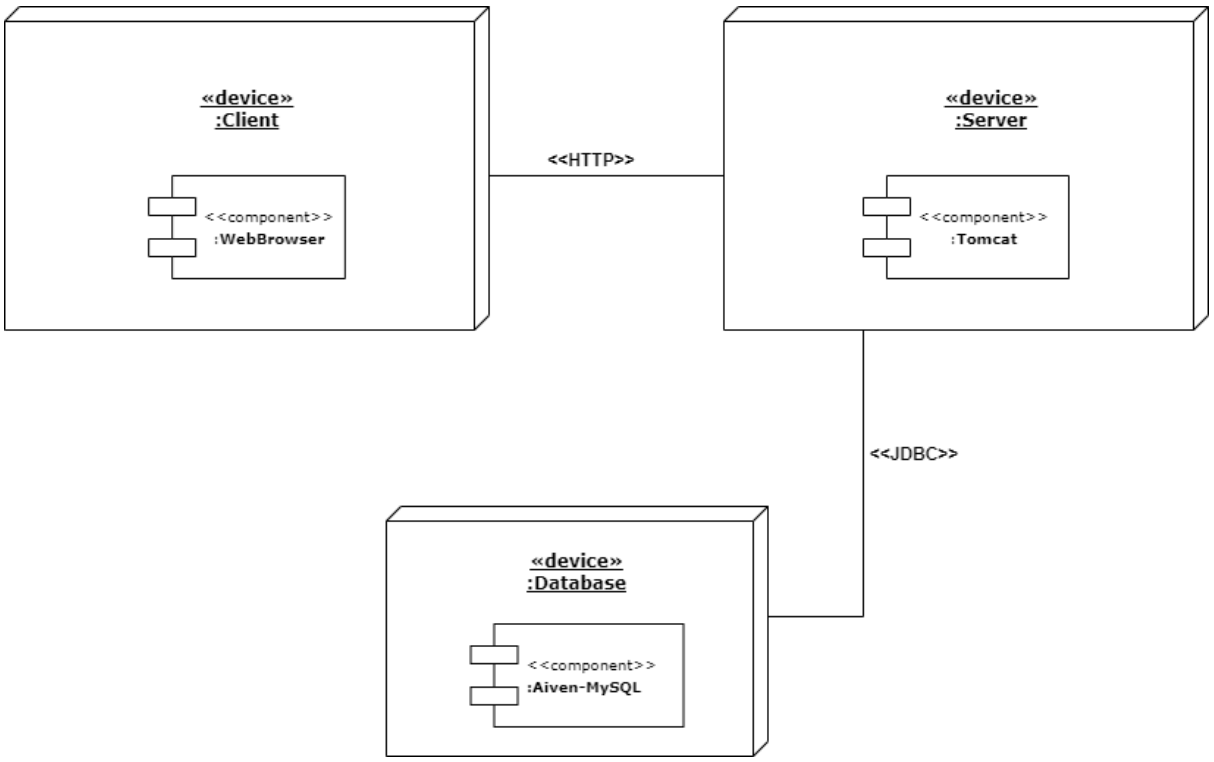
Diagramma Architetturale:



3.3 Mapping Hardware/Software

L' applicazione che verrà sviluppata si basa su una piattaforma hardware costituita da un server che risponde alle richieste effettuate dai clienti dai loro browser web. Essendo che il nostro sistema è una web app, il client browser comunicherà al server tramite richieste http, usando Tomcat, che a sua volta comunicherà col Database.

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



3.4 Gestione dei dati persistenti

Per gestire il salvataggio dei dati persistenti del sistema, abbiamo deciso di utilizzare un database relazionale, che permette di gestire facilmente l'accesso concorrente ai dati e di garantire la loro consistenza tramite l'uso del DBMS "Aiven".
La scelta è stata fatta per rispettare gli obiettivi di progettazione, offrendo:

- **Integrità dei dati:** "Aiven" applica vincoli per garantire l'integrità dei dati e li verifica durante gli aggiornamenti del database.
- **Privacy dei dati:** Diversi utenti possono accedere solo a specifiche parti del database e con permessi differenziati.
- **Affidabilità dei dati:** "Aiven"consente di effettuare backup e offre metodi per ripristinare il database in caso di guasti software o hardware.
- **Atomicità delle operazioni**: Le transazioni vengono eseguite interamente o non vengono eseguite affatto, mantenendo la coerenza del database con il modello reale.

Abbiamo deciso di utilizzare un database in cloud tramite Aiven. Questa soluzione ci consente di lavorare su un'unica istanza condivisa, riducendo i problemi legati

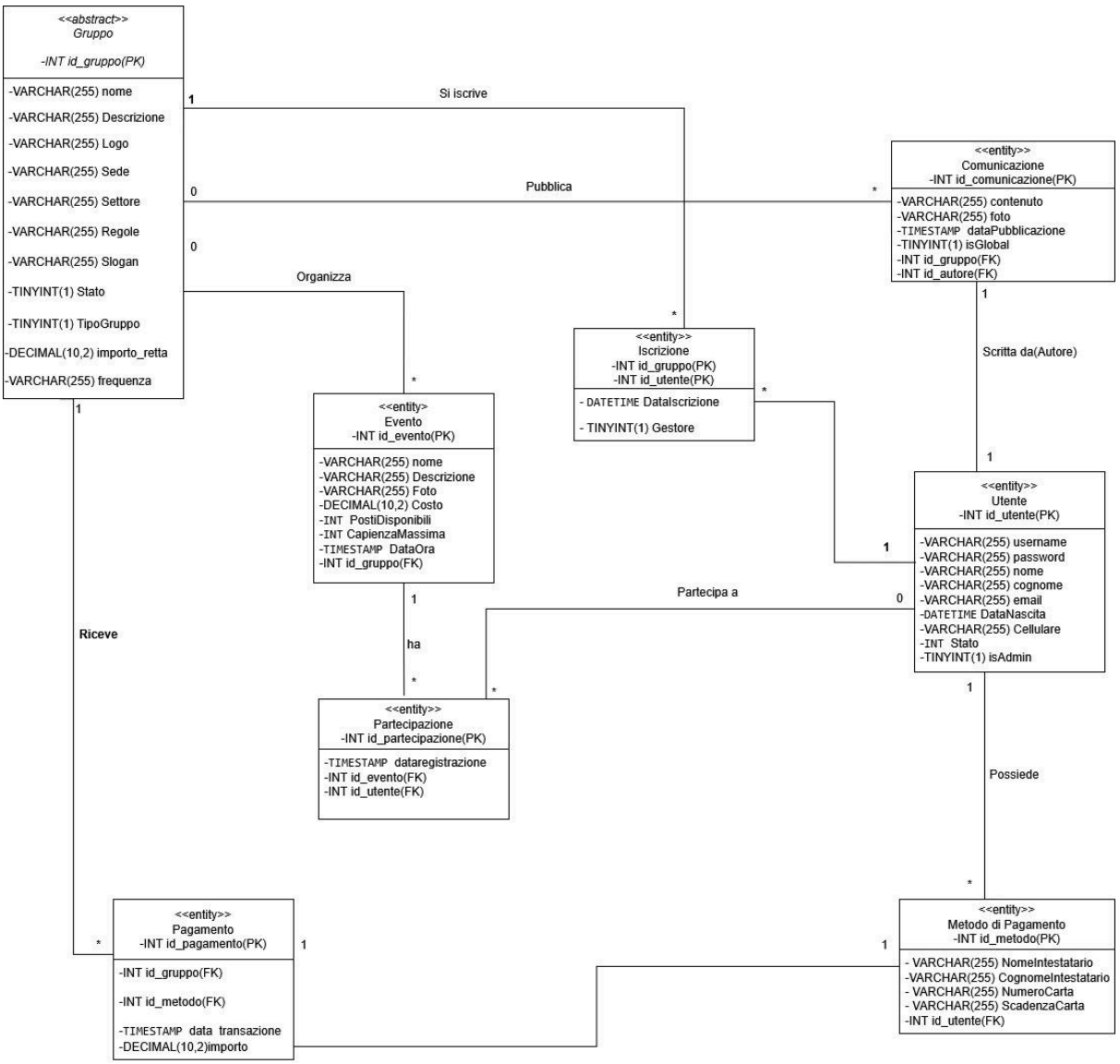
all'hosting locale del database e, di conseguenza, il tempo speso per risolvere eventuali errori.

3.4.1 Mapping dell'Ereditarietà (Strategia Single Table)

Per la persistenza della gerarchia delle classi Gruppo (che include le specializzazioni Club e Associazione), è stata scelta la strategia Single Table.
Questa scelta comporta la creazione di un'unica tabella Gruppo nel database che contiene:

- Tutti gli attributi della classe padre.
- Tutti gli attributi delle classi figlie (es. importo_retta per i Club).
- Una colonna discriminante (TipoGruppo) per distinguere se la riga rappresenta un Club o un'Associazione.

I campi non pertinenti per una specifica tipologia (es. la retta per un'Associazione) avranno valore NULL.
Questa strategia è stata preferita per massimizzare le performance in lettura, evitando JOIN costose.



Questa è la struttura che utilizzeremo per immagazzinare i dati persistenti.

3.5 Controllo degli accessi e sicurezza

Di seguito viene mostrata la matrice degli accessi per poter tenere traccia di quali attori possono accedere a quali servizi offerti dal sistema.

	Utente	Utente Tesserato	Utente Gestore	Admin
Autenticazione	Registrazione, Login, Logout, RecuperaPassword	Registrazione, Login, Logout, RecuperaPassword	Registrazione, Login, Logout, RecuperaPassword	Login, Logout, RecuperaPassword

Gestione Account	VisualizzaProfilo, ModificaDati, VisualizzaMetodi Pagamento	VisualizzaProfilo, ModificaDati, VisualizzaIscrizioniGr uppi, VisualizzaPagamenti VisualizzaMetodiPa gamento,	VisualizzaProfilo, ModificaDati, VisualizzaIscrizioni Gruppi, VisualizzaPagam enti VisualizzaMetodiP agamento,	VisualizzaProf ilo, ModificaDati,
Gestione Comunica zioni	VisualizzaComun icazioniGlobali	VisualizzaComunica zioniGruppo, VisualizzaComunica zioniGlobali	InviaComunicazi oneGruppo, VisualizzaComuni cazioniGruppo, VisualizzaComuni cazioniGlobali, EliminaComunica zioneGruppo	InviaComuni cazioneGlob ale, VisualizzaCo municazioni Globali, EliminaComu nicazioneGlo bale
Gestione eventi		IscrizioneEvento, VisualizzaCalendari oEventi, VisualizzaEvento	CreaEvento, IscrizioneEvento ModificaEvento, RimuoviEvento VisualizzaCalend arioEventi	
Gestione Sistema	SegnalaProblemi	SegnalaProblemi	SegnalaProblemi	BannaUtente , SbannaUtent e, SciogliGrupp o, AttivaManut enzione, DisattivaMan utenzione VisualizzaList aClienti,
Gestione Gruppo	IscrizioneGruppo VisualizzaGruppo	IscrizioneGruppo VisualizzaGruppo,	KickUtente, IscrizioneGruppo,	

	, CreaGruppo,	CreaGruppo,	VisualizzaGruppo, GeneraReport, CreaGruppo.	
Gestione Pagamenti	AggiungiMetodo Pagamento, RimuoviMetodoP agamento.	AggiungiMetodoPa gamento, RimuoviMetodoPag amento, PagaRetta.	AggiungiMetodo Pagamento, RimuoviMetodoP agamento, PagaRetta, ImpostaAbbona mento, ModificaAbbona mento	

3.6 Controllo globale del software

Il sistema è un sito web in cui ogni funzionalità viene avviata a seguito di una richiesta dell'utente, inviata tramite l'interfaccia grafica del sito. Quando l'utente desidera accedere a una determinata funzionalità, come effettuare il login o visualizzare una finestra specifica, invia un comando che viene interpretato dal sistema.
La richiesta viene gestita da un componente dedicato che analizza l'input dell'utente, esegue le operazioni necessarie sul server tramite il backend e produce una risposta che, a sua volta, viene gestita dal codice lato client. Il server è responsabile dell'indirizzamento del flusso verso i moduli applicativi appropriati, come la logica di controllo e la logica applicativa, per garantire una corretta elaborazione e un'esperienza fluida per l'utente.
Essendo un sito web che interagisce con un server, il sistema adotta un approccio basato su HTTP(Request-Response), in cui ogni azione dell'utente genera una richiesta HTTP che viene elaborata dal server,il sistema adotta un modello di controllo globale **procedure-driven**.

3.7 Condizioni Limite

Nonostante l'infrastruttura sottostante (TomCat e DBMS) gestisca automaticamente le eccezioni di basso livello, il sistema prevede use case specifici per la gestione delle condizioni limite, che sono poste sotto il controllo diretto dell'Amministratore di Sistema (ADMIN) che garantisce l'integrità dei dati e la continuità del servizio.
I casi d'uso che gestiscono le condizioni limite sono i seguenti:

- Avvio del server Tomcat
- Arresto del server Tomcat

- Attivazione modalità manutenzione
- Disattivazione modalità manutenzione

Identificativo: *UC_LC_1*	Avvio del server Tomcat	Data:	19/12/2025
		Vers.	0.1
		Autore:	Domenico Ricciardelli
Descrizione	L'amministratore avvia l'istanza del server Tomcat per rendere disponibile l'applicazione web.		
Attore Principale	ADMIN: un admin che vuole avviare l'applicazione web.		
Attori Secondari	N/A		
Entry Condition	L'admin ha accesso fisico o remoto alla macchina		
Exit Condition On Success	Il servizio Tomcat è attivo sulla porta configurata e l'app è raggiungibile.		
Exit Condition On Failure	Il server rimane spento, le risorse non vengono allocate.		
Rilevanza/User Priority	molto bassa		
Frequenza Stimata	1/mese		
Flusso di eventi Principale/Main Scenario			
1	Admin:	Un admin deve avviare il server Tomcat dell'applicazione web. Accede alla cartella bin di Tomcat tramite terminale/prompt e esegue lo script di avvio.	
2	Server:	Tomcat effettua il deploy dei file .war dell'applicazione web.	
Scenario/Flusso di eventi alternativo:Porta già occupata			
1.a1	Server:	Il server rileva che la porta 8080 è usata da un altro processo.	

1.a2	Server:	L'avvio fallisce e viene generato un log di errore.
Scenario/Flusso di eventi di ERRORE:Mancanza di connessione al database		
1.err1	Server:	Durante la fase di inizializzazione, l'applicazione tenta di stabilire una connessione con il Database Server.
1.err2	Server:	Il database non risponde o l'indirizzo IP non è raggiungibile
1.err3	Server:	Il sistema solleva un'eccezione di tipo CommunicationException.

Identificativo: *UC_LC_2*	Arresto del server Tomcat	Data:	20/12/2025
		Vers.	0.1
		Autore:	Domenico Ricciardelli
Descrizione	L'admin termina l'esecuzione del server Tomcat per manutenzione o aggiornamento dell'app.		
Attore Principale	ADMIN: un admin che vuole arrestare l'applicazione web.		
Attori Secondari	N/A		
Entry Condition	1) L'admin ha accesso fisico o remoto alla macchina. 2)Il server Tomcat è attualmente in stato running.		
Exit Condition On Success	Il processo Java relativo a Tomcat è terminato e le porte sono liberate.		
Exit Condition On Failure	Il server rimane attivo.		
Rilevanza/User Priority	molto bassa		

Frequenza Stimata	1/mese	
Flusso di eventi Principale/Main Scenario		
1	Admin:	Un admin deve arrestare il server Tomcat dell'applicazione web. Accede alla cartella bin di Tomcat tramite terminale e esegue lo script di arresto.
2	Server:	Tomcat invia un segnale di stop a tutti i moduli e le web app caricate e chiude le connessioni ai database e rilascia i thread attivi.
Scenario/Flusso di eventi di ERRORE:Il server non risponde al comando di arresto		
2.err1	Admin:	Invia il comando di shutdown, ma il terminale restituisce un errore o rimane in attesa infinita.
2.err2	Server:	Un thread dell'applicazione impedisce la chiusura pulita.
2.err3	Admin:	Esegue la chiusura forzata del processo.
Scenario/Flusso di eventi di ERRORE:Permessi insufficienti		
2.err4	Admin:	Esegue lo script di shutdown.
2.err5	Server:	Il sistema nega l'accesso perché l'admin non ha i privilegi necessari per terminare il processo.

Identificativo: *UC_LC_3*	Attivare Modalità Manutenzione	Data:	20/12/2025
		Vers.	0.1
		Autore:	Domenico

			Ricciardelli
Descrizione	L'amministratore mette temporaneamente l'applicazione in modalità manutenzione per consentire aggiornamenti, interventi tecnici o correzioni di bug, impedendo l'accesso agli utenti non amministratori fino al termine delle operazioni.		
Attore Principale	ADMIN: un admin che vuole attivare la modalità manutenzione del sistema.		
Attori Secondari	N/A		
Entry Condition	L'admin è autenticato nel sistema e dispone dei permessi per modificare lo stato operativo dell'applicazione.		
Exit Condition On Success	L'applicazione entra correttamente in modalità manutenzione: l'accesso degli utenti non amministratori è temporaneamente disabilitato, e viene mostrata una pagina di avviso.		
Exit Condition On Failure	Il sistema non riesce a modificare lo stato dell'applicazione e mostra un messaggio di errore all'amministratore.		
Rilevanza/User Priority	molto bassa		
Frequenza Stimata	1/mese		
Flusso di eventi Principale/Main Scenario			
1	Admin:	L'amministratore stabilisce che è necessario avviare una sessione di manutenzione del sistema. Accede alla sezione "Gestione sistema".	
2	Server:	Il sistema mostra le opzioni di configurazione, tra cui l'attivazione della modalità manutenzione.	
3	Admin:	L'amministratore sceglie l'opzione "Attiva modalità manutenzione"	
4	Sistema:	Il sistema chiede una conferma per evitare l'attivazione accidentale della modalità manutenzione.	

5	Admin:	L'amministratore conferma l'operazione.
6	Sistema:	Il sistema imposta lo stato dell'applicazione su "manutenzione", disattiva temporaneamente l'accesso per gli utenti non amministratori e mostra una pagina di avviso.
7	Admin:	L'amministratore visualizza un messaggio di conferma dell'attivazione della modalità manutenzione
Scenario/Flusso di eventi di alternativa: Modalità manutenzione già attiva		
3.a1	Sistema:	Il sistema rileva che la modalità manutenzione è già attiva.
3.a2	Sistema:	Il sistema mostra un messaggio: "La modalità manutenzione è già attiva" e non applica modifiche. Il flusso termina.
Scenario/Flusso di eventi di ERRORE:Problema di connessione		
3.err1	Sistema:	Il sistema non riesce a modificare lo stato dell'applicazione a causa di un problema di connessione.
3.err2	Sistema:	Il sistema mostra un messaggio di errore "Impossibile completare l'operazione. Riprovare più tardi." e reindirizza l'amministratore alla pagina "Gestione sistema".

Identificativo: *UC_LC_4*	Disattivare Modalità Manutenzione	Data:	20/12/2025
		Vers.	0.1
		Autore:	Salvatore Pio Ruggiero
Descrizione	L'amministratore disattiva la modalità manutenzione dell'applicazione, ripristinando lo stato operativo normale e consentendo nuovamente l'accesso a tutti gli utenti.		
Attore Principale	Admin: un utente con privilegi di amministratore che vuole disattivare la modalità manutenzione del sistema.		
Attori Secondari	N/A		
Entry Condition	L'admin è autenticato nel sistema e dispone dei permessi per modificare lo stato operativo dell'applicazione.		
Exit Condition On Success	L'applicazione esce correttamente dalla modalità manutenzione; il sistema ripristina lo stato operativo normale e riattiva l'accesso per tutti gli utenti.		
Exit Condition On Failure	Il sistema non riesce a modificare lo stato dell'applicazione e mostra un messaggio di errore all'amministratore.		
Rilevanza/User Priority	molto bassa		
Frequenza Stimata	1/mese		
Flusso di eventi Principale/Main Scenario			
1	Admin:	L'amministratore stabilisce che le operazioni di manutenzione sono terminate e decide di ripristinare il normale funzionamento dell'applicazione. Accede alla sezione "Gestione sistema".	
2	Server:	Il sistema mostra le opzioni di configurazione, tra cui la disattivazione della modalità manutenzione.	
3	Admin:	L'amministratore seleziona l'opzione	

		"Disattiva modalità manutenzione".
4	Sistema:	Il sistema richiede una conferma per evitare la disattivazione accidentale.
5	Admin:	L'amministratore conferma l'operazione.
6	Sistema:	Il sistema imposta lo stato dell'applicazione su "attivo", ripristina l'accesso per tutti gli utenti e mostra un messaggio di conferma all'amministratore.
Scenario/Flusso di eventi di alternativa: Applicazione già in stato attivo		
4.a1	Sistema:	Il sistema rileva che la modalità manutenzione non è attiva
4.a2	Sistema:	Il sistema mostra un messaggio "L'applicazione è già attiva" e non applica modifiche.
Scenario/Flusso di eventi di ERRORE:Problema di connessione		
4.err1	Sistema:	Il sistema non riesce a modificare lo stato dell'applicazione a causa di un problema di connessione.
4.err2	Sistema:	Il sistema mostra un messaggio di errore "Impossibile completare l'operazione. Riprovare più tardi." e reindirizza l'admin alla pagina "Gestione sistema".

4 Servizi dei Sottosistemi

In questa sezione vengono descritti i servizi di ogni sottosistema precedentemente elencati

Sottosistema Autenticazione

Servizio	Descrizione	Operazioni
Registrazione	Questa funzione permette a un utente di registrarsi	Registrazione
Login	Questa funzione permette a un utente di autenticarsi	Login
Logout	Questa funzione permette a un utente di effettuare l'uscita dal sistema	Logout
Recupera Password	Questa funzione permette a un utente di recuperare la password	ResetPassword

Sottosistema Gestione Account

Servizio	Descrizione	Operazioni
VisualizzaProfilo	Questa funzione permette all'utente di visualizzare il proprio account	GestioneAccount
ModificaDati	Questa funzione permette all'utente di poter modificare i propri dati (anagrafici e di accesso)	GestioneAccount, ModificaDatiAnagrafici, ModificaDatiAccesso
VisualizzaIscrizioniGruppi	Questa funzione permette all'utente di poter visualizzare le proprie iscrizioni ai gruppi	GestioneAccount
VisualizzaPagamenti	Questa funzione permette all'utente di	GestioneAccount

	poter visualizzare i pagamenti effettuati e da effettuare	
VisualizzaMetodiPagamento	Questa funzione permette all'utente di visualizzare i propri metodi di pagamento	GestioneAccount

Sottosistema Gestione Pagamenti

Servizio	Descrizione	Operazioni
Aggiungi Metodo di Pagamento	Questa funzione permette all'utente di aggiungere metodi di pagamento	GestioneAccount, AggiungiMetodoPagamento
Rimuovi Metodo di Pagamento	Questa funzione permette all'utente di rimuovere i propri metodi di pagamento	GestioneAccount, RimuoviMetodoPagamento
Pagare retta	Questa funzione permette di far pagare all'utente tesserato la retta del club	PagaRetta
Modificare l'importo della retta	Questa funzione permette all'utente gestore di un club di modificare l'importo della retta di un club	GestioneGruppo, ModificaAbbonamento
Impostare l'importo della retta	Questa funzione permette all'utente gestore di impostare l'importo della retta di un club	GestioneGruppo, ImpostaAbbonamento

Sottosistema Gestione Gruppo

Servizio	Descrizione	Operazioni
Iscrizione a un Gruppo	Questa funzione permette a un utente di iscriversi a un gruppo che sia club o associazione	Gestione Gruppo, IscrizioneGruppo
Visualizza un Gruppo	Questa funzione permette all'utente di visualizzare un gruppo	Gestione Gruppo, Visualizza Gruppo
Crea un Gruppo	Questa funzione permette all'utente di creare un gruppo(Associazione o club)	Gestione Gruppo CreaGruppo
Espellere un Utente	Questa funzione permette a un utente gestore di espellere un utente dal gruppo	Gestione Gruppo kickUtente
Genera un Report	Questa funzione permette a un utente gestore di generare un report settimanale o mensile	Gestione Gruppo GeneraReport

Sottosistema Gestione Eventi

Servizio	Descrizione	Operazioni
Iscrizione ad un Evento	Questa funzione permette a un utente tesserato di partecipare a un evento	Gestione Eventi, IscrizioneEvento
Crea un Evento	Questa funzione permette a un utente gestore di creare un evento	Gestione Eventi, CreaEvento

Modifica un Evento	Questa funzione permette a un utente gestore di modificare un evento.	Gestione Eventi, ModificaEvento
Rimuovi un Evento	Questa funzione permette a un utente gestore di rimuovere un evento.	Gestione Eventi, RimuoviEvento
Visualizza il Calendario degli Eventi	Questa funzione permette a un utente tesserato o gestore di visualizzare il calendario degli eventi.	Gestione Eventi, VisualizzaCalendarioEventi
Visualizza un Evento	Questa funzione permette a un utente tesserato o gestore di visualizzare un evento.	Gestione Eventi, VisualizzaEvento

Sottosistema Gestione Comunicazioni

Servizio	Descrizione	Operazioni
Invia una Comunicazione di Gruppo	Questa funzione permette a un utente gestore di inviare comunicazioni di gruppo	Gestione Comunicazione, InviaComunicazioneGruppo
Visualizza le Comunicazioni di Gruppo	Questa funzione permette a un utente di visualizzare le comunicazioni di gruppo	Gestione Comunicazione, VisualizzaComunicazioniGruppo
Visualizza le Comunicazioni Globali	Questa funzione permette a un utente di visualizzare le comunicazioni globali	Gestione Comunicazione, VisualizzaComunicazioniGlobali
Elimina una Comunicazione di	Questa funzione	Gestione Comunicazione,

Gruppo	permette a un utente gestore di rimuovere una comunicazione di gruppo	EliminaComunicazioneGruppo
Invia una Comunicazione Globale	Questa funzione permette a un utente admin di inviare una comunicazione globale.	Gestione Comunicazione, InviaComunicazioneGlobale
Elimina una Comunicazione Globale	Questa funzione permette a un utente admin di rimuovere una comunicazione globale.	Gestione Comunicazione, EliminaComunicazioneGlobale

Sottosistema Gestione Sistema

Servizio	Descrizione	Operazioni
Segnala un Problema	Questa funzione permette a un utente di segnalare un problema tramite e-mail	SegnalaProblema
Banna un Utente	Questa funzione permette a un utente admin di bannare un utente	BannaUtente
Sbanna un utente	Questa funzione permette a un utente admin di sbannare un utente	SbannaUtente
Sciogli un Gruppo	Questa funzione permette a un utente admin di sciogliere un gruppo	SciogliGruppo
Attiva la modalità Manutenzione	Questa funzione permette a un utente admin di attivare la modalità manutenzione	AttivaManutenzione

Disattiva la modalità Manutenzione	Questa funzione permette a un utente admin di disattivare la modalità manutenzione	DisattivaManutenzione
Visualizza tutti i Clienti	Questa funzione permette a un utente admin di visualizzare la lista di tutti i clienti.	VisualizzaClienti

5 Design Pattern

I design pattern utilizzati nel sistema sono i seguenti:

- **Proxy**: Questo pattern è essenziale in un'architettura 3-tier, poiché fa da intermediario tra livello di presentazione e logica di business, per gestire controlli di sicurezza.
- **Facade**: Questo pattern è essenziale per nascondere la complessità delle interazioni tra i vari sottosistemi, fornendo al livello di presentazione (i Controller) un'interfaccia unica e semplificata per eseguire operazioni complesse.

5.1 Proxy

5.1.1 Contesto

Nel sistema, abbiamo deciso di implementare il pattern **Protection Proxy** per blindare le funzionalità critiche riservate agli amministratori. Dato che l'applicazione espone metodi sensibili come il ban degli utenti (`bannaUtente`) o la gestione della modalità manutenzione (`attivaManutenzione` e `disattivaManutenzione`), è necessario garantire che nessun accesso non autorizzato raggiunga la logica di business e il database. Il Proxy agisce come un intermediario di sicurezza: intercetta la chiamata dal livello di presentazione, verifica i privilegi dell'utente e solo in caso di esito positivo delega l'esecuzione all'oggetto reale.

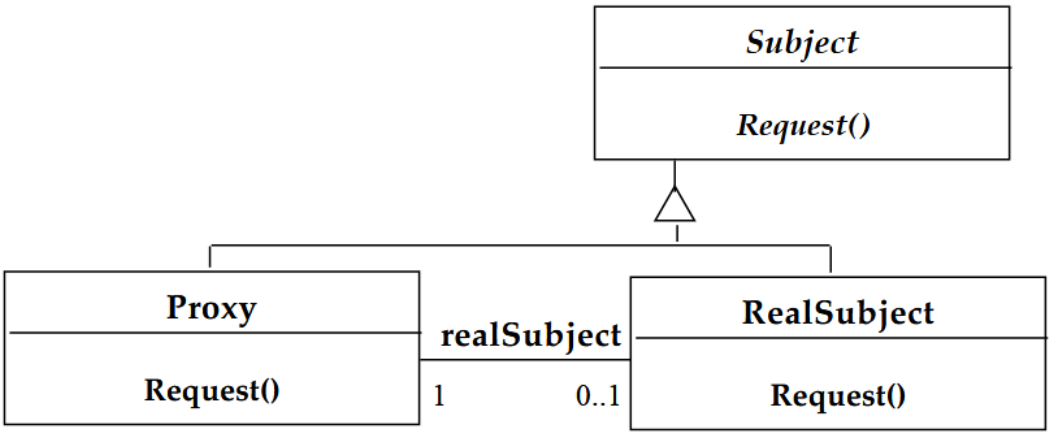
5.1.2 Struttura e partecipanti

L'implementazione del Proxy nel sistema ClubConnect si applica specificamente al sottosistema Gestione Sistema.

Componenti principali:

- **Subject** (`GestioneSistemaInterface`): Un'interfaccia Java che definisce i metodi comuni come `bannaUtente(int id)` o `attivaManutenzione()`. Garantisce che il Client (le Servlet) possa interagire con il Proxy e il RealSubject in modo trasparente.
- **Proxy** (`GestioneSistemaProxy`): È la classe che Implementa l'interfaccia e mantiene un riferimento al RealSubject. Nel costruttore riceve l'oggetto `UtenteBean` della sessione corrente. Ogni metodo implementato esegue prima un controllo privato (`checkAdminPermission`): se l'utente non ha il flag `isAdmin`, viene lanciata una `SecurityException`; se il controllo passa, la richiesta viene inoltrata al RealSubject.
- **RealSubject** (`GestioneSistemaBean`): Contiene la logica di business effettiva e le chiamate ai DAO . Questa classe non contiene controlli di sicurezza, poiché assume che chi la chiama sia già stato autorizzato.

Diagramma semplificato del funzionamento:



5.1.3 Dettagli sulle Istanze

Per chiarire il ciclo di vita degli oggetti e il flusso di controllo, l'esecuzione avviene nel seguente modo:

1. La Servlet (Client) recupera l'utente dalla sessione HTTP.
2. La Servlet crea un'istanza di `GestioneSistemaProxy`, passando l'oggetto utente al costruttore.
3. La Servlet invoca un metodo operativo, ad esempio `sistema.bannaUtente(id)`.
4. All'interno del metodo, il Proxy esegue il controllo di sicurezza verificando se l'utente è un amministratore.

5. Se l'utente non è autorizzato, il Proxy blocca l'esecuzione lanciando una `SecurityException` (che la Servlet catturerà per reindirizzare all'errore).
6. Se l'utente è autorizzato, il Proxy utilizza la sua istanza interna di `GestioneSistemaBean` (creata nel costruttore) per eseguire l'operazione sul database.

5.1.4 Vantaggi e Svantaggi

Vantaggi:

- **Separazione delle Responsabilità**: `GestioneSistemaBean` (Il RealSubject, la classe che gestisce la logica business) rimane pulito e si occupa solo di logica SQL. Tutta la complessità dei controlli if/else sui ruoli (`isAdmin`) è spostata nel `GestioneSistemaProxy`.
- **Maggiore Sicurezza**: Fornisce uno scudo protettivo. L'oggetto reale non viene mai istanziato o chiamato se il controllo di sicurezza nel Proxy fallisce. Questo previene accessi non autorizzati o accidentali alle funzioni critiche.
- **Flessibilità**: Si possono aggiungere nuovi controlli modificando solo il Proxy o aggiungendone un altro, senza toccare il codice delicato che interagisce con i dati.

Svantaggi:

- **Aumento della Complessità:** Il pattern ci costringe a creare un'interfaccia (`GestioneSistemaInterface`) che altrimenti non sarebbe strettamente necessaria in un'architettura semplice, aumentando il numero di file da gestire.
- **Overhead di Performance:** Ogni operazione di amministrazione richiede un passaggio aggiuntivo: l'istanziazione e il controllo del Proxy, anche se l'impatto è trascurabile rispetto alla sicurezza guadagnata.

5.2 Facade

5.2.1 Contesto

Nel sistema, il **pattern Facade** viene utilizzato per semplificare l'interazione tra i controller e la logica di business complessa che richiede la coordinazione di più sottosistemi. Questo è fondamentale per gestire operazioni articolate, come l'iscrizione a un evento, che coinvolge simultaneamente la verifica dei posti e l'invio della conferma.

Abbiamo scelto Facade perché l'architettura modulare del sistema rischiava di esporre troppa complessità ai controller. Il pattern ci permette di mascherare le dipendenze tra i vari service (GestioneEventi, GestioneComunicazioni), fornendo un unico punto di accesso per le operazioni transazionali.

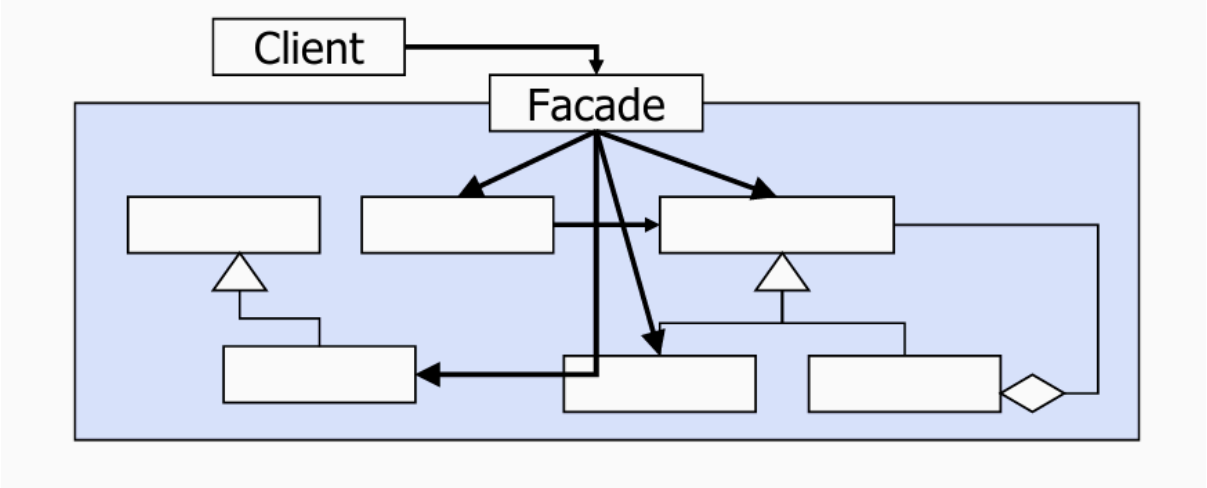
5.2.2 Struttura e partecipanti

L'implementazione del Facade nel nostro sistema si basa sulla classe `IscrizioneFacade`, che funge da interfaccia unificata per orchestrare le chiamate verso i diversi Bean del livello Application. Questo permette al Controller di invocare un solo metodo senza dover conoscere la logica interna di ogni sottosistema.

Componenti principali:

- **Facade** (`IscrizioneFacade`): È la classe che fornisce l'interfaccia unificata. Contiene il metodo `iscriviUtente(UtenteBean utente, int idEvento)` che incapsula la complessità del flusso di iscrizione.
- **Sottosistema 1** (`EventoService`): Responsabile della logica legata all'evento. Il Facade lo utilizza per verificare la disponibilità dei posti (`retrieveEvento`), registrare la partecipazione (`registraPartecipazione`) e aggiornare il contatore dei posti (`diminuisciPosti`).
- **Sottosistema 2** (`ComunicazioneService`): Responsabile delle notifiche. Il Facade lo invoca alla fine del processo per inviare la conferma all'utente (`inviaConfermaIscrizione`).
- **Client** (`IscrizioneEventoServlet`): Il controller che riceve la richiesta HTTP dall'utente. Invece di dover istanziare e gestire i due Bean separatamente, il Client istanzia solo `IscrizioneFacade` e invoca l'unico metodo `iscriviUtente`.

Diagramma semplificato del funzionamento:



5.2.3 Dettagli sulle Istanze

Per chiarire l'uso delle istanze, il flusso di esecuzione è il seguente:

1. Il Client crea un'istanza di `IscrizioneFacade`.
2. Il Client chiama `facade.iscriviUtente(...)`.
3. All'interno del metodo, il Facade crea (o recupera) un'istanza di `EventoService` e verifica i posti.
4. Se i posti sono disponibili, il Facade usa l'istanza di `EventoService` per salvare l'iscrizione.
5. Successivamente, il Facade crea un'istanza di `ComunicazioneService` e invoca il metodo di notifica.
6. Il Facade restituisce un booleano al Client indicando il successo o il fallimento dell'intera transazione.

5.2.4 Vantaggi e Svantaggi

Vantaggi:

- **Disaccoppiamento dei Controller**: I Controller (come `IscrizioneEventoServlet`) non hanno dipendenze dirette verso `ComunicazioneService` o `EventoService`. Questo permette di modificare la logica di notifica o di verifica posti senza dover mai rimettere mano al codice della Servlet.
- **Schermatura dei Sottosistemi**: Il Client viene isolato dai componenti interni. Ad esempio, se l'invio della conferma dovesse passare da una notifica interna a

una Email (tramite `sendEmail`), il Controller rimarrebbe totalmente ignaro del cambiamento.

- **Semplificazione dell'Interfaccia**: L'operazione di iscrizione, che richiede l'interazione coordinata tra recupero evento, verifica posti, registrazione partecipazione e invio conferma, viene ridotta alla singola invocazione di `iscriviUtente()`.
- **Centralizzazione della Logica**: Il flusso dell'operazione è gestito unicamente in `IscrizioneFacade`. Questo garantisce che non si verifichino mai situazioni in cui un utente viene iscritto (nel database eventi) senza che parta la relativa comunicazione (nel sottosistema comunicazioni).

Svantaggi:

- **Rischio "God Object"**: Se in futuro decidessimo di aggiungere a `IscrizioneFacade` anche la gestione dei pagamenti, dei rimborsi e della creazione gruppi, la classe diventerebbe troppo complessa. Per mitigare questo rischio nel progetto, abbiamo limitato le responsabilità della Facade alla sola orchestrazione dell'iscrizione.
- **Performance**: L'introduzione del livello `IscrizioneFacade` aggiunge un passaggio nella catena di chiamate tra Servlet e DAO. Nel contesto di un'applicazione web come ClubConnect, l'overhead di pochi millisecondi è considerato trascurabile rispetto ai benefici di manutenibilità ottenuti.