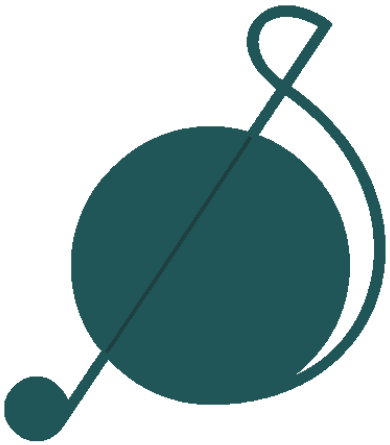
Laurea Triennale in informatica-Università di Salerno
Corso di
Ingegneria del software- Prof. Carmine Gravino



Spotibase

System Design Document

Spotibase

Docente
Gravino Carmine

Studenti
Silvio Venturino 0512108837
Umberto Della Monica 0512107742
Gagliarde Nicolapio 0512106980
Alessandro Aquino 0512106527

Revision History

Data	Versione	Descrizione	Autori
12/11/21	0.1	Creazione copertina, sommario, stesura design goals, design trade-off, acronimi, riferimenti, panoramici del documento, prima stesura della sezione "sistema proposto"	Tutto il team
19/11/21	0.2	Aggiunta deployment diagram, gestione dati persistenti, controllo accessi e sicurezza, flusso globale, condizioni limite, mapping HW/SW e diagrammi	Tutto il team
20/11/21	0.3	Aggiunta tabelle dati persistenti	NG
23/11/21	0.4	Aggiunta sezione dei sottoservizi	AA
26/11/21	0.5	Correzione di errori nei diagrammi	SV, UM
30/11/21	0.6	Aggiunta sezione scopo del sistema e correzione errori	SV, NG
01/12/21	1.0	Revisione documento	Tutto il team
06/12/21	1.1	Correzione utente guest e utente loggato in utente.	Tutto il team
10/12/21	1.2	Correzione architettura da MVC a three tier, correzione paragrafi, correzione trade-off memoria, correzione schemi	Tutto il team

Team Members

Nome	Ruolo nel progetto	Email	Acronimo
Alessandro Aquino	Team member	a.aquino33@studenti.unisa.it	AA
Nicolapio Gagliarde	Team member	n.gagliarde@studenti.unisa.it	NG
Umberto Della Monica	Team member	u.dellamonica2@studenti.unisa.it	UM
Silvio Venturino	Team member	s.venturino@studenti.unisa.it	SV

Sommario

1. Introduzione

1.1 Scopo del sistema

Lo scopo di Spotibase è quello di offrire ai propri utenti un'esperienza musicale completa, proponendo un ampio numero di canzoni e generi musicali. La semplicità di utilizzo, l'interfaccia estremamente intuitiva e i tool messi a disposizione, fanno di Spotibase un applicativo moderno, adatto a tutti i tipi di utente.
Il sistema progettato è una web app alla quale avrà accesso l'utente potrà registrarsi, per disporre di servizi aggiuntivi.
Il sistema tratta:

- Area Utente: riproduzione di brani musicali, aggiunta brani al carrello, possibilità di registrarsi, acquistare e scaricare canzoni, creare playlist e aggiungere tra i preferiti le canzoni.

1.2 Design Goals

I Design Goals sono organizzati in cinque categorie: Performance, Dependability, Cost, Maintenance, End User and Criteria.
I Design Goals identificati nel nostro sistema sono i seguenti:

Criteri di performance:

- **Tempo di risposta:**
  Il software deve consentire una navigazione rapida (meno di 1 secondo) a tutti i tipi di utenti, quindi tempi di risposta minimi nello svolgimento delle funzionalità da esso offerte.
- **Memoria:**
  La memoria complessiva del sistema dipenderà dalla memoria utilizzata per il mantenimento del Database e dal numero dei file mp3 e immagini che il sistema deve gestire.
  Nello specifico, per ogni 100 mila brani, i file mp3 occuperanno all'incirca 350 Gigabyte. Si prevede quindi un web server con una memoria di massa pari a 1 TB.

Criteri di affidabilità:

- **Robustezza:**
  Il sistema informerà l'utente di eventuali input errati attraverso messaggi di errore
- **Affidabilità**:
  Il sistema deve garantire il corretto funzionamento dei servizi offerti all'utente. Ad esempio la canzone riprodotta dal sistema sarà quella richiesta dall'utente. Il sistema verificherà la corretta creazione di playlist utilizzando i dati inseriti dall'utente. Il sistema gestirà in modo affidabile le preferenze che l'utente potrà esprimere, l'inserimento degli album e delle canzoni nel carrello e i relativi acquisti.
- **Disponibilità:**
  Una volta online, il sistema sarà disponibile per tutti gli utenti.
- **Tolleranza ai guasti:**
  Il sistema potrebbe essere soggetto a fallimenti dovuti a varie cause, ad esempio la rottura della macchina che ospiterà il

database. Per risolvere il problema, periodicamente sarà previsto un salvataggio dei dati sotto forma di codice SQL necessario alla rigenerazione del database

- **Sicurezza:**
  L'accesso al sistema sarà garantito mediante email e password.

Criteri di costi:

- **Costo di sviluppo:**
  È stimato un costo complessivo di 250 ore per la progettazione e lo sviluppo del sistema (50 per ogni membro del progetto).
- **Costo di deployment:**
  È stimato dal costo del web server e dai tools necessari al sistema.

Criteri di manutenzione:

- **Estensibilità:**
  Sarà possibile aggiungere nuove funzionalità al sistema, dettate dalle esigenze del cliente o dall'avvento di nuove tecnologie.
- **Modificabilità:**
  Il sistema sarà progettato in modo da rendere agevole la modifica di funzioni esistenti.
- **Leggibilità:**
  Il codice può essere compreso leggendo la documentazione allegata.
- **Adattabilità:**
  Il sistema è stato progettato per offrire i servizi legati alla riproduzione musicale e all'acquisto di brani. Di conseguenza spostare il sistema su un dominio applicativo diverso comporta sostanziali modifiche.
- **Portabilità:**
  L'interazione con il sistema avviene tramite browser, quindi possiamo definirlo portabile. Poiché il sistema viene sviluppato come una web application, esso è accessibile da qualunque

dispositivo, che sia esso mobile o meno, purché abbia un browser installato.
Inoltre il sistema è stato progettato per funzionare su una Java Virtual Machine, questo permette di spostare il sistema su macchine e sistemi operativi diversi.
Queste caratteristiche garantiscono la portabilità del sistema.

- **Tracciabilità dei requisiti:**
  La tracciabilità dei requisiti sarà possibile grazie ad una matrice di tracciabilità, attraverso la quale sarà possibile retrocedere al requisito associato ad ogni parte del progetto. La tracciabilità sarà garantita dalla fase di progettazione fino al testing.

Criteri di usabilità:

- **Usabilità:**

il sistema sarà di facile comprensione e utilizzo, permettendo di effettuare in modo semplice e immediato le varie operazioni grazie a un'interfaccia intuitiva, di facile comprensione e utilizzo. L'intuitività è garantita in quanto il sistema avrà una buona prevedibilità, cioè la risposta del sistema ad un'azione utente sarà corrispondente alle aspettative. L'utente saprà utilizzare le funzioni del sistema dopo 10 minuti di utilizzo.

- **Utilità:**

il lavoro dell'utente verrà supportato nel miglior modo possibile dal sistema, infatti l'utente compirà le operazioni consentite al fine di garantire un'esperienza di riproduzione musicale ottimale

1.2.1 Design Trade-off

Performance vs Memoria:

Il sistema deve garantire tempi di risposta minori di 1 secondo a discapito della memoria utilizzata. Poiché deve essere inviata una o più risposte ad ogni utente in base al servizio richiesto.

***Tempo di rilascio* vs Funzionalità:**

Con tempi di rilascio brevi, possono essere rilasciate versioni beta con funzionalità principali, ma nei tempi giusti. Ad esempio una prima versione può essere rilasciata con le funzionalità principali (es. riproduzione canzone) e successivamente verranno aggiunte funzionalità secondarie (es. gestione della coda di riproduzione).

***Tempo di rilascio* vs Qualità:**

Con tempi di rilascio brevi consegnamo la versione con dei bug a discapito della qualità, per ricevere feedback dal cliente ed essere coerenti con i tempi presi in accordo.

***Tempo di rilascio* vs Staffing:**

Aumentare il numero di unità del team potrebbe non influenzare il tempo di rilascio dell'applicativo. Si preferisce quindi, sfruttare le risorse che si hanno a disposizione, a fronte di un risparmio di risorse economiche.

Trade off	
Performance	Memoria
Tempo Di Rilascio	Funzionalità
Tempo Di Rilascio	Qualità
Tempo Di Rilascio	Staffing

1.3 Definizioni, acronimi e abbreviazioni

Definizioni

Canzone: breve composizione musicale.

Album: insieme di brani/canzoni che appartengono allo stesso artista.

Utente: utilizzatore della piattaforma.

Playlist: insieme di canzoni che non appartengono allo stesso artista ma scelte dall'utente.

Artista: creatore di canzoni e/o album.

Preferenza: scelta di un utente secondo al gusto soggettivo di valutazione

Admin: gestore della piattaforma.

Carrello: insieme di canzoni destinate all'acquisto.

Acronimi

SDD: Software Design Document (Documento di descrizione del design del software)

MU= Mock-Up

U=Utente

DP = Dati Persistenti

UCBC = Use Case Boundary Condition

1.4 Riferimenti

Libri:

- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition
- Software Engineering Tenth Edition

Autori:

- Bernd Bruegge & Allen H. Dutoit
- Ian Sommerville

Documenti: RAD_Spotibase – Requirements Analysis Document

1.5 Panoramica del documento

- **Capitolo 1:**
  Contiene l'introduzione agli obiettivi del sistema, i design goals, i trade-off e un elenco di definizioni, acronimi e abbreviazioni utili alla comprensione dell'intera documentazione.
- **Capitolo 2**
  Contiene la descrizione del sistema Corrente, o un sistema similare di riferimento.
- **Capitolo 3**
  Contiene la descrizione del sistema che verrà realizzato, degli obiettivi che andrà a realizzare, in cui sarà gestita la decomposizione in sottosistemi, il mapping Hardware/Software, la gestione dei dati persistenti, il controllo degli accessi e sicurezza, il controllo del flusso globale del sistema, le condizioni limite.
- **Capitolo 4**
  Contiene la rappresentazione dei servizi dei sottosistemi.

2.Architettura del Sistema Corrente

Attualmente esistono sistemi simili che permettono la riproduzione streaming on demand di canzoni come Spotify, Deezer, Amazon Music… . Questi sistemi anche se gestiscono la riproduzione e ascolto offline non hanno una vera e propria sezione e-commerce quindi con la possibilità di aggiungere al carrello e acquistare i file mp3.

3. Architettura del Sistema Proposto

3.1 Panoramica

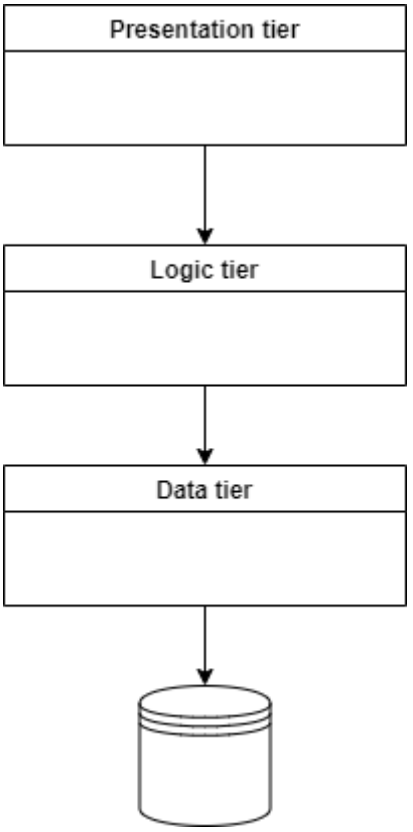
Il sistema che si vuole realizzare rientra nel campo della Green-field Engineering.
Il sistema proposto Spotibase è un'applicazione web. Il Sistema è rivolto a tutte le persone che vogliano ascoltare musica e vogliano tenere sotto controllo la loro libreria di canzoni.
Tutti gli utenti avranno la possibilità di registrarsi al sito oppure di continuare a navigare senza dover registrarti. Il software da noi proposto metterà a disposizione varie funzionalità, a seconda se l'utente ha effettuato l'accesso oppure no.
L'utente che ha effettuato l'accesso potrà inserire la preferenza per ogni canzone, potrà aggiungere le canzoni al carrello e acquistarle direttamente dalla piattaforma, potrà creare un numero massimo di playlist stabilito in base al tipo di abbonamento acquistato.

Nello specifico il sistema implementa un'architettura three-tier, diffusa nello sviluppo web e di Sistemi software object-oriented, in grado di separare la logica di presentazione dei dati dalla logica di business. Si tratta di un'architettura dove le funzionalità del sito sono separate e suddivise su più livelli di comunicazione tra loro.

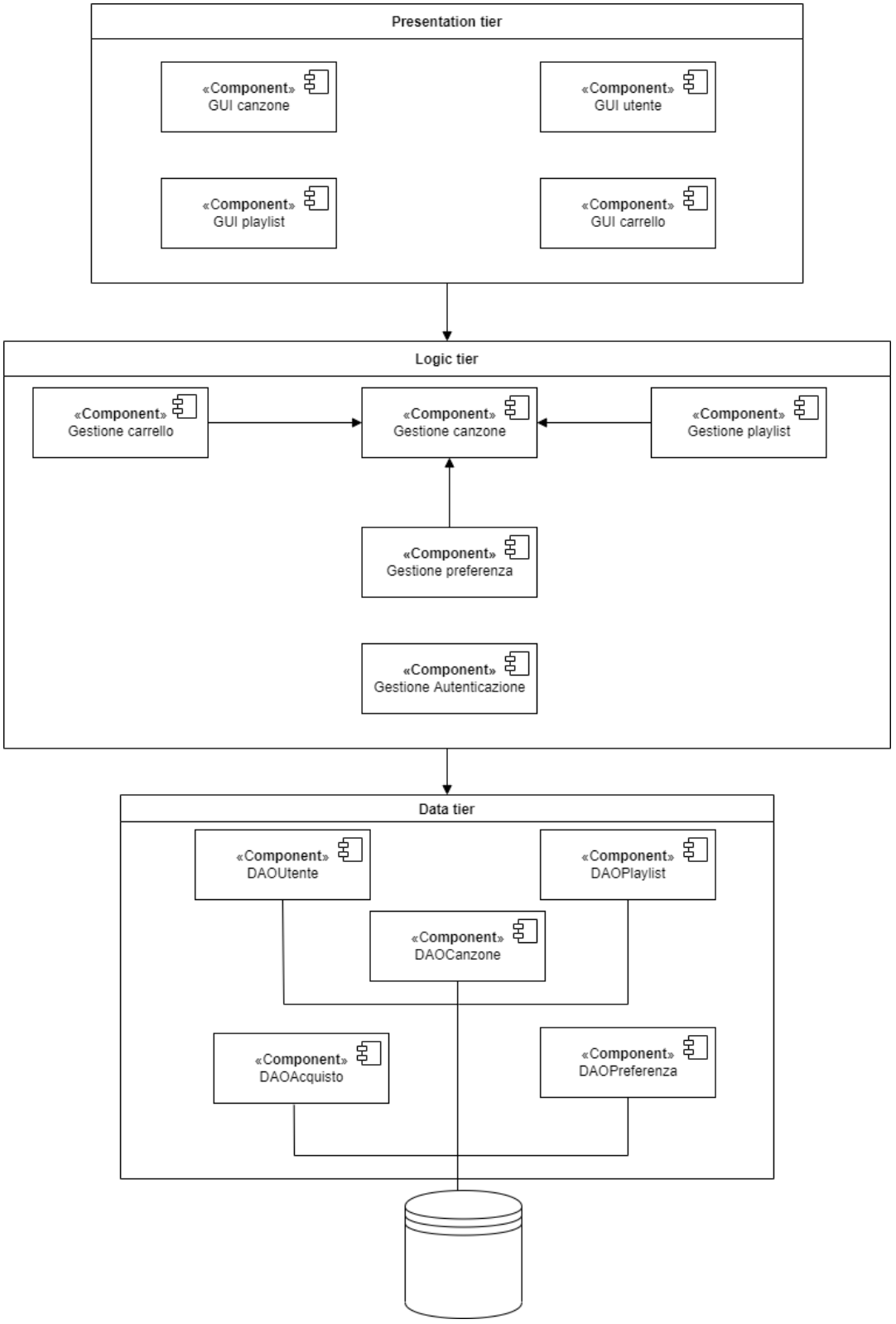
3.2 Decomposizione in sottosistemi

La decomposizione prevista per il sistema è composta da tre tier che si occupano di gestirne aspetti e funzionalità differenti:

- **Presentation tier**: raccoglie e gestisce elementi di interfaccia grafica e gli eventi generati su di essi
- **Logic tier**: si occupa della sequenza di interazioni con l'utente, modificando lo stato degli altri due tier;
- **Data tier**: si occupa della gestione dei dati utili all'applicazione;



Component Diagram



Il Logic tier gestisce 5 sottosistemi:

- Gestione Canzone
- Gestione Carrello
- Gestione Playlist
- Gestione di Autenticazione
- Gestione Preferenza

Il Presentation tier gestisce 4 sottosistemi:

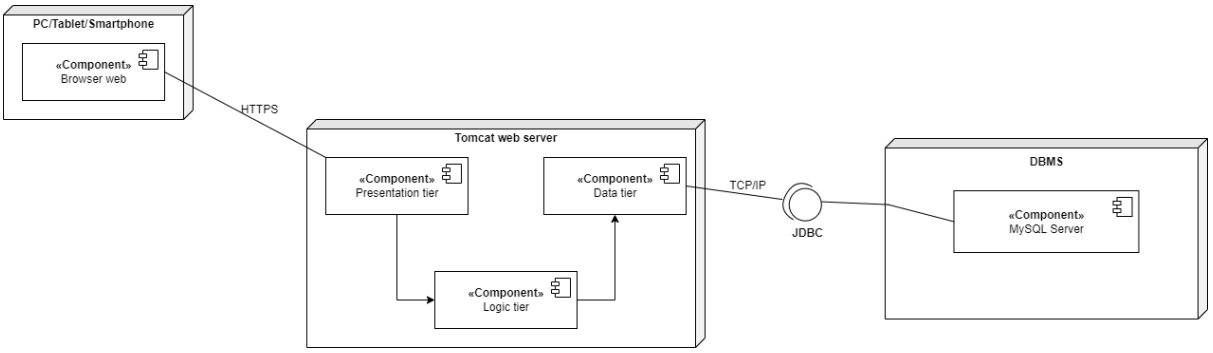
- GUI Carrello
- GUI Playlist
- GUI Utente
- GUI Canzone

Il Data tier gestisce 5 sottosistemi:

- DAOCanzone
- DAOUtente
- DAOAcquisto
- DAOPreferenza
- DAOPlaylist

Deployment Diagram

L'utente (Client) richiede le funzionalità tramite l'interfaccia che il sistema mette a disposizione a patto che si possieda un browser capace di interpretare JavaScript, in modo che le funzioni definite dal sistema possano eseguire in maniera corretta.
Il Client si connette al Sistema sul quale vengono eseguite le funzioni apposite richieste dal client.
La parte DBMS racchiude e gestisce la persistenza dei dati.
Il Sistema viene eseguito sul web server Tomcat.



3.3 Mapping Hardware/Software

Il sistema che si desidera sviluppare utilizzerà una struttura hardware composta da un Server che risponderà ai servizi richiesti dal client (un qualsiasi browser).
Il client è una macchina attraverso la quale un utente può collegarsi, usando una connessione a internet, per accedere al sistema mentre la macchina server gestisce la logica e i dati persistenti inseriti nel database.

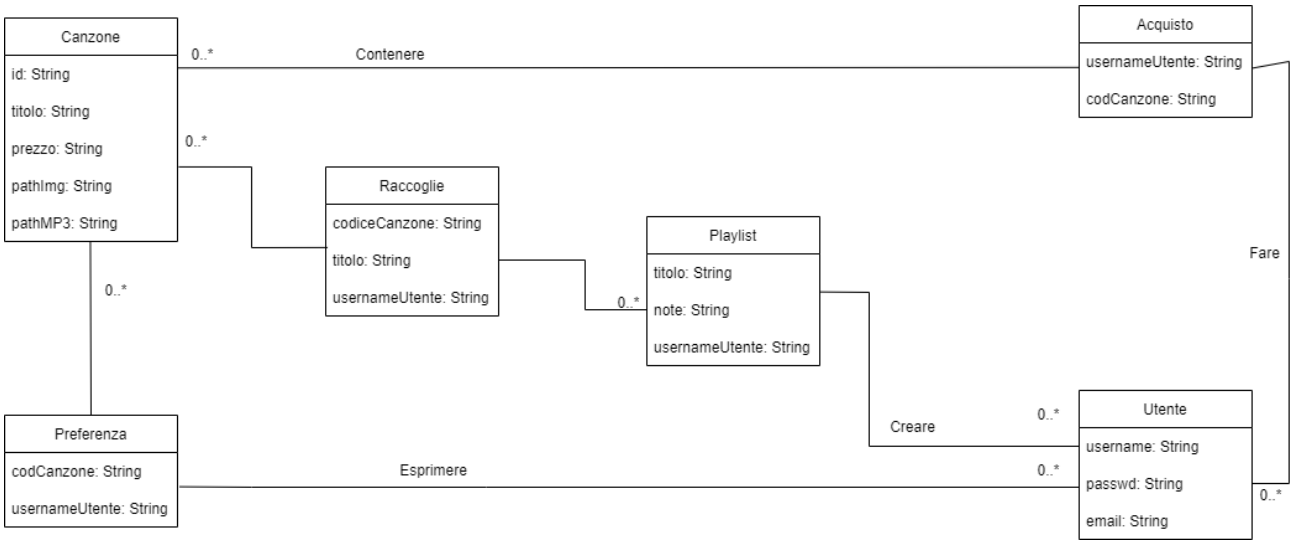
Client e server saranno connessi tramite il protocollo HTTPS, con cui il client inoltra richieste al server che provvederà a fornire i servizi richiesti. Inoltre verrà utilizzata la notazione JSON con la tecnica AJAX per permettere lo scambio di dati tra client e server in modo asincrono, aumentando quindi l'usabilità del sistema.

Le componenti hardware e software di cui ha bisogno il client sono un computer, un tablet oppure un qualsiasi dispositivo mobile dotato di connessione internet e un browser.

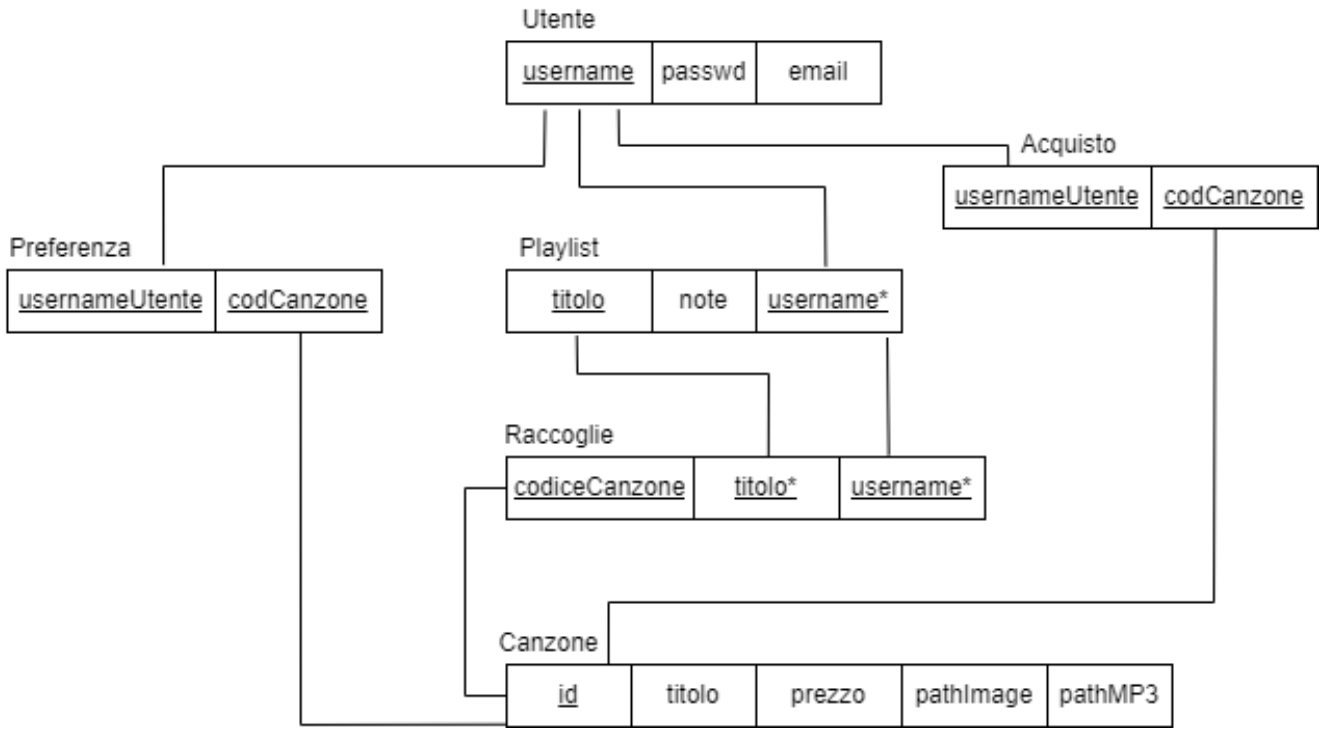
Per quanto riguarda il server, vi è la necessità di avere a disposizione una macchina avente connessione a internet e con capacità di immagazzinare grandi quantità di dati. La componente di cui si ha bisogno è un DBMS, nel nostro caso MySQL, per consentire il salvataggio dei dati in maniera persistente. Inoltre, è necessario usufruire dei servizi offerti da un web server, nello specifico Tomcat, per consentire la comunicazione con più client.

3.4 Gestione dati persistenti

Per la memorizzazione dei dati è stato scelto di utilizzare un Database Relazionale per consentire brevi tempi di risposta e ridurre i limiti di spazio di archiviazione.
I dati presenti nel database sono privatizzati, di conseguenza il DBMS permette di creare utenti diversi con permessi di lettura e scrittura opportuni.
Inoltre il DBMS gestisce i problemi di concorrenza e permette di effettuare il backup dei dati attraverso l'export dei file SQL.
Di seguito riportato il class diagram solo delle informazioni che restano persistenti nel Database, escludendo il carrello che verrà salvato in sessione. L'acquisto verrà invece salvato per tener traccia dello storico degli acquisti di un utente e per eventuali scelte implementative future.



Lo schema seguente evidenzia le chiavi e in che modo le istanze sono collegate tra di loro.



Di seguito vengono riportare le tabelle del database.

Canzone

NOME	TIPO	CONSTRAINTS	KEY
id	char(5)	not null	primary key
titolo	varchar(80)	not null	
prezzo	decimal(3,2)	not null	
pathImage	varchar(300)	not null	
pathMP3	varchar(300)	not null	

Utente

NOME	TIPO	CONSTRAINTS	KEY
username	varchar(40)	not null	primary key
passwd	varchar(128)	not null	
email	varchar(320)	not null, unique	

Raccoglie

NOME	TIPO	CONSTRAINTS	KEY
codiceCanzone	char(5)	not null	foreign key, primary key
titolo	varchar(100)	not null	foreign key, primary key
usernameUtente	varchar(40)	not null	foreign key, primary key

Playlist

NOME	TIPO	CONSTRAINTS	KEY
titolo	varchar(100)	not null	primary key
note	varchar(2000)		
usernameUtente	varchar(40)	not null	foreign key, primary key

Acquisto

NOME	TIPO	CONSTRAINS	KEY
usernameUtente	varchar(40)	not null	foreign key, primary key
codCanzone	varchar(5)	not null	foreign key, primary key

Preferenza

NOME	TIPO	CONSTRAINTS	KEY
codCanzone	char(5)	not null	foreign key, primary key
usernameUtente	varchar(40)	not null	foreign key, primary key

3.5 Controllo degli accessi e sicurezza

L'utente ha diversi permessi, un utente che non ha effettuato il login ha un sottoinsieme delle funzionalità messe a disposizione per l'utente che ha effettuato il login. Utilizziamo quindi una matrice degli accessi per capire cosa può fare un utente che ha fatto il login e cosa può fare un utente che non ha effettuato l'accesso al sistema.

Quando un utente prova ad effettuare un'operazione da lui non permessa, viene rimandato ad una pagina di errore.

Nella seguente tabella verranno riportate:

- L'intestazione in alto mostra i sottosistemi coinvolti;
- L'intestazione a sinistra gli attori;
- Il contenuto della tabella mostra le intersezioni tra attore e sottosistema che rappresenta il permesso di quell'attore su quel sottosistema.

	Autenticazione	**Canzone**	**Carrello**	**Acquisto**	**Preferenza**	**Playlist**
Utente che non ha effettuato il login	1)registrazione 2)login	1)riprodurre canzone 2)aggiunta canzone	1)visualizza carrello 2)rimuovere canzoni			
Utente che ha effettuato il login		1)riprodurre canzone 2)aggiunta canzone al carrello 3)Acquistare canzone 4)Scaricare canzone	1)visualizza carrello 2)rimuovere canzoni dal carrello 3)acquistare carrello	1)visualizzare gli acquisti effettuati	1)aggiungere preferenza canzone 2)rimuovere preferenza canzone 3)visualizzare le preferenze	1)creare playlist 2)inserire canzoni playlist 3)rimuovere canzoni playlist 3)creare playlist 4)cancellare playlist

L'attore può accedere al sistema attraverso un'interfaccia che chiederà email e password, il sistema farà i dovuti controlli e attiverà i permessi di acquisto, preferenza e gestione playlist. L'attivazione dei permessi sarà gestita dagli oggetti boundary e dagli oggetti control. Il sistema comunicherà con i browser degli utenti attraverso il protocollo sicuro HTTPS, e utilizzerà un protocollo per crittografare la password dell'utente.

3.6 Controllo flusso globale del sistema

Il flusso del sistema di Spotibase richiede una continua interazione dell'utente, per cui il controllo del flusso globale del sistema è di tipo event-driven, vale a dire che le azioni del sistema sono guidate dall'input dell'utente.
Per quanto riguarda la concorrenza, le funzionalità offerte dal Web Server, garantiscono un'interazione concorrente con tutti gli utenti connessi al sistema. Infatti, ogni utente connesso al sistema, tramite il suo browser web, avrà un thread dedicato tramite il quale il server interagirà con lui e si occuperà di eventuali sezioni critiche.

3.7 Condizione limite

Nel presente paragrafo verranno presentate le boundary conditions inerenti all'avvio del sistema, spegnimento del sistema, fallimento del sistema ed errore di accesso ai dati persistenti.

Avvio del sistema

Identificativo	UCBC_1	Data	04/01/22
		Versione	0.1
Nome	**Avvio del sistema**	Autore	Umberto Della Monica
Descrizione	Lo UC permette l'avvio del sistema		
Attore Principale	**Responsabile del sistema** E' interessato ad accendere il sistema		
Attori secondari	NA		
Entry Condition	Questo UCBC inizia quando il responsabile del sistema accede alla macchina server		
Exit condition On success	Questo UCBC termina quando il sistema viene avviato correttamente		
Exit condition On failure	Questo UC termina quando il sistema non viene avviato correttamente		
FLUSSO DI EVENTI PRINCIPALE			
1	Responsabile del sistema:	Esegue sulla macchina il comando che avvia il sistema	
2	Sistema:	Carica in memoria i componenti necessari per l'esecuzione del sistema	
Scenario/Flusso di eventi di ERRORE: sistema non riesce a caricare in memoria le componenti			
2.1	**Sistema:**	Visualizza un alert di errore al responsabile del sistema	

Spegnimento del Sistema

Identificativo	UCBC_2	Data	04/01/2022
		Versione	0.1
Nome	**Spegnimento del Sistema**	Autore	Umberto Della Monica
Descrizione	Lo UCBC permette lo spegnimento del sistema		
Attore Principale	**Responsabile del sistema**		
Attori secondari	NA		
Entry Condition	Questo UCBC inizia quando: L'Amministratore accede al Server AND Il Sistema è stato precedentemente avviato AND Il Sistema non è stato ancora spento		
Exit condition On success	Questo UCBC termina quando il sistema viene spento correttamente		
Exit condition On failure	Questo UCBC termina quando il sistema non viene spento correttamente		
FLUSSO DI EVENTI PRINCIPALE			
1	Responsabile del Sistema:	Invia un segnale di spegnimento al Sistema	
2	Sistema:	Dealloca le risorse e cancella i dati non persistenti e termina l'esecuzione del sistema.	

Fallimento del sistema

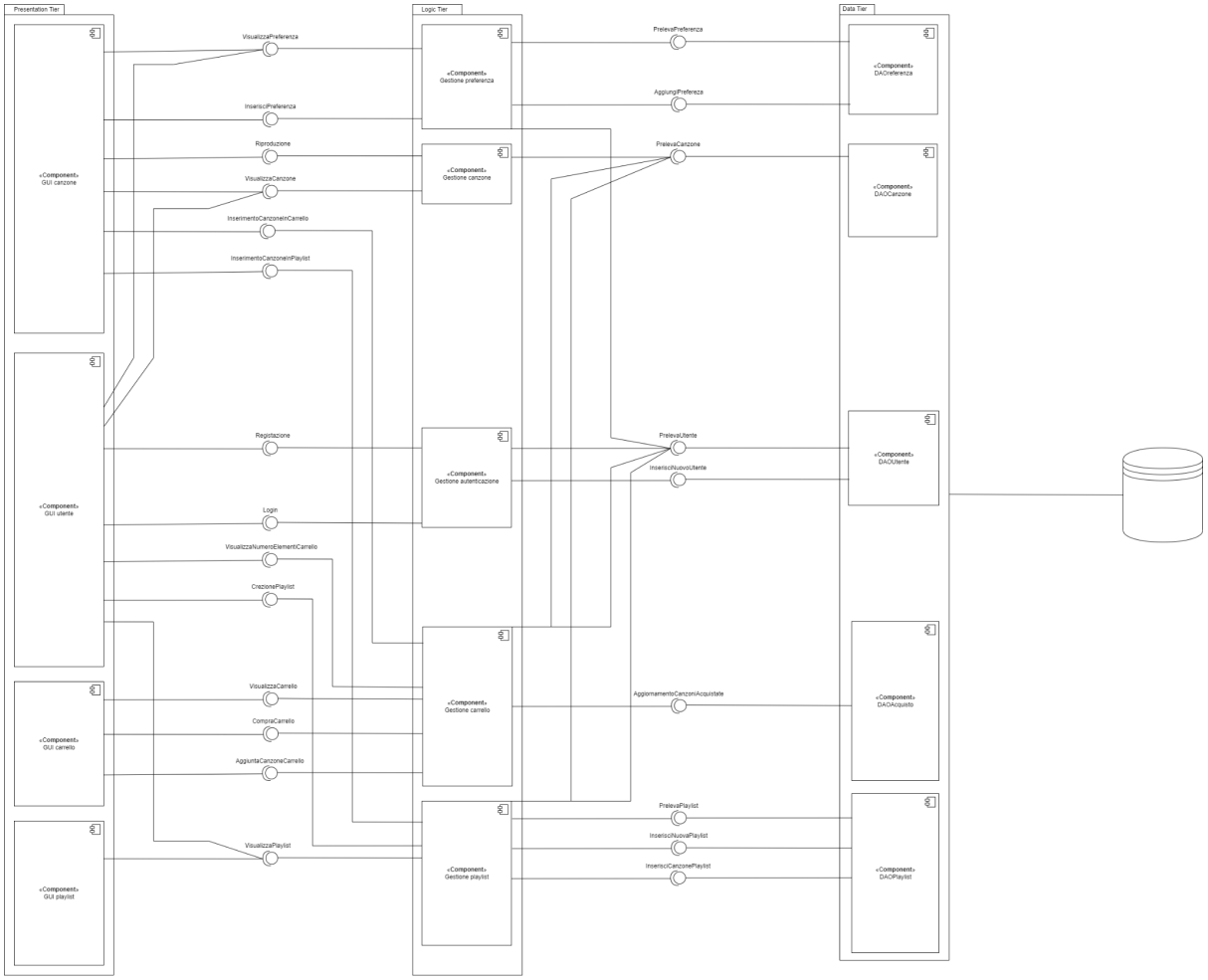
Identificativo	UCBC_3	Data	04/01/22
		Versione	0.1
Nome	**Fallimento del sistema**	Autore	Silvio Venturino
Descrizione	Lo UC definisce il comportamento del sistema in caso di fallimento		
Attore Principale	**Responsabile del sistema**		
Attori secondari	NA		
Entry Condition	Questo UCBC inizia quando il sistema termina inaspettatamente		
Exit condition On success	Questo UCBC termina quando il sistema viene riavviato correttamente		
Exit condition On failure	Questo UC termina quando il sistema non viene riavviato correttamente		
FLUSSO DI EVENTI PRINCIPALE			
1	Responsabile del sistema:	**Include UCBC_1**	

Errore di accesso ai dati persistenti

Identificativo	UCBC_4	Data	04/01/2022
		Versione	0.1
Nome	**Accesso ai Dati Persistenti**	Autore	Silvio Venturino
Descrizione	L'UCBC descrive il comportamento del sistema qualora fosse impossibile accedere ai dati persistenti o questi risultassero corrotti.		
Attore Principale	**Responsabile del sistema**		
Attori secondari	NA		
Entry Condition	Questo UCBC inizia quando: Il Sistema non può accedere ai dati persistenti OR I dati persistenti risultano corrotti.		
Exit condition On success	Questo UCBC termina quando il sistema riprende il normale funzionamento		
Exit condition On failure	Questo UCBC termina quando il sistema non riprende il normale funzionamento		

FLUSSO DI EVENTI PRINCIPALE		
1	Sistema:	Mostra una pagina di errore.
2	Responsabile del Sistema:	Ripristina il funzionamento del DBMS.
3	Responsabile del Sistema:	**Include UCBC_1**

4. Servizi dei Sottosistemi



Servizi offerti dal Logic tier per il sottosistema Presentation tier Gui Carrello:

- Visualizza Carrello;
- Compra Carrello;
- AggiuntaCanzoneCarrello

Servizi offerti dal Logic tier per il sottosistema Presentation tier Gui Canzone:

- Visualizza Preferenza
- Inserisci Preferenza
- Riproduzione Canzone
- Visualizza Canzone
- Inserimento Canzone In Carrello
- Inserimento Canzone In Playlist
- Visualizza Playlist

Servizi offerti dal Logic tier per il sottosistema Presentation tier Gui Utente:

- Visualizza Canzone
- Visualizza Preferenza
- Registrazione
- Login
- Visualizza Numero Elementi Carrello
- Creazione Playlist

Servizi offerti dal Logic tier per il sottosistema Presentation tier Gui Playlist:

- Visualizza Playlist

Servizi offerti dal Data tier per il sottosistema Logic tier Gestione Preferenza:

- Preleva preferenza

- Aggiungi Preferenza
- Preleva Canzone
- Preleva Utente

Servizi offerti dal Data tier per il sottosistema Logic tier Gestione Canzone:

- Preleva Canzone

Servizi offerti dal Data tier per il sottosistema Logic tier Gestione autenticazione:

- Preleva Utente
- Inserisci Nuovo Utente

Servizi offerti dal Data tier per il sottosistema Logic tier Gestione Carrello:

- Preleva Utente
- Preleva Canzone
- Aggiunta Canzone Carrello

Servizi offerti dal Data tier per il sottosistema Logic tier Gestione Playlist:

- Preleva Canzone
- Preleva Utente
- Preleva Playlist
- Inserisci Nuova Playlist
- Inserisci Canzone Playlist