



System Design Document

Introduzione	2
Obiettivi del sistema	2
Obiettivi di progettazione	3
Panoramica	5
Architettura del sistema proposto	6
Panoramica	6
Decomposizione in sottosistemi	7
Mappatura hardware/software	8
Gestione dei dati persistenti	9
Sicurezza e controllo degli accessi	14
Controllo del flusso globale del sistema	17
Condizioni limite	17

1. Introduzione

1.1. Obiettivi del sistema

Esistono sul Web diverse piattaforme volte alla recensione di contenuto multimediale, ma raramente queste si specializzano per una particolare categoria di media: la musica.

Siti che si occupano di poter far recensire esclusivamente contenuto musicale esistono, ma dimostrano di essere sistemi datati e non user-friendly, dando l'impressione di essere utilizzati esclusivamente da un'utenza elitista molto limitata.

La piattaforma commissionata **soundrate** punta ad attrarre qualsiasi categoria di utenza, con preferenze e musicali quanto più dispiagate, e permettere ai suoi utenti di pubblicare recensioni in modo semplice e agevole.

Oltre a poter pubblicare recensioni di progetti musicali, gli utenti potranno prendere visione e assegnare un voto (che potrà essere *positivo* o *negativo*) alle recensioni altrui, e effettuare segnalazioni se il contenuto di queste è ritenuto inappropriato.

L'utente disporrà inoltre di un proprio profilo, dove saranno raccolte tutte le recensioni pubblicate da esso, e di una personale lista d'ascolto di cui potrà usufruire per tenere traccia dei progetti musicali che intende ascoltare e, eventualmente, recensire.

1.2. Obiettivi di progettazione

Seguono gli obiettivi di progettazione individuati per il sistema.

1.2.1. Prestazioni

- **Tempi di risposta**
Il sistema deve garantire tempi di risposta brevi per ogni funzionalità, contenuti entro il secondo per gli utenti dotati di una connessione ad Internet moderna di almeno 7 Mbps.
- **Memoria**
Per la persistenza dei dati si utilizzerà una base di dati relazionali sotto il sistema di gestione di basi di dati relazionali. Per ottimizzare le prestazioni, il sistema implementerà una cache interna che conserverà i risultati delle richieste idempotenti più recenti.
- **Throughput**
Il sistema dovrà essere capace di gestire quantità elevate di richieste concorrentemente senza che ciò infici sul corretto funzionamento delle funzionalità del sistema, scalando e ridimensionandosi autonomamente se necessario.

1.2.2. Affidabilità

- **Robustezza**
Il sistema dovrà effettuare controlli estensivi al ricevere delle richieste per evitare di effettuare operazioni che lo porteranno ad uno stato inconsistente. Il sistema segnalerà ai client l'annullamento delle richieste non corrette.
- **Sicurezza**
Il sistema dovrà supportare delle funzionalità usufruibili esclusivamente dagli utenti autenticati dotati dei privilegi necessari.
Le informazioni sensibili degli utenti vanno trattate in maniera sicura.
- **Disponibilità**
Il sistema dovrà essere online, raggiungibile e in funzionamento in qualsiasi momento.

1.2.3. Costo

- **Costi**

Il sistema non dovrà utilizzare componenti richiedenti una licenza a pagamento. L'intero sistema risiederà su un host con modello di pagamento *pay as you go*.

1.2.4. Manutenzione

- **Portabilità**

Il sistema dovrà essere dispiegabile su qualsiasi application server che implementi le specifiche *Java EE 8* con il minor numero di modifiche richieste per la riconfigurazione.

- **Estendibilità**

Il sistema dovrà supportare una facile implementazione di future funzionalità.

1.2.5. Utenza finale

- **Usabilità**

Il sistema dovrà essere intuitivo nell'utilizzo e presentare un'interfaccia semplice ed attraente che non risulti sopraffacente all'utente; anche le categorie di utenza meno esperte dovranno essere in grado di usufruire del sistema.

L'interfaccia dovrà guidare l'utente in maniera intuitiva e richiedere da questo un numero di passi possibili particolarmente ridotto per portare a termine qualsiasi azione.

Il sito dovrà essere in lingua inglese.

Il sistema dovrà supportare un'interfaccia utente fluida sia per dispositivi desktop che dispositivi mobile.

1.3. Panoramica

Il documento è composto di tre capitoli.

Nel primo capitolo sono introdotti gli obiettivi di progettazione del sistema.

Nel secondo capitolo viene illustrata l'architettura del sistema proposto. Più dettagliatamente, vengono descritti:

- la decomposizione del sistema in sottosistemi di funzionalità.
- la mappatura hardware/software, riguardante la scelta della configurazione hardware del sistema, le componenti "off-the-shelf", la comunicazione tra le componenti del sistema.
- la gestione dei dati persistenti memorizzati dal sistema e l'infrastruttura di gestione richiesta per memorizzarli e gestirli.
- la sicurezza e il controllo degli accessi basati sui privilegi di cui dispongono gli utenti.
- il controllo del flusso globale, una panoramica sulla sequenza di operazioni svolte sul sistema.
- condizioni limite, ovvero avvio, spegnimento e fallimento del sistema.

Infine, nel terzo capitolo vengono illustrati i diversi sottosistemi che compongono il sistema e i servizi offerti da questi.

2. Architettura del sistema proposto

2.1. Panoramica

La piattaforma consisterà di una web application con stile architetturale **Model-View-Controller (MVC)**. Le architetture MVC consistono di tre componenti:

- il **model** si occupa di gestire i dati dell'applicazione e riceve input dal controller.
- per **view** si intende la presentazione grafica del modello in un particolare formato.
- il **controller** riceve input dai client (normalmente attraverso la view), che viene eventualmente convalidato e poi trasferito al model.

L'application server su cui verrà dispiegata la piattaforma dovrà inoltre essere in grado di comunicare con i nodi client e con la base di dati non risiedente sulla stessa macchina.

2.1.1. Vantaggi dell'architettura MVC

- **Sviluppo simultaneo:** diversi sviluppatori possono lavorare contemporaneamente su model, view e controller.
- **Alta coesione:** l'architettura MVC consente il raggruppamento logico di azioni correlate su uno stesso controller. Anche le view per uno specifico model vengono raggruppate insieme.
- **Basso accoppiamento:** per natura dell'architettura MVC si ha basso accoppiamento tra model, view e controller.
- **Agevolazione alle modifiche:** data la separazione delle responsabilità tra model, view e controller, la manutenzione e gli sviluppi futuri vengono agevolati.
- **Più viste per model:** model distinti possono avere più viste.

2.1.2. Svantaggi dell'architettura MVC

- **Navigabilità del codice:** la navigazione dell'architettura può risultare complicata data i nuovi strati di astrazione introdotti.
- **Consistenza tra multipli artefatti:** decomporre una caratteristica in tre tre artefatti può causare dispersione.
- **Alta curva di apprendimento:** gli sviluppatori che lavorano con architetture MVC devono essere abili in diverse tecnologie.

2.2. Decomposizione in sottosistemi

Trattandosi di una web application, l'intero sistema è logicamente separato in tre strati:

- Il **presentation layer** consiste dell'interfaccia tramite la quale i client interagiranno con il sistema, o più precisamente con l'**application logic layer**.
- L'**application logic layer** si occuperà, come il nome suggerisce, della logica dell'applicazione; questo dovrà quindi implementare tutte le funzionalità necessarie individuate in seguito ad analisi dei requisiti (quindi le funzionalità relative al *catalogo*, agli *utenti*, alle *recensioni*, *amministrative*). L'**application logic layer** non dovrà conoscere i client che usufruiranno del sistema, ma dovrà essere in grado di ricevere richieste da questi e rispondere di conseguenza; dovrà inoltre comunicare con lo **storage layer** per la gestione dei dati persistenti.
- Lo **storage layer** si occuperà della gestione dei dati persistenti e dovrà poter essere adoperato tramite **application logic layer** (o da eventuali applicazioni future che andranno ad espandere il sistema).

2.3. Mappatura hardware/software

Il sistema usufruirà di un'architettura client/server multistrato composta di tre strati logici: **presentation layer**, **application logic layer**, **storage layer**. Trattandosi di una web application, il *presentation layer* risiederà sull'application server, ma sarà ultimamente in esecuzione lato client tramite qualsiasi web browser supportante *HTML5*, *CSS3* e *ECMAScript 6*, sia su dispositivi desktop che mobile. Questo strato comunicherà con l'*application logic layer*, implementato sull'application server su cui verrà dispiegata la piattaforma, tramite protocollo *Hypertext Transfer Protocol Secure (HTTPS)*.

L'application server utilizzato sarà un'istanza di *Apache Tomee*, in esecuzione su cluster scalabili messi a disposizione dal servizio di containerizzazione *Amazon Elastic Container Service (Amazon ECS)*. Lo *storage layer* si occuperà della gestione dei dati persistenti tramite un'istanza del Relational Database Management System (RDBMS) *MySQL*, in esecuzione tramite il servizio *Amazon Relational Database Service (Amazon RDS)*. La comunicazione tra application server e lo strato di persistenza avverrà tramite l'utilizzo di un connector *Java Database Connectivity (JDBC)*.

Essendo ex novo, il sistema non dipenderà da componenti legacy.

Per quanto riguarda il catalogo della piattaforma, il sistema si affiderà all'API del servizio di streaming di contenuto musicale *Deezer* per ottenere informazioni su artisti e progetti musicali, e per permettere agli utenti del sistema di effettuare ricerche sul catalogo.

2.4. Gestione dei dati persistenti

Per la gestione dei dati persistenti si utilizzerà una base di dati sotto il sistema di gestione di basi di dati relazionali MySQL, di proprietà di Oracle. MySQL è un RDBMS gratuito ed open source, non richiedente licenza per scopi commerciali. MySQL è ampiamente utilizzato in produzione in numerose applicazioni, in particolare web application, come WordPress, Drupal, Joomla, Facebook e Twitter. È noto per rispettare la consistenza dei suoi dati, anche grazie all'utilizzo di transazioni, per la sua scalabilità, per le alte prestazioni fornite, e, più generalmente, per essere particolarmente robusto ed efficiente in tutti gli ambiti principali che riguardano un sistema di gestione di basi di dati relazionali.

L'application logic layer interagirà con lo storage layer tramite l'implementazione di Tomee di *Java Persistence API (JPA)*, *OpenJPA*. JPA è ampiamente usato per il suo alto livello di astrazione dalle basi di dati relazionali, essendo in grado di effettuare la mappatura da classi Java a entità relazionali preservando i concetti fondamentali della programmazione orientata agli oggetti, come l'ereditarietà, e permettendo di stabilire relazioni tra classi come in un modello relazionale; in seguito a questa mappatura, JPA è in grado di effettuare implicitamente la conversione da oggetti a entità relazionali.

Per garantire consistenza tra i dati, JPA supporta le transazioni, che possono essere container o application managed (nel primo caso implicite, nel secondo caso esplicite, ovvero dichiarate dal programmatore).

Per effettuare operazioni CRUD sulle basi di dati sottostanti, JPA fornisce strumenti come *Java Persistence Query Language (JPQL)* e la *Criteria API* (quest'ultima particolarmente efficace con la *Metamodel API*), che permettono essenzialmente di scrivere query "orientate agli oggetti". Queste vengono poi tradotte in normali query SQL ed eseguite tramite JDBC sui DBMS sottostanti.


La base di dati del sistema sarà composta dalle seguenti entità relazionali:




- **user**: rappresenta un account utente registrato al sistema e contiene informazioni su questo come nome utente, indirizzo e-mail e ruolo.
- **review**: rappresenta una recensione pubblicata per un progetto musicale; composta da contenuto testuale e un voto intero, contiene riferimenti all'utente che ha pubblicato la recensione e all'identificativo del progetto recensito.
- **vote**: rappresenta un voto assegnato da un utente ad una recensione; contiene il valore del voto che può essere positivo o negativo, e riferimenti all'utente votatore e alla recensione votata.
- **report**: rappresenta una segnalazione di un utente ad una recensione; contiene riferimenti all'utente segnalatore e alla recensione segnalata.
- **backlog entry**: rappresentata una voce nella lista d'ascolto di un utente registrato alla piattaforma; contiene riferimenti all'utente a cui appartiene la lista d'ascolto e all'identificativo del progetto musicale inserito in lista.





Sono assenti entità che rappresentano progetti musicali e artisti in quanto le informazioni su queste vengono fornite esternamente dall'API di Deezer citata precedentemente.






2.4.1. Struttura delle entità relazionali




Di seguito sono riportate le strutture delle entità relazionali, comunemente denominate *tabelle* nel gergo dei RDBMS, che compongono lo schema della base di dati utilizzato dal sistema.

user		
Campo	Vincoli	Tipo
username	 PRIMARY KEY	VARCHAR(36)
email	UNIQUE; NOT NULL	VARCHAR(255)
password	NOT NULL	CHAR(60)
role	NOT NULL; DEFAULT('USER')	VARCHAR(20)
signUpDate	NOT NULL; DEFAULT now()	DATETIME
pictureUrl		VARCHAR(255)

review		
Campo	Vincoli	Tipo
reviewerUsername	 PRIMARY KEY;  FOREIGN KEY REFERENCES user(username)	VARCHAR(36)
reviewedAlbumId	 PRIMARY KEY	BIGINT
content	NOT NULL	VARCHAR(5000)
rating	UNSIGNED; NOT NULL; CHECK (rating >= 1 AND rating <= 10)	INT
publicationDate	NOT NULL; DEFAULT now()	DATETIME

vote		
Campo	Vincoli	Tipo
voterUsername	 PRIMARY KEY	VARCHAR(36)
reviewerUsername	 PRIMARY KEY;  FOREIGN KEY REFERENCES review(reviewerUse rname)	VARCHAR(36)
reviewedAlbumId	 PRIMARY KEY; FOREIGN KEY REFERENCES review(reviewedAlb umId)	BIGINT
value	NOT NULL; CHECK (value = -1 OR value = +1)	INT

report		
Campo	Vincoli	Tipo
reporterUsername	 PRIMARY KEY	VARCHAR(36)
reviewerUsername	 PRIMARY KEY;  FOREIGN KEY REFERENCES review(reviewerUse rname)	VARCHAR(36)
reviewedAlbumId	 PRIMARY KEY;  FOREIGN KEY REFERENCES review(reviewedAlb umId)	BIGINT

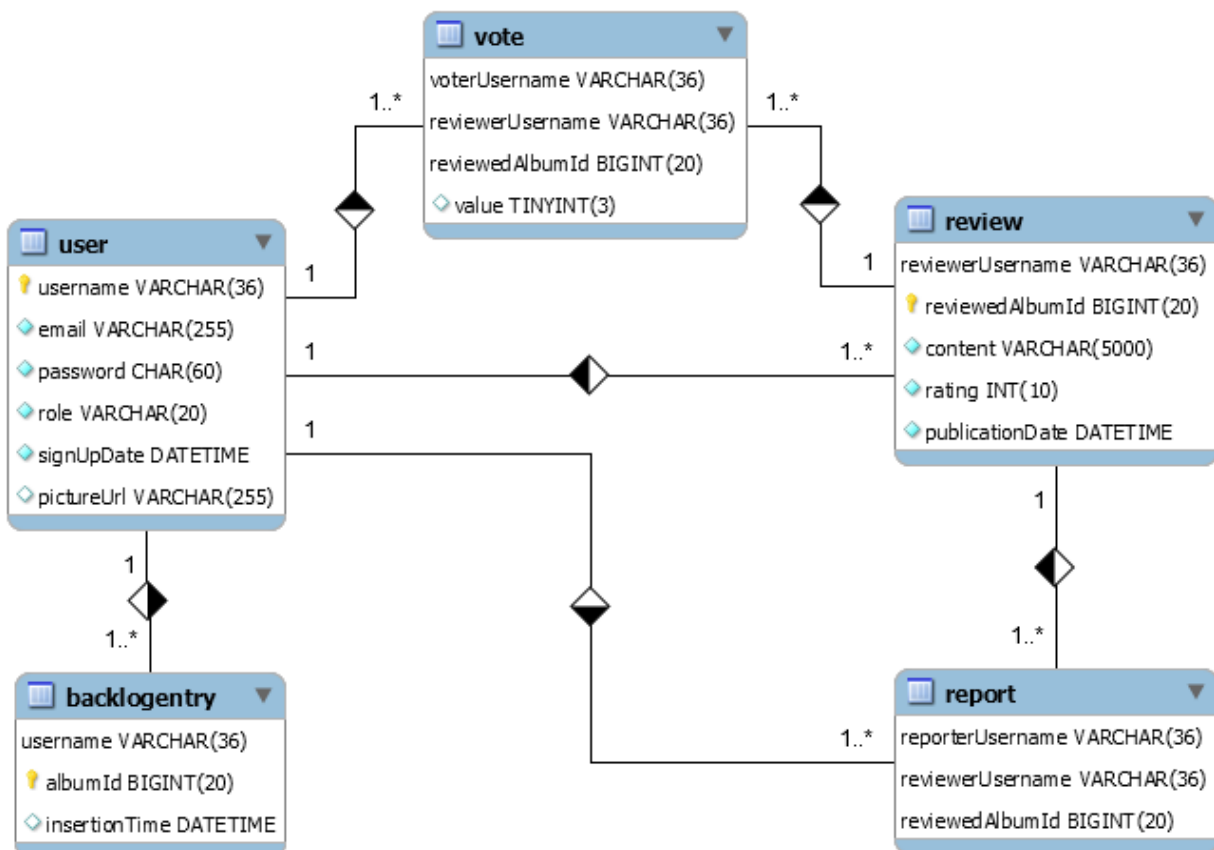
backlogEntry		
Campo	Vincoli	Tipo
username	 PRIMARY KEY;  FOREIGN KEY REFERENCES user(username)	VARCHAR(36)
albumId	 PRIMARY KEY	BIGINT
insertionTime	NOT NULL; DEFAULT now()	DATETIME

2.4.2. Schema logico

Segue il diagramma *Enhanced Entity-Relationship (EER)*, generato utilizzando lo strumento per progettazione di basi di dati per DBMS MySQL *MySQL Workbench*, della base di dati utilizzata dal sistema, descrivente le relazioni tra le entità che compongono lo schema e i loro attributi.

Legenda:

- I campi contrassegnati da un rombo pieno sono non facoltativi, ovvero devono sempre contenere un valore non nullo.
- I campi contrassegnati da un rombo vuoto sono facoltativi e possono contenere un valore nullo.
- I campi contrassegnati da una chiave sono attributi o fanno parte di una combinazione di attributi che formano la chiave primaria per l'entità di cui fanno parte.
- I campi non contrassegnati da alcun simbolo sono attributi o fanno parte di una combinazione di attributi che formano la chiave primaria per l'entità di cui fanno parte e sono chiavi esterne, ovvero fanno riferimento a chiavi primarie di altre entità.



2.5. Sicurezza e controllo degli accessi

Le informazioni sensibili degli utenti vanno trattate in maniera sicura; in particolare, per la memorizzazione delle password degli utenti sulla base di dati, queste passeranno per la funzione di password hashing irreversibile *bcrypt*, il cui risultato sarà ultimamente persistito. Una funzione di hashing irreversibile prende come argomento una stringa da cui ne viene costruita una nuova di lunghezza fissa, restituita come risultato, attraverso la quale non è possibile risalire alla stringa originaria. Le funzioni di password hashing sono idempotenti, ovvero per lo stesso input producono sempre lo stesso risultato, il che ha degli effetti collaterali indesiderati: ad esempio, se due utenti utilizzano la stessa password, questa viene memorizzata sulla base di dati utilizzando la stessa sequenza prodotta dalla funzione di hashing; ciò presenta dei problemi per la sicurezza perché, nell'eventualità in cui un utente malintenzionato dovesse riuscire ad accedere ai record degli utenti memorizzati sulla base di dati, questo si accorgerebbe di più sequenze uguali nei campi delle password degli utenti, e potrebbe di conseguenza trarre la conclusione che le password utilizzate da questi siano sequenze notoriamente insicure (o, in altre parole, brutti esempi di password come *password123* o simili) ed eventualmente ottenere l'accesso a più account. Per aggirare questo problema vengono introdotti i *salt*, ovvero ulteriori sequenze di caratteri, solitamente generati casualmente, che vengono combinate alla password dell'utente prima di passarla per la funzione di hashing; ciò ha l'effetto di garantire che, anche nel caso in cui più utenti dovessero utilizzare la stessa password, le sequenze memorizzate sulla base di dati siano teoricamente tutte distinte tra loro. Esiste la possibilità che la funzione di hashing generi la stessa sequenza per password diverse, o che dei salt generati casualmente possano essere identici per la stessa password, annullando quindi il vantaggio dell'introduzione dei salt per il gruppo di password affetto, ma queste sono così basse da risultare trascurabili.

La comunicazione tra i client e l'application server avverrà tramite il protocollo di comunicazione sicuro *HyperText Transfer Protocol Secure (HTTPS)*, per garantire che il trasferimento di informazioni tra le due componenti non sia intercettata (cruciale, ad esempio, durante il processo di autenticazione o registrazione in quanto la password viene trasferire dal client al server in *plain text*).

Per poter usufruire della maggior parte delle funzionalità offerte dal sistema, gli utenti dovranno autenticarsi. L'autenticazione avviene fornendo al sistema la combinazione di nome utente e password associate all'account a cui ci si vuole autenticare. Gli utenti devono essere in grado di poter ripristinare l'accesso al proprio account reimpostando la password tramite un meccanismo di recupero password, fornendo al sistema l'indirizzo e-mail associato al proprio account utente. Gli utenti, se autenticati, dovranno essere in grado di aggiornare sia l'indirizzo e-mail collegato al proprio account, sia la password.

Il sistema prevede l'esistenza di diverse categorie di utenza, con privilegi differenti. Segue la matrice degli accessi che descrive quali categorie di utenza possono accedere a quali funzionalità offerte dal sistema.

Gruppo utenza / Funzionalità	Utente non autenticato	Utente autenticato	Moderatore	Amministratore
Account utente	<ul style="list-style-type: none"> ✓ Sign up ✓ Log in ✗ Log out ✓ Recupero credenziali ✗ Aggiornamento credenziali ✓ Visualizzazione e profilo utente 	<ul style="list-style-type: none"> ✗ Sign up ✗ Log in ✓ Log out ✗ Recupero credenziali ✓ Aggiornamento o credenziali ✓ Visualizzazione e profilo utente 	<ul style="list-style-type: none"> ✗ Sign up ✗ Log in ✓ Log out ✗ Recupero credenziali ✓ Aggiornamento o credenziali ✓ Visualizzazione e profilo utente 	<ul style="list-style-type: none"> ✗ Sign up ✗ Log in ✓ Log out ✗ Recupero credenziali ✓ Aggiornamento o credenziali ✓ Visualizzazione e profilo utente
Recensioni	<ul style="list-style-type: none"> ✗ Pubblicazione recensione ✗ Aggiornamento recensione ✗ Rimozione recensione ✗ Votazione recensione ✓ Visualizzazione e recensione ✗ Segnalazione recensione 	<ul style="list-style-type: none"> ✓ Pubblicazione recensione ✓ Aggiornamento o recensione ✓ Rimozione recensione ✓ Votazione recensione ✓ Visualizzazione e recensione ✓ Segnalazione recensione 	<ul style="list-style-type: none"> ✓ Pubblicazione recensione ✓ Aggiornamento o recensione ✓ Rimozione recensione ✓ Votazione recensione ✓ Visualizzazione e recensione ✓ Segnalazione recensione 	<ul style="list-style-type: none"> ✓ Pubblicazione recensione ✓ Aggiornamento o recensione ✓ Rimozione recensione ✓ Votazione recensione ✓ Visualizzazione e recensione ✓ Segnalazione recensione
Catalogo	<ul style="list-style-type: none"> ✓ Ricerca catalogo ✓ Visualizzazione e progetto musicale ✓ Visualizzazione e artista ✗ Inserimento progetto in lista d'ascolto ✗ Rimozione progetto da lista d'ascolto ✗ Visualizzazione lista d'ascolto 	<ul style="list-style-type: none"> ✓ Ricerca catalogo ✓ Visualizzazione e progetto musicale ✓ Visualizzazione e artista ✓ Inserimento progetto in lista d'ascolto ✓ Rimozione progetto da lista d'ascolto ✓ Visualizzazione e lista d'ascolto 	<ul style="list-style-type: none"> ✓ Ricerca catalogo ✓ Visualizzazione e progetto musicale ✓ Visualizzazione e artista ✓ Inserimento progetto in lista d'ascolto ✓ Rimozione progetto da lista d'ascolto ✓ Visualizzazione e lista d'ascolto 	<ul style="list-style-type: none"> ✓ Ricerca catalogo ✓ Visualizzazione e progetto musicale ✓ Visualizzazione e artista ✓ Inserimento progetto in lista d'ascolto ✓ Rimozione progetto da lista d'ascolto ✓ Visualizzazione e lista d'ascolto
Amministrative	<ul style="list-style-type: none"> ✗ Aggiornamento privilegi utenti ✗ Rimozione recensioni altrui ✗ Visualizzazione segnalazioni ✗ Rimozione 	<ul style="list-style-type: none"> ✗ Aggiornamento privilegi utenti ✗ Rimozione recensioni altrui ✗ Visualizzazione segnalazioni ✗ Rimozione 	<ul style="list-style-type: none"> ✗ Aggiornamento privilegi utenti ✓ Rimozione recensioni altrui ✓ Visualizzazione e segnalazioni ✓ Rimozione 	<ul style="list-style-type: none"> ✓ Aggiornamento o privilegi utenti ✓ Rimozione recensioni altrui ✓ Visualizzazione e segnalazioni ✓ Rimozione

	segnalazione	segnalazione	segnalazione	segnalazione
--	--------------	--------------	--------------	--------------

2.6. Controllo del flusso globale del sistema

I client effettuano richieste HTTP all'application server, che le re-indirizza sulle apposite servlet che si occupano di fornire le view; queste le elaborano e rispondono ai client con le view richieste, generate dinamicamente tramite pagine JSP; tramite appositi elementi delle view, i client effettuano ulteriori richieste HTTP asincrone all'application server, che nuovamente le smista alle apposite servlet di controllo che le elaborano e rispondono di conseguenza; nell'elaborazione delle richieste, l'application server interagisce continuamente con la base di dati sottostante, interrogandola o aggiornandone il contenuto.

2.7. Condizioni limite

Le condizioni limite descrivono l'avvio, lo spegnimento e i comportamenti errati del sistema.

2.7.1. Avvio del sistema

Il sistema durante l'avvio si assicura che il sistema di gestione di basi di dati sia in esecuzione, e in caso contrario la avvia, dopodiché inizializza delle risorse interne come pool di sessioni e connessioni alla base di dati. Una volta effettuato l'avvio, il sistema è pronto a ricevere richieste e a fornire i servizi offerti ai client interessati.

2.7.2. Spegnimento del sistema

Allo spegnimento, il sistema smette di ricevere richieste, completa o tenta di terminare le operazioni in corso, e dealloca tutte le risorse occupate in fase di inizializzazione e durante l'esecuzione del sistema. Infine, viene terminato il sistema di gestione di basi di dati.

2.7.3. Fallimento del sistema

L'architettura del sistema utilizza servizi forniti da *Amazon Web Service (AWS)*, che si occupano dell'infrastruttura del sistema e della sua manutenzione automaticamente; gli errori hardware risultano quindi improbabili, e sono gestiti automaticamente dai servizi stessi nelle rare occorrenza.

Nel caso di errore software, il sistema non dovrà arrestarsi ma limitarsi a segnalare l'eccezione avvenuta, e continuare nel suo normale funzionamento in caso di evento non catastrofico.

I dati persistenti sono nuovamente gestiti da servizi offerti da AWS, quindi la consistenza dei dati, backup, e altre operazioni di manutenzione vengono assicurate dal servizio.