**Università degli Studi di Salerno**

Corso di Ingegneria del Software

**POORIFY**

System Design Document

Versione 0.3 / 22.11.2022





|  |  |
| --- | --- |
| Anna Linda Brenga | 0512109669 |
| Alessandro Farina | 0512109741 |
| Christian Mascolo | 0512111139 |
| Jessica Zampetti | 0512109894 |

**INDICE**

[1. INTRODUZIONE 3](#_Toc120005952)

[1.1 SCOPO DEL SISTEMA 3](#_Toc120005953)

[1.2 OBIETTIVI DI DESIGN 4](#_Toc120005954)

[1.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI 5](#_Toc120005955)

[1.4 RIFERIMENTI 5](#_Toc120005956)

[1.5 OVERVIEW 5](#_Toc120005957)

[2. SISTEMA ATTUALE 6](#_Toc120005958)

[3. SISTEMA PROPOSTO 7](#_Toc120005959)

[3.1 PANORAMICA 7](#_Toc120005960)

[3.2 SUBSYSTEM DECOMPOSITION 8](#_Toc120005961)

[3.3 HARDWARE/SOFTWARE MAPPING 9](#_Toc120005962)

[3.4 PERSISTENT DATA MANAGEMENT 10](#_Toc120005963)

[3.5 ACCESS CONTROL AND SECURITY 11](#_Toc120005964)

[3.6 GLOBAL SOFTWARE CONTROL 12](#_Toc120005965)

[3.7 BOUNDARY CONDITIONS 13](#_Toc120005966)

[4. SUBSYSTEM SERVICES 15](#_Toc120005967)

# **1. INTRODUZIONE**

## **SCOPO DEL SISTEMA**

Poorify è una piattaforma di streaming musicale che si pone come obiettivo quello di semplificare la fruizione di opere musicali attraverso un sistema gratuito e user-friendly.

In particolare, Poorify punta a fornire agli utenti uno strumento che permetta loro di riprodurre liberamente brani presenti, creare playlist e seguire i profili degli iscritti e agli artisti una piattaforma dove fare l’upload delle loro opere al fine di accrescere la propria fama

## **OBIETTIVI DI DESIGN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Label** | **Criteria Group** | **Design Criterion** | **Description** | **Origin** |
| DG1 | Performance | Memory | In base alle stime effettuate il server utilizzato deve avere una capacità di 250GB. | RNF4, 5, 6 & 7 |
| DG2 | Dependability | Robustness | Il sistema deve notificare l’utente in caso di input errato e gestire propriamente le possibili eccezioni. | Dominio Applicativo |
| DG3 | Dependability | Reliability | L’output delle operazioni deve coincidere con quanto specificato negli Use Cases e quindi con le aspettative degli utenti. | Dominio Applicativo |
| DG4 | Dependability | Security | Il sistema non deve permette agli utenti di accedere a pagine per cui non possiedono l’autorizzazione tramite dei filtri. | RNF2 |
| DG5 | Maintenance | Extensibility | Il sistema deve facilitare la futura aggiunta di nuove funzionalità. | Dominio Applicativo |
| DG6 | Maintenance | Readability | Il codice deve essere modulare e commentato in modo da semplificarne la lettura e la comprensione. | RNF8 |
| DG7 | End User | Usability | Il sistema deve notificare l’utente circa l’esito delle operazioni in caso di errore con degli opportuni messaggi. | RNF1 |

|  |  |
| --- | --- |
| **Trade-Off** | **Rational** |
| Space vs Speed | Il sistema deve garantire l’accesso a una grande quantità di contenuti, questo potrebbe causare dei rallentamenti durante la ricerca o la fruizione degli stessi. |
| Delivery Time vs Functionality | È richiesto di sviluppare l’intero sistema proposto, implementando tutti gli Use Cases, entro i tempi previsti per le deadlines. |
| Delivery Time vs Usability | Per vincoli di tempo si è deciso di non rendere la grafica piattaforma responsive. |

## **1.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI**

* Piattaforma: Applicazione Web (Poorify)
* RAD: Requirements Analysis Document
* UCBC: Use Case Boundary Condition
* RNF: Requisito Non Funzionale
* DG: Design Goal

## **1.4 RIFERIMENTI**

Nel corso del documento facciamo riferimento ai Requisiti Non Funzionali, agli Use Cases e al Class Diagram presenti nel documento RAD.

## **1.5 OVERVIEW**

Nei paragrafi successivi presentiamo:

* L’analisi dell’architettura di una piattaforma concorrente a Poorify.
* La decomposizione del sistema in sottosistemi.
* La strategia di deployment su hardware.
* La strategia adottata per la gestione dei dati persistenti.
* Le varie Boundary Condition del sistema.
* I servizi offerti dai vari sottosistemi.

# 

# **2. SISTEMA ATTUALE**

Il principale concorrente di Poorify è il noto servizio di streaming Spotify. La piattaforma del colosso svedese si basa fondamentalmente su un’architettura orientata ai servizi con un backend scritto prevalentemente in Python e Java. I messaggi vengono scambiati tramite protocollo Hermes, che sostituisce la più classica soluzione enterprise di HTTP e XML. Il principale strumento di persistenza impiegato è PostgreSQL, un DBMS a oggetti, mentre le tracce audio sono salvate su server Amazon S3. La piattaforma sfrutta inoltre machine learning e IA per fornire contenuti personalizzati in base ai gusti dell’utente.

# **3. SISTEMA PROPOSTO**

## **3.1 PANORAMICA**

Poorify è una piattaforma browser distribuita.

Nei paragrafi che seguono illustriamo l’architettura di massima dell’intero sistema. Elenchiamo brevemente le componenti principali dell’applicazione:

* Un front-end web dinamico sviluppato con HTML, JS e CSS.
* Un back-end per la logica di business basato su Serlvet Java.
* Un insieme di classi DAO che collegano la logica alla persistenza tramite JDBC.
* Un database relazionale MySQL e un File System per la persistenza dei dati.
* Una serie di terminali Python per le operazioni esterne dell’amministratore.

## **3.2 SUBSYSTEM DECOMPOSITION**

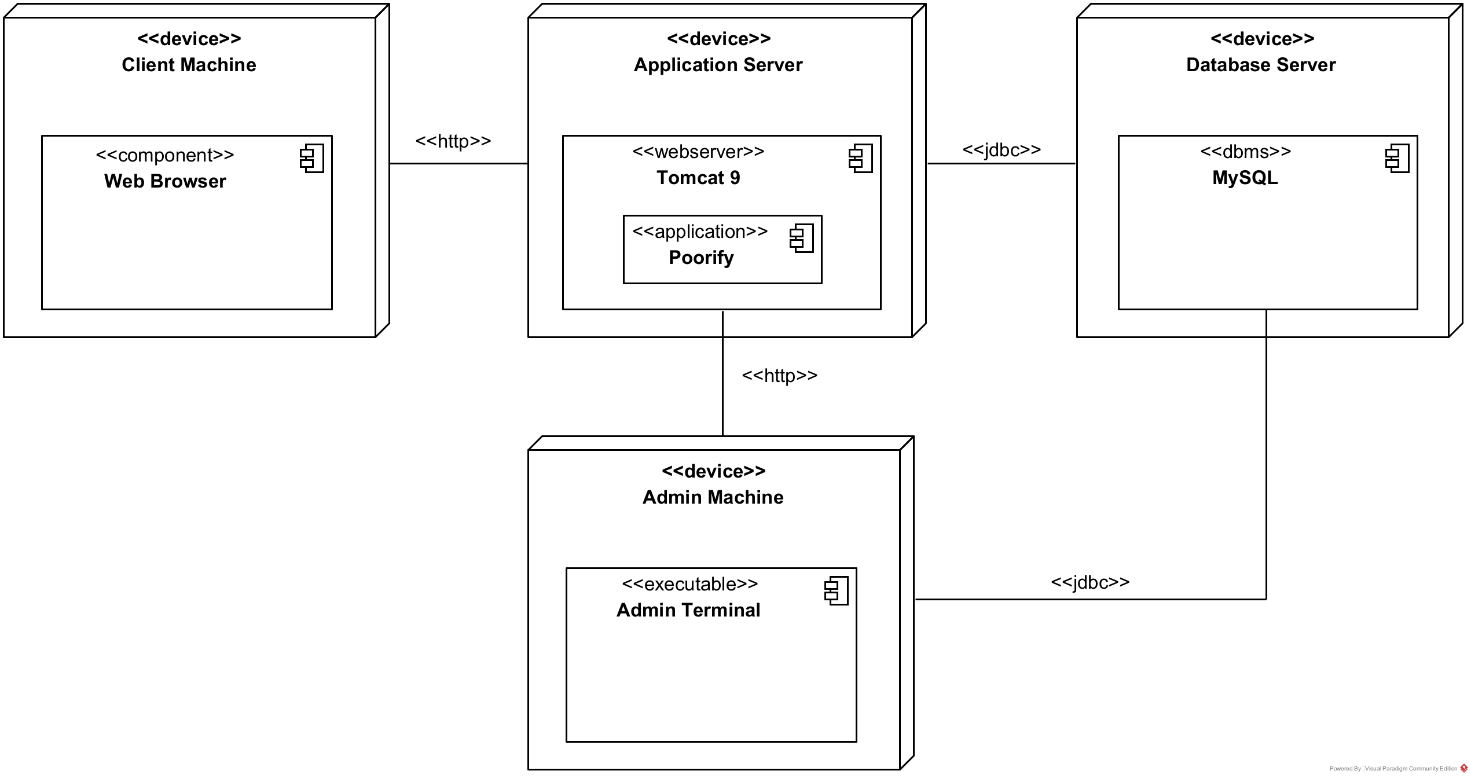
La piattaforma Poorify sfrutta una architettura web three-tier:

* **Presentation layer**: Interfaccia impiegata dagli utenti finali per interagire con la piattaforma tramite web browser dalle loro macchine client.
* **Logic layer**: La logica di business che gestisce le operazioni del sistema. Dividiamo tale strato in cinque sottosistemi/partizioni:
* *Gestione Utente*: per la registrazione, l’autenticazione, la modifica e l’eliminazione del profilo dei fruitori.
* *Gestione Playlist*:per la creazione, l’eliminazione, aggiunta/rimozione di brani e la modifica delle playlist (inclusa la possibilità di rendere la playlist pubblica/privata e singola/collaborativa).
* *Gestione Album*: per l’upload e l’eliminazione degli album.
* *Gestione Navigazione*: per la ricerca dei contenuti e la navigazione tra le varie pagine della piattaforma.
* *Gestione Riproduzione*: per la riproduzione dei brani e la gestione della coda di ascolti.
* **Data layer**: Lo strato software che permette di effettuare le operazioni su database necessarie al funzionamento della piattaforma.

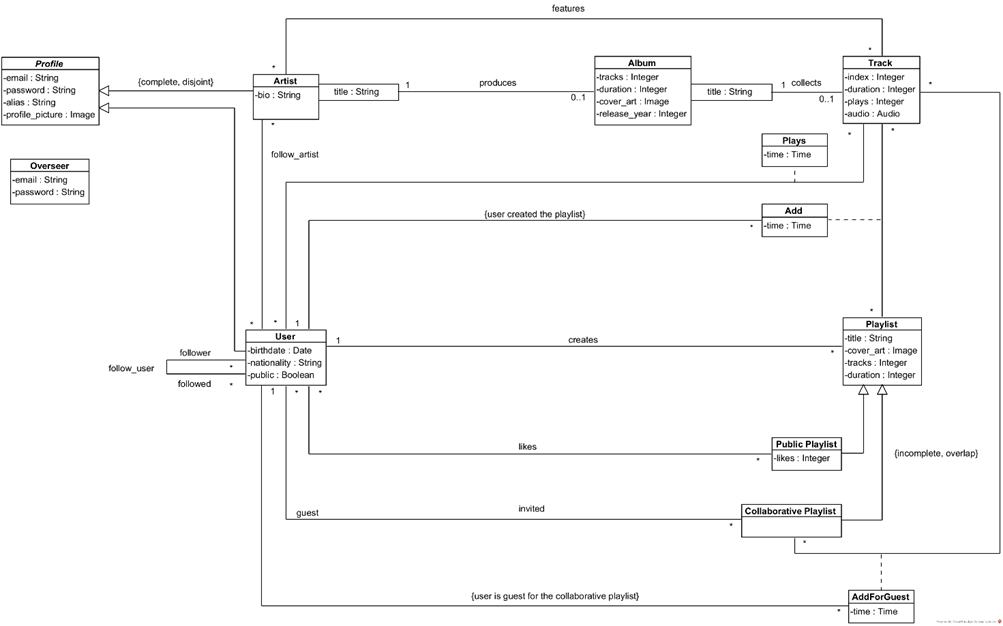
## **3.3 HARDWARE/SOFTWARE MAPPING**

Gli utenti accedono alla piattaforma Poorify tramite web browser dalla propria macchina client stabilendo una connessione http con un server. L’applicazione viene eseguita nel web container Tomcat che interagisce con un database relazionale MySQL, potenzialmente distribuito su molteplici macchine server.

Gli amministratori del sistema possono interagire sia con Tomcat per effettuare startup e shutdown del sistema, sia operare direttamente su database tramite un terminale dedicato per aggiungere nuovi impiegati supervisori o lanciare delle query per l’aggiornamento o la creazione di contenuti.



## **3.4 PERSISTENT DATA MANAGEMENT**



Facciamo riferimento al class diagram per selezionare quali attributi memorizzare su database relazionale e per quali invece impiegare il file system. In linea generale preferiamo il file system per dati voluminosi, come immagini e tracce audio.

I file multimediali sono acceduti tramite un path relativo a partire da una cartella di default all’interno dell’applicazione, è quindi possibile far migrare il sistema su altre macchine server senza dover cambiare tali riferimenti.

La root del nostro file system presenta le seguenti sottocartelle:

* **/Profile**: per *profile\_picture* della classe *Profile*, e di conseguenza *User* e *Artist*.
* **/Album**: per *cover\_art* della classe *Album*.
* **/Playlist**: per *cover\_art* della classe *Playlist*.
* **/Track**: per *audio* della classe *Track*.

I file salvati sono denominati *x.estensione* dove x è l’id associato alla entry corrispondente nel database. Ad esempio *profile\_picture* dell’utente/artista con id 27 è salvata come 27.png nella sottocartella */Profile* mentre l’attributo *audio* del brano con id 45 è salvato come 45.mp3 nella sottocartella */Track*.

Le restanti informazioni presenti nel class diagram, essendo meno complesse e perlopiù testuali o numeriche, sono rese persistenti su database.

## **3.5 ACCESS CONTROL AND SECURITY**

Illustriamo nella seguente tabella le operazioni che ogni tipologia di utente può eseguire all’interno della piattaforma.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Utente Non Registrato | Utente | Artista | Supervisore |
| Gestione Utente | Registrazione | Autenticazione  Disconnessione  Modifica Profilo  Visualizza Profilo  Eliminazione Profilo  Seguire (Smettere di seguire) Artista o Utente Pubblico | Autenticazione  Disconnessione  Modifica Profilo  Visualizza Profilo  Eliminazione Profilo | Autenticazione  Disconnessione  Eliminazione Profilo Altrui |
| Gestione Playlist |  | Creazione Playlist  Rendere Singola o Collaborativa  Rendere Pubblica o Privata  Invitare Guest per Collaborativa  Aggiungere un Brano  Eliminare un Brano  Lasciare Like  Modifica Playlist  Cambio Ordine Brani  Eliminazione Playlist |  | Eliminazione Playlist |
| Gestione Album |  |  | Upload Album  Eliminazione Album | Eliminazione Album |
| Gestione Navigazione |  | Ricerca Contenuti  Visualizza Pagina Artista  Visualizza Pagina Album  Visualizza Pagina Playlist  Visualizza Pagina Utente  Navigazione Pagine  Generazione Feed Utente | Visualizza Pagina Album  Navigazione Pagine | Ricerca Contenuti  Visualizza Pagina Artista  Visualizza Pagina Album  Visualizza Pagina Playlist  Visualizza Pagina Utente  Navigazione Pagine |
| Gestione Riproduzione |  | Riproduzione Brano  Riproduzione Playlist/Album  Aggiungere Brano alla Coda  Saltare Brano in Coda |  | Riproduzione Brano |

La piattaforma sfrutta dei filtri per garantire la sicurezza impedendo agli utenti di eseguire operazioni e accedere a pagine per cui non possiedono l’autorizzazione. Inoltre i supervisori hanno il compito di eliminare i contenuti ritenuti offensivi o inopportuni e nel caso eliminare il profilo del responsabile.

## **3.6 GLOBAL SOFTWARE CONTROL**

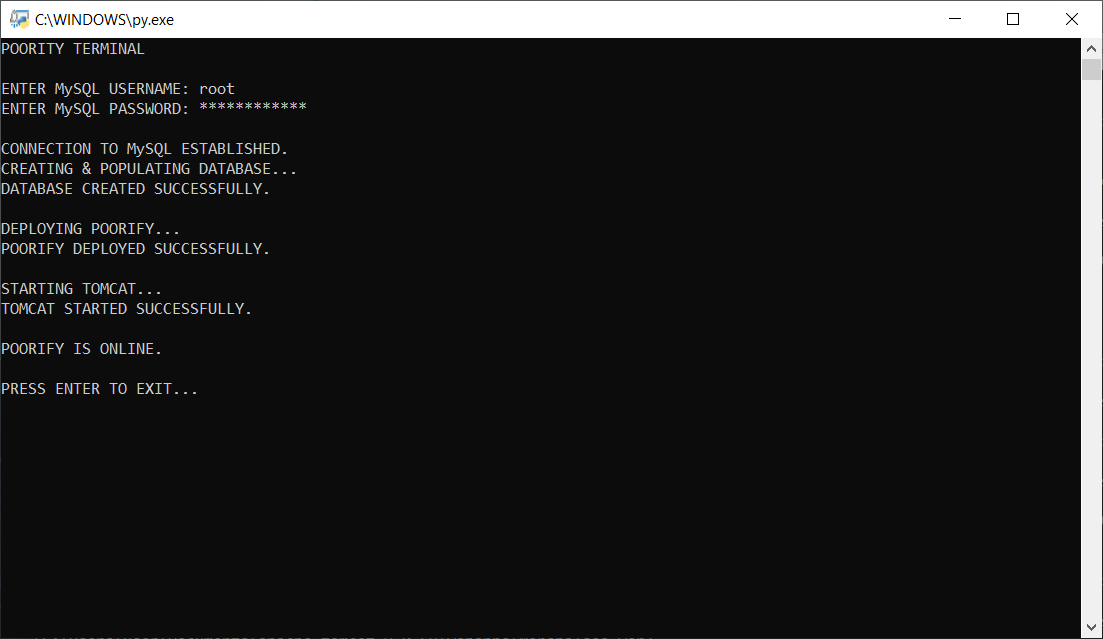
L’architettura della piattaforma Poorify è intrinsecamente Event-driven in quanto le servlet reagiscono agli eventi generati dall’input utente, ciò permette di aggiungere nuove funzionalità in maniera semplice senza dover modificare il codice già scritto. L’applicazione è anche distribuita e quindi alcune operazioni su database possono causare delle indesiderate race conditions. Rilsulta pertanto necessario gestire la concorrenza tramite thread e applicare dei meccanismi di sincronizzazione per garantire la coerenza delle risorse della piattaforma.

## **BOUNDARY CONDITIONS**

Startup e shutdown del sistema sono gestiti tramiti due eseguibili Python.

* *startup.py* effettua la configurazione del sistema. In questa fase viene creato e popolato, previo inserimento delle credenziali dell’amministratore del sistema, il database MySQL impiegato dalla piattaforma, e viene eseguito lo startup del web container Tomcat.
* *overseer.py* permette di gestire (aggiungere/cambiare/eliminare) i supervisori della piattaforma tramite alcune query predefinite.
* *wrapped.py* per generare per ciascun utente una playlist con i suoi brani più ascoltati durante l’anno trascorso.
* *charts.py* per generare delle classifiche dei brani più ascoltati in ogni nazione dove è disponibile il servizio.
* *shutdown.py* effettua lo shutdown del web container Tomcat, la piattaforma risulterà quindi offline.

Il sistema deve risultare robusto nella gestione dell’input utente, ad esempio interrompendo l’operazione sul nascere. In caso di eccezioni o fallimenti non previsti però non si potrà garantire il corretto funzionamento della piattaforma e potrebbe risultare necessario un riavvio.



Configurazione completata con successo con *startup.py*

**UCBC-1: STARTUP**

|  |  |
| --- | --- |
| **USE CASE NAME** | Startup |
| **PARTICIPATING ACTORS** | Amministratore del Sistema |
| **ENTRY CONDITION** | * L’amministratore esegue *startup.py*. |
| **FLOW OF EVENTS** | 1. L’amministratore fornisce username e password del database MySQL. 2. Il programma provvede a creare e popolare il database per applicazione tramite i file *creazione.sql* e *popolamento.sql*. 3. Il programma provvede a effettuare il deploy del file *Poorify.war* sul web container Tomcat. 4. Il programma effettua lo startup di Tomcat. |
| **EXIT CONDITION** | * La piattaforma Poorify è online. |
| **EXCEPTIONS** | * Se al punto 1 le credenziali fornite sono errate, l’amministratore viene notificato con un messaggio di errore e può ritentare l’autenticazione. (UCBC-1.1: CredenzialiStartupErrate) * Se al punto 2 startup.py non trova creazione.sql o popolamento.sql il programma termina con un messaggio di errore. (UCBC-1.2: FileMySQLNonTrovati) * Se al punto 3 startup.py non trova Poorify.war il programma termina con un messaggio di errore. (UCBC-1.3: FileWarNonTrovato) * Se al punto 4 startup.py non riesce a avviare Tomcat il programma termina con un messaggio di errore. (UCBC-1.4: TomcatNonAvviato) |

**UCBC-2: SHUTDOWN**

|  |  |
| --- | --- |
| **USE CASE NAME** | Shutdown |
| **PARTICIPATING ACTORS** | Amministratore del Sistema |
| **ENTRY CONDITION** | * L’amministratore esegue *shutdown.py*. |
| **FLOW OF EVENTS** | 1. L’amministratore conferma di voler eseguire lo shutdown del sistema. 2. Il programma effettua lo shutdown del web container Tomcat. |
| **EXIT CONDITION** | * La piattaforma Poorify è offline. |
| **EXCEPTIONS** | * Se al punto 1 l’amministratore non conferma lo shutdown, il programma termina normalmente e la piattaforma è ancora online. (UCBC-2.1: ShutdownRimandato) |

# **4. SUBSYSTEM SERVICES**

|  |  |  |
| --- | --- | --- |
| **Sottosistemi** | **Servizi** | **Operazioni** |
| Gestione Utente | Registrazione  Autenticazione  Modifica Profilo  Eliminazione Profilo  Funzioni Social | Registrazione  Autenticazione, Disconnessione  Modifica Profilo  Eliminazione Profilo  Seguire/Smettere di seguire Artisti/Utenti Pubblici |
| Gestione Playlist | Creazione Playlist  Gestione Brani  Modifica Attributi Playlist  Cambiamento Tipologia  Eliminazione Playlist  Feedback  Cambio Ordine Brani  Invito Utenti Collaborativa | Creazione Playlist  Aggiunta Brano, Rimozione Brano  Modifica Playlist  Rendere Singola/Collaborativa, Privata/Pubblica  Eliminazione Playlist  Lasciare Like, Togliere Like  Cambio Ordine Brani  Invito Utenti Collaborativa |
| Gestione Album | Upload Album  Eliminazione Album | Upload Album  Eliminazione Album |
| Gestione Navigazione | Ricerca Contenuti  Navigazione Pagine  Visualizzazione Pagine | Ricerca Contenuti  Navigazione Pagine  Visualizza Pagina Utente/Artista/Album/Playlist |
| Gestione Riproduzione | Riproduzione Brano  Riproduzione Generale  Gestione Coda di Ascolti | Riproduzione Brano  Riproduzione Playlist, Riproduzione Album  Aggiunta Brano In coda, Saltare Brano Corrente |