soundrate

Object Design Document

1. Introduzione

Dopo la stesura del documento di analisi dei requisiti e di progettazione del sistema ci si appresta a definire gli aspetti implementativi. Il documento di progettazione degli oggetti pone come obiettivo la produzione di un modello coerente con le funzionalità e le altre informazioni individuate nei documenti precedenti.

Il documento prevede la descrizione di trade-off da osservare durante lo sviluppo del sistema, linee guida per la documentazione delle interfacce e della stesura del codice, una panoramica dei design pattern utilizzati dalle classi che compongono il sistema e dei package che le contengono, e una descrizione delle interfacce dei sottosistemi individuati.

1.1. Object design trade-offs

Seguono i trade-off trovati nello sviluppo del sistema.

1.1.1. Prestazioni Vs. Costi

Il sistema fa affidamento a tecnologie open source non richiedenti licenze a pagamento per l'utilizzo di queste, tuttavia ampiamente e notoriamente diffuse, come il sistema di gestione di basi di dati relazionali *MySQL* e l'application server *Apache TomEE*. Il committente si impegna nel pagare i servizi offerti da *Amazon Web Service* (*AWS*) per l'hosting dell'application server e della base di dati, il che risulta decisamente più economico del costruire un'infrastruttura privata da dover manutenere a lungo termine, e offre ulteriori vantaggi sotto un punto di vista di prestazioni, come scalabilità automatica, replica e distribuzione trasparente di contenuto tramite nodi geografici.

1.1.2. Interfaccia Vs. Usabilità

La piattaforma **soundrate** nasce per semplificare e rendere attraenti le procedure di espressione di opinioni musicali, e di conseguenza vanta di un'interfaccia utente intuitiva e concisa con lo scopo di semplificare l'utilizzo del sistema ai suoi utenti finali.

1.1.3. Sicurezza Vs. Efficienza

A causa del tempo limitato per lo sviluppo del sistema, la sicurezza coprirà esclusivamente gli aspetti strettamente necessari, come la comunicazione sicura tra client e server, l'accesso ristretto alle funzionalità amministrative e la necessità di autenticazione per usufruire delle funzionalità chiave della piattaforma. Sarà possibile in futuro estendere il sistema implementando autenticazione a due fattori o altre tecnologie per la protezione degli account utente; tuttavia, come già descritto nella documentazione precedente, le credenziali degli utenti della piattaforma vengono memorizzate in maniera sicura, con rischio virtualmente inesistente che agenti malintenzionati possano ottenerne l'accesso.

1.2. Linee guida per la documentazione delle interfacce

Per assicurare un'alta leggibilità e di conseguenza una buona manutenibilità del codice, gli sviluppatori devono sottostare a delle linee guida per la stesura del codice.

1.2.1. Code style

Gli sviluppatori devono attenersi quanto più possibile a delle linee guida ben precise dettate dalle convenzioni di stile del codice *Google Java*. In particolare, il codice deve essere indentato utilizzando tab e non spazi, ed essere formattato seguendo uno stile conciso e comprensibile.

1.2.2. Naming convention

È buona pratica che i nomi da utilizzare siano:

- descrittivi;
- pronunciabili;
- di uso comune;
- non abbreviati (se non per variabili temporanee);
- contenenti esclusivamente caratteri consentiti (*A-Z*, *a-z*, *0-9*).

1.2.2.1. Classi

I nomi delle classi devono rispettare lo stile *upper camel case* (o *Pascal case*), dove la prima lettera di ogni parola è in maiuscolo. I nomi delle classi non dovrebbero contenere acronimi o abbreviazioni, a meno che l'abbreviazione non sia molto più diffusa della forma normale, come ad esempio URL o HTML.

Esempi:

- `class User {}`
- `class AvatarGenerator {}`

1.2.2.2. Metodi e funzioni

I nomi dei metodi e delle funzioni devono rispettare lo stile *lower camel case* (o *Dromedary case*), dove la prima lettera della prima parola è in minuscolo e la prima lettera di ogni parola successiva è in maiuscolo. I nomi di metodi e funzioni devono essere composti da verbi o da più parole che cominciano per un verbo.

Esempi:

- `run();`
- `runFast();`
- `getName();`

1.2.2.3. Variabili

Anche i nomi delle variabili locali, istanze di variabili, e variabili di classe devono rispettare lo stile *lower camel case*. I nomi delle variabili non dovrebbero iniziare per underscore (_) o per il simbolo del dollaro ($), anche se entrambi sono permessi.

È importante far sì quanto più possibile che i nomi delle variabili siano corti ma significativi; la scelta del nome di una variabile dovrebbe essere mnemonica. Nomi di variabili composte da un singolo carattere andrebbero evitati se non per variabili temporanee come contatori o variabili di buffer.

Esempi:

- `int i;`
- `char c;`
- `float averageRating;`

1.2.2.4. Costanti

I nomi delle costanti dovrebbero essere composte da parole in maiuscolo separate da underscore. I nomi delle costanti possono anche contenere cifre se appropriato, ma non come primo carattere.

Esempi:

- `static final int MAX_LENGTH = 10;`
- `const ERROR_401 = “Unauthorized”;`

2. Design pattern

Per assicurare lo sviluppo di un sistema coeso e facilmente manutenibile, si utilizzano diversi design pattern riutilizzabili per diverse componenti all'interno dell'applicazione.

2.1. Singleton

Il design pattern del *singleton* permette di restringere il numero di istanze di una classe ad una singola istanza.

2.1.1. Funzionalità

Il pattern del singleton permette la riduzione della memoria utilizzata per le classi che non hanno bisogno di mantenere uno stato o che possono usufruire di uno stato condiviso tra i suoi utilizzatori, in quanto esiste una singola istanza per ogni classe che adotta l'utilizzo del pattern.

Il design pattern del singleton viene ampiamente utilizzato per l'accesso centralizzato a funzionalità e componenti in un'unica entità condivisa dai suoi utilizzatori.

2.1.2. Utilizzo

Il sistema utilizza il design pattern del singleton per la maggior parte delle classi che non mantengono uno stato o con stato condiviso, che quindi possono essere accedute concorrentemente, come classi d'utilità, classi realizzanti i servizi che implementano le diverse funzionalità individuate (funzionalità riguardanti gli account utenti, le recensioni, il catalogo, ecc.), o classi che si occupano di mantenere in memoria i risultati delle richieste recenti sul catalogo.

2.2. Data access object (DAO)

Il design pattern del *data access object* (*DAO*) è un pattern utilizzato per fornire un'interfaccia ad una o più basi di dati o altri tipi di meccanismi di persistenza.

2.2.1. Funzionalità

Il pattern del DAO fornisce l'accesso a delle operazioni specifiche sui dati persistenti senza esporre dettagli sulla base di dati sottostante attraverso la mappatura di chiamate interne all'applicazione al layer di persistenza.

Un data access object permette di separare l'insieme di dati di cui un'applicazione richiede l'utilizzo da come gli stessi vengono trattati ad un livello più basso, permettendo all'utilizzatore di ignorare la tipologia di base di dati sottostante e le strutture dati che utilizzate.

2.2.2. Utilizzo

Il sistema adotta il pattern DAO per le classi che implementano i servizi offerti dalla piattaforma e che devono di conseguenza lavorare con dati persistenti.

2.3. Observer

Il design pattern dell'*observer* è un pattern in cui un oggetto detto *subject* notifica un insieme di *osservatori* riguardo i suoi cambi di stato o esecuzioni di azioni.

2.3.1. Funzionalità

Il pattern observer viene generalmente usato in sistemi di gestione di eventi in cui il subject viene detto "sorgente del flusso di eventi", mentre gli observer vengono denominati "pozzi di eventi". Questo pattern risulta particolarmente utile per soddisfare dipendenze tramite accoppiamento debole.

2.3.2. Utilizzo

Il pattern observer viene utilizzato dal sistema per aggiornare le sessioni degli utenti autenticati alla piattaforma se lo stato del loro account utente viene aggiornato, per mantenere le sessioni consistenti con lo stato dell'account utente.

Viene inoltre utilizzato per aggiornare il contenuto delle cache implementate dall'applicazione in maniera trasparente, senza richiedere alcuna azione esplicita dall'utilizzatore.

Il pattern observer può inoltre essere utilizzato per risolvere problemi di logica trasversale, come il logging delle operazioni svolte dal sistema.

3. Package

Trattandosi di un'applicazione web basata sulla piattaforma *Java Enterprise Edition*, questa consiste principalmente di due porzioni: una contenente classi e risorse Java, e una contenente risorse web come pagine web (in questo caso pagine e frammenti *JSP*) e ulteriore contenuto come fogli di stile CSS e file JavaScript per modellare l'aspetto dell'interfaccia e per permettere agli utenti di interagire con gli elementi che la compongono.

La prima delle due porzioni, su cui risiede la logica applicativa, consiste di due package principali, suddivisi a loro volta in subpackage:

- il package `application` contiene i subpackage che si occupano di implementare la logica applicativa, ovvero:
  - il package `entities` contiene le classi che vengono mappate alle entità della base di dati relazionale utilizzata dalla piattaforma (utente, recensione, voto, segnalazione).
  - il package `model` contiene le classi che si occupano di implementare le funzionalità offerte dal sistema (funzionalità riguardanti gli utenti, funzionalità riguardanti le recensioni, funzionalità riguardanti il catalogo della piattaforma) e un subpackage `exceptions` contenente le eccezioni lanciabili dalle classi che implementano i servizi.
  - altri subpackage di ruolo secondario che implementano la cache del sistema, classi di utilità, e componenti proprie della piattaforma *Java EE* come interceptor o event listener.
- il package `endpoints` contiene le classi che implementano gli endpoint HTTP che si occupano di soddisfare le richieste HTTP provenienti dai client, e consite dei seguenti subpackage:
  - il package `services` contiene le classi che si occupano di soddisfare le richieste degli utenti che richiedono l'interazione con i servizi offerti dal sistema (ad esempio, la registrazione di un nuovo utente, l'aggiornamento di una recensione, l'autenticazione al proprio account utente, la restituzione delle recensioni segnalate).
  - il package `dispatchers` contiene le classi che si occupano di inoltrare le richieste dei client alle pagine web della piattaforma, restituendole. Questo package è a sua volta separato nei package `fragments` e `pages`, dove le classi del primo inoltrano le richieste a frammenti di pagine (ad esempio l'header), mentre il secondo a pagine web intere, che a loro volta fanno uso di frammenti.

Le classi nel package `endpoints` si occupano quindi di rispondere direttamente alle richieste dei client, interagendo con le classi dei subpackage di ruolo primario (`entities`, `model`) presente nel package `application`; le classi del package `application` non avvia interazioni con le classi del package `endpoints`, ignorandone la presenza.

3.1. Package `application.entities`

File	Descrizione
`User.java`	Classe rappresentante un utente registrato al sistema.
`Review.java`	Classe rappresentante una recensione pubblicata da un utente registrato per un progetto musicale.
`Vote.java`	Classe rappresentante un voto assegnato da un utente registrato a una recensione
`Report.java`	Classe rappresentante una segnalazione effettuata da un utente registrato a una recensione.
`BacklogEntry.java`	Classe rappresentante un'entrata di un progetto musicale nella lista d'ascolto di un utente registrato.

3.2. Package `application.model`

File	Descrizione
`UsersAgent.java`	Classe che implementa i servizi volti a soddisfare le funzionalità riguardanti gli account utenti.
`ReviewsAgent.java`	Classe che implementa i servizi volti a soddisfare le funzionalità riguardanti le recensioni.
`CatalogAgent.java`	Classe che implementa i servizi volti a soddisfare le funzionalità riguardanti il catalogo della piattaforma.

3.3. Package `endpoints.services`

File	Descrizione
`AuthenticationService.java`	Classe implementante gli endpoint che si interfacciano alle funzionalità riguardanti l'autenticazione degli utenti (login, logout, signup).
`CatalogService.java`	Classe implementante gli endpoint che si interfacciano alle funzionalità riguardanti il catalogo della piattaforma (aggiornamento lista d'ascolto, ecc.).
`ReviewsService.java`	Classe implementante gli endpoint che si interfacciano alle funzionalità riguardanti le recensioni (pubblicazione, aggiornamento, rimozione di recensioni, ecc.).
`UsersService.java`	Classe implementante gli endpoint che si interfacciano alle funzionalità riguardanti gli account utenti (aggiornamento indirizzo e-mail, recupero password, ecc).

3.4. Package `endpoints.dispatchers.fragments`

File	Descrizione
`HeaderFragmentServlet.java`	Servlet che restituisce il frammento che compone l'header delle pagine web della piattaforma.
`SignInFragmentServlet.java`	Servlet che restituisce il frammento che compone il modal contenente i form per il log in e per il sign up degli utenti.
`UserSettingsFragmentServlet.java`	Servlet che restituisce il frammento che compone il modal contenente i form per l'aggiornamento dell'indirizzo e-mail e della password degli utenti.

3.5. Package `endpoints.dispatchers.pages`

File	Descrizione
`IndexPageServlet.java`	Servlet che restituisce la homepage della piattaforma.
`AlbumPageServlet.java`	Servlet che restituisce la pagina web per un determinato album.
`ArtistPageServlet.java`	Servlet che restituisce la pagina web per un determinato artista.
`SearchPageServlet.java`	Servlet che restituisce la pagina web contenente i risultati di una ricerca.
`ReviewPageServlet.java`	Servlet che restituisce la pagina web per una determinata recensione.
`UserPageServlet.java`	Servlet che restituisce la pagina web per un determinato utente.
`BacklogPageServlet.java`	Servlet che restituisce la pagina web contenente la lista d'ascolto di un determinato utente.
`RecoverPageServlet.java`	Servlet che restituisce la pagina web per il recupero delle proprie credenziali.
`ResetPageServlet.java`	Servlet che restituisce la pagina web per reimpostare la password associata al proprio account utente.
`ReportsPageServlet.java`	Servlet che restituisce la pagina web per moderatori per permettere la rimozione di recensioni segnalate e delle segnalazioni di queste.

4. Interfacce delle classi

Seguono le descrizioni delle interfacce delle classi implementanti le funzionalità della piattaforma (appartenenti al package `application.model`) e delle classi che permettono ai suoi utenti di poter usufruire di queste tramite richieste HTTP (appartenenti al package `endpoints.services`).

4.1. Interfacce classi in `application.model`

Le classi di questo package realizzano la logica di business del sistema e sono alla base delle funzionalità offerte dalla piattaforma.

4.1.1. Interfaccia classe `UsersAgent`

4.1.1.1. `getUsers`

Restituisce una lista degli utenti registrati al sistema.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<User> getUsers()`
- `public List<User> getUsers(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUsers()`
- `usersAgent.getUsers(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.2. `getUser`

Restituisce un utente il cui username è specificato se questo è registrato al sistema, altrimenti `null`.

Interfacce

- `public User getUser(@NotNull final String username)`

Contesto

- `usersAgent.getUser(username)`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e diverso da `null`.

4.1.1.3. `getUserByEmail`

Restituisce un utente il cui indirizzo e-mail è specificato se questo è registrato al sistema, altrimenti `null`.

Interfacce

- `public User getUserByEmail(@NotNull @Email final String username)`

Contesto

- `usersAgent.getUserByEmail(email)`

Condizioni d'entrata

- Il parametro `email` deve essere specificato e deve essere un indirizzo e-mail valido.

4.1.1.4. `createUser`

Registra uno specifico nuovo utente al sistema.

Interfacce

- `public void createUser(@NotNull final User user)`

Contesto

- `usersAgent.createUser(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- L'utente specificato viene registrato al sistema.

Eccezioni

- Un utente con lo stesso username dell'utente specificato è già presente nel sistema.
- Un utente con lo stesso indirizzo e-mail dell'utente specificato è già presente nel sistema.
- L'oggetto `user` non rispetta i criteri di validazione.

4.1.1.5. `updateUser`

Aggiorna le informazioni dell'utente specificato.

Interfacce

- `public void updateUser(@NotNull final User user)`

Contesto

- `usersAgent.updateUser(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Le informazioni dell'utente specificato vengono aggiornate.

Eccezioni

- L'utente specificato non è registrato al sistema.
- Un altro utente con lo stesso indirizzo e-mail dell'utente specificato è già presente nel sistema.
- L'oggetto `user` non rispetta i criteri di validazione.

4.1.1.6. `deleteUser`

Rimuove uno specifico utente dal sistema.

Interfacce

- `public void deleteUser(@NotNull User user)`

Contesto

- `usersAgent.deleteUser(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- L'utente viene rimosso dal sistema.
- Le recensioni dell'utente vengono rimosse dal sistema insieme ai conseguenti voti e segnalazioni associati a queste.
- I voti assegnati dall'utente vengono rimosse dal sistema.
- Le segnalazioni dell'utente vengono rimossi dal sistema.

Eccezioni

- L'utente specificato non è registrato al sistema.

4.1.1.7. `getUserReviews`

Restituisce una lista delle recensioni pubblicate da uno specifico utente se questo è registrato al sistema e almeno una sua recensione è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Review> getUserReviews(@NotNull final User user)`
- `public List<Review> getUserReviews(@NotNull final User user, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUserReviews(user)`
- `usersAgent.getUserReviews(user, index, limit)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.8. `getUserNumberOfReviews`

Restituisce il numero di recensioni pubblicate da uno specifico utente se questo è registrato al sistema e almeno una sua recensione è registrata, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserNumberOfReviews(@NotNull final User user)`

Contesto

- `usersAgent.getUserNumberOfReviews(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.9. `deleteUserReviews`

Rimuove le recensioni pubblicate da uno specifico utente se questo è registrato al sistema.

Interfacce

- `public void deleteUserReviews(@NotNull final User user)`

Contesto

- `usersAgent.deleteUserReviews(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se l'utente è registrato al sistema, le sue recensioni vengono rimosse insieme ai conseguenti voti e segnalazioni associati a queste.

4.1.1.10. `getUserVotes`

Restituisce una lista di voti assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno un voto assegnato da questo è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getUserVotes(@NotNull final User user)`
- `public List<Vote> getUserVotes(@NotNull final User user, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUserVotes(user)`
- `usersAgent.getUserVotes(user, index, limit)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.11. `getUserNumberOfVotes`

Restituisce il numero di voti assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno un voto assegnato da questo è registrato, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserNumberOfVotes(@NotNull final User user)`

Contesto

- `usersAgent.getUserNumberOfVotes(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.12. `deleteUserVotes`

Rimuove i voti assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema.

Interfacce

- `public void deleteUserVotes(@NotNull final User user)`

Contesto

- `usersAgent.deleteUserVotes(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se l'utente è registrato al sistema, i suoi voti assegnati vengono rimossi.

4.1.1.13. `getUserUpvotes`

Restituisce una lista di voti positivi assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno un voto positivo assegnato da questo è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getUserUpvotes(@NotNull final User user)`
- `public List<Vote> getUserUpvotes(@NotNull final User user, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUserUpvotes(user)`
- `usersAgent.getUserUpvotes(user, index, limit)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.14. `getUserNumberOfUpvotes`

Restituisce il numero di voti positivi assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno un voto positivo assegnato da questo è registrato, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserNumberOfUpvotes(@NotNull final User user)`

Contesto

- `usersAgent.getUserNumberOfUpvotes(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.15. `deleteUserUpvotes`

Rimuove i voti positivi assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema.

Interfacce

- `public void deleteUserUpvotes(@NotNull final User user)`

Contesto

- `usersAgent.getUserNumberOfUpvotes(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se l'utente è registrato al sistema, i suoi voti positivi assegnati vengono rimossi.

4.1.1.16. `getUserDownvotes`

Restituisce una lista di voti negativi assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno un voto negativo assegnato da questo è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getUserDownvotes(@NotNull final User user)`
- `public List<Vote> getUserDownvotes(@NotNull final User user, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUserDownvotes(user)`
- `usersAgent.getUserDownvotes(user, index, limit)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.17. `getUserNumberOfDownvotes`

Restituisce il numero di voti negativi assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno un voto negativo assegnato da questo è registrato, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserNumberOfDownvotes(@NotNull final User user)`

Contesto

- `usersAgent.getUserNumberOfDownvotes(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.18. `deleteUserDownvotes`

Rimuove i voti negativi assegnati a delle recensioni da uno specifico utente se questo è registrato al sistema.

Interfacce

- `public void deleteUserDownvotes(@NotNull final User user)`

Contesto

- `usersAgent.deleteUserDownvotes(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se l'utente è registrato al sistema, i suoi voti negativi vengono rimossi.

4.1.1.19. `getUserReports`

Restituisce una lista delle segnalazioni effettuate a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno una sua segnalazione è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Report> getUserReports(@NotNull final User user)`
- `public List<Report> getUserReports(@NotNull final User user, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUserReports(user)`
- `usersAgent.getUserReports(user, index, limit)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.20. `getUserNumberOfReports`

Restituisce il numero di segnalazioni effettuate a delle recensioni da uno specifico utente se questo è registrato al sistema e almeno una sua segnalazione è registrata, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserNumberOfReports(@NotNull final User user)`

Contesto

- `usersAgent.getUserNumberOfReports(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.21. `deleteUserReports`

Rimuove le segnalazione assegnate a delle recensioni da uno specifico utente se questo è registrato al sistema.

Interfacce

- `public void deleteUserReports(@NotNull final User user)`

Contesto

- `usersAgent.deleteUserReports(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se l'utente esiste, le sue segnalazioni vengono rimosse.

4.1.1.22. `getUserBacklog`

Restituisce una lista delle entrate in lista d'ascolto di uno specifico utente se questo è registrato al sistema ed esiste almeno un inserimento nella sua lista d'ascolto, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<BacklogEntry> getUserBacklog(@NotNull final User user)`
- `public List<BacklogEntry> getUserBacklog(@NotNull final User user, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `usersAgent.getUserBacklog(user)`
- `usersAgent.getUserBacklog(user, index, limit)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.1.23. `getUserBacklogLength`

Restituisce il numero di entrate nella lista d'ascolto di uno specifico utente se questo è registrato al sistema ed esiste almeno un inserimento nella sua lista d'ascolto, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserBacklogLength(@NotNull final User user)`

Contesto

- `usersAgent.getUserBacklogLength(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.24. `clearUserBacklog`

Rimuove le entrate presenti nella lista d'ascolto di uno specifico utente se questo è registrato al sistema.

Interfacce

- `public void clearUserBacklog(@NotNull final User user)`

Contesto

- `usersAgent.clearUserBacklog(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se l'utente esiste, le entrate della sua coda d'ascolto vengono rimosse.

4.1.1.25. `getUserAverageAssignedRating`

Restituisce il voto medio assegnato da un utente nelle sue recensioni se questo è registrato al sistema e almeno una sua recensione è registrata, altrimenti `null`.

Interfacce

- `public Double getUserAverageAssignedRating(@NotNull final User user)`

Contesto

- `usersAgent.getUserAverageAssignedRating(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.1.26. `getUserReputation`

Restituisce la reputazione di uno specifico utente (calcolata dal valore dei voti assegnati alle sue recensioni) se questo è registrato al sistema e almeno un voto è registrato per una sua recensione, altrimenti `0`.

Interfacce

- `public @NotNull Integer getUserReputation(@NotNull final User user)`

Contesto

- `usersAgent.getUserReputation(user)`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.

4.1.2. Interfaccia classe `ReviewsAgent`

4.1.2.1. `getReviews`

Restituisce una lista delle recensioni pubblicate sul sistema.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Review> getReviews()`
- `public List<Review> getReviews(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReviews()`
- `reviewsAgent.getReviews(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.2. `getReview`

Restituisce una recensione di cui viene specificato l'identificativo (dato dalla combinazione dello username dell'utente che la ha pubblicata e dall'identificativo del progetto musicale per cui è stata pubblicata) se questa è registrata sul sistema, altrimenti `null`.

Interfacce

- `public Review getReview(@NotNull final String reviewerUsername, @NotNull final Long reviewedAlbumId)`

Contesto

- `reviewsAgent.getReview(reviewerUsername, reviewedAlbumId)`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e diverso da `null`.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

4.1.2.3. `createReview`

Registra una nuova specifica recensione sul sistema.

Interfacce

- `public void createReview(@NotNull final Review review)`

Contesto

- `reviewsAgent.createReview(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- La recensione specificata viene registrata sul sistema.

Eccezioni

- Una recensione con lo stesso identificativo è già registrata sul sistema.
- L'oggetto `review` non rispetta i criteri di validazione.

4.1.2.4. `updateReview`

Aggiorna le informazioni della recensione specificata.

Interfacce

- `public void updateReview(@NotNull final Review review)`

Contesto

- `reviewsAgent.updateReview(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Le informazioni della recensione specificata vengono aggiornate.

Eccezioni

- La recensione specificata non è registrata sul sistema.
- L'oggetto `review` non rispetta i criteri di validazione.

4.1.2.5. `deleteReview`

Rimuove una specifica recensione dal sistema.

Interfacce

- `public void deleteReview(@NotNull Review review)`

Contesto

- `reviewsAgent.deleteReview(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- La recensione viene rimossa dal sistema insieme ai voti e alle segnalazioni associati a questa.

Eccezioni

- La recensione specificata non è registrata sul sistema.

4.1.2.6. `getVotes`

Restituisce una lista dei voti a recensioni registrati sul sistema.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getVotes()`
- `public List<Vote> getVotes(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getVotes()`
- `reviewsAgent.getVotes(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.7. `getVote`

Restituisce un voto di cui viene specificato l'identificativo (dato dalla combinazione dello username dell'utente che lo ha assegnato e dell'identificativo della recensione per cui viene assegnato) se questo è registrato sul sistema, altrimenti `null`.

Interfacce

- `public Vote getVote(@NotNull final String voterUsername, @NotNull final String reviewerUsername, @NotNull final Long reviewedAlbumId)`

Contesto

- `reviewsAgent.getVote(voterUsername, reviewerUsername, reviewedAlbumId)`

Condizioni d'entrata

- Il parametro `voterUsername` deve essere specificato e diverso da `null`.
- Il parametro `reviewerUsername` deve essere specificato e diverso da `null`.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

4.1.2.8. `createVote`

Registra un nuovo specifico voto sul sistema.

Interfacce

- `public void createVote(@NotNull final Vote vote)`

Contesto

- `reviewsAgent.createVote(vote)`

Condizioni d'entrata

- Il parametro `vote` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Il voto specificato viene registrato sul sistema.

Eccezioni

- Un voto con lo stesso identificativo è già registrato sul sistema.
- L'oggetto `vote` non rispetta i criteri di validazione.

4.1.2.9. `updateVote`

Aggiorna le informazioni del voto specificato.

Interfacce

- `public void updateVote(@NotNull final Vote vote)`

Contesto

- `reviewsAgent.updateVote(vote)`

Condizioni d'entrata

- Il parametro `vote` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Le informazioni del voto specificato vengono aggiornate.

Eccezioni

- Il voto specificato non è registrato sul sistema.
- L'oggetto `vote` non rispetta i criteri di validazione.

4.1.2.10. `deleteVote`

Rimuove uno specifico voto dal sistema.

Interfacce

- `public void deleteVote(@NotNull Vote vote)`

Contesto

- `reviewsAgent.deleteVote(vote)`

Condizioni d'entrata

- Il parametro `vote` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Il voto viene rimosso dal sistema.

Eccezioni

- Il voto specificato non è registrato sul sistema.

4.1.2.11. `getReports`

Restituisce una lista delle segnalazioni registrate sul sistema.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Report> getReports()`
- `public List<Report> getReports(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReports()`
- `reviewsAgent.getReports(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.12. `getReport`

Restituisce una segnalazione di cui viene specificato l'identificativo (dato dalla combinazione dello username dell'utente che la ha effettuata e dell'identificativo della recensione per cui viene effettuata) se questa è registrata sul sistema, altrimenti `null`.

Interfacce

- `public Report getReports(@NotNull final String reporterUsername, @NotNull final String reviewerUsername, @NotNull final Long reviewedAlbumId)`

Contesto

- `reviewsAgent.getReport(reporterUsername, reviewerUsername, reviewedAlbumId)`

Condizioni d'entrata

- Il parametro `reporterUsername` deve essere specificato e diverso da `null`.
- Il parametro `reviewerUsername` deve essere specificato e diverso da `null`.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

4.1.2.13. `createReport`

Registra una nuova specifica segnalazione sul sistema.

Interfacce

- `public void createReport(@NotNull final Report report)`

Contesto

- `reviewsAgent.createReport(report)`

Condizioni d'entrata

- Il parametro `report` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- La segnalazione specificata viene registrata sul sistema.

Eccezioni

- Una segnalazione con lo stesso identificativo è già registrata sul sistema.
- L'oggetto `report` non rispetta i criteri di validazione.

4.1.2.14. `updateReport`

Aggiorna le informazioni della segnalazione specificata.

Interfacce

- `public void updateReport(@NotNull final Report report)`

Contesto

- `reviewsAgent.updateReport(report)`

Condizioni d'entrata

- Il parametro `report` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Le informazioni della segnalazione specificata vengono aggiornate.

Eccezioni

- La segnalazione specificata non è registrata sul sistema.
- L'oggetto `report` non rispetta i criteri di validazione.

4.1.2.15. `deleteReport`

Rimuove una specifica segnalazione dal sistema.

Interfacce

- `public void deleteReport(@NotNull Report report)`

Contesto

- `reviewsAgent.deleteReport(report)`

Condizioni d'entrata

- Il parametro `report` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- La segnalazione viene rimossa dal sistema.

Eccezioni

- La segnalazione specificata non è registrata sul sistema.

4.1.2.16. `getReviewVotes`

Restituisce una lista di voti assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getReviewVotes(@NotNull final Review review)`
- `public List<Vote> getReviewVotes(@NotNull final Review review, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReviewVotes(review)`
- `reviewsAgent.getReviewVotes(review, index, limit)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.17. `getReviewNumberOfVotes`

Restituisce il numero di voti assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto per questa è registrato, altrimenti `0`.

Interfacce

- `public @NotNull Integer getReviewNumberOfVotes(@NotNull final Review review)`

Contesto

- `reviewsAgent.getReviewNumberOfVotes(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

4.1.2.18. `deleteReviewVotes`

Rimuove i voti assegnati ad una specifica recensione se questa è registrata sul sistema.

Interfacce

- `public void deleteReviewVotes(@NotNull final Review review)`

Contesto

- `reviewsAgent.deleteReviewVotes(user)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se la recensione è registrato al sistema, i voti assegnati a questa vengono rimossi.

4.1.2.19. `getReviewUpvotes`

Restituisce una lista di voti positvi assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto positivo per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getReviewUpvotes(@NotNull final Review review)`
- `public List<Vote> getReviewUpvotes(@NotNull final Review review, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReviewUpvotes(review)`
- `reviewsAgent.getReviewUpvotes(review, index, limit)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.20. `getReviewNumberOfUpvotes`

Restituisce il numero di voti positivi assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto positivo per questa è registrato, altrimenti `0`.

Interfacce

- `public @NotNull Integer getReviewNumberOfUpvotes(@NotNull final Review review)`

Contesto

- `reviewsAgent.getReviewNumberOfUpvotes(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

4.1.2.21. `deleteReviewUpvotes`

Rimuove i voti positivi assegnati ad una specifica recensione se questa è registrata sul sistema.

Interfacce

- `public void deleteReviewUpvotes(@NotNull final Review review)`

Contesto

- `reviewsAgent.deleteReviewUpvotes(user)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se la recensione è registrato al sistema, i voti positivi assegnati a questa vengono rimossi.

4.1.2.22. `getReviewDownvotes`

Restituisce una lista di voti negativi assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto negativo per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Vote> getReviewDownvotes(@NotNull final Review review)`
- `public List<Vote> getReviewDownvotes(@NotNull final Review review, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReviewDownvotes(review)`
- `reviewsAgent.getReviewDownvotes(review, index, limit)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.23. `getReviewNumberOfDownvotes`

Restituisce il numero di voti negativi assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto negativo per questa è registrato, altrimenti `0`.

Interfacce

- `public @NotNull Integer getReviewNumberOfDownvotes(@NotNull final Review review)`

Contesto

- `reviewsAgent.getReviewNumberOfDownvotes(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

4.1.2.24. `deleteReviewUpvotes`

Rimuove i voti negativi assegnati ad una specifica recensione se questa è registrata sul sistema.

Interfacce

- `public void deleteReviewDownvotes(@NotNull final Review review)`

Contesto

- `reviewsAgent.deleteReviewDownvotes(user)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se la recensione è registrato al sistema, i voti negativi assegnati a questa vengono rimossi.

4.1.2.25. `getReviewReports`

Restituisce una lista delle segnalazioni effettuate per una specifica recensione se questa è registrata al sistema e almeno una segnalazione ricevuta da questa è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Report> getReviewReports(@NotNull final Review review)`
- `public List<Report> getReviewReports(@NotNull final Review review, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReviewReports(review)`
- `reviewsAgent.getReviewReports(review, index, limit)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.26. `getReviewNumberOfReports`

Restituisce il numero di segnalazioni effettuate per una specifica recensione se questa è registrata al sistema e almeno e almeno una segnalazione ricevuta da questa è registrata, altrimenti `0`.

Interfacce

- `public @NotNull Integer getReviewNumberOfReports(@NotNull final Review review)`

Contesto

- `reviewsAgent.getReviewNumberOfReports(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

4.1.2.27. `deleteReviewReports`

Rimuove le segnalazione assegnate ad una specifica recensione se questa è registrata al sistema.

Interfacce

- `public void deleteReviewReports(@NotNull final Review review)`

Contesto

- `reviewsAgent.deleteReviewReports(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se la recensione esiste, le segnalazioni per questa vengono rimosse.

4.1.2.28. `getReviewScore`

Restituisce il punteggio di una specifica recensione (calcolata dal valore dei voti assegnati a questa) se questa è registrata al sistema e almeno un voto è registrato per questa, altrimenti `0`.

Interfacce

- `public @NotNull Integer getReviewScore(@NotNull final Review review)`

Contesto

- `reviewsAgent.getReviewScore(review)`

Condizioni d'entrata

- Il parametro `review` deve essere specificato e diverso da `null`.

4.1.2.29. `getTopReviews`

Restituisce una lista delle recensioni pubblicate sul sistema ordinate in maniera decrescente in base al punteggio di queste se almeno una recensione è registrata al sistema, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Review> getTopReviews()`
- `public List<Review> getTopReviews(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getTopReviews()`
- `reviewsAgent.getTopReviews(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.2.30. `getReportedReviews`

Restituisce una lista delle recensioni pubblicate sul sistema per cui è registrata almeno una segnalazione se almeno una segnalazione è registrata al sistema, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Review> getReportedReviews()`
- `public List<Review> getReportedReviews(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `reviewsAgent.getReportedReviews()`
- `reviewsAgent.getReportedReviews(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.3. Interfaccia classe CatalogAgent

4.1.3.1. `getBacklogEntries`

Restituisce una lista delle entrate delle liste d'ascolto degli utenti registrate sul sistema.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<BacklogEntry> getBacklogEntry()`
- `public List<BacklogEntry> getBacklogEntry(@Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `catalogAgent.getBacklogEntries()`
- `catalogAgent.getBacklogEntries(index, limit)`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.3.2. `getBacklogEntry`

Restituisce una entrata di lista d'ascolto di cui viene specificato l'identificativo (dato dalla combinazione dello username dell'utente a cui appartiene la lista d'ascolto in cui è inserita l'entrata e dall'identificativo del progetto musicale per cui è stata pubblicata) se questa è registrata sul sistema, altrimenti `null`.

Interfacce

- `public BacklogEntry getBacklogEntry(@NotNull final String username, @NotNull final Long albumId)`

Contesto

- `catalogAgent.getBacklogEntry(username, albumId)`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e diverso da `null`.
- Il parametro `albumId` deve essere specificato e diverso da `null`.

4.1.3.3. `createBacklogEntry`

Registra una nuova specifica entrata per la lista d'ascolto di un utente.

Interfacce

- `public void createBacklogEntry(@NotNull final BacklogEntry backlogEntry)`

Contesto

- `catalogAgent.createBacklogEntry(backlogEntry)`

Condizioni d'entrata

- Il parametro `backlogEntry` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- La entrata di lista d'ascolto specificata viene registrata sul sistema.

Eccezioni

- Una entrata di lista d'ascolto con lo stesso identificativo è già registrata sul sistema.
- L'oggetto `backlogEntry` non rispetta i criteri di validazione.

4.1.3.4. `updateBacklogEntry`

Aggiorna le informazioni dell'entrata di lista d'ascolto specificata.

Interfacce

- `public void updateBacklogEntry(@NotNull final BacklogEntry backlogEntry)`

Contesto

- `catalogAgent.updateBacklogEntry(backlogEntry)`

Condizioni d'entrata

- Il parametro `backlogEntry` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Le informazioni dell'entrata di lista d'ascolto specificata vengono aggiornate.

Eccezioni

- L'entrata di lista d'ascolto specificata non è registrata sul sistema.
- L'oggetto `backlogEntry` non rispetta i criteri di validazione.

4.1.3.5. `deleteBacklogEntry`

Rimuove una specifica entrata di lista d'ascolto.

Interfacce

- `public void deleteBacklogEntry(@NotNull BacklogEntry backlogEntry)`

Contesto

- `catalogAgent.deleteBacklogEntry(backlogEntry)`

Condizioni d'entrata

- Il parametro `backlogEntry` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- L'entrata di lista d'ascolto specificata viene rimossa dal sistema.

Eccezioni

- L'entrata di lista d'ascolto specificata non è registrata sul sistema.

4.1.3.6. `getAlbum`

Restituisce un progetto musicale di cui viene specificato l'identificativo se questo è presente nel catalogo del sistema, altrimenti `null`.

Interfacce

- `public Album getAlbum(@NotNull final Long albumId)`

Contesto

- `catalogAgent.getAlbum(albumId)`

Condizioni d'entrata

- Il parametro `albumId` deve essere specificato e diverso da `null`.

4.1.3.7. `getArtist`

Restituisce un artista musicale di cui viene specificato l'identificativo se questo è presente nel catalogo del sistema, altrimenti `null`.

Interfacce

- `public Artist getArtist(@NotNull final Long artistId)`

Contesto

- `catalogAgent.getArtist(artistId)`

Condizioni d'entrata

- Il parametro `artistId` deve essere specificato e diverso da `null`.

4.1.3.8. `getGenre`

Restituisce un genere musicale di cui viene specificato l'identificativo se questo è presente nel catalogo del sistema, altrimenti `null`.

Interfacce

- `public Genre getGenre(@NotNull final Long genreId)`

Contesto

- `catalogAgent.getGenre(genreId)`

Condizioni d'entrata

- Il parametro `genreId` deve essere specificato e diverso da `null`.

4.1.3.9. `getAlbumReviews`

Restituisce una lista delle recensioni pubblicate per uno specifico progetto musicale se questo è presente nel catalogo del sistema e almeno una recensione per questo è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public List<Review> getAlbumReviews(@NotNull final Album album)`
- `public List<Review> getAlbumReviews(@NotNull final Album album, @Min(0) final Integer index, @Min(1) final Integer limit)`

Contesto

- `catalogAgent.getAlbumReviews(album)`
- `catalogAgent.getAlbumReviews(album, index, limit)`

Condizioni d'entrata

- Il parametro `album` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

4.1.3.10. `getAlbumNumberOfReviews`

Restituisce il numero di recensioni pubblicate per uno specifico progetto musicale se questo è presente nel catalogo del sistema e almeno una recensione per questo è registrata, altrimenti `0`.

Interfacce

- `public @NotNull Integer getAlbumNumberOfReviews(@NotNull final User user)`

Contesto

- `catalogAgent.getAlbumNumberOfReviews(album)`

Condizioni d'entrata

- Il parametro `album` deve essere specificato e diverso da `null`.

4.1.3.11. `deleteAlbumReviews`

Rimuove le recensioni pubblicate per uno specifico progetto musicale se questo è presente nel catalogo del sistema.

Interfacce

- `public void deleteAlbumReviews(@NotNull final Album album)`

Contesto

- `catalogAgent.deleteAlbumReviews(album)`

Condizioni d'entrata

- Il parametro `album` deve essere specificato e diverso da `null`.

Condizioni d'uscita

- Se il progetto musicale è presente nel catalogo del sistema, le recensioni per questo vengono rimosse insieme ai conseguenti voti e segnalazioni associati a queste.

4.1.3.12. `getAlbumAverageRating`

Restituisce il voto medio assegnato ad un progetto musicale se questo è presente nel catalogo del sistema e almeno una recensione per questo è registrata, altrimenti `null`.

Interfacce

- `public Double getAlbumAverageRating(@NotNull final Album album)`

Contesto

- `catalogAgent.getAlbumAverageRating(album)`

Condizioni d'entrata

- Il parametro `album` deve essere specificato e diverso da `null`.

4.1.3.13. `getArtistAlbums`

Restituisce i progetti musicali pubblicati da uno specifico artista musicale se questo è presente sul catalogo del sistema con almeno un suo progetto, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public Albums getArtistAlbums(@NotNull final Artist artist)`
- `public Albums getArtistAlbums(@NotNull final Artist artist, @NotNull @Min(0) final Integer index, @NotNull @Min(1) final Integer limit)`

Contesto

- `catalogAgent.getArtistAlbums(artist)`
- `catalogAgent.getArtistAlbums(artist, index, limit)`

Condizioni d'entrata

- Il parametro `artist` deve essere specificato e diverso da `null`.
- Se specificati, i parametro `index` e `limit` devono essere maggiori o uguali rispettivamente di `0` e di `1`.

4.1.3.14. `getArtistNumberOfReviews`

Restituisce il numero di recensioni pubblicate per i progetti musicali di uno specifico artista musicale se questo è presente nel catalogo del sistema e almeno una recensione per almeno un progetto pubblicato da questo è registrata, altrimenti `0`.

Interfacce

- `public @NotNull Integer getArtistNumberOfReviews(@NotNull final Artist artist)`

Contesto

- `catalogAgent.getArtistNumberOfReviews(artist)`

Condizioni d'entrata

- Il parametro `artist` deve essere specificato e diverso da `null`.

4.1.3.15. `getArtistAverageRating`

Restituisce il voto medio assegnato ad un artista musicale (calcolato in base ai voti medi di ogni suo progetto musicale) se questo è presente nel catalogo del sistema e almeno una recensione per almeno un suo progetto musicale è registrata, altrimenti `null`.

Interfacce

- `public Double getArtistAverageRating(@NotNull final Artist artist)`

Contesto

- `catalogAgent.getArtistAverageRating(artist)`

Condizioni d'entrata

- Il parametro `artist` deve essere specificato e diverso da `null`.

4.1.3.16. `getTopAlbums`

Restituisce i progetti musicali presenti nel catalogo della piattaforma ordinati in base alla loro popolarità se almeno un progetto è presente nel catalogo della piattaforma, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public Albums getTopAlbums()`
- `public Albums getTopAlbums(@NotNull @Min(0) final Integer index, @NotNull @Min(1) final Integer limit)`

Contesto

- `catalogAgent.getTopAlbums()`
- `catalogAgent.getTopAlbums(index, limit)`

Condizioni d'entrata

- Se specificati, i parametro `index` e `limit` devono essere maggiori o uguali rispettivamente di `0` e di `1`.

4.1.3.17. `searchAlbums`

Restituisce i progetti musicali presenti nel catalogo della piattaforma che corrispondono ai termini di ricerca specificati se almeno un progetto rispetta questi termini, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public Albums searchAlbums()`
- `public Albums searchAlbums(@NotNull @Min(0) final Integer index, @NotNull @Min(1) final Integer limit)`

Contesto

- `catalogAgent.searchAlbums()`
- `catalogAgent.searchAlbums(index, limit)`

Condizioni d'entrata

- Se specificati, i parametro `index` e `limit` devono essere maggiori o uguali rispettivamente di `0` e di `1`.

4.1.3.18. `searchArtists`

Restituisce gli artisti musicali presenti nel catalogo della piattaforma che corrispondono ai termini di ricerca specificati se almeno un artista rispetta questi termini, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- `public Artists searchArtists()`
- `public Artists searchArtists(@NotNull @Min(0) final Integer index, @NotNull @Min(1) final Integer limit)`

Contesto

- `catalogAgent.searchArtists()`
- `catalogAgent.searchArtists(index, limit)`

Condizioni d'entrata

- Se specificati, i parametro `index` e `limit` devono essere maggiori o uguali rispettivamente di `0` e di `1`.

4.2. Interfacce classi in `endpoints.services`

Le classi di questo package si interfacciano alle classi del package `application.model` per realizzare le funzionalità offerte dal sistema, fornendo un'interfaccia verso queste agli utenti della piattaforma.

4.2.1. Interfaccia classe `AuthenticationService`

4.2.1.1. `logIn`

Permette ad un utente che fornisce le sue credenziali di autenticarsi al proprio account registrato sul sistema.

Interfacce

- ```
  @Path("/log-in")
  @POST
  public Response logIn(@FormParam("username") @NotBlank
  final String username, @FormParam("password") @NotBlank
  final String password, @Context final HttpServletRequest
  request)
  ```

Contesto

- `POST /log-in?username=${username}&password=${password}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `password` deve essere specificato e non vuoto.
- L'utente invocante il metodo non è autenticato al sistema.

Condizioni d'uscita

- L'utente viene autenticato dal sistema al suo account.

Eccezioni

- Le credenziali fornite non sono valide, ovvero non corrispondono a quelle di nessun utente registrato alla piattaforma.
  ```

4.2.1.2. logOut

Permette ad un utente autenticato al sistema di de-autenticarsi.

Interfacce

- ```
  @Path("/log-out")
  @POST
  public Response logOut(@Context final HttpServletRequest
  request)
  ```

Contesto

- **POST /log-out**

Condizioni d'entrata

- L'utente invocante il metodo è autenticato al sistema.

Condizioni d'uscita

- L'utente viene de-autenticato dal sistema dal suo account.
  ```

4.2.1.3. signUp

Permette ad un utente di registrare un account utente al sistema.

Interfacce

- ```
  @Path("/sign-up")
  @POST
  public Response signUp(@FormParam("username")
  @Pattern(regexp = User.USERNAME_PATTERN) final String
  username, @FormParam("password") @NotBlank @Email final
  String email, @FormParam("password") @NotNull
  @Pattern(regexp = User.PASSWORD_PATTERN) final String
  password, @Context final HttpServletRequest request)
  ```

Contesto

- ```
  POST
  /sign-up?username=${username}&email=${email}&password=${p
  assword}
  ```

Condizioni d'entrata

- Il parametro `username` deve essere specificato e rispettare una specifica espressione regolare (`^(?=.{3,36}$)[a-zA-Z0-9]+([_-]?[a-zA-Z0-9])*$`, ovvero 3-36 caratteri di lunghezza e esclusivamente caratteri alfanumerici).
- Il parametro `email` deve essere specificato e deve essere un indirizzo e-mail valido.
- Il parametro `password` deve essere specificato e rispettare una specifica espressione regolare (`^(?=(.*\d){2})[0-9a-zA-Z]{8,72}$`, ovvero 8-72 caratteri di lunghezza e contenente almeno due cifre).
- L'utente invocante il metodo non è autenticato al sistema.

Condizioni d'uscita

- Un nuovo account utente viene registrato al sistema utilizzando le informazioni fornite.
- L'utente viene autenticato dal sistema al suo account.

Eccezioni

- Un account utente con lo stesso nome utente è già registrato al sistema.
- Un account utente con lo stesso indirizzo e-mail è già registrato al sistema.

4.2.2. Interfaccia classe `UsersService`

4.2.2.1. `getUser`

Restituisce la rappresentazione di un account utente specificato in formato JSON (ignorandone le informazioni sensibili) se questo è registrato al sistema, altrimenti `null`.

Interfacce

- ```
  @Path("/get-user")
  @GET
  public Response getUser(@QueryParam("username") @NotBlank
  final String username)
  ```

Contesto

- `GET /get-user?username=${username}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
  ```

4.2.2.2. `getUserReviews`

Restituisce la rappresentazione in formato JSON delle recensioni pubblicate da uno specifico utente se questo è registrato al sistema e almeno una sua recensione è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-user-reviews")
  @GET
  public Response getUser(@QueryParam("user") @NotBlank
  final String username, @QueryParam("index") @Min(0) final
  Integer index, @QueryParam("limit") @Min(1) final Integer
  limit, @Context final HttpServletRequest request)
  ```

Contesto

- `GET /get-user-review?user=${username}`
- ```
  GET
  /get-user-review?user=${username}&index=${index}&limit=${
  limit}
  ```

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- L'utente specificato non è registrato al sistema.

4.2.2.3. `getUserVotes`

Restituisce una lista di voti assegnati a delle recensioni da uno specifico utente in formato JSON se questo è registrato al sistema e almeno un voto assegnato da questo è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-user-votes")
  @GET
  public Response getUserVotes(@QueryParam("user")
  @NotBlank final String username, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-user-votes?user=${username}`
- `GET /get-user-votes?user=${username}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- L'utente specificato non è registrato al sistema.
  ```

4.2.2.4. `getUserVotes`

Restituisce una lista di voti positivi assegnati a delle recensioni da uno specifico utente in formato JSON se questo è registrato al sistema e almeno un voto positivo assegnato da questo è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-user-upvotes")
  @GET
  public Response getUserUpvotes(@QueryParam("user")
  @NotBlank final String username, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-user-upvotes?user=${username}`
- `GET /get-user-upvotes?user=${username}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- L'utente specificato non è registrato al sistema.
  ```

4.2.2.5. `getUserDownvotes`

Restituisce una lista di voti negativi assegnati a delle recensioni da uno specifico utente in formato JSON se questo è registrato al sistema e almeno un voto negativo assegnato da questo è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-user-downvotes")
  @GET
  public Response getUserDownvotes(@QueryParam("user")
  @NotBlank final String username, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-user-downvotes?user=${username}`
- `GET /get-user-downvotes?user=${username}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `user` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- L'utente specificato non è registrato al sistema.
  ```

4.2.2.6. `getUserReports`

Restituisce la rappresentazione delle segnalazioni effettuate da uno specifico utente in formato JSON se questo è registrato al sistema e almeno una sua segnalazione è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-user-reports")
  @GET
  public Response getUserReports(@QueryParam("user")
  @NotBlank final String username, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-user-reports?user=${username}`
- `GET /get-user-reports?user=${username}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`
- L'utente invocante il metodo deve essere autenticato al sistema e disporre di privilegi di moderazione.

Eccezioni

- L'utente specificato non è registrato al sistema.
  ```

4.2.2.7. `getUserBacklog`

Restituisce la rappresentazione delle delle entrate in lista d'ascolto di uno specifico utente in formato JSON se questo è registrato al sistema ed esiste almeno un inserimento nella sua lista d'ascolto, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-user-backlog")
  @GET
  public Response getUserBacklog(@QueryParam("user")
  @NotBlank final String username, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-user-backlog?user=${username}`
- `GET /get-user-backlog?user=${username}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- L'utente specificato non è registrato al sistema.
  ```

4.2.2.8. `updateUserEmail`

Aggiorna l'indirizzo e-mail associato all'account di un utente specificato se questo è registrato al sistema.

Interfacce

- ```
  @Path("/update-user-email")
  @POST
  public Response updateUserEmail(@FormParam("username")
  @NotBlank final String username, @FormParam("cpassword")
  @NotBlank final String currentPassword,
  @FormParam("nemail") @NotBlank @Email final String
  newEmail, @Context final HttpServletRequest request)
  ```

Contesto

- ```
  POST
  /update-user-email?username=${username}&cpassword=${curre
  ntPassword}&nemail=${newEmail}
  ```

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `currentPassword` deve essere specificato e non vuoto.
- Il parametro `newEmail` deve essere specificato ed essere un indirizzo e-mail valido.
- L'utente invocante il metodo deve essere autenticato al sistema e il suo account deve corrispondere a quello specificato dal parametro `username`.

Eccezioni

- L'utente specificato non è registrato al sistema.
- Il parametro `currentPassword` non corrisponde alla password dell'utente specificato.
- L'indirizzo e-mail fornito corrisponde già a quello di un account utente diverso da quello dell'utente invocante il metodo.

4.2.2.9. `updateUserPassword`

Aggiorna la password associata all'account di un utente specificato se questo è registrato al sistema.

Interfacce

- ```
  @Path("/update-user-password")
  @POST
  public Response updateUserPassword(@FormParam("username")
  @NotBlank final String username, @FormParam("cpassword")
  @NotBlank final String currentPassword,
  @FormParam("npassword") @NotNull @Pattern(regexp =
  User.PASSWORD_PATTERN) final String newPassword, @Context
  final HttpServletRequest request)
  ```

Contesto

- ```
  POST
  /update-user-password?username=${username}&cpassword=${cu
  rrentPassword}&npassword=${newPassword}
  ```

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `currentPassword` deve essere specificato e non vuoto.
- Il parametro `newPassword` deve essere specificato e rispettare una specifica espressione regolare (`^(?=(.*\d){2})[0-9a-zA-Z]{8,72}$`, ovvero 8-72 caratteri di lunghezza e contenente almeno due cifre).
- L'utente invocante il metodo deve essere autenticato al sistema e il suo account deve corrispondere a quello specificato dal parametro `username`.

Eccezioni

- L'utente specificato non è registrato al sistema.
- Il parametro `currentPassword` non corrisponde alla password dell'utente specificato.

4.2.2.10. `updateUserRole`

Aggiorna il ruolo associato all'account di un utente specificato se questo è registrato al sistema.

Interfacce

- ```
  @Path("/update-user-role")
  @POST
  public Response updateUserRole(@FormParam("username")
  @NotBlank final String username, @FormParam("role")
  @NotNull final User.Role role, @Context final
  HttpServletRequest request)
  ```

Contesto

- `POST /update-user-role?username=${username}&role=${role}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `role` deve essere specificato e corrispondere a un ruolo valido definito dal sistema.
- L'utente invocante il metodo deve essere autenticato al sistema e disporre di privilegi di amministrazione.

Eccezioni

- L'utente specificato non è registrato al sistema.
  ```

4.2.2.11. `recoverUserAccount`

Invia una e-mail contenente un link per il recupero della password di un account utente di cui viene specificato l'indirizzo e-mail associato a questo.

Interfacce

- ```
  @Path("/recover-user-account")
  @POST
  public Response recoverUserAccount(@FormParam("email")
  @NotBlank @Email final String email, @Context final
  HttpServletRequest request, @Context final UriInfo
  uriInfo)
  ```

Contesto

- `POST /recover-user-account?email=${email}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `role` deve essere specificato e corrispondere a un ruolo valido.
- L'utente invocante il metodo non deve essere autenticato.

Eccezioni

- L'indirizzo e-mail specificato non corrisponde a nessun account utente registrato al sistema.
  ```

4.2.2.12. `resetUserPassword`

Reimposta la password di un account utente tramite un token specifico inviato per e-mail.

Interfacce

- ```
  @Path("/reset-user-password")
  @POST
  public Response resetUserPassword(@FormParam("token")
  @NotBlank final String token, @FormParam("password")
  @NotNull @Pattern(regexp = User.PASSWORD_PATTERN) final
  String password, @Context final HttpServletRequest
  request)
  ```

Contesto

- `POST /reset-user-password?token=${token}&password=${password}`

Condizioni d'entrata

- Il parametro `token` deve essere specificato e non vuoto.
- Il parametro `password` deve essere specificato e rispettare una specifica espressione regolare (`^(?=(.*\d){2})[0-9a-zA-Z]{8,72}$`, ovvero 8-72 caratteri di lunghezza e contenente almeno due cifre).
- L'utente invocante il metodo non deve essere autenticato.

Eccezioni

- Il token è malformato o scaduto.
- L'utente specificato nel token non è più registrato al sistema.
  ```

4.2.3. Interfaccia classe `ReviewsService`

4.2.3.1. `getReview`

Restituisce la rappresentazione in formato JSON di una recensione specificata se questa è registrata sul sistema, altrimenti `null`.

Interfacce

- ```
  @Path("/get-review")
  @GET
  public Response getReview(@QueryParam("reviewer")
  @NotBlank final String reviewerUsername,
  @QueryParam("album") @NotNull final Long reviewedAlbumId)
  ```

Contesto

- ```
  GET
  /get-review?reviewer=${reviewerUsername}&album=${reviewed
  AlbumId}
  ```

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

4.2.3.2. `getReviewVote`

Restituisce la rappresentazione in formato JSON di uno specifico voto se questo è registrato sul sistema, altrimenti `null`.

Interfacce

- ```
  @Path("/get-review-vote")
  @GET
  public Response getReviewVote(@QueryParam("voter")
  @NotBlank final String voterUsername,
  @QueryParam("reviewer") @NotBlank final String
  reviewerUsername, @QueryParam("album") @NotNull final
  Long reviewedAlbumId, @Context final HttpServletRequest
  request)
  ```

Contesto

- ```
  GET
  /get-review-vote?voter=${voterUsername}&reviewer=${review
  erUsername}&album=${reviewedAlbumId}
  ```

Condizioni d'entrata

- Il parametro `voterUsername` deve essere specificato e non vuoto.
- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

Eccezioni

- L'utente votatore specificato non è registrato al sistema.
- La recensione specificata non è registrata sulla piattaforma.

4.2.3.3. `getReviewVotes`

Restituisce la rappresentazione in formato JSON di una lista di voti assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-review-votes")
  @GET
  public Response getReviewVotes(@QueryParam("reviewer")
  @NotBlank final String reviewerUsername,
  @QueryParam("album") @NotNull final Long reviewedAlbumId,
  @QueryParam("index") @Min(0) final Integer index,
  @QueryParam("limit") @Min(1) final Integer limit,
  @Context final HttpServletRequest request)
  ```

Contesto

- `GET /get-review-votes?reviewer=${reviewerUsername}&album=${reviewedAlbumId}`
- `GET /get-review-votes?reviewer=${reviewerUsername}&album=${reviewedAlbumId}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- La recensione specificata non è registrata sulla piattaforma.
  ```

4.2.3.4. `getReviewUpvotes`

Restituisce la rappresentazione in formato JSON di una lista di voti positivi assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto positivo per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-review-upvotes")
  @GET
  public Response getReviewUpvotes(@QueryParam("reviewer")
  @NotBlank final String reviewerUsername,
  @QueryParam("album") @NotNull final Long reviewedAlbumId,
  @QueryParam("index") @Min(0) final Integer index,
  @QueryParam("limit") @Min(1) final Integer limit,
  @Context final HttpServletRequest request)
  ```

Contesto

- `GET /get-review-upvotes?reviewer=${reviewerUsername}&album=${reviewedAlbumId}`
- `GET /get-review-upvotes?reviewer=${reviewerUsername}&album=${reviewedAlbumId}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

Eccezioni

- La recensione specificata non è registrata sulla piattaforma.
  ```

4.2.3.5. `getReviewDownvotes`

Restituisce la rappresentazione in formato JSON di una lista di voti negativi assegnati ad una specifica recensione se questa è registrata sul sistema e almeno un voto negativo per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-review-downvotes")
  @GET
  public Response
  getReviewDownvotes(@QueryParam("reviewer") @NotBlank
  final String reviewerUsername, @QueryParam("album")
  @NotNull final Long reviewedAlbumId, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-review-downvotes?reviewer=${reviewerUsername}&album=${reviewedAlbumId}`
- `GET /get-review-downvotes?reviewer=${reviewerUsername}&album=${reviewedAlbumId}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.

Eccezioni

- La recensione specificata non è registrata sulla piattaforma.
  ```

4.2.3.6. `getReviewReport`

Restituisce la rappresentazione in formato JSON di una specifica segnalazione se questa è registrato sul sistema, altrimenti `null`.

Interfacce

- ```
  @Path("/get-review-report")
  @GET
  public Response getReviewReport(@QueryParam("reporter")
  @NotBlank final String reporterUsername,
  @QueryParam("reviewer") @NotBlank final String
  reviewerUsername, @QueryParam("album") @NotNull final
  Long reviewedAlbumId, @Context final HttpServletRequest
  request)
  ```

Contesto

- ```
  GET
  /get-review-report?voter=${voterUsername}&reviewer=${revi
  ewerUsername}&album=${reviewedAlbumId}
  ```

Condizioni d'entrata

- Il parametro `reporterUsername` deve essere specificato e non vuoto.
- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato al sistema e disporre di privilegi di moderazione.

Eccezioni

- L'utente segnalatore specificato non è registrato al sistema.
- La recensione specificata non è registrata sulla piattaforma.

4.2.3.7. `getReviewReports`

Restituisce la rappresentazione in formato JSON di una lista di segnalazioni effettuate per una specifica recensione se questa è registrata sul sistema e almeno una segnalazione per questa è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-review-reports")
  @GET
  public Response getReviewReports(@QueryParam("reviewer")
  @NotBlank final String reviewerUsername,
  @QueryParam("album") @NotNull final Long reviewedAlbumId,
  @QueryParam("index") @Min(0) final Integer index,
  @QueryParam("limit") @Min(1) final Integer limit,
  @Context final HttpServletRequest request)
  ```

Contesto

- `GET /get-review-reports?reviewer=${reviewerUsername}&album=${reviewedAlbumId}`
- `GET /get-review-reports?reviewer=${reviewerUsername}&album=${reviewedAlbumId}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato al sistema e disporre di privilegi di moderazione.

Eccezioni

- La recensione specificata non è registrata sulla piattaforma.
  ```

4.2.3.8. `getReportedReviews`

Restituisce la rappresentazione in formato JSON della lista delle recensioni pubblicate sul sistema per cui è registrata almeno una segnalazione se almeno una segnalazione è registrata al sistema, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-reported-reviews")
  @GET
  public Response getReportedReviews(@QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-reported-reviews`
- `GET /get-reported-reviews?index=${index}&limit=${limit}`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.
- L'utente invocante il metodo deve essere autenticato al sistema e disporre di privilegi di moderazione.
  ```

4.2.3.9. publishReview

Registra una nuova specifica recensione sul sistema o la aggiorna se questa esiste già.

Interfacce

- ```
  @Path("/publish-review")
  @POST
  public Response publishReview(@FormParam("reviewer")
  @NotBlank final String reviewerUsername,
  @FormParam("album") @NotNull final Long reviewedAlbumId,
  @FormParam("content") @NotBlank @Size(min =
  Review.MIN_CONTENT_LENGTH, max =
  Review.MAX_CONTENT_LENGTH) final String content,
  @FormParam("rating") @NotBlank
  @Min(Review.MIN_ALLOWED_RATING),
  @Min(Review.MIN_ALLOWED_RATING) final Integer rating,
  @Context final HttpServletRequest request)
  ```

Contesto

- ```
  POST
  /publish-review?reviewer=${reviewerUsername}&album=${revi
  ewedAlbumId}&content=${content}&rating=${rating}
  ```

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- Il parametro `content` deve essere specificato e rientrare la lunghezza minima e massima del corpo di una recensione.
- Il parametro `rating` deve essere specificato e rientrare il valore minimo e massimo di un voto assegnabile ad un progetto musicale (tra `1` e `10` inclusivi).
- L'utente invocante il metodo deve essere autenticato alla piattaforma e il suo account utente deve corrispondere a quello specificato dal parametro `reviewer`.

Condizioni d'uscita

- La recensione specificata viene registrata o aggiornata sul sistema.

Eccezioni

- L'utente recensore non viene trovato sul sistema.
- Il progetto musicale specificato non viene trovato sul catalogo della piattaforma.

4.2.3.10. `deleteReview`

Rimuove una specifica recensione dal sistema.

Interfacce

- ```
  @Path("/delete-review")
  @POST
  public Response deleteReview(@FormParam("reviewer")
  @NotBlank final String reviewerUsername,
  @FormParam("album") @NotNull final Long reviewedAlbumId,
  @Context final HttpServletRequest request)
  ```

Contesto

- `POST /delete-review?reviewer=${reviewerUsername}&album=${reviewedAlbumId}`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato alla piattaforma e il suo account utente deve corrispondere a quello specificato dal parametro `reviewer`, o essere dotato di privilegi di moderazione.

Condizioni d'uscita

- La recensione specificata viene eliminata dal sistema.

Eccezioni

- La recensione specificata non viene trovata sul sistema.
  ```

4.2.3.11. voteReview

Registra un nuovo voto per una specifica recensione sul sistema o lo aggiorna o rimuove se questo esiste già.

Interfacce

- ```
  @Path("/vote-review")
  @POST
  public Response publishReview(@FormParam("voter")
  @NotBlank final String voterUsername,
  @FormParam("reviewer") @NotBlank final String
  reviewerUsername, @FormParam("album") @NotNull final Long
  reviewedAlbumId, @FormParam("vote") @NotNull final String
  voteValueParameter, @Context final HttpServletRequest
  request)
  ```

Contesto

- `POST /vote-review?voter=${voterUsername}&reviewer=${reviewerUsername}&album=${reviewedAlbumId}&vote=${voteValue}`

Condizioni d'entrata

- Il parametro `voterUsername` deve essere specificato e non vuoto.
- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato alla piattaforma e il suo account utente deve corrispondere a quello specificato dal parametro `voter`.

Condizioni d'uscita

- Il voto viene pubblicato, aggiornato o rimosso dal sistema.

Eccezioni

- L'utente votatore non viene trovato sul sistema.
- La recensione non viene trovata sul sistema.
  ```

4.2.3.12. `reportReview`

Registra una segnalazione per una specifica recensione sul sistema.

Interfacce

- ```
  @Path("/report-review")
  @POST
  public Response reportReview(@FormParam("reporter")
  @NotBlank final String reporterUsername,
  @FormParam("reviewer") @NotBlank final String
  reviewerUsername, @FormParam("album") @NotNull final Long
  reviewedAlbumId, @Context final HttpServletRequest
  request)
  ```

Contesto

- ```
  POST
  /report-review?reporter=${reporterUsername}&reviewer=${re
  viewerUsername}&album=${reviewedAlbumId}
  ```

Condizioni d'entrata

- Il parametro `reporterUsername` deve essere specificato e non vuoto.
- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato alla piattaforma e il suo account utente deve corrispondere a quello specificato dal parametro `reporter`.

Condizioni d'uscita

- La segnalazione viene registrata sul sistema.

Eccezioni

- L'utente segnalatore non viene trovato sul sistema.
- La recensione non viene trovata sul sistema.

4.2.3.13. `deleteReviewReports`

Rimuove le segnalazione assegnate ad una specifica recensione se questa è registrata al sistema.

Interfacce

- ```
  @Path("/delete-review-reports")
  @POST
  public Response
  deleteReviewReports(@FormParam("reviewer") @NotBlank
  final String reviewerUsername, @FormParam("album")
  @NotNull final Long reviewedAlbumId, @Context final
  HttpServletRequest request)
  ```

Contesto

- `POST /delete-review-reports?reviewer=${reviewerUsername}&album=${reviewedAlbumId}`

Condizioni d'entrata

- Il parametro `reviewerUsername` deve essere specificato e non vuoto.
- Il parametro `reviewedAlbumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato alla piattaforma e disporre di privilegi di moderazione.

Condizioni d'uscita

- Le segnalazioni per la recensione specificata vengono rimosse.

Eccezioni

- La recensione non viene trovata sul sistema.
  ```

4.2.4. Interfaccia classe `CatalogService`

4.2.4.1. `getBacklogEntry`

Restituisce la rappresentazione in formato JSON di una entrata in lista d'ascolto specificata se questa è registrata sul sistema, altrimenti `null`.

Interfacce

- ```
  @Path("/get-backlog-entry")
  @GET
  public Response getBacklogEntry(@QueryParam("user")
  @NotBlank final String username, @QueryParam("album")
  @NotNull final Long albumId)
  ```

Contesto

- `GET /get-backlog-entry?user=${username}&album=${albumId}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `albumId` deve essere specificato e diverso da `null`.
  ```

4.2.4.2. `getAlbum`

Restituisce la rappresentazione in formato JSON di un progetto musicale se questo è presente sul catalogo della piattaforma, altrimenti `null`.

Interfacce

- ```
  @Path("/get-album")
  @GET
  public Response getAlbum(@QueryParam("album") @NotNull
  final Long albumId)
  ```

Contesto

- `GET /get-album?album=${albumId}`

Condizioni d'entrata

- Il parametro `albumId` deve essere specificato e diverso da `null`.
  ```

4.2.4.3. `getArtist`

Restituisce la rappresentazione in formato JSON di un artista musicale se questo è presente sul catalogo della piattaforma, altrimenti `null`.

Interfacce

- ```
  @Path("/get-artist")
  @GET
  public Response getArtist(@QueryParam("artist") @NotNull
  final Long artistId)
  ```

Contesto

- `GET /get-artist?artist=${artistId}`

Condizioni d'entrata

- Il parametro `artistId` deve essere specificato e diverso da `null`.
  ```

4.2.4.4. `getGenre`

Restituisce la rappresentazione in formato JSON di un genere musicale se questo è presente sul catalogo della piattaforma, altrimenti `null`.

Interfacce

- ```
  @Path("/get-genre")
  @GET
  public Response getGenre(@QueryParam("genre") @NotNull
  final Long genreId)
  ```

Contesto

- `GET /get-genre?genre=${genreId}`

Condizioni d'entrata

- Il parametro `artistId` deve essere specificato e diverso da `null`.
  ```

4.2.4.5. `getAlbumReviews`

Restituisce la rappresentazione in formato JSON della lista delle recensioni pubblicate per uno specifico progetto musicale se questo è presente nel catalogo del sistema e almeno una recensione per questo è registrata, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-album-reviews")
  @GET
  public Response getAlbumReviews(@QueryParam("album")
  @NotNull final Long albumId, @QueryParam("index") @Min(0)
  final Integer index, @QueryParam("limit") @Min(1) final
  Integer limit, @Context final HttpServletRequest request)
  ```

Contesto

- `GET /get-album-reviews?album=${albumId}`
- `GET /get-album-reviews?album=${albumId}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `albumId` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- Il progetto musicale specificato non è presente nel catalogo della piattaforma.
  ```

4.2.4.6. `getArtistAlbums`

Restituisce la rappresentazione in formato JSON della lista dei progetti musicali pubblicati da uno specifico artista musicale se questo è presente nel catalogo del sistema e almeno un suo progetto musicale è registrato, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-artist-albums")
  @GET
  public Response getArtistAlbums(@QueryParam("artist")
  @NotNull final Long artistId, @QueryParam("index")
  @Min(0) final Integer index, @QueryParam("limit") @Min(1)
  final Integer limit, @Context final HttpServletRequest
  request)
  ```

Contesto

- `GET /get-artist-albums?artist=${artistId}`
- `GET /get-artist-albums?artist=${artistId}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `albumId` deve essere specificato e diverso da `null`.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.

Eccezioni

- L'artista musicale specificato non è presente nel catalogo della piattaforma.
  ```

4.2.4.7. `getTopAlbums`

Restituisce la rappresentazione in formato JSON dei progetti musicali presenti nel catalogo della piattaforma ordinati in base alla loro popolarità se almeno un progetto è presente nel catalogo della piattaforma, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/get-top-albums")
  @GET
  public Response getTopAlbums(@QueryParam("index") @Min(0)
  final Integer index, @QueryParam("limit") @Min(1) final
  Integer limit)
  ```

Contesto

- `GET /get-top-albums`
- `GET /get-top-albums?index=${index}&limit=${limit}`

Condizioni d'entrata

- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.
  ```

4.2.4.8. `searchAlbums`

Restituisce la rappresentazione in formato JSON dei progetti musicali presenti nel catalogo della piattaforma che corrispondono ai termini di ricerca specificati se almeno un progetto rispetta questi termini, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/search-albums")
  @GET
  public Response searchAlbums(@QueryParam("q") @NotBlank
  final String query, @QueryParam("index") @Min(0) final
  Integer index, @QueryParam("limit") @Min(1) final Integer
  limit)
  ```

Contesto

- `GET /search-albums?q=${query}`
- `GET /search-albums?q=${query}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `query` deve essere specificato e non vuoto.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.
  ```

4.2.4.9. `searchArtists`

Restituisce la rappresentazione in formato JSON degli artisti musicali presenti nel catalogo della piattaforma che corrispondono ai termini di ricerca specificati se almeno un artista rispetta questi termini, altrimenti `null`.
I parametri `index` e `limit` sono opzionali e servono a specificare, rispettivamente, l'indice della lista da cui iniziare a restituire i risultati, e il numero massimo di risultati da restituire.

Interfacce

- ```
  @Path("/search-artists")
  @GET
  public Response searchArtists(@QueryParam("q") @NotBlank
  final String query, @QueryParam("index") @Min(0) final
  Integer index, @QueryParam("limit") @Min(1) final Integer
  limit)
  ```

Contesto

- `GET /search-artists?q=${query}`
- `GET /search-artists?q=${query}&index=${index}&limit=${limit}`

Condizioni d'entrata

- Il parametro `query` deve essere specificato e non vuoto.
- Se specificato, il parametro `index` deve essere maggiore o uguale a `0`.
- Se specificato, il parametro `limit` deve essere maggiore o uguale a `1`.
  ```

4.2.4.10. `updateBacklog`

Inserisce una specifica entrata nella lista d'ascolto di un utente o la rimuove se questa è già presente.

Interfacce

- ```
  @Path("/update-backlog")
  @POST
  public Response updateBacklog(@FormParam("user")
  @NotBlank final String username, @FormParam("album")
  @Min(0) final Long albumId, @Context final
  HttpServletRequest request)
  ```

Contesto

- `POST /update-backlog?user=${username}&album=${albumId}`

Condizioni d'entrata

- Il parametro `username` deve essere specificato e non vuoto.
- Il parametro `albumId` deve essere specificato e diverso da `null`.
- L'utente invocante il metodo deve essere autenticato al sistema e il suo account deve corrispondere a quello specificato dal parametro `username`.

Condizioni d'uscita

- L'entrata viene registrata o rimossa dal sistema.

Eccezioni

- L'utente specificato non viene trovato sul sistema.
- Il progetto musicale specificato non viene trovato sul catalogo del sistema.
  ```