Università degli Studi di Salerno

Corso di Ingegneria del Software

POORIFY

Object Design Document

Versione 1.1 / 13.01.2023

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

INDICE

1. INTRODUCTION 3
1.1 OBJECT DESIGN TRADE-OFFS 3
1.1.1 DESIGN PATTERNS 3
1.2 INTERFACE DOCUMENTATION GUIDELINES 4
1.2.1 NAMING CONVENTIONS 4
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 4
1.4 REFERENCES 4
2. PACKAGES 5
3. CLASS INTERFACES GLOSSARY 7
3.1 CLASS DIAGRAM TO EER DIAGRAM 7
3.2 INTERFACES SPECIFICATION 9
3.2.1 BEAN 9
3.2.2 DAO 33
3.2.3 CONTROL 43

1. INTRODUCTION

1.1 OBJECT DESIGN TRADE-OFFS

Il principale trade-off individuato in fase di System Design è Space vs Speed, ovvero garantire la fruizione di una grande quantità di contenuti in tempi rapidi.
Ovviamente un sistema distribuito scalabile deve garantire all'utenza tempi di risposta ottimali, ma per la nostra applicazione possiamo tollerare dei rallentamenti dovuti allo streaming o al download dei contenuti multimediali dal server remoto.

1.1.1 DESIGN PATTERNS

Elenchiamo di seguito i Design Pattern utilizzati per realizzare l'applicazione. In molti casi si è deciso di non attenersi all'implementazione canonica ma piuttosto di seguire la filosofia dietro il design pattern scelto.

- **COMMAND**
  Per permettere all'utente la navigazione delle pagine già visitate tramite *Navigator.java,* una struttura dati basata su due stack che conservano oggetti *Page.java,* wrapper che memorizza tipo e id del contenuto visitato.
- **STRATEGY**
  All'interno del metodo *getTracklist(Order order)* della classe *PlaylistBean.java* così da poter restituire diversi ordinamenti degli *AddedBean.java* che compongono la lista di canzoni presenti nella playlist. Il metodo crea un *TreeSet* passando al costruttore un *Comparator* diverso a seconda dell'ordinamento da realizzare in base al parametro *Order*.
- **PROXY**
  Per semplificare le query di retrieve delle canzoni all'interno di una playlist il metodo di *PlaylistDAO.java* restituisce degli *AddedBeanProxy.java* che contengono solo l'id dei contenuti desiderati. In secondo momento, nelle Servlet, si provvede a costruire i relativi *AddedBean.java* utilizzando gli altri DAO per recuperare gli oggetti veri e propri partendo dagli id dell'oggetto proxy.
- **ADAPTER**
  Per incapsulare in un unico punto la logica di upload dei file su blob container remoto Azure all'interno di *Uploader.java*, classe singleton che si interfaccia con le API del componente off-the-shelf.
- **OBSERVER**
  Per notificare l'utente di alcuni eventi, quali l'aggiunta di un brano a una playlist, tramite un pop up Javascript che può essere attivato da qualunque altro script passando come parametro il messaggio di notifica.

1.2 INTERFACE DOCUMENTATION GUIDELINES

1.2.1 NAMING CONVENTIONS

Forniamo delle linee guida per la stesura del codice durante la fase di implementazione così da poter essere consistenti e renderne più semplice la lettura sia agli sviluppatori interni al team che a terzi.

- **CLASSES**
  - I nomi delle classi devono rispettare la notazione *UpperCamelCase.*
  - Le classi Bean/DTO devono chiamarsi *NameBean.*
  - Le classi che interagiscono con il layer della persistenza devono chiamarsi *NameDAO*.
  - Le classi che modellano gli oggetti Control, ossia le nostre Servlet, devono essere denominate con un nome che ne esplicita la funzionalità offerta.
- **METHODS**
  - I nomi dei metodi devono rispettare la notazione *lowerCamelCase.*
- **PACKAGE ORGANIZATION**
  - Deve essere creato un package per ogni sottosistema.
  - Le classi che realizzano il sottosistema, principalmente Servlet, DAO e Bean, devono essere contenute nello stesso package.

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

- OCL: Object Constraint Language
- DAO: Data Access Object
- DTO: Data Transfer Object

1.4 REFERENCES

Nel corso del documento facciamo riferimento a artefatti precedenti quali il **Requirements Analysis Document** (RAD) e al **System Design Document** (SDD).

2. PACKAGES

Con riferimento alla divisione in sottosistemi riportata nel SDD elenchiamo le componenti che compongono ciascun package.

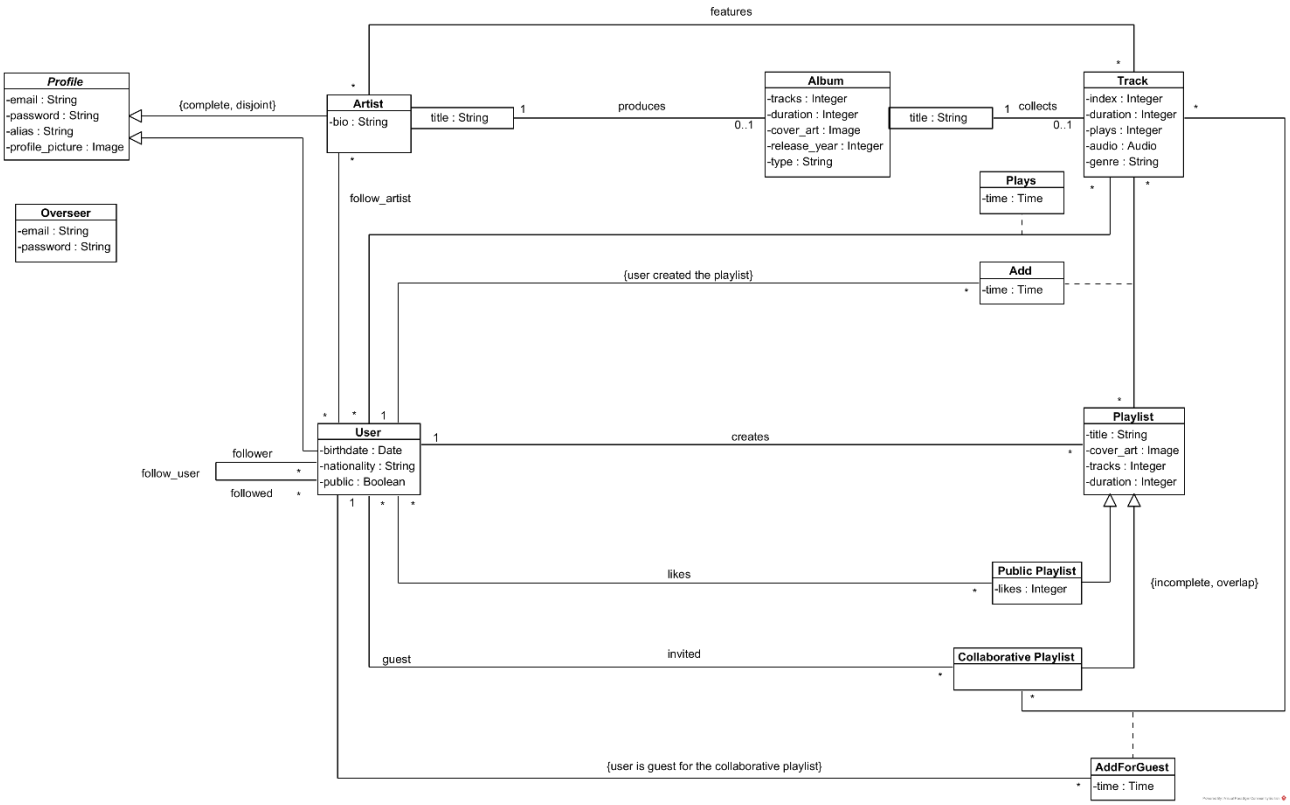
- **PROFILE**
  - ProfileBean.java
  - UserBean.java
  - ArtistBean.java
  - OverseerBean.java
  - NationBean.java
  - ProfileDAO.java
  - Login.java
  - Logout.java
  - Signup.java
  - CheckNations.java
  - CheckCredentials.java
  - CheckFollowing.java
  - GetUser.java
  - GetArtist.java
  - FollowArtist.java
  - UnfollowArtist.java
  - FollowUser.java
  - UnfollowUser.java
  - EditProfile.java
  - DeleteProfile.java
  - SearchForGuests.java
- **PLAYLIST**
  - PlaylistBean.java
  - AddedBean.java
  - AddedBeanProxy.java
  - PlaylistDAO.java
  - GetPlaylist.java
  - CreatePlaylist.java
  - DeletePlaylist.java
  - EditPlaylist.java
  - AddTrack.java
  - RemoveTrack.java
  - MakePrivate.java
  - MakePublic.java

    - MakeSingle.java
    - MakeCollaborative.java
    - LikePublicPlaylist.java
    - UnlikePublicPlaylist.java
    - AddGuest.java
    - CheckLike.java
    - ChangeOrder.java
- **ALBUM**
    - AlbumBean.java
    - AlbumDAO.java
    - GetAlbum.java
    - UploadAlbum.java
    - DeleteAlbum.java
- **TRACK**
    - TrackBean.java
    - ListeningQueue.java
    - TrackDAO.java
    - Play.java
    - PlayAlbum.java
    - PlayPlaylist.java
    - AddToQueue.java
    - Skip.java
- **NAVIGATION**
    - Page.java
    - Navigator.java
    - PrevPage.java
    - NextPage.java
    - ResultsContainer.java
    - Search.java

3. CLASS INTERFACES GLOSSARY

3.1 CLASS DIAGRAM TO EER DIAGRAM

Partendo dal Class Diagram mostrato in precedenza nel RAD e nel SDD, effettuiamo le dovute modifiche per poter mappare le classi su delle entità da rendere persistenti su database relazionale SQL.



Rimuoviamo i seguenti attributi che NON andranno memorizzati su database bensì su file system. Il razionale dietro tale scelta è discusso in dettaglio nel SDD.

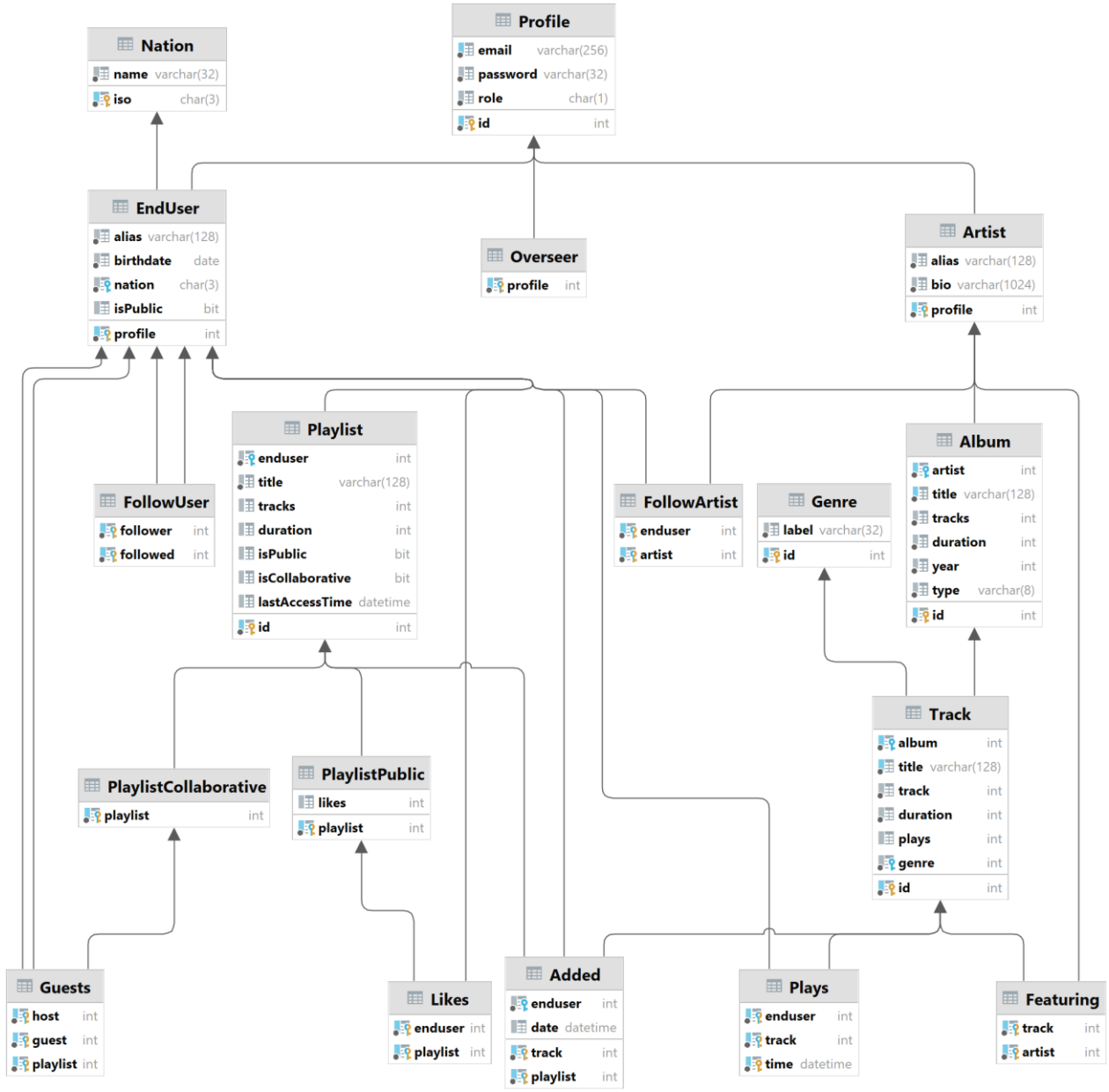
- *profile_picture* della classe **Profile**
- *cover_art* della classe **Playlist**
- *cover_art* della classe **Album**
- *audio* della classe **Track**

Decidiamo di trattare gli attributi *nationality* della classe **User** e *genre* della classe **Track** come delle entità separate ai fini della normalizzazione del modello relazionale.

Modifichiamo inoltre la gerarchia che coinvolge le classi **Profile**, **Artist** e **User** includendo anche **Overseer** per facilitare la gestione delle credenziali e in seguito del login alla piattaforma.

Dotiamo infine le entità principali di un id che funge da chiave primaria.

Questo è il diagramma ER risultante generato dal tool integrato di IntelliJ:



3.2 INTERFACES SPECIFICATION

3.2.1 BEAN

Nome Classe	ProfileBean
Modulo	Profile
Variabili di Istanza	-id: int -email: String -password: String -role: Role
Descrizione	Entità astratta che contiene le informazioni comuni a tutte le tipologie di profili utente della piattaforma.
Firme dei Metodi	+ ProfileBean(role: Role): ProfileBean + ProfileBean(id: int, email: String, password: String, role: Role): ProfileBean + getId(): int + getEmail(): String + getPassword(): String + getRole(): Role + setId(id: int): void + setEmail(email: String): void + setPassword(password: String): void + setRole(role: Role): void
Pre e Post Condizioni	• context ProfileBean::ProfileBean(role) **pre:** true **post:** self.id = 0 **and** self.email = null **and** self.password = null **and** self.role = role **and** result = self • context ProfileBean::ProfileBean(id, email, password, role) **pre:** true **post:** self.id = id **and** self.email = email **and** self.password = password **and** self.role = role **and** result = self • context ProfileBean::getId() **pre:** true

	post: result = self.id • context ProfileBean::getEmail() **pre:** true **post:** result = self.email • context ProfileBean::getPassword() **pre:** true **post:** result = self.password • context ProfileBean::getRole() **pre:** true **post:** result = self.role • context ProfileBean::setId(id) **pre:** true **post:** self.id = id • context ProfileBean::setEmail(email) **pre:** true **post:** self.email = email • context ProfileBean::setPassword(password) **pre:** true **post:** self.password = password • context ProfileBean::setRole(role) **pre:** true **post:** self.role = role
Invarianti	

Nome Classe	UserBean<<extends>>ProfileBean
Modulo	Profile
Variabili di Istanza	-alias: String -birthdate: String -nation: NationBean -isPublic: boolean -playlists: Collection<PlaylistBean> -likedPlaylists: Collection<PlaylistBean> -artists: Collection<ArtistBean> -followers: Collection<UserBean> -following: Collection<UserBean>
Descrizione	Entità che rappresenta l'utente principale della piattaforma.

Firme dei Metodi	+ UserBean(): UserBean + UserBean(id: int, email: String, password: String, alias: String, birthdate: String, nation: NationBean, isPublic: boolean): UserBean + getAlias(): String + getBirthdate(): String + getNation(): NationBean + isPublic(): boolean + getPlaylists(): Collection<PlaylistBean> + getLikedPlaylists(): Collection<PlaylistBean> + getArtists(): Collection<ArtistBean> + getFollowers(): Collection<UserBean> + getFollowing(): Collection<UserBean> + setAlias(alias: String): void + setBirthdate(birthdate: String): void + setNation(nation: NationBean): void + setPublic(aPublic: boolean): void + setPlaylists(playlists: Collection<PlaylistBean>): void + setLikedPlaylists(likedPlaylists: Collection<PlaylistBean>): void + setArtists(artists: Collection<ArtistBean>): void + setFollowers(followers: Collection<UserBean): void + setFollowing(following: Collection<UserBean): void
Pre e Post condizioni	• context UserBean::UserBean() **pre:** true **post:** self.id = 0 **and** self.email = null **and** self.password = null **and** self.role = USER **and** self.alias = null **and** self.birthdate = null **and** self.nation = null **and** self.isPublic = false **and** self.playlists = null **and** self.likedPlaylists = null **and** self.artists = null **and** self.followers = null **and** self.following = null **and** result = self • context UserBean::UserBean(id, email, password, alias, birthdate, nation, isPublic) **pre:** true **post:** self.id = id **and** self.email = email **and**

	self.password = password **and** self.role = USER **and** self.alias = alias **and** self.birthdate = birthdate **and** self.nation = nation **and** self.isPublic = isPublic **and** self.playlists = null **and** self.likedPlaylists = null **and** self.artists = null **and** self.followers = null **and** self.following = null **and** result = self • context UserBean::getAlias() **pre:** true **post:** result = self.alias • context UserBean::getBirthdate() **pre:** true **post:** result = self.birthdate • context UserBean::getNation() **pre:** true **post:** result = self.nation • context UserBean::isPublic() **pre:** true **post:** result = self.isPublic • context UserBean::getPlaylists() **pre:** true **post:** result = self.playlists • context UserBean::getLikedPlaylists() **pre:** true **post:** result = self.likedPlaylists • context UserBean::getArtists() **pre:** true **post:** result = self.artists • context UserBean::getFollowers() **pre:** true **post:** result= self.followers • context UserBean::getFollowing() **pre:** true **post:** result = self.following • context UserBean::setAlias(alias) **pre:** true **post:** self.alias = alias • context UserBean::setBirthdate(birthdate) **pre:** true **post:** self.birthdate = birthdate • context UserBean::setNation(nation)

	pre: true **post:** self.nation = nation • context UserBean::setPublic(aPublic) **pre:** true **post:** self.isPublic = aPublic • context UserBean::setPlaylists(playlists) **pre:** true **post:** self.playlists = playlists • context UserBean::setLikedPlaylists(likedPlaylists) **pre:** true **post:** self.likedPlaylists = likedPlaylists • context UserBean::setArtists(artists) **pre:** true **post:** self.artists = artists • context UserBean::setFollowers(followers) **pre:** true **post:** self.followers = followers • context UserBean::setFollowing(following) **pre:** true **post:** self.following = following
Invarianti	○ context ProfileBean **inv:** self.playlists->forAll(p: PlaylistBean | p.host = self **or** p.guests->includes(self))

Nome Classe	ArtistBean <<extends>> ProfileBean
Modulo	Profile
Variabili di Istanza	-alias: String -bio: String -albums: Collection<AlbumBean> -topTracks: Collection<TrackBean>
Descrizione	Entità che rappresenta gli artisti che possono caricare contenuti sulla piattaforma.
Firme dei Metodi	+ ArtistBean(): ArtistBean + ArtistBean(id: int, email: String, password: String, alias: String, bio: String): ArtistBean + getAlias(): String

	+ getBio(): String + getAlbums(): Collection<AlbumBean> + getTopTracks(): Collection<TrackBean> + setAlias(alias: String): void + setBio(bio: String): void + setAlbums(albums: Collection<AlbumBean>): void + setTopTracks(topTracks: Collection<TrackBean): void
Pre e Post Condizioni	• context ArtistBean()::ArtistBean **pre:** true **post**: self.id = 0 **and** self.email = null **and** self.password = null **and** self.role = ARTIST **and** self.alias = null **and** self.bio = null **and** self.albums = null **and** self.topTracks = null **and** result = self • context ArtistBean()::ArtistBean(id, email, password, alias, bio) **pre:** true **post**: self.id = id **and** self.email = email **and** self.password = password **and** self.role = ARTIST **and** self.alias = alias **and** self.bio = bio **and** self.albums = null **and** self.topTracks = null **and** result = self • context ArtistBean()::getAlias() **pre:** true **post:** result = self.alias • context ArtistBean()::getBio() **pre:** true **post:** result = self.bio • context ArtistBean()::getAlbums() **pre:** true **post:** result = self.albums • context ArtistBean()::getTopTracks() **pre:** true **post:** result = self.topTracks • context ArtistBean()::setAlias(alias) **pre:** true **post:** self.alias = alias • context ArtistBean()::setBio(bio: String) **pre:** true **post:** self.bio = bio • context ArtistBean()::setAlbums(albums)

	pre: true **post:** self.albums = albums • context ArtistBean()::setTopTracks(topTracks) **pre:** true **post:** self.topTracks = topTracks
Invarianti	○ context ArtistBean **inv:** self.albums->forAll(a: AlbumBean | a.artist = self) ○ context ArtistBean **inv:** self.topTracks->forAll(t: TrackBean | t.album.artist = self **or** t.featuring->includes(self))

Nome Classe	OverseerBean<<extends>>ProfileBean
Modulo	Profile
Variabili di Istanza	
Descrizione	Entità che rappresenta i supervisori della piattaforma
Firme dei Metodi	+ OverseerBean(): OverseerBean + OverseerBean(id: int , email: String, password: String): OverseerBean
Pre e Post condizioni	• context OverseerBean::OverseerBean() **pre:** true **post:** self.id = 0 **and** self.email = null **and** self.password = null **and** self.role = OVERSEER **and** result = self • context OverseerBean::OverseerBean(id, email, password) **pre:** true **post:** self.id = id **and** self.email = email **and** self.password = password **and** self.role = OVERSEER **and** result = self
Invarianti	

Classe	NationBean
Modulo	Profile
Variabili di Istanza	- iso: String - name: String

Descrizione	Entità che rappresenta la nazionalità di un utente.
Firma dei Metodi	+ NationBean(): NationBean + NationBean(iso: String, name: String): NationBean + getIso(): String + getName(): String + setIso(iso: String): void + setName(name: String): void
Pre e Post Condizioni	• context NationBean::NationBean() **pre:** true **post:** self.iso = 0 **and** self.name = null **and** result = self • context NationBean::NationBean(iso, name) **pre:** true **post:** self.iso = iso **and** self.name = name **and** result = self • context NationBean::getIso() **pre:** true **post:** result = self.iso • context NationBean::getName() **pre:** true **post:** result = self.name • context NationBean::setIso(iso) **pre:** true **post:** self.iso = iso • context NationBean::setName(name) **pre:** true **post:** self.name = name
Invarianti	○ context NationBean **inv: if** self.iso <> null **then** self.iso.size = 3 **endif**

Nome Classe	AlbumBean
Modulo	Album
Variabili di sistema	-id: int -title: String -tracks: int

	-duration: int -year: int -type: String -artist: ArtistBean -tracklist: Collection<TrackBean>
Descrizione	Entità che rappresenta un album caricato da un artista sulla piattaforma
Firme dei Metodi	+ AlbumBean(): AlbumBean + AlbumBean(id: int, title: String, tracks: int, duration: int, year: int, type: String): AlbumBean + getId(): int + getTitle(): String + getTracks(): int + getDuration(): int + getYear(): int + getType(): String + getArtist(): ArtistBean + getTracklist(): Collection<TrackBean> + setId(id: int): void + setTitle(title: String): void + setTracks(tracks: int): void + setDuration(duration: int): void + setYear(year: int): void + setType(type: String): void + setArtist(artist: ArtistBean): void + setTracklist(tracklist : Collection<TrackBean>): void
Pre e Post-condizioni	• context AlbumBean()::AlbumBean() **pre:** true **post**: self.id = 0 **and** self.title = null **and** self.tracks = 0 **and** self.duration = 0 **and** self.year = 0 **and** self.type = null **and** self.artist = null **and** self.tracklist = null **and** result = self • context AlbumBean()::AlbumBean(id: int, title: String, tracks: int, duration: int, year: int, type: String) **pre:** true **post**: self.id = id **and** self.title = title **and** self.tracks = tracks **and** self.duration = duration **and** self.year

	= year **and** self.type = type **and** self.artist = null **and** self.tracklist = null **and** result = self • context AlbumBean::getId() **pre:** true **post**: result = self.id • context AlbumBean::getTitle() **pre:** true **post**: result = self.title • context AlbumBean::getTracks() **pre:** true **post**: result = self.tracks • context AlbumBean::getDuration() **pre:** true **post**: result = self.duration • context AlbumBean::getYear() **pre:** true **post**: result = self.year • context AlbumBean::getType() **pre:** true **post**: result = self.type • context AlbumBean::getArtist() **pre:** true **post**: result = self.artist • context AlbumBean::getTracklist() **pre:** true **post**: result = self.tracklist • context AlbumBean::setId(id) **pre:** true **post**: self.id = id • context AlbumBean::setTitle(title) **pre:** true **post**: self.title = title • context AlbumBean::setTracks(tracks) **pre:** true **post**: self.tracks = tracks • context AlbumBean::setDuration(duration) **pre:** true **post**: self.duration = duration • context AlbumBean::setYear(year) **pre:** true **post**: self.year = year

	• context AlbumBean::setType(type: String) **pre:** true **post**: self.type = type • context AlbumBean::setArtist(artist) **pre:** true **post**: self.artist = artist • context AlbumBean::setTracklist(tracklist): **pre:** true **post**: self.tracklist = tracklist
Invarianti	○ context AlbumBean **inv:** self.tracklist->forAll(t: TrackBean | t.album = self) ○ context AlbumBean **inv:** self.artist.albums->includes(self)

Nome Classe	PlaylistBean
Modulo	Playlist
Variabili di Istanza	-id: int -host: UserBean -title: String -tracks: int -duration: int -likes: int -lastAccessTime: String -isPublic: Boolean -isCollaborative: Boolean -guests: Collection<UserBean> -tracklist: Collection<AddedBean>
Descrizione	Entità che rappresenta una playlist realizzata da un utente
Firme dei Metodi	+ PlaylistBean(): PlaylistBean + PlaylistBean(id: int, title: String, tracks: int, duration: int, isPublic: boolean, isCollaborative: boolean, lastAccessTime: String) : PlaylistBean + getId():int + getTitle(): String + getTracks(): int

	+ getDuration(): int + isPublic(): boolean + getLikes(): int + isCollaborative(): boolean + getLastAccessTime(): String + getHost(): UserBean + getGuests(): Collection<UserBean> + getTracklist(): Collection<AddedBean> + getTracklist(order: Order): Collection<AddedBean> + setId(id: int): void + setTitle(title: String): void + setTracks(tracks: int): void + setDuration(duration : int) : void + setPublic(aPublic: boolean): void + setLikes(likes: int): void + setCollaborative(collaborative: boolean): void + setLastAccessTime(lastAccessTime: String): void + setHost(host: UserBean): void + setGuests(guests: Collection<UserBean>): void + setTracklist(tracklist: Collection<AddedBean>): void
Pre e Post condizioni	• context PlaylistBean::PlaylistBean() **pre:** true **post:** self.id = 0 **and** self.title = null **and** self.tracks = 0 **and** self.duration = 0 **and** self.isPublic = false **and** self.isCollaborative = false **and** self.lastAccessTime = null **and** self.host = null **and** self.guests = null **and** self.tracklist = null **and** result = self • context PlaylistBean::PlaylistBean(id, title, tracks, duration, isPublic, isCollaborative, lastAccessTime) **pre:** true **post:** self.id = id **and** self.title = title **and** self.tracks = tracks **and** self.duration = duration **and** self.isPublic = isPublic **and** self.isCollaborative = isCollaborative **and** self.lastAccessTime = lastAccessTime **and** self.host = null **and** self.guests = null **and** self.tracklist = null **and** result = self • context PlaylistBean::getId()

	pre: true **post**: result = self.id • context PlaylistBean::getTitle() **pre:** true **post**: result = self.title • context PlaylistBean::getTracks() **pre:** true **post**: result = self.tracks • context PlaylistBean::getDuration() **pre:** true **post**: result = self.duration • context PlaylistBean::isPublic() **pre:** true **post**: result = self.isPublic • context PlaylistBean::getLikes() **pre:** true **post**: result = self.likes • context PlaylistBean::isCollaborative() **pre:** true **post**: result = self.isCollaborative • context PlaylistBean::getLastAccessTime() **pre:** true **post**: result = self.lastAccessTime • context PlaylistBean::getHost() **pre:** true **post**: result = self.host • context PlaylistBean::getGuests() **pre:** true **post**: result = self.guests • context PlaylistBean::getTracklist() **pre:** true **post**: result = self.tracklist • context PlaylistBean::getTracklist(order) **pre:** true **post**: **if** order = DATE **then** result = self.tracklist->sortedBy(t | t.date) **else if** order = TITLE **then** result = self.tracklist->sortedBy(t | t.track.title) **else if** order = ARTIST **then** result = self.tracklist->sortedBy(t | t.artist.alias) **else if** order = ALBUM **then** result = self.tracklist->sortedBy(t | t.album.title) **else if** order = DURATION **then**

	result = self.tracklist->sortedBy(t | t.track.duration) **endif** • context PlaylistBean::setId(id) **pre:** true **post**: self.id = id • context PlaylistBean::setTitle(title) **pre:** true **post**: self.title = title • context PlaylistBean::setTracks(tracks) **pre:** true **post**: self.tracks = tracks • context PlaylistBean::setDuration(duration) **pre:** true **post**: self.duration = duration • context PlaylistBean::setPublic(aPublic) **pre:** true **post**: self.isPublic = aPublic • context PlaylistBean::setLikes(likes) **pre:** true **post**: self.likes = likes • context PlaylistBean::setCollaborative (collaborative) **pre:** true **post**: self.isCollaborative = collaborative • context PlaylistBean::setLastAccessTime (lastAccessTime) **pre:** true **post**: self.lastAccesTime = lastAccessTime • context PlaylistBean::setHost(host) **pre:** true **post**: self.host = host • context PlaylistBean::setGuests(guests) **pre:** true **post**: self.guests = guests • context PlaylistBean::setTracklist(tracklist) **pre:** true **post:** self.tracklist = tracklist
Invarianti	○ context PlaylistBean **inv:** self.tracklist->forAll(a: AddedBean | a.playlist = self)

Nome Classe	AddedBean
Modulo	Playlist
Variabili di Istanza	-user: UserBean -track: TrackBean -playlist: PlaylistBean -date: String
Descrizione	Entità che rappresenta un brano contenuto in una playlist in seguito all'aggiunta da parte di un utente
Firme dei Metodi	+ AddedBean(): AddedBean + AddedBean(user: UserBean, playlist: PlaylistBean, date: String, track: TrackBean): AddedBean + getUser(): UserBean + getTrack(): TrackBean + getPlaylist(): PlaylistBean + getDate(): String + setUser(user: UserBean): void + setTrack(track: TrackBean): void + setPlaylist(playlist: PlaylistBean): void + setDate(date: String): void
Pre e Post condizioni	• context AddedBean::AddedBean() **pre:** true **post:** self.user = null **and** self.playlist = null **and** self.date = null **and** self.track = null **and** result = self • context AddedBean::AddedBean(user, playlist, date, track) **pre:** true **post:** self.user = user **and** self.playlist = playlist **and** self.date = date **and** self.track = track **and** result = self • context AddedBean::getUser() **pre:** true **post:** result = self.user • context AddedBean::getTrack() **pre:** true

	post: result = self.track • context AddedBean::getPlaylist() **pre:** true **post:** self = self.playlist • context AddedBean::getDate() **pre:** true **post:** result = self.date • context AddedBean::setUser(user) **pre:** true **post:** self.user = user • context AddedBean::setTrack(track) **pre:** true **post:** self.track = track • context AddedBean::setPlaylist(playlist) **pre:** true **post:** self.playlist = playlist • context AddedBean::setDate(date) **pre:** true **post:** self.date = date
Invarianti	○ context AddedBean **inv:** self.playlist.tracklist->includes(self) ○ context AddedBean **inv:** self.playlist.host = self.user **or** self.playlist.guests->includes(self.user) ○ context AddedBean **inv:** self.user.playlists->includes(self.playlist)

Nome Classe	AddedBeanProxy
Modulo	Playlist
Variabili di Istanza	-user: int -track: int -playlist: int -date: String
Descrizione	Entità che rappresenta un brano contenuto in una playlist in seguito all'aggiunta da parte di un utente
Firme dei Metodi	+ AddedBeanProxy(): AddedBeanProxy

	+ AddedBeanProxy(user: int, playlist: int, date: String, track: int): AddedBeanProxy + getUser(): int + getTrack(): int + getPlaylist(): PlaylistBean + getDate(): String + setUser(user: int): void + setTrack(track: int): void + setPlaylist(playlist: int): void + setDate(date: String): void
Pre e Post condizioni	• context AddedBeanProxy:: AddedBeanProxy() **pre:** true **post:** self.user = 0 **and** self.playlist = 0 **and** self.date = null **and** self.track = 0 **and** result = self • context AddedBeanProxy:: AddedBeanProxy(user, playlist, date, track) **pre:** true **post:** self.user = user **and** self.playlist = playlist **and** self.date = date **and** self.track = track **and** result = self • context AddedBeanProxy::getUser() **pre:** true **post:** result = self.user • context AddedBeanProxy::getTrack() **pre:** true **post:** result = self.track • context AddedBeanProxy::getPlaylist() **pre:** true **post:** self = self.playlist • context AddedBeanProxy::getDate() **pre:** true **post:** result = self.date • context AddedBeanProxy::setUser(user) **pre:** true **post:** self.user = user • context AddedBeanProxy::setTrack(track) **pre:** true **post:** self.track = track • context AddedBeanProxy::setPlaylist(playlist) **pre:** true

	post: self.playlist = playlist • context AddedBeanProxy::setDate(date) **pre:** true **post:** self.date = date
Invarianti	

Nome Classe	TrackBean
Modulo	Track
Variabili di Istanza	-id: int -title: String -index: int -duration: int -plays: int -genre: String -album: AlbumBean -featuring: Collection<ArtistBean>
Descrizione	Entità che rappresenta un brano caricato sulla piattaforma
Firme dei Metodi	+ TrackBean(): TrackBean + TrackBean(id: int , title: String, index: int, duration: int, plays: int, genre : String): TrackBean + getId(): int + getTitle(): String + getIndex(): int + getDuration(): int + getPlays(): int + getGenre(): String + getAlbum(): AlbumBean + getFeaturing(): Collection<ArtistBean> + setId(id: int): void + setTitle(title: String): void + setIndex(index: int): void + setDuration(duration: int) : int + setPlays(plays: int): void + setGenre(genre : String) : void + setAlbum(album: AlbumBean): void

	+ setFeaturing(featuring: Collection<ArtistBean>): void
Pre e Post condizioni	• context TrackBean()::TrackBean() **pre:** true **post:** self.id = 0 **and** self.title = null **and** self.index = 0 **and** self.duration = 0 **and** self.plays = 0 **and** self.genre = null **and** self.album = null **and** self.featuring = null **and** result = self • context TrackBean()::TrackBean(id, title, index, duration, plays, genre) **pre:** true **post:** self.id = id **and** self.title = title **and** self.index = index **and** self.duration = duration **and** self.plays = plays **and** self.genre = genre **and** self.album = null **and** self.featuring = null **and** result = self • context TrackBean()::getId() **pre:** true **post:** result = self.id • context TrackBean()::getTitle() **pre:** true **post:** result = self.title • context TrackBean()::getIndex() **pre:** true **post:** result = self.index • context TrackBean()::getDuration() **pre:** true **post:** result = self.duration • context TrackBean()::getPlays() **pre:** true **post:** result = self.plays • context TrackBean()::getGenre() **pre:** true **post:** result = self.genre • context TrackBean()::getAlbum() **pre:** true **post:** result = self.album • context TrackBean()::getFeaturing() **pre:** true **post:** result = self.featuring

	• context TrackBean()::setId(id) **pre:** true **post:** self.id = id • context TrackBean()::setTitle(title) **pre:** true **post:** self.title = title • context TrackBean()::setIndex(index) **pre:** true **post:** self.index = index • context TrackBean()::setDuration(duration) **pre:** true **post:** self.duration = duration • context TrackBean()::setPlays(plays) **pre:** true **post:** self.plays = plays • context TrackBean()::setGenre(genre) **pre:** true **post:** self.genre = genre • context TrackBean()::setAlbum(album) **pre:** true **post:** self.album = album • context TrackBean()::setFeaturing(featuring) **pre:** true **post:** self.featuring = featuring
Invarianti	○ context TrackBean **inv:** self.album.tracklist ->includes(self)

Classe	ListeningQueue
Modulo	Track
Variabili di Istanza	- userQueue: Queue<Integer> - autoQueue: Queue<Integer>
Descrizione	Entità che rappresenta la coda di ascolti dell'utente
Firma dei Metodi	+ ListeningQueue(): ListeningQueue + addToUserQueue(int id): void + addToAutoQueue(int id): void + clearAutoQueue(): void

	+ poll(): int
Pre e Post Condizioni	• context ListeningQueue:: ListeningQueue () **pre:** true **post:** self.userQueue = ArrayDeque() **and** self.autoQueue = ArrayDeque() • context Page::addToUserQueue(int id) **pre:** true **post:** addToUserQueue->last() = id • context Page::addToAutoQueue(int id) **pre:** true **post:** addToAutoQueue->last() = id • context Page::clearAutoQueue() **pre:** true **post:** addToAutoQueue->size() = 0 • context Page::poll() **pre:** true **post: if** userQueue.size() > 0 **then** result = userQueue->poll() **else if** autoQueue.size() > 0 **then** result = autoQueue->poll() **else** result = 0 **endif**
Invarianti	

Classe	Page
Modulo	Navigation
Variabili di Istanza	- id: int - type: Type
Descrizione	Entità che rappresenta una pagina visitata dall'utente
Firma dei Metodi	+ Page(id: int, type: Type): Page + getId(): int + getType(): Type
Pre e Post Condizioni	• context Page::Page(id: int, type: Type) **pre:** true **post:** self.id = id **and** self.type = type **and** result = self • context Page::getId() **pre:** true

	post: result = self.id • context Page::getType() **pre:** true **post:** result = self.type
Invarianti	

Classe	Navigator
Modulo	Navigation
Variabili di Istanza	- prev: Stack<Page> - current: Page - next: Stack<Page>
Descrizione	Entità che rappresenta una pagina visitata dall'utente
Firma dei Metodi	+ Navigator(): Navigator + getCurrent(): Page + setCurrent(page: Page): void + save(page: Page): void + prev(): Page + next(): Page + hasPrev(): boolean + hasNext(): boolean
Pre e Post Condizioni	• context Navigator::Navigator() **pre:** true **post:** self.prev = null **and** self.current = null **and** result = self • context Navigator::getCurrent() **pre:** true **post:** result = self.current • context Navigator::setCurrent(page) **pre:** true **post:** self.current = page • context Navigator::save(page) **pre:** true **post:** self.prev->top() = self.current **and** self.next.size = 0 • context Navigator::prev() **pre:** true

	post: self.next->top() = self@pre.current **and** self.current = self.prev->pop() **and** result = self.current • context Navigator::next() **pre:** true **post:** self.prev->top() = self@pre.current **and** self.current = self.next->pop() **and** result = self.current • context Navigator::hasPrev() **pre:** true **post: if** self.prev.size > 0 **then** result = true **else** result = false **endif** • context Navigator::hasNext() **pre:** true **post: if** self.next.size > 0 **then** result = true **else** result = false **endif**
Invarianti	

Nome Classe	ResultsContainer
Modulo	Navigation
Variabili di Istanza	-users: Collection<UserBean> -artists: Collection<ArtistBean> -tracks: Collection<TrackBean> -albums: Collection<AlbumBean> -playlists: Collection<PlaylistBean>
Descrizione	Entità wrapper per i risultati forniti dalla ricerca
Firme dei Metodi	+ ResultsContainer():ResultsContainer + empty(): boolean + getUsers(): Collection<UserBean> + getArtists(): Collection<ArtistBean> + getTracks(): Collection<TrackBean> + getAlbums(): Collection<AlbumBean> + getPlaylists(): Collection<PlaylistBean> + setUsers(users: Collection<UserBean>): void + setArtists(artists: Collection<ArtistBean>): void + setTracks(tracks: Collection<TrackBean>): void + setAlbums(albums: Collection<AlbumBean>): void

	+ setPlaylists(playlists: Collection<PlaylistBean>): void
Pre e Post condizioni	• context ResultsContainer::ResultsContainer() **pre:** true **post:** self.users = self.artists = self.tracks = self.albums = self.playlists = null **and** result = self • context ResultsContainer::empty() **pre:** true **post: if** self.users = self.artists = self.tracks = self.albums = self.playlists = null **then** result = true **else** result = false **endif** • context ResultsContainer::getUsers() **pre:** true **post:** result = self.users • context ResultsContainer::getArtists() **pre:** true **post:** result = self.artists • context ResultsContainer::getTracks() **pre:** true **post:** result = self.tracks • context ResultsContainer::getAlbums() **pre:** true **post:** result = self.albums • context ResultsContainer::getPlaylists() **pre:** true **post:** result = self.playlists • context ResultsContainer::setUsers(users) **pre:** true **post:** self.users = users • context ResultsContainer::setArtists(artists) **pre:** true **post:** self.artists = artists • context ResultsContainer::setTracks(tracks) **pre:** true **post:** self.tracks = tracks • context ResultsContainer::setAlbums(albums) **pre:** true **post:** self.albums = albums • context ResultsContainer::setPlaylists(playlists)

	pre: true **post:** self.playlists = playlists
Invarianti	

3.2.2 DAO

Consideriamo il database come un insieme di set/collections con i quali i DAO interagiscono:

- playlists
- added
- tracks
- likes
- nations
- followings
- profiles
- users
- artists
- albums
- featurings
- plays

La maggioranza dei metodi dei DAO restituiscono un booleano che funge da acknowledgement per comunicare agli oggetti Control, le Servlet, l'esito dell'operazione.

Nome Classe	PlaylistDAO
Modulo	Playlist
Variabili di Istanza	-connection: Connection
Descrizione	Bean Singleton che gestisce la persistenza relativa al modulo playlist
Firma dei Metodi	+ PlaylistDAO(connection: Connection): PlaylistDAO + get(id: int): PlaylistBean + add(user: int, title: String): boolean + getLastFromUser(user: int): int + getFromUser(id: int): Collection<Playlist> + getFromLikes(id: int): Collection<Playlist> + getPublicFromUser(id: int): Collection<Playlist>

	+ getAdded(id: int): Collection<AddedProxyBean> + getLikes(id: int): int + addTrackToPlaylist(user: int, track: int, playlist: int): boolean + removeTrackFromPlaylist(track: int, playlist: int): boolean + remove(id: int): boolean + like(user: int, playlist: int): boolean + unlike(user: int, playlist: int): boolean + checkLike(user: int, playlist: int): boolean + setPublic(id: int): boolean + setPrivate(id: int): boolean + setCollaborative(id: int): boolean + setSingle(id: int): boolean + getCountFromUser(id: int): int + changeTitle(id: int, title: String): boolean + addGuest(host: int, guest: int, playlist: int): boolean
Pre e Post condizioni	• context PlaylistDAO::PlaylistDAO(connection) **pre:** true **post:** self.connection = connection • context PlaylistDAO::get(id: int) **pre:** true **post:** result = playlists->select(p | p.id = id) • context PlaylistDAO::add(user: int, title: String) **pre:** true **post:** playlists->exists(p | p.host = user **and** p.title = title) • context PlaylistDAO:: getLastFromUser(user: int) **pre:** true **post: let** playlist: PlaylistBean = playlists->select(p | p.host = user **and** playlists->forAll(y | y.host = user **and** p.id >= y.id)) result = playlist.id • context PlaylistDAO::getFromUser(id: int) **pre:** true **post:** result = playlists->select(p | p.host = id) • context PlaylistDAO::getFromLikes(id: int) **pre:** true

	post: result = playlists->select(p | likes->exists(l | l.user = id **and** l.playlist = p.id) • context PlaylistDAO::getPublicFromUser(id: int) **pre:** true **post:** result = playlists->select(p | p.host = id **and** p.isPublic = true) • context PlaylistDAO::getAdded(id: int) **pre:** true **post:** result = added->select(a | a.playlist = id) • context PlaylistDAO::getAdded(id: int) **pre:** true **post:** result = added->select(a | a.playlist = id) • context PlaylistDAO::getLikes(id: int) **pre:** true **post:** result = likes->count(l | l.playlist = id) • context PlaylistDAO::addTrackToPlaylist(user: int, track: int, playlist: int) **pre:** true **post:** added->exists(a | a.user = user **and** a.track = track **and** a.playlist = playlist) = true • context PlaylistDAO::removeTrackFromPlaylist(track: int, playlist: int) **pre:** true **post: not** added->exists(a | a.user = user **and** a.track = track **and** a.playlist = playlist) • context PlaylistDAO::remove(id: int) **pre:** true **post: not** playlists->exists(p | p.id = id) • context PlaylistDAO::like(user: int, playlist: int) **pre:** true **post:** likes->exists(l | l.user = user **and** l.playlist = playlist) • context PlaylistDAO::unlike(user: int, playlist: int) **pre:** true **post: not** likes->exists(l | l.user = user **and** l.playlist = playlist) • context PlaylistDAO::checkLike(user: int, playlist: int) **pre:** true

	post: result = likes->exists(l | l.user = user **and** l.playlist = playlist) • context PlaylistDAO::setPublic(id: int) **pre:** true **post:** playlists->exists(p | p.id = id **and** p.isPublic) • context PlaylistDAO::setPrivate(id: int) **pre:** true **post:** playlists->exists(p | p.id = id **and not** p.isPublic) • context PlaylistDAO::setCollaborative(id: int) **pre:** true **post:** playlists->exists(p | p.id = id **and** p.isCollaborative) • context PlaylistDAO::setSingle(id: int) **pre:** true **post:** playlists->exists(p | p.id = id **and not** p.isCollaborative) • context PlaylistDAO::getCountFromUser(id: int) **pre:** true **post:** result = playlists->count(p | p.host = id) • context PlaylistDAO::changeTitle(id: int, title: String) **pre:** true • **post:** playlists->exists(p | p.id = id **and** p.title = title) • context PlaylistDAO::addGuest(host: int, guest: int, playlist: int) **pre:** true **post:** guests->exists(g | g.host = host **and** g.guest = guest **and** g.playlist = playlist)
Invarianti	

Nome classe	AlbumDAO
Modulo	Album
Variabili di Istanza	-connection: Connection

Descrizione	Bean Singleton che gestisce la persistenza relativa al modulo album
Firma dei metodi	+ AlbumDAO(connection: Connection): AlbumDAO + get(id: Int): AlbumBean + getFromArtist(id: Int): Collection<AlbumBean> + getFromTrack(id: Int): AlbumBean + add(artist: int, title: String, tracks: int, duration: int, year: int, type: String): boolean + get(artist: int, title: String): int + remove(id: int): Void
Pre e Post Condizioni	• context AlbumDAO::AlbumDAO(connection) **pre:** true **post:** self.connection = connection • context AlbumDAO::get(id) **pre:** true **post:** result = albums→select(a | a.id = id) • context AlbumDAO::getFromArtist(id) **pre:** true **post:** result = albums→select(a | a.artist = id) • context AlbumDAO::getFromTrack(id) **pre:** true **post:** result = albums→select(a | a.tracklist->exists(t | t.id = id)) • context AlbumDAO::add(artist, title, tracks, duration, year, type) **pre:** true **post:** albums→exists(a | a.artist = artist **and** a.title = title **and** a.tracks = tracks **and** a.duration = duration **and** a.year = year **and** a.type = type) • context AlbumDAO::get(artist, title) **pre:** true **post:** result = albums→select(a | a.id **where** a.artist = artist **and** a.title = title) • Context AlbumDAO::remove(id) **pre**: true **post: not** albums->exists(a | a.id = id)

Invarianti	

Nome classe	ProfileDAO
Modulo	Profile
Variabili di Istanza	-connection: Connection
Descrizione	Bean Singleton che gestisce la persistenza relativa al modulo profile
Firma dei metodi	+ ProfileDAO(connection: Connection): ProfileDAO + get(id: int): ProfileBean + get(email: String, password: String): ProfileBean + get(id: int, role: String): ProfileBean + check(email: String, password: String): boolean + getAllNations(): Collection<NationBean> + getFollowersFromUser(id: int): Collection<UserBean> + getFollowingFromUser(id: int): Collection<UserBean> + getArtistsFromUser(id: int): Collection<ArtistBean> + getFeaturingFromTrack(id: int): Collection<ArtistBean> + getFromAlbum(id: int): ArtistBean + getFromTrack(id: int): ArtistBean + getHostFromPlaylist(id: int): UserBean + getGuestsFromPlaylist(id: int): Collection<UserBean> + add(user: UserBean): void + add(artist: ArtistBean): void + add(overseer: OverseerBean): void + followArtist(user: int, artist: int): boolean + unfollowArtist(user: int, artist: int): boolean + followUser(follower: int, followed: int): boolean + unfollowUser(follower: int, followed: int): boolean + changeAlias(id: int, alias: String): void + changePassword(id: int, password: String): void + changePublic(id: int, isPublic: Boolean): void

	+ changeBio(id: int, bio: String): void + checkFollowing(user: int, followed: int): Boolean + removeUser(id: int): void + removeArtist(id: int): void
Pre e Post Condizioni	• context ProfileDAO:: ProfileDAO (connection) **pre:** true **post:** self.connection = connection • context ProfileDAO::get(id) **pre:** true **post:** result = profiles→select(p | p.id = id) • context ProfileDAO::get(email, password) **pre:** true **post:** result = profiles→select(p | p.email = email **and** p.password = password) • context ProfileDAO::get(id, role) **pre:** true **post:** result = profiles→select(p | p.id = id **and** p.role = role) • context ProfileDAO::check(email, password) **pre:** true **post:** result = profiles→exists(p | p.email = email **and** p.password = password) • context ProfileDAO::getAllNations() **pre:** true **post:** result = nations • context ProfileDAO::getFollowersFromUser(id: int) **pre:** true **post:** result = users->select(u | followings->exists(f | f.follower = u.id **and** f.followed = id)) • context ProfileDAO::getFollowingFromUser(id: int) **pre:** true **post:** result = users->select(u | followings->exists(f | f.follower = id **and** f.followed = u.id)) • context ProfileDAO::getArtistsFromUser(id: int) **pre:** true **post:** result = artists->select(a | followings->exists(f | f.follower = id **and** f.followed = a.id)) • context ProfileDAO::getFeaturingFromTrack(id: int) **pre:** true

	post: result = artists->select(a | featurings->exists(f | f.artist = a.id **and** f.track = id)) • context ProfileDAO::getFromAlbum(id: int) **pre:** true **post:** result = artists->select(a | albums->exists(l | l.artist = a.id **and** l.id = id)) • context ProfileDAO::getFromTrack(id: int) **pre:** true **post:** result = artists->select(a | tracks->exists(t | t.artist = a.id **and** t.id = id)) • context ProfileDAO::getHostFromPlaylist(id: int) **pre:** true **post:** result = users->select(u | playlists->exists(p | p.host = u.id **and** p.id = id)) • context ProfileDAO::getGuestFromPlaylist(id: int) **pre:** true **post:** result = users->select(u | playlists->exists(p | p.id = id **and** p.guests->exists(g | g.id = u.id))) • context ProfileDAO::add(user: UserBean) **pre:** true **post:** users->exists(u | u = user) • context ProfileDAO::add(artist: ArtistBean) **pre:** true **post:** artists->exists(a | a = artist) • context ProfileDAO::add(overseer: OverseerBean) **pre:** true **post:** overseers->exists(o | o = overseer) • context ProfileDAO::followArtist(user: int, artist: int) **pre:** true **post:** followings->exists(f | f.follower = user **and** f.followed = artist) • context ProfileDAO::unfollowArtist(user: int, artist: int) **pre:** true **post: not** followings->exists(f | f.follower = user **and** f.followed = artist) • context ProfileDAO::followUser(follower: int, followed: int) **pre:** true

	post: followings->exists(f | f.follower = follower **and** f.followed = followed) • context ProfileDAO::unfollowUser(follower: int, followed: int) **pre:** true **post: not** followings->exists(f | f.follower = follower **and** f.followed = followed) • context ProfileDAO::changeAlias(id: int, alias: String) **pre:** true **post:** users->exists(u | u.id = id **and** u.alias = alias) **or** artists->exists(a | a.id = id **and** a.alias = alias) • context ProfileDAO::changePassword(id: int, password: String) **pre:** true **post:** profiles->exists(p | p.id = id **and** p.password = password) • context ProfileDAO::changePublic(id: int, isPublic: boolean) **pre:** true **post:** users->exists(u | u.id = id **and** u.isPublic = isPublic) • context ProfileDAO::changeBio(id: int, bio: String) **pre:** true **post:** artists->exists(a | a.id = id **and** a.bio = bio) • context ProfileDAO::checkFollowing(user: int, followed: int) **pre:** true **post:** result = followings->exists(f | f.follower = user **and** f.followed = followed) • context ProfileDAO::removeUser(id: int) **pre:** true **post: not** users->exists(u | u.id = id) • context ProfileDAO::removeArtist(id: int) **pre:** true **post: not** artists->exists(a | a.id = id)
Invarianti	

Nome classe	TrackDAO
Modulo	Track
Variabili di Istanza	-connection: Connection
Descrizione	Bean Singleton che gestisce la persistenza relativa al modulo track
Firma dei metodi	+ TrackDAO(connection: Connection): TrackDAO + get(id: int): TrackBean + getFromAlbum(id: int): Collection<TrackBean> + getTopFiveFromArtist(id: int): Collection<TrackBean> + addPlay(user: int, track: int): void + add(album: int, title: String, track: int, duration: int, genre: int): Boolean + get(album: int, title: String): int
Pre e Post Condizioni	• context TrackDAO:: TrackDAO (connection) **pre:** true **post:** self.connection = connection • context ProfileDAO::get(id) **pre:** true **post:** result = tracks→select(p | p.id = id) • context ProfileDAO::getFromAlbum(id) **pre:** true **post:** result = profiles→select(p | p.album = id) • context ProfileDAO::getTopFiveFromArtist(id) **pre:** true **post:** result = tracks->sortedBy(t | t.plays)->subSequence(0, 4) • context ProfileDAO::addPlay(user, track) **pre:** true **post:** plays->count(p | p.user = user **and** p.track = track) = plays@pre->count(p | p.user = user **and** p.track) + 1 • context ProfileDAO::add(album, title, track, duration, genre) **pre:** true

	post: tracks->exists(t | t.album = album **and** t.title = title **and** t.track = track **and** t.duration = duration **and** t.genre = genre) • context ProfileDAO::get(album, title) **pre:** true **post:** result = track->(t | t.album = album **and** t.title = title).id
Invarianti	

3.2.3 CONTROL

Il metodo ***init()*** di ogni Servlet inizializza i DAO singleton necessari allo svolgimento dell'operazione recuperandoli dal ServletContext. La logica di business è contenuta interamente nel metodo ***doPost(request: HttpServletRequest, response: HttpServletResponse)***.

Fatta questa premessa, per una maggiore chiarezza e semplicità, riportiamo per ogni oggetto Control solo i DAO utilizzati e le condizioni del metodo doPost(…). In particolare, facciamo riferimento ai principali metodi dei DAO invocati e alle informazioni salvate nell'oggetto session.

Indichiamo con *bean: TipoBean = request->get()* le varie operazioni di recupero di parametri per comporre il bean.

Nome Classe	GetAlbum
Modulo	Album
Variabili di Istanza	-albumDAO: AlbumDAO -profileDAO: ProfileDAO -trackDAO: TrackDAO
Descrizione	Servlet che permette di recuperare le informazioni relative a un album
Definizioni	• context GetAlbum **def** session: HttpSession = request->getSession() **def** id: String = request.getParameter("id")
Pre e Post condizioni	• context GetAlbum::doPost(…) **pre:** id <> null

	post: let album: AlbumBean = albumDAO->get(id) session->getAttribute("Album") = album
Invarianti	

Nome Classe	UploadAlbum
Modulo	Album
Variabili di Istanza	-albumDAO: AlbumDAO -trackDAO: TrackDAO
Descrizione	Servlet che permette di effettuare l'upload di un album e dei relativi brani sulla piattaforma
Definizioni	• context UploadAlbum **def** session: HttpSession = request->getSession() **def** album: AlbumBean = request->get() **def** tracklist: Collection<TrackBean> = request->get()
Pre e Post condizioni	• context UploadAlbum::doPost(…) **pre:** album <> null **and** tracklist.size > 0 **post:** albumDAO->add(album) **and** tracklist->forAll(t: TrackBean | trackDAO->add(t)
Invarianti	

Nome Classe	DeleteAlbum
Modulo	Album
Variabili di Istanza	-albumDAO: AlbumDAO -trackDAO: TrackDAO
Descrizione	Servlet che permette di recuperare le informazioni relative a un album
Definizioni	• context DeleteAlbum **def** session: HttpSession = request->getSession() **def** id: String = request.getParameter("id")
Pre e Post condizioni	• context DeleteAlbum::doPost(…) **pre:** id <> null

	post: let album: AlbumBean = albumDAO->get(id) album.tracklist->forAll(t: TrackBean | trackDAO->remove(t)) **and** albumDAO->remove(album)
Invarianti	

Nome Classe	NextPage
Modulo	Navigation
Variabili di Istanza	
Descrizione	Servlet che permette all'utente di tornare a una pagina visitata in precedenza
Definizioni	• context NextPage **def** session: HttpSession = request->getSession() **def** navigator: Navigator = session.getAttribute("Navigator")
Pre e Post condizioni	• context NextPage::doPost(…) **pre:** true **post: let** page: Page = navigator->next() result = page
Invarianti	

Nome Classe	PrevPage
Modulo	Navigation
Variabili di Istanza	
Descrizione	Servlet che permette all'utente di tornare a una pagina visitata in precedenza
Definizioni	• context PrevPage **def** session: HttpSession = request->getSession() **def** navigator: Navigator = session.getAttribute("Navigator")
Pre e Post condizioni	• context PrevPage::doPost(…) **pre:** true **post: let** page: Page = navigator->prev() result = page
Invarianti	

Nome Classe	Search
Modulo	Navigation
Variabili di Istanza	-playlistDAO: PlaylistDAO -profileDAO: ProfileDAO -albumDAO: AlbumDAO -trackDAO: TrackDAO
Descrizione	Servlet che permette la ricerca di contenuti
Definizioni	• context Search **def** session: HttpSession = request->getSession() **def** search: String = request.getParameter(“search”)
Pre e Post condizioni	• context Search::doPost(…) **pre:** true **post: let** users: Collection<UserBean> = profileDAO.searchUsers(search) **let** artists: Collection<ArtistBean> = profileDAO.searchArtist(search) **let** tracks: Collection<TrackBean> = trackDAO.search(search) **let** albums: Collection<AlbumBean> = albumDAO.search(search) **let** playlists: Collection<Playlists> = playlistDAO.search(search) **let** container: ResultsContainer = ResultsContainer() container.users = users **and** container.artists = artists **and** container.tracks = tracks **and** container.albums = albums **and** container.playlists = playlists **and** session.getAttribute(“Container”) = container
Invarianti	

Nome Classe	AddGuest
Modulo	Playlist

Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di aggiungere un utente guest a una playlist collaborativa
Definizioni	• context AddGuest **def** session: HttpSession = request->getSession() **def** host: String = session->getAttribute("Profile")->getId() **def** guest: String = request.getParameter("guest") **def** playlist: String = request.getParameter("playlist")
Pre e Post condizioni	• context AddGuest::doPost(…) **pre:** guest <> null **and** playlist <> null **post:** playlistDAO->addGuest(host, guest, playlist)
Invarianti	

Nome Classe	AddTrack
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di aggiungere un brano a una playlist
Definizioni	• context AddTrack **def** session: HttpSession = request->getSession() **def** user: String = session->getAttribute("Profile")->getId() **def** track: String = request.getParameter("track") **def** playlist: String = request.getParameter("playlist")
Pre e Post condizioni	• context AddGuest::doPost(…) **pre:** track <> null **and** playlist <> null **post:** playlistDAO->addTrack(user, track, playlist)
Invarianti	

Nome Classe	ChangeOrder
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di cambiare l'ordine di visualizzazione dei brani in una playlist
Definizioni	• context ChangeOrder **def** session: HttpSession = request->getSession() **def** order: String = request.getParameter("order")
Pre e Post condizioni	• context ChangeOrder::doPost(…) **pre:** order **in** {"date", "title", "duration", "artist", "album"} **post:** session->getAttribute("Order") = order
Invarianti	

Nome Classe	CheckLike
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di controllare se un utente ha lasciato like a una playlist pubblica
Definizioni	• context CheckLike **def** session: HttpSession = request->getSession() **def** user: String = session->getAttribute("Profile")->getId() **def** playlist: String = request.getParameter("playlist")
Pre e Post condizioni	• context CheckLike::doPost(…) **pre:** playlist <> null **post:** result = playlistDAO.checkLike(user, playlist)
Invarianti	

Nome Classe	CreatePlaylist

Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di creare una nuova playlist, inizialmente vuota
Definizioni	• context CreatePlaylist **def** session: HttpSession = request->getSession() **def** user: UserBean = session->getAttribute("Profile")
Pre e Post condizioni	• context CreatePlaylist::doPost(…) **pre:** true **post: let** playlist: PlaylistBean = PlaylistBean() playlist->getHost() = user **and** user->getPlaylists()->exists(p | p = playlist)
Invarianti	

Nome Classe	DeletePlaylist
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di eliminare una playlist
Definizioni	• context DeletePlaylist **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter("id")
Pre e Post condizioni	• context DeletePlaylist::doPost(…) **pre:** id <> null **post:** playlistDAO.remove(id)
Invarianti	

Nome Classe	EditPlaylist
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO

Descrizione	Servlet che permette di modificare una playlist
Definizioni	• context EditPlaylist **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter(“id”) **def** title: String = request->getParameter(“title”)
Pre e Post condizioni	• context EditPlaylist::doPost(…) **pre:** id <> null **post: if** title <> null **then** playlistDAO.changeTitle(id, title) **endif**
Invarianti	

Nome Classe	GetPlaylist
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO -profileDAO: ProfileDAO
Descrizione	Servlet che permette di recuperare tutte le informazioni relative a una playlist
Definizioni	• context GetPlaylist **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter(“id”)
Pre e Post condizioni	• context GetPlaylist::doPost(…) **pre:** id <> null **post: let** playlist: PlaylistBean = playlistDAO->get(id) playlist->setHost(profileDAO->getFromPlaylist(playlist->getId()) **let** tracklist: Collection<AddedBean> = playlistDAO->getFromPlaylist(playlist->getId()) playlist->setTracklist(tracklist) session.getAttribute(“Playlist”) = playlist
Invarianti	

Nome Classe	LikePublicPlaylist
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette a un utente di mettere like a una playlist pubblica
Definizioni	• context LikePublicPlaylist **def** session: HttpSession = request->getSession() **def** user: UserBean = session->getAttribute("Profile") **def** id: String = request->getParameter("id")
Pre e Post condizioni	• context LikePublicPlaylist::doPost(…) **pre:** id <> null **post:** playlistDAO.like(user, id) **and** user->getLikedPlaylists->exists(p | p.getId() = id)
Invarianti	

Nome Classe	UnlikePublicPlaylist
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette a un utente di togliere like a una playlist pubblica
Definizioni	• context UnlikePublicPlaylist **def** session: HttpSession = request->getSession() **def** user: UserBean = session->getAttribute("Profile") **def** id: String = request->getParameter("id")
Pre e Post condizioni	• context UnlikePublicPlaylist::doPost(…) **pre:** id <> null **post:** playlistDAO.unlike(user, id) **and** user->getLikedPlaylists->forAll(p | p.getId() <> id)
Invarianti	

Nome Classe	MakeCollaborative
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di rendere una playlist collaborativa
Definizioni	• context MakeCollaborative **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter(“id”)
Pre e Post condizioni	• context MakeCollaborative::doPost(…) **pre:** id <> null **post:** playlistDAO.makeCollaborative(id)
Invarianti	

Nome Classe	MakeSingle
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di rendere una playlist singola
Definizioni	• context MakeSingle **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter(“id”)
Pre e Post condizioni	• context MakeSingle::doPost(…) **pre:** id <> null **post:** playlistDAO.makeSingle(id)
Invarianti	

Nome Classe	MakePublic
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di rendere una playlist pubblica

Definizioni	• context MakePublic **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter(“id”)
Pre e Post condizioni	• context MakePublic::doPost(…) **pre:** id <> null **post:** playlistDAO.makePublic(id)
Invarianti	

Nome Classe	MakePrivate
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di rendere una playlist privata
Definizioni	• context MakePrivate **def** session: HttpSession = request->getSession() **def** id: String = request->getParameter(“id”)
Pre e Post condizioni	• context MakePrivate::doPost(…) **pre:** id <> null **post:** playlistDAO.makePrivate(id)
Invarianti	

Nome Classe	RemoveTrack
Modulo	Playlist
Variabili di Istanza	-playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di rimuovere un brano da una playlist
Definizioni	• context RemoveTrack **def** session: HttpSession = request->getSession() **def** track: String = request.getParameter(“track”) **def** playlist: String = request.getParameter(“playlist”)

Pre e Post condizioni	• context RemoveTrack::doPost(…) **pre:** track <> null **and** playlist <> null **post:** playlistDAO->removeTrack(track, playlist)
Invarianti	

Nome Classe	CheckCredentials
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di controllare le credenziali inserite in fase di autenticazione
Definizioni	• context CheckCredentials **def** session: HttpSession = request->getSession() **def** email: String = request.getParameter(“email”) **def** password: String = request.getParameter(“password”)
Pre e Post condizioni	• context CheckCredentials::doPost(…) **pre:** email <> null **and** password <> null **post:** result = profileDAO->check(email, password)
Invarianti	

Nome Classe	CheckFollowing
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che ci permette di controllare se un utente segue il profilo di un altro utente o di un artista
Definizioni	• context CheckFollowing **def** session: HttpSession = request->getSession() **def** user: String = session->getAttribute(“Profile”)->getId() **def** followed: int = request.getParameter(“id”) **def** type: String = request.getParameter(“type”)

Pre e Post condizioni	• context CheckFollowing::doPost(…) **pre:** followed <> null **and** type <> null **post:** result = profileDAO->checkFollowing(user, followed)
Invarianti	

Nome Classe	CheckNations
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di recuperare tutte le nazione per la visualizzazione dinamica su jsp
Definizioni	• context CheckNations **def** session: HttpSession = request->getSession()
Pre e Post condizioni	• context CheckNations::doPost(…) **pre:** true **post:** result = profileDAO.getAllNations()
Invarianti	

Nome Classe	DeleteProfile
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di eliminare un profilo
Definizioni	• context DeleteProfile **def** session: HttpSession = request->getSession() **def** profile: ProfileBean = session->getAttribute(“Profile”)
Pre e Post condizioni	• context DeleteProfile::doPost(…) **pre:** profile <> null **post: if** profile->getRole() == “user” **then** profileDAO->removeUser(profile->getId()) **else** profileDAO->removeArtist(profile->getId())

Invarianti	

Nome Classe	EditProfile
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di modificare alcune informazioni del proprio profilo (alias, password, visibilità, etc.)
Definizioni	• context EditProfile **def** session: HttpSession = request->getSession() **def** profile: String = session->getAttribute("Profile") **def** id: int = profile->getId() **def** alias: String = request.getParameter("alias") **def** password: String = request.getParameter("password") **def** part: Part = request.getPart("cover") **def** isPublic: boolean = request.getParameter("isPublic") **def** bio: String = request.getParameter("bio")
Pre e Post condizioni	• context EditProfile::doPost(…) **pre:** id <> null **post: if** alias <> null **then** profileDAO.changeAlias(id, alias) **if** password <> null **then** profileDAO.changePassword(id, password) **if** isPublic <> null **then** profileDAO.changePublic(id, part) **if** bio <> null **and** profile->getRole() == "artist" **then** profileDAO.changeTitle(id, bio)
Invarianti	

Nome Classe	FollowArtist
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di seguire (follow) un artista

Definizioni	• context FollowArtist **def** session: HttpSession = request->getSession() **def** profile: UserBean = session->getAttribute(“Profile”) **def** user: int = profile->getId() **def** artist: int = request.getParameter(“id”)
Pre e Post condizioni	• context FollowArtist::doPost(…) **pre:** user <> null **and** artist <> null **post:** profileDAO->followArtist(user, artist)
Invarianti	

Nome Classe	FollowUser
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di seguire (follow) un utente
Definizioni	• context FollowUser **def** session: HttpSession = request->getSession() **def** profile: UserBean = session->getAttribute(“Profile”) **def** follower: int = profile->getId() **def** followed: int = request.getParameter(“id”)
Pre e Post condizioni	• context FollowUser::doPost(…) **pre:** follower <> null **and** followed <> null **post:** profileDAO->followUser(follower, followed)
Invarianti	

Nome Classe	GetArtist
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO -albumDAO: AlbumDAO -trackDAO: TrackDAO

Descrizione	Servlet che permette di recuperare le informazioni relative a un Artista
Definizioni	• context GetArtist **def** session: HttpSession = request->getSession() **def** id: int = request.getParameter(“id”)
Pre e Post condizioni	• context GetArtist::doPost(…) **pre:** id<> null **post: let** artist: ArtistBean = ProfileDAO->get(id) session->getAttribute(“Artist”) = artist
Invarianti	

Nome Classe	GetUser
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO -playlistDAO: PlaylistDAO -albumDAO: AlbumDAO
Descrizione	Servlet che permette di recuperare le informazioni relative a un Utente
Definizioni	• context GetUser **def** session: HttpSession = request->getSession() **def** id: int = request.getParameter(“id”)
Pre e Post condizioni	• context GetUser::doPost(…) **pre:** id <> null **post: let** user: UserBean = ProfileDAO->get(id) session->getAttribute(“User”) = user
Invarianti	

Nome Classe	Login
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO -playlistDAO: PlaylistDAO -albumDAO: AlbumDAO

Descrizione	Servlet che permette l'autenticazione
Definizioni	• context Login **def** session: HttpSession = request->getSession() **def** email: String = request.getParameter("email") **def** password: String = request.getParameter("password")
Pre e Post condizioni	• context Login::doPost(…) **pre:** email <> null **and** password <> null **post: let** profile: ProfileBean = ProfileDAO->get(id) session->getAttribute("Profile") = profile
Invarianti	

Nome Classe	Logout
Modulo	Profile
Variabili di Istanza	
Descrizione	Servlet che permette la disconnessione dall'applicazione
Definizioni	• context Logout **def** session: HttpSession = request->getSession()
Pre e Post condizioni	• context Logout::doPost(…) **pre:** true **post:** session->invalidate()
Invarianti	

Nome Classe	SearchForGuests
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di cercare utenti Guests per una playlist collaborativa
Definizioni	• context SearchForGuests **def** session: HttpSession = request->getSession()

	def search: String = request.getParameter(“search”)
Pre e Post condizioni	• context SearchForGuests::doPost(…) **pre:** search <> null **post:** result = profileDAO->searchUsersByAlias(search)
Invarianti	

Nome Classe	Signup
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO -playlistDAO: PlaylistDAO
Descrizione	Servlet che permette di registrarsi alla piattaforma
Definizioni	• context Signup **def** session: HttpSession = request->getSession() **def** email: String = request.getParameter(“email”) **def** password: String = request.getParameter(“password”) **def** alias: String = request.getParameter(“alias”) **def** type: String = request.getParameter(“type”)

Pre e Post condizioni	• context Signup::doPost(…) **pre:** email <> null **and** playlist <> null **and** alias <> null **post: if** type == “user” **then** profileDAO->add(user) **else** profileDAO->add(artist)
Invarianti	

Nome Classe	UnfollowArtist
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO

Descrizione	Servlet che permette di smettere di seguire (unfollow) un Artista
Definizioni	• context UnfollowArtist **def** session: HttpSession = request->getSession() **def** profile: UserBean = session->getAttribute(“Profile”) **def** user: int = profile->getID() **def** artist: int = request.getParameter(“id”)
Pre e Post condizioni	• context UnfollowArtist::doPost(…) **pre:** user <> null **and** artist <> null **post:** profileDAO->unfollowArtist(user, artist) profile->getArtists()->remove(artist)
Invarianti	

Nome Classe	UnfollowUser
Modulo	Profile
Variabili di Istanza	-profileDAO: ProfileDAO
Descrizione	Servlet che permette di smettere di seguire (unfollow) un Utente
Definizioni	• context UnfollowUser **def** session: HttpSession = request->getSession() **def** profile: UserBean = session->getAttribute(“Profile”) **def** follower: int = profile->getID() **def** followed: int = request.getParameter(“id”)
Pre e Post condizioni	• context UnfollowUser::doPost(…) **pre:** user <> null **and** artist <> null **post:** profileDAO->unfollowUser(follower, followed) profile->getFollowing()->remove(followed)
Invarianti	

Nome Classe	AddToQueue

Modulo	Track
Variabili di Istanza	
Descrizione	Servlet che permette di aggiungere una canzone alla coda degli ascolti
Definizioni	• context AddToQueue **def** session: HttpSession = request->getSession() **def** id: int = request.getParameter(“id”) **def** queue: ListeningQueue = session->getAttribute(“ListeningQueue”)
Pre e Post condizioni	• context AddToQueue::doPost(…) **pre:** id <> null **and** ListeningQueue <> null **post:** queue->addToUserQueue(id)
Invarianti	

Nome Classe	Play
Modulo	Track
Variabili di Istanza	-trackDAO: TrackDAO -albumDAO: AlbumDAO -profileDAO: ProfileDAO
Descrizione	Servlet che permette di riprodurre un brano
Definizioni	• context Play **def** session: HttpSession = request->getSession() **def** userId: int = session->getAttribute(“Profile”)->getId() **def** trackId: int = request.getParameter(“trackId”)
Pre e Post condizioni	• context Play::doPost(…) **pre:** userId <> null **and** trackId <> null **post:** trackDAO->addPlay(userId, trackId)
Invarianti	

Nome Classe	PlayAlbum
Modulo	Track

Variabili di Istanza	
Descrizione	Servlet che permette di riprodurre un album
Definizioni	• context PlayAlbum **def** session: HttpSession = request->getSession() **def** index: int = request.getParameter(“index”) - 1 **def** album: AlbumBean = session->getAttribute(“Album”) **def** queue: ListeningQueue = session->getAttribute(“ListeningQueue”)
Pre e Post condizioni	• context PlayAlbum::doPost(…) **pre:** index <> null **and** album <> null **and** queue <> null **post:** queue->forAll(t: TrackBean | trackDAO->add(t))
Invarianti	

Nome Classe	PlayPlaylist
Modulo	Track
Variabili di Istanza	
Descrizione	Servlet che permette di riprodurre una playlist
Definizioni	• context PlayPlaylist **def** session: HttpSession = request->getSession() **def** index: int = request.getParameter(“index”) - 1 **def** playlist: AlbumBean = session->getAttribute(“Playlist”) **def** queue: ListeningQueue = session->getAttribute(“ListeningQueue”)
Pre e Post condizioni	• context PlayPlaylist::doPost(…) **pre:** index <> null **and** playlist <> null **and** queue <> null **post:** queue->forAll(t: TrackBean | trackDAO->add(t))
Invarianti	

Nome Classe	Skip
Modulo	Track

Variabili di Istanza	
Descrizione	Servlet che permette di saltare una canzone della Coda di Ascolti
Definizioni	• context Skip **def** session: HttpSession = request->getSession() **def** queue: ListeningQueue = session->getAttribute(“ListeningQueue”)
Pre e Post condizioni	• context Skip::doPost(…) **pre:** queue <> null **post: let** id = queue->poll()
Invarianti	