# User

**Поля:**

tracks: Track[](Protected): Массив треков пользователя

playlists: Playlist[](Protected): Массив плейлистов пользователя

albums: Album[](Protected): Массив альбомов пользователя

name: String (Protected): Никнейм пользователя

email: String (Protected): Email учетной записи пользователя

password: String (Protected): Пароль от учетной записи пользователя

isBlocked: bool (Protected): Показывает, был ли заблокирован конкретный пользователь

**Методы:**

getName(): String (Protected): Возвращает никнейм пользователя

setName(): void (Protected): Устанавливает никнейм пользователя

getEmail(): String (Protected): Возвращает email пользователя

setEmail(): void (Protected): Возвращает email пользователя

getPassword(): String (Protected): Возвращает пароль пользователя

setPassword(): String (Protected): Устанавливает пароль пользователя

# UserWindow

**Поля:**

user: User(Private): Ссылка на конкретного пользователя

# LibraryWindow

**Поля:**

playlists: Playlist[](Private): Массив плейлистов, в котором находится плейлисты, созданных и добавленных пользователем в библиотеку

tracks: Track[](Private): Массив треков, добавленных пользователем в библиотеку

albums: Album[](Private): Массив альбомов, добавленных пользователем в библиотеку

**Методы:**  
deleteTrackFromLibrary(Track track): void(Private) - Удаляет выбранный пользователем трек из массива треков

createPlaylist(): void(Private): Создает и добавляет в библиотеку пользователя созданный плейлист

deletePlaylistFromLibrary(): void(Private): Удаляет выбранный плейлист из библиотеки пользователя

deleteAlbumFromLibrary(Album album): void(Private): Удаляет выбранный пользователем альбом из массива альбомов

# SearchComponent

**Поля:**

response: String(Private): Строка запроса

**Методы:**

search(): void(Private): Метод, отвечающий за обработку запроса

deleteResponse(): void(Private): Метод, отвечающий за очистку строки запроса

# ResponseComponent

**Методы:**

showListTrack(): void(Private): Метод, отвечающий за показ данных, соответствующих запросу

addTrackToLibrary(Track track): void(Private): Метод, который добавляет в массив треков пользователя выбранный пользователем трек

addAlbumToLibrary(Album album): void(Private): Метод, который добавляет в массив альбомов пользователя выбранный пользователем альбом

# ReccomendationWindow

**Поля:**

user: User(Private): Ссылка на конкретного пользователя

**Методы:**

searchTrackByGenre(): void(Private): Метод, который ищет схожие по жанру треки

createPlaylistByGenres(): Playlist(Private): Метод, создающий плейлист и добавляющий в него треки, схожие по жанрам

showRecomendationTracks(): void(Private): Метод, отображающий треки, схожие с жанрами треков пользователя

# SettingsComponent

**Методы:**

deleteAccount(): void(Private): Метод, удаляющий аккаунт

swithTheme(): void(Private): Метод, который меняет тему приложения по выбору пользователя

exit(): void(Private): Метод, который закрывает все окна и выводит начальное окно с выбором авторизации и регистрации

# Admin

**Поля:**

tracks: Track[](Private): Массив треков. По умолчанию = null

playlists: Playlist[](Private): Массив плейлистов. По умолчанию = null

albums: Album[](Private): Массив альбомов. По умолчанию = null

isBlocked: bool (Private): Поле, которое показывает, заблокирован ли администратор (по умолчанию false)

Класс Admin не имеет методов

# AdminWindow

**Поля:**

admin: Admin(Private): Ссылка на конкретного администратора

**Методы:**

blockUser(User user): void(Private): Блокирует учетную запись указанного пользователя

deleteUser(User user): void(Private): Удаляет учетную запись указанного пользователя

changeRole(User user, String newRole): void(Private): Изменяет роль указанного пользователя

# Artist

**Поля:**

playlists: Playlist[](Private): Массив плейлистов. По умолчанию = null

avatar: String (Private): Аватар артиста.

text: String (Private): Описание, связанное с артистом.

genres: String[] (Private): Массив жанров, связанных с артистом.

**Методы:**

getGenres(): String[] (Public): Возвращает массив жанров, связанных с артистом.

setGenres(genres: String[]): void (Public): Устанавливает массив жанров для артиста.

# ArtistWindow

**Поля:**

artist: Artist(Private): Ссылка на конкретного исполнителя

# InformationComponent

**Поля:**

artist: Artist(Private): Ссылка на конкретного исполнителя

**Методы:**

changeInformation(): void(Private): Метод, который перезаписывает данные, находящиеся в текстовом поле окна исполнителя

# AvatarComponent

**Поля:**

artist: Artist(Private): Ссылка на конкретного исполнителя

**Методы:**

loadAvatar(): void(Private): Метод, который удаляет старый аватар и на его место устанавливает изображение, загруженное пользователем

deleteAvatar(): void(Private): Метод, который удаляет аватар исполнителя и на его место устанавливает базовую картинку

# LibraryArtistComponent

**Поля:**

tracks: Track[] (Private): Массив треков исполнителя

albums: Album[] (Private): Массив альбомов испоSлнителя

**Методы:**

loadTrackToApp(): void(Private): Метод, который добавляет трек в библиотеку артиста

deleteTrackFromApp(): void(Private): Метод, который удаляет трек с площадки

createAlbum(): void(Private): Метод, который создает альбом из загруженных треков на площадке

# AutorisationOrRegistrationWindow

**Методы:**

setEmail(email: String): Устанавливает email.

setPassword(password: String): Устанавливает пароль.

checkEmailAndPassword(email: String, password: String): Проверяет введенные email и пароль.

changeFunction(): Изменяет функцию окна между авторизацией и регистрацией.

createUser(email: String, password: String): Создает нового пользователя с заданным email и паролем.

# TrackContainer (Интерфейс)

**Методы:**

addTrackToPlaylist(track: Track, playlist: Playlist): Добавляет трек в плейлист.

deleteTrackFromPlaylist(track: Track, playlist: Playlist): Удаляет трек из плейлиста.

getTracks(): Возвращает массив треков в контейнере.

getName(): Возвращает имя контейнера.

setName(name: String): Устанавливает имя контейнера.

getDescription(): Возвращает описание контейнера.

setDescription(description: String): Устанавливает описание контейнера.

setPlaylistPicture(playlistPicture: String): Устанавливает изображение контейнера.

# Track

**Поля:**

name: string(Private): Название трека.

genre: string(Private): Жанр трека.

artist: string(Private): Исполнитель трека.

songPicture: string(Private): Изображение трека.

audio: string(Private): Ссылка на аудиофайл трека.

duration: time(Private): Продолжительность трека.

**Методы:**

getName(): string(Private): Возвращает название трека.

setName(name: String): void(Private): Устанавливает название трека.

getGenre(): string(Private): Возвращает жанр трека.

setGenre(genre: String): void(Private): Устанавливает жанр трека.

getArtist(): string(Private): Возвращает исполнителя трека.

setArtist(artist: String): void(Private): Устанавливает исполнителя трека.

setSongPicture(songPicture: String): string(Private): Устанавливает ссылку на изображение трека.

getAudio(): string(Private): Возвращает аудиофайл трека.

setAudio(audio: String): void(Private): Устанавливает аудиофайл трека.

getDuration(): string(Private): Возвращает продолжительность трека.

# Playlist

**Поля:**

tracks: Track[](Private): Массив треков, входящих в плейлист.

name: string(Private): Название плейлиста.

description: string(Private): Описание плейлиста.

playlistPicture: string(Private): Изображение плейлиста.

# Album

**Поля:**

tracks: Track[](Private): Массив треков, входящих в альбом.

name: string(Private): Название альбома.

description: string(Private): Описание альбома.

albumPicture: string(Private): Ссылка на изображение альбома.

# Player

**Поля:**

track: Track(Private): Ссылка на текущий трек.

**Методы:**

trackPlay(Track track): void(Private): Начинает воспроизведение текущего трека.

trackPause(Track track): void(Private): Приостанавливает воспроизведение текущего трека.

trackRewind(Track track):void(Private): Перематывает текущий трек.

changeVolumeLevel(Track track): void(Private): Изменяет уровень громкости воспроизведения.