Sistema de Música

Christyan Assolini, Vitor Hugo

Engenharia de Software
Universidade da Região de Joinville (UNIVILLE) – Joinville, SC – Brazil
christyan.assolini@univille.br, vitor.hugo.ramos.dos.santos@univille.br

1. Introdução

Este sistema é uma plataforma de streaming musical que oferece aos usuários a oportunidade de explorar uma ampla biblioteca de músicas, elaborar playlists, acompanhar seus artistas preferidos e descobrir novos sons. A estrutura do sistema estará organizada em duas categorias principais de usuários: ouvintes e artistas. Os ouvintes terão a chance de pesquisar e escutar músicas, criar e compartilhar suas playlists, além de seguir os artistas que admiram. Por outro lado, os artistas poderão subir suas canções, acompanhar dados de reproduções e se conectar com seus fãs.

2. Requisitos Funcionais

2.1 História de Usuário: Cadastro e Autenticação

Como um usuário, eu quero me cadastrar no sistema com meu nome de usuário, senha, e-mail e outras informações pessoais para que eu possa acessar minhas playlists e músicas. Após o cadastro, desejo fazer login usando minhas credenciais, com o sistema verificando e autenticando meu acesso. Além disso, gostaria de poder alterar minhas informações de perfil e redefinir minha senha caso seja necessário.

2.2 História de Usuário: Reproduzir e Descobrir Músicas

Como um ouvinte, eu quero pesquisar músicas, álbuns e artistas no aplicativo para encontrar e reproduzir diretamente as músicas que eu gosto. Também quero que o sistema permita que eu adicione essas músicas às minhas playlists e que eu possa curtir as músicas que eu mais gosto. O sistema deve, ainda, me sugerir novas músicas e álbuns com base no meu histórico de reprodução.

2.3 História de Usuário: Criação e Gerenciamento de Playlists

Como um ouvinte, eu quero criar playlists personalizadas para organizar minhas músicas favoritas. Quero poder adicionar ou remover músicas das minhas playlists a qualquer momento, além de poder escolher se a playlist será pública ou privada.

Também gostaria de poder compartilhar minhas playlists com amigos e receber sugestões de músicas para adicioná-las com base no meu gosto musical.

2.4 História de Usuário: Upload de Músicas e Interação com Seguidores (Artista)

Como um artista, eu quero fazer upload das minhas músicas no sistema e associá-las a álbuns que eu criei, para que meus seguidores possam ouvi-las e interagir com elas. Ao subir uma música, quero poder incluir informações como título, gênero e capa do álbum. Também gostaria de ver estatísticas detalhadas sobre a reprodução das minhas músicas, além de notificar meus seguidores sempre que lançar uma nova música ou álbum.

2.5 História de Usuário: Criação e Organização de Álbuns (Artista)

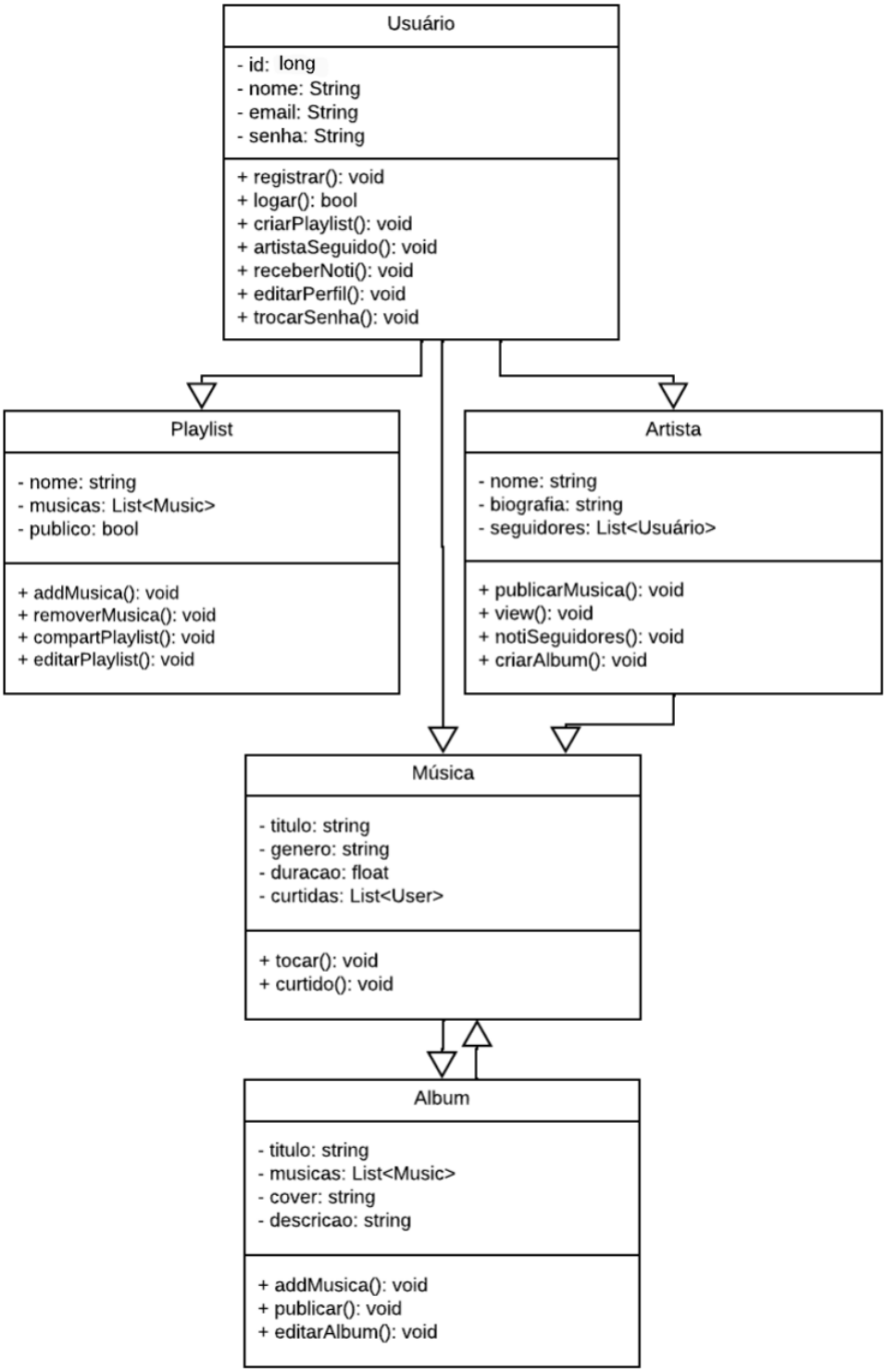
Como um artista, eu quero organizar minhas músicas em álbuns, podendo adicionar várias músicas em um único álbum e publicá-lo de forma que seja visualmente atraente e profissional. Também desejo editar as informações do álbum, como a capa, título, descrição e a lista de músicas, sempre que necessário. Isso me permite lançar um produto musical coerente e bem estruturado.

2.6 História de Usuário: Seguimento de Artistas e Notificações

Como um ouvinte, eu quero seguir meus artistas favoritos para que eu possa receber atualizações automáticas sobre novos lançamentos, álbuns e shows. Desejo ser notificado sempre que um artista que sigo publicar novas músicas ou realizar atualizações em seus álbuns, permitindo que eu me mantenha atualizado sobre as atividades dos artistas que mais gosto.

3. Diagrama de Classes (UML)

No diagrama de classes UML, são representadas as classes de um sistema, suas propriedades, comportamentos, e os relacionamentos entre elas. Onde podemos ver as classes usuário, artista, playlist, música e album.



3.1. Entidade Sistema De Música

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Column;

@Entity
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String nome;
    @Column(nullable = false, unique = true)
    private String email;
    @Column(nullable = false)
    private String senha;
    public Usuario() {
    }
       public Usuario(String nome, String email, String
senha) {
        this.nome = nome;
        this.email = email;
        this.senha = senha;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
```

```
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }
    public void registrar() {
    }
    public boolean logar() {
        return true;
    }
    public void criarPlaylist() {
    }
    public void artistaSeguido() {
    }
    public void receberNoti() {
    }
    public void editarPerfil() {
    }
    public void trocarSenha() {
    }
}
```

Figura 1. Sistema de Música Usuário.

```
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.ManyToMany;

@Entity
public class Artista {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String nome;

    @Column(length = 2000)
    private String biografia;

    @ManyToMany
    private List<Usuario> seguidores;
    public Artista() {
    }
    public Artista(String nome, String biografia) {
        this.nome = nome;
        this.biografia = biografia;
    }
    public Long getId() {
        return id;
    }
    public String getNome() {
        return nome;
```

```
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getBiografia() {
        return biografia;
    }
    public void setBiografia(String biografia) {
        this.biografia = biografia;
    }
    public List<Usuario> getSeguidores() {
        return seguidores;
    }
    public void setSeguidores(List<Usuario> seguidores) {
        this.seguidores = seguidores;
    }
    public void publicarMusica() {
    }
    public void view() {
    }
    public void notiSeguidores() {
    }
    public void criarAlbum() {
    }
}
```

Figura 2. Sistema de Música artista.

```
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.OneToMany;
```

```
import javax.persistence.CascadeType;

@Entity
public class Album {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String titulo;
    @OneToMany(cascade = CascadeType.ALL)
    private List<Musica> musicas;
    @Column(nullable = false)
    private String cover;
    @Column(length = 500)
    private String descricao;
    public Album() {
    }
       public Album(String titulo, String cover, String
descricao) {
        this.titulo = titulo;
        this.cover = cover;
        this.descricao = descricao;
    }
    public Long getId() {
        return id;
    }
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public List<Musica> getMusicas() {
```

```
        return musicas;
    }
    public void setMusicas(List<Musica> musicas) {
        this.musicas = musicas;
    }
    public String getCover() {
        return cover;
    }
    public void setCover(String cover) {
        this.cover = cover;
    }
    public String getDescricao() {
        return descricao;
    }
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
    public void addMusica(Musica musica) {
        musicas.add(musica);
    }
    public void publicar() {
    }
    public void editarAlbum() {
    }
}
```

Figura 3. Sistema de Música album.

```
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Column;
```

```
import javax.persistence.OneToMany;
import javax.persistence.CascadeType;

@Entity
public class Playlist {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String nome;
    @OneToMany(cascade = CascadeType.ALL)
    private List<Musica> musicas;
    @Column(nullable = false)
    private boolean publico;
    public Playlist() {
    }
    public Playlist(String nome, boolean publico) {
        this.nome = nome;
        this.publico = publico;
    }
    public Long getId() {
        return id;
    }
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
    public List<Musica> getMusicas() {
        return musicas;
    }
```

```
    public void setMusicas(List<Musica> musicas) {
        this.musicas = musicas;
    }
    public boolean isPublico() {
        return publico;
    }
    public void setPublico(boolean publico) {
        this.publico = publico;
    }
    public void addMusica(Musica musica) {
        musicas.add(musica);
    }
    public void removerMusica(Musica musica) {
        musicas.remove(musica);
    }
    public void compartPlaylist() {
    }
    public void editarPlaylist() {
    }
}
```

Figura 4. Sistema de Música playlist.

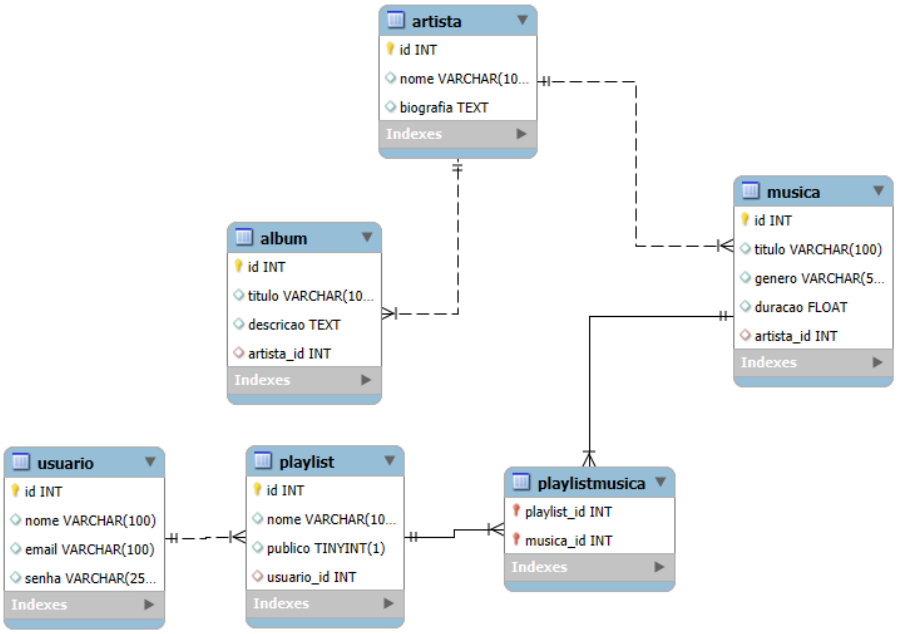
```
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.ManyToMany;

@Entity
public class Musica {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String titulo;
    @Column(nullable = false)
    private String genero;
    @Column(nullable = false)
    private float duracao;
    @ManyToMany
    private List<Usuario> curtidas;
    public Musica() {
    }
      public Musica(String titulo, String genero, float
duracao) {
        this.titulo = titulo;
        this.genero = genero;
        this.duracao = duracao;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public String getGenero() {
        return genero;
    }
```

```
    public void setGenero(String genero) {
        this.genero = genero;
    }
    public float getDuracao() {
        return duracao;
    }
    public void setDuracao(float duracao) {
        this.duracao = duracao;
    }
    public List<Usuario> getCurtidas() {
        return curtidas;
    }
    public void setCurtidas(List<Usuario> curtidas) {
        this.curtidas = curtidas;
    }
    public void tocar() {
    }
    public void curtido() {
    }
}
```

Figura 5. Sistema de Música musica.

4. Banco de dados



4. Conclusão

O desenvolvimento deste sistema de música proporcionou uma solução abrangente para usuários que buscam uma experiência completa de streaming musical, tanto para ouvintes quanto para artistas. Com funcionalidades como criação de playlists, upload de músicas, recomendação personalizada e seguimento de artistas, a plataforma oferece uma experiência dinâmica e personalizada.

Referências

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749. https://doi.org/10.1109/TKDE.2005.99

Anderson, C. (2006). The Long Tail: Why the Future of Business is Selling Less of More. Hyperion.

Pressman, R. S. (2019). Engenharia de Software: Uma Abordagem Profissional (9ª ed.). McGraw-Hill.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3ª ed.). Addison-Wesley Professional.