Sistema de Música XPO

Isabele Vitória Pires, Julia Lopes

Engenharia de Software
Universidade da Região de Joinville (UNIVILLE) – Joinville, SC – Brazil

isabele.pires@univille.br, jlopes@univille.br

1. Introdução

O XPO é uma plataforma digital projetada para oferecer uma experiência de streaming de música altamente personalizável e eficiente, semelhante ao Spotify. Com recursos que incluem desde a descoberta de novas músicas e a criação de playlists personalizadas até recomendações baseadas em preferências individuais, o XPO se destaca por priorizar a acessibilidade e a personalização.

A plataforma oferece streaming contínuo de músicas em alta qualidade, além de permitir o gerenciamento de bibliotecas pessoais de forma intuitiva e o compartilhamento fácil de conteúdo entre usuários, criando uma experiência musical integrada e sob medida para cada perfil.

2. Requisitos Funcionais

Os requisitos funcionais do Sistema XPO serão apresentados em forma de história de usuário.

2.1. História de Usuário 01

Como um verdadeiro amante de música, adoro explorar diferentes gêneros e artistas, e valorizo receber recomendações que reflitam meus gostos e hábitos de escuta. Ao longo do tempo, percebo que minhas preferências evoluem, e gosto de descobrir novas músicas que complementam o que já aprecio.

O diagrama de classes foi elaborado para representar as principais entidades do sistema de música, centrando-se no usuário como o ator principal. A classe **Usuário** contém um identificador e um nome, além de uma lista de playlists associadas, refletindo a capacidade do usuário de gerenciar suas coleções. A **Playlist** tem um relacionamento Many-to-One com **Usuário** e um relacionamento Many-to-Many com **Música**, permitindo que cada playlist contenha várias músicas e que uma música possa ser parte de várias playlists. A classe **Música** inclui informações como título e artista, enquanto a classe **Gênero** categoriza as músicas, facilitando a personalização de recomendações. A classe **Recomendação** é fundamental, representando o desejo do usuário de descobrir novas músicas alinhadas aos seus gostos. Este design oferece flexibilidade para futuras expansões, como novos algoritmos de recomendação, ao mesmo tempo que mantém a clareza e a organização do código, assegurando uma implementação eficiente das funcionalidades desejadas.

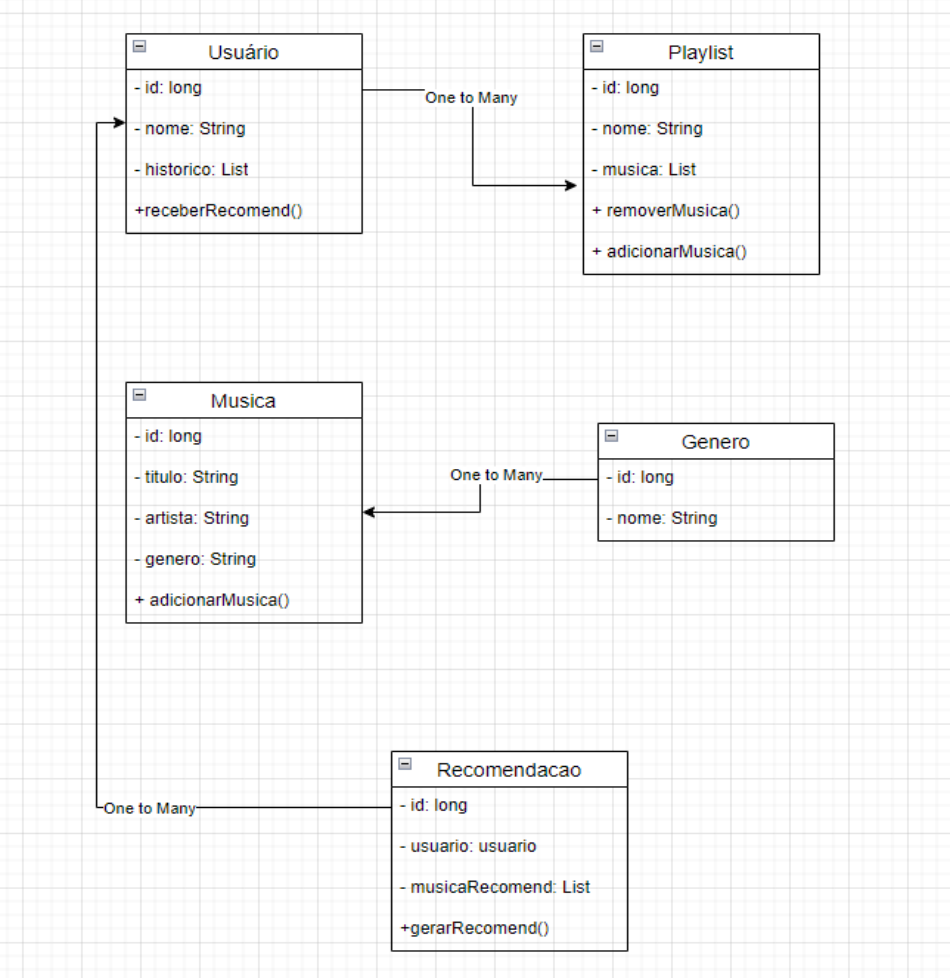


Figura 01. Diagrama de classe das entidades da História de Usuário 01.

A entidade Usuário é central, com atributos como id e nome, permitindo a identificação do usuário. O relacionamento 1

entre Usuário e Playlist mostra que cada usuário pode ter várias playlists, o que é comum em plataformas de música.

A entidade Playlist tem um relacionamento N

com Música, permitindo que cada playlist contenha várias músicas e uma música faça parte de várias playlists, capturando a flexibilidade na criação de coleções musicais.

A inclusão da entidade Gênero categoriza as músicas e facilita a exploração de estilos. A relação N:1 entre Música e Gênero indica que cada música pertence a um único gênero, mas um gênero pode ter muitas músicas.

A entidade Recomendação atende à necessidade do usuário de receber sugestões personalizadas. A relação N:1 entre Recomendação e Música permite que várias

recomendações incluam a mesma música, enquanto a relação com Usuário indica que as recomendações são personalizadas para cada um.

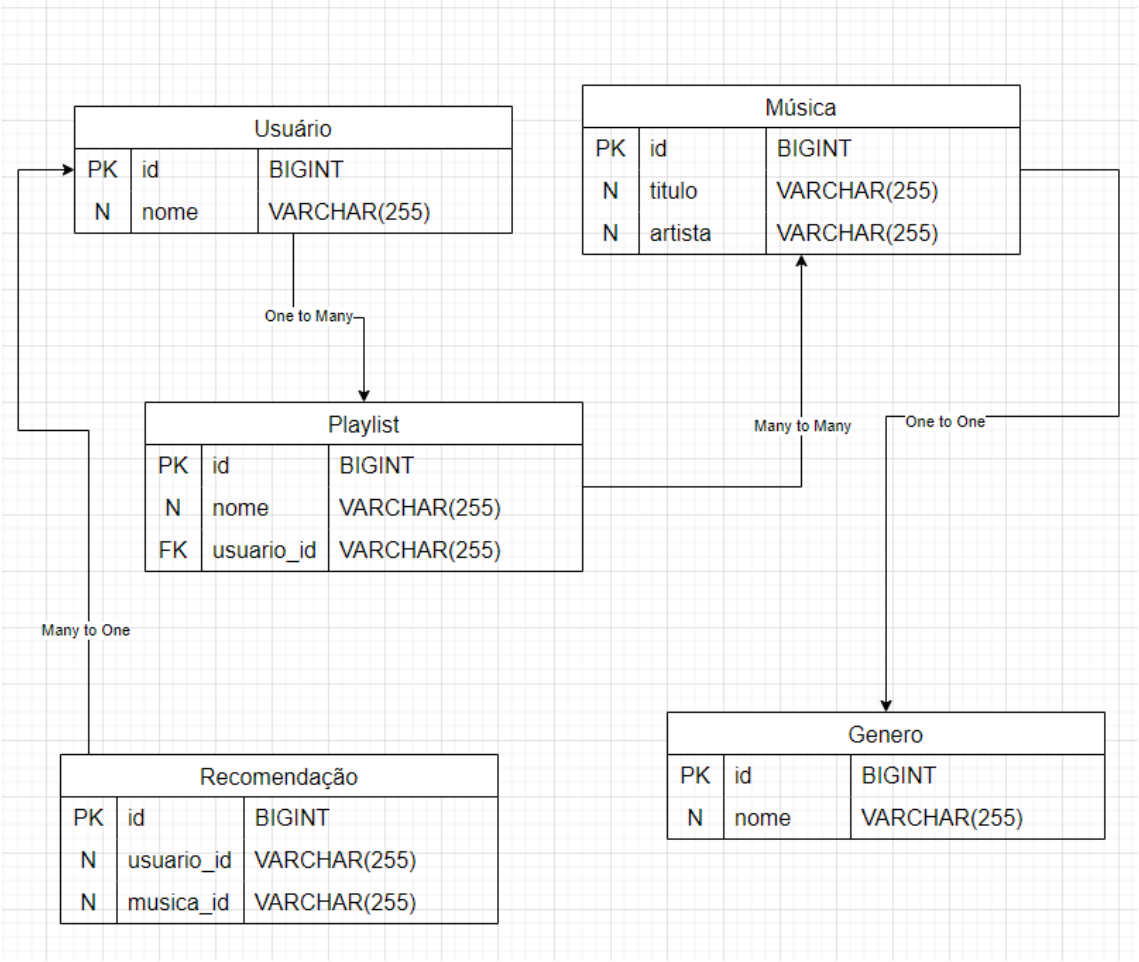


Figura 2. Modelo Entidade Relacionamento da História de Usuário 01.

2.2. História de Usuário 02

Como um usuário que escuta música em diferentes momentos do dia, quero poder criar e gerenciar playlists específicas para cada atividade, como treinos na academia, relaxamento ou foco no trabalho. Isso me permite acessar rapidamente as músicas ideais para cada situação, sem perder tempo procurando faixas que combinem com o momento. Dessa forma, posso aproveitar ao máximo minha experiência de escuta, tendo a trilha sonora certa para me motivar, relaxar ou me concentrar quando necessário.

2.3 História de Usuário 03

Como alguém que adora compartilhar músicas com amigos, gostaria de ter a opção de enviar playlists ou faixas específicas diretamente para meus contatos na plataforma ou

em redes sociais. Assim, posso facilmente recomendar minhas músicas favoritas e compartilhar descobertas musicais com quem tem gostos semelhantes, criando uma conexão mais próxima através da música.

3. Codificação

O modelo de entidades é composto por três classes principais: **Usuario**, **Playlist** e **Musica**. A classe **Usuario** representa os usuários da aplicação, contendo um identificador e um nome, e pode ter várias playlists associadas a ele. A **Playlist** organiza as músicas e se relaciona com um único usuário, permitindo que cada playlist contenha várias músicas. Por sua vez, a classe **Musica** descreve faixas musicais, com atributos como título e artista, e pode ser parte de várias playlists. Esses relacionamentos entre as classes configuram um sistema flexível que reflete a experiência musical dos usuários.

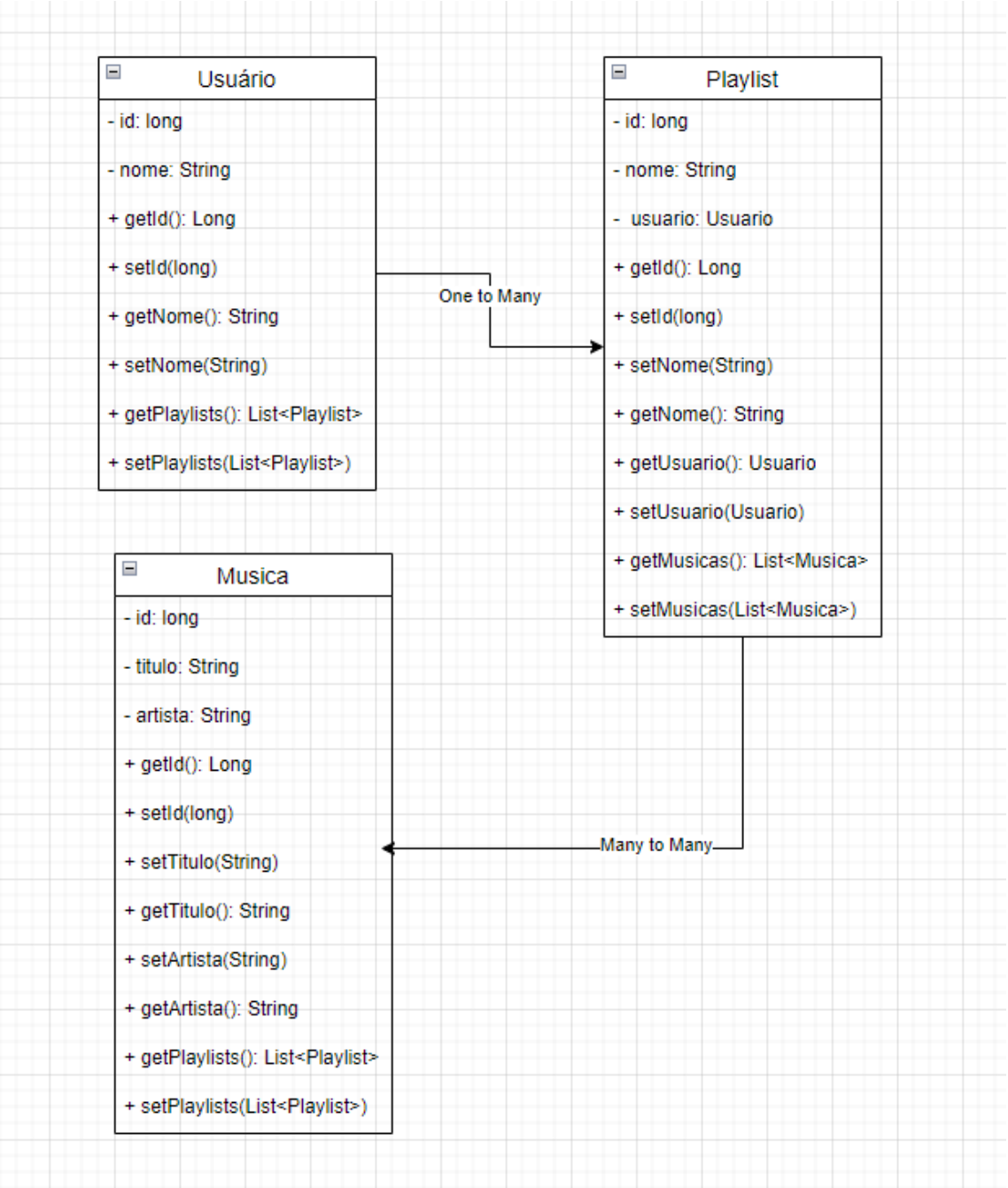


Imagem do diagrama de classes do Sistema XPO

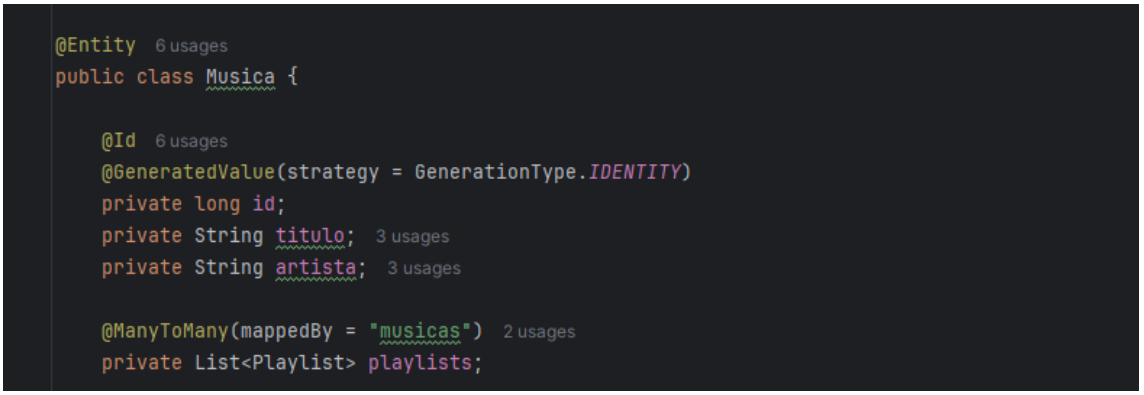
3.1. Entidades

O código da classe Playlist descreve um sistema de gerenciamento de músicas, onde a relação com a entidade Usuario é um Many-to-One (N:1), permitindo que um usuário tenha várias playlists, cada uma associada a apenas um usuário. A relação com a entidade Musica é um Many-to-Many (N), onde cada playlist pode incluir várias músicas e uma música pode estar em várias playlists. Essa conexão é gerenciada por uma tabela de junção chamada playlist_musica, facilitando a organização das músicas em diferentes playlists e promovendo uma experiência musical personalizada para os usuários.

```
public class Playlist {  9 usages

    @Id  6 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String nome;  3 usages

    @ManyToOne  3 usages
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;

    @ManyToMany  2 usages
    @JoinTable(
            name = "playlist_musica",
            joinColumns = @JoinColumn(name = "playlist_id"),
            inverseJoinColumns = @JoinColumn(name = "musica_id")
    )
```

Imagem do Código da entidade Playlist

O relacionamento N entre Musica e Playlist permite que os usuários criem playlists diversificadas e personalizadas. Com isso, uma única música pode aparecer em várias playlists, enquanto uma playlist pode incluir várias músicas, promovendo uma experiência de escuta rica e variada. Essa abordagem é comum em aplicativos de música, onde os usuários desejam organizar suas canções de maneira prática e acessível.



```
@Entity  6 usages
public class Musica {

    @Id  6 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String titulo;  3 usages
    private String artista;  3 usages

    @ManyToMany(mappedBy = "musicas")  2 usages
    private List<Playlist> playlists;
```

Imagem do Código da entidade Música

O relacionamento entre as classes Usuario e Playlist é um exemplo de relação 1(um para muitos). Isso significa que um único usuário pode ter várias playlists, mas cada playlist pertence a apenas um usuário. A classe Usuario contém uma lista de playlists (List<Playlist>) para representar essa associação. Na tabela Playlist, há uma coluna usuario_id que funciona como chave estrangeira, referenciando o id do usuário, garantindo a integridade referencial. Esse relacionamento é fundamental para permitir que os usuários organizem suas músicas em diferentes playlists, refletindo suas preferências e hábitos de escuta em uma plataforma de música.

```java
public class Usuario {  6 usages

    @Id  6 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String nome;  3 usages

    @OneToMany(mappedBy = "usuario")  2 usages
    private List<Playlist> playlists;

    public long getId() {  no usages
        return id;
    }
```

4. Banco de dados

O banco de dados **XPO_Musica** é estruturado para gerenciar a experiência musical dos usuários, incorporando três entidades principais: **Usuario**, **Musica** e **Playlist**. A tabela **Usuario** armazena informações dos usuários, enquanto a tabela **Musica** contém dados sobre as faixas, como título e artista. A tabela **Playlist** relaciona-se com o usuário, permitindo que ele crie várias playlists. A tabela **Playlist_Musica** estabelece um relacionamento muitos-para-muitos, possibilitando que uma playlist contenha várias músicas e que uma música esteja em várias playlists. Essa estrutura oferece flexibilidade na organização da biblioteca musical dos usuários.

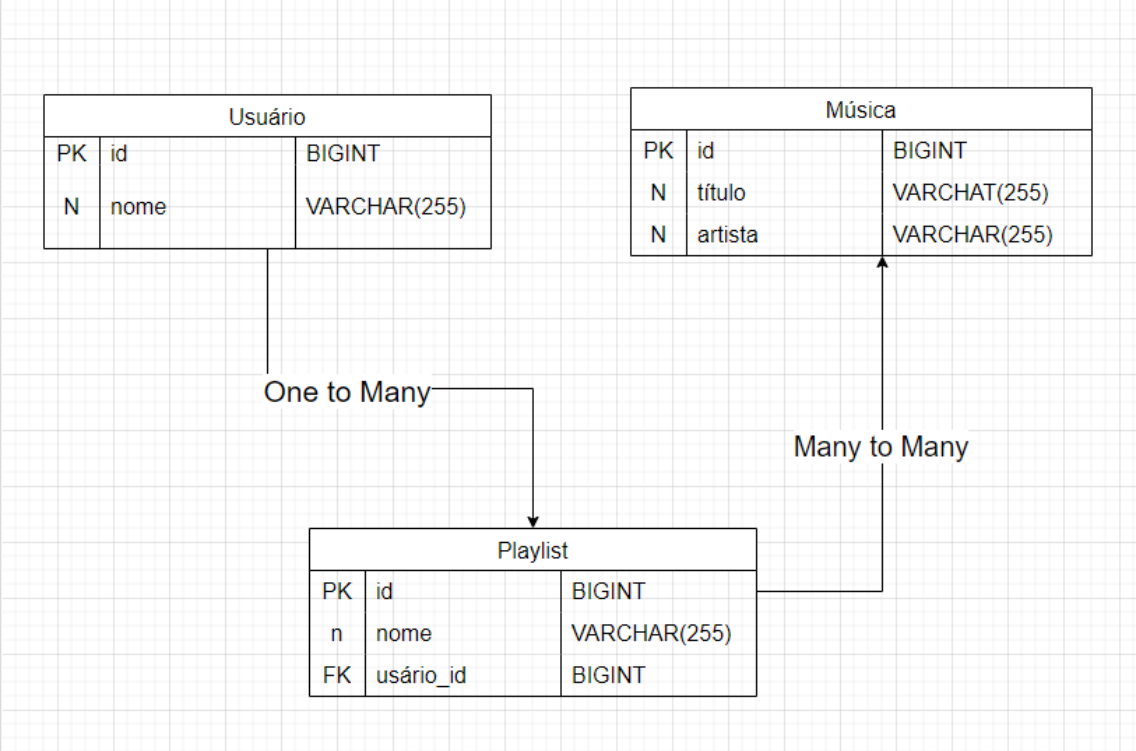


Imagem do Modelo Entidade Relacionamento do Sistema XPO

4. Conclusão

O sistema de música XPO foi projetado para oferecer uma experiência de streaming personalizada e eficiente, atendendo às necessidades dos usuários de forma intuitiva. Através das histórias de usuário, pudemos identificar os principais requisitos funcionais, como a personalização de playlists, a descoberta de novas músicas e o compartilhamento de conteúdo entre usuários. O desenvolvimento do XPO visa proporcionar uma plataforma acessível e flexível, capaz de se adaptar aos diferentes momentos do dia e preferências musicais dos usuários, garantindo uma experiência de uso satisfatória e inovadora. A implementação dos relacionamentos entre entidades e a estruturação eficiente do banco de dados são essenciais para o bom funcionamento da plataforma, contribuindo para a gestão eficiente de bibliotecas musicais e a entrega de recomendações precisas. Com isso, o XPO tem potencial para se consolidar como uma solução completa e competitiva no mercado de streaming musical.

Referências

Boulic, R. and Renault, O. (1991) "3D Hierarchies for Animation", In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) "Motion Capture White Paper", http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, December.

Holton, M. and Alexander, S. (1995) "Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials", Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.