Spotify
Documento de Arquitetura de Software

Gustavo Hammerschmidt
PUCPR - Pontíficia Universidade Católica Paraná
Ciência da Computação
Email: g.hammerschmidt@pucpr.edu.br

Karlos Eduardo de Oliveira Silva
PUCPR - Pontíficia Universidade Católica Paraná
Engenharia de Software
Email: karlos.silva@pucpr.edu.br

Data	Versão	Descrição	Autor
18/11/2020	1.0	Versão Inicial	Gustavo Hammerschmidt
25/11/2020	2.0	Versão Final	Gustavo Hammerschmidt

Conteúdo

1 Introdução

Este documento de arquitetura de Software visa a apresentar a arquitetura da aplicação Spotify, diagramas de classe e componentes e seus casos de uso.

1.1 Finalidade

Este documento fornece uma visão arquitetural abrangente do sistema, usando diversas visões de arquitetura para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

1.2 Escopo

Este Documento de Arquitetura de Software se aplica ao Sistema de Streaming de Músicas do Spotify.

1.3 Definição de Termos

A definição de termos utilizados pode ser encontrada nas referências mencionadas abaixo.

1.3.1 Referências

As referências[1][2][3][4][5][6][7] estão anexadas ao fim deste documento.

2 Representação da Arquitetura

Este documento apresenta a arquitetura como uma série de visões: visão de casos de uso, visão de processos, visão de implantação e visão de implementação. Essas visões são apresentadas como Modelos do ASTAH e utilizam a Linguagem Unificada de Modelagem (UML).

3 Objetivos e Restrições da Arquitetura

É objetivo deste documento definir a estruturação dos comportamentos da aplicação Spotify e sua arquitetura de software; são restrições da arquitetura o conhecimento do funcionamento da aplicação por completo como contratos e pagamentos, e a qualidade mínima dos produtos exibidos e a limitação das informações obtidas aos anos em que foram publicadas - o que pode infligir em uma desatualização deste documento em relação ao estado atual da aplicação. A aplicação também pode ser restringida por motivos de negócios e/ou sociopolíticos e geográficos.

4 Requisitos da Aplicação

4.1 Requisitos Funcionais

Os usuários devem poder buscar músicas, adicioná-las e removê-las de suas listas de músicas favoritas, e escutar o rádio da música(uma lista de músicas relacionadas). Eles também devem logar na aplicação para utilizá-la, e podem criar playlists de músicas, adicionando e removendo músicas a elas, e definir se ela é pública(visível a outros usuários) ou privada. Os usuários premium, que assinaram à aplicação podem baixar músicas, podcasts e playlists em seus dispositivos. Usuários premium podem

escolher qual música reproduzir sem escutar anúncios entre as músicas, os usuários free podem reproduzir aleatoriamente a lista de músicas favoritas e, em enquanto escutam as músicas, propagandas e anúncios são reproduzidos entre as músicas.

4.2 Requisitos Não-Funcionais

É importante para a aplicação que o ambiente de desenvolvimento e testagem de códigos seja de fácil manutenibilidade para os desenvolvedores; assim como é importante que a aplicação cliente tenha um bom desempenho e esteja disponível em multiplataformas para amplificar o público-alvo. Portanto, considera-se fundamental que os dados de seus usuários sejam mantidos com segurança e que a aplicação no lado do servidor tenha uma boa performance e seja capaz de terminar tarefas com eficiência.

5 Visão de Casos de Uso

Uma descrição da visão de casos de uso da arquitetura de software. A Visão de Casos de Uso é uma entrada importante para a seleção do conjunto de cenários e/ou casos de uso que são o foco de uma iteração. Ela descreve o conjunto de cenários e/ou os casos de uso que representam alguma funcionalidade central e significativa. Também descreve o conjunto de cenários e/ou casos de uso que possuem cobertura arquitetural substancial (que experimenta vários elementos de arquitetura) ou que enfatizam ou ilustram um determinado ponto complexo da arquitetura.

Os casos de uso deste sistema estão listados a seguir.

* Buscar.
* Adicionar Música à lista de músicas favoritas.
* Remover Música da lista de músicas favoritas.
* Escutar Rádio da música.
* Baixar Podcast.
* Baixar Playlist.
* Baixar Música.
* Escolher Qual Música Reproduzir.
* Logar.
* Fazer Playlist.
    ** Tornar Pública.
    ** Tornar Privada.
    ** Adicionar Música.
    ** Remover Música.

5.1 Casos de uso

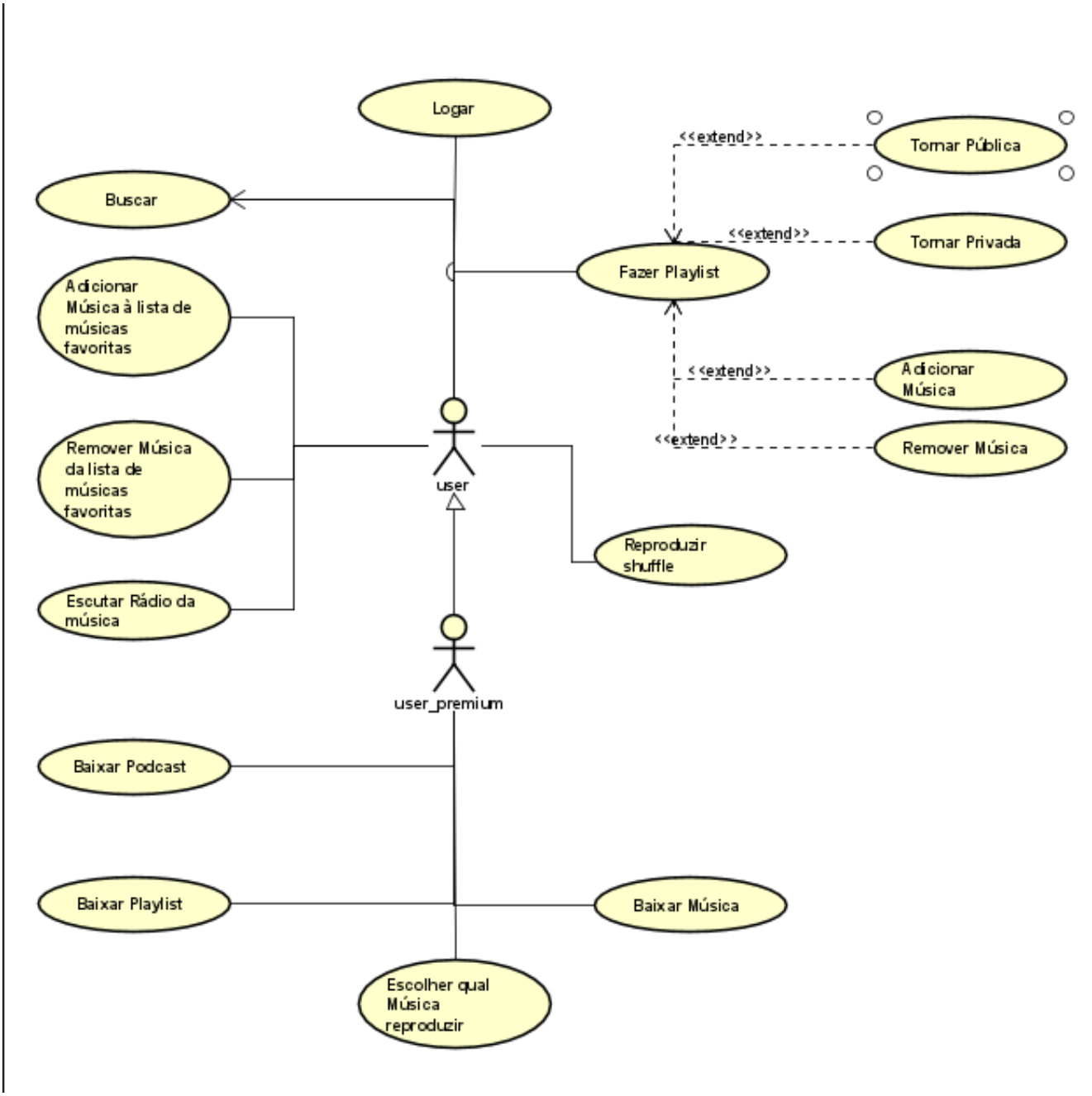

Figura 1. Casos de uso.

A imagem acima demonstra todos os casos de uso listados e suas interações.

5.2 Detalhamento dos Casos de Uso

Uma sinopse do funcionamento e utilidade de cada caso de uso:

Caso de Uso	Detalhes
Buscar	O usuário pode buscar uma música pelo seu nome.
Adicionar Música à lista de músicas favoritas	O usuário pode marcar uma música dos resultados de uma busca como favorita, e essa música aparecerá na sua lista de músicas favoritas.
Remover Música da lista de músicas favoritas	O usuário pode remover uma música de sua lista de músicas favoritas.
Escutar Rádio da Música	O usuário pode iniciar uma playlist rádio com músicas relacionadas a música de início.
Baixar Podcast	O usuário premium pode baixar podcasts em seu dispositivo.
Baixar Playlist	O usuário premium pode baixar playlists em seu dispositivo.
Baixar Música	O usuário premium pode baixar músicas em seu dispositivo.
Escolher Qual Música Reproduzir	O usuário premium pode escolher qual música reproduzir, o usuário free pode escolher reproduzir uma lista de músicas aleatoriamente.
Logar	O usuário precisa logar na aplicação antes de usá-la.
Fazer Playlist	O usuário pode criar uma playlist: lista de músicas.
Tornar Playlist Pública	O usuário pode definir a visibilidade da playlist como pública.
Tornar Playlist Privada	O usuário pode definir a visibilidade da playlist como privada.
Adicionar Música à Playlist	O usuário pode adicionar músicas às suas playlists.
Remover Música da Playlist	O usuário pode remover músicas de suas playlists.

5.3 Casos de Uso Críticos

Abaixo, encontra-se três casos de uso críticos para aplicação e suas descrições.

5.3.1 Caso Crítico 1: Logar

Ator: User

Resumo: O usuário pode se logar informando o número de telefone e senha, ou pela conta do google, facebook ou apple.

Mockup:

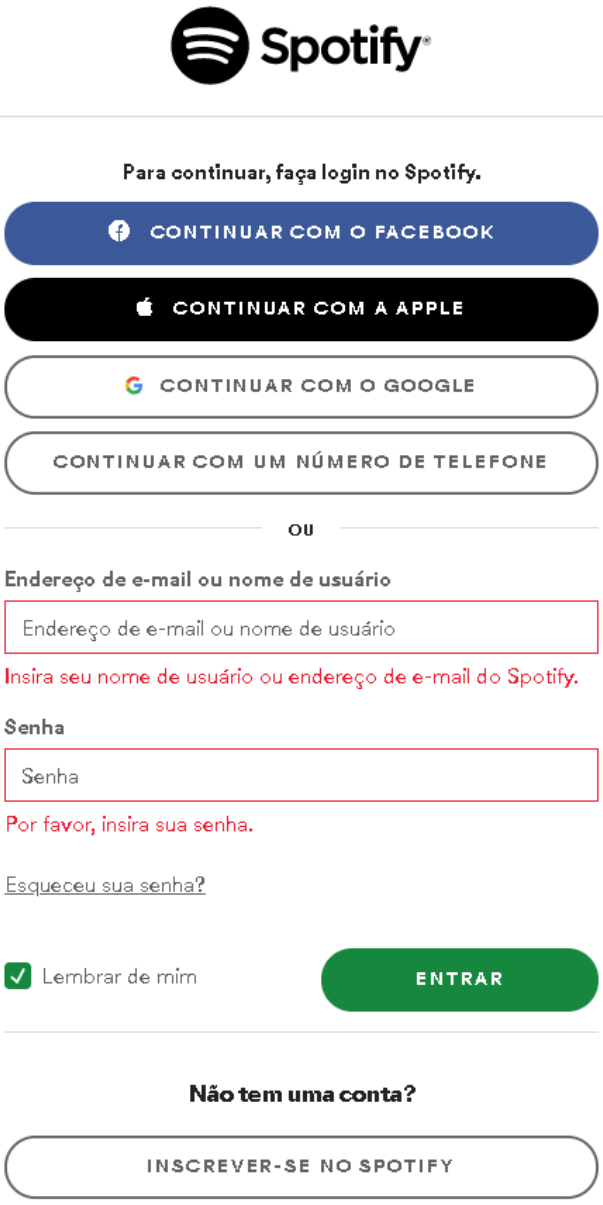

Figura 2. Login spotify.

Diagrama de classe:

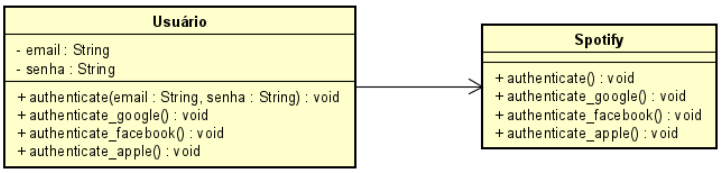

Figura 3. Diagrama de Classes 1.

Diagrama de Sequência:

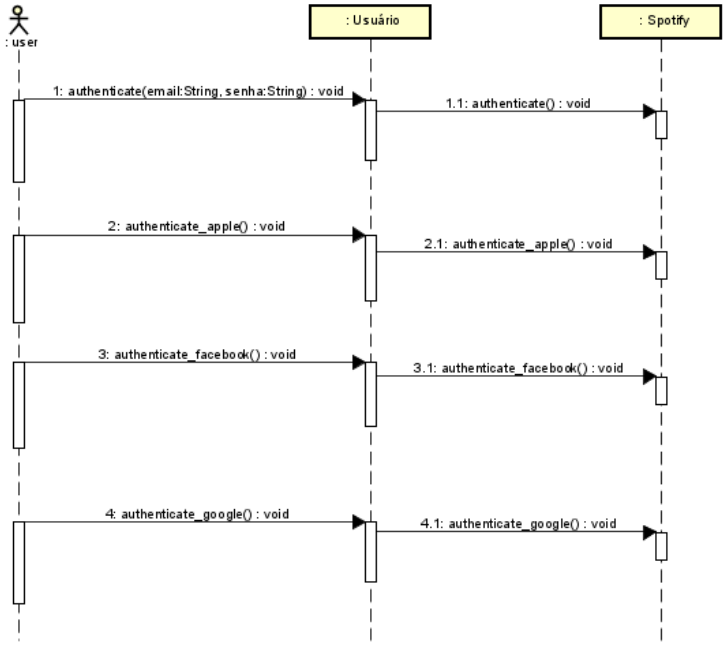

Figura 4. Diagrama de Sequência 1.

5.3.2 Caso Crítico 2: Reproduzir Shuffle

Ator: User

Resumo: O usuário pode reproduzir músicas da sua playlist de forma aleatória.

Mockup:

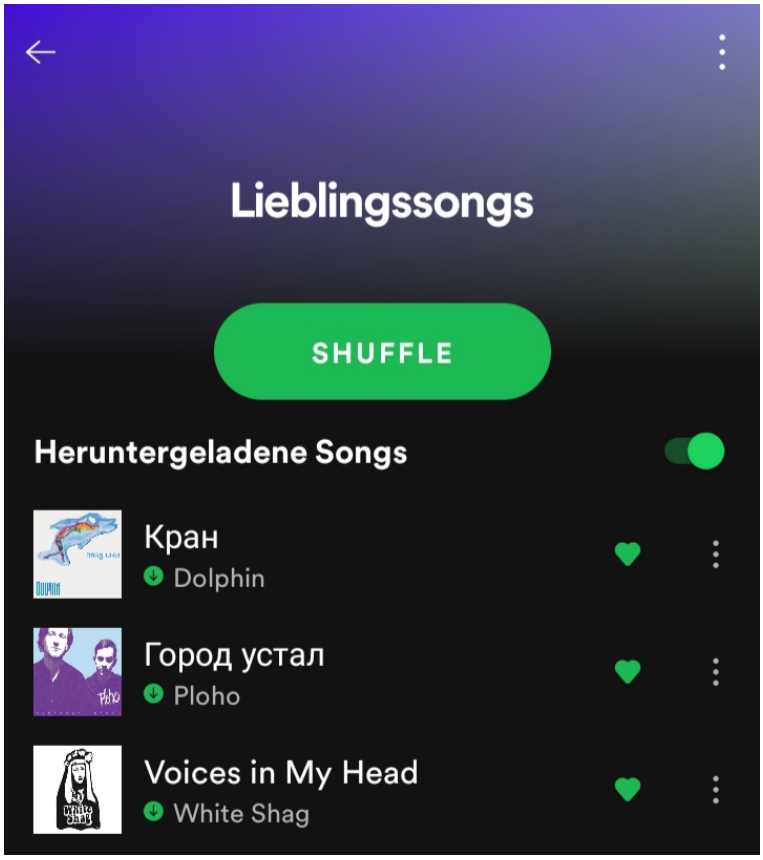

Figura 5. Mockup Shuffle

Diagrama de classe:

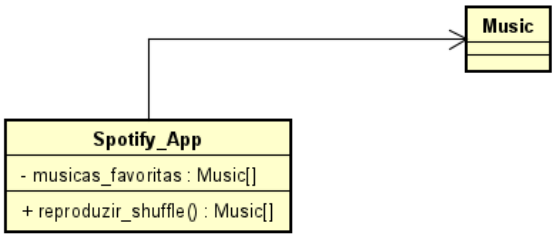

Figura 6. Diagrama de Classes 2.

Diagrama de Sequência:

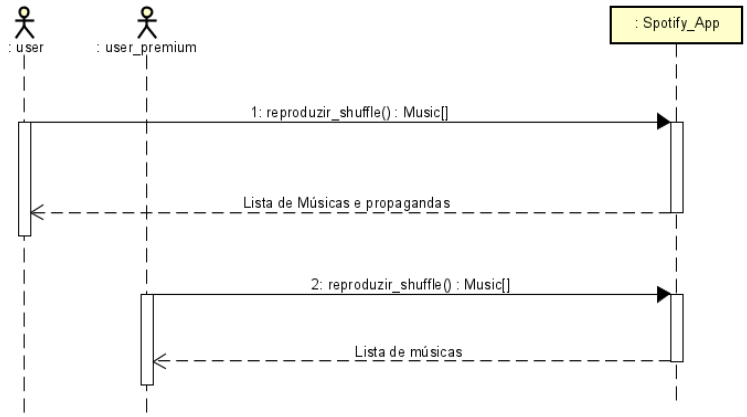

Figura 7. Diagrama de Sequência 2.

5.3.3 Caso Crítico 3: Baixar Música

Ator: User_premium

Resumo: O usuário premium pode fazer o dowload da música desejada.

Mockup:

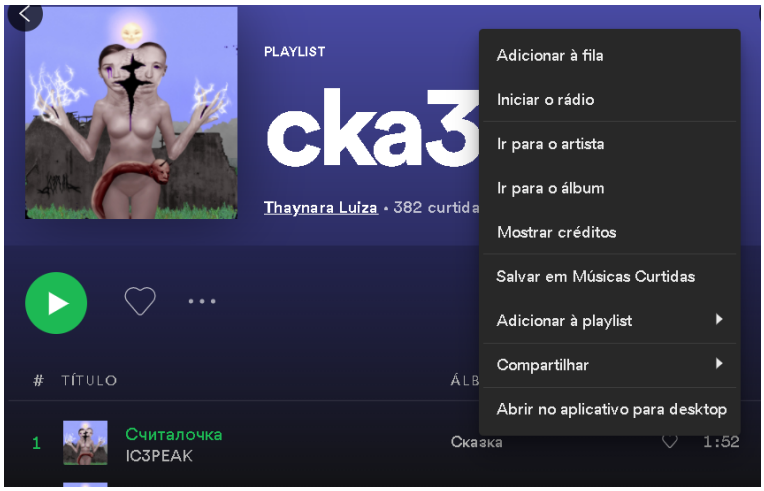

Figura 8. Mockup Baixar Música.

Diagrama de classe:

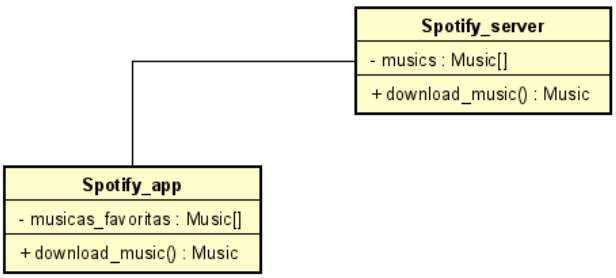

Figura 9. Diagrama de Classes 3.

Diagrama de Sequência:

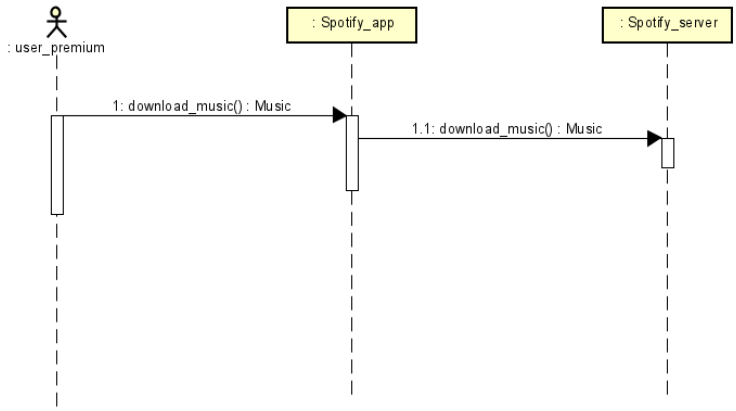

Figura 10. Diagrama de Sequência 3.

6 Padrões Arquiteturais Selecionados

As arquiteturas escolhidas para o projeto Spotify foram a de microserviços e arquitetura em Layers, pois a arquitetura está distribuída geograficamente e, constantemente, é atualizada, bem como seus serviços são demandados de muitos lugares – ademais, a arquitetura em layers divide a aplicação em camadas para então estruturá-la da melhor forma hierárquica possível, de modo a prover uma estrutura de microserviços geograficamente distribuída.

6.1 Acesso do Usuário à Aplicação

O diagrama abaixo demonstra como o usuário se comunica com a aplicação.

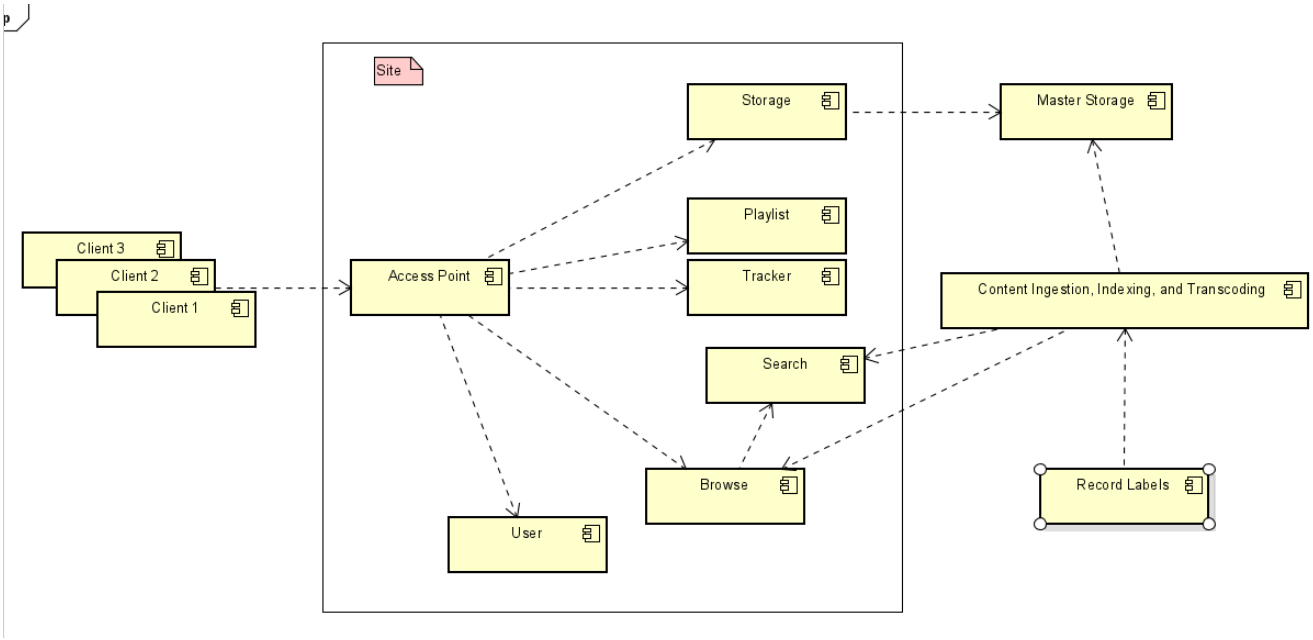

Figura 11. Interação do usuário com a plataforma.

6.2 Localização de Arquivos pela a Aplicação

A figura abaixo demonstra como os arquivos da aplicação são salvos.

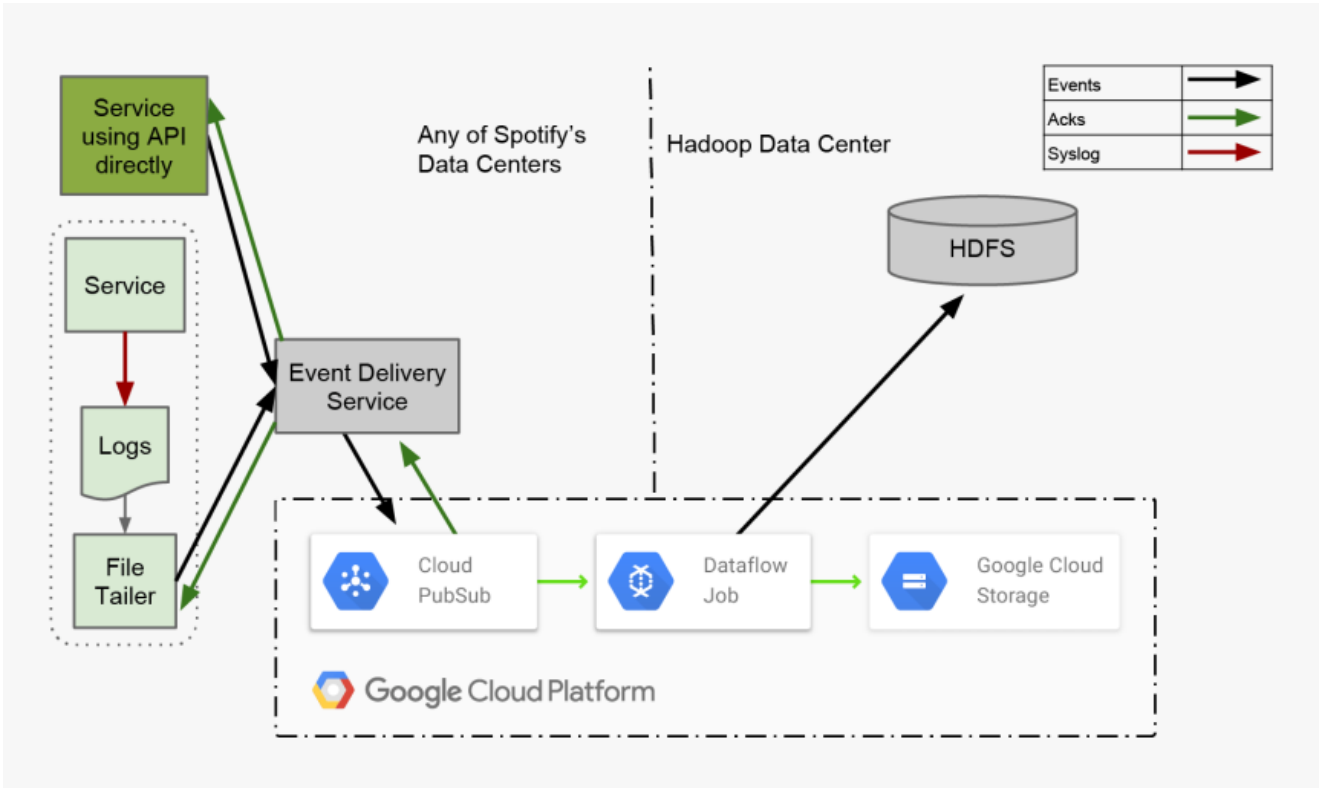

Figura 12. Salvamento de arquivos.

A figura abaixo demonstra como arquivos são requisitados e. no caso das playlists, até modificados.

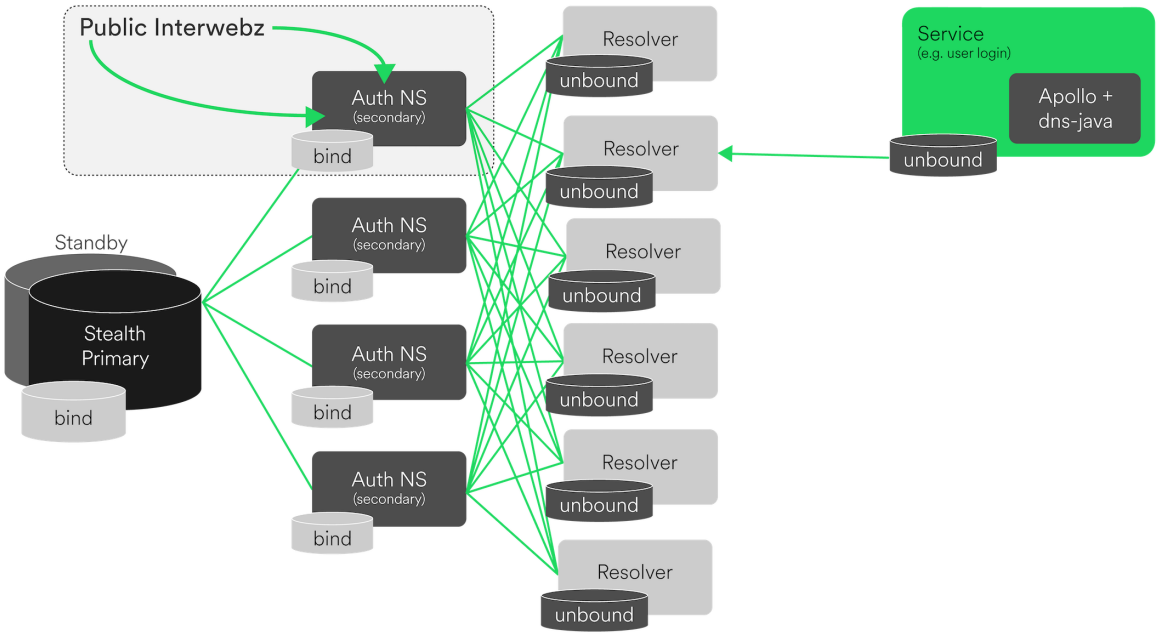

Figura 13. Requisito de arquivo.

7 Arquitetura Proposta

Esta é a arquitetura proposta para a aplicação: uma arquitetura de microserviços em conjunto com a estruturação de seus componentes em layers - que estão distribuídos geograficamente-; e a utilização do serviço de armazenamento distribuído Hadoop, HDFS, para o armazenamento de arquivos construído conforme a arquitetura Space-Based. Veja a figura abaixo para compreender a estruturação da aplicação.

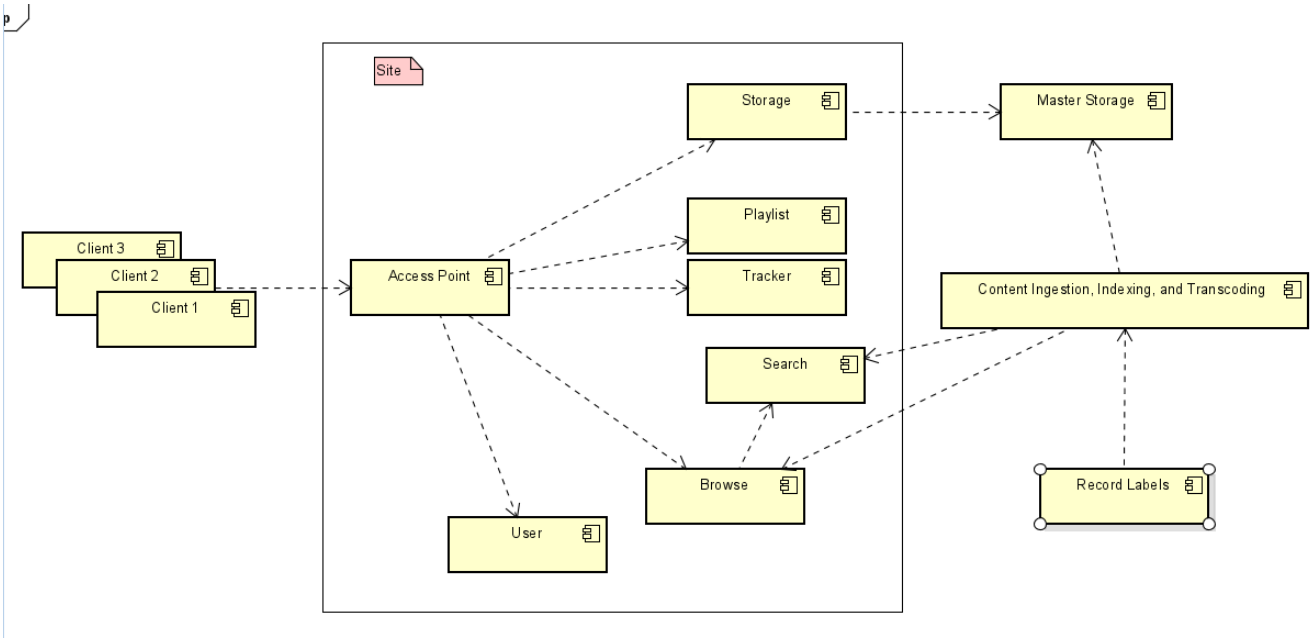

Figura 14. Arquitetura Proposta.

8 Cenários Descritos e Analisados - Análise Arquitetural

Abaixo, esta uma lista dos cenários descritos e suas análises.

Atributo de Qualidade	Descrição do Cenário	
Cenário 1: **Manutenibilidade**	**Fonte**	Aplicação
	Estímulo	Tornar a aplicação de fácil acesso e manutenção
	Artefato	Access Point.
	Ambiente	Controle do comportamento da aplicação
	Resposta	Cobertura de teste e acompanhamento da equipe
	Medida	Time analista observando comportamentos.
Cenário 2: Desempenho	**Fonte**	Aplicação
	Estímulo	Tempo de Resposta Efetivo
	Artefato	Browser, Search e Access Point
	Ambiente	Funcionamento ideal em servidor
	Resposta	Desenvolvimento procedural
	Medida	Processamento com base na performance e tempo tomado.
Cenário 3: Disponibilidade	**Fonte**	Usuário
	Estímulo	Acessibilidade em diferentes plataformas.
	Artefato	Access Point.
	Ambiente	Ser Responsivo em diferentes sistemas
	Resposta	Desenvolvimento de soluções multiplataforma.
	Medida	Implementar as funcionalidades da aplicação em diferentes plataformas.
Cenário 4: Segurança	**Fonte**	Aplicação.
	Estímulo	Segurança dos dados sensíveis do Usuário
	Artefato	User e Access Point.
	Ambiente	Proteção de recursos da aplicação
	Resposta	Seguir os padrões da LGPD
	Medida	Manter os dados tratados e em um local Seguro.
Cenário 5: Performance	**Fonte**	Aplicação
	Estímulo	Funcionar em tempo de execução com eficiência.
	Artefato	Storage, Browse e Access Point.
	Ambiente	Desempenho da aplicação no servidor.
	Resposta	Desenvolvimento de Clean Code e técnicas de programação funcional.
	Medida	Manter o versionamento de Código em repositório.

Figura 15. Cenários descritos.

Análise do Cenário 1	
Sumário	Manutenção no código da aplicação
Objetivo de Negócio	Facilitar a manutenção da aplicação pelas equipes.
Atributo de Qualidade	Manutenibilidade
Abordagem	Análise do comportamento da aplicação pelas equipes de QA.
Riscos	Qualidade da análise
Tradeoffs	Melhorias na aplicação e uso de uma equipe de análise.

Figura 16. Análise Cenário 1.

Análise do Cenário 2	
Sumário	Tempo que o servidor leva para resolver os problemas de um usuário.
Objetivo de Negócio	Prover desempenho para a aplicação no servidor.
Atributo de Qualidade	Desempenho.
Abordagem	Desenvolvimento procedural da aplicação.
Riscos	Nenhum.
Tradeoffs	Tempo de eficiência na aplicação no servidor com o tempo da rede.

Figura 17. Análise Cenário 2.

Análise do Cenário 3	
Sumário	O sistema deve estar disponível em diferentes plataformas.
Objetivo de Negócio	Ofertar a aplicação a um público maior.
Atributo de Qualidade	Disponibilidade
Abordagem	Implementação de soluções gerais e multiplataforma.
Riscos	Eficiência da aplicação em diferentes plataformas.
Tradeoffs	Amplitude da distribuição contra a eficiência.

Figura 18. Análise Cenário 3.

Análise do Cenário 4	
Sumário	A aplicação deve garantir a segurança dos dados do usuário.
Objetivo de Negócio	Ser uma aplicação confiável.
Atributo de Qualidade	Segurança
Abordagem	Uso de validação dos dados e aplicação da LGPD.
Riscos	Vulnerabilidade a ataques não sofridos
Tradeoffs	Complexidade da aplicação em prol da segurança dos dados.

Figura 19. Análise Cenário 4.

Análise do Cenário 5	
Sumário	O sistema deve performar de forma que tenha o melhor desempenho possível no servidor.
Objetivo de Negócio	Garantir que a aplicação resolva os pedidos do usuário.
Atributo de Qualidade	Performance
Abordagem	Desenvolvimento com base em técnicas de programação procedural e funcional e testes da aplicação.
Riscos	O Desenvolvimento de uma aplicação eficiente pode consumir muito tempo.
Tradeoffs	Custo de desenvolvimento com base em melhor performance da aplicação.

Figura 20. Análise Cenário 5.

9 Visão Lógica

A visão lógica da aplicação Spotify é composta pelos 6 componentes abaixo:

* Apresentação: contém as distribuições da aplicação-cliente e como interpretar os dados grafica e logicamente.
* Aplicação: contém a lógica da aplicação e o método main, responsável por coordenar a execução dos componentes da aplicação.
* Dados do Usuário: contém os dados referentes ao usuário.
* Buscar: contém os métodos e funções que operam em cima dos dados da database e retorna uma lista de apresentação das músicas e/ou playlists.
* Músicas e Listas: contém o indexador dos dados do usuário e administra a ordem e estrutura do conteúdo salvo por ele.
* Armazenamento: contém as funcionalidades que operam sobre o sistema de dados distribuídos da aplicação.

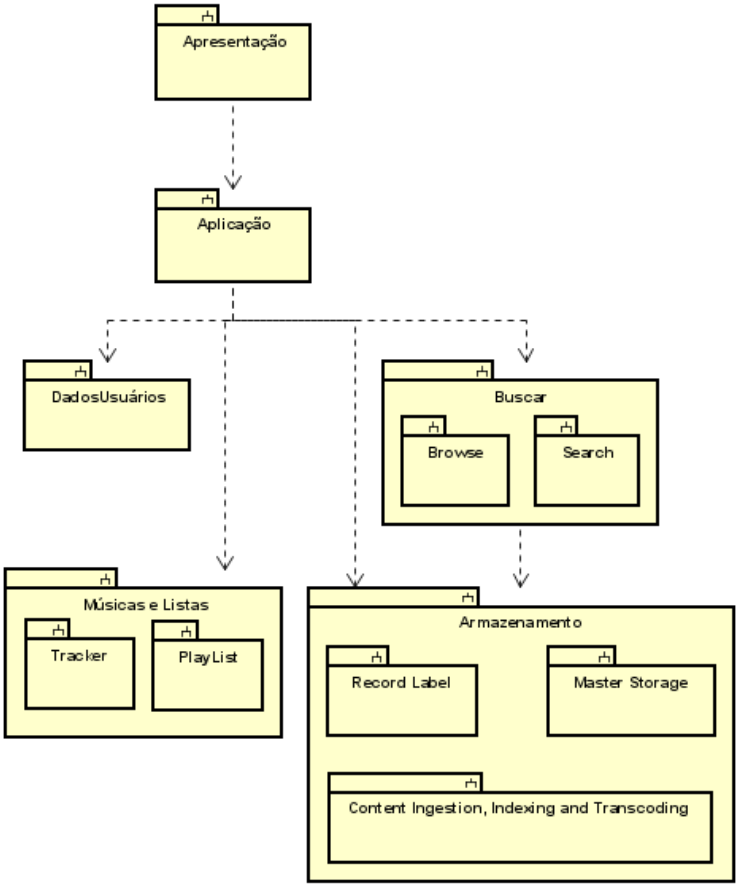

Figura 21. Estrutura da aplicação.

9.1 Pacote de Apresentação

O pacote de apresentação cliente contém as distribuições da aplicação Spotify. Cada pacote dentro do componente apresentação possui a versão para uma plataforma.

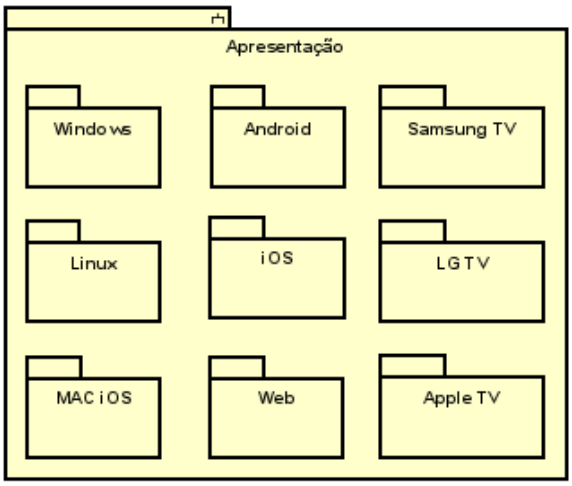

Figura 22. Componente Apresentação.

9.2 Pacote da Aplicação

O pacote aplicação contém handlers de outros componentes com os quais interage, e, também, possui um pacote com Access Point e a classe main, que coordena a execução da aplicação.

9.2.1 Pacote Access Point Contido

O pacote Access Point contém as classes que coordenam a aplicação e interagem com os handlers de outros componentes. Apenas as funções citadas nos casos de uso críticos foram mencionadas no diagrama abaixo. O pacote também fornece uma interface de acesso a aplicação geral para a implementação multiplataforma da aplicação.

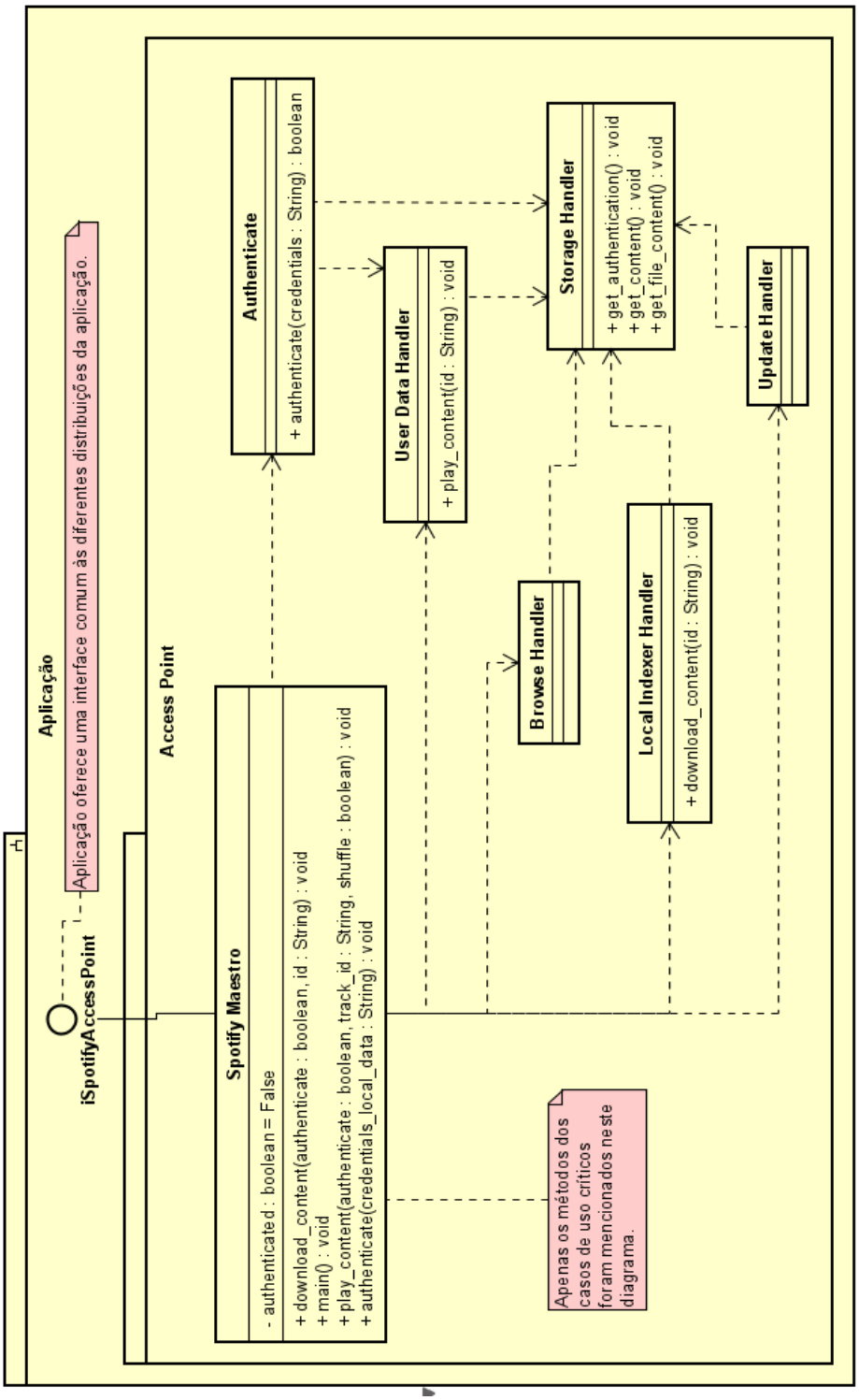

Figura 23. Pacote do Access Point.

9.2.2 Pacotes Contidos no Pacote Aplicação

O pacote aplicação contém os 4 pacotes apontados no diagrama abaixo, todos eles possuem um modo de execução independente, que entra em ação toda vez que o handler das classes Access Point o requisitam.

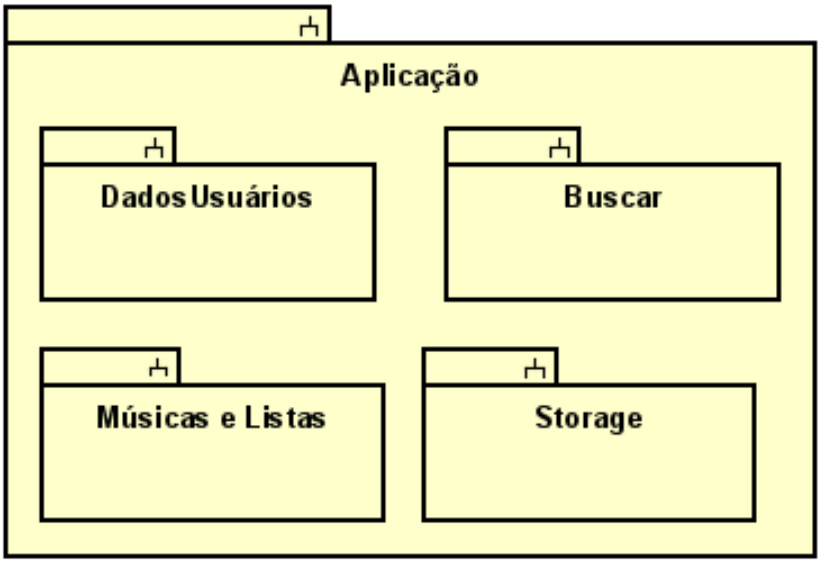

Figura 24. Pacotes contidos no pacote aplicação.

9.3 Pacote de Dados do Usuário

Contém o pacote handler utilizado pelo componente Access Point do diagrama de componentes da arquitetura proposta. Ele administra a interação da aplicação com a parte do armazenamento dedicada aos dados dos usuários.

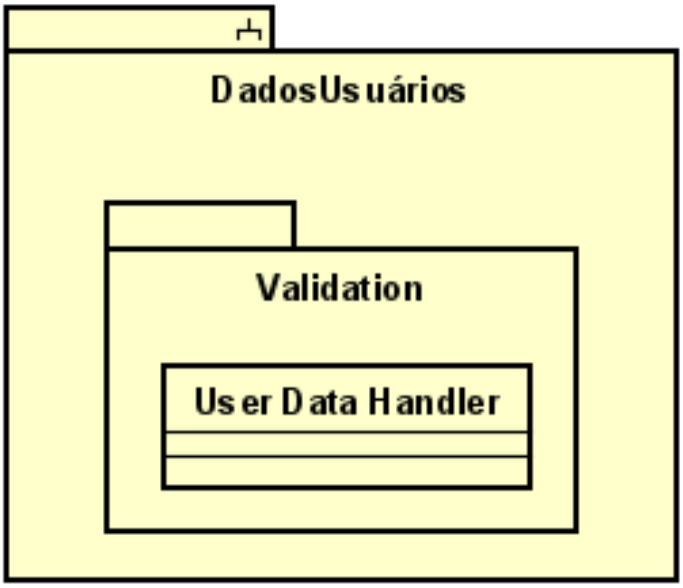

Figura 25. Componente Dados do Usuário.

9.4 Pacote do Componente Buscar

O pacote Buscar contém os componentes Browse e Search do diagrama de componentes da arquitetura proposta. Eles são responsáveis por interagir com o armazenamento das músicas e playlists, e por retorná-las de forma apresentável ao usuário.

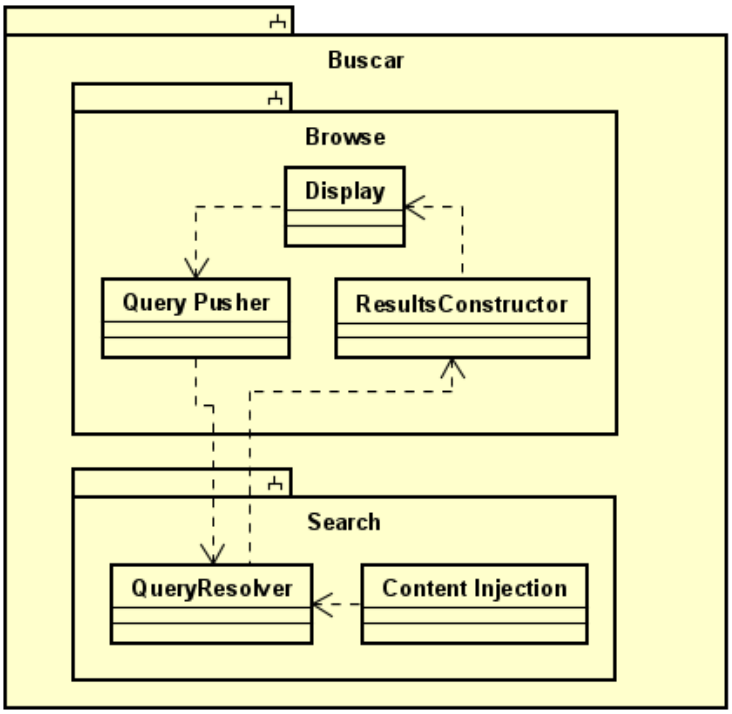

Figura 26. Componente Buscar.

9.5 Pacote de Armazenamento

O pacote armazenamento contém os handlers utilizados pelo Access Point para interagir com o sistema de arquivos HDFS, também contém o handler de adição de novas labels e o pacote de indexação com handler utilizado pelos componentes Browse e Search.

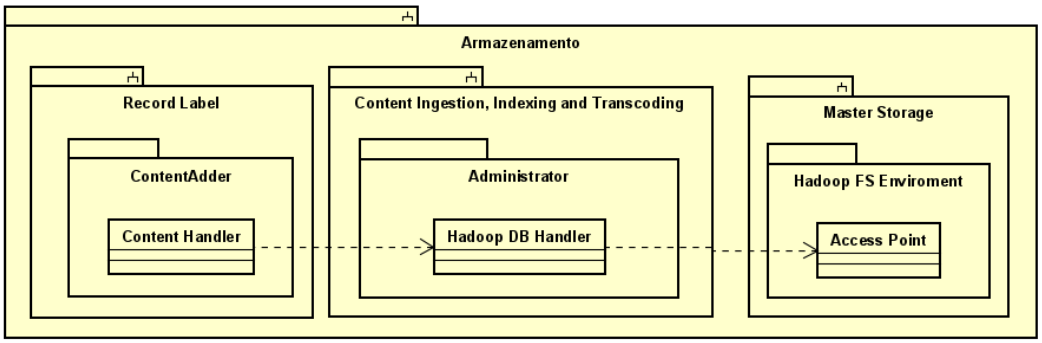

Figura 27. Componente Armazenamento.

10 Visão de Processo

Esta seção descreve a decomposição do sistema em processos. Neste momento, é possível decompor as rotinas da aplicação em subprocessos como demonstrado abaixo.

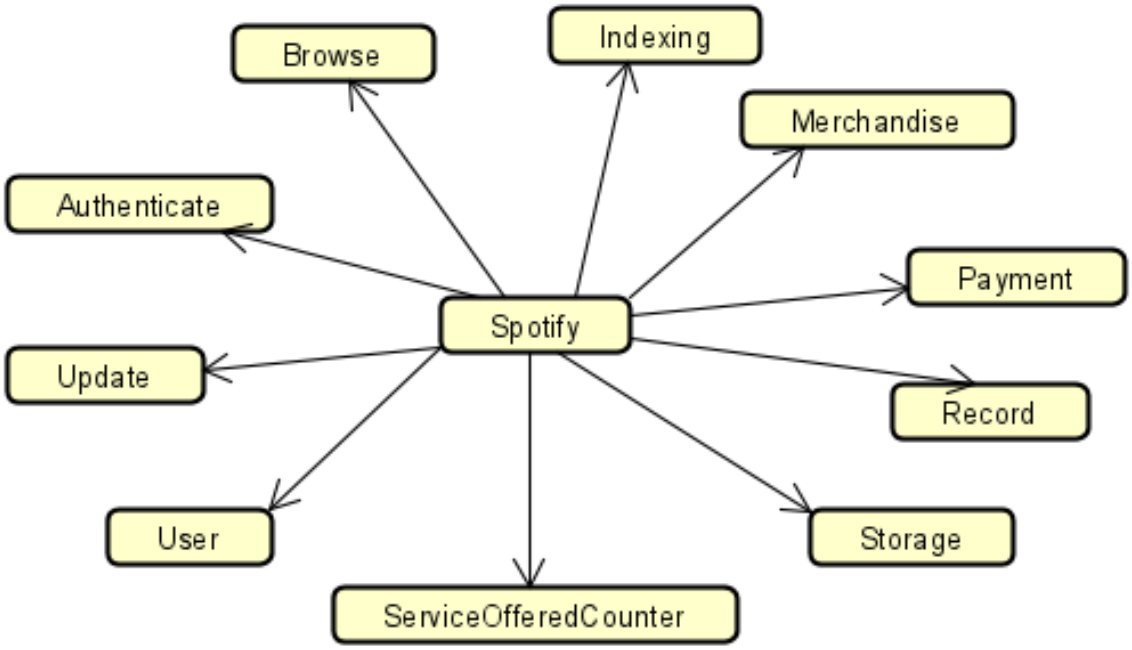

Figura 28. Visão dos processos da arquitetura.

11 Arquitetura X Componentes X Tecnologias

A arquitetura foi estruturada de forma a permitir uma divisão da aplicação em layers, a permitir uma estruturação dos dados como microserviços – onde a estrutura de indexação e manutenção dos conteúdos, HDFS, Hadoop Distributed File System, possibilita a organização dos arquivos em uma arquitetura Space-Based.

As tecnologias usadas no lado das aplicações-cliente variam, pois há uma grande variedade de distribuições. Para as versões Android, é utilizado Java; para iOS, é utilizado Swift; para Windows, MacOS e linux, é utilizado Electron; e, para as distribuições das fabricantes de Smart TVs, é utilizado C++. A aplicação servidor é feita em NODE.js. E, no lado da DataBase, foi utilizado Java com HDFS para armazenar os dados.

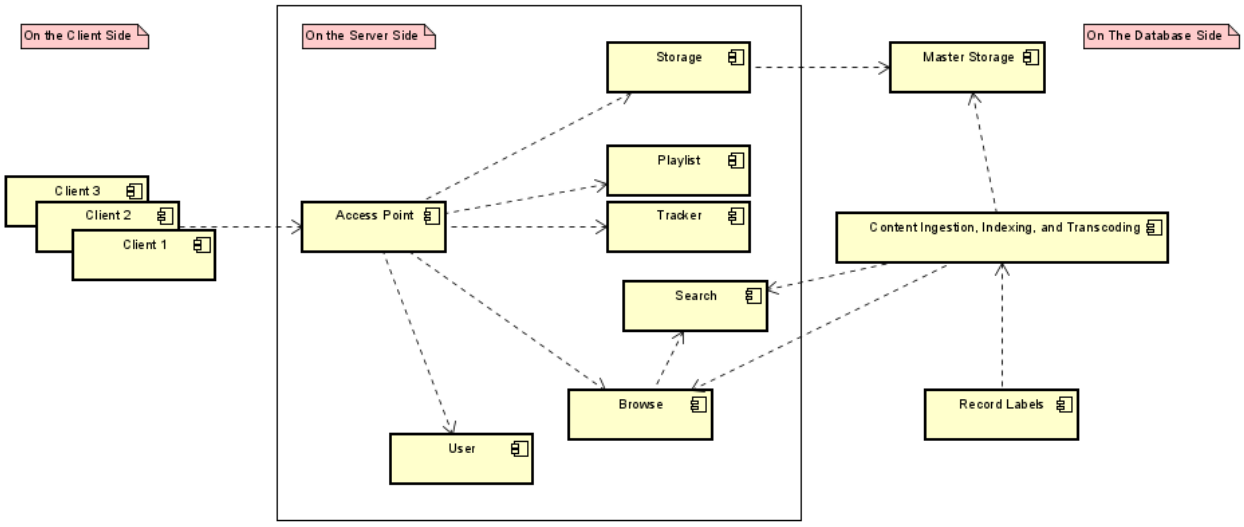

Figura 29. Divisão da aplicação em Layers.

12 Visão de Implantação

A aplicação será implantada em 4 níveis. No primeiro, o cliente, quem solicita serviços à aplicação. No segundo o Resolver local da aplicação que comunica o pedido ao nó authenticate da região. No terceiro, o no authenticate responde ao serviço ou o encaminha para o nível 4. No quarto, a aplicação responde aos pedidos que não estão em âmbito regional, atualizando o cache dos nós pedintes de forma a manter o serviço requisitado para demandas a posteriori. As comunicações entre níveis são feitas por meio da internet e cada nível é um componente da aplicação implementado em um servidor de forma geográfica e lógica. Veja a figura abaixo para compreender a implantação, a figura 13 serve de referência a essa estrutura.

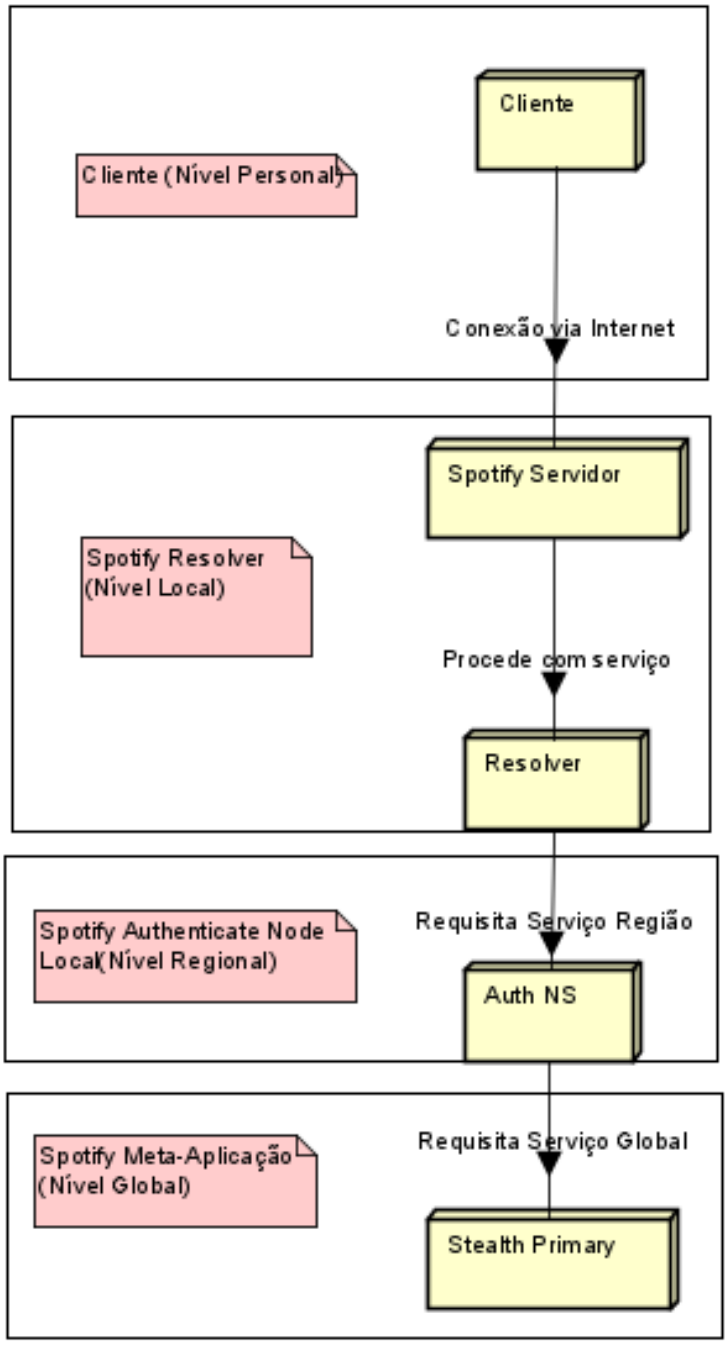

Figura 30. Implementação da arquitetura.

13 Visão de Implementação

O software do servidor reside em 3 camadas distribuídas de forma geográfica e hierárquica. O cliente fornece uma camada de acesso secundária.

14 Tamanho e Desempenho

O software suportará, em escala global, a propagação de serviços a múltiplos usuários em rede simultaneamente.

15 Qualidade

O software administrará serviços distribuídos geograficamente e indexará novos conteúdos conforme forem adicionados; também, promoverá o uso fácil desta arquitetura pelo usuário com interfaces de uso fácil e autodescritiva.

Referências

[1] L. Root. "Spotify's Love/Hate Relationship with DNS". Em: *LYNN ROOT – WORDS* (). DOI: https://www.roguelynn.com/words/spotifys-love-hate-relationship-with-dns/.

[2] A. Martin. "Big Data Spotify Data Architecture". Em: *KindPNG* (). DOI: https://www.kindpng.com/imgv/JTxbbm_big-data-spotify-data-architecture-hd-png-download/.

[3] JetBrains. "What is Spotify's architecture?" Em: *Quora* (). DOI: https://www.quora.com/What-is-Spotifys-architecture?share=1.

[4] Kevin Goldsmith. "GOTO 2015 - Microservices at Spotify - Kevin Goldsmith". Em: *Youtube* (). DOI: https://www.youtube.com/watch?v=7LGPeBgNFuU&feature=emb_logo.

[5] Gonzalo P. "Microservices Architecture at Spotify". Em: *CODEBASE* (). DOI: https://medium.com/codebase/microservices-architecture-at-spotify-beac905e9622.

[6] Spotify Engineering. "Spotify Modernizes Client-Side Architecture to Accelerate Service on All Devices". Em: *Spotify R&D — Engineering* (). DOI: https://engineering.atspotify.com/2020/05/28/spotify-modernizes-client-side-architecture-to-accelerate-service-on-all-devices/.

[7] Spotify Engineering. "When Should I Write an Architecture Decision Record". Em: *Spotify R&D — Engineering* (). DOI: https://engineering.atspotify.com/2020/04/14/when-should-i-write-an-architecture-decision-record/.