DOCUMENTAÇÃO DE DESENVOLVIMENTO DE SOFTWARE

Sonz

Gabriel Viegas Capecci – 82213442
Gabriel Mariotti Higa – 822141216
Gabriela Dardis Rodrigues – 822141330
Igor Britto - 822141647
Maria Fernanda Mendes Tobias – 822137255
Renato Peduto Filho - 822126254

2025-1

Documentação de Desenvolvimento de Software

1. Iniciação do Projeto

1.1 Termo de Abertura do Projeto

Documento que autoriza formalmente o início do projeto e define o gerente responsável, objetivos, justificativas e stakeholders principais.

O projeto tem como objetivo desenvolver uma plataforma de streaming de música digital que permita a todo tipo de usuários ouvirem músicas sob demanda em dispositivos móveis e desktops, apenas possuindo o dispositivo e acesso a internet.

Com a grande expansão da tecnologia a demanda por consumo digital cresceu exponencialmente, assim se enxergou uma oportunidade de monetizar uma plataforma de streaming através de planos de assinaturas e publicidades pagas.

O gerente de projeto será nomeado pela liderança executiva tendo em sua autonomia, liderar equipes de produto, engenharia, jurídico e marketing, além de monitorar negociações com produtoras e artistas.

1.2 Registro das Partes Interessadas

Processo de identificação de indivíduos, grupos ou organizações que podem afetar ou serem afetados pelo projeto.

- **Usuários gratuitos:** Usuários que desejam acesso sem custo, mesmo que devam ver e ouvir as publicidades;
- **Usuários premium:** Usuários que desejam uma experiência sem interrupções, com qualidade e exclusividade;
- **Gravadoras e artistas:** Usuários que desejam adicionar seu conteúdo na plataforma tendo em vista uma monetização e visibilidade;
- **Equipe técnica:** responsável por garantir escalabilidade e segurança;
- **Equipes de negócios e marketing:** foco em engajamento, crescimento e ROI;

- **Executivos/investidores:** Usuários que desejam um retorno financeiro e melhorar sua posição de mercado.

2. Planejamento do Projeto

2.1 Plano de Gerenciamento do Projeto

Documento que consolida todos os planos auxiliares e guia a execução, monitoramento e encerramento do projeto.
Será utilizado o modelo de desenvolvimento ágil SCRUM com sprints de duas semanas, além das outras sprints. O plano abrange a integração entre áreas como engenharia, UX/UI, licenciamento e marketing. Define que as entregas mínimas (MVP) incluirão: cadastro/login, streaming básico, biblioteca de músicas e playlists personalizadas.

2.2 Estimativas de Projeto

As estimativas para o desenvolvimento da plataforma Sonz foram baseadas no uso da metodologia ágil SCRUM, estruturada em sprints quinzenais. O projeto foi planejado para ser concluído em 7 sprints, totalizando cerca de 14 semanas, o que compreende todas as etapas desde a preparação do ambiente, desenvolvimento do MVP até a validação final com usuários reais.

A estimativa de tempo considera:

- A implementação das funcionalidades principais do MVP;
- Atividades de testes automatizados e manuais em paralelo ao desenvolvimento;
- Realização de testes de performance, usabilidade e validação.

A estimativa de recursos humanos inclui:

- Um gerente de projeto;
- Desenvolvedores distribuídos por áreas (streaming, playlists, recomendações, etc.);
- Equipe de QA composta por líder, analistas, engenheiro de automação e analista de UX.

A estimativa de ferramentas e infraestrutura contempla:

- Ambientes dedicados (desenvolvimento, staging e produção);
- Ferramentas de automação de testes e CI/CD (Selenium, GitHub Actions); Monitoramento de desempenho (Firebase, Power BI);
- Gestão e comunicação com JIRA, Slack e Confluence.

2.3 Declaração do Escopo

2.3.1. Justificativa do Projeto

Com a grande expansão da tecnologia a demanda por consumo de música digital tem crescido exponencialmente com a popularização de dispositivos móveis e conexões rápidas à internet, assim se enxergou uma oportunidade de monetizar uma plataforma de streaming através de planos de assinaturas e publicidades pagas.
Este projeto visa desenvolver uma plataforma de streaming de música digital que permita a todo tipo de usuários ouvirem músicas sob demanda em dispositivos móveis e desktops, apenas possuindo o dispositivo e acesso a internet.

2.3.2. Objetivos do Projeto

Desenvolver um MVP (versão mínima viável) de uma plataforma de streaming de música que permita ouvir faixas sob demanda, fazer buscas, criar playlists e receber recomendações personalizadas. A ideia é que o app funcione bem em web e mobile, e já tenha integração com redes sociais, com foco em atrair usuários e validar o modelo de negócio.

2.3.3. Entregas Principais

O projeto vai entregar uma versão inicial (MVP) com os seguintes recursos:

- Plataforma acessível via web e dispositivos móveis;
- Catálogo de músicas com opção de busca e filtros;
- Player de música com funções básicas (play, pause, próxima faixa);
- Criação e compartilhamento de playlists;

- Sistema de recomendação automática de músicas;
- Integração com redes sociais (login e compartilhamento).

2.3.4. Critérios de Aceitação

Para considerar o projeto pronto, os seguintes pontos precisam estar funcionando:

- Plataforma sendo executada normalmente no computador e no celular;
- O usuário consegue procurar músicas e usar filtros;
- O player toca música sem travar e com os botões funcionando;
- Dá pra criar playlists e compartilhar seu link para outras pessoas;
- O sistema sugere músicas com base no que o usuário escuta (através do algoritmo).

2.3.5. Fora do Escopo (MVP)

Por ser um MVP, algumas funções mais avançadas como vídeos, não vão ser incluídas nesse primeiro momento incluindo:

- Podcasts;
- Videoclipes.

2.3.6. Premissas

- A plataforma só vai funcionar com conexão à internet;
- O time de desenvolvimento vai ser pequeno, mas com pessoas especializadas (time SCRUM);
- Teremos um catálogo de músicas liberado para uso;
- A primeira forma de monetização será com anúncios e teste de plano pago.

2.3.7. Restrições

- O orçamento é limitado, então não dá pra investir em tudo de uma vez;
- A gente tem um prazo curto para lançar e testar a ideia;
- Não haverá suporte offline.

2.4 Estrutura Analítica do Projeto (WBS)

A Estrutura Analítica do Projeto serve para dividir o projeto em partes menores, para facilitar o controle e a organização do que precisa ser feito. No nosso caso, a ideia é criar uma plataforma de streaming de música. Abaixo segue a divisão do que o projeto vai envolver:

Início do Projeto

- Criar o termo de abertura;
- Identificar as pessoas interessadas no projeto (stakeholders).

Planejamento

- Criar o plano do projeto;
- Planejar o que vai estar no escopo;
- Definir o que vai ter no MVP;
- Organizar o backlog e sprints com o time.

Desenvolvimento da Plataforma

- Backend (parte do sistema que roda por trás):
  - Login e cadastro;
  - Streaming de músicas;
  - Banco de dados das músicas;
- Frontend Web (interface pro usuário no computador):
  - Tela de login;
  - Tela de busca de música;
  - Reprodutor de música.
- Aplicativo Mobile (para celular):
  - Versão Android;
  - Versão iOS.

Funcionalidades do MVP

- Player com botões básicos (play, pause, pular música);
- Busca com filtros;

- Criar e compartilhar playlists;
- Recomendação automática de músicas.

Testes e Ajustes

- Testar se todas funcionalidades estão funcionando em conjunto;
- Receber feedback dos usuários e stakeholders;
- Ajustar as coisas que não estiverem boas e estejam apresentando problemas.

Lançamento

- Publicar a plataforma;
- Monitorar seu uso e resolver possíveis problemas.

2.5 Plano de Gerenciamento de Requisitos

Esse plano serve para mostrar como vamos descobrir o que o sistema precisa ter e como vamos garantir que isso seja feito.

- Vamos conversar com os stakeholders a fim de entender o que eles esperam da plataforma;
- Os requisitos serão colocados no backlog no JIRA;
- O Product Owner vai ajudar a decidir o que é mais importante e suas prioridades;
- A cada sprint, vamos validar se as entregas atendem o que os usuários queriam;
- Se aparecer algo novo, a gente avalia nas reuniões de planejamento e vê se vale a pena incluir.

2.6 Plano de Comunicação

Aqui a gente define como todo mundo envolvido vai se manter informado durante o projeto.

- A equipe vai se comunicar diariamente pelo **Slack**;
- As tarefas e entregas vão ser controladas pelo **JIRA**;
- Toda semana vai ter reunião rápida de acompanhamento;

- A cada duas semanas, vamos fazer reuniões maiores (Sprint review e Sprint retrospective);
- Os líderes e investidores vão receber relatórios simples com o que foi feito, o que falta e os problemas encontrados.

2.7 Plano de Gerenciamento de Riscos

Esse plano é para identificar problemas que podem atrapalhar o projeto e pensar em soluções antes que eles aconteçam.

- **Atraso nas entregas**
    - **Chance de acontecer:** Alta;
    - **Impacto:** Alto;
    - **Ação planejada:** Reduzir o escopo, se necessário, e focar nas funcionalidades mais importantes para garantir a entrega do MVP no prazo.

- **Problemas com licenciamento de músicas**
    - **Chance de acontecer:** Média;
    - **Impacto:** Alto;
    - **Ação planejada:** Antecipar as negociações com produtoras e manter um catálogo alternativo de músicas já liberadas.

- **Poucos usuários no início do lançamento**
    - **Chance de acontecer:** Média;
    - **Impacto:** Médio;
    - **Ação planejada:** Investir em divulgação nas redes sociais e coletar feedback dos primeiros usuários para melhorar a plataforma.

- **Erros e falhas no sistema**
    - **Chance de acontecer:** Alta;

    - **Impacto:** Alto;
    - **Ação planejada:** Realizar testes constantes durante as sprints e corrigir os problemas rapidamente assim que forem identificados.

- **Falha na integração com redes sociais**

    - **Chance de acontecer:** Média;
    - **Impacto:** Baixo;
    - **Ação planejada:** Fazer testes com as APIs das redes sociais e manter uma versão alternativa sem a integração, se for necessário.

3. Planejamento dos Testes

3.1 Cronograma de atividades

O cronograma será alinhado com as sprints quinzenais do Scrum. Considerando um período estimado de 3 a 4 meses para entrega do MVP, temos:

Semana	Sprint	Teste
1-2	1	Definição de casos de teste, setup de ambiente, testes unitários para login/cadastro
3-4	2	Testes unitários e de integração para player básico
5-6	3	Testes de busca, filtros e criação de playlists
7-8	4	Testes de recomendação automática e integração com as redes
9-10	5	Testes de sistema (end-to-end), testes de performance
11-12	6	Testes de usabilidade (UX) com usuários reais
13-14	7	Validação final

3.2 Alocação de Recursos

Obteve-se as seguintes alocações de recursos tanto no âmbito de pessoas quanto em um ambiente mais técnico:

Equipe de QA (Qualidade)

1. Líder de QA: Planejamento, coordenação e supervisão.
2. Analistas de Teste: Escrita de casos de teste, execução de testes manuais e automatizados.
3. Engenheiro de Testes Automatizados: Automatização com Selenium e integração com GitHub.
4. Analista de UX: Planejamento e execução de testes com usuários reais.

Recursos Tecnológicos

- Ferramentas: Selenium (testes UI), GitHub (CI/CD), JIRA (gestão de tarefas), Confluence (relatórios).
- Ambientes: Desenvolvimento, Staging e Produção.
- Dispositivos de Teste: Navegadores variados (Chrome, Firefox, Safari), dispositivos Android e iOS.

3.3 Marcos do Projeto

Os marcos do projeto estão diretamente ligados com as finais das Sprints mencionadas. Com isso, verificam-se os seguintes tópicos:

Marco	Descrição	Final de qual Sprint?
M1	Conclusão dos testes mencionados e configuração do ambiente	1
M2	Finalização de ao menos 50% dos testes unitários e de integração	3

M3	Testes automatizados finalizados em mais de 80% dos casos	5
M4	Validação dos usuários em relação a usabilidade da plataforma	6
M5	Aprovação final do MVP	7

3.4 Casos de Teste (Detalhados)

ID	Caso de Teste	Entrada	Resultado Esperado	Tipo
CT01	Login com credenciais válidas	Usuário e senha válidos	Redireciona para tela inicial	Funcional
CT02	Login com credenciais inválidas	Usuário e senha inválidos	Exibir mensagem de erro: "Senha incorreta"	Funcional
CT03	Login com usuário não cadastrado	Usuário e senha inexistente	Exibir mensagem: "Usuário não encontrado"	Funcional
CT04	Busca por música existente	"Travis Scott"	Lista com resultados relevantes	Funcional

CT05	Buscar música inexistente	Buscar por termo que não existe	Exibir mensagem: "Nenhuma música encontrada"	Funcional
CT06	Criar nova playlist	Nome da playlist	Playlist criada com sucesso	Funcional
CT07	Adicionar música à playlist	Selecionar música e adicionar na playlist desejada	Música aparece na playlist	Funcional
CT08	Remover música da playlist	Selecionar música e remover da playlist desejada	Música removida da playlist	Funcional
CT09	Reproduzir música	Selecionar música	Reprodução inicia corretamente	Integração
CT10	Pausar música	Pressionar o botão de “pause”	Música para de tocar	Integração
CT11	Avançar e retroceder música	Pressionar o botão de avançar ou retroceder	Música muda conforme ação	Integração

CT12	Recomendação com base em histórico	Histórico de audições	Lista personalizada de músicas	Funcional
CT13	Layout responsivo no mobile	Acessar via smartphone	Layout se adapta corretamente	Interface
CT14	Microsserviços estão se comunicando	Teste de API entre serviços	Comunicação estabelecida com sucesso	Integração
CT15	Logout do sistema	Pressionar o botão de “logout”	Usuário é deslogado e redirecionado para login	Funcional

3.5 Roteiro de Testes

Testes de Login e Autenticação

Caso de Teste	Objetivo	Passos	Resultado Esperado
CT01 - Login com credenciais válidas	Verificar autenticação correta	1. Abrir página de login. 2.Informar usuário/senha válidos. 3. Clicar em "Entrar"	Usuário é autenticado e redirecionado para a tela principal

CT02 - Login com senha incorreta	Garantir falha em credenciais inválidas	1. Informar usuário válido. 2.Informar senha incorreta. 3. Tentar entrar	Exibir mensagem de erro: "Senha incorreta"
CT03 - Login com usuário não cadastrado	Garantir falha ao usuário inexistente	1. Informar usuário inexistente. 2. Tentar entrar	Exibir mensagem: "Usuário não encontrado"

Testes de Busca de Música

Caso de Teste	Objetivo	Passos	Resultado Esperado
CT04 - Buscar música existente	Verificar busca funcional	1. Acessar a página de busca. 2.Digitar nome de música existente. 3.Confirmar busca	Exibir lista de músicas com o termo exibida
CT05 - Buscar música inexistente	Garantir resposta correta para música não cadastrada	1. Buscar por termo que não existe	Exibir mensagem: "Nenhuma música encontrada"

Testes de Playlists

Caso de Teste	Objetivo	Passos	Resultado Esperado
CT06 - Criar nova playlist	Validar criação de playlist	1. Na tela de playlists, clicar "Criar nova" 2. Informar nome 3. Salvar	Playlist criada e exibida na lista
CT07 - Adicionar música à playlist	Verificar adição de música na playlist	1. Selecionar playlist 2. Adicionar música 3. Confirmar	Música aparece na playlist
CT08 - Remover música da playlist	Validar remoção de música	1. Selecionar música na playlist 2. Remover música	Música removida da playlist

Testes de Reprodução

Caso de Teste	Objetivo	Passos	Resultado Esperado
CT09 - Reproduzir música	Verificar funcionalidade de play	1. Selecionar música 2. Clicar em "Play"	Música começa a tocar
CT10 - Pausar música	Verificar funcionalidade de pausa	1. Enquanto música toca 2. Clicar em "Pause"	Música para de tocar

CT11 - Avançar e retroceder música	Validar controle de avanço e retrocesso	1. Enquanto música toca 2. Clicar em avançar/retroceder	Música muda conforme ação

Testes de Recomendação

Caso de Teste	Objetivo	Passos	Resultado Esperado
CT12 - Exibir recomendações baseadas no histórico	Validar algoritmo básico de recomendação	1. Ouvir músicas variadas 2. Acessar página de recomendações	Músicas relacionadas são exibidas

Testes de Logout

Caso de Teste	Objetivo	Passos	Resultado Esperado
CT13 - Logout do sistema	Garantir encerramento de sessão seguro	1. Usuário logado, clicar em "Logout"	Usuário é deslogado e redirecionado para login

3.6 Critérios de Aceitação

1. Todas as funcionalidades do MVP foram implementadas e testadas;
2. Interface amigável e funcional;
3. Sem erros críticos;

4. Feedback positivo de pelo menos 20 usuários.

3.7 Recursos Necessários

1. Equipe: 2 testadores;
2. Desktops ou smartphones disponíveis;
3. Internet estável;
4. Acesso ao código-fonte e ambientes;
5. Ferramentas de testes automatizados;
6. Documentação.

3.8 Riscos e Mitigações

Risco	Impacto	Mitigação
Falta de tempo para testes	Alto	Início antecipado dos testes em paralelo ao desenvolvimento
Ambiente instável	Médio	Utilizar containers/Docker para padronização
Equipe inexperiente com testes	Médio	Capacitação especializada e uso de boas práticas
Mudança no escopo durante os testes	Alto	Garantir a rastreabilidade dos requisitos

4. Execução do Projeto

4.1 Documentos Técnicos do Desenvolvimento

Requisitos Funcionais e Não Funcionais:

1. **Funcionais:**
   Streaming de música;
   Criação de playlists;
   Recomendação baseada no histórico de reprodução;
   Integração com smart speaker;
   Reprodução de músicas e controle do player;

2. **Não Funcionais:**
   Latência mínima para streaming;
   Alta disponibilidade;
   Compatibilidade multi-plataforma: Incluindo iOS, Android, e web;
   Reprodução em segundo plano;
   Cobertura total dos requisitos com testes.

Arquitetura de Software:

1. Arquitetura de microsserviços para escalabilidade e manutenção, com serviços dedicados para streaming, biblioteca, perfis e recomendações. Uso de CDN para acelerar a entrega do conteúdo.

Especificações Técnicas:

1. Backend em Python e Java, front end em React, HTML e CSS. Além do Apache Kafka para mensagens em tempo real. APIs RESTful para comunicação entre serviços.

Documentação de Segurança:

1. Autenticação via OAuth 2.0 e criptografia ponta a ponta dos dados, com HTTPS em todas as comunicações. Documentação e padrões são armazenados em repositórios versionados como GitHub, garantindo controle das alterações. Jira é usado para gerenciar tarefas e manter a documentação alinhada ao desenvolvimento.

4.2 Plano de Qualidade do Software

Objetivos de Qualidade:

1. Interface intuitiva, desempenho eficiente e disponibilidade acima de 99,9%.

Critérios de Aceitação:

1. Cobertura mínima de 95% em testes unitários e sem regressões após atualizações.

Processos de Teste:

1. Unitários, integração, stress, load e usabilidade com grupos de usuários.

Ferramentas e Métodos:

1. Selenium para testes automatizados e GitHub para integração e entrega contínua (CI/CD).
2. Práticas como integração contínua, revisão de código e monitoramento constante garantem estabilidade, permitindo correções rápidas e minimizando riscos.

4.3 Plano de Recursos Humanos

Organização:

1. Gerente de projeto supervisiona e comunica com stakeholders. Equipes divididas em squads por funcionalidade (streaming, perfis, etc.), além de QA e DevOps.

Alocação:

1. Equipes alocadas conforme etapas, com revisões trimestrais para ajustes.

Treinamento:

1. Workshops mensais sobre novas tecnologias e treinamentos em frameworks e segurança.

Comunicação:

1. Reuniões diárias de stand-up e uso de Slack e Jira para comunicação e gestão de tarefas. A equipe reúne desenvolvedores, engenheiros de DevOps, analistas de qualidade e gerentes, organizados em squads para garantir agilidade e integração técnica.

5. Monitoramento e Controle

5.1 Relatórios de Progresso

Durante as diversas fases de desenvolvimento do Sonz, os relatórios de progresso atuaram como o pulso do projeto. Eles forneciam uma visão clara e periódica do status das atividades, dos recursos consumidos e dos desafios enfrentados. Para o Sonz, esses relatórios teriam sido cruciais para:

- **Acompanhamento da Implementação de Funcionalidades:** Relatórios detalham o avanço na criação de funcionalidades-chave, como o módulo de streaming de áudio, a criação e gestão de playlists, a integração com diferentes dispositivos e o suporte a podcasts. Métricas de progresso, como percentual de conclusão das histórias de usuário ou épicos, teriam sido utilizadas para visualização.
- **Controle de Qualidade e Desempenho:** Relatórios periódicos teriam incluído métricas de qualidade do código, como a taxa de bugs encontrados por funcionalidade, a cobertura de testes automatizados e o tempo de resposta do aplicativo.
- **Análise de Cronograma e Orçamento:** O progresso era constantemente comparado com o planejamento original. Relatórios visuais teriam sido usados para indicar se o desenvolvimento de novas funcionalidades ou a otimização de existentes estavam dentro do prazo e do orçamento. Por exemplo, se o desenvolvimento do algoritmo de recomendação de músicas estivesse atrasado, um relatório de progresso destacaria esse desvio, permitindo à gerência realocar recursos ou redefinir prazos.

- **Gestão de Riscos e Impedimentos:** Os relatórios também eram a ferramenta para comunicar novos riscos identificados, como uma dificuldade na integração com um novo sistema operacional, problemas de licenciamento de músicas, etc. Isso permitia que a liderança tomasse decisões informadas para resolver os impedimentos e manter o fluxo de trabalho.

Esses relatórios, gerados regularmente (semanalmente, quinzenalmente, ou em cadências ágeis), teriam fornecido a transparência necessária para todas as partes interessadas – desde os desenvolvedores até os executivos – garantindo que o projeto Sonz se mantivesse alinhado com seus objetivos e visão de produto.

ID	Descrição Resumida	Status	Squad Responsável	Observações
RF-001	Criação de playlists personalizadas	Implementado	Squad Playlists	Funcionalidade disponível nas versões desktop e mobile.
RF-002	Recomendação baseada no histórico de reprodução	Em desenvolvimento	Squad Discover	Ajustes no algoritmo de sugestão baseados em testes A/B.
RF-003	Integração com smart speakers	Planejado	Squad Connect	Em fase de análise de compatibilidade com APIs da Alexa e Google Assistant.
RF-004	Relatórios de progresso para gestores	Implementado	Squad Infra	Dashboard integrado ao Confluence e Jira.
RF-005	Reprodução de músicas e	Implementado	Squad Player	Funcionalidade principal estável

	controle do player			em todas as plataformas.
RF-006	Reprodução de podcasts com controles avançados	Implementado	Squad Podcasts	Inclui avanço rápido, retrocesso e salvar episódios.
RNF-001	Cobertura total de requisitos com testes	Em validação	Squad QA Central	Cobertura de 93% alcançada, testes adicionais em execução.
RNF-002	Reprodução em segundo plano	Implementado	Squad Mobile	Testado em Android e iOS, comportamento validado.

5.2 Matriz de Rastreabilidade de Requisitos

A matriz de rastreabilidade de requisitos permite o acompanhamento preciso de cada requisito desde sua origem até sua entrega final, assegurando que nenhum item essencial seja perdido ao longo das fases de desenvolvimento. No projeto de melhorias e expansão do Sonz. Ao longo do desenvolvimento, os requisitos foram organizados em categorias funcionais, como reprodução de música, criação de playlists, recomendação personalizada, suporte a podcasts, compatibilidade com dispositivos e melhoria de infraestrutura. Para cada um deles, foram registrados:

- A **origem do requisito** (ex.: feedback de usuários, demanda de mercado ou objetivos estratégicos);
- A **funcionalidade associada** no sistema;
- O **teste correspondente** que validaria sua correta implementação;
- O **status de desenvolvimento** (planejado, em progresso, validado ou implementado);
- O **responsável pela entrega**, geralmente um squad especializado.

Com base nessa estrutura, a matriz ajudou a garantir que cada requisito fosse rastreável em todas as fases. Essa rastreabilidade também permitiu responder rapidamente a mudanças nos requisitos e ajustar o escopo de desenvolvimento sem comprometer a qualidade final do produto.

I'D do Requisito	Descrição do Requisito	Origem	Funcionalidade Associada	Teste Correspondente	Status de Implementação	Responsável
RF-001	O usuário deve conseguir criar playlists personalizadas	Feedback de usuários	Gerenciamento de playlists	Teste de criação, edição e exclusão	Implementado	Squad Playlists
RF-002	Recomendação de músicas baseada no histórico de reprodução	Análise de dados	Algoritmo de recomendação	Teste de sugestão com base em comportamento	Em desenvolvimento	Squad Discover
RF-003	Compatibilidade com smart speakers (Alexa, Google Home)	Expansão de mercado	Integração com dispositivos externos	Teste de pareamento e controle por voz	Planejado	Squad Connect
RF-004	Relatórios de progresso acessíveis por gestores	Processo interno	Dashboard de monitoramento de squads	Teste de visualização de progresso	Implementado	Squad Infra
RF-005	O usuário deve conseguir reproduzir músicas e	Funcionalidade básica	Player de músicas	Teste de play, pause, próximo e	Implementado	Squad Player

	controlar o player			anterior		
RF-006	O usuário deve conseguir ouvir podcasts com controles específicos	Demanda de conteúdo	Player de podcasts	Teste de avanço rápido e controle de tempo	Implementado	Squad Podcasts
RNF-001	Garantir que 100% dos requisitos tenham cobertura de testes	Requisitos de qualidade	Cobertura de testes automatizados	Verificação de cobertura via CI/CD	Em validação	Squad QA Central
RNF-002	O usuário realiza a reprodução em segundo plano ao utilizar outros aplicativos.	Funcionalidade básica	Player de músicas	Teste de usabilidade no segundo plano e de integração	Implementado	Squad Mobile

5.3 Controle de Mudanças

Solicitação de Mudança

1. Qualquer mudança desejada é registrada no JIRA como uma issue com tipo "Change".
2. Deve conter: descrição, justificativa, impacto estimado (escopo, tempo, custo) e prioridade.

Análise Técnica e de Impacto

1. Realizada pelo Gerente do Projeto, Product Owner e Tech Lead.
2. Critérios analisados:
    a. Estimativa por complexidade técnica;
    b. Impacto no cronograma (Gantt);

c. Riscos associados (avaliados via matriz de risco);
d. Orçamento (com base em projeções no Excel/Power BI).

Decisão

1. Pequenas mudanças são aprovadas em Sprint Plannings.
2. Mudanças com impacto médio ou alto são encaminhadas ao Comitê de Mudanças (formado por GP, PO, QA e líder técnico).
3. Decisões são registradas e comunicadas via Confluence e Slack.

Implementação

1. Mudança aprovada é quebrada em tarefas técnicas e adicionada ao Product Backlog no JIRA.
2. Alterações no código são versionadas no GitHub, com Pull Requests e Code Review.

Monitoramento

1. As mudanças são monitoradas por meio de dashboards do JIRA e relatórios semanais enviados via Confluence para os stakeholders.

5.4 Análise de Desempenho

A análise de desempenho do projeto visa acompanhar a execução das atividades em tempo real, assegurando que os resultados estejam alinhados com os objetivos estratégicos e operacionais. Isso é feito com base em KPIs acompanhados por meio de ferramentas integradas.

5.4.1. Ferramentas de Monitoramento

1. JIRA Dashboards: acompanhamento de progresso por sprint.
2. GitHub e Selenium: automação de testes e integração contínua.
3. Firebase Performance Monitoring: análise em tempo real da performance do app.
4. SonarQube: verificação de qualidade e cobertura de código.
5. Power BI: geração de dashboards executivos com dados extraídos via API do JIRA.

5.4.2. Mecanismos de Controle

1. Sprint Reviews: análise de progresso e validação dos entregáveis.
2. Retrospectivas: identificação de falhas e proposição de melhorias.
3. Dashboards Operacionais: atualizados automaticamente no Power BI e acessados via Confluence.
4. Alertas Automatizados: via Firebase e GitHub para problemas de performance ou falhas de build.

5.5 Itens de Configuração

Tipo	Identificador	Descrição	Responsável
Documento de Requisitos	DOC-REQ-V1	Especificação funcional e não funcional	Product Owner
Código Fonte	SRC-FRONT-1	Front-end em React, HTML e CSS (web)	Squad Web
Código Fonte	SRC-BACK-1	Back-end Node.js com API REST	Squad Backend
Pipeline CI/CD	CI-GHA WORKFLOW	Workflow do GitHub Actions para build, testes e deploy	Eng. DevOps
Casos de Teste	QA-TEST UNIT	Testes unitários automatizados (Jest/Selenium)	QA
Relatório de	QA-REL	Documentação de	QA

Bugs	BUGS	defeitos por sprint	
Documento de Arquitetura	DOC-ARQ-V1	Desenho da arquitetura de microsserviços	Arquiteto de Software
Documento do GCS	DOC-GCS-V1	Este próprio documento de controle de configuração	Gerente de Projeto

5.5.1. Controle de Versão

Ferramentas utilizadas:

1. Git + GitHub: Controle de versão distribuído do código-fonte;
2. GitHub Releases: Marcação de versões do produto (v1.0, v1.1, etc.);
3. Confluence: Controle de versões de documentação técnica.

Tagging de versões:

1. Código-fonte: v1.0. v1.1.0, v2.0.0;
2. Documentos: Sufixo -v1,-v2, etc.

5.5.2. Controle de Acesso

Cargo	Permissões
Gerente de Projeto	Admin (merge, deploy e gerenciamento de acesso)
Desenvolvedores	Push em branches de feature

QA	Acesso leitura/escrita em testes e relatórios
DevOps	Acesso total ao pipeline e arquivos Docker

Todo o gerenciamento via GitHub Teams e integrações serão com autenticação via SSO.

5.5.3. Auditoria e Verificações

1. Auditoria Semanal de Código: via Pull Requests e Code Review;
2. Verificação de Integridade: por hashes SHA no Git e validação de builds;
3. Auditoria de Deploy: registros de execução no GitHub Actions (logs de pipelines);
4. Auditoria de Documentação: controle de versões no Confluence e aprovações.

5.5.4. Rastreabilidade

1. Requisito no JIRA;
2. Sprint e Epic associadas;
3. Commits no Git com mensagens padronizadas.

5.5.5. Mecanismos de Backup

1. Repositórios GitHub com backup automático diário (via GitHub Enterprise);
2. Documentos Confluence com versionamento nativo e exportações semanais.

5.5.6. Histórico de Alterações

O histórico de alterações registra todas as modificações feitas no software e na documentação, permitindo rastrear o que foi alterado, quando e por quem. Isso garante transparência, facilita auditorias, ajuda a recuperar versões anteriores em caso de problemas, melhora a comunicação da equipe e contribui para o controle de qualidade e gestão de riscos no projeto.

5.5.7. Repositório SCM

No Sonz, o repositório de Gestão de Configuração de Software (SCM) foi centralizado no GitHub, garantindo controle total sobre os artefatos de desenvolvimento: código-fonte (frontend e backend), documentos técnicos, plano de testes, scripts de deploy e históricos de alterações. O repositório atua como hub de integração com:

1. JIRA: para rastreamento de tarefas e mudanças;
2. Selenium e CI/CD: para testes automatizados e validação contínua;
3. Confluence: para documentação formal do projeto.

5.5.8. Controle de Alterações

Fluxo adotado:

1. As mudanças são registradas no JIRA como issues do tipo Change Request;
2. Avaliação realizada por Product Owner, Tech Lead e Squad responsável;
3. Alterações pequenas são aprovadas em Sprint Planning. Já mudanças médias/grandes exigem validação do Comitê de Mudanças;
4. Todas as alterações passam por code review, testes automatizados e tagging no GitHub antes da integração.

5.5.9. Auditoria de Configuração

Durante o desenvolvimento do Sonz, a auditoria de configuração foi realizada com foco em manter a conformidade técnica dos artefatos críticos, principalmente nas entregas do MVP.

A auditoria verificou:

1. Se os testes definidos foram cobertos em CI;
2. Se as mudanças nos SCIs foram documentadas no Confluence;
3. Se o autor da mudança foi registrado nos commits e issues correspondentes;
4. Se houve compatibilidade com os demais módulos e com o roadmap do projeto;
5. Se os critérios de qualidade foram respeitados.

5.5.10. Gestão de Impacto

Devido à arquitetura de microsserviços, o Sonz implementou a gestão de impacto como processo central para evitar falhas em cascata entre módulos como player, recomendações e autenticação.

Práticas adotadas:

1. Identificação antecipada dos módulos e squads afetados por mudanças (usando JIRA dependências);
2. Avaliação de impacto direto (forward) e reverso (backward), especialmente entre repositórios do backend e mobile;
3. Comunicação estruturada via Slack e reuniões diárias;
4. Realização de testes de integração entre serviços após qualquer modificação crítica.

6. Encerramento do Projeto

6.1 Resumo do Projeto

Este documento formaliza o encerramento do projeto de Desenvolvimento da Plataforma Sonz. O projeto teve como principal objetivo desenvolver uma versão mínima viável (MVP) de uma plataforma de streaming de música digital, acessível via web e dispositivos móveis. A justificativa para o projeto baseou-se na crescente demanda por consumo de música digital e na oportunidade de monetização através de assinaturas e publicidade. O escopo original do MVP concentrou-se em funcionalidades essenciais como cadastro/login, streaming básico, busca e filtros no catálogo de músicas, criação e compartilhamento de playlists, e um sistema inicial de recomendação personalizada, além de integração básica com redes sociais. Não houve mudanças significativas aprovadas no escopo do MVP durante a execução.

6.2 Aceitação Formal das Entregas

Confirma-se por meio deste termo que todas as entregas principais definidas no escopo do projeto foram concluídas e formalmente aceitas. As entregas incluem a plataforma

MVP funcional acessível via web e dispositivos móveis (Android e iOS), o catálogo de músicas com funcionalidades de busca e filtros, o player de música com controles básicos, a capacidade de criar e compartilhar playlists, o sistema de recomendação automática baseado no histórico do usuário e a integração inicial com redes sociais para login e compartilhamento. As entregas foram validadas em relação aos critérios de aceitação estabelecidos, que incluíam a execução estável da plataforma nos ambientes definidos, a funcionalidade correta da busca e filtros, a reprodução de música sem interrupções significativas, a criação e compartilhamento efetivo de playlists e a geração de sugestões de músicas pertinentes. A validação final ocorreu ao término da Sprint 7, conforme planejado, e os resultados dos testes de sistema, performance e usabilidade confirmaram o atendimento aos critérios.

6.3 Sumário de Desempenho do Projeto

O desempenho do projeto foi monitorado continuamente em relação às linhas de base de escopo, cronograma, custo e qualidade. Em termos de escopo, o MVP será entregue conforme planejado na Declaração de Escopo, incluindo todas as funcionalidades previstas e excluindo itens definidos como fora do escopo inicial (podcasts, videoclipes, modo offline). O cronograma foi gerenciado através da metodologia ágil Scrum, com sprints quinzenais. O projeto foi concluído dentro do período estimado de 7 sprints (aproximadamente 14 semanas), atingindo os marcos planejados (M1 a M5) nas Sprints correspondentes. A qualidade foi gerenciada através de um plano dedicado, com a execução de testes unitários, de integração, de sistema, de performance e de usabilidade ao longo das sprints, garantindo que as entregas atendessem aos padrões definidos e aos critérios de aceitação.

6.3 Sumário dos Riscos e Problemas

Durante o planejamento, foram identificados riscos potenciais, incluindo atrasos nas entregas, desafios no licenciamento de músicas, baixa adoção inicial por usuários, ocorrência de erros e falhas no sistema, e dificuldades na integração com redes sociais. Para cada risco, foram planejadas ações de mitigação, como o foco no escopo essencial do MVP para evitar atrasos, a antecipação de negociações de licenciamento, o investimento em divulgação inicial e a realização de testes contínuos para identificar e

corrigir falhas rapidamente. Ao longo da execução, os riscos foram monitorados e as ações planejadas foram implementadas conforme necessário. Problemas pontuais que surgiram durante o desenvolvimento e testes foram registrados, priorizados e resolvidos pela equipe dentro das sprints, garantindo o progresso contínuo do projeto.

6.3 Lições Aprendidas

O projeto Sonz proporcionou aprendizados valiosos. A adoção da metodologia Scrum com sprints quinzenais mostrou-se eficaz para gerenciar o desenvolvimento iterativo e incremental, permitindo flexibilidade e feedback constante. A definição clara do escopo do MVP foi crucial para manter o foco e garantirá a entrega dentro das restrições de tempo e orçamento. A complexidade das negociações de licenciamento de músicas (se aplicável) reforçou a necessidade de iniciar essas discussões o mais cedo possível em projetos futuros. Os testes de usabilidade realizados com usuários reais forneceram insights importantes que ajudaram a refinar a experiência do usuário antes do lançamento. A importância de testes contínuos e da rápida correção de bugs foi evidente para manter a qualidade e a estabilidade da plataforma. A comunicação será eficaz através das ferramentas definidas (Slack, JIRA) e das cerimônias do Scrum (reuniões diárias, reviews, retrospectivas) sendo um passo fundamental para o alinhamento da equipe e o sucesso do projeto.

6.3 Plano de Transição

A plataforma Sonz MVP, juntamente com sua documentação técnica associada (gerada durante o desenvolvimento) e manuais de usuário (se criados), será formalmente transferida para a equipe de operações e manutenção para sustentação e futuras evoluções.

6.3 Liberação de Recursos

Com a conclusão formal do projeto e a aceitação das entregas finais, todos os recursos humanos e materiais alocados especificamente para o projeto de Desenvolvimento da Plataforma Sonz estão oficialmente liberados. Isso inclui a equipe de desenvolvimento (backend, frontend web, mobile), a equipe de QA (líder, analistas, engenheiro de

automação, analista de UX) e quaisquer outros recursos temporários ou equipamentos utilizados exclusivamente pelo projeto.

6.3 Encerramento Administrativo e Contratual

Todas as atividades administrativas relacionadas ao projeto foram concluídas. A documentação final do projeto, incluindo o Termo de Abertura, Planos de Gerenciamento (Projeto, Requisitos, Comunicação, Riscos, Qualidade, Recursos Humanos), Declaração de Escopo, WBS, Matriz de Rastreabilidade de Requisitos, Relatórios de Progresso, Registros de Mudanças, Casos de Teste, Relatórios de Teste e este Termo de Encerramento, foi devidamente arquivada conforme os procedimentos da organização. Quaisquer contratos pendentes com fornecedores ou terceiros relacionados a este projeto foram finalizados e encerrados.

7. Referências

ATLASSIAN. Atlassian | Software Development and Collaboration Tools. Disponível em: <https://www.atlassian.com>.

ATLASSIAN. Gerenciamento de configuração: definição e benefícios | Atlassian. Disponível em: <https://www.atlassian.com/br/microservices/microservices-architecture/configuration-management?utm_source=chatgpt.com>. Acesso em: 27 mai. 2025.

GitHub.com Help Documentation. Disponível em: <https://docs.github.com>.

Firebase Performance Monitoring. Disponível em: <https://firebase.google.com/docs/perf-mon>.

JENKINS. Jenkins User Documentation. Disponível em: <https://www.jenkins.io/doc/>.

SeleniumHQ Browser Automation. Disponível em: <https://www.selenium.dev>.

SonarQube Documentation Homepage. Disponível em: <https://docs.sonarsource.com>. Acesso em: 27 mai. 2025.

JULCSC. Power BI documentation - Power BI. Disponível em: <https://learn.microsoft.com/power-bi/>. Acesso em: 27 mai. 2025.

Home | Scrum Guides. Disponível em: <https://scrumguides.org>.

CHINA, C. R.; GOODWIN, M. Gerenciamento de configuração. Disponível em: <https://www.ibm.com/br-pt/think/topics/configuration-management?utm_source=chatgpt.com>. Acesso em: 27 mai. 2025.

Guia para Monitoramento e Controle do Projeto | Smartsheet. Disponível em: <https://pt.smartsheet.com/content/project-monitoring-control>.

Relatório de Progresso: Como Escrever, Estruturar e Tornar Visual. Disponível em: <https://piktochart.com/pt-br/blog/relatorio-de-progresso/>. Acesso em: 10 jun. 2025.

COMPANY, T. Testing Company - 9 dicas para criar casos de teste de forma correta e eficaz. Disponível em:

<https://testingcompany.com.br/blog/9-dicas-para-criar-casos-de-teste-de-forma-correta-e-eficaz>. Acesso em: 27 mai. 2025.

KYMBERLI DE SOUZA. PMBOK: o que é e como aplicá-lo com 6 dicas | Zeev. Disponível em: <https://zeev.it/blog/pmbok-o-que-e-e-como-aplicar/>.

Gerenciamento de escopo PMBOK: conheça 6 etapas essenciais. Disponível em: <https://www.projectbuilder.com.br/blog/gerenciamento-de-escopo-pmbok/>.

As etapas do Gerenciamento de Projetos: O monitoramento e controle - PMKB. Disponível em: <https://pmkb.com.br/as-etapas-do-gerenciamento-de-projetos-o-monitoramento-e-controle/>. Acesso em: 27 mai. 2025.