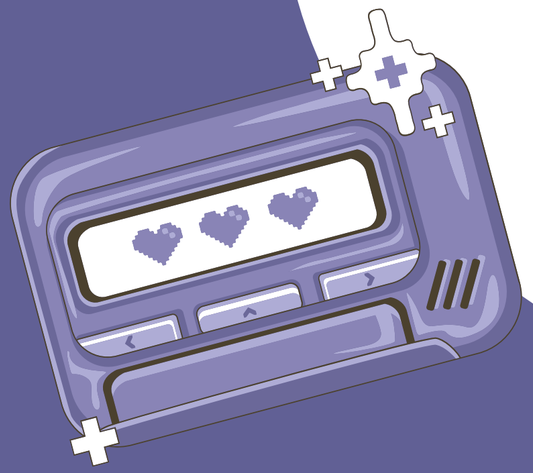
Melodify

Carol Baião, Daniel Vitor, Gustavo Perdigão
Vinicius Augusto e Yago Arruda

Introdução e Objetivos



INTRODUÇÃO

- Importância da música no cotidiano

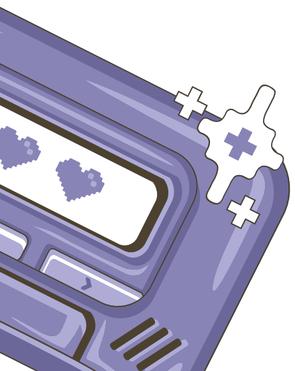
A música desperta emoções, cria conexões e está presente em diversos momentos da vida das pessoas.

- Limitação dos sistemas atuais

Faltam recursos para interações sociais focadas em música, e os usuários ficam limitados a redes genéricas ou fóruns dispersos.

- Crescimento das plataformas de streaming

Oferecem um vasto catálogo, mas as recomendações são baseadas apenas em algoritmos, sem considerar o lado humano do gosto musical.



- Proposta do projeto

Criar uma rede social musical, onde usuários possam avaliar, comentar e recomendar músicas, promovendo descoberta colaborativa e valorizando novos artistas.

OBJETIVOS

Objetivo Geral: Desenvolver uma rede social voltada para recomendação musical, permitindo que os usuários avaliem, comentem, compartilhem músicas e interajam com outros ouvintes, promovendo uma descoberta musical mais personalizada e colaborativa.

- Facilitar a descoberta de novas músicas

Oferecer recomendações personalizadas com base em avaliações e interações reais entre usuários.

- Valorizar a opinião da comunidade

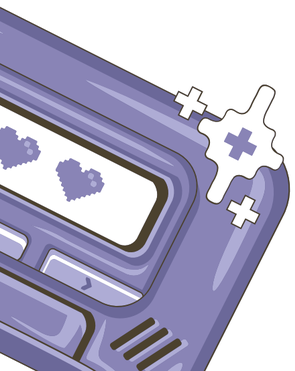
Estimular a criação de análises, comentários e críticas musicais, dando voz ativa aos ouvintes.

- Estimular a interação entre ouvintes

Criar um espaço para troca de opiniões, comentários e recomendações musicais de forma colaborativa.

- Unir tecnologia e experiência social

Integrar funcionalidades digitais com a experiência de compartilhar gostos musicais.



Projeto do Sistema

PERSONAS

- Felipe Mendes – O Explorador Musical

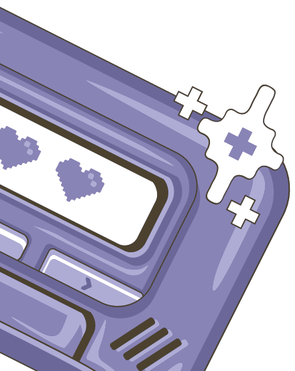
Jovem curioso por novos estilos; busca recomendações personalizadas e quer compartilhar suas descobertas.

- Mariana Rocha – A Ouvinte Casual

Profissional ocupada; procura recomendações rápidas e confiáveis para montar playlists sem perder tempo.

- Larissa Mendes – A Crítica Musical

Jornalista especializada; deseja publicar análises detalhadas, engajar com a comunidade e manter um histórico organizado.



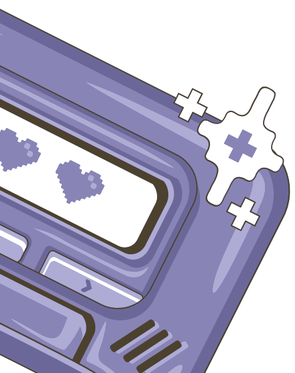
- Gustavo Ferreira – O Desenvolvedor Entusiasta de IA

Estudante de computação; quer testar e melhorar algoritmos de recomendação, focando no aspecto técnico da plataforma.

REQUISITOS

- FUNCIONAIS:

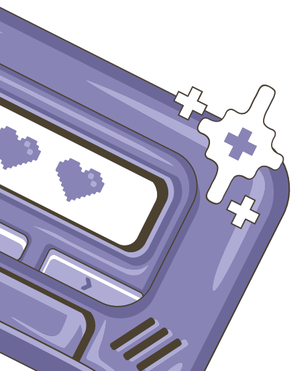
ID	Descrição do Requisito	Prioridade
RF-001	A aplicação deve permitir cadastro de usuario	ALTA
RF-002	A aplicação deve permitir deletar o usuario	MÉDIA
RF-003	A aplicação deve permitir alterar os dados do usuario	MÉDIA
RF-004	A aplicação deve autenticar o login	ALTA
RF-005	A aplicação deve gerenciar a conexão a API do spotify	ALTA
RF-006	A aplicação deve permitir buscar por uma musica	MÉDIA
RF-007	A aplicação deve permitir o usuario a analisar uma musica	ALTA
RF-008	A aplicação deve permitir o usuario a deletar uma analise	BAIXA
RF-009	A aplicação deve permitir o usuario a comentar em uma analise	MÉDIA
RF-010	A aplicação deve permitir o usuario a deletar um comentario	BAIXA
RF-011	A aplicação deve recomendar musicas por artista	BAIXA
RF-012	A aplicação deve recomendar musicas por ano de lançamento	MÉDIA
RF-013	A aplicação deve recomendar musicas por gênero	BAIXA



REQUISITOS

- NÃO FUNCIONAIS:

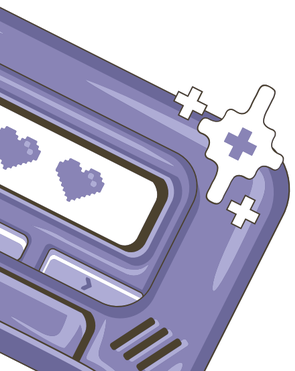
ID	Descrição do Requisito	Prioridade
RNF-001	A aplicação deve funcionar adequadamente em navegadores de dispositivos moveis e computadores	BAIXA
RNF-002	A aplicação deve processar requisições do usuário em no máximo 3s	BAIXA
RNF-003	A aplicação deve ser intuitiva para um uso simples	BAIXA



REQUISITOS

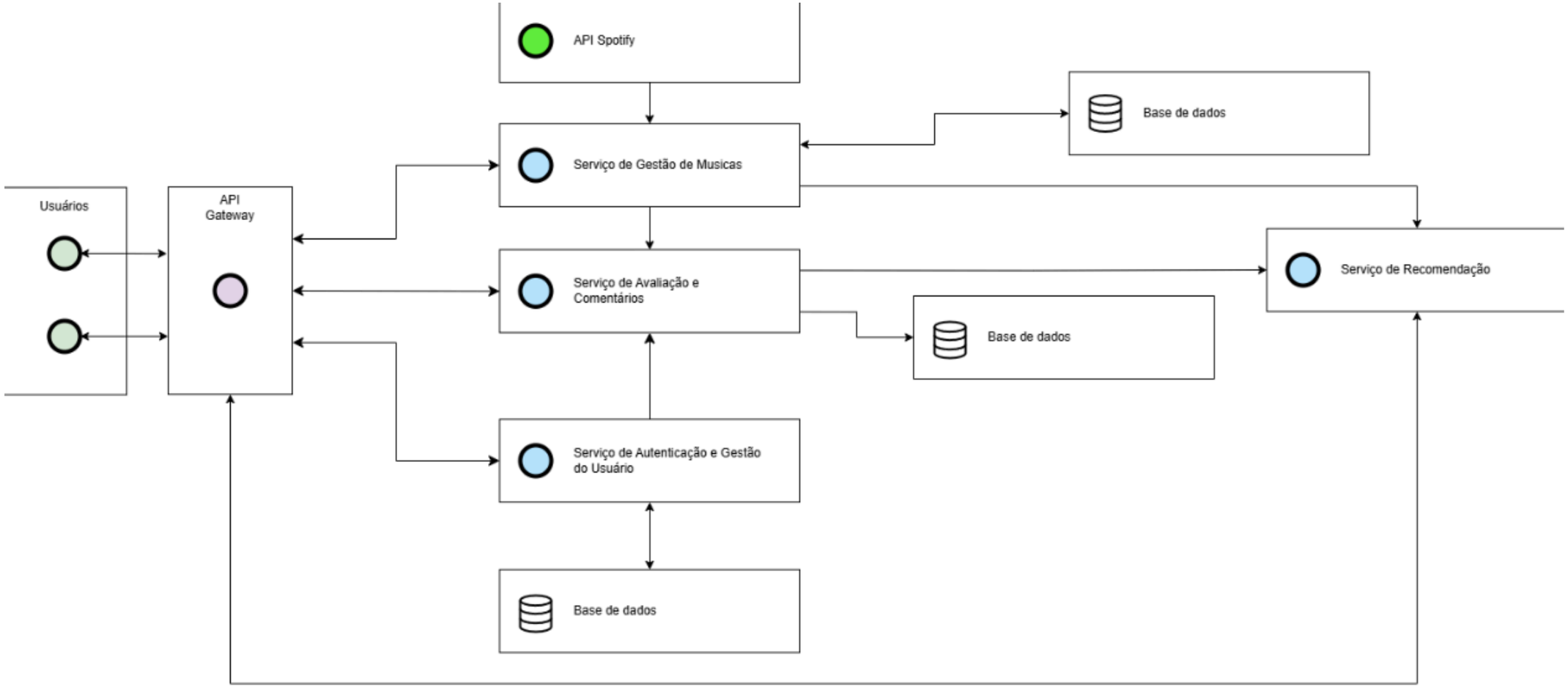
- RESTRIÇÕES:

ID	Restrição
01	O projeto deverá ser entregue até o final do semestre
02	Cada membro deve contribuir com pelo menos um serviço da API
03	Usar API do spotify como suporte a obtenção dos dados das musicas e não como uma ferramenta principal



ARQUITETURA

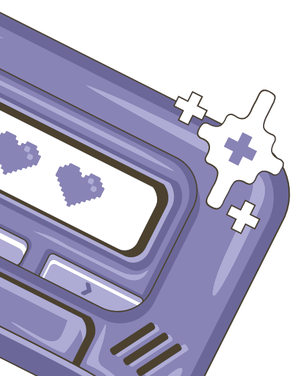
- ARQUITETURA DA SOLUÇÃO: MICROSSERVIÇOS



TECNOLOGIAS

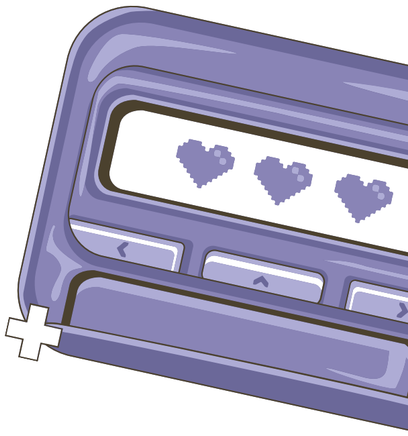
Tecnologias utilizadas no desenvolvimento do projeto:

- C#
  - para a lógica do back-end;
- API Spotify
  - integração para busca e dados de músicas;
- MySQL
  - armazenamento e gestão dos dados das músicas, avaliações, comentários e usuários;
- GitHub
  - para versionamento do código;



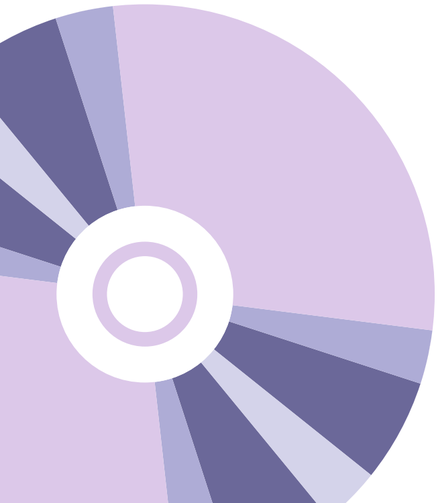
Desenvolvimento do Sistema

PLANEJAMENTO

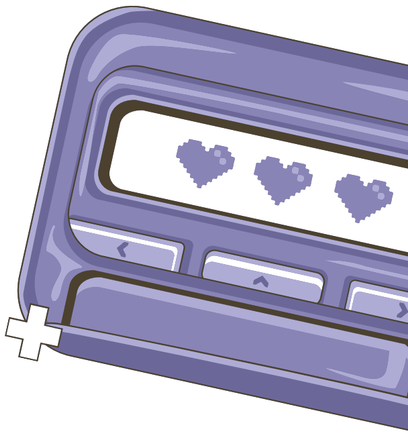


- SPRINT 1:

Responsável	Tarefa/Requisito	Iniciado em	Prazo
Gustavo	README introdutório	07/03/2025	07/03/2025
Daniel	Objetivos	07/03/2025	07/03/2025
Carolina	Problema	07/03/2025	07/03/2025
Carolina	Justificativa	07/03/2025	07/03/2025
Vinicius	Introdução	07/03/2025	07/03/2025
Yago	Público-Alvo	07/03/2025	07/03/2025
Todos do Grupo	Requisitos Funcionais	14/03/2025	14/03/2025
Todos do Grupo	Requisitos Não Funcionais	14/03/2025	14/03/2025
Yago	Restrições	14/03/2025	14/03/2025
Daniel e Vinicius	Personas	14/03/2025	14/03/2025
Carolina	Histórias de Usuário	14/03/2025	14/03/2025
Yago	Tecnologias Utilizadas	21/03/2025	23/03/2025
Gustavo	Divisão de Papéis	21/03/2025	23/03/2025
Vinicius	Quadro de Tarefas	21/03/2025	23/03/2025
Daniel	Processos	21/03/2025	23/03/2025
Yago	Ferramentas	21/03/2025	23/03/2025
Carolina	Gerar Tarefas no Github Projects	21/03/2025	23/03/2025

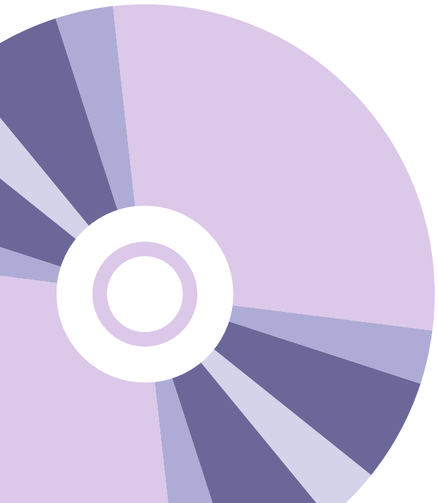


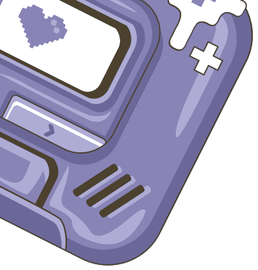
PLANEJAMENTO



- SPRINT 2:

Responsável	Tarefa/Requisito	Iniciado em	Prazo
Daniel	Avaliação	11/04/2025	01/06/2025
Daniel	Comentários	11/04/2025	01/06/2025
Yago	Autenticação	10/05/2025	01/06/2025
Yago	Gestão de Usuários	11/04/2025	01/06/2025
Gustavo e Vinicius	Gestão de músicas	30/04/2025	01/06/2025
Carol	Recomendação de músicas	22/05/2025	01/06/2025
Todos do grupo	Base de dados	11/04/2025	01/06/2025





METODOLOGIA ÁGIL

- Para auxiliar no processo de desenvolvimento, foi utilizado o Scrum como metodologia ágil;

- Estrutura das tarefas no Kanban:
  - Backlog: Lista geral de tarefas por prioridade;
  - To Do: Tarefas selecionadas para a sprint;
  - In Progress: Tarefas em andamento;
  - Review: Tarefas sendo revisadas;
  - Done: Tarefas finalizadas;

- Organização:
  - Sprint Planning: Planejamento de entregas por sprint
  - Week Planning: Acompanhamento semanal das tarefas e ajustes

- Papéis desempenhados pela equipe:
  - Product Owner;
  - Scrum Master;
  - Equipe de Desenvolvimento;

ENDPOINTS

- Autenticação de usuários:
  - POST /login
    - Autentica o usuário e retorna o token JWT.
  - GET /usuarios
    - Lista todos os usuários (requer token).
  - POST /usuarios
    - Cadastra novo usuário.
  - GET /usuarios/{id}
    - Retorna dados de um usuário específico.
  - PUT /usuarios/{id}
    - Atualiza dados do usuário.
  - DELETE /usuarios/{id}
    - Remove um usuário do sistema.

- Status do Usuário:
  - GET /status
    - Lista todos os status cadastrados.
  - POST /status
    - Cadastra novo status.
  - GET /status/{id}
    - Retorna status específico.
  - PUT /status/{id}
    - Atualiza o status existente.
  - DELETE /status/{id}
    - Remove o status.

ENDPOINTS

- Avaliações de Músicas
  - GET /avaliacao
    - Lista todas as avaliações feitas.
  - POST /avaliacao
    - Cria uma nova avaliação.
  - GET /avaliacao/{id}
    - Retorna uma avaliação específica.
  - PUT /avaliacao/{id}
    - Edita a avaliação.
  - DELETE /avaliacao/{id}
    - Exclui a avaliação.

- Comentários nas Avaliações
  - GET /comentario
    - Lista todos os comentários.
  - POST /comentario
    - Adiciona um novo comentário.
  - GET /comentario/{id}
    - Retorna comentário específico.
  - PUT /comentario/{id}
    - Edita o comentário.
  - DELETE /comentario/{id}
    - Remove o comentário.

ENDPOINTS

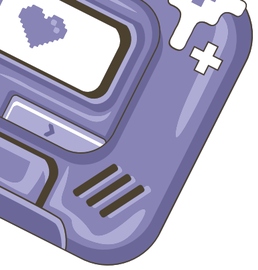
- Músicas:
  - GET /music
    - Lista todas as músicas registradas.
  - GET /music/{id}
    - Retorna música por ID.
  - PUT /music/{id}
    - Atualiza dados da música.
  - DELETE /music/{id}
    - Remove a música do sistema.
  - GET /music/search
    - Busca música por nome ou artista (via Spotify).
  - POST /music/add-by-spotify-id/{id}
    - Adiciona uma música usando o ID do Spotify.
  - POST /music/add-multiple-by-spotify-id
    - Adiciona múltiplas músicas por IDs do Spotify.

- Playlists:
  - GET /playlist
    - Lista todas as playlists existentes.
  - GET /playlist/{id}
    - Retorna uma playlist específica.
  - GET /playlist/user/{userId}
    - Retorna playlists de um usuário.
  - POST /playlist/user/{userId}
    - Cria uma nova playlist para o usuário.
  - POST /playlist/{playlistId}/add-music/{musicId}
    - Adiciona uma música à playlist.
  - DELETE /playlist/{playlistId}/remove-music/{musicId}
    - Remove uma música da playlist.

ENDPOINTS

- Integração com Spotify
  - GET /api/spotify/search
    - Realiza busca no Spotify.
  - GET /api/spotify/music/{spotifyId}
    - Retorna dados da música por ID do Spotify.

- Recomendação de Músicas
  - GET /recomendacao/byGenero
    - ▶ Recomenda músicas por gênero.
  - GET /recomendacao/byAno
    - ▶ Recomenda músicas por ano.
  - GET /recomendacao/byArtista
    - ▶ Recomenda músicas por artista



TESTES

- Ferramentas Utilizadas
  - Swagger: Testes manuais e documentação da API
  - Postman: Testes REST com diferentes cenários
  - Newman: Execução automatizada de testes e simulação de carga

- Testes realizados:
  - Testes Unitários: Funções de validação, entidades e regras de negócio
  - Testes de Integração: Comunicação entre controllers, serviços e banco
  - Testes de Carga: Simulação de múltiplos acessos simultâneos

- Exemplos de Casos de Teste
  - Criar usuário com dados válidos → 201 Created
  - Login com senha errada → 401 Unauthorized
  - Buscar música existente → 200 OK
  - Deletar comentário inexistente → 404 Not Found
  - Recomendação por gênero inválido → 404 Not Found

Fim