Gerenciador de Downloads do YouTube

Guilherme Silva Sampaio e Pedro Henrique Gonçalves Mota

O que é o Sistema?

O Gerenciador de Downloads do YouTube é uma aplicação web fullstack com o objetivo de permitir que usuários baixem vídeos e áudios diretamente do YouTube de forma simples, eficiente e moderna, demonstrando a integração entre frontend e backend com comunicação em tempo real. O sistema será construído com **React + Vite** no frontend e **FastAPI (Python)** no backend, utilizando **HTTP** e **WebSocket** para comunicação, além de dependências como **pytubefix**, **MoviePy** e **YouTube API** e conteinerização para suporte a IPV6.

A aplicação contará com uma interface responsiva e intuitiva, que permitirá buscar vídeos diretamente pela API do YouTube ou inserir URLs de forma manual. Os usuários poderão escolher entre baixar arquivos em formato MP4 (vídeo) ou MP3 (áudio), e ainda realizar downloads em lote, que serão compactados automaticamente em um arquivo ZIP. Todo o progresso será exibido em tempo real por meio de WebSockets, garantindo uma experiência fluida e informativa.

Modelo Cliente-Servidor

O projeto segue o modelo cliente-servidor, onde o frontend atua na porta 5173 e o backend na porta 8000. A interface web se comunica com o servidor para processar as requisições de download, conversão e compactação de arquivos. Além disso, o sistema foi projetado com foco em escalabilidade e desempenho, sendo totalmente containerizado com Docker e compatível com IPv6.

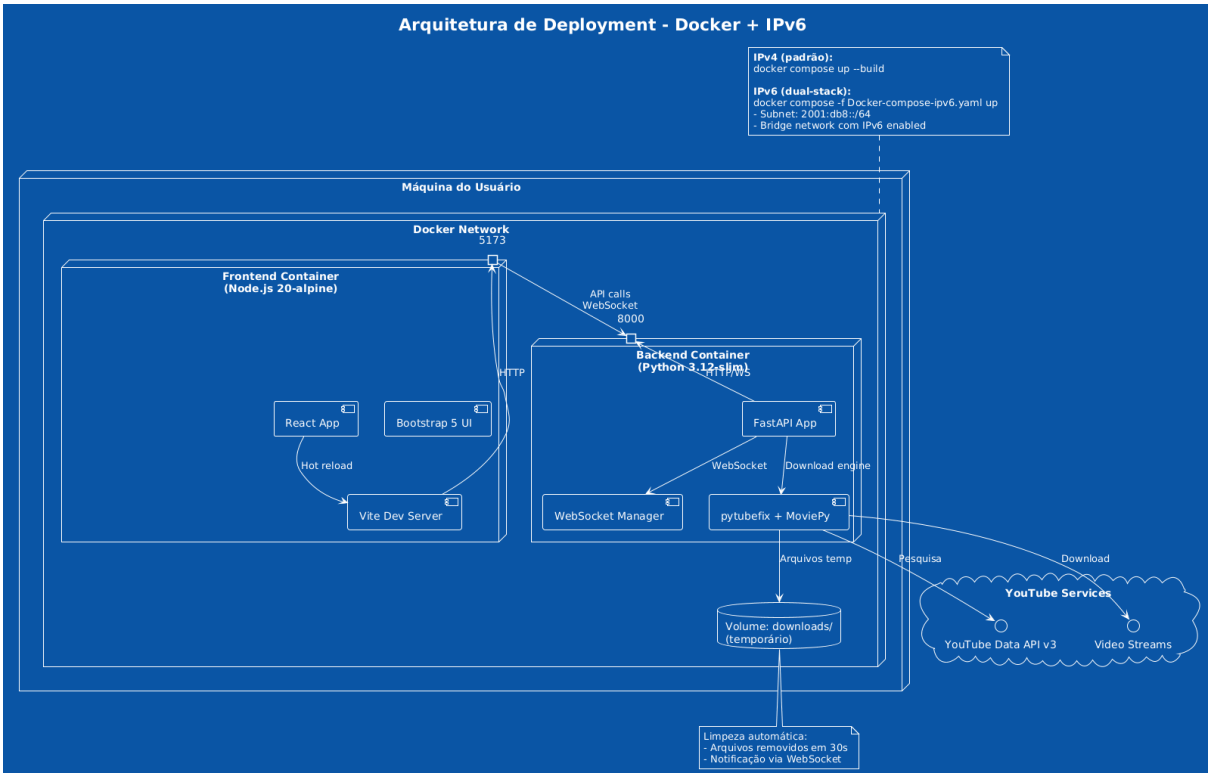

Figura 1: Diagrama da arquitetura cliente - servidor do sistema

Onde queremos chegar?

Os principais resultados que desejamos alcançar é a construção de um sistema fullstack completo e funcional, suporte a múltiplos formatos de mídia (MP3, MP4, ZIP), comunicação em tempo real e interface moderna com design adaptável a dispositivos móveis. Como possíveis melhorias futuras, estão previstas funcionalidades como autenticação de usuários, histórico de downloads e playlists personalizadas.

Diagrama de UseCase

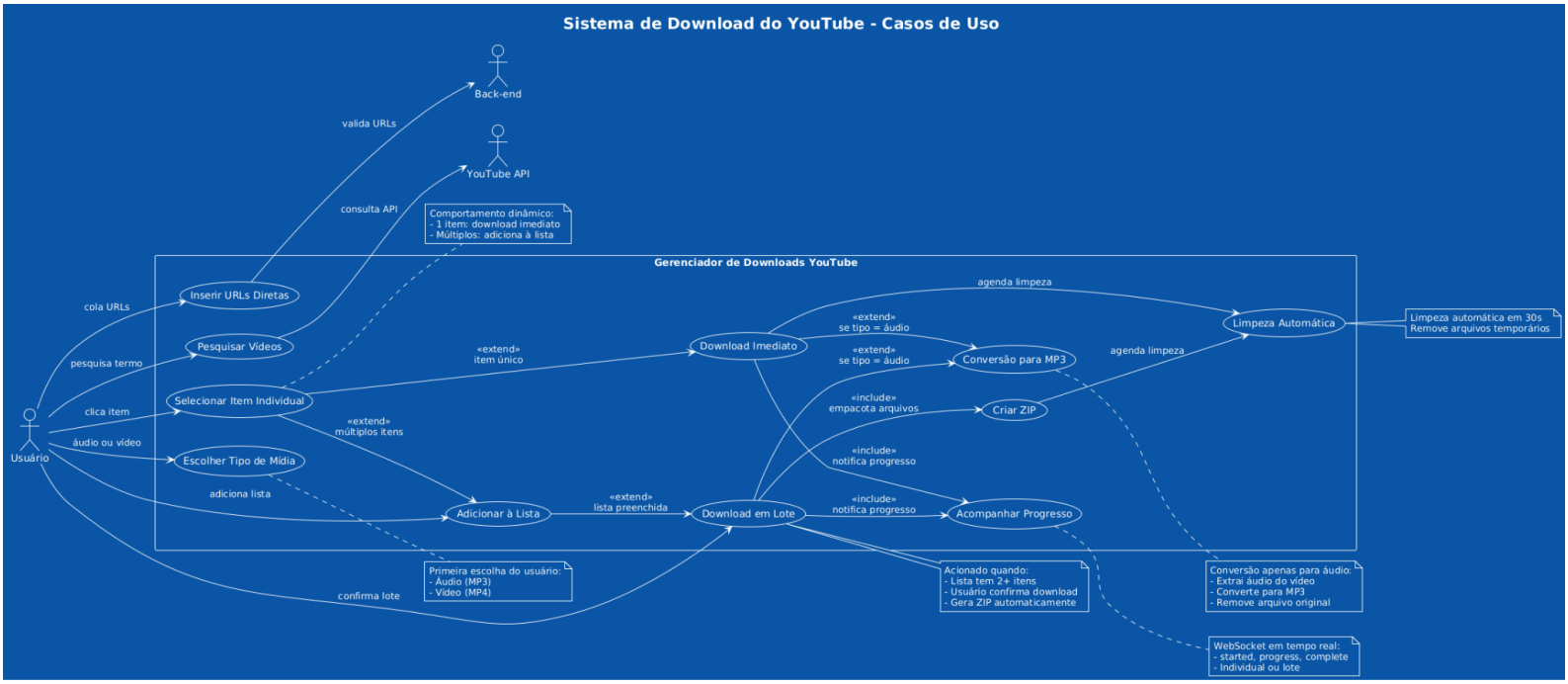

Figura 2: Diagrama de UseCase

Descrição do Diagrama de UseCase

Este diagrama representa um sistema de download do YouTube otimizado para diferentes cenários de uso.

Atores Principais:

- **Usuário:** Interage com a interface para pesquisar e baixar conteúdo.
- **YouTube API:** Fornece dados dos vídeos.
- **Back-end:** Processa validações e operações do sistema.

Fluxo Principal:

1. **Entrada de Dados:** O usuário pesquisa vídeos ou insere URLs diretas.
2. **Seleção de Tipo:** Escolhe o formato (áudio - MP3 ou vídeo - MP4) antes de selecionar o conteúdo.

3. **Comportamento Inteligente:**
    - **1 item:** Download automático e imediato.
    - **Múltiplos itens:** Adiciona à lista para download em lote.

Funcionalidades Especiais:

- **Download em Lote (ZIP):** Gera um arquivo ZIP automaticamente ao selecionar 2 ou mais itens.
- **Conversão Condicional:** Converte para MP3 apenas quando o tipo selecionado for áudio.
- **Progresso em Tempo Real:** Utiliza WebSocket para notificar o status do download instantaneamente.
- **Limpeza Automática:** Remove arquivos temporários após 30 segundos.

Arquitetura:

O sistema utiliza relacionamentos <<include>> para funcionalidades obrigatórias (progresso, ZIP) e <<extend>> para opcionais (conversão de áudio, diferentes tipos de seleção), garantindo flexibilidade e eficiência no processo.

Diagrama de Sequência

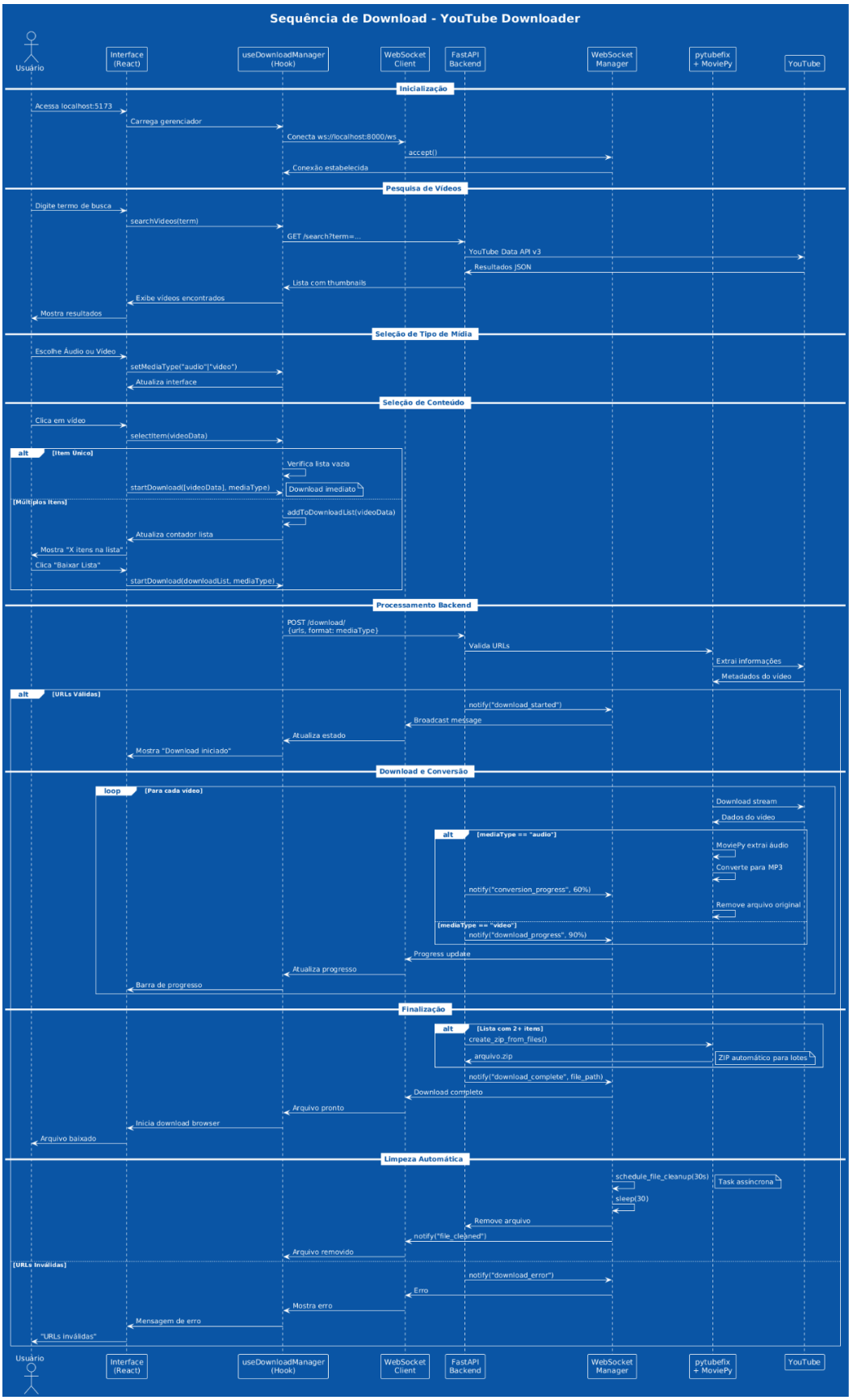

Figura 3: Diagrama de sequência

Requisitos Funcionais (RF)

- RF01 - Pesquisa de Vídeos: busca por termo no YouTube (YouTube API).
- RF02 - Inserção de URLs Diretas: aceita URLs válidas do YouTube.
- RF03 - Seleção Múltipla: permite selecionar vários vídeos.
- RF04 - Escolha de Formato: opção entre MP3 (áudio) e MP4 (vídeo).
- RF05 - Download Individual: baixa arquivos únicos via navegador.
- RF06 - Download em Lote: gera ZIP com múltiplos vídeos.
- RF07 - Progresso em Tempo Real: mostra progresso via WebSocket.
- RF08 - Validação de URLs: checa URLs antes do download.
- RF09 - Limpeza Automática: remove arquivos temporários após 30s.
- RF10 - Monitoramento de Saúde: endpoint /health para status da aplicação.

Requisitos Não Funcionais (RNF)

- RNF01 - Performance: até 5 downloads simultâneos (AsyncIO).
- RNF02 - Usabilidade: interface intuitiva e responsiva (Bootstrap 5).
- RNF03 - Compatibilidade de Rede: suporte IPv4 e IPv6.
- RNF04 - Disponibilidade: WebSocket estável com ping/pong.
- RNF05 - Escalabilidade: deployment via Docker.
- RNF06 - Segurança: validação e sanitização de entradas.
- RNF07 - Manutenibilidade: código modular e documentado.
- RNF08 - Portabilidade: execução consistente com .env e containers.
- RNF09 - Confiabilidade: tratamento de erros e logs estruturados.
- RNF10 - Eficiência de Recursos: limpeza automática de temporários.
- RNF11 - Tempo de Resposta: interface fluida durante downloads.
- RNF12 - Interoperabilidade: compatível com browsers modernos (ES6+).

Fluxo de Atividades

A ordem de desenvolvimento do projeto é a seguinte:

1. Desenvolvimento das rotas no backend utilizando o Swagger do FastAPI para testes, validação e verificação do sistema.
2. Criação e integração da interface desenvolvida com React.
3. Estabelecimento da conexão WebSocket.
4. Criação dos contêineres com suporte a IPv6.
5. Testar as funcionalidades do Sistema

Distribuição de Tarefas

Guilherme Sampaio

- Desenvolvimento do backend
- Organização da arquitetura do sistema
- DevOps - conteinerização
- Uso do websocket no backend

Pedro Mota

- Desenvolvimento do frontend
- Integração com o backend
- Testes
- Uso do websocket no frontend