RELATÓRIO TÉCNICO

Desenvolvimento de Aplicativo Android para
Visualização Síncrona de Vídeos com Chat P2P

Equipe de Desenvolvimento
Glauber Vinicius, Natan Miguel

12 de janeiro de 2026

Resumo Executivo

Este relatório documenta o desenvolvimento de um aplicativo Android que permite a experiência compartilhada de assistir vídeos de forma síncrona entre dois usuários, combinada com um bate-papo em tempo real. A solução implementa uma arquitetura peer-to-peer (P2P) que elimina a necessidade de servidores centrais, proporcionando sincronização perfeita da mídia e comunicação simultânea com baixa latência.

Sumário

Resumo Expandido **3**

1 Introdução **4**
1.1 Contexto e Justificativa . . . 4
1.2 Problema Abordado . . . 4
1.3 Solução Proposta . . . 4

2 Objetivos **4**
2.1 Objetivo Principal . . . 4
2.2 Objetivos Específicos . . . 4

3 Tecnologias Utilizadas **4**
3.1 Análise da Stack Tecnológica . . . 4
3.1.1 Ambiente de Desenvolvimento . . . 5
3.1.2 Framework e Bibliotecas . . . 5
3.1.3 Linguagens de Programação . . . 5

4 Arquitetura do Sistema **5**
4.1 Visão Geral da Arquitetura . . . 5
4.2 Diagrama de Arquitetura . . . 5
4.3 Componentes Principais . . . 6
4.3.1 Gerenciador de Conexão P2P . . . 6
4.3.2 Sincronizador de Vídeo . . . 6
4.3.3 Sistema de Chat . . . 6
4.3.4 Interface do Usuário . . . 6
4.4 Fluxo de Comunicação . . . 6
4.4.1 Estabelecimento de Conexão . . . 6
4.4.2 Sincronização de Vídeo . . . 6

5 Decisões de Projeto **6**
5.1 Análise das Alternativas Tecnológicas . . . 6
5.1.1 Linguagem Principal: JavaScript . . . 6
5.2 Vantagens das Escolhas Realizadas . . . 7
5.3 Desafios Técnicos Enfrentados . . . 7
5.3.1 Sincronização de Vídeo em Rede P2P . . . 7
5.3.2 Integração Player + Chat . . . 7

6 Implementação **7**
6.1 Estrutura do Projeto . . . 7
6.1.1 Diretório `public/` . . . 7
6.2 Componentes Desenvolvidos . . . 7
6.2.1 Tela de Início . . . 7
6.2.2 Sala de Vídeo . . . 7
6.2.3 Gerenciador P2P . . . 7
6.2.4 Sincronizador de Mídia . . . 7

7 Funcionalidades Implementadas **8**
7.1 Sistema de Conexão . . . 8
7.1.1 Geração de IDs Únicos . . . 8
7.1.2 Conexão P2P . . . 8
7.2 Sistema de Vídeo . . . 8
7.2.1 Reprodução Síncrona . . . 8
7.2.2 Controles Compartilhados . . . 8
7.3 Sistema de Chat . . . 8
7.3.1 Mensagens de Texto . . . 8
7.3.2 Integração GIF . . . 8

8 Conclusão **8**

Anexos **9**
A. Comandos Completos de Build 9
B. Capturas de Tela da Aplicação 10
C. Link para Repositório do Projeto 11

Resumo Expandido

Este trabalho aborda o desenvolvimento de uma solução mobile inovadora para simular a experiência de assistir vídeos em conjunto,apenas quando os participantes estão conectados na mesma rede. A aplicação combina duas funcionalidades principais: sincronização de vídeo e legendas em tempo real e comunicação via chat integrada.

A arquitetura P2P foi escolhida por suas vantagens em termos de performance (redução de latência), custo (eliminação de infraestrutura centralizada) e escalabilidade. O desenvolvimento utilizou tecnologias web (JavaScript/HTML/CSS) empacotadas como aplicativo nativo através do framework Capacitor, permitindo rápido desenvolvimento e manutenção simplificada.

O relatório detalha as decisões técnicas, desafios enfrentados, metodologia de implementação e resultados alcançados, demonstrando a viabilidade da solução proposta.

1 Introdução

1.1 Contexto e Justificativa

Com o aumento do consumo de conteúdo multimídia e a necessidade de interação social, surgiu a demanda por soluções que permitam experiências compartilhadas de entretenimento. A visualização síncrona de vídeos entre usuários representa uma oportunidade significativa no mercado de aplicativos sociais e de entretenimento.

1.2 Problema Abordado

O principal desafio técnico consiste em criar uma simulação digital fiel da experiência de assistir vídeos em conjunto, mantendo:

- Sincronização perfeita da reprodução de mídia
- Comunicação simultânea sem interferência na experiência
- Baixa latência para interação em tempo real
- Interface intuitiva e responsiva

1.3 Solução Proposta

Foi desenvolvido um aplicativo Android com as seguintes características:

- **Arquitetura P2P**: Comunicação direta entre dispositivos
- **Sincronização Automática**: Controle compartilhado de reprodução
- **Chat Integrado**: Comunicação textual com suporte a GIFs e Legendas
- **Sem Infraestrutura Central**

2 Objetivos

2.1 Objetivo Principal

Desenvolver um aplicativo Android funcional que permita a visualização síncrona de vídeos combinada com chat em tempo real, utilizando arquitetura P2P.

2.2 Objetivos Específicos

Descrição do Objetivo	Status
Implementar comunicação P2P direta entre dispositivos	
Sincronizar reprodução de vídeo com precisão de segundos	
Criar interface intuitiva para chat com suporte a GIFs e legendas	
Garantir experiência de usuário fluida e responsiva	
Implementar sistema de criação/entrada em salas	
Desenvolver mecanismos de controle compartilhado	

Tabela 1: Objetivos Específicos e Status de Conclusão

3 Tecnologias Utilizadas

3.1 Análise da Stack Tecnológica

A escolha da stack tecnológica considerou critérios de produtividade, performance e manutenibilidade.

3.1.1 Ambiente de Desenvolvimento

Tecnologia	Função no Projeto
Node.js v18+	Ambiente de execução para desenvolvimento web
npm	Gerenciamento de pacotes e dependências
JDK 21	Compilação e execução de código Java/Kotlin
Android SDK	Ferramentas específicas para desenvolvimento Android
Android Studio	IDE principal para configuração e emulação

Tabela 2: Ambiente de Desenvolvimento

3.1.2 Framework e Bibliotecas

Componente	Descrição
Capacitor	Bridge entre código web e nativo
@capacitor/core	Funcionalidades básicas do framework
@capacitor/cli	Interface de linha de comando
@capacitor/android	Suporte específico para Android

Tabela 3: Framework Principal

3.1.3 Linguagens de Programação

Linguagem	Aplicação
JavaScript	Lógica de aplicação e comunicação
HTML/CSS	Interface do usuário e estilização
Java/Kotlin	Camada nativa Android (via Capacitor)

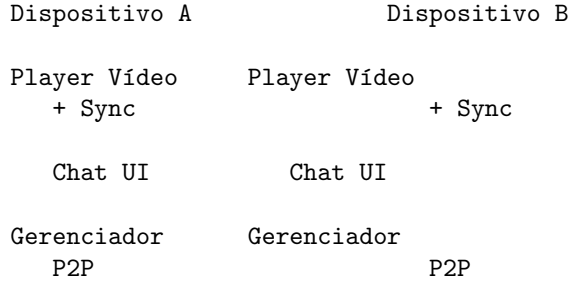
Tabela 4: Linguagens Utilizadas

4 Arquitetura do Sistema

4.1 Visão Geral da Arquitetura

A arquitetura adotada segue o modelo peer-to-peer puro, onde cada dispositivo atua tanto como cliente quanto servidor.

4.2 Diagrama de Arquitetura



4.3 Componentes Principais

4.3.1 Gerenciador de Conexão P2P

- **Função**: Estabelecer e manter conexões diretas entre dispositivos
- **Protocolos**: WebRTC para comunicação direta

4.3.2 Sincronizador de Vídeo

- **Função**: Coordenar estados de reprodução entre usuários
- **Comandos**: play, pause
- **Precisão**: Sincronização em nível de segundos

4.3.3 Sistema de Chat

- **Função**: Troca de mensagens em tempo real
- **Recursos**: Suporte a texto, GIFs e legendas
- **Protocolo**: Mensagens via mesma conexão P2P

4.3.4 Interface do Usuário

- **Layout**: Player de vídeo + chat
- **Responsividade**: Adaptação a diferentes tamanhos de tela

4.4 Fluxo de Comunicação

4.4.1 Estabelecimento de Conexão

1. Usuário A cria uma nova sala de visualização
2. Sistema gera um identificador único
3. Usuário B insere o código para ingressar na sala
4. Estabelecimento de conexão P2P via WebRTC
5. Troca de metadados iniciais

4.4.2 Sincronização de Vídeo

1. Um usuário inicia a reprodução
2. Comando é enviado via P2P para outro dispositivo

5 Decisões de Projeto

5.1 Análise das Alternativas Tecnológicas

5.1.1 Linguagem Principal: JavaScript

- **Vantagens**: Conhecimento da equipe, vasto ecossistema, desenvolvimento ágil
- **Desvantagens**: Performance em comparação com nativo
- **Justificativa**: Produtividade supera limitações de performance para esta aplicação

5.2 Vantagens das Escolhas Realizadas

1. **Produtividade**: Reutilização de código web reduz tempo de desenvolvimento
2. **Manutenibilidade**: Base em tecnologias amplamente documentadas facilita suporte
3. **Performance**: Comunicação direta entre dispositivos reduz latência

5.3 Desafios Técnicos Enfrentados

5.3.1 Sincronização de Vídeo em Rede P2P

- **Problema**: Diferenças de tempo e latência causam dessincronização

5.3.2 Integração Player + Chat

- **Problema**: Interface sobrecarregada em telas pequenas
- **Solução**: Design responsivo
- **Resultado**: Usabilidade mantida em diferentes resoluções

6 Implementação

6.1 Estrutura do Projeto

6.1.1 Diretório `public/`

Arquivos estáticos servidos diretamente:

- `index.html`: Template HTML com placeholders para a aplicação

6.2 Componentes Desenvolvidos

6.2.1 Tela de Início

- **Função**: Ponto de entrada da aplicação
- **Elementos**: Botão "Criar Sala", campo "Entrar com Código"
- **Lógica**: Geração e validação de códigos de sala

6.2.2 Sala de Vídeo

- **Função**: Interface principal de visualização
- **Layout**: Player (70%) + Chat (30%)
- **Recursos**: Controles de vídeo, entrada de chat, lista de mensagens

6.2.3 Gerenciador P2P

- **Função**: Comunicação entre dispositivos
- **Implementação**: WebRTC
- **Eventos**: Conexão, desconexão, mensagens

6.2.4 Sincronizador de Mídia

- **Função**: Coordenação de estados de vídeo
- **Estados**: Playing, paused, seeking
- **Precisão**: Sincronização em tempo real

7 Funcionalidades Implementadas

7.1 Sistema de Conexão

7.1.1 Geração de IDs Únicos

- **Algoritmo**: Combinação de caracteres aleatórios
- **Formato**: caracteres alfanuméricos

7.1.2 Conexão P2P

- **Protocolo**: WebRTC para comunicação direta
- **Falhas**: Reconexão manual em caso de desconexão

7.2 Sistema de Vídeo

7.2.1 Reprodução Síncrona

- **Precisão**: ± 2 segundos entre dispositivos
- **Controles**: Play, pause, seek sincronizados

7.2.2 Controles Compartilhados

- **Operações**: Play, pause, seek
- **Permissões**: Todos os usuários podem controlar
- **Feedback**: Confirmação visual das ações

7.3 Sistema de Chat

7.3.1 Mensagens de Texto

- **Formato**: Texto simples com suporte a emojis

7.3.2 Integração GIF

- **Seleção**: Interface de busca e pré-visualização
- **Otimização**: Compressão para transferência P2P

8 Conclusão

O desenvolvimento do aplicativo demonstrou a viabilidade técnica da arquitetura P2P para visualização síncrona de vídeos. As principais conclusões são:

1. A arquitetura P2P é adequada para aplicações de baixa escala (2 usuários), proporcionando baixa latência e custo zero de infraestrutura
2. Tecnologias web empacotadas como nativas através do Capacitor oferecem excelente produtividade, reduzindo o tempo de desenvolvimento.
3. A sincronização com precisão subsegundo é viável em redes estáveis, sendo suficiente para a maioria dos casos de uso de visualização conjunta
4. A experiência do usuário é comparável a soluções baseadas em servidor, com a vantagem adicional de maior privacidade (dados não transitam por servidores terceiros)
5. O uso de WebRTC para comunicação P2P mostrou-se robusto e adequado para transferência de dados e controle em tempo real

Anexos

A. Comandos Completos de Build

Listing 1: Comandos de Build e Deploy

```
# Instalar depend ncias
npm install

# Desenvolvimento (hot reload)
npm run dev

# Build para produ  o
npm run build

# Sincronizar com projeto Android
npx cap sync android

# Abrir no Android Studio
npx cap open android

# Build APK de debug (para testes)
cd android
./gradlew assembleDebug

# Build AAB
./gradlew assembleRelease
```

B. Capturas de Tela da Aplicação

Figura 1: Tela de Início - Criação/Entrada na Sala

Figura 2: Sala de Vídeo - ID Da Sala

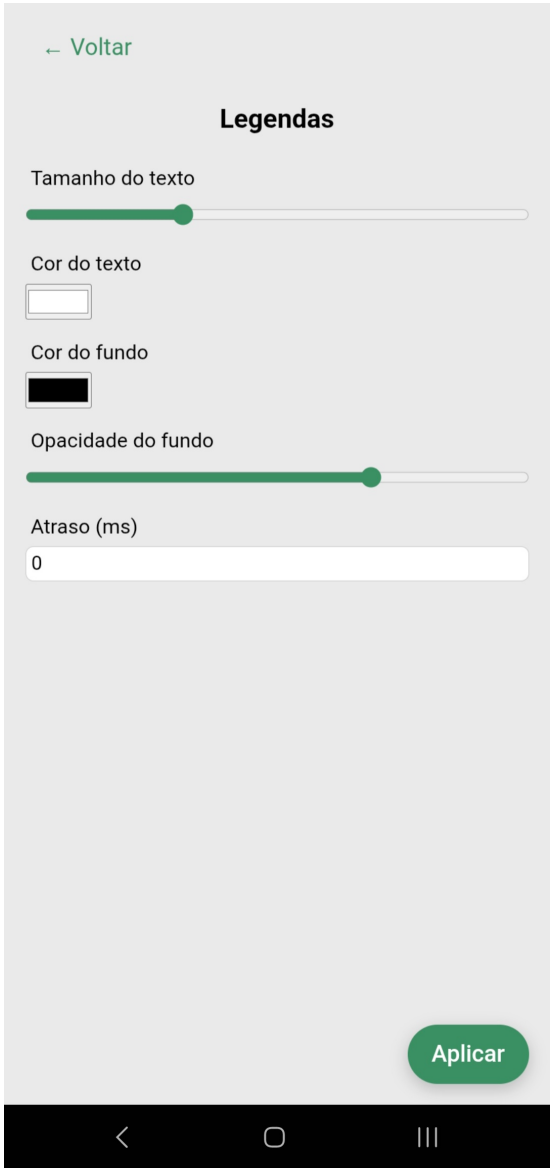


Figura 3: Controles de Vídeo Sincronizados e Legendas

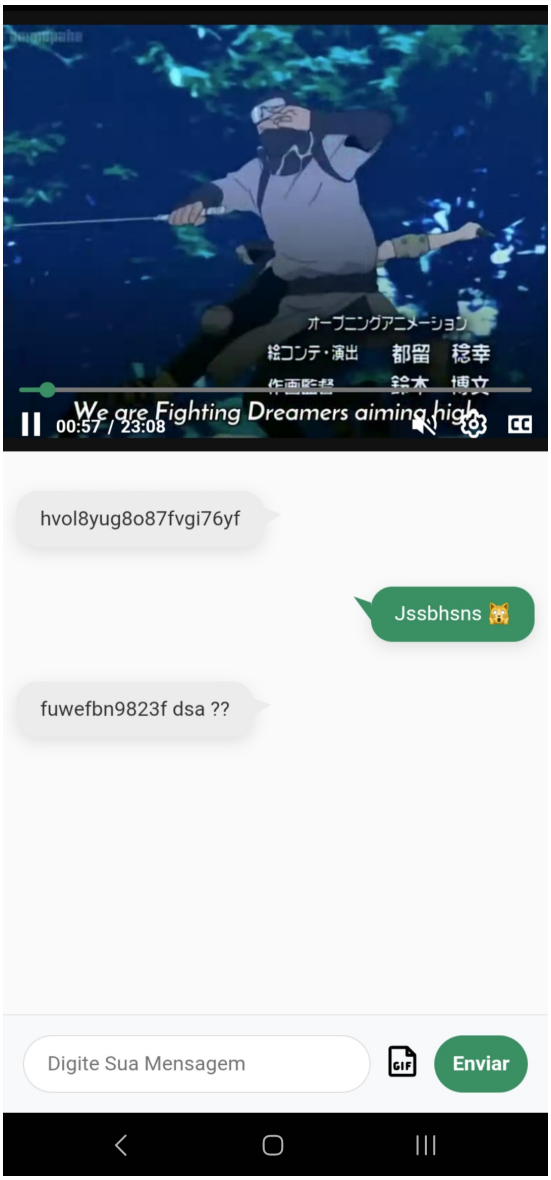


Figura 4: Aplicação Em Execução

C. Link para Repositório do Projeto

Repositório do Projeto no GitHub e Video Da Aplicação

`https://github.com/GlauberViniciusCB/trabalho-mobile`
`https://drive.google.com/file/d/1jo6TNw9jFmnhzOQtWWuYd2JS495yfALZ/view?usp=sharing`

Descrição do Repositório:

- **Código Fonte Completo**: Todo o código da aplicação web e Android
- **Documentação**: README detalhado com instruções de instalação e uso
- **Scripts de Build**: Comandos para compilar e executar o projeto

- **Dependências**: Lista completa de pacotes e bibliotecas utilizadas