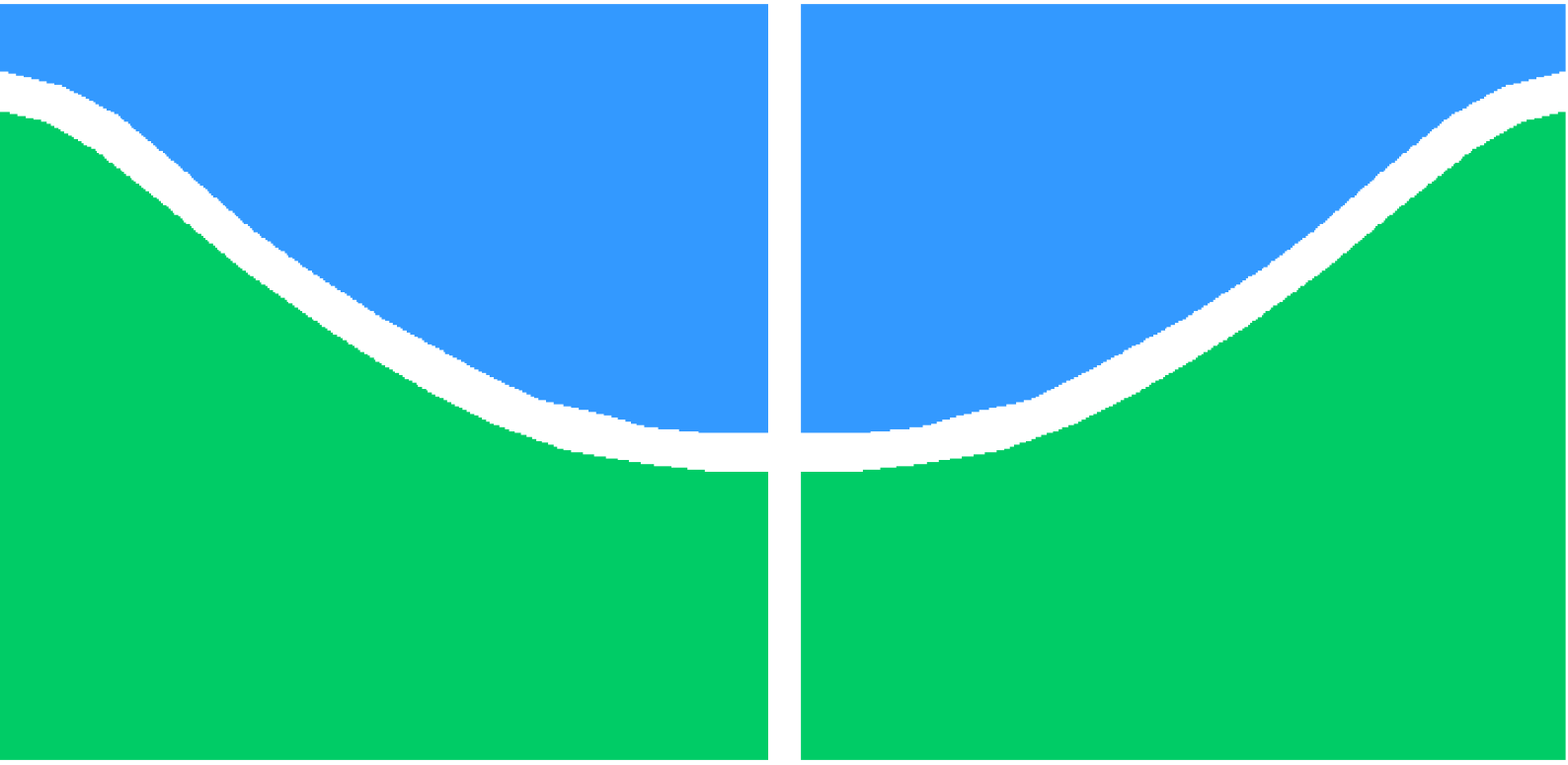

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Construção de uma aplicação de videoconferência com WebRTC

Autor: Gabriel Marcolino Rodrigues,

Shaíne Aparecida Cardoso de Oliveira,
Victor de Souza Cabral

Professor: Fernando W.Cruz

Brasília, DF

2024

Gabriel Marcolino Rodrigues,
Shaíne Aparecida Cardoso de Oliveira,
Victor de Souza Cabral

Construção de uma aplicação de videoconferência com WebRTC

Trabalho da disciplina de Redes do curso de Engenharia de Software da Universidade de Brasília.

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

Orientador: Fernando W.Cruz

Brasília, DF
2024

Sumário

1 Introdução

Nos últimos anos, o avanço das tecnologias de comunicação e o aumento da demanda por soluções remotas tornaram as ferramentas de videoconferência indispensáveis em diversos contextos, como o corporativo, educacional e social. Esse cenário foi intensificado pela pandemia de COVID-19, que acelerou a adoção de plataformas digitais para manter a interação entre pessoas de diferentes partes do mundo. No entanto, apesar da ampla oferta de soluções comerciais, como Zoom e Google Meet, muitas dessas ferramentas apresentam limitações quanto à personalização, integração com sistemas específicos e altos custos de manutenção.

Neste contexto, surge a necessidade de desenvolver uma aplicação de videoconferência personalizada, utilizando o protocolo WebRTC (Web Real-Time Communication). O WebRTC é uma tecnologia de código aberto que permite comunicação em tempo real diretamente entre navegadores, sem a necessidade de plugins ou intermediários, oferecendo baixa latência e alta qualidade de áudio e vídeo. Essa solução é ideal para criar sistemas customizados que atendam a requisitos específicos, como a integração com outros serviços ou funcionalidades adicionais de controle e segurança.

O presente relatório tem como objetivo descrever o processo de construção de uma aplicação de videoconferência utilizando WebRTC, abordando desde a definição do problema até a implementação da solução proposta. Serão discutidos aspectos técnicos do desenvolvimento, como a configuração dos servidores de sinalização, manipulação de streams de mídia, e a criação de interfaces de usuário, bem como as dificuldades encontradas ao longo do processo e as estratégias adotadas para superá-las.

2 WebRTC e suas características

WebRTC (Web Real-Time Communication) é uma tecnologia de código aberto que possibilita a comunicação em tempo real entre navegadores, permitindo o estabelecimento de conexões ponto a ponto sem a necessidade de software adicional. Em outras palavras, o WebRTC permite que o navegador transmita áudio e vídeo ao vivo entre você e outros usuários diretamente pela web.

Essa comunicação ocorre por meio de canais específicos criados pelo navegador, que se conectam ao site visitado, trocando informações, como o endereço IP. Embora isso viabilize uma interação eficiente, esses canais podem, em algumas circunstâncias, contornar certos métodos de criptografia da conexão.

2.1 Como funciona o WebRTC

Exemplo de uma chamada ponto a ponto

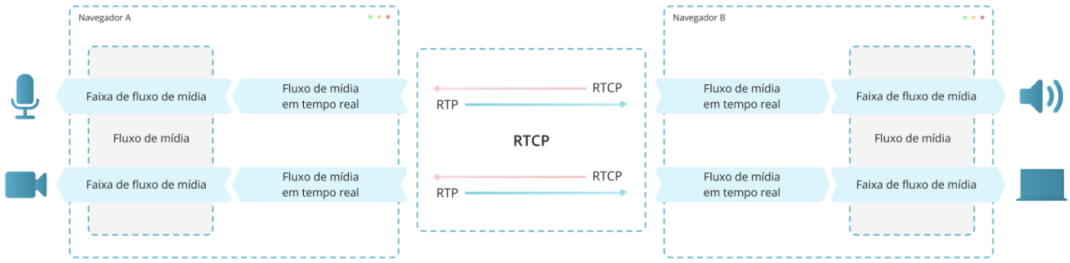

Figura 1 – Funcionamento do WebRTC

1. Um usuário abre uma página WebRTC.
2. Um navegador pode solicitar acesso a uma webcam e a um microfone. Nesse caso, o usuário precisa conceder permissão para que o aplicativo WebRTC tenha acesso aos dispositivos do usuário. Também há casos em que essa permissão não é necessária, por exemplo, ao assistir a uma transmissão ao vivo.

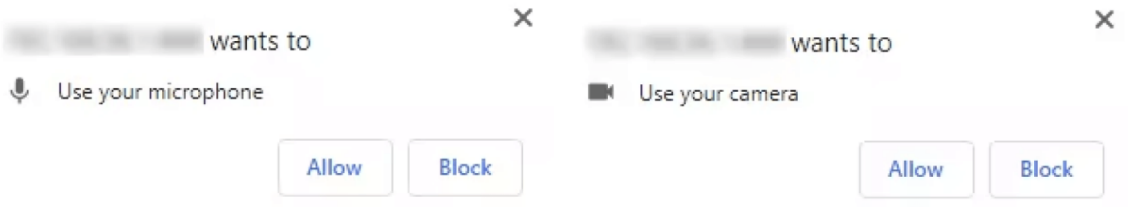

Figura 2 – Habilita câmera e microfone

3. O pacote de Protocolo de Descrição de Sessão (SDP) é gerado no navegador que inicia a conexão. Na verdade, é um arquivo de texto que contém detalhes importantes de conexão, por exemplo, tipo de dados de mídia (áudio, vídeo ou conteúdo), codecs, quais parâmetros são suportados por um navegador, etc.

4. Dependendo de como a tecnologia é implementada, um iniciador de conexão transfere esse pacote para outros participantes. Geralmente se utiliza um servidor de sinalização e protocolo WebSocket para esse propósito.

5. No lado receptor, um navegador recebe um pacote SDP e, em seguida, gera um semelhante que também obtém os dados do primeiro pacote. O segundo pacote é enviado de volta ao lado inicial. Agora, os dois clientes já se conhecem de certa forma.

6. Dependendo de sua implementação, o estado da conexão de rede é analisado juntamente com as etapas anteriores. Os clientes recebem um endereço de servidor STUN que é usado para descobrir o endereço IP externo do dispositivo. Em seguida, é comparado ao endereço IP interno para determinar se o NAT está sendo usado com a conexão e, em caso afirmativo, como os pacotes UDP são roteados. Em casos mais complexos, por ex. quando o NAT duplo é usado, os desenvolvedores usam servidores TURN. Eles são essencialmente repetidores transformando uma conexão ponto a ponto (P2P) em uma conexão cliente-servidor-cliente.

7. Depois que essas etapas forem concluídas com êxito, a conexão será estabelecida. O evento onicecandidate é solicitado periodicamente para transferir informações sobre endereços IP, configurações de NAT e tentativas de conexão entre clientes.

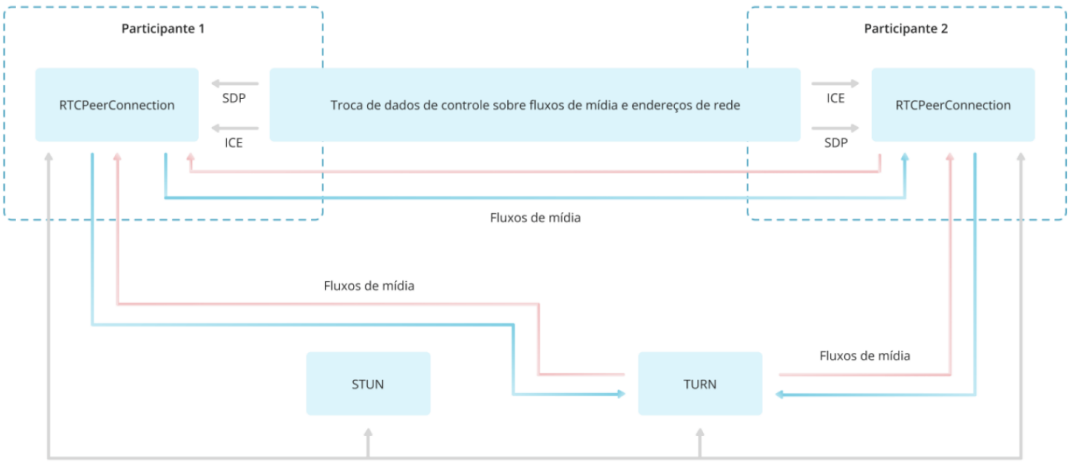

Figura 3 – Fluxo de dados

2.2 Vantagens e Desvantagens

Vantagens:

- Não requer instalação de software.
- Comunicação de alta qualidade graças a:
  - Codecs modernos de vídeo e áudio;
  - Ajuste automático de qualidade;
  - Cancelamento integrado de eco e ruído;
  - Controle Automático de Ganho (AGC).
- Foco de segurança forte: todas as conexões são protegidas e criptografadas de acordo com os protocolos DTLS e SRTP.
- O WebRTC opera apenas no protocolo HTTPS, exigindo que o site possua um certificado assinado.
- Suporte SVC adicionado como parte da implementação dos codecs VP9 e AV1.
- Apesar dos navegadores ainda não suportarem SVC, o TrueConf possibilita o SVC em navegadores de clientes.
- Recurso nativo de compartilhamento de área de trabalho.
- Interfaces de controle baseadas em HTML5 e JavaScript.

- Projeto de código aberto que pode ser incorporado ao seu produto ou serviço.
- Multiplataforma: o WebRTC funciona igualmente bem em qualquer sistema operacional — desktop ou móvel — se o navegador oferecer suporte, economizando recursos de desenvolvimento.

Desvantagens:

- As soluções WebRTC são incompatíveis entre si, pois o padrão define apenas os métodos para transferência de dados de vídeo e áudio.
- Desenvolvedores são responsáveis por decidir os métodos de endereçamento, rastreamento de status, troca de mensagens e arquivos, agendamento, etc.
- Não é possível realizar chamadas de um aplicativo WebRTC para outro.
- Usuários preocupados com segurança podem se decepcionar, pois o WebRTC detecta os endereços IP reais dos usuários.
- Nem proxies nem a rede Tor podem manter o anonimato completo no WebRTC.
- Para ocultar seu endereço IP, é possível utilizar serviços de VPN ou um servidor TURN.
- Se necessário, o WebRTC pode ser desabilitado.
- WebRTC não oferece suporte a controle remoto da área de trabalho.
- Embora seja possível transmitir a tela do dispositivo, isso ocorre como um fluxo de vídeo unilateral, semelhante a uma imagem de câmera, sem interação com a fonte.
- Essa limitação existe por razões de segurança: o código JavaScript não pode controlar nada fora da janela atual do navegador.
- Para obter mais funcionalidades, como controle remoto da área de trabalho, é possível usar aplicativos personalizados fornecidos por provedores de videoconferência.

2.3 WebRTC x Websocket

Arquitetura	Tecnologia de comunicação em tempo real	Protocolo de comunicação bidirecional
Caso de uso	Videoconferência, comunicação peer-to-peer	Chat em tempo real, ambientes colaborativos
Latência	Comunicação de baixa latência com WebRTC	Comunicação de baixa latência com WebSocket
Streaming de vídeo	Suporte nativo para streaming de vídeo em tempo real	Focado principalmente na comunicação de dados
Sinalização	Requer um servidor de sinalização para estabelecimento de sessão	Não requer sinalização explícita
Travessia de firewall	Incorpora técnicas de travessia NAT	Pode exigir configuração adicional para travessia NAT
Escalabilidade	Escalabilidade limitada devido à natureza peer-to-peer	Altamente escalável com arquitetura baseada em servidor
Complexidade	Pode ser complexo de implementar e gerenciar	Implementação e gerenciamento relativamente simples
Uso de banda	Utilização eficiente de banda para streaming de mídia	Sobrecarga mínima para comunicação de dados
Segurança	WebRTC implementa criptografia e autenticação para proteger a transmissão de dados pela internet	WebSocket implementa conexões seguras com wss:// usando soluções de criptografia SSL/TLS

3 Arquitetura utilizada

3.1 Visão Geral

A aplicação é um sistema de videoconferência e chat em tempo real. Ela é projetada para permitir a comunicação por vídeo e mensagens de texto entre usuários. A arquitetura é composta por várias partes que interagem para criar uma experiência de videoconferência interativa e funcional.

3.2 Módulos da Aplicação

3.2.1 Frontend

O frontend da aplicação é responsável pela interface com o usuário. Ele utiliza HTML para estruturar a página, CSS para estilizar os elementos e JavaScript para adicionar interatividade. A seguir está a descrição dos principais arquivos e módulos do frontend:

- **index.html**: Define a estrutura básica da página, incluindo containers para diferentes seções como a dashboard, messenger e call container. Ele organiza a disposição dos elementos e inclui referências aos arquivos de estilo e scripts JavaScript.
- **style.css**: Contém as regras de estilo para a aplicação. Define a aparência dos botões, containers, textos e outros elementos visuais. Inclui estilos para estados de foco e hover de botões, disposição de containers e formatação de mensagens.
- **JavaScript**:
  - **gerenciaDialogo.js**: Responsável pela manipulação e atualização dos elementos DOM, como exibição e ocultação de containers e atualização de mensagens.
  - **constantes.js**: Define constantes usadas em vários lugares do código, facilitando a manutenção e a leitura.
  - **gerenciaInterface.js**: Lida com a interface do usuário, incluindo a configuração e manipulação dos elementos visuais, como a atualização da interface durante uma chamada.
  - **gerenciaWebSocket.js**: Gerencia a comunicação via WebSocket, conectando a aplicação com o servidor para a troca de mensagens e eventos em tempo real.

  - **estados.js**: Implementa o gerenciamento de estado da aplicação, armazenando dados como mensagens e informações do usuário.
  - **webRTCHandler.js**: Lida com a configuração e gerenciamento das conexões WebRTC, que são essenciais para a transmissão de vídeo e áudio em tempo real.
  - **main.js**: Coordena a lógica principal da aplicação, integrando os diferentes módulos e gerenciando a comunicação entre eles.

3.2.2 Backend

- **Gerenciamento das Conexões**: Estabelece e mantém conexões WebSocket para comunicação em tempo real.
- **Sinalização WebRTC**: Facilita o processo de sinalização necessário para iniciar e gerenciar chamadas WebRTC.
- **Armazenamento de Dados**: Pode incluir a persistência de dados de usuários e sessões, dependendo das necessidades da aplicação.

3.3 Estrutura do Código

3.3.1 HTML

O código HTML estrutura a página da seguinte forma:

- **.logo_container**: Exibe o logo da aplicação.
- **.main_container**: Contém todos os principais containers da aplicação.
- **.dashboard_container**: Inclui a seção para inserção e exibição de códigos de chamada.
- **.messenger_container**: Gerencia o chat, incluindo a área de mensagens e o campo para novas mensagens.
- **.call_container**: Configura a área para chamadas de vídeo, incluindo o vídeo local e remoto e os botões de controle da chamada.

3.3.2 CSS

O CSS define o estilo visual da aplicação:

- **Estilo dos Botões**: Define o estilo padrão dos botões e suas alterações durante o hover e o foco. Isso inclui transparência, bordas e efeitos de sombra.
- **Layout dos Containers**: Define o tamanho, a posição e o estilo dos principais containers, como a área de chamadas, o painel de controle e o chat.
- **Estilo das Mensagens**: Define como as mensagens são exibidas, incluindo a formatação para mensagens enviadas e recebidas.

3.3.3 JavaScript

Os scripts JavaScript são responsáveis pela lógica e funcionalidade da aplicação:

- **Manipulação do DOM**: Atualiza a interface do usuário com base nas ações do usuário e nas atualizações de estado.
- **Comunicação WebSocket**: Gerencia a troca de dados em tempo real entre o cliente e o servidor.
- **Gerenciamento de Conexões WebRTC**: Configura e controla as transmissões de vídeo e áudio.

3.4 Testes e Funcionamento

Para tornar a aplicação funcional e testá-la:

- **Testar Funcionalidades**:
  - **Conectar-se a uma Chamada**: Teste a inserção e o gerenciamento do código de chamada.
  - **Envio e Recebimento de Mensagens**: Verifique a funcionalidade de envio e recebimento de mensagens no chat.
  - **Chamada de Vídeo**: Teste a qualidade e a estabilidade da chamada de vídeo, verificando a transmissão de vídeo local e remoto.

4 Metodologia utilizada

Para o desenvolvimento da aplicação de videoconferência utilizando WebRTC, a equipe se organizou de forma colaborativa, dividindo as tarefas em grupos menores a parte do frontend e features que são apenas para aprimorar o projeto, de acordo com as especializações e habilidades de cada integrante. No entando, a construção de servidor e conexão com o websocket foi realizado de forma conjunta, pois era o objetivo principal do trabalho.

Foram realizadas conversas informais pelo WhatsApp e tivemos encontros diários com durações variáveis, pois dependia da disponibilidade de horário dos integrantes do grupo.

4.0.1 Encontro 1: Definição do Escopo e Planejamento Inicial (10/09/2024)

- **Objetivo:** Definir o escopo do projeto e as principais funcionalidades da aplicação.
- **Decisões Tomadas:**
  - Definição das funcionalidades básicas: chamadas de vídeo/voz e chat.
  - Definição das tecnologias usadas: NodeJS no backend, JavaScript, HTML e CSS no frontend.

4.0.2 Encontro 2: Configuração do ambiente de desenvolvimento (11/09/2024)

- **Objetivo:** Subir todo ambiente de desenvolvimento e começar a implementação do Websocket.
- **Decisões Tomadas:**
  - Implementação inicial de servidores STUN.
  - Início do cliente.

4.0.3 Encontro 3: Desenvolvimento da Interface de Usuário (12/09/2024)

- **Objetivo:** Iniciar o desenvolvimento do frontend utilizando ReactJS.
- **Decisões Tomadas:**
  - Estruturação da interface com componentes para videochamada e chat.
  - Definição de eventos como iniciar/encerrar chamada.

4.0.4 Encontro 4: Integração Backend-Frontend (13/09/2024)

- **Objetivo:** Integrar o backend com o frontend para permitir a comunicação em tempo real.
- **Decisões Tomadas:**
  - Integração dos servidores de sinalização com a interface para criação de salas de videoconferência.
  - Testes de conexão ponto a ponto entre dois navegadores.
  - Melhorias no layout, como posicionamento dos vídeos e otimização do chat.

4.0.5 Encontro 5: Criação do relatório e ajustes finais (14/09/2024)

- **Objetivo:** Realizar os últimos testes e construir o relatório.
- **Decisões Tomadas:**
  - Ajustes no frontend.
  - Construção do relatório.

4.0.6 Encontro 6: Finalização do relatório e gravação de vídeo (15/09/2024)

- **Objetivo:** Revisar o projeto e preparar a documentação final.
- **Decisões Tomadas:**
  - Finalizar o relatório.
  - Revisão do código e da documentação..
  - Gravação do vídeo.

5 Conclusão

5.1 Conclusão

Diante do tempo limitado disponível para a rotina que os alunos se encontravam no momento, especialmente por estarmos no fim do semestre, enfrentamos desafios para implementar funcionalidades adicionais planejadas. Conseguimos desenvolver as funcionalidades básicas da aplicação de videoconferência e a interface de usuário com chat, mas não conseguir adicionar características adicionais, como levantar a mão e otimizações de performance. No entanto, o projeto nos proporcionou um bom entendimento do funcionamento do WebRTC e das tecnologias envolvidas, como Angular e NodeJS.

O grupo teve um pouco de dificuldade (unânime) de administrar o tempo de desenvolvimento com o fim de semestre da faculdade, pois foi composto com apenas três alunos e precisava que todos participassem e entendessem da parte principal do projeto.

5.1.1 Participação dos Membros

Cada membro do grupo teve um papel crucial no desenvolvimento da aplicação. Abaixo, apresentamos a avaliação individual de cada integrante sobre o projeto, sua participação, e a autoavaliação.

5.1.1.1 Gabriel Marcolino

- **Aprendizado e Comentários:** Durante o desenvolvimento do projeto, aprendi bastante sobre o WebRTC e sua integração com aplicações web.
- **Participação:** Participei de todo o projeto, assim como todos os integrantes do grupo.

5.1.1.2 Shaíne Oliveira

- **Aprendizado e Comentários:** O projeto foi uma excelente oportunidade para trabalhar com WebRTC e aprender mais sobre o desenvolvimento frontend com Angular.
- **Participação:** Trabalhei em todo o projeto, assim como os outros integrantes.

5.1.1.3 Victor Cabral

- **Aprendizado e Comentários:** A experiência me trouxe um bom entendimento de como integrar diferentes tecnologias e em especial a utilização de WebRTC.
- **Participação:** Participei de toda a construção do projeto.

5.1.2 Tabela de Autoavaliação

Aluno	**Nota**
Gabriel Marcolino	10/10
Shaíne Oliveira	10/10
Victor Cabral	10/10

Tabela 1 – Autoavaliação dos membros do grupo

5.1.3 Link do repositório e do vídeo

No repositório, encontra-se as instruções de como rodar, vídeo de apresentação e todo o código.

- Repositório <https://github.com/ShaineOliveira/FRC-2024>

-Vídeo de apresentação <https://www.youtube.com/watch?v=pfXgswrdlng>

6 Referências Bibliográficas

UNIVERSIDADE DE BRASÍLIA. *WebRTC*. Disponível em: <https://aprender3.unb.br/pluginfile.php/2924042/mod_resource/content/1/webrtc.pdf>. Acesso em: [11/09/2024].

TRUECONF. *WebRTC*. Disponível em: <https://trueconf.com/pt-br/webrtc.html>. Acesso em: [11/09/2024].

WebRTC. *Primeiros passos com o WebRTC*. Disponível em: <https://webrtc.org/getting-started/overview?hl=pt-br>. Acesso em: [12/09/2024].

NORDVPN. *O que é a WebRTC?* Disponível em: <https://nordvpn.com/pt-br/blog/o-que-e-a-webrtc/?srsltid=AfmBOopKNigLj7jM9wxh78rt7ysow8mRR8KLDL7fLjVvabjYoyXr1TfZ> Acesso em: [12/09/2024].

WIKIPEDIA. *WebSocket*. Disponível em: <https://en.wikipedia.org/wiki/WebSocket>. Acesso em: [13/09/2024].

DIGITAL SAMBA. *WebRTC vs WebSocket*. Disponível em: <https://www.digitalsamba.com/blog/webrtc-vs-websocket>. Acesso em: [13/09/2024].

VIDEOSDK. *WebRTC vs WebSocket*. Disponível em: <https://www.videosdk.live/blog/webrtc-vs-websocket>. Acesso em: [13/09/2024].