UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

VITOR LIMA VANACOR

Interface Administrativa para Visualização de Atividade em Sistema de Videoconferência

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Marcelo Pimenta

Porto Alegre
2019

AGRADECIMENTOS

Agradeço à minha família pela criação carinhosa que recebi, ensinando-me desde sempre solidariedade, empatia e humildade. Amo vocês.

Agradeço à minha namorada por compartilhar comigo tantos momentos maravilhosos, todos os dias. Viver contigo é incrível. Tu é incrível. Te amo.

Agradeço aos meus amigos tanto pelas conversas, festas e jogos quanto pelo apoio nos momentos difíceis. Vocês trazem cores ao meu dia.

Agradeço aos meus professores do colégio por me fornecerem uma sólida base de conhecimentos, além de me estimularem a pensar criticamente a sociedade.

Agradeço aos meus colegas e ex-colegas do CPD, do PRAV e do Mconf pelos ensinamentos e parcerias que me incentivam a iniciar minha carreira profissional motivado e confiante.

Agradeço aos professores da Universidade por me fornecerem uma educação abrangente nessa fascinante área que é a computação. Em especial, ao amigo e professor que me orientou, por me tranquilizar e me aconselhar durante a escrita deste trabalho.

RESUMO

Para qualquer administrador de um serviço, é importante ter a possibilidade de examinar dados gerados pela utilização de ferramentas e problemas que ocorrem no processo. Este trabalho apresenta as modificações feitas em um sistema para registrar informações sobre seu uso e ocorrências de erros, além do desenvolvimento de um aplicativo web administrativo para clara visualização desses registros. O sistema em questão é o *software* de videoconferência chamado Multipresença, utilizado para a realização de atendimentos no projeto "TeleOftalmo - Olhar Gaúcho", serviço que oferece exames oftalmológicos à distância para pessoas que moram longe da região central da capital do Rio Grande do Sul. A solução desenvolvida neste trabalho permite aos administradores do TeleOftalmo verificar os horários em que o Multipresença está sendo utilizado e saber quando houver alguma falha causada por falta de conexão com a internet.

Palavras-chave: Interface administrativa. videoconferência. aplicativo web.

Administrative Interface for Visualization of Activity in Videoconferencing System

ABSTRACT

For any service administrator, it is important to be able to examine data generated by the usage of tools and the problems that occur in the process. This term paper presents the modifications made to a system to record information about its usage and the occurrences of errors, as well as the development of an administrative web application for clear visualization of these records. The system in question is the videoconferencing software called Multipresence, used to perform video calls in the project "TeleOftalmo - Olhar Gaúcho", a service that offers remote ophthalmological examinations for people who live far from the central region of the capital of Rio Grande do Sul. The solution developed in this work allows TeleOftalmo administrators to check on how Multipresence is being used and to know when there is a failure caused by a lack of internet connection.

Keywords: Admin interface. Videoconference. Web Application.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

HTTP	Hypertext Transfer Protocol
REST	Representational state transfer
API	Application programming interface
JSON	Javascript Object Notation
RPC	Remote Procedure Call
SIP	Session Initiation Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
RNP	Rede Nacional de Ensino e Pesquisa
PRAV	Projetos em Áudio e Vídeo
SPA	Single-Page Application
SUS	System Usability Scale

SUMÁRIO

1 INTRODUÇÃO

Ter acesso a informações sobre a utilização de uma ferramenta, principalmente em serviços cujo seu papel é central, de maneira clara e compreensível pode ser valioso para administradores e gestores, pois possibilita verificar o que está funcionando bem e o que pode ser melhorado. No caso de ferramentas de software isso é especialmente prático, pois pode ser feito automaticamente, bastando que o sistema reporte os devidos dados e que haja uma interface intuitiva para visualizá-los.

É isto que os administradores do TeleOftalmo desejam do Multipresença, o sistema de videoconferência utilizado em seu projeto. Eles gostariam de poder examinar a atividade do sistema através da visualização de quando e por quem o sistema está sendo utilizado, quando as videochamadas estão sendo realizadas, qual a qualidade da transmissão dos vídeos nessas chamadas e quais erros ocorreram.

1.1 TeleOftalmo

O "TeleOftalmo - Olhar Gaúcho" é um serviço de telediagnóstico em oftalmologia que possui o objetivo de melhorar o acesso da população ao diagnóstico e manejo de problemas oftalmológicos, além de qualificar a lista de espera para consultas com um especialista. Em 2017, ano em que o TeleOftalmo iniciou, pelo menos 9 mil pessoas esperavam atendimento oftalmológico no Rio Grande do Sul, em um tempo de espera para consulta que ultrapassava um ano (ROESLER; LONGONI; VALLE, 2018). O projeto tem contribuído para a redução dessa fila, e desde o seu início já foram realizados mais de 15 mil telediagnósticos (TeleOftalmo. . . , 2019).

Funciona da seguinte maneira: um médico da Atenção Primária no posto de saúde encaminha o paciente para avaliação oftalmológica através da Plataforma Eletrônica do Telessaúde; então, a equipe do Telessaúde realiza o agendamento com o paciente. A avaliação é feita remotamente pelos oftalmologistas do projeto, com apoio presencial da equipe de enfermagem no consultório remoto. Após os exames, o laudo é enviado pela plataforma de Telessaúde ao médico solicitante com recomendações de conduta. Óculos são fornecidos sem custo a pacientes com erro de refração (miopia, hipermetropia, astigmatismo e presbiopia).

O projeto é uma parceria do TelessaúdeRS-UFRGS, da Secretaria Estadual da Saúde e da Associação Hospitalar Moinhos de Vento, através do Programa de Desen-

volvimento Institucional do Sistema Único de Saúde (PROADI-SUS) do Ministério da Saúde. A Figura 1.1 mostra as cidades em que os consultórios remotos estão alocados.

Figura 1.1: Cidades em que estão alocados os consultórios TeleOftalmo

Fonte: (TeleOftalmo. . . ,)

1.2 Multipresença

O software utilizado pelo TeleOftalmo para transmissão de vídeo nos atendimentos à distância é o Multipresença, um sistema de videoconferência versátil que pode ser usado para conferências de sala, telepresença ou ainda no computador pessoal (ROESLER; LONGONI; MARINS, 2015). Seu desenvolvimento iniciou em 2015, pelo PRAV[1] (Projetos em Áudio e Vídeo), financiado pela RNP[2] (Rede Nacional de Ensino e Pesquisa), e tem como objetivo ser uma solução de baixo custo e amplamente adaptável a diferentes necessidades. Foi escolhido pelo TeleOftalmo por oferecer diferenciais como:

- Comunicação ponto-a-ponto entre dispositivos na videoconferência, ou seja, áudio e vídeo são transmitidos diretamente de um computador a outro, sem a necessidade de um servidor intermediário;

[1] http://www.inf.ufrgs.br/prav/
[2] https://www.rnp.br/

- Transmissão de vídeo em alta definição (Full HD, resolução de 1920x1080p);
- Integração com o tipo de câmera utilizada nos consultórios: câmeras IP, que são câmera conectadas diretamente na rede dados;
- Possibilidade de realizar chamadas em que um ponto central se comunica com diversos pontos remotos, sem que estes últimos se comuniquem entre si;
- Desenvolvimento de novas funcionalidades sob demanda, de acordo com as necessidades do projeto.

A versão do Multipresença utilizada pelo TeleOftalmo possui modificações específicas para se adequar melhor ao projeto. Ele será melhor descrito no capítulo 3.

1.3 Objetivo

O objetivo deste trabalho é fornecer aos administradores do TeleOftalmo uma maneira de visualizar informações sobre o uso do sistema de videoconferência utilizado no projeto, o Multipresença. Isso inclui as modificações realizadas no sistema para a obtenção destas informações e o desenvolvimento de uma interface administrativa na forma de um aplicativo web para sua visualização. Espera-se que esta solução proporcione aos responsáveis pelo projeto dados relevantes para a tomada de decisões e facilite a identificação da origem de problemas, possibilitando que esses sejam corrigidos e deixem de prejudicar a experiência dos usuários.

1.4 Estrutura do Trabalho

A introdução apresentou um resumo do problema a ser resolvido, qual o contexto em que ele está inserido e o objetivo da proposta de solução. O capítulo 2 apresentará fundamentos, conceitos e tecnologias utilizadas no desenvolvimento da solução, essenciais para o entendimento do trabalho. O capítulo 3 apresentará o estado atual do sistema em que a solução opera, que apesar de funcionar bem, não registra nenhuma informação sobre seu uso. O capítulo 4 detalhará a solução proposta, descrevendo os requisitos levantados, a metodologia de desenvolvimento utilizada, a implementação em si e seu funcionamento, além da avaliação por usuários. O capítulo 5 apresentará resultados obtidos, limitações e trabalhos futuros.

2 FUNDAMENTOS, CONCEITOS E TECNOLOGIAS

Este capítulo trará conceitos e tecnologias que foram fundamentais no desenvolvimento do projeto. As primeiras seções trazem conceitos centrais ao tema da monografia, e as subsequentes resumem os principais *frameworks*, ferramentas e bibliotecas utilizados. Por fim, a última seção apresenta um breve resumo de protocolos, arquiteturas, padrões e outras tecnologias que serão citadas no decorrer do texto.

2.1 Videoconferência

A videoconferência permite que usuários fisicamente distantes entre si comuniquem-se por áudio e vídeo aproximadamente em tempo real. Essa tecnologia possibilita uma grande economia de tempo e recursos, por evitar o deslocamento dos envolvidos, e já possui aplicações de sucesso tanto para uso pessoal quanto nas áreas de negócios, saúde e educação. Pode ser realizada por dispositivos exclusivos - em que há câmera, microfone e central de processamento dedicados exclusivamente para videoconferência - ou por software, em que se utilizam câmera e microfone genéricos conectados em um computador pessoal (ROESLER et al., 2012).

2.2 Qt

Qt (pronuncia-se como a palavra inglesa *cute*) é um *framework* da linguagem C++ para desenvolvimento de aplicações desktop, sistemas embarcados e aplicativos móveis. Está disponível em licença comercial e código aberto. Por ser compatível com diversas plataformas, facilita o desenvolvimento de funcionalidades que podem ser utilizadas em Linux, Windows, OS X, Android, iOS e outros (THELIN, 2007).

Seus benefícios incluem uma ferramenta para desenvolvimento de interfaces gráficas com aparência nativa em diferentes plataformas (denominada *widgets toolkit*), um sistema de gerenciamento de eventos assíncronos, utilidades para facilitar programação concorrente, operações de rede e outros mecanismos de I/O, além de estruturas de dados abstratas, como pilhas, filas e mapas *hash*.

2.3 Ruby on Rails

Ruby on Rails é um *framework* da linguagem Ruby para desenvolvimento de aplicações web. É gratuito e de código aberto. Um de seus grandes diferenciais em relação a outros *frameworks* é sua forte preferência por "convenção sobre configuração", filosofia por vezes considerada controversa. Ele assume que o código é escrito seguindo certos princípios, "The Rails Way", (FERNANDEZ; FAUSTINO; KUSHNER, 2014) e automatiza diversas interações entre componentes baseado nesse pressuposto. Isso possui vantagens, pois dispensa o programador de perder tempo tomando decisões pouco importantes e especificando interações extremamente comuns, além de facilitar a transferência de conhecimento entre diferentes projetos, já que eles necessariamente terão a mesma estrutura e seguirão as mesmas convenções. Por outro lado, essa automatização pode ofuscar a origem de comportamentos indesejados, já que eles podem ter sido criados pela automatização "invisível" (muitas vezes referenciada até como "mágica do Rails") e não por intervenção explícita do programador.

Uma das suas convenções mais proeminentes é a utilização do padrão arquitetural "Model-View-Controller" (MVC), que visa dividir as responsabilidades da aplicação em três partes. "Model" (ou Modelo) são as regras e lógica de negócio e representação interna dos dados da aplicação, independentemente de sua exibição. "View" (ou Apresentação) responsabiliza-se pela representação visual do modelo ao usuário. "Controller" (ou Controlador) realiza a interação entre os outros dois, pois recebe mensagens externas (do usuário, através da Apresentação, ou de outros programas), valida-as e então converte-as em comandos para o Modelo (LEFF; RAYFIELD, 2001).

Outro padrão utilizado é o "Object-relational mapping" (ORM), no Rails feito pela classe "ActiveRecord", que visa facilitar a interação com bancos de dados relacionais utilizando linguagens orientadas a objeto. Ele mapeia tabelas do banco de dados para classes, permitindo manipular seus registros como instâncias da classe em que campos são representados por atributos.

2.4 Node.js

Node.js é um interpretador de código JavaScript multiplataforma e de código aberto que revolucionou a indústria por possibilitar que a mesma linguagem utilizada em clientes web (JavaScript, que roda no navegador) fosse usada para desenvolver servido-

res. É baseado no interpretador V8 desenvolvido pela Google, e sua arquitetura baseada em eventos assíncronos, não-bloqueantes e de *thread* única permite a manipulação de milhares de conexões ou requisições simultâneas em tempo real.

2.5 React.js

React.js é uma biblioteca JavaScript para criar interfaces de usuário de maneira declarativa e baseada em componentes reutilizáveis. Desenvolvida e mantida pela empresa Facebook, teve seu código aberto em 2013.

É dita "declarativa" pois desenvolver interfaces com ela é menos baseado em escrever comandos e mais em descrever como as informações devem ser apresentadas baseado no estado da aplicação, que se altera de acordo com interações do usuário ou outros eventos emitidos pelo navegador. Dá preferência por "composição sobre herança", estimulando que novos elementos sejam criados compondo outros já existentes, em vez de estendendo a funcionalidade de elementos mais genéricos.

2.6 Outros conceitos e tecnologias

- SIP (*Session Initiation Protocol*): Protocolo para sinalização de eventos em sessões multimídia interativas. Não se responsabiliza pelo transporte das mídias em si, apenas para indicar início, modificação e encerramento de sessões.
- RTP (*Real-time Transport Protocol*): Protocolo que especifica como sistemas gerenciam sua transmissão de mídia em tempo real através da rede.
- SQLite: Biblioteca para linguagem C que implementa um Sistema de Gerenciamento de Banco de Dados SQL que é compacto, veloz e autocontido.
- REST (*Representational state transfer*): Estilo de arquitetura para *Web services* em que clientes manipulam "recursos" - geralmente, representações textuais de registros de um banco de dados - providos pelo servidor através de um conjunto prédefinido de operações que não mantêm estado.
- RailsAdmin: Pacote para o framework Rails que gera automaticamente uma interface administrativa para facilmente ler, criar, atualizar e deletar registros do banco de dados.
- RTSP (*Real Time Streaming Protocol*): Protocolo para estabelecer e controlar ses-

sões de fluxos multimídia.

- WebSocket: Protocolo desenvolvido para prover comunicação bidirecional entre agentes web, atua sobre uma única conexão TCP e permite transmissões muito mais eficientes entre cliente e servidor (LUBBERS; GRECO, 2015)
- RPC (*Remote Procedure Call*): Padrão usado em sistemas distribuídos que permite a um programa chamar uma função de um programa localizado em outra máquina, abstraindo a comunicação através da rede.
- Webhook: Um método de integração em que um serviço web inscreve-se para receber atualizações de determinados eventos disparados por outro serviço web.
- JSON-RPC: Um protocolo de RPC que utiliza JSON (*JavaScript Object Notation*).

3 MULTIPRESENÇA - VISÃO GERAL

Para que se entenda as decisões tomadas no projeto e implementação da solução, é importante conhecer o funcionamento do Multipresença. A seguir, sua arquitetura e modo de utilização serão explicados.

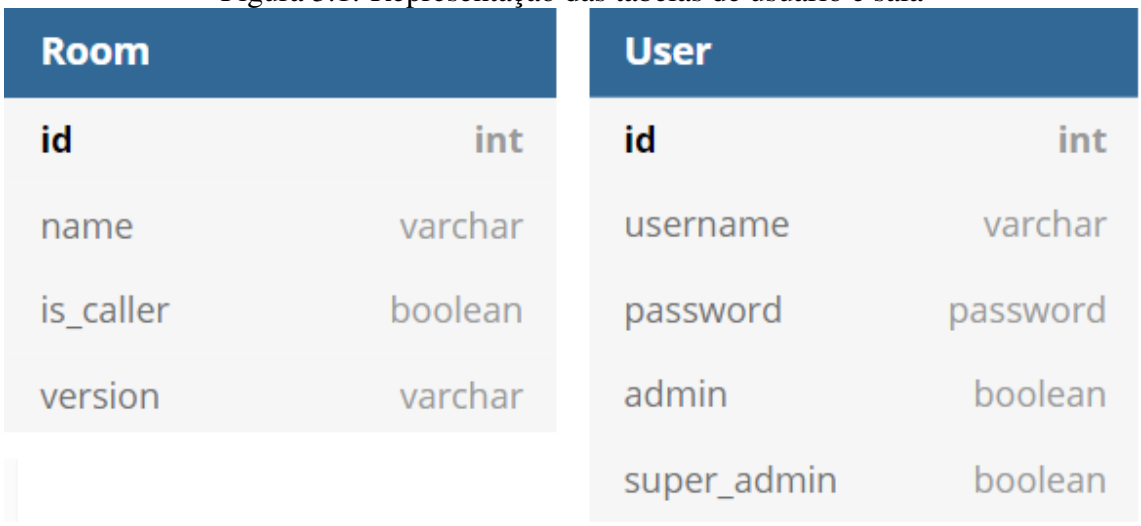
3.1 Salas

Um conceito importante para se entender o Multipresença é o de "Sala". Uma sala representa o espaço físico em que as máquinas (computadores) utilizadas nas videoconferências estão localizadas. Salas são criadas no banco de dados por um administrador, e cada máquina que instala o Multipresença cadastra-se em uma delas. Na versão original do Multipresença, é possível cadastrar diversas máquinas em uma mesma sala, algo que não é necessário para o TeleOftalmo, pois cada sala representa um consultório e somente um computador é utilizado para a videoconferência. Atualmente, há doze salas registradas:

- Quatro localizam-se na sede do TeleOftalmo, no bairro Moinhos de Vento em Porto Alegre. São denominadas "Centros de Controle" ou "Pontos de Atendimento", pois a partir delas que os médicos oftalmologistas realizam o atendimento remoto aos seus pacientes.
- As demais salas estão espalhadas pelo Estado do Rio Grande do Sul, e são denominadas "Pontos Remotos" ou "Pontos de Coleta", pois para elas que os pacientes são encaminhados a fim de terem seus exames oftalmológicos coletados. Nelas, as equipes de enfermagem operam o Multipresença quando necessário.

Os computadores utilizados para realizar as videoconferências estão configurados para inicializar o Multipresença junto com o sistema operacional, facilitando para médicos e enfermeiros, pois basta ligar a máquina e o ambiente estará pronto para que os atendimentos iniciem. Uma sala é dita "conectada" ou "online" quando o Multipresença está executando devidamente no computador associado a tal sala. Uma sala é dita "disponível" quando, além de conectada, ela não está em uma videoconferência.

Figura 3.1: Representação das tabelas de usuário e sala



Elaborada pelo autor

3.2 Usuários

Há três tipos de usuários que interagem com o sistema:

- Os médicos oftalmologistas, que iniciam as chamadas de vídeo com os pontos remotos;
- A equipe de enfermagem presente nos pontos de coleta, que recebem a chamada e podem interagir com os especialistas através de áudio, vídeo e texto;
- Os administradores, que instalam a aplicação nas máquinas, realizam as configurações necessárias e prestam suporte na ocorrência de problemas técnicos

Os pacientes não são listados pois interagem apenas indiretamente com o sistema, sem manipulá-lo, comunicando-se por áudio e vídeo com o médico realizando o atendimento.

3.3 Arquitetura

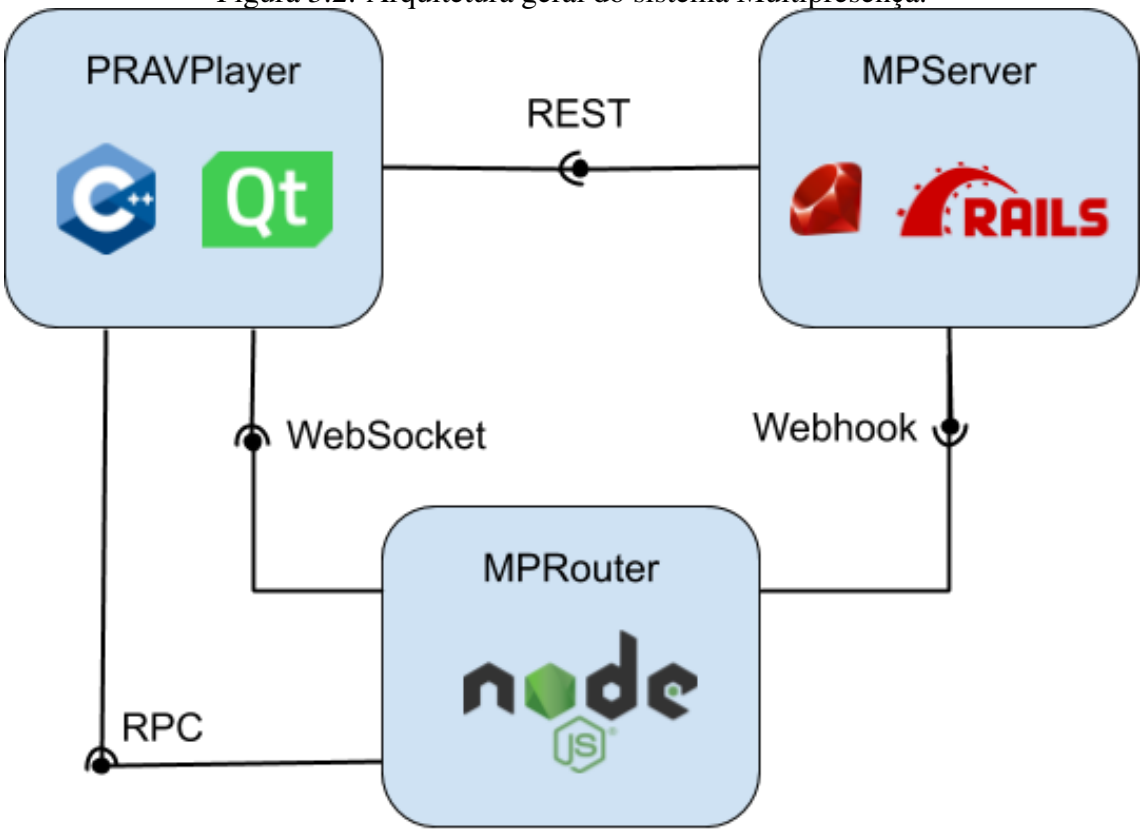
O Multipresença é um sistema distribuído com três componentes:

1. **PRAVPlayer**: Um cliente desktop;
2. **MPServer**: Um servidor RESTful;
3. **MPRouter**: Um servidor WebSocket e JSON-RPC.

A Figura 3.2 ilustra os protocolos de comunicação utilizados entre os compo-

nentes. O PRAVPlayer acessa a API REST exposta pelo MPServer através de requisições HTTP. Além disso, possui um cliente JSON-RPC para realizar chamadas remotas ao MPRouter. Também comunica-se com ele através de WebSocket, que permite comunicação bidirecional (LUBBERS; GRECO, 2015). Por fim, a comunicação entre MPServer e MPRouter utiliza Webhooks, onde o primeiro envia uma mensagem ao segundo quando um modelo é alterado no banco de dados.

Figura 3.2: Arquitetura geral do sistema Multipresença.



Fonte: Elaborado pelo autor

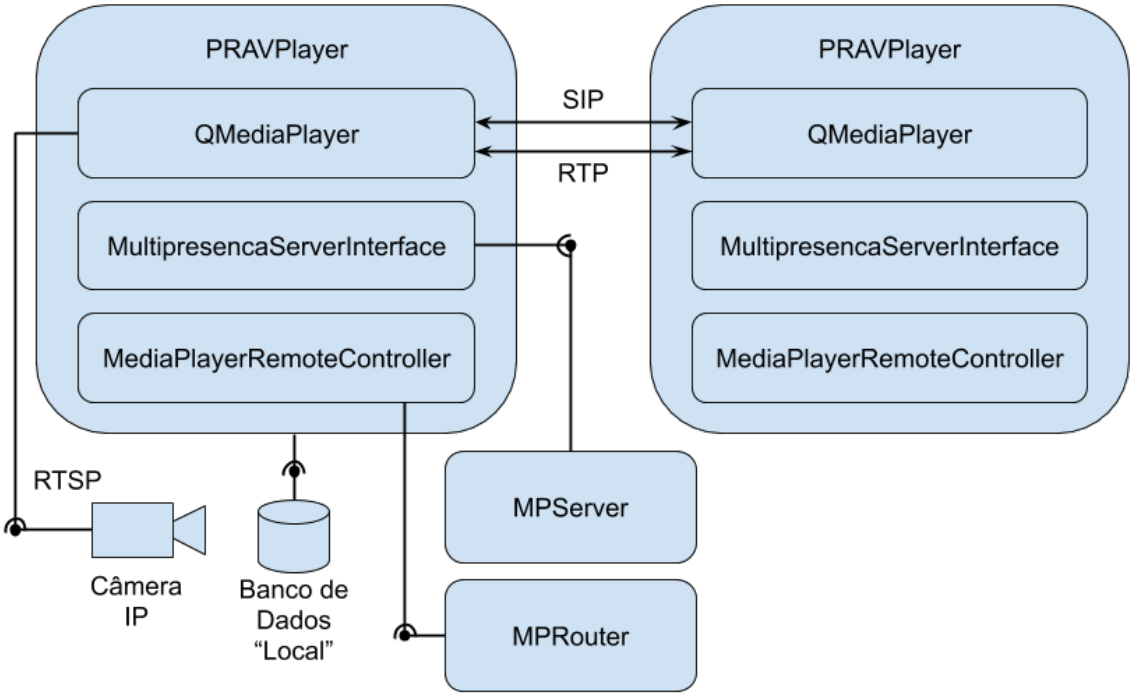
3.3.1 PRAVPlayer

Uma aplicação desktop escrita em C++ utilizando o framework Qt que roda em Windows e é instalada nas máquinas dos usuários. É o módulo principal do sistema e o único com o qual os usuários comuns (não-administradores) interagem diretamente, sendo responsável pela realização da videoconferência em si.

Possui três módulos, ilustrados na Figura 3.3:

- QMediaPlayer: Módulo responsável pela captura (recebimento de áudio e vídeo vindo de câmeras e microfones), codificação (conversão de áudio e vídeo "crus"

Figura 3.3: Arquitetura do PRAVPlayer.



Fonte: Elaborado pelo autor

para um formato comprimido, a fim de ser transmitido) e decodificação (o contrário), exibição e transmissão das mídias de áudio e vídeo. Além disso, gerencia as chamadas SIP.

- MultipresencaServerInterface: Biblioteca para abstrair a comunicação com o MPServer, mapeando métodos em C++ (com parâmetros e retornos apropriadamente tipados) para requisições HTTP.
- MediaPlayerRemoteController: Biblioteca para abstrair a comunicação com o MPRouter, mapeando métodos em C++ para chamadas JSON-RPC ou mensagens assíncronas enviadas por WebSocket.

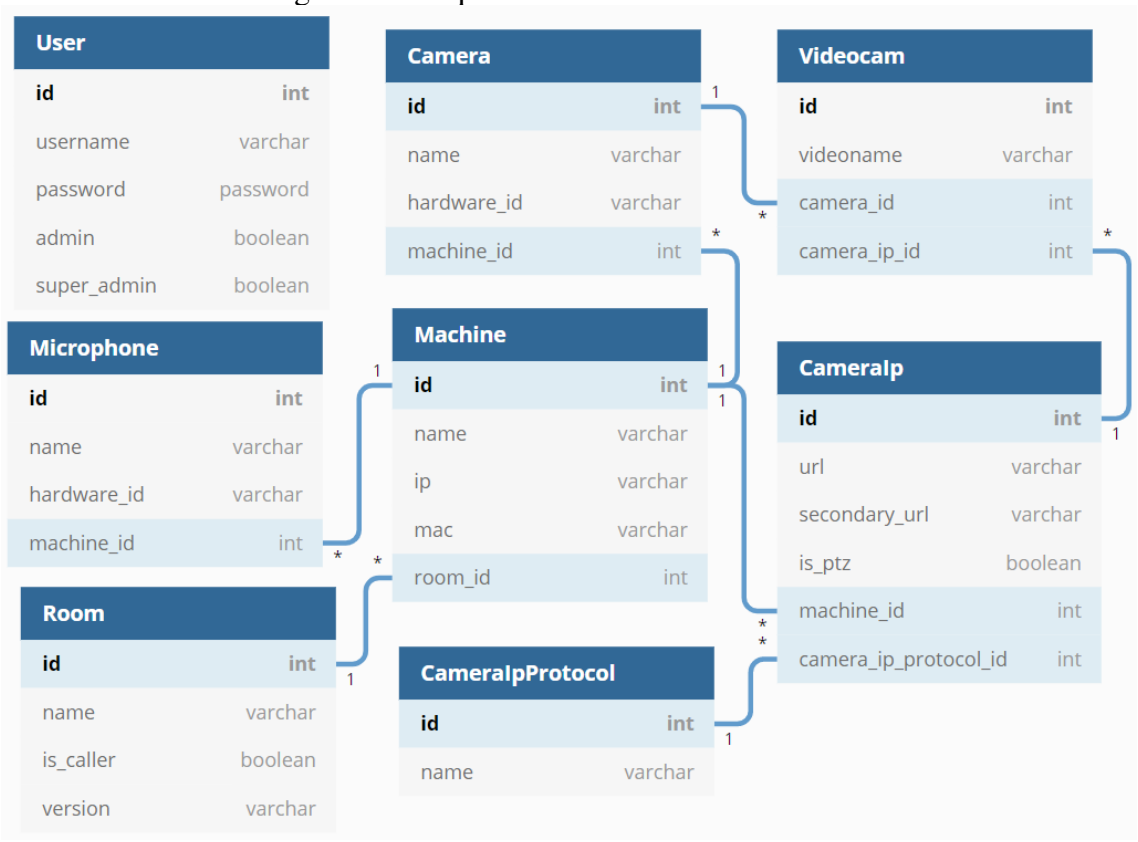
Também possui uma interface gráfica feita com o *widgets toolkit* do Qt. Comunica-se com um banco de dados SQLite para persistir e buscar configurações e outros dados que precisem ser persistidos.

3.3.2 MPServer

Um servidor web Ruby on Rails que exibe uma API baseada em REST. O Sistema de Gerenciamento de Banco de Dados utilizado é o PostgreSQL. Apesar do Rails ser um framework *full-stack*, que provê tanto lógica de negócio quanto apresentação, no

Multipresença do TeleOftalmo ele é usado apenas como *back-end* e sua função é basicamente fornecer uma interface conveniente para o banco de dados. A Figura 3.4 mostra um diagrama Entidade-Relacionamento com o esquema das principais tabelas do banco.

Figura 3.4: Esquema Geral do Banco de Dados



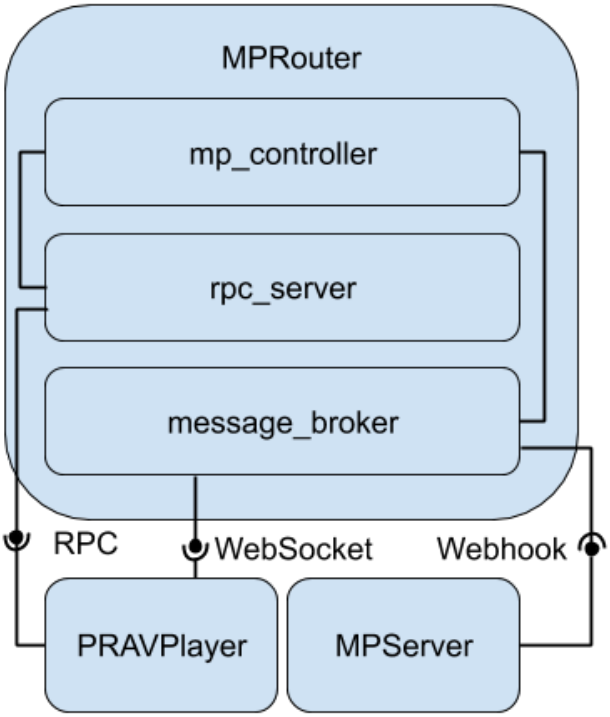
Fonte: Elaborado pelo autor

3.3.3 MPRouter

Um servidor em Node.js responsável por manter as informações dinâmicas, que não necessitam ser persistidas, como o estado das salas (se conectadas ou desconectadas) e das chamadas entre pontos. Possui três módulos, ilustrados na Figura 3.5

- `mp_controller`: Módulo principal, mantém o estado da aplicação através de classes que correspondem aos modelos do bando de dados. A função dos outros módulos é fornecer interface para manipular e consultar esse estado.
- `message_broker`: Atua como um servidor WebSocket para o PRAVPlayer e como um cliente que se inscreve por Webhook em eventos do MPServer. Possui uma classe para gerenciar as mensagens que chegam por ambos os canais.
- `rpc_server`: Um servidor JSON-RPC que o PRAVPlayer utiliza para informar

Figura 3.5: Arquitetura MPRouter



Fonte: Elaborado pelo autor

ao MPRouter que uma chamada está sendo iniciada ou encerrada.

3.4 Exemplos de Uso

Nesta seção será apresentado como o Multipresença

3.4.1 Uso pelo Oftalmologista

O oftalmologista faz login no sistema operacional e o Multipresença inicializa automaticamente. A tela inicial está na Figura 3.6. No canto inferior esquerdo encontram-se os “vídeos locais”. São os vídeos do próprio consultório médico, os que serão vistos pelo paciente após o início da chamada. À direita encontra-se o menu lateral, principal meio de interação do usuário com a aplicação. No canto superior direito, há quatro botões, respectivamente: Logout, Chat, Configurações e Fechar.

- “Logout” será utilizado para trocar o usuário da sessão Multipresença. Na verdade, ele não está presente na versão atual, sua implementação é descrita na seção “Autenticação no Cliente Desktop” do capítulo “Desenvolvimento da Solução”.

Figura 3.6: Tela inicial em um ponto de atendimento.

Fonte: O autor

- “Chat” abre uma janela que permite ao oftalmologista trocar mensagens de texto instantâneas com os médicos assistentes presentes nos pontos remotos que estejam disponíveis. O botão também pisca para avisar quando uma mensagem for recebida. A funcionalidade não será mais detalhada pois foge ao escopo do trabalho;
- “Configurações” abre uma janela para configuração de dispositivos de áudio e vídeo, como microfones e câmeras. É utilizado apenas por administradores;
- “Fechar” encerra a aplicação.

Ocupando a maior parte do menu lateral estão os botões correspondentes aos pontos remotos. Botões de pontos remotos não disponíveis ficam desabilitados e não podem ser clicados. Ao clicar em um botão habilitado ele fica azul, indicando que está selecionado. É possível que vários estejam selecionados ao mesmo tempo. Quando ao menos um estiver selecionado, o botão “Conectar”, inicialmente desabilitado, fica habilitado (indicado pela cor verde) e pode ser clicado. Quando isso for feito, será iniciada uma videoconferência entre o ponto de atendimento e os pontos remotos selecionados.

A Figura 3.7 mostra a tela de chamada nos pontos de atendimento. No centro da janela aparecerão os vídeos das câmeras localizadas nos pontos remotos selecionados para a conferência. Um dos vídeos fica em destaque - maior, à esquerda - enquanto os demais aparecem em tamanho reduzido, empilhados na direita. Clicar em um dos vídeos menores o coloca em destaque, permitindo ao oftalmologista escolher qual deve receber destaque a cada momento. O menu lateral pode ser escondido para não ficar sobreposto

Figura 3.7: Tela de videoconferência em um ponto de atendimento.

Fonte: O autor

aos vídeos reduzidos.

O menu lateral passa a exibir as salas presentes na videoconferência. Para cada sala, é mostrado:

- Nome da Sala;
- Botões para chat e alerta, à direita do nome da sala;
- Botões de controle de áudio, para deixar de receber ou enviar áudio para salas específicas;
- Vídeos da Sala, sendo que para cada vídeo há:
  - Nome do Vídeo.
  - Indicação visual da taxa de transmissão do vídeo.

  O botão “Encerrar”, em vermelho, finaliza a conferência.

3.4.2 Uso pela equipe de enfermagem remota

Os enfermeiros assistentes, presentes nos pontos de coleta, têm uma interação mais “passiva” com o sistema. Como mostrado na Figura 3.8, não há menu lateral. Eles apenas esperam até que uma chamada seja iniciada com sua sala.

Ao receber uma ligação, a janela passa a exibir os vídeos do ponto de atendimento que iniciou a ligação. O paciente e o médico assistente interagem por áudio e vídeo com

Figura 3.8: Tela inicial em um ponto remoto.

Fonte: O autor

o oftalmologista, sem nenhum tipo de contato com os outros pontos de coleta que possam estar sendo atendidos ao mesmo tempo. Não é possível encerrar a chamada, esta ação pode apenas ser feita a partir do ponto de atendimento que iniciou a chamada.

Figura 3.9: Tela de videoconferência em um ponto remoto.

Fonte: O autor

3.4.3 Uso pelo administrador

Como citado anteriormente, o administrador é responsável pelas configurações do sistema e por prestar suporte na ocorrência de problemas. Algumas de suas atribuições são:

- Registrar as salas no banco de dados. Isto é feito através da interface administrativa do MPServer;
- Instalar o Multipresença nos computadores;
- Cadastrar o computador no banco de dados. Na primeira vez que o Multipresença é executado em um computador, é exibida uma janela de cadastro, e o administrador seleciona a qual sala a máquina deve ser associada.
- Configurar microfone e webcams. Clicando no botão de configurações (ícone de engrenagem), é aberta uma janela em que é possível selecionar quais câmeras conectadas ao computador serão utilizadas, selecionar uma resolução, bitrate etc.
- Configurar câmeras IP. Através da interface administrativa, o administrador cadastra as câmeras IP no banco de dados inserindo suas URLs e especificando as máquinas associadas a elas. O vídeo de uma câmera IP aparece como miniatura no canto inferior esquerdo da tela na máquina associada a ela, junto com os vídeos das webcams conectadas; e todos são enviados como vídeos daquela sala.
- Prestar suporte. Por exemplo, se a videochamada cai, se a transmissão perde qualidade ou se alguma máquina não consegue se conectar, o administrador age: verificando se há problema na rede, se as informações salvas no banco estão corretas e até mesmo acessando remotamente os computadores nos ponto de coleta para tentar resolver o problema. Se for constatado que o problema está no código da aplicação, o desenvolvedor (o autor) é informado.

4 DESENVOLVIMENTO DA SOLUÇÃO

Este capítulo iniciará abordando os requisitos, tarefas e cronograma acordados com o cliente para o desenvolvimento da solução. Após, serão explicadas as modificações realizadas no Multipresença para ser possível coletar e armazenar os dados requisitados. Por fim, será descrito o desenvolvimento do aplicativo web para visualização das informações registradas.

4.1 Requisitos

Os esclarecimentos sobre as funcionalidades desejadas foram discutidos em trocas de e-mail e reuniões informais entre a equipe de desenvolvimento - composta pelo autor e o gerente do projeto Multipresença - e o cliente. O que se desejava era uma interface administrativa com informações sobre o uso do sistema. Levando em consideração o tempo e recursos disponíveis (4 meses e 1 desenvolvedor 30h/semana), acordou-se que deveriam ser desenvolvidas as seguintes estórias de usuário:

- Como super-administrador, quero criar e editar usuários, para poder identificar os médicos quando forem utilizar o sistema.
- Como administrador, quero ver as sessões ativas, para saber quais médicos estão utilizando o sistema no momento, assim como em qual sala e desde quando.
- Como administrador, quero ver as chamadas ativas, para saber quais pontos remotos estão sendo atendidos no momento
- Como administrador, quero ver as sessões já ocorridas, para saber quem estava utilizando o sistema em uma determinada data, assim como em qual sala, desde quando e até quando.
- Como administrador, quero ver as chamadas já ocorridas, para saber quais pontos remotos foram atendidos em uma determinada data.
- Como administrador, quero ver a qualidade da transmissão de vídeo durante as chamadas, para saber se a conexão de internet entre as salas está satisfatória.
- Como administrador, quero saber quando ocorreram erros no sistema, para tentar rastrear sua origem e assim evitar que se repitam

Estas estórias de usuário partem de alguns pressupostos que não são realidade:

por exemplo, de que se sabe qual é o usuário que está utilizando o Multipresença e de que sessões, chamadas e erros são salvos no banco de dados. Para resolver isso, foram identificados os seguintes pré-requisitos que também deveriam ser incluídos nas tarefas de desenvolvimento:

- Antes de acessar o aplicativo web, o usuário deve se autenticar para garantir que somente administradores o acessem.
- Quando o cliente desktop iniciar, o usuário utilizando-o deve ser autenticado.
- Após a autenticação do usuário no cliente desktop, uma nova sessão deve ser registrada.
- Quando uma videoconferência iniciar, novas chamadas entre os envolvidos devem ser registradas.
- Quando uma videoconferência encerrar, os términos das chamadas devem ser registrados.
- Quando o sistema for encerrado corretamente, o término da sessão deve ser registrado.
- Quando o sistema for interrompido incorretamente, um novo erro deve ser registrado.
- Periodicamente, durante uma chamada, a variação da qualidade de transmissão de vídeo deve ser registrada.

A metodologia de desenvolvimento baseia-se nos princípios do manifesto ágil (BECK et al., 2001) portanto as entregas seriam feitas incrementalmente, possibilitando ao cliente testar as novas funcionalidades em um ambiente de homologação e prover *feedback* aos desenvolvedores rapidamente, evitando que o resultado só fosse visto pelo cliente no final de todo o processo de desenvolvimento. Assim, o seguinte cronograma foi montado:

- 18/Março - 8/Abril: Autenticação nos clientes desktop e no aplicativo web
- 8/Abril - 22/Abril: Coleta e armazenamento dos registros de sessões e chamadas
- 22/Abril - 20/Maio: Exibição dos registros de sessões e chamadas
- 20/Maio - 3/Junho: Coleta, armazenamento e exibição de registros de erros no sistema
- 3/Junho - 24/Junho: Armazenamento e exibição da qualidade das transmissões de vídeo

4.2 Autenticação no Cliente Desktop

Houve ênfase por parte do cliente na importância da autenticação ser automática - com exceção apenas da primeira vez - e baseada no usuário do sistema operacional logado no momento. Ou seja, deve ser possível determinar o usuário da sessão a ser registrada sem a necessidade de inserir as credenciais todas as vezes, mantendo exatamente a mesma experiência anterior à existência de autenticação.

Primeiramente, foi implementado uma maneira de persistir a informação sobre usuários que já fizeram login. A aplicação utiliza um banco de dados SQLite para salvar configurações e outros dados que precisem de persistência. Então, criou-se nesta base uma nova tabela, `Users`, com os seguintes campos:

- `localUsername`, do tipo TEXT; nome do usuário no sistema operacional. Único, chave-primária.
- `username`, do tipo TEXT; nome do usuário salvo no banco de dados remoto do Multipresença.
- `password`, do tipo TEXT; senha do usuário. É salva para a realização da autenticação automática.

Então, foi desenvolvida a janela de login. A Figura 4.1 mostra três estados diferentes: À esquerda, antes da primeira tentativa de autenticação. O botão "Entrar" está verde pois está selecionado. No centro, campos e botões estão desabilitados enquanto a autenticação é processada. À direita, após uma falha de autenticação, há uma mensagem que é diferente dependendo se a falha foi de autenticação ou conexão.

Figura 4.1: Três estados diferentes da janela de login no cliente desktop.

Fonte: O autor

Também foi criado um método `authenticate` na classe "MultipresencaServe-

rInterface" que recebe as credenciais como parâmetro e realiza a requisição para a rota de autenticação. Caso seja bem-sucedida, o servidor retorna no corpo da resposta o id do usuário autenticado e um token, que é salvo em memória e utilizado para autenticar requisições subsequentes.

Quando o programa é aberto, busca-se no banco de dados local por um registro cujo campo `localUsername` corresponda ao do usuário atual do sistema operacional. Se não existir, exibe a janela de login para que o usuário insira suas credenciais. Se existir, envia o `username` e `password` do registro encontrado para a rota de autenticação. Se a autenticação for bem-sucedida, o programa inicia normalmente. Se falhar, a janela de login informa se o erro foi de autenticação ou de conexão e permite que o usuário tente novamente. Este processo está ilustrado no diagrama de fluxo da Figura 4.2

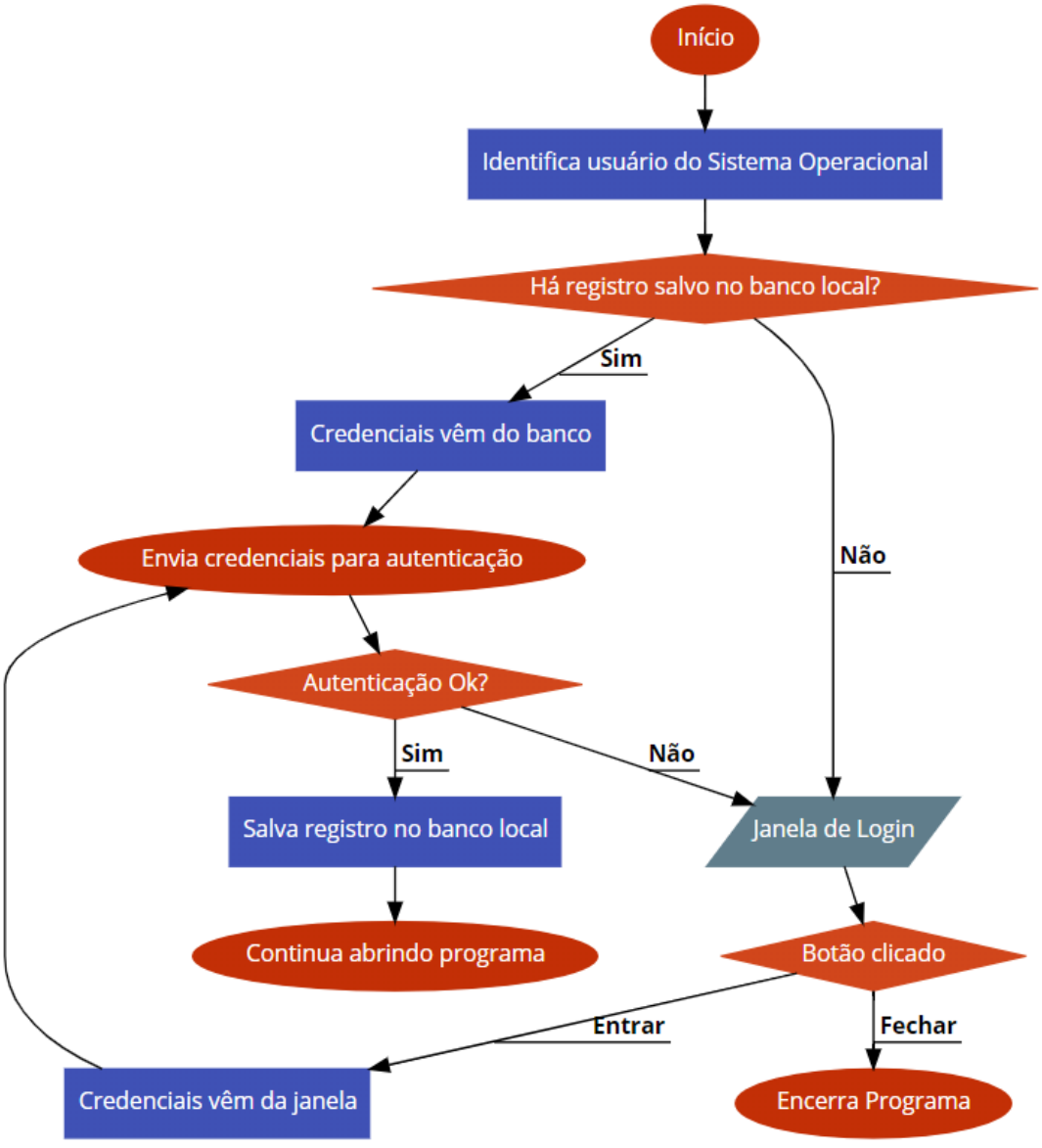
Também foi implementada uma funcionalidade de "logout", colocada em um botão à esquerda do botão de chat (previamente mostrado na Figura 3.6). Seu comportamento é simples: deleta-se do banco de dados local o registro da tabela `Users` cujo campo `localUsername` corresponder ao usuário logado no sistema operacional e reinicia-se a aplicação. Dessa forma, não haverá autenticação automática, e será possível entrar como um um novo usuário.

4.3 Coleta e Armazenamento do Registro de Sessões

A primeira informação que se tem interesse em armazenar é a de "sessão", para registrar quem, em qual sala, e em qual horário foi iniciado ou encerrado o programa. Para tanto, criou-se no banco de dados a tabela `DesktopSession`, com os seguintes campos:

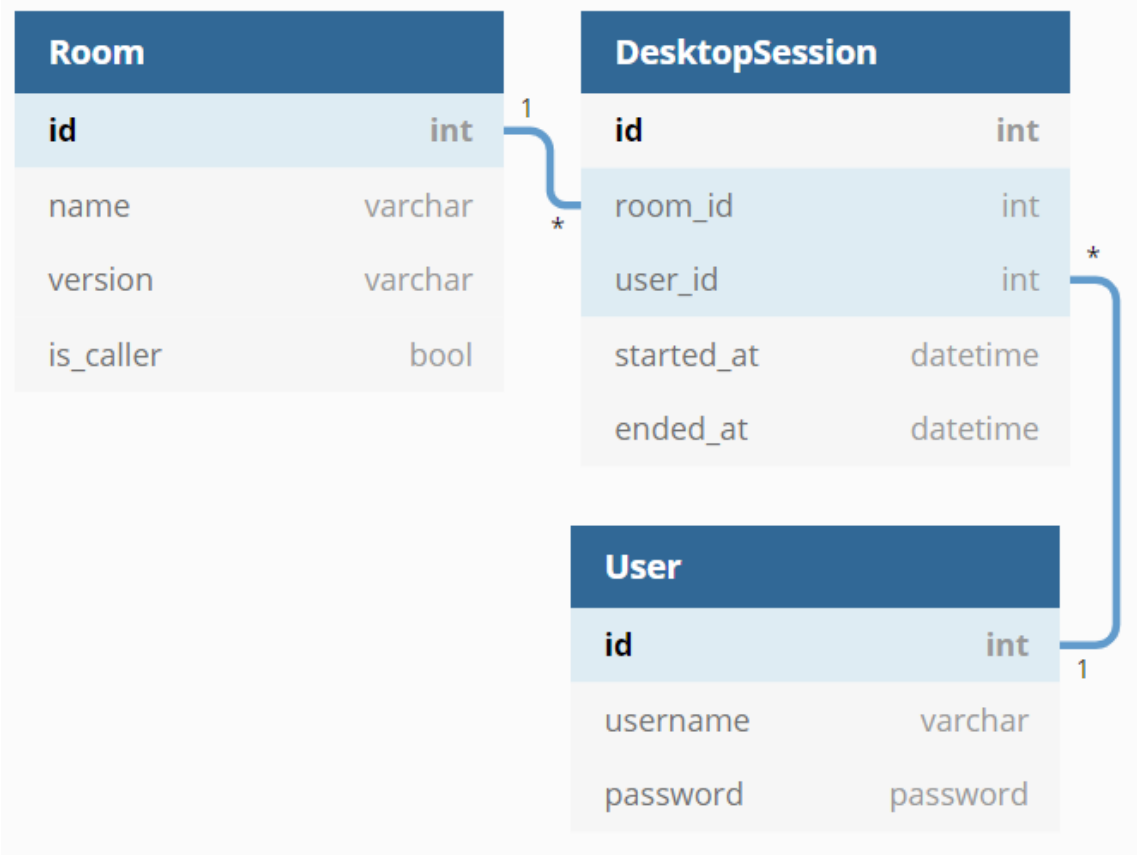
- `id`: Inteiro que identifica unicamente uma `DesktopSession`, é a chave-primária.
- `user_id`: Inteiro que corresponde ao identificador de um registro na tabela `Users`, informando quem iniciou a sessão.
- `room_id`: Inteiro que corresponde ao identificador de um registro na tabela `Rooms`, informando em qual sala iniciou-se a sessão.
- `started_at`: Campo do tipo DATETIME, indica o horário em que a sessão iniciou.
- `ended_at`: Campo do tipo DATETIME, indica o horário em que a sessão terminou.

Figura 4.2: Fluxo desenhado para a autenticação do cliente desktop.



Fonte: Elaborado pelo autor

Figura 4.3: Diagrama Entidade-Relacionamento da tabela de sessões desktop.



Fonte: O autor

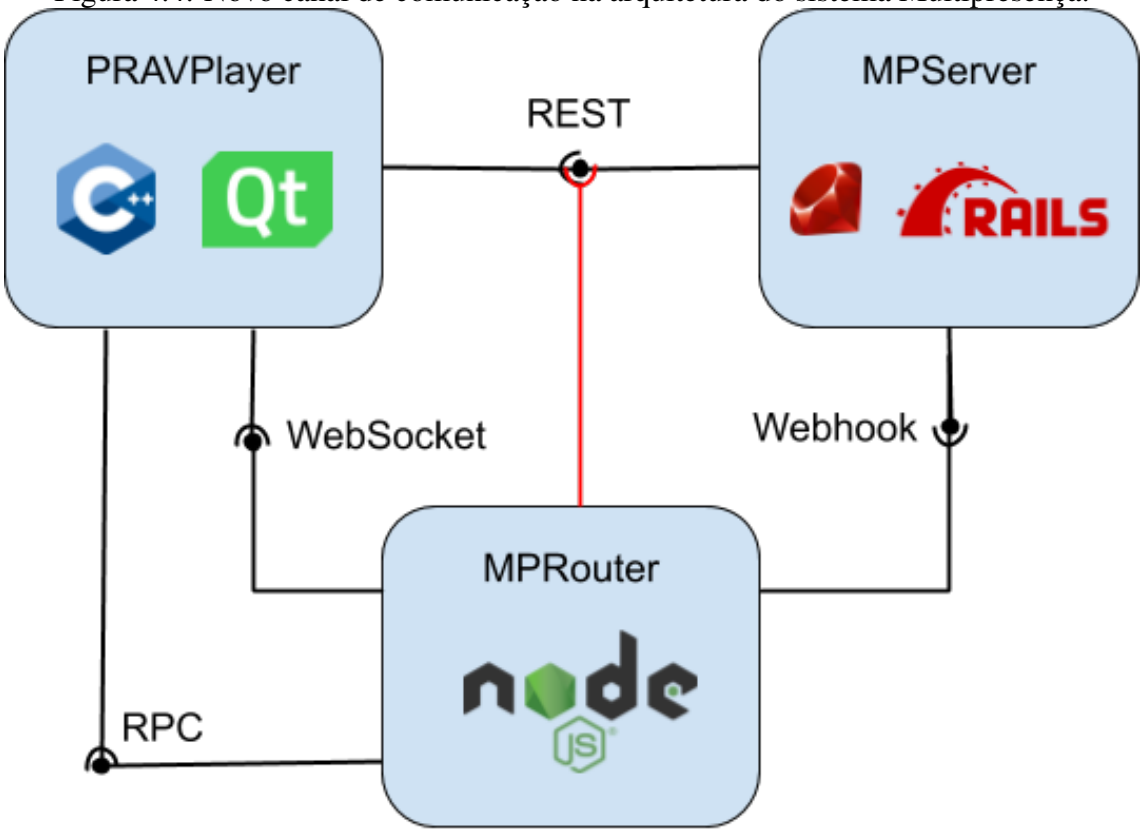
A Figura 4.3 contém um diagrama entidade-relacionamento da nova tabela e suas relações com as previamente existentes. Também foram criadas, no MPServer, duas rotas para registrar o início e fim de uma sessão:

- `/desktop_sessions/start`: Recebe requisições HTTP do tipo POST; deve ser passado o id do usuário iniciando a sessão e da sala em que ela está sendo iniciada. Cria um registro `DesktopSession` no banco com as informações recebidas, com o campo `started_at` no horário atual e o campo `ended_at` nulo, indicando que a sessão ainda está em andamento. Além disso, encerra quaisquer sessões não finalizadas naquela sala, evitando sessões sem fim.
- `/desktop_sessions/end`: Recebe requisições HTTP do tipo POST; deve ser passado o id da sessão a ser encerrada. Se o id corresponder a uma sessão válida e não-encerrada, preenche o campo `ended_at` com o horário atual.

Criadas as rotas necessárias na API, é preciso que as requisições corretas sejam feitas no início e final de cada sessão. As requisições à aplicação Rails, até então, eram realizadas apenas pelo cliente desktop. Entretanto, percebeu-se que fazer o cliente ser responsável por encerrar a sessão seria problemático, pois no caso de uma interrupção

abrupta na conexão (como uma queda de luz no consultório), não seria possível registrar o encerramento no momento certo. Por esse motivo, decidiu-se que seria mais correto o servidor MPRouter realizar as requisições de final de sessão, pois ele pode detectar quando uma conexão WebSocket é encerrada. Como descrito na seção "Arquitetura" do capítulo anterior, o MPRouter não utilizava a API REST do MPServer (Figura 3.2). Esse novo canal de comunicação teve de ser criado, ilustrado na Figura 4.4.

Figura 4.4: Novo canal de comunicação na arquitetura do sistema Multipresença.



Fonte: Elaborado pelo autor

Então, quando a o servidor Node detectar que uma conexão WebSocket foi interrompida, por qualquer que seja o motivo, ele realizará a requisição de encerramento da sessão correspondente ao MPServer. Determinou-se que o MPRouter também seria responsável por registrar o início da sessão, a fim de manter as operações simétricas. A primeira mensagem que é enviada pelo cliente ao servidor após o estabelecimento de uma conexão WebSocket é do tipo *SUBSCRIBE*, que identifica a máquina realizando a conexão. Acrescentou-se à esta mensagem informações do usuário - id e token obtidos na autenticação - que serão imediatamente utilizadas pelo MPRouter para realizar a requisição de início da sessão.

4.4 Coleta e Armazenamento do Registro de Chamadas

Outra informação que deseja-se registrar é a de chamadas entre pontos. Pelo mesmo motivo das sessões explicado anteriormente, decidiu-se que as requisições para registrar seu início e fim seriam feitas pelo servidor Node, que mantém o estado das chamadas sendo realizadas. Criou-se no banco de dados a tabela "Call", com os seguintes campos:

- `id`: Inteiro que identifica unicamente uma DesktopSession, é a chave-primária.
- `caller_session_id`: Inteiro que corresponde ao identificador de um registro na tabela DesktopSessions, informando a sessão que iniciou a chamada. Sempre será a sessão de um Ponto de Atendimento.
- `receiver_session_id`: Inteiro que corresponde ao identificador de um registro na tabela DesktopSessions, informando a sessão que recebeu a chamada. Sempre será a sessão de um Ponto de Coleta.
- `started_at`: Campo do tipo DATETIME, indica o horário em que a chamada iniciou.
- `ended_at`: Campo do tipo DATETIME, indica o horário em que a chamada terminou.

Na aplicação Rails, criou-se duas rotas para registrar o início e fim de uma chamada:

- `/calls/start`: Recebe requisições HTTP do tipo POST; deve ser passado os ids da sessão iniciando a chamada e da sessão recebendo-a. Cria um registro *Call* no banco com as informações recebidas, com o campo `started_at` no horário atual e o campo `ended_at` nulo, indicando que a chamada ainda está em andamento. Além disso, encerra quaisquer chamadas não finalizadas pela sessão recebedora, visto que uma sala não pode receber mais de uma chamada por vez.
- `/calls/end`: Recebe requisições HTTP do tipo POST; deve ser passado o *id* da sessão recebedora da ligação. Se o *id* corresponder ao de uma sessão recebedora válida cuja última chamada está em andamento, preenche o campo `ended_at` desta chamada com o horário atual.

Apesar de uma única videoconferência envolver um ponto de atendimento e diversos pontos de coleta, esta lógica determina que em uma chamada há apenas um ligador

e um recebedor, e não diversos recebedores. Por exemplo, se o ponto de atendimento A ligar para os pontos remotos B, C e D, serão registradas 3 “Calls”: Uma de A pra B, outra de A pra C e outra de A pra D. Dessa forma, se houver algum problema e um ponto remoto desconectar, isso será devidamente registrado, pois uma das chamadas será encerrada enquanto as demais terão continuado.

4.5 Coleta e Armazenamento do Registro de Erros

Os problemas mais comuns relatados pelo cliente eram a perda de qualidade da chamada e eventuais desconexões causados pela instabilidade da internet nos pontos de coleta. Portanto, o principal erro a ser registrado era o da perda de conexão com o servidor.

Para o registro dos erros, foram criadas duas tabelas: `ErrorType`, para categorizar os erros sem redundância, seguindo a forma normal de bancos de dados (DATE, 2012); e `ErrorReport`, representando a ocorrência do problema. Para esta última foram definidos os campos:

- `id`: Inteiro que identifica unicamente um `ErrorReport`, é a chave-primária;
- `desktop_session_id`: Inteiro que corresponde ao identificador de um registro na tabela `DesktopSession`, indicando a sessão em que o erro ocorreu;
- `error_type_id`: Inteiro que corresponde ao identificador de um registro na tabela `ErrorType`, o tipo do erro ocorrido.
- `reported_at`: Campo do tipo DATETIME, indica o horário em que o erro ocorreu.

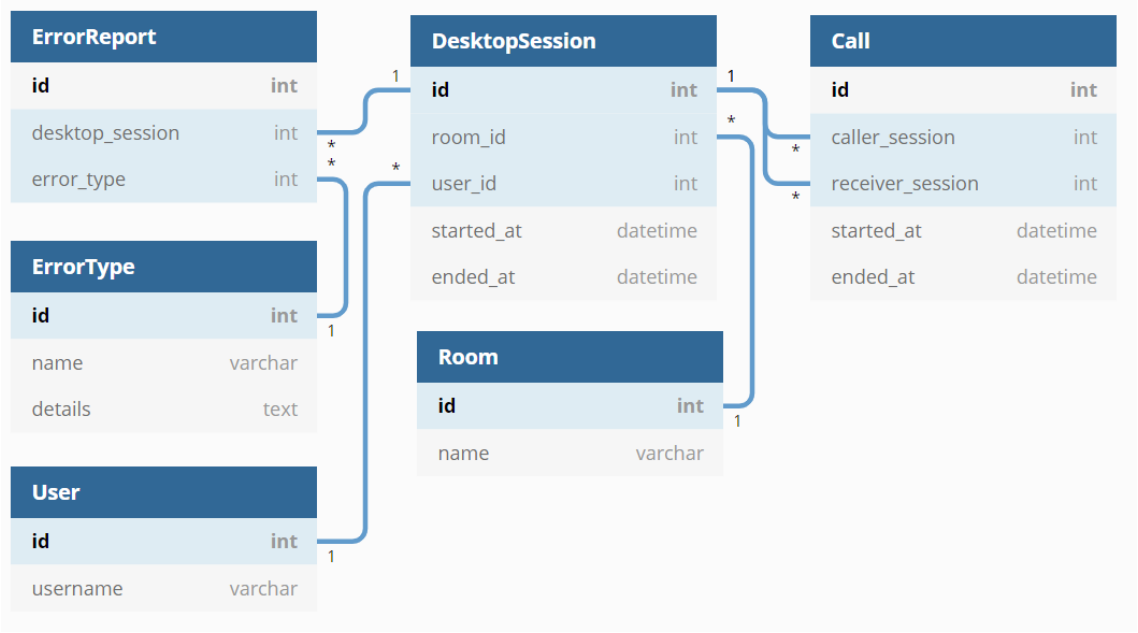
E para a tabela `ErrorType`, os campos:

- `id`: Inteiro que identifica unicamente um `ErrorType`, é a chave-primária;
- `name`: Campo do tipo VARCHAR, um nome curto e significativo para o tipo de erro;
- `details`: Campo do tipo TEXT, uma descrição mais detalhada sobre o tipo de erro.

A Figura 4.5 contém um diagrama entidade-relacionamento incluindo chamadas e registros de erros.

O único tipo de erro registrado em um primeiro momento será o de conexão interrompida. Para isso, utilizou-se o fato de que o protocolo WebSocket especifica diferentes

Figura 4.5: Diagrama Entidade-Relacionamento com Chamadas e Registros de Erro.



Fonte: Elaborado pelo autor

códigos para serem utilizados no encerramento de uma conexão. Até então, estes códigos estavam sendo ignorados. Portanto, primeiramente foi necessário fazer com que o término correto do programa no cliente (clicando no botão Fechar) encerrasse a conexão WebSocket com o servidor enviando o código 1000, indicando término normal (como mostrado na Figura 4.6). Assim, quando o programa for fechado corretamente, o servidor saberá que o encerramento foi normal pois terá recebido o código apropriado. Então, se o cliente for desconectado abruptamente (por queda de luz ou da internet, por exemplo), o servidor não receberá um *frame* de encerramento (o que é interpretado como código 1006, como mostrado na Figura 4.6) e saberá que deve registrar um novo `ErrorReport` do tipo `connection_interrupted` no banco de dados.

4.6 Configuração inicial do aplicativo Web

O aplicativo web foi desenvolvido utilizando *React.js*, uma biblioteca JavaScript para criar interfaces de maneira declarativa. Ela é especialmente apropriada para criar Aplicações de Página Única (*Single-Page Applications*), que baixam toda a lógica da aplicação no primeiro carregamento e então passam a requisitar somente os dados, quando forem necessários. Isso representa uma melhora na experiência do usuário, pois não exige que a página toda recarregue sempre que o conteúdo for ser alterado (NYGÅRD, 2015). A configuração inicial do projeto foi feita utilizando-se o script *create-react-app*, que gera

Figura 4.6: Parte dos códigos do evento *close* do WebSocket

Código de Status	Nome	Descrição
`0–999`		**Reservado e não utilizado.**
`1000`	`CLOSE_NORMAL`	Encerramento normal. A conexão foi completada com sucesso sempre que o propósito para o qual ela foi criada tenha sido atingida.
`1001`	`CLOSE_GOING_AWAY`	O "endpoint" desapareceu, por causa de uma falha no servidor ou por que o navegador navegou para fora da página que abriu a conexão.
`1002`	`CLOSE_PROTOCOL_ERROR`	O "endpoint" finalizou a conexão devido a um erro de protocolo.
`1003`	`CLOSE_UNSUPPORTED`	A conexão está sendo finalizada por causa de o dado do "endpoint" recebido ser de um tipo que não pode ser aceito (por exemplo, um "text-only endpoint" recebido como dado binário).
`1004`		**Reservado.** Um significado pode ser definido futuramente.
`1005`	`CLOSE_NO_STATUS`	**Reservado.** Indica que um código "no status" foi fornecido mesmo que qualquer outro código seja esperado.
`1006`	`CLOSE_ABNORMAL`	**Reservado.** Usado para indicar que uma conexão foi fechada anormalmente (isto é, sem o "close frame" ter sido enviado) quando um "status code" é esperado.

Fonte: https://developer.mozilla.org/pt-BR/docs/Web/API/CloseEvent

um ambiente com diversas ferramentas importantes para o desenvolvimento.

Para obter um design alinhado com boas práticas de Experiência de Usuário (UX), utilizou-se a biblioteca Material UI, que disponibiliza componentes React que seguem o padrão de aparência e interatividade definidos pelo Material Design, desenvolvido pela Google.

4.7 Autenticação no Navegador

O primeiro passo no desenvolvimento da aplicação foi a criação de uma tela de autenticação, pois o acesso à ela deve ser restrito a usuários com permissão de administrador. Esta tela deve receber as credenciais - nome de usuário e senha - do usuário e enviá-las em uma requisição para a rota de autenticação no *backend*. Caso as credenciais correspondam ao de um usuário administrador, a resposta do servidor incluirá um token, que é armazenado no navegador e utilizado para autenticar as requisições subsequentes. Seguindo o padrão *Container and Presentational Components* (ABRAMOV, 2015), foram criados dois componentes *React*:

- `LoginFormContainer`: Componente contendo apenas lógica, sem se responsabilizar pela apresentação. Lida com as validações *client-side* das credenciais e com a chamada à API do servidor. Para realizar as requisições, foi utilizada a biblioteca *axios*.
- `LoginForm`: Componente de apresentação, responsável por renderizar os elementos da tela - como botões, *labels* e *inputs* - aplicando seus respectivos estilos.

A Figura 4.7 mostra como ficou a tela de login. Após uma autenticação exitosa, há o direcionamento para a página principal da aplicação.

4.8 Layout Geral

O layout do aplicativo web é formado por três elementos, mostrados na Figura 4.8:

- A) Uma barra superior permanente, que exibe o logotipo do TeleOftalmo e um botão para abrir ou fechar o menu lateral;
- B) Um menu lateral temporário, com as opções de páginas disponíveis e um botão para fazer *logout*.

Figura 4.7: Tela de Login.

Fonte: O autor

Figura 4.8: Layout da Interface Administrativa.

Fonte: O autor

- C) O conteúdo principal da página selecionada, que varia de acordo com a opção escolhida no menu.

4.9 Gerenciamento de Usuários

A tabela de usuários já existia no banco de dados, mas quase não era utilizada pela versão do Multipresença customizada para o TeleOftalmo. Existia apenas um usuário administrador, utilizado para acessar a interface administrativa do banco de dados. Para as novas funcionalidades requisitadas, é necessário que cada médico especialista tenha um usuário próprio, para ser possível identificar a pessoa utilizando o sistema. Por isso, apesar de já ser possível gerenciar usuários pela interface do RailsAdmin, esta funcionalidade também foi colocada na aplicativo desenvolvido para facilitar o CRUD de usuários.

Figura 4.9: Tela de Gerenciamento de Usuários

Fonte: O autor

A Figura 4.9 mostra a tela de gerenciamento de usuário. Ela está disponível apenas para usuários com permissão de "Super Admin". Ao clicar no Botão "Criar Usuário" é aberto um modal, mostrado na Figura 4.10, apresentando um formulário para ser preenchido com as informações do usuário a ser criado. Ao clicar em "Salvar", é verificado no cliente se nenhum campo está vazio e se a senha bate com a confirmação. É feita uma requisição POST enviando os dados para o servidor, onde as mesmas verificações são feitas (pois não se pode confiar em nada que vem do cliente), além de outras. Por exemplo, é verificado se o nome de usuário escolhido já não existe. Se houver erros, a resposta de re-

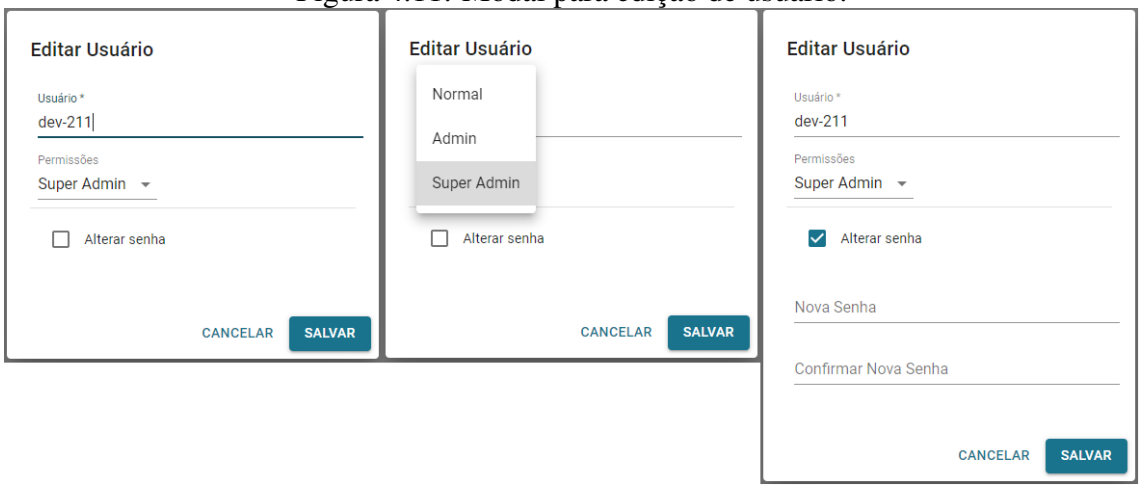
quisição incluirá quais são eles, para que sejam mostrados ao usuário no lugar apropriado do formulário. Se não houver erros, o usuário é criado no banco e adicionado à lista.

Figura 4.10: Modal para criação de usuário.

Fonte: O autor

Clicando em algum usuário já existente, abre-se um modal semelhante, como mostrado na Figura 4.11. A principal diferença é que os campos de senha só aparecem caso o checkbox "Alterar senha" seja marcado. Um usuário não pode alterar o próprio nível de permissão.

Figura 4.11: Modal para edição de usuário.



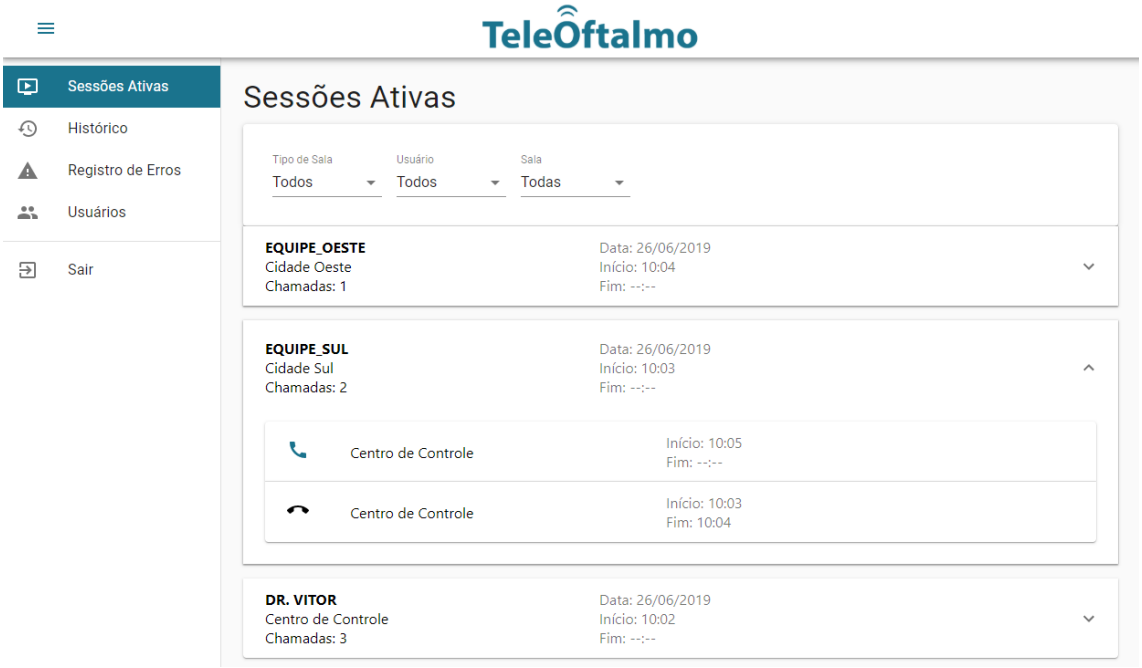
Fonte: O autor

Clicar no ícone de lixeira à direita de um usuário da lista exibe um diálogo de confirmação, para evitar que um usuário seja deletado acidentalmente. Um usuário não pode excluir a si próprio, então o ícone com lixeira não é mostrado na lista para o usuário logado.

4.10 Sessões Ativas

A tela de Sessões Ativas permite ao administrador verificar quais usuários (e em quais salas) estão utilizando o sistema no momento.

Figura 4.12: Sessões Ativas.



Fonte: O autor

Na Figura 4.12, no primeiro retângulo abaixo do título "Sessões Ativas" localizam-se os filtros de busca. Eles iniciam na opção "Todos(as)", ou seja, sem filtrar nenhum resultado: serão buscadas todas as `DesktopSession` cujo campo `ended_at` estiver nulo. Os filtro disponíveis são:

- Tipo de Sala: Possui as opções "Pontos de Atendimento" e "Pontos de Coleta". Filtram as sessões de acordo com o campo `is_caller` das salas. Também afetam o filtro "Sala", reduzindo suas opções para apenas as salas do tipo selecionado;
- Usuário: Suas opções são carregadas dinamicamente, e são buscados apenas usuários que possuem alguma sessão em uma sala Ponto de Atendimento. Isso porque os usuários nos pontos de coleta serão fixos, então mostrá-los nas opções seria redundante com o filtro de "Sala";
- Sala: Suas opções são carregadas dinamicamente, assim como o filtro dos usuários. Todas as salas são buscadas, mas apenas são exibidas nas opções aquelas cujo tipo corresponde ao selecionado no filtro "Tipo de Sala". A Figura 4.13 mostra as opções disponíveis quando o "Tipo de Sala" está como "Todos".

Figura 4.13: Opções para filtro de sala.

Fonte: O autor

E abaixo dos filtros, se houver alguma sessão ativa que corresponda aos filtros definidos, são listados os "Itens de Sessão", que mostram as seguintes informações: nome do usuário, em negrito, no campo superior esquerdo do item; nome da sala, logo abaixo do nome do usuário; o número de chamadas realizadas por aquela sessão; e data e horários de início e fim da sessão. A seta à direita indica que há mais informações caso o item seja clicado.

Clicando em um item de sessão, suas chamadas são exibidas em uma lista. Para cada chamada, um ícone à esquerda indica o estado da chamada: telefone preto virado para baixo em chamadas finalizadas e azul virado à esquerda para chamadas em andamento. À direita do ícone, o nome da sala com a qual a chamada está sendo feita; e mais à direita, horários de início e fim da chamada.

4.11 Histórico de Sessões

Tela para buscar qualquer sessão já realizada. É muito similar à "Sessões Ativas": ambas são o mesmo componente React, apenas com um parâmetro diferente para indicar que a busca não deve ser apenas nas sessões não-encerradas. Há um filtro de "Data", com quatro opções: "Hoje", "Ontem", "Últimos 7 dias" e "Definir intervalo", mostrada na Figura 4.14. As três primeiras são autoexplicativas. A última, ao ser selecionada, revela dois campos de seleção de data: "Do dia" e "Ao dia", permitindo ao usuário filtrar apenas

Figura 4.14: Histórico de Sessões.

Fonte: O autor

sessões dentro de um determinado intervalo, mostrado na Figura 4.15.

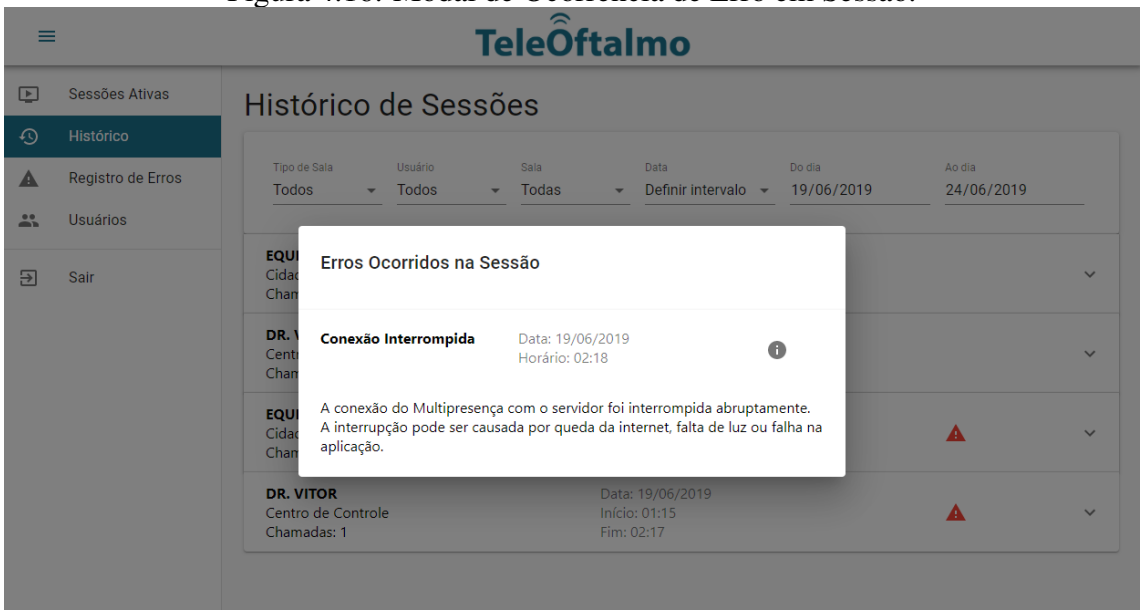
Figura 4.15: Filtro de sessões por intervalo.

Fonte: O autor

Quando a data selecionada corresponder a um só dia (como “Hoje” e “Ontem”), todas as sessões do dia são buscadas no servidor, e os demais filtros são aplicados apenas na interface. Dessa forma, mudar o filtro de sala ou usuário não gera novas requisições, tornando a interação mais rápida. Entretanto, quando a busca envolver mais de um dia, os demais filtros também são utilizados nos parâmetros de URL da requisição, reduzindo o escopo da busca no banco de dados e o tamanho da resposta. Isso foi feito pois quando há um intervalo de vários dias, o número de resultados retornados pode crescer bastante,

sendo que possivelmente há interesse apenas em sessões de um usuário ou sala.

Figura 4.16: Modal de Ocorrência de Erro em Sessão.



Fonte: O autor

Quando tiver sido registrado um erro em uma sessão, haverá um ícone vermelho no componente que representa a sessão. Ao clicar-se no ícone, é aberto um modal listando tipo, data e horário dos erros ocorridos na sessão, mostrado na Figura 4.16. Clicar no ícone de informação revela os detalhes daquele tipo de erro.

4.12 Registro de Erros

A última tela, mostrada na Figura 4.17 é similar à de histórico de sessões - os filtros são exatamente os mesmos - mas com a finalidade de mostrar apenas ocorrências de erros. Para cada erro, é mostrado seu nome (em negrito, na esquerda), a sala em que ocorreu (logo abaixo do nome), data e horário. Mais à direita há um ícone de informação que, como antes, revela os detalhes do tipo de erro, como mostrado na Figura 4.18.

Expandir o item de erro exibe a sessão em que ele ocorreu. Assim como os demais itens de sessão, pode ser expandido para revelar as chamadas realizadas na sessão.

4.13 Avaliação

Para verificar se a interface desenvolvida é eficaz, eficiente e satisfatória para o cliente, aplicou-se a ferramenta SUS (*System Usability Scale*) (BROOKE et al., 1996),

Figura 4.17: Tela de Registros de Erros.

Fonte: O autor

Figura 4.18: *Tooltip* com detalhes do erro.

Fonte: O autor

que fornece uma maneira simples de atribuir um valor numérico para a usabilidade de um sistema - atributo que é naturalmente subjetivo. Consiste em um questionário, entregue ao participante após ele interagir com o sistema por alguns minutos e realizar algumas tarefas, com uma lista de 10 (dez) afirmações em que o participante deve atribuir para cada uma delas uma pontuação de 1 (um) a 5 (cinco) segundo a Escala Likert: 1 para "Discordo totalmente" e 5 para "Concordo totalmente".

Idealmente será aplicado em diversos usuários, mas apenas um havia disponibilidade a tempo de finalizar este trabalho. É o principal administrador do TeleOftalmo, portanto suas impressões são de extrema pertinência. As afirmações do questionário, assim como o valor que o administrador atribuiu a cada uma delas, estão na tabela 4.1. Calculando-se a pontuação total como proposto por Brooke et al. (1996), chega-se em 85 (oitenta e cinco), valor que segundo Bangor, Kortum and Miller (2009) pode ser considerado "Excelente".

Tabela 4.1: Resultado da Aplicação do SUS

Afirmação	*Valor*
1. Eu acho que gostaria de usar esse sistema com frequência	5
2. Eu acho o sistema desnecessariamente complexo	1
3. Eu achei o sistema fácil de usar	4
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	4
5. Eu acho que as várias funções do sistema estão muito bem integradas	5
6. Eu acho que o sistema apresenta muita inconsistência	1
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente	5
8. Eu achei o sistema esquisito de usar	1
9. Eu me senti confiante ao usar o sistema	5
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema	3

Fonte: O Autor

5 CONCLUSÃO

Transformar eventos do sistema em dados e visualizá-los através uma interface amigável pode empoderar o administrador de um sistema, por permitir ter *insights* sobre sua utilização e rastrear problemas. Neste trabalho foram descritas as modificações realizadas no sistema Multipresença para que sessões, chamadas e erros fossem devidamente registrados no banco de dados. Além disso, foi descrito o aplicativo web desenvolvido para visualizar essas informações. Os resultados da avaliação com usuário mostram que o sistema satisfaz as necessidades do cliente.

5.1 Limitações

O trabalho, evidentemente, tem algumas limitações. A principal delas é que os dados buscados no servidor não são salvos no navegador, fazendo com que a requisição precise ser repetida toda vez que a página é renderizada: isso causa uma demora desnecessária já que em grande parte das vezes os dados a serem exibidos serão os mesmos que anteriormente. Além disso, não há paginação dos resultados, e isso faz com que o tempo de carregamento seja muito alto quando muitos itens correspondem à busca: por exemplo, com sessões em um intervalo de meses.

5.2 Trabalhos Futuros

Como sugestões de trabalhos futuros para dar continuidade a este, tem-se:

- Atualizar dados em tempo real, fazendo com que não seja necessário carregar a página para receber as informações mais recentes;
- Exportar relatórios para PDF, permitindo ter um registro formal sobre o uso do sistema;
- Detectar e registrar outros tipos de erros, aumentando o poder de rastreamento de falhas no sistema;
- Registrar e exibir a qualidade de transmissão das chamadas, permitindo verificar se a velocidade de conexão da internet está apropriada.

REFERÊNCIAS

ABRAMOV, D. **Presentational and Container Components**. 2015. <https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0>. Acessado em 3 de Julho de 2019.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, Usability Professionals' Association, v. 4, n. 3, p. 114–123, 2009.

BECK, K. et al. **Princípios por trás do Manifesto Ágil**. 2001. <https://www.manifestoagil.com.br/principios.html>. Acessado em 3 de Julho de 2019.

BROOKE, J. et al. Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London–, v. 189, n. 194, p. 4–7, 1996.

DATE, C. **Database Design and Relational Theory**. [S.l.]: O'Reilly, 2012.

FERNANDEZ, O.; FAUSTINO, K.; KUSHNER, V. **The Rails 4 Way**. [S.l.]: Addison-Wesley Professional, 2014.

LEFF, A.; RAYFIELD, J. T. Web-application development using the model/view/controller design pattern. In: **Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing**. Washington, DC, USA: IEEE Computer Society, 2001. (EDOC '01), p. 118–. ISBN 0-7695-1345-X. Available from Internet: <http://dl.acm.org/citation.cfm?id=645344.650161>.

LUBBERS, P.; GRECO, F. **HTML5 Web Sockets: A quantum leap in scalability for the Web**. 2015. <http://www.websocket.org/quantum.html>. Acessado em 3 de Julho de 2019.

NYGÅRD, K. Single page architecture as basis for web applications. In: . [S.l.: s.n.], 2015.

ROESLER, V. et al. Mconf: An open source multiconference system for web and mobile devices. In: **Multimedia - A Multidisciplinary Approach to Complex Issues**. [S.l.]: InTech, 2012.

ROESLER, V.; LONGONI, G.; MARINS, A. Multipresença: um sistema de videoconferência adaptável, escalável e interoperável. In: **Conferencia de Directores de Tecnologia da Informacion, TICAL**. [S.l.: s.n.], 2015.

ROESLER, V.; LONGONI, G.; VALLE, R. Videoconferência multiambientes: o sistema multipresença com foco na área da saúde. In: **Anais Estendidos do XXIV Simpósio Brasileiro de Sistemas Multimídia e Web**. Porto Alegre, RS, Brasil: SBC, 2018. p. 187–190. ISSN 2596-1683. Available from Internet: <https://portaldeconteudo.sbc.org.br/index.php/webmedia_estendido/article/view/4081>.

TeleOftalmo Dados. 2019. <https://www.ufrgs.br/telessauders/dados-teleoftalmo/>. Acessado em 3 de Julho de 2019.

TeleOftalmo Inicial. <https://www.ufrgs.br/telessauders/telediagnostico/teleoftalmo/>. Acessado em 3 de Julho de 2019.

THELIN, J. **Foundations of Qt Development**. [S.l.]: Apress, 2007.