INSTITUTO FEDERAL DE SANTA CATARINA

MATHEUS MEDEIROS

Aplicativo Progressive Web App para Interação, Gerenciamento e Monitoramento com o Laboratório Remoto de FPGAs (eLab FPGA)

São José - SC

dezembro/2023

APLICATIVO PROGRESSIVE WEB APP PARA INTERAÇÃO, GERENCIAMENTO E MONITORAMENTO COM O LABORATÓRIO REMOTO DE FPGAS (ELAB FPGA)

Projeto de Trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina

Orientador: Prof. Marcos Moecke, Dr.

São José - SC

dezembro/2023

RESUMO

Laboratórios remotos têm se tornado uma prática cada vez mais comum e importante nas universidades nos últimos anos, sobretudo na educação em áreas científicas e engenharias, beneficiando alunos, professores e instituições de ensino. Um laboratório remoto oferece aos alunos a oportunidade de controlar a distância equipamentos físicos e ferramentas de *softwares*, com experiência de aprendizagem semelhante a um laboratório real. Isso traz vantagens não apenas aos alunos que podem aprender de forma mais autônoma, mas também para os professores que podem orientar os alunos independente da presença física no laboratório. Já as universidades obtêm benefícios em relação à gestão mais eficiente de seus recursos, ao custo de aquisição de novos equipamentos e *softwares*, além de disponibilizar a qualquer hora seus equipamentos para ensino e pesquisa. Neste sentido e com o propósito de facilitar o acesso a recursos de diferentes plataformas de *Field Programmable Gate Array* (FPGA) em um ambiente virtual, o projeto de laboratório eLab é desenvolvido para oferecer controle, monitoramento e acesso a programação de *kits* de desenvolvimento FPGA remotamente. Este trabalho tem o objetivo de implementar uma plataforma de interação, video monitoramento e gerenciamento de acesso aos *kits* de FPGA. Para atingir esse objetivo, será desenvolvida uma aplicação *Progressive Web App* (PWA) que permitirá a interação com os componentes, chaves e botões nos kits de FPGA do eLab. Além disso, será possível acompanhar os resultados por meio de monitoramento em vídeo em tempo real e agendar horários para acessar um dos kits.

Palavras-chave: Laboratório Remoto. FPGA. Progressive Web App.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura básica *Complex Programmable Logic Device* (CPLD) . . . 14
Figura 2 – Arquitetura básica de um FPGA 14
Figura 3 – Arquitetura básica de um Bloco Lógico Configurável (CLB) 15
Figura 4 – Exemplo de negociação de uma abertura de *WebSocket* 18
Figura 5 – Diagrama de blocos do laboratório remoto 21

LISTA DE ABREVIATURAS E SIGLAS

ACID Atomicidade, Consistência, Isolamento e Durabilidade.

API *Application Programming Interface.*

CLB Bloco Lógico Configurável.

CPLD *Complex Programmable Logic Device.*

CSS *Cascading Style Sheets.*

DLP Dispositivo Lógico Programável.

DOM *Document Object Model.*

DSP *Digital Signal Processor.*

FLASH EEPROM *Flash Electrically Erasable Programmable Read-Only Memory.*

FPGA *Field Programmable Gate Array.*

GAL *Generic Array Logic.*

GPIO *General Purpose Input Output.*

GT-MRE Grupo de Trabalho em Experimentação Remota.

HTML *Hypertext Markup Language.*

HTTPS *Hypertext Transfer Protocol Secure.*

I/O Entrada e Saída.

ICE *Interactive Connectivity Establishment.*

IP *Internet Protocol.*

JSON *JavaScript Object Notation.*

JTAG *Joint Test Action Group.*

LED *Light Emitting Diode.*

LUT *Look-Up Table.*

MCU *Multipoint Control Unit.*

NAT *Network Address Translation.*

NoSQL *Not Only SQL.*

P2P *Peer-to-Peer.*

PAL *Programmable Array Logic.*

PLA *Programmable Logic Array.*

PLD *Programmable Logic Device.*

PLL *Phase-Locked Loop.*

PWA *Progressive Web App.*

PXI *PCI eXtensions for Instrumentation.*

RELLE *Remote Labs Learning Environment.*

RExLab Laboratório de Experimentação Remota.

RTC *Real-Time Communications.*

SFU *Selective Forwarding Unit.*

SGBD Sistema de Gerenciamento de Banco de Dados.

SGBDR Sistema de Gerenciamento de Banco de Dados Relacional.

SPLD *Simple Programmable Logic Device.*

SQL *Structured Query Language.*

SRAM *Static Random Access Memory.*

STUN *Session Traversal Utilities for NAT.*

TURN *Traversal Using Relays around NAT.*

UFSC Universidade Federal de Santa Catarina.

UML *Unified Modeling Language.*

URI *Uniform Resource Identifier.*

URL *Uniform Resource Locator.*

VHDL *VHSIC Hardware Description Language.*

VISIR *Virtual Instrument Systems in Reality.*

WEB *World Wide Web.*

WebRTC *Web Real-Time Communications.*

WS *Websocket.*

WSS *Websocket Secure.*

SUMÁRIO

1 INTRODUÇÃO . . . 9
1.1 Objetivo geral . . . 10
1.1.1 Objetivos específicos . . . 10

2 FUNDAMENTAÇÃO TEÓRICA . . . 11
2.1 Laboratórios remotos . . . 11
2.1.1 Soluções existentes . . . 12
2.2 Dispositivos lógicos programáveis . . . 13
2.2.1 SPLD . . . 13
2.2.2 CPLD . . . 13
2.2.3 FPGA . . . 14
2.3 PWA . . . 16
2.3.1 *Service worker* . . . 17
2.4 *Websockets* . . . 17
2.5 WebRTC . . . 18
2.6 Banco de dados . . . 19

3 PROPOSTA . . . 21
3.1 Cronograma de atividades . . . 23

REFERÊNCIAS . . . 24

1 INTRODUÇÃO

Field Programmable Gate Array (FPGA) é um circuito integrado composto por uma matriz bidimensional de células logicas interligadas por chaves programáveis que podem ser configuradas independentemente, conforme a necessidade de processamento (CHU, 2008). Segundo Mayoz et al. (2020) a arquitetura paralela, baixo consumo de energia e adaptação a algoritmos personalizados faz com que está tecnologia desempenha um papel significativo na indústria. A programação de um FPGA é conduzida por meio de linguagens de programação de descrição de *hardware* como *VHSIC Hardware Description Language* (VHDL) e Verilog, utilizando ferramentas de programação de *software* específicas para cada plataforma de *hardware* de diversos fabricantes, tais como Quartus Prime da Intel, Vivado da AMD, além de outras opções disponíveis gratuitamente.

A instalação destas plataformas e ferramentas pode ser complexa e demorada para iniciantes em programação de *hardware*, com problemas de compatibilidade de *hardware* e *software*. Dado a demanda do mercado por profissionais na área de Dispositivo Lógico Programável (DLP) e a dificuldade de treinamento devido aos diferentes modelos de FPGAs e programação de *hardwares* que são projetadas e mapeadas especificamente para os dispositivos FPGA de cada fabricante. Uma alternativa viável é utilizar plataformas de programação em nuvem com diferentes FPGAs acessíveis de qualquer dispositivo com acesso à internet.

As aulas práticas de laboratório desempenham um papel importante na formação de diversas disciplinas das engenharias, incluindo aquelas relacionadas à eletrônica digital, que representam uma das áreas de aplicação prática das FPGAs, desta forma os professores possibilitam que os alunos testem seu conhecimento na prática por meio de experimentos realizados em ambiente de laboratório (BASCIL; YAZICI; TEMURTAS, 2012).

Plataformas de ensino online que oferecem acesso remoto a *kits* de desenvolvimento em FPGA ou experimentos de variadas áreas da engenharia já estão disponíveis em laboratórios como o LabsLand-FPGA Laboratory (INTEL, 2023), RELLE (CATARINA, 2023) e VISIR (GARCIA-LORO et al., 2019). Laboratórios remotos têm o potencial de proporcionar às instituições acadêmicas a capacidade de oferecer um ambiente acadêmico a estudantes de acordo com suas necessidades, e docentes fazer avaliação de seus alunos por meio de monitoramento de video (PRETZ, 2020), como consequência elimina a necessidade de acesso físico aos equipamentos e componentes necessários e permite a repetição de cenários de experimentos. Conforme Post et al. (2019), um dos benefícios que os laboratórios remotos oferecem aos alunos em processo de aprendizagem é a capacidade de

organizar suas tarefas de acordo com suas necessidades individuais. Isso significa que os alunos podem realizar suas atividades em qualquer lugar e a qualquer momento, o que promove um sentimento maior de autonomia e motivação.

Neste contexto, o eLab é um laboratório que visa oferecer uma plataforma remota com acesso a diferentes *kits* de FPGA com monitoramento através de video. A estrutura do laboratório é composto por laboratório remoto, servidor de aplicação e interface de usuário. O laboratório remoto contém *kits* de FPGA conectado através dos pinos *General Purpose Input Output* (GPIO) a uma placa de controle. Para haver interação entre placa de controle e o servidor, é utilizado um microcontrolador para tratar as mensagens vindas do servidor e controlar as entradas do FPGA, que será através da GPIO. O servidor contém um sistema de gerência capaz de realizar o controle de autenticação do usuário, troca de dados com a placa de controle, gravação da placa FPGA e monitoramento de video atuando como um gateway de mídia.

1.1 Objetivo geral

O objetivo deste trabalho é viabilizar o acesso dos usuários ao laboratório remoto por meio de uma interface *World Wide Web* (WEB), permitindo que interajam com os *kits* de FPGAs. Além de poder avaliar o desempenho do experimento em tempo real através do monitoramento de vídeo.

1.1.1 Objetivos específicos

Para alcançar o objetivo geral foi estabelecido os seguinte objetivos específicos que orientarão o desenvolvimento do projeto:

1. Desenvolvimento de API REST, como interface para acesso aos recursos do banco de dados;
2. Implementação um sistema de monitoramento remoto por imagem de vídeo dos Kits FPGA;
3. Desenvolvimento de uma aplicação PWA, como forma de interface de acesso e controle ao laboratório remoto;
4. Modificações no software da placa de controle hardware, responsável pela comunicação entre aplicação e o sistema de controle de dados.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica para o desenvolvimento deste trabalho, abordando conceitos relacionados a laboratórios remotos seção 2.1, Dispositivo Lógico Programável (DLP) seção 2.2, *Progressive Web App* (PWA) seção 2.3, *websockets* seção 2.4, *Web Real-Time Communications* (WebRTC) seção 2.5 e banco de dados seção 2.6.

2.1 Laboratórios remotos

Laboratórios remotos são ferramentas compostas por *hardware* e *software* instalados nas universidades, que permite aos alunos acessar remotamente equipamentos físicos com experiência semelhante a uma aula prática presencial, além da possibilidade de compartilhar recursos entre diferentes instituições de ensino (ORDUñA et al., 2016). Nestes laboratórios podem ser realizados experimentos de diferentes áreas do conhecimento, como robótica, eletrônica, física, química entre outros.

É destacado por Orduña et al. (2016), que em uma diversidade de laboratórios cada um tem sua arquitetura, mas algumas características da estrutura desses tipos de laboratório deve ser comum a todos, como gerenciamento de usuário com autenticação e autorização, agenda de experimentos, para garantir a disponibilidade dos equipamentos, e por fim uma interface de usuário para interação com os equipamentos. No caso de circuitos digitais a disponibilidade de um equipamento de monitoramento de vídeo é um recurso importante neste tipo de laboratório, por permitir acompanhar em tempo real o experimento, comportamento dos periféricos de entrada e saída como interruptores, botões, *Light Emitting Diodes* (LEDs), entre outros dispositivos (MAYOZ et al., 2020)

Conforme Orduña et al. (2016), um ponto-chave para os laboratórios remotos é o seu compartilhamento entre universidades, considerando três abordagens, (1) laboratórios totalmente abertos para quem quiser utilizar, (2) laboratórios com antecedência devem comunicar a lista de usuários que terão acesso ao sistema e (3) federalização, onde um laboratório é compartilhado entre as universidades que fazem parte da federação, sem a necessidade de solicitar acesso ao laboratório.

Os laboratórios remotos nas engenharias e em outros cursos das universidades visa, oferecer aos alunos um reforço das aulas teóricas aplicadas em sala de aula, sobretudo a grande quantidades de horas de laboratório que necessitaria para aplicar o seu conhecimento (SIERRA-FERNáNDEZ et al., 2020). Conforme destaca Post et al. (2019), a educação em laboratórios remotos permite a orientação simultânea de vários alunos,

tornando-se, assim, uma opção mais eficiente e econômica para as universidades. Essa eficiência decorre da capacidade de permitir que o alunos acessem o laboratório de qualquer lugar e a qualquer momento, além de não necessitar de tempo de configuração e desmontagem do equipamento ou experimento.

2.1.1 Soluções existentes

Nos últimos anos, os laboratórios remotos vêm ganhando notoriedade. Deste modo, para o ensino das engenharias surge a *e-Engineering*, com objetivo de automatizar e aprimorar o processo de ensino-aprendizagem. Seu propósito é facilitar aos alunos o acesso a equipamentos e experimentos de diferentes áreas de modo a complementar o seu currículo (SIERRA-FERNáNDEZ et al., 2020).

Iniciativas como *LabsLand* demostram esse avanço ao longo dos anos, que tem como um de seus objetivos construir uma rede de laboratórios, conectando escolas e universidades do mundo com laboratórios físicos, disponibilizados remotamente (LABSLAND, 2023). Ainda segundo LabsLand (2023), laboratórios está disponíveis em dois formatos (1) tempo real, quando o equipamento é acessado ao mesmo tempo que experimento está ocorrendo, e (2) ultraconcorrentes quando o experimento é pré-carregado. Na arquitetura da plataforma as funcionalidades de gerenciamento comuns a todos os tipos de laboratório remoto são disponibilizados através da biblioteca *WebLabLib*, já na plataforma WEB também é disponibilizado a *LabsLand Web IDE* feita para edição, síntese e compilação de arquivos VHDL e Verilog (MAYOZ et al., 2020).

Plataformas como o *Virtual Instrument Systems in Reality* (VISIR) (IFSC, 2023), é um laboratório remoto com foco no ensino teórico e prático de circuitos elétricos e eletrônicos. O VISIR tem como proposta oferecer aos alunos um laboratório muito próximo de um laboratório real, desta forma administrador, alunos, instrutores, professores e convidados podem interagir com instrumentos e equipamentos mediante a uma interface WEB (TAWFIK et al., 2013). O VISIR possui em sua estrutura uma plataforma *PCI eXtensions for Instrumentation* (PXI), matriz de comutação de relés, servidor de medição e servidor de equipamento (TAWFIK et al., 2013).

Outro exemplo é o *Remote Labs Learning Environment* (RELLE)(CATARINA, 2023), laboratório remoto desenvolvido pelo Grupo de Trabalho em Experimentação Remota (GT-MRE) do Laboratório de Experimentação Remota (RExLab), na Universidade Federal de Santa Catarina (UFSC), que visa oferecer experimentos remotos de física, biologia e robótica, entre outros e disponibilizar um sistema de gerenciamento de usuários, com autenticação, fila de acesso ao experimento e emissão de relatórios, além de oferecer a possibilidade de desenvolver novos experimentos na plataforma a partir de especificações do sistema (SIMãO, 2016). A arquitetura do laboratório é composta por quatro módulos sistema físico, disponibilidade do experimento na rede, gerenciamento do usuá-

rios e interface de acesso ao laboratório. O sistema físico inclui o experimento, o vídeo monitoramento e a placa de aquisição e controle de dados (SIMãO, 2016).

2.2 Dispositivos lógicos programáveis

Programmable Logic Device (PLD) é um dispositivo eletrônico cujo *hardware* tem sua construção baseada em circuitos combinacionais que podem ser programados para implementar diferentes circuitos digitais, desta forma é possível realizar diferentes implementações de circuitos sem a necessidade de alterar o *hardware* físico, apenas realizando a reconfiguração (PEDRONI, 2010b). Segundo Floyd (2007) as categorias de lógica programável são *Programmable Logic Device* (PLD), que se divide em *Simple Programmable Logic Device* (SPLD) e *Complex Programmable Logic Device* (CPLD), e *Field Programmable Gate Array* (FPGA).

2.2.1 SPLD

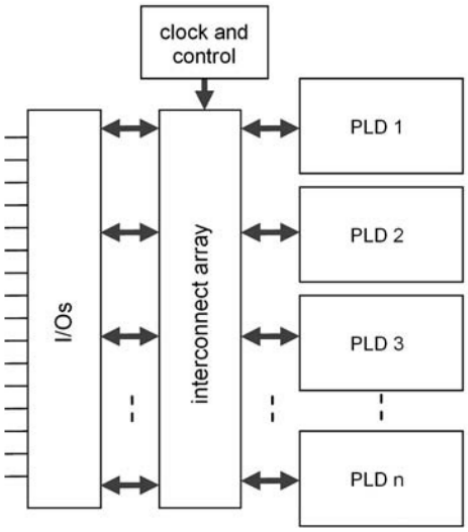
Por volta da década de 1970 os primeiros PLDs foram introduzidos no mercado, sendo chamados de *Programmable Logic Array* (PLA) e *Programmable Array Logic* (PAL), porém apresentava limitações devido a sua estrutura ter apenas portas tradicionais *AND* e *OR* sem a utilização de *flip-flops*, portanto era realizado a implementado penas circuitos combinacionais (PEDRONI, 2010a). A maior quantidade de *Simple Programmable Logic Device* (SPLD) são baseados em PLA e *Generic Array Logic* (GAL) (FLOYD, 2007).

Os *chips* do tipo PAL foram um avanço da tecnologia para superar limitações do PLA (COSTA, 2014). A arquitetura básica de um PLA é composto por um arranjo de portas *AND*, seguido por um arranjo de portas *OR* que são fixos (PEDRONI, 2010b). De acordo com Pedroni (2010a), considerando a arquitetura PLA foi adicionado macrocélula, *flip-flops*, porta *XOR* e multiplexadores que permitiam ser programados, além disso, foi adicionado arranjos *AND* programáveis que estão ligados a um arranjo *OR* fixo dentro das macrocélula.

2.2.2 CPLD

Lançado na década de 80, os *Complex Programmable Logic Device* (CPLD) são uma evolução dos SPLDs, este são *chips* programáveis e reprogramáveis, possuem alto desempenho, baixo custo e alta capacidade de integração (COSTA, 2014). Conforme apresentado na Figura 1 a arquitetura básica de um CPLD é composta por vários SPLDs em um mesmo *chip*, que são interligados por um barramento de interconexões complexas programáveis, além de conter circuitos de Entrada e Saída (I/O) mais modernos, muitos pinos I/O e suporte a *Joint Test Action Group* (JTAG) (PEDRONI, 2010b).

Figura 1 – Arquitetura básica CPLD

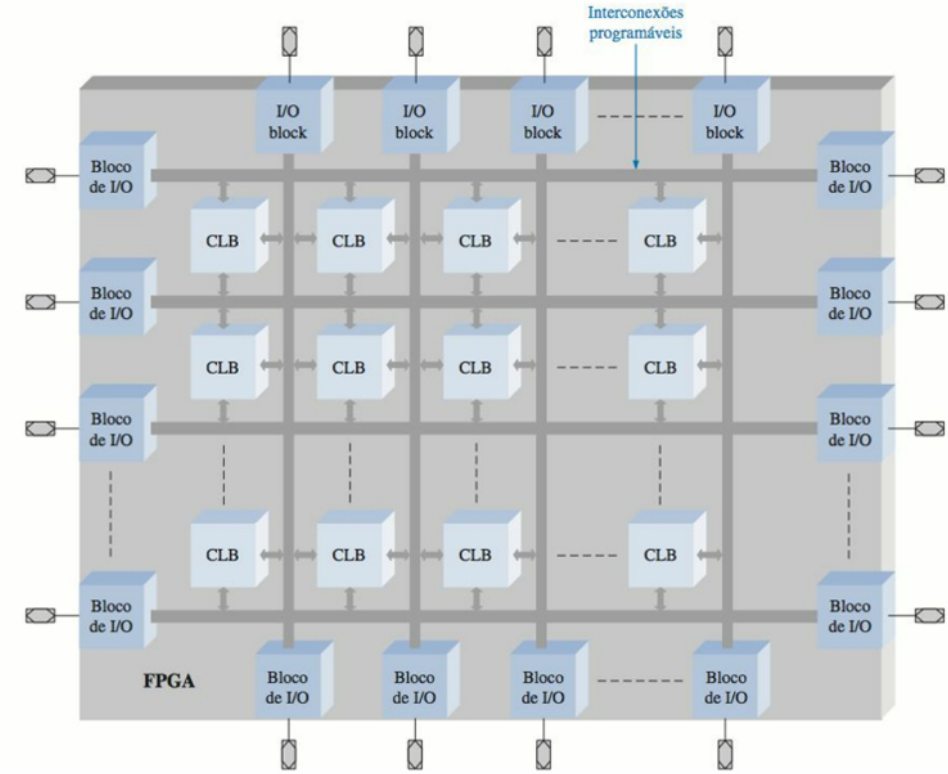


Fonte: (PEDRONI, 2010a)

2.2.3 FPGA

Também lançado na década de 80 *Field Programmable Gate Array* (FPGA) tem sua estrutura diferente dos SPLDs, deste modo é composta por três componentes básicos CLBs, interconexões programáveis e blocos de Entrada e Saída (I/O) (FLOYD, 2007).

Figura 2 – Arquitetura básica de um FPGA

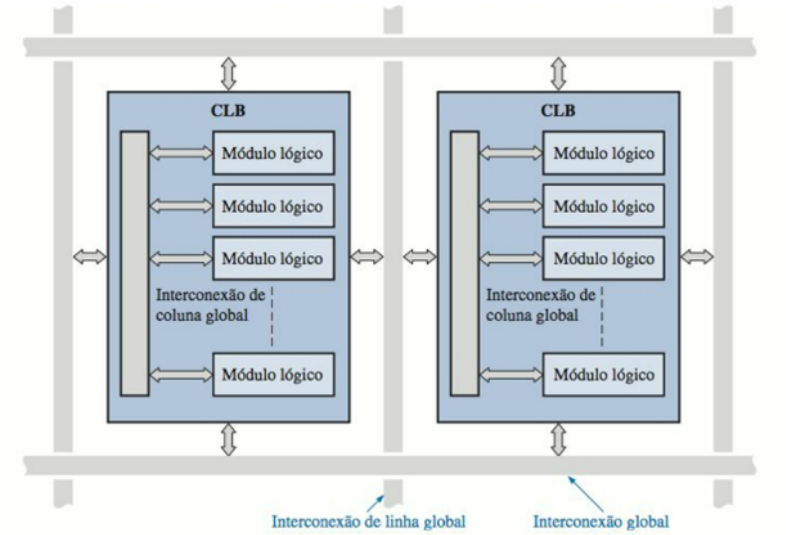


Fonte: (FLOYD, 2007)

Conforme ilustrado na Figura 2, os CLBs forma uma estrutura bidimensional, interligada por um barramento de interconexões programáveis, organizados como canais de roteamento vertical e horizontal, que interligam os CLBs e os blocos I/O (COSTA, 2014). Ainda segundo, Costa (2014) as interligações são realizadas conforme a necessidade do projeto.

Os CLBs que ficam entre as interconexões do FPGA é formado por um conjunto de módulos lógicos Figura 3, que podem configurados como logica combinacional ou registrada (FLOYD, 2007). Um típico módulo logico que os CLBs possuem é a *Look-Up Table* (LUT), a qual é um bloco de memória programável utilizado para gerar funções lógicas combinacionais de soma de produtos (FLOYD, 2007). Costa (2014) destacam que a LUT são voláteis, ou seja, perdem seu conteúdo armazenado quando o FPGA é desligada, porém, pode ser utilizados blocos de memória como *Flash Electrically Erasable Programmable Read-Only Memory* (FLASH EEPROM) para carregar automaticamente as células de armazenamento quando o dispositivo for realimentado.

Figura 3 – Arquitetura básica de um CLB



Fonte: (FLOYD, 2007)

É destacado por Pedroni (2010b), que os FPGAs incluem outros blocos como *Static Random Access Memory* (SRAM), *Digital Signal Processor* (DSP) e circuitos *Phase-Locked Loop* (PLL). Os FPGAs que utilizam SRAM, incluem uma memória não volátil *chip*, utilizado para armazenar a configuração do FPGA para quando for realimentado, os blocos DSP faz o processamento de dados em tempo real e tem sua aplicação em processamento de sinais digitais que representam sinais analógicos (FLOYD, 2007). Já o PLL é responsável por realizar o gerenciamento do *clock* do FPGA (PEDRONI, 2010b).

2.3 PWA

O termo *Progressive Web App* (PWA), foi criado por Alex Russell e Frances Berriman (RUSSELL, 2015), que descrevem a concepção de uma aplicação WEB que pode ser instalada no dispositivo do usuário através de um navegador. Essas aplicações oferecem a capacidade de operar *offline*, enviar notificações e acessar recursos do dispositivo móvel de forma segura,

Os PWAs fazem uso de um conjunto de estratégias e técnicas, como das *Application Programming Interfaces* (APIs) disponíveis nos navegadores para criar uma experiência semelhante à de aplicativos nativos (FORTUNATO; BERNARDINO, 2018). O termo *'progressive'* é utilizado para indicar que uma aplicação deve ser compatível com diferentes navegadores, adaptando-se aos recursos específicos de cada um, o que acarreta em diferentes experiências dos usuários (OSMANI, 2015).

Um recurso importante nos PWAs é o *service workers*, um conjunto de APIs que possibilita o armazenamento em cache, gerenciamento de notificações *push*, pré-carregamento, entre outras funcionalidades. De acordo com Malavolta et al. (2017) algumas vantagens dessas aplicações em comparação com as nativas são:

- Disponibilidade independente da conexão: A utilização dos *service workers* permite que o aplicativo seja executado mesmo sem conexão com a internet, podendo as funcionalidades *online*, *offline* e com conexões instáveis;
- Tempo de carregamento rápido: a aplicação carregar quase que instantaneamente, independente da rede utilizada, isso é possível devido ao uso de *service workers* que armazenam cache;
- Notificação *push*: é possível enviar notificações para o usuário mesmo quando o aplicativo não está em execução, que ajuda no engajamento do usuário;
- Atalho na tela inicial: é possível fixar um atalho na tela inicial do dispositivo, o que é sugerido pelo próprio navegador a partir do interesse do usuário;
- Aparência nativa: os PWA podem ter sua aparência de modo semelhante a um aplicativo nativo, sem que seja apresenta a interface do navegador.

O desenvolvimento de uma aplicação PWA envolve a utilização de tecnologias, como *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript* (FORTUNATO; BERNARDINO, 2018). Para facilitar e acelerar o processo de desenvolvimento, são empregados *frameworks* como *Angular* (ANGULAR, 2023), *Vue* (VUE.JS, 2023) e *Ionic* (IONIC, 2023), ou bibliotecas como *React* (REACT, 2023).

2.3.1 *Service worker*

Os *service workers* são *scripts* JavaScript que atuam como intermediários *proxy* entre o navegador e o servidor, para aprimorar a confiabilidade ao proporcionar funcionalidades *offline* e otimizar o desempenho das páginas, conforme Gaunt (2017) e Google (2021) . Segundo Hume (2017), a eficiência dos PWAs está ligada aos *service workers*, os quais operam em segundo plano nos navegadores, sem a necessidade de interação com o usuário, o que viabiliza a execução de tarefas como notificações, sincronização em segundo plano e acesso *offline* a aplicação.

Gaunt (2017) destaca características importantes do *service workers* como, JavaScript *worker* não possui acesso direto ao *Document Object Model* (DOM)[1], é encerrado quando não está em uso e reiniciado quando necessário, faz o uso de *Promises*[2] e por fim, um *proxy* de rede programável, o que permite controlar as requisições de rede.

Para que uma aplicação WEB faça uso do *service workers*, deve ser servida por meio de uma conexão segura, ou seja, *Hypertext Transfer Protocol Secure* (HTTPS), e que o navegador ofereça suporte a *service workers* (GAUNT, 2017). O *service worker* atua como um controlador de cada requisição feita à aplicação, permitindo redirecionamentos, atrasos ou até mesmo interrupções nas solicitações (HUME, 2017). No entanto, ainda segundo Hume (2017), o registro de um *service worker* suspeito na rede pode resultar no redirecionamento das requisições para um servidor malicioso, representando um risco de segurança. Por esse motivo, o *service worker* deve ser servido por uma conexão segura HTTPS (GAUNT, 2017).

2.4 *Websockets*

O *websockets* é um protocolo de comunicação assíncrono que permite a troca de mensagens de texto e dados binários entre um cliente e um servidor, este protocolo opera em um canal de comunicação bidirecional *full-duplex*, permitindo eventos sem a necessidade de solicitações prévias (GRIGORIK, 2013).Em uma API *websocket*, tanto o cliente quanto o servidor podem enviar e receber mensagens a qualquer momento, sem a necessidade de uma solicitação prévia (LOMBARDI, 2015). Grigorik (2013) destaca que este tipo de informação pode ir de cargas mais simples como *JavaScript Object Notation* (JSON) até formatos de mensagens binárias mais complexas.

A conexão *websocket* é iniciada por meio de uma *Uniform Resource Locator* (URL), que atua como um *endpoint* ao qual o cliente se conecta (WANG; SALIM; MOSKOVITS, 2013). Essa conexão pode ser estabelecida por meio de uma *Uniform Resource Identifier*

[1] DOM é a representação de dados dos objetos que compõem a estrutura e o conteúdo de um documento na WEB (MOZILLA, 2023a)

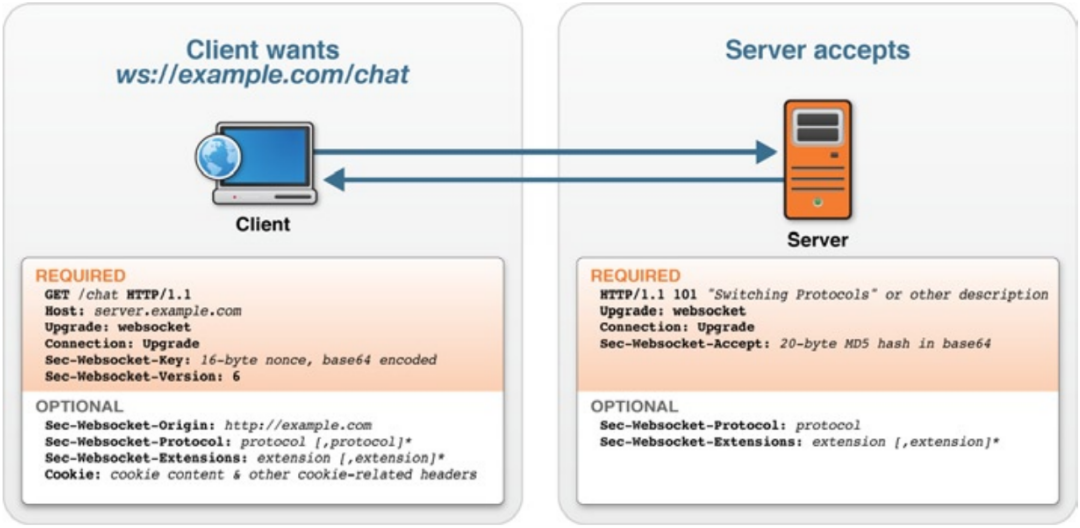
[2] *Promises* é um objeto que representa a eventual conclusão ou falha de uma operação assíncrona (MOZILLA, 2023c)

(URI), *Websocket* (WS) para conexões não criptografadas, e *Websocket Secure* (WSS) para aquelas que requerem criptografia (WANG; SALIM; MOSKOVITS, 2013).

Uma conexão *websocket* opera com base em eventos, disparando quatro tipos: *Open*, *Message*, *Error*, e *Close*, como definido por WHATWG (2023). Já os métodos como *send* e *close* são utilizados para enviar mensagens do cliente para o servidor e encerrar a conexão (LOMBARDI, 2015). Enquanto as atribuições *readyState*, *bufferedAmount* e *protocol* representam o estado da conexão, a quantidade de bytes enfileirados e o protocolo compreendido pelo cliente, respectivamente (WANG; SALIM; MOSKOVITS, 2013).

A comunicação entre cliente e servidor em um protocolo *websocket* é estabelecida por meio de um *handshake*. A Figura 4 ilustra o processo em que o cliente, por meio de uma requisição HTTP, solicita uma conexão com o servidor, indicando no cabeçalho *Upgrade* sua intenção de alterar o protocolo de comunicação para *websocket*. Se o servidor estiver disponível, a solicitação é aceita, sendo estabelecido assim um canal de comunicação *websocket* (WANG; SALIM; MOSKOVITS, 2013).

Figura 4 – Exemplo de negociação de uma abertura de *WebSocket*



Fonte: (WANG; SALIM; MOSKOVITS, 2013)

2.5 WebRTC

Web Real-Time Communications (WebRTC) é composto por um conjunto de padrões e protocolos que permitem a comunicação em tempo real entre navegadores, compartilhando áudio, vídeo e dados por meio de uma APIs JavaScript, deste modo o WebRTC transforma esse tipo de comunicação um padrão para os navegadores, não havendo necessidade de instalação de *plugins* ou *softwares* adicionais (GRIGORIK, 2013).

Uma aplicação que usa WebRTC interage com um navegador por meio de uma API WebRTC padronizada, que permite que seja acessado as funções do navegador em

Real-Time Communications (RTC), sem intermediário. Dentre estas funções estão gerenciamento de conexão, negociação da capacidade de codificação e decodificação, controle de mídia, travessia de *Network Address Translation* (NAT) e *firewall*, entre outros (LORETO; ROMANO, 2014).

Inicialmente, a implementação original do WebRTC tinha como objetivo oferecer uma comunicação *Peer-to-Peer* (P2P), no entanto, também suporta outras arquiteturas, como *Multipoint Control Unit* (MCU) e *Selective Forwarding Unit* (SFU) (IYENGAR, 2021).

Em uma comunicação P2P, todos os participantes estão diretamente conectados, e compartilham mídia entre si. Na MCU, os participantes se conectam a um servidor central responsável por juntar as mídias de todos os participantes e distribuí-las para os demais em um único fluxo. Por fim, o SFU, onde os participantes se conectam a um servidor central e enviam a sua mídia, e o servidor então se encarrega de distribuir a mídia para os demais participantes. Um servidor SFU atua como um transmissor, que recebe fluxos de mídia e os distribui, o que o leva a ter carga minima de processamento (IYENGAR, 2021).

Para se estabelecer a ligação entre os participantes, é necessário contornar problemas que bloqueariam a conexão, como *firewalls*; dispositivo não possuir *Internet Protocol* (IP) público; e roteador não permitir a conexão entre os demais participantes. Para isso, o WebRTC utiliza o protocolo *Interactive Connectivity Establishment* (ICE) com *Session Traversal Utilities for NAT* (STUN) e/ou *Traversal Using Relays around NAT* (TURN) (MOZILLA, 2023b).

2.6 Banco de dados

Um Sistema de Gerenciamento de Banco de Dados (SGBD) conforme descrito por Silberschatz, Korth e Sudarshan (2016), é uma coleção de dados que são inter-relacionados em um conjunto de sistemas que permitem o acesso e modificação desses dados. A coleção de dados pode ser entendida como o banco de dados projetados para gerenciar uma grande quantidade de informações, além disso o banco de dados de deve garantir a integridade e segurança dos dados, isso mediante a falhas no sistema ou acesso indevidos (SILBERSCHATZ; KORTH; SUDARSHAN, 2016). Os SGBD podem ser do tipo relacional ou não relacional, *Structured Query Language* (SQL) e *Not Only SQL* (NoSQL), respectivamente.

O Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR), é projetado com um esquema predefinido e fixo para organizar os dados em tabelas que podem ser escalados verticalmente Soni e Jyotinagar (2023), Heuser (2009) destaca além das tabelas que formam o banco de dados, cada tabela deve conter colunas que representam atributos e restrição de integridade. Os banco de dados SQL são projetados para serem utilizados em

aplicações que necessitam de recursos de consultas complexas, e transações que envolvem Atomicidade, Consistência, Isolamento e Durabilidade (ACID). Ainda segundo Soni e Jyotinagar (2023), o banco de dados SQL tem como limitações a escalabilidade, pois lidar com grande volume de dados pode resultar em ter um sistema integrados de processamento e distribuição dos dados, e flexibilidade, onde a estrutura de dados é rígida e não permite alterações.

Banco de dados NoSQL pode fazer o processamento de muitos dados em variados formatos, mas para que isso ser possível é necessário ser mais flexível e deixar algumas catacrésticas do banco de dados relacional SQL de lado (SILVA et al., 2021). Soni e Jyotinagar (2023) afirma que o banco de dados NoSQL podem tratar de dados não estruturados e proporcionando maior escalabilidade, no entanto, pode acarretar e problemas de consistência transacional, quando comparado com o banco de dados SQL. As limitações do banco de dados não relacional NoSQL segundo Soni e Jyotinagar (2023) são a consistência dos dados, pois não oferece a garantia da ACID, dados heterogêneos, porque não possui um esquema de dados predefinido, segurança por serem abertos e escaláveis, portanto mais propícios a ataques, por fim *backup* e recuperação de dados, modelagem do banco e ferramentas de gerenciamento, todo estes que diferem do banco de dados relacional, deste modo maior desafio para os desenvolvedores acostumados com o banco de dados relacional.

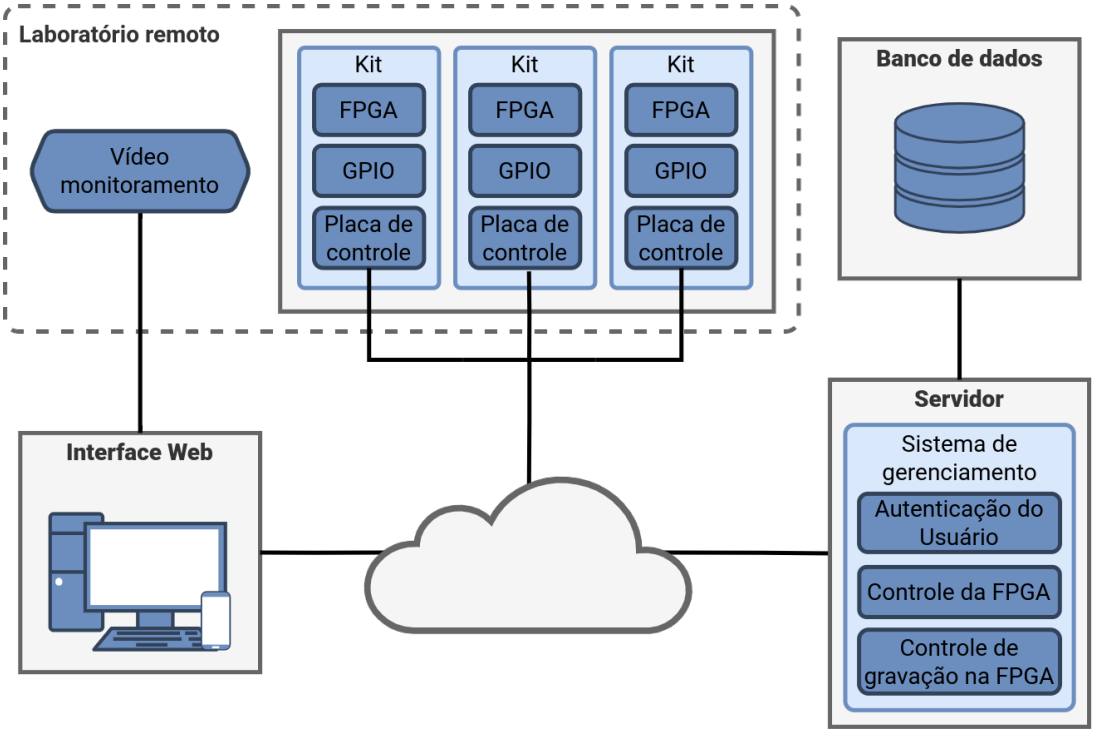
De acordo com Silva et al. (2021) a escolha de uma SGBD para um projeto é uma tarefa complexa e não deve ser deixada de lado, para isso deve ser considerado o tipo de dados que serão armazenados, interface disponível para usuário e desenvolvimento e a capacidade de se conectar com outros SGBD.

3 PROPOSTA

O eLAB tem como objetivo a criação de um laboratório remoto para desenvolvimento em FPGA. Este trabalho se propõe a desenvolver uma plataforma WEB para interação com o laboratório remoto, permitindo o acesso aos *kits* de FPGA por meio de uma aplicação PWA disponível para computadores e dispositivos móveis. A plataforma permitirá a interação com chaves, botões e LEDs dos *kits* de FPGA através da placa de controle. Além disso, oferecerá gerenciamento de acesso aos *kits* por agendamento de horário e monitoramento por vídeo.

O laboratório, conforme apresentado na Figura 5, é composto por *kits* e cada um inclui uma FPGA, uma placa de controle conectada à FPGA por meio dos pinos de entrada e saída denominados GPIO, e um sistema de monitoramento de vídeo. Cada *kit* está diretamente conectado ao servidor, o qual possui um sistema de gerenciamento responsável pela autenticação do usuário, controle da FPGA, gestão da gravação na placa de FPGA, e monitoramento de vídeo dos *kits* de FPGA. Além disso, o servidor inclui um banco de dados que armazena as informações dos usuários e os dados de acesso ao laboratório. A aplicação PWA, por meio da interface *web*, interage com o servidor e recebe o monitoramento de vídeo, permitindo que o usuário interaja com o *kit* de FPGA e observe o resultado em tempo real.

Figura 5 – Diagrama de blocos do laboratório remoto



Fonte: Elaborado pelo autor

Conforme o propósito deste trabalho, o projeto foi dividido em seis etapas, sendo a primeira consiste em realizar o levantamento dos requisitos funcionais do sistema e descrever o conjunto de interações entre a aplicação e servidor já existente no projeto, como resultado deve se obter um diagrama de caso de uso conforme a padronização *Unified Modeling Language* (UML) e o desenvolvimento de uma API REST detalhando os processos para acessar os recursos do servidor, desta forma havendo uma padronização para troca de informação com a aplicação PWA.

A segunda etapa, a modelagem da aplicação, será realizada com base na prioridade dos requisitos para garantir o seu pleno funcionamento, seguido da implementação da aplicação PWA utilizando a biblioteca React *front-end*. Nesta mesma etapa será realizada a implementação do sistema de gerenciamento para agendamento de horários com base no envio de comando via *websocket* ao servidor para interação com o *kit* de FPGA, cada *kit* poderá ser acessível por vários usuários, mas apenas um usuário que realizou agendamento do horário ou acessou um *kit* que estava disponível tem prioridade no controle. Deste modo o usuário deve ter total gestão do seu *kit* FPGA, podendo enviar comandos de controle e garantindo prioridade sobre os demais usuário que também estão acessando o mesmo *kit*, para este processo será utilizado a biblioteca *socket.io*. Deste modo, será necessário realizar um levantamento da estrutura atual do banco de dado do eLAB, para verificar se será possível realizar modificações ou se será necessário criar um novo banco de dados para armazenar os horários de agendamento, *status* e qual dos *kit* de FPGA que foi agendado e o usuário que realizou o agendamento.

Na terceira etapa, será realizada a implementação do sistema de monitoramento remoto por vídeo, para isso será utilizado a API WebRTC que realiza a comunicação entre cliente e servidor. Para tal será utilizado um SFU que receberá o fluxo de mídia de vídeo do servidor de controle e fará a sua distribuição para todos os usuários que se conectarem a sala SFU. Os usuários devem ser configurados apenas para receber fluxo de mídia.

Na quarta etapa será realizada a integração do sistema de monitoramento com a aplicação PWA. Na quinta etapa será feito a integração e testes dos *Kits* de FPGA com a interface de acesso ao laboratório, a partir do resultado testes e pode ser necessário realizar modificações no *software* da placa de controle do *hardware*. Ao fim de cada uma das etapas será realizado a documentação dos resultados obtidos durante o processo e finalização da escrita do TCC.

3.1 Cronograma de atividades

Quadro 1 – Cronograma de atividades

Atividades	Mês					
	Fev	Mar	Abr	Mai	Jun	Jul
A1	√					
A2	√	√				
A3		√	√			
A4			√	√		
A5				√	√	
A6	√	√	√	√	√	√

Fonte: Elaborado pelo autor

Conforme o Quadro 1, o cronograma de atividades foi dividido em seis etapas. A seguir, são descritas as atividades de cada etapa:

- A1: levantar os requisitos funcionais do banco de dados existente. Desenvolver um diagrama de caso de uso e implementar uma API REST com base nos requisitos identificados.
- A2: projetar e implementar uma aplicação *Progressive Web App* (PWA) e um sistema de agendamento de horários para acesso aos *kits* de FPGA.
- A3: implementar o sistema de monitoramento remoto do laboratório.
- A4: integrar monitoramento remoto com a aplicação PWA.
- A5: integrar e testar acionamento das chaves, botões e LEDs dos *kits* de FPGA com a interface de acesso ao laboratório. Realizar possíveis modificações no *software* da placa de controle do *hardware*.
- A6: documentar os resultados obtidos durante o processo e finalizar a escrita do TCC.

REFERÊNCIAS

ANGULAR. *Deliver web appswith confidence*. 2023. Disponível em: https://angular.io/. 16

BASCIL, M. S.; YAZICI, I.; TEMURTAS, F. A fpga based remote accessible digital system laboratory prototype. *2012 International Conference on E-Learning and E-Technologies in Education, ICEEE 2012*, p. 49–53, 2012. 9

CATARINA, U. F. de S. *RELLE - Ambiente de Aprendizagem com Experimentos Remotos*. 2023. Disponível em: http://relle.ufsc.br/labs. 9, 12

CHU, P. P. *FPGA Prototyping by VHDL Examples*. [S.l.]: Wiley, 2008. 9

COSTA, C. *Projetos de Circuitos Digitais com FPGA*. [S.l.: s.n.], 2014. ISBN 978-85-365-0585-5. 13, 15

FLOYD, T. *Sistemas Digitais: Fundamentos e Aplicações*. 9. ed. [S.l.]: Bookman, 2007. ISBN 9788577801077. 13, 14, 15

FORTUNATO, D.; BERNARDINO, J. Progressive web apps: An alternative to the native mobile apps. p. 1–6, 2018. 16

GARCIA-LORO, F. et al. Pilar: Sharing visir remote labs through a federation. *IEEE Global Engineering Education Conference, EDUCON*, IEEE Computer Society, April-2019, p. 102–106, 4 2019. ISSN 21659567. 9

GAUNT, M. Service workers: an introduction. *Hacker Bits*, v. 12, 2017. 17

GOOGLE. *Service worker overview*. 2021. Disponível em: https://developer.chrome.com/docs/workbox/service-worker-overview?hl=en. 17

GRIGORIK, I. *Browser APIs and Protocols: WebSocket - High Performance Browser Networking (O'Reilly)*. O'Reilly Media, 2013. Disponível em: https://hpbn.co/websocket/. 17, 18

HEUSER, C. A. *Projeto de banco de dados : Volume 4 da Série Livros didáticos informática UFRGS*. [S.l.]: Bookman, 2009. ISBN 9788577804528. 19

HUME, D. A. *Progressive Web Apps*. [S.l.]: Manning Publications Co., 2017. 200 p. ISBN 9781617294587. 17

IFSC. *Laboratório Remoto VISIR+*. 2023. Disponível em: https://visir.florianopolis.ifsc.edu.br/visir/index.php/pt. 12

INTEL. *Remote Access to FPGA Lab*. 2023. Disponível em: https://www.intel.com/content/www/us/en/developer/videos/remote-access-to-fpga-lab.html. 9

IONIC. *The Cross-Platform App Development Leader*. 2023. Disponível em: https://ionicframework.com/. 16

IYENGAR, M. *WebRTC Architecture Basics: P2P, SFU, MCU, and Hybrid Approaches*. Medium, 2021. Disponível em: https://medium.com/securemeeting/webrtc-architecture-basics-p2p-sfu-mcu-and-hybrid-approaches-6e7d77a46a66. 19

LABSLAND. *Experimentar de verdade*. 2023. Disponível em: https://labsland.com/pt_BR/about. 12

LOMBARDI, A. *WebSocket: Lightweight Client-Server Communications*. O'Reilly, 2015. ISBN 9781449369279. Disponível em: https://books.google.com.br/books?id=QsTqjgEACAAJ. 17, 18

LORETO, S.; ROMANO, S. P. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*. [S.l.]: O'Reilly Media, 2014. ISBN 9781449371838. 19

MALAVOLTA, I. et al. Assessing the impact of service workers on the energy efficiency of progressive web apps. p. 35–45, 2017. 16

MAYOZ, C. A. et al. Fpga remote laboratory: Experience of a shared laboratory between upna and unifesp. *Proceedings - 2020 14th Technologies Applied to Electronics Teaching Conference, TAEE 2020*, Institute of Electrical and Electronics Engineers Inc., 7 2020. 9, 11, 12

MOZILLA. *Introduction to the DOM*. 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. 17

MOZILLA. *Introduction to WebRTC protocols*. 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols. 19

MOZILLA. *Using promises*. 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises. 17

ORDUñA, P. et al. Labsland: A sharing economy platform to promote educational remote laboratories maintainability, sustainability and adoption. p. 1–6, 2016. 11

OSMANI, A. *Getting started with Progressive Web Apps*. 2015. Disponível em: https://addyosmani.com/blog/getting-started-with-progressive-web-apps/. 16

PEDRONI, V. A. *Circuit Design and Simulation with VHDL*. 2nd ed.. ed. [S.l.]: MIT Press, 2010. ISBN 9788120343016. 13, 14

PEDRONI, V. A. *Eletrônica Digital Moderna e VHDL*. [S.l.]: Elsevier, 2010. ISBN 9788535234657. 13, 15

POST, L. S. et al. Effects of remote labs on cognitive, behavioral, and affective learning outcomes in higher education. *Computers & Education*, Pergamon, v. 140, p. 103596, 10 2019. ISSN 0360-1315. 9, 11

PRETZ, K. *Remote Learning Made Easier With This Startup's Online Engineering Labs - IEEE Spectrum*. 2020. Disponível em: https://spectrum.ieee.org/remote-learning-made-easier-with-this-startups-online-engineering-labs#toggle-gdpr. 9

REACT. *The library for web and native user interfaces*. 2023. Disponível em: https://react.dev/. 16

RUSSELL, A. *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. 2015. Disponível em: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/. 16

SIERRA-FERNáNDEZ, J. M. et al. Reconfigurable web-interface remote lab for instrumentation and electronic learning. p. 713–717, 2020. 11, 12

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de banco de dados*. [S.l.]: Elsevier Brasil, 2016. 19

SILVA, L. F. C. et al. *Banco de Dados Não Relacional*. [S.l.]: Sagah, 2021. ISBN 9786556901534. 20

SIMãO, J. P. S. Relle: Sistema de gerenciamento de experimentos remotos. 2016. Disponível em: https://repositorio.ufsc.br/xmlui/handle/123456789/157371. 12, 13

SONI, M.; JYOTINAGAR, V. Sql vs nosql databases for the microservices: A comparative survey. p. 17–22, 2023. 19, 20

TAWFIK, M. et al. Virtual instrument systems in reality (visir) for remote wiring and measurement of electronic circuits on breadboard. *IEEE Transactions on Learning Technologies*, v. 6, n. 1, p. 60–72, 2013. 12

VUE.JS. *The Progressive JavaScript Framework*. 2023. Disponível em: https://vuejs.org/. 16

WANG, V.; SALIM, F.; MOSKOVITS, P. *The Definitive Guide to HTML5 WebSocket*. Apress, 2013. (Books for professionals by professionals). ISBN 9781430247418. Disponível em: https://books.google.com.br/books?id=YIgdrqKZzYMC. 17, 18

WHATWG. *WebSockets*. 2023. Disponível em: https://websockets.spec.whatwg.org/. 18