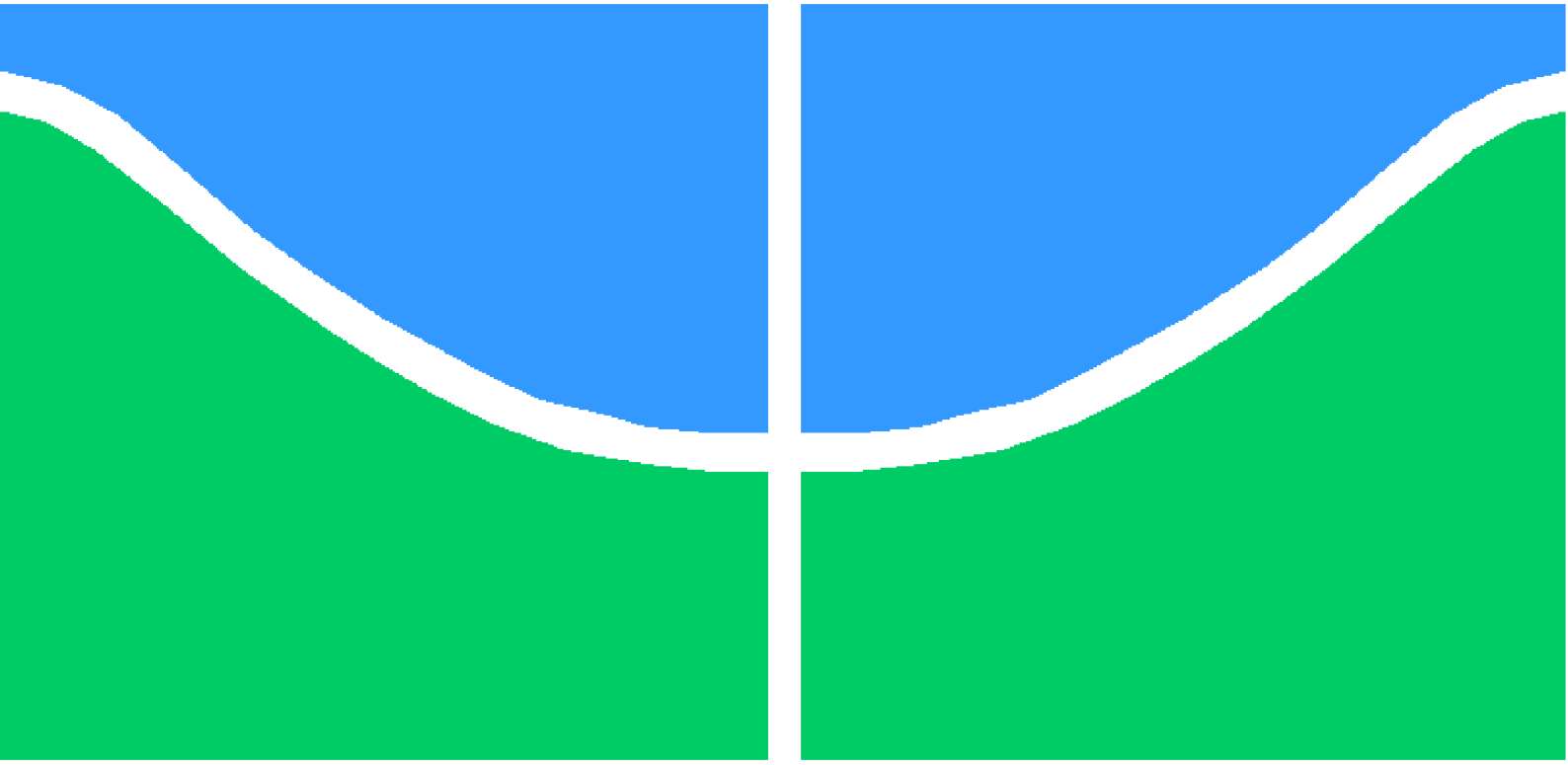


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados

Autor: André Lucas Ferreira Lemos de Souza

Orientador: Prof. Dr. Renato Coral Sampaio

Brasília, DF

2023

André Lucas Ferreira Lemos de Souza

Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Renato Coral Sampaio

Brasília, DF

2023

André Lucas Ferreira Lemos de Souza

Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Brasília, DF
2023

Agradecimentos

Agradeço aos professores do curso de Engenharia de Software, pelo empelho durante toda a nossa formação e por apresentarem projetos e tecnologias que nos tiram sempre da zona de conforto, preparando-nos para o mercado de trabalho. A meu pai, meu irmão e amigos por sempre estarem ao meu lado e nunca me deixarem desistir. E um agradecimento mais que especial a minha mãe Érica Iara, que não está mais entre nós fisicamente para desfrutar desse momento, mas que sempre esteve ao meu lado e foi meu alicerce para chegar até aqui.

"É no problema da educação que assenta o grande segredo do aperfeiçoamento da humanidade."

(Immanuel Kant)

Resumo

O ensino prático é fundamental para que os alunos concluam melhor o curso e estejam preparados para o mercado de trabalho. Na área de Sistemas Embarcados, a realização de práticas em laboratório é imprescindível para a formação de novos profissionais. O sugimento da portaria nº 2.117 do Ministério da Educação que oferta parte da carga horária de cursos de graduação presenciais na modalidade de ensino a distância e a pandemia causada pelo COVID-19, destacaram ainda mais a necessidade de uma alternativa aos laboratórios físicos. O objetivo da plataforma que será desenvolvida, com o auxílio de *softwares* abertos, é permitir que os estudantes realizem remotamente os experimentos práticos de Sistemas Embarcados por meio de um laboratório remoto. Os aspectos metodológicos utilizados constituem uma pesquisa descritiva, dividida nas fases de planejamento, coleta de dados, análise de dados e relato dos resultados. Espera-se que este projeto ajude o processo de aprendizagem dos alunos de graduação e pós-graduação, viabilizando a realização de experimentos por meio remoto e o cumprimento dos atingir os objetivos propostos para os projetos.

Palavras-chave: Sistemas Embarcados, COVID-19, Laboratório Remoto, Práticas em Laboratório.

Abstract

Practical teaching is essential for students to better complete the course and be prepared for the job market. In the area of Embedded Systems, carrying out laboratory practices is essential for the training of new professionals. With the introduction of Ordinance No. 2,117 of the Ministry of Education, which offers part of the workload of on-site undergraduate courses in the distance learning modality and the pandemic caused by COVID-19, they further highlighted the need for an alternative to physical laboratories. The objective of the platform that will be developed, with the help of open software, is to allow students to remotely carry out practical experiments on Embedded Systems through a remote laboratory. The methodological aspects used constitute a descriptive research. It is expected that this project will help the learning process of undergraduate and graduate students, where they can carry out experiments and achieve the proposed objectives for the projects.

Key-words: Embedded Systems, COVID-19, Remote Laboratory, Laboratory Practices.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UnB	Universidade de Brasília
COVID-19	Doença do Coronavírus, enquanto “19” se refere a 2019
SSH	Cápsula de Segurança
VPN	Rede privada virtual
TV	Televisores
VOD	Vídeo sob demanda
MEC	Ministério da Educação
WebRTC	Comunicações Web em Tempo Real
HTTP	Protocolo de Transferência de Hipertexto

Sumário

1 Introdução

Classificada como multidisciplinar, a área de Sistemas Embarcados inclui conhecimentos de eletrônica, algoritmos, sistemas operacionais, processamento de sinais e teoria de controle. O uso de laboratórios no ensino de sistemas embarcados é fundamental para o processo de ensino e aprendizagem dos alunos. Por meio de vivências nesses ambientes, os alunos têm a oportunidade de consolidar seus conhecimentos teóricos para resolução de problemas práticos, auxiliando na formação da sua carreira.

Em 2019, o Ministério da Educação (MEC) publicou a portaria Nº 2.117, de 6 de dezembro de 2019, que "... dispõe sobre a oferta de carga horária na modalidade de Ensino a Distância - EaD em cursos de graduação presenciais ofertados por Instituições de Educação Superior - IES pertencentes ao Sistema Federal de Ensino, com observância da legislação educacional em vigor" (UNIAO, 2019). Assim, as IES poderão ofertar até 40% da carga horária na modalidade de EAD na organização pedagógica e curricular de seus cursos de graduação presenciais.

Devido à pandemia do COVID-19, escolas e universidades enfrentaram problemas em como prosseguir com a aprendizagem, mantendo professores e alunos a salvo de uma emergência de saúde pública. A maioria das instituições optaram por suspender todas as aulas, incluindo laboratórios, ofertando seus cursos de forma remota para impedir a maior propagação do vírus que causa o COVID-19 (HODGES et al., 2020).

Um dos modelos que contribuem para o ensino a distância, especialmente no ensino de engenharia, porém com dificuldade de implementação são os experimentos por laboratório remoto, que permitem que os alunos executem o trabalho prático necessário de maneira semelhante à realizada nos laboratórios físicos, se for corretamente projetado (MOHAMMED; ZOGHBY; ELMESALAWY, 2020). A ideia principal dos laboratórios remotos é que as universidades de todo o mundo possam instalar e hospedar experimentos físicos, que alunos acessem por meio da internet (AKTAN et al., 1996).

O estudo de diferentes tecnologias para laboratórios de controle remoto é um dos temas que tem atraído muitos pesquisadores e empresas nos últimos anos. Em virtude da pandemia de COVID-19, e com as possibilidades a partir da portaria do MEC, essa pesquisa foi intensificada. De fato, a prática de experimentos é considerada uma das ausências mais críticas no sucesso dos processos de ensino a distância em escolas e universidades. Portanto, é necessário focar no desenvolvimento de laboratórios de controle remoto eficientes e confiáveis que possam ser adaptados à parte prática em todos os cursos e programas educacionais. A experimentação remota decorre de uma motivação diferente: ela permite que os professores demonstrem melhor os conceitos em uma sala de aula

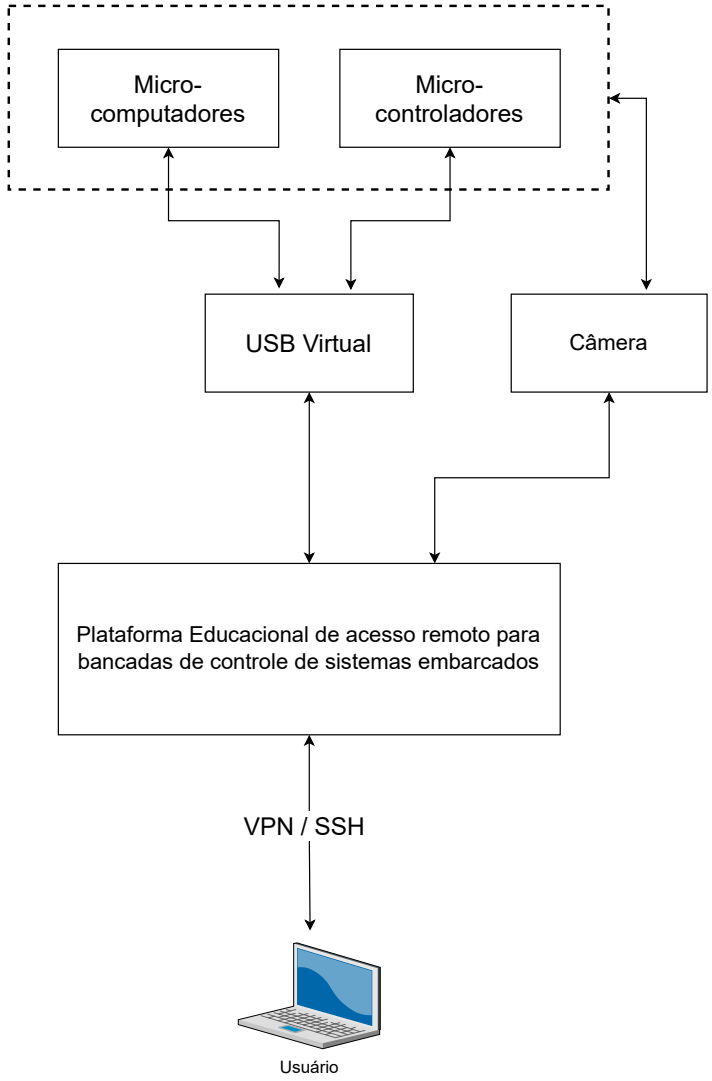
tradicional simplesmente conectando-se a um laboratório remoto e executando um experimento. Ele também pode fornecer uma solução econômica para cursos de engenharia a distância (MOHAMMED; ZOGHBY; ELMESALAWY, 2020).

Laboratórios remotos são baseados em laboratórios físicos que são controlados remotamente por meio de uma plataforma. Na maioria das vezes, são equipamentos caros e sensíveis, limitando a quantidade de experimentos. Esses laboratórios oferecem uma oportunidade dos estudantes realizarem exercícios fora do laboratório físico em horários apropriados, a princípio através de agendamento (KARAKASIDIS, 2013).

Deste modo, observamos que no cenário atual a concretização tecnológica segue em paralelo ao desenvolvimento educacional. Assim, uma plataforma para acesso remoto a laboratórios pode ser a solução para estes problemas em que os alunos enfrentam, podendo coletar dados reais do campo de experimentos que são necessários para que eles se envolvam nos experimentos como na realidade.

A ideia inicial deste sistema de laboratório remoto é oferecer aos usuários a capacidade de se conectar ao laboratório através de uma conexão segura por SSH. A plataforma intuitiva permitirá acessar experimentos de forma fácil e eficiente, fornecendo ao usuário uma USB virtual para interagir com controladores remotamente e uma câmera para fornecer *feedback* em tempo real. Dessa forma, os usuários poderão conduzir seus experimentos com total autonomia e sem se preocupar com a distância física dos equipamentos. A ideia descrita acima está ilustrada na Figura 1.

Figura 1 – Diagrama do Laboratório



Fonte: Autor

1.1 Problema

Nesse cenário de pandemia do COVID-19, que infectou milhões de pessoas pelo mundo, e com a portaria do MEC nº 2.117, que dispõe de 40% da carga horária ofertada na modalidade de ensino a distância nos cursos presenciais, atividades educacionais e laboratoriais também sofreram mudanças drásticas.

Importante mudanças foram impostas principalmente relacionadas à estrutura e ao funcionamento do ensino que envolvem alunos e professores, criando soluções para que todos trabalhassem em casa. Dessa forma, tornou-se ainda mais difícil a realização de atividades laboratoriais e experimentais, evidenciando assim a necessidade de uma estrutura de ensino para laboratórios práticos de forma remota.

A partir das ausências encontradas, a pergunta de pesquisa determinada neste trabalho é:

Como implementar uma plataforma educacional de acesso remoto para bancadas de sistemas embarcados, com o auxílio de ferramentas opensource integradas, para possibilitar que alunos se conectem aos laboratórios reais?

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho é especificar e implementar uma plataforma educacional de acesso remoto para bancadas de sistemas embarcados.

1.2.2 Objetivos Específicos

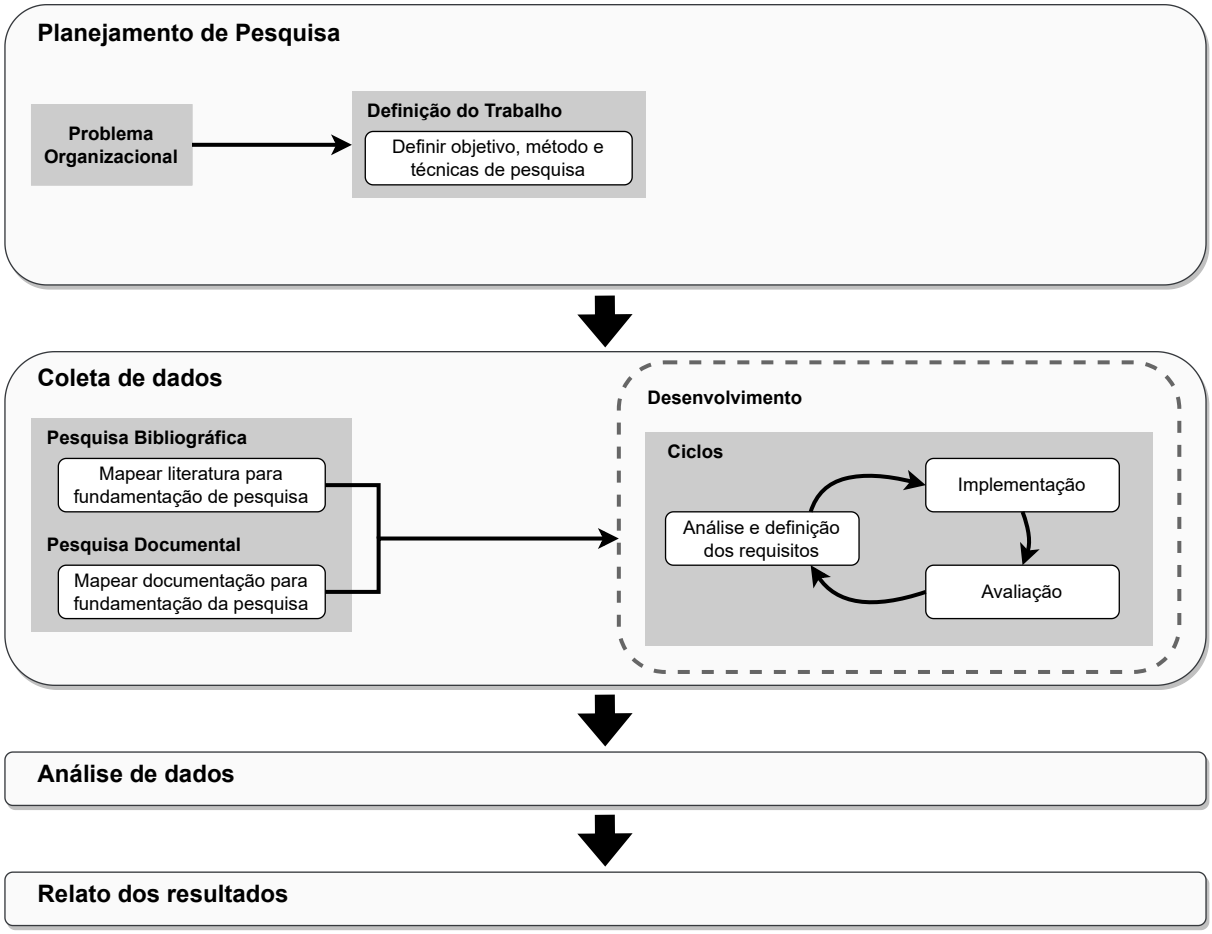
- Investigar tecnologias adequadas ao desenvolvimento de uma plataforma educacional de acesso remoto;
- Elicitar, modelar e analisar requisitos para uma plataforma educacional de acesso remoto para bancada de sistemas embarcados;
- Iniciar o desenvolvimento da plataforma educacional de acesso remoto para bancadas de sistemas embarcados.

1.3 Metodologia

O plano metodológico deste trabalho é composto por quatro fases: Planejamento de pesquisa, Coleta de dados, Análise de dados, Relato dos resultados, como apresentado na Figura 2.

A metodologia deste trabalho foi definida e classificada de acordo com o seu objetivo. Essa é uma pesquisa de natureza Aplicada, do tipo Descritiva. Para a Coleta dos dados são empregadas as técnicas pesquisa bibliográfica e pesquisa documental, seguida de um "processo de desenvolvimento" da plataforma.

Figura 2 – Plano metodológico



Fonte: Autor

1.3.1 Procedimentos Metodológicos

Conforme o plano metodológico, a fase de Coleta de Dados compreende o emprego das técnicas Revisão bibliográfica e Documental. Em seguida, um processo para o desenvolvimento da Plataforma:

1. **Pesquisa bibliográfica:** Levantamento criterioso das fontes a partir do tema, para aprimorar a visão do contexto e melhor adequar o escopo, para assim definir uma base teórica concreta para apoiar os objetivos do trabalho.
2. **Pesquisa documental:** Estudo de ferramentas e tecnologias que auxiliarão na execução do trabalho, serão abordadas no referencial e no desenvolvimento do trabalho.
3. **Processo de Desenvolvimento:** Definição dos procedimentos metodológicos utilizados para a execução do trabalho.
    - **Análise de requisitos:** Definição da abordagem para levantamento e análise de requisitos da plataforma.

- **Desenvolvimento da solução:** Implementação da plataforma web, seguindo a metodologia de desenvolvimento e os requisitos elicitados.
- **Validação dos resultados obtidos:** Análise e avaliação da implementação da solução e se esta atende aos objetivos do trabalho.

1.4 Organização do Trabalho

Este trabalho está organizado nos seguintes capítulos:

- **Capítulo 1 - Introdução:** Neste capítulo foram apresentados o contexto do trabalho, o problema de pesquisa, os objetivos deste trabalho e, uma síntese da metodologia planejada;
- **Capítulo 2 - Referencial teórico:** Descreve os conceitos que fundamentam este trabalho. Tem como propósito esclarecer e analisar conhecimentos relevantes quanto aos tópicos abordados;
- **Capítulo 3 - Metodologia:** Descreve os conceitos que fundamentam a metodologia planejada. Tem como propósito definir os processos e como serão implementados;
- **Capítulo 4 - Proposta:** Apresenta a proposta deste trabalho, desde os requisitos gerais do sistema até a proposta de implementação de uma plataforma educacional de acesso remoto para bancadas de controle de sistemas embarcados.
- **Capítulo 5 - Desenvolvimento:** Descreve o desenvolvimento da plataforma que permite aos usuários se conectar ao laboratório remoto e acessar os experimentos. São apresentadas as tecnologias utilizadas para a criação da plataforma, bem como a descrição dos recursos e funcionalidades oferecidos aos usuários.
- **Capítulo 6 - Resultados:** Apresenta os resultados obtidos através do desenvolvimento do sistema. Aqui, são discutidos os resultados dos experimentos conduzidos através da plataforma, bem como a eficiência e a facilidade de uso do sistema.
- **Capítulo 7 - Conclusão:** Por fim, é apresentada a conclusão do trabalho, onde são discutidos os principais pontos abordados ao longo do trabalho, incluindo o desenvolvimento da plataforma e os resultados obtidos. Aqui, são apresentadas as principais contribuições do trabalho e as perspectivas para futuros estudos e melhorias.

2 Referencial Teórico

2.1 Considerações Iniciais

Neste capítulo é apresentada a revisão dos principais tópicos relacionados. Com o propósito de esclarecer e analisar conhecimentos e técnicas relevantes a respeito dos assuntos que unidos formam a base estrutural deste trabalho.

2.2 Sistemas embarcados

Um sistema embarcado é um sistema baseado em microprocessador projetado para controlar uma função ou um intervalo de funções, em vez de ser programado pelo usuário final como um computador. O usuário pode escolher as funções, mas não pode alterar a função do sistema adicionando/substituindo *software*. Com um computador, é exatamente isso que um usuário pode fazer: o computador é um processador de texto e, posteriormente, é um console de jogos, apenas mudando o *software*. Os sistemas embarcados são projetados para realizar tarefas específicas (HEATH, 2002).

Em virtude da grande utilização de sistemas de controle e microcontroladores em diversas áreas e produtos tecnológicos, tem crescido bastante o interesse de estudantes nessa área de atuação. Diante da necessidade do mercado de profissionais na área de sistemas embarcados, algumas instituições de ensino superior oferecem cursos de sistemas embarcados tanto na graduação quanto na pós-graduação em engenharia (PARREIRA et al., 2021).

Por se caracterizar por campos interdisciplinares e multidisciplinares, incluindo conhecimentos em eletrônica, processamento de sinais e teoria de controle, algoritmos, sistemas operacionais, o uso de laboratórios no ensino de sistemas embarcados é fundamental para o processo de ensino e aprendizagem do aluno. Com experiências nesses ambientes, os alunos têm a oportunidade de consolidar os conhecimentos teóricos do curso para resolver problemas práticos na vida profissional. Desta forma, os educadores desenvolveram vários métodos de ensino de laboratório para envolver e motivar os alunos na prática de ensino e coordenar aspectos da teoria em situações práticas (MATOS et al., 2019; PARREIRA et al., 2021).

2.3 Laboratórios remotos

Laboratório pode significar várias coisas. Pode se referir tanto às instalações físicas quanto aos equipamentos usados para experimentos. Logo a prática de acessar os recursos do laboratório para atividades práticas pode ser chamado de experimento. De acordo com a (CARNEGIE, 2014), realizar um experimento de um local remoto é chamado de experimento remoto. O experimento permite que os usuários interajam com o mundo real por meio de sistemas eletrônicos de controle e monitoramento acessados por dispositivos computacionais (SILVA et al., 2020).

Laboratórios remotos para o ensino educacional em engenharias estabelecem um método bastante comum utilizado pelas universidades. Dessa forma, oferece um horário flexível para os estudantes, buscando melhores e maiores resultados com os recursos disponíveis. Com isso, utilizando os laboratórios remotos para controlar remotamente os dispositivos físicos, que ficam limitados pela quantidade (SANTOS, 2014).

Segundo Weddell, Bones e Wareing (2014), Laboratórios remotos não permitem que os alunos interajam fisicamente com equipamentos ou aparelhos usados para um experimento em um ambiente de laboratório. Assim, eles são mais seguros, reduzindo problemas associados a falta de equipamentos, vandalismo e danos ao equipamento por uso inadequado. A segurança dos alunos também é melhorada através do isolamento físico de equipamentos perigosos. Esses fatores podem ter vários benefícios, na forma de custos reduzidos de conformidade com saúde e segurança. Até pouco tempo, os desafios tecnológicos envolvidos na comunicação limitavam o crescimento de laboratórios remotos. Entretanto, melhorias e inovações recentes nas tecnologias começaram rapidamente a remover esses impedimentos.

Existem várias categorias em que laboratórios remotos podem ser classificados. Baseado em Karakasidis (2013), podemos apresentar três categorias:

1) Plataformas baseadas na utilização de laboratórios em espaços físicos. Eles são baseados em laboratórios físicos que são controlados remotamente pela internet. Em muitos casos, diz respeito a equipamentos caros ou sensíveis. Esses laboratórios oferecem a oportunidade de exercício fora do laboratório em horários apropriados para os alunos, embora necessitem de agendamento.

2) Plataformas baseadas apenas em software e simuladas. As simulações digitais são baseadas em modelos matemáticos. O grau de semelhança com o equipamento real depende do caso. Na verdade, existem duas subcategorias. Um que o experimento tenta reproduzir apenas esquematicamente o ambiente laboratorial e casos em que o ambiente laboratorial é reproduzido para criar realismo.

3) Plataformas baseadas tanto em laboratórios em espaços físicos, quanto laboratórios simulados.

Dentre algumas vantagens dos laboratórios remotos, estão:

1. Elimina restrições de tempo e espaço;
2. Os experimentos são reais e executados em tempo real;
3. Compartilhamento de materiais caros;
4. Reduzindo o custo de equipamentos usados em laboratório;
5. Reduzindo a manutenção de equipamentos usados em laboratório;
6. Protegendo os alunos durante os testes e experimentos;
7. Protegendo materiais contra o uso indevido ou erro humano;

Em todos os casos foram apresentadas as características particulares de cada aplicação, destacando elementos particulares interessantes seja do lado da tecnologia, seja do lado do desenvolvimento educacional e das técnicas que são utilizadas.

Neste trabalho enfatizamos o foco no desenvolvimento da plataforma baseada na utilização de laboratórios físicos, possibilitando por meio da internet acesso a recursos que na maioria das vezes não possuem suporte para todos. Além de possibilitar que um maior número de alunos os acessem, os laboratórios remotos podem tornar o aprendizado mais flexível em relação ao tempo e local, podendo ser feita conforme a disponibilidade de horário. Conforme ilustradas na Tabela 1, são grandes as vantagens obtidas através dos laboratórios remotos.

2.4 Acesso remoto (SSH)

O Secure Shell (SSH) é uma ferramenta importante para conexões seguras. Trata-se de uma ferramenta de rede privada virtual (VPN) baseada em software que pode criar uma conexão segura sempre que necessário. Pode ser usado para fazer login com segurança em *host* remoto, desde que você tenha as autorizações adequadas (BOTH, 2020).

Embora seu nome seja *Secure Shell*, o SSH não é exatamente um *shell*, é um grupo de protocolos de conexão que permite que computadores compartilhem links e informações criptografadas de forma segura. Além disso, ele fornece autenticação confiável para os *hosts* e usuários, criptografa a comunicação e transmissão entre eles e garante a integridade das transmissões, detectando e notificando se os dados estiverem alterados ou ausentes (BOTH, 2020).

Desse modo, através de um computador ligado à internet, usuários conseguem acessar e modificar arquivos. A proposta do acesso remoto e do protocolo é criar uma

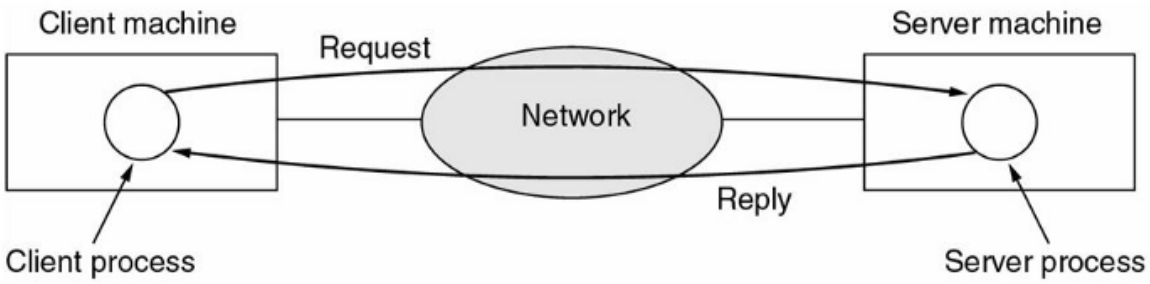
conexão segura, já que não haverá nenhuma invasão nesses arquivos. Assim garante que somente os dois pontos tenham acesso a esta informação, o servidor e o computador que enviou os dados para o acesso remoto.

2.5 Plataforma web

Plataformas web são desenvolvidas para tornar operações mais rápidas, simples e eficazes. Permitir o acesso remoto é uma dentre as vantagens mais relevantes na utilização desta plataforma, onde colaboradores e usuários com acesso a internet possam acessar o software remotamente e utilizar suas funcionalidades, sem que seja necessário a instalação no computador.

Essas soluções são para computadores com navegadores ou clientes front-end que se comunicam com servidores back-end pela Internet. Essa forma de processamento distribuído, na qual os computadores clientes solicitam serviços e os servidores fornecem um conjunto de serviços em uma rede de computadores, é conhecida como arquitetura cliente-servidor (FREITAS; SILVA, 2020; TANENBAUM, 2003).

Figura 3 – O modelo cliente/servidor envolve solicitações e respostas



Fonte: (TANENBAUM, 2003)

Conforme na Figura 3, na arquitetura cliente/servidor as plataformas trabalham por meio de requisições e respostas. Após o servidor receber a solicitação do cliente, ele executa a tarefa solicitada ou encontra os dados solicitados e envia o retorno com a resposta (TANENBAUM, 2003). No caso da plataforma web, essa resposta pode ser exibida para o usuário através do *front-end*. Essas tecnologias serão melhor descritas nas seções a seguir.

2.5.1 Back-end

O *back-end* se refere a qualquer parte da plataforma que os usuários não veem. Conhecida também como arquitetura do servidor, o *back-end* é responsável por integrar as

funcionalidades internas do servidor, sendo uma camada de acesso a dados (GOIS, 2002). A função do *back-end* está relacionada com banco de dados, servidores e segurança. Dessa forma, o banco de dados responsável por armazenar as informações e o servidor por garantir a operação dos computadores.

Dentre as operações de *back-end* podem incluir também login, serviços de pesquisa e qualquer outra função do qual as informações e dados precisam ser manipulados, enviados ou recuperados de um banco de dados.

2.5.2 Front-end

O desenvolvimento web *front-end*, também conhecido como arquitetura do cliente, é tudo que o usuário consegue enxergar e interagir na aplicação. A combinação da aparência, interface do usuário e a sensação ao usar. Através dessa plataforma o cliente consegue acessar os serviços e recursos disponíveis do servidor, realizando requisições.

Devido à quantidade de funcionalidades oferecidas em suas interfaces de usuários, as aplicações web acabam tendo desafios na estruturação de suas arquiteturas, fazendo com que dificulte a manutenção. Com isso, é comum a utilização de *frameworks* para auxiliar no desenvolvimento. Os *frameworks* e bibliotecas permitem uma melhor estruturação e simplicidade, facilitando assim a manutenção e criação de aplicações (KALUZA; TROSKOT; VUKELIC, 2018). Atualmente existem vários *frameworks* que podem ser utilizados para *front-end*, sendo os mais comuns ReactJS, VueJS e Angular.

2.6 Streaming de vídeo

Assistir vídeos online tornou-se uma forma muito popular de entretenimento. Nos últimos anos, observou-se que o *streaming* de vídeo é responsável pela maior parte do tráfego e continua a aumentar a cada ano (JULURI; TAMARAPALLI; MEDHI, 2016). Em virtude disso, o uso de vídeos também como material didático está cada vez maior nas instituições de ensino.

Existem duas formas principais de entregar conteúdo de vídeo pela internet: vídeo sob demanda (VOD) e transmissão ao vivo. Na transmissão ao vivo o conteúdo pode ser transmitido à medida que é criado, ou com um período de atraso (*delay*), sendo passado direto da câmera para o dispositivo do usuário (AMARAL, 2020). Por outro lado, o vídeo sob demanda o conteúdo permanece armazenado em bibliotecas de servidores e fica disponível para os usuários a qualquer momento e a qualquer dispositivo (AMARAL, 2020).

No caso da transmissão ao vivo, existe um fator adicional que entra em jogo: a latência. A latência é definida como a diferença de tempo entre um evento sendo capturado

e o momento em que ele é apresentado ao usuário (TIDESTROM, 2019). No caso de bancadas de sistemas embarcados, é importante a comunicação em tempo real onde a baixa latência não afeta o tempo de espera entre as respostas, tornando melhor a análise de dados recebidos.

O WebRTC é uma forma de atingir baixas latências de video, destinado a comunicações em tempo real ponto a ponto, ele contém a capacidade de *streaming* de vídeo em baixas latências. WebRTC é um projeto que foi aberto ao público pelo Google em junho de 2011, o objetivo do projeto é habilitar aplicativos de navegador de comunicação em tempo real (RTC). O WebRTC fornece um conjunto de protocolos que permite o suporte de comunicações entre navegadores. Por outro lado, a finalidade do WebRTC é a comunicação ponto a ponto em tempo real pela web. Portanto, o foco principal é manter as latências baixas, ao custo de uma diminuição na qualidade de não ter certeza absoluta de que tudo é recebido (TIDESTROM, 2019).

2.7 Programação de firmware de modo remoto

Firmware é um software de controle de baixo nível para sistemas embarcados. O *firmware* define exatamente a função do microcontrolador, quando ele está lendo a série de tarefas e comandos em código da memória do programa. Para obter segurança, o *firmware* depende da eficiência e robustez (HOLLY, 2020).

Dado o fato que o *firmware* tem amplo papel dentro de um sistema embarcado, são necessárias atualizações que permitam o gerenciamento de *bugs* e outras vulnerabilidades características de sistemas baseados em software, bem como a evolução do sistema. A alternativa de sistemas embarcados para atualizar *firmware* pela internet é bem conhecida e bastante utilizado em sistemas ARM e sistemas operacionais (JURKOVIC; SRUK, 2014).

As atualizações de *firmware* do sistema são implementadas com base em acesso direto à memória no agente de software. A conexão com a Internet é feita para permitir a comunicação com o servidor. Avalia o processo de implementação da atualização de software descrita por uma série de medições fornecendo informações sobre as propriedades e características. A atualização de software por agentes de software também é adequado para implantação em uma região de sistemas embarcados (JURKOVIC; SRUK, 2014).

Bootloader é um software que grava *firmware* na memória do programa flash do dispositivo durante a fase de inicialização. Grande parte dos microcontroladores possuem *bootloader* em sua memória padrão. Esses carregadores de inicialização são executados após a configuração ou reinicialização e aguardam o evento para gravar o *firmware* (HOLLY, 2020).

2.8 Projetos similares

A Experimentação Remota é uma área de pesquisa que visa ampliar a capacidade experimental para além de seus limites, utilizando os recursos da Internet e de outros meios tecnológicos capazes de prover acesso remoto, possibilitando o compartilhamento de recursos de um modo geral. Em outras palavras, é possível operar-se um equipamento remotamente, ou seja, de um local distante do mesmo. Neste contexto, serão apresentados projetos similares que exploram essa área de pesquisa e ampliam o conhecimento sobre as possibilidades e aplicações da Experimentação Remota.

2.8.1 Relle

O Ambiente de Aprendizado com Experimentos Remotos (Relle) é uma ferramenta que permite o controle e a manipulação de experimentos a distância. Ele é desenvolvido pelo Grupo de Trabalho em Experimentação Remota (GT-MRE) da Universidade Federal de Santa Catarina, localizado no Laboratório de Experimentação Remota (RExLab). O projeto é financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e pela Rede Nacional de Ensino e Pesquisa (RNP), por meio de seus programas de PD Temáticos (RELLE, 2023).

O RExLab surgiu em 1997 na Universidade Federal de Santa Catarina e, hoje em dia, é uma rede de 12 universidades em 5 países diferentes, com o objetivo de democratizar o conhecimento científico e tecnológico, incentivando jovens a seguirem carreiras na área e integrando a educação científica no ensino (RELLE, 2023).

A plataforma é um ambiente virtual que permite a realização de experimentos físicos de forma remota através da internet, com o objetivo de utilizar as tecnologias de informação e comunicação como meio para melhorar a qualidade da aprendizagem prática dos estudantes, especialmente nas áreas de ciência, tecnologia, engenharia e matemática. Além disso, a experimentação remota estimula a criatividade, experimentação e promove a interdisciplinaridade na educação.

2.8.2 LabRemotoFPGA

O projeto “Laboratório remoto baseado em FPGA aplicado nas disciplinas de Prática de Eletrônica Digital 1 e 2 da Faculdade UnB Gama” foi desenvolvido por Rodrigo Bonifácio de Medeiros como Trabalho de Conclusão de Curso para obtenção do Bacharelado em Engenharia Eletrônica na Universidade de Brasília em 2018. O projeto foi orientado pelo Prof. Dr. Daniel Mauricio Muñoz Arboleda.

O objetivo do projeto foi criar um laboratório remoto baseado em FPGA para as disciplinas de Prática de Eletrônica Digital 1 e 2 na Faculdade UnB Gama, com o

intuito de resolver a falta de kits de desenvolvimento da Basys 3 na instituição. Com isso, os alunos poderão trabalhar fora da instituição e melhorar o processo de ensino-aprendizagem, além de ser uma solução mais barata do que adquirir novos kits. O projeto inclui a implementação de uma arquitetura de hardware em FPGA e um modelo de aplicação cliente-servidor para uma interface web gráfica. Onde o usuário pode criar seu projeto VHDL e interagir com a placa remotamente, usando chaves, botões, LEDS e *displays* de sete segmentos virtuais na interface para simular os componentes físicos do FPGA. Além disso, é possível acompanhar ao vivo os resultados da placa Basys 3 através de uma transmissão web. Experimentos com circuitos sequenciais e máquinas de estados mostraram a correta interação entre o servidor, a placa e o usuário, tornando a solução viável e facilmente replicável para outros servidores e FPGAs (MEDEIROS, 2018).

3 Metodologia

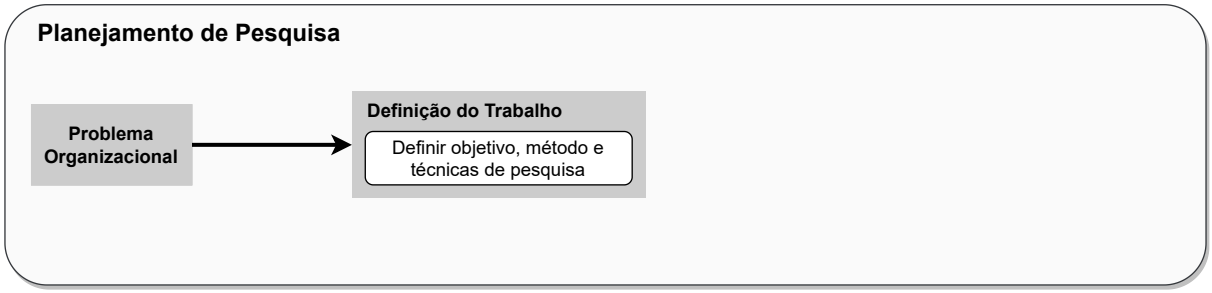
3.1 Considerações Iniciais

Dado o objetivo deste trabalho, especificar e implementar uma plataforma educacional de acesso remoto para bancadas de sistemas embarcados, neste capítulo é retomado o plano metodológico apresentado brevemente no Capítulo 1.

3.2 Fase de Planejamento da Pesquisa

Nesta fase de planejamento da pesquisa foram definidos o tema, os objetivos de pesquisa, o plano e a classificação metodológica, conforme apresentado na Figura 4.

Figura 4 – Planejamento da pesquisa

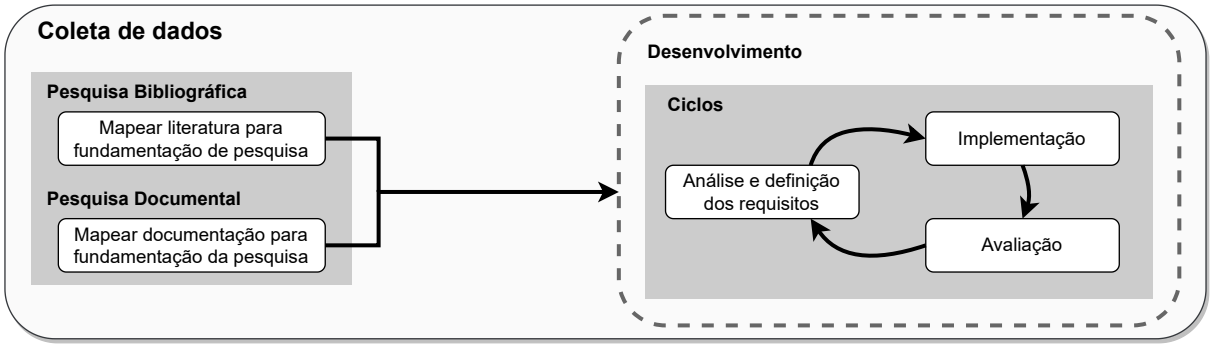


Fonte: Autor

3.3 Fase de Coleta de Dados

Nesta fase de coletas de dados são definidos os métodos de pesquisa bibliográfica e pesquisa documental, que são associados a etapa de desenvolvimento.

Figura 5 – Coleta de dados



Fonte: Autor

Conforme o plano metodológico, a fase de Coleta de Dados compreende o emprego das técnicas Revisão bibliográfica e Documental e de um processo para o desenvolvimento da Plataforma:

1. **Pesquisa bibliográfica:** Levantamento de fontes a partir do tema, para aprimorar a visão do contexto e melhor adequar o escopo, para assim definir uma base teórica concreta para apoiar os objetivos do trabalho.
2. **Pesquisa documental:** Estudo de ferramentas e tecnologias que auxiliarão na execução do trabalho, serão abordadas no referencial e no desenvolvimento do trabalho.
3. **Processo de Desenvolvimento:** Definição dos procedimentos metodológicos utilizados para a execução do trabalho, constituído de:

3.3.1 Pesquisa Bibliográfica

Baseada em livros, artigos, monografias e outros materiais aceitos pela comunidade científica, foi realizada a pesquisa bibliográfica deste trabalho a cerca de plataformas web de acesso remoto para laboratórios reais de sistemas embarcados. Foram levantados também materiais que exemplificam a arquitetura e a aplicação de métodos bem como a avaliação. Para isso foram utilizados recursos como IEEE, Scopus, CAPES, Google Acadêmico e documentações próprias das tecnologias.

Assim as etapas para realização do desenvolvimento a cerca objeto de estudo foram definidas, a fim de obter fundamentação para determinar o método mais adequado para a resolução do problema.

3.3.2 Pesquisa Documental

Nesta seção, serão listadas as principais tecnologias e ferramentas escolhidas para utilização no desenvolvimento da plataforma proposta por este trabalho. Serão apresentadas as tecnologias em si e as ferramentas que darão suporte em geral.

- **Node.js**

  Node.js é um software multiplataforma que renderiza código JavaScript para o lado do servidor (*back-end*), sem a necessidade de um navegador. Utilizado para criar facilmente aplicativos escaláveis e rápidos, além de fornecer uma vasta biblioteca de módulos que simplificam bastante o desenvolvimento (Node.js, 2022).

- **NestJS**

  NestJS é uma plataforma de desenvolvimento de aplicativos em Node.js com foco em aplicativos de alta performance e escalabilidade. Ele fornece uma estrutura modular

baseada em TypeScript para a criação de aplicativos *back-end* com suporte a vários protocolos, incluindo HTTP e WebSockets. Além disso, o NestJS oferece recursos avançados, como testes unitários, validação de dados, autenticação e autorização, e muito mais.

- **React.js**

  React é uma biblioteca de desenvolvimento baseada em JavaScript, mantida pelo Facebook e pela comunidade de software aberto. A biblioteca surgiu por volta de 2013 e hoje se tornou uma das bibliotecas de *front-end* mais utilizadas para o desenvolvimento web. O React fornece várias extensões para o suporte de diversas arquiteturas de aplicativos, além da interface do usuário. A popularidade do React se deu pela facilidade na criação de aplicações, o desempenho aprimorado e a grande reutilização dos componentes, que reduz o tempo de desenvolvimento (REACT..., 2022).

- **TypeScript**

  TypeScript é uma linguagem de programação baseada no JavaScript que surgiu com o objetivo de estender, sendo assim compatível com bibliotecas baseadas em JavaScript, como Node e React. O TypeScript adiciona extensões ao JavaScript destinadas a torná-lo mais claro em relação aos tipos de dados usados no código. Por ser mais confiável, normalmente é utilizado para prevenir *bugs*, por sempre apontar erros de tipo na compilação no momento do desenvolvimento (TYPESCRIPT, 2022).

- **GitHub**

  GitHub é um recurso de programação bastante popular, usado para o compartilhamento e armazenamento remoto de repositórios e códigos. Utilizado para facilitar o gerenciamento e a colaboração de projetos, é um serviço de interface web de hospedagem de repositório Git, que busca simplificar diversos recursos. O mesmo será utilizado para armazenamento e controle de tarefas do repositório da plataforma desenvolvida.

3.3.3 Processo de Desenvolvimento

O processo de desenvolvimento adotado neste trabalho é baseado em boas práticas, técnicas e metodologias da Engenharia de Software estudadas durante a graduação, bem como o desenvolvimento, os testes, a integração contínua, dentre outros aspectos. Para o desenvolvimento da plataforma deste trabalho foram escolhidos alguns elementos de metodologias ágeis, decorrentes do Scrum e do Kanban. Como se trata de um projeto desenvolvido individualmente, serão feitas algumas adaptações nas metodologias para que se encaixem melhor a este trabalho.

O **Scrum** é um método do qual se pode empregar vários processos e técnicas, tornando mais clara e eficiente a abordagem de gerenciamento e desenvolvimento de produtos, de modo a melhorá-los (SCHWABER; SUTHERLAND, 2020). Do Scrum serão utilizadas as seguintes práticas:

- **Product Backlog:** Uma lista de tarefas a partir dos requisitos do software, que pode variar de acordo com o domínio e evolução do trabalho. As tarefas serão descritas e disponibilizadas no GitHub.
- **Sprint:** "Sprints são o bater de coração do Scrum, onde as ideias são transformadas em valor"(SCHWABER; SUTHERLAND, 2020). É o período que as atividades definidas no *Backlog* são executadas, geralmente tem duração de 1 a 3 semanas.

O método **Kanban** entra em complemento ao Scrum, utilizando um painel que define as etapas das tarefas e o fluxo que elas devem seguir até serem entregues. Melhorando a visão do todo do produto e acompanhando a realização dos itens dispostos na etapa do *backlog*. Isso torna o desenvolvimento mais ágil e enxuto, facilitando o gerenciamento do projeto.

3.3.3.1 Análise e definição de requisitos

O objetivo desta etapa é entender melhor o funcionamento e as necessidades do projeto, elicitar os requisitos a serem desenvolvidos e a partir dos requisitos levantados realizar a criação do *backlog* do produto. É necessário entender as necessidades do produto para o levantamento dos requisitos funcionais e não funcionais. De acordo com o entendimento do projeto, é feita a análise da perspectiva mais técnica para definição das tarefas, que serão descritas no repositório do projeto e avaliadas pela complexidade. Por fim, deve-se criar o *backlog*, adicionando as novas tarefas que foram definidas nos passos anteriores. Durante o decorrer do projeto podem ocorrer mudanças e até o surgimento de tarefas.

3.3.3.2 Implementação

À medida que as tarefas são definidas, a implementação do sistema começa e os modelos selecionados são integrados ao sistema. Um dos critérios de aceitação de uma tarefa pode ser a presença de testes e a implementação de novas funcionalidades, avaliando o desempenho das funcionalidades e modelos desenvolvidos.

3.3.3.3 Avaliação

Analisar os dados coletados e avaliar se as ações implementadas estão atingindo os objetivos desejados. Essa análise pode ser usada como insumo para criação de novos ciclos.

3.4 Análise dos resultados

Como citado na seção 1.3, serão descritos e especificados os resultados de todas as fases, resultando nos capítulos finais do trabalho. Com o objetivo de pesquisar sobre a plataforma e sua implementação, tendo como resultado a documentação e o artefato de software.

4 Proposta do Trabalho

4.1 Considerações Iniciais

Dado o objetivo deste trabalho, neste capítulo é apresentado um detalhamento da proposta deste trabalho, bem como as atividades já realizadas e as atividades a serem realizadas. Seguido do cronograma de atividades.

4.2 Levantamento dos requisitos

Este capítulo tem como objetivo levantar os requisitos para o trabalho, bem como delimitar o escopo do mesmo. Os requisitos principais, que serão melhor detalhados posteriormente, são os seguintes:

- **Cadastro de professores:** O sistema deve ser capaz de cadastrar professores, que terão acesso ao cadastro de turmas, experimentos e alunos.
- **Cadastro de turmas:** O sistema deve permitir que professores cadastrados realizem o cadastro de turmas o vínculo dessas turmas aos alunos e experimentos.
- **Cadastro dos experimentos:** O sistema deve permitir que professores cadastrados realizem o cadastro de experimentos e vincular os experimentos às turmas.
- **Cadastro de alunos:** O sistema deve permitir que professores cadastrados realizem o cadastro de alunos para acesso às turmas.
- **Sistema de agendamento:** O sistema deve permitir que alunos cadastrados façam agendamento de experimentos vinculados a sua turma, conforme a disponibilidade.
- ***Streaming* de Câmera:** O sistema deve apresentar um *streaming* de câmera onde os usuários possam ver os resultados dos experimentos na bancada, em tempo real.
- **Dashboard:** O sistema deve apresentar um *dashboard*, por meio de um WebView embutido, onde os usuários possam manipular e acompanhar os experimentos.
- **Acesso via VPN e/ou SSH:** O sistema deve permitir o acesso de usuários por meio de VPN e/ou de SSH, conforme seja determinado.
- **Acesso à dispositivo USB remoto:** O sistema deve ser capaz de permitir acesso a dispositivo USB remoto, onde o usuário se conecta ao USB remoto pelo sistema como se o dispositivo estivesse conectado à sua própria máquina.

4.3 Atividades realizadas

Determinado o tema, o objeto de estudo e o problema, foi realizada uma pesquisa e descrição bibliográfica, que serão detalhadas nos tópicos a seguir:

1. **Pesquisa bibliográfica**

   Levantamento das informações teóricas que servem de sustentação para o trabalho foram atingidas por meio de pesquisas em bases científicas. Nos capítulo 1 são introduzidos os conceitos, problemas e objetivos para o desenvolvimento da plataforma de ensino remoto para laboratórios físicos de Sistemas Embarcados, além da definição inicial da metodologia.

2. **Pesquisa documental**

   No capítulo 2 são documentados e descritos os conceitos que fundamentam este trabalho. São representados tópicos a respeito de temas e tecnologias que serão utilizadas para execução da construção desta plataforma.

3. **Elaboração da proposta**

   No capítulo 3 é elaborada a proposta deste trabalho. Dentre os tópicos é descrita como foi realizada a pesquisa bibliográfica, é feita a pesquisa das tecnologias que serão utilizados e a descrição da metodologia. Dentro da metodologia é abordado os métodos de desenvolvimento, como será realizada a análise dos requisitos e a coleta e análise dos resultados. Foi feito o levantamento dos requisitos principais, por fim, os resultados preliminares serão apresentados.

4.3.1 Considerações Finais

Neste capítulo foi apresentada a proposta do trabalho, que busca colaborar com a solução dos problemas encontrados acerca do ensino remoto de laboratórios de Sistemas Embarcados. A proposta tem como objetivo a especificação de uma plataforma que auxilie no ensino prático de Sistemas Embarcados por meio de laboratórios remotos, visando ampliar e flexibilizar o ensino, além de viabilizar a prática para alunos de graduação e pós-graduação. É importante ressaltar que a ideia não é abordar todos os requisitos e entregar a plataforma completa, mas sim desenvolver o sistema web para o agendamento e controle remoto, bem como explorar tecnologias como o SSH e câmeras para viabilizar o acesso remoto. Com esse foco delimitado, busca-se alcançar resultados significativos na implementação das funcionalidades essenciais para o ensino prático em bancadas de sistemas embarcados, proporcionando uma base sólida para trabalhos futuros de desenvolvimento e aprimoramento da plataforma educacional de acesso remoto.

5 Desenvolvimento

Neste capítulo, será discutido o desenvolvimento dos recursos listados como requisitos do Laboratório Remoto. O projeto pode ser encontrado em um repositório aberto no GitHub no seguinte link: <https://github.com/orgs/TCC-Andre/repositories>.

Este capítulo visa esclarecer os tópicos mencionados na metodologia e revisão da literatura, detalhar as etapas de desenvolvimento do projeto, decisões arquiteturais e explicar alguns arquivos de código.

5.1 Requisitos do sistema

A concepção de requisitos é uma parte fundamental do processo de desenvolvimento de software. Esses elementos fornecem uma visão clara do que o sistema deve fazer e como ele deve ser construído.

A utilização do *backlog* facilita a visualização dos requisitos, por se tratar de uma lista de tarefas a serem realizadas. É uma representação visual dos requisitos e ajuda a garantir que o desenvolvimento do software esteja alinhado. O *backlog* também permite a priorização das tarefas, o que é importante para garantir que o desenvolvimento seja realizado de forma eficiente e no prazo estipulado.

Para a priorização foi utilizada a técnica Moscow, uma ferramenta de gestão de projetos que ajuda a priorizar tarefas e requisitos em um projeto de software. Ela classifica os itens do *backlog* em quatro categorias: *Must have, Should have, Could have e Won't have*.

1. ***Must have:*** São tarefas ou requisitos que são considerados críticos para o sucesso do projeto. Essas tarefas são consideradas *"must"* porque são consideradas obrigatórias para o funcionamento correto do sistema ou para atender às expectativas dos usuários.

2. ***Should have:*** São tarefas ou requisitos que são importantes, mas não são críticos. Essas tarefas devem ser realizadas logo após as tarefas *"must have"*.

3. ***Could have:*** São tarefas ou requisitos que são considerados úteis, mas não são prioritários. Essas tarefas podem ser realizadas se houver tempo e recursos disponíveis.

4. ***Won't have:*** são tarefas ou requisitos que não serão realizados neste projeto.

A seguir, o Quadro 2 apresenta a distribuição e priorização dos requisitos definidos no *product backlog*.

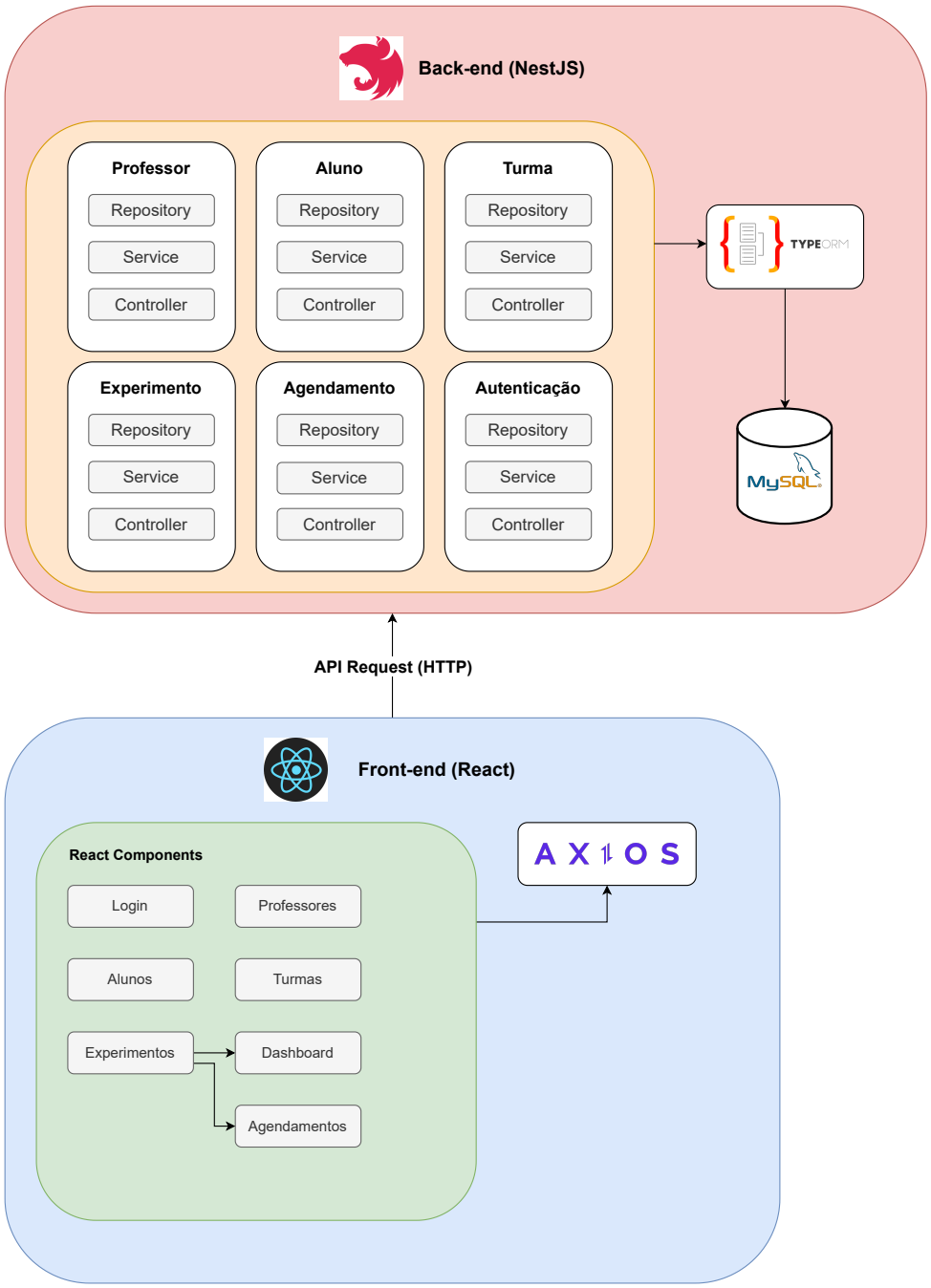
Tabela 1 – Product Backlog

Código	**Requisito**	**Prioridade**
RF01	O sistema deve permitir que o professor efetue o cadastro de turmas	*Must*
RF02	O sistema deve permitir que o professor efetue o cadastro de alunos	*Must*
RF03	O sistema deve permitir que o professor efetue o cadastro de experimentos	*Must*
RF04	O sistema deve permitir que usuários realizem login	*Must*
RF05	O sistema deve ordenar os agendamentos por disponibilidade	*Should*
RF06	O sistema deve permitir que usuários realizem agendamento de horário nos experimentos	*Must*
RF07	O sistema deve permitir que usuários realizem os experimentos	*Must*
RF08	O sistema deve apresentar aos usuários um *dashboard* embutido para manipulação do experimento	*Must*
RF09	O sistema deve apresentar aos usuários um *streaming* de câmera com o experimento na bancada em tempo real	*Must*
RF10	O sistema deve permitir aos usuários acesso à dispositivo USB remoto	*Must*

5.2 Arquitetura do sistema

A escolha da arquitetura baseia-se em conceitos importantes para a construção de sistemas robustos e escaláveis. A separação de responsabilidades entre as camadas é fundamental, com o React sendo utilizado para criar a interface do usuário e o NestJS fornecendo a *API* de dados. Além disso, o uso de tais tecnologias juntas permite um desenvolvimento mais rápido e eficiente devido à sua ampla comunidade e ferramentas disponíveis. De certa forma, essa arquitetura é uma boa opção para uma solução escalável, eficiente e bem organizada para sistemas web.

Figura 6 – Arquitetura do sistema



Fonte: Autor

O diagrama de arquitetura da Figura 6 demonstra como o NestJS e o React trabalham juntos para construir uma aplicação eficiente. O NestJS é responsável por gerenciar todos os dados que são enviados e recebidos através da *API* da aplicação. Ele realiza consultas ao banco de dados MySQL usando o TypeORM, que é uma biblioteca poderosa para lidar com o mapeamento de objetos relacionais. Além disso, o NestJS fornece segurança e validação dos dados recebidos para garantir que a aplicação seja protegida contra ataques e erros de entrada de dados.

Por sua vez, o React é responsável por construir a interface do usuário, que é a parte da aplicação que o usuário final vê e interage. Ele usa a biblioteca Axios para consumir os dados da API e exibi-los na tela. Isso permite que o usuário tenha acesso a todas as informações relevantes da aplicação de maneira clara e intuitiva.

Em resumo, o NestJS e o React são componentes importantes e complementares da aplicação, trabalhando juntos para garantir que os dados sejam gerenciados de forma segura e eficiente, enquanto a interface do usuário é construída de maneira atrativa e fácil de usar.

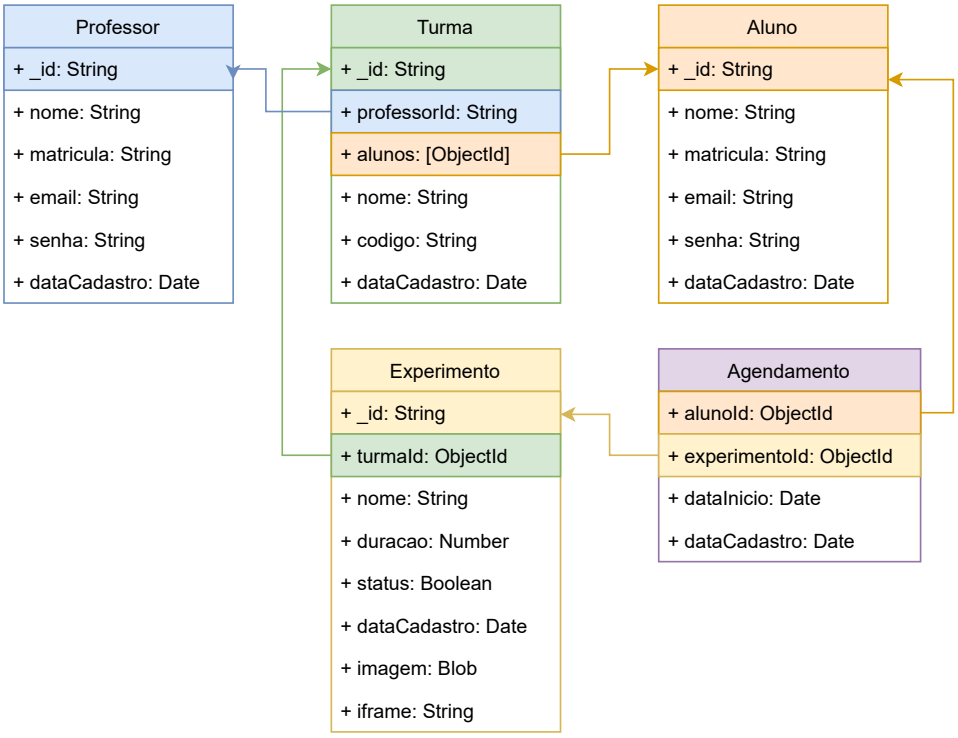
5.2.1 Back-end

O *back-end* da aplicação foi desenvolvido em NestJS. Ele é o coração da nossa aplicação, fornecendo uma estrutura modular que permite a criação de *endpoints* da *API* de maneira rápida e eficiente. Esses *endpoints* são os pontos de contato, onde solicita e recebe dados da aplicação.

O banco de dados relacional que estamos usando é o MySQL. Ele é responsável por armazenar e gerenciar todos os dados da aplicação, incluindo informações sobre usuários, experimentos e outros itens importantes. Além disso, é responsável por garantir que as informações sejam armazenadas de forma segura e acessíveis quando necessário. O NestJS se comunica com o banco de dados MySQL através de uma camada de acesso a dados, que é responsável por gerenciar as operações de inserção, atualização, consulta e exclusão de dados.

Para melhor compreensão e documentação, foi desenvolvido o diagrama do banco de dados da Figura 6.

Figura 7 – Diagrama do banco de dados



Fonte: Autor

O diagrama é uma representação visual das tabelas e colunas presentes no banco de dados. Ele ajuda a ilustrar a estrutura do banco de dados e como as informações são armazenadas e relacionadas entre si.

Para fazer a conexão e interagir com o MySQL, estamos usando a biblioteca TypeORM. Ela é uma das bibliotecas mais populares e amplamente utilizadas para acesso a dados, e oferece uma ampla gama de recursos que tornam a integração com o MySQL muito mais fácil. Além disso, o TypeORM é altamente customizável, o que significa que podemos ajustá-lo de acordo com as necessidades específicas da nossa aplicação.

Em resumo, o NestJS, o MySQL e o TypeORM formam a base sólida da nossa aplicação. Juntos, eles garantem que a aplicação seja rápida, segura e fácil de usar, oferecendo aos usuários uma ótima experiência.

5.2.1.1 Estrutura arquitetural do NestJS

Em resumo, a arquitetura do NestJS é composta por três camadas básicas: Controlador, Serviço e Repositório, cada um com sua responsabilidade específica.

Controlador é a camada responsável por lidar com as requisições HTTP e responder a elas. Eles são responsáveis por receber e validar os dados de entrada, chamar o Serviço adequado e retornar uma resposta para o cliente.

Serviço é a camada que contém a lógica de negócios da aplicação. Ele é responsável por realizar operações complexas e interagir com outras fontes de dados, como bancos de dados, *APIs* externas, entre outros. O Serviço é invocado pelo Controlador para realizar tarefas específicas e retornar os resultados para o Controlador, que, por sua vez, retorna a resposta ao cliente.

Repositório é a camada responsável por lidar com a persistência de dados. Ele é usado para acessar e manipular os dados armazenados em fontes de dados, como bancos de dados, arquivos, etc. O Repositório é invocado pelo Serviço para recuperar ou armazenar dados.

Abaixo temos figuras como exemplo esta arquitetura aplicada dentro do desenvolvimento do *back-end*.

Figura 8 – Controlador

```typescript
@ApiTags('Alunos')
@Controller('alunos')
export class AlunosController {
  constructor(private readonly alunosService: AlunosService) {}

  @Post()
  create(@Body() cadastrarAlunoDto: CadastrarAluno): Promise<Aluno> {
    return this.alunosService.create(cadastrarAlunoDto);
  }

  @Get()
  findAll(): Promise<Aluno[]> {
    return this.alunosService.findAll();
  }

  @Get(':id')
  findOne(@Param('id', ParseUUIDPipe) id: string): Promise<Aluno> {
    return this.alunosService.findOne(id);
  }

  @Put(':id')
  update(
    @Param('id', ParseUUIDPipe) id: string,
    @Body() editarAlunoDto: EditarAluno,
  ) {
    return this.alunosService.update(id, editarAlunoDto);
  }

  @Delete(':id')
  remove(@Param('id') id: string): Promise<void> {
    return this.alunosService.remove(id);
  }
}
```

Fonte: Autor

Figura 9 – Serviço e Repositório

```typescript
@Injectable()
export class AlunosService {
  constructor(
    @InjectRepository(Aluno)
    private readonly alunosRepository: Repository<Aluno>,
  ) {}

  async create(cadastrarAlunoDto: CadastrarAluno): Promise<Aluno> {
    const user = new Aluno();
    user.nome = cadastrarAlunoDto.nome;
    user.matricula = cadastrarAlunoDto.matricula;
    user.email = cadastrarAlunoDto.email;
    const hash = await bcrypt.hash(cadastrarAlunoDto.senha, saltRounds);
    user.senha = hash;
    user.turma = cadastrarAlunoDto.turma;
    user.dataCadastro = dayjs().format();

    const usuarioExistente = await this.alunosRepository.findOneBy({
      matricula: user.matricula,
    });

    if (usuarioExistente) {
      throw new HttpException(
        {
          statusCode: HttpStatus.BAD_REQUEST,
          message: 'Matricula já cadastrada!',
        },
        HttpStatus.BAD_REQUEST,
      );
    } else {
      return this.alunosRepository.save(user);
    }
  }

  async findAll(): Promise<Aluno[]> {
    return this.alunosRepository.find();
  }

  findOne(id: string): Promise<Aluno> {
    return this.alunosRepository.findOneBy({ id: id });
  }

  async remove(id: string): Promise<void> {
    await this.alunosRepository.delete(id);
  }

  async findOneMatricula(matricula: string): Promise<Aluno | undefined> {
    return this.alunosRepository.findOneBy({ matricula: matricula });
  }

  async update(id: string, editarAlunoDto: EditarAluno): Promise<Aluno> {
    const user = await this.alunosRepository.findOneBy({ id: id });

    user.nome = editarAlunoDto.nome;
    user.matricula = editarAlunoDto.matricula;
    user.email = editarAlunoDto.email;

    return this.alunosRepository.save(user);
  }
}
```

Fonte: Autor

Na Figura 8 o Controlador recebe as requisições em formato HTTP, lida com as entradas, chama o serviço e retorna ao usuário a reposta adequada. E na Figura 9 podemos ver o serviço, que é quem lida com a lógica e realiza as operações, e o repositório, que é utilizado para salvar e buscar os dados dentro do banco MySQL.

5.2.1.2 Criação e gerenciamento de professores, alunos e turmas

Trata-se de uma interface via *API* que permite a criação e gerenciamento de informações de professores, alunos e turmas. O **professor**, trata-se de um registro com dados de um professor matriculado na instituição e que terá papel de administrador dentro do sistema. O **aluno** trata-se de um registro com dados de um aluno matriculado e que poderá acessar os experimentos do sistema. E a **turma** trata-se de um registro com dados da turma, que será vinculada a professores e experimentos, e que contém os alunos.

Os endpoints são responsáveis por garantir a integridade dos dados, possibilitando sua criação, busca, atualização e exclusão de forma segura e eficiente.

As seguintes funções estão presentes nos três *endpoints*:

- **Criar**: Envia uma requisição HTTP POST para o *endpoint* de professores com os dados do professor a ser criado. O servidor cria um novo registro de professor com base nestes dados e retorna uma resposta HTTP com o status 201 (Criado) e os dados do professor recém-criado.
- **Buscar Todos:** Envia uma requisição HTTP GET para o *endpoint* de professores sem nenhum parâmetro adicional. O servidor retorna uma resposta HTTP com o status 200 (OK) e uma lista de todos os registros de professores.
- **Buscar Um:** Envia uma requisição HTTP GET para o *endpoint* de professores com o ID do professor desejado como parâmetro. O servidor retorna uma resposta HTTP com o status 200 (OK) e os dados do professor específico.
- **Atualizar:** Envia uma requisição HTTP PUT para o *endpoint* de professores com o ID do professor e os dados atualizados como parâmetros. O servidor atualiza o registro de professor com base nestes dados e retorna uma resposta HTTP com o status 200 (OK) e os dados do professor atualizados.
- **Excluir:** Envia uma requisição HTTP DELETE para o *endpoint* de professores com o ID do professor como parâmetro. O servidor exclui o registro de professor e retorna uma resposta HTTP com o status 204 (Sem Conteúdo).

5.2.1.3 Criação e gerenciamento de experimentos

Trata-se de uma interface via *API* que permite a criação e gerenciamento de informações sobre experimentos. O **experimento** trata-se de um registro com dados de um experimento que pode ser acessado por alunos e professores, além de ser vinculado a turmas. O *endpoint* é similar aos anteriores, que possibilita sua criação, busca, atualização e exclusão, com o adição da consulta de experimentos disponíveis para o aluno. Além de receber dados com a imagem do experimento e o *iframe* que será embutido no *front-end*.

- **Criar**: Envia uma requisição HTTP POST para o *endpoint* de experimentos com os dados do professor a ser criado.
- **Buscar Todos:** Envia uma requisição HTTP GET para o *endpoint* de experimentos sem nenhum parâmetro adicional.
- **Buscar Um:** Envia uma requisição HTTP GET para o *endpoint* de experimentos com o ID do professor desejado como parâmetro.
- **Buscar experimentos disponíveis para o aluno:** Envia uma requisição HTTP GET para o *endpoint* de experimentos com o ID do aluno desejado como parâmetro.
- **Atualizar:** Envia uma requisição HTTP PUT para o endpoint de experimentos com o ID do professor e os dados atualizados como parâmetros.
- **Excluir:** Envia uma requisição HTTP DELETE para o *endpoint* de experimentos com o ID do professor como parâmetro.

5.2.1.4 Criação e gerenciamento de agendamentos

Trata-se de uma interface via *API* que permite a criação e gerenciamento de informações sobre agendamentos. O **agendamento** refere-se a um registro com dados de um agendamento de um experimento, que uma vez agendado só pode ser acessado pelo aluno que realizou o agendamento. Da mesma forma, o *endpoint* é similar aos anteriores, possibilita criação, busca, atualização e exclusão. Além de listar os horários disponíveis, de acordo com a data passada como parâmetro.

- **Criar**: Envia uma requisição HTTP POST para o *endpoint* de agendamentos com os dados do professor a ser criado.
- **Buscar Todos:** Envia uma requisição HTTP GET para o *endpoint* de experimentos sem nenhum parâmetro adicional.
- **Buscar Um:** Envia uma requisição HTTP GET para o *endpoint* de experimentos com o ID do professor desejado como parâmetro.

- **Atualizar:** Envia uma requisição HTTP PUT para o *endpoint* de experimentos com o ID do professor e os dados atualizados como parâmetros.
- **Excluir:** Envia uma requisição HTTP DELETE para o *endpoint* de experimentos com o ID do professor como parâmetro.
- **Listar horários disponíveis:** Envia uma requisição HTTP GET para o *endpoint* de agendamentos com a data e o ID do experimentos desejado como parâmetro e retorna a lista com os horários disponíveis.

5.2.2 Front-end

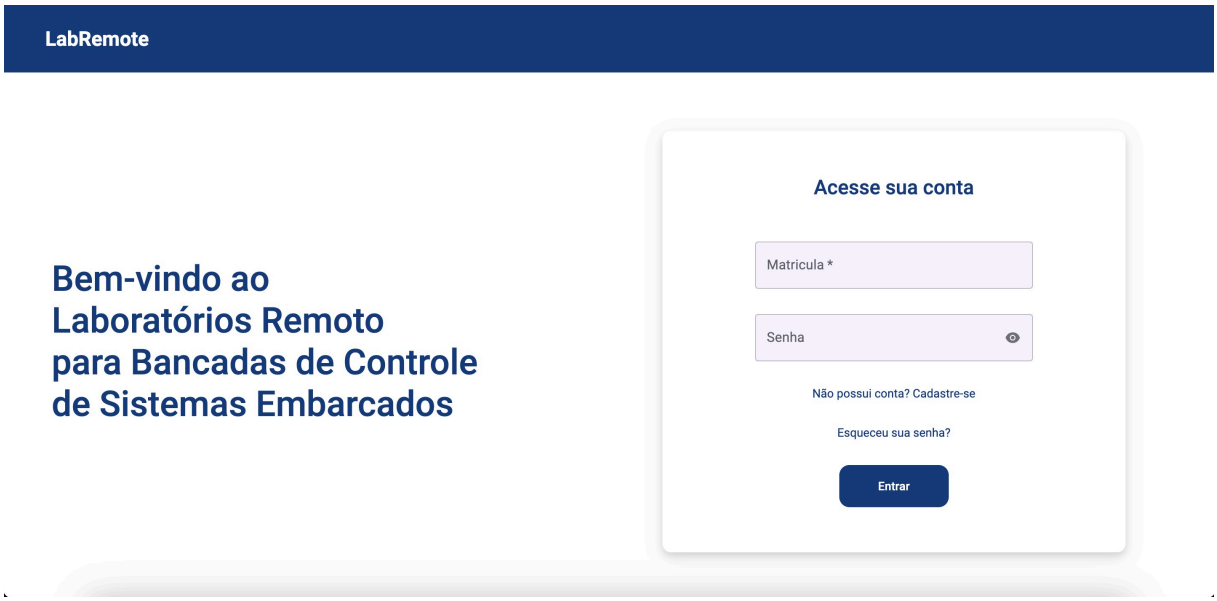
Front-end é a parte da aplicação que é responsável pela interface com o usuário. É onde a aparência e a interação da aplicação são definidas. Para a construção do *front-end* da aplicação foram utilizadas as tecnologias React, Axios e *React Components*. React é uma biblioteca JavaScript poderosa que permite a criação de interfaces de usuário de forma eficiente. Axios é uma biblioteca que facilita as requisições HTTP no *front-end*. Os *React Components* são os componentes visuais que compõem a interface do usuário e formam as telas que serão mostradas a seguir.

- ***Home/login***: É a primeira tela que o usuário irá visualizar, nela será possível realizar o login e acessar as informações da página inicial da aplicação.
- **Gerenciar alunos:** Nesta tela será possível gerenciar as informações dos alunos, incluindo adição, edição ou remoção de informações.
- **Gerenciar turmas:** Neste módulo será possível gerenciar as turmas, incluindo adição, edição e remoção de turmas.
- **Gerenciar experimentos:** Neste módulo será possível gerenciar as informações dos experimentos, incluindo adição, edição e remoção de experimentos.
- ***Home* logada/lista de experimentos:** Esta tela exibe a lista com os experimentos disponíveis na turma que o aluno está cadastrado.
- **Experimentos:** Esta tela permitirá ao usuário visualizar informações detalhadas sobre os experimentos, incluindo descrição, duração e objetivos.
- **Agendamento do experimento:** Neste módulo será possível o aluno agendar experimentos conforme os dias e a disponibilidade.
- ***Dashboard* do experimento:** Este é o painel de controle dos experimentos, onde serão exibidos gráficos e estatísticas sobre o desempenho dos experimentos realizados, permitindo uma análise mais detalhada.

5.2.2.1 *Home / Login*

O usuário acessa a aplicação e é direcionado para a tela *Home/Login*. Se já tiver uma conta, ele pode fazer login informando seu nome de usuário e senha. Caso contrário, ele deve solicitar ao professor seu acesso. Conforme mostrado na Figura 10.

Figura 10 – Tela home/login

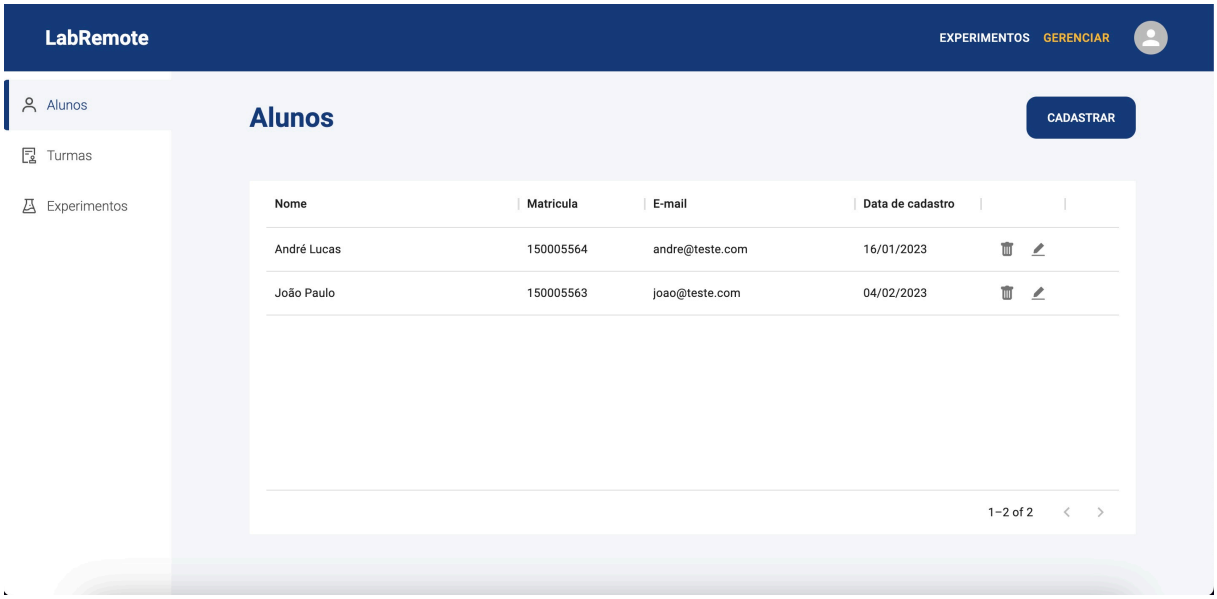


Fonte: Autor

5.2.2.2 Gerenciar alunos

Depois de fazer login, caso seja um administrador, o usuário pode acessar a tela de gerenciamento de alunos. Aqui, ele pode visualizar uma lista de todos os alunos, bem como adicionar, editar ou remover informações de alunos. Conforme mostrado na Figura 11.

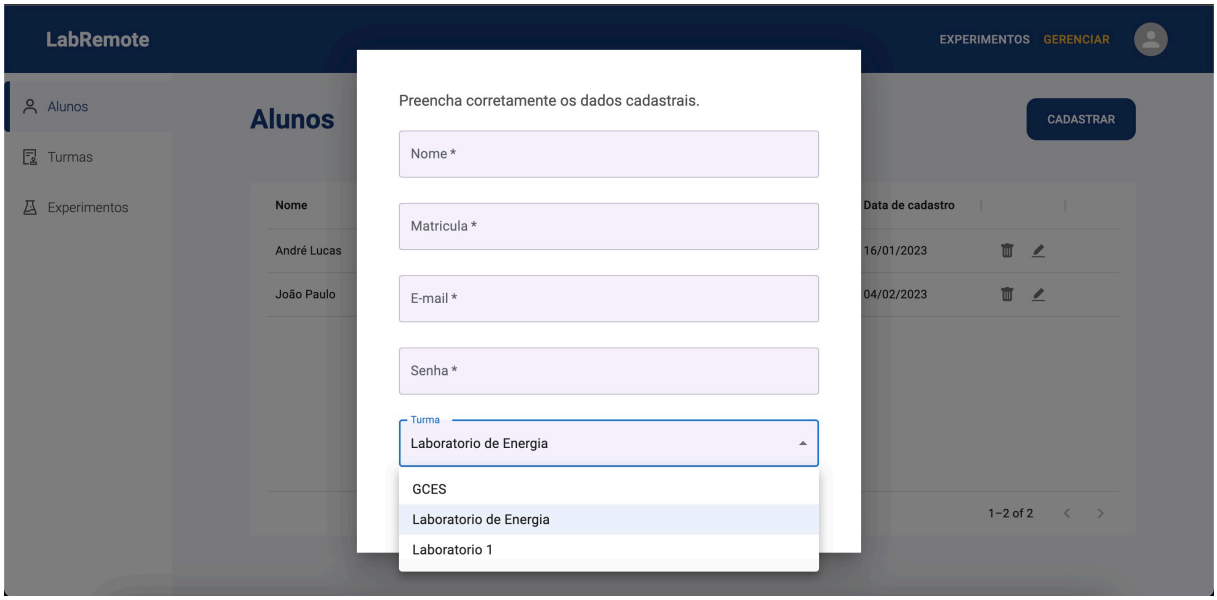
Figura 11 – Tela de gerenciar alunos



Fonte: Autor

A tela de cadastro de alunos é intuitiva e fácil de usar. A imagem abaixo ilustra como os professores podem adicionar novos alunos às suas turmas, fornecendo informações básicas como nome, e-mail, número de matrícula e definindo uma senha.

Figura 12 – Tela de cadastro alunos



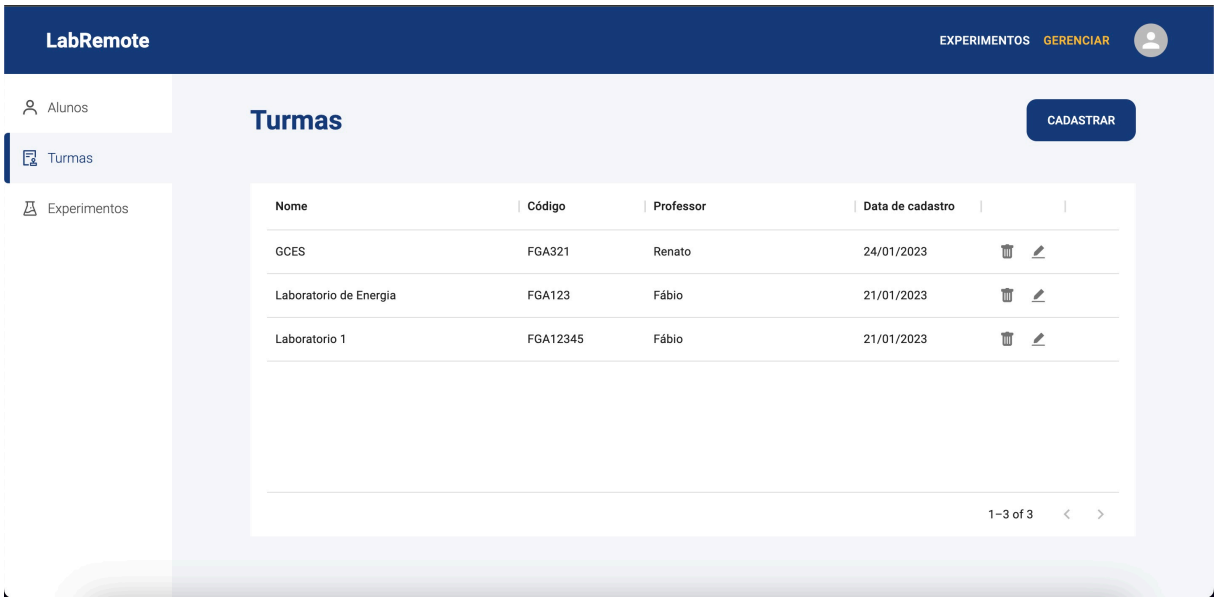
Fonte: Autor

A interface permite que os professores vejam uma lista de todos os alunos cadastrados, com opções para editar ou excluir as informações de cada um.

5.2.2.3 Gerenciar turmas

Da mesma forma, caso seja um administrador, a partir da tela de gerenciamento de turmas, o usuário pode visualizar uma lista de todas as turmas, bem como adicionar, editar ou remover informações de turmas. Conforme mostrado na Figura 13.

Figura 13 – Tela de gerenciar turmas

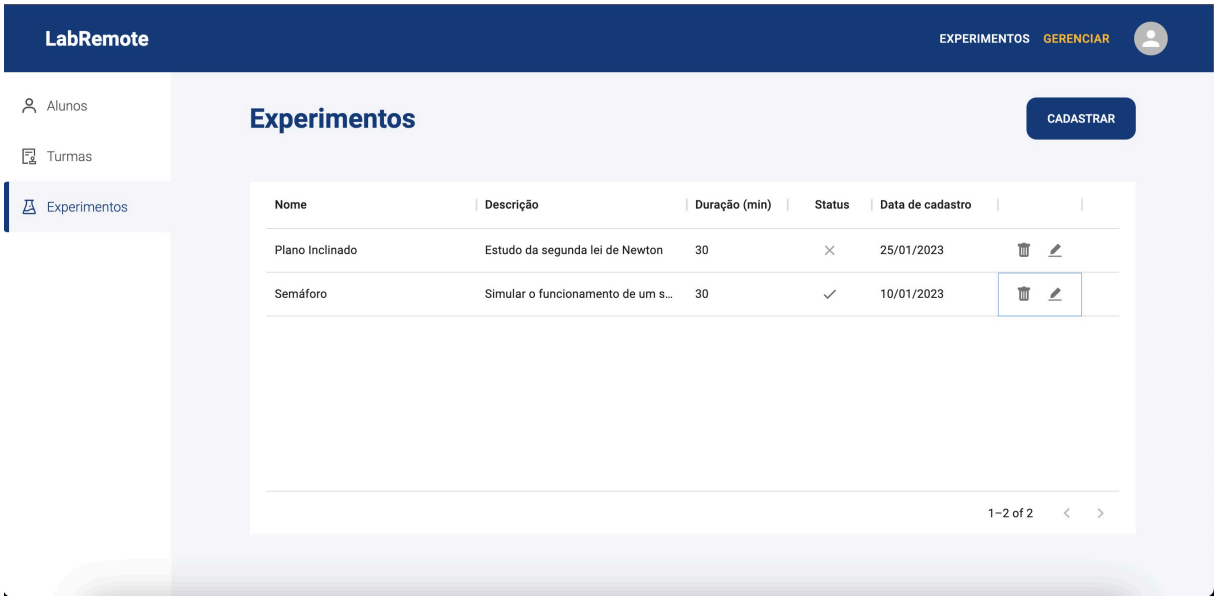


Fonte: Autor

5.2.2.4 Gerenciar experimentos

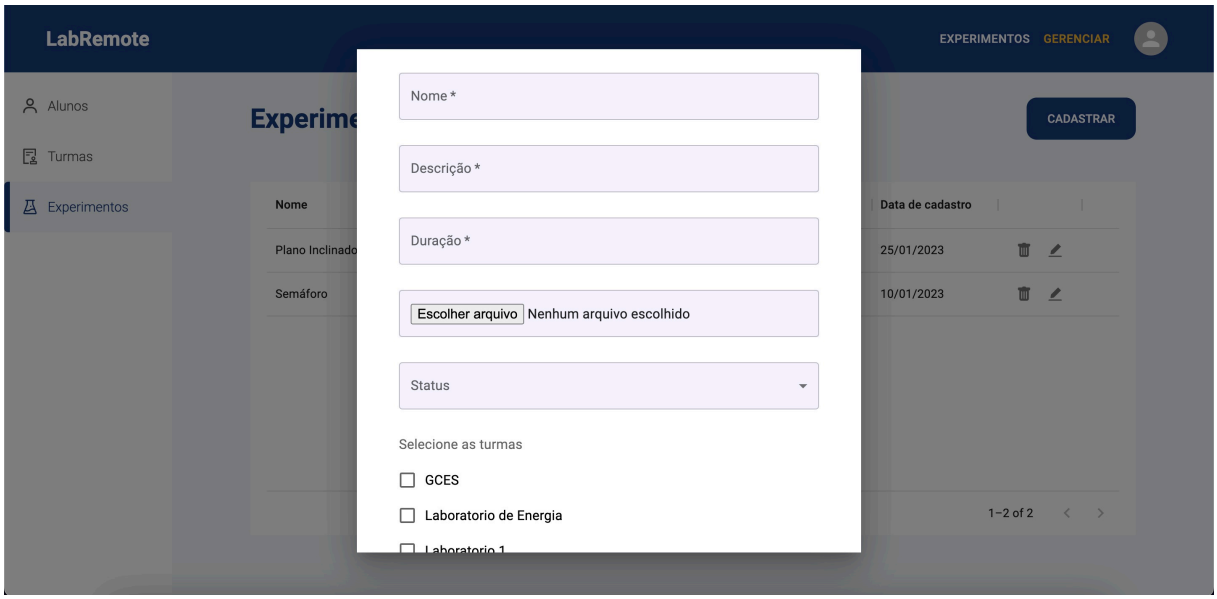
Caso seja um administrador, a tela de gerenciamento de experimentos permite ao usuário visualizar uma lista de todos os experimentos, bem como adicionar, editar ou remover informações sobre experimentos. Conforme mostrado na Figura 14.

Figura 14 – Tela de gerenciar experimentos



Fonte: Autor

Figura 15 – Tela de cadastro de experimentos



Fonte: Autor

Como exemplo na Figura 15, a tela para cadastro de experimentos é acessada pelos professores na interface da aplicação. Ela possibilita a inserção de informações importantes sobre o experimento, como uma imagem representativa e a turma vinculada. O processo de cadastro é intuitivo e fácil de ser realizado.

Além disso, o professor também pode adicionar através de um *input* o *iframe* do *dashboard*, que é uma ferramenta visual que permite aos alunos interagirem com o experimento e receberem *feedback* imediato. O *iframe* é integrado na tela de forma simples e

intuitiva, tornando o processo de criação de experimentos muito mais eficiente e agradável para os professores.

Em resumo, é uma ferramenta fundamental para o sucesso da aplicação, permitindo que os professores criem e gerenciem seus experimentos de forma intuitiva e eficiente, oferecendo aos alunos uma melhor experiência.

5.2.2.5 Home logada / Lista de experimentos

Depois de realizar o login, o usuário terá acesso a *home* logada ou lista de experimentos. Onde estarão listados todos os experimentos vinculados a turma no qual o aluno foi cadastrado. Conforme mostrado na Figura 16.

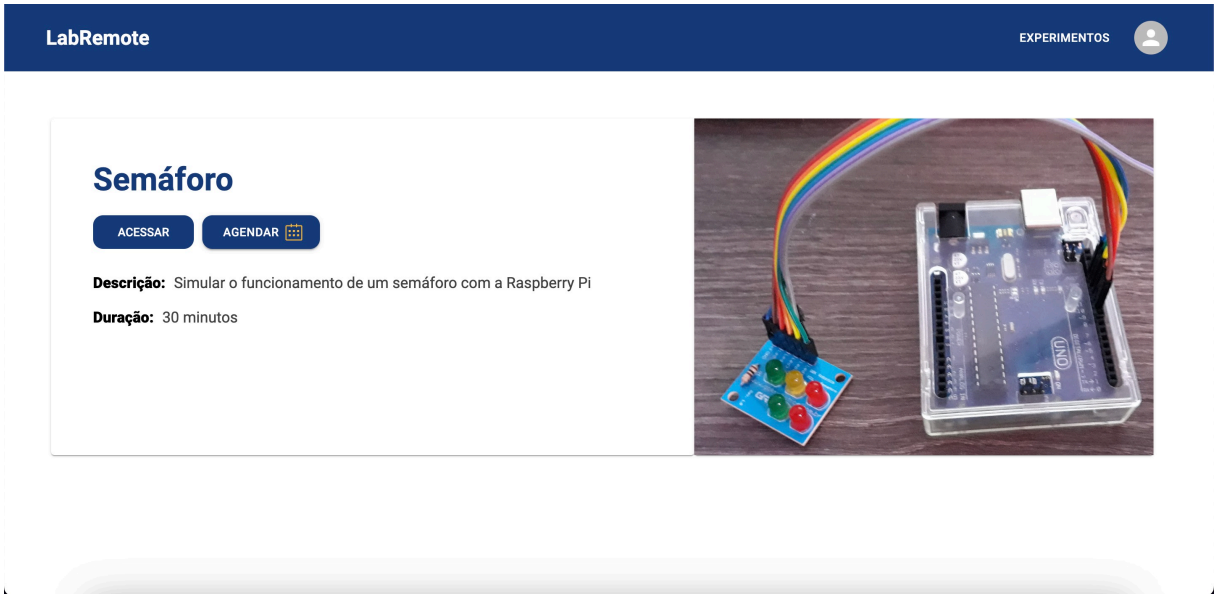
Figura 16 – Tela do experimento

Fonte: Autor

5.2.2.6 Experimento

Depois de clicar em um experimento específico na tela com a lista de experimentos, o usuário é direcionado para a tela do experimento, onde pode visualizar informações detalhadas, incluindo descrição, objetivo e material necessário. Conforme mostrado na Figura 17.

Figura 17 – Tela do experimento

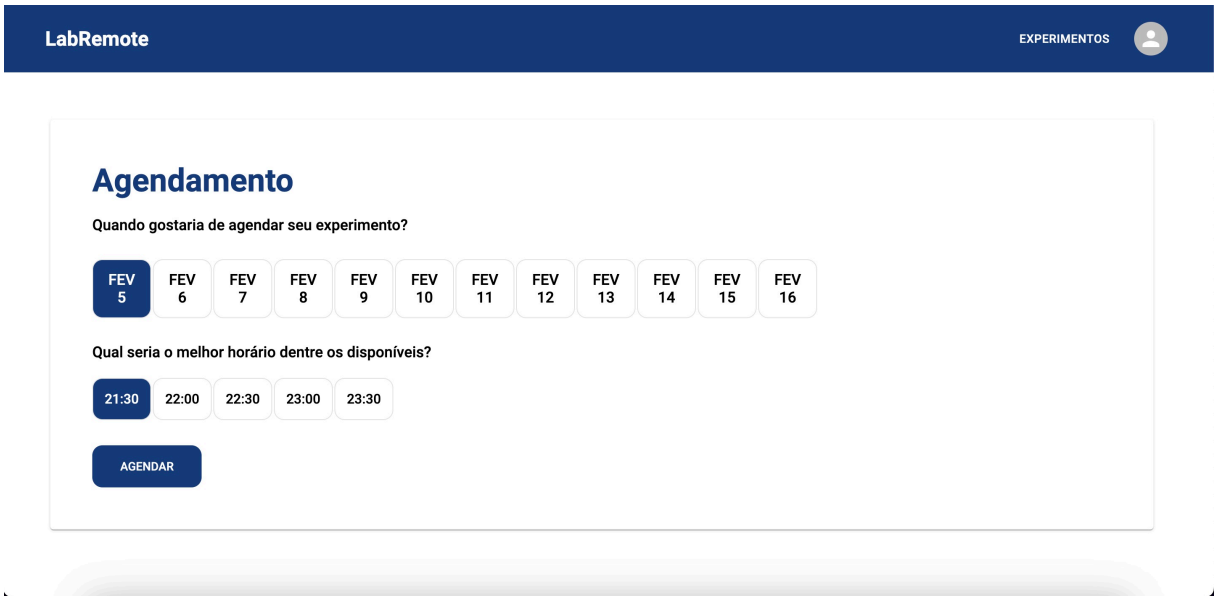


Fonte: Autor

5.2.2.7 Agendamento de experimento

A partir da tela do experimento, é possível acessar o agendamento, onde o usuário pode selecionar um horário e agendar um experimento para ela, assim alocando o aluno ao experimento agendado. Conforme mostrado na Figura 18.

Figura 18 – Tela de agendamento



Fonte: Autor

5.2.2.8 Dashboard de experimento

Depois de agendado, o usuário pode acessar a tela de *dashboard* do experimento, onde pode visualizar gráficos e estatísticas sobre o desempenho dos experimentos realizados, permitindo uma análise mais profunda e detalhada. Conforme mostrado na Figura 19.

Figura 19 – Tela de dashboard do experimento

Fonte: Autor

5.3 Qualidade

A qualidade do software é um fator crítico. É importante garantir que o código seja escrito de forma clara, eficiente e segura. Para ajudar a alcançar esse objetivo, existem diversas ferramentas que podem ser usadas. O ESLint, o Prettier e o SonarCloud são exemplos de ferramentas que ajudam a garantir a qualidade do software, ao corrigir erros comuns, garantir a consistência da formatação do código e identificar problemas de segurança e performance. Utilizando essas ferramentas, é possível desenvolver software de boa qualidade, o que pode ajudar a alcançar o sucesso do projeto de diferentes maneiras.

1. **ESLint:** É uma ferramenta de análise estática de código que ajuda a encontrar e corrigir padrões comuns de erros no código. Ele também fornece sugestões para escrever código mais limpo e consistente, o que ajuda a manter a qualidade do código ao longo do tempo.

2. **Prettier:** É uma ferramenta de formatação de código que ajuda a garantir a consistência na formatação do código. Isso é especialmente importante em projetos em equipe, onde vários desenvolvedores podem estar trabalhando no mesmo código.

Além disso, o Prettier ajuda a evitar debates desnecessários sobre a formatação do código.

3. **SonarCloud:** É uma ferramenta de análise de código que ajuda a encontrar problemas de segurança, performance, qualidade de código e outros problemas comuns. Ele fornece uma visão geral da qualidade do código e ajuda a identificar pontos de melhoria, o que pode levar a códigos mais seguros, eficientes e de alta qualidade.

Figura 20 – Painel do SonarCloud

Fonte: Autor

A Figura 20 mostra o painel da ferramenta SonarCloud, analisando a aplicação *back-end* e a *front-end*. A ferramenta avalia os repositórios em relação a métricas essenciais da qualidade de software, como segurança, manutenibilidade e boas práticas de codificação. O painel também apresenta gráficos e relatórios detalhados que mostram pontos fracos e oportunidades de melhoria no código.

Ao investir em ferramentas de qualidade de software, é possível desenvolver bons projetos, com códigos claros, eficientes e seguros. Por fim, é importante destacar que o uso dessas ferramentas é parte integrante do processo de desenvolvimento de software e pode ajudar a garantir o padrão a longo prazo do projeto.

6 Resultados

Com base nos objetivos apresentados na seção 1.1, na apresentação dos requisitos para o Laboratório Remoto na seção 5 e na implementação e disponibilização da aplicação ao final deste projeto, podemos afirmar que alcançamos o exito na realização e entrega.

Este estudo teve como objetivo investigar as tecnologias adequadas para o desenvolvimento de uma plataforma educacional de acesso remoto, com foco em bancadas de sistemas embarcados. Para alcançar esse objetivo, foram realizadas atividades de elicitação, modelagem e análise de requisitos para a plataforma. Como resultado, foi elaborado um sistema web eficiente, capaz de atender às necessidades específicas de ensino e aprendizagem em bancadas de sistemas embarcados, com recursos de agendamento e acesso remoto. Embora não tenha sido possível implementar toda a "plataforma remota"planejada, a solução desenvolvida representa um passo significativo em direção à efetivação do objetivo proposto. Recomenda-se que futuros trabalhos se concentrem na continuação do desenvolvimento da plataforma, de forma a aprimorar ainda mais a experiência educacional remota para os estudantes envolvidos.

O Quadro 2 apresenta as principais atividades executadas e avaliadas em relação a plataforma.

Tabela 2 – Atividades executadas

Requisito	
Permitir que o professor efetue o cadastro de turmas	√
Permitir que o professor efetue o cadastro de alunos	√
Permitir que o professor efetue o cadastro de experimentos	√
Permitir que usuários realizem login	√
Ordenar os agendamentos por disponibilidade	√
Permitir que usuários realizem agendamento de horário nos experimentos	√
Permitir que usuários realizem os experimentos	√
Apresentar aos usuários um dashboard embutido para manipulação do experimento	√

A ausência de algumas funcionalidades se deve à diversas razões, tais como a complexidade, e principalmente, a limitação de tempo. Embora tenha trabalhado para incluir o maior número possível de funcionalidades, algumas não puderam ser concluídas dentro do prazo. No entanto, as funcionalidades ausentes serão especificadas para possíveis trabalhos futuros.

7 Conclusão

Neste trabalho, foi destacada a importância do ensino prático na área de Sistemas Embarcados, sendo fundamental para a formação adequada dos futuros profissionais. A pandemia e a portaria nº 2.117 do Ministério da Educação também destacou a necessidade de alternativas aos laboratórios físicos. Para atender a essa necessidade, foi desenvolvida a plataforma para viabilizar a realização de experimentos práticos de Sistemas Embarcados de forma remota.

Ao longo deste projeto de conclusão de curso, que incluiu desde a revisão da literatura até implementação de um laboratório, foi criado um sistema que oferece uma *API* combinada com uma interface de fácil utilização. Esta ferramenta permite a criação e gestão de professores, alunos, turmas, experimentos e agendamentos.

O *back-end* foi desenvolvido em Typescript com NestJS. Ele é o centro da nossa aplicação, fornecendo uma estrutura modular que permitiu a criação de *endpoints* da *API* de maneira rápida e eficiente. Esses *endpoints* são os pontos de contato, onde a aplicação faz solicitações e recebe dados através de requisições HTTP.

Já o *front-end*, a interface intuitiva da nossa aplicação, foi criado usando o *framework React*. Ele é responsável por apresentar os dados de maneira clara e fácil de usar para o usuário final. As requisições da aplicação são realizadas através da *API* fornecida pelo *back-end*, que retorna os dados necessários para preencher as informações na interface. A combinação do *back-end* robusto e da interface intuitiva do *React* torna a nossa aplicação fácil de usar e escalável.

Como resultado, espera-se que esta plataforma contribua para o processo de aprendizagem dos estudantes de graduação e pós-graduação e os ajude a alcançar seus objetivos.

7.1 Trabalhos futuros

Considerando as pendências resultantes da falta de atendimento a todos os requisitos elicitados neste projeto, bem como as oportunidades de melhoria no sistema, os seguintes pontos podem ser propostos como possíveis tarefas futuras:

- Requisitos não cumpridos
  - **RF09:** *Streaming* de câmera com o experimento na bancada em tempo real
  - **RF10:** Permitir aos usuários acesso à dispositivo USB remoto
  - **RF11:** Acesso via SSH

- Melhorias sugeridas
  - Criação de um *bucket* S3 para armazenar imagens, o que facilitaria a organização e a recuperação de imagens no futuro. Além da configuração de acesso seguro e controlado ao *bucket*, garantindo a privacidade e segurança das imagens armazenadas.
  - Desenvolvimento do processo de *deploy* da aplicação, possibilitando implantação de *pipelines* automatizados de integração e entrega contínua, o que permitiria a realização de *deploys* de forma mais rápida, segura e controlada.

Referências

AKTAN, B. et al. Distance learning applied to control engineering laboratories. *IEEE Transactions on Education*, v. 39, n. 3, p. 320–326, 1996. Citado na página 12.

AMARAL, C. *Streaming de vídeo sob demanda: tudo o que você precisa saber*. 2020. Disponível em: <https://k2ponto.com.br/blog/streaming-de-video-sob-demanda-tudo-o-que-voce-precisa-saber/>. Citado na página 22.

BOTH, D. Remote access with ssh. In: _____. *Using and Administering Linux: Volume 3: Zero to SysAdmin: Network Services*. Berkeley, CA: Apress, 2020. p. 67–86. ISBN 978-1-4842-5485-1. Disponível em: <https://doi.org/10.1007/978-1-4842-5485-1_5>. Citado na página 20.

CARNEGIE, M. U. The virtual lab: Engineering the future. In: . [s.n.], 2014. Disponível em: <http://users.ece.cmu.edu/~stancil/virtual-lab/virtual-lab.html>. Citado na página 19.

FREITAS, F. M.; SILVA, J. H. C. d. WebTCC - Plataforma para controle e acompanhamento de trabalhos finais de curso. 2020. Disponível em: <https://repositorio.ifsc.edu.br/handle/123456789/2457>. Citado na página 21.

GOIS, L. A. d. Implementação dos serviços de comunicação e monitoração em um ambiente para desenvolvimento de sistemas distribuídos. ago. 2002. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/300>. Citado na página 22.

HEATH, S. *Embedded systems design*. [S.l.]: Elsevier, 2002. Citado na página 18.

HODGES, C. B. et al. The difference between emergency remote teaching and online learning. Educause, 2020. Citado na página 12.

HOLLY, O. Remote firmware update management of embedded devices. *Department of science and innovations, FEI STU in Bratislava*, 2020. Disponível em: <https://www.zahorack.com/wp-content/uploads/2020/08/HOLLY_SVOC_2020.pdf>. Citado na página 23.

JULURI, P.; TAMARAPALLI, V.; MEDHI, D. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys Tutorials*, v. 18, n. 1, p. 401–418, 2016. Citado na página 22.

JURKOVIC, G.; SRUK, V. Remote firmware update for constrained embedded systems. In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. [S.l.: s.n.], 2014. p. 1019–1023. Citado na página 23.

KALUZA, M.; TROSKOT, K.; VUKELIC, B. COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT. *Zbornik Veleučilišta u Rijeci*, v. 6, n. 1, p. 261–282, maio 2018. ISSN 1848-1299, 1849-1723. Disponível em: <https://hrcak.srce.hr/199922>. Citado na página 22.

KARAKASIDIS, T. Virtual and remote labs in higher education distance learning of physical and engineering sciences. In: *2013 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.: s.n.], 2013. p. 798–807. Citado 2 vezes nas páginas 13 e 19.

MATOS, S. et al. Bancada de transferência de calor para práticas de teoria de controle. In: . [S.l.: s.n.], 2019. Citado na página 18.

MEDEIROS, R. B. d. *Laboratório Remoto Baseado em FPGA aplicado NAS disciplinas de prática de eletrônica digital 1 E 2 da faculdade Unb Gama*. 2018. Disponível em: <https://bdm.unb.br/handle/10483/23653>. Citado na página 25.

MOHAMMED, A. K.; ZOGHBY, H. M. E.; ELMESALAWY, M. M. Remote controlled laboratory experiments for engineering education in the post-covid-19 era: Concept and example. In: IEEE. *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*. [S.l.], 2020. p. 629–634. Citado 2 vezes nas páginas 12 e 13.

Node.js. *Node.js*. 2022. Disponível em: <https://nodejs.org/en/>. Citado na página 27.

PARREIRA, P. F. P. et al. Laboratório virtual de sistemas embarcados: uma abordagem de ensino remoto usando softwares gratuitos. *Simpósio Brasileiro de Automação Inteligente - SBAI*, v. 1, n. 1, out. 2021. ISSN 2175-8905. Disponível em: <https://www.sba.org.br/open_journal_systems/index.php/sbai/article/view/2843>. Citado na página 18.

REACT – Uma biblioteca JavaScript para criar interfaces de usuário. 2022. Disponível em: <https://pt-br.reactjs.org/>. Citado na página 28.

RELLE. 2023. Disponível em: <http://relle.ufsc.br/about>. Citado na página 24.

SANTOS, M. P. d. LABVAE : uma solução para experimentação de aprendizagem eletrônica. set. 2014. Disponível em: <http://ri.ufs.br/jspui/handle/riufs/3350>. Citado na página 19.

SCHWABER, K.; SUTHERLAND, J. The scrum guide. *Scrumguides. Org*, p. 17, 2020. Citado na página 29.

SILVA, J. B. d. et al. Laboratórios Remotos como Alternativa para Atividades Práticas em Cursos na Modalidade EAD. *EaD em Foco*, v. 10, n. 2, jul. 2020. ISSN 2177-8310. Disponível em: <https://eademfoco.cecierj.edu.br/index.php/Revista/article/view/942>. Citado na página 19.

TANENBAUM, A. S. *Redes de computadoras*. [S.l.]: Pearson educación, 2003. Citado na página 21.

TIDESTROM, J. *Investigation into low latency live video streaming performance of WebRTC*. [s.n.], 2019. Disponível em: <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-249446>. Citado na página 23.

TYPESCRIPT. 2022. Disponível em: <https://www.typescriptlang.org/>. Citado na página 28.

UNIAO, I. N. d. B. Diário Oficial da. *PORTARIA Nº 2.117, DE 6 DE DEZEMBRO DE 2019 - DOU - Imprensa Nacional*. 2019. Disponível em: <https://www.in.gov.br/web/dou>. Citado na página 12.

WEDDELL, S. J.; BONES, P. J.; WAREING, N. M. A remote laboratory for learning embedded systems & control. In: . [S.l.: s.n.], 2014. Citado na página 19.