CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM DESENVOLVIMENTO SOFTWARE MULTIPLATFORMA

GUILHERME LUÍS RODRIGUES SILVA
JORGE LUIZ PATROCÍNIO DOS SANTOS
YAGO RAPHAEL DE MELO MOURO
RAFAEL VICTOR REDOVAL DE SOUSA

PLATAFORMA PARA MONITORAMENTO E ANÁLISE DE TEMPERATURA E UMIDADE

FRANCA/SP
2025

1 INTRODUÇÃO

A introdução deste projeto visa apresentar de forma clara e objetiva a proposta de criação de um dashboard que realiza a monitorização e exibição dos dados coletados por meio de um Arduino físico, equipado com sensor de umidade e temperatura. Trata-se de uma solução que integra dispositivos de Internet das Coisas (IoT) a uma API REST, desenvolvida em Node.js e Express, permitindo a captação, o armazenamento e a disponibilização dos dados para visualização em tempo real, tanto na plataforma web quanto na versão mobile desenvolvida em React Native.

O sistema proposto não se limita a um único dispositivo, podendo agregar múltiplos Arduinos, o que viabiliza a ampliação do monitoramento para diversos ambientes e aplicações, como o controle ambiental em áreas industriais ou laboratórios de pesquisa. A transmissão dos dados é realizada via conexão Wi-Fi, garantindo a agilidade e confiabilidade na comunicação entre o hardware e a camada de software. Além disso, o uso do MongoDB como banco de dados NoSQL assegura escalabilidade e flexibilidade na gestão das informações.

Esta solução se destaca pela utilização de tecnologias modernas e uma arquitetura robusta que facilita a integração de diferentes componentes. O dashboard foi concebido com foco na usabilidade e na apresentação intuitiva dos dados, sendo capaz de gerar gráficos interativos que permitem uma análise aprofundada dos valores coletados. A implementação do Swagger com OpenAPI 3.0 para a documentação da API contribui para a transparência e o facilitamento das futuras manutenções e evoluções do sistema.

Portanto, o projeto se configura como uma ferramenta inovadora, que alia a precisão dos dados coletados por sensores físicos à eficiência das tecnologias web e mobile, proporcionando uma visão clara e atualizada das condições monitoradas. Essa integração entre hardware e software não apenas otimiza o gerenciamento dos dados, mas também potencializa a tomada de decisões com base em informações precisas e em tempo real.

2 VIABILIDADE DO PROJETO

A viabilidade do projeto de desenvolvimento do dashboard de monitoramento está intrinsecamente ligada à sua capacidade de atender a uma demanda real por

soluções de monitoramento em tempo real, de maneira eficiente e com alto valor agregado aos seus usuários. A análise de viabilidade permite comprovar a utilidade do sistema proposto, identificar oportunidades para aprimoramento, avaliar o perfil dos usuários e mensurar os recursos disponíveis para a implementação e manutenção da solução.

Neste cenário, a proposta de valor do projeto reside na criação de uma plataforma digital que integra de forma harmoniosa tecnologias de hardware e software, possibilitando a coleta, o armazenamento e a visualização de dados oriundos de sensores de umidade e temperatura instalados em Arduinos. Essa integração não só garante a precisão na coleta dos dados, mas também permite o acesso às informações de maneira prática e intuitiva, seja por meio da versão web ou do aplicativo mobile.

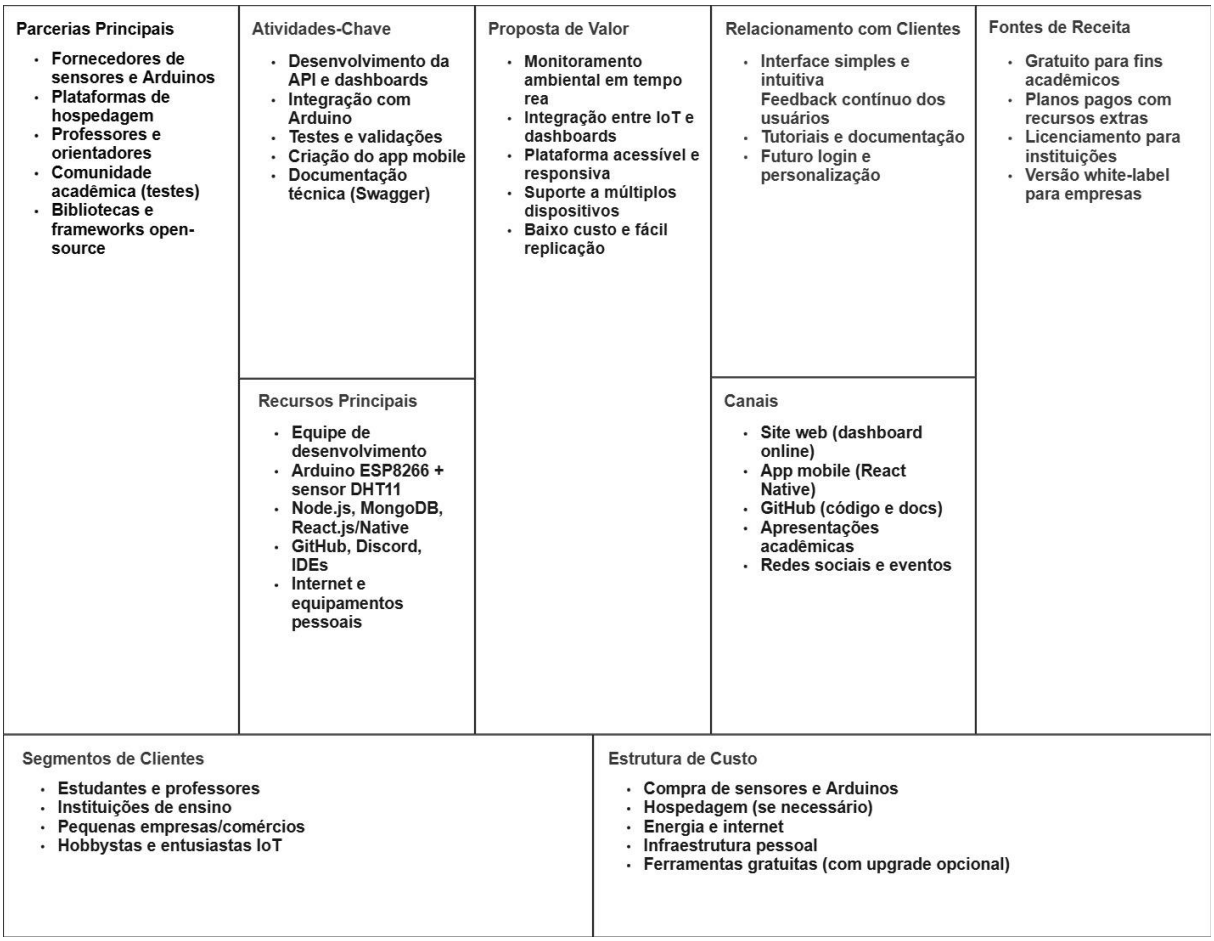
Além disso, o projeto adota uma arquitetura robusta, utilizando o Node.js e o Express para a construção da API, o MongoDB para a gestão flexível e escalável dos dados, e frameworks modernos como React.js e React Native para a criação de interfaces responsivas e de fácil navegação. Essa combinação de tecnologias assegura que a plataforma seja não apenas eficiente e de alta performance, mas também adaptável a futuras expansões, podendo suportar a integração de múltiplos dispositivos simultaneamente.

A viabilidade técnica e operacional do sistema demonstra que, com os recursos atuais e as tecnologias escolhidas, é possível desenvolver uma solução que atenda tanto às necessidades imediatas quanto às demandas de evolução do ambiente monitorado. A utilização de metodologias ágeis durante o desenvolvimento também permite a constante validação e refinamento dos requisitos, garantindo alinhamento com as expectativas dos usuários e stakeholders.

Por fim, a aplicação da ferramenta Business Model Canvas (BMC) ajudará a estruturar e validar o modelo de negócio, evidenciando os canais de entrega, a proposta de valor, os segmentos de clientes e as principais parcerias estratégicas. Dessa forma, a viabilidade do projeto se manifesta não apenas do ponto de vista tecnológico, mas também pela sua capacidade de gerar valor e sustentação no mercado, reafirmando a pertinência e a relevância desta solução inovadora.

2.1 CANVAS DE NEGÓCIO

Figura 1 – Business Model Canvas – BMC



2.1.1 PARCERIAS PRINCIPAIS

As parcerias principais são os apoios externos e recursos estratégicos que viabilizam o desenvolvimento e funcionamento do projeto:

- Fornecedores de componentes eletrônicos: Fontes de aquisição de Arduinos, sensores DHT11 e módulos ESP8266 utilizados para coleta dos dados ambientais.
- Serviços de hospedagem: A ser definido, buscamos plataformas com bom desempenho, suporte a back-end em Node.js e opção gratuita ou de baixo custo.
- Comunidade acadêmica: Apoio de colegas da Fatec para realização de testes, feedback sobre usabilidade e sugestões de melhoria contínua.
- Orientadores e professores: Suporte metodológico e técnico, com contribuições estratégicas durante todas as fases do projeto.

- Bibliotecas e frameworks open-source: Utilização de ferramentas gratuitas como React.js, React Native, Express, entre outras, que reduzem custos e agilizam o desenvolvimento.

2.1.2 ATIVIDADES-CHAVE

As atividades-chave representam as principais ações que garantirão a entrega e qualidade do projeto:

- Desenvolvimento da API REST: Implementação de uma API robusta em Node.js com integração ao banco de dados MongoDB e recebimento de dados dos sensores.
- Desenvolvimento do dashboard web: Construção de uma interface interativa com React.js para exibição de gráficos e dados em tempo real.
- Criação do aplicativo mobile: Desenvolvimento do app em React Native, com foco em acessibilidade e leitura prática dos dados.
- Integração com os dispositivos físicos: Programação e configuração dos Arduinos para envio contínuo de dados via Wi-Fi.
- Documentação técnica: Utilização do Swagger com OpenAPI 3.0 para gerar documentação clara da API e facilitar futuras manutenções.
- Testes e validações: Garantia de performance, estabilidade, segurança dos dados e boa experiência de uso.

2.1.3 PROPOSTA DE VALOR

A proposta de valor é o diferencial que torna o projeto útil, único e relevante para os usuários:

- Fornece uma solução completa para o monitoramento ambiental por sensores físicos, integrando hardware e software em um único sistema.
- Permite visualização em tempo real de dados de temperatura e umidade por meio de gráficos dinâmicos e acessíveis.
- Interface moderna, responsiva e intuitiva, tanto na versão web quanto mobile.
- Escalável para múltiplos dispositivos, com baixo custo de implementação e fácil replicação para diferentes ambientes.
- Tecnologia acessível com uso de componentes de baixo custo e software livre.

2.1.4 RELACIONAMENTO COM CLIENTES

Formas de interação e suporte ao usuário:

- Interface clara e objetiva, sem a necessidade de conhecimento técnico para uso básico.
- Feedback contínuo: O projeto poderá evoluir conforme sugestões de uso real dos usuários e testes acadêmicos.
- Tutoriais e documentação: Suporte ao uso da aplicação por meio de guias explicativos e documentação da API.
- Controle por usuário (futuramente): Possibilidade de personalização e autenticação individual conforme evolução do sistema.

2.1.5 FONTES DE RECEITA

Mesmo sendo inicialmente um projeto acadêmico, há possibilidade de monetização futura:

- Versão gratuita com funcionalidades básicas para usuários comuns e estudantes.
- Planos pagos com recursos adicionais, como monitoramento avançado, histórico detalhado, alarmes de notificação e integração com serviços externos.
- Comercialização como solução white-label para empresas que necessitam de monitoramento local ou remoto.
- Licenciamento para escolas, laboratórios e pequenas indústrias com foco em controle ambiental interno.

2.1.6 RECURSOS PRINCIPAIS

Recursos indispensáveis para o desenvolvimento e funcionamento do projeto:

- Equipe de desenvolvimento: Guilherme Luís Rodrigues Silva, Jorge, Yago Mouro e Rafael.
- Dispositivos físicos: Arduino NodeMCU ESP8266, sensores DHT11 e acessórios para prototipagem.

- Tecnologias de software: Node.js, Express, React.js, React Native, MongoDB, Jest, Redux, Swagger.
- Ferramentas de apoio: GitHub para versionamento, Figma ou Canva para prototipação e Discord para comunicação em equipe.
- Serviços online: Plataformas de hospedagem para disponibilização da API e do frontend.

2.1.7 CANAIS

Meios utilizados para disponibilizar, divulgar e manter o projeto:

- Site web com acesso ao dashboard em navegadores como Google Chrome.
- Aplicativo mobile disponível para testes em Android. GitHub como repositório público de código-fonte e documentação técnica.
- Divulgação em redes sociais, apresentações acadêmicas, feiras tecnológicas e portfólios dos integrantes.
- Demonstrações práticas do funcionamento com Arduinos em sala de aula e eventos.

2.1.8 SEGMENTOS DE CLIENTES

Público-alvo e beneficiários diretos da solução:

- Estudantes e pesquisadores que precisam monitorar variáveis ambientais em tempo real.
- Instituições de ensino técnico e superior que utilizam sensores em práticas laboratoriais.
- Pequenas empresas ou comércios que desejam controlar temperatura e umidade de ambientes específicos (estoques, câmaras frias etc.).
- Amantes de tecnologia, hobbystas e desenvolvedores que buscam uma solução IoT acessível para integrar sensores a dashboards visuais.

2.1.9 ESTRUTURA DE CUSTO

Custos diretos e indiretos envolvidos no projeto:

- Aquisição de Arduinos, sensores DHT11 e módulos Wi-Fi (ESP8266).
- Eventual contratação de plano de hospedagem para manter a aplicação online.

- Equipamentos pessoais utilizados no desenvolvimento (computadores, smartphones etc.).
- Despesas com internet e energia elétrica durante o período de desenvolvimento e testes.
- Todas as bibliotecas utilizadas são open-source, mas há possibilidade de uso de ferramentas pagas em versões futuras.

3 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos é uma etapa crucial no processo de desenvolvimento de sistemas, pois permite identificar de forma clara as necessidades dos usuários e os objetivos do projeto, garantindo que a solução final atenda de maneira eficiente ao problema proposto. É por meio dessa fase que se estabelece o escopo funcional e não funcional do sistema, reduzindo riscos de falhas, retrabalho e desalinhamento entre as expectativas dos usuários e o produto entregue.

No contexto do projeto de desenvolvimento de um dashboard para monitoramento de temperatura e umidade com integração a dispositivos Arduino, o levantamento de requisitos foi essencial para compreender como os dados deveriam ser tratados, armazenados e apresentados, além de entender quais funcionalidades seriam indispensáveis para o uso prático e contínuo do sistema. A definição dos requisitos também possibilitou identificar aspectos técnicos relevantes, como o formato de comunicação entre o hardware e o backend, a estrutura de dados adequada para o banco de dados NoSQL, e os requisitos de interface para uma boa experiência do usuário tanto na versão web quanto mobile.

Além disso, a análise dos requisitos permitiu antecipar preocupações com a escalabilidade do sistema, já que este deverá estar apto a lidar com múltiplos dispositivos enviando dados simultaneamente, mantendo a integridade das informações e a performance na exibição gráfica dos dados. Portanto, esse processo se configurou como um pilar estratégico para o desenvolvimento da solução, servindo de base sólida para as demais etapas do projeto e assegurando que a entrega final seja funcional, confiável e aderente às necessidades reais do seu público-alvo.

3.1 ELICITAÇÃO E ESPECIFICAÇÃO DOS REQUISITOS

A elicitação de requisitos é uma etapa essencial do processo de desenvolvimento de software, pois é nela que se busca compreender profundamente as necessidades dos usuários e transformá-las em funcionalidades práticas e objetivas. Para o projeto do dashboard de monitoramento de temperatura e umidade, a elicitação foi realizada por meio de conversas informais com estudantes e professores da área de tecnologia, bem como por meio da própria análise técnica das demandas comuns em soluções que envolvem Internet das Coisas (IoT) e visualização de dados em tempo real.

Durante esse processo, foi possível identificar os elementos indispensáveis para garantir o bom funcionamento do sistema, considerando aspectos como a recepção contínua de dados de sensores físicos, a visualização clara e responsiva desses dados, o suporte à expansão do sistema para múltiplos dispositivos Arduino, e a necessidade de personalização visual que facilite a interpretação dos dados de forma ágil. Um dos requisitos mais citados foi a importância de o sistema funcionar tanto em dispositivos desktop quanto em smartphones, o que reforçou a necessidade de uma versão mobile adaptada e funcional.

Além disso, a proposta de permitir que o dashboard sinalize de forma visual os níveis de temperatura e umidade — utilizando cores como verde, amarelo e vermelho — foi apontada como um diferencial prático para facilitar o entendimento imediato do estado do ambiente monitorado. A ausência de login também foi definida estrategicamente para simplificar o acesso à aplicação, considerando que o sistema não precisa individualizar usuários ou armazenar dados sensíveis.

Com base nessas necessidades, foram elaboradas as seguintes histórias de usuário:

3.1.1 MONITORAMENTO EM TEMPO REAL

- Como usuário, desejo visualizar em tempo real os dados de temperatura e umidade enviados por sensores, para acompanhar as condições do ambiente.
- Como usuário, desejo que os dados sejam atualizados automaticamente no dashboard sem precisar recarregar a página.

3.1.2 SUPORTE A MÚLTIPLOS DISPOSITIVOS

- Como usuário, desejo que o sistema aceite múltiplos Arduinos conectados, para monitorar diferentes locais em um único ambiente de visualização.
- Como usuário, desejo poder identificar de qual Arduino os dados estão vindo, para saber qual ambiente está sendo monitorado.

3.1.3 SINALIZAÇÃO VISUAL POR CORES

- Como usuário, desejo que o dashboard utilize cores para representar os níveis de temperatura e umidade, para facilitar a identificação de situações críticas (vermelho), alerta (amarelo) e normalidade (verde).

3.1.4 ACESSO MOBILE E USABILIDADE

- Como usuário, desejo acessar o sistema por meio do celular com a mesma eficiência da versão web, para acompanhar os dados mesmo quando estiver em movimento.
- Como usuário, desejo que a interface seja responsiva, adaptando-se automaticamente ao tamanho da tela do dispositivo utilizado.

3.1.5 HISTÓRICO E ANÁLISE DE DADOS

- Como usuário, desejo visualizar dados anteriores por meio de gráficos para analisar a variação de temperatura e umidade ao longo do tempo.
- Como usuário, desejo selecionar o intervalo de tempo que quero visualizar nos gráficos, para facilitar a análise por período.

3.1.6 PERFORMANCE E ESTABILIDADE

- Como usuário, desejo que o sistema carregue rapidamente e funcione de forma estável, mesmo com vários Arduinos enviando dados simultaneamente.
- Como usuário, desejo que o sistema funcione sem necessidade de autenticação, acessando o dashboard diretamente.

3.1.7 PERSONALIZAÇÃO DA INTERFACE

- Como usuário, desejo poder alterar o intervalo de atualização automática dos dados, para ajustar conforme a minha necessidade.
- Como usuário, desejo personalizar a visualização dos gráficos, escolhendo entre diferentes tipos (linha, barra etc.).

3.2. BPMN

As Figuras 2 a 4 apresentam os diagramas de BPMN, que consistem na modelagem dos processos envolvidos na coleta, envio, tratamento e visualização dos dados de temperatura e umidade. A partir desses diagramas, é possível compreender o fluxo completo de comunicação entre os dispositivos físicos (Arduinos), a API responsável pelo processamento dos dados, o banco de dados e a interface de visualização acessada pelos usuários. Essa modelagem contribui para uma visão clara dos processos que serão automatizados pelo sistema, evidenciando a integração entre hardware e software no monitoramento ambiental em tempo real.

Figura 2 – Diagrama de BPMN do Arduino

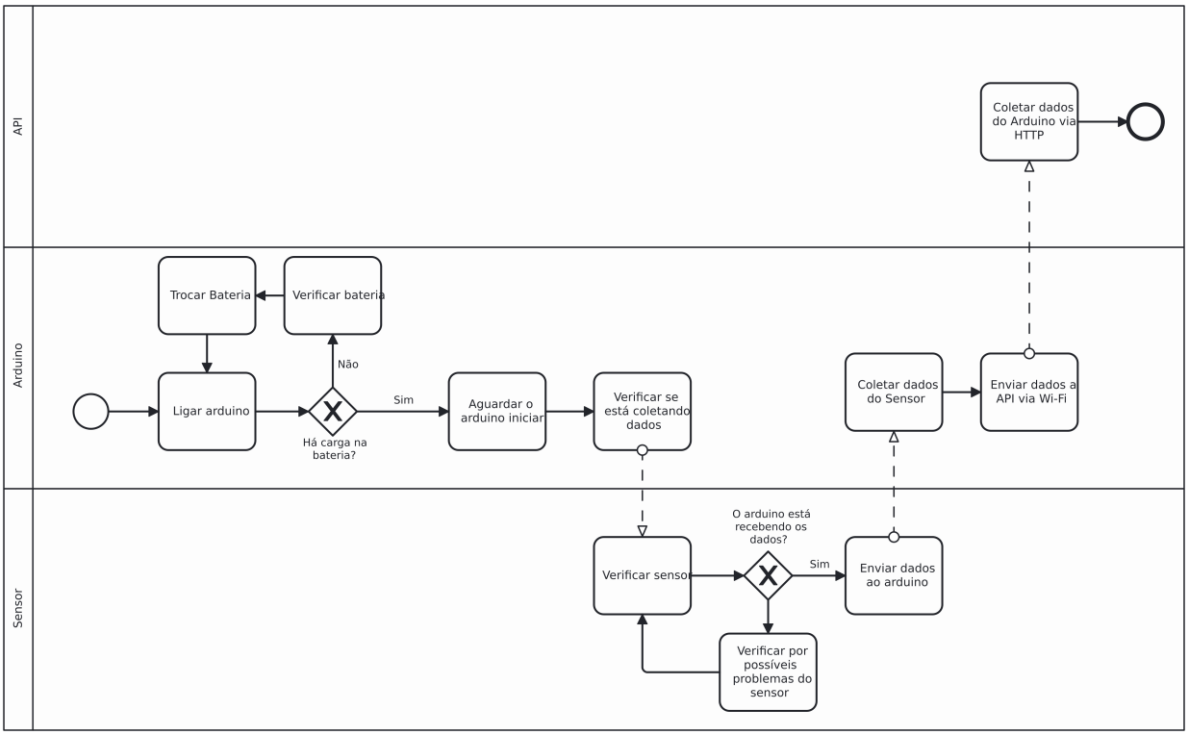


Figura 3 – Diagrama de BPMN de Back-End

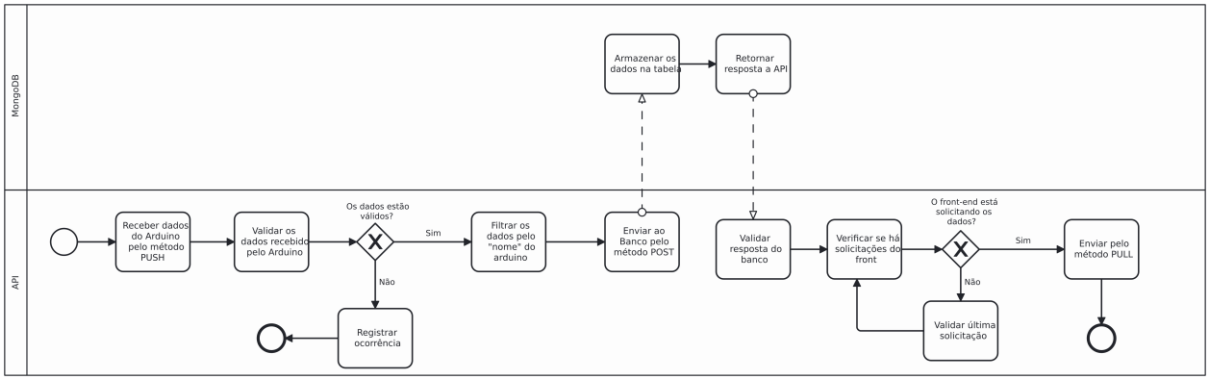
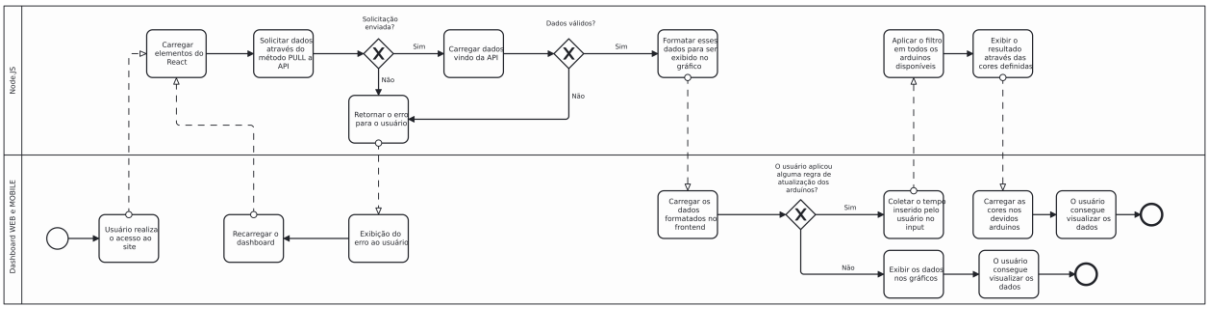


Figura 4 – Diagrama de BPMN de Front-End



3.3 REQUISITOS FUNCIONAIS

Os Requisitos Funcionais são um componente essencial do processo de desenvolvimento de software. Eles consistem na descrição detalhada das funcionalidades e recursos que o sistema deve fornecer para atender às necessidades dos usuários finais. Esses requisitos definem o que o sistema deve fazer, abrangendo o comportamento da aplicação, suas interações e fluxos esperados. Para o projeto do dashboard de monitoramento ambiental, os requisitos foram elaborados a partir do levantamento e análise das necessidades identificadas durante a fase de elicitação. Abaixo, no Quadro 1, estão especificados os requisitos funcionais do sistema:

Quadro 1 – Requisitos Funcionais do sistema

RF001-Receber Dados dos Dispositivos	Categoria: ( ) Oculto (X)Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve receber dados de temperatura e umidade enviados por Arduinos via conexão Wi-Fi, utilizando requisições HTTP.		
RF002-Armazenar Dados no Banco de Dados	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve armazenar os dados recebidos dos sensores em um banco de dados		

NoSQL (MongoDB), associando-os ao respectivo dispositivo.		
RF003-Exibir Dados em Tempo Real no Dashboard Web	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve exibir, por meio de gráficos interativos, os dados recebidos em tempo real na interface web.		
RF004-Exibir Dados no Aplicativo Mobile	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve disponibilizar os dados de monitoramento em uma versão mobile acessível e responsiva.		
RF005-Classificar Valores com Cores Visuais	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve classificar os dados de temperatura e umidade com cores visuais (verde, amarelo ou vermelho), de acordo com os limites definidos.		
RF006-Suportar Múltiplos Dispositivos	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve aceitar e distinguir dados provenientes de diferentes dispositivos Arduino conectados simultaneamente.		
RF007-Atualização Automática dos Dados	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O dashboard deve atualizar os dados automaticamente sem necessidade de recarregar a página.		
RF008-Visualizar Histórico por Período	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário selecione um intervalo de tempo para visualizar o histórico de dados em gráficos.		
RF009-Configurar Intervalo de Atualização	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve oferecer uma opção para o usuário definir o intervalo de atualização automática dos dados no dashboard.		

3.4 REQUISITOS NÃO FUNCIONAIS

Os Requisitos Não Funcionais são outra categoria importante de requisitos de software, que complementam os Requisitos Funcionais. Eles descrevem os atributos do sistema que não estão diretamente relacionados às funcionalidades, mas que são essenciais para o desempenho adequado e para a experiência do usuário.

Esses requisitos incluem características como desempenho, compatibilidade, usabilidade e segurança, que garantem a robustez e confiabilidade do sistema. No Quadro 2 são especificados os Requisitos Não Funcionais do projeto do dashboard de monitoramento do ambiente.

Quadro 2 – Requisitos Não Funcionais do sistema

RNF001-Compatibilidade com Navegadores	O sistema deve ser compatível com os principais navegadores modernos, como Google Chrome, Mozilla Firefox, Microsoft Edge e Opera.	Tipo: Plataforma	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF002-Interface Responsiva	O sistema deve apresentar uma interface responsiva, adaptando-se corretamente a diferentes tamanhos de tela, como smartphones, tablets e desktops.	Tipo: Interface	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF003- Interface Intuitiva e Simples	O sistema deve possuir uma interface amigável, de fácil navegação e uso, mesmo para usuários com pouco conhecimento técnico.	Tipo: Interface	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF004-Tempo de Resposta Rápido	O sistema deve apresentar respostas rápidas às ações do usuário, com tempo médio de carregamento inferior a 2 segundos por interação.	Tipo: Plataforma	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF005- Atualização Automática de Dados	O sistema deve atualizar automaticamente os dados exibidos no dashboard sem necessidade de intervenção do usuário.	Tipo: Plataforma	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF006-Suporte a Múltiplos Dispositivos Simultâneos	O sistema deve ser capaz de processar e exibir os dados de múltiplos Arduinos conectados simultaneamente, sem comprometimento da performance.	Tipo: Plataforma	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF007-Códigos de Cor para Interpretação Visual	O sistema deve utilizar cores (verde, amarelo, vermelho) para representar diferentes faixas de valores de temperatura e umidade.	Tipo: Interface	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF008- Comunicação Segura via API	A comunicação entre os Arduinos e a API deve seguir boas práticas de segurança e integridade de dados durante a transmissão.	Tipo: Segurança	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

3.5 REGRAS DE NEGÓCIO

As Regras de Negócio são um conjunto de diretrizes que definem como as atividades de um sistema devem ser realizadas. Elas representam as políticas,

práticas, procedimentos e restrições que governam o comportamento da aplicação e garantem a integridade dos dados e do fluxo funcional. No caso do projeto de monitoramento de temperatura e umidade via dispositivos Arduino, essas regras foram definidas com base em critérios técnicos e operacionais essenciais ao funcionamento seguro e preciso do sistema. O Quadro 3 mostra as Regras de Negócio do projeto.

Quadro 3 – Regras de Negócio do sistema.

RN001 - Identificação Obrigatória do Dispositivo
Descrição: Somente serão aceitos dados enviados por Arduinos que estejam devidamente identificados com um ID válido no corpo da requisição, para garantir a rastreabilidade dos dados por origem.
RN002 - Faixa Válida de Temperatura e Umidade
Descrição: O sistema apenas armazenará dados de temperatura entre -20°C e 80°C, e umidade entre 0% e 100%. Leituras fora desses intervalos serão descartadas por segurança e confiabilidade.
RN003 - Atualização Contínua de Dados
Descrição: Os dados devem ser atualizados automaticamente pelo Arduino em intervalos definidos. O dashboard será alimentado continuamente sem necessidade de ação manual do usuário.
RN004 - Classificação por Faixa com Código de Cor
Descrição: A exibição dos dados no dashboard seguirá uma escala de cores baseada em faixas de valor predefinidas: verde (normal), amarelo (atenção) e vermelho (alerta), conforme os parâmetros definidos no backend.
RN005 - Visualização Pública dos Dados
Descrição: Todos os dados recebidos serão de visualização pública, sem necessidade de autenticação. O sistema foi projetado para acesso direto ao dashboard, com foco em simplicidade e transparência.
RN006 - Armazenamento com Timestamp
Descrição: Todos os dados recebidos deverão conter um carimbo de data e hora (timestamp) para que possam ser organizados corretamente em ordem cronológica e utilizados nos gráficos históricos.
RN007 - Responsividade Obrigatória da Interface
Descrição: A visualização dos dados deve funcionar em qualquer dispositivo, garantindo que a experiência do usuário seja a mesma em telas grandes (web) e pequenas (mobile).

3.6 CASOS DE USO

Nesta seção são apresentadas as características dos Casos de Uso definidas a partir das análises dos Requisitos Funcionais (RF), Não Funcionais (RNF) e Regras de Negócio (RN).

Índice de Casos de Uso:

UC001: Receber Dados dos Dispositivos.

UC002: Armazenar Dados

UC003: Exibir Dados no Dashboard

UC004: Exibir Dados no Mobile

UC005: Classificação por Cores

UC006: Suportar Múltiplos Dispositivos

UC007: Atualização Automática

UC008: Visualização de Histórico

UC009: Configurar Intervalo de Atualização.

Quadro 4 – Use Case Receber Dados dos Dispositivos

Caso de Uso – Receber Dados dos Dispositivos	
ID	UC 001
Descrição	Este caso de uso tem por objetivo permitir que o sistema receba dados de temperatura e umidade enviados por Arduinos por meio de requisições HTTP.
Ator Primário	Usuário do sistema
Pré-condição	O Arduino deve estar conectado à rede Wi-Fi e configurado para enviar dados.
Cenário Principal	1. O Arduino realiza uma leitura dos sensores. 2. O Arduino envia os dados em formato JSON para a API via requisição HTTP. 3. A API recebe a requisição e verifica o conteúdo. 4. A API valida os dados recebidos. 5. O sistema confirma o recebimento com status de sucesso.
Pós-condição	Os dados são processados e preparados para serem armazenados.
Cenário Alternativo	3a – A requisição está malformada: 3a.1 O sistema retorna erro 400 - "Requisição inválida".

Quadro 5 – Use Case Armazenar Dados no Banco de Dados

Caso de Uso – Armazenar Dados no Banco de Dados	
ID	UC 002
Descrição	Este caso de uso tem por objetivo armazenar os dados recebidos dos sensores em um banco de dados NoSQL (MongoDB).
Ator Primário	Sistema
Pré-condição	A API deve ter recebido dados válidos.
Cenário Principal	1. O sistema valida os dados recebidos da API. 2. O sistema inclui um timestamp ao registro. 3. O sistema associa os dados ao identificador do dispositivo. 4. Os dados são salvos no MongoDB.
Pós-condição	Os dados ficam disponíveis para consultas posteriores.
Cenário Alternativo	4a – Falha na conexão com o banco de dados: 4a.1 O sistema registra erro interno e retorna código 500.

Quadro 6 – Use Case Exibir Dados no Dashboard Web

Caso de Uso – Exibir Dados no Dashboard Web	
ID	UC 003
Descrição	Este caso de uso tem por objetivo exibir os dados mais recentes recebidos dos sensores em tempo real na interface web.
Ator Primário	Usuário do sistema
Pré-condição	O sistema deve conter dados recentes no banco.
Cenário Principal	1. O usuário acessa o dashboard via navegador. 2. O sistema solicita os dados recentes à API.

	3. A API retorna os últimos dados coletados. 4. O sistema renderiza os dados em gráficos dinâmicos. 5. O usuário visualiza os dados atualizados.
Pós-condição	O usuário obtém a leitura atual do ambiente monitorado.
Cenário Alternativo	2a – A API não responde: 2a.1 O sistema exibe uma mensagem: "Erro ao carregar dados."

Quadro 7 – Use Case Exibir Dados no Mobile

Caso de Uso – Exibir Dados no Aplicativo Mobile	
ID	UC 004
Descrição	Este caso de uso tem por objetivo permitir que o usuário visualize os dados no aplicativo mobile de forma otimizada.
Ator Primário	Usuário do sistema
Pré-condição	O sistema deve estar funcional e conectado.
Cenário Principal	1. O usuário acessa o aplicativo. 2. O aplicativo envia requisição para buscar os dados atuais. 3. A API responde com os dados. 4. O aplicativo exibe as informações em formato adaptado ao celular.
Pós-condição	O usuário visualiza os dados em tempo real no mobile.
Cenário Alternativo	2a – Falha na requisição: 2a.1 O app exibe a mensagem: "Erro ao buscar dados. Verifique sua conexão."

Quadro 8 – Use Case Classificação por Cores

Caso de Uso – Classificação por Cores	
ID	UC 005
Descrição	Este caso de uso tem por objetivo permitir que o sistema exiba os dados de temperatura e umidade com cores visuais conforme faixas predefinidas.
Ator Primário	Sistema
Pré-condição	Os dados devem estar disponíveis e dentro das faixas válidas.
Cenário Principal	1. O sistema processa o valor recebido do sensor. 2. O sistema verifica a qual faixa o valor pertence. 3. O sistema aplica a cor correspondente (verde, amarelo ou vermelho). 4. A interface exibe a cor no gráfico ou indicador visual.
Pós-condição	Os dados aparecem destacados com base em seu nível de criticidade.
Cenário Alternativo	2a – O valor está fora da faixa configurada: 2a.1 O sistema ignora a cor ou exibe uma cor padrão de erro (ex: cinza).

Quadro 9 – Use Case Suporte a Múltiplos Dispositivos

Caso de Uso – Suportar Múltiplos Dispositivos	
ID	UC 006
Descrição	Este caso de uso tem por objetivo permitir que o sistema receba e trate dados provenientes de mais de um Arduino simultaneamente.
Ator Primário	Sistema
Pré-condição	Cada Arduino deve enviar dados com identificador único.
Cenário Principal	1. Dois ou mais Arduinos enviam dados à API. 2. O sistema identifica os dispositivos pelos seus IDs. 3. Os dados são armazenados no banco de forma separada por origem. 4. O dashboard exibe as leituras individualmente por dispositivo.
Pós-condição	Os dados de todos os Arduinos são exibidos corretamente e de forma separada.
Cenário Alternativo	2a – Dispositivo sem ID definido: 2a.1 O sistema descarta a requisição e registra o erro.

Quadro 10 – Use Case Atualização Automática dos Dados

Caso de Uso – Atualização Automática dos Dados	
ID	UC 007
Descrição	Este caso de uso tem por objetivo atualizar automaticamente os dados exibidos no dashboard em tempo real, sem necessidade de ação do usuário.
Ator Primário	Sistema
Pré-condição	Deve haver conexão estável com a API.
Cenário Principal	1. O usuário abre o dashboard. 2. O sistema inicia a rotina de atualização automática conforme intervalo configurado. 3. A cada ciclo, o sistema requisita novos dados da API. 4. O sistema atualiza os gráficos com os novos valores.
Pós-condição	Os dados são exibidos continuamente, mantendo a interface atualizada.
Cenário Alternativo	3a – Falha na atualização automática: 3a.1 O sistema tenta novamente e exibe aviso: "Falha na atualização. Recarregando..."

Quadro 11 – Use Case Visualização de Histórico por Período

Caso de Uso – Visualização de Histórico por Período	
ID	UC 008
Descrição	Este caso de uso tem por objetivo permitir ao usuário visualizar dados históricos com base em um intervalo de tempo selecionado.
Ator Primário	Usuário do sistema
Pré-condição	Devem existir dados salvos com carimbo de data e hora (timestamp).
Cenário Principal	1. O usuário acessa o dashboard. 2. O usuário seleciona um intervalo de datas (início e fim). 3. O sistema busca os dados correspondentes no banco. 4. O sistema exibe os dados históricos em gráfico.
Pós-condição	O usuário consegue analisar o comportamento das variáveis ao longo do tempo.
Cenário Alternativo	3a – Nenhum dado encontrado no intervalo: 3a.1 O sistema exibe a mensagem: "Sem dados disponíveis nesse período."

Quadro 12 – Use Case Configurar Intervalo de Atualização

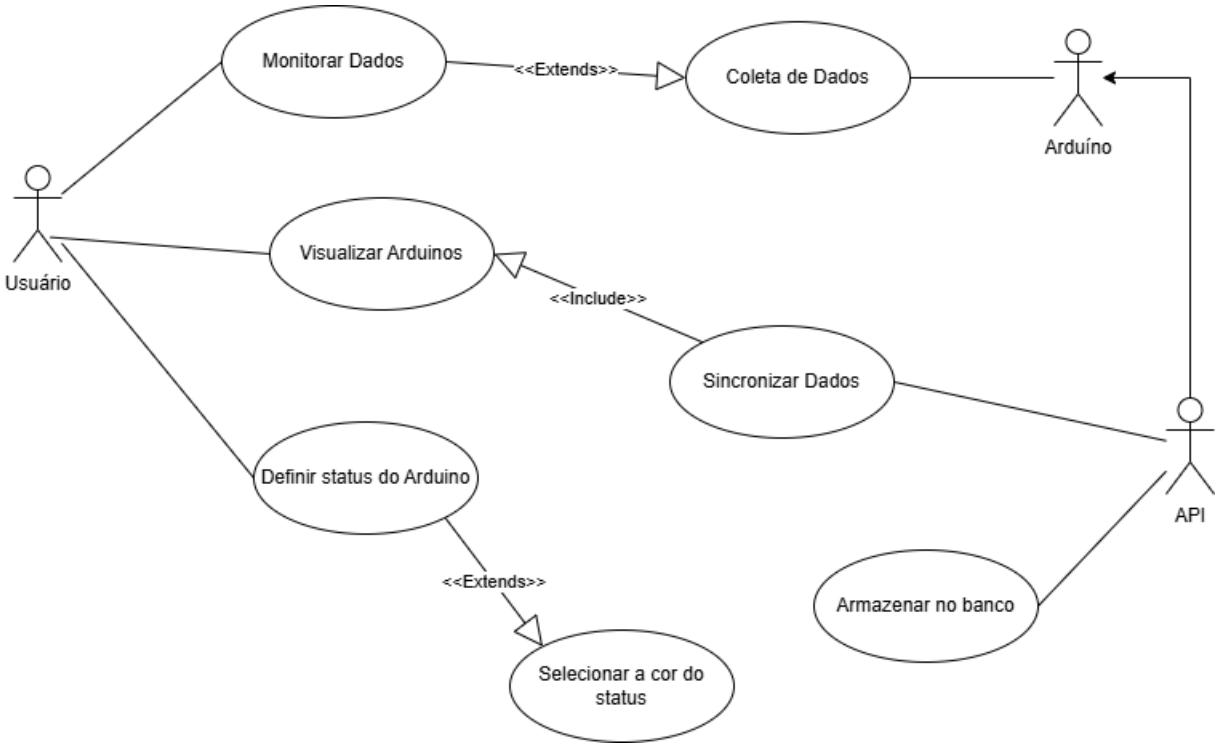
Caso de Uso – Configurar Intervalo de Atualização	
ID	UC 009
Descrição	Este caso de uso tem por objetivo permitir que o usuário defina o intervalo de tempo para que os dados do dashboard sejam atualizados automaticamente.
Ator Primário	Usuário do sistema
Pré-condição	O dashboard deve estar carregado e funcionando corretamente.
Cenário Principal	1. O usuário acessa o menu de configurações. 2. O sistema exibe as opções de intervalo de atualização (ex: 5s, 10s, 30s). 3. O usuário seleciona o intervalo desejado. 4. O sistema aplica a nova configuração. 5. O dashboard passa a atualizar com a nova frequência.
Pós-condição	Os dados são atualizados no novo intervalo definido pelo usuário.
Cenário Alternativo	3a – O usuário seleciona um valor inválido: 3a.1 O sistema exibe a mensagem: "Intervalo inválido. Selecione uma opção válida."

3.7. DIAGRAMA DE CASO DE USO

O Diagrama de Caso de Uso é uma ferramenta essencial na modelagem de sistemas de software, pois permite representar graficamente as interações entre os usuários (atores) e as funcionalidades do sistema. A partir da análise dos requisitos funcionais e das regras de negócio, esse diagrama facilita a visualização do comportamento do sistema sob a perspectiva do usuário, evidenciando os principais cenários de uso.

No contexto do projeto de dashboard de monitoramento ambiental, o diagrama de caso de uso representa as interações entre o usuário e o sistema de visualização de dados, bem como entre os dispositivos Arduino e a API responsável pelo recebimento e armazenamento das informações. Essa representação é fundamental para compreender como cada elemento se comunica com o sistema e quais são as funcionalidades essenciais que ele oferece.

Figura 5 - Diagrama de Caso de Uso



4.0 – PROTÓTIPO DA APLICAÇÃO

Os protótipos de tela desenvolvidos para este projeto têm como objetivo representar visualmente a interface do sistema de monitoramento de umidade em tempo real. Através da prototipação, foi possível planejar e validar a disposição dos elementos no dashboard, como os gráficos de leitura dos sensores, os indicadores

visuais por cores (verde, amarelo e vermelho), e os controles de configuração de horários e limites de umidade.

Além disso, a prototipação foi essencial para garantir que a interface ofereça uma boa experiência tanto em dispositivos desktop quanto na versão mobile, já que o sistema é responsivo e pode ser acessado de diferentes plataformas. Essa etapa permitiu antecipar e corrigir possíveis problemas de usabilidade, assegurando que o sistema seja intuitivo, prático e funcional para os usuários, mesmo antes do início da implementação.

Figura 6 – Prototipação de Tela: Home

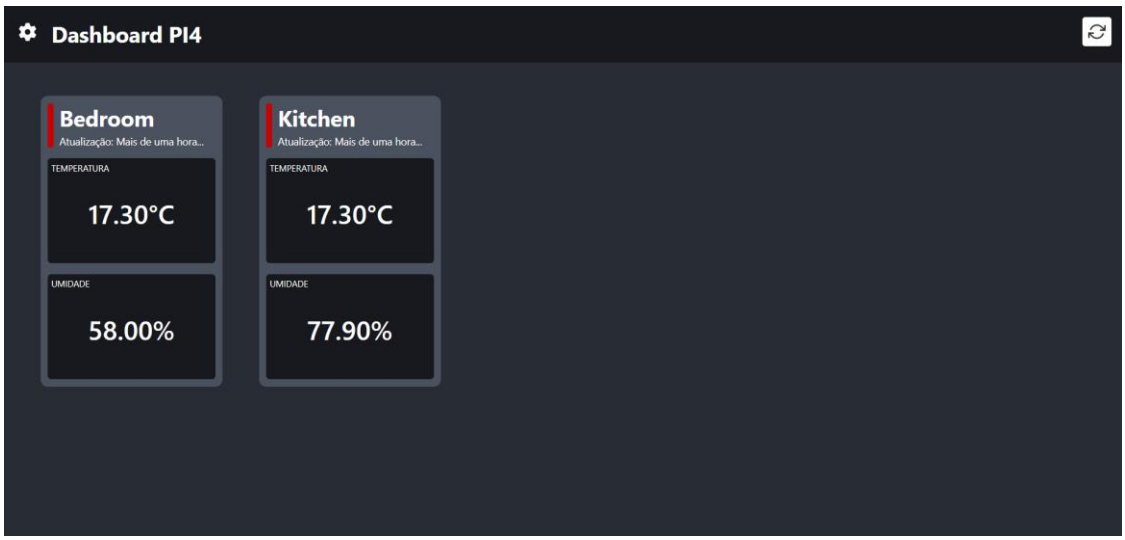


Figura 7 – Prototipação de Tela: Apresentação dos gráficos

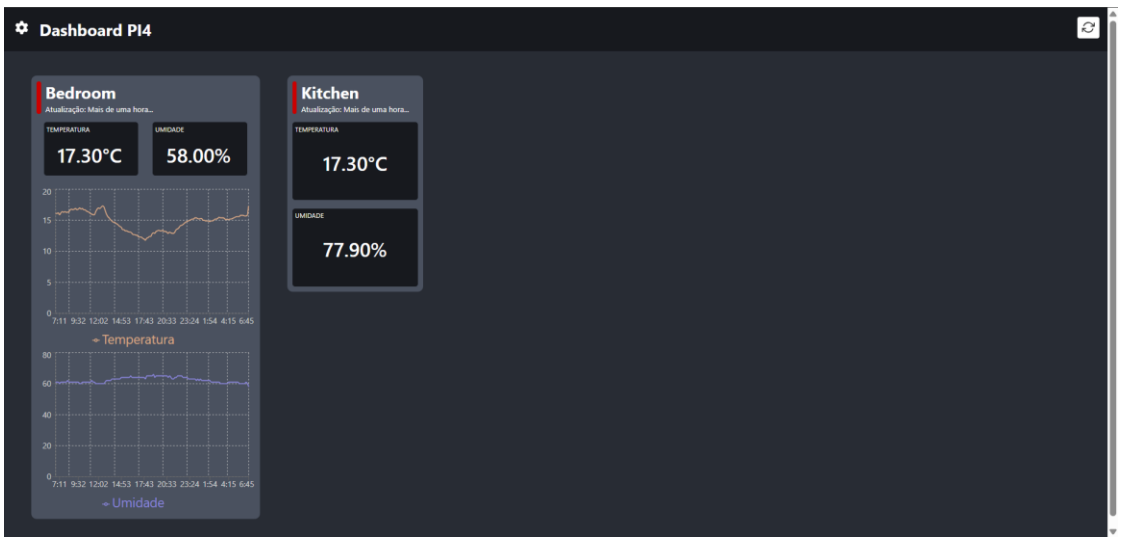


Figura 8 – Prototipação de Tela: Temperatura

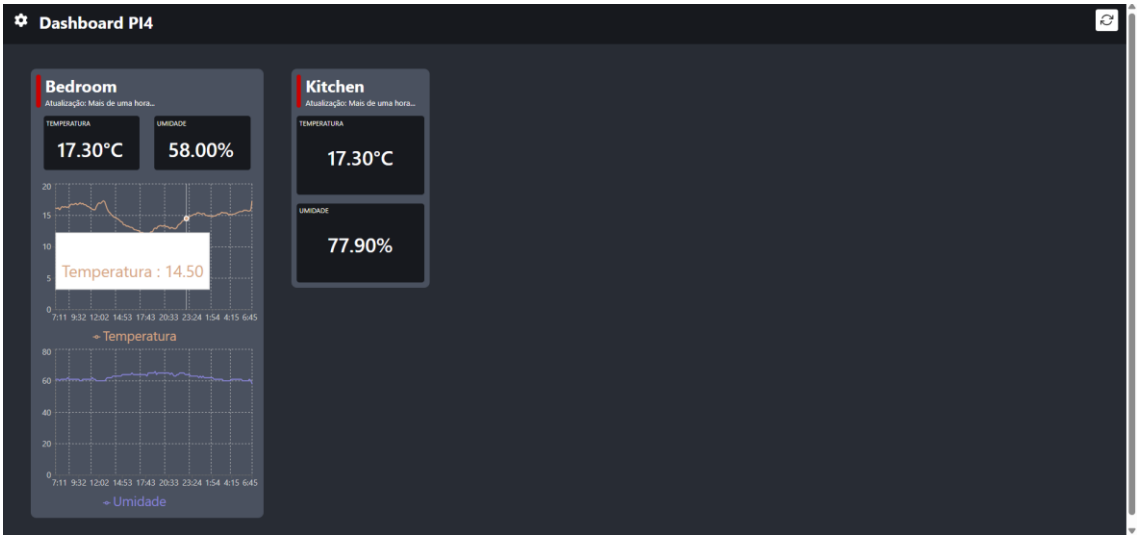


Figura 9 – Prototipação de Tela: Umidade

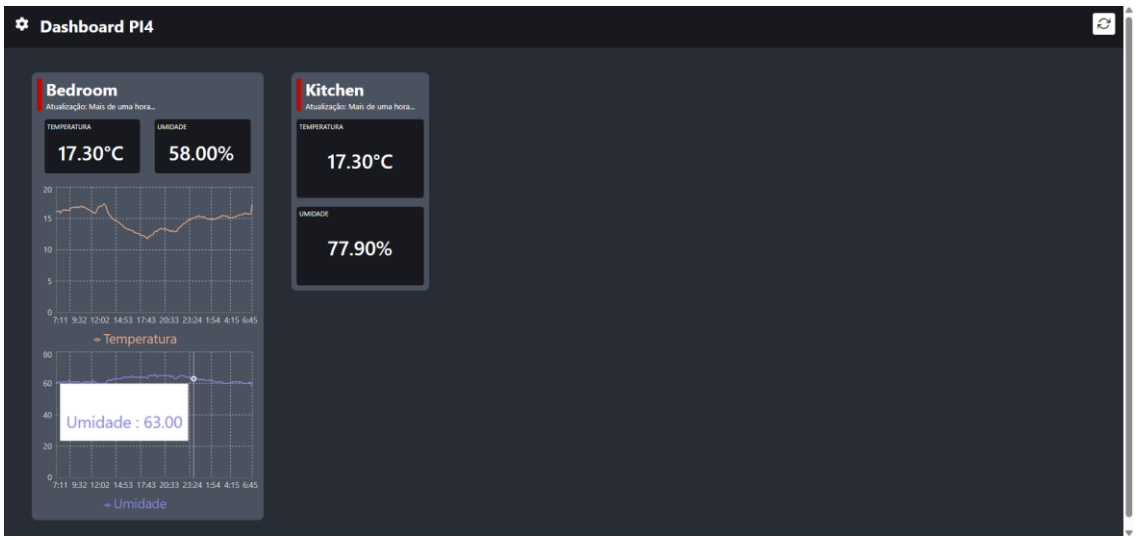


Figura 10 – Prototipação de Tela: Tela de filtros

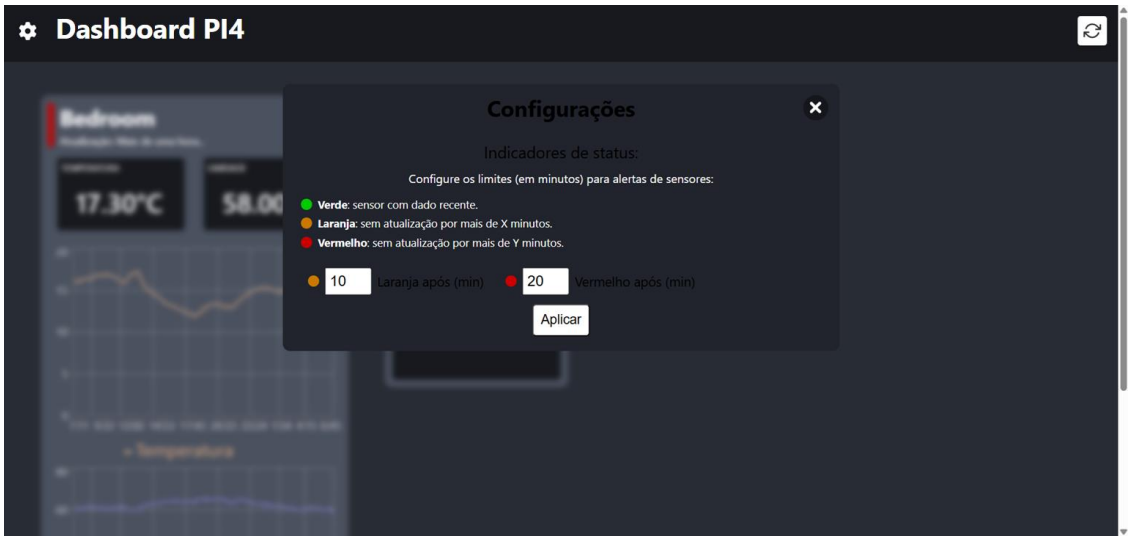


Figura 11 – Prototipação de Tela: Aplicação dos filtros

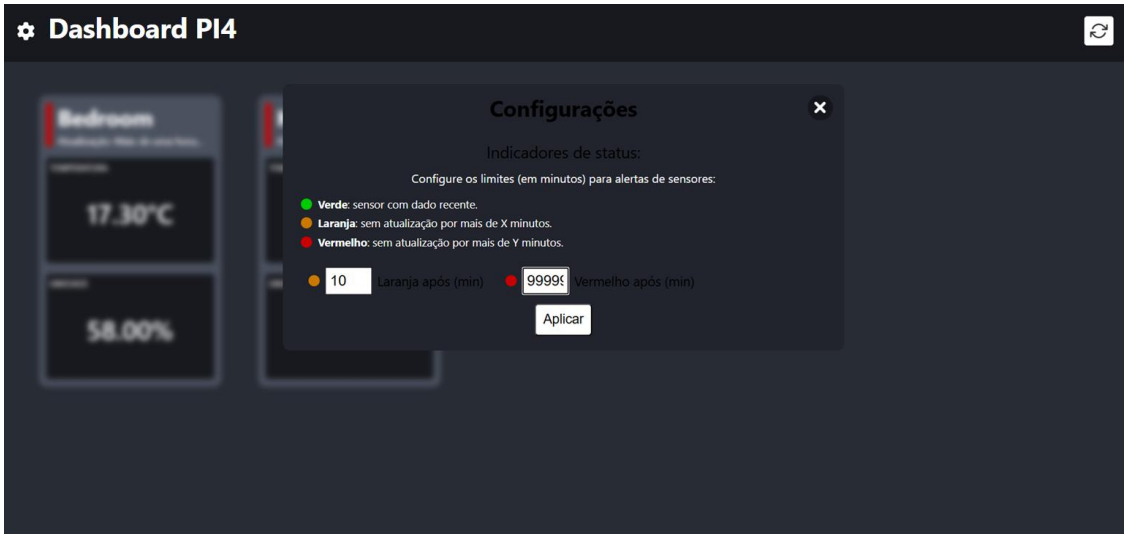


Figura 12 – Prototipação de Tela: Atualização dos sensores

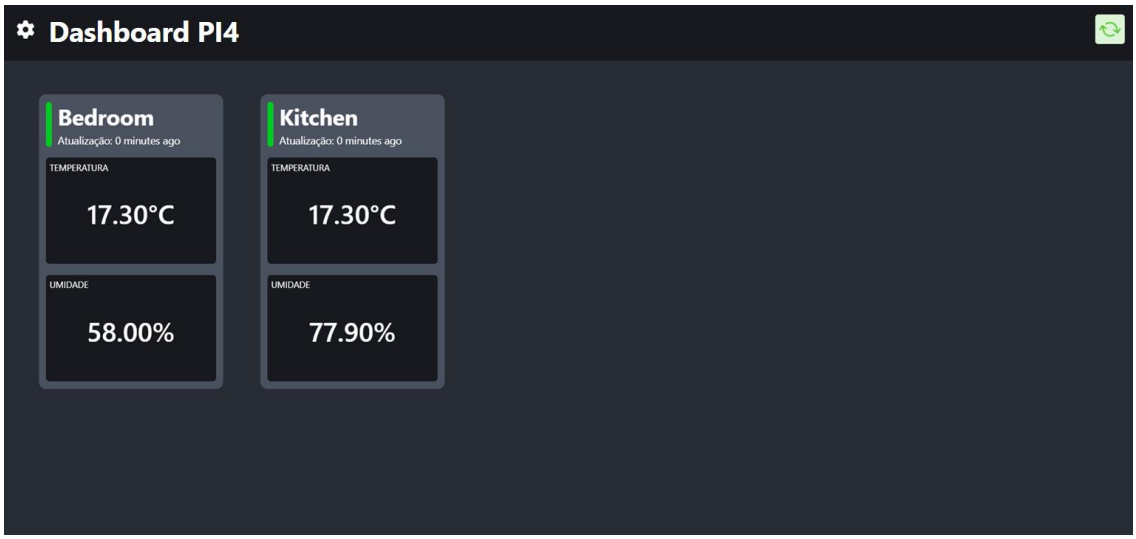


Figura 13 – Prototipação de Tela: Home Mobile

Figura 14 – Prototipação de Tela: Filtros no Mobile

Figura 15 – Prototipação de Tela: Gráficos no Mobile