

Angelo Rodrigo Ribeiro da Silva

Solus, um software para monitoramento de painéis solares

Boituva

2018

Angelo Rodrigo Ribeiro da Silva

Solus, um software para monitoramento de painéis solares

Documentação para o software desenvolvido como trabalho de conclusão de curso para análise e desenvolvimento de sistemas no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, câmpus de Boituva.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO
PAULO - IFSP

Curso de Análise e Desenvolvimento de Sistemas

Orientador: Prof. Dr. Marcelo Figueiredo Polido.

Boituva

2018

Angelo Rodrigo Ribeiro da Silva

Solus, um software para monitoramento de painéis solares

Documentação para o software desenvolvido como trabalho de conclusão de curso para análise e desenvolvimento de sistemas no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, câmpus de Boituva.

Boituva, 4 de dezembro de 2018

Banca examinadora:

(Titulação, nome completo, instituição)

(Titulação, nome completo, instituição)

(Titulação, nome completo, instituição)

Agradecimentos

Agradeço a meus amigos, que não me deixarem abalar nos momentos onde os prazos não pareciam ter fim, agradecimentos especiais aos meus amigos Cleiton Santos, Letícia Callistro, Wellington Sato, Dante Mesquita, Virginia Saucedo e Jaíne Santos, obrigado por entenderem minha ausência no processo de desenvolvimento do trabalho.

À minha família, por sempre fornecer a base que eu precisei para meus estudos.

E ao meu orientador Prof. Dr. Marcelo Figueiredo Polido, que auxiliou na construção dos protótipos e teve a disponibilidade e paciência para com o desenvolvimento deste trabalho.

Resumo

A aplicação desenvolvida baseia-se na dificuldade durante a análise de dados meteorológicos perante a prévia instalação e configuração de painéis solares. Este documento visa documentar e descrever um software com foco na captura e análise de dados meteorológicos, produzindo assim, resultados e informações necessárias para o usuário.

Palavras-chave: nodejs. painéis solares. energia solar. arduino.

Abstract

The application developed is based on the difficulty during the analysis of meteorological data before the previous installation and configuration of solar panels. This document aims to document and describe a software focused on the capture and analysis of meteorological data, thus producing results and information necessary to the user.

Keywords: nodejs. solar panels. solar energy. arduino.

Lista de ilustrações

Figura 1 – Diagrama de caso de uso	18
Figura 2 – Representação da arquitetura	22
Figura 3 – Diagrama de classes	24
Figura 4 – Diagrama de sequência	26
Figura 5 – Diagrama de estado	28
Figura 6 – Tela de login	37
Figura 7 – Dashboard	37
Figura 8 – Resultados de estatísticas	38
Figura 9 – Gráficos das estatísticas	38
Figura 10 – Números dos gráficos das estatísticas	39
Figura 11 – Listagem de estações meteorológicas	39
Figura 12 – Modal de adição e edição dos dados de uma estação	40
Figura 13 – Listagem de usuários	40
Figura 14 – Modal de adição ou edição de usuário	41
Figura 15 – Visualização de perfil	41
Figura 16 – Edição de perfil	42

Lista de tabelas

Tabela 1 – Consultar estatísticas	19
Tabela 2 – Especificações do caso de uso consultar medidas	19
Tabela 3 – Especificação do caso de uso manter usuários	20
Tabela 4 – Especificação do caso de uso manter estações meteorológicas	20
Tabela 5 – Especificação do caso de uso editar perfil	20
Tabela 6 – Especificação da Model base	25
Tabela 7 – Especificação da Medida	25
Tabela 8 – Especificação da Estação Meteorológica	25
Tabela 9 – Especificação do Usuário	25
Tabela 10 – Especificação do sensor de umidade e temperatura em relação a umidade	29
Tabela 11 – Especificações do sensor de temperatura ambiente	30
Tabela 12 – Especificações do sensor de luminosidade LDR	30
Tabela 13 – Especificações do sensor de nível de chuva	30
Tabela 14 – Especificações do sensor de capacidade solar	30
Tabela 15 – Especificações do sensor de radiação uv	31
Tabela 16 – Descrição das rotas da API	33
Tabela 17 – Especificações das rotas de listagem e retorno de arduino	34
Tabela 18 – Descrição dos campos adicionais	34
Tabela 19 – Especificações das rotas de atualização e cadastro de arduino	34
Tabela 20 – Especificações da rota exclusão de arduino	34
Tabela 21 – Especificações das rotas de listagem e retorno de usuário	35
Tabela 22 – Especificações das rotas de atualização e cadastro de usuário	35
Tabela 23 – Especificações da rota de login de usuário	35
Tabela 24 – Especificações da rota exclusão de usuário	35
Tabela 25 – Especificações da rota de cadastro de medida	36
Tabela 26 – Especificações da rota de consulta de estatísticas do arduino	36

Lista de abreviaturas e siglas

API Application Programming Interface (em português Interface de Programação de Aplicações).

HTTP HyperText Transfer Protocol (em português Protocolo de transferência de hipertexto).

IFSP Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

RESTFUL API Full Representational State Transfer Application Programming Interface (em português é um serviço de API que segue a risca as definições da abstração REST)

Sumário

1	INTRODUÇÃO	12
1.1	Objetivo	12
1.2	Justificativa	12
1.3	Estrutura do TCC	13
2	DESENVOLVIMENTO	14
2.1	Metodologias	14
2.1.1	Programação funcional	14
2.1.2	Controle de qualidade	14
2.1.2.1	Testes de interface	14
2.1.2.2	Testes unitários	15
2.1.3	Diagramas UML 2.0	15
2.2	Ferramentas utilizadas	15
2.2.1	Astah	15
2.2.2	Javascript ES8	16
2.2.3	MongoDB	16
2.3	Engenharia de requisitos	16
2.3.1	Cenários do sistema	17
2.3.2	Requisitos funcionais	17
2.3.2.1	Detalhamento dos casos de uso	19
2.3.3	Requisitos não funcionais	20
2.3.3.1	Performance	21
2.3.3.2	Segurança	21
2.3.4	Arquitetura macro	22
2.4	Diagramas do sistema	23
2.4.1	Diagrama de classes	23
2.4.2	Diagrama de sequência	26
2.4.2.1	Controlador	26
2.4.2.2	Repositório	27
2.4.2.3	Modelo	27
2.4.3	Diagrama de estado	27
2.5	Arquitetura das estações meteorológicas	28
2.5.1	Arduino	29
2.5.2	Sensores	29
2.5.3	NodeMCU	31

2.6	Arquitetura do software	31
2.6.1	Servidor	32
2.6.1.1	Banco de dados	32
2.6.1.2	Tratamento dos dados meteorológicos	33
2.6.1.3	Análise dos dados	33
2.6.1.4	Documentação da API	33
2.6.2	Cliente	36
2.6.2.1	Descrição das telas	36
3	CONCLUSÃO	43
	REFERÊNCIAS	44

1 Introdução

A captação de energia elétrica no Brasil, é composta por diversas fontes primárias, dentre as quais, se destacam o uso de energia eólica, energia hidráulica e energia solar, analisadas as diversas fontes de energia, dentre as tais, a energia solar se destaca por sua baixa implicação a questões ambientais.

A energia solar, vem sendo cada vez mais amplamente utilizada no Brasil, tanto por empresas de pequeno a grande porte, quanto pelo mercado residencial, segundo a projeção da Associação Brasileira de Energia Solar Fotovoltaica (Absolar), tal forma de captação de energia representa 0,83% de toda a energia captada no país (KOLOSZUK; SAUAIA, 2018).

Mesmo com o amadurecimento da captação de energia solar no país, para a instalação de uma estação de captação de energia solar, é recomendado que seja feita uma prévia análise de dados no local, para que, resultados satisfatórios possam ser obtidos a mínimo prazo.

O trabalho desenvolvido visa fornecer uma ferramenta para a captura e análise de dados meteorológicos, visando a geração de informações necessárias para o usuário.

1.1 Objetivo

O trabalho tem como objetivo fornecer uma ferramenta para captura, persistência e análise de dados meteorológicos, fornecendo a usuários, informações necessárias para o acompanhamento da eficiência energética de painéis solares.

Aproveitando-se do baixo custo de equipamentos eletrônicos para disponibilizar um software como um todo de baixo custo e de fácil instalação.

Para se atingir o objetivo especificado, foi construído um software de amostragem de dados ao usuário, realizando previamente um tratamento e análise de dados.

1.2 Justificativa

A justificativa do desenvolvimento do trabalho se da pela iniciativa da instalação de uma usina solar no Instituto Federal de São Paulo, no Campus localizado em Boituva.

Foi implementado um laboratório de $30m^2$, composto por uma estação solarimétrica e um sistema de $5kW$ com rastreador solar, entre outras tecnologias.

O projeto de exploração de energia fotovoltaica proporcionou a criação do curso

de instalador de sistemas fotovoltaicos, onde a ferramenta visa ser utilizada de forma a agrupar os dados capturados em diferentes pontos (IFSP, 2018).

1.3 Estrutura do TCC

O trabalho apresentado, primeiramente procura contextualizar acerca da atual situação da captação de energia solar, procurando explicar conceitos básicos de energia fotovoltaica e decorre sobre as dificuldades e cuidados entorno do assunto.

Subsequindo, são apresentadas as metodologias utilizadas no desenvolvimento do projeto, decorrendo acerca das ferramentas utilizadas e conceitos nos quais o projeto se baseia, no desenvolvimento do trabalho, em seguida, é apresentado ao usuário o projeto do sistema, onde questões teóricas acerca da implementação são discutidas, seguindo o desenvolvimento do projeto, se apresenta a arquitetura e documentação do software, detalhando questões internas do desenvolvimento do software e as justificativas pelas tecnologias utilizadas.

Por fim, são apresentadas as considerações finais do projeto, demonstrando os resultados obtidos, comparando os mesmos com os requisitos, procurando esclarecer os objetivos atingidos ou não, e então, é feita uma projeção do projeto para o futuro, apresentando um caminho para o projeto de manutenção e continuidade do software.

2 Desenvolvimento

2.1 Metodologias

2.1.1 Programação funcional

Foi utilizado para a elaboração da arquitetura do software, conceitos de programação funcional, sua maior característica é que, se uma expressão possui um valor bem definido, então, dada uma entrada de dados, a ordem a qual o computador carregar a avaliação não afeta o resultado (BIRD; WADLER, 1988).

A programação funcional, visa, através da utilização do conceito matemático de funções, proporcionar para a arquitetura do software, um modelo onde funções recebem parâmetros e então, através dos parâmetros que foram inseridos na função, um valor é sempre retornado para este parâmetro, este conceito é chamado de função pura, uma vez que um valor x é colocado com entrada, sempre se terá como resposta o valor y . Através das funções puras, o código se torna mais previsível e mais suscetível a testes automatizados, fazendo comparação com o paradigma de programação orientada a objetos, muitas vezes, o objeto está em um estado inválido na memória, podendo assim, acarretar erros.

Porém, nem sempre o controle de estado colateral é levado como lei, porém, ele é contido dentro de funções que estão encapsuladas em locais específicos da aplicação, desta forma, o risco de erros diminui, visando a alta disponibilidade do sistema, que, agora, não depende mais de uma alta quantidade de contextos externos.

2.1.2 Controle de qualidade

Foram utilizadas, juntamente ao desenvolvimento do software, técnicas de controle de software orientadas ao teste, onde, a aplicação é submetida a seção de testes, tanto manuais como automatizados, foram utilizados testes de interface no sistema e testes unitários automatizados.

2.1.2.1 Testes de interface

Para que fosse validada a funcionalidade do sistema como um todo, testes de interface no sistema foram aplicados, onde, diversos usuários foram submetidos a utilização do sistema, informando possíveis dificuldades na utilização do mesmo e erros, que, uma vez corrigidos, voltam a serem analisados por usuários, até que se possa ser atingida uma mínima aceitação.

2.1.2.2 Testes unitários

Segundo Beck, (2002, p.10), "Desenvolvimento orientado a testes, é uma maneira de gerenciar o medo durante a programação".

O conceito de testes unitários em desenvolvimento de software, é fornecer a uma unidade, que pode ser considerada como uma função ou uma instrução do sistema, valores esperados, valores esses, que são comparados com os retornos das funções, e que, uma vez não correspondendo, emitem um erro ao desenvolvedor, que pode, com a visualização facilitada do problema, corrigir tal ponto de forma assertiva.

Tal conceito fora aplicado, para garantir, de forma unitária a qualidade e segurança do código fonte do sistema, garantindo assim, a qualidade da aplicação, desde os primeiros momentos de sua implementação.

2.1.3 Diagramas UML 2.0

Assim como definido por Guedes, (2018), a UML:

É uma linguagem utilizada para modelar softwares baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. (p.17).

No trabalho apresentado, mesmo que não utilizados conceitos de orientação a objetos, a linguagem de modelagem UML foi utilizada como ferramenta para detalhar partes do software, descrevendo desde os casos de uso da aplicação, as funcionalidades do sistema.

2.2 Ferramentas utilizadas

2.2.1 Astah

Astah Community é um software para modelagem UML com suporte a UML 2, desenvolvido pela Change Vision, e disponível para diversos sistemas operacionais. Anteriormente conhecido por JUDE, um acrônimo de Java and UML Developers Environment (Ambiente para Desenvolvedores UML e Java) (LIMA, 2016).

O mesmo foi utilizado no projeto para a criação dos diagramas de descrição software, auxiliando no desenho dos casos de uso na engenharia de requisitos e nos diagramas de classe, diagramas de sequência, estado e implementação.

A utilização do software Astah, foi motivada pela sua confiabilidade na construção dos diagramas, fornecendo uma plataforma com redundâncias e por fornecer uma licença gratuita de sua versão profissional a universidades e estudantes de engenharia de software.

2.2.2 Javascript ES8

O Javascript é um linguagem de programação interpretada multiparadigma, aceitando estilos variados de programação como orientação a objetos, programação funcional e imperativa (MOZILLA, 2018). O Javascript é uma linguagem de programação primariamente desenvolvida para o lado do cliente, sendo executada no navegador do usuário, porém, com o advento do motor de processamento do motor do Google Chrome, o v8, a linguagem foi implementada também do lado servidor, utilizando o motor de processamento nodejs, que é baseada no motor do Chrome (NODEJS, 2018).

Para a implementação do projeto, foi utilizada a linguagem de programação Javascript em sua versão 8, ou como definida pela especificação do Javascript, o EcmaScript 2017.

A utilização da linguagem Javascript, tanto para a construção da aplicação no lado servidor, quando para o lado cliente, foi determinada por sua alta performance e por proporcionar ferramentas que permitem a programação de forma funcional, detalhes acerca da utilização da linguagem são melhor definidos na seção acerca dos detalhes da implementação do software, em 2.6.

2.2.3 MongoDB

O MongoDB é um banco de dados de documento com suporte a escalabilidade e flexibilidade para execução de consultas e indexação necessária (MONGODB, 2018).

O mesmo foi utilizado no projeto em sua versão 4.1, a justificativa de seu uso, se da pela sua facilidade ao se trabalhar com escrita massiva de informações, proporcionando a aplicação performance e disponibilidade.

2.3 Engenharia de requisitos

Assim como descrito por Pressman (2011):

O amplo espectro de tarefas e técnicas que levam a um entendimento dos requisitos é denominado engenharia de requisitos. Na perspectiva do processo de software, a engenharia de requisitos é uma ação de engenharia de software importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho. (p.126).

Neste capítulo, há a utilização da engenharia de requisitos para auxiliar a documentação do projeto de software, descrevendo os cenários de utilização de sistema, onde é descrito o comportamento esperado pelo usuário através de entrevistas, após, utilizando

os resultados obtidos com a modelagem através de cenários, os requisitos são definidos e são criados os contratos de uso do usuário com o sistema através de casos de uso, então, é descrita a arquitetura macro do software, procurando contextualizar acerca do desenho do projeto.

2.3.1 Cenários do sistema

Há diversas formas de se medir a qualidade de um software, porém, não há outra forma de se medir a qualidade e se um objetivo foi atingido no desenvolvimento de um sistema do que a satisfação dos usuários. Se entendido como os usuários desejam interagir com o sistema, o que é esperado de entradas, processamentos e saídas, será possível a equipe construir um projeto mais objetivo e proveitoso (PRESSMAN, 2011).

Portanto, nesta seção são descritos os cenários do software através de uma descrição que foi construída utilizando conteúdos das entrevistas que foram feitas com as partes interessadas no projeto.

No Instituto Federal de São Paulo, no campus localizado em Boituva, o professor responsável pelo projeto descreve o sistema como uma aplicação capaz de capturar dados meteorológicos através de sensores próximos a painéis solares, ele necessita dessas informações sendo demonstradas através de gráficos, para que uma análise possa ser feita, além da disposição das informações através de gráficos, é esperado do sistema demonstrar ao usuário informações de mínimas e máximas das informações capturadas. É desejado pelo usuário que o sistema seja capaz de lidar com diversas estações meteorológicas, reunindo os dados, portando, o sistema necessita de um cadastro das estações meteorológicas, guardando informações e identificações.

Através do texto acima, podemos entender as necessidades do usuário para com o sistema, o cenário de software descrito é definido dentro dos requisitos nas seções seguintes.

2.3.2 Requisitos funcionais

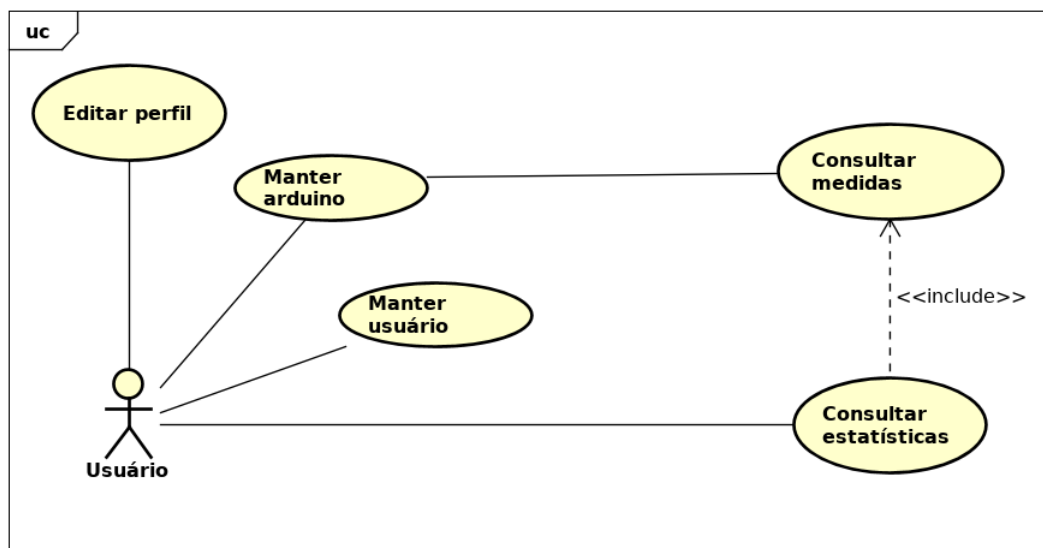
Os requisitos funcionais de um software são utilizados para descrever o que ele deve fazer, eles giram em torno do tipo de sistema desenvolvido, de seus usuários e da forma como uma equipe de desenvolvimento de software busca entender o que é necessário para criar um software que atenda as necessidades de um usuário. Ao serem descritas as funcionalidades básicas de um sistema, a fazemos de forma abstrata, para que os interessados no desenvolvimento do projeto possam entender o que será feito. Porém, os mesmos podem ser detalhados de forma específica, declarando suas entradas, saídas e exceções (SOMMERVILLE, 2011).

Foi utilizado para descrever os requisitos funcionais do sistema, a modelagem de requisitos através do diagrama de caso de uso, que busca visualizar, especificar e

documentar o comportamento de um ator ao utilizar o sistema. (GUEDES, 2018).

O diagrama de caso de uso descrito na figura 1, resume os requisitos funcionais que foram definidos ao longo do desenvolvimento do projeto.

Figura 1 – Diagrama de caso de uso



Fonte: Produção do autor.

Através do diagrama do caso de uso contido na figura 1, é possível identificarmos apenas um ator na aplicação, que é o usuário do sistema, interessado em visualizar informações geradas pela análise estatística dos dados capturados. O mesmo interage com a aplicação através da interface de usuário, realizando cadastros e consultas.

O requisito funcional **Consultar estatísticas** é o principal requisito do sistema, onde o usuário fará a filtragem dos dados, trazendo a ele as estatísticas geradas pela aplicação. O usuário irá utilizar de campos de entrada de dados para escolher qual a estação meteorológica ele deseja visualizar, a data inicial para realizar as consultas, a data final e o intervalo de tempo para cálculo das médias. Após a consulta das informações, o usuário irá obter como resultado as informações de mínimas e máximas das medidas e gráficos que o informam as médias para cada uma das propriedades meteorológicas.

O requisito funcional representado pelo caso de uso **Manter usuário**, descreve as ações do usuário acerca do cadastro de usuários no sistema. Através desta funcionalidade, o usuário poderá, através da interface do usuário na web, utilizando tabelas, realizar o cadastro, consultas, atualização de informações e exclusão de usuários. Durante a interação do usuário com a persistência de dados de usuários no sistema, caso erros ocorram, mensagens de erro auxiliares serão mostradas ao usuário, o auxiliando a encontrar qual o erro no processo realizado.

O requisito funcional **Manter arduino**, é muito similar ao caso de uso anterior **Manter usuário**, e representa a persistência de informações acerca das estações meteorológicas no sistema, utilizando as quatro operações básicas de cadastro, consulta, atualização e exclusão de dados.

O requisito funcional de representado pelo caso de uso **Editar Perfil** representa a visualização e edição dos dados do usuário que está utilizando o sistema no momento, ele pode escolher editar suas informações, que exibe um formulário com os atuais dados preenchidos, onde o usuário pode realizar a alteração de suas informações.

2.3.2.1 Detalhamento dos casos de uso

Utilizando de tabelas, este capítulo descreve de forma detalhada cada um dos casos de uso, apresentado as descrições, os atores relacionados, as pré condições, exceções e fluxos alternativos.

Tabela 1 – Consultar estatísticas

Realizar consultas com filtros	
Descrição	Recebe do usuário os filtros para seleção das informações, então, realiza uma análise estatística dos dados requisitados e os exibe para o usuário utilizando listagens e gráficos.
Atores	Usuário
Pré-condições	Banco de dados em correto funcionamento Filtros corretos passados pelo usuário Listagem de estações meteorológicas retornando ao mínimo uma estação
Exceções e fluxos alternativos	Caso a seleção não retorne nenhuma informação ao usuário, uma mensagem de dados inexistentes é exibida Em caso de perca de conexão com banco de dados, exibe uma mensagem de erro ao usuário Caso não existam estações meteorológicas cadastradas, o filtro de seleção de estação meteorológica exibe um aviso ao usuário

Tabela 2 – Especificações do caso de uso consultar medidas

Consultar medidas	
Descrição	Consulta as medidas que estão persistidas no banco de dados para uma estação meteorológica, no software desenvolvido, atualmente este caso de uso é apenas acessado pela geração de estatísticas
Pré-condições	Credenciais de acesso ao banco de dados Banco de dados em correto funcionamento
Exceções e fluxos alternativos	Caso não existam medidas capturadas para a estação meteorológica, retorna uma lista vazia de dados

Tabela 3 – Especificação do caso de uso manter usuários

Manter usuários	
Descrição	Realiza uma listagem dos dados dos usuários atualmente registrados no sistema, fornecendo a possibilidade de cadastrar um novo usuário, atualizar os dados de um usuário ou excluir um usuário do sistema
Pré-condições	Credenciais de acesso ao banco de dados Banco de dados em correto funcionamento
Exceções e fluxos alternativos	Caso não existam usuários cadastrados no sistema, exibe uma mensagem de aviso ao usuário Para o cadastro ou atualização dos dados de um usuário, caso existam dados inválidos que foram dados como entrada, ao confirmar a operação, exibe uma mensagem de erro ao usuário

Tabela 4 – Especificação do caso de uso manter estações meteorológicas

Manter estações meteorológicas	
Descrição	Realiza uma listagem dos dados das estações meteorológicas atualmente registradas no sistema, fornecendo a possibilidade de cadastrar uma nova estação, atualizar os dados de uma estação ou excluir uma estação meteorológica do sistema
Pré-condições	Credenciais de acesso ao banco de dados Banco de dados em correto funcionamento
Exceções e fluxos alternativos	Caso não existam estações cadastradas no sistema, exibe uma mensagem de aviso ao usuário Para o cadastro ou atualização dos dados de uma estação, caso existam dados inválidos que foram dados como entrada, ao confirmar a operação, exibe uma mensagem de erro ao usuário

Tabela 5 – Especificação do caso de uso editar perfil

Editar perfil	
Descrição	Exibe os dados do usuário que está utilizando o sistema no momento, dando a possibilidade do usuário alterar os próprios dados
Pré-condições	Credenciais de acesso ao banco de dados Banco de dados em correto funcionamento
Exceções e fluxos alternativos	Caso na edição do perfil do usuário atual existam dados inválidos que foram dados como entrada, ao confirmar a operação, exibe uma mensagem de erro ao usuário

2.3.3 Requisitos não funcionais

Os requisitos não funcionais, como seu próprio nome demonstra, não estão ligados diretamente ao que um software deve fazer, porém, tais requisitos são considerados tão importantes quanto os requisitos funcionais, pois podem comprometer a usabilidade e funcionalidades de um sistema (SOMMERVILLE, 2011).

Comumente, requisitos não funcionais estão ligados a questões de disponibilidade de software, tempo de resposta ao usuário e outras partes que se comunicam com o sistema e segurança.

Uma das formas de determinar a arquitetura do sistema seguindo os requisitos não funcionais é restringir a forma como as funcionalidades primárias do sistema trabalham,

trabalhando no projeto e na implementação do software de forma restrita dentro destes padrões.

Nas seções subsequentes os tópicos de disponibilidade, tempo de resposta e segurança do sistema são melhor detalhados.

2.3.3.1 Performance

Para a boa usabilidade do usuário ao interagir com um sistema, um crucial requisito funcional é o tempo de resposta de uma aplicação, pois, dependendo da lentidão com a qual as informações são disponibilizadas, um usuário pode ter problemas e, em muitos casos, não utilizar o sistema por conta do tempo de resposta.

Na era da informação, a informação é trabalhada de forma quase instantânea, portanto, tal requisito não pode ser deixado em segundo plano ao trabalharmos no projeto de um software.

No sistema desenvolvido, todo o desenvolvimento do sistema foi pensando de forma a fornecer ao usuário um baixo tempo de resposta através do uso de técnicas de desenvolvimento de software focadas em alta disponibilidade, desde a implementação da arquitetura do software e de como o banco de dados é utilizado, até a forma como os dados são carregados na interface do usuário. Para atingir tal objetivo, optou-se por disponibilizar ao usuário o mínimo de informações necessárias para a interação com o sistema.

Detalhes acerca da forma como o sistema é implementado para fornecer ao usuário a melhor experiência possível desde sua implementação em baixo nível são melhor descritos na seção sobre a arquitetura do software em 2.6.

2.3.3.2 Segurança

Outro requisito muito importante para qualquer sistema é a segurança, tanto nas informações que ali estão contidas quanto no resultado final que é apresentado ao usuário, com a evolução da tecnologia, as pessoas passaram a estar muito mais próximas a sistemas de informação, colocando ali, informações sensíveis.

Neste projeto, para que se pudesse proporcionar ao usuário melhor segurança em seus dados e nas informações meteorológicas que são informadas, foram utilizadas técnicas de criptografia de informações, focando primariamente em informações que são sensíveis, e, para acesso ao sistema, técnicas de bloqueio de informações através de chaves foram utilizadas, melhores detalhes acerca de como tecnologias utilizadas funcionam são relatados na seção 2.6, onde a arquitetura do software proposto é apresentada.

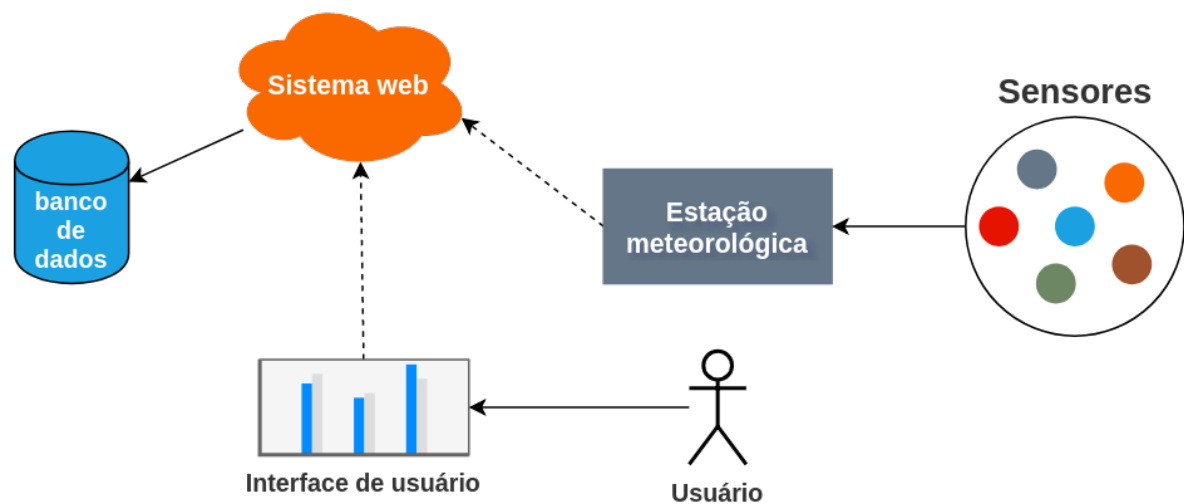
2.3.4 Arquitetura macro

A arquitetura de um software segundo Sommerville, (2011):

É o elo crítico entre o projeto e a engenharia de requisitos, pois identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles. O resultado do processo de projeto de arquitetura é um modelo de arquitetura que descreve como o sistema está organizado em um conjunto de componentes de comunicação. (p.104).

Procurando descrever os casos de uso do sistema, optou-se por utilizar uma descrição da arquitetura macro do software através de figuras, descrevendo o todo do sistema através de simples componentes, pode-se conferir a arquitetura através da figura 2

Figura 2 – Representação da arquitetura



Fonte: Produção do autor.

Através do desenho da arquitetura do software podemos identificar as relações entre os diversos componentes do sistema, porém, de uma forma a qual a qual as partes interessadas no projeto ainda podem interagir e entender o que está sendo proposto.

No diagrama, podemos identificar os sensores conectados diretamente com a estação meteorológica (traço preenchido), enviando as informações capturadas para a estação meteorológica através de conexão web (traço pontilhado), que, após receber as informações as envia para o sistema Web, que as persiste em banco de dados.

O usuário, interage com o sistema diretamente através da interface que disponibiliza gráficos para que o acompanhamento possa ser realizando, consultando as informações através de conexão web com o sistema.

Dessa forma, pode-se descrever a arquitetura do projeto, visando, atingir o objetivo definido através da engenharia de requisitos com um projeto agora melhor detalhado, tanto para as partes interessadas, quanto para a equipe de software, nas seções seguintes, é descrita de forma técnicas os componentes do sistema, através de diagramas UML.

2.4 Diagramas do sistema

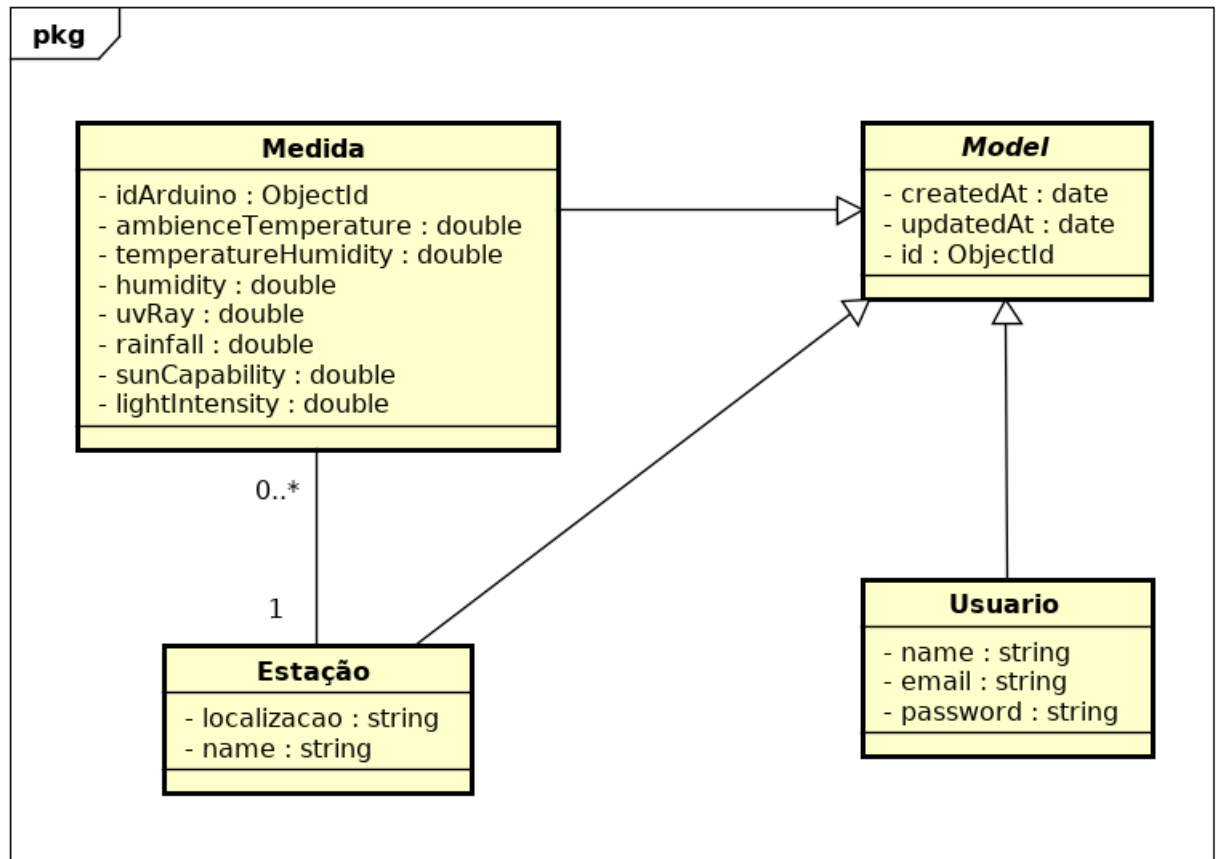
Foram utilizados para a descrição dos componentes da aplicação os diagramas UML, os diagramas de classe, de sequência e de estado descrevem o software nos capítulos seguintes.

2.4.1 Diagrama de classes

"Os diagramas de classe são usados no desenvolvimento de um modelo de sistema orientado a objetos para mostrar as classes de um sistema e as associações entre essas classes. Em poucas palavras, uma classe de objeto pode ser pensada como uma definição geral de um tipo de objeto do sistema."(SOMMERVILLE, 2011).

O sistema desenvolvido implementa o paradigma de programação funcional de forma pouco restrita, portanto, alguns conceitos de programação orientada a objetos foram descartados. Porém, o diagrama de classes também pode representar as características das entidades do sistema.

Figura 3 – Diagrama de classes



Fonte: Produção do autor.

Como podemos verificar no diagrama descrito na figura 3 as entidades do sistema consistem na medida, que é a informação que foi capturada pela estação meteorológica, que pertence a uma estação, a entidade de estação meteorológica, que possui nenhuma a muitas medidas capturadas, e a entidade de usuário, que representa um usuário cadastrado no sistema.

As propriedades de cada entidade são especificadas abaixo, descartando aqui, seus comportamentos, pois, as mesmas são manipuladas através das camadas, que são descritas nas seções que descrevem o comportamento do sistema, através de diagramas de sequência e estado, nas respectivas seções 2.4.2 e 2.4.3.

Tabela 6 – Especificação da Model base

Model	
Descrição	É a model base da aplicação, que advem do framework utilizado pra mapeamento dos objetos para o banco de dados
Propriedades	
id	Identifica de forma única um objeto no banco de dados
createdAt	Data em que o objeto foi armazenado
updatedAt	Data em que alguma das propriedades do objeto foi alterada pela última vez

Tabela 7 – Especificação da Medida

Medida	
Descrição	Representada a medida que foi capturada pela estação meteorológica
Propriedades	
idArduino	Identificação do arduino que realizou a captura
ambienteTemperatura	O valor que foi capturado pelo sensor de temperatura ambiente
temperatureHumidity	Valor da temperatura capturada pelo sensor agregado de temperatura e umidade
humidity	Valor da umidade capturada pelo sensor agregado de temperatura e umidade
uvRay	Nível de radiação uv que foi medida
rainfall	Porcentagem de chuva capturada
sunCapability	Capacidade solar que foi medida
lightIntensity	Nível da intensidade de luz capturada

Tabela 8 – Especificação da Estação Meteorológica

Estação meteorológica	
Descrição	Representa uma estação meteorológica cadastrada no sistema
Propriedades	
location	Descrição de onde a estação meteorológica está posicionada
name	Nome de identificação da estação meteorológica

Tabela 9 – Especificação do Usuário

Usuário	
Descrição	Representa um usuário cadastrado no sistema
Propriedades	
name	Nome de identificação do usuário
email	E-mail que identifica o usuário e é utilizado para login no sistema
password	Senha do usuário ao acessar o sistema, a informação é criptografada

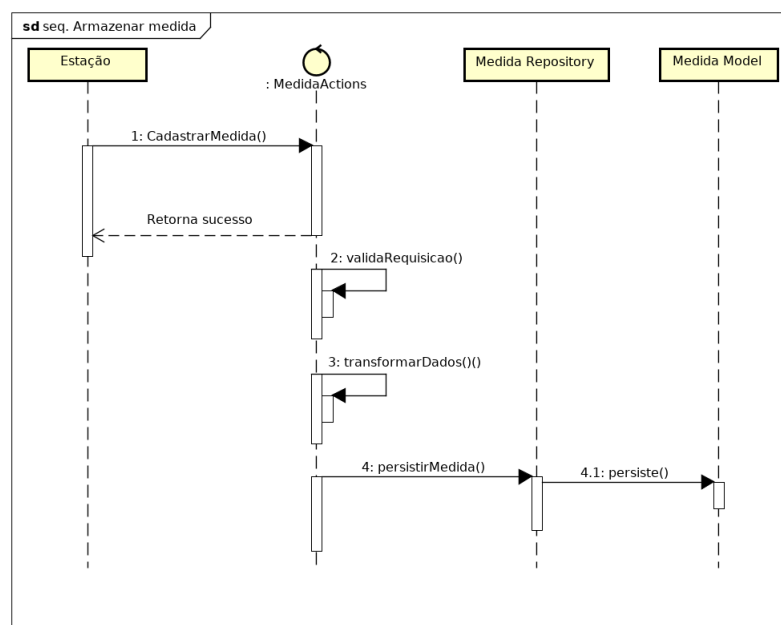
2.4.2 Diagrama de sequência

"Os diagramas de sequência em UML são usados, principalmente, para modelar as interações entre os atores e os objetos em um sistema e as interações entre os próprios objetos. A UML tem uma sintaxe rica para diagramas de sequência, que permite a modelagem de vários tipos de interação."(SOMMERVILLE, 2011).

Nesta seção, é apresentado o diagrama de sequência que representa o armazenamento de uma medida capturada por uma estação meteorológica, apresentando desde a saída dos dados, até a sua persistência em banco.

Para melhor aproveitamento da estação meteorológica e para que não haja necessidade de espera de uma resposta do software, utilizou-se da programação assíncrona para responder com sucesso a estação, os dados são enviados, e uma resposta de sucesso é emitido a estação, porém, os dados são processados em seguida, deixando livre a fila de envio na estação.

Figura 4 – Diagrama de sequência



Fonte: Produção do autor.

Como descrito no diagrama de sequência representado na figura 4 a aplicação é composta por 3 principais camadas, seguindo um modelo que são descritas nos capítulos seguintes.

2.4.2.1 Controlador

Na camada de controle, após ser recebida uma mensagem da estação meteorológica, a mesma retorna sucesso para a estação, porém sem a confirmação das informações, que

são logo em seguida, validadas, e então são tratadas utilizando as regras descritas na seção 2.6.1.2.

2.4.2.2 Repositório

A camada de repositório trabalha como uma intermediária entre o controlador da aplicação onde a validação dos dados é feita e a camada de persistência da aplicação, ela trabalha verificando se os dados persistidos estão corretos e fornece um local comum para consulta e armazenamento das informações.

2.4.2.3 Modelo

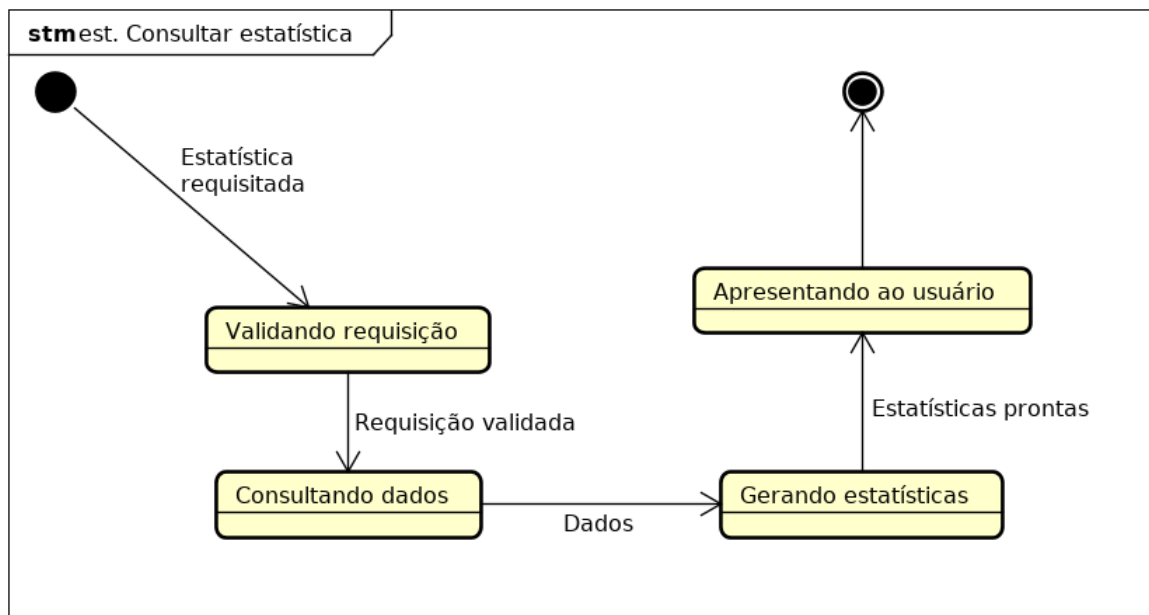
A camada de modelos, é composta pelas entidades do sistema, é a única camada da aplicação que possui acesso direto ao banco de dados, e é gerenciada através da ferramenta de comunicação com o banco de dados, abstraindo assim, o acesso ao baixo nível da persistência do software.

2.4.3 Diagrama de estado

"O diagrama de máquina de estados demonstra o comportamento de um elemento por meio de um conjunto finito de transições de estado, ou seja, uma máquina de estados."(GUEDES, 2018).

O diagrama de estado abaixo compreende a consulta de estatísticas na aplicação, demonstrando, o estado de uma requisição no sistema até sua resposta ao usuário.

Figura 5 – Diagrama de estado



Fonte: Produção do autor.

Na figura 5 podemos identificar os estados como sendo a estatística requisitada ao sistema, a requisição sendo identificada como válida, então, os dados são encontrados no sistema e as estatísticas são apresentadas ao usuário.

No estado identificado como **Validando requisição**, a mesma é validada, identificando se o usuário realmente possui acesso ao sistema e se os filtros que foram aplicados são também válidos, após os dados são consultados e passados para a geração de estatísticas, nos estados **Consultando dados** e **Gerando estatísticas**, respectivamente, por fim, no estado identificado como **Apresentando ao usuário** os gráficos são renderizados na tela, e o ciclo de vida da requisição termina.

2.5 Arquitetura das estações meteorológicas

Para a construção da arquitetura da estação meteorológica, foi utilizado como central o microcontrolador arduino, diversos sensores para captura das informações e a placa WiFi NodeMCU para conexão a internet e gerenciamento das requisições com o servidor.

Mais detalhes acerca dos componentes utilizados são descritos nas seções Ambiente de desenvolvimento integrado subsequentes.

2.5.1 Arduino

Como definido em seu site oficial: "O Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software de fácil utilização. É destinado a qualquer pessoa que construa projeto interativos." (ARDUINO, 2018).

Ele fornece um controlador eletrônico único baseado no microprocessador ATmega328, desenvolvido em c++, componentes eletrônicos oficiais para a construção de aplicações interativas e uma IDE (Ambiente de desenvolvimento integrado) para a construção de sistemas.

Para a construção da estação meteorológica descrita, foi utilizado arduino UNO, em sua versão contendo 6 portas analógicas, 6 portas digitais uma cpu com velocidade de 16 MHz e 32 KB de armazenamento interno.

Foram utilizados diversos sensores conectados as portas seriais do arduino para coleta de dados, após coletados os dados, o arduino monta a requisição que o servidor espera em formato JSON, é um formato de dados similar ao XML, porém, utiliza a sintaxe de objetos do javascript, para a construção do JSON não foi utilizada nenhuma biblioteca, por conta da quantidade de memória limitada do arduino.

2.5.2 Sensores

Para a captura das informações foram utilizados os sensores de temperatura, umidade do ar com temperatura em relação a umidade, sensor de chuva digital/analógico, sensor de intensidade luminosa, radiação UV e capacidade solar.

Os sensores utilizados se conectam com o arduino através de suas portas seriais, emitindo as informações nas portas seriais para que o arduino possa as capturar.

Os sensores utilizados são melhor descritos nas tabelas de descrição abaixo.

Tabela 10 – Especificação do sensor de umidade e temperatura em relação a umidade

Sensor de umidade e temperatura relativa DHT11	
Descrição	Capturam os valores de umidade e temperatura em relação a umidade
Conexão	Se conecta com o arduino através da porta analógica 0 (A0)
Identificação dos valores	
Umidade	É representada por porcentagem (0 a 100%)
Temperatura	Capta a temperatura atual em °C

Tabela 11 – Especificações do sensor de temperatura ambiente

Sensor de temperatura ambiente Thermistor	
Descrição	Captura o valor da temperatura ambiente
Conexão	Se conecta com o arduino através da porta analógica 1 (A1)
Identificação dos valores	
Temperatura	Capta a temperatura ambiente atual em °C

Tabela 12 – Especificações do sensor de luminosidade LDR

Sensor de luminosidade LDR	
Descrição	Captura o valor da intensidade luminosa
Conexão	Se conecta com o arduino através da porta analógica 2 (A2)
Identificação dos valores	
< 100	Claridade Intensa
100 - 300	Claridade alta
300 - 500	Claridade acima da média
500 - 700	Claro
700 - 850	Pouca Claridade
850 - 1000	Muito Pouca Claridade
> 1000	Escuro

Tabela 13 – Especificações do sensor de nível de chuva

Sensor de chuva	
Descrição	Captura através do sensor digital se está chovendo (1) ou não (0) e através do sensor analógico de 0 a 4.5
Conexão	Se conecta com o arduino através da porta analógica 3 (A3)
Identificação dos valores	
< 2.00	Sem Chuva
2.00 - 3.00	Nuvens Esparsas
3.00 - 3.99	Nublado
4.00 - 4.50	Chuva
> 4.50	Chuva Intensa

Tabela 14 – Especificações do sensor de capacidade solar

Sensor de capacidade solar	
Descrição	Captura a capacidade solar utilizando fotocélula
Conexão	Se conecta com o arduino através da porta analógica 5 (A5)
Identificação dos valores	
< 1,00	Fraca
1,00 - 1,49	Média
1,50 - 1,99	Boa
>= 2,00	Ótima

Tabela 15 – Especificações do sensor de radiação uv

Sensor de radiação uv						
Descrição	Captura índice uv de forma analógica (0 - 240)					
Conexão	Se conecta com o arduino através da porta analógica 4 (A4)					
Identificação dos valores						
Índice UV	0	1	2	3	4	5
Voltagem (mV)	<50	227	318	408	503	606
Valor analógico	<10	46	65	83	103	124
Índice UV	6	7	8	9	10	11+
Voltagem (mV)	696	795	881	976	1079	1170
Valor analógico	142	162	180	200	221	240

Alguns dados foram tratados antes de serem armazenados em banco de dados, detalhes acerca da transformação de dados que foi aplicada são melhor abordados na seção 2.6.1.2.

2.5.3 NodeMCU

Para a conexão e envio dos dados para o servidor, foi utilizada a placa wifi NodeMCU ESP8266, conectada nas portas TX e RX.

A placa NodeMCU fornece possibilidade de conexão com rede wifi e roteamento wifi e pode ser programada em linguagem LUA ou utilizando a IDE do Arduino.

No projeto, o arduino se conecta com o NodeMCU através das portas TX e RX, onde o NodeMCU espera que os dados que foram capturados pelo Arduino são escritos em sua porta serial, então, os dados são enviados através de requisição HTTP para o servidor.

As configurações da placa NodeMCU são definidas no começo de seu arquivo de código fonte (linhas 3 - 12).

2.6 Arquitetura do software

Segundo Martin, um dos maiores pensadores sobre o assunto da qualidade de software na modernidade, em seu livro Clean Architecture (Arquitetura limpa): "O software foi inventado para ser soft". Pretendia ser uma maneira de alterar facilmente o comportamento das máquinas. Se quiséssemos que o comportamento das máquinas fosse difícil de mudar, teríamos chamado de hardware". (MARTIN, 2017).

A arquitetura de software desenvolvida utiliza da separação dos componentes de software em pacotes que são exportados e podem ser utilizados em qualquer parte da

aplicação, separando as responsabilidades do sistema, utilizando também os benefícios de técnicas do paradigma de programação funcional, foi possível isolar os pontos de efeitos colaterais da aplicação, deixando os comportamentos da aplicação, mais previsíveis.

Para a construção, foi utilizado clássico modelo de cliente servidor, onde a aplicação do lado cliente tem a responsabilidade de renderizar o que o cliente vê (front end) e a aplicação no lado servidor tem a responsabilidade de aplicar regras de negócio, realizar armazenamento e análise dos dados solicitados (back end), no sistema construído, o lado cliente e o lado servidor se comunicam através de protocolo HTTP, utilizando o conceito restful no lado servidor que é descrito na seção 2.6.1 e o lado cliente utiliza a tecnologia SPA (Single Page Application), que é detalhada na seção 2.6.2

2.6.1 Servidor

O lado do servidor foi construído utilizando NodeJS, um runtime para Javascript no lado servidor, foi disponibilizada para a aplicação uma API (Application Programming Interface) via HTTP para a comunicação tanto da interface do usuário quanto para o arduino, utilizando os métodos RESTful. Uma RESTful API é uma interface de acesso a aplicação aplicação que utiliza requisições http para a criação, atualização, consulta e exclusão de dados (ROUSE, 2016). No sistema desenvolvido, foi utilizado o formato de dados JSON.

A documentação da api pode ser conferida na seção 2.6.1.4.

2.6.1.1 Banco de dados

O banco de dados utilizado foi o MongoDB, um banco de dados não relacional, o desenho do banco de dados é controlado pela aplicação, que define quais campos devem ser indexados e como os dados devem se "relacionar", cada informação capturada fica armazenada dentro de uma coleção, utilizando o formato JSON.

A escolha de um banco de dados não relacional, foi motivada pela facilidade ao se trabalhar com esquemas maleáveis, podendo ter suas informações mutadas, deixando a cargo da aplicação a tomada de decisão.

O banco MongoDB foi escolhido pela sua facilidade ao trabalhar com escrita de dados concorrente, pois, as informações são, em primeiro instante processadas e armazenadas em cache, e já são retornadas para a aplicação, somente após, o MongoDB se encarrega de realizar a inserção dos dados em disco.

Com todas as vantagens do banco de dados, a aplicação tira vantagem, se beneficiando no que toca performance, disponibilidade e mutabilidade das informações.

2.6.1.2 Tratamento dos dados meteorológicos

Os dados que foram capturados pelas estações, tiveram seus valores mantidos, com exceção dos valores de intensidade de luz e quantidade de chuva, o valor de intensidade de luz precisou ser invertido para melhor visualização das informações, considerando x como o valor que foi capturado, com sua máxima identificada como 1024, a equação $1024 - x$ é aplicada, invertendo o valor, caso o resultado seja negativo, o valor 0 é considerado em seu lugar, para a quantidade de chuva, a mesma regra foi aplicada, porém, o valor também foi convertido para %, então a fórmula para obtenção do valor que será armazenado para quantidade de chuva será $((1024 - x)/1024)100$, novamente, dado resultado negativo, será considerada a porcentagem 0.

2.6.1.3 Análise dos dados

São retornados ao usuário os dados de médias, mínimas e máximas, por serem os valores mais significantes para o acompanhamento meteorológico na situação identificada. Para obtenção das médias dos valores, foi aplicada a função de agregação do MongoDB, reunindo as informações por intervalo de tempo e aplicando a função de média sobre o resultado.

2.6.1.4 Documentação da API

Na tabela 16 são detalhadas as rotas da api, documentando os métodos que foram utilizados, os endereços de cada uma das rotas e sua descrição.

Tabela 16 – Descrição das rotas da API

Método	Rota	Descrição
GET	/arduino	Lista os arduinos
GET	/arduino/:id	Retorna os dados de um arduino
POST	/arduino	Cadastra um novo arduino
POST, PATCH, PUT	/arduino/:id	Atualiza os dados de um arduino
DELETE	/arduino/:id	Deleta um arduino e suas medidas capturadas
GET	/user	Lista os usuários
GET	/user/:id	Retorna os dados de um usuário
POST	/user	Cadastra um novo usuário
POST, PATCH, PUT	/user/:id	Atualiza os dados de um usuário
DELETE	/user/:id	Deleta um usuário
POST	/user/login	Recebe as credenciais e retorna o token
POST	/measure	Cadastra uma medida capturada
GET	/statistic/:id	Retorna as estatísticas de dados do arduino

Nas tabelas abaixo, podemos conferir o detalhamento de cada uma das rotas da API, para todos os casos, os dados que forem retornados da API, são encapsulados na

variável `data` dentro do corpo de resposta da requisição HTTP, no caso de listagem de dados, a variável `data` é um vetor que armazena as entidades listadas e no retorno de uma única entidade, a variável `data` é um objeto contendo os dados que foram retornados.

Juntamente com os dados, o campo `success` é retornado, quando verdadeiro, a requisição foi processada com sucesso, caso contrário, a requisição foi processada sem sucesso. Para todos os retornos, considere também os campos adicionais informados na tabela 18.

Tabela 17 – Especificações das rotas de listagem e retorno de arduino

GET /arduino e GET /arduino/:id		
Saída		
Propriedade	Tipo	Descrição
name	String	Nome da estação
location	String	Localização da estação

Tabela 18 – Descrição dos campos adicionais

Campos adicionais		
Propriedade	Tipo	Descrição
_id	String	Identificador da entidade no banco de dados
createdAt	String	Data de inserção do registro no banco de dados
updatedAt	String	Data da última atualização em dados do registro

Tabela 19 – Especificações das rotas de atualização e cadastro de arduino

POST /arduino e POST /arduino/:id		
Entrada		
Propriedade	Tipo	Descrição
name	String	Nome da estação
location	String	Localização da estação
Saída		
Propriedade	Tipo	Descrição
name	String	Nome da estação
location	String	Localização da estação

Tabela 20 – Especificações da rota exclusão de arduino

DELETE /arduino/:id		
Saída		
Propriedade	Tipo	Descrição
name	String	Nome da estação
location	String	Localização da estação

Tabela 21 – Especificações das rotas de listagem e retorno de usuário

GET /user e GET /user/:id		
Saída		
Propriedade	Tipo	Descrição
name	String	Nome do usuário
email	String	E-mail do usuário
password	String	Senha criptografada do usuário

Tabela 22 – Especificações das rotas de atualização e cadastro de usuário

POST /user e POST /user/:id		
Entrada		
Propriedade	Tipo	Descrição
name	String	Nome do usuário
email	String	E-mail do usuário
password	String	Senha do usuário
Saída		
Propriedade	Tipo	Descrição
name	String	Nome do usuário
email	String	E-mail do usuário
password	String	Senha criptografada do usuário

Tabela 23 – Especificações da rota de login de usuário

POST /user/login		
Entrada		
Propriedade	Tipo	Descrição
email	String	E-mail do usuário
password	String	Senha do usuário
Saída		
Propriedade	Tipo	Descrição
name	String	Nome do usuário
email	String	E-mail do usuário
password	String	Senha criptografada do usuário
token	String	Token para autenticação no sistema

Tabela 24 – Especificações da rota exclusão de usuário

DELETE /user/:id		
Saída		
Propriedade	Tipo	Descrição
name	String	Nome do usuário
email	String	E-mail do usuário
password	String	Senha criptografada do usuário

Tabela 25 – Especificações da rota de cadastro de medida

POST /measure		
Entrada		
Propriedade	Tipo	Descrição
arduinoId	String	Id da estação
uvRay	Number	Medida de radiação UV
rainfall	Number	Medida da quantidade de chuva
sunCapability	Number	Nível de capacidade solar
humidity	Number	Porcentagem de umidade
ambienceTemperature	Number	Medida da temperatura ambiente
temperatureHumidity	Number	Medida da temperatura em relação a umidade
lightIntensity	Number	Medida da intensidade de luz

Tabela 26 – Especificações da rota de consulta de estatísticas do arduino

GET /statistic/:id		
Entrada		
Propriedade	Tipo	Descrição
from	String	Data inicial
to	String	Data final
interval	String	Intervalo da seleção
Saída		
Propriedade	Tipo	Descrição
uvRay	Number	Medida de radiação UV
rainfall	Number	Medida da quantidade de chuva
sunCapability	Number	Nível de capacidade solar
humidity	Number	Porcentagem de umidade
ambienceTemperature	Number	Medida da temperatura ambiente
temperatureHumidity	Number	Medida da temperatura em relação a umidade
lightIntensity	Number	Medida da intensidade de luz

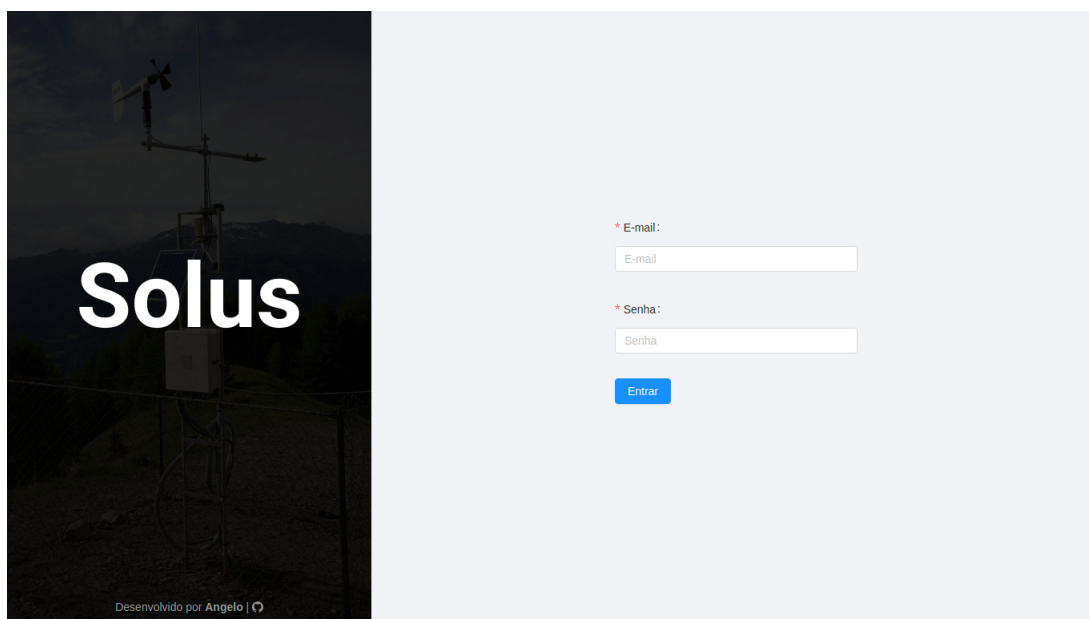
2.6.2 Cliente

Foi utilizado para a construção da aplicação lado cliente a biblioteca REACT, ela é uma biblioteca baseada em componentes e fornece, através da linguagem javascript diversas ferramentas para a construção de SPA's (Single Page Applications) (REACT, 2016). As SPA'S (Single Page Applications) são aplicações desenvolvidas para browser, utilizando javascript, que são capazes de fornecer ao usuário uma experiência sem atualização do navegador, de forma dinâmica.

2.6.2.1 Descrição das telas

Abaixo, pode-se conferir a documentação e descrição de cada uma das telas.

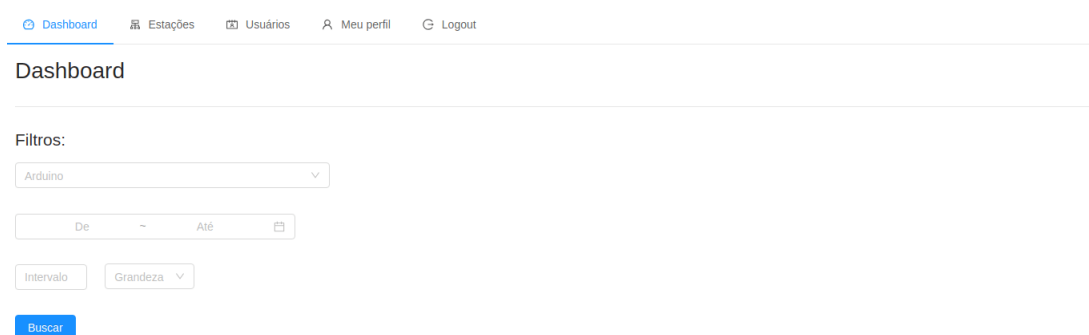
Figura 6 – Tela de login

A imagem mostra a interface de login do sistema Solus. À esquerda, há uma imagem de fundo escura com o nome 'Solus' em branco e o texto 'Desenvolvido por Angelo' no canto inferior esquerdo. À direita, há um formulário de login com campos para 'E-mail' e 'Senha', ambos precedidos por um asterisco vermelho. Abaixo dos campos, há um botão azul com o texto 'Entrar'.

Fonte: Produção do autor.

Na tela de login, podemos conferir o formulário de acesso ao sistema, requisitando o e-mail e senha do usuário e o botão de acesso, ao ser confirmado o login no sistema o usuário é levado até a tela principal do sistema.

Figura 7 – Dashboard

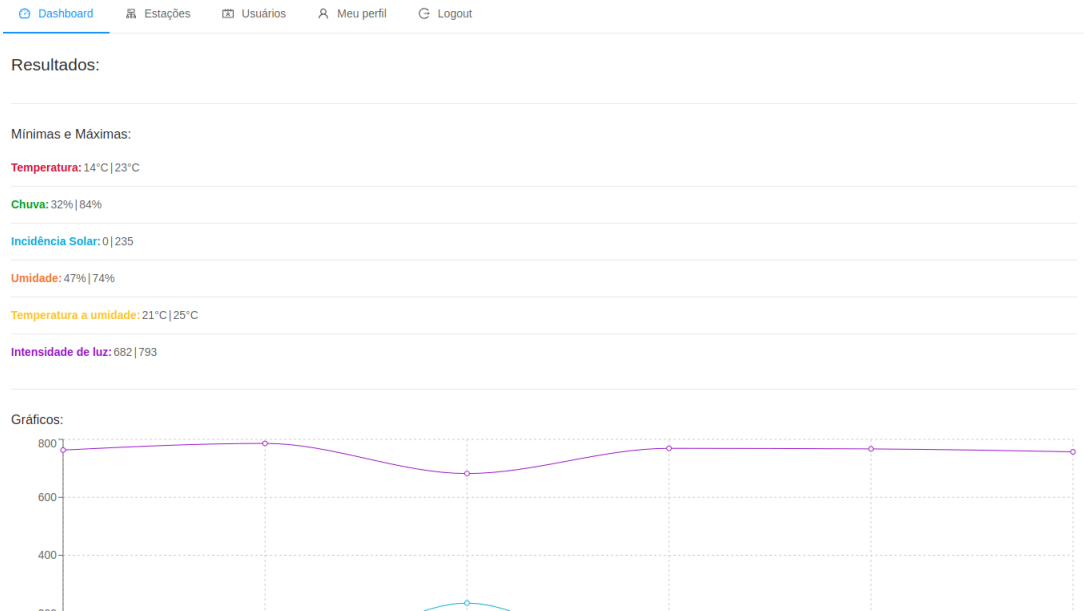
A imagem mostra o dashboard do sistema Solus. No topo, há uma barra de navegação com links para 'Dashboard', 'Estações', 'Usuários', 'Meu perfil' e 'Logout'. Abaixo, o título 'Dashboard' é seguido por uma seção de filtros. Os filtros incluem um menu suspenso para 'Arduíno', campos para 'De' e 'Até' com um ícone de calendário, e botões para 'Intervalo' e 'Grandeza' com um menu suspenso. Um botão azul 'Buscar' está na base dos filtros.

Fonte: Produção do autor.

Conforme pode-se conferir na figura 7 na tela inicial do sistema, podemos conferir os filtros de dados, onde ao se escolher as informações de opção da estação meteorológica

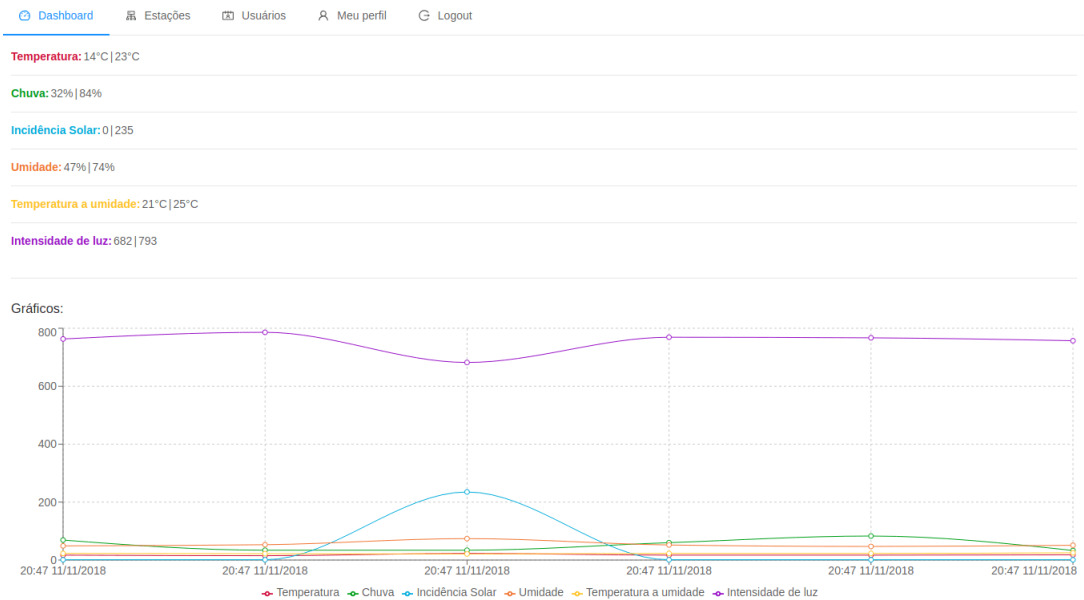
desejada, data inicial e data final de captura e qual o intervalo de tempo e confirmar a seleção de estatísticas, os resultados da figura 8 são exibidos.

Figura 8 – Resultados de estatísticas



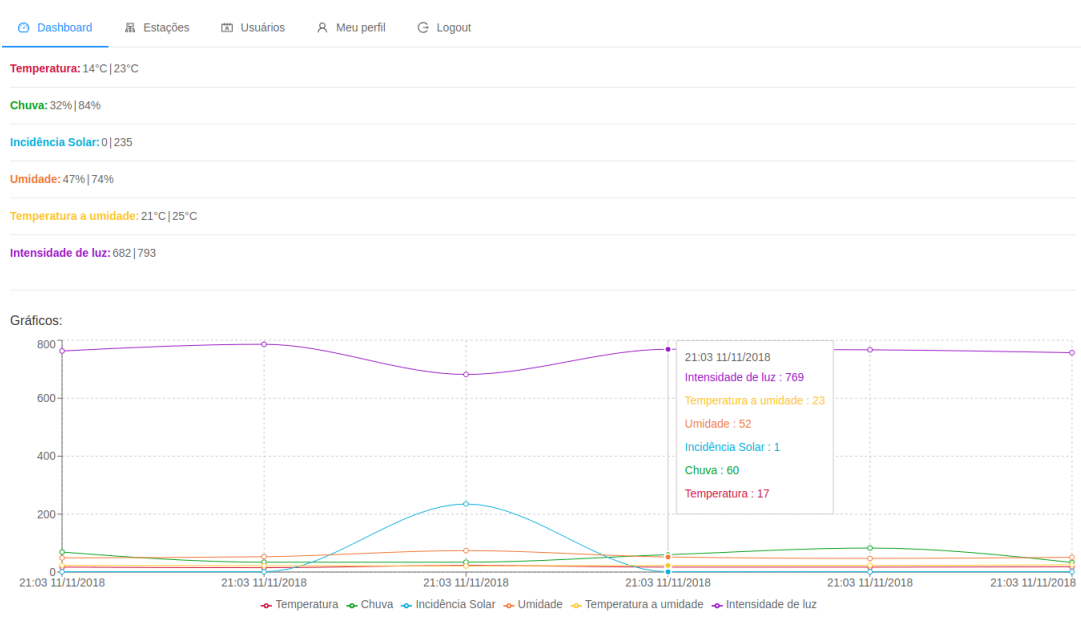
Fonte: Produção do autor.

Figura 9 – Gráficos das estatísticas



Fonte: Produção do autor.

Figura 10 – Números dos gráficos das estatísticas



Fonte: Produção do autor.

São exibidas ao usuário as medidas de mínimas e máximas dos dados e os gráficos das médias que foram calculadas logo abaixo.

Figura 11 – Listagem de estações meteorológicas

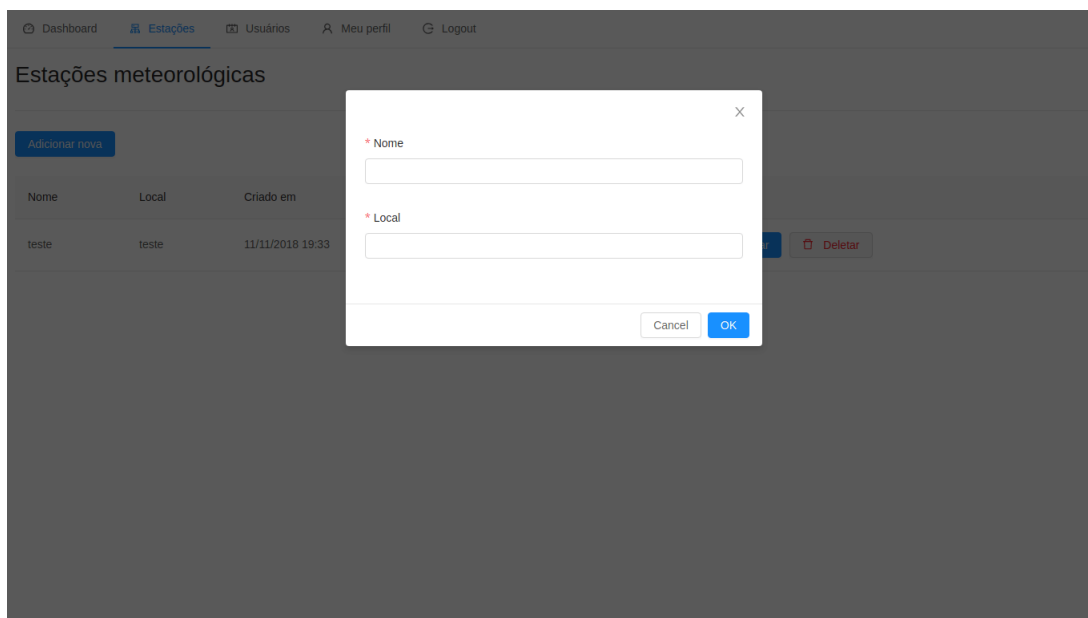
Dashboard	Estações	Usuários	Meu perfil	Logout
Estações meteorológicas				
Adicionar nova				
Nome	Local	Criado em	Última modificação	Ações
teste	teste	11/11/2018 19:33	11/11/2018 19:33	Editar Deletar

Fonte: Produção do autor.

Na listagem de estações meteorológicas, é exibida ao usuário uma tabela com as informações das estações meteorológicas cadastradas, há no topo da tabela a opção de

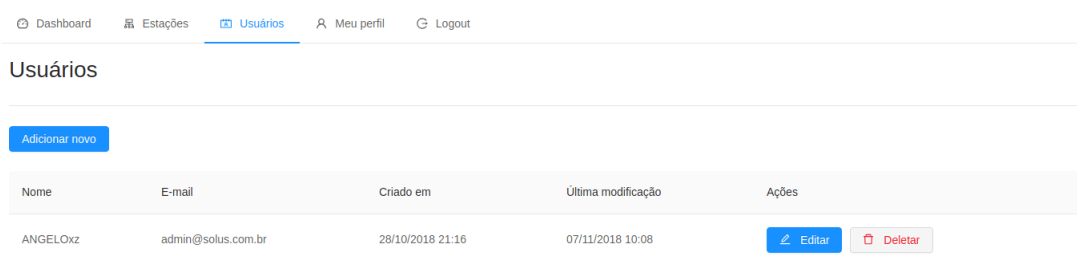
adicionar uma nova estação meteorológica e para cada uma das linhas da tabela são exibidas as ações de editar ou excluir o registro de uma estação meteorológica, os modais de cadastro e edição podem ser conferidos na figura 12, porém, no caso da edição, o modal já abre com as informações preenchidas.

Figura 12 – Modal de adição e edição dos dados de uma estação



Fonte: Produção do autor.

Figura 13 – Listagem de usuários



Fonte: Produção do autor.

A tela de listagem de usuários, segue o mesmo comportamento da tela de listagem de estações meteorológicas, porém, para o modal de cadastro de usuário, é adicionado o campo opcional para edição de senha e e-mail.

Figura 14 – Modal de adição ou edição de usuário

The screenshot shows a web application interface with a dark sidebar and a light main content area. The sidebar contains navigation links: Dashboard, Estações, **Usuários**, Meu perfil, and Logout. The main content area has a header 'Usuários' and a table with columns 'Nome' and 'E-mail'. A table row shows 'ANGELOxz' and 'admin@solus.com.br'. To the right of the table is an 'Ações' column with 'Editar' and 'Deletar' buttons. A modal form is open in the center, titled 'Usuários' with a close button. The form has three input fields: 'Nome', 'E-mail', and 'Senha', each with a red asterisk indicating it is required. At the bottom of the modal are 'Cancelar' and 'Salvar' buttons.

Fonte: Produção do autor.

Figura 15 – Visualização de perfil

The screenshot shows the 'Meu perfil' page in a web application. The sidebar contains navigation links: Dashboard, Estações, Usuários, **Meu perfil**, and Logout. The main content area has a header 'Meu perfil'. Below the header is a section titled 'Meus dados' containing the following information: 'Nome: ANGELOxz', 'Email: admin@solus.com.br', 'Data de cadastro: 21:16 28/10/2018', and 'Última modificação: 10:08 7/11/2018'. At the bottom of the section is an 'Editar' button with a pencil icon.

Fonte: Produção do autor.

Na tela **Meu perfil** são exibidos os dados do usuário que está utilizando o sistema

no momento, caso o usuário abra a função de edição, é exibido o formulário abaixo, que ao ser confirmado, é fechado novamente, exibindo somente suas informações já atualizadas.

Figura 16 – Edição de perfil

A imagem mostra a interface de usuário para a edição de perfil. No topo, há uma barra de navegação com links para Dashboard, Estações, Usuários, Meu perfil (destacado) e Logout. Abaixo, o título 'Meu perfil' é seguido por uma seção 'Meus dados' que contém campos para Nome (preenchido com 'ANGELOxz'), E-mail (preenchido com 'admin@solus.com.br') e Senha (com o placeholder 'Senha'). Botões 'Salvar' e 'Cancelar' estão localizados abaixo dos campos. Na base da seção, há um botão 'Editar' com um ícone de lápis.

Fonte: Produção do autor.

3 Conclusão

O sistema desenvolvido é capaz de fornecer uma ferramenta ao especialista para análise e acompanhamento de informações importantes no acompanhamento de painéis solares, atingido assim seus objetivos, porém, dadas as facilidades de mudança no software e na forma como os dados foram modelados, acrescentaria mais valor ao software, serem adicionadas novas medidas meteorológicas, enriquecendo mais a forma como o usuário é informado.

Medições importantes como a velocidade do vento e a quantidade em milímetros de chuva não foram adicionados, e trariam grandes avanços ao projeto. Funcionalidades geográficas como identificação das estações meteorológicas através de posicionamento em mapa seriam interessantes ferramentas para a análise meteorológica proposta.

Referências

- ARDUINO. 2018. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>. Acesso em: 03 nov. 2018. Citado na página 29.
- BECK, K. *Test Driven Development By Example*. 1. ed. [S.l.]: Yahoo, 2002. Citado na página 15.
- BIRD, R.; WADLER, P. *Introduction to functional programming*. 1. ed. [S.l.: s.n.], 1988. Citado na página 14.
- EVANS, E. *Domain Driven Design: Atacando as complexidades no coração do software*. 3. ed. [S.l.]: Dog Ear Publishing, 2014. Nenhuma citação no texto.
- GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1. ed. [S.l.]: Addison-Wesley Professional, 1994. Nenhuma citação no texto.
- GUEDES, G. T. A. *UML 2 - Uma Abordagem Prática*. 3. ed. [S.l.]: novatec, 2018. Citado 3 vezes nas páginas 15, 18 e 27.
- IFSP. 1º workshop do projeto de eficiência energética e p&d realizado entre o ifsp campus boituva. 2018. Disponível em: <<https://btv.ifsp.edu.br/index.php/component/content/article/17-ultimas-noticias/2372-1-workshop-do-projeto-de-eficiencia-energetica-e-p-d-realizado-entre-o-ifsp-campus-boituva>>. Acesso em: 02 nov. 2018. Citado na página 13.
- KOLOSZUK, R.; SAUAIA, R. Renováveis no brasil: Maturidades diferentes para cada fonte exigem cuidados especiais. 2018. Disponível em: <<http://www.absolar.org.br/noticia/artigos-da-absolar/renovaveis-no-brasil-maturidades-diferentes-para-cada-fonte-exigem-cuidados-especiais.html>>. Acesso em: 09 nov. 2018. Citado na página 12.
- LIMA, D. de. Modele softwares com astah community. *Techtudo*, 2016. Disponível em: <<https://www.techtudo.com.br/tudo-sobre/astah-community.html>>. Acesso em: 03 nov. 2018. Citado na página 15.
- MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. 1. ed. [S.l.]: Prentice Hall, 2017. Citado na página 31.
- MARTINS, F. R. et al. Projeto sonda – rede nacional de estações para coleta de dados meteorológicos aplicados ao setor de energia. Congresso Brasileiro de Energia Solar, n. 1, 2007. Nenhuma citação no texto.
- MONGODB. What is mongodb. 2018. Disponível em: <<https://www.mongodb.com/what-is-mongodb>>. Acesso em: 03 nov. 2018. Citado na página 16.
- MOZILLA. Uma reintrodução ao javascript (tutorial de js). 2018. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/A_re-introduction_to_JavaScript>. Acesso em: 03 nov. 2018. Citado na página 16.

NODEJS. About node.js. 2018. Disponível em: <<https://nodejs.org/en/about/>>. Acesso em: 03 nov. 2018. Citado na página 16.

PRESSMAN, R. *Engenharia de Software: Uma Abordagem Profissional*. 7. ed. [S.l.]: AMGH, 2011. Citado 2 vezes nas páginas 16 e 17.

QUEIROZ, A. Javascript assíncrono: callbacks, promises e async functions. 2018. Disponível em: <<https://medium.com/@alcidesqueiroz/javascript-ass%C3%ADncrono-callbacks-promises-e-async-functions-9191b8272298>>. Acesso em: 30 out. 2018. Nenhuma citação no texto.

REACT. What is react. 2016. Disponível em: <<https://reactjs.org/tutorial/tutorial.html#what-is-react>>. Acesso em: 02 nov. 2018. Citado na página 36.

ROUSE, M. Restful api. 2016. Disponível em: <<https://searchmicroservices.techtarget.com/definition/RESTful-API>>. Acesso em: 02 nov. 2018. Citado na página 32.

SADALAGE, P.; FLOWER, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. 1. ed. [S.l.]: Addison-Wesley Professional, 2012. Nenhuma citação no texto.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.]: Pearson Education, 2011. Citado 5 vezes nas páginas 17, 20, 22, 23 e 26.