Angelo Rodrigo Ribeiro da Silva

Solus, um software para monitoramento de painéis solares

Boituva

2018, v-0.0.1

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: Dr. Marcelo Figueiredo Polido

Boituva

2018, v-0.0.1

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

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

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

Lista de tabelas

Lista de abreviaturas e siglas

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

HTTP HyperText Transfer Protocol (em português Protocolpo 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

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, dentre as tais, a energia solar de destaca por sua baixa implicação a questões ambientais.

A energia solar, vem sendo cada vez mais amplamente utilizada 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 (2018).

Mesmo com o amadurecimento da captação de energia solar no país, o projeto de instalação de uma estação de captura de energia solar, necessita ser feito juntamente com 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 (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.

Subseguindo, 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 Referencial Teórico

TODO: REFERENCIAL TEORICO

3 Desenvolvimento

3.1 Metodologias

3.1.1 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.

3.1.2 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.

3.1.2.1 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.

3.1.2.2 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 acertiva.

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.

3.1.3 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.

3.2 Ferramentas utilizadas

3.2.1 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 e por fornecer uma licença gratuita de sua versão PRO a universidades e estudantes de engenharia de software no geral.

3.2.2 Javascript ES8

3.2.3 Nginx

3.2.4 MongoDB

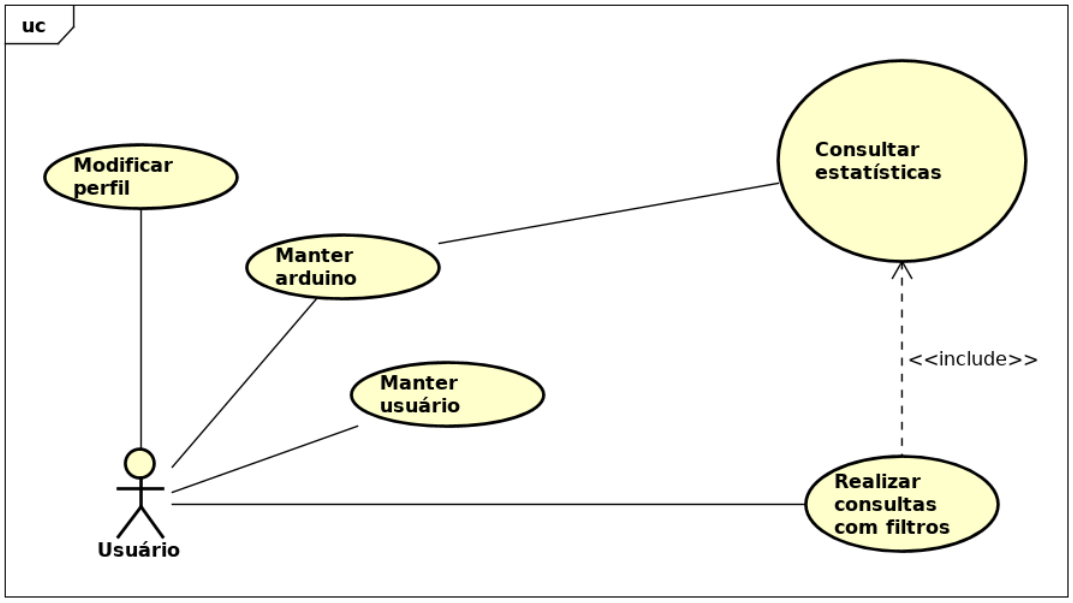
3.3 Engenharia de requisitos

No capitulo, são descritos os requisitos funcionais e não funcionais do sistema, que descrevem o que a aplicação necessita abranger no projeto.

3.3.1 Requisitos funcionais

Para facilitar a visualização dos requisitos funcionais do sistema, optou-se por utilizar o diagrama de descrição de casos de uso da UML.

Figura 1 – Diagrama de caso de uso



Através do diagrama do caso de uso contido na figura 3.3.1, é possível identificarmos os usuários da aplicação, capazes de realizar determinadas ações no sistema, podemos identificar ações que somente são executadas pelo usuário e ações que são executadas de forma autônoma pela aplicação, tais relações de usuários com seus casos de uso são descritos nos capítulos subsequentes.

3.3.1.1 Requisitos funcionais do usuário

Tabela 1 – Especificações do caso de uso capturar dados meteorológicos

Capturar dados meteorológicos	
Descrição	Captura os dados meteorológicos através de sensores conectados ao microcontrolador arduino e os envia através de requisição HTTP para a API
Atores	Sistema
Pré-condições	Credenciais de acesso a API API em correto funcionamento

Tabela 2 – Especificações do caso de uso armazenar dados meteorológicos

Armazenar dados meteorológicos	
Descrição	Após receber os dados captados pelo arduino, a api os valida e então, os armazena em banco de dados
Atores	Sistema
Pré-condições	Banco de dados em correto funcionamento Recebimento de dados através das rotas da API
Exceções e fluxos alternativos	Em caso de dados inválidos, a API os descarta Em caso de perca de conexão com banco de dados, a API os armazena em memória

Tabela 3 – Especificações do caso de uso realizar consultas com filtros

Realizar consultas com filtros	
Descrição	Recebe do usuário os filtros para seleção das informações, então, realiza uma análise estatística dos dados requeridos e os exibe para o usuário utilizando gráficos.
Atores	Usuário
Pré-condições	Credenciais de acesso ao banco de dados para realizar as consultas Banco de dados em correto funcionamento Filtros corretos passados pelo usuário
Exceções e fluxos alternativos	Caso ele não encontre dados na seleção, exibe uma mensagem de não encontrado Em caso de perca de conexão com banco de dados, exibe uma tela de erro ao usuário

Tabela 4 – Especificações do caso de uso realizar análise estatística dos dados

Realizar análise estatística dos dados	
Descrição	Realiza calculo estatístico de informações, retornando informações relevantes como média, moda e desvio padrão.
Atores	Usuário
Pré-condições	Credenciais de acesso ao banco de dados Banco de dados em correto funcionamento
Exceções e fluxos alternativos	Caso não existam dados para serem analisados, joga uma exceção de argumentos inválidos

3.3.2 Requisitos não funcionais

Os requisitos não funcionais do sistema complementam os requisitos funcionais, como melhorias para as especificações.

3.3.2.1 Segurança

O projeto precisa trabalhar de forma segura, então, esse requisito pede para que a API possua autenticação das aplicações clientes e que também, a aplicação web possua autenticação, para o usuário visualizar as informações e realizar consultas, ele precisa estar autenticado.

3.3.2.2 Disponibilidade

Para garantir uma melhor análise e fidelidade dos dados, o sistema precisa funcionar durante 24 horas por dia e 7 dias por semana, para isso, redundâncias precisam ser trabalhadas.

3.3.2.3 Performance

Outro requisito não funcional do sistema é a performance, as informações precisam ser processadas de forma rápida, problemas como lentidão no processamento podem acabar travando a entrada de dados no sistema.

3.3.3 Cenários do sistema

TODO: Cenários do sistema

3.3.4 Arquitetura macro

TODO: Arquiretura macro

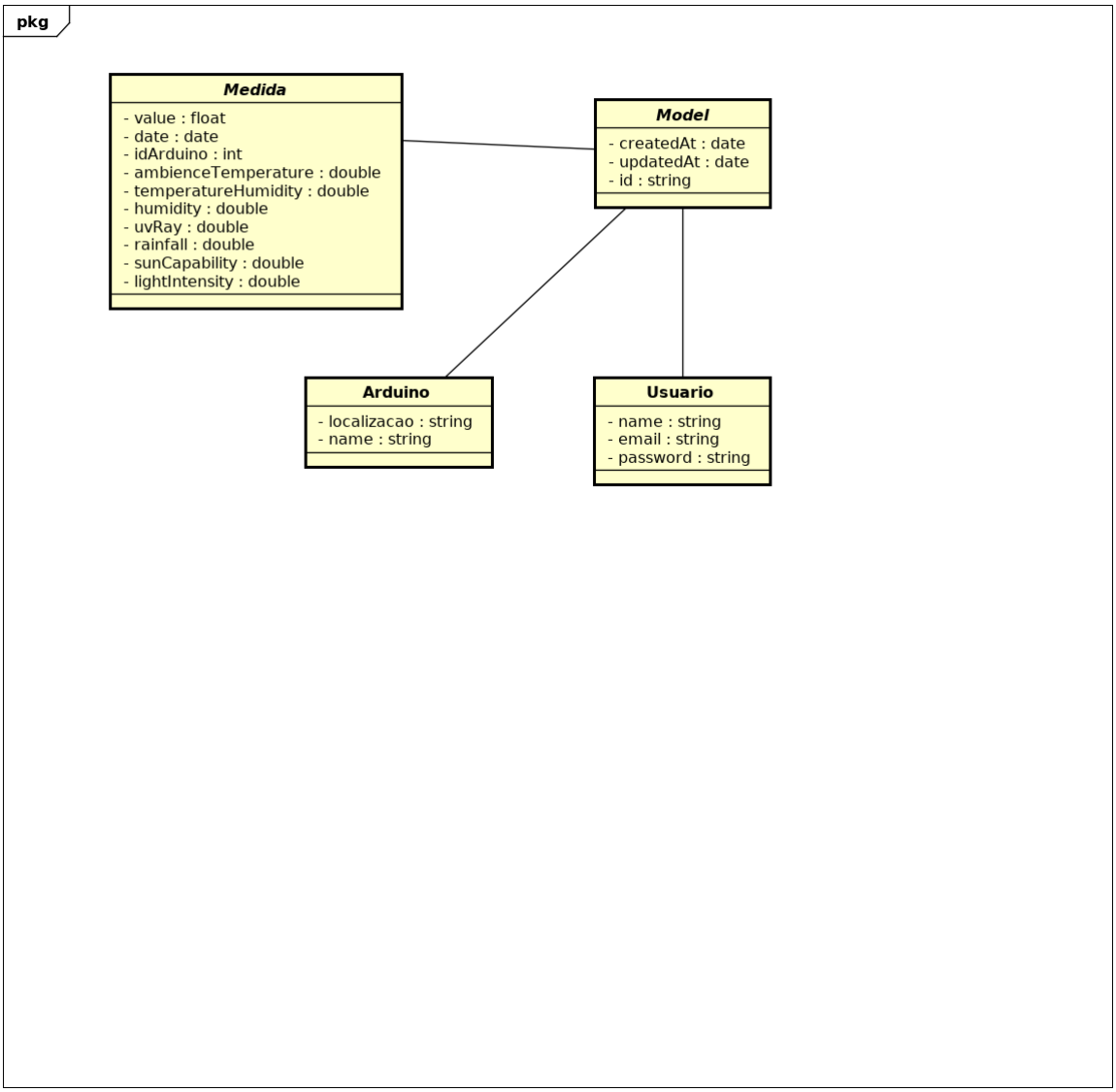
3.4 Diagramas do sistema

TODO: Diagramas do sistema

3.4.1 Diagrama de classes

Como podemos verificar no diagrama descrito na figura 3.4.1 as entidades do sistema consistem na medida, que é a informação que foi capturada pelo arduino, as entidades arduino e usuário também estão presentes, todas extendem da Model do Mongoose.

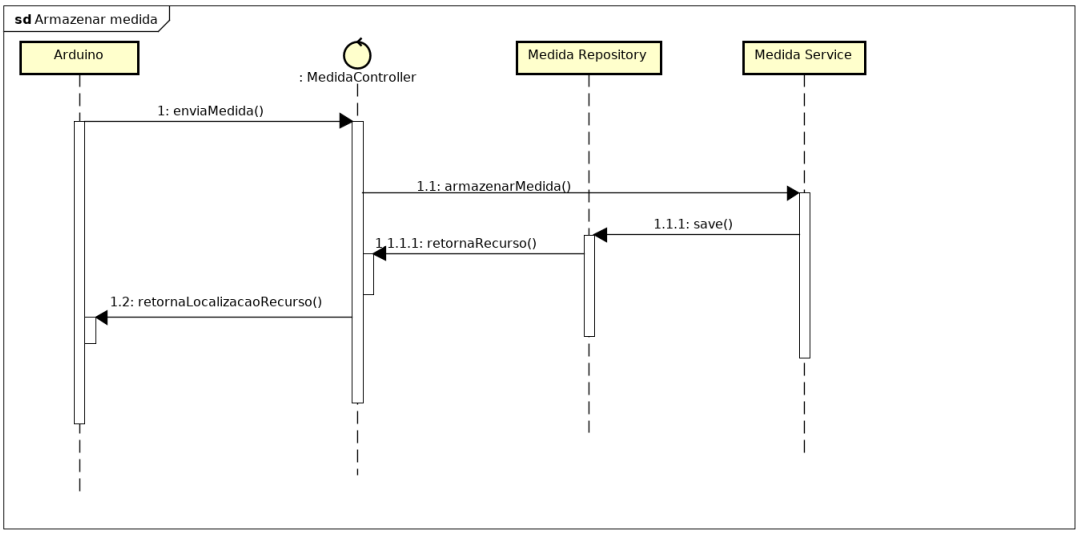
Figura 2 – Diagrama de classes



3.4.2 Diagrama de sequência

Apresentação do diagrama de sequencia do projeto.

Figura 3 – Diagrama de sequência



Como descrito no diagrama de sequência representado na figura 3.4.2 a aplicação é composta por 3 principais camadas.

3.4.2.1 Controlador

Na camada de controle, os dados são recebidos e passam por uma básica validação através, porém não sofrem a interferência das regras de negócio.

Nessa camada, os dados são recebidos e retornados ao cliente, é uma interface de acesso a aplicação, geralmente, essa camada recebe objetos de requisições HTTP.

3.4.2.2 Serviço

Na camada de serviço as ações são os casos de uso, o caso de uso responsável por aquela ação trabalha, inserindo os dados através do repositório no banco de dados, e então, retorna as informações.

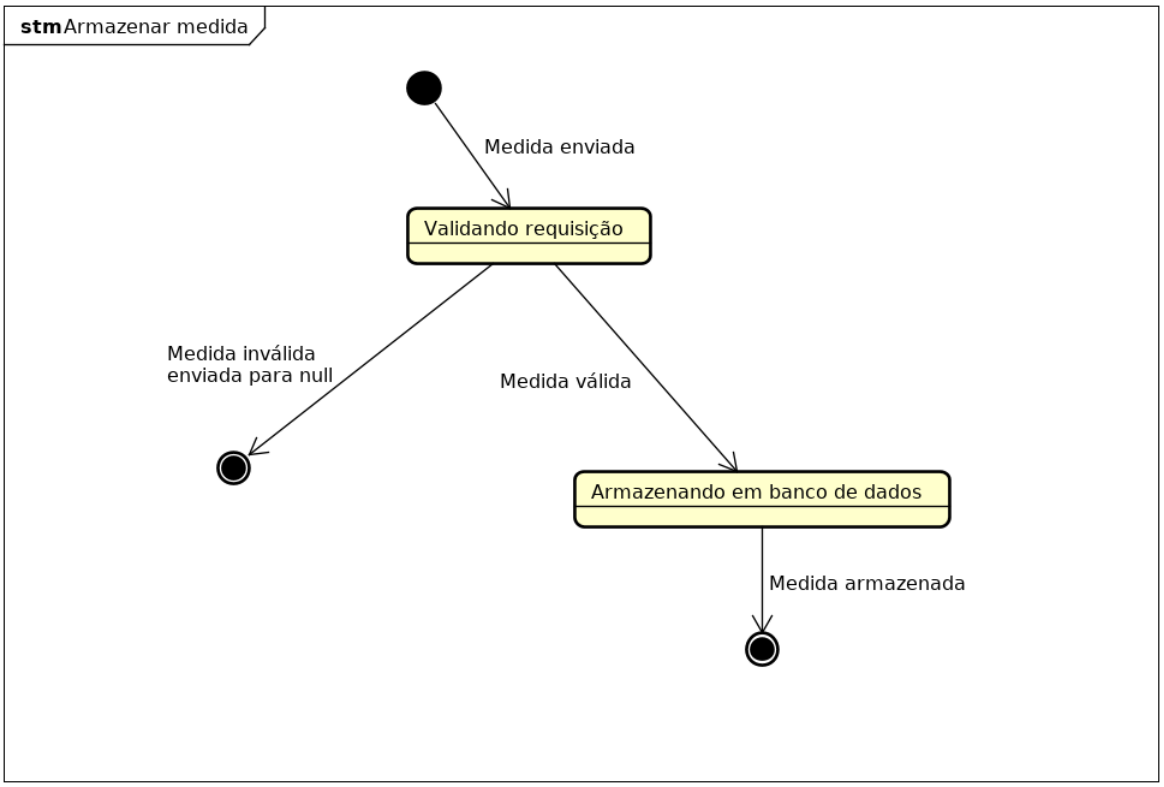
3.4.2.3 Repositório

Dentro da camada de repositório, são recebidos dados que, através de uma camada de infraestrutura são persistidos, retornando então, uma entidade.

3.4.3 Diagrama de estado

Apresentação dos estados de uma entidade de medida que foi capturada ao longo da interação com o sistema.

Figura 4 – Diagrama de estado



Conforme descrito na figura 3.4.3 os estados durante o armazenamento de uma de uma medida são:

3.4.3.1 Validando requisição

A autenticação do arduino na API foi feita e a requisição HTTP para o armazenamento de uma medida foi feita até a API. Nesse estado, a requisição está passando por uma validação básica, verificando os tipos e formato dos dados.

3.4.3.2 Armazenando em banco de dados

A medida é valida e está sendo armazenada no banco de dados, após, ela é retornada a camada de serviço.

3.5 Arquitetura das estações meteorológicas

Foi utilizado para a captura dos dados o microcontrolador arduino, que, com sensores conectados a sua porta serial, prepara os dados e envia uma string json contendo as medidas para o nodeMCU. A placa serial nodeMCU é responsável por se conectar a internet através de Wifi e envia o json para a api.

3.5.1 Sensores

Os sensores do arduino, são ligados através da protoboard até o arduino, onde a captação é feita e as informações são tratadas, os dados são montados dentro de uma string JSON, que é enviada para o nodeMCU.

Não foi utilizada a biblioteca para a montagem da string JSON por conta da memória limitada do controlador.

3.5.2 NodeMCU

Na placa de WiFi NodeMCU, as informações são recebidas através da porta serial e o json recebido é enviado para a api.

3.6 Arquitetura do software

A arquitetura da aplicação desenvolvida pode ser resumida em dados sendo capturados através de sensores conectados a um microcontrolador arduino, serão captadas as seguintes informações: temperatura, umidade do ar, temperatura em relação a umidade, porcentagem de chuva, radiação uv, intensidade luminosa e capacidade solar.

Dados esses, que serão enviados através de requisições HTTP para uma API, serão armazenadas em banco de dados e então, será feita uma análise estatística dessas informações.

A interface do usuário final com a aplicação, será feita através de uma aplicação web, onde os dados analisados serão disponibilizados e o usuário fará consultas a essas informações.

3.6.1 Servidor

TODO: ARQUITETURA DE SERVIDOR

3.6.1.1 Implementação do servidor

TODO: IMPLEMENTACAO DO SERVIDOR

3.6.1.2 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 quis 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 maleaveis, 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.

3.6.1.3 Tratamento dos dados

Os dados foram exibidos da mesma forma que foram capturados, com exceção das informações de intensidade de luz, que foi invertida para melhor exibição no gráfico, e pela medida de nível de chuva, que além de invertida, foi transformada em porcentagem.

3.6.1.4 Análise dos dados

3.6.1.4.1 Médias

A média aplicada em cima dos dados, é separada por um intervalo de tempo, como exemplo, caso o usuário informe que o intervalo requisitado é de uma hora, os dados são agrupados por cada hora e então, uma média é aplicada nesses dados, exibindo as médias em intervalos de horas no gráfico.

3.6.1.4.2 Mínimas e Máximas

TODO: Mínimas e máximas

3.6.1.5 Documentação da API

Podemos conferir abaixo a documentação das rotas da API.

Tabela 5 – 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

3.6.2 Cliente

Foi utilizado para a construção do lado da interface do cliente a biblioteca react, que foi utilizada por sua alta performance na renderização de componentes no navegador, a interface aproveita da funcionalidade de renderização virtual da biblioteca para aumentar a performance e diminuir o custo computacional.

3.6.2.1 Descrição das telas

TODO: DESCRICAO DAS TELAS

4 Conclusão

TODO: Conclusão

4.1 Resultados

TODO: Resultados

4.2 Projeções futuras

TODO: Projeções futuras

Referências

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 12.

BECK, K. *Test Driven Development By Example*. 1. ed. [S.l.]: Yahoo, 2002. Citado na página 16.

BIRD, R.; WADLER, P. *Introduction to functional programming*. 1. ed. [S.l.: s.n.], 1988. Citado na página 15.

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 na página 16.

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-commmunity.html>. Acesso em: 03 nov. 2018. Citado na página 16.

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.

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.

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.