Angelo Rodrigo Ribeiro da Silva

Solus, um software para monitoramento de painéis solares

Brasil

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

Brasil

2018, v-0.0.1

Angelo Rodrigo Ribeiro da Silva
Solus, um software para monitoramento de painéis solares/ Angelo Rodrigo Ribeiro da Silva. – Brasil, 2018, v-0.0.1-
25 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Marcelo Figueiredo Polido

Trabalho de conclusão de curso – INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO – IFSP
Curso de Análise e Desenvolvimento de Sistemas, 2018, v-0.0.1.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

Resumo

O trabalho desenvolvido se baseia na dificuldade durante a análise de dados meteorológicos na prévia instalação de painéis solares. Este documento visa descrever uma aplicação web focada em captar e descrever dados meteorológicos utilizando métricas de estatística.

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

Abstract

The thesis developed is based on the difficulty in the analysis of meteorological data in the installation of solar panels. This document aims to describe a web software focused on capturing and describing meteorological data using statistics metrics.

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

1.1 Contextualização

Na sociedade contemporânea, ao percebermos o problema por trás das fontes de energia elétrica comumente utilizadas, diversas fontes de energia menos degradantes ao meio ambiente vem sendo estudadas.

Em meio ao estudo de diferentes fontes de energia a energia solar vem mostrando seu valor, se sobresaindo quando analisadas questões como facilidade de instalação e manutenção, discrição e custo. Visto que, outras fontes de energia renovavéis são de díficil acesso.

Porém mesmo com as vantagens da utilização de energia solar, a instalação não é trivial, e precisa ser feita sobre uma prévia análise das condições climáticas da região, visando maior aproveitamento dos recursos investidos.

1.2 Tema

O mote acerca do desenvolvimento desse trabalho, gira entorno do auxílio da tomada de decisão através do desenvolvimento de um software de análise e visualização de dados.

1.3 Objetivo

Este desenvolvimento tem como objetivo introduzir a questão da análise de dados meteorológicos e tomada de decisão acerca da instalação de painéis solares e documentar através de metodologias de descrição de software, a arquitetura e desenvolvimento de uma aplicação que visa auxiliar na resolução deste problema.

1.4 Delimitação do Problema

Não existe hoje uma forma prática, de baixo custo e precisa de realizar a análise de dados ante a instalação de painéis solares, visando, com uma massa de dados dispostos de maneira simples e organizada, auxiliar a tomada de decisão na instalação destes painéis, procurando através de variáveis como posição, ângulo entre outras, extrair o máximo de performance na captação de energia solar.

1.5 Justificativa

Existe um projeto de instalação de uma usina solar no IFSP, no campus localizado em Boituva, portanto, o tema do projeto foi escolhido, para que se possa através da análise prévia de dados, se encontrar a melhor configuração para os painéis fotovoltaicos, atingindo assim, uma maior captação de energia e aproveitamento de investimento.

1.6 Método

A metodologia de trabalho escolhida para este projeto, utiliza algumas convenções da metologia SCRUM, porém, pelo tamanho limitado da equipe, o projeto foi trabalhado sendo ditado pela metodologia KANBAN. Para medirmos a qualidade do software, foi decidido optar pelo desenvolvimento da aplicação utilizando-se de técnicas de controle de qualidade orientadas ao teste.

2 Descrição Geral do Sistema

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.

2.1 Principais Envolvidos e suas Características

2.1.1 Usuários do Sistema

O sistema visa atender especialistas que precisam realizar tomadas de decisão.

Isso inclui também, clientes que, antes de realizar a instalação de painéis solares, precisam analisar se o investimento será compensado. E também, empresas de instalação de painéis solares, que gostariam de fazer uma análise de viabilidade mudando local, angulo e fazendo outras pesquisas acerca da instalação ou da manutenção de painéis fotovoltaicos.

2.1.2 Desenvolvedores do Sistema

Os envolvidos no desenvolvimento do projeto, são o orientador, Dr. Marcelo Polido, que ficará responsável pelos requisitos do sistema, ele irá coordenar o que será implementado e irá ditar as entregas incrementais. Também responsável por requisitos do projeto está o Professor Mario Pin, que será algo próximo de um Product Owner, ele será o primeiro cliente final da aplicação, irá utilizar o sistema para realizar análise de dados.

O planejamento e desenvolvimento do projeto, ficará por conta do aluno responsável pela defesa do mesmo, Angelo Silva.

O projeto é open source, ou seja, aberto para a comunidade no github, recebendo então, pequenas contribuições esporádicas de outros desenvolvedores ao longo do ciclo de vida do projeto.

2.1.3 Tecnologias Empregadas

A aplicação foi desenvolvida utilizando arduino para gerenciamento e captura dos dados utilizando requisições HTTP através das bibliotecas do arduino. A conexão com a internet foi feita utilizando um nodeMCU Wifi, as informações são capturadas, é feita uma validação e formatação básica dos dados e então os mesmos são enviados para uma RESTFUL API construída com NODEJS.

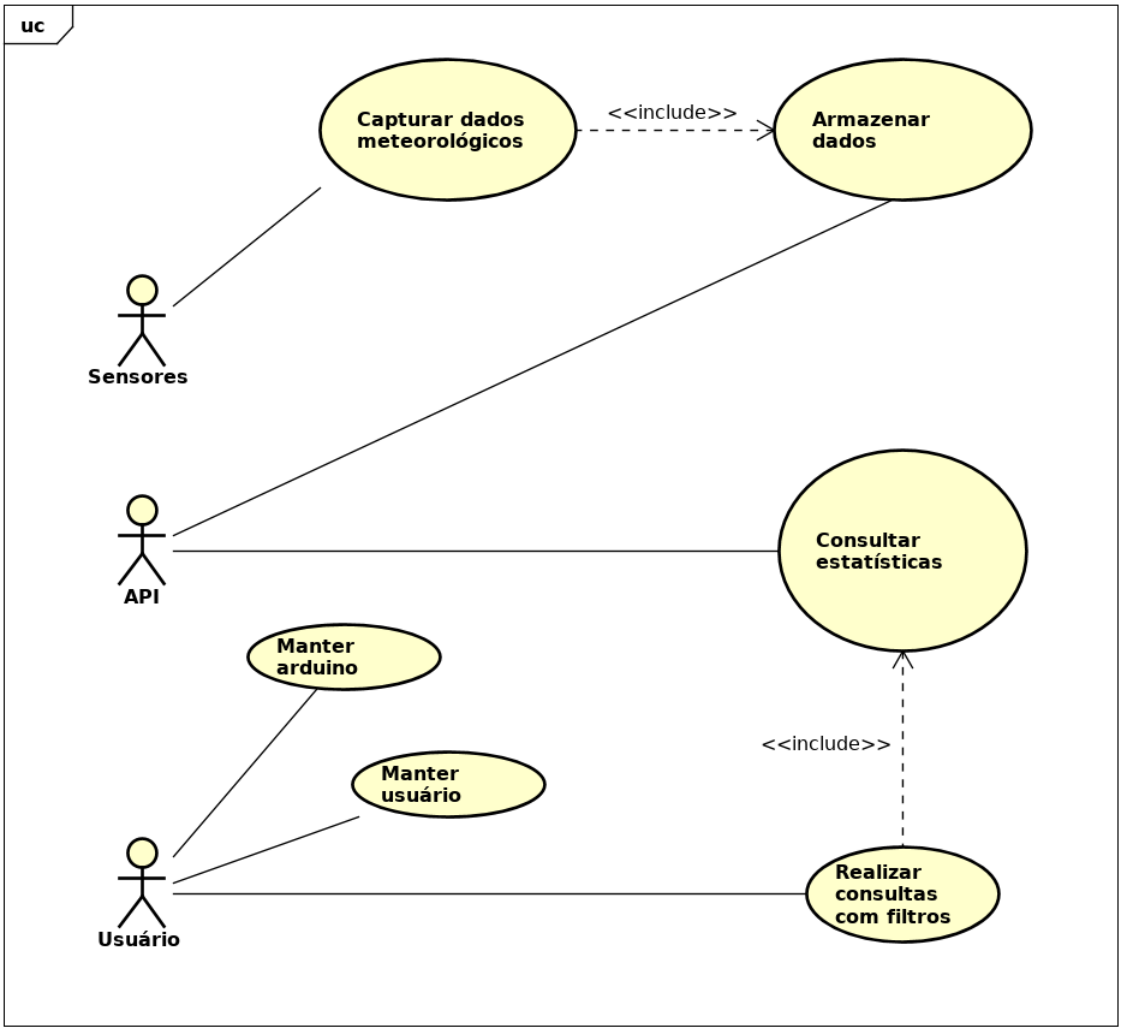
O banco de dados utilizado foi o MongoDB, com um framework de mapeamento chamado Mongoose,

3 Requisitos do sistema

3.1 Requisitos funcionais

Os requisitos funcionais do sistema são definidos pelas características para que um MVP possa ser entregue. Os requisitos são descritos no diagrama de caso de uso abaixo.

Figura 1 – Diagrama de caso de uso



Os atores do sistema foram definidos com o o próprio sistema, que poderá realizar acesso a alguns casos de uso e o usuário final. Seguem abaixo as especificações dos casos de uso.

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
Exceções e fluxos alternativos	Em caso de perca de conexão com internet ou a API, armazena as informações temporariamente no arduino

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

3.2 Requisitos não funcionais

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

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.

3.2.2 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.

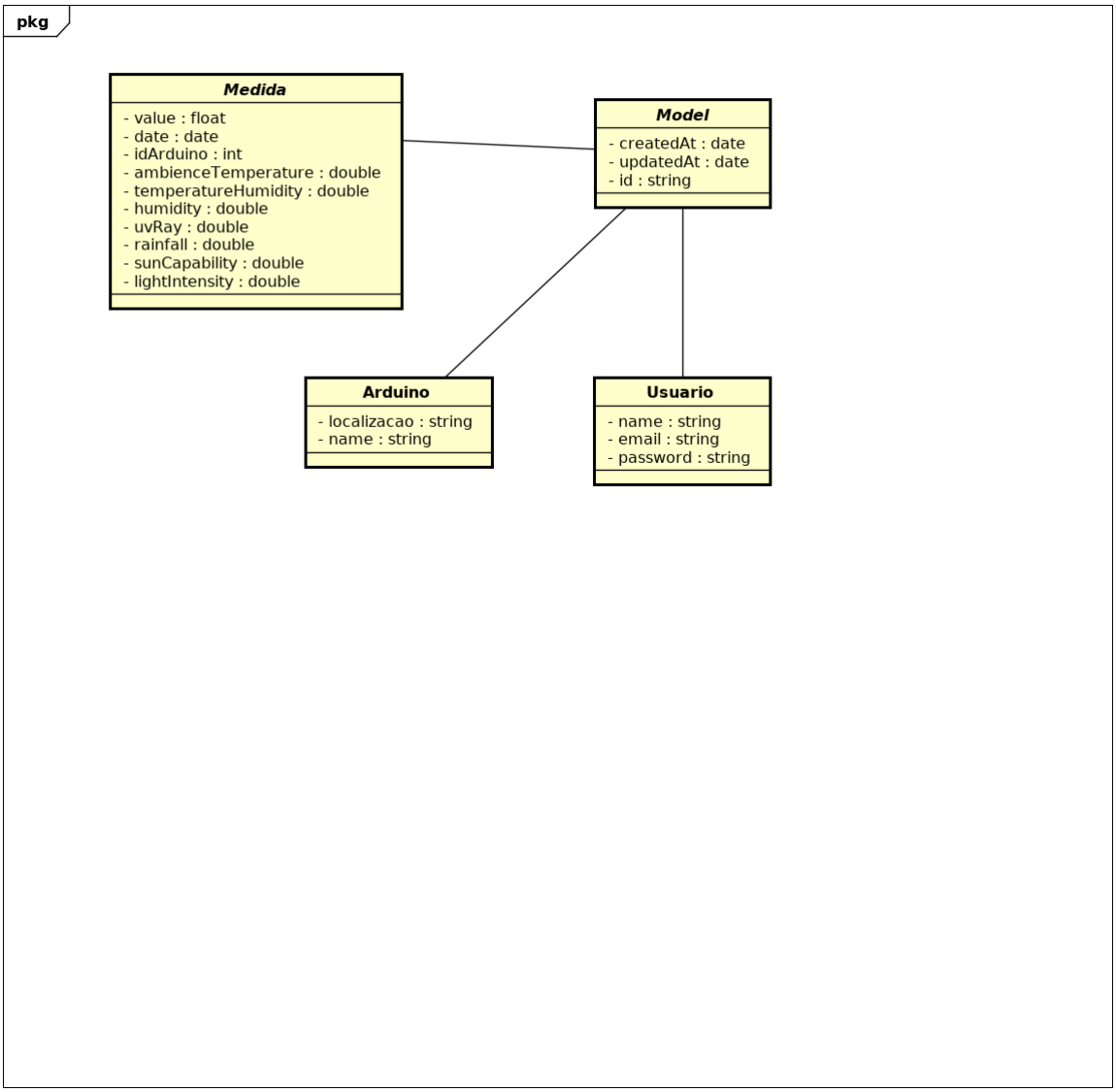
3.2.3 Performance

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.

4 Diagrama de classes

Como podemos verificar no diagrama descrito na figura 4 as entidades do sistema consistem na medida, que é a informação que foi capturada pelo arduino, uma entidade abstrata que é entendida pelas classes concretas das medidas, temos também as entidades que representam o arduino, possuindo a localização do mesmo e o usuário, ambos extendendo da classe autentical.

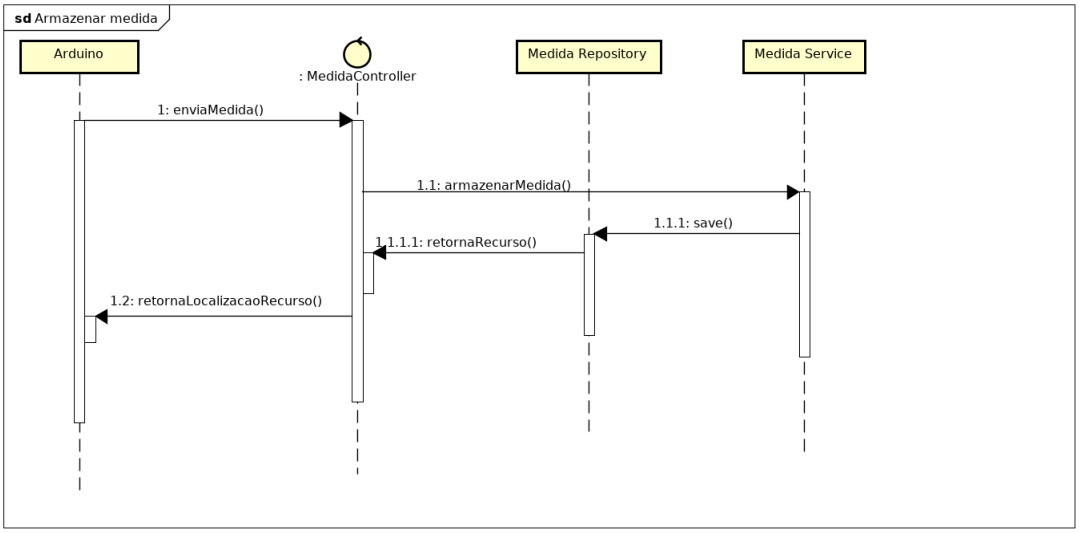
Figura 2 – Diagrama de classes



5 Diagrama de sequência

Apresentação do diagrama de sequencia do projeto.

Figura 3 – Diagrama de sequência



Como descrito no diagrama de sequência representado na figura 5 a aplicação é composta por 3 principais camadas.

5.1 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.

5.2 Serviço de validação

No serviço de validação, regras de negócio são aplicadas a medida para verificar se a mesma é valida para ser armazenada.

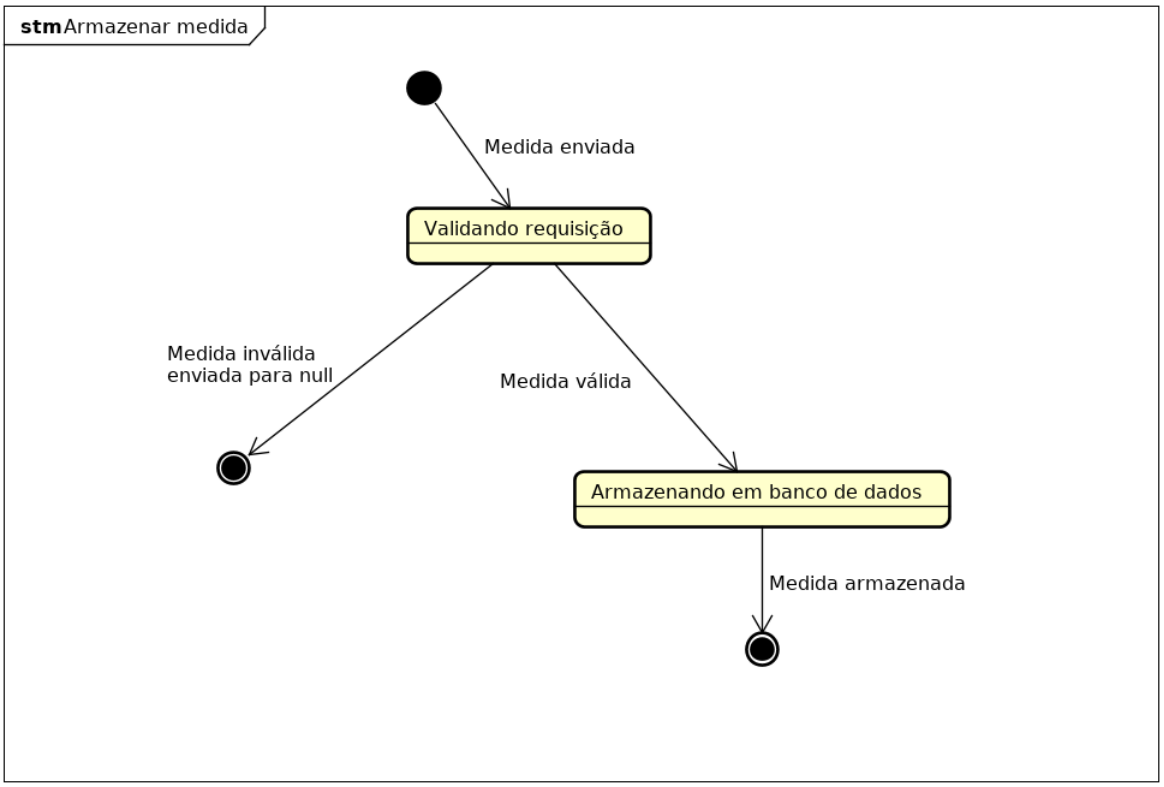
5.3 Repositório

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

6 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 6 os estados durante o armazenamento de uma de uma medida são:

6.1 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.

6.2 Validando regras de negócio

A medida passou pelo controlador e está sendo validada através dos filtros, onde as regras de negócio de validação estão verificando se aquela medida faz sentido, baseando-se

nas medidas captadas anteriormente.

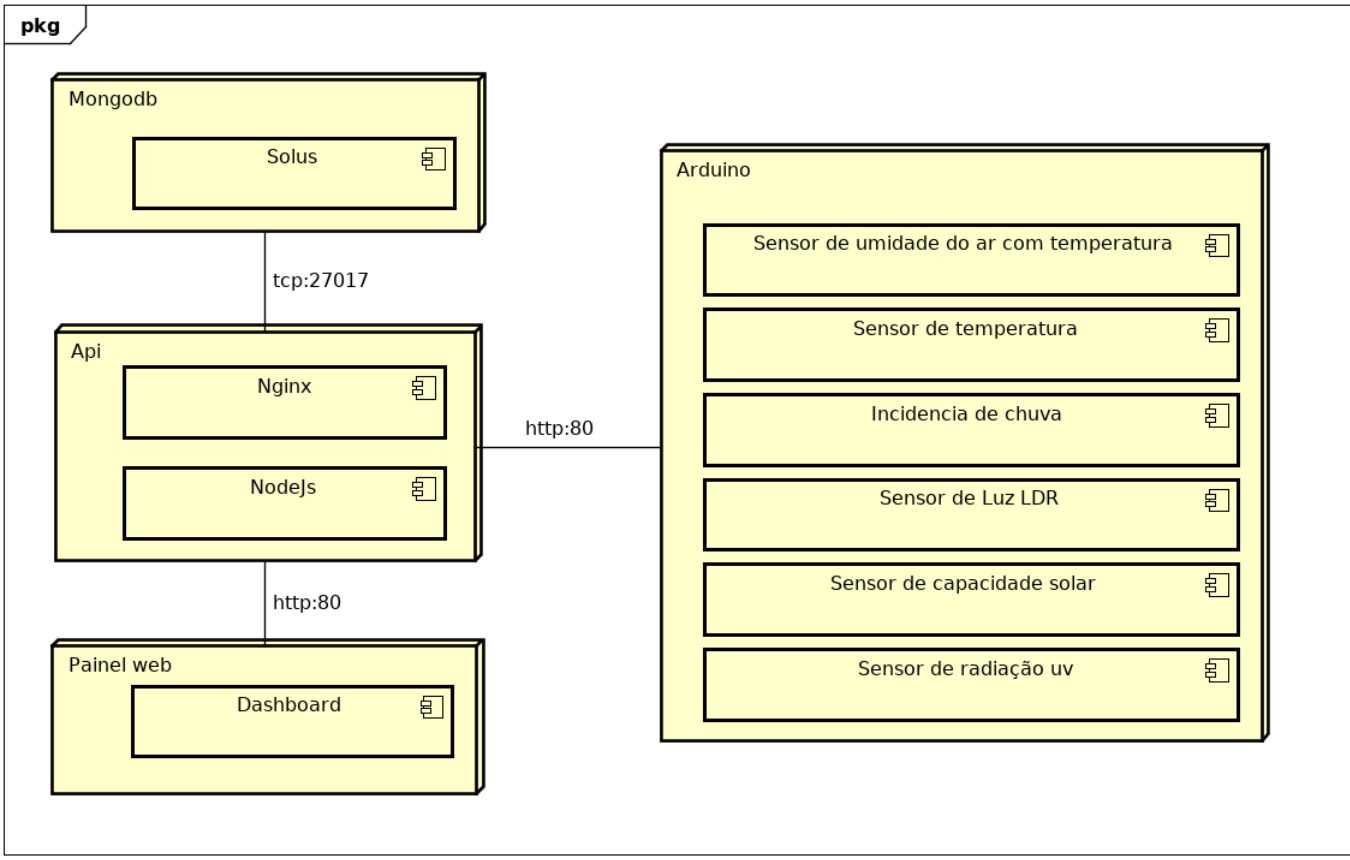
6.3 Armazenando em banco de dados

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

7 Diagrama de implementação

A seguir o diagrama de implementação do sistema, que descreve a forma como os componentes de hardware e software interagem entre si.

Figura 5 – Diagrama de implementação



8 Descrição e arquitetura do 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"o modelo do banco de dados pode ser definido como, cada informação capturada fica armazenada dentro de uma coleção, utilizando o formato JSON.

9 Descrição e 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	/arduino/:id	Atualiza os dados de um arduino
DELETE	/arduino/:id	Deleta um arduino e suas medidas capturadas
GET	/arduino:id/measure	Retorna as medidas capturadas por um arduino
POST	/arduino/:id/measure	Cadastra uma medida em um arduino
GET	/measure/:id	Retorna os dados de uma medida
POST	/authenticate	Retorna o Json Web Token de uma credencial
POST	/location	Cria uma nova localização
GET	/location/:id	Retorna os dados de uma localização
POST, PATCH	/location/:id	Atualiza os dados de uma localização
DELETE	/location/:id	Deleta uma localização

9.1 Métodos HTTP utilizados

Como podemos visualizar na tabela 5, o método POST fica designado a criar um recurso quando a requisição for enviada para uma rota sem a identificação. O método POST também é utilizado para atualizar um recurso quando a requisição for enviada para um recurso identificado. Para a atualização de um recurso também pode ser utilizado o método PATCH. Para a consulta de recursos é utilizado o método GET, o método DELETE é utilizado para excluir um recurso.

9.2 Autenticação

Uma exceção do método POST é na autenticação, os campos nome de usuário e senha são enviados para a API, que retorna um Json Web Token para que possa ser feita a autenticação de forma moderna.

10 Conclusão

Conclusão

Referências

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.

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.

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.