

UTFPR-UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Bacharelado em Engenharia de Software - 6º Período

DISCIPLINA: *Oficina de Integração 2 – IF66K-ES61*

PROFESSOR: *André Luiz Przybysz*

Documento de Projeto de Software

No Céu Tempão

Amilton Fontoura de Camargo Junior

Carlos Henrique dos Santos

Tiago Pagotto

Cornélio Procópio

2017

Sumário

Introdução	3
Contexto	3
Justificativa	3
Proposta	4
Organização do Documento	4
Descrição Geral do Sistema	5
Objetivos (Gerais e Específicos)	5
Limites e Restrições	6
Descrição dos Usuários do Sistema	6
Desenvolvimento do Projeto	7
Tecnologias e ferramentas	7
Metodologia de desenvolvimento	8
Cronograma previsto	9
Requisitos do Sistema	10
Requisitos Funcionais	10
Tabela 2 - Requisitos funcionais do projeto.	11
Requisitos Não-funcionais	11
Diagramas de Casos de Uso	12
Análise do Sistema	15
Arquitetura do Sistema	15
Modelo do Banco de Dados	16
Modelo Conceitual	16
Modelo Lógico	17
Dicionário de dados	18
Diagrama de Classes	20
Diagrama de Atividades	20

Implementação	25
Protótipos de Telas	25
Descrição do código	25
Considerações Finais	27
Bibliografia	28

1 Introdução

1.1 Contexto

O clima é um fator fundamental para a vida na terra. Esta influência que o clima possui no âmbito social traz fascínio e medo à humanidade desde os primórdios. Para melhor compreender e não ser surpreendida, a humanidade observa e tenta prever padrões climáticos. Contudo, o clima não possui fácil previsão, pois existem n variáveis neste processo e, em geral, toda previsão feita em uma região é resultado de alguns, mas não todos, fatores que mais influenciam o tempo daquela região.

Desta maneira, neste documento é proposto um modelo de medição de temperatura e umidade para a cidade de Cornélio Procópio. Esta análise será realizada com o uso de um sistema e um microcontrolador Arduino em um local físico da Universidade Tecnológica Federal do Paraná (UTFPR). Desta forma, os alunos, professores, servidores e visitantes da universidade podem ter uma noção real de como está o clima onde estudam e trabalham e, através de um histórico de temperaturas e umidades dos últimos dias, torna-se possível analisar o tempo nesta região.

1.2 Justificativa

É nítida a volatilidade do clima na região de Cornélio Procópio. Um exemplo claro desta afirmação pode ser notado em um dia de primavera ou até mesmo de outono, onde nestas épocas o clima é muito instável e pode-se notar sintomas das 4 estações do ano em um único dia.

Alunos, colaboradores e servidores estão sujeitos e quase nunca preparados para as variações climáticas. Através deste projeto, pretende-se

implantar, subjetivamente, uma estação de monitoramento de temperatura e umidade na UTFPR-CP. Esta estação fornecerá os dados meteorológicos através de uma página web desenvolvida por nosso grupo de trabalho.

1.3 Proposta

O sistema a ser desenvolvido servirá como um canal informativo sobre as condições atuais e passadas do clima na UTFPR. À partir das leituras medidas pelos sensores de temperatura e umidade, um banco de dados armazenará as informações e, assim, uma página web poderá exibir seu conteúdo de forma numérica, gráfica e dinâmica.

Como atualmente não existe tal sistema em funcionamento no câmpus, seu impacto será gradual e de maneira positiva, pois será um canal a mais de informações para aqueles que desejam monitorar a temperatura e umidade nesta região.

1.4 Organização do Documento

Este documento foi seccionado em oito partes principais para melhor entendimento do sistema a ser desenvolvido.

Inicialmente, na introdução, cita-se brevemente as motivações que fizeram com que fosse escolhido o desenvolvimento desse projeto. Este segmento é subdividido em quatro tópicos, são eles: contexto, justificativa, proposta e organização do documento.

Posteriormente, na segunda seção do trabalho, é apresentada uma descrição geral do sistema, sub seccionada em: objetivos (gerais e específicos), limites e restrições, além da descrição dos usuários do sistema.

Na seção seguinte (terceira), são contemplados os temas ligados ao

desenvolvimento do projeto. Nesta, são detalhadas as tecnologias e ferramentas, a metodologia e o cronograma previsto para a produção do sistema.

O quarto tópico deste documento, requisitos do sistema, aborda os requisitos funcionais e não funcionais e, complementarmente, nesta subdivisão do trabalho são apresentados os diagramas de casos de uso do software.

Sequencialmente, é feita uma análise física do sistema, a qual trata dos tópicos: arquitetura do sistema, modelo de banco de dados (BD), modelo conceitual, modelo lógico, dicionário de dados, diagrama de classes e diagrama de atividades.

Tendo todos estes artefatos das seções anteriores do documento, torna-se possível a implementação (sexto tópico) e, nesta parte do trabalho, são disponibilizados os protótipos de telas e a descrição do código.

Por fim, são tratadas as considerações finais e a bibliografia utilizada para a realização deste documento, respectivamente nas seções sétima e oitava.

2 Descrição Geral do Sistema

2.1 Objetivos (Gerais e Específicos)

Objetivos gerais:

Para este projeto, o objetivo principal é gerar um website de monitoramento climático para a UTFPR. Juntamente com este website, será montada uma estação climática em Arduino, para funcionar como um servidor de dados climáticos em tempo real. Para a exibição no website, um servidor web também será criado, para que o mesmo obtenha os dados da estação, salve-os em um banco de dados e, também, processe as informações e exiba-as na página. A página web exibirá, de

forma textual e gráfica, o histórico climático do câmpus, baseando-se nos dados armazenados no banco de dados do servidor.

Objetivos específicos:

- Criar uma estação de coleta de dados climáticos;
- Tornar a estação acessível remotamente via wireless;
- Criar um servidor web com banco de dados;
- Tornar o servidor web operante sob um computador local;
- Criar uma página web exibindo os dados climáticos armazenados no servidor;
- Tornar a página dinâmica, com dados textuais e gráficos.

2.2 Limites e Restrições

Este projeto não contemplará:

- Um servidor web operante em qualquer região (será um servidor local);
- O monitoramento do clima em regiões fora da UTFPR;
- O monitoramento de chuva (apenas de umidade relativa do ar e temperatura);
- Previsões climáticas.

2.3 Descrição dos Usuários do Sistema

O sistema destina-se à comunidade em geral, mas especialmente aos alunos, professores, servidores e visitantes da UTFPR câmpus Cornélio Procopio. Os usuários/atores são:

- **Estação meteorológica** - dispositivo que, através dos sensores, faz a leitura dos dados do ambiente e os retorna por meio de uma conexão

wi-fi (sem fio);

- **Servidor de banco de dados** - processo lógico em um computador que armazena e recupera informações;
- **Servidor web** - processo lógico em um computador que processa as informações programadas em páginas PHP e retornam ao usuário como páginas HTML;
- **Usuário** - internauta que acessou a página web e terá acesso aos dados climáticos.

3 Desenvolvimento do Projeto

3.1 Tecnologias e ferramentas

As tecnologias utilizadas neste projeto serão:

- **Microcontrolador:** Arduino BlackBoard v1.0;
- **Circuitos:** Protoboard;
- **Leitura:** Sensor de temperatura e umidade DHT11;
- **Comunicação sem fio:** Módulo wi-fi ESP8266;
- **Comunicação com fio:** Fios de cores sortidas para ligação dos componentes;
- **Servidor local:** Servidor MAMP PRO v3;
- **Processamento de scripts:** Servidor PHP v5.6.2;
- **Banco de dados:** MySQL v4.2.10;
- **Modelagem:** Astah Professional v7.1;
- **Prototipação:** Balsamiq Mockups v3;
- **Documentação:** Google Docs;
- **Edição de código:** Adobe Dreamweaver CS5, Notepad++ v7.3.2;
- **Edição de imagens:** Adobe Fireworks CS5, Adobe Photoshop CC.

3.2 Metodologia de desenvolvimento

A metodologia usada para o desenvolvimento do software será uma customização do *framework* Scrum, devido às suas características fundamentais (agilidade, interatividade e incrementabilidade). Além destas características, a escolha pelo Scrum se justifica pelo fato de já ter sido utilizado com sucesso em projetos anteriores.

O Scrum é uma metodologia de desenvolvimento de software, que iniciou com o Manifesto Ágil, e assim simplificar o modelo vigente de desenvolvimento em cascata para mudar o paradigma e passar a trabalhar com interações. É baseado em pequenas equipes. Ele permite a comunicação entre os membros da equipe. Entretanto, há uma grande quantidade de softwares desenvolvidos por programadores solos. Um software sendo desenvolvido por um só programador pode ainda se beneficiar de alguns princípios do Scrum, como: um backlog de produto, um backlog de sprint, um sprint e uma retrospectiva de sprint. Scrum Solo é uma versão adaptada para uso de programadores solo. [5]

A base fundamental da metodologia **Scrum** é composta por: [6]

Product Owner: é o ponto central com poderes de liderança sobre o produto. Ele é o único responsável por decidir quais recursos e funcionalidades serão construídos e qual a ordem que devem ser feitos. É responsabilidade dele manter e comunicar a todos os outros participantes uma visão clara do que a equipe Scrum está buscando alcançar no projeto. Como tal, ele é responsável pelo sucesso global da solução. Para garantir que a equipe construa rapidamente o que o Product Owner precisa, ele deve colaborar ativamente com o ScrumMaster e equipe de desenvolvimento e deve estar disponível para responder às perguntas tão logo estas são feitas.

ScrumMaster é responsável por ajudar a todos os envolvidos a entender e abraçar os valores, princípios e práticas do Scrum. Ele age como

um Coach, executando a liderança do processo e ajudando a equipe Scrum (e o resto da organização) a desenvolver sua própria abordagem do Scrum, que tenha a melhor performance, respeitando as particularidades da organização. O ScrumMaster também tem um papel de facilitador. Ele deve ajudar a equipe a resolver problemas e fazer melhorias no uso do Scrum. Ele também é responsável por proteger a equipe contra interferências externas e assume um papel de liderança na remoção de impedimentos que podem atrapalhar a produtividade.

Time Scrum: no desenvolvimento tradicional de software são abordados vários tipos de trabalho, tais como: arquiteto, programador, testador, administrador de banco de dados, Designer, e assim por diante. No Scrum é definido o papel do Time de Desenvolvimento, que é simplesmente a junção de todas essas pessoas em uma equipe multidisciplinar, e que são responsáveis pela concepção, construção e testes do produto. A ideia principal é que a equipe de desenvolvimento se auto-organiza para determinar a melhor maneira de realizar o trabalho para atingir a meta estabelecida pelo Product Owner.

Com a execução, geram-se os artefatos listados a seguir.

Product Backlog: no Scrum, sempre se faz o trabalho mais importante primeiro. O Product Owner, com ajuda do resto da equipe Scrum e as partes interessadas, é o responsável por determinar e gerir a sequência deste trabalho e comunicando-o na forma de uma lista de prioridades conhecida como o Product Backlog. O Product Owner, em conjunto com as demais partes interessadas no produto, definem os itens do Product Backlog. Em seguida, ele garante que os itens do Backlog são colocadas na sequência correta (usando fatores como valor, custo, conhecimento e risco), de modo que os itens de alto valor, aparecerá no topo do backlog do produto e os itens de menor valor aparecer em direção ao fundo. O Product backlog é um documento que está constantemente evoluindo. Os itens podem ser adicionados, excluídos e revisto pelo Product Owner por conta de mudanças

nas condições de negócios, ou conforme a compreensão da equipe Scrum sobre o produto aumenta. Em geral a atividade de criar e de refinar os itens do product backlog, estimando o tamanho e esforço de cada item, é chamada de *Grooming*.

Sprints: no Scrum, o trabalho é realizado em iterações ou ciclos de até um mês de calendário chamado de Sprints. O trabalho realizado em cada sprint deve criar algo de valor tangível para o cliente ou usuário. Sprints são *timeboxed* (duração fixa) para que tenham sempre um início e fim data fixa, e, geralmente, todos eles devem estar com a mesma duração.

Sprint Planning: o product backlog pode representar muitas semanas ou até meses de trabalho, o que é muito mais do que pode ser concluído em um único e curto sprint. Para determinar quais os subconjuntos de itens do Product Backlog mais importantes para construir no próximo sprint, o product owner, junto com o time de desenvolvimento e ScrumMaster, devem realizar o Sprint Planning (planejamento de sprint). Durante o planejamento do sprint, a equipe de desenvolvimento e o product owner devem chegar a um acordo sobre qual o Objetivo do Sprint. Com este objetivo em mãos, eles determinam quais os itens do backlog devem ser priorizados para serem executados neste Sprint.

Daily Scrum: todos os dias, idealmente no mesmo horário, os membros da equipe de desenvolvimento devem realizar uma reunião com tempo definido (15 minutos ou menos), chamado Daily Scrum. Esta reunião também é muitas vezes chamada de Stand-Up Meeting, por causa de uma prática recomendada para que a reunião seja feita em pé (com a intenção de fazer com que a reunião seja rápida). Uma abordagem comum nesta reunião é o Scrum Master perguntar para cada membro da equipe três perguntas:

1. O que fiz ontem que ajudou o time a atingir a meta do sprint?
2. O que vou fazer hoje para ajudar o time a atingir a meta do sprint?
3. Existe algum impedimento que não permita a mim ou ao time atingir

a meta do sprint?

Definition of Done (Definição de Pronto): no Scrum considera-se como resultado do Sprint produto ou funcionalidade concluída. Embora isso varie significativamente de um extremo ao outro para cada time Scrum, os integrantes devem ter um entendimento compartilhado do que significa o trabalho estar completo, assegurando a transparência. Esta é a “Definição de Pronto” para o Time Scrum e é usado para assegurar quando o trabalho está completado no incremento do produto.

Sprint Review (Revisão do Sprint): no final do Sprint, existem duas atividades adicionais que são fundamentais. Uma delas é chamada Sprint Review. Esta é uma reunião informal, e a apresentação do incremento destina-se a motivar e obter comentários e promover a colaboração.

Sprint Retrospective (Retrospectiva do Sprint): enquanto o objetivo do Sprint Review é verificar necessidades de adaptações no produto, o Sprint Retrospective tem como objetivo verificar necessidades de adaptações no processo de trabalho. A Retrospectiva do Sprint ocorre depois da Revisão da Sprint e antes da reunião de planejamento da próxima Sprint. Esta é uma reunião time-boxed de três horas para uma Sprint de um mês.

Portanto, devido ao curto tempo disponível para desenvolvimento, revisão e testes em nosso cronograma, os Sprints (período de implementação) terão duração máxima de 1 semana e, quando finalizado, haverá uma reunião de, no máximo, 15 minutos, para demonstração e validação/incrementação das funcionalidades implementadas no Sprint anteriormente citado.

O Scrum será estruturado conforme a imagem apresentada abaixo.

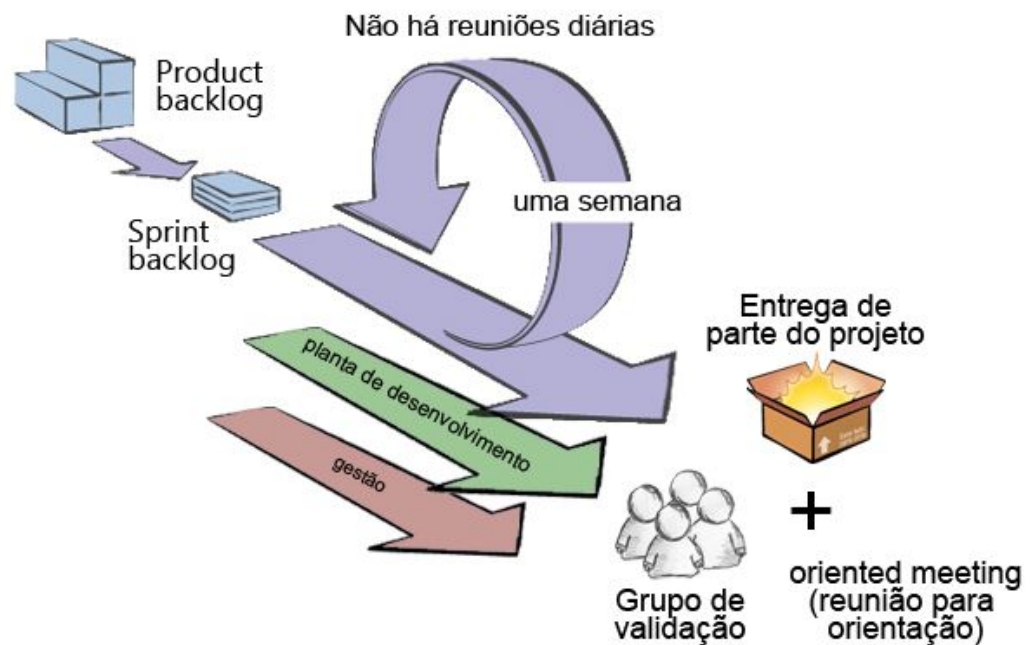


Figura 1 - Estrutura customizada do *framework* Scrum.

Serão colecionados em nosso repositório do processo os seguintes artefatos: protótipo do software, Product backlog (lista de funcionalidades a serem implementadas), Sprint backlog (funcionalidades que serão implementadas naquele Sprint) e cronograma.

3.3 Cronograma previsto

PERÍODO DE TEMPO	ATIVIDADE A SER REALIZADA	RESPONSÁVEL
11/04 até 14/04	Diagramação do projeto	Amilton
15/04	Início do desenvolvimento e divisão de tarefas	Amilton

16/04 até 23/04	1° Sprint (implementação das funcionalidades do Sprint backlog)	Tiago
24/04	Reunião de validação e/ou incrementação do 1° Sprint	Carlos
25/04 até 02/05	2° Sprint (implementação das funcionalidades do Sprint backlog)	Tiago
03/05	Reunião de validação e/ou incrementação do 2° Sprint	Tiago
04/05 até 11/05	3° Sprint (implementação das funcionalidades do Sprint backlog)	Carlos
12/05	Reunião de validação e/ou incrementação do 3° Sprint	Carlos
13/05 até 20/05	4° Sprint (implementação das funcionalidades do Sprint backlog)	Tiago
21/05	Reunião de validação e/ou incrementação do 4° Sprint	Tiago
22/05 até 29/05	5° Sprint (implementação das funcionalidades do Sprint backlog)	Carlos
30/05	Reunião de validação e/ou incrementação do 5° Sprint	Carlos
31/05 até 07/06	Finalização da implementação e revisão do código	Amilton
08/06 até 15/06	Fase de testes do software finalizado	Amilton
16/06 até 21/06	Correção de possíveis erros e bugs não	Todos

	corrigidos previamente	
22/06	Entrega do software completo e com funcionamento perfeito (como contratado) ao cliente	Todos

Tabela 1 - Cronograma de desenvolvimento do projeto.

4 Requisitos do Sistema

4.1 Requisitos Funcionais

A tabela 2 seguinte ilustra os requisitos funcionais do projeto.

ID	REQUISITO FUNCIONAL	PRIORIDADE
RF01	O sistema deve medir a umidade relativa do ar.	Essencial
RF02	O sistema deve medir a temperatura do ar.	Essencial
RF03	O sistema deve funcionar 100% online.	Importante
RF04	O sistema deve possuir um banco de dados no mesmo servidor da página web.	Importante
RF05	O sistema deve exibir um histórico de temperaturas na região.	Desejável
RF06	O sistema deve exibir um histórico da umidade relativa do ar na região.	Desejável
RF07	O sistema deve exibir a temperatura média na região no histórico.	Importante
RF08	O sistema deve exibir a umidade relativa do ar média na região no histórico.	Importante

Tabela 2 - Requisitos funcionais do projeto.

4.2 Requisitos Não-funcionais

A tabela seguinte ilustra os requisitos não funcionais do projeto.

ID	REQUISITO NÃO FUNCIONAL	CATEGORIA
----	-------------------------	-----------

RNF01	O sistema deve ser implementado em PHP 5 e HTML.	Implementação
RNF02	O sistema deve emitir um alerta caso a temperatura ultrapasse os 30 graus célsius.	Implementação
RNF03	O sistema deve emitir um alerta caso a temperatura diminua abaixo de 10 graus célsius.	Implementação
RNF04	As informações do clima devem ser atualizadas para exibição ao usuário no máximo de 1 em 1 hora.	Confiabilidade
RNF05	O usuário deve ser capaz de ver como está o clima atual em até 10 segundos.	Usabilidade
RNF06	O sistema deve ser capaz de ser visto nos principais navegadores.	Implementação
RNF07	O sistema deve mostrar um histórico com os dados dos últimos 7 dias.	Usabilidade
RNF08	O sistema deve atualizar o banco de dados a cada 60 segundos.	Confiabilidade
RNF09	A página web deve ser no padrão adaptável a monitores de resolução 1024x768 pixels.	Usabilidade
RNF10	O sistema deve fazer um backup diário do banco de dados.	Segurança

Tabela 3 - Requisitos não funcionais do projeto.

4.3 Diagramas de Casos de Uso

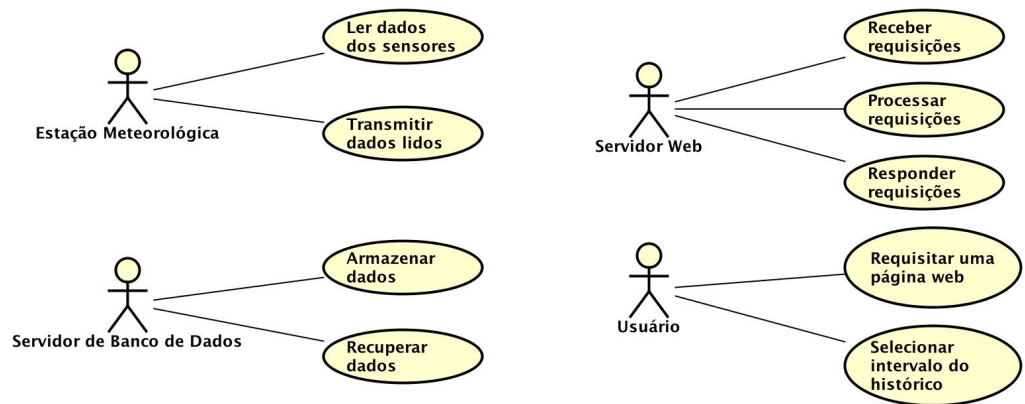


Figura 2 - Caso de uso: visão geral.

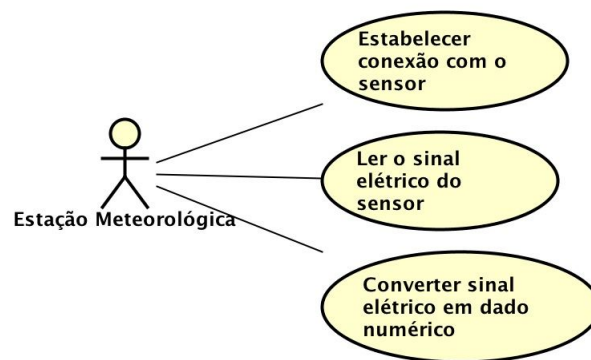


Figura 3 - Caso de uso: ler dados dos sensores.

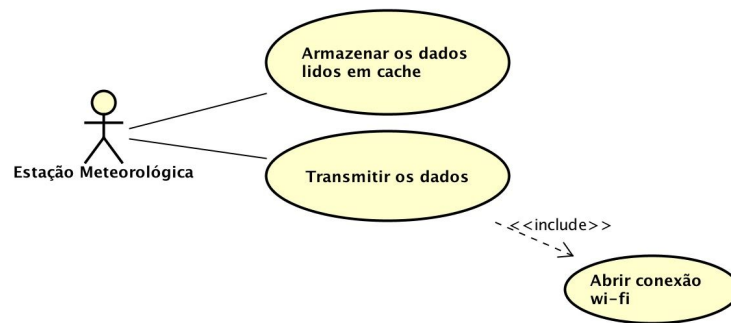


Figura 4 - Caso de uso: transmitir dados dos sensores.

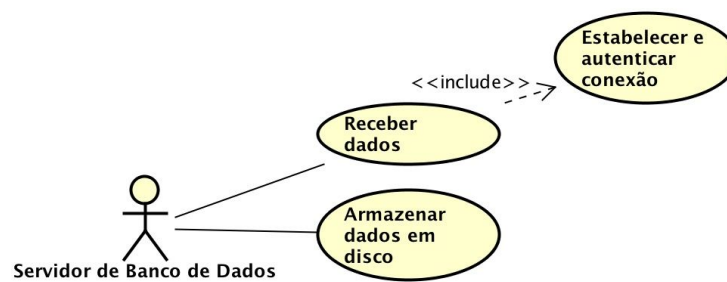


Figura 5 - Caso de uso: armazenar dados.

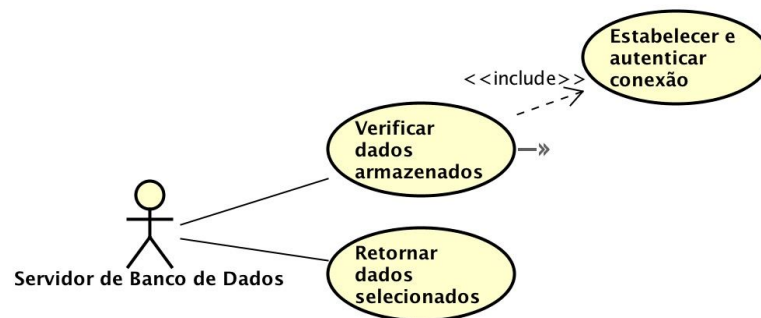


Figura 6 - Caso de uso: recuperar dados.

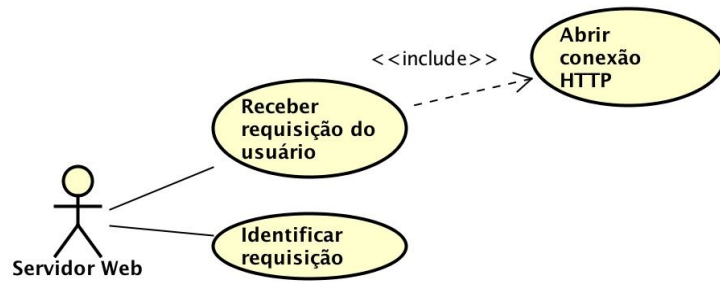


Figura 7 - Caso de uso: receber requisições.

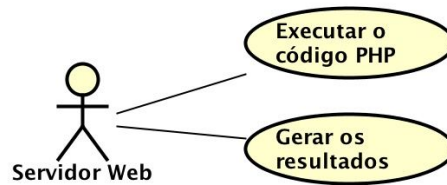


Figura 8 - Caso de uso: processar requisições.

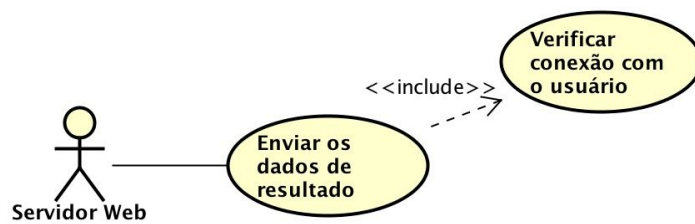


Figura 9 - Caso de uso: responder requisições.

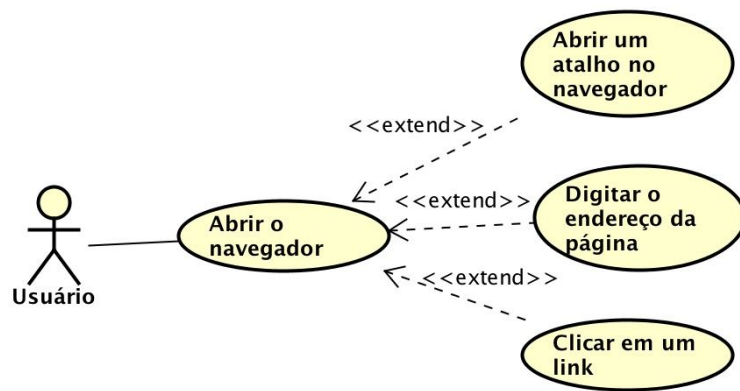


Figura 10 - Caso de uso: requisitar uma página web.

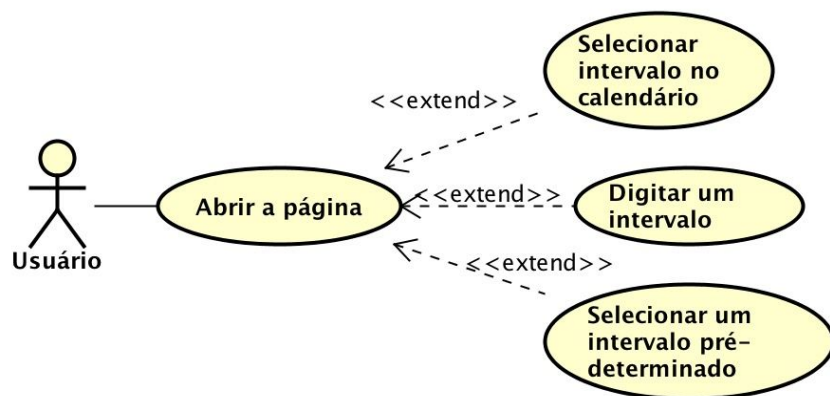


Figura 11 - Caso de uso: selecionar um intervalo do histórico.

5 Análise do Sistema

5.1 Arquitetura do Sistema

O diagrama da Figura 12 representa a arquitetura do sistema de estação meteorológica.



Figura 12 - Arquitetura do sistema.

No diagrama da Figura 12, o Arduino representa a estação meteorológica, que realiza a medição da temperatura e umidade do ambiente em que ele se situa. O computador funciona como um servidor web e de banco de dados. E, para realizar a ponte de conexão entre estes dois componentes acima citados, é utilizado um roteador wireless.

5.2 Modelo do Banco de Dados

5.2.1 Modelo Conceitual

A Figura 13 seguinte demonstra como foi estruturado o modelo conceitual do banco de dados do sistema.

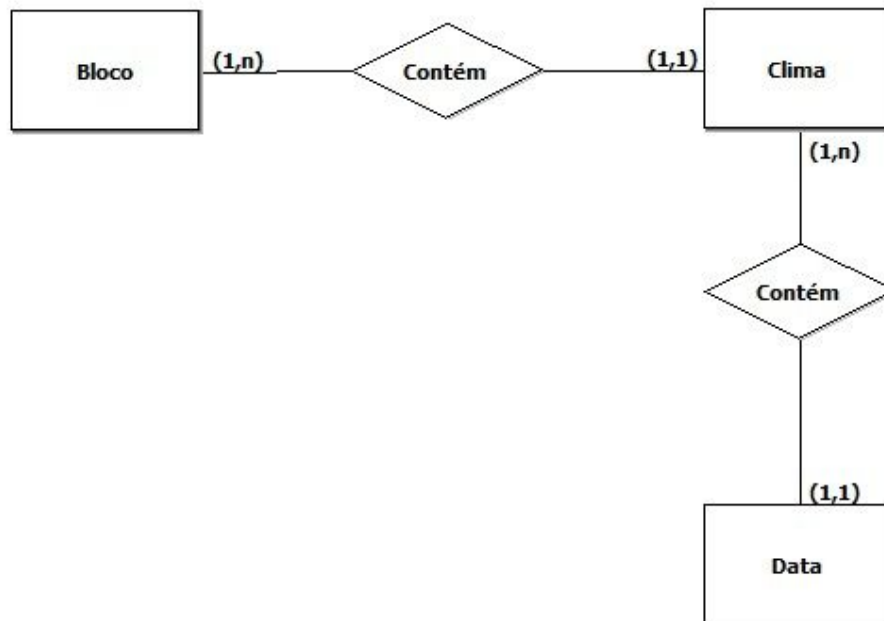


Figura 13 - Modelo conceitual do banco de dados.

5.2.2 Modelo Lógico

A Figura 14 seguinte demonstra como foi estruturado o modelo lógico do banco de dados do sistema.

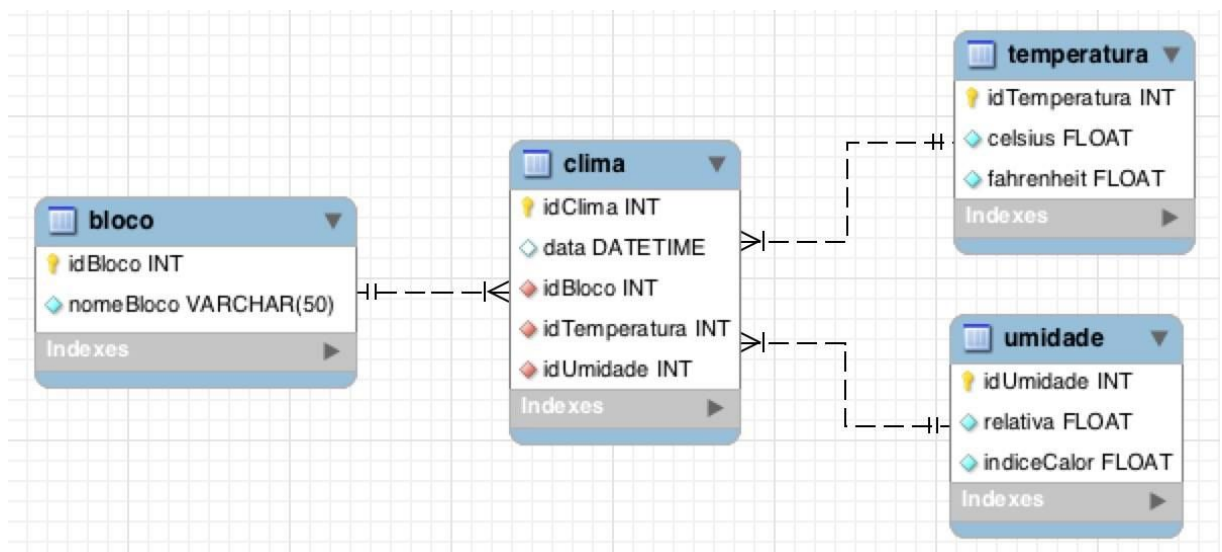


Figura 14 - Modelo lógico do banco de dados.

5.2.3 Dicionário de dados

Nas tabelas 4, 5, 6 e 7 seguintes, os itens do modelo lógico acima são

explicados na forma de um dicionário de dados.

BLOCO				
ATRIBUTO	TIPO	DESCRIÇÃO	NULL	CHAVE
idBloco	Int	Identificador de registro em Bloco	NOT NULL	Primária
nomeBloco	Varchar	Nome identificador do referido Bloco	NOT NULL	-----

Tabela 4 - Dicionário de dados: Bloco.

CLIMA				
ATRIBUTO	TIPO	DESCRIÇÃO	NULL	CHAVE
idClima	Int	Atributo identificador em Clima	NOT NULL	Primária
data	DateTime	Atributo identificador de tempo	NOT NULL	-----
idBloco	Int	Chave estrangeira da tabela Bloco	NOT NULL	Estrangeira
idTemperatura	Int	Chave estrangeira da Tabela Temperatura	NOT NULL	Estrangeira
idUmidade	Int	Chave estrangeira da Tabela Umidade	NOT NULL	Estrangeira

Tabela 5 - Dicionário de dados: Clima.

TEMPERATURA				
ATRIBUTO	TIPO	DESCRIÇÃO	NULL	CHAVE
idTemperatura	Int	Atributo identificador de Temperatura	NOT NULL	Primária
celsius	Float	Atributo identificador da escala do tipo celsius	NOT NULL	-----
fahrenheit	Float	Atributo identificador da escala do tipo fahrenheit	NOT NULL	-----

Tabela 6 - Dicionário de dados: Temperatura.

TEMPERATURA				
ATRIBUTO	TIPO	DESCRIÇÃO	NULL	CHAVE
idUmidade	Int	Atributo identificador de Umidade	NOT NULL	Primária
relativa	Float	Quantidade de umidade presente no ar	NOT NULL	-----
indiceCalor	Float	Quantidade estimada de calor com base na umidade	NOT NULL	-----

Tabela 6 - Dicionário de dados: Umidade.

5.3 Diagrama de Classes

Nas Figuras 15 e 16 seguintes constam os diagramas de classes do projeto.

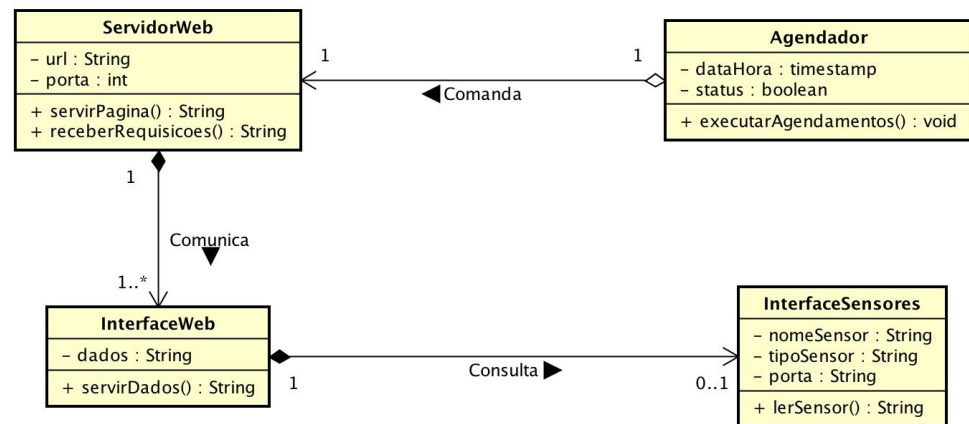


Figura 15 - Diagrama de classes: Servidor Web.

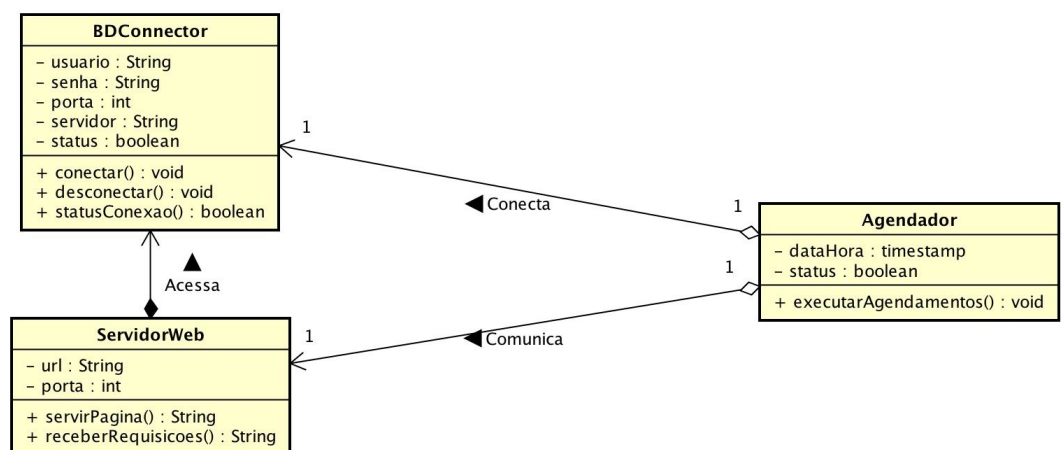


Figura 16 - Diagrama de classes: Estação meteorológica.

5.4 Diagrama de Atividades

Nas Figuras 17, 18, 19 e 20 seguintes constam os diagramas de atividades do projeto.

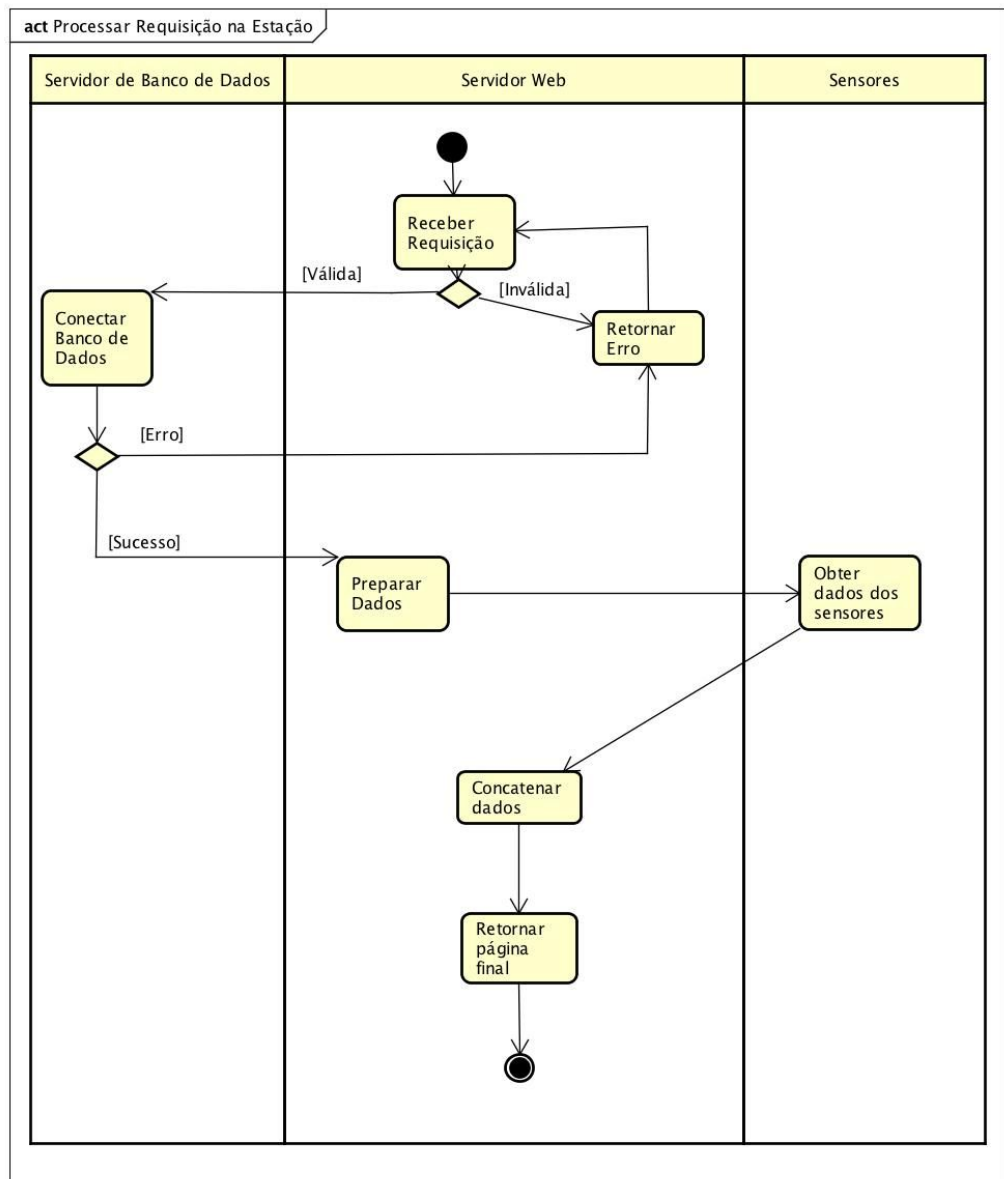


Figura 17 - Diagrama de atividade: Processar requisição na estação.

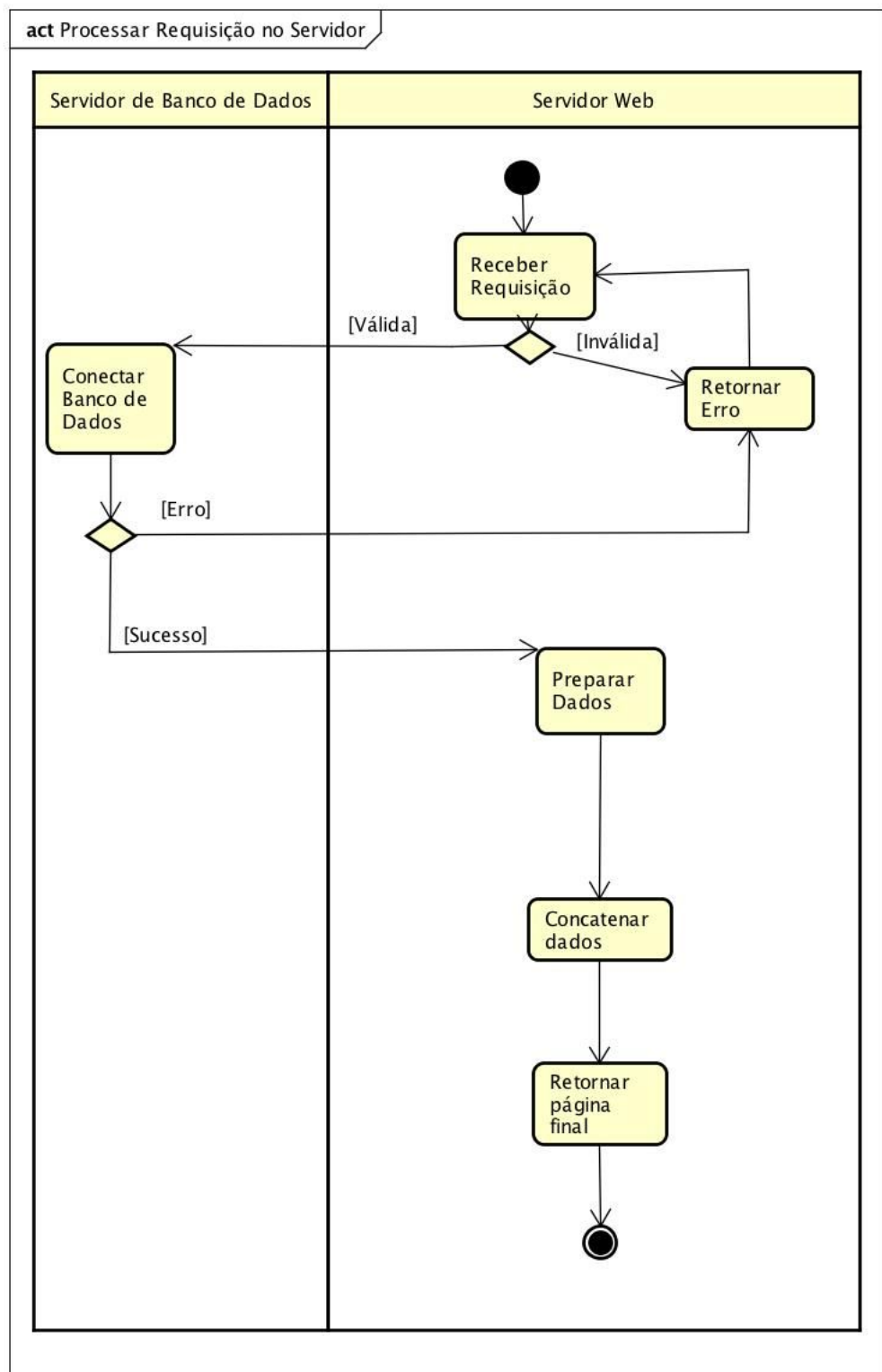


Figura 18 - Diagrama de atividade: Processar requisição no servidor.

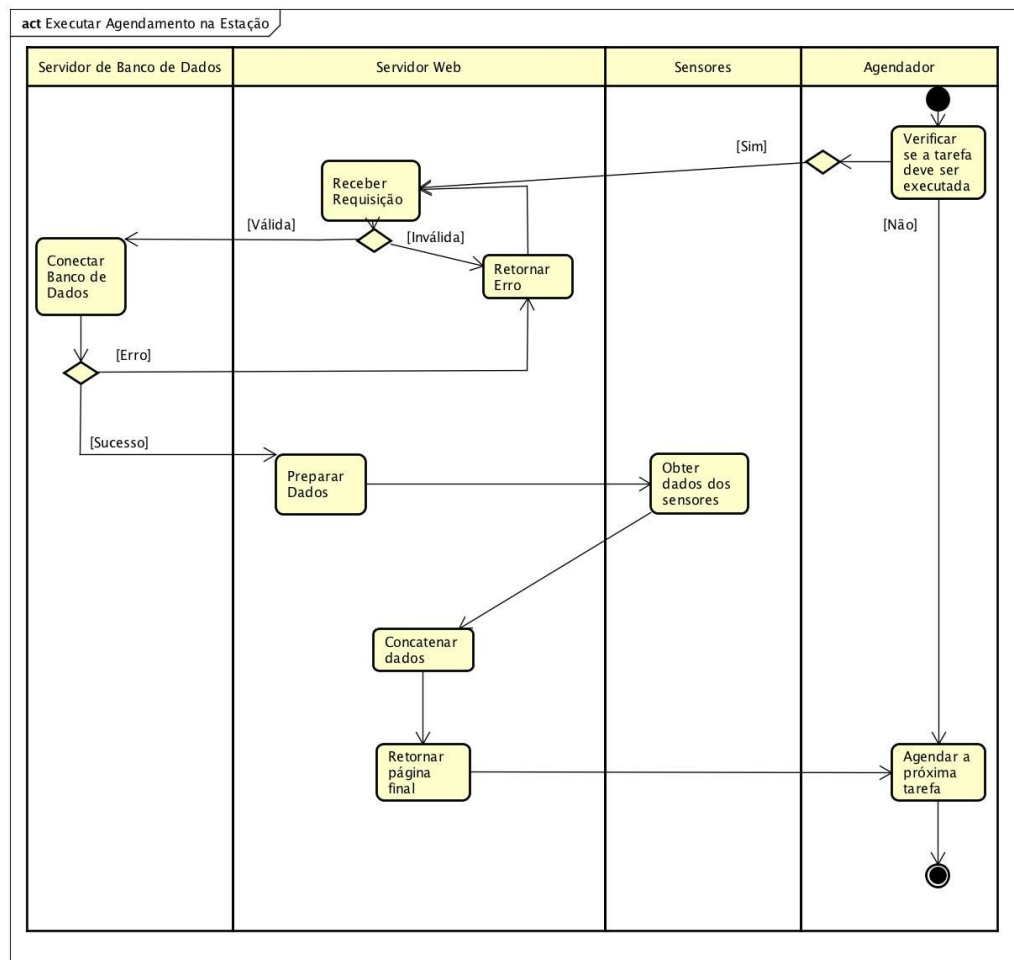


Figura 19 - Diagrama de atividade: Executar agendamento na estação.

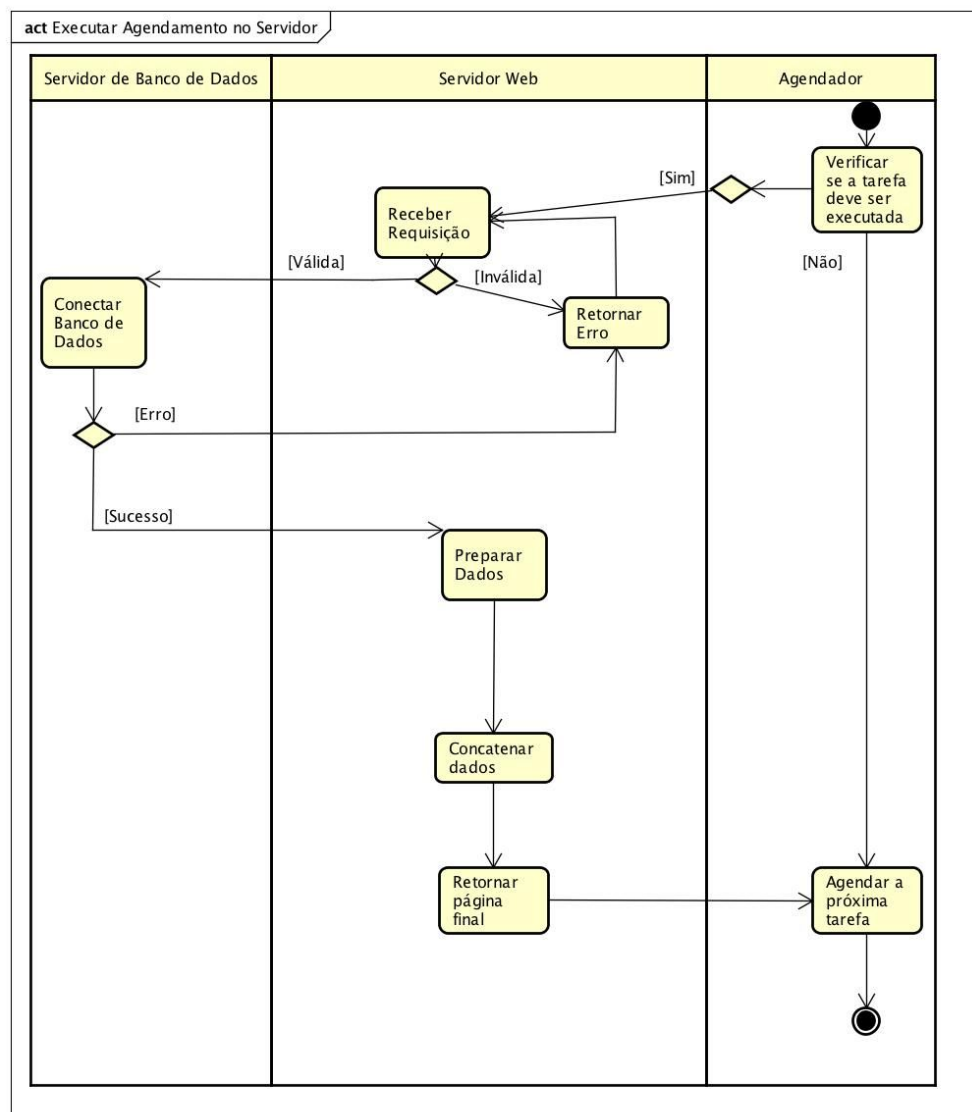


Figura 20 - Diagrama de atividade: Executar agendamento no servidor.

6 Implementação

6.1 Protótipos de Telas

Para elaboração da página web, foi utilizado o protótipo de tela seguinte aqui apresentado.

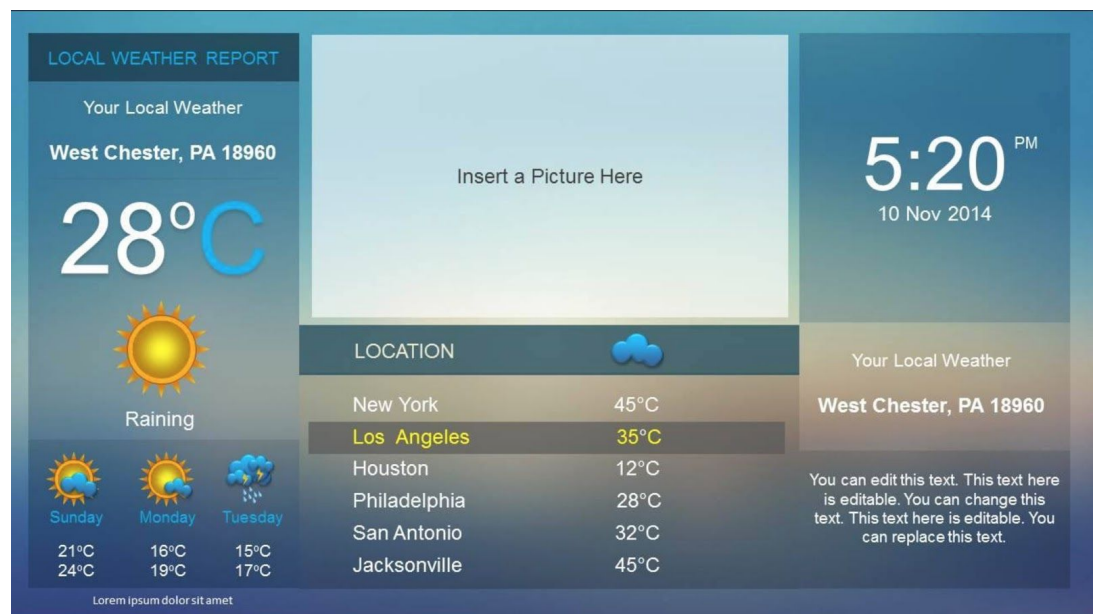
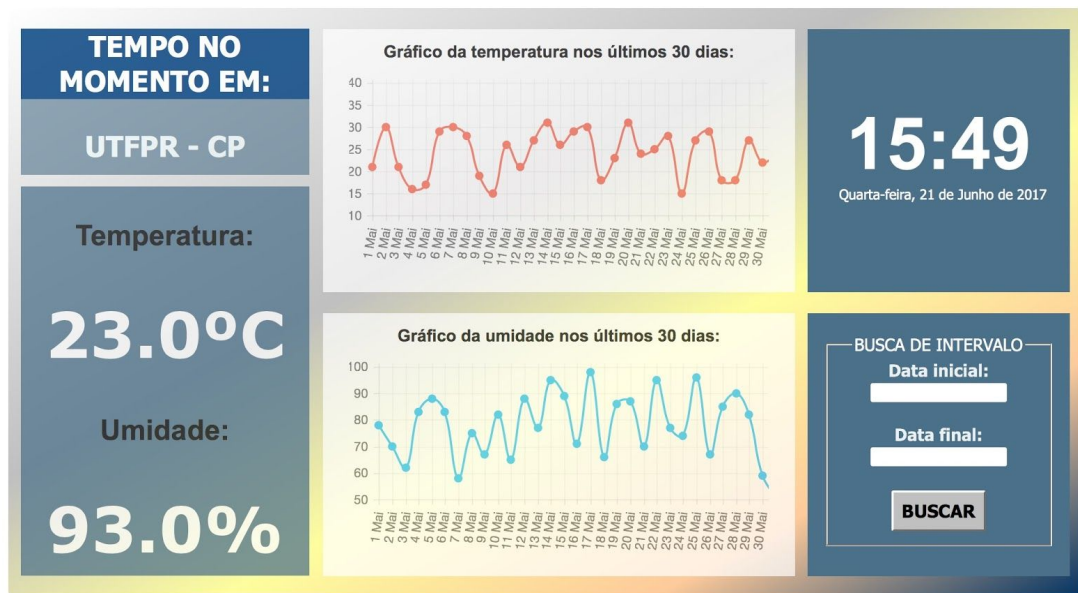


Figura 21 - Protótipo de tela.

O resultado é o apresentado abaixo.



© NO CÉU TEMPÃO
Trabalho ministrado na disciplina de Oficina de Integração 2

Figura 22 - Protótipo da página web.

O protótipo em Arduino não possui interface gráfica. Portanto, o seu esquema final de ligação está apresentado a seguir.

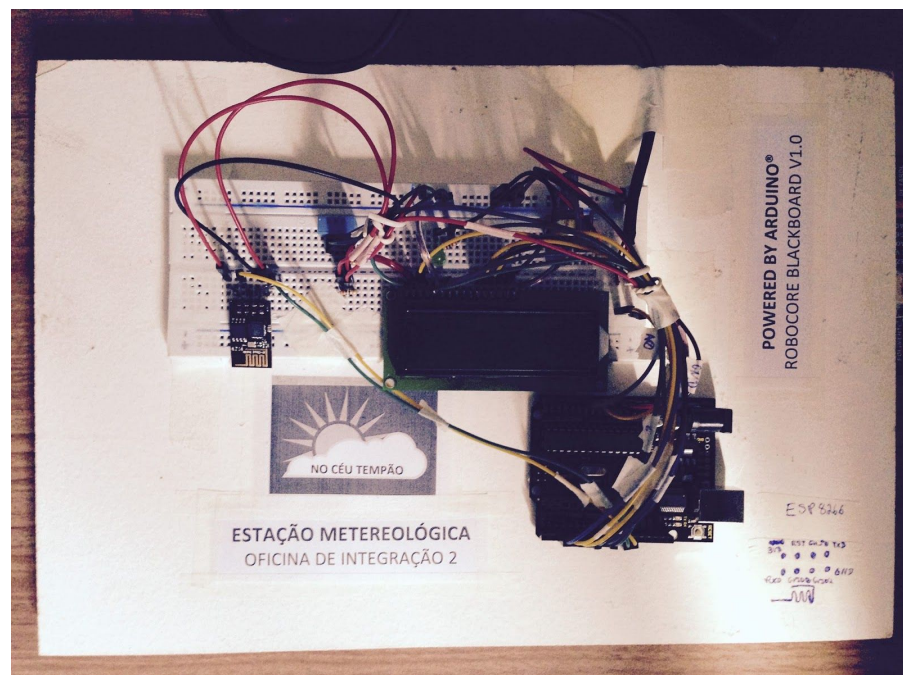
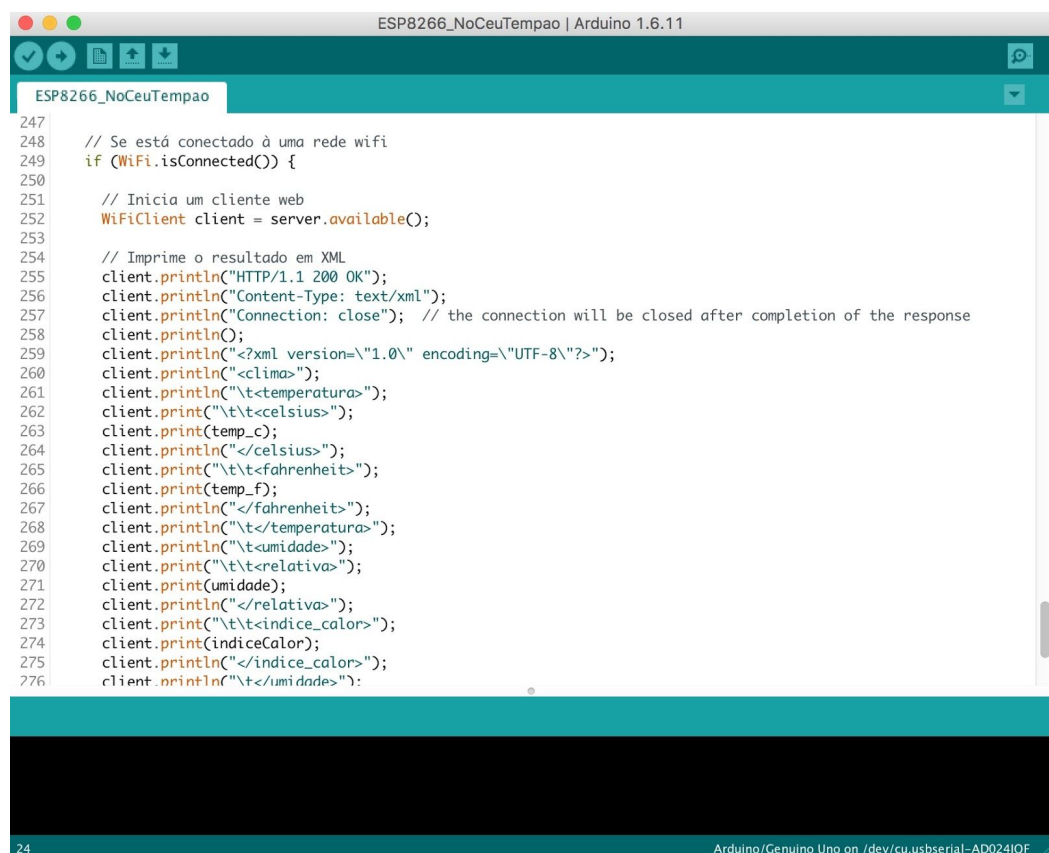


Figura 23 - Protótipo em Arduino.

6.2 Descrição do código

Na estação meteorológica, usando Arduino, há duas programações: uma do microcontrolador ESP8266, que é uma placa usada para conectar na rede WiFi, e ela própria é um Arduino separado. Este código é estruturado, de arquivo único (arquivo .ino), responsável pela recepção dos dados de temperatura e umidade e geração de um arquivo XML, além de conectar automaticamente a uma rede wifi e retornar com o seu endereço IP. Uma porção do seu código está exibido abaixo.



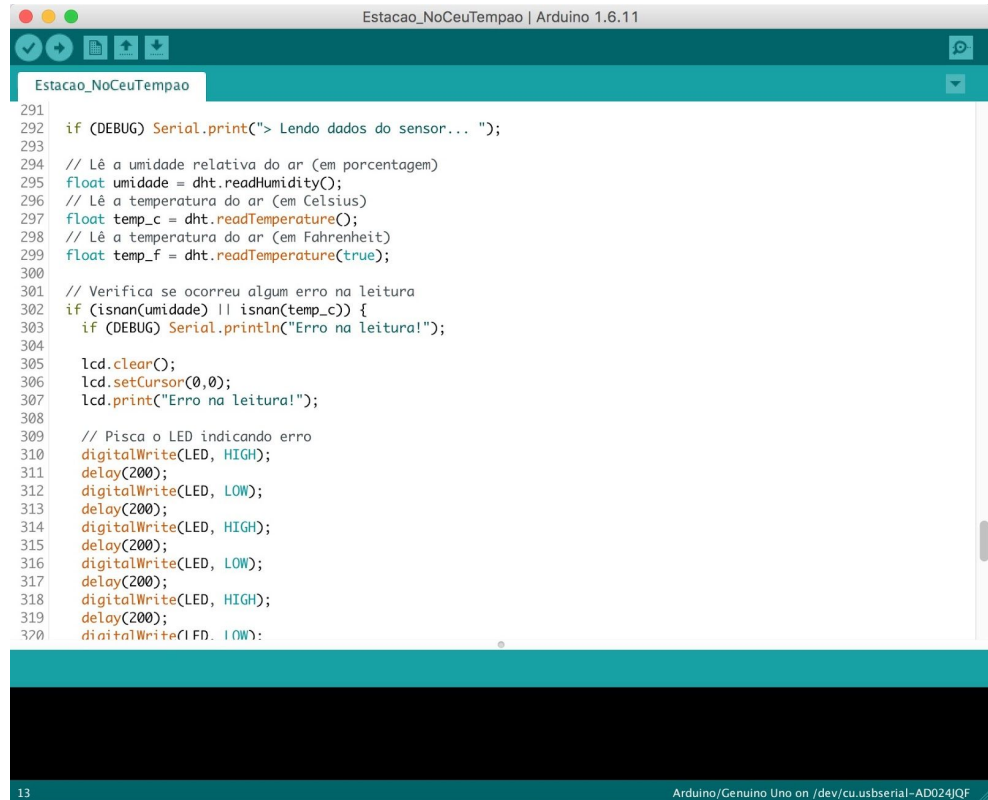
```

247
248 // Se está conectado à uma rede wifi
249 if (WiFi.isConnected()) {
250
251     // Inicia um cliente web
252     WiFiClient client = server.available();
253
254     // Imprime o resultado em XML
255     client.println("HTTP/1.1 200 OK");
256     client.println("Content-Type: text/xml");
257     client.println("Connection: close"); // the connection will be closed after completion of the response
258     client.println();
259     client.println("<?xml version='1.0' encoding='UTF-8'?'>");
260     client.println("<clima>");
261     client.println("<t<temperatura>");
262     client.print("<t<celsius>");
263     client.print(temp_c);
264     client.println("</celsius>");
265     client.print("<t<fahrenheit>");
266     client.print(temp_f);
267     client.println("</fahrenheit>");
268     client.println("</temperatura>");
269     client.println("<t<umidade>");
270     client.print("<t<relativa>");
271     client.print(umidade);
272     client.println("</relativa>");
273     client.print("<t<indice_calor>");
274     client.print(indiceCalor);
275     client.println("</indice_calor>");
276     client.println("<t</umidade>");
  
```

Figura 22 - Fragmento de código do ESP8266.

Na parte geral da estação, há o controle do sensor de temperatura e umidade, controle do *display* LCD e alimentação do sistema. Neste microcontrolador, é feita a leitura do sensor, os devidos cálculos, comunicação com o ESP8266 e exibição dos dados referentes à rede conectada, endereço IP e dados lidos do sensor no *display* LCD. Uma parte do seu código está

impressa abaixo.



```

291
292   if (DEBUG) Serial.print("> Lendo dados do sensor... ");
293
294   // Lê a umidade relativa do ar (em porcentagem)
295   float umidade = dht.readHumidity();
296   // Lê a temperatura do ar (em Celsius)
297   float temp_c = dht.readTemperature();
298   // Lê a temperatura do ar (em Fahrenheit)
299   float temp_f = dht.readTemperature(true);
300
301   // Verifica se ocorreu algum erro na leitura
302   if (isnan(umidade) || isnan(temp_c)) {
303     if (DEBUG) Serial.println("Erro na leitura!");
304
305     lcd.clear();
306     lcd.setCursor(0,0);
307     lcd.print("Erro na leitura!");
308
309     // Pisca o LED indicando erro
310     digitalWrite(LED, HIGH);
311     delay(200);
312     digitalWrite(LED, LOW);
313     delay(200);
314     digitalWrite(LED, HIGH);
315     delay(200);
316     digitalWrite(LED, LOW);
317     delay(200);
318     digitalWrite(LED, HIGH);
319     delay(200);
320     digitalWrite(LED, LOW);
  
```

Figura 23 - Fragmento de código do Arduino Blackboard.

Para que houvesse a comunicação entre os dois microcontroladores, um protocolo de comunicação via serial foi criado, com mensagens personalizadas e comandos feitos única e exclusivamente para a comunicação neste projeto, enviando as informações lidas do sensor pela rede de internet.

A página web é simples e exibe apenas os dados que foram capturados pela estação e salvos no banco de dados da aplicação no servidor web. Há uma biblioteca de funções que auxilia na conexão com a estação meteorológica e manipulação dos dados no banco de dados. Uma parte de seu código pode ser vista no fragmento abaixo.

```

99         </div>
100     <br>
101     <div class="linhaForm">
102         <label class="rotulo" for="dataf">Data final: </label>
103         <input class="campo" type="date" name="dataFim" class="InputFields"></input>
104     </div>
105     <br>
106     <button type="submit" id="BotaoSearch"> BUSCAR </button>
107 </fieldset>
108 </form>
109 </div>
110
111 </div>
112
113 <script>
114     var _ultimos30diasT = {
115         labels : ["13 Mai", "14 Mai", "15 Mai", "16 Mai", "17 Mai", "18 Mai", "19 Mai", "20 Mai", "21 Mai", "22 Mai", "23 Mai", "24 Mai", "25 Mai", "26 Mai",
116         "27 Mai", "28 Mai", "29 Mai", "30 Mai", "31 Mai", "1 Jun", "2 Jun", "3 Jun", "4 Jun", "5 Jun", "6 Jun", "7 Jun", "8 Jun", "9 Jun", "10 Jun", "11 Jun", "12 Jun",],
117         datasets :
118         [
119             {
120                 label: "TEMPERATURA",
121                 fillColor : "rgba(255,99,71,0.01)",
122                 strokeColor : "rgba(255,99,71,1)",
123                 pointColor : "rgba(255,99,71,1)",
124                 pointStrokeColor : "rgba(255,255,255,0)",
125                 pointHighlightFill : "#fff",
126                 pointHighlightStroke : "rgba(255,99,71,1)"
127             }
128         ]
129     };
130 </script>
131 </body>

```

Figura 24 - Fragmento de código da página web.

```

1 // Converte o objeto XML para Array
2 $res = json_decode(json_encode($xml), true);
3
4 // Retorna com o array de dados
5 return $res;
6
7 // Se é para puxar os dados do banco
8 } else {
9     // Puxa o último registro do banco de dados
10     $r = mysql_fetch_array(query("SELECT * FROM (SELECT b.nomeBloco, c.data, t.celsius, t.fahrenheit, u.relativa, u.indiceCalor FROM bloco b INNER JOIN clima
11 c INNER JOIN temperatura t INNER JOIN umidade u GROUP BY c.data) s ORDER BY s.data DESC LIMIT 1"));
12
13     // Converte os dados do banco para o formato esperado pela aplicação
14     $res = array();
15     $res['temperatura']['celsius'] = $r['celsius'];
16     $res['temperatura']['fahrenheit'] = $r['fahrenheit'];
17     $res['umidade']['relativa'] = $r['relativa'];
18     $res['umidade']['indiceCalor'] = $r['indiceCalor'];
19
20     // Retorna com o array de dados
21     return $res;
22 }
23 }

```

Figura 25 - Fragmento de código da biblioteca de funções.

7 Considerações Finais

Durante o desenvolvimento deste sistema, o grupo de trabalho teve contato com diversos conceitos da área de programação que não possuíam conhecimento, o que demandou horas extras de trabalho. Um exemplo disso é o fato de somente um dos integrantes do grupo ter conhecimento prévio sobre programação de arduino.

Na parte de desenvolvimento WEB, o grupo não teve muita dificuldade, pois a interface gerada é muito simples. A parte mais complicada desse quesito foi o uso da tag <canvas> para gerar o gráfico de umidade e temperatura. Outro ponto importante é que, como não houve tempo hábil e disponibilidade para manter o servidor conectado 24h por dia durante todo o mês à estação, o recurso de busca por data não foi implementado. Com a falta dos dados medidos, os dados expostos no gráfico também foram simulados, para facilitar a visualização e permitir uma visão de como ficará o sistema depois de executado no mundo real.

Além destes aspectos, na realização deste projeto, o grupo teve conhecimento, através de pesquisas exaustivas, sobre conceitos inerentes a temperatura, umidade e climatologia.

8 Bibliografia

- ASCOM. **A importância da meteorologia vai muito além de saber “se vai chover hoje”**. Disponível em:
<<http://www.crea-se.org.br/a-importancia-da-meteorologia-vai-muito-alem-de-saber-se-vai-chover-hoje/>>. Acesso em 08 de Abril de 2017.
- FARIA, Caroline. **Previsão do Tempo**. Disponível em:
<<http://www.infoescola.com/meteorologia/previsao-do-tempo/>>. Acesso em 08 de Abril de 2017.
- FABRI, José Augusto; GONÇALVES, José Antonio; L'ERARIO, Alexandre; PAGOTTO, Tiago. **Scrum Solo: Processo de software para desenvolvimento individual**. Disponível em:
<<https://engenhariasoftware.files.wordpress.com/2016/04/scrum-solo.pdf>>. Acesso em 08 de Abril de 2017.