

# **APLICAÇÃO PARA CONTROLE DE FLUXO E MENSAGERIA ENTRE DISPOSITIVOS IOT**

Aluno: Silvio Greuel

Orientador: Dalton Solano dos Reis

# Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Especificação
- Implementação
- Operacionalidade
- Resultados e discussões
- Conclusões e sugestões

# Introdução

- Controle centralizado da mensageria
- Gestão dos fluxos de mensagens entre dispositivos IoT
- Multiplataforma com PWA
- Utilização de tecnologias de código aberto

# Objetivos

- Realizar a gestão dos fluxos de mensageria de dispositivos IoT de modo centralizado sem a necessidade de efetuar alterações no firmware
- Explorar integração de instalação nativa no sistema operacional, validando o acesso off-line a aplicação
- Explorar o uso da aplicação utilizando uma bancada de homologação, descrevendo de modo que possa ser futuramente reproduzido
- Levantar casos de uso no ambiente agrário e apresentar uma solução aplicável utilizando o aplicativo produzido por este trabalho
- Levantar casos de uso no ambiente industrial e apresentar uma solução aplicável utilizando o aplicativo produzido por este trabalho

# Fundamentação Teórica

## Internet das Coisas (IoT)

- Dados sobre o ambiente físico
- Ações sobre o ambiente físico
- Informação
- Expansão da internet
- Conectividade

# Fundamentação Teórica

## Progressive Web App (PWA)

- Tecnologias web
- Progressiva
- Responsiva
- Independência da rede
- Detectável
- Instalável
- Compartilhável

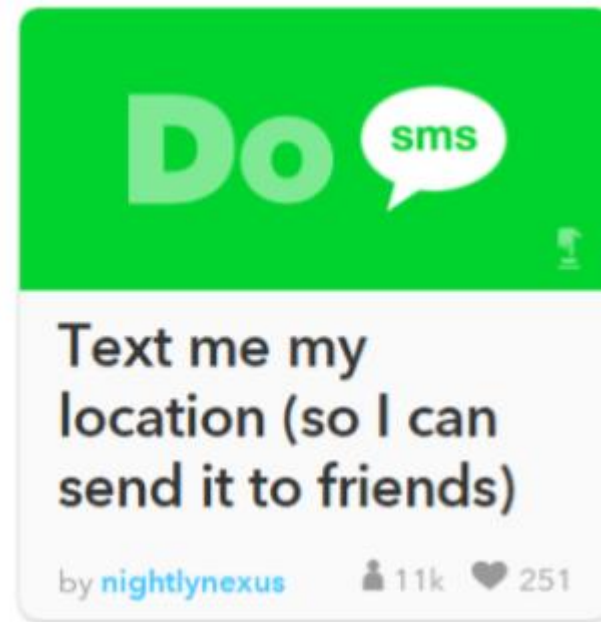
# Trabalhos Correlatos

Características	Correlatos	Vorapojsut (2015)	ADLINK (2017)	Fransson, Driaguine (2017)
Aborda aplicação WEB		SIM	NÃO	SIM
Aborda interface móvel		SIM	NÃO	SIM
Aborda multiplataforma		NÃO	NÃO	SIM
Aborda definição do controle de fluxos		SIM	NÃO	NÃO
Aborda definição do controle de dispositivos		SIM	NÃO	NÃO
Aborda utilização do Protocolo REST		NÃO	SIM	NÃO
Aborda utilização do Protocolo AMQP		NÃO	SIM	NÃO
Aborda utilização do Protocolo MQTT		NÃO	SIM	NÃO

# Trabalhos Correlatos

## Vorapojisut (2015)

- Modelo “Se Isso, Então Aquilo”

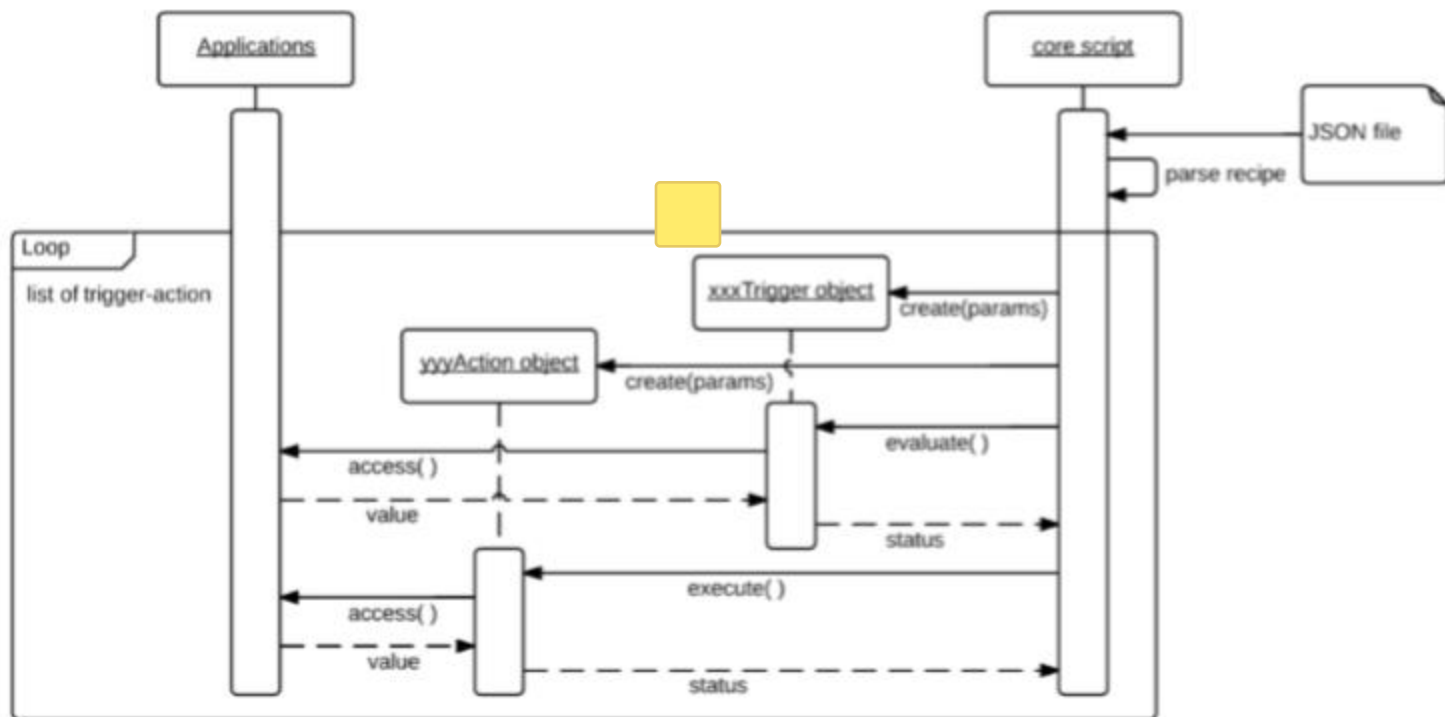




# Trabalhos Correlatos

## Vorapojpisut (2015)

- Gatilhos, Ações



# Trabalhos Correlatos

## ADLINK (2017).

- MQTT
- AMQP
- REST (HTTP)

# Trabalhos Correlatos

## ADLINK (2017).

Características	MQTT	AMQP	REST(HTTP)
Modelo de Abstração	PUB/SUB	P2P ou PUB/SUB	Request/Reply
Arquitetura	Centralizada	Centralizada	Cliente-Servidor
Interoperabilidade	Parcial	Sim	Sim
Roteamento de mensagens	Topicos	Exchanges, Filas, Bindings	N/A
Encoding	Binário	Binário	Texto Plano

# Trabalhos Correlatos

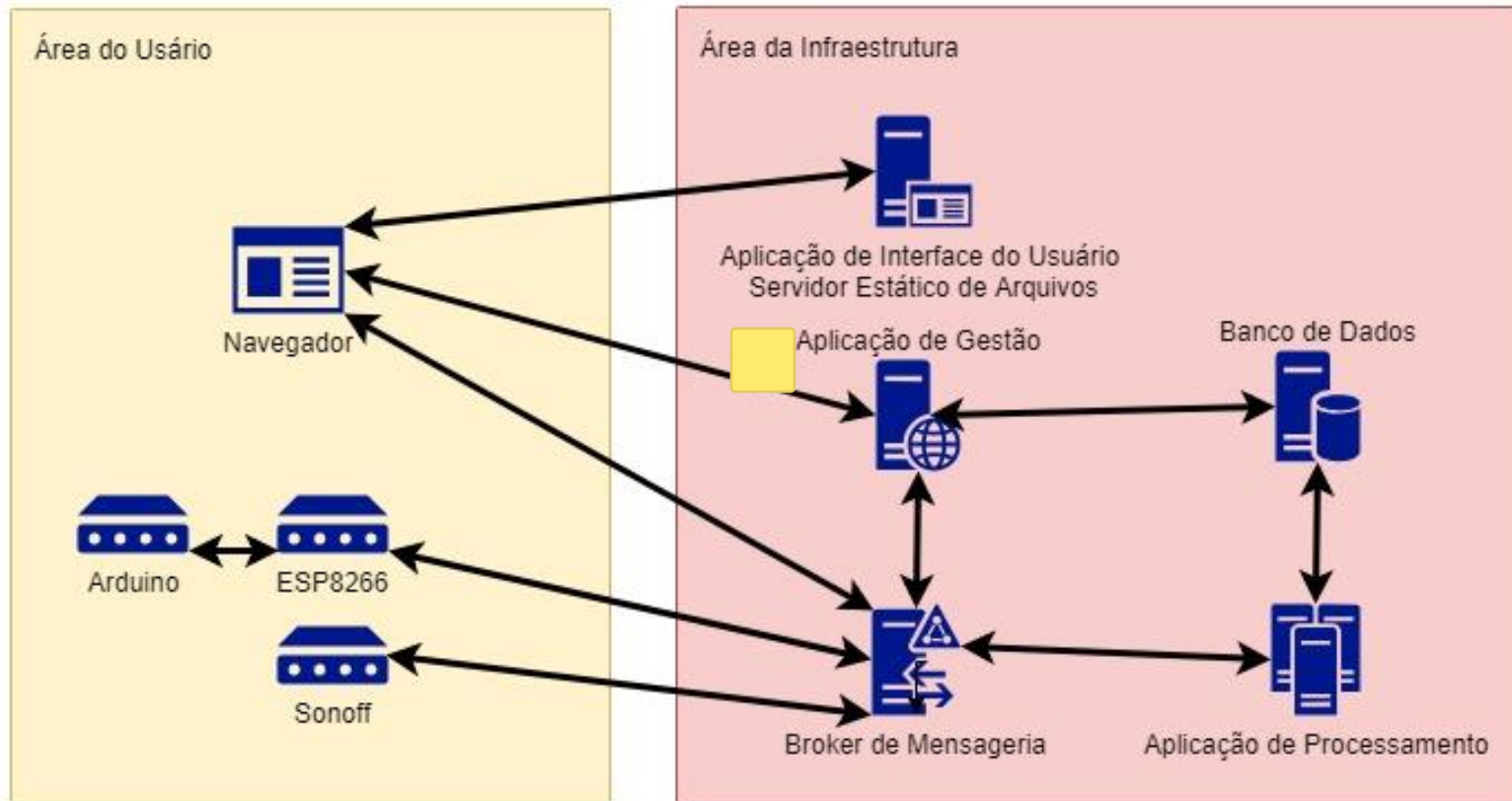
## Fransson e Driaguine (2017).

- Push messages
- Bluetooth
- Offline mode
- Background sync
- Vibration
- Fullscreen

# Especificação

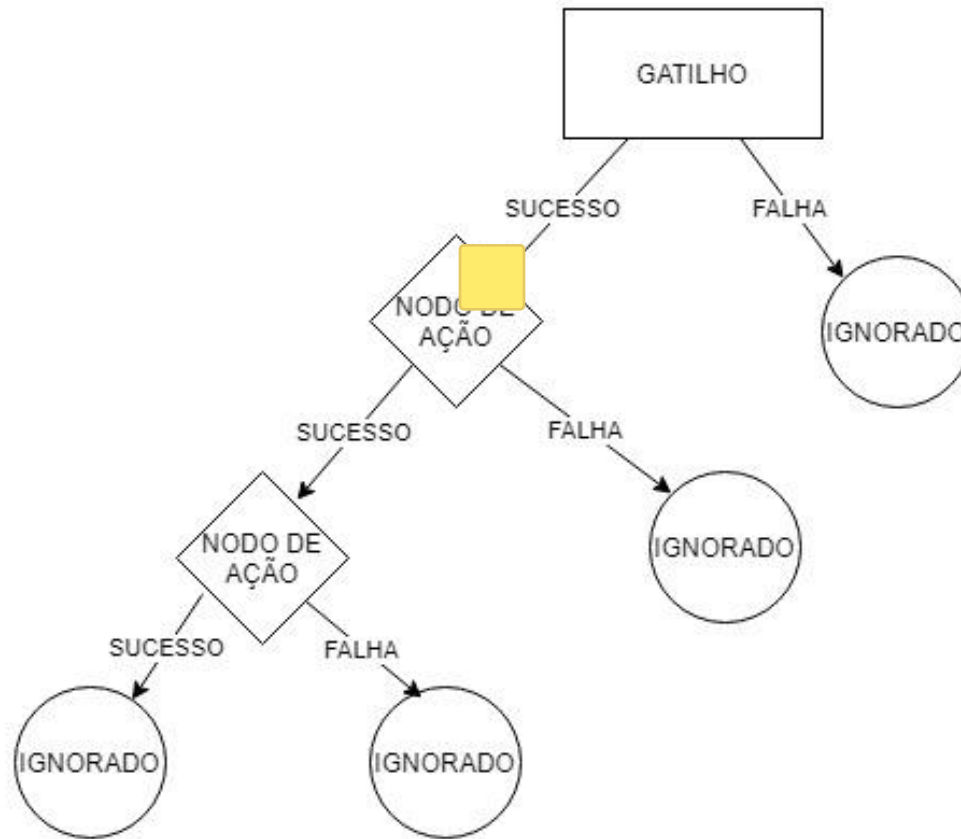
- (Apresentar os principais diagramas desenvolvidos na especificação e que permitem compreender os elementos essenciais do trabalho)
- (Cuidado com a legibilidade das figuras – redesenhe-as caso necessário)
- (Tempo estimado – 8 minutos)

# Implementação Arquitetura Geral

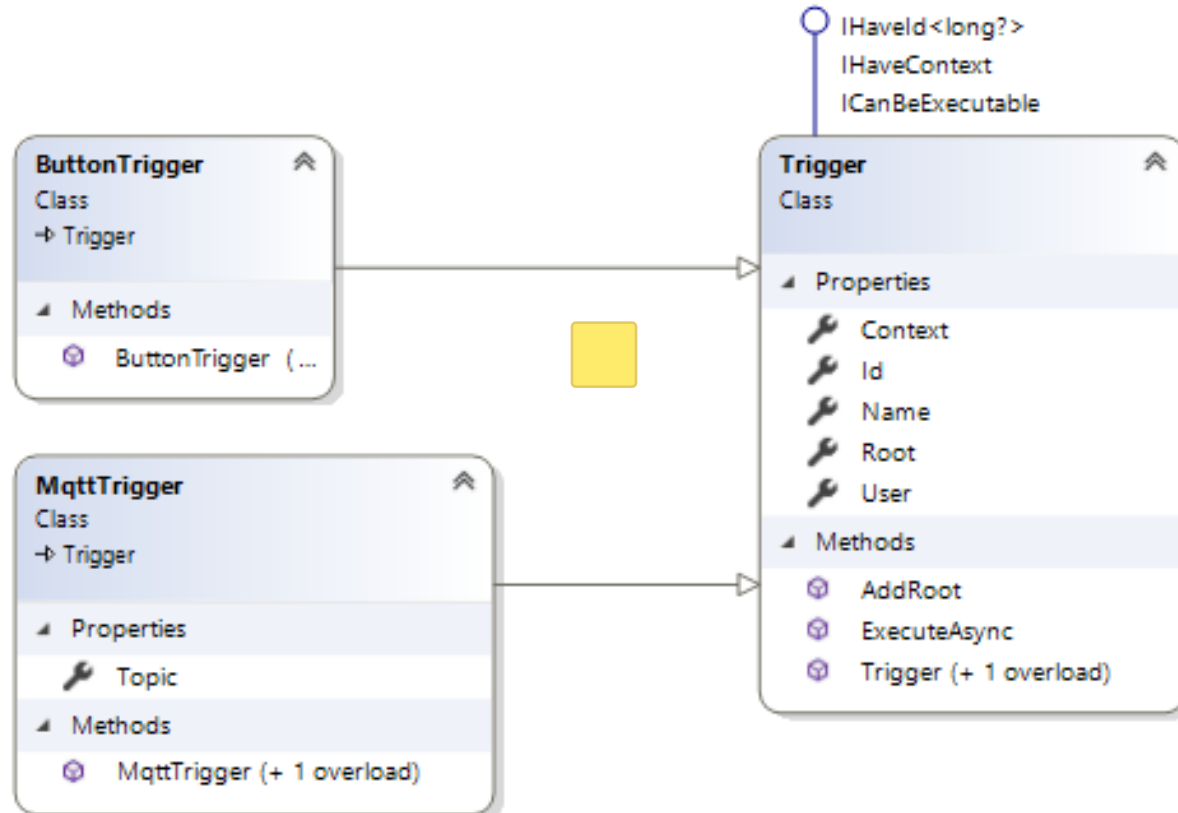


# Implementação

## Domínio Árvore de Decisão

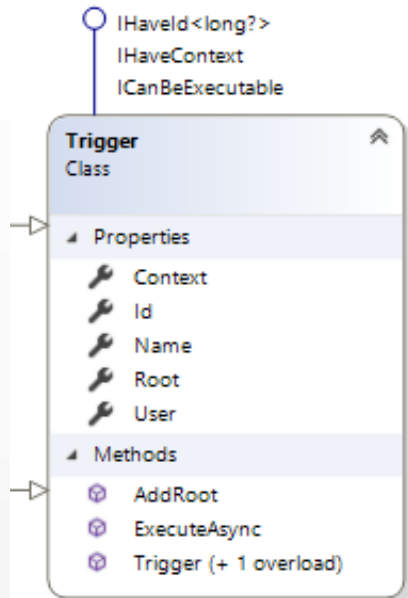


# Implementação Domínio Gatilho





# Implementação Domínio Gatilho



```
public class Trigger : IHaveId<long?>, IHaveContext, ICanBeExecutable
{
    public long? Id { get; set; }
    public string Name { get; set; }
    public virtual Node Root { get; set; }
    public virtual User User { get; set; }
    public virtual IDictionary<string, dynamic> Context { get; set; }

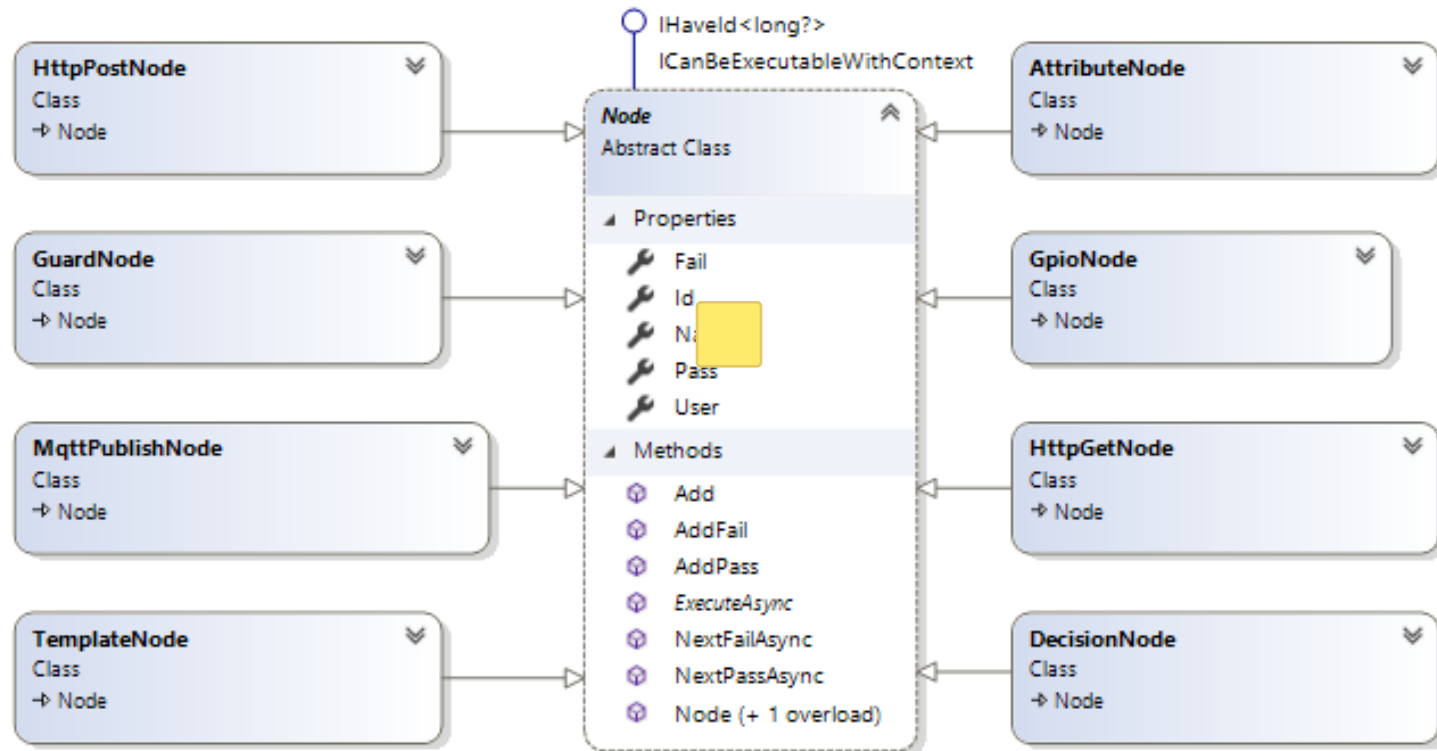
    public Trigger() { }

    public Trigger(User user, string name)
    {
        this.Name = name;
        this.User = user;
    }

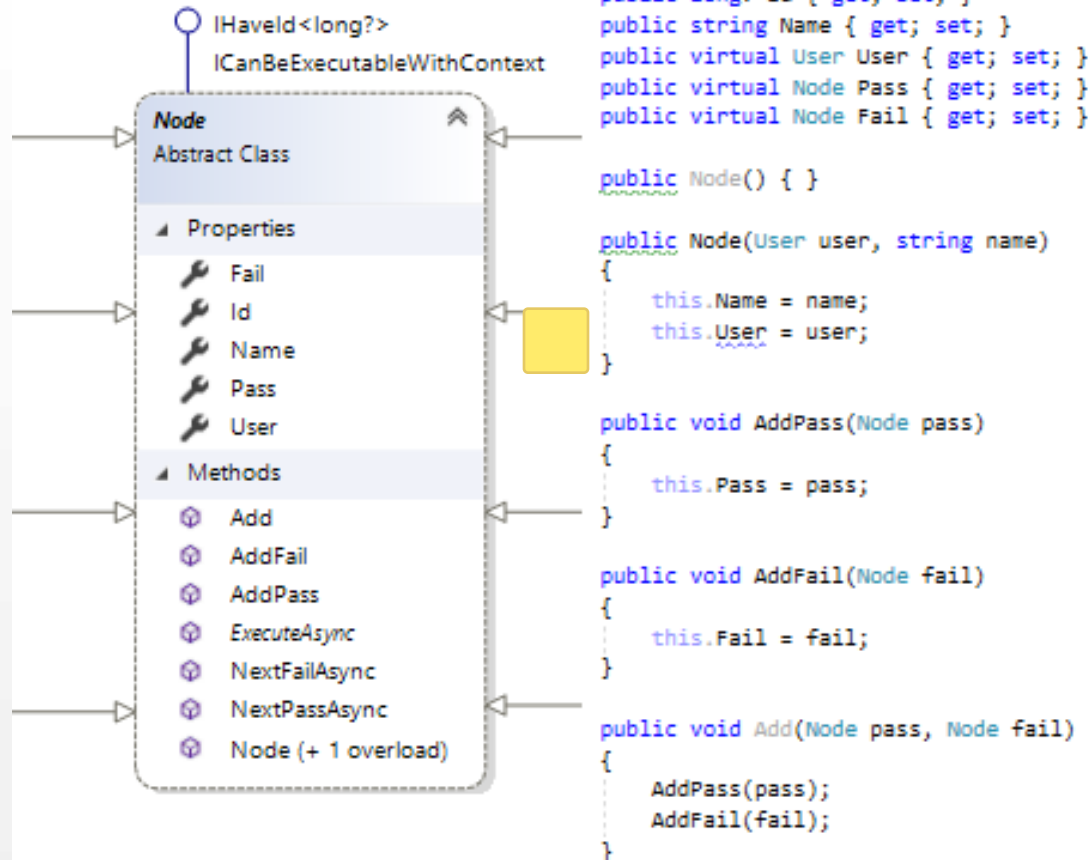
    public async Task ExecuteAsync()
    {
        if (Root == null) { return; }
        await Root.ExecuteAsync(Context);
    }

    public void AddRoot(Node root)
    {
        Root = root;
    }
}
```

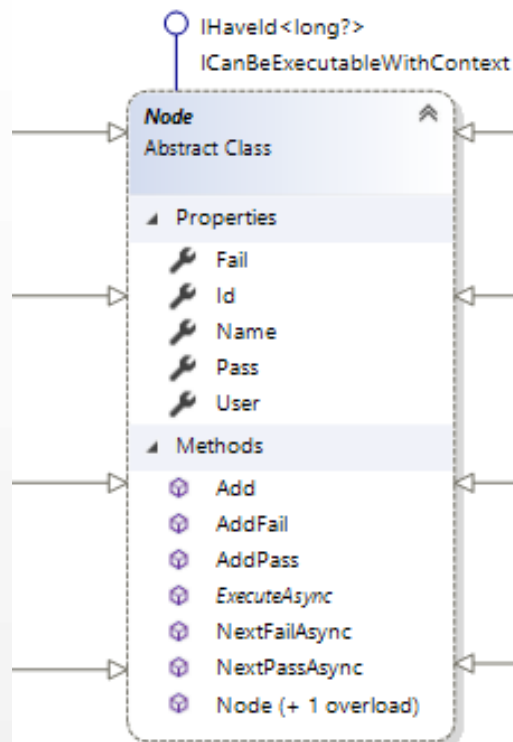
# Implementação Domínio Nodo



# Implementação Domínio Nodo



# Implementação Domínio Nodo



```
public async Task NextPassAsync(IDictionary<string, dynamic> context)
{
    try
    {
        if (Pass == null) return;
        await Pass.ExecuteAsync(context);
    }
    catch (Exception e)
    {
        Log.Logger.Information($"EXCEPTION {e.Message}");
        context["exception"] = e.Message;
    }
}
```

```
public async Task NextFailAsync(IDictionary<string, dynamic> context)
{
    try
    {
        if (Fail == null) return;
        await Fail.ExecuteAsync(context);
    }
    catch (Exception e)
    {
        Log.Logger.Information($"EXCEPTION {e.Message}");
        context["exception"] = e.Message;
    }
}
```

```
public abstract Task ExecuteAsync(IDictionary<string, dynamic> context);
```

# Implementação Domínio Nodo

```
public class HttpGetNode : Node
{
    public string Url { get; set; }
    public string Field { get; set; }

    public HttpGetNode() { }

    public HttpGetNode(User user, string name, string url, string field) : base(user, name)
    {
        this.Url = url;
        this.Field = field;
    }

    public override async Task ExecuteAsync(IDictionary<string, dynamic> context)
    {
        var url = string.Format(new TemplateFormatProvider(), Url, context);
        var http = new HttpClient();
        var request = new HttpRequestMessage(HttpMethod.Get, url);
        var response = await http.SendAsync(request);
        context[Field] = await response.Content.ReadAsStringAsync();

        Log.Logger.Information($"GET {url} = {Field}:{context[Field]}");
        await NextPassAsync(context);
    }
}
```

# Implementação PWA App Manifest

```
{  
  "name": "asdf-flow-pwa",  
  "short_name": "asdf-flow-pwa",  
  "icons": [  
    {  
      "src": "/img/icons/android-chrome-192x192.png",  
      "sizes": "192x192",  
      "type": "image/png"  
    },  
    {  
      "src": "/img/icons/android-chrome-512x512.png",  
      "sizes": "512x512",  
      "type": "image/png"  
    }  
  ],  
  "start_url": "/index.html",  
  "display": "standalone",  
  "background_color": "#000000",  
  "theme_color": "#4DBA87"  
}
```

# Implementação PWA Bluetooth

```
connectToDevice: function (device) {  
  return (device ? Promise.resolve(device) : this.requestBluetoothDevice())  
    .then((device) => this.connectDeviceAndCacheCharacteristic(device))  
    .then((characteristic) => this.startNotifications(characteristic))  
    .catch((error) => {  
      this.log(error);  
      return Promise.reject(error);  
    });  
},
```

# Implementação PWA Bluetooth

```
requestBluetoothDevice: function() {  
  this.log('Requesting bluetooth device...');  
  
  return navigator.bluetooth.requestDevice({  
    filters: [{services: [this.serviceUuid]}],  
  }).then((device) => {  
    this.log('"' + device.name + '" bluetooth device selected');  
  
    this.device = device;  
    this.device.addEventListener('gattserverdisconnected', this.handleDisconnection.bind(this));  
  
    return this.device;  
  });  
},
```



# Implementação PWA Bluetooth

```
connectDeviceAndCacheCharacteristic: function(device) {  
  if (device.gatt.connected && this.characteristic) {  
    return Promise.resolve(this.characteristic);  
  }  
  
  this.log('Connecting to GATT server...');  
  You, 16 days ago • Adding bluetooth  
  return device.gatt.connect().  
    then((server) => {  
      this.log('GATT server connected', 'Getting service...');  
      return server.getPrimaryService(this.serviceUuid);  
    }).  
    then((service) => {  
      this.log('Service found', 'Getting characteristic...');  
      return service.getCharacteristic(this.characteristicUuid);  
    }).  
    then((characteristic) => {  
      this.log('Characteristic found');  
  
      this.characteristic = characteristic; // Remember characteristic.  
  
      return this.characteristic;  
    });  
},
```

# Implementação PWA Bluetooth

```
startNotifications: function(characteristic) { You, 16 days ago • A  
  this.log('Starting notifications...');  
  
  return characteristic.startNotifications().  
    then(() => {  
      this.log('Notifications started');  
  
      characteristic.addEventListener('characteristicvaluechanged',  
        this.handleCharacteristicValueChanged.bind(this));  
    });  
},
```

# Implementação PWA Bluetooth

```
handleCharacteristicValueChanged: function(event) { You,
  let value = new TextDecoder().decode(event.target.value);

  for (let c of value) {
    if (c === this.receiveSeparator) {
      let data = this.receiveBuffer.trim();
      this.receiveBuffer = ' ';

      if (data) {
        this.receive(data);
      }
    } else {
      this.receiveBuffer += c;
    }
  }
},
```

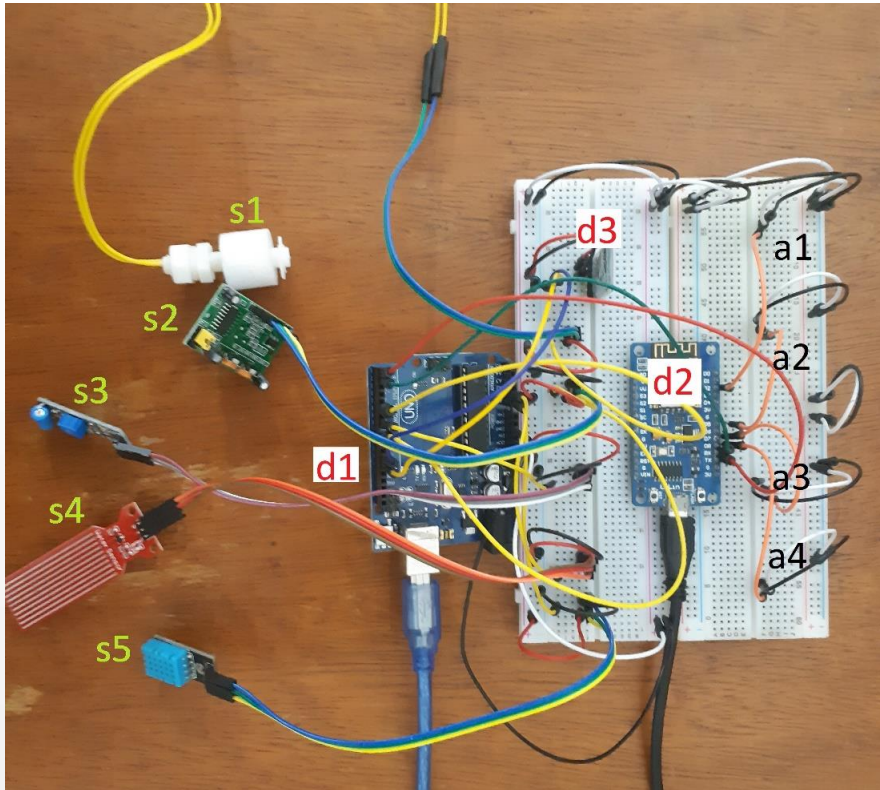
# Implementação PWA NFC

```
write: function(message) {
  alert('Writing')
  navigator
    .nfc
    .push(message)
    .then(() => {
      alert("Message pushed.");
    })
    .catch((error) => {
      alert("Push failed :-( again.");
    });
},
read: function() {
  alert('Reading')
  navigator
    .nfc
    .watch((message) => {
      alert('Message received')
      this.processMessage(message)
    })
    .then(() => alert("Added a watch."))
    .catch(err => alert("Adding watch failed: " + err.name))
},
```

# Implementação PWA Notificação

```
notify: function(message) {  
  Notification.requestPermission(permission => {  
    this.permission = permission  
    if (this.permission !== 'granted') {return;}  
  
    this.messages.push(message)  
  
    const options = {  
      body: message,  
      icon: 'img/icon/icon-file-150x150.png',  
      vibrate: [100, 50, 100],  
      data: {  
        dateOfArrival: Date.now(),  
        primaryKey: 1  
      },  
    },  
  )  
  navigator  
    .serviceWorker  
    .getRegistration()  
    .then(reg => {  
      reg.showNotification(message, options)  
    })  
})  
}
```

# Implementação IoT Bancada



# Implementação IoT Bancada

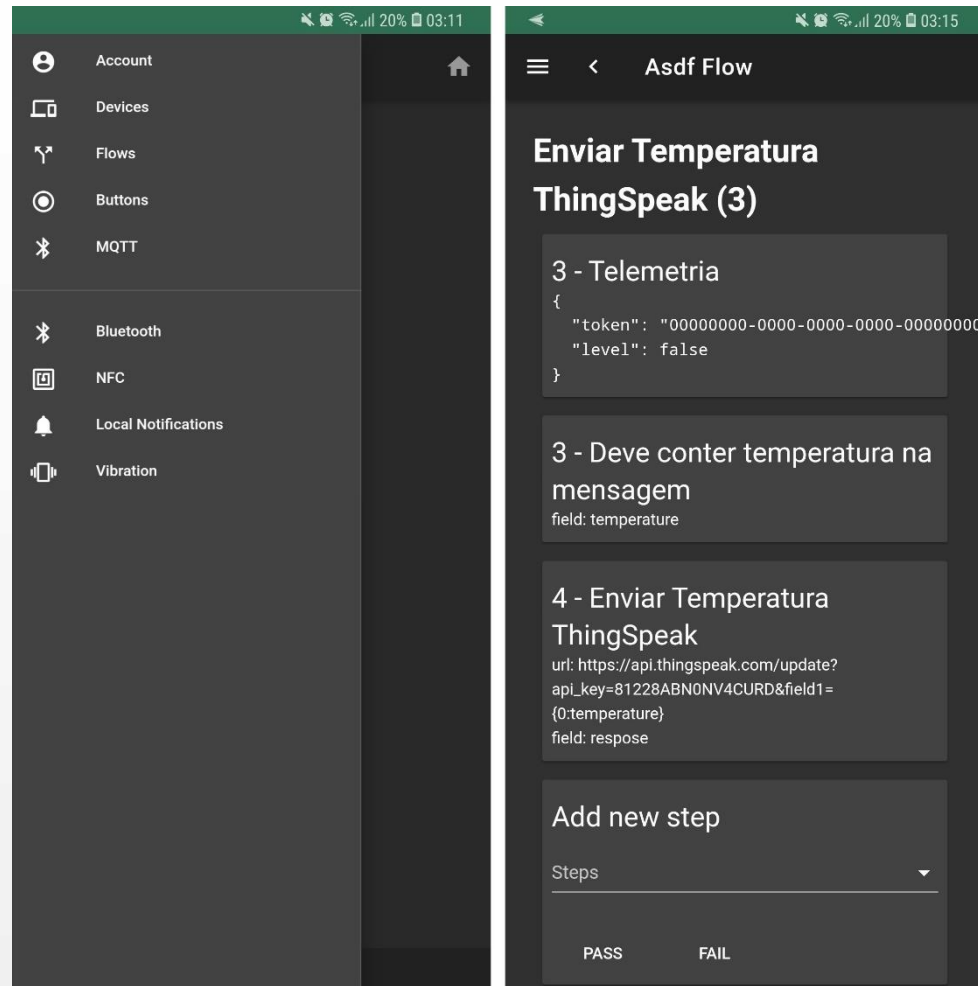
```
void bluetoothLoop(  
  float water, float temperature, float humidity, bool vibration, bool motion, bool level) {  
  printHeadersOnBluetooth();  
  
  printLoopCounterOnBluetooth();  
  
  printSensorValueOnBluetooth(FIELD_DHT11_HUMIDITY, humidity);  
  printSensorValueOnBluetooth(FIELD_DHT11_TEMPERATURE, temperature);  
  printSensorValueOnBluetooth(FIELD_WATER, water);  
  printSensorValueOnBluetooth(FIELD_VIBRATION, vibration);  
  printSensorValueOnBluetooth(FIELD_MOTION, motion);  
  printSensorValueOnBluetooth(FIELD_LEVEL, level);  
  
  printEndOnBluetooth();  
}
```

# Implementação IoT Bancada

```
void esploop(float water, float temperature, float humidity, bool vibration, bool motion, bool level) {  
    int millsBetween = 500;  
  
    printHeaders();  
    printSensorValue(FIELD_DHT11_HUMIDITY, humidity);  
    printEnd();  
    delay(millsBetween);  
  
    printHeaders();  
    printSensorValue(FIELD_DHT11_TEMPERATURE, temperature);  
    printEnd();  
    delay(millsBetween);  
  
    printHeaders();  
    printSensorValue(FIELD_WATER, water);  
    printEnd();  
    delay(millsBetween);  
  
    printHeaders();  
    printSensorValue(FIELD_VIBRATION, vibration);  
    printEnd();  
    delay(millsBetween);  
  
    printHeaders();  
    printSensorValue(FIELD_MOTION, motion);  
    printEnd();  
    delay(millsBetween);  
  
    printHeaders();  
    printSensorValue(FIELD_LEVEL, level);  
    printEnd();  
    delay(millsBetween);  
}
```

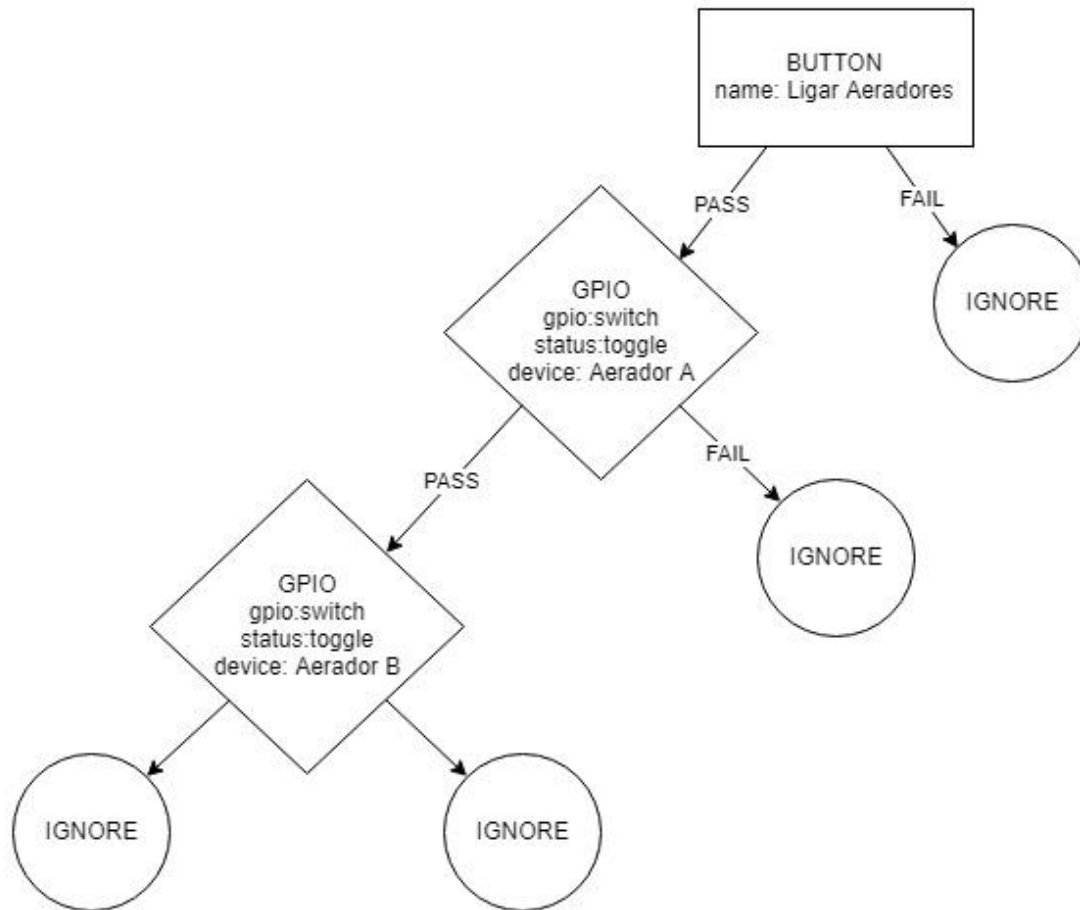


# Operacionalidade PWA



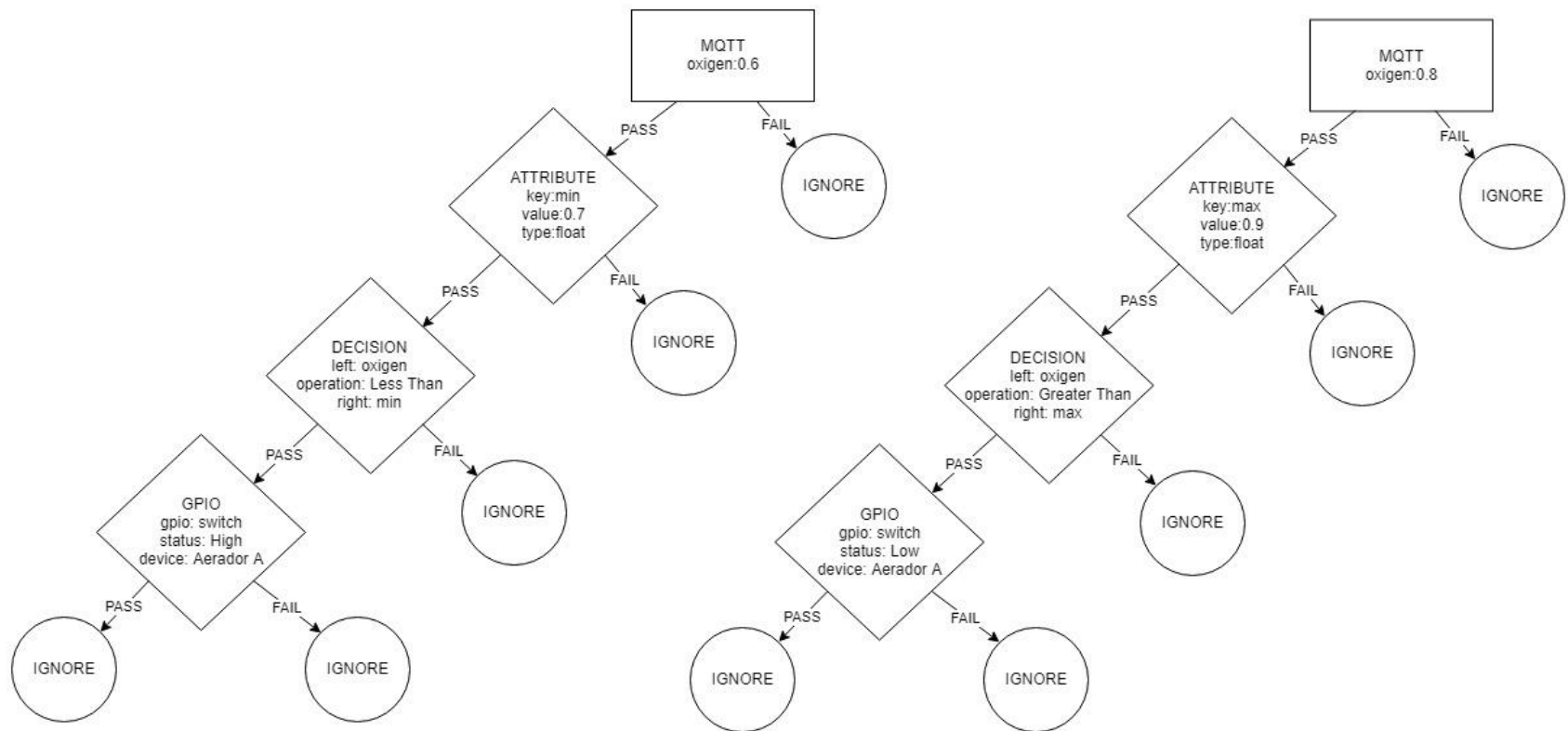
# Resultados e Discussões

## Solução Piscicultura



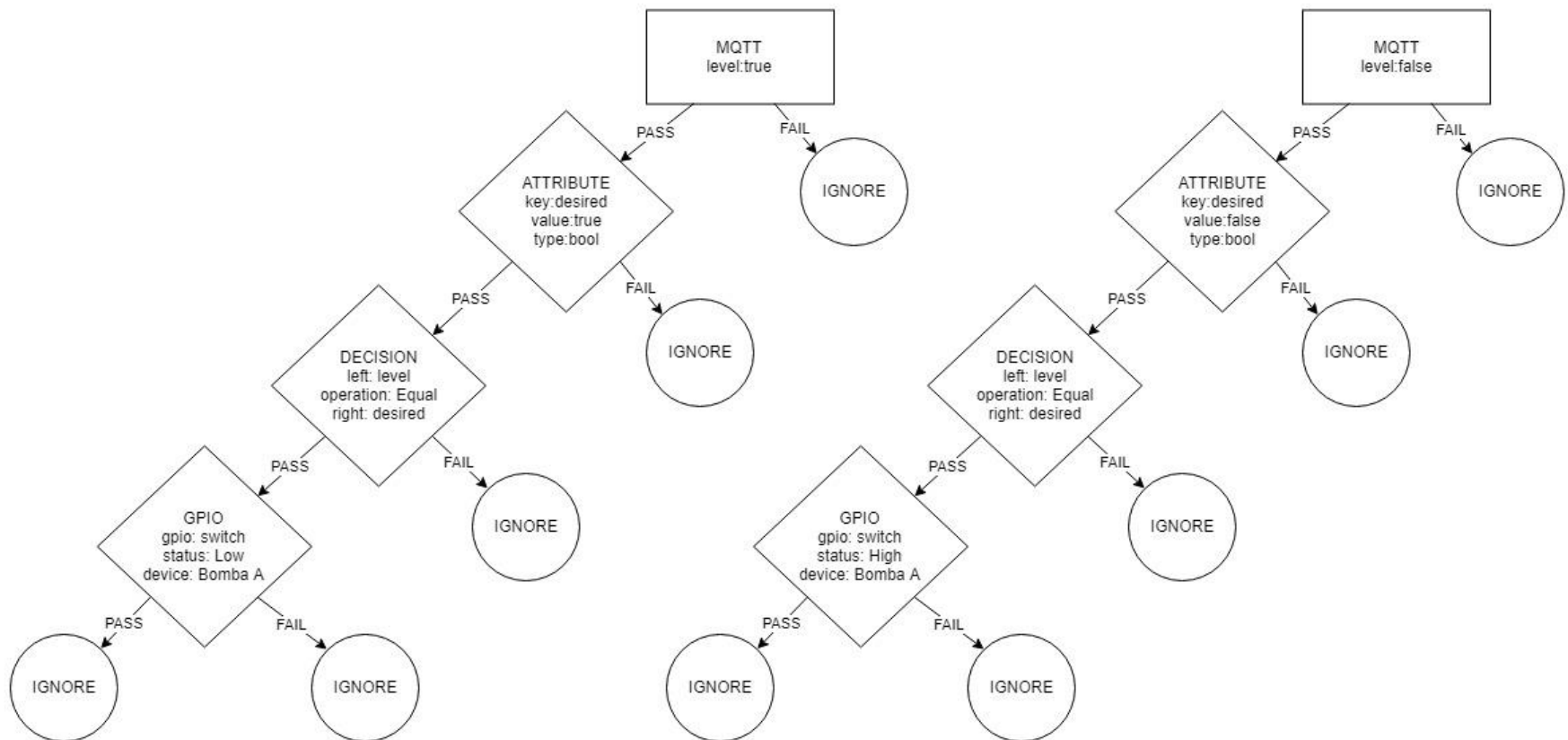
# Resultados e Discussões

## Solução Piscicultura



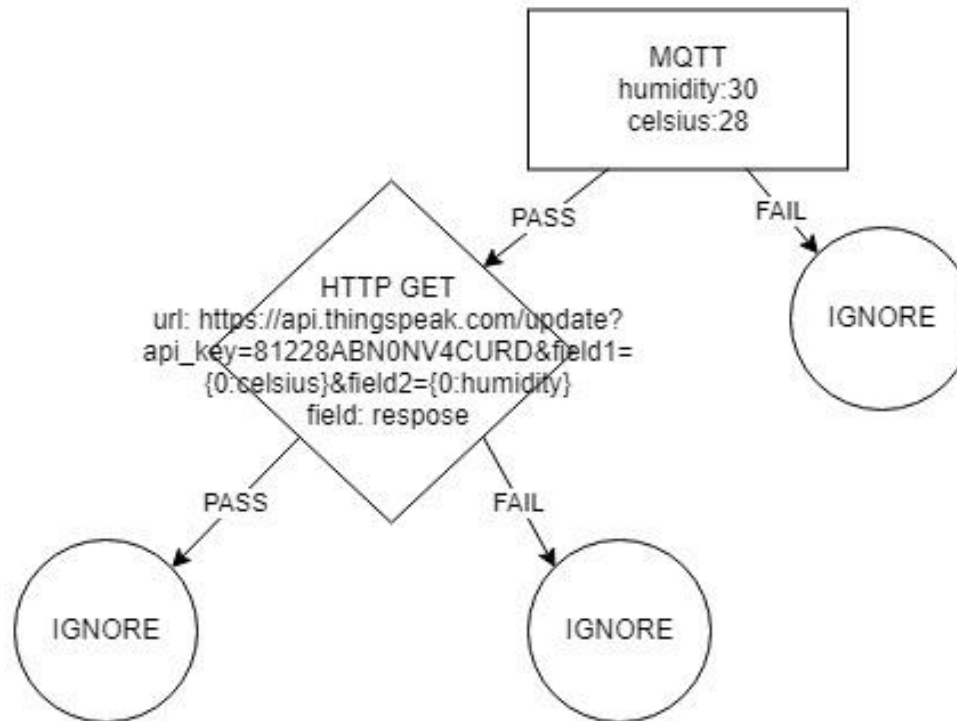
# Resultados e Discussões

## Solução Arroz Irrigado




# Resultados e Discussões

## Solução Fiação



# Conclusões e Sugestões

- Alterar a arquitetura para que a execução do fluxo possa ocorrer em uma rede local ou internamente nos dispositivos
- Desenvolver novos gatilhos e novos nodos de ação, de forma a  ender novos casos de uso
- Definir de maneira mais detalha o protocolo de mensagem, adicionando novos tipos de interesse ao IoT, por exemplo um tipo gpio, definindo os estados possíveis deste tipo

# Conclusões e Sugestões

- Desenvolver um firmware universal compatível com o protocolo de
- Adicionar a possibilidade de realizar o envio do firmware para o dispositivo utilizando a aplicação de interface de usuário e a API WebUSB