node SERVER

Rafael Hoffmann

raelhoff@{gmail.com/edu.univali.br}

UNIVALI

Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

JSONFILE

Fácil leitura / gravação de arquivos JSON.

Instalação do módulo: npm install jsonfile

Site oficial: https://www.npmjs.com/package/jsonfile

JSONFILE - (file.json)

```
{
    "modulo":"JSONFILE",
    "curso":"NodeJs"
}
```

JSONFILE - (readfile.js)

```
var jsonfile = require('jsonfile')

var file = './data.json'

jsonfile.readFile(file, function(err, obj) {
        if(err) {
                console.log("Erro")
        }else{
                console.log(" \nInformações Assíncronas: \n");
                console.log(obj);
        }
})

var json = jsonfile.readFileSync(file);
```

JSONFILE - (writefile.js)

```
var jsonfile = require('jsonfile')

var file = './data1.json'

////////////Assíncrono//////////////////////
var obj = {name: 'JP'}
jsonfile.writeFile(file, obj, function (err) {
  if(err) console.error(err)
})

////////////Síncrono//////////////////////
var obj2 = {type2: 'Sincrono'}
jsonfile.writeFileSync(file, obj2)
```

Desafio 1

Dentro da pasta Desafio 1:

- Crie um novo módulo readfile.js para realizar a leitura do arquivo file.json. Lembrando que este módulo deve ser exportado na forma de uma função.
- O módulo server.js deve ser configurado com a porta que está definida no arquivo file.json
- O módulo server.js deve retornar a lista de contatos do arquivo file.json quando for realizado um get (‘/contatos’)

XMLDOC

xmldoc permite analisar documentos XML com facilidade.

Instalação do módulo: npm install xmldoc

Site oficial: https://www.npmjs.com/package/xmldoc

XMLDOC

```
<informacoes>
	<dia>01/06/2017</dia>
	<pessoas>
		<pessoa>
			<sexo>feminino</sexo>
			<primeironome>Anna</primeironome>
			<últimonome>Smith</últimonome>
		</pessoa>
		<pessoa>
			<sexo>masculino</sexo>
			<primeironome>Pedro</primeironome>
			<últimonome>Souza</últimonome>
		</pessoa>
	</pessoas>
</informacoes>
```

XMLDOC

```
var data = fs.readFileSync('./file.xml');
var XMLPessoa = new xmldoc.XmlDocument(data);

console.log('\n' + XMLPessoa.childNamed("dia").toString());
console.log(XMLPessoa.valueWithPath("dia") + "\n");


XMLPessoa.childNamed("pessoas").eachChild(function(child, index, array){
	//console.log(array);
	console.log('\n' + child.childNamed("sexo").toString());
	console.log(child.valueWithPath("sexo"));
	console.log(child.childNamed("primeironome").toString());
	console.log(child.valueWithPath("primeironome"));
	console.log(child.childNamed("últimonome").toString());
	console.log(child.valueWithPath("últimonome"));
})
```

Desafio 2

Dentro da pasta Desafio 2:

- Crie um novo módulo readfile.js para realizar a leitura do arquivo file.xml. Lembrando que este módulo deve ser exportado na forma de uma função.
- O módulo server.js deve ser configurado com a porta que está definida no arquivo file.xml
- O módulo server.js deve retornar a lista de contatos do arquivo file.xml quando for realizado um get (‘/contatos’)

LOG4JS

LOG4JS permite criar logs com facilidade.

- Substitui a função console.log
- O log do console é colorido para stdout ou stderr
- Permite a criação de arquivo sendo configurável com base no tamanho

Instalação do módulo: npm install log4js

site oficial: https://www.npmjs.com/package/log4js

LOG4JS

Vamos ver um exemplo

Desafio 3

Dentro da pasta Desafio 3, atualize o projeto (lista dinâmica de contatos) de forma que seja registrado todas as requisições por meio do módulo LOG4JS os métodos:

- get('/contatos') -> para retornar a lista de contatos
- get('/contatos/:id') -> retorna um contato específico
- post('/contatos') -> adicionar um novo contato na lista. Exemplo:
- put('/contatos/:id') -> atualizar cadastro. Exemplo:
- delete('/contatos') -> delete a primeira posição do cadastro cadastro. Dica, utilize a função

- Modifique o arquivo ControlLog.js para excluir arquivos que foram criados no último minuto (60000 ms)
- Crie um setInterval para chamar o método ApagaLog.js a cada minuto

(60000 ms)

C-STRUCT

Suporta os seguintes recursos:

- int8, int16, int24, int32, int40 e int48.
- String.
- Boolean, float e double.

Instalação do módulo: npm install c-struct

Site oficial: https://www.npmjs.com/package/c-struct

C-STRUCT

Vamos ver um exemplo

Desafio 4

O exemplo visto demonstrou a utilização do módulo C-STRUCT empregado na comunicação UDP/DGRAM.

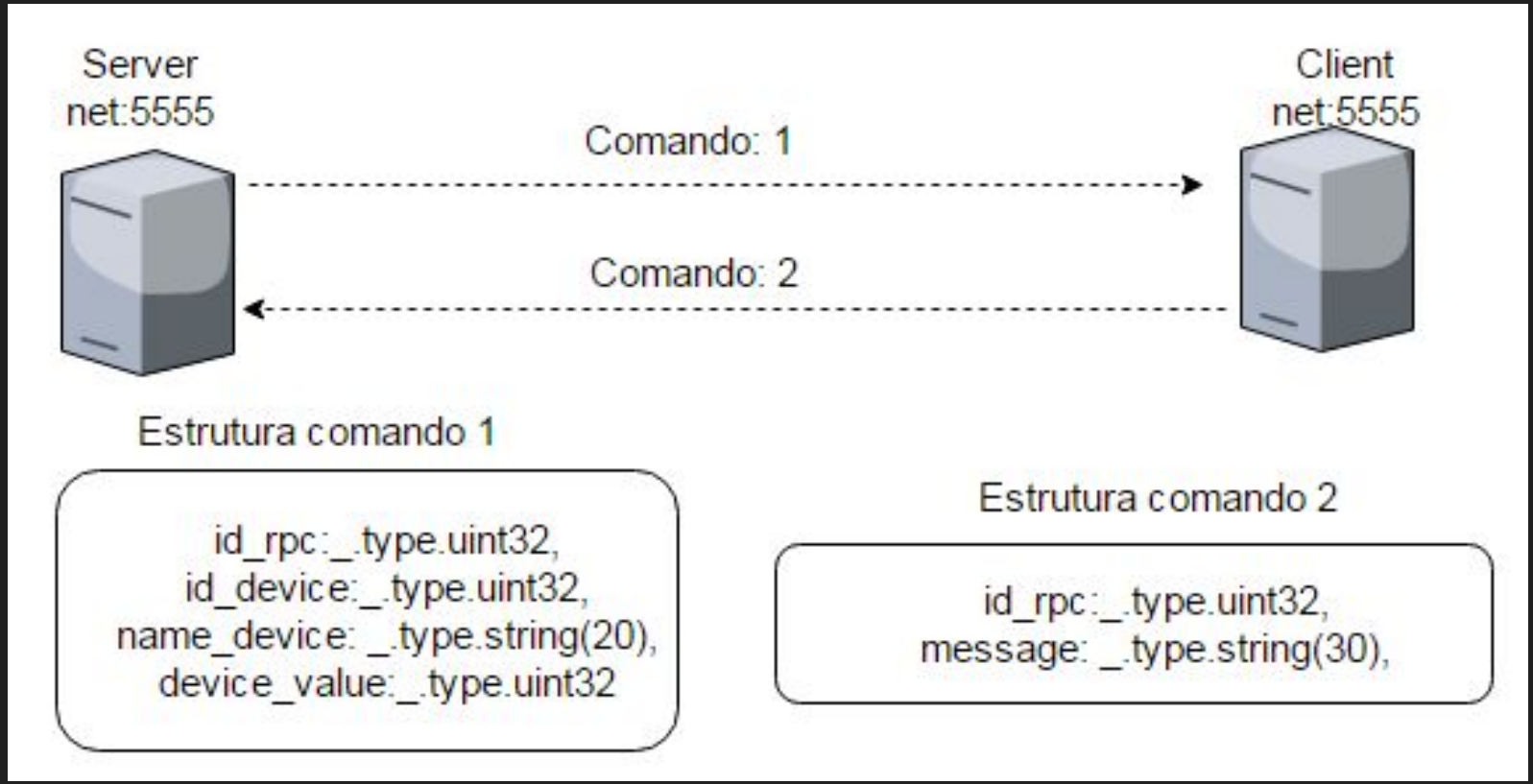
Agora realize a mesma comunicação utilizando o módulo Net/TCP.

Dicas:

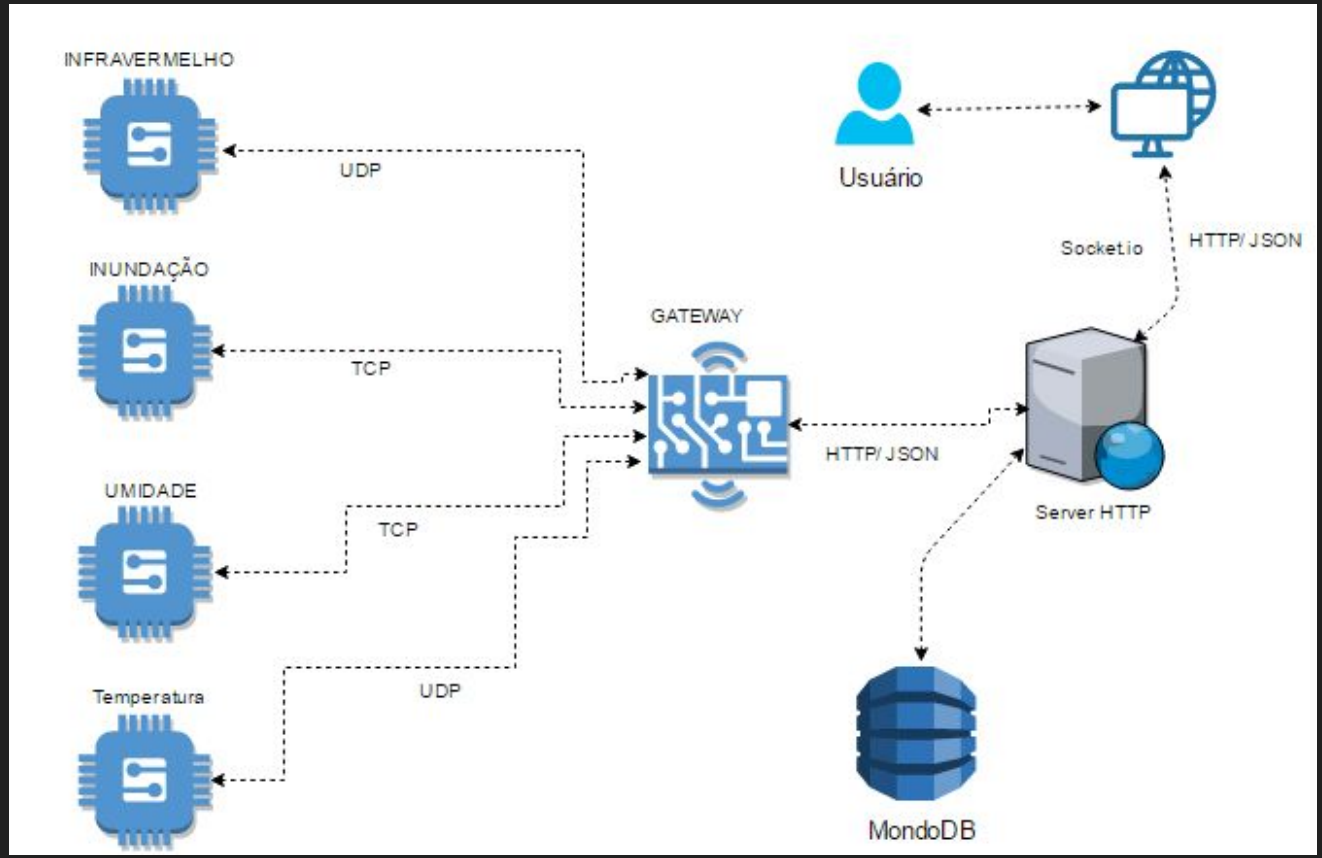
- Utilize as mesmas estruturas empregados no exemplo DGRAM.

```
var Datagrama1 = new _.Schema({
    id_rpc:_.type.uint32,
    message: _.type.string(30),});
```

Desafio 4


Server
net:5555
Client
net:5555
Comando: 1
Comando: 2
Estrutura comando 1
id_rpc:_.type.uint32,
id_device:_.type.uint32,
name_device: _.type.string(20),
device_value:_.type.uint32
Estrutura comando 2
id_rpc:_.type.uint32,
message: _.type.string(30),

Projeto piloto



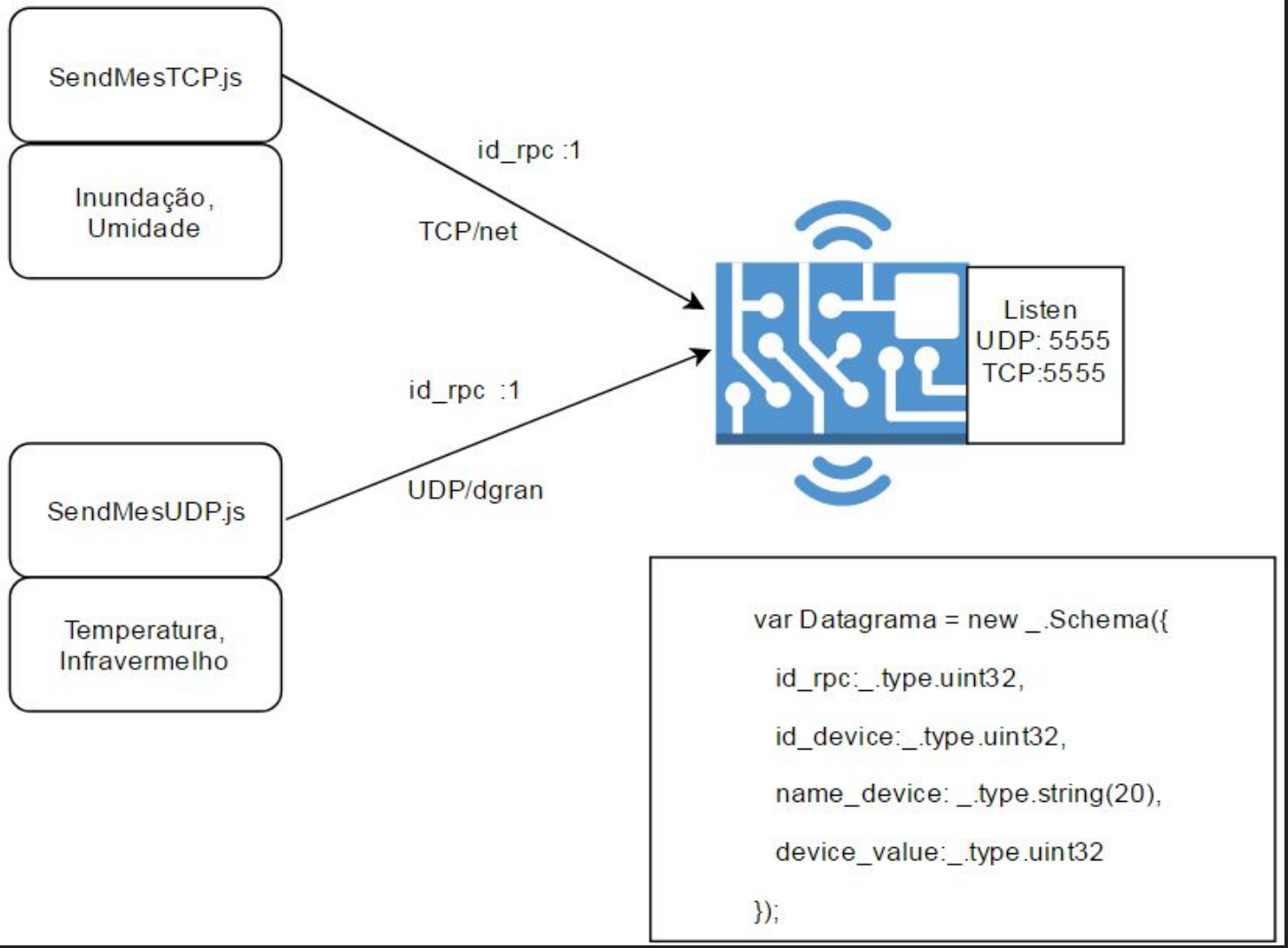
Projeto piloto - (Sensores)

Dentro da pasta Projeto Piloto:

1. Crie uma pasta sensores
2. Crie um novo módulo SendMesTCP.js para enviar mensagens TCP utilizando a estrutura da variável Datagrama.
   - Dicas: poderá ser enviado as informações dos dispositivos (umidade e inundação)
   - Após enviar mensagem, o cliente deve desconectar do server.
3. Crie um novo módulo SendMesUDP.js para enviar mensagens UDP utilizando a estrutura da variável Datagrama.

Dica: Deve ser enviado as informações dos dispositivos(temperatura e infravermelho)

```
var Datagrama = new _.Schema({
    id_rpc:_.type.uint32,
    id_device:_.type.uint32,
    name_device: _.type.string(20),
    device_value:_.type.uint32
});
```


SendMesTCP.js
Inundação,
Umidade
id_rpc :1
TCP/net
Listen
UDP: 5555
TCP:5555
id_rpc :1
SendMesUDP.js
UDP/dgran
Temperatura,
Infravermelho
var Datagrama = new _.Schema({
id_rpc:_.type.uint32,
id_device:_.type.uint32,
name_device: _.type.string(20),
device_value:_.type.uint32
});

Projeto piloto - (Gateway)

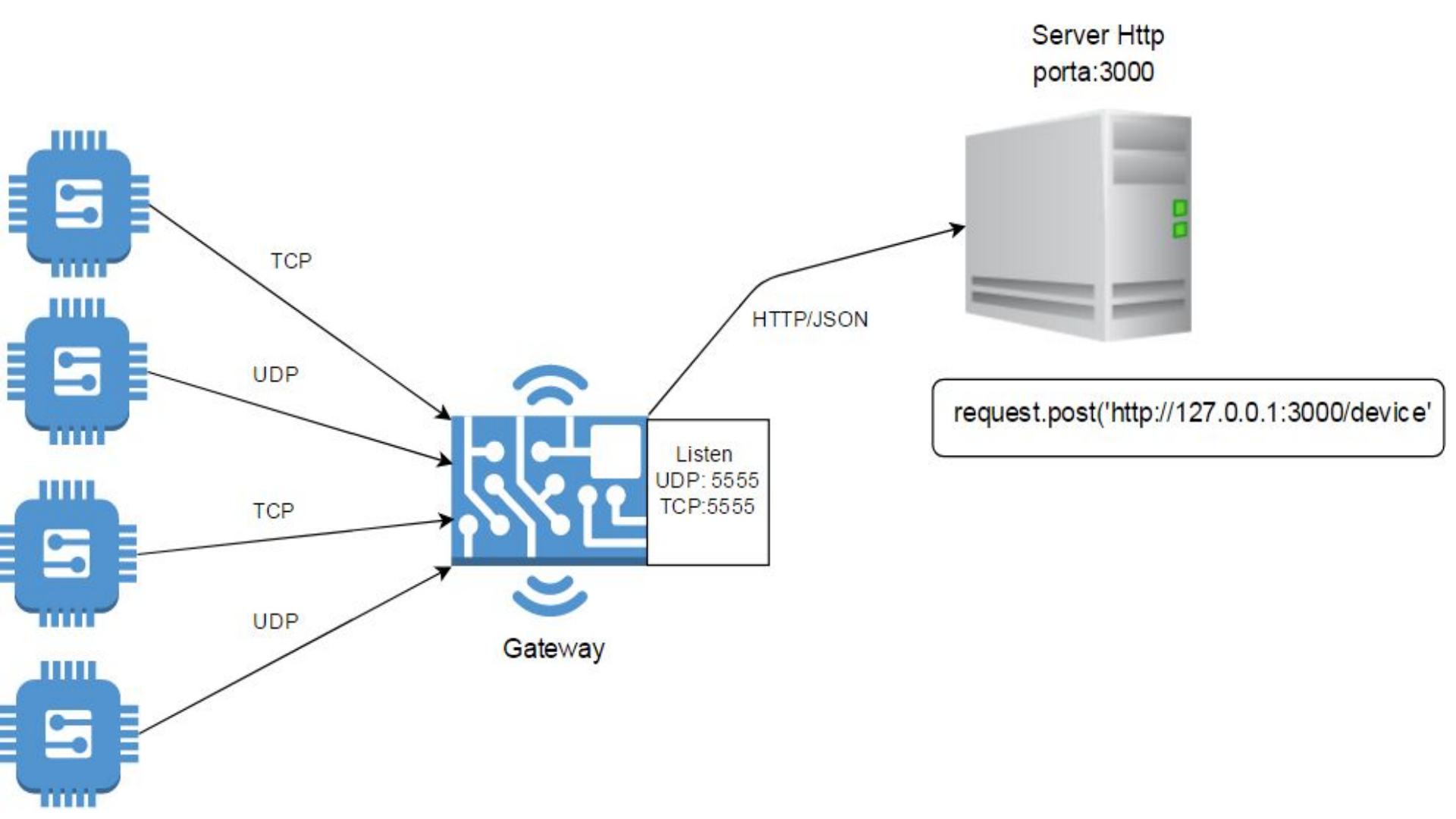
Dentro da pasta Projeto Piloto:

1. Crie uma pasta Gateway.
2. Crie um novo módulo main.js. Este módulo será responsável por receber todas as notificações (UDP e TCP) .
3. Após receber a notificação (Datagrama) o mesmo deverá transmitir a mensagem recebida no formato JSON para o servidor Web por meio do módulo request.

   Dica:

   - Utilizar o método Post no request
   - request.post('http://127.0.0.1:3000/device'

```
var Datagrama = new _.Schema({
    id_rpc:_.type.uint32,
    id_device:_.type.uint32,
    name_device: _.type.string(20),
    device_value:_.type.uint32
});
```


Server Http
porta:3000
TCP
UDP
TCP
UDP
HTTP/JSON
Listen
UDP: 5555
TCP:5555
Gateway
request.post('http://127.0.0.1:3000/device'

Projeto piloto - (ServidorWeb)

Dentro da pasta Projeto Piloto:

1. Crie uma pasta ServidorWeb

2. Crie um novo módulo server.js. Este módulo será responsável por receber todas as notificações (HTTP - '/device') . Dicas:
   - Utilize o módulo express, bory-parser

3. Após receber a notificação (no formato JSON) o mesmo deverá transmitir a mensagem recebida para a página Web por meio do módulo socket.io. Dicas:

   - io.emit('devices', req.body);


Server Http
porta:3000
Página Web
socket.io
Usuário
HTTP/JSON
Listen
UDP: 5555
TCP:5555
127.0.0.1:3000
Lista de Devices:
2 | UMIDADE | 10
3 | INUNDACAO | 0
1 | TEMPERATURA | 30
4 | INFRAVERMELHO | 15

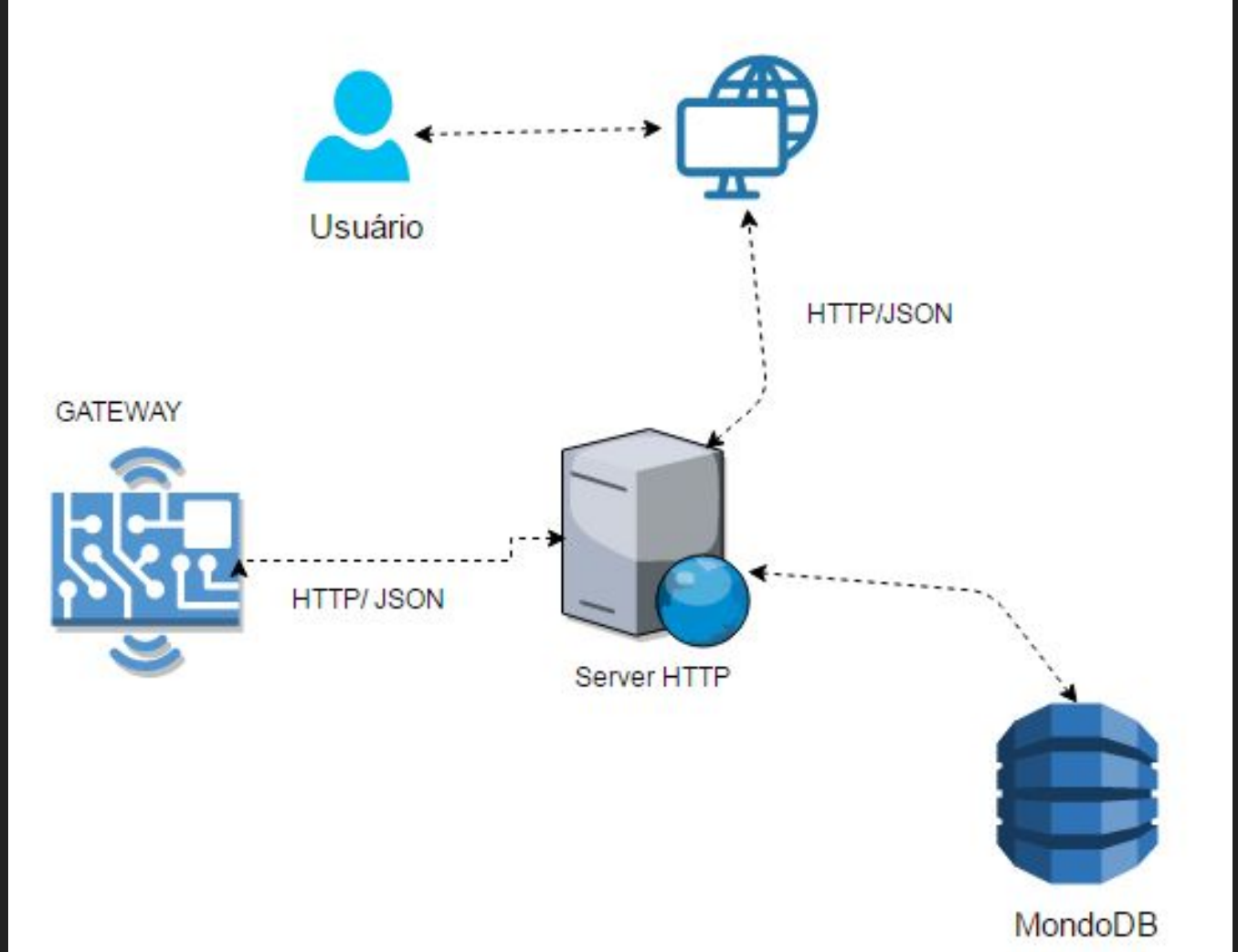
Projeto piloto - (ServidorWeb_database)

Dentro da pasta Projeto Piloto:

1. Realize a integração entre o módulo mongoose com o módulo servidor Web. Dica: require('../mongoose/database.js')('mongodb://localhost/Devices');

2. Save todos as informações enviadas pelo Gateway no banco dados

3. Retorne no formato JSON todas as informações que foram salvas no banco de dados.Dica:
   - app.get('/devices') return json


Usuário
HTTP/JSON
GATEWAY
HTTP/ JSON
Server HTTP
MondoDB

Referências e Sugestões

Vídeo aulas do YouTube

- AngularJS - Rodrigo Branas
  https://www.youtube.com/watch?v=_y7rKxqPoyg

Livros da casa da código (https://www.casadocodigo.com.br)

- Construindo APIs REST com Node.js
  https://www.casadocodigo.com.br/products/livro-apis-nodejs

- Aplicações web real-time com Node.js
  https://www.casadocodigo.com.br/products/livro-nodejs

- ECMAScript 6 - Entre de cabeça no futuro do JavaScript
  https://www.casadocodigo.com.br/products/livro-ecmascript6