

Serviço Nacional de Aprendizagem Comercial do Rio Grande do Sul  
Faculdade Senac Porto Alegre  
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Fabiano da Rosa Gomes

# **Ysto** **Automação Residencial**

Porto Alegre

2017

Fabiano da Rosa Gomes

**Ysto  
Automação Residencial**

Relatório Final de Projeto, apresentado como requisito parcial à obtenção do grau de Técnologo em Análise e Desenvolvimento de Sistemas, pela Faculdade Senac Porto Alegre. Orientador: Prof. Me. Luciano Zanuz.

Porto Alegre

2017

## RESUMO

O presente projeto propõem um sistema de automação residencial, provendo o controle e o acionamento de dispositivos que são alimentados por energia elétrica em corrente alternada, disponível nas tomadas residenciais. Para tanto, usa um conjunto composto de uma placa ESP8266 (wifi) e uma RaspberryPI (central de controle). Nesta central de controle é possível manipular mais de um módulo wifi, ligando e desligando suas saídas através do protocolo MQTT. Foi utilizado as linguagens de programação C, Python e Javascript no desenvolvimento deste projeto.

**Palavras-chaves:** automação residencial. iot. mqtt. python. javascript.

# LISTA DE ILUSTRAÇÕES

Figura 1 – Possibilidades da domótica nos dias de hoje . . . . .	13
Figura 2 – Exemplos de IHM convencionais . . . . .	14
Figura 3 – Levantamento aproximado de custos . . . . .	16
Figura 4 – Alternativa pronta para o MA . . . . .	16
Figura 5 – Diagrama esquemático de uma RaspberryPI . . . . .	18
Figura 6 – Diagrama esquemático de uma ESP8266-01 . . . . .	19
Figura 7 – Topologia da solução . . . . .	22
Figura 8 – Esquemático eletrônico do MA . . . . .	23
Figura 9 – Arduíno IDE . . . . .	23
Figura 10 – Gravação do <i>firmware</i> no MA . . . . .	24
Figura 11 – Central de controle . . . . .	24
Figura 12 – Esquemático de serviços da central de controle . . . . .	25
Figura 13 – Mensagem sendo enviada via MQTT . . . . .	26
Figura 14 – Mensagem sendo recebida via MQTT . . . . .	26
Figura 15 – Modelo publish/subscribe . . . . .	27
Figura 16 – Um exemplo de estrutura de tópicos . . . . .	27
Figura 17 – O papel do broker . . . . .	27
Figura 18 – Estrutura de tópicos adotada neste projeto . . . . .	28
Figura 19 – Funcionamento do Scrum solo . . . . .	29
Figura 20 – Levantamento de requisitos . . . . .	30
Figura 21 – Manipulação de artefatos na Sprint . . . . .	31
Figura 22 – Entrega de software . . . . .	32
Figura 23 – Gestão de projeto . . . . .	32
Figura 24 – Documento de escopo . . . . .	34
Figura 25 – Detalhamento sprint 1 . . . . .	36
Figura 26 – Montagem das placas de fixação dos módulos . . . . .	36
Figura 27 – Módulo ESP8266 tipo 01 . . . . .	37
Figura 28 – Módulo ESP8266 tipo 12 . . . . .	37
Figura 29 – Detalhamento sprint 2 . . . . .	38
Figura 30 – Detalhamento sprint 3 . . . . .	39
Figura 31 – Detalhamento sprint 4 . . . . .	39
Figura 32 – Detalhamento sprint 5 . . . . .	40
Figura 33 – Testes unitários da API . . . . .	41

Figura 34 – Detalhamento sprint 6 . . . . .	42
Figura 35 – Estrutura do projeto . . . . .	43
Figura 36 – Detalhamento sprint 7 . . . . .	44
Figura 37 – Detalhamento sprint 8 . . . . .	44
Figura 38 – Acionamento de carga . . . . .	45
Figura 39 – Leitura de dados . . . . .	45
Figura 40 – Acionamento através da API . . . . .	46
Figura 41 – Recurso users na API . . . . .	47
Figura 42 – Recurso devices na API . . . . .	47
Figura 43 – Controle de acesso da API . . . . .	48
Figura 44 – Solicitação ou request . . . . .	48
Figura 45 – Resposta ou response . . . . .	49
Figura 46 – Representação de devices . . . . .	50
Figura 47 – Representação de users . . . . .	51
Figura 48 – ER do banco de dados . . . . .	52
 Figura 49 – Informações da API . . . . .	54
Figura 50 – Requisição de token de acesso . . . . .	55
Figura 51 – Manutenção de usuários . . . . .	56
Figura 52 – Manutenção de dispositivos . . . . .	57
Figura 53 – Atualização de estado de um dispositivo . . . . .	58
Figura 54 – Tela de login . . . . .	59
Figura 55 – Tela do painel de controle . . . . .	60
Figura 56 – Tela de cadastro de novos usuários . . . . .	61
Figura 57 – Tela de cadastro de novos dispositivos . . . . .	62
Figura 58 – Tela de controle dos dispositivos . . . . .	63
 Figura 59 – Testes funcionais automatizados . . . . .	64
Figura 60 – Embalagem conceitual do módulo auxiliar . . . . .	65
Figura 61 – <i>Case</i> comercial da central de controle . . . . .	65
 Figura 62 – Controle horas trabalhadas . . . . .	68

## LISTA DE TABELAS

Tabela 1 – Presença dos produtos eletroeletrônicos nos domicílios brasileiros . . . . .	11
Tabela 2 – User Stories . . . . .	35
Tabela 3 – Product Backlog . . . . .	35

## LISTA DE ABREVIATURAS E SIGLAS

AC	Alternate Current
API	Application Program Interface
AURESIDE	Associação Brasileira de Automação Residencial e Predial
CHPD	Chip Select
CORS	Cross-Origin Resource Sharing
CRUD	Acrônimo do inglês para Create, Read, Update e Delete
ER	Entidade Relacionamento
GND	Ground
GPIO	General Purpose Input/Output
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	Integrated Development Environment
IHM	Interface Homem Máquina
JSON	JavaScript Object Notation
M2M	Machine to Machine
MA	Módulo Auxiliar
MD5	Message-Digest algorithm 5
MQTT	Message Queue Telemetry Transport
MVR	Model-View-Router
NPM	Node Package Manager
ORM	Object-relational mapping
PSP	Personal Software Process
PWA	Progressive Web Apps
REST	REpresentational State Transfer
RST	Reset

RX	Receiver
SOC	System-On-Chip
TX	Transmitter
VCC	Power Supply
W3C	World Wide Web Consortium

# SUMÁRIO

<b>1 Apresentação geral do projeto . . . . .</b>	<b>11</b>
<b>2 Definição do problema . . . . .</b>	<b>14</b>
<b>3 Objetivos . . . . .</b>	<b>17</b>
3.1 Objetivo geral . . . . .	17
3.2 Objetivos específicos . . . . .	17
<b>4 Análise de tecnologias e ferramentas . . . . .</b>	<b>18</b>
4.1 RaspberryPI . . . . .	18
4.2 ESP8266 . . . . .	18
4.3 Arduíno . . . . .	19
4.4 Javascript . . . . .	19
4.5 Python . . . . .	19
4.6 SQLite . . . . .	20
4.7 MQTT . . . . .	20
4.8 Mosquitto . . . . .	20
4.9 Debian Linux . . . . .	20
4.10 Geany IDE . . . . .	20
4.11 Schemacrawler . . . . .	21
4.12 Git SCM . . . . .	21
4.13 Gliffy . . . . .	21
4.14 VueJS . . . . .	21
4.15 Skeleton . . . . .	21
<b>5 Descrição da solução . . . . .</b>	<b>22</b>
5.1 Módulos Auxiliares . . . . .	22
5.1.1 Esquema eletrônico do MA . . . . .	22
5.1.2 Programação do <i>firmware</i> do MA . . . . .	23
5.2 Central de controle . . . . .	24
5.3 Funcionamento do protocolo MQTT . . . . .	25
5.3.1 Tópico para controle de um MA . . . . .	27
5.4 Ponto de entrada no sistema . . . . .	28
<b>6 Abordagem de desenvolvimento . . . . .</b>	<b>29</b>
6.1 Levantamento de requisitos . . . . .	30
6.2 Sprint . . . . .	30

6.3	Entrega . . . . .	31
6.4	Gestão . . . . .	32
6.5	Repositório de artefatos . . . . .	33
6.6	Scrum solo neste projeto . . . . .	33
<b>7</b>	<b>Arquitetura do sistema . . . . .</b>	<b>34</b>
7.1	Modelagem funcional . . . . .	34
7.1.1	Documento de escopo . . . . .	34
7.1.2	User Stories . . . . .	34
7.1.3	Sprints . . . . .	35
7.2	Modelagem do processo de negócio . . . . .	45
7.2.1	Firmware do módulo auxiliar . . . . .	45
7.2.2	Controle de dispositivos através da API . . . . .	45
7.3	Modelagem de dados . . . . .	46
7.3.1	Modelagem da API . . . . .	46
7.3.2	Definição de recursos . . . . .	47
7.3.3	Controle de acesso . . . . .	47
7.3.4	Estrutura de mensagens . . . . .	48
7.3.5	Código de status . . . . .	49
7.4	Modelagem do banco de dados . . . . .	50
<b>8</b>	<b>Funcionamento do sistema . . . . .</b>	<b>53</b>
8.1	Firmware para controle do módulo WIFI . . . . .	53
8.2	API para controle de Hardware . . . . .	53
8.2.1	Sobre esta API . . . . .	53
8.2.2	Solicitação de token . . . . .	54
8.2.3	Manutenção de usuários do sistema . . . . .	55
8.2.4	Manutenção de dispositivos do sistema . . . . .	56
8.2.5	Controle de dispositivo . . . . .	57
8.3	Interface Gráfica para controle do sistema . . . . .	58
8.3.1	Acesso ao sistema . . . . .	58
8.3.2	Painel de controle . . . . .	59
8.3.3	Cadastro de novos usuários . . . . .	60
8.3.4	Cadastro de novos dispositivos . . . . .	61
8.3.5	Controle de dispositivos . . . . .	62
8.3.6	Retorno para tela inicial . . . . .	63
<b>9</b>	<b>Validação . . . . .</b>	<b>64</b>
9.1	Testes funcionais automatizados - API . . . . .	64
9.2	Embalagem e apresentação . . . . .	64

9.3	Alcance de acionamento . . . . .	65
9.4	Interface com usuário . . . . .	66
9.5	Escopo e oportunidade . . . . .	66
9.5.1	Sinalização para salas de espera . . . . .	66
9.5.2	Sinalização para praças de alimentação . . . . .	66
9.5.3	Situação atual . . . . .	66
<b>10</b>	<b>Considerações finais . . . . .</b>	<b>68</b>
10.1	Funcionalidades futuras . . . . .	68
10.1.1	Segurança . . . . .	68
10.1.2	Hospedagem em servidor externo . . . . .	69
10.1.3	Mais opções de dispositivos . . . . .	69
<b>Referências . . . . .</b>		<b>70</b>

# 1 APRESENTAÇÃO GERAL DO PROJETO

A quantidade de produtos eletroeletrônicos cresce cada vez mais nas residências brasileiras, conforme mostra a pesquisa realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE) em 2016. A tabela 1 apresenta a presença em números percentuais de alguns eletrodomésticos nos lares brasileiros.

Tabela 1 – Presença dos produtos eletroeletrônicos nos domicílios brasileiros

Produto	2012	2013	2014	2015
Fogões	98,75%	98,76%	98,81%	98,84%
Televisores	97,20%	97,16%	97,14%	97,14%
Refrigeradores	96,65%	97,21%	97,56%	97,83%
Rádio	80,86%	75,71%	72,08%	69,23%
Máquinas de lavar	55,14%	57,46%	58,68%	61,14%
Freezer	16,66%	17,05%	16,48%	16,88%

Fonte: IBGE 2016

Junto com o crescimento de produtos deste segmento, o acesso à internet também cresceu muito nos últimos anos. Em matéria do portal G1 de 06 de Abril de 2016<sup>1</sup>, é destaque a quantidade de casas com acesso a internet, 50 por cento em média. Estes números acabam fortalecendo um outro mercado que nasce quase como uma consequência do consumo de eletrônicos e acesso a internet, o mercado de automação residencial. Automação residencial, ou domótica, segundo o Departamento de Informática (DIN) da Universidade Estadual de Maringá (UEM) pode ser definido como:

O termo domótica, é uma fusão da palavra latina domus (casa) e do moderno robótica. A domótica, que também pode ser referenciada por expressões como "smart building", "intelligent building", "edifícios inteligentes", é um novo domínio de aplicação tecnológica, tendo como objetivo básico melhorar a qualidade de vida, reduzindo o trabalho doméstico, aumentando o bem estar e a segurança de seus habitantes e visa também uma utilização racional e planejada dos diversos meios de consumo. A domótica procura uma melhor integração através da automatização nas áreas de segurança, de comunicação e de controle, e gestão de fluídos. (MARINGA, 2017)

Com base nas definições conceituais anteriores, é correto afirmar que a domótica cuida da integração de sistemas e tem por finalidade tornar o ambiente doméstico e os diversos equipamentos que fazem parte dele mais fáceis de serem utilizados por qualquer morador da casa, com ou sem conhecimentos especializados em tecnologia da informação. Investir em automação residencial garante praticidade e objetividade no dia a dia sem

<sup>1</sup> <http://g1.globo.com/tecnologia/noticia/2016/04/internet-chega-pela-1-vez-mais-de-50-das-casas-no-brasil-mostra-ibge.html> acessado em 14/11/2017

prejudicar o conforto e segurança<sup>2</sup>. Este mercado oferta diversas possibilidades, como o desenvolvimento de sistemas personalizados para atender necessidades específicas, a busca por redução de consumo energético ou ainda a preocupação com a segurança podem ser a motivação para busca de soluções nesta área. Analisando um mercado maior que o nosso e que normalmente dita as tendências de futuro, a Associação Brasileira de Automação Residencial e Predial (AURESIDE) destaca:

Pesquisas da Parks Associates mostram que mais de 100 milhões de lares dos EUA não tinham um dispositivo doméstico inteligente no final de 2016. Analistas da empresa internacional observam que alcançar essas famílias exige investimentos contínuos para criar experiências únicas e personalizadas para os moradores... A empresa de pesquisa, em seu serviço NUMBERS, estima que até 2020, mais de 12 milhões de lares americanos terão um detector inteligente de vazamento de água, mais de 40 milhões terão um termostato inteligente, quase 50 milhões terão uma lâmpada inteligente e cerca de 14 milhões terão um controlador doméstico inteligente. A Conferência CONNECTIONS deste ano examinará as oportunidades de investimento no IOT que impulsionarão esse crescimento e sua adoção em soluções domésticas inteligentes. (AURESIDE, 2017).

Diante deste cenário promissor, soluções mais acessíveis precisam ser criadas pensando na realidade brasileira. Falar sobre ter uma opção acessível não está ligado somente as questões financeiras, embora estas tenham um peso significativo na escolha. Facilidade de obter os componentes, configurá-los e integrar isso com padrões existentes são o diferencial deste projeto.

Este projeto apresenta uma solução para automação residencial, baseada em padrões abertos que não dependem do pagamento de licenças ou direitos autorais, o que facilita a integração com outros sistemas ou dispositivos. Também utiliza componentes eletrônicos de fácil acesso e com preços razoáveis, levando em conta as soluções existentes de mercado nacional. A figura 1 mostra, de forma esquemática, as possibilidades disponíveis com um sistema de domótica em nossas residências. Possibilidades estas que vão do controle de acesso a residencia, monitoramento de áreas específicas, controle do ambiente através de sensores de temperatura ou ainda a integração com outros dispositivos inteligentes como um aparelho de TV ou central de multimídia.

---

<sup>2</sup> <http://overbr.com.br/artigos/por-que-investir-em-automacao-residencial> acessado em 14/11/2017

Figura 1 – Possibilidades da domótica nos dias de hoje



Fonte: Website da Aureside (AURESIDE, 2017)

## 2 DEFINIÇÃO DO PROBLEMA

Diariamente novos dispositivos são inseridos no cotidiano da sociedade, essa variedade de "coisas" que fazem parte do cotidiano podem acabar se tornando um problema, principalmente para pessoas com pouca prática na configuração e utilização destes novos e modernos aparelhos. Neste ambiente desconexo, qualquer nova aquisição, como um novo ar condicionado ou uma fechadura eletrônica passam a ser mais um dispositivo com características novas e configurações diferentes que necessitam ser administrados por estes moradores. As primeiras interfaces homem máquina (IHM) que utilizam o controle remoto, por rádio frequência, normalmente são as opções mais comuns e variadas nas residências atuais. A figura 2 mostra algumas variações desta interface para controle das coisas de casa.

Figura 2 – Exemplos de IHM convencionais



Fonte: Website da digitaltrends

É comum que todos estes aparelhos funcionem de forma independente e desconectadas, cabendo aos usuários desvendar os segredos de cada IHM. O que deveria trazer facilidade e comodidade pode facilmente se tornar um forte motivo para aquela nova aquisição cair rapidamente em desuso ou simplesmente não ser utilizada da maneira correta. Outro cenário muito comum é a eleição de um "especialista" em novas tecnologias, as vezes um filho ou sobrinho descolado que domina as novidades recém chegadas. O fato é que esse acúmulo de possibilidades gera insatisfação e porque não dizer frustração nos usuários que gostariam apenas de desfrutar das soluções e confortos que estes bens prometem trazer e nem sempre cumprem essa tarefa a contento.

Nos últimos anos, uma tendência tem se mostrado forte, a integração destes apare-

lhos, empresas grandes como Google<sup>1</sup> , Amazon<sup>2</sup> e Apple<sup>3</sup> tem investido tempo e dinheiro para criar soluções que entregam uma experiência mais integrada ao cotidiano das pessoas. Este processo envolve a adaptação dos aparelhos existentes em conjunto com uma integração a novos sistemas e interfaces mais amigáveis. Todas as soluções citadas anteriormente utilizam o hábito dos moradores para fazer uma mapa e uma agenda de eventos, sugerindo com o passar do tempo certas ações que antes dependiam exclusivamente do controle humano, isso introduz o que convencionou chamar de Aprendizagem de Máquina associado a Sistemas de Recomendações. Estes sistemas prometem ser assistentes pessoais e conforme as demonstrações destes fabricantes, realmente vão mudar a maneira como nos relacionamos com as máquinas.

Neste processo, a captura de hábitos e o aprendizado que estas novas tecnologias prometem conseguir vão levar às próximas gerações a experimentar uma integração com as máquinas muito mais natural e porque não dizer orgânica. Fazer diferente o que de certa forma se acostuma pelo hábito é o que as soluções destas empresas propõem, interfaces de controle por voz, aprendizado de máquina e conexão com serviços externos são algumas das novidades que fazem parte do núcleo desta nova família de aplicativos e dispositivos.

Este mercado apostava em três itens para captar estes novos consumidores:

- a) Conforto;
- b) Segurança;
- c) Integração.

Nem sempre a tecnologia costuma ficar disponível para a grande maioria de forma rápida, tornar projetos de eletrônica e informática uma realidade no Brasil muitas vezes é um desafio e isso por vezes nos deixa nos tempos das cavernas em relação a países desenvolvidos, mesmo no mais otimista dos cenários ainda estamos longe de ter acesso as soluções de domótica existentes no exterior. As principais questões que dificultam o acesso a essas soluções são em primeiro lugar a falta de interesse comercial de trazer isso para nosso país, nossa infraestrutura embora tenha melhorado, ainda está longe do ideal. Os custos de serviço de *telecom* associado as altas taxas de importação de eletrônicos inviabilizam a chegada destas facilidades as lojas do nosso país. Estas dificuldades não vão fazer com que as pessoas deixem de querer isso e de alguma forma estas necessidades serão sanadas, dentro deste contexto que nasce o projeto de automação residencial chamado Ysto, este projeto se propõem a contemplar os itens acima citados de forma mais simples e a um uma fração do preço dos existentes na atualidade.

---

<sup>1</sup> <https://madeby.google.com/home/>

<sup>2</sup> <https://www.amazon.com/dp/B00X4WHP5E>

<sup>3</sup> <https://www.apple.com/homepod>

Fazer a aquisição de dados, transformá-los em informação útil e decidir o que fazer com esta informação disponibilizando uma interface simples e unificada será o escopo deste projeto, tudo isso a um custo mínimo. A figura 3 mostra uma estimativa de preço para montagem de uma central de controle e um módulo auxiliar.

Figura 3 – Levantamento aproximado de custos

<b>Custos aproximados do projeto</b>		
Item	Descrição	Valor
01	RaspberryPI	R\$30,00
01	Módulo ESP-01	R\$15,00
01	Opto acoplador	R\$3,00
02	Resistores R330 OHMs	R\$1,00
01	Caixa patola	R\$10,00
01	Case acrílico RaspberryPI	R\$20,00
01	Shield com relé	R\$15,00
01	Conversor AC-DC 5V	R\$15,00
01	Regulador DC 3V3	R\$5,00
Total		R\$114,00

Fonte: Autor do projeto

Estes valores foram coletados em sites como mercado livre e solda fria, são apenas estimativas e médias de preços, ou seja, podem variar. Uma alternativa muito interessante para aquisição dos componentes principais do sistema, a RaspberryPI e o ESP-01 são os sites chineses que disponibilizam estes componentes a preços muito mais atrativos. Apenas como efeito de ilustração, o site Aliexpress vende conjuntos de cinco ESP-01 a R\$32,00 é possível optar por um módulo pronto como o da figura 4.

Figura 4 – Alternativa pronta para o MA



Fonte: Website da Aliexpress

Este módulo reduziria nosso custo para aproximadamente R\$60,00 e teríamos algo muito prático e de fácil utilização.

## 3 OBJETIVOS

Esta seção apresenta os objetivos do trabalho, divididos em objetivo geral e objetivos específicos.

### 3.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um sistema que viabilize a automação de atividades em uma casa, a partir de uma interface simples de controle e acionamento de dispositivos, este controle remoto universal é acessado através do celular de qualquer um dos moradores.

### 3.2 Objetivos específicos

Para atingir o objetivo geral deste projeto, será necessário atingir os seguintes objetivos específicos:

- a) Implementar um aplicativo que possa ser acessado via smartphone, Android e IOS, para controlar os dispositivos;
- b) Elaborar um controle de acesso através deste aplicativo para impedir que ações indesejadas ocorram;
- c) Elaborar uma interface de controle que possa ser conectado a eletrodomésticos comuns permitindo controlá-los de forma remota.

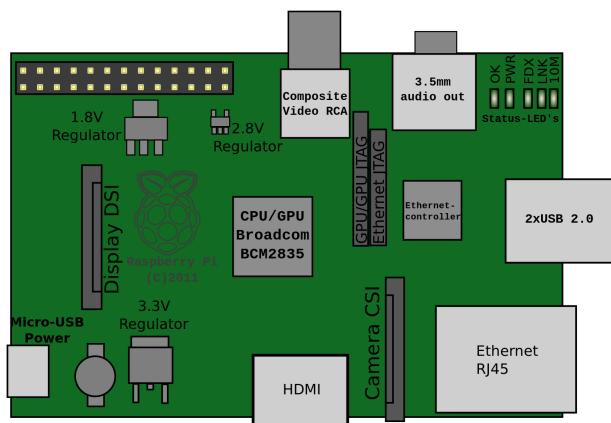
## 4 ANÁLISE DE TECNOLOGIAS E FERRAMENTAS

No desenvolvimento deste trabalho foram usados as seguintes ferramentas e tecnologias.

### 4.1 RaspberryPI

O RaspberryPI é um computador com dimensões reduzidas, do tamanho de um cartão de crédito, que disponibiliza diversos IO para integração. Escolhido por oferecer um bom poder de processamento (que impacta no tempo de resposta dos acionamentos), apresentar um baixo custo e de fácil aquisição no mercado local. É a central de controle do projeto, onde ficam hospedados os serviços e os aplicativos deste projeto, a figura 5 mostra uma placa de forma esquemática. (RASPBERRYPI, 2017)

Figura 5 – Diagrama esquemático de uma RaspberryPI

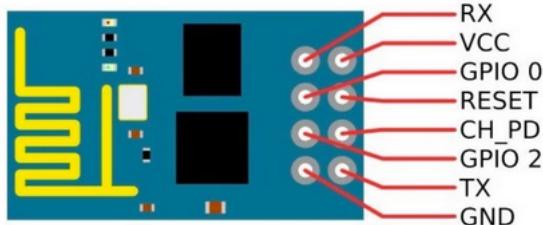


Fonte: Wikipedia

### 4.2 ESP8266

O ESP8266 é um System-On-Chip (SOC) com WIFI embutido, é produzido pela empresa Espressif, disponibiliza diversos IO e é compatível com o *framework* do Arduíno. (Espressif, 2017). Neste projeto, funciona como um acoplamento a eletrodomésticos existentes, trazendo um interface de comunicação WIFI para dispositivos que não a possuem, é o chamado Módulo Auxiliar (MA), a figura 6 mostra o modelo 01 desta placa e sua pinagem. (EPRESSIF, 2017)

Figura 6 – Diagrama esquemático de uma ESP8266-01



Fonte: Autor do projeto

### 4.3 Arduíno

Segundo definição do site Embarcados:

Arduíno é uma plataforma de código aberto (hardware e software) criada em 2005 pelo italiano Massimo Banzi (e outros colaboradores) para auxiliar no ensino de eletrônica para estudantes de design e artistas. O objetivo principal foi o de criar uma plataforma de baixo custo, para que os estudantes pudessem desenvolver seus protótipos com o menor custo possível. Outro ponto interessante do projeto, foi a proposta de criar uma plataforma de código aberto, disponível para a comunidade o que ajudou em muito no seu desenvolvimento. (EMBARCADOS, 2017)

Foi a ferramenta escolhida para o desenvolvimento do *firmware* do Módulo Auxiliar (MA) em função da facilidade de uso e por oferecer um grande número de bibliotecas para os mais diversos hardwares de mercado. (ARDUINO, 2017)

### 4.4 Javascript

Javascript é a principal linguagem de programação para desenvolvimento de aplicativos para browsers de internet. É utilizada para o desenvolvimento da interface com usuário final.

### 4.5 Python

Segundo o site do projeto:

Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. (FOUNDATION, 2017)

Foi a alternativa encontrada, quando dos problemas de performance na utilização de NodeJS (NODEJS, 2017). Em um período curto de tempo, quatro dias, toda API para controlar os módulos auxiliares foi reescrita.

## 4.6 SQLite

A wikipedia faz uma tradução do site do projeto, segundo eles:

SQLite é uma biblioteca em linguagem C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo Sistema de Gerenciamento de Banco Dados (SGBD) separado. (SQLITE, 2017)

Foi a opção escolhida devido a performance e facilidade de implementação para ambientes com Linux embarcado.

## 4.7 MQTT

MQTT é um protocolo de comunicação do tipo Machine to Machine (M2M) desenvolvido pela IBM no inicio dos anos 2000 como solução de comunicação entre hardwares de baixa capacidade de computação de dados com equipamentos mais robustos. Este protocolo serve de "cola" entre todos os entes de um sistema heterogêneo, onde servidores, computadores pessoais, smartfones e microcontroladores precisam conversar. Neste projeto é utilizado como o meio de comunicação entre todos os MAs e a central de controle. (IBM, 2017)

## 4.8 Mosquitto

Mosquitto é um Broker, ou central de controle de mensagens, que utiliza o protocolo MQTT para troca de mensagens em dispositivos diversos sobre o protocolo TCP. Está instalado e configurado como serviço na RaspberryPI portanto, fazendo parte da central de controle.(MOSQUITTO, 2017)

## 4.9 Debian Linux

Debian Linux é uma das primeiras distribuições Linux 1993, conhecido por seu rigoroso critério na aprovação de pacotes e na consequente robustez do sistema. Oferece uma licença para uso pessoal ou comercial que não onera o projeto. É o sistema operacional da central de controle. (DEBIAN, 1993)

## 4.10 Geany IDE

Segundo o site do projeto, Geany é um editor de textos que usa o kit gráfico GTK+ com características básicas de uma *Integrated Development Environment* (IDE), tradução do autor. Todo o código escrito neste projeto usou esta IDE. (GEANY, 2017)

#### 4.11 Schemacrawler

Segundo o site do projeto, Schemacrawler é uma ferramenta de descoberta e compreensão de esquemas de banco de dados, tradução do autor. Esta ferramenta foi utilizada para a partir do banco de dados gerado pelo módulo de *Object-relational mapping* (ORM), mapear as tabelas e relacionamentos, traduzindo para um diagrama Entidade Relacionamento. (SUALEH, 2017)

#### 4.12 Git SCM

Segundo o site do projeto, Git é um sistema de controle distribuído de código fonte aberto, projetado para lidar com tudo, de pequenos a grandes projetos com velocidade e eficiência, tradução do autor. Todo material deste projeto incluindo este relatório estão versionados e controlados com o Git. (GIT, 2017)

#### 4.13 Gliffy

Gliffy é uma extensão do google chrome para criação e edição de diagramas, usada neste projeto para elaboração dos diagramas de BPM e fluxo de dados. (GLIFFY, 2017)

#### 4.14 VueJS

Vuejs é um framework *open-source* utilizado para desenvolvimento de interfaces gráficas para ambientes web. Foi escolhido para este projeto devido a simplicidade e facilidade de uso, envolve um *setup* de ambiente de desenvolvimento que requer poucos recursos(é possível iniciar o desenvolvimento sem o uso de NodeJS e sua coleção de ferramentas). Separa o tratamento de dados da visualização destes o que facilita a modularização do projeto. (VUEJS, 2017)

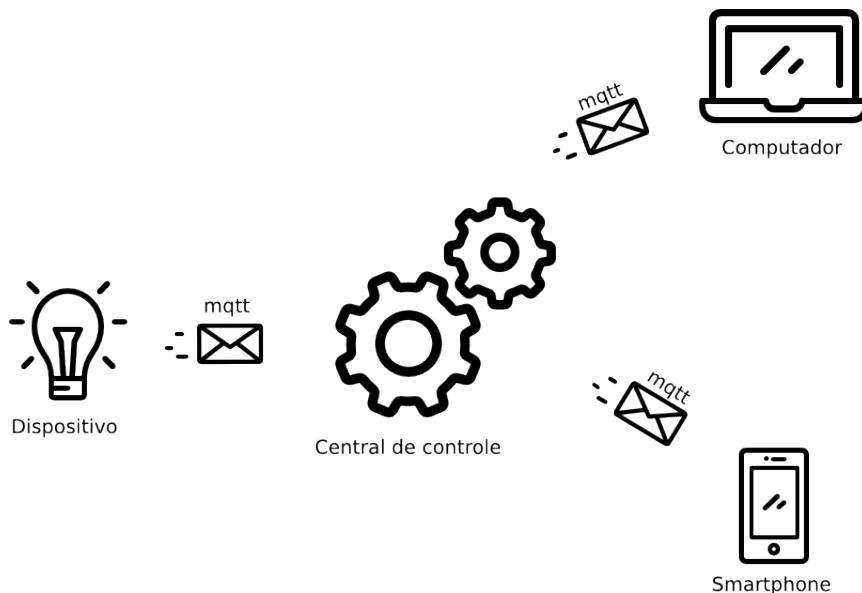
#### 4.15 Skeleton

Skeleton é um *template* para desenvolvimento de interfaces web, oferece uma estrutura mínima de html e css. Foi a opções escolhida em função do tamanho reduzido, da simplicidade para iniciar o desenvolvimento e rapidez em obter resultados. (SKELETON, 2017)

## 5 DESCRIÇÃO DA SOLUÇÃO

Este projeto possui duas grandes partes, os Módulos Auxiliares (MA) e a central de controle. A figura 7 mostra um diagrama geral de comunicação de um dispositivo acoplado a um Módulo Auxiliar (MA) com a central de controle.

Figura 7 – Topologia da solução



Fonte: Autor do projeto

O principal diferencial deste projeto é a API de controle dos dispositivos, esta API segue o padrão REST de implementação o que permite que qualquer cliente que tenha a capacidade de integrar-se a APIs deste tipo possam fazer uso dela.

### 5.1 Módulos Auxiliares

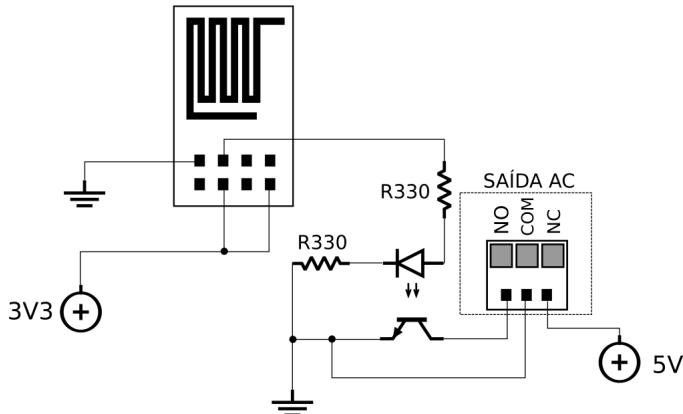
Estes módulos são acoplados a eletrodomésticos e são compostos por uma placa ESP8266 modelo ESP-01, todo processo de desenvolvimento do *firmware* até a gravação deste *firrmware* na placa ESP-01 é feito utilizando a IDE do Arduíno. Também são usados bibliotecas do framework Arduíno para comunicação através do protocolo MQTT com a central de controle.

#### 5.1.1 Esquema eletrônico do MA

Foi desenvolvido um circuito para acoplar a placa ESP-01 a um relé, este é responsável por fazer o acionamento de cargas AC (Corrente Alternada, tradução do inglês).

A figura 8 mostra este circuito de forma esquemática.

Figura 8 – Esquemático eletrônico do MA



Fonte: Autor do projeto

### 5.1.2 Programação do *firmware* do MA

No processo de desenvolvimento do *firmware* do MA foi utilizado a IDE Arduíno, a figura 9 mostra com um trecho do firmware de controle do MA.

Figura 9 – Arduíno IDE

```

app-modulo.ino | Arduino 1.8.0
Arquivo Editar Sketch Ferramentas Ajuda
app-modulo.ino
#include "ESP8266.h"

#define TOPIC "MA003/relay"
#define STATUS_TOPIC "/status"
#define RELAY_PIN 3

char* relayTopic = TOPIC;
char* statusTopic = STATUS;

const int relayPin = RELAY_PIN;

netInfo homeNet = {
    .mqttHost = "192.168.10.101",
    .ssid = "mozelino",
    .pass = "costinhos",
    .mqttPort = 1883
};

ESPHelper myESP(homeNet);

void setup() {
    //setup the rest of ESPHelper
    myESP.addSubscription(relayTopic);
    myESP.begin();
    myESP.setCallback(callback);
    pinMode(relayPin, OUTPUT);
    delay(100);
}

void loop(){
    //loop ESPHelper and wait for commands from mqtt
    myESP.loop();
    yield();
}

```

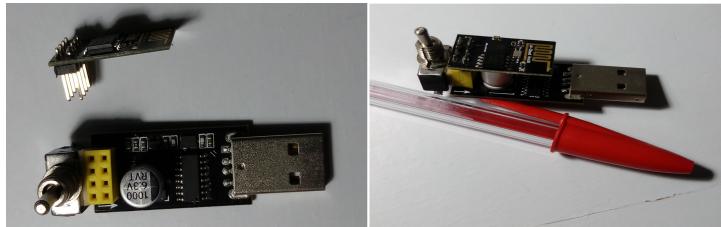
Salvo.

Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFFS), ck\_Disabled, None err/dev/ttyUSB0

Fonte: Autor do projeto

O processo de gravação do *firmware* na placa ESP-01 utiliza um gravador que é acoplado a uma das portas USB do computador, a figura 10 mostra como é este gravador e a montagem do módulo para gravação de *firmware*.

Figura 10 – Gravação do *firmware* no MA



Fonte: Autor do projeto

## 5.2 Central de controle

A central de controle é um micro computador com dimensões reduzidas, o modelo é uma RaspberryPI tipo B com 512MB de RAM e um cartão de memória do tipo SD card com 8GB de memória disponível. Neste computador estão instalados os seguintes programas:

- a) Sistema operacional Debian na versão 8;
- b) Broker MQTT Mosquitto versão 1.4.14;
- c) NodeJS versão 8.2.1;
- d) Sqlite3 versão 3.8.7.

Na figura 11 é possível ver o computador montado em uma caixa de acrílico.

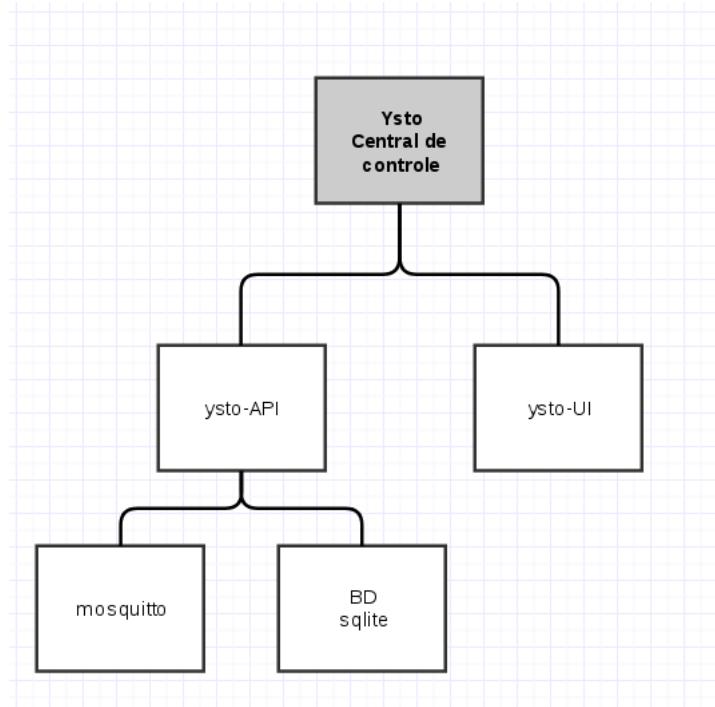
Figura 11 – Central de controle



Fonte: Autor do projeto

Na figura 12 mostra a coleção de serviços e aplicativos que estão hospedados na central de controle.

Figura 12 – Esquemático de serviços da central de controle



Fonte: Autor do projeto

A troca de mensagens será feita utilizando o protocolo chamado Telemetry Transport Message Queue (MQTT), este protocolo além de possuir um baixo consumo de banda (as mensagens são relativamente pequenas em comparação com outros protocolos que trafegam no mesmo meio, http por exemplo) ainda fornece ferramentas necessárias para:

- Fazer o controle de quem pode trocar mensagens na nossa rede de dispositivos;
- Garantir a entrega das mensagens;
- Integração com aplicativos de mercado em diversas plataformas mobile ou *Desktop*.

Este Dashboard será um Appweb do tipo *Mobile First* ou seja, seus layouts serão adaptados para as telas de telefones, tablets e computadores.

### 5.3 Funcionamento do protocolo MQTT

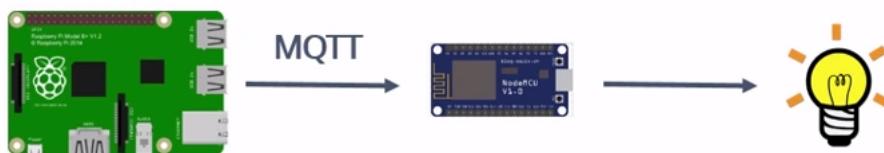
Este projeto baseia-se fortemente no funcionamento do protocolo MQTT (IBM, 2017), é importante entender como este funciona pois tem reflexo direto em como este

projeto opera. Este protocolo de comunicação permite estabelecer uma maneira simples de comunicação entre múltiplos dispositivos, podendo:

- Enviar um comando para uma saída;
- Ler uma entrada e publicar os dados lidos.

A figura 13 exemplifica o funcionamento do envio de um comando para uma saída do sistema.

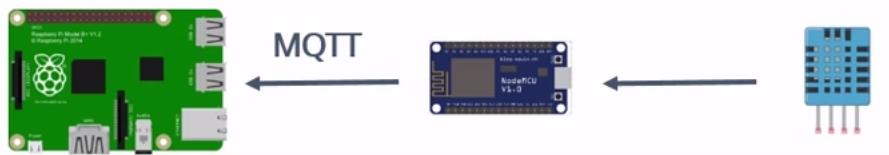
Figura 13 – Mensagem sendo enviada via MQTT



Fonte: Random Nerd Tutorials

Já figura 14 demonstra o recebimento de dados de um sensor de temperatura.

Figura 14 – Mensagem sendo recebida via MQTT



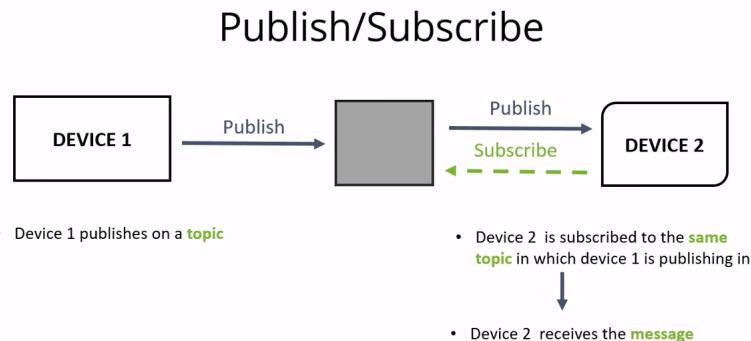
Fonte: Random Nerd Tutorials

Existem alguns conceitos básicos que envolvem o uso deste protocolo, são eles:

- Publish/Subscribe :** Um dispositivo pode publicar mensagens para outros dispositivos, ou um dispositivo pode ser inscrito em um tópico qualquer e passar a receber as mensagens deste tópico;
- Messages:** São as informações trocadas entre os dispositivos, podem ser comandos ou apenas informações;
- Topics:** É a maneira como um dispositivo demonstra interesse em determinadas mensagens, pode também ser definido como o lugar onde um dispositivo deseja publicar suas mensagens;
- Broker:** É o responsável por receber todas as mensagens, fazer o filtro e publicar nos respectivos tópicos.

A figura 15 demonstra de forma esquemática as diversas ações que ocorrem durante uma troca de mensagens entre dispositivos.

Figura 15 – Modelo publish/subscribe



Fonte: Random Nerd Tutorials

A figura 16 mostra como uma estrutura de tópicos pode ser montada para fazer o acionamento de um dispositivo, no caso uma lâmpada.

Figura 16 – Um exemplo de estrutura de tópicos



Fonte: Random Nerd Tutorials

A figura 17 mostra o papel do Broker dentro deste cenário.

Figura 17 – O papel do broker

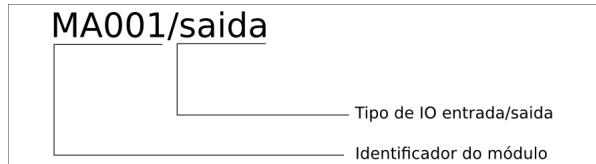


Fonte: Random Nerd Tutorials

### 5.3.1 Tópico para controle de um MA

De acordo com o descrito na seção de descrição da solução, este projeto está baseado no conceito *Publish/Subscribe* onde uma estrutura de tópicos é necessária para a execução de ações e coleta de dados, a figura 18 mostra como é nossa estrutura de tópicos.

Figura 18 – Estrutura de tópicos adotada neste projeto



Fonte: Random Nerd Tutorials

#### 5.4 Ponto de entrada no sistema

Para os usuários ou administradores do sistema, tudo será controlado por acesso a URL da rede local, no endereço <http://ysto.local>. Basta então, através do seu apontar para este endereço estando conectado a rede local do ysto.

## 6 ABORDAGEM DE DESENVOLVIMENTO

Este projeto terá diversas funcionalidades que se relacionam entre si, para organizar e sistematizar o processo de desenvolvimento foi usado uma adaptação do Scrum<sup>1</sup> o Scrum Solo<sup>2</sup>. O Scrum Solo é a combinação do framework Scrum Personal Software Process (PSP), onde:

O Scrum é um framework ágil para gerenciamento de projetos que se destaca por sua abordagem enxuta (lean) de desenvolvimento. Por ser um modelo iterativo e incremental, o Scrum divide o projeto em vários sprints (ciclos curtos de desenvolvimento) consecutivos que ocorrerão de acordo com a prioridade do product owner (proprietário do produto). Cada período de sprint é definido, geralmente, entre duas e quatro semanas. Durante esse tempo, o scrum team (analista e programadores) se dedica ao máximo para ter um pequeno conjunto de funcionalidades codificadas e testadas. (ScrumSolo, 2017) O PSP é um processo de melhoria projetado para ajudar os desenvolvedores a controlar, administrar e aperfeiçoar sua competência para produzir software de qualidade. O propósito do PSP é ajudar o desenvolvedor a melhorar a sua forma de trabalho, entendendo sua própria performance e sabendo onde e como melhorá-la. A filosofia por trás do PSP é que a competência de uma organização para construir softwares de determinado tamanho e grau de complexidade decorre, em parte, da habilidade individual de seus engenheiros. O PSP se baseia no princípio do conhecimento, avaliação e melhorias contínuas do processo individual. (SCRUMSOLO, 2017)

A figura 19 mostra como funciona os ciclos de iterações dentro do Framework Scrum Solo.

Figura 19 – Funcionamento do Scrum solo



Fonte: Website do Scrum Solo

As subseções a seguir apresentam a sequencia de etapas para facilitar o acompanhamento e a execução do projeto propostas pelo framework Scrum Solo.

<sup>1</sup> <https://www.scrum.org/>

<sup>2</sup> <https://engenhariasoftware.wordpress.com/2016/04/17/scrum-solo-2/>

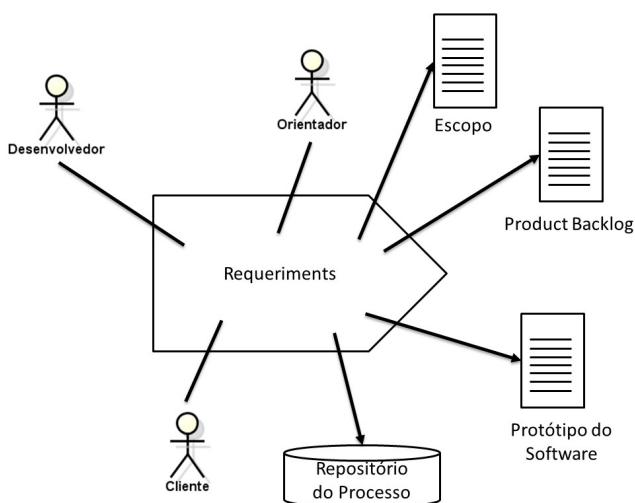
## 6.1 Levantamento de requisitos

Esta é a primeira etapa do framework, aqui é feito a descrição dos principais pontos do software aplicados a solução do problema, para isso os seguintes artefatos são gerados:

- a) Documento de escopo;
- b) Product Backlog;
- c) Protótipos de software.

A figura 20 mostra como estes artefatos estão relacionados com os atores nesta etapa.

Figura 20 – Levantamento de requisitos



Fonte: Website do Scrum Solo

## 6.2 Sprint

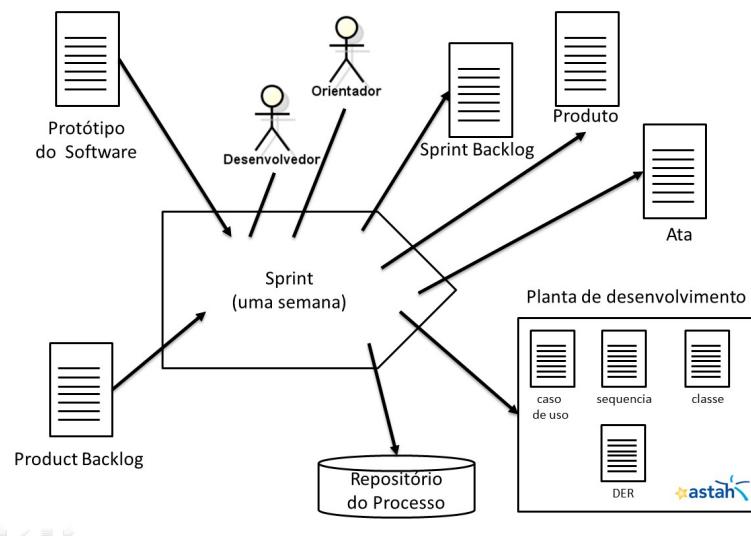
Esta etapa, que pode ocorrer N vezes dentro de um projeto, movimenta os artefatos anteriores e faz a adição dos seguintes:

- a) Sprint Backlog;
- b) Planta de desenvolvimento, este relatório;
- c) Ata de reunião, substituído por encontros regulares com o orientador e por análise deste relatório em documento de revisão;

- d) Produto parcial, com alguma funcionalidade pronta.

A figura 21 mostra como fica o relacionamento entre os atores e os referidos artefatos.

Figura 21 – Manipulação de artefatos na Sprint

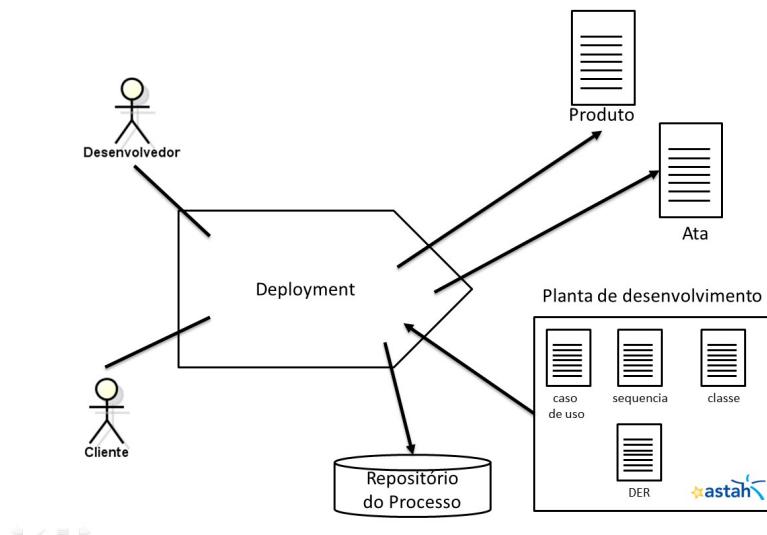


Fonte: Website do Scrum Solo

### 6.3 Entrega

Periodicamente é definido uma etapa de entrega, onde um conjunto de funcionalidades prontas e testadas é apresentado e entregue para o cliente, a figura 22 mostra como é o relacionamento dos atores com os artefatos nesta etapa.

Figura 22 – Entrega de software

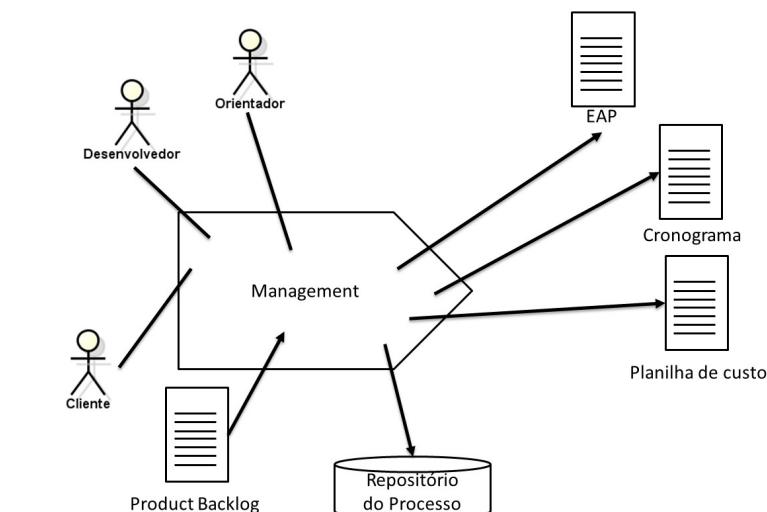


Fonte: Website do Scrum Solo

## 6.4 Gestão

Também de forma periódica, ocorrem reuniões de acompanhamento e gestão do projeto. Nesta etapa, são avaliados o progresso do projeto bem como eventuais mudanças de escopo, mudanças estas que podem alterar os artefatos elaborados até então. A figura 23 mostra como fica o relacionamento dos atores com os artefatos nesta etapa.

Figura 23 – Gestão de projeto



Fonte: Website do Scrum Solo

## 6.5 Repositório de artefatos

O Scrum Solo prevê a existência de um repositório na nuvem para todos os artefatos que compõem um projeto, neste projeto é usado o seguinte endereço Scrum-Repo<sup>3</sup>.

## 6.6 Scrum solo neste projeto

O desenvolvimento deste projeto, através da adaptação deste framework, mostrou-se muito produtivo. A oferta de documentos de modelo para dar início ao projeto propriamente dito, é muito prático e eficiente. O projeto de exemplo também ajuda muito na hora de fazer a estruturação do projeto e de definir uma maneira de interagir com o orientador. Utilizamos os seguintes artefatos nesta implementação:

- a) Documento de escopo;
- b) User stories;
- c) Product backlog;
- d) Sprint backlog;
- e) Retrospectiva de sprint;
- f) Calendário de reuniões com o orientador do projeto.

---

<sup>3</sup> <https://www.dropbox.com/sh/ku5z0fzbermoq49/AACzrYhZ4mBv89ZaJfMosRHKa?dl=0>

## 7 ARQUITETURA DO SISTEMA

A seguir os artefatos que serão gerados ao longo do desenvolvimento deste projeto e que servirão de suporte ao longo do processo de desenvolvimento.

### 7.1 Modelagem funcional

Nesta etapa são levantadas o Documento de Escopo, as e as Sprints Backlog.

#### 7.1.1 Documento de escopo

Este é o ponto de início para desenvolvimento do projeto, nele deve ser descrito em alto nível, que tipo de problema este projeto resolve. A figura 24 mostra este documento.

Figura 24 – Documento de escopo

#### **Escopo**

Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Data da criação	09 de Abril de 2017
Data da reunião	03 de Abril de 2017
Data de revisão	02 de Setembro de 2017

#### **Escopo do problema**

Integrar o controle dos eletrodomésticos de maneira simples a um custo baixo.

#### **Perfil do cliente**

Qualquer família que possua eletrodomésticos e rotinas diárias, que gostaria automatizar e tomar o processo de utilização e realização de tarefas mais fácil.

#### **Product backlog**

Itens identificados para compor a lista de funcionalidades:

- Capturar o acionamento de dispositivos;
- Processar essa captura para armazenamento;
- Permitir o acionamento de dispositivos de forma manual ou automática.

Fonte: Autor do projeto

#### 7.1.2 User Stories

Este artefato faz a coleta, em um nível abstrato e sem riqueza de detalhes, das funcionalidades que os entes envolvidos no uso do sistema desejam que exista. Recebem um atributo identificador e uma descrição resumida, a tabela 2.

Tabela 2 – User Stories

Id	Descrição
US001	Como usuário do sistema eu gostaria de ligar ou desligar dispositivos, uma lâmpada por exemplo, usando meu Smartphone.
US002	Como usuário do sistema eu gostaria de saber quantos e quais dispositivos fazem parte da minha rede.
US003	Como usuário do sistema eu gostaria de saber a situação destes dispositivos, estão conectados? Em que estado estão?
US004	Como usuário do sistema eu gostaria de acessar meu perfil e verificar as ações que realizei em um período específico de tempo.
US005	Como administrador do sistema eu gostaria de cadastrar outros usuários para a utilização do sistema.
US006	Como usuário do sistema eu gostaria de receber notificações no meu Smartphone sobre ações que estão ocorrendo em minha casa, mesmo estando fora dela.

Fonte: Produzido pelo autor

A tabela 3 mostra a listagem de User Stories, ordenada por prioridade.

Tabela 3 – Product Backlog

Id	Descrição
US001	Controlar o acionamento de dispositivos.
US002	Listagem de dispositivos.
US003	Status de dispositivos.
US004	Histórico de ações de usuários.
US005	CRUD de usuários.
US006	Notificações via Smartphone.

Fonte: Produzido pelo autor

### 7.1.3 Sprints

Aqui são listadas as sprints realizadas neste projeto, juntamente com suas revisões. Revisões estas que descrevem fatos ocorridos durante o desenvolvimento do projeto.

**Sprint 1** A Sprint 1 iniciou com o desenvolvimento da US001, esta foi subdividida em tarefas mais detalhadas conforme a figura 25.

Figura 25 – Detalhamento sprint 1

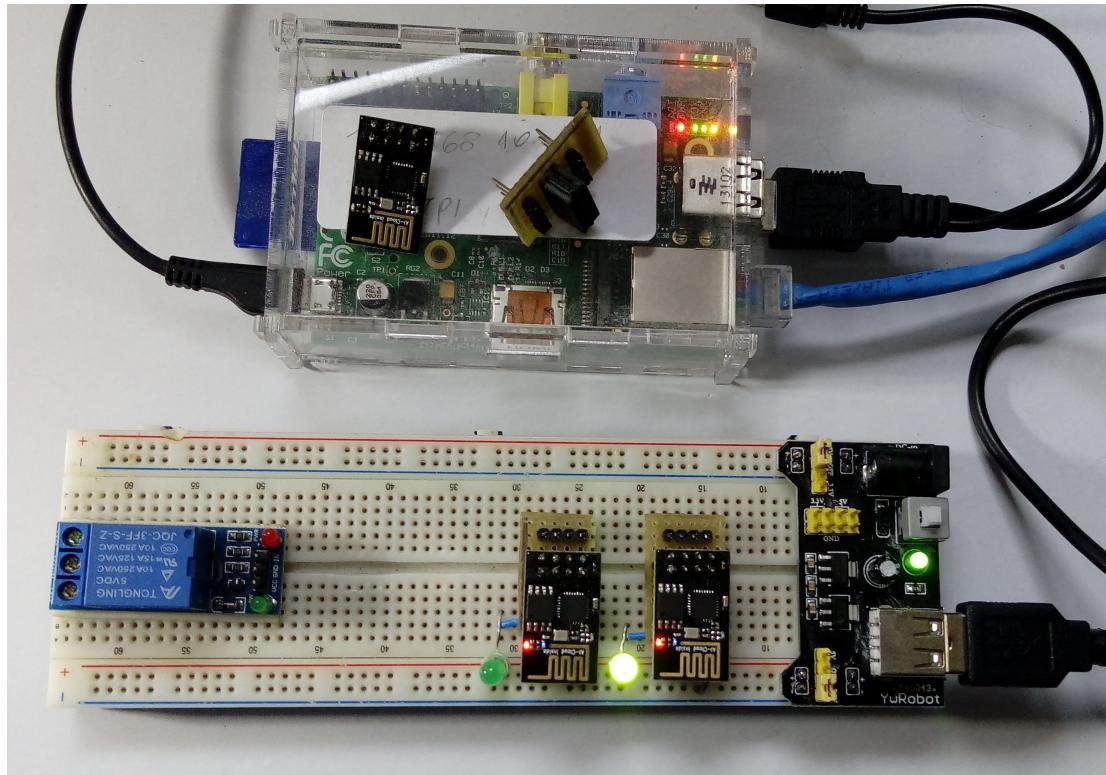
Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Período	10/04 à 23/04/2017
Identificação da Sprint	Controlar o acionamento de dispositivos (US001)

ID	Descrição	Data de execução	Tempo (minutos)
1	Montagem dos módulos para captura de dados	14 à 16/04	480
2	Gravação do interpretador JS no módulo	21 à 23/04	120

Fonte: Autor do projeto

**Retrospectiva da Sprint 1** Realizado a montagem de placas auxiliares para fixação no protoboard , posteriormente esse circuito é utilizado para demonstrar o funcionamento dos módulos que coletam as informações e fazem os acionamentos. A figura 26 mostra a etapa final de montagem.

Figura 26 – Montagem das placas de fixação dos módulos



Fonte: Autor do projeto

Na realização desta etapa, o modelo escolhido apresentou problemas no uso do

interpretador de comando JavaScript, o Espruino(Espruino, 2017). Existe um Bug que não permite a gravação de informações na região não volátil de memória do módulo, isso inviabiliza temporariamente a utilização desta versão do módulo ESP8266. A figura 27 mostra como é o referido módulo.

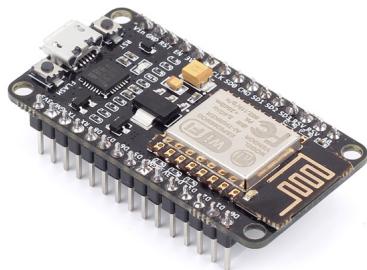
Figura 27 – Módulo ESP8266 tipo 01



Fonte: Website da Espressif

Este Bug foi relatado aos autores do projeto e após a correção é possível retomar o uso desta versão de hardware. Para dar continuidade ao projeto será utilizado uma variação de placa da família ESP8266, a tipo 12 ou também conhecida como Node MCU. A figura 28 mostra como é esta placa.

Figura 28 – Módulo ESP8266 tipo 12



Fonte: Website da Espressif

Este modelo não apresentou os problemas descritos anteriormente, nele temos mais capacidade de armazenamento e um número maior de General Purpose Input/Output.

**Sprint 2** A Sprint 2 deu seguimento as atividades da US001, novas tarefas foram identificadas dentro deste contexto e adicionadas, conforme mostra a figura 29.

Figura 29 – Detalhamento sprint 2

Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Período	24/04 à 05/05/2017
Identificação da Sprint	Controlar o acionamento de dispositivos (US001)

ID	Descrição	Data de execução	Tempo (minutos)
1	Comunicação do módulo auxiliar com o servidor onde está o Ysto	29/04/17	60
2	Implementação do protocolo MQTT no módulo auxiliar	30/04/17	60
3	Teste de comunicação usando o protocolo MQTT entre o módulo e servidor	05/05/17	180

Fonte: Autor do projeto

**Retrospectiva da Sprint 2** Realizado um teste com uma biblioteca disponibilizada para o framework Arduíno (Arduino, 2017), a ESPHelper (ESPHelper, 2017), com ela foi possível a utilização da placa ESP-01, mantendo as dimensões reduzidas do nosso módulo de acesso e captura de informações do mundo físico. Para a realização dos testes de comunicação com o servidor, já utilizando o lugar final onde este ficará armazenado, foi utilizado o cliente para o protocolo MQTT chamado Mosquitto (Mosquitto, 2017).

**Sprint 3** Esta Sprint deve ser o encerramento do Firmware conforme a figura 30 contempla as seguintes atividades.

Figura 30 – Detalhamento sprint 3

Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Período	24/04 à 05/05/2017
Identificação da Sprint	Controlar o acionamento de dispositivos (US001)

ID	Descrição	Data de execução	Tempo (minutos)
1	Comunicação do módulo auxiliar com o servidor onde está o Ysto	29/04/17	60
2	Implementação do protocolo MQTT no módulo auxiliar	30/04/17	60
3	Teste de comunicação usando o protocolo MQTT entre o módulo e servidor	05/05/17	180

Fonte: Autor do projeto

**Retrospectiva da Sprint 3** Realizado a implementação de acionamento do módulo auxiliar.

**Sprint 4** A figura 31 mostra o planejamento da sprint 4.

Figura 31 – Detalhamento sprint 4

Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Período	22/05 à 02/06/2017
Identificação da Sprint	Controlar o acionamento de dispositivos (US001)

ID	Descrição	Data de execução	Tempo (minutos)
1	Integração fw dos módulos com o broker (US001)	27/05/17	300
2	Integração do serviço local com um serviço de controle na nuvem, CloudMQTT (US001)	03/06/17	180

Fonte: Autor do projeto

**Retrospectiva da Sprint 4** Nesta Sprint foi fixado uma estrutura de tópicos para controlar os Módulos Auxiliares (MA), também foi feita a ligação do serviço Mosquitto instalado na RaspberryPI com um serviço que roda na nuvem chamado CloudMQTT<sup>1</sup>, isso

<sup>1</sup> <https://www.cloudmqtt.com/>

permite controlar os módulo auxiliares em qualquer lugar onde existe acesso a internet.

**Sprint 5** A figura 32 mostra o planejamento da sprint 5.

Figura 32 – Detalhamento sprint 5

Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Período	04/09 à 15/09/2017
Identificação da Sprint	Implementação da API (US002, US003, US005)

ID	Descrição	Data de execução	Tempo (minutos)
1	Cadastro de usuários	04/09 à 08/09	570
2	Cadastro de dispositivos	11/09 à 15/09	570

Fonte: Autor do projeto

**Retrospectiva da Sprint 5** Nesta sprint ocorreu a implementação das US002, US003 e US005, em conjunto a esse desenvolvimento, testes unitários foram executados, a figura 33 mostra o relatório de testes realizados até o momento.

Figura 33 – Testes unitários da API

```
ssh /home/fabiano
root@ysto:~/ysto/api.git# npm test
> ysto-api@1.0.0 test /root/ysto/api.git
> NODE_ENV=test mocha test/**/*.js

Routes: Devices
  GET /devices
    status 200
    ✓ returns a list of devices
  POST /devices
    status 200
    ✓ creates a new device
  GET /devices/:id
    status 200
    ✓ returns one device
    status 404
    ✓ throws error when device not exist
  PUT /devices/:id
    status 204
    ✓ updates a device
  DELETE /devices/:id
    status 204
    ✓ removes a device

Routes: Index
  GET /
    ✓ returns the API status

Routes: Token
  POST /token
    status 200
    ✓ returns authenticated user token
    status 401
    ✓ throws error when password is incorrect
    ✓ throws error when email not exist
    ✓ throws error when email and password are blank

Routes: Users
  GET /user
    status 200
    ✓ returns an authenticated user
  DELETE /user
    status 204
    ✓ deletes an authenticated user
  POST /users
    status 200
    ✓ creates a new user

14 passing (20s)
root@ysto:~/ysto/api.git# █
```

Fonte: Autor do projeto

**Sprint 6** A figura 34 mostra o planejamento da sprint 6.

Figura 34 – Detalhamento sprint 6

Nome do projeto	Ysto Automação Residencial
Nome do desenvolvedor	Fabiano da Rosa Gomes
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas
Período	25/09 à 06/10/2017
Identificação da Sprint	Implementação da API (US004)

ID	Descrição	Data de execução	Tempo (minutos)
1	Token e ações de controle	25/09 à 06/10	1920

Fonte: Autor do projeto

**Retrospectiva da Sprint 6** Finalizamos a sprint 6 dentro do cronograma definido, todos os *end-points* foram testados de forma automática.

**Cancelamento da Sprint 7** Na sprint 7 estava previsto a implementação da Interface com o usuário (UI), mas no decorrer de uso da API desenvolvida na sprint 6 problemas de performance foram percebidos, durante o período de pesquisa para entender o que estava acontecendo, o seguinte levantamento foi realizado:

- a ) Tempo de inicio de testes automáticos: 10 – 15 segundos;
- b ) Tempo de resposta para um acionamento de módulo auxiliar: 5 – 8 segundos;
- c ) Quantidade de arquivos necessários para o funcionamento da API (módulos do NodeJS); 10.537 entre módulos e aplicativo;
- d ) Quantidade de espaço ocupado no cartão SD pela API, em bytes: 64MB de arquivos

Essa parece ser uma característica do NodeJS, que tem como lema "Baterias não inclusas". Essa política tem seus prós e contras, depois de muito pesquisar pareceu ser a melhor alternativa trocar a tecnologia de desenvolvimento da API. Python foi escolhido, justamente por fazer uma proposta do tipo "Baterias inclusas", o que impacta na quantidade de bibliotecas externas que é necessária para fazer uma API RESt funcionar. Com a refatoração em Python a API reduziu para um tamanho de 200KB em arquivos, a figura 35 mostra como a árvore da aplicação ficou estruturada.

Figura 35 – Estrutura do projeto

```

fish /home/fabiano/projects/ysto/vuejs-teste
ssh /home/fabiano/projects          x   fish /home/fabiano/projects/ysto/vuejs-teste
fabiano@my-pc ~/p/y/vuejs-teste> tree -L 4 -I '*.pyc'
.
├── api.py
├── LICENSE
├── model
│   ├── device.py
│   ├── __init__.py
│   └── user.py
├── README.md
├── ssl
│   ├── __init__.py
│   ├── server.pem
│   └── SslServer.py
└── test_api.py

	ui
	├── static
	│   ├── css
	│   │   ├── normalize.css
	│   │   └── skeleton.css
	│   ├── img
	│   │   └── favicon.png
	│   ├── lib
	│   │   └── vue.js
	│   └── main.js
	└── views
	    └── index.html

ysto.db

8 directories, 17 files
fabiano@my-pc ~/p/y/vuejs-teste>

```

Fonte: Autor do projeto

Além dos números e a simplicidade da estrutura do projeto, agora o acionamento de dispositivos é praticamente instantâneo (algo em torno de UM segundo entre a seleção da ação e o acionamento).

**Cancelamento da Sprint 8** Na sprint 8 estava previsto a finalização da UI, este trabalho foi iniciado, mas nos primeiros testes de integração com API foi percebido um problema de acesso aos recursos da API. A UI era um aplicativo que estava exposto na porta 3002 da central de controle e a API estava na porta 3001. Tanto o aplicativo cliente, no caso um browser quanto o servidor implementam políticas de segurança para evitar que ataques maliciosos possam comprometer o funcionamento de aplicativos pra internet, esta proteção baseia-se no conceito conhecido como *Cross-Origin Resource Sharing* (CORS).

Diversas tentativas foram feitas para fazer com que a API se tornasse disponível pra UI, dentre elas:

- a ) Do lado da UI envio de cabeçalhos com filtro aberto para domínios: Access-Control-Allow-Origin: \*
- b ) Do lado do servidor, permitir esta troca entre aplicativos do mesmo domínio;
- c ) Instalação de servidores utilizados em aplicativos comerciais como o nginx e o lighttpd que possuem receitas para liberação do CORS;

d ) Instalação de extensões no browser para fazer a liberação de troca de mensagens entre aplicativos que ficam sob o mesmo domínio.

O fato é que nada disso funcionou, a opção final foi mesclar os dois aplicativos, desta forma a API e a UI agora são um projeto único e está disponível na porta 3001 da central de controle. A UI está na raiz do domínio <IP>/ e a API recebeu o seguinte endereço <IP>/api.

**Sprint 7** A figura 36 mostra o planejamento da sprint 7.

Figura 36 – Detalhamento sprint 7

Nome do projeto	Ysto Automação Residencial		
Nome do desenvolvedor	Fabiano da Rosa Gomes		
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas		
Período	13/11 à 17/11/2017		
Identificação da Sprint	Implementação da UI (US006)		
ID	Descrição	Data de execução	Tempo (minutos)
1	UI mobile first	13/11 à 17/11	1440

Fonte: Autor do projeto

**Retrospectiva da Sprint 7** Implementação realizada sem problemas, utilizado o template Skeleton (SKELETON, 2017) para a criação da UI.

**Sprint 8** A figura 37 mostra o planejamento da sprint 8.

Figura 37 – Detalhamento sprint 8

Nome do projeto	Ysto Automação Residencial		
Nome do desenvolvedor	Fabiano da Rosa Gomes		
Nome do curso	Tecnólogo em análise e desenvolvimento de sistemas		
Período	20/11 à 24/11/2017		
Identificação da Sprint	Implementação da UI (US006)		
ID	Descrição	Data de execução	Tempo (minutos)
1	Controle de eventos	20/11 à 24/11	1440

Fonte: Autor do projeto

**Retrospectiva da Sprint 8** TODO: Na data da impressão do relatório não havia executado esta sprint ainda.

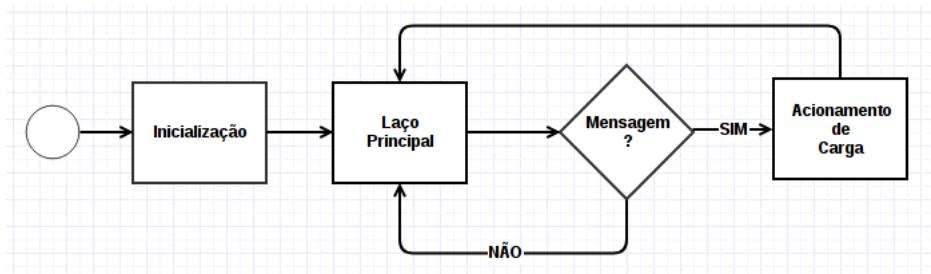
## 7.2 Modelagem do processo de negócio

Diagramas de fluxo de dados, BPM, e Diagramas de sequência que descrevem o funcionamento do projeto.

### 7.2.1 Firmware do módulo auxiliar

O módulo auxiliar é o responsável pela interação do sistema com o mundo físico, cabe a ele fazer a leitura de input e o acionamento do output. O firmware responsável por fazer estas leituras e escritas está dividido em duas partes, a figura 38 mostra como é feito o acionamento de uma carga.

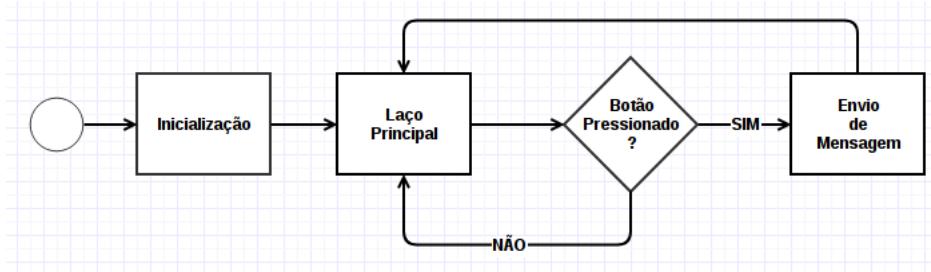
Figura 38 – Acionamento de carga



Fonte: Autor do projeto

Já a figura 39 mostra o que ocorre quando o módulo faz uma leitura.

Figura 39 – Leitura de dados



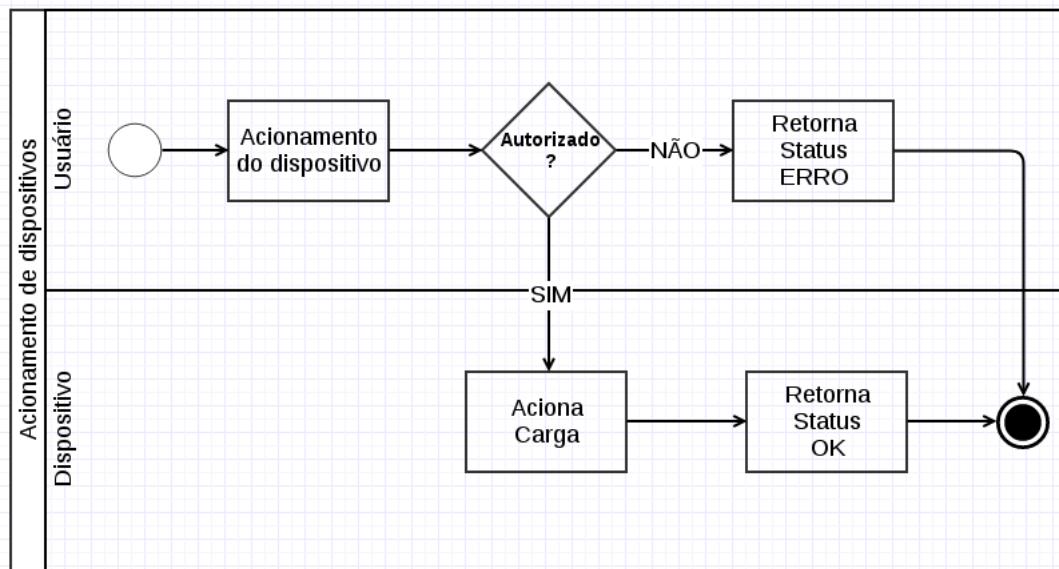
Fonte: Autor do projeto

Para o desenvolvimento deste firmware foi definido que o pino 2 do ESP-01 será o responsável por acionar cargas e o pino 3 será o responsável pela leitura de dados.

### 7.2.2 Controle de dispositivos através da API

O controle dos dispositivos é feito através da API de controle, a figura 40 mostra uma sequência de acionamento de dispositivo.

Figura 40 – Acionamento através da API



Fonte: Autor do projeto

## 7.3 Modelagem de dados

Diagrama Entidade-Relacionamento e modelo conceitual.

### 7.3.1 Modelagem da API

Ysto usa uma Application Program Interface (API) que utiliza o padrão REST, segundo (Saudate, 2016, p.4):

REST significa REpresentational State Transfer (ou Transferência de Estado Representativo, em tradução livre), e é um estilo de desenvolvimento de web services que teve origem na tese de doutorado de Roy Fielding (2000). Este, por sua vez, é coautor de um dos protocolos mais usados no mundo, o HTTP (HyperText Transfer Protocol). Assim, é notável que o protocolo REST é guiado (dentre outros preceitos) pelo que seriam as boas práticas de uso de HTTP:

- Uso adequado dos métodos HTTP;
- Uso adequado de URLs;
- Uso de códigos de status padronizados representação de sucessos ou falhas;
- Uso adequado de cabeçalhos HTTP e Interligações entre vários recursos diferentes.

O desenvolvimento desta API permite um maior desacoplamento de funções do sistema com a interface de uso.

### 7.3.2 Definição de recursos

Recursos são o ponto central de qualquer API REST, eles "são o conjunto de dados que trafegam pelo protocolo"(Saudate, 2016, p.5). Os chamados "verbos" do HTTP são utilizados como um padronizador de ações, sendo este procedimento uma excelente simplificação para ações repetitivas dentro de um sistema. As figuras que seguem, fazem justamente a ligação entre estes "verbos", POST, GET, PUT e DELETE com as ações que eles representam. A figura 41 mostra como o recurso users é tratados dentro da API.

Figura 41 – Recurso users na API

<b>Recurso</b>	<b>POST create</b>	<b>GET read</b>	<b>PUT update</b>	<b>Delete delete</b>
/users	Cria um novo usuário	Lista usuários	Atualiza todos os usuários	Apaga todos os usuários
/users/123	Trata como uma coleção e cria um novo usuário nela	Mostra os dados do usuário com id=123	Se existe atualiza o id=123 Se não cria um novo usuário	Remove o usuário id=123

Fonte: Autor do projeto

A figura 42 mostra como o recurso devices é tratado dentro da API.

Figura 42 – Recurso devices na API

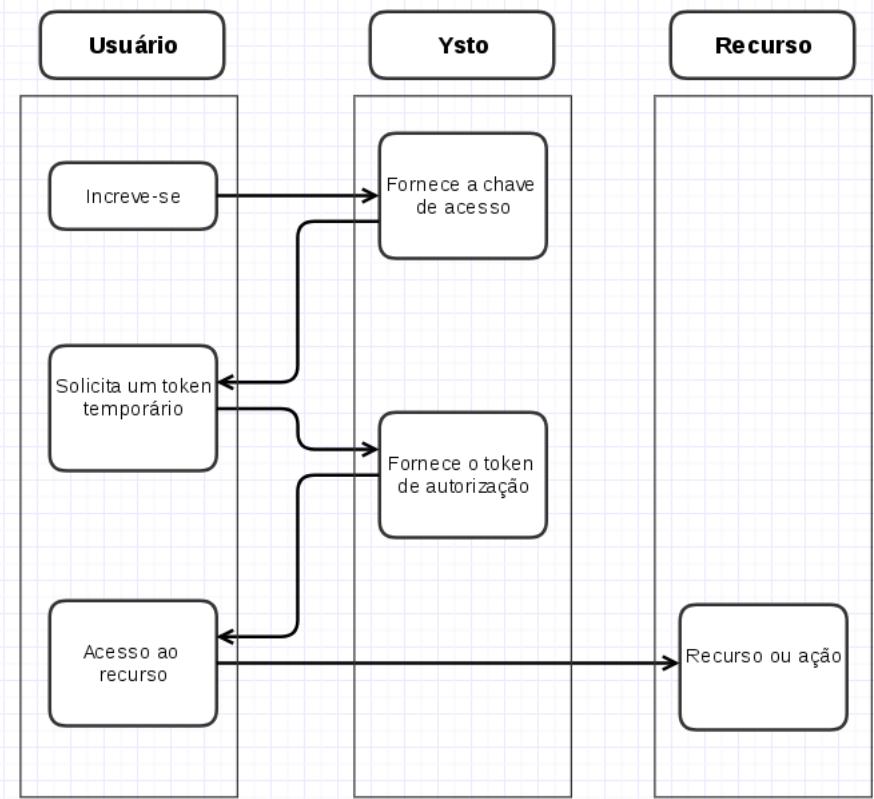
<b>Recurso</b>	<b>POST create</b>	<b>GET read</b>	<b>PUT update</b>	<b>Delete delete</b>
/devices	Cria um novo dispositivo	Lista dispositivos	Atualiza todos os dispositivos	Apaga todos os dispositivos
/devices/123	Trata como uma coleção e cria um novo dispositivo nela	Mostra os dados do dispositivo com id=123	Se existe atualiza o id=123 Se não cria um novo dispositivo	Remove o dispositivo id=123

Fonte: Autor do projeto

### 7.3.3 Controle de acesso

Todas as ações e recursos da API estão protegidas, o sistema de proteção para o uso deste sistema foi da adoção de tokens de segurança. Neste modelo de segurança, apenas mensagens devidamente assinadas chegam ao destino correto, executando a ação desejada. A figura 43 mostra como essa autenticação ocorre e como o usuário chega ao recurso desejado.

Figura 43 – Controle de acesso da API

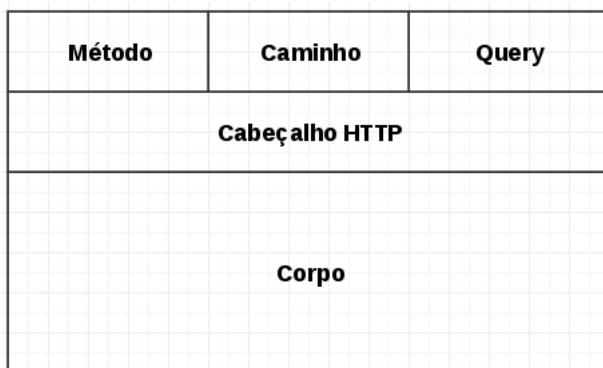


Fonte: Autor do projeto

### 7.3.4 Estrutura de mensagens

No momento da solicitação de um recurso ou ação, temos um HTPP, a figura 44 mostra a estrutura desta mensagem.

Figura 44 – Solicitação ou request



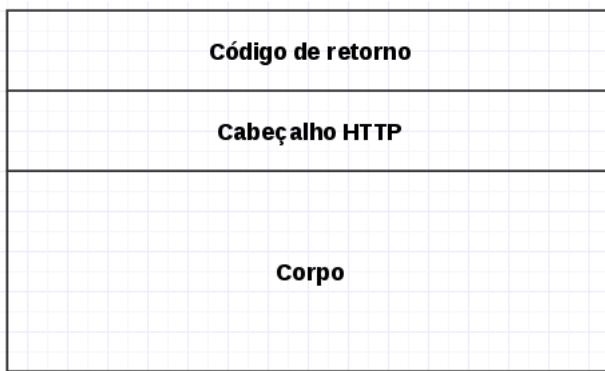
Fonte: Autor do projeto

Onde, cada campo poderia ser exemplificado da seguinte forma:

- a) Método: GET, POST, PUT ou DELETE;
- b) Caminho: /api/v1/home/devices
- c) Query: ?search&devices&id=123
- d) Cabeçalho: Contém o token de acesso;
- e) Corpo quando necessário, em formato JSON JavaScript Object Notation.

A resposta a uma solicitação, obedece o padrão mostrado na figura 45.

Figura 45 – Resposta ou response



Fonte: Autor do projeto

### 7.3.5 Código de status

Dentro do modelo REST, os códigos de retorno do servidor são muito importantes e determinam em que estado nossa solicitação se encontra, esta é a conveção do protocolo HTTP (Saudate, 2016, p.27):

- a) 1xx - Informacionais;
- b) 2xx- Códigos de sucesso;
- c) 3xx- Códigos de redirecionamento;
- d) 4xx- Erros causados pelo cliente;
- e) 5xx - Erros originados no servidor.

Dentro deste padrão esta API adota os seguintes valores de status para as requisições de recursos:

- a) 200 OK Para operações realizadas com sucesso;

- b) 204 No Content Para operações de PUT, POST ou DELETE, onde o servidor pode se recusar a enviar conteúdo;
- c) 400 Bad Request Resposta de erro genérico para qualquer erro de processamento;
- d) 401 Unauthorized Para solicitações não autorizadas ou com token de acesso inválido;
- e) 404 Not Found Para recursos que não existem no nosso contexto.

## 7.4 Modelagem do banco de dados

Para a interação com o banco de dados foi utilizado o ORM Sequelize e sendo que a atualização deste modelo se dá sempre que uma alteração nos modelos da aplicação ocorrem. A figura 46 mostra como fica a estrutura do modelo que representa os devices.

Figura 46 – Representação de devices

```
module.exports = (sequelize, DataType) => {
  const Devices = sequelize.define("Devices", {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoincrement: true
    },
    description: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    switch_on: {
      type: DataType.BOOLEAN,
      allowNull: false,
      defaultValue: false
    },
    on_line: {
      type: DataType.BOOLEAN,
      allowNull: false,
      defaultValue: false
    }
  }, {
    classMethods: {
      associate: models => {
        Devices.belongsTo(models.Users);
      }
    }
  });
  return Devices;
};
```

Fonte: Autor do projeto

Na figura 47 é exibido a estrutura que representa os Users.

Figura 47 – Representação de users

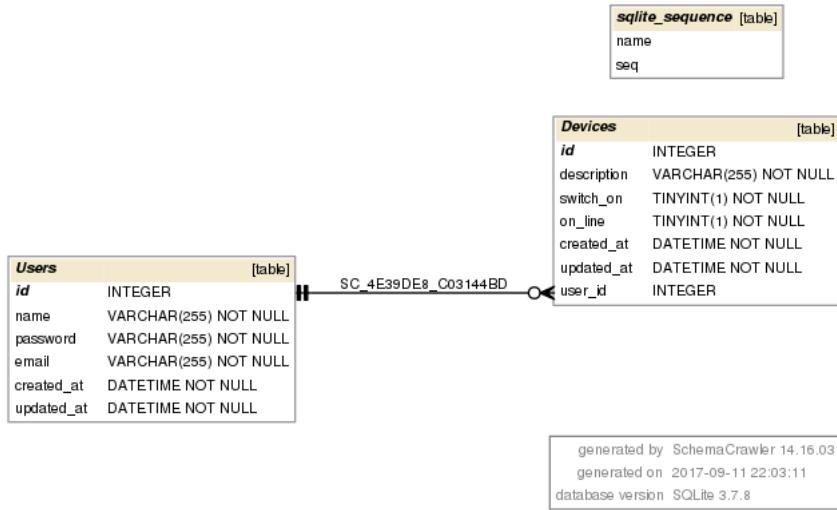
```
import bcrypt from "bcrypt";

module.exports = (sequelize, DataType) => {
  const Users = sequelize.define("Users", {
    id: {
      type: DataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    name: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    password: {
      type: DataType.STRING,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    },
    email: {
      type: DataType.STRING,
      unique: true,
      allowNull: false,
      validate: {
        notEmpty: true
      }
    }
  }, {
    hooks: {
      beforeCreate: user => {
        const salt = bcrypt.genSaltSync();
        user.password = bcrypt.hashSync(user.password, salt);
      }
    },
    classMethods: {
      associate: models => {
        Users.hasMany(models.Devices);
      },
      isPassword: (encodedPassword, password) => {
        return bcrypt.compareSync(password, encodedPassword);
      }
    }
  });
  return Users;
};
```

Fonte: Autor do projeto

A modelagem ER (Entidade Relacionamento) do banco de dados é apresentada na figura 48.

Figura 48 – ER do banco de dados



Fonte: Autor do projeto

## 8 FUNCIONAMENTO DO SISTEMA

Nesta seção estão dispostas as funcionalidades e interfaces desenvolvidas ao longo do projeto.

### 8.1 Firmware para controle do módulo WIFI

Foi utilizado um microcontrolador fabricado pela empresa Espressif o ESP8266, este microcontrolador é controlado por um *firmware* escrito utilizando o *framework* Arduíno. Este framework divide-se em duas partes, seu núcleo, com as funções básicas de controle do hardware e uma lista de bibliotecas que permitem a expansão e o acesso a novos periféricos de outros modelos de placas micro controladas, como a ESP8266 que possui uma interface de comunicação via WIFI. Através desta interface de rede WIFI, são efetuadas as trocas de mensagens via TCP com a central de controle, utilizando o protocolo de comunicação MQTT, a responsabilidade da central de controle é de fazer o gerenciamento de dispositivos conectados na rede e a distribuição de mensagens.

Na central de controle um serviço de controle de mensagens, chamado *broker* de mensagens, recebe e envia mensagens ela é uma camada intermediária entre os Módulos Auxiliares (controlados pelo ESP8266) e as ações que o usuário deseja fazer, de um lado esta central sabe "conversar" com os módulos, de outro utilizando a API para controle ela troca mensagens com o usuário.

### 8.2 API para controle de Hardware

Esta API é o caminho pelo qual aplicativos variados podem se comunicar com a ponta final do sistema, os Módulos Auxiliares. Para que um aplicativo possa utilizá-la, basta que este consiga consumir uma API RESt, esta é uma forma de tornar este sistema atrativo para outros desenvolvedores que tenham interesse em fazer algum tipo de integração com um sistema de domótica simplificado como é o Ysto.

Esta API é a figura central do sistema, todas as ações e controles de comportamento do sistema passam por ela. As sub-seções a seguir descrevem de forma mais detalhada seu funcionamento.

#### 8.2.1 Sobre esta API

Este *end-point* traz informações sobre a API, dados como nome e versão. A figura 49 monstra esta chamada através de um cliente de RESt genérico.

Figura 49 – Informações da API

The screenshot shows a user interface for making HTTP requests. At the top, there's a header bar with a 'Request' tab, a refresh icon, settings, and a '+' button. Below it, the 'URL' field contains 'http://ysto.local:3001/api'. The 'Method' dropdown is set to 'GET'. To the right of the method is a blue 'Send request' button. Underneath these fields are two links: 'Headers >' and 'Basic auth >'. The main area is titled 'Response (0.421s) - http://ysto.local:3001/api'. It displays a green '200 OK' status. Below the status, there's another 'Headers >' link. The most prominent part is a code block showing a JSON object:

```
{
  "versao": "2.0.0",
  "about": "ysto-API"
}
```

Fonte: Autor do projeto

### 8.2.2 Solicitação de token

Este *end-point* espera o envio das credenciais de um usuário válido para então devolver um token de segurança, por motivo de simplificação este token não possui um tempo de validade, ficando a critério do desenvolvedor elaborar uma política para lidar com a validade deste token, uma boa prática seria armazenar este token no que a World Wide Web Consortium (W3C) chama de *session storage*, desta forma o token existe apenas no tempo de "vida" da janela do aplicativo. A figura 50 mostra o envio de credenciais e o recebimento de um token válido.

Figura 50 – Requisição de token de acesso

The screenshot shows a POST request to `http://ysto.local:3001/api/auth`. The request body is set to `JSON` and contains two fields: `email` (fabiano@mail.net) and `password` (123456). The response is a `200 OK` status with a large JSON token string.

```

Request
URL: http://ysto.local:3001/api/auth
Method: POST
Send request

Headers >
Basic auth >
Request body >
Type: JSON
email: fabiano@mail.net
password: 123456
+Add parameter

Response (0.35s) - http://ysto.local:3001/api/auth
200 OK
Headers >
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6W3siaWQiOjF9XX0.eetLj4DKGNshMe9uCZmtv0ayZPFHva_PsrqANvG6pRI"

```

Fonte: Autor do projeto

### 8.2.3 Manutenção de usuários do sistema

Este *end-point* faz as operações de CRUD dos usuários do sistema, a idéia é que o banco seja criado com um usuário padrão e a partir deste usuário todas as ações relacionadas a usuários sejam feitas a partir dele, conforme a necessidade dos moradores. Este é um *end-point* que exige um token de autenticação válido no cabeçalho da mensagem, solicitações que não possuem esta assinatura serão recusadas pela central de controle, a figura 51 mostra o cadastro de um usuário via API.

Figura 51 – Manutenção de usuários

The screenshot shows a request configuration and a response view.

**Request:**

- URL: http://ysto.local:3001/api/users
- Method: GET
- Headers:
  - Authorization: bearer eyJhbGciOiJIUzI1Nils
  - +Add header
  - Basic auth >

**Response (0.044s) - http://ysto.local:3001/api/users**

**200 OK**

**Headers:**

```
Date: Sat, 18 Nov 2017 12:49:01 GMT
Server: WSGIServer/0.1 Python/2.7.9
Content-Length: 196
Content-Type: application/json
Cache-Control: no-cache
```

**Body:**

```
{
  "users": [
    {
      "name": "Fabiano Gomes",
      "created_at": "2017-10-14 17:44:50",
      "updated_at": "2017-10-14 17:44:50",
      "id": 1,
      "password": "e10adc3949ba59abbe56e057f20f883e",
      "email": "fabiano@mail.net"
    }
  ]
}
```

Fonte: Autor do projeto

### 8.2.4 Manutenção de dispositivos do sistema

Este *end-point* faz as operações de CRUD de dispositivos que são controlados pelo sistema, é importante entender que esta é uma ação que deve estar integrada com o *firmware* de controle dos Módulos Auxiliares, ou seja, o *firmware* deve ser gravado com o mesmo tópico cadastrado na API. Atualmente a API controla apenas um tópico chamado relay, desta forma uma mensagem será endereçada da seguinte forma, tópico de destino <NOME-DISPOSITIVO>/output e conteúdo da mensagem [1 | 0] para ligar ou desligar a saída do módulo referenciado. A figura 52 mostra como é feito o cadastro de um dispositivo, este *ed-point* da API só aceita solicitações mediante um token válido.

Figura 52 – Manutenção de dispositivos

The screenshot shows a request configuration window and its corresponding response.

**Request:**

- URL: http://ysto.local:3001/api/devices
- Method: GET
- Headers:
  - Authorization: bearer eyJhbGciOiJIUzI1NlslnR5cCl6lkpXVCJ9.eyJpZCI6W3siaWQiOjFf
  - +Add header
  - Basic auth >

**Response (0.25s) - http://ysto.local:3001/api/devices**

**200 OK**

**Headers:**

```
Date: Sat, 18 Nov 2017 13:09:36 GMT
Server: WSGIServer/0.1 Python/2.7.9
Content-Length: 319
Content-Type: application/json
Cache-Control: no-cache
```

**Body:**

```
{
  "devices": [
    {
      "user_id": 33,
      "description": "MA002",
      "created_at": "2017-1015 13:35:03",
      "updated_at": "2017-1015 13:36:12",
      "switch_on": 0,
      "on_line": 0,
      "id": 2
    },
    {
      "user_id": 33,
      "description": "MA003",
      "created_at": "2017-1014 10:18:22",
      "updated_at": "2017-1015 12:00:30",
      "switch_on": 0,
      "on_line": 0,
      "id": 670
    }
  ]
}
```

Fonte: Autor do projeto

### 8.2.5 Controle de dispositivo

Este *end-point* faz o controle de acionamento dos dispositivos cadastrados, ele recebe o comando de acionamento (ligar ou desligar) e repassa para o *broker*, este por sua vez faz o envio para o Módulo Auxiliar correspondente, a figura 53 mostra este envio. Da mesma forma que os demais tópicos, este depende da assinatura por um token válido.

Figura 53 – Atualização de estado de um dispositivo

The screenshot shows the POSTMAN interface with two main sections: Request and Response.

**Request:**

- URL:** http://lysto.local:3001/api/devices/670
- Method:** PUT
- Headers:** Authorization: bearer eyJhbGciOiJIUzI1Nils
- Request body:**
  - Type: JSON
  - id: 670
  - switch\_on: 1

**Response (0.325s) - http://lysto.local:3001/api/devices/670**

**200 OK**

**Headers:**

```
Date: Sat, 18 Nov 2017 13:48:17 GMT
Server: WSGIServer/0.1 Python/2.7.9
Content-Length: 191
Content-Type: application/json; charset=utf-8
```

**Body:**

```
{"devices": [{"user_id": 33, "description": "MA003", "created_at": "2017-10-14 10:18:22", "updated_at": "2017-11-18 13:48:17", "switch_on": 1, "on_line": 0, "id": 670}]}
```

Fonte: Autor do projeto

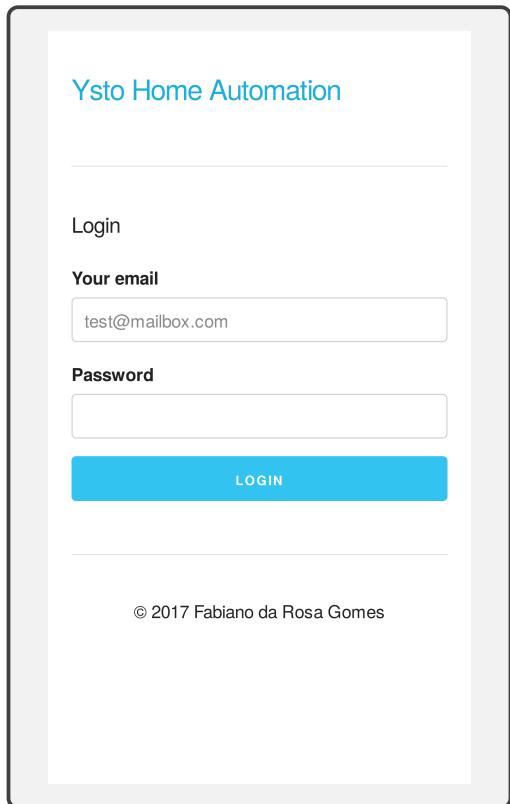
## 8.3 Interface Gráfica para controle do sistema

Esta interface é um exemplo de como a API pode ser utilizada, fica hospedada na central de controle e pode ser acessada via *web browser* por dispositivos móveis e *desktops* independente do sistema operacional.

### 8.3.1 Acesso ao sistema

Na tela de acesso, figura 54, deve ser fornecido um email de usuário válido e sua senha.

Figura 54 – Tela de login

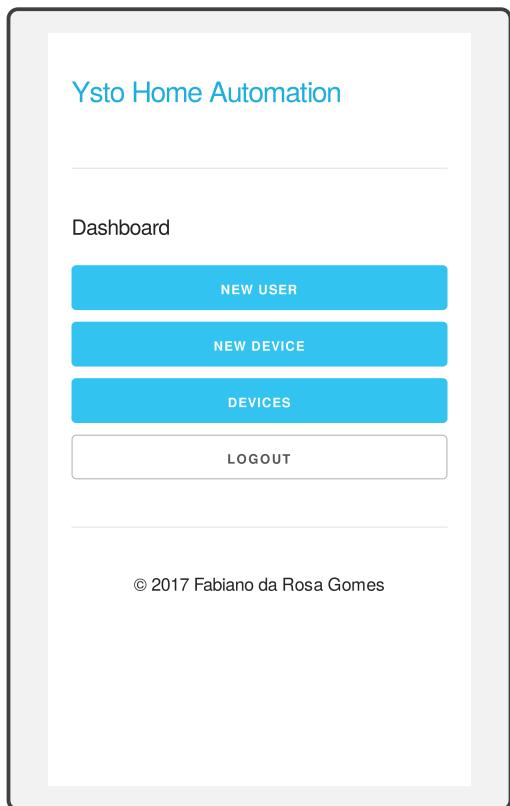


Fonte: Autor do projeto

### 8.3.2 Painel de controle

No painel de controle é oferecido uma lista de possibilidades para o usuário navegar no sistema, a figura 55 mostra estas possibilidades.

Figura 55 – Tela do painel de controle

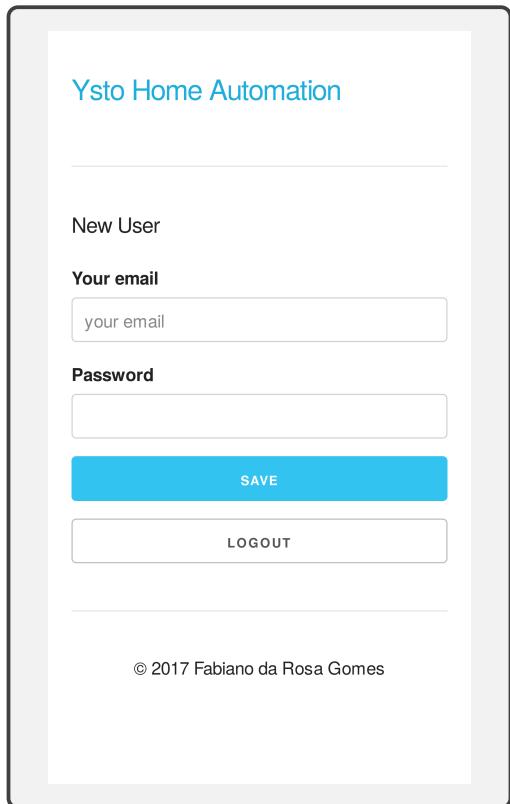


Fonte: Autor do projeto

### 8.3.3 Cadastro de novos usuários

Esta funcionalidade faz o cadastro de um novo usuário do sistema, utilizando uma email válido e a senha definida pelo usuário, apenas um *hash* MD5 é armazenado no banco de dados. A figura 56 mostra como é esta interface.

Figura 56 – Tela de cadastro de novos usuários

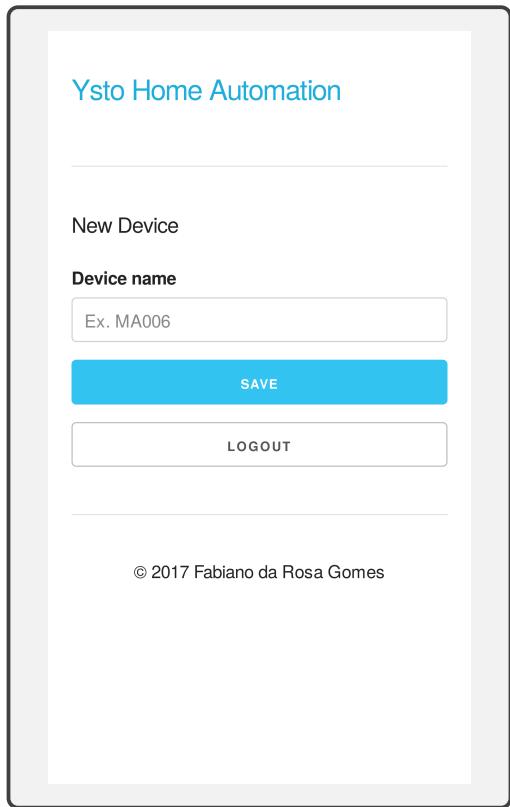


Fonte: Autor do projeto

#### 8.3.4 Cadastro de novos dispositivos

Esta funcionalidade faz o cadastro de novos dispositivos no sistema, importante lembrar que este nome deve ser o mesmo que é gravado no *firmware* dos módulos de controle. A figura 57 mostra esta funcionalidade.

Figura 57 – Tela de cadastro de novos dispositivos

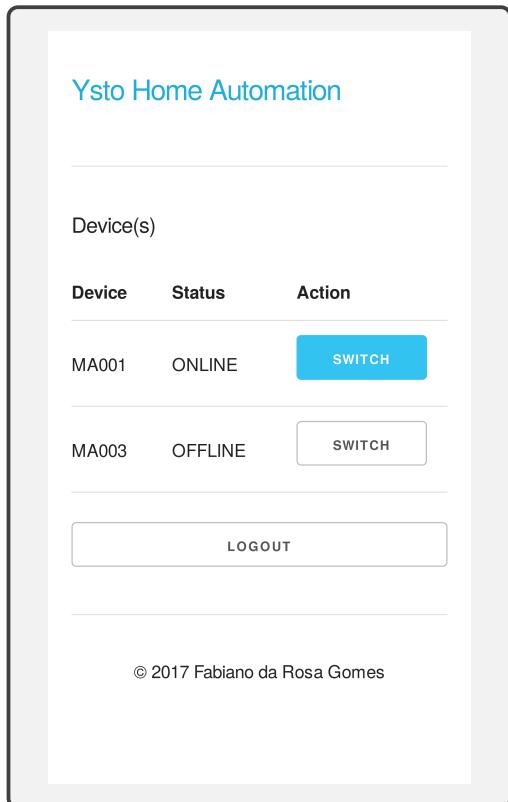


Fonte: Autor do projeto

### 8.3.5 Controle de dispositivos

Nesta tela é apresentado a lista de dispositivos cadastrados é possível fazer a troca de estado utilizando o botão *switch*, a figura 58 mostra como ficou esta tela.

Figura 58 – Tela de controle dos dispositivos



Fonte: Autor do projeto

Por apresentar um *layout* simples e com nome de funções de fácil entendimento, a utilização desta interface pretende oferecer um manuseio rápido e fácil mesmo para leigos no assunto.

### 8.3.6 Retorno para tela inicial

Para voltar para a tela de inicio, basta "clicar" no título da página de cada tela, no caso "Ysto Home Automation".

## 9 VALIDAÇÃO

A seguir é descrita a estratégia de validação selecionada para este projeto, a mesma busca verificar a usabilidade e bom funcionamento do sistema, assim como aferir se o mesmo atende as expectativas dos usuários naquilo em que se propõe.

### 9.1 Testes funcionais automatizados - API

O motor deste projeto é a API que acessa os módulos auxiliares, através de um suíte de testes automatizados, foram testados os seguintes *end-points* da API:

- a ) <IP>/api/: Sobre esta API;
- b ) <IP>/api/auth: Devolve um token de autorização para acesso as áreas restritas da API;
- c ) <IP>/api/devices: Retorna todos os dispositivos cadastrados;
- d ) <IP>/api/users: Retorna todos os usuários cadastrados

A figura 59 mostra o resultado destes testes.

Figura 59 – Testes funcionais automatizados

```
root@ysto:~/ysto/vuejs-teste# ./test_api.py
test_about (__main__.TestAPI) ... ok
test_get_devices (__main__.TestAPI) ... ok
test_get_devices_id (__main__.TestAPI) ... ok
test_get_users (__main__.TestAPI) ... ok
test_token_ok (__main__.TestAPI) ... ok

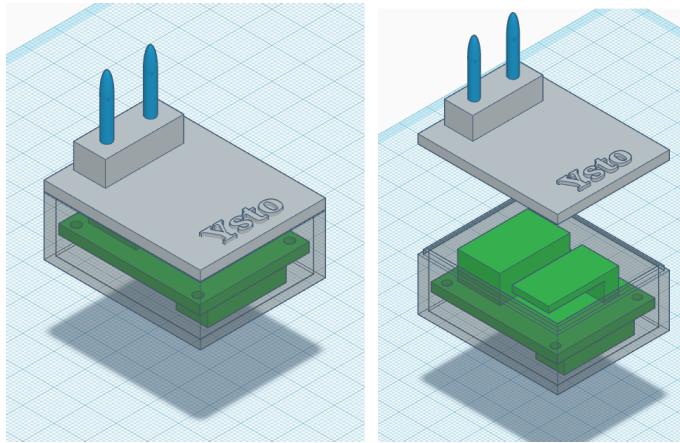
-----
Ran 5 tests in 0.114s
OK
root@ysto:~/ysto/vuejs-teste# █
```

Fonte: Autor do projeto

### 9.2 Embalagem e apresentação

Por se tratar de uma proposta de produto, é importante que este esteja disposto de uma forma que facilite o seu manuseio. A figura 60 mostra um conceito de embalagem para o módulo auxiliar.

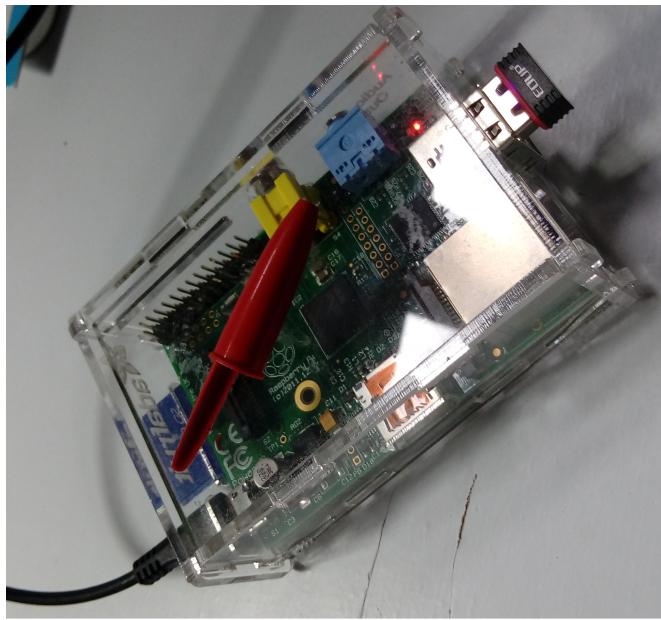
Figura 60 – Embalagem conceitual do módulo auxiliar



Fonte: Autor do projeto

Já a figura 61 usa um *case* comercial para a RaspberryPI a central de controle.

Figura 61 – *Case* comercial da central de controle



Fonte: Autor do projeto

### 9.3 Alcance de acionamento

Os testes foram realizados com base em um terreno com 12 metros de frente por 30 metros de profundidade. Trata-se de um projeto que depende da qualidade do sinal da rede wifi, nestes ambientes sempre existe uma oscilação de sinal que deve ser desconsiderado para fins de testes práticos, em testes de laboratório isso poderia ser melhor explorado.

Para os testes do Módulo Auxiliar, o que impacta diretamente no desempenho

de resposta é a qualidade da antena do módulo, em testes com variação de distância dentro do espaço descrito, ocorreram algumas falhas de acionamento quando nos limites do terreno. Um repetidor de sinal resolveria isso facilmente, não foi possível fazer esta verificação.

## 9.4 Interface com usuário

A idéia da interface com usuário é a de ser um painel de controle muito simples e de fácil manuseio, os conceitos foram repassados para uma família de 5 pessoas e para mais 2 pessoas de fora. Na época da explicação de como o projeto funciona, a interface com usuário não estava pronta e funcionando o que prejudica a análise do produto.

## 9.5 Escopo e oportunidade

Este projeto nasceu com uma proposta genérica de domótica residencial, não possuía nenhum cliente em potencial além das necessidades do próprio desenvolvedor. Este cenário mudou a partir do momento que outras pessoas passaram a tomar conhecimento do que este projeto se propunha a resolver e duas novas oportunidades surgiram.

### 9.5.1 Sinalização para salas de espera

Uma aplicação que não exigiria nenhuma adaptação da proposta atual seria a utilização do Ysto como um sinalizador de salas de consultórios médicos, onde é necessário avisar que um paciente chegou e está esperando. Este projeto pode ser usado para este fim uma vez que a recepcionista poderia administrar a central de controle e sempre que precisasse sinalizar para um médico em uma determinada sala, bastaria acionar o tópico correspondente a esta sala. Os tópicos poderiam ser formatados da seguinte forma, <NOME-DA-SALA>/saída. A carga a ser acionada pelo módulo auxiliar pode ser uma luminária comercial a gosto do decorador da clínica.

### 9.5.2 Sinalização para praças de alimentação

Da mesma forma uma praça de alimentação poderia utilizar de luminárias fixadas em suas mesas para fazer o controle e aviso de que o pedido está disponível, uma sugestão de tópico para este caso seria <MESA-CLIENTE>/saída.

### 9.5.3 Situação atual

Ambas possibilidades tem grande potencial e poderão se tornar uma proposta real, a idéia de utilizar como um sinalizador nasceu em uma conversa com a dona de uma clínica

médica que viu neste projeto a possibilidade de uso em um curto espaço de tempo (em negociação).

## 10 CONSIDERAÇÕES FINAIS

Este projeto foi desenvolvido ao longo do ano de 2017, a figura 62 mostra a quantidade de horas trabalhada.

Figura 62 – Controle horas trabalhadas

Controle de horas do projeto	
Descrição	Horas
Desenvolvimento - Sprint 1	10
Desenvolvimento - Sprint 2	5
Desenvolvimento - Sprint 3	5
Desenvolvimento - Sprint 4	8
Desenvolvimento - Sprint 5	19
Desenvolvimento - Sprint 6	32
Desenvolvimento - Sprint 7	24
Desenvolvimento - Sprint 8	24
Reuniões de acompanhamento	8
Documentação	64
Pesquisa	64
Bug-fix de funcionalidades	64
<hr/>	
Total de horas trabalhadas	327

Fonte: Autor do projeto

Foi uma atividade que envolveu pesquisa, gerenciamento de tempo e criatividade, nem sempre tudo ocorre como planejado e fazer adaptações e descobrir melhores soluções fizeram parte desta atividade. Ao longo do desenvolvimento deste projeto muitas ideias e conceitos interessantes foram descobertos, mas nem sempre foi possível adicioná-los ao projeto, por isso é importante ao menos citá-los como uma referência para implementações futuras.

### 10.1 Funcionalidades futuras

Estas funcionalidades foram separadas em dois grupos.

#### 10.1.1 Segurança

Toda informação deveria estar transitando em um meio seguro para troca de mensagens, no caso a utilização do protocolo http com sua camada de segurança em ssl, isso envolve o serviço da API e a troca de mensagens do *broker* no protocolo mqtt.

### 10.1.2 Hospedagem em servidor externo

Hospedar em um serviço externo traria a possibilidade de controlar os dispositivos de casa em qualquer lugar onde os usuários pudessem acessar a internet. A dificuldade estava em fazer um aplicativo que soubesse "chavear" entre uma rede local e um serviço na web, com o uso de *Progressive Web Apps* (PWA) a mesma estrutura de aplicativo web instalado na central de controle funcionaria para controle usando o serviço hospedado. Outro ponto que seria necessário lidar, o *broker* precisa existir no serviço remoto, isso poderia ser resolvido com a criação de *containers* Docker com a mesma infraestrutura da central de controle. Nos teste realizados durante o TCC-1 foi feito a ligação do *broker* local com um *broker* remoto, estas tentativas foram deixadas de lado visando um produto mínimo viável.

### 10.1.3 Mais opções de dispositivos

Atualmente a API sabe lidar apenas com o acionamento de relés, seria muito interessante adicionar funcionalidades como acionamento por *timer*, leitura de sensores de temperatura e detecção de presença.

## REFERÊNCIAS

- ARDUINO. *Framework para computação Física*. 2017. Disponível em: <<https://www.arduino.cc>>. Acesso em: 2017-03-14. Citado na página 19.
- AURESIDE. *Sobre o mercado de automação em 2016/17*. 2017. Disponível em: <<http://plataformaconnectar.blogspot.com.br/2017/05/o-mercado-de-automacao-residencial.html>>. Acesso em: 2017-05-20. Citado 2 vezes nas páginas 12 e 13.
- DEBIAN. *Debian o Sistema operacional Universal*. 1993. Disponível em: <<https://debian.org>>. Acesso em: 2017-03-14. Citado na página 20.
- EMBARCADOS. *O que é Arduino*. 2017. Disponível em: <<https://www.embarcados.com.br/arduino/>>. Acesso em: 2017-06-03. Citado na página 19.
- EPRESSIF. *Fabricante de chip SOC ESP8266*. 2017. Disponível em: <<https://espressif.com>>. Acesso em: 2017-03-14. Citado na página 18.
- FOUNDATION, P. S. *Sobre Python*. 2017. Disponível em: <<https://www.python.org/about/>>. Acesso em: 2017-11-14. Citado na página 19.
- GEANY. *Geany IDE*. 2017. Disponível em: <<https://www.geany.org>>. Acesso em: 2017-08-27. Citado na página 20.
- GIT. *Git SCM*. 2017. Disponível em: <<https://git-scm.com>>. Acesso em: 2017-08-27. Citado na página 21.
- GLIFFY. *Gliffy*. 2017. Disponível em: <<https://www.gliffy.com>>. Acesso em: 2017-09-02. Citado na página 21.
- IBM. *Protocolo de comunicação para IoT*. 2017. Disponível em: <<http://mqtt.org>>. Acesso em: 2017-03-14. Citado na página 20.
- MARINGA, U. E. de. *O que é domótica*. 2017. Disponível em: <<http://www.din.uem.br/ia/intelige/domotica/int.htm>>. Acesso em: 2017-11-14. Citado na página 11.
- MOSQUITTO. *Broker opensource para o protocolo MQTT*. 2017. Disponível em: <<https://mosquitto.org>>. Acesso em: 2017-05-05. Citado na página 20.
- NODEJS. *JavaScript server side*. 2017. Disponível em: <<https://nodejs.org>>. Acesso em: 2017-03-14. Citado na página 19.
- RASPBERRYPI. *Computador miniaturizado*. 2017. Disponível em: <<http://www.raspberrypi.org>>. Acesso em: 2017-03-14. Citado na página 18.
- SCRUMSOLO. *Framework para controle e desenvolvimento de projetos ágeis*. 2017. Disponível em: <<https://scrum solo.wordpress.com>>. Acesso em: 2017-03-14. Citado na página 29.
- SKELETON. *Sobre skeleton*. 2017. Disponível em: <<http://getskeleton.com>>. Acesso em: 2017-11-14. Citado 2 vezes nas páginas 21 e 44.

SQLITE. *Sobre Sqlite*. 2017. Disponível em: <<https://pt.wikipedia.org/wiki/SQLite>>. Acesso em: 2017-11-14. Citado na página 20.

SUALEH. *SchemaCrawler*. 2017. Disponível em: <<http://sualeh.github.io/SchemaCrawler/>>. Acesso em: 2017-08-29. Citado na página 21.

VUEJS. *Sobre vuejs*. 2017. Disponível em: <<https://vuejs.org/>>. Acesso em: 2017-11-14. Citado na página 21.