



# Sistema de Regras para "Internet of Things"

**Relatório Final**

Inteligência Artificial

2017/2018

Mestrado Integrado em Engenharia Informática e  
Computação

Beatriz de Henriques Martins - up201502858@fe.up.pt  
João Pedro Furriel de Moura Pinheiro - up201104913@fe.up.pt  
Leonardo Manuel Gomes Teixeira - up201502848@fe.up.pt

20 de Maio de 2018

# Folha de Rosto

*Nome da Disciplina:* Inteligência Artificial

*Ano:* 2017/2018

*Data:* 20 de maio de 2018

*Título do trabalho:* Sistema de Regras para "Internet of Things"

*Nome dos elementos do grupo:*

- up201104913@fe.up.pt - João Pedro Furriel de Moura Pinheiro
- up201502848@fe.up.pt - Leonardo Manuel Gomes Teixeira
- up201502858@fe.up.pt - Beatriz de Henriques Martins

## Conteúdo

<b>1</b>	<b>Objetivo</b>	<b>4</b>
<b>2</b>	<b>Especificação</b>	<b>5</b>
2.1	Análise detalhada do tema . . . . .	5
2.2	Representação do Conhecimento e Explicação de <i>datasets</i> . . . . .	5
2.2.1	Sensores . . . . .	6
2.2.2	Dispositivos . . . . .	6
2.2.3	Regras . . . . .	6
2.3	Técnicas e Algoritmos . . . . .	7
2.4	Arquitetura . . . . .	7
<b>3</b>	<b>Desenvolvimento</b>	<b>8</b>
3.1	Ferramentas, Linguagens de Programação e Ambientes de Desen- volvimento . . . . .	8
3.2	Diagrama de classes . . . . .	8
3.3	Detalhes de Desenvolvimento . . . . .	9
<b>4</b>	<b>Experiências</b>	<b>10</b>
<b>5</b>	<b>Conclusões</b>	<b>10</b>
<b>6</b>	<b>Melhoramentos</b>	<b>10</b>
<b>7</b>	<b>Recursos</b>	<b>11</b>
7.1	Bibliografia . . . . .	11
7.2	Software . . . . .	11
7.3	Participação . . . . .	11
<b>8</b>	<b>Apêndice</b>	<b>12</b>

# 1 Objetivo

Este trabalho consiste na criação de um Sistema de regras para *Internet of Things*.

*Internet of Things*, *Internet das Coisas* - IOT, é a capacidade de objectos e/ou serviços *Web* do dia-a-dia serem ligados à internet e entre si de maneira a enviar e receber dados e alterar o seu estado com base em regras pré-definidas. Este conceito tem uma infinidade de aplicações. Não só na Domótica, área mais conhecida como *Casas Inteligentes*, onde é possível ativar o aquecimento central ou fechar as persianas a partir do nosso telemóvel ou automaticamente através de inputs de sensores, esta área também é aplicada na Saúde, por exemplo, no controlo das bombas de insulina através de uma aplicação móvel.

O objetivo deste trabalho é o desenvolvimento de um sistema de regras de inferência para que estes dispositivos possam operar autonomamente mediante o envio e recepção de dados entre os mesmos.

## 2 Especificação

### 2.1 Análise detalhada do tema

Este projeto baseia-se no conceito de IOT e possui uma infinidade de aplicações, no sentido em que analisámos alguns temas possíveis para a realização deste projeto. Deste modo, propomo-nos a desenvolver um sistema de IOT para Domótica.

A Domótica é uma das áreas de IOT em maior expansão, em que os vários objetos e sensores instalados numa smart house são integrados e onde o utilizador da mesma poderá definir as suas regras através de uma aplicações. As regras definidas e a definir são do tipo IFTTT, *If this than that*, que deste modo criam condições não ambíguas.

Relativamente à divisão do trabalho, este está dividido em quatro grandes partes:

1. Definição de um conjunto de regras para vários dispositivos;
2. Calculo de funções de *fuzzify* e *defuzzify*;
3. Desenvolvimento de uma interface amiga do utilizador;
4. Fase de testes.

Tiramos partido do suporte que o *Jess* fornece ao uso de conhecimento difuso/incerto, mais concretamente do *FuzzyJess* por meio da biblioteca *FuzzyJ ToolKit*.

### 2.2 Representação do Conhecimento e Explicação de *datasets*

No contexto do nosso projeto, os *datasets* existentes são os **sensores**, como por exemplo um sensor de temperatura na sala ou um sensor de humidade do solo do jardim, os **dispositivos** da casa, como o ar-condicionado do quarto, ou o sistema de rega do jardim, e por fim, as **regras** de inferência correspondentes, como por exemplo *abrir a janela da sala se estiver 'calor'* ou *ligar o sistema de rega se a humidade do solo do jardim for 'baixa'*. É através dos sensores (factos) e das regras associadas que os dispositivos serão activados e manipulados automaticamente.

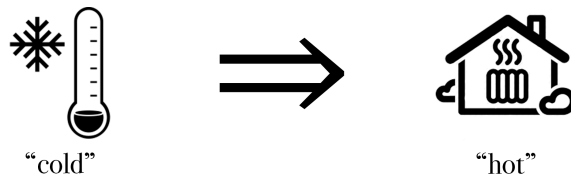


Figura 1: Exemplo de uma regra simples

### 2.2.1 Sensores

Os sensores são objetos da class *Sensor* e, como qualquer sensor normal, têm um valor numérico associado (Ex: Termómetro: 27,3°C; Sensor de movimento: 1 -> SIM). Além disso, cada sensor têm também uma *FuzzyVariable* que associa o respectivo valor numérico a vários estados, cada um definido por um *FuzzySet*, como por exemplo, Termómetro: {'cold': ZFuzzySet(10.0, 20.0), 'warm': PIFuzzySet(20.0, 10.0), 'hot': SFuzzySet(20.0, 30.0)}.

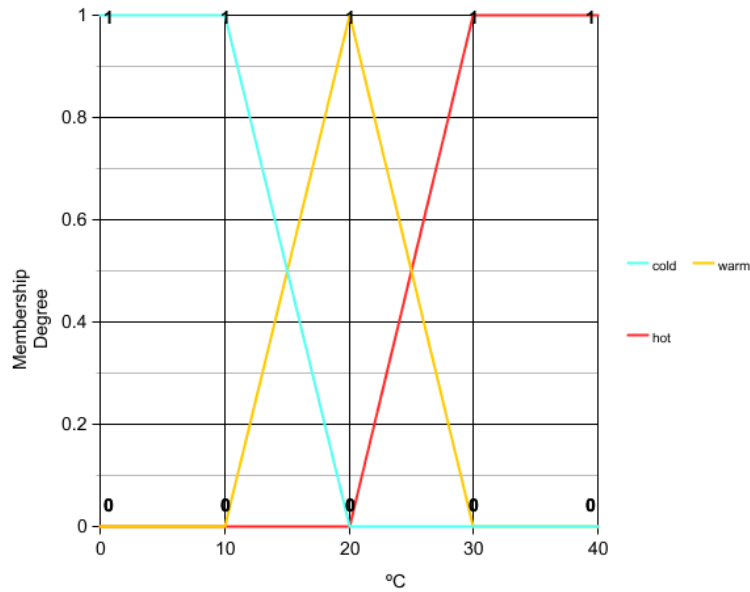


Figura 2: Gráfico aproximado da *FuzzyVariable* do Termómetro

### 2.2.2 Dispositivos

Os dispositivos são objetos da class *Device*, podendo estes ser *FuzzyDevice* (Ex: Ar-condicionado) ou *SimpleDevice* (Ex: Porta da sala).

Assim como os sensores, cada dispositivo *FuzzyDevice* também tem uma *FuzzyVariable* associada, que define o *FuzzySet* de cada termo (Figura 9).

### 2.2.3 Regras

As regras são definidas através de uma declaração (*rules.clp*) e criadas no Java (*JessManipulator*) pela função *createNewVersatileRule*. As novas regras são guardadas no ficheiro *rules.clp*.

```
1 (defrule testSimpleRuleF-N
2
3   (TemperatureSensor (name "Living Room Temperature Sensor") (
4     fuzzyValue ?v0&:(fuzzy-match ?v0 "cold")))
5
6   =>
7   (assert (Living_Room_Heater_1 (new FuzzyValue ?*heaterTemperature
8     * "high")))
```

Listing 1: Exemplo de definição de uma regra

## 2.3 Técnicas e Algoritmos

O motor de inferência responsável pela ativação das regras é o *Rete*, mais especificamente *FuzzyRete*. Este analisa os factos existentes e unifica com as regras guardadas na base de conhecimento, sendo este o comportamento de um Sistema de Regras. Cada variável possui um grau de pertença a um conjunto, podendo não pertencer completamente a este. Assim, é possível calcular o FuzzyValue de cada expressão numérica de sensores e dispositivos.

Usando a regra presente na Imagem 1 para fazer a analogia para o nosso projeto. “Se está *cold*, então aquecimento está *hot*” o motor de inferência calcula a função necessária para determinar, no processo de desfusãoção, uma temperatura concreta para a água armazenada no aquecimento.

O algoritmo de desfusãoção que utilizamos foi o *weighted Average Defuzzify*. Este algoritmo calcula a média ponderada de cada saída do conjunto de regras armazenadas na base de conhecimento do sistema.

## 2.4 Arquitectura

Através de uma interface, o utilizador introduz os dados, ou seja, altera o valor dos sensores. A *GUI* envia os dados para o *JessManipulator*, que por sua vez, envia para o motor de inferência. Depois de recalcular a função de desfusãoção os dados são enviados pelo caminho inverso, motor de inferência para *JessManipulator* e de seguida para a *GUI*.

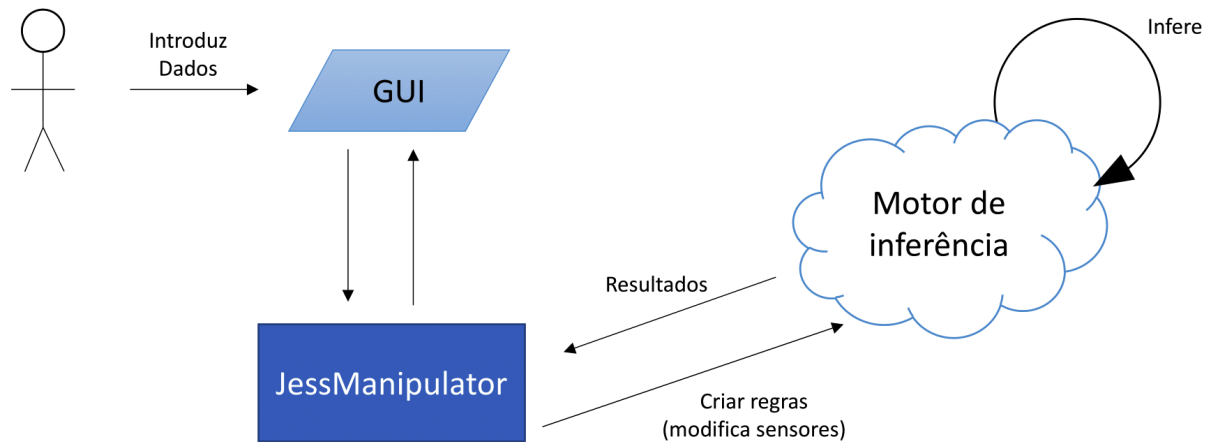


Figura 3: Arquitectura do sistema

## 3 Desenvolvimento

### 3.1 Ferramentas, Linguagens de Programação e Ambientes de Desenvolvimento

Este projeto foi desenvolvido em dois Sistemas Operativos distintos, *Linux 17.10* e *mac OS High Sierra*.

Os ambientes de desenvolvimento utilizados foram:

- *Eclipse Oxygen* para desenvolvimento do código *Java* e *debugging*;
- *Visual Studio Code* para manipulação de *Jess*, *FuzzyJess* e regras.

As linguagens de programação usadas foram:

- *Jess* - para interpretação de regras existentes no Sistema de Regras, usando a biblioteca FuzzyJ Toolkit;
- *Java* - invocação para leitura de regras e escrita de métodos em *Jess*;
- *Java Swing* - para implementação da interface gráfica.

### 3.2 Diagrama de classes

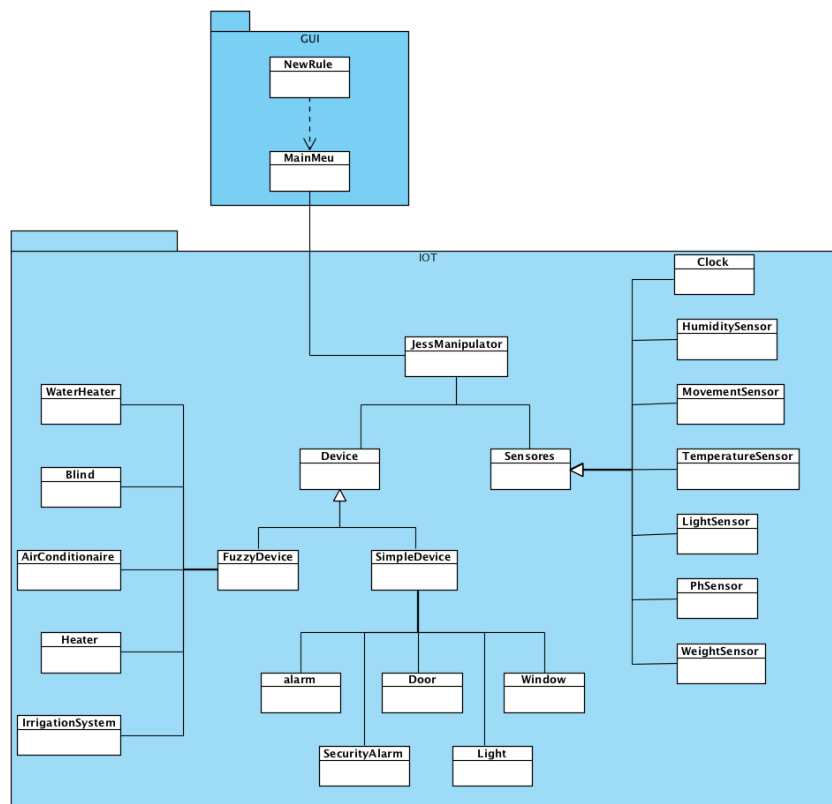


Figura 4: Diagrama de classes



### 3.3 Detalhes de Desenvolvimento

Alguns detalhes da implemetação são:

- Os sensores são os *factos* do motor de inferência, sendo estes instâncias da classe *Sensor* e os valores de cada um são campos de cada instância. NOTA: *?\*tempFvar\** - representa a *FuzzyVariable*

```
1 (bind ?temperatureSensorSmallBathroom (new TemperatureSensor "
    Small Bathroom Temperature Sensor" 30 ?*tempFvar* "
    temperatureSensorSmallBathroom"))
2 (add ?temperatureSensorSmallBathroom)
```

Listing 2: Criação do *ShadowFact* de um sensor

- Tanto os *Sensor* como os *FuzzyDevice* contêm uma *FuzzyVariable* que define os *FuzzySets* de cada estado possível do sensor ou dispositivo. Como por exemplo:

```
1 (defglobal ?*fanSpeed* = (new nrc.fuzzy.FuzzyVariable "fan
    speed" 0.0 1000.0 "RPM"))
2
3 (?*fanSpeed* addTerm "low"
4 (new nrc.fuzzy.ZFuzzySet 0.0 400.0))
5
6 (?*fanSpeed* addTerm "medium"
7 (new nrc.fuzzy.PIFuzzySet 500.0 200.0))
8
9 (?*fanSpeed* addTerm "high"
10 (new nrc.fuzzy.SFuzzySet 600.0 1000.0))
```

Listing 3: Criação de um *FuzzyDevice*

- Todas as regras foram definidas na linguagem *Jess*, logo quando uma nova regra é criada pelo utilizador é gerado o código *Jess* correspondente (na função *createNewVersatileRule*) e executado no motor *Rete*.

## 4 Experiências

A base do nosso projeto é um Sistema de Regras. As experiências efectuadas consistem na alteração dos input dos sensores e verificar se acontece alguma coisa nos dispositivos relacionados.

Através de uma interface é possível manipular os dados dos sensores (numa situação real os *inputs* seriam automáticos) e verificar as conclusões nos estados dos dispositivos.

## 5 Conclusões

Em suma, este é um tema bastante interessante e de vasta aplicação tanto no quotidiano como, por exemplo, em áreas de investigação e saúde. Permitindo ao grupo desenvolver uma aplicação bastante próxima da realidade e com resultados bastante positivos na área da Domótica.

O grupo teve algumas dificuldades com a linguagem *Jess* e com a biblioteca de *FuzzyJess* devido à sintaxe pouco amigável e à falta de documentação fiável.

Também sentimos algumas dificuldades na elaboração da estrutura do trabalho porque o grupo não tinha quase conhecimento nenhum sobre o tema.

## 6 Melhoramentos

Alguns aspectos a melhorar para o nosso projeto seria a implementação de algumas API's, nomeadamente uma API meteorológica para o input automático da temperatura e da humidade em alguns sensores. Também seria interessante fazer um protótipo para *android*.

## 7 Recursos

### 7.1 Bibliografia

- <https://ifttt.com/discover>
- <https://en.wikipedia.org/wiki/IFTTT>
- [https://pt.wikipedia.org/wiki/Internet\\_das\\_coisas](https://pt.wikipedia.org/wiki/Internet_das_coisas)
- <https://pt.wikipedia.org/wiki/Dom%C3%B3tica>
- <https://www.luzesom.pt/pt/instalacao-de-equipamentos/domotica/>
- <http://paginas.fe.up.pt/eol/AIAD/aulas/jess.pdf>
- «Jess Tutorial», Maarten Menken <mrmenken@cs.vu.nl> Vrije Universiteit, Amsterdam, The Netherlands

### 7.2 Software

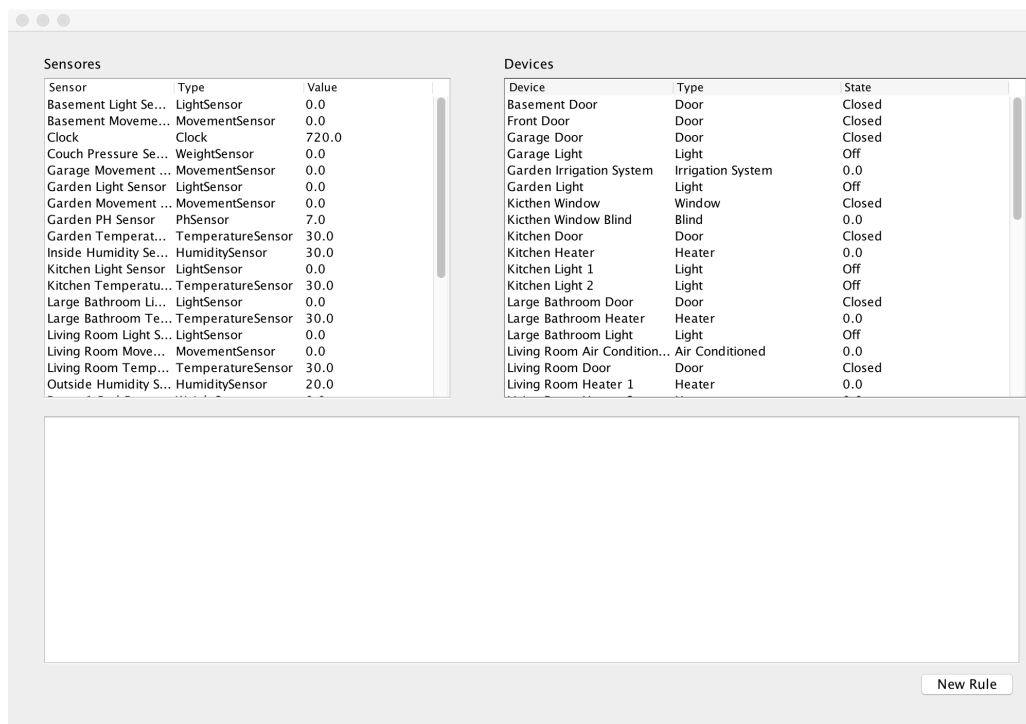
- *JESS*
- *JAVA*

### 7.3 Participação

- Beatriz: 33.3%
- João: 33.3%
- Leonardo: 33.3%

## 8 Apêndice

Quando iniciamos o programa não existem regras, apenas sensores e dispositivos, como podemos observar na figura seguinte:



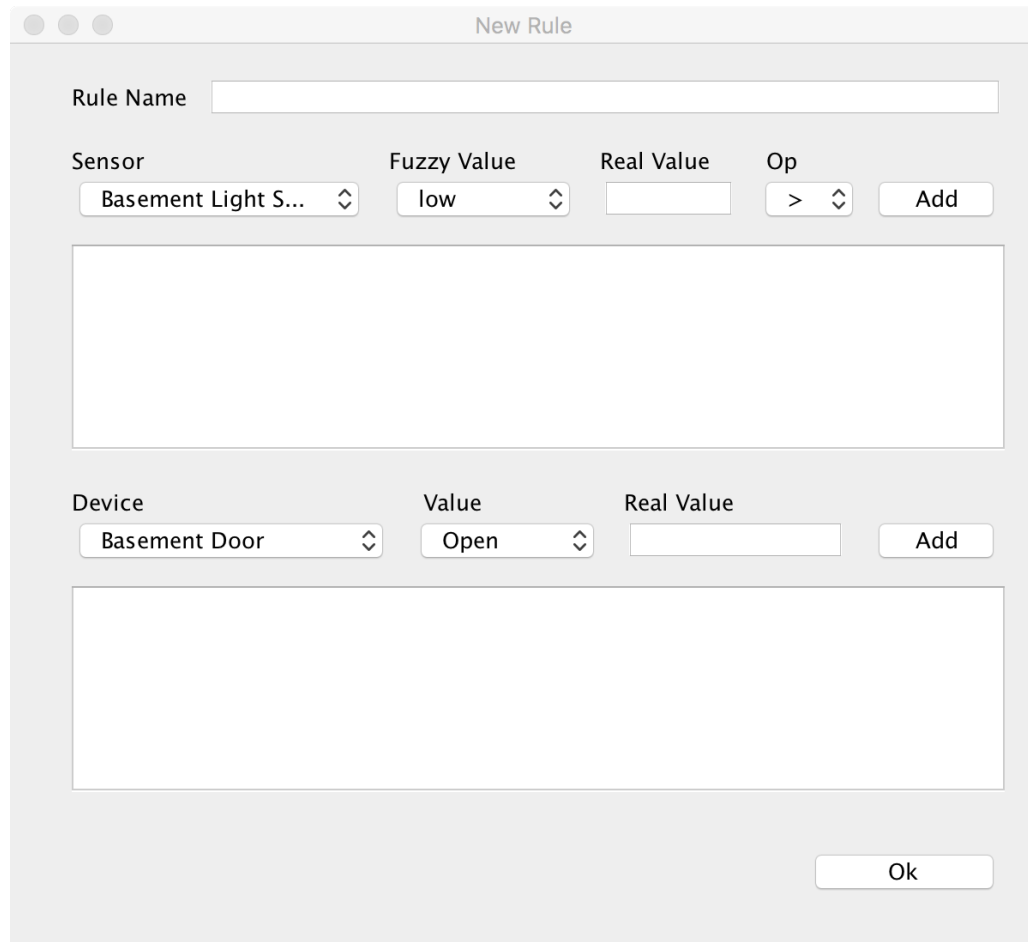
Sensores		
Sensor	Type	Value
Basement Light Se...	LightSensor	0.0
Basement Moveme...	MovementSensor	0.0
Clock	Clock	720.0
Couch Pressure Se...	WeightSensor	0.0
Garage Movement ...	MovementSensor	0.0
Garden Light Sensor	LightSensor	0.0
Garden Movement ...	MovementSensor	0.0
Garden PH Sensor	PhSensor	7.0
Garden Temperat...	TemperatureSensor	30.0
Inside Humidity Se...	HumiditySensor	30.0
Kitchen Light Sensor	LightSensor	0.0
Kitchen Temperatu...	TemperatureSensor	30.0
Large Bathroom Li...	LightSensor	0.0
Large Bathroom Te...	TemperatureSensor	30.0
Living Room Light S...	LightSensor	0.0
Living Room Move...	MovementSensor	0.0
Living Room Temp...	TemperatureSensor	30.0
Outside Humidity S...	HumiditySensor	20.0

Devices		
Device	Type	State
Basement Door	Door	Closed
Front Door	Door	Closed
Garage Door	Door	Closed
Garage Light	Light	Off
Garden Irrigation System	Irrigation System	0.0
Garden Light	Light	Off
Kitchen Window	Window	Closed
Kitchen Window Blind	Blind	0.0
Kitchen Door	Door	Closed
Kitchen Heater	Heater	0.0
Kitchen Light 1	Light	Off
Kitchen Light 2	Light	Off
Large Bathroom Door	Door	Closed
Large Bathroom Heater	Heater	0.0
Large Bathroom Light	Light	Off
Living Room Air Condition...	Air Conditioned	0.0
Living Room Door	Door	Closed
Living Room Heater 1	Heater	0.0

New Rule

Figura 5: Janela inicial

Para adicionar uma nova regra, basta clicar no botão “New Rule”. Este abre uma outra janela para a criação da nova regra.



The image shows a 'New Rule' dialog box with a light gray background and standard window controls (three dots) in the top-left corner. The title bar reads 'New Rule'. The dialog is divided into two main sections for defining rules. The top section has a 'Rule Name' text field at the top. Below it, there are four columns: 'Sensor', 'Fuzzy Value', 'Real Value', and 'Op'. The 'Sensor' column has a dropdown menu with 'Basement Light S...' selected. The 'Fuzzy Value' column has a dropdown menu with 'low' selected. The 'Real Value' column has an empty text field. The 'Op' column has a dropdown menu with '>' selected. To the right of these columns is an 'Add' button. Below this row is a large empty rectangular box. The bottom section has three columns: 'Device', 'Value', and 'Real Value'. The 'Device' column has a dropdown menu with 'Basement Door' selected. The 'Value' column has a dropdown menu with 'Open' selected. The 'Real Value' column has an empty text field. To the right of these columns is an 'Add' button. Below this row is another large empty rectangular box. At the bottom right of the dialog is an 'Ok' button.

Sensor	Fuzzy Value	Real Value	Op
Basement Light S...	low		>

Device	Value	Real Value
Basement Door	Open	

Figura 6: Janela para nova regra

De seguida, é necessário definir os sensores que irão disparar a regra. Os campos *Real Value* e *Op* apenas são usados quando se quer definir uma regra concreta. Caso contrário, utiliza-se o campo *Value* para escolher o estado difuso pretendido. Para os dispositivos procede-se de modo semelhante, tendo em conta que o campo *Real Value* é utilizado quando se quer definir um valor concreto para um *FuzzyDevice*.

New Rule

Rule Name

Sensor  Fuzzy Value  Real Value  Op

Garage Movement Sensor = yes

Device  Value  Real Value

Garage Light = On

Figura 7: Criação de uma nova regra

Após criar a regra é possível observá-la na lista de regras na janela inicial

Sensores

Sensor	Type	Value
Basement Light Se...	LightSensor	0.0
Basement Moveme...	MovementSensor	0.0
Clock	Clock	720.0
Couch Pressure Se...	WeightSensor	0.0
Garage Movement ...	MovementSensor	0.0
Garden Light Sensor	LightSensor	0.0
Garden Movement ...	MovementSensor	0.0
Garden PH Sensor	PhSensor	7.0
Garden Temperat...	TemperatureSensor	30.0
Inside Humidity Se...	HumiditySensor	30.0
Kitchen Light Sensor	LightSensor	0.0
Kitchen Temperatu...	TemperatureSensor	30.0
Large Bathroom Li...	LightSensor	0.0
Large Bathroom Te...	TemperatureSensor	30.0
Living Room Light S...	LightSensor	0.0
Living Room Move...	MovementSensor	0.0
Living Room Temp...	TemperatureSensor	30.0
Outside Humidity S...	HumiditySensor	20.0

Devices

Device	Type	State
Basement Door	Door	Closed
Front Door	Door	Closed
Garage Door	Door	Closed
Garage Light	Light	Off
Garden Irrigation System	Irrigation System	0.0
Garden Light	Light	Off
Kicthen Window	Window	Closed
Kicthen Window Blind	Blind	0.0
Kitchen Door	Door	Closed
Kitchen Heater	Heater	0.0
Kitchen Light 1	Light	Off
Kitchen Light 2	Light	Off
Large Bathroom Door	Door	Closed
Large Bathroom Heater	Heater	0.0
Large Bathroom Light	Light	Off
Living Room Air Condition...	Air Conditioned	0.0
Living Room Door	Door	Closed
Living Room Heater 1	Heater	0.0

Ligar\_Luzes: If Garage Movement Sensor is yes then Garage Light is set to On

New Rule

Figura 8: Depois de criar uma nova regra

Também é possível escolher um sensor e modificar o valor (*Value*). Referindo que isto apenas é necessário, uma vez que não usamos sensores reais ou nenhuma API externa para o mesmo efeito.

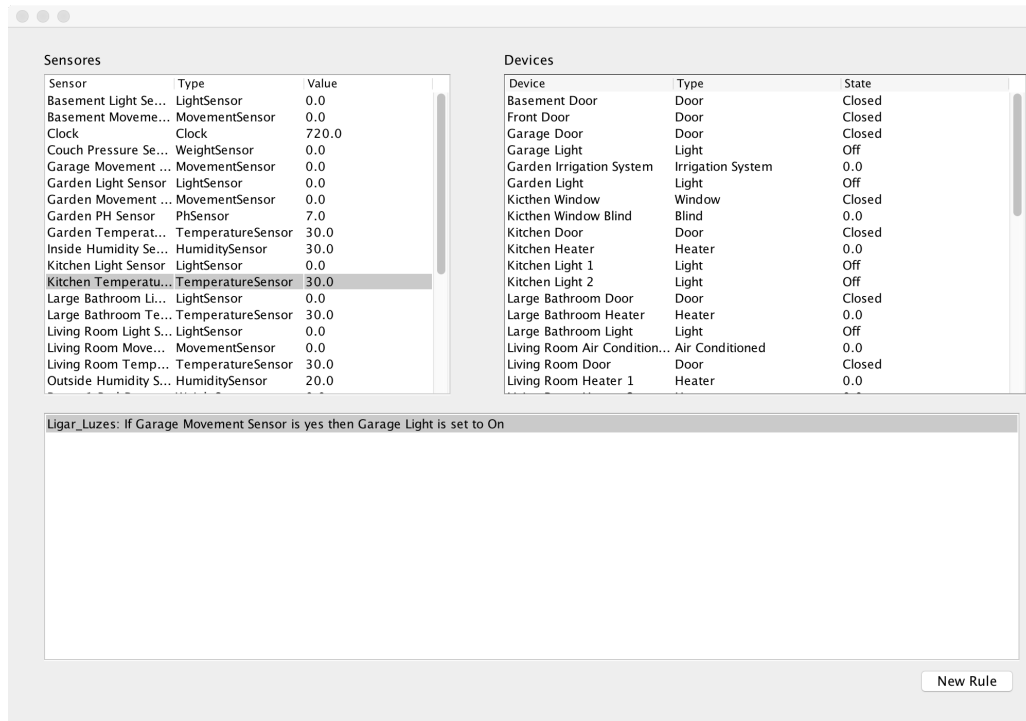


Figura 9: Modificar o estado de um sensor