



Universidade do Porto  
Faculdade de Engenharia

**FEUP**

# **Sistema de Regras para “Internet of Things”**

*Relatório Final*

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação



# Página do Rosto

**Título do Projeto:** Sistema de Regras para “Internet of Things”

**Curso:** Mestrado Integrado em Engenharia Informática e Computação

**Disciplina:** Inteligência Artificial

**Ano:** 2014/2015

**Grupo:**

- André Humberto Trigo de Bordalo Moraes - ei07122
- Turma 1
- ei07122@fe.up.pt
  
- João Alberto Trigo de Bordalo Moraes - ei12040
- Turma 1
- ei12040@fe.up.pt
  
- Maria João Pombinho Miranda – ei12046
- Turma 1
- ei12046@fe.up.pt

**Para o docente:** Professor Doutor Engenheiro Henrique Lopes Cardoso

**Data de Entrega:** 31 de Maio de 2015

## Objetivo

Pretende-se com este trabalho implementar um sistema que estabeleça ligações lógicas entre dispositivos tendo em conta o interesse e preferências do utilizador. Por dispositivos entende-se qualquer objecto com existência real (um electrodoméstico, por exemplo) bem como serviços e aplicações *web* (como é caso o *facebook*, *gmail*, *dropbox*, *twitter*, entre outras).

# Especificação

- **Análise detalhada do Tema:**

Este projecto baseia-se no conceito de *The Internet of Things*. Este conceito, de uma forma simples, consiste em criar um sistema que interligue vários dispositivos com diferentes funções e estabelecer troca de informações ou serviços entre si. Analisando as aplicações associadas, existe uma infinidade de possibilidades: desde áreas como a domótica, na concepção de casas inteligentes e automáticas - instalando *SmartGrids* no seio de um lar, por exemplo -; áreas de transporte, de forma a regular trânsito, gerir rotas marítimas; entre outras aplicabilidades.

Como há grande diversidade de áreas e o próprio conceito não implica exclusivamente dispositivos electrónicos, mas também *software*, serviços e aplicações *web*, o nosso projecto incide em juntar vários serviços *web* e proporcionar facilidades, optimizações e envio de informação ao seu utilizador.

Relativamente à divisão do trabalho, está dividido em quatro grandes partes: identificação das principais aplicações e serviços *web* de eleição da maioria dos utilizadores, os critérios de utilização popularidade, frequência de utilização e utilidade; funcionalidades mais utilizadas nestas aplicações e serviços; introduzir as *APIs* existentes destas aplicações e estabelecer conectividade entre si; e, por fim, fase de testes.

No que diz respeito à implementação da conectividade, teremos em conta as necessidades do utilizador e introduzir no contexto de IFTTT( *If This, Then That*), isto é, se o utilizador procurar ser notificado no *Facebook* se o tempo está quente, deverá aparecer uma mensagem com esse aviso na mesma aplicação , como por exemplo.

Tiramos partido do suporte que o *Jess* fornece ao uso de conhecimento difuso/incerto, mais concretamente do *FuzzyJess*.

- **Explicação de *datasets*:**

No contexto do nosso projecto, os únicos *datasets* são os factos introduzidos pelo utilizador. É através dos factos que as regras serão disparadas e com o seu devido tratamento, através de *handlers* que mediante o disparo de determinadas regras, alguma acção é realizada (nomeadamente envio de e-mails ou comunicações com o *Facebook*).

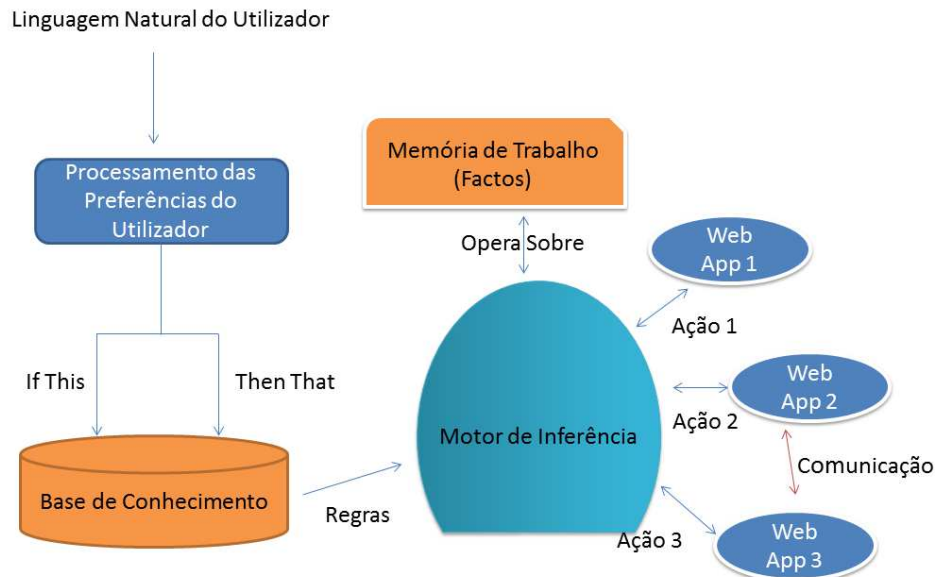
- **Técnicas, algoritmos:**

O algoritmo por detrás do disparo das regras é o RETE. Este algoritmo consiste em analisar os factos existentes na memória de trabalho e unificar com as regras existentes na base de conhecimento, disparando as regras relevantes, e, tirando as devidas conclusões, tal como um sistema pericial se deve comportar. Para suportar esta análise de regras e factos recorreremos à ferramenta *Jess*. Na realidade, o nosso projecto não recorre exclusivamente ao RETE; utilizamos o *FuzzyRETE* com o objetivo de introduzir o conceito difuso a variáveis que achamos necessárias, como é o caso da temperatura. Destaca-se que o *FuzzyRETE* é uma implementação por cima do RETE. Assim sendo, o que o nosso projecto usa é o *FuzzyRETE*.

- **Representação do Conhecimento:**

O conhecimento que o nosso sistema pericial adquire é por via das regras nele implementadas. Estas regras são do tipo IFTTT, tal como fora mencionado anteriormente. Mesmo com conceitos difusos, o sistema está preparado para compreender o seu significado e disparar as regras quando necessário.

- **Arquitetura:**



Em seguida será apresentada uma ilustração da arquitetura por detrás do nosso projecto:

**Legenda:** Arquitetura do sistema a desenvolver.

Por via de uma interface gráfica, o utilizador indica as suas preferências. Em seguida estas preferências serão analisadas e convertidas para regras do tipo IFTTT. Sempre que um evento acontece, ele é armazenado em forma de facto na Memória de Trabalho e, quando o Motor de Inferência é executado, tenta unificar os factos em memória com cabeças de regras e lança as acções, que no nosso caso são representadas pelas *Web App*.

## Desenvolvimento

### • Ferramentas, Linguagens e Ambientes Utilizados:

Este projecto foi realizado em ambiente Windows, mais concretamente em Windows 7 e 8.1. Relativamente aos ambientes de desenvolvimento, utilizamos a versão do Eclipse Luna – para desenvolvimento do código java e respectivo *debugging* –, editores de texto como *Notepad++* *Sublime text* – para desenvolvimento do código *Jess* e *FuzzyJess* –, e ainda o ficheiro *jess.bat* – para o *debugging* das regras feitas em *Jess*.

Estando subentendido no parágrafo anterior, as linguagens usadas foram: *java*, *Jess* e *FuzzyJess*. A primeira serve para implementar a interface gráfica, comunicação com APIs externas, nomeadamente *Facebook*, e ainda invocar alguns métodos para leitura das regras, escritas em *Jess*. As restantes linguagens serviram para indicar as regras existentes no nosso sistema pericial. Salienta-se que na escrita das regras foi necessário algum código *java*.

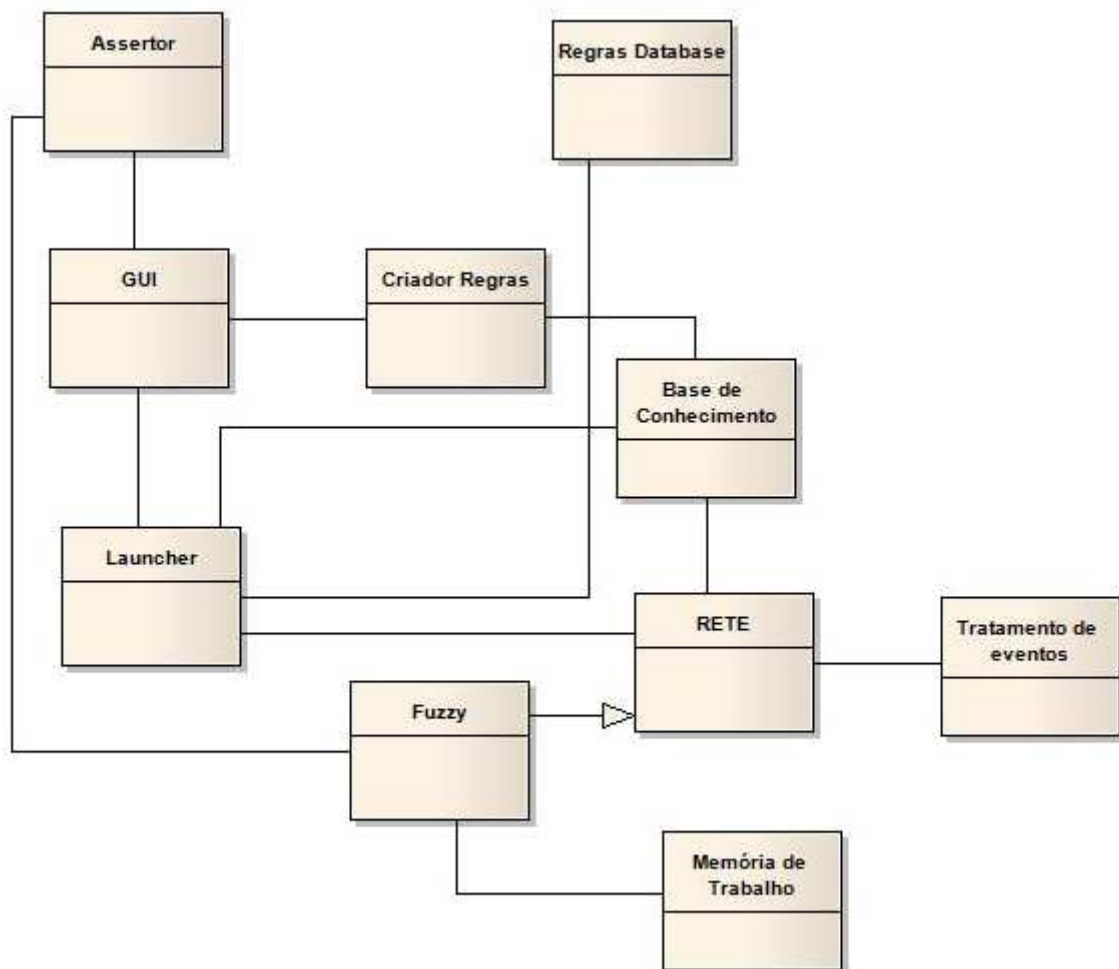
Também mencionado anteriormente, recorreremos a algumas APIs para integrar alguma complexidade no nosso sistema pericial: é o caso da API do *Facebook* e ainda a API de email do *java*. Nesta última, enviamos para o servidor de email da FEUP a mensagem a enviar e depois este, o servidor, trata de reencaminhar para o destino.

### • Estrutura da Aplicação:

No diagrama a baixo está representado a estrutura da aplicação. A aplicação é iniciada pelo Launcher que inicializa o motor RETE e o *FuzzyRETE* (Fuzzy), a Base de Conhecimento, onde se encontram as



regras, e ainda a interface gráfica (GUI). Na GUI é possível utilizar o Assertor para inserir os factos que, após processados entrarão na Memória de Trabalho. Com a Base de Conhecimento e Memória de Trabalho carregados no sistema, é Função do RETE avaliar e dispara as regras e,



dependendo das regras disparadas, elas serão tratadas pelo Tratamento de Eventos ( *EventHandler*) e alguma acção extra será acionada.

**Legenda:** Diagrama da aplicação

### • Detalhes da Implementação:

A implementação dos conceitos difusos no nosso sistema pericial requereu de código e bibliotecas externas ao *Jess* mas que o usam. Utilizou-

se a biblioteca *fuzzyJ* para tal efeito. Alguns métodos desta biblioteca estão *depercated* visto que esta biblioteca é um projecto canadiano que deixou de ser actualizado, tendo algumas funcionalidades por implementar. Esta biblioteca encontra-se num repositório público, indicado na bibliografia.

Há a implementação de tratadores (*handlers*) em que o objectivo destes é comunicar com APIs. São usadas as APIs do *Facebook* para publicar algumas mensagens no estado pessoal do utilizador e do email do *java* para este ser notificado na sua caixa de correio de alguns acontecimentos disparados pelas regras.

## Experiências

Sendo o nosso projecto um sistema pericial, as experiências efectuadas a este relacionam-se com o teste das regras e verificação e validação das conclusões tiradas pelas regras. Através da inserção de factos na memória de trabalho do nosso sistema, este tira as conclusões devidas: se o sistema percebe que o tempo está quente, é de manhã e está sol, este notifica que as janelas vão ser abertas e comunica no estado do *Facebook* do utilizador que «Vai Estar Um Dia Quente». Se a altura do dia for noite, o sistema notifica que vai fechar as janelas. Sujeitando o nosso sistema a este tipo de experiências pudemos deduzir que ele está preparado quer para comunicar com dispositivos externos a ele quer para tirar conclusões como um perito.

Chama-se à atenção de que a nossa aplicação como não é oficial nem é registada, isto é, chamada aplicação *developer*, tem alguns problemas no que diz respeito ao acesso a determinados conteúdos no *Facebook* e ao fazer chamadas muitas vezes seguidas, pode conduzir a banimentos temporários.

## Conclusões

O conceito de *Internet of Things* é muito vasto e relaciona muitos aspectos que permitem facilidade na vida quotidiana.

Desenvolver um sistema pericial não é fácil. Requer bastante pesquisa no que diz respeito aos conceitos e variáveis que o sistema vai ter em conta. No nosso caso, tivemos em conta algumas variáveis meteorológicas com o objectivo de avisar e atuar em alguns equipamentos dentro de uma casa e contas pessoais do utilizador – email e *Facebook*.

Ao longo do desenvolvimento deste projecto deparamo-nos com algumas dificuldades, nomeadamente, na informação útil sobre o *Jess* e mais acentuada no *FuzzyJess*. Para além disto, a informação para *developers* de *java* nas APIs da *Google+* e *Facebook*, não são muito intuitivas e fáceis de utilizar.

## Melhoramentos

De forma a melhorar o nosso projeto, um dos primeiros aspectos seria o nosso sistema comunicar com mais APIs, e serviços *web*. Outra possibilidade de melhoria seria, através da análise de mais variáveis, o sistema tirar mais conclusões.

## Recursos

Os recursos que temos utilizado dividem-se em dois tipos:

- **Bibliográficos:**

Sítios para aprofundar conhecimentos:

[http://en.wikipedia.org/wiki/Internet\\_of\\_Things#Applications](http://en.wikipedia.org/wiki/Internet_of_Things#Applications)

<http://whatis.techtarget.com/definition/Internet-of-Things>

[https://web.fe.up.pt/~eol/IA/1415/APONTAMENTOS/5\\_Ln.pdf](https://web.fe.up.pt/~eol/IA/1415/APONTAMENTOS/5_Ln.pdf)

[https://web.fe.up.pt/~eol/IA/1415/APONTAMENTOS/4\\_SP.pdf](https://web.fe.up.pt/~eol/IA/1415/APONTAMENTOS/4_SP.pdf)

[https://web.fe.up.pt/~eol/IA/1415/APONTAMENTOS/3b\\_RI%20.pdf](https://web.fe.up.pt/~eol/IA/1415/APONTAMENTOS/3b_RI%20.pdf)

<https://eden.dei.uc.pt/~ec/teoricas/Isp1.pdf>

[http://en.wikipedia.org/wiki/Expert\\_system](http://en.wikipedia.org/wiki/Expert_system)

<https://ifttt.com/products>

- **Software:**

O software utilizado é o *Jess*, na versão 7.1. Para aprofundar estes conhecimentos recorreremos aos seguintes sítios:

<http://paginas.fe.up.pt/~eol/AIAD/aulas/jess.pdf>

<http://www.jessrules.com/>

<http://herzberg.ca.sandia.gov/docs/70/api/jess/>

[http://ctat.pact.cs.cmu.edu/docs/ctat\\_2\\_1/writing\\_jess\\_rules.html](http://ctat.pact.cs.cmu.edu/docs/ctat_2_1/writing_jess_rules.html)

E ao Manual de tutorial:

*Jess Tutorial*, Maarten Menken, from Vrije Universiteit, Amsterdam, The Netherlands, 24 dezembro 2002, contacto: mrmenkens.vu.nl.

Objetos de Consulta sobre o *FuzzyJess* (instalação, código e documentação):

[http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/isa/2010\\_1/APIdocs/nrc/fuzzy/jess/](http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/isa/2010_1/APIdocs/nrc/fuzzy/jess/)

<http://rorchard.github.io/FuzzyJ/>

[http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/isa/2010\\_1/fuzzyJDocs/FuzzyJess.html](http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/isa/2010_1/fuzzyJDocs/FuzzyJess.html)

<http://www.csie.ntu.edu.tw/~sylee/courses/FuzzyJ/FuzzyJess.htm>

<https://github.com/rorchard/FuzzyJ>

API do *Facebook*:

<https://developers.facebook.com/tools/explorer>

API de emails do *java*:

<http://www.oracle.com/technetwork/java/javamail/index.html>

- **Esforço de cada elemento do grupo:**

André: 40%

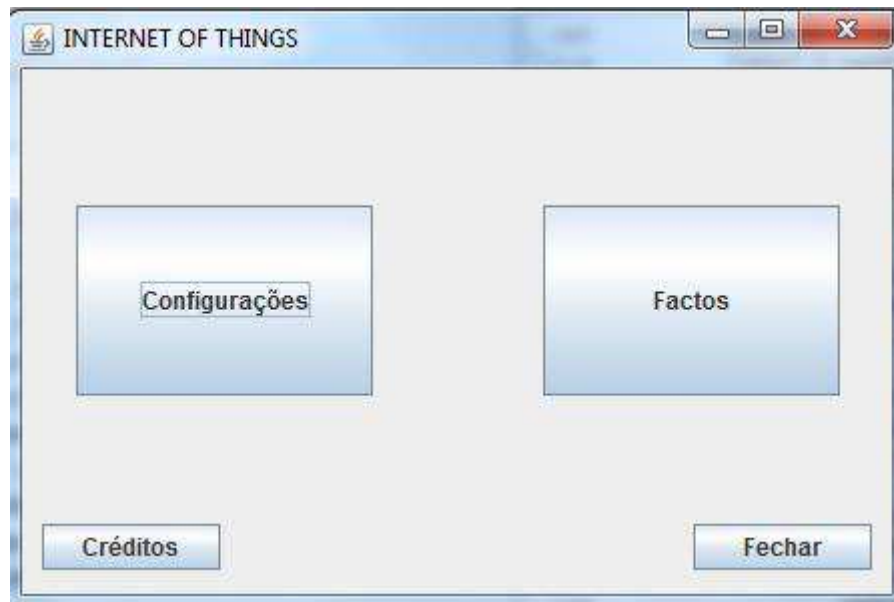
João: 40%

Maria: 20%

# Apêndice

## Manual de Utilização:

Ao  
inicializar  
programa,  
aparecerá a  
seguinte  
interface  
gráfica:



0

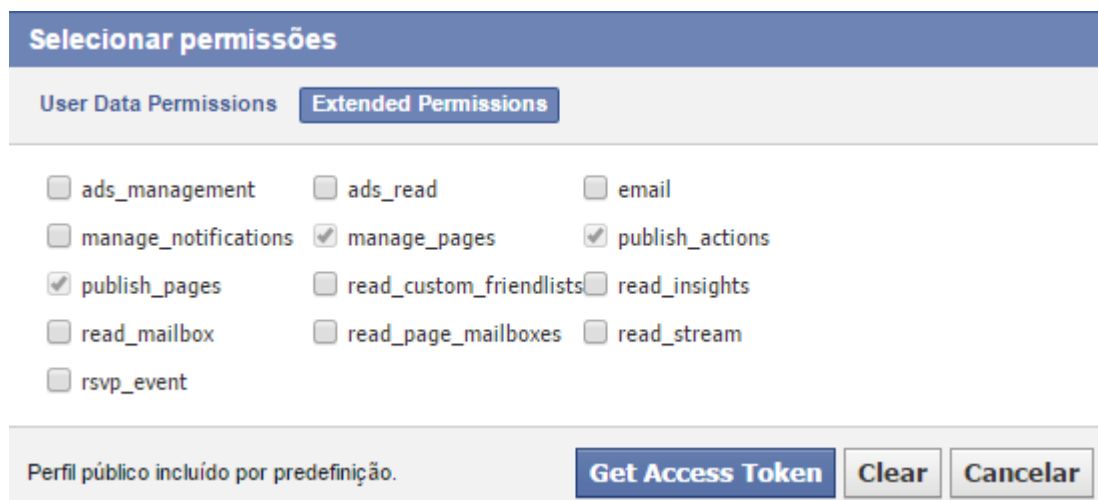
### **Legenda:** Interface Inicial

Nesta interface o utilizador deve primeiro fazer algumas configurações.  
Ao clicar em «Configurações», aparecerá a seguinte interface gráfica:



### **Legenda:** Interface das Configurações

Nesta interface o utilizador poderá carregar as informações existentes no ficheiro “configs.txt” ou inserir o valor nos campos sugeridos e actualizar esse ficheiro. Chama-se especial atenção para o campo «Token do Facebook». Para obter este token, o utilizador deve abrir este link, <https://developers.facebook.com/tools/explorer>, clicar em «Get Token» >



**Selecionar permissões**

User Data Permissions   **Extended Permissions**

☐ ads\_management   ☐ ads\_read   ☐ email

☐ manage\_notifications   ☒ manage\_pages   ☒ publish\_actions

☒ publish\_pages   ☐ read\_custom\_friendlists   ☐ read\_insights

☐ read\_mailbox   ☐ read\_page\_mailboxes   ☐ read\_stream

☐ rsvp\_event

Perfil público incluído por predefinição.   **Get Access Token**   Clear   Cancelar

«Get Acesso Token» > «Extended Permission» e, por fim seleccionar as seguintes opções da lista apresentada:



**Configurações**

Credenciais Sifeup

Email Notificação   Token do Facebook

Sifeup

Senha

Cancelar   Carregar   Guardar

**Legenda:** Permissões permitidas.

Com



The screenshot shows a Java Swing window titled "Menu Factos". Inside the window, the title "Estado Meteriológico" is centered at the top. Below the title, there are four columns of buttons representing different meteorological conditions: "Tempo", "Intensidade Tempo", "Temperatura", and "Momento Dia". The "Tempo" column has buttons for "Sol" and "Chuva". The "Intensidade Tempo" column has buttons for "Alta" and "Baixa". The "Temperatura" column has buttons for "Quente", "Ameno", and "Frio". The "Momento Dia" column has buttons for "Manhã", "Tarde", and "Noite". At the bottom right of the window, there are two buttons: "Cancel" and "Submit".

as

permissões seleccionadas, clicar em «Get Access Token». Na queixa de texto «Access Token», copiar o conteúdo daí e colocar no campo da nossa interface gráfica e clicar em «Guardar». Estas configurações servem para introduzir a comunicação entre a API de emails do *java* e a API do *Facebook* pois estas requerem autenticação e alguns privilégios de acesso.

Ao clicar em «Guardar» o utilizador volta para o Menu Inicial e está pronto para usar o nosso sistema pericial. Clicando no botão «Factos», aparecerá a seguinte GUI:

### **Legenda:** Menu dos Factos

Neste menu o utilizador indica as condições meteorológicas disponíveis pelos botões e ao clicar em «Submit», aparecerá na consola as conclusões retiradas.