Universidade do Minho
Mestrado em Engenharia Informática

Perfil Sistemas Inteligentes
Agentes Inteligentes

Arquiteturas Orientadas a Serviços

Gestão de um Ambiente Inteligente baseado em Tecnologia OSGi

janeiro 2015

Figura 1: Ana Margarida Ferreira Cruz, PG27747

Figura 2: Isabel Maria Ferreira Cruz, PG27746

Figura 3: Serafim Miguel da Costa Pinto, PG28506

Resumo

No âmbito da Unidade Curricular de Agentes Inteligentes, do Perfil Sistemas Inteligentes do Mestrado em Engenharia Informática, este relatório pretende apresentar o Projeto de Agentes e Sistemas Multiagentes, adotando uma Arquitetura Orientada a Serviços - OSGi, bem como as decisões que foram tomadas e as suas dificuldades. Este trabalho tem como objetivo principal desenvolver um sistema inteligente capaz de controlar o ambiente de uma casa ou local de trabalho, através de serviços e sensores (fictícios).

Conteúdo

1 Introdução **3**
- 1.1 Contextualização . . . 3
- 1.2 Caso de Estudo . . . 3
- 1.3 Objetivos . . . 4
- 1.4 Ferramentas . . . 4
- 1.5 Estrutura do Relatório . . . 4

2 Desenvolvimento **6**
- 2.1 Arquitetura Orientada ao Serviço . . . 6
- 2.2 Framework OSGi . . . 6
  - 2.2.1 Bundles . . . 7
  - 2.2.2 Serviços . . . 7
- 2.3 Sensores . . . 7
- 2.4 Bundles Implementados . . . 8
  - 2.4.1 TempHum . . . 8
  - 2.4.2 Luminosidade . . . 8
  - 2.4.3 WeatherStation . . . 8
  - 2.4.4 Actuador . . . 8
  - 2.4.5 Leituras . . . 9
  - 2.4.6 SensorAlarm . . . 9
  - 2.4.7 HouseManager . . . 9
- 2.5 Interação com o Utilizador . . . 10
  - 2.5.1 Janela Principal . . . 10
  - 2.5.2 Limites dos Sensores . . . 11
  - 2.5.3 Leituras dos Sensores . . . 11

3 Notas Finais **12**
- 3.1 Conclusões . . . 12

Lista de Figuras

1 Ana Margarida Ferreira Cruz, PG27747 2
2 Isabel Maria Ferreira Cruz, PG27746 2
3 Serafim Miguel da Costa Pinto, PG28506 2

2.1 Janela principal "*House Manager*" 10
2.2 Menu "Opções" 10
2.3 Janela - Limites dos Sensores 11
2.4 Janela - Leituras dos Sensores 11

Capítulo 1

Introdução

1.1 Contextualização

Atualmente, a autonomia, o conforto, e o bem-estar são atributos essenciais para o bem-estar das pessoas. No entanto, nem todas as casas ou locais de trabalho têm as condições necessárias para esses atributos serem satisfeitos, e por isso este projeto tem por objetivo contribuir para um melhoramento da qualidade de vida das pessoas.

É sobre esta temática que incide este trabalho prático. Trata-se de um sistema funcional capaz de integrar serviços que em tempo real recolhem dados (temperatura, humidade e luminosidade) de um ambiente através de sensores. De modo ao utilizador ter a possibilidade de gerir esse ambiente através dos atuadores disponíveis (ar condicionado, desumidificador e lâmpada).

1.2 Caso de Estudo

O trabalho prático consiste no desenvolvimento de um sistema distribuído para a gestão de um ambiente inteligente, nomeadamente no que diz respeito à sua temperatura, humidade e luminosidade.

A concretização deste projeto pressupõe a realização de um conjunto de etapas:

1. Analisar os dados fornecidos pela equipa docente e os objetivos a atingir, bem como perceber os principais conceitos que estão por base nas Arquiteturas Orientadas a Serviços. Foi fornecido aos alunos um projeto designado "*Environment*" que contém o simulador do ambiente, que gera os valores dos sensores de temperatura, humidade e luminosidade, internos e externos. Simula ainda os atuadores, com os quais se pode interagir ligando e desligando-os, bem como aumentando e diminuindo a sua intensidade. Também foi fornecido aos alunos o início de uma pequena interface para posteriormente ser desenvolvida pelos mesmos, para ser usada pelo

utilizador, e por fim um exemplo de um serviço que dá acesso o valor de um sensor (neste caso a temperatura). Os objetivos do trabalho serão mencionados e descritos mais à frente;

2. Desenvolver os serviços necessários que serão capazes de interagir com os sensores e os atuadores no ambiente, isto é, obter os valores lidos pelos sensores, bem como poder controlar o ambiente através dos atuadores presentes. Adicionalmente, implementar um sistema de notificações quando os sensores se encontram fora dos valores limite definidos pelo utilizador, e também uma parte mais estatística com as leituras dos sensores;

3. Construção da interface gráfica, de forma a conseguir a aplicação inteiramente funcional com o utilizador.

1.3 Objetivos

Como foi mencionado anteriormente, o foco deste projeto passa por desenvolver um sistema que seja capaz de ser monitorizado por um utilizador, podendo este controlar um ambiente. Tendo em conta os requisitos deste projeto, podem ser definidos os seguintes objetivos principais a atingir:

- Conhecer os principais conceitos que definem o paradigma das Arquiteturas Orientadas a Serviços;
- Implementar diferentes módulos independentes entre si, com funcionalidades bem definidas;
- Construir um sistema de maior dimensão assente na orquestração de serviços mais simples.

1.4 Ferramentas

Para desenvolver o sistema foi necessário instalar o conjunto de especificações OSGi[1]. Para desenvolver o código do trabalho prático foi necessário instalar o *software* Eclipse[2]. Este relatório foi desenvolvido recorrendo ao LaTeX.

1.5 Estrutura do Relatório

Numa primeira fase, é apresentado no relatório o Resumo deste trabalho prático, onde é descrito o problema e a justificação da sua solução.

De seguida é apresentada a Introdução, onde é exposto o caso de estudo e onde é feita uma pequena contextualização do problema, bem como os objetivos a alcançar com este trabalho prático.

[1] http://www.knopflerfish.org/download.html
[2] https://eclipse.org

Seguidamente surge o Desenvolvimento onde serão apresentados os capítulos que descrevem todo o processo para a aplicação final. Com grande foco na tecnologia OSGi, explicando os seus conceitos necessários para a compreensão do relatório. Nestes capítulos são abordados os diversos tipos de sensores e os atuadores presentes no ambiente.

Posteriormente é apresentada a Conclusão onde o grupo faz uma apreciação crítica sobre o trabalho prático, apontando os seus pontos fortes e fracos.

Por fim a Bibliografia é apresentada após a Conclusão, e nela é apresentada a lista das fontes bibliográficas consultadas durante a realização deste trabalho.

Capítulo 2

Desenvolvimento

2.1 Arquitetura Orientada ao Serviço

Service-Oriented Architecture (SOA) é uma arquitetura de *software* cujo princípio fundamental prega que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços.

A característica fundamental de um SOA é de que existem serviços com interfaces definidas que podem ser executadas sem conhecer quem as executa. Cada componente no sistema dá a conhecer os serviços que disponibiliza ao mesmo tempo que encontra serviços que outros componentes disponibilizam na mesma forma.

SOA pode assim resolver algumas dificuldades. Os dispositivos dão a conhecer os seus serviços com a sua descrição e num formato normalizado. A finalidade é de que se dois dispositivos utilizam tecnologias diferentes, eles continuam a entender como executar os serviços um do outro. Como na arquitetura os componentes são autónomos, o sistema torna-se escalável e mais flexível, pois podem trabalhar independentemente com pouca ou nenhuma configuração do utilizador.

2.2 Framework OSGi

A *framework* OSGi executa sobre a linguagem de programação Java, e tem uma arquitetura orientada a serviços. Na *framework* é proposto a utilização de componentes de execução (*bundles*) que podem ser instalados, desinstalados e atualizados de um modo dinâmico e escalável.

Um *bundle*, para além de disponibilizar serviços, pode obter serviços de outro *bundle* através do registo de serviços oferecidos pela *framework*. Da mesma forma, pode também exportar e importar classes Java. Existe um controlo por parte da *framework* que gere dependências entre *bundles* e serviços, proporcionando aos *bundles* um correto funcionamento. A *framework* separa a implementação de serviços da sua especificação, permitindo assim aos programadores

uma maior independência no desenvolvimento dos serviços.

Uma das características mais importantes da *framework* é a de permitir a execução de *bundles* em tempo real através da *framework* de registo de serviços. Esta característica é bastante vantajosa pois o utilizador não necessita de reiniciar nem configurar o sistema após instalar novos *bundles*, novos serviços e atualizar *bundles* já existentes.

2.2.1 Bundles

Um *bundle* é uma unidade funcional com um ciclo de vida e com capacidade de carregar classes. Também contém bem definidos os métodos que lhe permitem estar ligado à *framework*. Trata-se de um arquivo JAR que contém as classes e os recursos que podem ser imagens, páginas HTML, entre outros. Como já foi referido, um *bundle* contém um conjunto de serviços.

Cada *bundle* contém um padrão de interface já definido, de forma a possibilitar a *framework* de entender e instalar o *bundle*. O "manifesto" é um ficheiro padrão de entrada no arquivo JAR. Este inclui meta-informações sobre o ficheiro, de forma padronizada, com o objetivo da *framework* ser capaz de processar o *bundle* corretamente.

O "ativador do *bundle*" é executado apenas quando o *bundle* é iniciado ou parado, e implementa os métodos *start* e *stop*. Quando o *bundle* é inicializado a *framework* analisa o cabeçalho do *manifesto*. Se o *bundle* necessita de importar pacotes, a *framework* procura o pacote em todos os *bundles* com o estado de "resolvido" e de seguida exporta os seus pacotes para a *framework*. É no método *start* que o *bundle* deve registar todos os serviços que presta. O *bundle* entra no estado de iniciado e, em caso de sucesso da ativação, o *bundle* passa para o estado de "ativo", e um outro evento notifica os observadores interessados de que o *bundle* foi iniciado.

2.2.2 Serviços

Para implementar um serviço, é necessário definir uma interface que diz o que o serviço faz, e as correspondentes classes que definem como o serviço é realmente executado. Existe uma separação da interface e da implementação, garantindo assim que a interface do serviço se mantenha estável, mesmo que a implementação sofra alterações.

2.3 Sensores

Neste trabalho prático, como já foi referido anteriormente, os sensores são fictícios, pelo que é a classe *Environment* que contém o motor de simulação dos sensores. O grupo não utilizou sensores reais, mas os que foram utilizados neste trabalho prático são:

- Temperatura - valor da temperatura no interior e exterior da casa;

- Humidade - valor da humidade no interior e exterior da casa;
- Luminosidade - valor da luminosidade no interior e exterior da casa;
- Janela - indica se a janela está ou não fechada.

2.4 Bundles Implementados

Nesta secção serão citados e descritos todos os *bundles* desenvolvidos neste trabalho prático.

2.4.1 TempHum

Este *bundle* fornece um serviço que acede ao *Environment* e devolve os valores da temperatura e humidade no interior da casa. Quando inicializado devolve a seguinte mensagem:

```
TempHum Service is starting!
Service registered: TempHumService
```

2.4.2 Luminosidade

Este *bundle* fornece um serviço que acede aos valores do sensor da luminosidade no interior da casa. Quando inicializado devolve a seguinte mensagem:

```
Luminosidade Service is starting!
Service registered: LuminosidadeService
```

2.4.3 WeatherStation

Como o próprio nome indica este *bundle* oferece um serviço que obtém todos os valores dos sensores exteriores da casa - temperatura, humidade e luminosidade. Quando inicializado devolve a seguinte mensagem:

```
WeatherStation Service is starting!
Service registered: WeatherStationService
```

2.4.4 Actuador

Os atuadores do sistema são:

- Ar Condicionado
- Desumidificador
- Lâmpada
- Janela

Este *bundle* concede um serviço que é aquele que faz a gestão do ambiente, isto é, além de obter o estado dos atuadores, pode interagir ligando e desligando-os, bem como aumentando ou diminuindo a sua intensidade. A interação com os atuadores reflete-se em alterações nos valores lidos nos sensores, como seria de esperar. Quando é inicializado devolve a seguinte mensagem:

```
Actuador Service is starting!
Service registered: ActuadorService
```

2.4.5 Leituras

Cada *bundle* anterior tem implementado uma memória interna das últimas dez leituras dos sensores. Desta forma, o *bundle* "Leituras" fornece um serviço que devolve o valor médio dessas mesmas leituras.

```
Leituras Service is starting!
Service registered: LeiturasService
```

2.4.6 SensorAlarm

Este *bundle* contém um serviço que é usado para definir o limites mínimo e o máximo de todos os sensores. Também, conforme a configuração do utilizador, devolve o estado do alarme, ou seja, se as condições do ambiente diferem das definições estabelecidas ou não.

```
SensorAlam Service is starting!
Service registered: SensorAlarmService
```

2.4.7 HouseManager

Este *bundle*, define todo o interface gráfico, que permite visualizar o estado dos sensores interiores e exteriores bem como dos atuadores, e interagir com eles. Também, define as notificações para o utilizador, possibilita a definição dos limites dos sensores e a consulta das leituras dos mesmos.

2.5 Interação com o Utilizador

2.5.1 Janela Principal

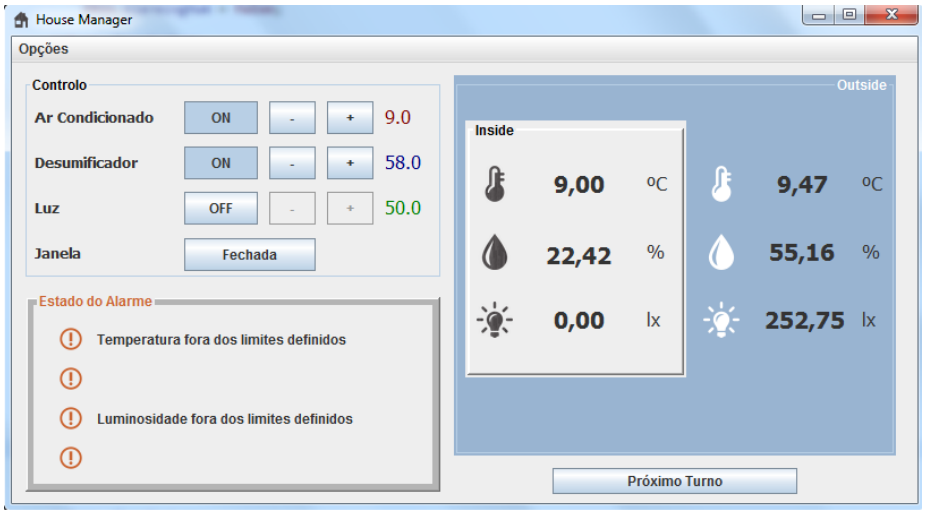

Figura 2.1: Janela principal "*House Manager*"

Como se pode verificar na figura 2.1, a aplicação "*House Manager*" exibe o valor dos sensores interiores e exteriores.

No canto superior esquerdo é possível ver os atuadores com os seus estados de ligados ou desligados, bem como as suas intensidades.

No lado inferior esquerdo são feitas as notificações ao utilizador de acordo com as suas configurações.

O botão "Próximo Turno" permite avançar a simulação para o turno seguinte.

Por fim, também é possível verificar a existência de um menu "Opções" (Figura 2.2).

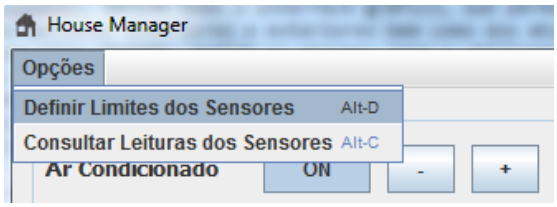

Figura 2.2: Menu "Opções"

2.5.2 Limites dos Sensores

Figura 2.3: Janela - Limites dos Sensores

Nesta janela (Figura 2.3) o utilizador pode definir para todos os sensores os valores limite que definem um intervalo. Também pode definir se deseja a janela fechada ou aberta.

2.5.3 Leituras dos Sensores

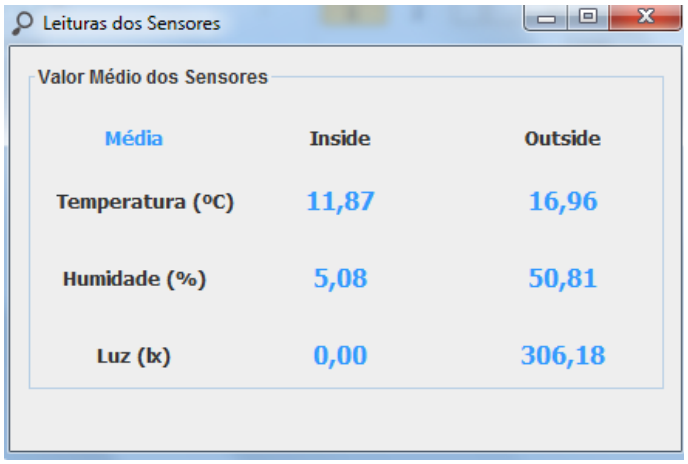

Figura 2.4: Janela - Leituras dos Sensores

É através da janela acima (Figura 2.4) que o utilizador pode consultar a média das últimas dez leituras feitas pelos sensores.

Capítulo 3

Notas Finais

3.1 Conclusões

Com este trabalho prático ficaram retidos os principais conceitos que definem o paradigma das Arquiteturas Orientadas a Serviços. Ficou-se a conhecer e a dominar a tecnologia OSGi. Percebeu-se que ao usar esta *framework* podem-se implementar diferentes módulos independentes entre si, com funcionalidades bem definidas, e até, em dispositivos diferentes. OSGi são especificações com foco na tecnologia Java que permitem a instalação de módulos em tempo real. Esta plataforma permite a integração de serviços totalmente independentes, se um deixar de funcionar não irá prejudicar em nada o funcionamento do sistema.

Outro aspeto importante a salientar, e foi algo que nos deu alguns problemas, foi o facto de em determinada altura do trabalho não ser possível manter a consistência dos dados, isto é, cada vez que era feita uma nova simulação, os novos valores gerados pelos sensores eram obtidos sem ter em conta as alterações efetuadas no *Environment*. Isto porque cada *bundle* tinha o seu próprio *Environment*, e não era de todo isso o desejado. Depois, através da ajuda da equipa docente, e com algum trabalho de pesquisa, foi possível solucionar este problema. E portanto, existe apenas um *Environment*, que é sempre o mesmo para todos os *bundles* durante a execução da aplicação.

Depois de uma avaliação ponderada, o grupo considerou que os requisitos pedidos foram cumpridos. O grupo acredita que este projeto, para além de ser viável, é também uma excelente oportunidade de criar algo com alguma importância e proveito para as pessoas no futuro e, por isso, o grupo está satisfeito com o trabalho realizado.

Bibliografia

[1] - Descrição de Arquiteturas Orientadas a Serviços:
`http://pt.wikipedia.org/wiki/Service-oriented_architecture`

[2] - Descrição da tecnologia OSGi:
`http://pt.wikipedia.org/wiki/OSGi`

[3] - Artigos do *site* RepositóriUM da Universidade do Minho:
`http://repositorium.sdum.uminho.pt/handle/1822/19001`

[4] - Desenvolvimento OSGi com Knopflerfish:
`http://robertgreiner.com/2010/02/osgi-development-with-knopflerfish-part1-the-setup/`

[5] - *Site* da tecnologia Knoplerfish:
`http://www.knopflerfish.org/`

[6] - Knopflerfish no *site* da Wikipédia:
`http://en.wikipedia.org/wiki/Knopflerfish`

[7] - OSGi no *site* da Wikipédia:
`http://pt.wikipedia.org/wiki/OSGi`