BlueStation 2
Reengenharia de um sistema de disseminação de mensagens

PROJETO DE MESTRADO

António Luís da Silva Gonçalves
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

agosto | 2014

BlueStation 2
Reengenharia de um sistema de disseminação de mensagens

PROJETO DE MESTRADO

António Luís da Silva Gonçalves
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
Leonel Domingos Telo Nóbrega

CO-ORIENTADOR
Luís Armando de Aguiar Oliveira Gomes

AGRADECIMENTOS

Ao longo de cinco anos muitas foram as pessoas que fizeram parte do meu percurso académico. Todas com o seu contributo seja direto ou indireto, com as suas críticas negativas ou positivas, fizeram-me lutar, crescer enquanto homem e nunca desistir dos meus sonhos, nomeadamente os relacionados com o meu futuro académico e profissional, mesmo quando pareciam difíceis e se avizinhavam de muito sacrifício.

O meu primeiro agradecimento é direcionado ao meu orientador Prof. Leonel Nóbrega, pois sem este grande senhor nada do trabalho desenvolvido seria possível. Graças à sua notável intelectualidade e experiência na área da programação, sempre me orientou para os caminhos que deveria seguir e com a sua preciosa ajuda pude concluir com sucesso os objetivos inicialmente propostos. Espero por em prática na minha vida futura, tanto os conhecimentos transmitidos por este grande senhor, bem como o seu exemplo de vida profissional, que mesmo com a quantidade de compromissos que tinha, estava sempre com um humor de causar inveja e que muitas vezes me contagiava, me motivava a nunca desistir e me fazia acreditar que tudo é possível quando queremos muito que o seja. O meu muito obrigado por tudo!

O meu segundo agradecimento é direcionado aos senhores Claudio Mantero, Pedro Vieira ambos dos Horários do Funchal, ao Prof. Luís Gomes e ao Tiago Camacho, ex-aluno da Universidade da Madeira, pois o trabalho a que dei continuidade foi iniciado por ele. O meu obrigado ao Sr. Claudio Mantero e ao Sr. Pedro Vieira pela sua prestabilidade, pelo interesse, pelo contributo prestado, por me tratarem sempre da melhor maneira e por serem pessoas que levo comigo com enorme carinho e amizade. O meu grande obrigado ao Tiago Camacho que desde o início sempre esteve disposto ajudar-me em todas as dúvidas, em explicar-me como o sistema estava implementado, mesmo quando o fuso horário era totalmente o oposto do meu. O meu muito obrigado ao Prof. Luís Gomes, pela sua prestabilidade, pelas suas ideias sempre construtivas e por estar sempre disposto a ajudar. O meu grande obrigado a todos!

Todo o trabalho realizado nesta jornada, não é apenas as inúmeras noites de pouco sono, as páginas escritas com imensos caracteres e as linhas de código que fizeram com que o objetivo final da minha dissertação fosse valorizado, reconhecido e cumprisse com o objetivo inicialmente proposto. Mas sim as palavras de quem eu amo, meu pai António, minha mãe Maria, minhas irmãs Verónica e Carina, e a minha namorada Fátima, que não ficam escritas em folhas de papel, mas sim escritas nas minhas recordações e tatuadas no meu coração. As linhas de código que apenas o meu "processador" sabe interpretar e que encontram o caminho do meu coração foram eles que escreveram, que tornaram este momento possível, porque sem eles nada disto teria sentido. Obrigado por tudo o que significam e continuam a significar para mim!

ABSTRACT

Many companies that are on the market for long time encounter problems in the use and maintenance of software systems. Many of these systems are made to run on a wide range of hardware and programmed in obsolete languages. Given these factors companies have three options:

1) Run the software in the disorganized state that is in and have increasingly high updating costs.

2) Recreate the software from scratch.

3) Conduct a reengineering of the software and thus implement a more current paradigm with or without changing the programming language.

Reverse engineering and/or reengineering is the way that many companies seek to evolve their software systems, avoiding large costs and difficult maintenance. Reengineering allows us recreate the existing system, with improvements in quality, maintenance, documentation and functionality.

At Horários do Funchal a distributed system is used, composed of a server and Bluetooth stations (Bluestation), which are used to diffuse content to customers/commuters who are at the bus stops that they use. Over the years the system has revealed some instabilities and limitations, which made them obsolete and unable to be used by the company. Due to these factors this master thesis aimed to do a reverse reengineering of the distributed system. Reengineering went through:

- Analyze all logical and physical components of the system (BlueStations and server);
- Find and fix flaws the system had;
- Optimize the performance of the whole system;
- Develop a new application that allow to interact with BlueStations and make possible the creation and management services.
- Solve usability problems, instability and lack of features that the current application presented.

So that at the end of this dissertation all the work done would put the system back in operation, with new quality attributes and answer the needs that the company intends for it.

KEYWORDS

Web Application

CORBA Architecture

Bluetooth

Reverse Engineering

REST Service

Distributed System

PALAVRAS CHAVE

Aplicação Web

Arquitetura CORBA

Bluetooth

Engenharia Reversa

Serviço REST

Sistema Distribuído

RESUMO

Muitas empresas que se encontram no mercado há já muito tempo deparam-se com problemas no uso e manutenção de sistemas de *software*. Muitos desses sistemas são construídos para serem executados numa gama diversificada de *hardware* e programados em linguagens obsoletas. Perante estes fatores as empresas têm três opções:

1) Manter o *software* na situação de desorganização em que se encontra e terem custos cada vez mais elevados para a sua atualização.

2) Reconstruir de raiz o *software*.

3) Realizar uma engenharia reversa sobre o *software* e assim implementar um paradigma mais atual com ou sem mudança de linguagem de programação.

A engenharia reversa é a forma que muitas empresas procuram para evoluir os seus sistemas de *software*, evitando manutenções difíceis e com grande custo. A engenharia reversa permite recriar o sistema existente, apresentando melhorias em termos de qualidade, manutenção, documentação e de funcionalidades.

Nos Horários do Funchal é utilizado um sistema distribuído, composto por um servidor e estações Bluetooth (BlueStation), que são utilizadas para a disseminação de conteúdos aos clientes que se encontram nas paragens dos autocarros. Ao longo dos anos o sistema tem revelado algumas instabilidades e limitações, que tornaram-no obsoleto e incapaz de ser utilizado pela empresa. Devido a estes fatores este trabalho de mestrado teve como principal objetivo fazer uma engenharia reversa ao sistema distribuído. Está engenharia passou por:

- Analisar todos os componentes lógicos e físicos do sistema (BlueStations e Servidor);
- Procurar e corrigir falhas que o sistema apresentava;
- Otimizar o desempenho de todo o sistema;
- Desenvolver uma nova aplicação que permitisse interagir com as BlueStations e possibilitasse a criação e gestão de serviços.
- Resolver os problemas de usabilidade, instabilidades e a falta de funcionalidades que aplicação atual apresentava.

para que no final desta dissertação todo o trabalho desenvolvido colocasse o sistema de novo em funcionamento, com novos atributos de qualidade e respondesse as necessidades que a empresa pretende para o mesmo.

TABELA DE CONTEÚDOS

Agradecimentos I
Abstract II
Keywords III
Palavras Chave IV
Resumo V
Tabela de Conteúdos VI
Lista de Figuras IX
Lista de Tabelas XI
Acrónimos XII
1. Introdução 1
1.1. Motivação 2
1.2. Objetivos 3
1.3. Desafios 4
1.4. Contribuição 6
1.5. Organização 7
2. Estado da Arte 9
2.1. Tecnologia Bluetooth 9
2.1.1. Versões da Especificação Bluetooth 9
2.1.2. Classes Bluetooth 10
2.1.3. Visibilidade 10
2.1.4. Descoberta de Dispositivos 11
2.1.5. Perfis Bluetooth 13
2.1.6. Busca de Serviços 14
2.1.7. Emparelhamento de Dispositivos 14
2.1.8. Protocolos Bluetooth 16
2.1.9. Topologias Bluetooth 18
2.1.10. Zonas Bluetooth 19
2.2. Arquitetura CORBA 20
2.2.1. Serviços 22
2.3. REST (*Representational State Transfer*) 25
2.3.1. Elementos Arquiteturais do Serviço REST 25
2.3.2. Exemplo de um serviço Web utilizando REST 28
3. Visão Geral do Sistema BlueStation 31
3.1. Arquitetura de Alto Nível do Sistema BlueStation 34

3.1.1. Componente BlueStation (Estação) 34
3.1.2. Broker 42
3.2.3 Unidade Central 43
4. BlueStation - Aplicação Consola de Administração 45
4.1. Fundamentação Tecnológica e Funcionalidades 45
4.1.1. Comandos Gerais 46
4.1.2. Comandos de Gestão dos Serviços 47
5. Análise do Problema 53
5.1. Participation Map 53
5.2. Diagrama de Interação 55
5.3. Fase de Análise e de *Debug* da Consola de Administração (aplicação existente) 57
5.4. Requisitos 58
5.4.1. Requisitos Funcionais 58
5.4.2. Requisitos Não-Funcionais 59
5.5. Protótipos de Média Fidelidade 60
5.5.1. Primeira Fase de Prototipagem 60
5.5.2. Segunda Fase de Prototipagem 64
6. BlueStation - Aplicação Web 67
6.1. Aspetos de Implementação 67
6.1.1. Arquitetura Geral da Aplicação Web 68
6.2. Fundamentação Tecnológica e Funcionalidades 69
6.2.1. Elementos Comuns a Todas as *Interfaces* 69
6.2.1.1. Estado de Funcionamento das Estações 69
6.2.1.2. Menu de Navegação 71
6.2.2. TextService.html 71
6.2.3. FileService.html 73
6.2.4. ServiceList.html 75
6.2.5. AddAddressBL.html 76
6.2.6. ListAddressBL.html 77
7. Comparação das Aplicações 79
7.1. Requisitos Necessários para Executar cada uma das Aplicações 79
7.2. *Interface* e Usabilidade 79
7.3. Funcionalidades 81
8. Teste de Usabilidade 83
8.1. Preocupações 83
8.2. Público-Alvo e Tarefas 84

8.3. Avaliação de Usabilidade da Aplicação Web 85
8.4. Resumo dos Resultados 85
8.5. Conclusões e Alterações 88
9. Conclusões e Trabalho Futuro 89
Referências Bibliográficas 91

LISTA DE FIGURAS

Figura 1 – Bluetooth: Códigos de Acesso de Procura Geral (GIAC) 12
Figura 2 – Bluetooth: Códigos de Acesso de Procura Limitada (LIAC) 12
Figura 3 – Bluetooth: Pilha de Protocolos 16
Figura 4 – Topologias Bluetooth 19
Figura 5 – Comunicação entre Cliente CORBA e Servidor CORBA 21
Figura 6 - Arquitetura CORBA 21
Figura 7 – Exemplo REST: Visão geral do serviço Web 28
Figura 8 – Exemplo REST: Método GET 29
Figura 9 - Exemplo REST: Método PUT 29
Figura 10 - Exemplo REST: Método DELETE 30
Figura 11 – Esquema Geral do Sistema BlueStation 31
Figura 12 – BlueStation (Estação) 32
Figura 13 – Arquitetura de Alto Nível do Sistema 34
Figura 14 - BlueStation - Elementos Físicos 35
Figura 15 - Visão geral dos Componente de Software da BlueStation 36
Figura 16 - Elementos de Software e Funcionamento do Sistema de Procura 37
Figura 17 - Elementos de Software e Funcionamento do Módulo de Publicação 38
Figura 18 - Elementos de Software e Funcionamento do Agendamento de Serviços 39
Figura 19 - Elementos de Software e Funcionamento do Sistema de Entrega 40
Figura 20 – Elementos de Software e Funcionamento do Sistema de Comunicação 41
Figura 21 - Elementos de Software e Funcionamento do Sistema de Logs 42
Figura 22 - Elementos de Software e Funcionamento do Broker 42
Figura 23 – Elementos de Software e Funcionamento da Unidade Central 43
Figura 24 - Modelo MVC da Consola de Administração 45
Figura 25 - Interface da Consola de Administração 46
Figura 26 – Participation Map: Cenário para o Administrador 53
Figura 27 - Participation Map: Cenário para o Utilizador 54
Figura 28 - Diagrama de interação: Operações possíveis da consola de administração 55
Figura 29 - Diagrama de interação: Passos para criar serviços na consola de administração 56
Figura 30 - Protótipo de Média Fidelidade: Interface Home 61
Figura 31 - Consola de Administração: Interface para a criação de um serviço 61
Figura 32 – Protótipo de Média Fidelidade: Interface Criação de um Serviço 62
Figura 33 - Protótipo de Média Fidelidade: Antes e Depois - Interface Criação de um Serviço de Texto 64
Figura 34 - Protótipo de Média Fidelidade: Depois - Interface Criação de um Serviço de Ficheiro 65
Figura 35 - Arquitetura Geral da Aplicação Web 68
Figura 36 - Print-screen: Barra de estado que informa se a estação está online ou offline 70
Figura 37 - Print-screen: Menu de Navegação das Interfaces 71
Figura 38 – Print-screen: Interface TextService.html 72
Figura 39 – Ficheiros que compõem a interface TextService.html e suas funcionalidades 73
Figura 40 – Print-screen: Interface FileService.html 74
Figura 41 - Ficheiros que compõem a interface FileService.html e suas funcionalidades 75
Figura 42 - Print-screen: Interface ServiceList.html 75
Figura 43 – Ficheiros que compõem a interface ServiceList.html e suas funcionalidades 76
Figura 44 – Print-screen: Interface AddAddressBL.html 76

Figura 45 - Ficheiros que compõem a interface AddAddressBL.html e suas funcionalidades.... 77
Figura 46 - Print-screen: Interface ListAddressBL.html..... 77
Figura 47 - Ficheiros que compõem a interface ListAddressBL.html e suas funcionalidades..... 78
Figura 48 – Print-screen: Interface da Aplicação Consola de Administração 80
Figura 49 – Print-screen: Interface da Aplicação Web..... 81

LISTA DE TABELAS

Tabela 1 - Versões da especificação Bluetooth 9
Tabela 2 - Classes Bluetooth 10
Tabela 3 - Alguns perfis Bluetooth 13
Tabela 4 – Bluetooth: Alguns protocolos da camada middleware 18
Tabela 5 - Elementos de dados do serviço REST 26
Tabela 6 - Conectores do serviço REST 27
Tabela 7 - Métodos REST 28

ACRÓNIMOS

ACL - Assynchronous Connection-Less

AFH - Adaptive Frequency-Hopping

A2DP - Advanced Audio Distribution Profile

BPP - Basic Printing Profile

CORBA - Common Object Request Broker

CORS - Cross-Origin Resource Sharing

CSS - Cascading Style Sheets

DII - Dynamic Invocation Interface

DSI - Dynamic Skeleton Interface

DUN - Dial-Up Networking

EDR - Enhanced Data Rate

FIFO - First In, First Out

FTP - File Transfer Profile

GIAC - General Inquiry Access Code

GPS - Global Positioning System

HID - Human Interface Device

HTML - HyperText Markup Language

IDL - Interface Definition Language

IIOP - Internet Inter-Orb Protocol

IPC - Inter-process Communication

ISM - Industrial Scientific Medical

JS – JavaScript

JSON - JavaScript Object Notation

JSP - JavaServer Page

LIAC - Limited Inquiry Access Code

LMP - Link Manager Protocol

L2CAP - Logical Link Control Adaptation Protocol

MAC - Media Access Control

MVC - Model-View-Control

NFC - Near Field Communication

OBEX - Object Exchange Protocol

OMG - Object Manager Group

OOB - Out Of Band

OPP - Object Push Profile

ORB - Object Request Broker

PAN - Personal Area Networks

PPP - Point-to-Point Protocol

REST - Representational State Transfer

RFCOMM - Radio Frequency Communications

SCO - Synchronous Connection-Oriented

SDP - Service Discovery Protocol

SIG - Special Interest Group

SSP - Secure Simple Pairing

TCP/IP - Transmission Control Protocol/Internet Protocol

TCS - Telephony Control Protocol

UMTS - Universal Mobile Telecommunication System

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

XML - eXtensible Markup Language

1. INTRODUÇÃO

Cada vez mais as tecnologias sem fios estão presentes no quotidiano das pessoas, seja para o acesso à internet através de redes WiFi, seja para operações mais sensíveis como o pagamento de serviços através da tecnologia NFC (Near Field Communication) ou para uma simples troca de ficheiros utilizando a tecnologia Bluetooth. O objetivo é tirar o maior partido destas tecnologias e colocar ao dispor dos utilizadores serviços relevantes que satisfaçam as suas necessidades.

No caso da empresa Horários do Funchal, este tipo de tecnologia é utilizado num sistema composto por um conjunto de componentes distribuídos, para monitorizar a utilização dos transportes públicos e simultaneamente para enviar pequenas mensagens para os utentes desta rede. A componente principal que constitui o sistema é chamada de estação (ou também, BlueStation), que não é mais do que um computador a correr um sistema desenvolvido para a disseminação de conteúdos sensíveis ao contexto, utilizando a tecnologia sem fios Bluetooth. Estas estações estão instaladas em diferentes paragens de autocarros e têm como função fazer procuras contínuas por dispositivos Bluetooth ativos e visíveis, no raio de alcance da BlueStation. As estações usam o endereço MAC (Media Access Control) da placa Bluetooth do dispositivo encontrado para identificar os utentes, enviando a estes os serviços que se encontram configurados nas estações, e também fornecem à empresa todas as informações relacionadas com os serviços enviados, assim como a paragem que o utente utilizou. Estes serviços são criados através de uma aplicação programada na linguagem Perl, que tem como *interface* a linha de comandos, e interpreta um conjunto de comandos para desencadear ações. É através desta aplicação que é feita a comunicação com as estações. As BlueStation estão preparadas para suportar dois tipos de serviços: texto e imagem. A estes serviços podem ser associados um conjunto de restrições, tais como: se o serviço é destinado a todos ou a um conjunto específico de dispositivos Bluetooth e ainda permite definir em que momento o serviço deve ser executado e finalizado.

A parte essencial desta tese consiste em criar uma nova aplicação de gestão e criação de serviços que substitua a existente, assim como resolver as limitações e instabilidades que o sistema apresenta tanto a nível de diagnóstico, de funcionamento e de interação. Todos os componentes do sistema, nomeadamente as estações e as bases de dados que guardam todas as informações da atividade que ocorrem no sistema foram analisadas, com a finalidade de entender toda a arquitetura do sistema, para que o problema fosse analisado da melhor forma possível, satisfazendo um conjunto de atributos de qualidade que estavam em falta e para colocar todas as funcionalidades do sistema operacionais.

Este sistema distribuído fornece à Horários do Funchal um sistema com inúmeras possibilidades de utilização, a um baixo custo, e permite recolher dados de movimentação dos seus utentes, nomeadamente a paragem de entrada e a de saída. Alguns dos dados recolhidos (por exemplo: hora e data) dizem respeito ao momento em que o dispositivo Bluetooth do cliente foi detetado pela estação e estes dados podem ser cruzados com os dados das validações originadas pelo sistema eletrónico de bilhética e o sistema de GPS (Global Positioning System) dos autocarros, a fim de serem tomadas decisões de negócio relevantes para a empresa.

1.1. MOTIVAÇÃO

O desenvolvimento de um sistema é um processo que nunca é dado por concluído, havendo sempre trabalho por fazer, seja em termos de correções de problemas, introdução de novas funcionalidades, melhorias de desempenho ou de usabilidade. É de enorme importância que o sistema possua uma boa arquitetura, para que todas essas tarefas sejam efetuadas facilmente e com o mínimo custo possível.

Proceder a alterações num sistema existente, mesmo para quem fez parte do seu desenvolvimento, poderá resultar numa tarefa árdua dependendo da complexidade que o sistema apresenta. E torna-se ainda mais difícil quando um elemento alheio ao processo de desenvolvimento fica responsável por estas modificações. Este é o trabalho fundamental desta tese, analisar um sistema existente, contribuindo para o seu melhoramento tanto a nível de funcionalidades, desempenho e de usabilidade.

O sistema existente visava colmatar dificuldades que a companhia de transportes públicos tem em obter informações mais precisas sobre como os seus serviços são utilizados. Ao mesmo tempo este sistema permite a empresa estar mais próxima dos seus clientes, fornecendo informações em tempo real sobre possíveis alterações das carreiras ou outro tipo de informação útil aos seus clientes e assim melhorar a qualidade do serviço prestado.

Com os dados recolhidos pelo sistema é possível analisar quais são as paragens e locais onde há uma maior utilização dos transportes públicos e ter uma estimativa do número de pessoas por paragem. Analisando estes dados podem ser tomadas decisões importantes de negócio, tais como:

- Reduzir ou aumentar o número de carreiras que passam em determinada zona;
- Tomar estratégias de marketing que incentivem e cativem mais os passageiros para a utilização dos transportes públicos;
- Reduzir o tráfego no centro das cidades, poupar possivelmente nos gastos de combustível e nas emissões de CO_2.

Com os pontos focados anteriormente esse sistema traz muitas vantagens mas é necessário torná-lo estável, adicionar novas funcionalidades e fazer a transição da aplicação de criação e gestão de serviços que tem revelado ser uma limitação na utilização deste sistema, para uma *interface* Web, simplificando assim a sua utilização, escalabilidade e interação.

1.2. OBJETIVOS

Como já foi referido anteriormente, o principal objetivo deste trabalho de dissertação foi analisar um sistema distribuído constituído por 12 estações e 1 servidor, a fim de resolver instabilidades que os equipamentos têm evidenciado, bem como limitações e problemas de usabilidade que a aplicação de gestão e criação de serviços tem colocado.

Este sistema é utilizado pela empresa Horários do Funchal, que possui estações Bluetooth espalhadas por algumas paragens. Na sede da empresa encontra-se um servidor, onde a aplicação que permite controlar e interagir com as estações está instalada, e também onde é guardada a base de dados que regista tudo o que acontece no sistema.

As tarefas para alcançar os objetivos baseiam-se em detetar e corrigir a razão pela qual as estações perdem a comunicação com o servidor, deixam de responder aos pedidos da aplicação e de enviar o conteúdo aos dispositivos Bluetooth dos utentes. Em relação à aplicação de gestão e criação de serviços as tarefas passam por desenvolver uma nova aplicação num ambiente de execução Web, abandonando assim o ambiente de linha de comandos, de forma a ser possível contornar os problemas de usabilidade e limitações na implementação de novas funcionalidades, tais como ter a possibilidade de diagnosticar o estado das estações em tempo real.

Assim o objetivo desta dissertação numa primeira fase consiste em detetar e resolver todas as instabilidades que o sistema apresenta, analisando toda a sua parte física e lógica (*Hardware* e *Software*). Numa segunda fase, desenvolver uma aplicação Web abandonando assim a aplicação existente, para que seja possível dar suporte a todas as funcionalidades de criação e de gestão dos serviços, bem como de diagnóstico das BlueStations e tornar todo o sistema operacional.

1.3. DESAFIOS

As decisões durante a realização desta dissertação foram tomadas essencialmente com base na análise prévia de todo o sistema, pensando nos desafios que o mesmo colocava, de forma a conseguir realizar os objetivos propostos.

Os desafios encontrados para a realização desta dissertação dividem-se em quatro categorias:

- **Sistema Operativo e Linguagem de Programação Utilizada**

    1) O sistema está programado em Perl, utiliza uma biblioteca chamada BlueZ, escrita em C, que trata das comunicações Bluetooth entre as estações e os dispositivos encontrados. O sistema utiliza alguns componentes escritos em Shell Script, que possuem instruções com um conjunto de comandos que fazem referência aos componentes que são necessários executar para que o sistema funcione. Analisar e entender um sistema que utiliza 2 linguagens de programação é, por si só, um grande desafio.

    2) Entender a arquitetura de um sistema que já está implementado, que possui centenas de linhas de código, que corre sobre o Linux, compreender o seu funcionamento e entender os problemas que o sistema apresenta é um desafio complexo. Complexo devido à quantidade de ficheiros que comunicam entre si para fazer o sistema funcionar e devido a tarefa de *debug* em linha de comandos ser um processo em si mesmo nada acessível e que revela algumas limitações para ambientes que utilizam *threads*, como é o caso deste sistema.

- **Sistema Distribuído e Tecnologia Utilizada**

    1) O sistema utiliza a tecnologia Bluetooth nas estações, que estão distribuídas pelas paragens, para enviar conteúdos aos dispositivos dos utilizadores. Esta tecnologia tem algumas limitações em termos de número de ligações em simultâneo e é necessário adaptar os conteúdos que são enviados aos utilizadores, para que a taxa de sucesso de envio seja melhor que atual.

    2) É desafiante analisar este sistema distribuído que faz uso da tecnologia UMTS (*Universal Mobile Telecommunication System*) nas BlueStation para comunicar com o servidor, enviando informações para a base de dados, e fazer a comunicação com a aplicação de gestão e criação de serviços. Para esta comunicação o sistema utiliza uma arquitetura para estabelecer e simplificar a troca de dados que tem por nome CORBA (*Common Object Request Broker*). É um desafio entender como é feita a implementação do CORBA, para que seja possível utilizar esta arquitetura de comunicação entre a aplicação Web e a unidade central, porque é a única na qual o sistema consegue comunicar e interpretar todos os pedidos.

- **Desempenho e Usabilidade**

    1) Genericamente diminuir a complexidade existente na utilização do sistema.

    2) Apresentar melhores níveis de desempenho e de estabilidade do sistema.

    3) Substituir a aplicação que é utilizada no servidor para a comunicação com as estações e para a criação dos conteúdos que são enviados aos utentes do serviço, por uma aplicação Web e que seja mais *user-friendly*.

- **Desafio Final**

    1) De forma a ser possível ultrapassar todos estes desafios, responder às exigências que o sistema apresenta e às necessidades de quem o utiliza, o desafio final passa por criar uma aplicação gráfica desenvolvida para a Web, que contorne todas as limitações existentes, adicionando mais funcionalidades ao sistema e melhorando as atuais.

1.4. CONTRIBUIÇÃO

As contribuições descritas nesta dissertação são as seguintes:

- Análise e manutenção de todos os elementos físicos (*Hardware*) e lógicos (*Software*) do sistema, corrigindo limitações e instabilidades, substituindo *hardware* danificado e aumentando o desempenho do sistema.

- Desenvolvimento de uma aplicação Web que comunica com um sistema distribuído constituído por vários computadores (chamados de BlueStations), que utilizam a tecnologia Bluetooth para detetar dispositivos e enviar conteúdos aos equipamentos encontrados. Esta aplicação foi desenvolvida recorrendo à implementação do serviço REST (*Representational State Transfer*) e de um cliente CORBA. A nova aplicação contribuiu para que novas funcionalidades fossem implementadas, para que a interação entre o administrador e a aplicação fosse simplificada e para que novos atributos de segurança e de registos de utilização fossem adicionados.

- Fornecer à empresa Horários do Funchal o seu sistema, com novos atributos de qualidade, ajudando-a a estar mais próxima dos seus clientes, poder melhorar a qualidade dos serviços que presta e ver realizada a necessidade de ter uma nova aplicação que facilite o uso do sistema distribuído.

- Contribuir para que a sociedade tenha ao seu dispor um serviço que a possa auxiliar na utilização da rede de transportes da empresa Horários do Funchal e tenha acesso a informações de forma gratuita.

- Acrescentar nova documentação à já existente sobre o sistema e utilizar uma linguagem de programação mais atual na nova aplicação do que a utilizada pela aplicação existente.

- Apresentação de informações importantes, que podem ser utilizadas para outros projetos que desejam utilizar a tecnologia Bluetooth ou outros componentes utilizados nesta dissertação, e até estratégias que podem ser adotadas para trabalhos de engenharia reversa.

1.5. ORGANIZAÇÃO

Esta dissertação encontra-se dividida em nove capítulos. O primeiro capítulo, introdutório, é descrita a motivação para a realização deste trabalho. Globalmente é descrita a realidade em que o sistema se encontra, os objetivos a serem atingidos, os desafios que o trabalho apresenta, as respetivas contribuições e a organização da dissertação, encerram o primeiro capítulo.

Relativamente ao segundo capítulo é apresentado em detalhe o *background* associado à dissertação, nomeadamente a tecnologia Bluetooth, a arquitetura CORBA e por fim o serviço REST.

No terceiro capítulo, é detalhado todo o sistema que suporta as BlueStations. Nesse capítulo é apresentada toda a arquitetura do sistema, é detalhado cada componente que a constituí e para terminar o capítulo é apresentado como é feita a comunicação com as estações Bluetooth.

O quarto capítulo detalha em termos de implementação e funcionalidades a aplicação existente para gerir e criar serviços (consola de administração), que são posteriormente associados às estações Bluetooth.

No quinto capítulo, é descrita a abordagem tomada para proceder à engenharia reversa do sistema, na qual se destacam técnicas de modelação, levantamento dos novos requisitos para a nova aplicação Web de gestão e criação de serviços, e por fim são apresentados os protótipos criados para a *interface* da nova aplicação.

Relativamente ao sexto capítulo, são descritos todos os aspetos de implementação da aplicação Web para a gestão e criação de serviços, assim como a sua arquitetura e todas as funcionalidades que a aplicação permite executar.

O sétimo capítulo apresenta a comparação entre a antiga aplicação para a gestão e criação de serviços com a nova aplicação Web desenvolvida para o mesmo propósito. Essa comparação é feita em termos de requisitos necessários para executar a aplicação, em termos de *interface*, usabilidade e por fim nas funcionalidades que ambas apresentam.

No oitavo e penúltimo capítulo, são descritos os testes de usabilidade efetuados à nova aplicação, sendo também efetuada uma análise aos resultados obtidos, transcrevendo as alterações que estes surtiram na aplicação.

Para finalizar, no nono e último capítulo, são apresentadas as conclusões e trabalhos futuros.

2. ESTADO DA ARTE

Neste capítulo será feita uma descrição das diferentes tecnologias envolvidas neste trabalho, nomeadamente a tecnologia Bluetooth, arquitetura CORBA e o serviço REST.

2.1. TECNOLOGIA BLUETOOTH

O nome da tecnologia Bluetooth foi atribuído pelo Bluetooth SIG (*Special Interest Group*) em referência ao rei dinamarquês Harold Blatand, cujo nome em inglês seria Harald Bluetooth [1].

O Bluetooth é uma especificação industrial para a área das redes sem fios PAN (*Personal Area Networks*). A tecnologia de radiofrequência Bluetooth opera na faixa de frequência dos 2.4 GHz, conhecida também como ISM (*Industrial Scientific Medical*).

O Bluetooth é classificado como uma rede *ad hoc,* caracterizada pela formação de grupos de transmissão (*piconets*) entre dispositivos que desejam conectar-se uns aos outros. A tecnologia Bluetooth possibilita que os dispositivos executem operações com autorização, sem nenhuma ação por parte do utilizador do dispositivo, bastando para isso ter no seu equipamento o Bluetooth ativo e visível. Este tipo de operações é muito utilizado para *marketing*, porque possibilita o envio de mensagens de forma gratuita a um grande número de equipamentos, sendo que o utilizador apenas decide se quer aceitar ou rejeitar a mensagem [2].

2.1.1. Versões da Especificação Bluetooth

A tecnologia Bluetooth ao longo dos anos tem sofrido algumas alterações, resultando em novas versões, que permitiram um aumento da taxa de transferência e uma redução dos consumos de energia. A lista das versões da especificação Bluetooth é detalhada na Tabela 1.

Versão	Taxa de Transferência	Principais Atualizações
1.1	1 Mbps	Primeira versão comercial
1.2	1 Mbps	Implementação de estratégia anti interferência (AFH – *Adaptive Frequency-Hopping*)
2.0 + EDR	2.1 Mbps	Melhoria da taxa de transferência (EDR – *Enhanced Data Rate*)
2.1 + EDR	2.1 Mbps	Processo de emparelhamento simples (SSP – *Secure Simple Pairing*)
3.0 + HS	24 Mbps	Uso do canal WiFi (IEEE 802.11) para transferência
4.0	1 Mbps	Consumo de energia reduzido

Tabela 1 - Versões da especificação Bluetooth

2.1.2. Classes Bluetooth

Cada dispositivo que suporta a tecnologia Bluetooth incorpora um controlador que é enquadrado em uma das 3 classes descritas na Tabela 2. As 3 classes diferenciam-se pela potência e alcance máximo que o controlador Bluetooth pode atingir. Quanto maior for o alcance, maior é a potência necessária para que seja atingida a distância desejada. Por essa razão os dispositivos normalmente utilizam as classes 2 ou 3, de forma a economizar energia.

Classe	Potência máxima permitida (mW-miliWatt / dBm-decibel miliwatt)	Alcance (Aproximadamente)
1	100 mW (20 dBm)	Até 100 metros
2	2.5 mW (4 dBm)	Até 10 metros
3	1 mW (0 dBm)	1 metro

Tabela 2 - Classes Bluetooth

O alcance dos dispositivos de classe 2 pode ser expandido, quando conectados a dispositivos de classe 1. Isso é conseguido graças à alta potência de transmissão do dispositivo de classe 1.

2.1.3. Visibilidade

Em termos de visibilidade um equipamento Bluetooth pode estar nos seguintes estados:

- **Visível**: O dispositivo neste estado pode ser descoberto por outro equipamento que realize o processo de descoberta na sua vizinhança. Este estado fica ativo por tempo indeterminado.

- **Visível por tempo limitado**: Semelhante ao estado anterior, com a diferença que os dispositivos ficam visíveis para serem descobertos por tempo limitado.

- **Oculto**: O dispositivo neste estado não responde a processos de descoberta realizados pelos seus vizinhos, mas permite ligar-se e ser descoberto por dispositivos que já o conhecem.

Os dispositivos Bluetooth que estejam nos modos visíveis ou visível por tempo limitado, esperam por pedidos de descoberta num canal exclusivo para essa finalidade. Cada canal tem uma série de frequências sendo que o dispositivo no estado visível itera por cada uma dessas frequências, para responder às requisições de descoberta.

Cada um dos estados tem associado a si um consumo de energia, sendo o estado visível o que mais energia utiliza, pelo facto de o sistema estar constantemente a iterar sobre as frequências de descoberta, a fim de responder a eventuais requisições. O estado visível por tempo limitado consome energia só pelo tempo em que itera pelas frequências de descoberta, (ou seja, consome energia para responder a pedidos de procura de outros dispositivos) passando depois deste tempo para o estado oculto. No estado oculto o dispositivo fica invisível e não há qualquer troca de dados com dispositivos que não o conhecem (ou seja, que nunca foram emparelhados).

Neste estado não há consumo de energia para o processo de descoberta, porque o dispositivo não responde a nenhum pedido de procura, sendo portanto o melhor estado para comunicar entre dispositivos que já se conhecem previamente.

2.1.4. Descoberta de Dispositivos

A descoberta de dispositivos é um processo durante o qual o dispositivo Bluetooth pode detetar outros dispositivos Bluetooth que estejam na sua vizinhança. Para este processo, o dispositivo que inicia a descoberta envia uma mensagem de *broadcast* (difusão) e aguarda pela resposta dos dispositivos que estiverem visíveis e que respondam à requisição.

A mensagem de *broadcast* enviada é acompanhada de um dos dois possíveis códigos de acesso, GIAC ou LIAC:

- **GIAC (*General Inquiry Access Code*)**: Este código de acesso específica que todos os dispositivos que estejam no estado visível devem responder à requisição.

- **LIAC (*Limited Inquiry Access Code*)**: Este código de acesso específica que todos os dispositivos que estejam visíveis temporariamente devem responder à requisição.

A utilização de dois códigos de acesso justifica-se na medida em que se deseje procurar por dispositivos que estejam sempre visíveis, ou dispositivos que estejam visíveis por tempo limitado. Esta forma de procura de dispositivos pode ser útil se, se quiser fazer chegar um pedido a um dispositivo que tem tempo limite de visibilidade, antes de enviar pedidos a dispositivos que estão sempre visíveis e que podem receber um pedido a qualquer momento. Quando uma descoberta é realizada com o LIAC, uma quantidade bastante reduzida de dispositivos responde à requisição, passando rapidamente para o modo oculto.

A resposta dos dispositivos à mensagem de *broadcast* inclui principalmente um identificador único do dispositivo, que é o endereço MAC. Quando um dispositivo tem na sua posse este identificador pode dar início a requisições de conexão ao dispositivo encontrado.

O processo de descoberta não possui nenhum controlo de concorrência. Ao mesmo tempo que um dispositivo está no processo de descoberta, outros podem estar a realizar outros processos. Sendo assim, um dispositivo que inicia um processo de descoberta pode estar em simultâneo a descobrir outros aparelhos e responder a requisições de descobertas de outros dispositivos.

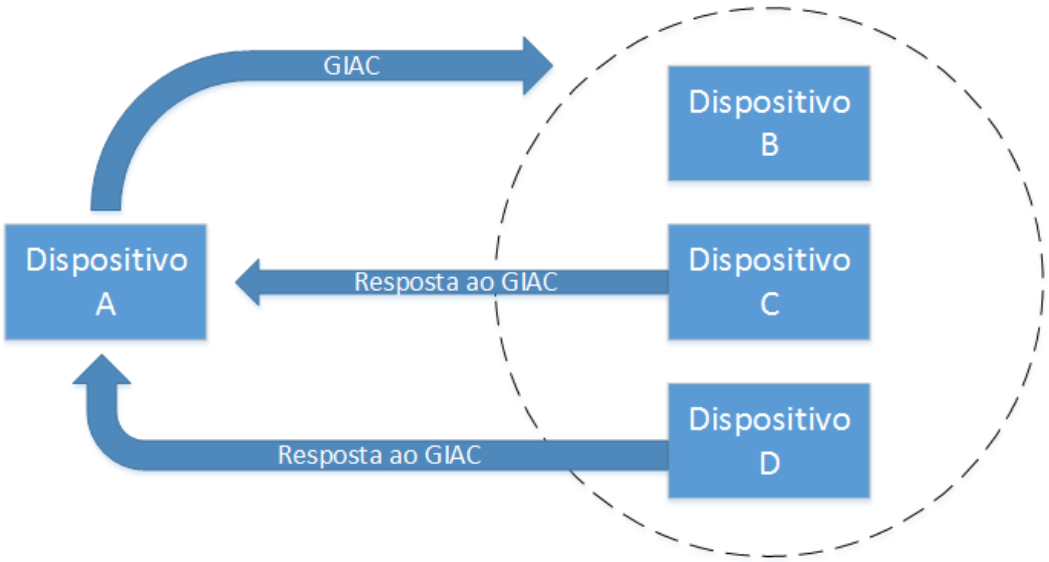

Figura 1 – Bluetooth: Códigos de Acesso de Procura Geral (GIAC)

Na Figura 1 o dispositivo A envia a mensagem de difusão com o código GIAC para os dispositivos na sua vizinhança. O dispositivo B está oculto e por isso não responde à requisição. O dispositivo C está visível por tempo limitado e o dispositivo D está visível, e por isso ambos vão dar resposta ao pedido GIAC. A resposta caso o dispositivo A envie a mensagem de difusão com o código LIAC encontra-se representada na Figura 2.

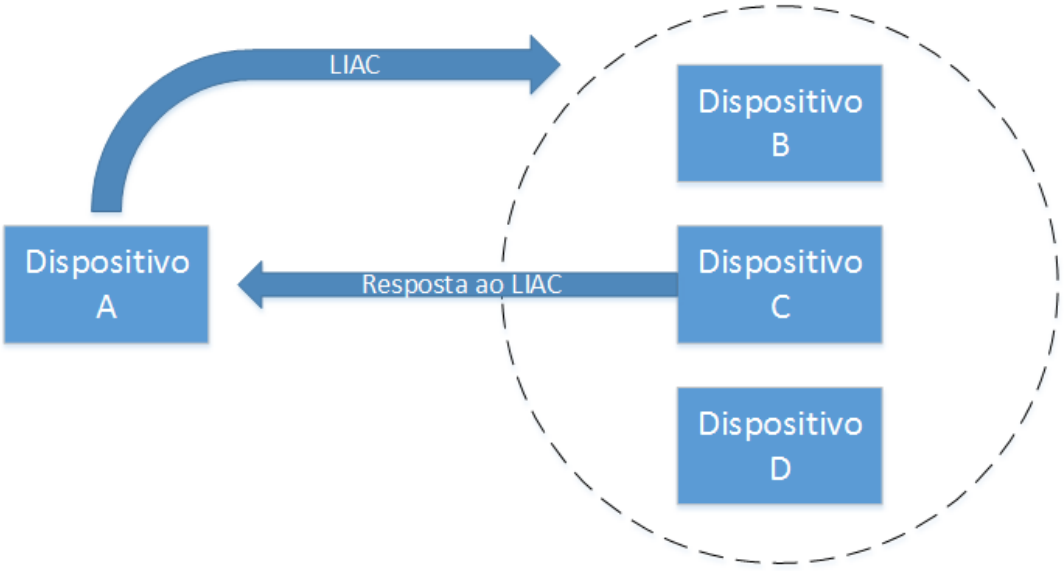

Figura 2 – Bluetooth: Códigos de Acesso de Procura Limitada (LIAC)

O dispositivo B está oculto e por isso não responde à requisição. O dispositivo C está visível por tempo limitado e por essa razão responde à requisição, o dispositivo D está visível por tempo indeterminado e por isso não vai responder à mensagem de difusão com o código LIAC.

2.1.4.1. Determinismo e Pior Caso

O processo de descoberta de dispositivos é não-determinístico. Para os mesmos dispositivos, sucessivos processos de descoberta podem devolver conjuntos diferentes de dispositivos encontrados. Isso deve-se a dois factos: resposta pseudoaleatória na frequência de descoberta a limitação do tempo no processo de descoberta.

Para evitar colisões com outros dispositivos, a resposta a uma requisição de descoberta por parte de um dispositivo visível, espera um tempo pseudoaleatório para responder. Em ambientes com um grande número de dispositivos envolvidos, podem ocorrer muitas interferências e com isso alguns dispositivos podem não ser descobertos devido ao tempo de descoberta vir a ser curto para que o dispositivo seja detetado.

O tempo-limite para o processo de descoberta é variável para cada dispositivo. Um período de transmissão/receção leva 1250 ms. Em condições normais os dispositivos executam 8 desses períodos, o que no total dá 10 s para um intervalo de busca recomendando, para que se descubram todos os dispositivos na vizinhança.

2.1.5. Perfis Bluetooth

Os perfis Bluetooth (ver Tabela 3) formalizam serviços de alto nível, nomeadamente aplicações e usos que podem ser utilizados na tecnologia Bluetooth. Por exemplo a troca de ficheiros, transmissão de voz, impressão, ligação à internet, etc.

A maneira como um dispositivo usa o Bluetooth vai depender das capacidades dos seus perfis. Os perfis especificam padrões que os fabricantes devem seguir para que a comunicação entre diferentes dispositivos Bluetooth seja compatível. Na Tabela 3 são descritos os perfis mais utilizados.

Sigla	**Perfil**	**Descrição**
A2DP	Advanced Audio Distribution Profile	Perfil de distribuição de áudio avançada
BPP	Basic Printing Profile	Perfil básico de impressão
DUN	Dial-Up Networking	Perfil de rede *dial-up*
FAX	FAX Profile	Perfil de Fax
FTP	File Transfer Profile	Perfil de transferência de ficheiros
HID	Human Interface Device	Perfil de dispositivos de *interface* humana
OPP	Object Push Profile	Perfil de envio direto de dados

Tabela 3 - Alguns perfis Bluetooth

2.1.6. Busca de Serviços

A tecnologia Bluetooth especifica um protocolo denominado SDP (*Service Discovery Protocol*), que é utilizado para descobrir quais os serviços Bluetooth disponíveis no dispositivo ao qual se conecta.

O protocolo segue a arquitetura cliente/servidor, onde o cliente é quem inicia o SDP e o servidor é o dispositivo-alvo. O dispositivo-alvo ao receber uma requisição SDP devolve uma lista denominada de *service records* (registos de serviço).

Para cada registo a lista contém informações específicas (tais como qual o canal de radiofrequência que deve ser utilizado, protocolo, etc) de como deve ser feita a conexão ao dispositivo-alvo, para realizar o serviço desejado.

2.1.7. Emparelhamento de Dispositivos

Para que dois dispositivos Bluetooth possam comunicar entre si, há uma etapa opcional chamada de emparelhamento. No ato da comunicação entre dois dispositivos é especificada uma chave de emparelhamento, que vai permitir a autenticação entre ambos e fazer com que a comunicação entre eles seja possível. Depois do processo de emparelhamento, os dispositivos guardam os endereços MAC, e em futuras ligações podem comunicar de forma transparente. Os endereços guardados podem ser eliminados, mas assim o processo de emparelhamento tem de ser novamente realizado.

O emparelhamento pode ser executado durante a execução do SDP. Quando um dispositivo-alvo decida que o emparelhamento é obrigatório, ele pode decidir que o emparelhamento seja feito antes ou depois do SDP. Neste processo é definida uma chave de emparelhamento, que é muito simples de quebrar devido ao seu tamanho. Portanto, os dispositivos que utilizem esse recurso devem utilizar chaves maiores que os quatro dígitos usuais.

O método de emparelhamento percorre as seguintes etapas:

- **Descoberta**: O dispositivo deve proceder à procura do dispositivo-alvo, com o qual deseja comunicar.

- **Busca de serviço**: O dispositivo-alvo ao ser questionado sobre os serviços que suporta, requisita uma autenticação.

- **Chave de emparelhamento**: A chave de emparelhamento é inserida em ambos os dispositivos que desejam comunicar. Esta chave normalmente é numérica e é digitada ao se transferirem dados para um dispositivo pela primeira vez.

De forma a simplificar o processo de emparelhamento, a partir da versão 2.1, foi incluído o mecanismo SSP (*Secure Simple Pairing*), que aceita alguns métodos diferentes para emparelhar dispositivos. Um dos métodos é o OOB (*Out Of Band*) que permite que o emparelhamento seja feito utilizando um mecanismo diferente do Bluetooth, por exemplo o NFC. Enquanto no

Bluetooth o tempo para o estabelecimento de uma ligação leva entre 6 a 10 segundos, no NFC ela ocorre em até 0.1 segundos.

Os métodos alternativos para executar o emparelhamento têm como objetivo agilizar e/ou implementar mais segurança a esse procedimento [1].

Para além do OOB, existem os seguintes métodos de emparelhamento:

- **Comparação Numérica**

O método comparação numérica é utilizado em cenários onde ambos os equipamentos tenham um ecrã, e possam introduzir um comando de confirmação dos números apresentados. Um bom exemplo é uma ligação Bluetooth entre um PC e um telemóvel.

Quando os dispositivos querem se conectar é apresentado em ambos os equipamentos um número de seis dígitos (de "000000" até "999999") é perguntado se o número apresentado é o mesmo nos dois equipamentos. Se em ambos os equipamentos a resposta for "sim", o emparelhamento é realizado com sucesso.

A comparação numérica é utilizada por dois motivos. O primeiro deve-se ao facto de muitos dispositivos não terem nomes exclusivos (terem nomes iguais) e o pedido de emparelhamento poder ser feito para o equipamento não desejado. Utilizando a comparação numérica este problema é ultrapassado, garantido que os dispositivos desejados estão conectados entre si. O segundo motivo para a utilização deste método é para evitar ataques *man-in-the-middle*.

- ***Just Works***

O método *Just Works* é projetado para ambientes em que pelo menos um dos dispositivos não possua ecrã capaz de exibir números de 6 dígitos nem teclado. Este método é aplicado por exemplo, aos auriculares Bluetooth. O *Just Works* utiliza a comparação numérica, mas o valor numérico não é apresentado ao utilizador, simplesmente é pedido que aceite o emparelhamento.

Em termos de segurança fornece a mesma que a comparação numérica, mas não oferece nenhuma proteção contra ataques *man-in-the-middle*.

- **Chave de Acesso**

Este método foi pensado para cenários em que um dispositivo tem capacidade de entrada (por exemplo: teclado) e o outro tem capacidade de saída (por exemplo: ecrã). É mostrado ao utilizador um número de seis dígitos (de "000000" até "999999") no dispositivo que possui capacidade de mostrar o código e o outro dispositivo introduz o código. Se o valor introduzido no segundo dispositivo estiver correto, o emparelhamento é iniciado [3].

2.1.8. Protocolos Bluetooth

A pilha de protocolos Bluetooth é definida pela organização Bluetooth SIG. A pilha é dividida em três grupos lógicos: grupo de protocolos de transporte, grupo de protocolos de *middleware* e o grupo de aplicação (ver Figura 3).

O grupo de protocolos de transporte permite aos dispositivos Bluetooth localizar outros dispositivos e gerir *links* físicos e lógicos para as camadas superiores da pilha de protocolos. As camadas Bluetooth *Radio*, *Baseband*, *Link Manager* e L2CAP (*Logical Link Control Adaptation Protocol*) fazem parte do grupo de protocolo de transporte. Estes protocolos suportam dois tipos de comunicação: síncrona e assíncrona. Todas as camadas deste grupo são indispensáveis para a comunicação entre dispositivos Bluetooth.

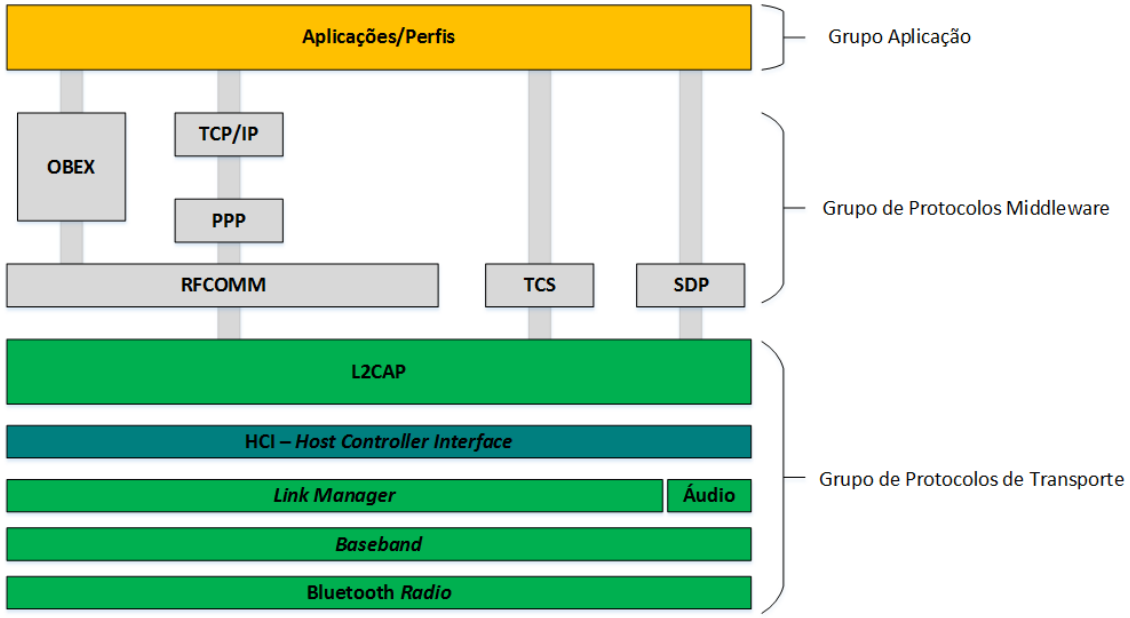

Figura 3 – Bluetooth: Pilha de Protocolos

2.1.8.1. Grupo de Protocolos de Transporte

- **Camada Bluetooth *Radio***: especifica os requisitos do dispositivo Bluetooth, incluindo frequência, recuperação de *clock*, deteção de dados, esquema de modulação e potência de transmissão.

- **Camada *Baseband***: especifica o estabelecimento de conexão com uma *piconet*, o endereçamento, formato do pacote, temporização e controlo de energia.

  Nesta camada são definidos quem é o dispositivo mestre e o dispositivo escravo. A camada *Baseband* também define procedimentos de processamento de pacotes, estratégias de deteção de erros, criptografia, transmissão e retransmissão de pacotes.

A tecnologia Bluetooth como lida com transmissão de dados, voz e vídeos, utiliza para cada tipo de dados diferentes ligações [6]. Estas ligações são de dois tipos: SCO (*Synchronous Connection-Oriented*) e ACL (*Assynchronous Connection-Less*).

A ligação SCO é utilizada principalmente para voz e vídeo, requerendo transmissões de dados rápidas e consistentes, fazendo ligação ponto-a-ponto entre dois dispositivos. Os pacotes transmitidos nesta ligação nunca são retransmitidos, mesmo que ocorram erros. Os pacotes SCO são tratados como prioritários em relação aos pacotes ACL. Este tipo de ligação, por ser síncrona, reserva e utiliza os *slots* de tempo de forma periódica, em intervalos de tempo fixos. Para este tipo de ligação existem diferentes taxas de transmissão. Para dados a taxa de transmissão é de 432 kbps e para pacotes de voz é de 64 kbps.

A ligação ACL é do tipo ponto-a-multiponto, ou seja, possui um mestre e vários escravos. Está ligação utiliza a transmissão assíncrona, que utiliza somente os *slots* de tempo que não foram previamente reservados para a transmissão SCO. Está ligação é usada para a transmissão de pacotes com dados e possui mecanismos para reenviar pacotes perdidos ou com erros. A velocidade de transmissão desta ligação é de até 721 kbps.

- **Camada *Link Manager***: Está camada implementa o LMP (*Link Manager Protocol*), que gere as prioridades do meio de transmissão entre os dispositivos. Este protocolo gere a taxa de transferência de dados e de áudio, bem como é responsável pela autenticação, por definir níveis de confiança entre dispositivos, implementar criptografia e controlar os gastos de energia.

- **Camada Áudio**: Os pacotes de áudio são tratados com prioridade para manter a qualidade esperada pelas aplicações que os utilizam. Os pacotes de áudio são direcionados diretamente para a *baseband*, para posteriormente serem transmitidos em pequenos pacotes diretamente para a *interface* Bluetooth *Radio*.

- ***Host Controller Interface***: Permite que as camadas superiores a esta *interface*, tenham acesso à *baseband*, *link manager* e outros registos de *hardware*.

- **Camada L2CAP**: A camada L2CAP é responsável pela multiplexação de dados, permitindo que vários protocolos e aplicações compartilhem a *interface* Bluetooth *radio*. Nesta camada é feita a segmentação de grandes pacotes utilizados pelas camadas superiores, para serem transmitidos pela *baseband*, bem como a remontagem desses pacotes nos dispositivos recetores [4] [5].

2.1.8.2. Grupo de Protocolos *Middleware*

Neste grupo de protocolos estão incluídos protocolos de terceiros e padrões industriais, que possibilitam que aplicações existentes e novas aplicações operem sobre a tecnologia Bluetooth. A descrição de alguns destes protocolos encontra-se na Tabela 4 [6].

Sigla	Descrição
RFCOMM - *Radio Frequency Communications*	Protocolo de substituição de cabo. Utilizado para criar uma porta serial (RS-232) virtual.
TCS - *Telephony Control Protocol*	Protocolo usado para controlo de chamadas de voz e dados entre dispositivos Bluetooth.
SDP - *Service Discovery protocol*	Protocolo usado para descobrir que serviços são suportados pelo outro dispositivo Bluetooth e quais os parâmetros que deve usar para se conectar a ele.
PPP - *Point-to-Point Protocol*	Protocolo padrão da Internet, utilizado para transportar datagramas de IP sobre uma ligação ponto-a-ponto.
TCP/IP - *Transmission Control Protocol/Internet Protocol*	Protocolos essenciais ao conjunto de protocolos TCP/IP.
OBEX - *Object Exchange Protocol*	Protocolo que facilita a troca de dados entre dispositivos Bluetooth.

Tabela 4 – Bluetooth: Alguns protocolos da camada middleware

2.1.8.3. Camada de Aplicações

Esta camada de aplicações, também conhecida como perfis (ver secção 2.1.5), identifica 13 aplicações e fornece pilhas de protocolos para que os recursos da tecnologia Bluetooth possam operar entre equipamentos de diferentes fabricantes [4] [5].

2.1.9. Topologias Bluetooth

A tecnologia Bluetooth possui dois tipos de topologias: *piconet* e *scatternet* (ver Figura 4). Nessas topologias existem dois tipos de conexões: ponto-a-ponto (*peer-to-peer*) e ponto-a-multiponto (*point-to-multipoint*). A ligação ponto-a-ponto permite a implementação da topologia *piconet*, onde um dos dispositivos é o mestre e outros dispositivos que compartilham o mesmo canal físico de comunicação são os escravos. Uma *piconet* é uma rede *ad-hoc*, que permite um dispositivo mestre se interconectar com até sete dispositivos ativos. Os dispositivos podem trocar de posição e o escravo pode passar para mestre e vice-versa.

A agregação de duas ou mais *piconets* forma uma *scatternet*. Neste caso, o mestre de uma *piconet* pode ser o escravo em uma *scatternet*. O compartilhamento do meio físico é feito apenas entre as *piconets*, e não entre toda a *scatternet* [1] [5].

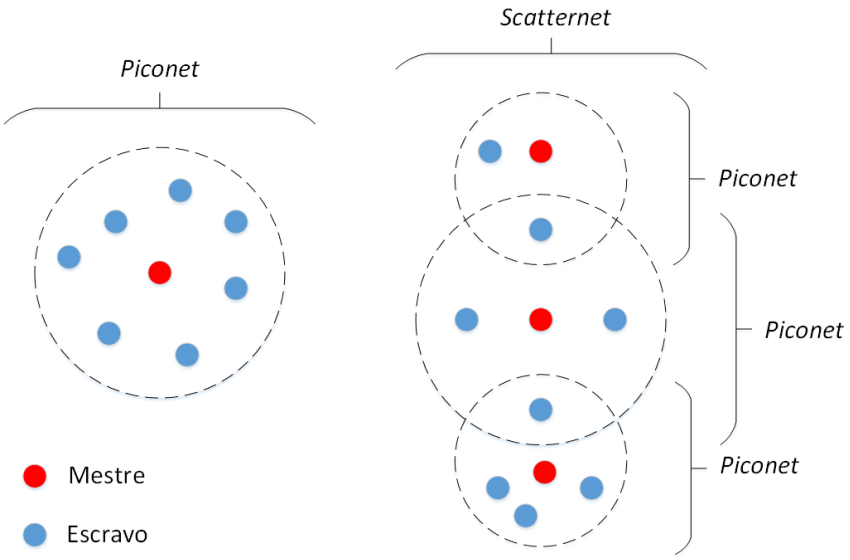

Figura 4 – Topologias Bluetooth

2.1.10. Zonas Bluetooth

As zonas Bluetooth são áreas de interatividade cobertas por um ou mais dispositivos Bluetooth (que funcionam como emissores de informações), que detetam outros dispositivos que tenham essa tecnologia, com o objetivo de transmitir e/ou recolher informações associadas aos dispositivos encontrados.

Estas zonas são muitas vezes utilizadas para fazer publicidade, por oferecerem uma solução eficaz e acessível para interagir com um grande público e oferecerem serviços que sejam relevantes para quem recebe esses conteúdos nos seus dispositivos.

As zonas Bluetooth quando comparadas com os métodos tradicionais de divulgação ou prestação de serviços possuem as seguintes vantagens [8]:

- O conteúdo é entregue diretamente no dispositivo do cliente.
- O cliente tem a possibilidade de só aceitar os conteúdos que desejar.
- Possibilidade de fazer chegar informações aos dispositivos em diferentes formatos (vídeo, áudios, etc).
- As zonas Bluetooth são portáteis e podem ser facilmente instaladas.
- As zonas Bluetooth podem recolher informações do serviço prestado e dos dispositivos, para serem analisados.
- As informações que a zona Bluetooth está a enviar podem ser facilmente alteradas.

2.2. ARQUITETURA CORBA

O padrão CORBA foi criado pela OMG (*Object Manager Group*) em 1989 para estabelecer e simplificar a interoperabilidade entre sistemas distribuídos.

De forma a entender o funcionamento desta arquitetura (ver Figura 5) é necessário entender dois conceitos essenciais, que são descritos abaixo:

- **IDL (*Interface Definition Language*):** para cada objeto CORBA existe necessariamente uma *interface* claramente definida e especificada em IDL. Nestas *interfaces* são definidos quais os serviços associados ao objeto, ou seja, quais as operações que o objeto suporta, sem que seja fornecida qualquer informação da implementação do objeto. Um objeto CORBA pode ter mais do que uma *interface* associada a ele, sendo neutra em relação à linguagem de programação implementada pelo cliente CORBA. O cliente CORBA não necessita conhecer como o objeto remoto foi implementado nem a sua localização.

- **ORB (*Object Request Broker*)**: é o componente que funciona como intermediário na transferência de mensagens entre o cliente CORBA e servidor CORBA.

  Um cliente CORBA é a entidade que pretende executar uma operação num objeto CORBA que se encontra no servidor e que é chamado remotamente. Quando um objeto remoto é invocado envolve a especificação do objeto destino, da operação a ser invocada e dos parâmetros que serão enviados e retornados.

  O ORB implementa os mecanismos necessários para localizar o objeto invocado pelo cliente e devolver os dados que resultam do pedido. O cliente pode fazer uma requisição tanto por meio do uso da *interface* de invocação dinâmica ou por meio de um *stub* (*interface* de invocação estática) gerado em IDL. Assim, um objeto recebe um pedido seja via *skeleton* (*interface* estática do servidor) gerado em IDL ou através de um *skeleton* dinâmico. Depois disso o ORB localiza a operação do objeto referenciado, transfere os parâmetros, e quando o pedido do cliente está completo, tanto o controlo como o resultado obtido pela requisição é retornado ao cliente (ver Figura 5).

  As invocações remotas do cliente CORBA via *stub* geralmente são síncronas, enquanto as invocações remotas via DII (*Dynamic Invocation Interface*) podem ser síncronas ou assíncronas. A invocação dinâmica foi concebida para colmatar a falta de flexibilidade na invocação estática. O padrão CORBA possui duas *interfaces* dinâmicas: a DII do lado do cliente e a DSI (*Dynamic Skeleton Interface*) do lado do servidor. Na DII o cliente CORBA invoca operações sobre o objeto CORBA sem a necessidade de ter informação estática sobre o mesmo, ou seja, o cliente não tem a si associado um *stub* para realizar a invocação do objeto, mas determina a *interface* do objeto, acedendo a um repositório de *interfaces* (ver Figura 5). Na DSI o servidor fornece as implementações das operações dos objetos CORBA, sem a necessidade do cliente conhecer a *interface* estática do objeto.

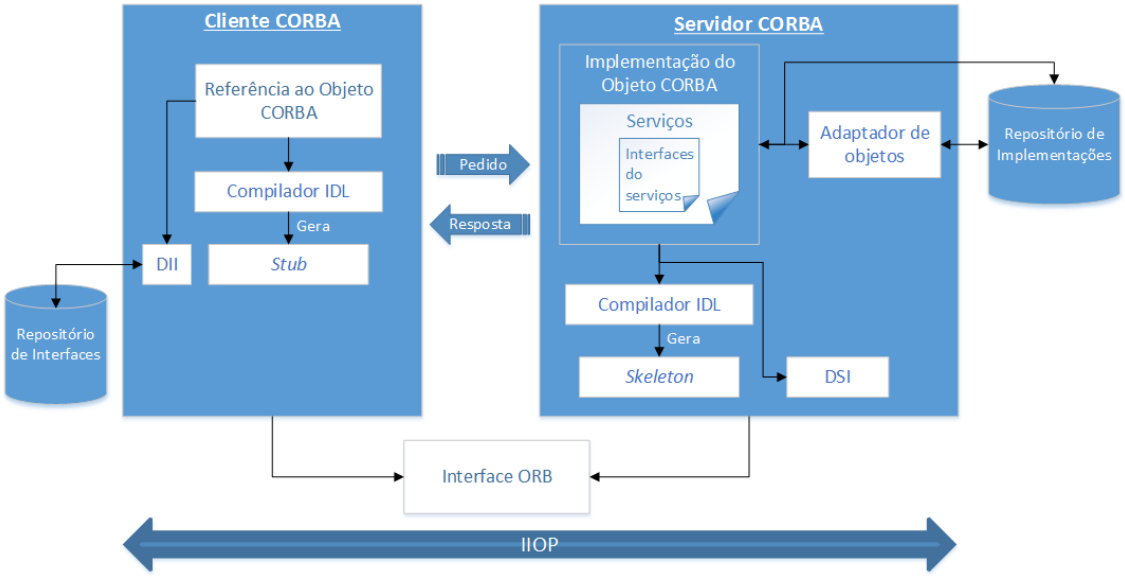

Figura 5 – Comunicação entre Cliente CORBA e Servidor CORBA

O protocolo IIOP (*Internet Inter-Orb Protocol*) baseado no TCP/IP (*Transmission Control Protocol/Internet Protocol*) garante a interoperabilidade entre ORBs localizados remotamente. A especificação CORBA permite que aplicações distribuídas desenvolvidas em linguagens de programação diferentes e que se executam sobre sistemas operativos distintos, possam interagir, possibilitando que vários objetos heterogéneos, distribuídos possam operar de forma transparente. O padrão CORBA é baseado na arquitetura cliente/servidor. Os pedidos dos clientes e as respostas aos pedidos são realizados utilizando o ambiente de comunicação ORB.

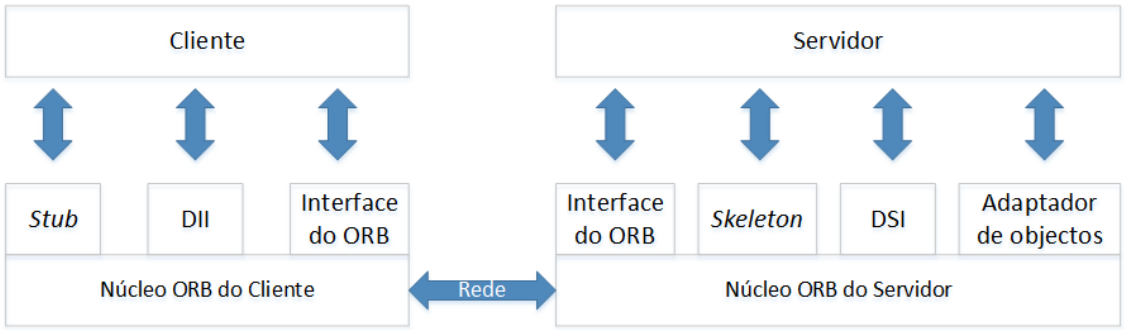

Figura 6 - Arquitetura CORBA

A Figura 6 mostra a arquitetura CORBA, sendo os seus principais componentes detalhados abaixo [9]:

- **Cliente:** tem como principal função requisitar serviços aos objetos localizados no servidor.

- ***Stub*:** é a *interface* de invocação estática, que é gerada a partir da compilação de uma IDL. Está *interface* é utilizada pelo cliente para realizar chamadas a objetos remotos que se encontram no servidor.

- **DII:** é a *interface* de invocação dinâmica, que permite ao cliente chamar um método no servidor sem que tenha conhecimento de sua *interface* em tempo de compilação.

- **Repositório de *Interfaces*:** é a base de dados onde se encontram todas as *interfaces* dos serviços conhecidos pelo ORB. Esta base de dados é utilizada pelos clientes que usam a DII para chamar os métodos de objetos nas quais as *interfaces* são desconhecidas em tempo de compilação.

- **Servidor:** tem como função receber solicitações dos clientes, processá-las e enviar uma mensagem com a resposta ao pedido recebido.

- ***Skeletons*:** é a *interface* estática do servidor, que recebe pedidos do cliente e encaminha-os ao objeto servidor.

- **DSI:** é a *interface* de *skeletons* dinâmica. Esta *interface* entrega chamadas de métodos a uma implementação de objeto cujas *interfaces* não são conhecidas em tempo de execução.

- **Repositório de Implementação:** este repositório contém dados que são necessários para que a ORB localize e ative implementações de objetos.

- **Adaptador de Objetos:** é um componente da arquitetura CORBA que só existe do lado do servidor. Tem como responsabilidades a criação de referências para objetos CORBA, ativação dos objetos e direcionamento das requisições.

- **Núcleo ORB:** responsável por intermediar a comunicação entre o cliente e o servidor.

2.2.1. Serviços

Para as aplicações distribuídas que utilizam o padrão CORBA, a utilização das IDLs e dos objetos que constituem o ORB, não são suficientes, uma vez que a IDL tem como objetivo padronizar as invocações dos clientes feitas aos métodos dos objetos que estão instanciados no servidor. Por sua vez o ORB tem como objetivo coordenar as invocações feitas pelos clientes e as mensagens de retorno. Posto isso vê-se a necessidade da criação de serviços para controlar e executar estas transações entre o cliente/servidor.

Em 2000 a OMG especificou 16 serviços fundamentais para que as aplicações orientadas a objetos pudessem ser executados numa arquitetura CORBA. Em seguida segue a descrição resumida desses 16 serviços:

1) ***Life Cycle Service*** **(Ciclo de vida):** define serviços e acordos para criar, mover, apagar e copiar objetos. Devido a este serviço os clientes podem instanciar objetos sem ser necessário saber a sua localização.

2) ***Relationship Service*** **(Relacionamento):** permite criar associações dinâmicas entre componentes que não se conhecem.

3) ***Naming Service*** **(Nome do Serviço):** permite que os objetos possam localizar e referenciar outros objetos pelo nome.

4) ***Object Transaction Service*** **(Transação):** serviço utilizado para transações**.** Num sistema distribuído um só objeto pode receber várias solicitações e por isso é necessário ter um serviço que mantenha a integridade das transações.

5) ***Trading Object Service*** **(Negócio):** Permite localizar qualquer serviço e também é utilizado pelo *Naming Service*.

6) ***Security Service*** **(Segurança):** este serviço fornece funcionalidades de autenticação, controlo de acesso, auditoria, comunicação segura, criptografia e ferramentas administrativas.

7) ***Concurrency Service*** **(Concorrência):** Permite que numa base de dados ou num sistema de ficheiros apenas um cliente de cada vez, tenha acesso a um registo ou a um ficheiro.

8) ***Persistent State Service*** **(Persistência):** este serviço proporciona aos objetos vários mecanismos de persistência dos dados, tais como base de dados relacionais, base de dados orientada a objetos e até em ficheiros.

9) ***Event Service*** **(Evento):** permite que os componentes subscrevam eventos específicos, sendo notificados assim que o evento esteja disponível.

10) ***Time Service*** **(Tempo):** fornece mecanismos para sincronizar relógios num sistema distribuído e permite ainda acionar eventos através do tempo.

11) ***Notificacion Service*** **(Notificação):** este serviço fornece aos clientes a possibilidade de selecionarem apenas os eventos que têm interesse em receber. Este serviço é uma extensão ao *Event Service* e adiciona a funcionalidade de transmitir eventos na forma de dados estruturados.

12) ***Collection Service*** **(Coleções):** este serviço permite criar agrupamentos de objetos que tenham comportamentos semelhantes, podendo eles serem acedidos via índice de objetos.

13) ***Property Service*** **(Propriedades):** este serviço permite adicionar propriedades aos objetos. Por exemplo, num sistema de *downloads*, é possível adicionar um contador ao objeto, que é incrementado sempre que é feito um *download*. Estas propriedades não fazem parte da implementação do objeto.

14) ***Query Service*** **(Procura):** tal como o nome do serviço indica, permite a procura de objetos através de *querys*. Por exemplo, numa base de dados orientada a objetos, pode ser utilizada a linguagem SQL para efetuar procuras.

15) ***Externalization Service*** **(Externalização):** serviço para gravar em disco ou em memória o estado de um objeto, para depois recarregar o objeto em outro momento.

16) ***Licensing Service*** **(Licenciamento):** este serviço permite criar um certo controlo de acesso aos objetos que se encontram no servidor, baseado em um contrato de licença de uso. É possível implementar o serviço de licenciamento que vai definir quais os tipos de clientes que podem ou não aceder os serviços de determinados objetos [8].

2.3. REST (*REPRESENTATIONAL STATE TRANSFER*)

Muitos dos serviços Web existentes não utilizam padrões Web para desenvolver as suas aplicações, resultando em fatores como a redução da interoperabilidade, aumento da latência da rede e futuros problemas na escalabilidade das aplicações [9]. O REST é um modelo que surgiu por Roy Fielding, idealizado para a comunicação entre duas aplicações, independentemente da plataforma/linguagem de desenvolvimento. O REST utiliza na troca de mensagens o protocolo HTTP, onde utiliza os métodos POST, GET, DELETE e PUT para manipular os dados de um recurso. Nesta arquitetura o cliente faz um pedido utilizando um URL, sendo que o serviço REST vai gerar um URI único para representar os dados. O REST trata uma aplicação Web como um conjunto de recursos, que representam um estado único da aplicação [10].

Um serviço Web utilizando o REST normalmente é implementado de duas formas diferentes:

- ***RESTFull* (ou *Hi-REST*):** é a implementação que faz uso de todas as potencialidades da arquitetura, utilizando todos os métodos do protocolo HTTP.

- ***Lo-REST*:** pode ser visto como a versão *lite* do REST, utiliza apenas os métodos GET e POST. Esta implementação é utilizada quando alguns *proxies* ou *firewalls* não permitem outros métodos que não esses. Para os métodos DELETE e PUT a estratégia que pode ser tomada, é utilizar um cabeçalho especial do protocolo HTTP (X-HTTP-*Method-Override*) ou então, pode ser utilizada uma variável específica para tal identificação do método.

O serviço REST permite o armazenamento em *cache*, a reutilização de interações, substituição dinâmica de componentes e o processamento de ações por parte de intermediários. Assim, responde as necessidades de um sistema distribuído com base na Web [11].

2.3.1. Elementos Arquiteturais do Serviço REST

O serviço REST na sua arquitetura distingue três elementos:

1) Elementos dos dados

2) Elementos de conexão (Conectores)

3) Elementos de processamento (Componentes)

2.4.1.2. Elementos dos Dados

O aspeto fundamental do REST é a origem e estado dos elementos dos dados. O REST identifica seis elementos dos dados: recurso, identificador do recurso, metadados do recurso, representação, metadados da representação e dados de controlo. Na Tabela 5 são apresentados os elementos de dados do serviço REST, seguido de um exemplo para cada elemento.

Elementos dos Dados	Exemplos Web
Recurso	O alvo pretendido de uma referência de *HyperText*
Identificador do Recurso	URL, URI
Representação	Documento HTML, Imagem JPEG
Metadados da Representação	Tipo de dados, tempo da última modificação
Metadados do Recurso	*Link* da fonte, *links* alternativos
Dados de Controlo	Controlo da *cache*

Tabela 5 - Elementos de dados do serviço REST

Recurso: Um recurso pode ser um documento, uma página Web ou até o resultado de uma pesquisa [12].

Os recursos são identificados e diferenciados dos outros, utilizando **identificadores de recursos**. Num ambiente Web estes identificadores são as URIs (*Uniform Resource Identifier*). Um recurso pode ser observado através das suas representações [9].

Representação: Uma representação é o resultado do que o recurso tem para apresentar e não o próprio recurso. Quando a representação de um recurso não sofre alterações, esse recurso é denominado de recurso estático. Por sua vez os recursos cujos valores se alteram ao longo do tempo e a sua semântica se mantenha, são chamados de recursos dinâmicos. A semântica (um conceito, uma identidade) é única e distingue um recurso do outro [12].

Metadados da Representação: Uma representação é constituída por uma sequência de *bytes* (o conteúdo) onde os metadados da representação descrevem essa sequência.

O formato de uma representação é chamado de *media type*. A escolha do *media type* da representação influencia o desempenho do sistema. Na maioria das aplicações Web o *media type* escolhido para as representações é o XML (*eXtensible Markup Language*) e JSON (*JavaScript Object Notation*) [9].

Metadados do Recurso: Os metadados do recurso descrevem o recurso, ou seja, fornecem informações que estão relacionadas com o recurso [10].

Dados de Controlo: São utilizados para parametrizar os pedidos e substituir o comportamento padrão de alguns elementos de ligação. O comportamento da *cache* pode ser alterado pelos dados de controlo incluídos numa mensagem de pedido ou resposta. Os dados de controlo definem o propósito de uma mensagem entre os componentes, tais como qual ação que está a ser solicitada ou o significado de uma resposta [14].

2.4.1.3. Conectores (Elementos de Conexão)

O REST utiliza vários conectores, que se encontram resumidos na Tabela 6. Esses conectores são utilizados para encapsular o acesso aos recursos e à transferência das representações dos recursos. Os conectores fornecem uma *interface* abstrata para comunicar com os componentes, aumentando assim a simplicidade e escondendo a implementação dos mecanismos de comunicação subjacentes aos recursos.

Conector	**Funções**
Cliente	Envia e recebe pedidos.
Servidor	Escuta pedidos e envia respostas.
Cache	Pode estar localizada no cliente ou no servidor, para armazenar respostas e pode ser compartilhada entre vários clientes.
Solucionador (resolver)	Transforma identificadores de recursos em endereços de rede.
Túnel	Permite que os componentes mudem dinamicamente o seu comportamento de ativo para o modo túnel.

Tabela 6 - Conectores do serviço REST

Em todas as interações REST cada pedido possui todas as informações necessárias para que um conector possa compreender o pedido, independentemente do pedido que o precedeu. Este procedimento traz as seguintes vantagens:

1) O conector não tem de manter o estado da aplicação entre os pedidos, reduzindo o consumo de recursos físicos e melhorando a escalabilidade.

2) Permite que as interações sejam processadas em paralelo.

3) Permite um intermediário para ver e entender um pedido de forma isolada, o que pode ser necessário e vantajoso, quando os serviços são reorganizados de forma dinâmica.

4) Obriga que toda a informação que é dada numa resposta esteja em *cache*.

2.4.1.4. Componentes

Os componentes REST são identificados pelo papel que desempenham dentro da aplicação Web [9]. De seguida são detalhados os componentes REST:

Agente de Utilizador (*User agent*): Usa o conector cliente para efetuar um pedido e receber a resposta ao pedido [14].

Servidor de Origem (*Origin server*): Usa o conector servidor para receber o pedido e é a fonte para as representações dos recursos. Cada servidor fornece uma *interface* genérica para os serviços que implementa, funcionando como uma hierarquia de recursos.

Componentes intermédios (*Intermediary componentes*): agem como conectores cliente e servidor, com o objetivo de transmitir os pedidos e respostas.

2.3.2. Exemplo de um serviço Web utilizando REST

De forma a entender-se melhor a arquitetura REST e os conceitos detalhados anteriormente, segue-se um exemplo muito simples de um serviço Web utilizando o REST.

Neste exemplo o serviço Web disponibiliza as seguintes funcionalidades:

- O utilizador pode fazer *upload* e eliminação de fotos.

Na Figura 7 é representado de forma geral o processo para aceder a um recurso utilizando o serviço REST.

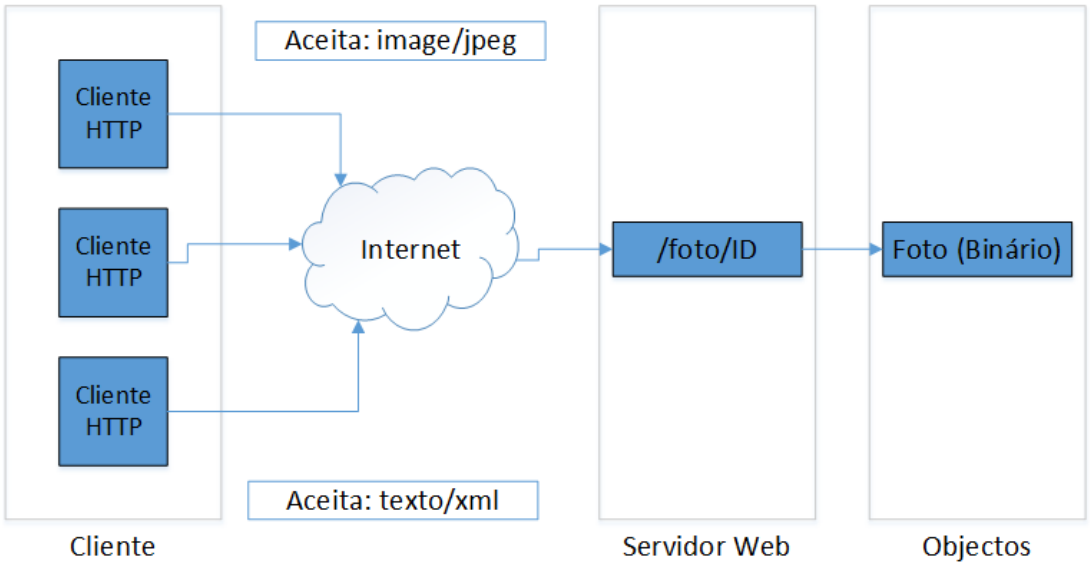

Figura 7 – Exemplo REST: Visão geral do serviço Web

As fotos são o recurso deste serviço. Associado a este recurso está a seguinte representação: representação em binário.

Para este exemplo o recurso é identificado e endereçado recorrendo a URI. Apenas os recursos podem ser endereçados e não as representações. Para aceder ao recurso Foto é utilizado o seguinte URI: /foto/ID. Onde o ID representa o identificador pelo qual a foto é conhecida no servidor Web.

De forma a interagir com o recurso do serviço Web, a especificação HTTP define um conjunto de métodos, mas apenas quatro são relevantes para a arquitetura REST. São eles: GET, POST, PUT e DELETE. A função de cada método é resumida na Tabela 7.

Método	**Função**
GET	Retorna o recurso
POST	Inserir, atualizar ou acrescentar dados num recurso
PUT	Criar, atualizar ou substituir um recurso
DELETE	Apagar um recurso

Tabela 7 - Métodos REST

Neste exemplo (ver Figura 8) ao utilizar o método GET aplicado à URI: /foto/ID (onde o ID é o nome que identifica o recurso, neste exemplo a foto) é retornada a representação binária (ou seja, a foto) correspondente ao ID. Quando o pedido GET é bem-sucedido é retornado como resposta o código 200, a informar do sucesso da operação.

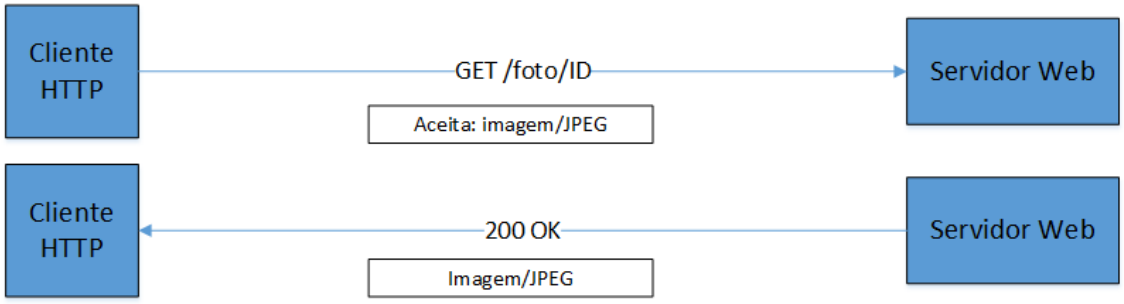

Figura 8 – Exemplo REST: Método GET

Neste exemplo o método POST não se insere em nenhuma funcionalidade do serviço. Porém o método POST poderia ser utilizado para adicionar metadados às fotos, ou seja, através do método POST seria possível adicionar propriedades como longitude e latitude.

Para o *upload* de fotos (ver Figura 9) é utilizado o método PUT. Para realizar essa operação é necessário o utilizador autenticar-se via HTTP AUTH. A resposta ao método PUT tem como código o número 201 que indica a criação do recurso e retorna também a URI que identifica o recurso criado. O método PUT é aplicado a URI: /foto/.

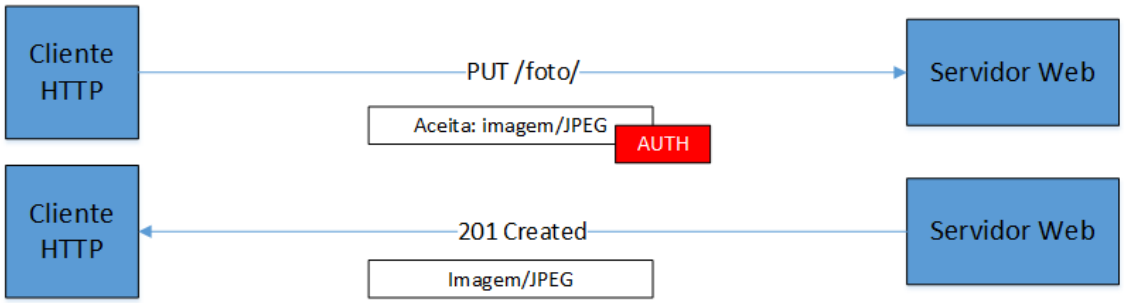

Figura 9 - Exemplo REST: Método PUT

Por fim o método DELETE é utilizado para eliminar um recurso. Este método para o presente exemplo (ver Figura 10) é aplicado utilizando a URI: /foto/ID (onde o ID é o nome que identifica a foto que será eliminada), e necessita que o cliente se autentifique no servidor. Quando o servidor aceita o pedido é devolvida uma mensagem para o cliente com o código 202, a informar que o pedido foi aceite [9] [12].

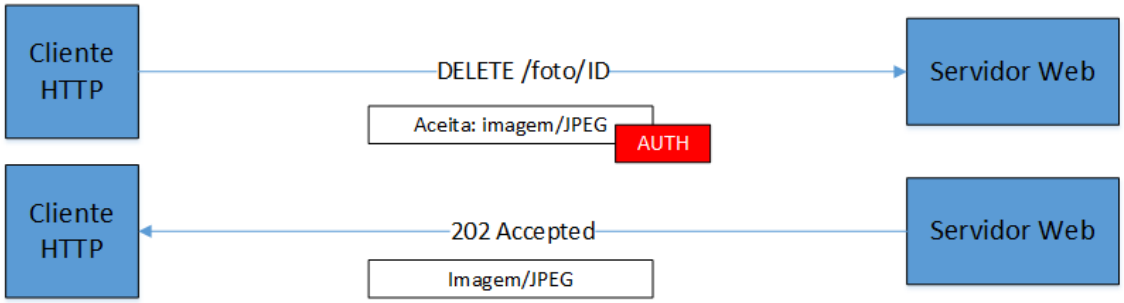

Figura 10 - Exemplo REST: Método DELETE

Neste capítulo, foi feito um estudo detalhado da tecnologia Bluetooth, da arquitetura CORBA e do que o REST oferece na implementação de um serviço Web. A finalidade de estudar estas componentes, foi a de, reunir informações sobre o Bluetooth e da arquitetura CORBA e assim entender como estas tecnologias são utilizadas e implementadas no sistema BlueStation.

Nos próximos capítulos é apresentado todos os componentes do sistema BlueStation e descrito todo o processo de desenvolvimento da nova aplicação que permite criar os conteúdos e associá-los as estações.

3. VISÃO GERAL DO SISTEMA BLUESTATION

Neste capítulo é descrito de uma forma muito geral o sistema existente, utilizado para controlar todas as BlueStations (ver Figura 11). O sistema é composto por um conjunto de estações chamadas BlueStation, que são pequenos computadores que estão equipados com dois *dongles* Bluetooth e uma *pen* de banda larga 3G (ver Figura 12).

Figura 11 – Esquema Geral do Sistema BlueStation

Esta configuração permite que o primeiro *dongle Bluetooth* tenha como função procurar todos os dispositivos Bluetooth ativos e visíveis na vizinhança da BlueStation, enquanto o outro *dongle* é responsável por enviar serviços aos utilizadores de equipamentos móveis que foram detetados pelo *dongle* de procura. A *pen* de banda larga 3G é responsável por permitir a comunicação com a unidade central (que se encontra no servidor, instalado na estação central dos Horários do Funchal). Este servidor dá suporte à aplicação de gestão e criação de serviços, armazena nas bases de dados todas as informações relacionadas com as BlueStations e a aplicação.

Quando as estações são iniciadas, elas contatam o componente *Broker* que faz parte da arquitetura CORBA. Este componente é um intermediário das comunicações entre as estações com a unidade central e vice-versa. Além disso faz com que as estações se registem no sistema com o nome da estação e não pelo seu IP. Isto é feito porque a *pen* de banda larga 3G possui IP dinâmico e por isso as estações são conhecidas pelo utilizador e pela unidade central através do nome da estação e não pelo IP.

O *dongle* usado para descobrir dispositivos Bluetooth comunica com o componente do sistema chamado agendamento de serviços. Assim que é encontrado um dispositivo, o componente verifica se o serviço deve ser executado. O componente de agendamento de serviços é uma pilha constituída por vários serviços, que com base no endereço MAC da placa Bluetooth do

dispositivo encontrado pelo *dongle*, analisa na pilha os serviços existentes e verifica se estão reunidas as condições para que o conteúdo do serviço seja enviado ao dispositivo encontrado.

Figura 12 – BlueStation (Estação)

Um serviço é constituído por um conjunto de atributos que são definidos durante a sua criação. Estes atributos são:

- Nome que é dado ao serviço;
- Conteúdo que é enviado aos utilizadores (este conteúdo pode ser de texto ou imagem);
- Estação ou estações que irão receber o serviço;
- Data e hora a que o serviço deve se iniciar e finalizar;
- Dispositivos a quem se destina o serviço (podem ser todos os dispositivos na vizinhança da BlueStation ou um conjunto de dispositivos específicos);
- Tempo, em segundos, que um dispositivo tem de estar visível para a BlueStation a fim de receber o serviço;
- Outros atributos específicos da tecnologia Bluetooth que não são relevantes para está dissertação e são facultativos para a criação de um serviço.

O sistema possui uma funcionalidade chamada de *Blacklist*, que tem como função bloquear determinado dispositivo através do seu endereço MAC. Esta funcionalidade pode ser utilizada

por exemplo, a pedido do utilizador caso este queira deixar de receber conteúdos no seu dispositivo, bastando para isso, fornecer ao administrador do sistema BlueStation o endereço MAC da placa Bluetooth do seu equipamento. Cada estação tem a sua própria *Blacklist*, ou seja, um dispositivo pode estar apenas bloqueado em uma estação e noutra não. Durante a execução dos serviços e antes do conteúdo ser enviado aos utilizadores a *Blacklist* é analisada e os endereços MAC que estiverem nesta lista são automaticamente excluídos de receberem qualquer serviço.

A unidade central está implementada num servidor dedicado para o sistema. Está unidade é constituída por duas bases de dados e a aplicação utilizada para comunicar remotamente com as estações, criar e gerir todos os serviços associados a elas. A base de dados SQL guarda todas as informações relacionadas com os dispositivos Bluetooth e com os serviços.

A aplicação para a criação e gestão dos conteúdos que são enviados para as estações está escrita em Perl e utiliza a arquitetura MVC (*Model-View-Control*) na sua implementação. A aplicação permite fazer a gestão de todas as estações, tal como modificar as configurações, associar serviços e gerir a *Blacklist* de dispositivos.

3.1. ARQUITETURA DE ALTO NÍVEL DO SISTEMA BLUESTATION

A Figura 13 representa a arquitetura do sistema de um ponto de vista muito genérico, focando-se nos 3 componentes (BlueStation, *Broker* e Unidade Central) que servem de base a todo o sistema e que vão ser vistos em mais detalhe ao longo desta dissertação.

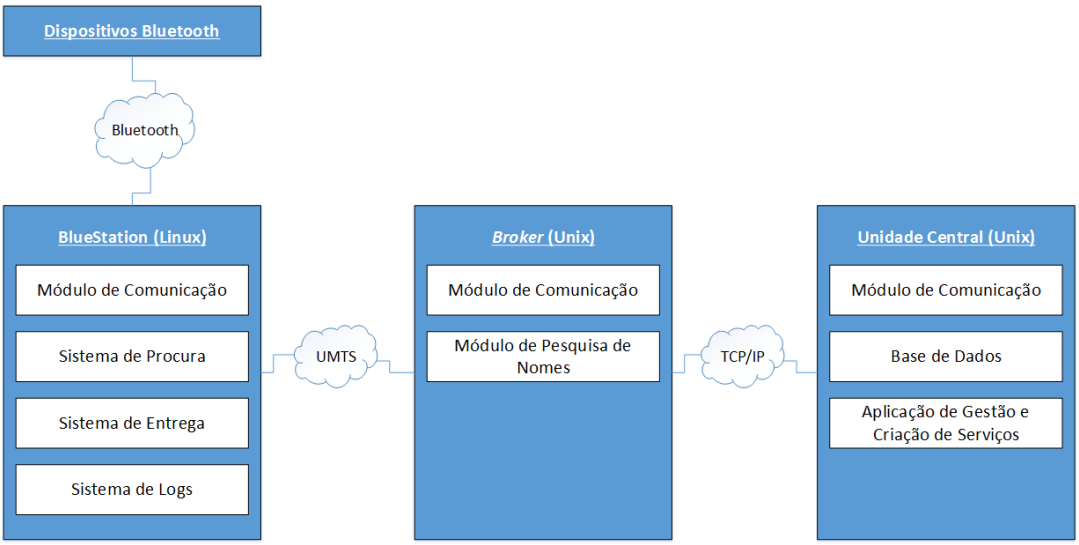

Figura 13 – Arquitetura de Alto Nível do Sistema

A componente BlueStation (Estação) representa as estações que se encontram instaladas em diferentes locais. Os dispositivos Bluetooth são os que enviam os conteúdos para o utilizador final, utilizando a tecnologia Bluetooth.

A componente *Broker* é um intermediário na comunicação entre as estações e a unidade central. O *Broker* faz a tradução dos IP das estações para nomes, ou seja, a estação é conhecida no sistema pelo seu nome e não pelo seu IP.

A componente Unidade Central é onde é feita toda a gestão das estações. Nesta unidade encontra-se a base de dados com as informações de tudo o que acontece no sistema e a aplicação de criação e gestão de conteúdos associados (serviços) as estações.

3.1.1. Componente BlueStation (Estação)

Como já foi referido anteriormente a BlueStation é um computador configurado com o sistema operativo Linux. As configurações Bluetooth são estabelecidas pela biblioteca BlueZ e é usado o OpenOBEX para enviar os conteúdos ao utilizador final, através do protocolo OBEX. A maioria dos componentes que compõem o sistema estão programados em Perl, mas para algumas funções externas, foram programados em Bash.

A BlueStation é a base que constitui este sistema distribuído. Cada estação possui três elementos físicos, dois *dongles* Bluetooth e uma *pen* de banda larga 3G. O elemento que está

constantemente ativo é o *dongle* que faz a procura por dispositivos Bluetooth na sua vizinhança. Este *dongle* é de classe 2 por razões de alcance, visto só interessarem os dispositivos que estão próximos das BlueStations. Se o ambiente em que as estações estão instaladas for muito propício a interferências, pode ser usado um *dongle* de classe 3, reduzindo ainda mais o raio de alcance. O segundo *dongle* utilizado para enviar os conteúdos aos dispositivos Bluetooth que se encontrem na vizinhança da BlueStation é de classe 1. É utilizada esta classe, por ter um raio de alcance maior que a classe 2 ou 3 e assim fazer chegar ao utilizador o conteúdo mesmo que esse esteja em movimento ou saia do raio de alcance do *dongle* de procura.

O facto de o *dongle* de entrega utilizar a classe 1 não representa um problema para as interferências, porque o conteúdo só é enviado aos dispositivos detetados pelo *dongle* de procura, que utiliza a classe 2 ou a 3. As estações são capazes de funcionar com apenas um *dongle* Bluetooth que faça a procura de dispositivos e a entrega dos conteúdos. Essa situação é desaconselhável porque os resultados de desempenho são muito pobres. No cenário atual, as estações estão a utilizar um *dongle* para enviar os conteúdos e um *dongle* para procura, mas o sistema está configurado para suportar mais *dongles* que façam a entrega de conteúdos e assim ter uma maior resposta, num cenário em que existam muitos dispositivos. De notar que o número de *dongles* suportados vai depender da potência que a entrada USB da estação possui. Ao aumentar o número de *dongles* Bluetooth está a ser introduzida também uma maior interferência entre os *dongles*.

Fisicamente, as estações possuem apenas duas portas USB, sendo que uma das portas está a ser ocupada por uma *pen* de banda larga 3G e na outra porta encontra-se um *hub* de 4 portas USB, onde são conectados os *dongles* Bluetooth (Ver Figura 14). O facto de a estação ter que usar um *hub* USB (que é alimentado pela estação) e a potência da porta USB da estação ser dividida pelas 4 portas do *hub*, limita a BlueStation a só suportar um *dongle* para a entrega dos conteúdos e outro para a procura. O último elemento físico, a *pen* de banda larga 3G é usada para a ligação a internet e para comunicar com a unidade central.

Figura 14 - BlueStation - Elementos Físicos

A BlueStation é composta por seis elementos lógicos, (ver Figura 15) sendo que alguns comunicam entre si de forma bidirecional e outros unidirecional. Esses seis elementos são: sistema de procura, módulo de publicação, agendamento de serviços, sistema de entrega, sistema de *logs* e sistema de comunicação.

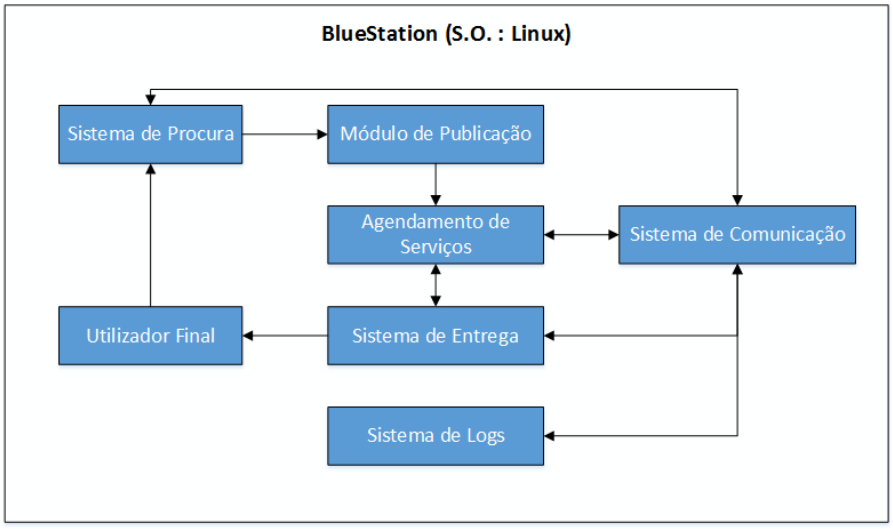

Figura 15 - Visão geral dos Componente de Software da BlueStation

3.1.1.1. Sistema de Procura

O sistema de procura é responsável por encontrar os dispositivos Bluetooth ativos e visíveis no raio de alcance da estação. Este elemento está escrito em Perl e em Bash. A aplicação que comunica com o *dongle* de Bluetooth está programada no ficheiro btreader.sh e está totalmente escrita em Bash. Esta aplicação faz uso da *interface* de configuração fornecida pela componente hcitool que faz parte da biblioteca BlueZ. Está aplicação fornece como *output* o endereço MAC do dispositivo Bluetooth encontrado, o nome pelo qual é identificado, data e hora em que foi encontrado e a classe, em hexadecimal, associada ao dispositivo. Esta classe permite reconhecer a que categoria o dispositivo pertence, ou seja, se é um *smartphone* ou um computador, por exemplo.

Esta aplicação permite alterar algumas configurações tais como: diferença de tempo entre cada procura por dispositivos Bluetooth, nome que é dado ao *dongle* Bluetooth e em que porta a aplicação deve se ligar para comunicar com o *dongle*.

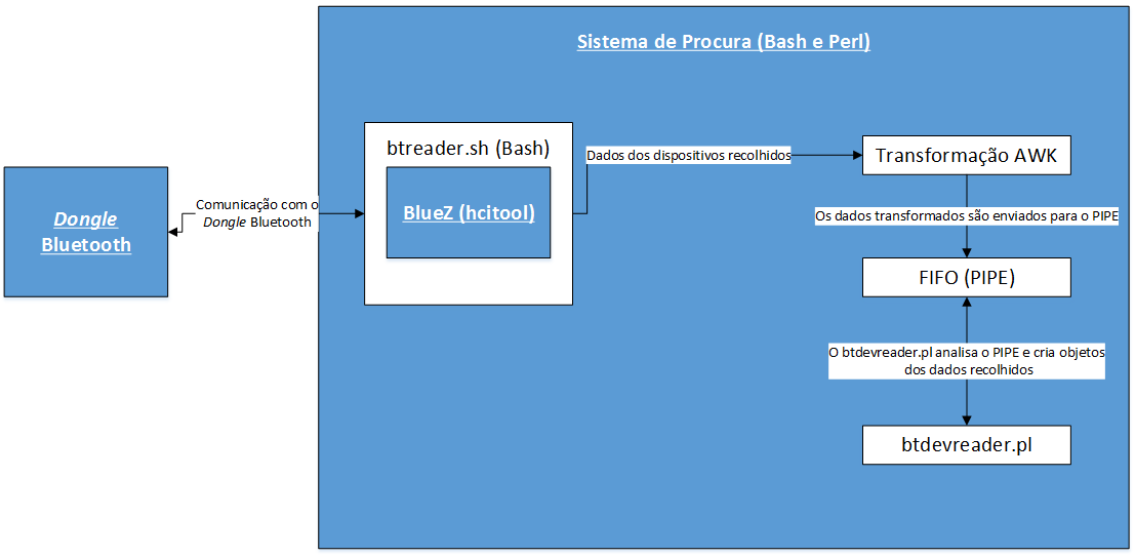

Figura 16 - Elementos de Software e Funcionamento do Sistema de Procura

Os dados recolhidos pela aplicação, sofrem uma transformação AWK e são enviados para um PIPE chamado FIFO (*First In*, *First Out*), que vai ser lido por outro componente chamado btdevread.pl que está escrito em Perl (ver Figura 16). Com os dados que são recebidos pelo PIPE, é criado um objeto desses dados e uma lista. Quando a lista dos elementos é atualizada com nova informação, este componente envia a lista de dispositivos recolhidos para o sistema de publicação utilizando IPC (*Inter-process Communication*) – *sockets* de domínio UNIX.

3.1.1.2. Módulo de Publicação

O módulo de publicação (ver Figura 17) utiliza o padrão de desenho Produtor/Subscritor. Este módulo disponibiliza 5 canais distintos para subscrição, são eles:

1) **DEV_ENTRY:** Indica que um novo dispositivo que não foi detetado num passado recente entrou no raio de alcance do *dongle* Bluetooth.

2) **DEV_EXIT:** Indica que os dispositivos que estavam no raio de alcance do *dongle* de procura, deixaram de estar nesse raio de alcance por um determinado período de tempo e consequentemente foram excluídos da lista de dispositivos.

3) **DEV_UPDATE:** Indica os dispositivos que existem atualmente no raio de alcance do *dongle* de procura.

4) **DEV_ENTRY_UPDATE:** Junta os canais DEV_ENTRY e DEV_UPDATE.

5) **DEV_ALL:** Refere-se a todos os canais mencionados.

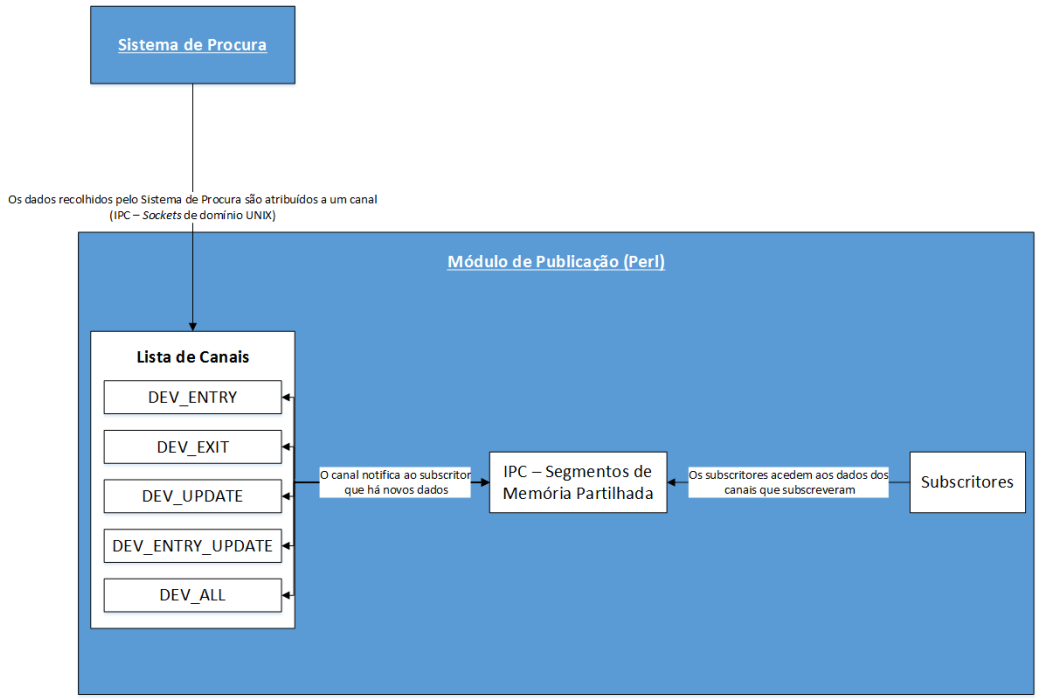

Figura 17 - Elementos de Software e Funcionamento do Módulo de Publicação

O módulo de publicação (ver Figura 17) funciona usando *sockets* de domínio UNIX e todo o módulo está escrito em Perl. O módulo de publicação espera que o sistema de procura envie os dados dos dispositivos capturados e consoante os dados recebidos, é atribuído um canal a estes dados. Quando os dados são atribuídos a um canal o subscritor desse canal é notificado sobre o novo conteúdo. De notar que o conteúdo é transmitido aos subscritores utilizando o método mais eficiente IPC – segmentos de memória partilhada.

3.1.1.3. Agendamento de Serviços

O agendamento de serviços (ver Figura 18) é atualmente o único subscritor do canal DEV_ENTRY_UPDATE. Este componente é responsável pela gestão da fila de serviços, fazendo a tradução de objetos de serviços para objetos comando e pela manutenção dos endereços na *Blacklist*.

O componente agendamento de serviços é um programa que utiliza segmentos de memória partilhada para comunicar entre os processos dos componentes locais. Este componente é notificado sempre que haja um novo dispositivo detetado ou quando a informação do dispositivo é atualizada. A verificação da lista de dispositivos é feita de uma forma síncrona, juntamente com o processo principal que analisa a fila de serviços, a lista de dispositivos e determina se um serviço deve ser executado.

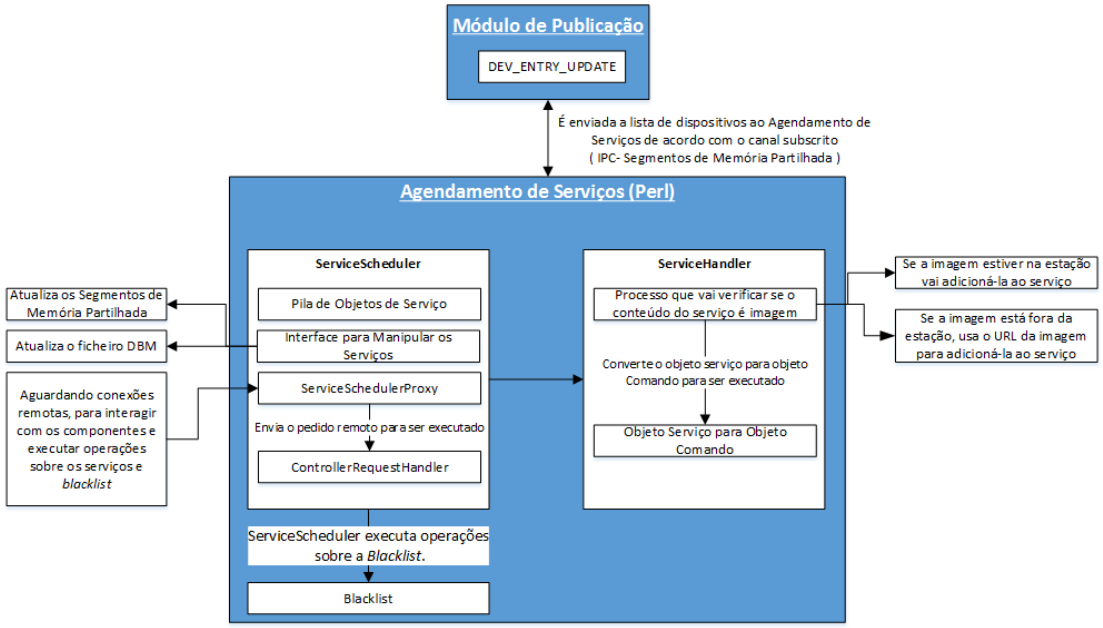

Figura 18 - Elementos de Software e Funcionamento do Agendamento de Serviços

Para um serviço ser executado tem de verificar um conjunto de requisitos presentes na pilha de objetos de serviço. A classe responsável por esta verificação é a ServiceScheduler, que quando o serviço cumpre com todos os requisitos marca-o como executável. A classe ServiceScheduler passa a responsabilidade de executar o serviço para a classe ServiceHandler (ver Figura 18).

A classe ServiceHandler vai criar um novo processo em que a primeira tarefa é verificar se o conteúdo que o serviço vai enviar ao utilizador final é de texto ou imagem. No caso de ser texto, não é necessário fazer qualquer operação, porque o conteúdo já se encontra dentro do objeto serviço quando este é criado. No caso de o conteúdo ser uma imagem é necessário ir buscar o conteúdo e associa-lo ao objeto serviço. O passo seguinte é traduzir o objeto serviço para um objeto comando. Feito isto, é aberta uma conexão usando um *socket* de domínio UNIX com o sistema de entrega que vai executar o objeto comando.

A classe ServiceScheduler implementa a *interface* para manipular o serviço atual. Manipular o objeto serviço, consiste em atualizar os segmentos de memória partilhada e o ficheiro DBM para que a persistência dos dados do serviço seja mantida. A atualização de um serviço que passa de ativo para não ativo, não é refletida no ficheiro DBM, por questões de desempenho.

Por fim o componente ServiceScheduler fica à escuta de conexões externas, permitindo ser controlado remotamente. A ligação remota é implementada pela classe ServiceSchedulerProxy que encarrega a execução ao ControllerRequestHandler. O ControllerRequestHandler fica responsável por enviar a resposta a quem emitiu o pedido. A estratégia de encarregar o pedido remoto a outro componente do sistema é baseada no padrão de desenho Proxy.

O processo de funcionamento do agendamento de serviços é esquematizado na Figura 18.

3.1.1.4. Sistema de Entrega

O sistema de entrega (ver Figura 19) é responsável por entregar ao utilizador final o conteúdo dos serviços. Depois do agendamento de serviços fazer a tradução do objeto serviço para objeto comando, é da responsabilidade do sistema de entrega, tentar a execução do serviço e fazer o conteúdo chegar ao utilizador final.

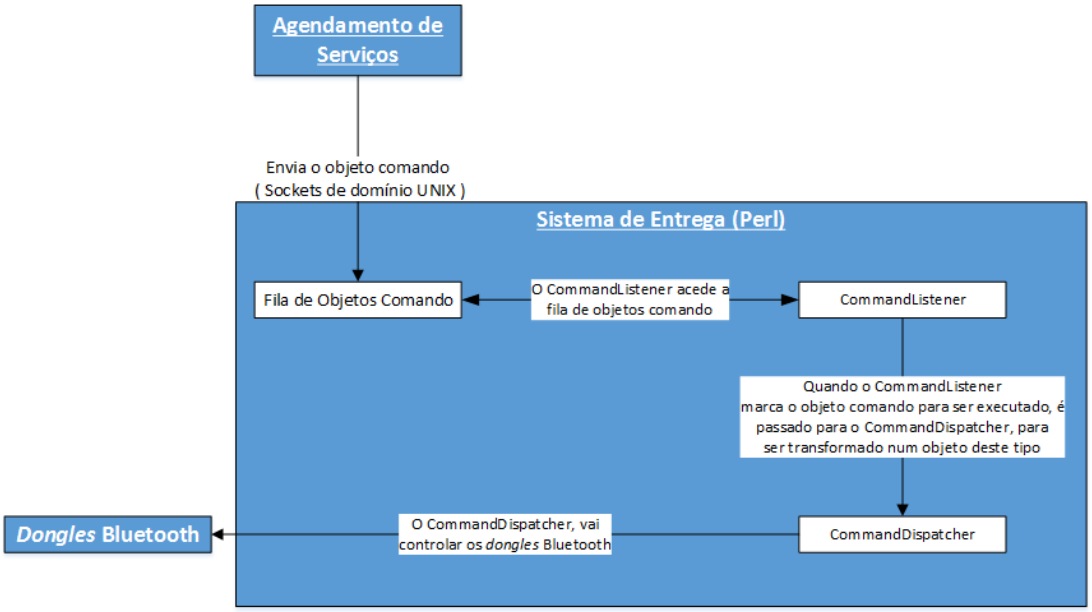

Figura 19 - Elementos de Software e Funcionamento do Sistema de Entrega

Tal como o componente agendamento de serviços, o sistema de entrega funciona num ambiente de múltiplos processos. Na sua inicialização são criados dois processos e quando um comando tem de ser executado, um novo processo é criado. A comunicação entre os processos é a mesma usada no agendamento de serviços, ou seja, o uso de segmentos de memória partilhada.

O componente CommandListener que faz parte do sistema de entrega, é o primeiro a receber o comando que é enviado pelo agendamento de serviços, utilizando *sockets* de domínio UNIX. Este componente vai executar uma de duas operações: adiciona o comando à fila ou marca o comando para ser executado.

Quando o comando é para ser executado, tem de ser transformado num objeto do tipo CommandDispatcher. O CommandDispatcher possui a lista das portas Bluetooth que possuem *dongles*, para que possa colocar o *dongle* em uso como ocupado. Além disso, e no caso de a estação estar a funcionar com um único *dongle* Bluetooth para fazer a procura de dispositivos e entrega de conteúdos, o CommandDispatcher controla o dispositivo para que as duas operações possam ser sincronizadas.

3.1.1.5. Sistema de Comunicação

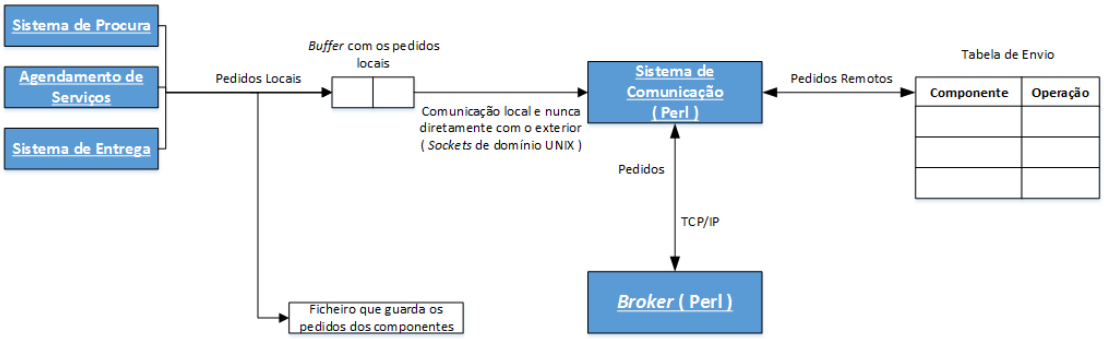

Figura 20 – Elementos de Software e Funcionamento do Sistema de Comunicação

O sistema de comunicação (ver Figura 20) é responsável por se ligar aos componentes exteriores, funcionando de forma similar a um *proxy*. Como foi visto nos componentes anteriores, no sistema de procura, agendamento de serviços e sistema de entrega, implementam-se mecanismos de controlo remoto, mas estes componentes só executam ligações que usam *sockets* de domínio UNIX.

Pensando na segurança do sistema, este componente foi criado exclusivamente para comunicações usando a internet e para que existisse um componente que não expusesse os componentes internos diretamente ao exterior.

Portanto o sistema de comunicação é o único que suporta pedidos locais e remotos. Para pedidos locais realizados por parte dos componentes locais que queiram comunicar com a unidade central, o sistema de comunicação envia o pedido ao *Broker* (que será detalhado mais à frente) que se encarregará de enviar à unidade central. Os pedidos dos componentes locais são enviados primeiro para um *buffer* de pedidos e guardados num ficheiro, sendo que o sistema de comunicação periodicamente analisa esse *buffer* e verifica se há conteúdos para enviar.

Para os pedidos remotos é utilizada uma tabela de envio, que regista qual o componente que fez o pedido e a operação desejada. Todo o processo de funcionamento do sistema de comunicação é exemplificado na Figura 20.

3.1.1.6. Sistema de *Logs*

A BlueStation possui um sistema de *logs* que é iniciado no arranque das estações, utilizando para isso o agendamento de tarefas do Linux, o crontab. O sistema de *logs* realiza a sua tarefa todos os dias às 23:59. A tarefa consiste em recolher todos os ficheiros de log, criados pelos componentes que constituem o sistema, comprimi-los num único ficheiro e enviar para o *buffer* de pedidos do componente de comunicação, que por sua vez envia para a unidade central. Todo o processo de funcionamento do sistema de *logs* é exemplificado na Figura 21.

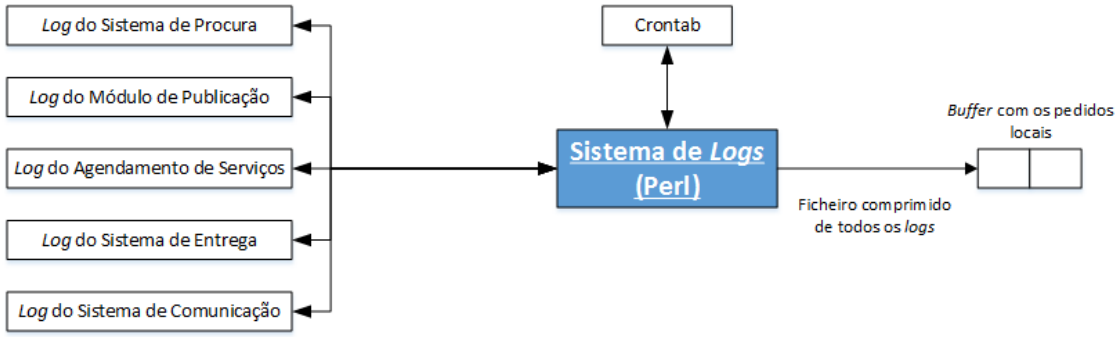

Figura 21 - Elementos de Software e Funcionamento do Sistema de Logs

3.1.2. *Broker*

O *Broker* (ver Figura 22) é o componente intermediário que é envolvido em todas as comunicações entre as BlueStations e a unidade central. O *Broker* é usado como servidor de nomes, mas também como encaminhador de pedidos/respostas.

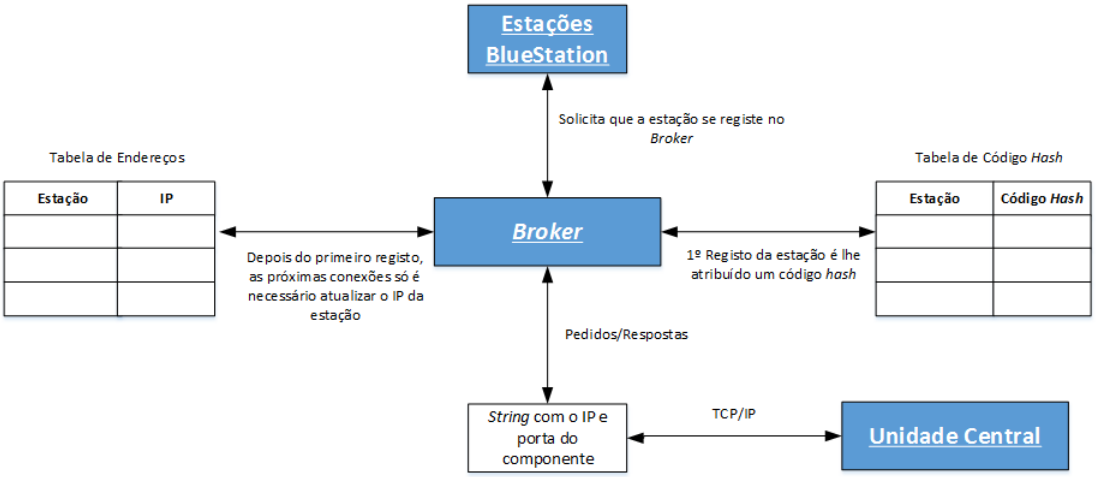

Figura 22 - Elementos de Software e Funcionamento do Broker

O *Broker* é usado em todas as ligações feitas entre as BlueStations e o servidor (nomeadamente, para a aplicação de gestão e criação de serviços como para a base de dados). As estações não estão habilitadas a receber ligações diretas do exterior, por isso são utilizadas técnicas de *tunneling* (Túnel) por onde a troca de informação deve passar.

O *Broker* mantém uma estrutura de dados que lhe permite associar o nome de uma estação a um endereço IP concreto. Esta estrutura é uma tabela de *hash*. Antes que os outros componentes possam comunicar com o *Broker*, a estação tem de ser a primeira a fazê-lo. Este processo é feito quando a estação é iniciada, e realiza um pedido ao *Broker* para se registar. Se a estação já está registada o que tem de fazer é solicitar a substituição do endereço IP, porque como o IP é atribuído dinamicamente, o endereço pode obviamente mudar.

Para uma estação não se fazer passar por outra, é implementado um mecanismo muito simples. No momento em que a estação é registada no *Broker* é lhe atribuído um código *hash* que lhe vai permitir identificar-se no *Broker* em pedidos futuros.

Como existem várias estações o *Broker* tem de responder a todos os pedidos e isso é feito usando múltiplos processos. Quando um novo pedido é feito, um novo processo filho é criado. Se o pedido é para ser encaminhado para uma estação ou para a unidade central, o *Broker* irá localizar o componente e estabelecer a ligação com ele. Este processo é independente do processo de registo, onde o *Broker* apenas mantém uma *string* composta pelo endereço IP e a porta do componente. Todo o processo de funcionamento do *Broker* é exemplificado na Figura 22.

3.2.3 Unidade Central

A unidade central (ver Figura 23) é o centro de controlo de todo o sistema. É na unidade central que toda a informação é armazenada, nas bases de dados existentes. Existem duas bases de dados, a blue_station.sql e a blue_synch.sql.

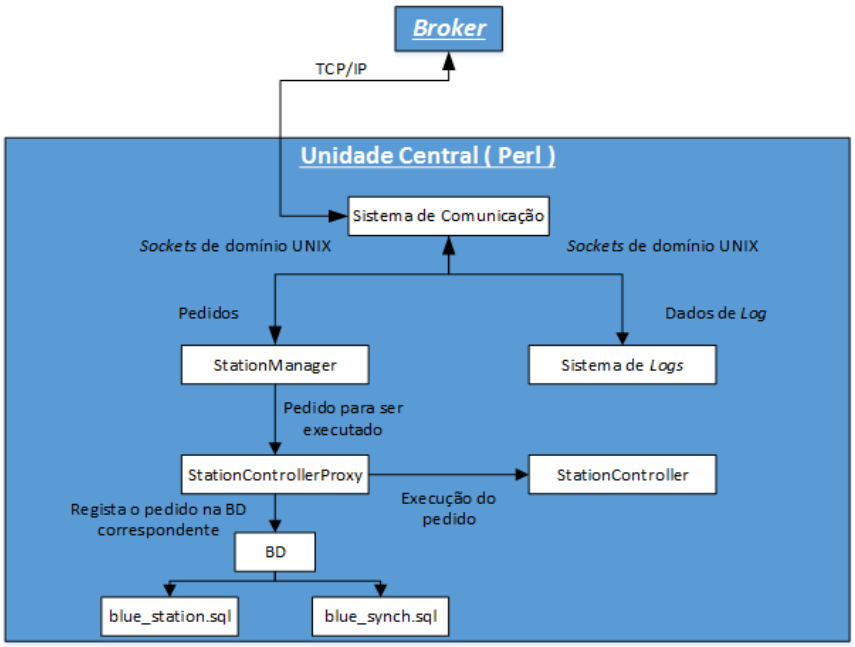

Figura 23 – Elementos de Software e Funcionamento da Unidade Central

A blue_synch.sql armazena as operações que são realizadas, tais como:

- Serviços adicionados
- Serviços removidos
- Adicionar um endereço a *Blacklist*
- Remover um endereço da *Blacklist*
- Identificação de quem realizou a operação
- Hora e data da operação
- Em que estação foi realizada a operação

A blue_station.sql armazena todas as informações das estações, tais como:

- Dispositivos recolhidos
- Serviços executados
- Dispositivos que recebem os serviços
- Dispositivos que não aceitaram os serviços
- Falhas ocorridas no envio de serviços

É na unidade central que é possível interagir com as estações de forma remota, através da aplicação de criação e gestão de serviços. Nesta aplicação é possível criar e eliminar serviços, adicionar e remover endereços da *Blacklist* e alterar configurações do *dongle* Bluetooth de procura.

O componente mais importante da unidade central é o gestor das estações. Este componente possibilita a comunicação remota com as estações e a comunicação com as bases de dados. O gestor das estações utiliza múltiplos processos que ficam à escuta de pedidos usando *sockets* de domínio UNIX.

Quando é feito um pedido para o gestor de estações, a classe StationManager cria um novo processo filho e um novo objeto do tipo StationControllerProxy que vai lidar com o pedido. Quando a classe StationControllerProxy recebe o pedido, regista o pedido na base de dados e só depois a classe encarrega a classe StationController da execução do pedido. É na classe StationController que se encontram todos os métodos relacionados com o controlo dos serviços, *Blacklist* e configurações das estações.

Como foi referido na descrição do sistema de *logs* das BlueStations, todas as operações realizadas são registadas num ficheiro de *log* que depois é enviado à unidade central. É da responsabilidade do componente gestor de *logs* que se encontra na unidade central, gerir estes ficheiros e armazená-los num arquivo correspondente a cada estação.

Tal como as estações, a unidade central tem de se registar no *Broker*. O componente chamado sistema de comunicação, presente na unidade central, utiliza a ligação à internet para efetuar o registo e também encaminhar os pedidos que chegam e saem da unidade central. Todo o processo de funcionamento da unidade central é exemplificado na Figura 23.

4. BLUESTATION - APLICAÇÃO CONSOLA DE ADMINISTRAÇÃO

Neste capítulo será feita uma descrição das componentes que constituem a aplicação existente para criação e gestão de serviços, bem como a descrição de todas as suas funcionalidades.

4.1. FUNDAMENTAÇÃO TECNOLÓGICA E FUNCIONALIDADES

A consola de administração está escrita em Perl e implementa o padrão arquitetural MVC (ver Figura 24). A consola da administração permite fazer toda a gestão e criação dos conteúdos (ou, serviços) que são adicionados as BlueStations. Esta aplicação interpreta um conjunto de comandos pré-definidos para executar uma operação e tem como *interface* a linha de comandos (ver Figura 25).

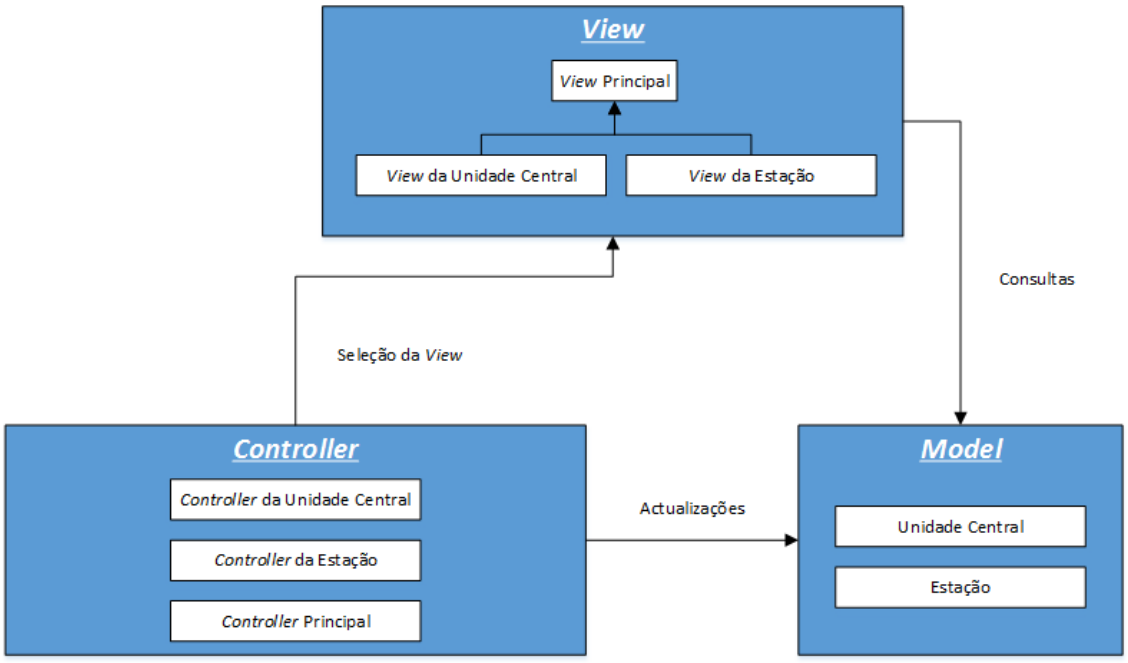

Figura 24 - Modelo MVC da Consola de Administração

Tal como a unidade central, a aplicação tem de se conectar ao *Broker*, para que através dos servidores de nomes, seja possível comunicar com as estações.

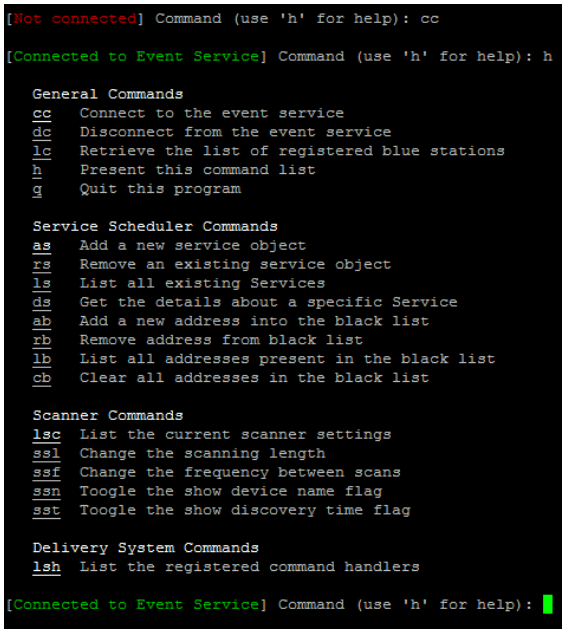

Figura 25 - Interface da Consola de Administração

A consola de administração está dividida em 2 tipos de comandos: comandos gerais e comandos para gestão dos serviços (ver Figura 25). Estes tipos de comandos são detalhados na secção que se segue.

4.1.1. Comandos Gerais

Quando a aplicação é iniciada a primeira informação que aparece é a seguinte:

[Not connected] Command (use 'h' for help):

Usando o comando ‘h’ o administrador tem acesso aos seguintes comandos *offline*, que são descritos a seguir:

- **cc –** Conectar-se ao serviço de eventos (ou seja, ter acesso ao menu dos comandos que permitem fazer a gestão e criação de serviços).
- **dc –** Desconectar-se do serviço de eventos.
- **lc –** Devolve a lista das estações registadas no sistema.
- **h –** Apresenta o menu de ajuda ao administrador.
- **q –** Sai da consola de administração**.**

Usando o comando '**lc**', é apresentada a lista das estações registadas, através do seu nome de serviço. Segue-se um possível resultado na execução deste comando:

[Not connected] Command (use 'h' for help): lc

Registered Stations:

Savoy, Forum, Miramar, CcBomJesus, Palacio, Autonomia, Universidade, Infancia, MadeiraShopping, Infante.

4.1.2. Comandos de Gestão dos Serviços

Quando o administrador pretende aceder ao serviço de eventos e interagir com todas as estações que estão registadas no sistema utiliza o comando '**cc**'. Após a ligação bem-sucedida, a barra de estado da aplicação reflete a mudança da seguinte forma:

[Connected to Event Service] Command (use 'h' for help):

Depois de já estar conectado ao serviço de eventos, o comando '**h**' irá mostrar uma lista renovada de comandos que podem ser executados e que são detalhados em seguida.

- **Comando 'ls':** Permite listar os serviços de uma ou mais estações

Se se pretender verificar quais os serviços que existem em uma determinada estação ou estações, emite-se o comando **'ls'**. Por exemplo, se se desejar ver quais serviços que estão presentes nas estações 'Miramar' e 'CCBomJesus' são executados os seguintes passos:

[Connected to Event Service] Command (use 'h' for help): ls

Enter location(s): Palacio Universidade

Service list for station(s) Palacio Universidade issued
Service list for station 'Palacio'
Current Registered Service List
ID '7' - Agenda Cultural - 09/08/13 – *;23:59
Service list for station 'Universidade'
Current Registered Service List
ID '9' - Atraso Carreira 13 - 10/08/13 – 16:30;17:00

Neste exemplo ambas as estações têm um serviço cada, para ser executado no agendamento correspondente.

- **Comando ‘as’:** Permite adicionar serviço

Para adicionar um novo serviço é executado o comando **‘as’** e de seguida devem ser preenchidos os seguintes campos:

[Connected to Event Service] Command (use 'h' for help): as

Enter location(s): miramar
Destination address(es): FF:FF:FF:FF:FF:FF
Date of execution(dd/mm/yy): 29/01/13
Time of execution(hh/mm): 15:00
Enter discovery duration: 60
Use textual content directly?(y/N): y
Enter textual content: Olá, isto é um teste.
Enter service name (Return for none): teste
Service addition issued for station(s) 'miramar'
Service with ID '1' addedd successfuly to station 'Miramar'

A fim de entender melhor o processo, segue-se a descrição em detalhe de todos os passos individuais.

- ***Location*** **(Localização)**: Nome da estação ou estações onde o administrador deseja adicionar o serviço. Os nomes das estações devem ser separados por espaço e as estações devem ser previamente registadas no sistema.

- ***Destination Addresses*** **(Endereços de Destino)**: Os endereços MAC das placas Bluetooth dos dispositivos dos utilizadores para os quais o serviço se destina. O uso do endereço especial 'FF: FF: FF: FF: FF: FF' é usado para *broadcast* (ou seja, envia o serviço para todos os dispositivos que estão na vizinhança da BlueStation). Para enviar o serviço a vários endereços MAC, estes têm de ser são separados por espaço.

- ***Date of Execution*** **(Data para iniciar a execução):** Define a data em que o serviço deve ser executado. Ao deixar este campo em branco o serviço será registrado para ser executado no presente dia. O administrador pode fazer uso de caracteres especiais, a fim de especificar datas concretas. Por exemplo, se o administrador desejar que um serviço se execute a partir de agora até 31 de Dezembro, terá de introduzir o seguinte comando, usando um caracter especial: *; 31/12/2012.

- ***Time of Execution*** **(Hora para iniciar a execução):** Semelhante à data, mas agora em relação à hora. Ao deixar o campo em branco o serviço vai ser executado até à meia-noite do presente dia. Se o administrador desejar que o serviço seja executado a partir de agora até às 22:00h, terá de introduzir o seguinte comando, usando um caracter especial: *; 22:00.

- ***Enter discovery duration*** **(Duração da Procura):** Define o tempo em segundos pelo qual os dispositivos têm de permanecer na vizinhança da BlueStation, a fim de serem selecionados para receberem o serviço.

- ***Content* (Conteúdo):** Se o administrador enviar conteúdo textual, só terá de premir **'y'** e introduzir a mensagem que deseja enviar aos utilizadores. Se o conteúdo não for texto e sim uma imagem, o administrador deve apresentar um URI ou então um URL. A URI pode ser um ficheiro local e a URL (Uniform Resource Locator) pode ser um ficheiro *online*. No caso em que o administrador utilize um URI, deve usar o comando: 'file:///caminho_do_ficheiro'. A URI deve ser acessível pela estação em si e não pela consola da administração. Por outras palavras, o serviço é executado nas estações e esse arquivo deve estar presente na própria estação. Se não, a execução do serviço falhará. Para a utilização de um URL é muito simples, basta o administrador introduzir o endereço completo que dá acesso ao ficheiro *online.* Por exemplo: *"*http://www.exemplo.com/teste.jpg"

- ***Service Name* (Nome do Serviço):** Nome que o administrador quer dar ao serviço e pelo qual o serviço é identificado nas BlueStations.

Após todos os campos terem sido preenchidos a aplicação tenta o envio do serviço e depois é mostrada a informação de sucesso ou de falha.

- **Comando 'ds':** Devolve os detalhes de um serviço

Este comando devolve os detalhes de um serviço, sendo para isso necessário que o administrador introduza a estação e o ID do serviço sobre o qual deseja obter mais informações.

[Connected to Event Service] Command (use 'h' for help): ds

Enter location(s): miramar
Introduce service ID: 1

Details for service '1' at 'miramar' issued
Details for service with ID '1'
Service fingerprint: d188cb842913d68ef32bb6e5d47d651e
Addition date: 29/01/13 14:58
Active: Yes
Target: Not set
Static Content:Yes
Statefull: Yes
Served Devices: None
Last Execution: None
Destination: FF:FF:FF:FF:FF:FF
Execution Date: 29/01/13
Execution Time: 15:00
Discovery Duration: 60
Locations: Miramar

- **Comando ‘rs’:** Permite remover serviços

Para remover um serviço que se encontra na estação é necessário introduzir o nome da estação e o ID do serviço. Para saber o ID do serviço é necessário executar o comando **‘ls’** a fim de listar todos os serviços que se encontram na estação e obter os IDs dos mesmos.

[Connected to Event Service] Command (use 'h' for help): rs

Enter location(s): miramar
Introduce service ID: 1

Service removal issued for service with ID '1' on station 'miramar'
Service with ID '1' successfuly removed from station 'miramar'

- **Comando ‘ab’:** Permite adicionar um endereço MAC à *Blacklist*

Quando é necessário ou desejável incluir um dispositivo para a lista negra, o administrador dispõe do comando **'ab'**. Por exemplo, para adicionar o dispositivo com o endereço AA:BB:CC:DD:EE:FF na estação miramar:

[Connected to Event Service] Command (use 'h' for help): ab

Enter location(s): miramar
Enter black list address: AA:BB:CC:DD:EE:FF

Addition of black list address 'AA:BB:CC:DD:EE:FF' for station 'miramar' issued
Successfuly added address 'AA:BB:CC:DD:EE:FF' to station 'Miramar' black list

Quando um endereço é adicionado à *BlackList*, ele não será contabilizado durante a execução de um serviço, ou seja, não receberá qualquer serviço das estações que tenham o seu endereço MAC nas suas *Blacklist*.

- **Comando ‘lb’:** Permite listar por estação os endereços MAC que se encontram na *Blacklist*

[Connected to Event Service] Command (use 'h' for help): lb

Enter location(s): Miramar

Black list issued for station(s) 'miramar'
1: AA:BB:CC:DD:EE:FF

Para o administrador consultar a *Blacklist* de uma estação, introduz a estação desejada e é devolvida a lista com os endereços MAC.

- **Comando 'rb':** Permite remover um endereço MAC que se encontra na *Blacklist* de uma estação

[Connected to Event Service] Command (use 'h' for help): rb

Enter location(s): miramar
Enter black list address: AA:BB:CC:DD:EE:FF

Removal for black list address 'AA:BB:CC:DD:EE:FF' on station(s) 'miramar' issued
Successfuly removed address 'AA:BB:CC:DD:EE:FF' from station 'Miramar' black list

Para o administrador remover um endereço MAC da *Blacklist* necessita de escolher a estação e depois introduzir o endereço MAC que deseja remover. Se o administrador quiser limpar todos os endereços na lista negra a partir de uma determinada estação ou estações deve utilizar o comando '**cb**'. Isso é muito útil quando existe um grande número de endereços na lista negra [15].

5. ANÁLISE DO PROBLEMA

Um dos primeiros passos realizados foi modelar o problema para evitar interpretações subjetivas e entender quais as atividades que o sistema suporta e os componentes envolvidos nessas atividades. A modelação ajuda a visualizar a aplicação em termos de estrutura, comportamentos e ajuda a ter um guia para a construção da nova aplicação.

5.1. PARTICIPATION MAP

Uma das formas mais simples de modelar uma atividade é através de um mapa simples do cenário onde a atividade é desempenhada. Criar uma representação dos participantes e as relações entre eles e com os vários artefactos envolvidos na atividade é fundamental para se ter uma visão geral do contexto e em que momento determinada atividade ocorre no sistema [16].

Para este sistema existem 2 cenários de atividades. O primeiro cenário (ver Figura 26) descreve como o administrador interage com o sistema.

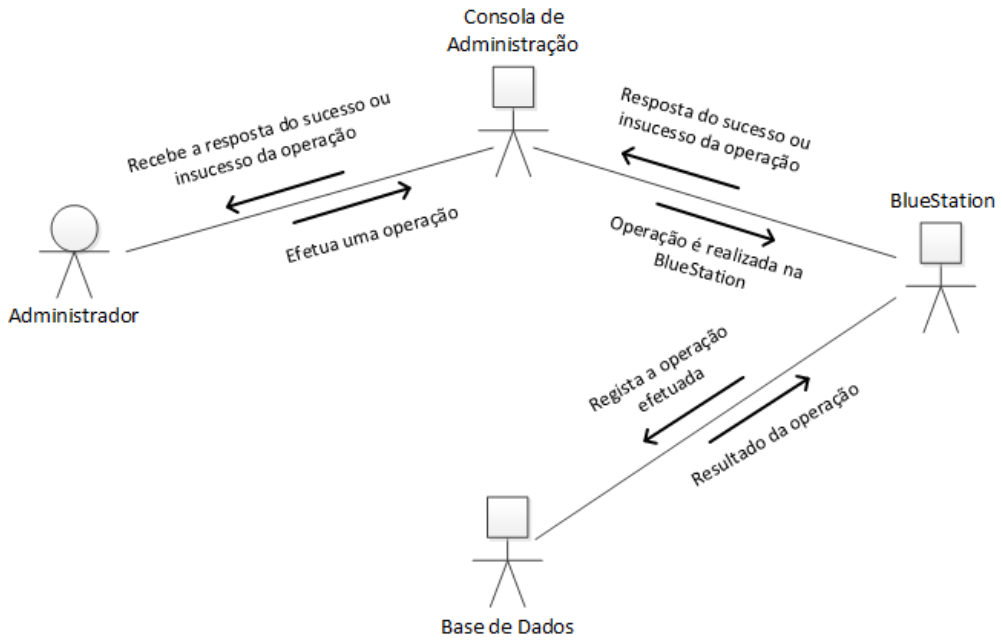

Figura 26 – Participation Map: Cenário para o Administrador

O segundo cenário (ver Figura 27) descreve as situações que podem ocorrer quando os utilizadores (ou seja, quem recebe os conteúdos enviados pelas estações) interagem ou não com o sistema.

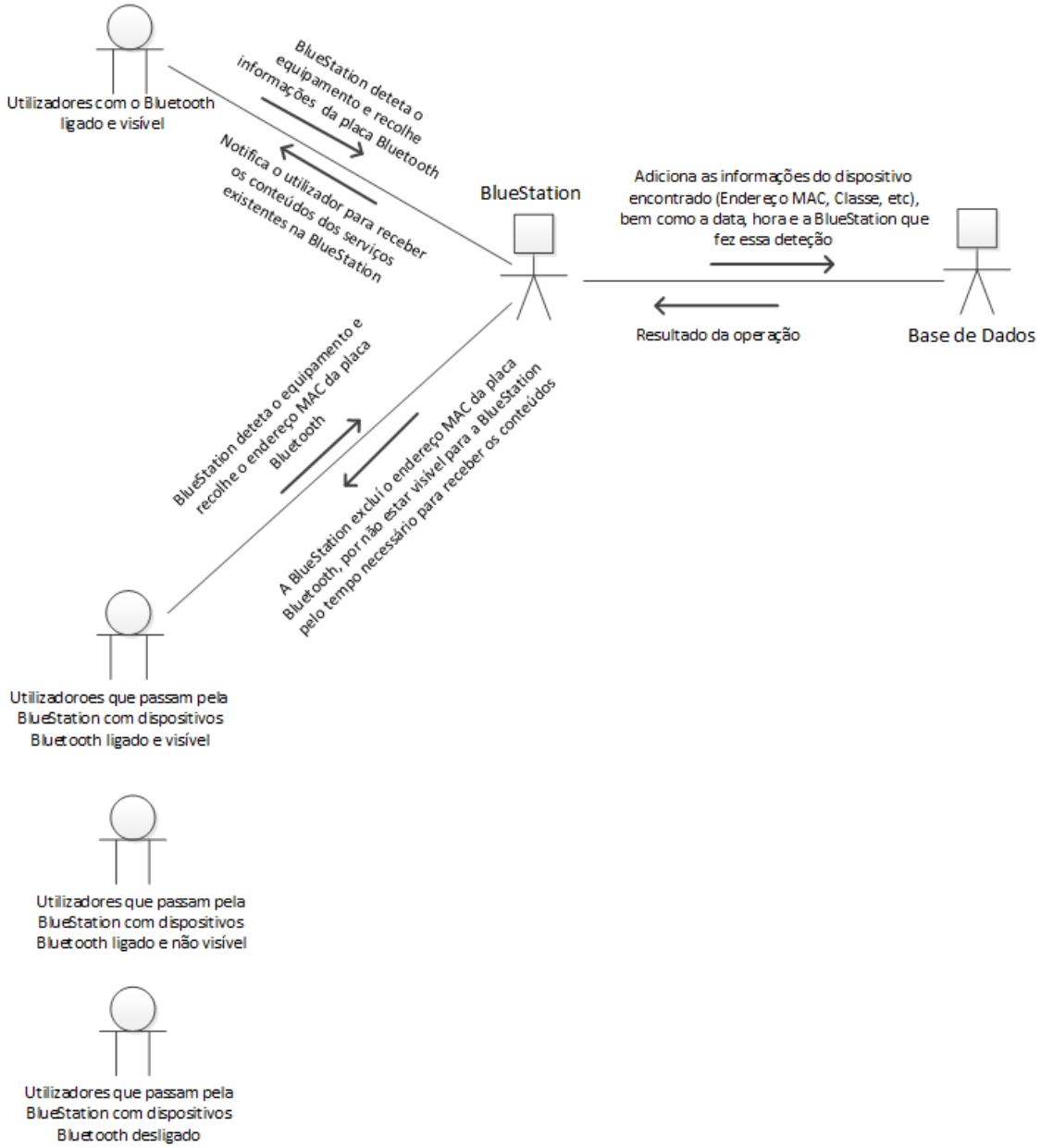

Figura 27 - Participation Map: Cenário para o Utilizador

5.2. DIAGRAMA DE INTERAÇÃO

Nos diagramas de interação que se seguem, o primeiro diagrama (ver Figura 28) representa os passos e os comandos possíveis que o administrador tem que introduzir para executar na consola de administração a operação desejada sobre as BlueStations. O comando **'as'** que permite a criação do serviço é detalhada na Figura 29.

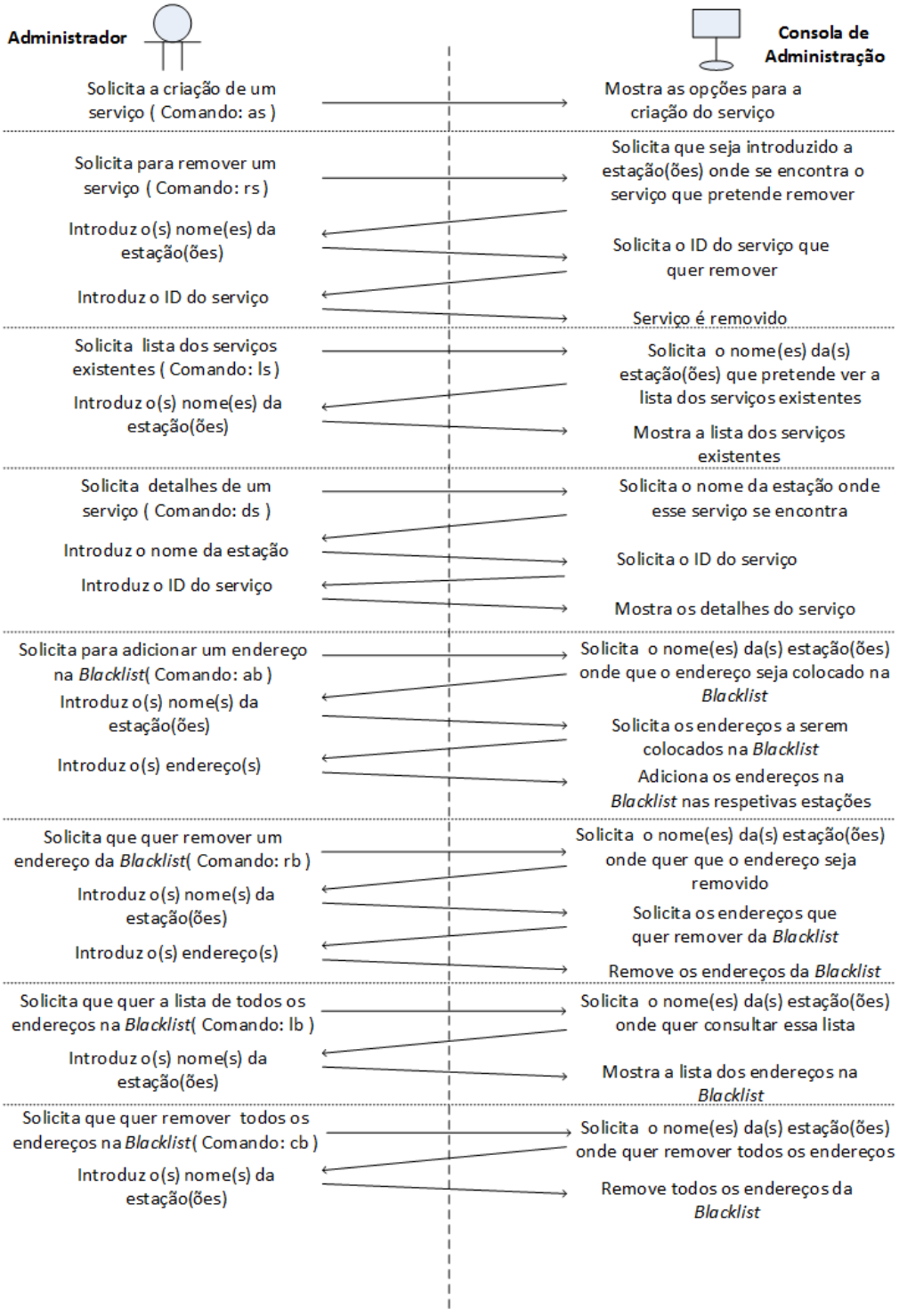

Figura 28 - Diagrama de interação: Operações possíveis da consola de administração

No segundo diagrama de interação (ver Figura 29) são detalhado os passos que o administrador tem que introduzir para criar um serviço e associá-lo a uma BlueStation, utilizando a consola de administração.

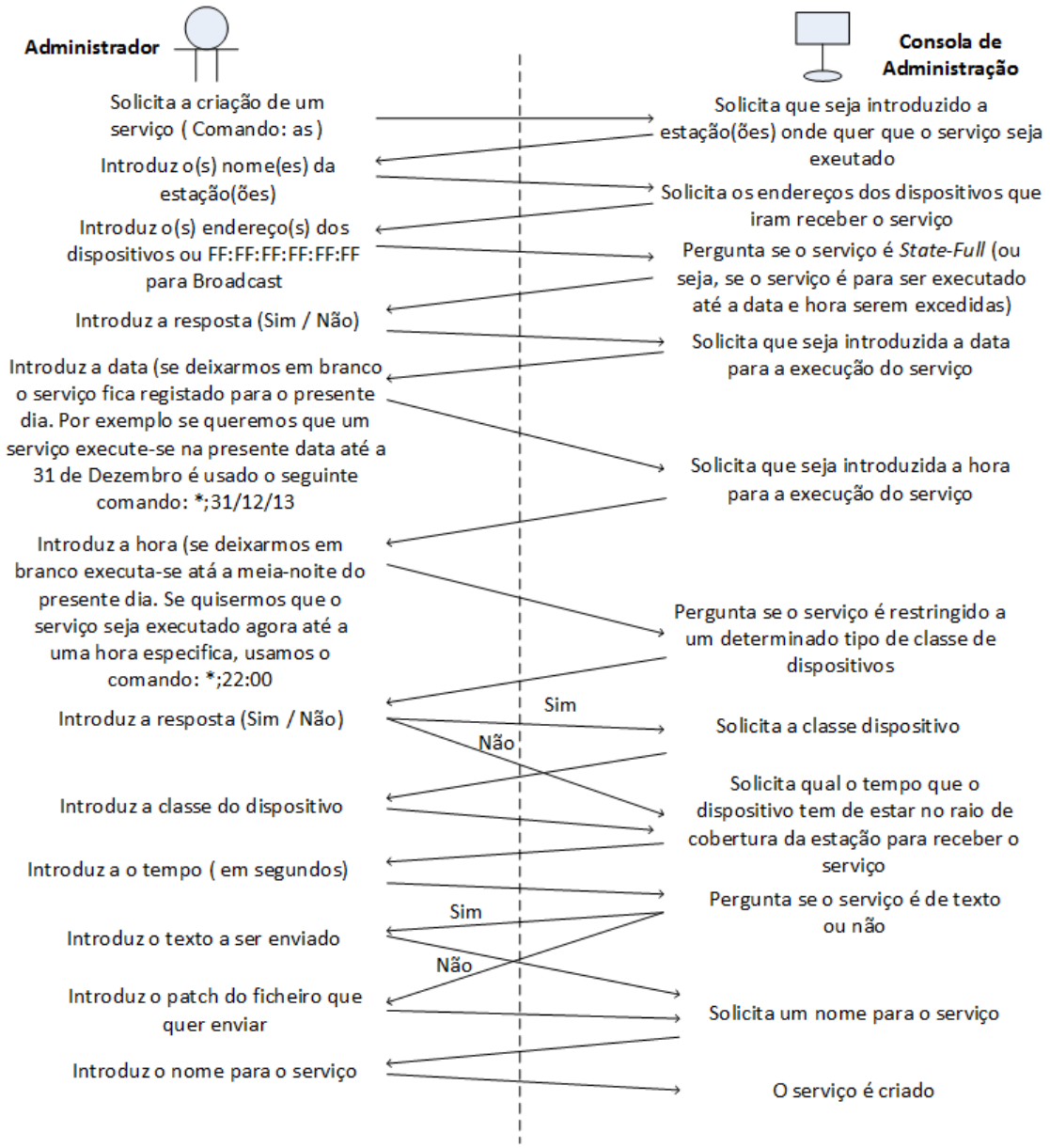

Figura 29 - Diagrama de interação: Passos para criar serviços na consola de administração

5.3. Fase de Análise e de *Debug* da Consola de Administração (aplicação existente)

Nesta secção são detalhados os passos fundamentais resultantes do processo de *debug* e de análise de todo o sistema e da aplicação existente para gestão e criação de serviços. A fase de análise e de *debug* tornaram possível conhecer toda a arquitetura do sistema, todo o seu funcionamento para assim poder tomar decisões acertadas. Resultante de todo esse processo foi possível implementar a aplicação Web para substituir a consola de administração e resolver os problemas já mencionados nos capítulos anteriores.

A consola de administração executa-se sobre a linha de comandos. Fazer o *debug* sobre este ambiente é uma tarefa árdua e pouco produtiva. Portanto, foram copiados todos os ficheiros da consola de administração e utilizado o IDE Komodo, que dá suporte à linguagem Perl, permitindo assim instalar as bibliotecas necessárias para correr a aplicação no IDE, e proceder à tarefa de *debug*. Ao longo do processo de *debug*, foi verificado que a consola de administração em todas as suas comunicações utiliza objetos ORB (*Object Request Broker*), ou seja, objetos de pedido para o *Broker* (ver secção 3.1.2). Esses objetos são definidos pela arquitetura CORBA. A arquitetura CORBA tornou possível a criação da aplicação Web sem necessidade de alterar qualquer componente base do sistema. Os objetos ORB utilizam o protocolo IIOP, que neste sistema está a utilizar a porta 10010. Estes objetos recebem o pedido do utilizador e enviam o pedido para o componente adequado, que depois devolve ao utilizador a resposta ao pedido.

Durante o processo de *debug* foi percetível que a consola de administração utiliza dois componentes essenciais, um para o envio dos pedidos e outro para o envio das respostas aos pedidos. Estes componentes estão implementados nas classes: CORBA_EventService.pm e CORBA_LocalStation.pm. Nestas classes encontram-se todos os métodos que são chamados para executar determinada operação nas estações. Estes métodos são acedidos pela aplicação através de mensagens que são enviadas da aplicação para os componentes CORBA que contêm um campo com o nome do método que querem aceder.

No processo de *debug* foram efetuadas todas as operações que o sistema permite fazer sobre as estações, para determinar e entender quais e como os objetos devem ser criados e associados às mensagens, a fim de realizar a operação desejada utilizando a arquitetura CORBA.

Os pontos focados anteriormente são os aspetos essenciais que o sistema existente utiliza e que era necessário implementar na nova aplicação de criação e gestão de serviços utilizando o ambiente Web.

Na secção 6 são explicados todos os aspetos de implementação, tanto da arquitetura CORBA, bem como de toda a aplicação Web.

5.4. REQUISITOS

Ao longo de várias reuniões com os responsáveis pelo sistema Bluetooth e membros dos Horários do Funchal, foram definidos um conjunto de requisitos funcionais e não-funcionais que pretendiam ver cumpridos para a nova aplicação de gestão e criação de serviços para as BlueStations. No desenvolvimento da aplicação Web foram tidos em consideração os requisitos que se seguem.

5.4.1. Requisitos Funcionais

Na conceção deste sistema, foram obtidos os seguintes requisitos funcionais:

- O sistema deve informar o administrador do estado de funcionamento das estações.
- O sistema deve permitir ao administrador criar, editar e eliminar serviços de texto e de imagem.
- O sistema deve mostrar todos os serviços ativos e os que já terminaram a sua execução nas estações.
- O sistema deve mostrar ao administrador todas as informações relacionadas com os serviços.
- O sistema deve suportar o carregamento de imagens para serem adicionadas aos serviços.
- O sistema deve permitir inserir destinatários de forma automática na criação dos serviços.
- O sistema deve permitir excluir, por estação, determinados dispositivos de receberem serviços.
- O sistema deve permitir listar e retirar determinados dispositivos que estão excluídos de receberem serviços.

5.4.2. Requisitos Não-Funcionais

Na conceção deste sistema, foram obtidos os seguintes requisitos não-funcionais:

- O sistema deve ter uma disponibilidade de 99%.
- O sistema deve assegurar a proteção dos dados de acessos não autorizados.
- O sistema deve assegurar que não há recolha de nenhum dado que viole a privacidade dos utilizadores.
- O sistema deve apresentar uma *interface* intuitiva e de fácil utilização, garantindo uma boa interação entre o administrador e o sistema.
- Garantir que o sistema permita uma fácil alteração das suas funcionalidades.
- O sistema deve permitir uma fácil integração de novas funcionalidades.
- Uma vez que as BlueStations estão sempre a enviar informações para o servidor, o tempo de execução das operações utilizando o sistema deve ser adequado, de modo a obter-se uma eficiência aceitável.
- O sistema deve criar um *backup* de todos os serviços que são criados pelo administrador.
- O sistema deve ser otimizado para ser usado no navegador Google Chrome.

5.5. PROTÓTIPOS DE MÉDIA FIDELIDADE

A prototipagem é uma tarefa fundamental na modelação do problema. Permite ter uma visão geral da *interface* do sistema, como esta pode suportar as funcionalidades requisitadas e como é feita a interação com a aplicação.

Foram efetuados protótipos de média fidelidade para que a visão da aplicação Web fosse mais próxima da versão final e fosse possível efetuar testes de usabilidade de uma forma rápida, pouco dispendiosa e de fácil adaptação aos problemas que os testes fossem apresentando. De notar que os utilizadores estavam habituados a uma aplicação que se executava sobre a consola e não em ambiente gráfico.

A prototipagem foi realizada em duas fases. A primeira fase, como é habitual acontecer, é muito propícia a que os problemas de usabilidade sejam numerosos. Devido a isso foi decidido realizar uma segunda fase. Na segunda fase os protótipos criados já têm as correções aos problemas detetados na primeira fase e assim é passada à fase de desenvolvimento da aplicação, que se aproxima de uma versão quase final com o mínimo de erros de usabilidade possíveis e com as funcionalidades exigidas.

5.5.1. Primeira Fase de Prototipagem

Na primeira fase de prototipagem os protótipos de média fidelidade criados foram a *interface home* (ver Figura 30) e a *interface* para criar um serviço (ver Figura 32). Só foram criados dois protótipos, porque as restantes *interfaces* seguem a mesma ideia de *design*.

- **Problema:** A aplicação consola de administração (ver Figura 25) foi desenvolvida pensando mais no sistema do que no utilizador, tornando o acesso às informações relevantes para quem administra o sistema, uma tarefa árdua e propicia a erros. Na projeção das soluções de desenho para a nova aplicação de gestão e criação de serviços, a importância foi dada ao administrador e não ao sistema. Assim sendo, o *design* das *interfaces* tinha de ser organizado colocando as informações relevantes (informações sobre as estações e serviços) sempre visíveis para o administrador, e organizar a *interface* de acordo com as funcionalidades que o sistema suporta.

- **Solução:** Com base nos problemas descritos anteriormente, este primeiro protótipo (ver Figura 30) apresenta no centro a listagem de todos os conteúdos que se encontram nas BlueStations, ou seja, as informações dos serviços, os endereços MAC excluídos que se encontram nas *Blacklist* e outras informações que pudessem ser consideradas relevantes ao administrador. No lado esquerdo da Figura 30 é apresentado o menu expansível que permite navegar pelas várias *interfaces* e executar diferentes operações. No topo da página são apresentadas as estações registadas no sistema, seguidas de um ícone que varia de cor consoante o estado de funcionamento da estação correspondente.

Na *interface home* (ver Figura 30) o objetivo é colocar à disposição do administrador um resumo global de todo o sistema, ou seja, a informação das tarefas que as BlueStations já realizaram ou estão a realizar, bem como o estado de funcionamento de todas as estações e o menu de navegação que permite efetuar as operações permitidas pelo sistema, sobre os serviços e *blacklist*.

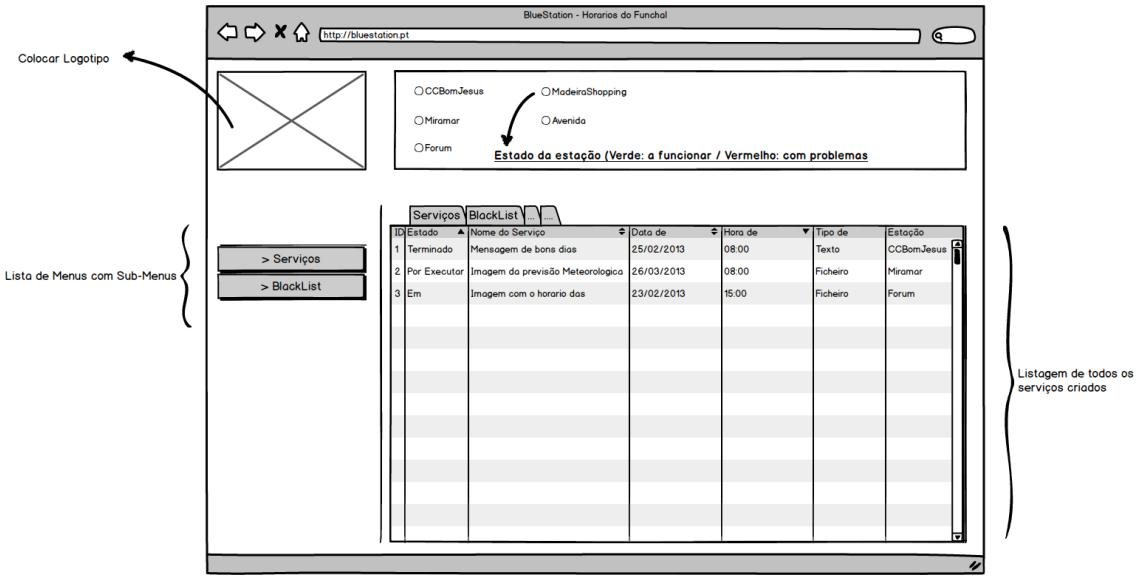

Figura 30 - Protótipo de Média Fidelidade: Interface Home

- **Problema:** Criar um novo serviço na consola de administração (ver Figura 31) era uma tarefa complexa e pouco acessível para um administrador, que podia ter que criar vários serviços por dia. Algumas informações pedidas pela aplicação não são triviais e consequentemente resultavam em erros, que impediam a execução do serviço ou numa execução que não a desejada pelo administrador. A consola de administração não fornece informação prévia de quais as estações que estão em funcionamento, permitindo que o administrador introduzisse em vão todas as informações e quando o serviço estava a ser enviado, recebesse a informação que a estação não estava operacional.

```
[Connected to Event Service] Command (use 'h' for help): as
Enter location(s): ccbomjesus
Destination address(es): FF:FF:FF:FF:FF:FF
Execution method(default is OBEXPUSH):
Set service as statefull?(y/N): n
Set service as frequent?(y/N): n
Date of execution(dd/mm/yy): 05/05/13
Time of execution(hh/mm): 12:00
Enter discovery duration: 60
Set class restrictions?(y/N): n
Use textual content directly?(y/N): y
Enter textual content: Isto e um teste
Enter service name (Return for none): teste
Service addition issued for station(s) 'ccbomjesus'
```

Figura 31 - Consola de Administração: Interface para a criação de um serviço

- **Solução:** Para resolver os problemas mencionados, o pretendido para a aplicação Web é minimizar o número de passos que o administrador tem de realizar para criar um serviço, através da predefinição de algumas opções que são mais comuns, da adição de *widgets* para o agendamento, da seleção automática das estações que estão operacionais e ocultando as informações (sendo elas definidas automaticamente pela aplicação Web e ocultas ao administrador) que dizem respeito à tecnologia Bluetooth e que não são importantes para a criação do serviço, do ponto de vista do administrador.

O segundo protótipo (ver Figura 32) apresenta a *interface* para a criação de serviços e as decisões de desenho tomadas.

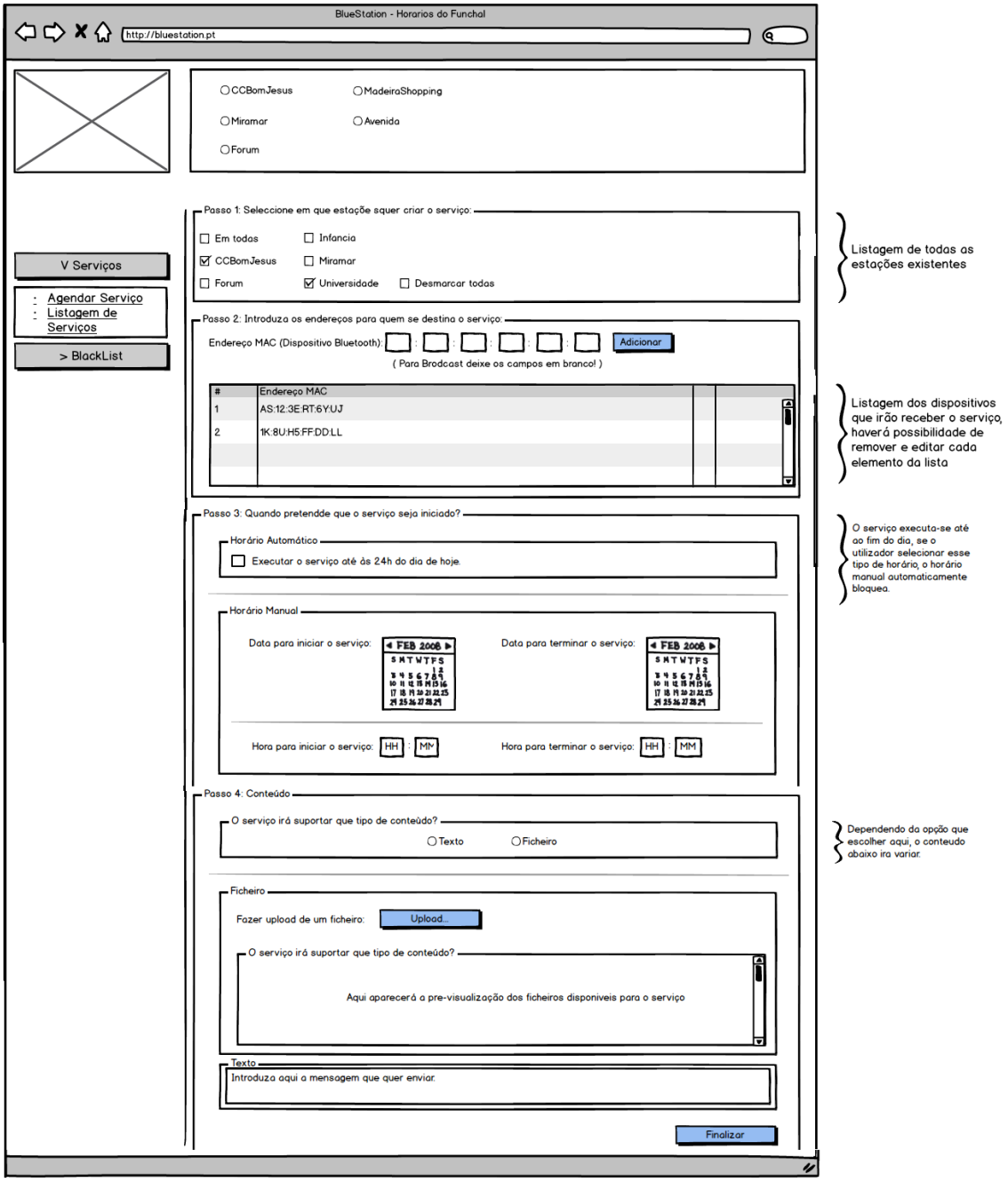

Figura 32 – Protótipo de Média Fidelidade: Interface Criação de um Serviço

Como é mostrado na Figura 32, no primeiro passo são listadas todas as estações registadas no sistema, para que o administrador selecione quais receberão o serviço; No passo dois o administrador escolhe a quem se destina o serviço, adicionando os endereços MAC dos dispositivos Bluetooth nas caixas de texto, que estão agrupadas seguindo a estrutura de um endereço MAC; No passo 3 o administrador escolhe o tipo de agendamento para o serviço; No quarto e último passo o administrador escolhe uma de duas opções, se o serviço é de texto ou de ficheiro. Dependendo da opção escolhida alguns componentes da *interface* ficaram bloqueados.

Para a primeira parte de prototipagem os testes de usabilidade aos protótipos foram realizados por quatro pessoas (nível universitário e com bons conhecimentos informáticos). A tarefa essencial para esta aplicação e com mais operações para o administrador era a de criar serviços para serem enviados para as estações. Portanto os dois protótipos anteriores das duas *interfaces* eram suficientes para os primeiros testes de usabilidade, e úteis para o *design* das restantes *interfaces*.

A cada pessoa foi atribuída uma tarefa diferente e a cada uma delas foi feita uma breve descrição acerca do propósito e objetivo da aplicação. Ao realizar estas tarefas, os participantes foram incentivados a partilharem eventuais dificuldades e aspetos que lhes fizessem confusão. Durante a realização das tarefas foram retiradas anotações dos problemas reportados pelos participantes e erros cometidos.

5.5.2. Segunda Fase de Prototipagem

Com base nas anotações retiradas dos testes de usabilidade efetuados aos primeiros protótipos, foi separada a criação de serviços em duas *interfaces.* Uma *interface* para o serviço de texto (ver Figura 33) e outra *interface* para a criação do serviço de ficheiro (ver Figura 34).

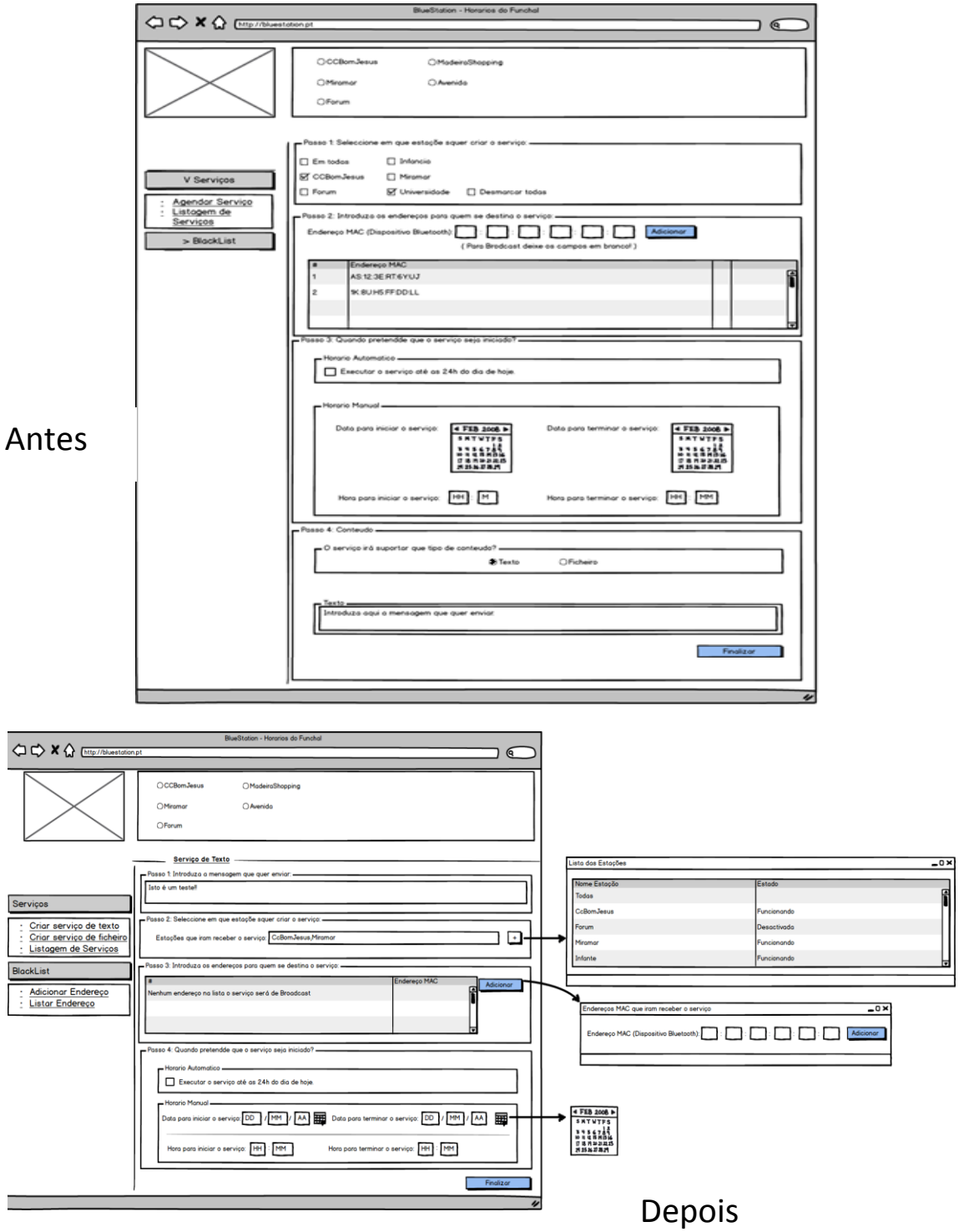

Figura 33 - Protótipo de Média Fidelidade: Antes e Depois - Interface Criação de um Serviço de Texto

Do protótipo (ver Figura 32) realizado na primeira fase de prototipagem, que juntava numa só *interface* a criação do serviço de texto e de ficheiro, nesta segunda foram criados dois protótipos de média fidelidade para cada um dos serviços. Esta decisão foi tomada devido aos participantes revelarem que a antiga *interface* continha muita informação, o que a tornava confusa. Em relação ao menu de navegação, este foi alterado para tornar todas as opções visíveis e não ocultas. A ordem de realização dos passos para criar o serviço foi alterada com base nas sugestões dadas pelos participantes, e de relevância para a criação do serviço. Com o objetivo de mostrar só a informação relevante ao administrador, a lista das estações passou a estar disponível através de um *pop-up* onde o administrador seleciona as estações que deseja enviar o serviço e só as selecionadas aparecem na *interface* principal. A decisão tomada para as estações foi aplicada também à inserção do endereço MAC. Em relação ao agendamento do serviço, foi adicionada a possibilidade de inserção da data de forma manual e assim disponibilizar duas formas de inserção, através do *pop-up* ou digitando manualmente (ver Figura 34).

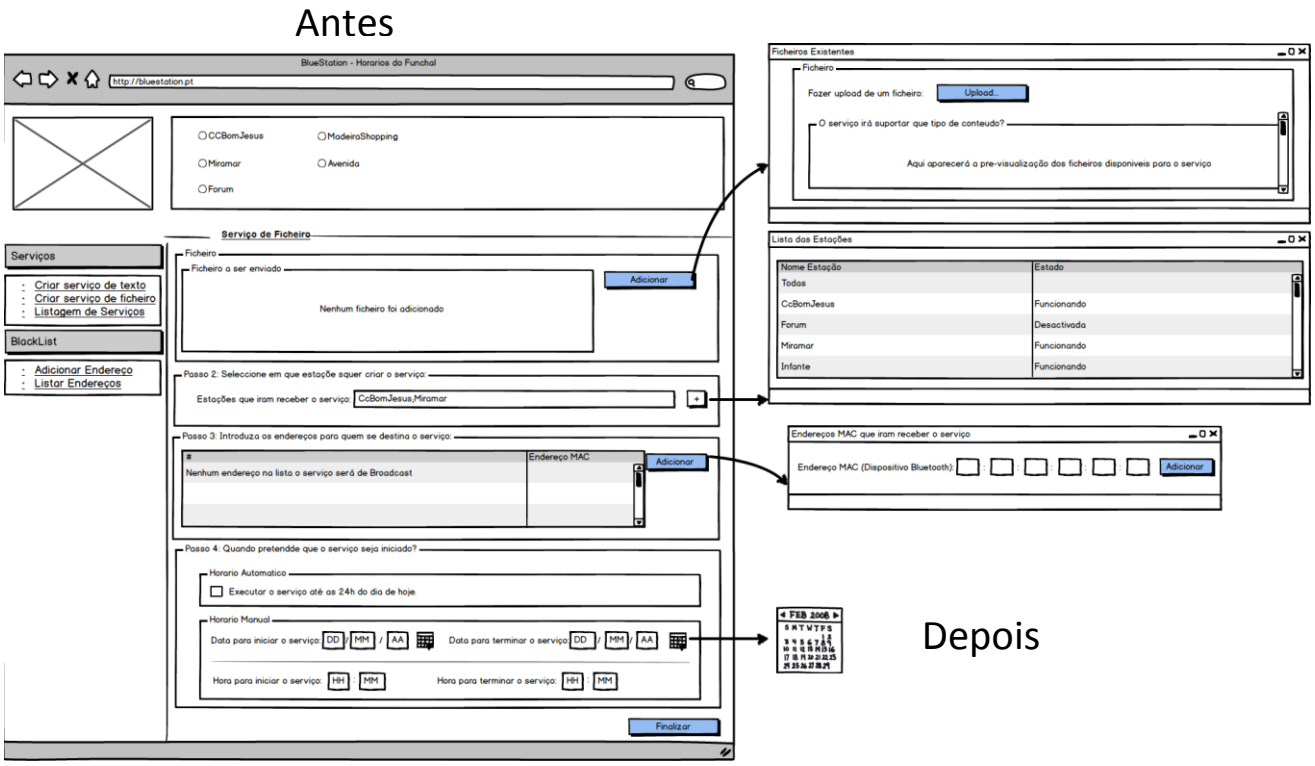

Figura 34 - Protótipo de Média Fidelidade: Depois - Interface Criação de um Serviço de Ficheiro

Depois de terminada a segunda fase de prototipagem e com base nos protótipos realizados e das conclusões retiradas dos testes de usabilidade, foi dado início à criação do *design* da aplicação Web, bem como à implementação de todas as funcionalidades.

6. BLUESTATION - APLICAÇÃO WEB

Neste capítulo será feita uma descrição de todo o desenvolvimento da nova aplicação para a criação e gestão de serviços.

6.1. ASPETOS DE IMPLEMENTAÇÃO

As decisões tomadas sobre os aspetos de implementação foram baseadas na análise teórica e técnica, pensando na melhor arquitetura que a nova aplicação teria de apresentar para que todos os objetivos fossem cumpridos.

Com base nos requisitos funcionais e não-funcionais, na arquitetura que o sistema apresenta, na tecnologia utilizada e no servidor que dá suporte a todo o sistema, foram tomadas as seguintes decisões de implementação:

1) Aspetos Técnicos

A consola de administração, que se executa sobre a linha de comandos, está instalada num servidor com IP público, para que a aplicação seja acedida de qualquer lugar e para que a comunicação entre o servidor e as estações seja possível.

O servidor HTTP instalado não possui todos os requisitos necessários para suportar a nova aplicação baseada na Web. Devido a esta limitação, a nova aplicação foi construída com recurso a um servidor HTTP *Tomcat*. Este servidor suporta *servelets*, JSP (*JavaServer Page*), possibilita uma melhor gestão de aplicações Web, permite ter um escalonamento da aplicação desenvolvida (nomeadamente a introdução de um sistema de *login* utilizando um *Active Directory*) e permite alterar configurações da aplicação Web sem ter que alterar o código-fonte. Assim o servidor *Tomcat* preenche todos os requisitos necessários para implementar todas as funcionalidades pretendidas para a aplicação Web.

2) Aspetos de Desenvolvimento

O facto de o sistema utilizar a arquitetura CORBA na sua implementação foi fundamental para que o desenvolvimento da *interface* Web fosse possível. Implementar o serviço CORBA recorrendo apenas à Web é uma tarefa difícil, devido às bibliotecas que têm de ser utilizadas para que o serviço funcione, e só foi possível realizar a sua implementação na aplicação Web recorrendo a um módulo intermediário utilizando a linguagem JAVA e utilizado o serviço REST para aceder a esses componentes (ver Figura 35). Os componentes da visualização da *interface* estão escritos em HTML (*HyperText Markup Language*) juntamente com a utilização de CSS (*Cascading Style Sheets*). Para conceber *scripts* foi utilizado o *Javascript* que vai fazer a ligação entre os formulários HTML com o serviço REST, e para o envio de ficheiros para o servidor foi utilizado JSP.

6.1.1. Arquitetura Geral da Aplicação Web

A arquitetura da aplicação Web (ver Figura 35) é construída por três entidades essenciais: as estações Bluetooth, as componentes que se encontram do lado do cliente e as do lado do servidor.

Do lado do cliente fazem parte os componentes responsáveis pela visualização e manipulação dos conteúdos que são apresentados pela aplicação Web. Com o objetivo de separar o código HTML da parte de formatação e aparência das páginas HTML é utilizada a linguagem de estilo CSS. É fundamental esta separação, permitindo uma maior eficiência no carregamento das páginas HTML por parte dos *browsers,* e permite que as alterações de estilo das páginas HTML possam ser feitas editando os ficheiros CSS.

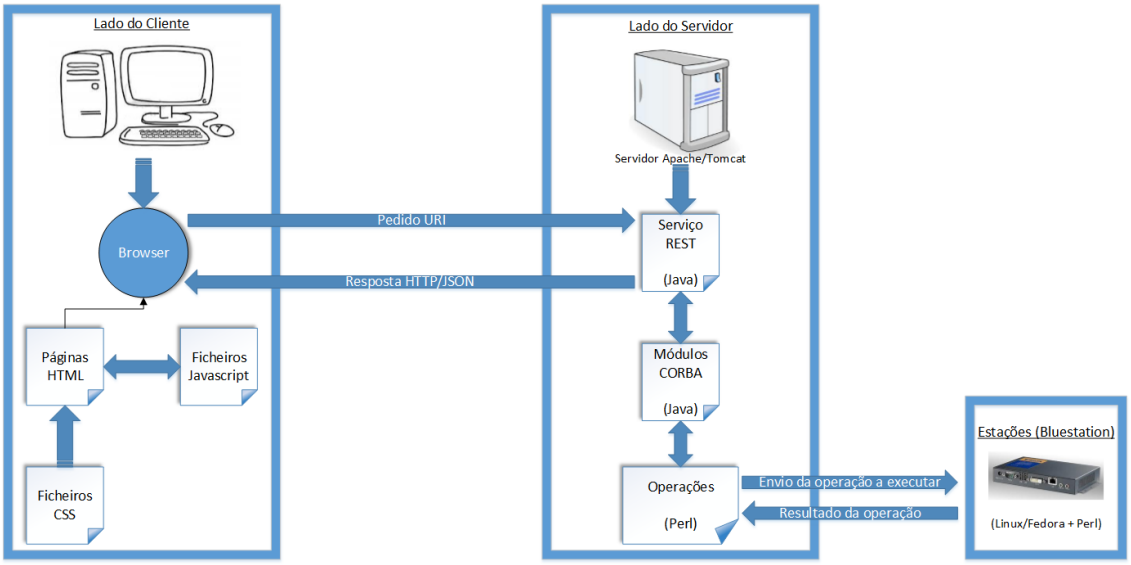

Figura 35 - Arquitetura Geral da Aplicação Web

Em relação aos ficheiros JS (*JavaScript*), estes fornecem um conjunto de funções que são executadas de acordo com as ações que o utilizador realize nas páginas HTML. Quando uma ação do utilizador necessita de um pedido ao servidor é executada uma função JS que vai utilizar a especificação CORS (*Cross-Origin Resource Sharing*). Esta especificação permite comunicar com o servidor através do *browser* e aceder aos seus recursos, que neste caso serão os módulos CORBA que permitirão comunicar com as estações. O acesso aos recursos é feito utilizando um identificador denominado de URI. Estes identificadores são interpretados pelo serviço REST que devolve a resposta para o lado do cliente no formato JSON ou em XML.

O lado do servidor é composto por três componentes: o serviço REST, os módulos CORBA e as operações que podem ser feitas sobre as BlueStations (ver Figura 35).

O serviço REST está implementado em Java e é o responsável por receber os pedidos URI provenientes da aplicação Web. O serviço REST para cada URI tem uma função dedicada na sua

implementação, que acede aos módulos CORBA para realizar as transformações necessárias para aceder ao serviço requisitado e assim dar resposta ao pedido do utilizador.

Nos módulos CORBA estão presentes os ficheiros essenciais que resultaram do processo de *debuging* descrito na secção 5.3. Estes módulos tornam possível a comunicação entre a aplicação Web e as estações. Todas as informações que são trocadas entre a aplicação Web e as BlueStations utilizam objetos ORB. No módulo CORBA existem duas classes essenciais, a EventPublisher e a EventSubscriber. A classe EventPublisher é responsável por enviar os objetos ORB para aceder as estações e a classe EventSubscriber é responsável por receber os objetos ORB provenientes das estações e extrair o objeto, de forma a ser possível aceder à sua informação e enviá-la para a aplicação Web.

Quando o objeto ORB é criado, nele vai associada a operação que o utilizador quer realizar (por exemplo, criar um novo serviço), esta operação é interpretada pelos módulos CORBA e o pedido é enviado para as estações que executarão a operação e devolverão o sucesso/insucesso da operação.

6.2. FUNDAMENTAÇÃO TECNOLÓGICA E FUNCIONALIDADES

A aplicação Web foi construída com base em cinco *interfaces* principais: TextService.html, FileService.html, ServiceList.html, AddAddressBL.html e ListAddressBL.html.

Nesta seção, serão detalhados todos os aspetos relacionados com estas *interfaces*, desde as funcionalidades suportadas por cada uma e os ficheiros que as constituem.

6.2.1. Elementos Comuns a Todas as *Interfaces*

Existem dois elementos que estão presentes em todas as *interfaces* que compõem a aplicação Web. São eles a barra, que mostra o estado de funcionamento das estações (ver Figura 36), e o menu de navegação, que permite navegar pelas diferentes *interfaces* HTML (ver Figura 37).

6.2.1.1. Estado de Funcionamento das Estações

No topo de todas as *interfaces* é apresentado o estado de funcionamento, quase em tempo real, das estações que estão registadas no sistema. Esta barra de estado das estações (ver Figura 36) está presente em todas as *interfaces* HTML que compõem a aplicação Web para que o administrador saiba quando alguma estação deixa de funcionar.

O estado *online* é representado na *interface* pelo ícone circular verde que se encontra antes do nome da estação. O estado *online* significa que a estação está a comunicar com o sistema e que está preparada para responder às operações que o administrador quiser executar.

Figura 36 - Print-screen: Barra de estado que informa se a estação está online ou offline

O estado *offline* é representado na *interface* pelo ícone circular cinzento que é apresentado antes do nome da estação. O estado *offline* significa que a estação poderá estar ou não ligada, só que não tem qualquer ligação com o sistema e assim não pode dar resposta a nenhum pedido proveniente da aplicação Web.

Numa primeira fase, para implementar este tipo de funcionalidade, foi utilizada a técnica arquitetural de disponibilidade *heartbeat* (*Ping/Echo*). Esta técnica foi colocada a funcionar utilizando o sistema de agendamento de tarefas *Crontab* do Linux e uma função implementada em JAVA que se executava na aplicação Web para realizar os *pings* às estações. O *Crontab* foi utilizado do lado das estações, para que quando a estação comunicasse com o sistema enviasse para a aplicação Web um ficheiro com o seu IP. Este IP era posteriormente acedido e lido pela função programada em JAVA, que iria realizar os *pings* às BlueStations. Ao realizar os *pings* as estações que respondessem ao pedido eram registadas, como estando operacionais, ou seja, *online* e as que a aplicação Web não obtivesse qualquer resposta eram marcadas como estando *offline*.

Esta implementação, depois de vários testes veio a revelar-se problemática para este tipo de sistema. As estações utilizam para se ligar à internet um *modem* 3G onde as velocidades são um pouco limitadas e muitas vezes as estações não respondiam aos *pings* mesmo quando a estação tinha ligação à internet. Outro dos problemas reportados foi o bloqueio completo da aplicação Web quando existiam mais de duas estações a funcionar ao mesmo tempo, e era necessário fazer os *pings* a cada uma das estações. Para contornar esta limitação e ter um sistema de diagnóstico das estações a funcionar, a estratégia final implementada foi utilizar o *Crontab* para o envio de um ficheiro por parte da estação, sendo que o ficheiro tem o nome da estação que o envia. A aplicação Web o que faz é verificar para cada estação registada no sistema se existe um ficheiro com o respetivo nome enviado por parte da estação.

O método consiste em ter em cada estação uma tarefa agendada no *Crontab* que, a cada 5 minutos, envia o ficheiro para o servidor e outra tarefa no servidor, também no *Crontab* que, a cada 15 minutos, elimina todos os ficheiros enviados pelas estações. Como a cada 5 minutos as estações enviam um ficheiro com o nome da estação e a cada 15 minutos o servidor elimina-o, quando a estação deixar de enviar o ficheiro para o servidor, significa que não tem ligação à internet e assim é possível saber que a estação não está a comunicar, ou seja, que está *offline*.

6.2.1.2. Menu de Navegação

Na parte lateral esquerda de todas as *interfaces* da aplicação Web são apresentadas cinco opções de navegação (Criar Serviço de Texto, Criar Serviço de Ficheiro, Listar Serviços, Adicionar Endereços e Listar Endereços) separadas em duas categorias (Serviços e *Blacklist*) como pode ser visto na Figura 37. Este menu faz parte de todas as *interfaces* da aplicação e permite aceder a todas as funcionalidades que a aplicação Web dispõe. Ao longo deste capítulo são descritas todas as funcionalidades que compõem estas duas categorias.

Figura 37 - Print-screen: Menu de Navegação das Interfaces

6.2.2. TextService.html

Esta *interface* é a primeira página apresentada aos utilizadores da aplicação, quando acedem ao *website*: http://62.28.14.4:8080/BlueStation/TextService.html. Esta *interface* permite criar um serviço de texto, ou seja, uma mensagem de texto para ser associada às estações que o administrador desejar, desde que a estação esteja no estado *online*. Na criação deste serviço de texto é possível definir alguns parâmetros tais como: a quem se destina o serviço e em que momento o serviço começa e termina a sua execução.

Esta *interface,* através das validações do formulário Web, só permite que um serviço de texto seja enviado para as estações selecionadas se todos os campos forem preenchidos corretamente. Esta *interface* está dividida em 4 passos (ver Figura 38).

No passo 1 o utilizador atribui um nome ao serviço e introduz a mensagem que será apresentada a quem aceitar o serviço no seu dispositivo Bluetooth.

No passo 2 são apresentadas todas as estações que estão registadas no sistema, com base no sistema de diagnóstico das estações (ver capítulo 6.2.1.1), as estações *online* cujas *checkbox* são automaticamente selecionadas (podendo o administrador desseleccionar) e as restantes cujas checkbox são bloqueadas, não permitindo ao administrador selecionar por engano uma estação que não esteja a funcionar e que nunca receberá o serviço.

No passo 3 é determinado a quem se destina o serviço. O serviço pode ser generalizado a todos os equipamentos Bluetooth que sejam detetados pelas estações ou então personalizado para

um conjunto de endereços MAC. Quando o utilizador escolhe enviar para um grupo restrito de endereços MAC novos campos são apresentados na *interface* (ver Figura 38). Este tipo de efeito na *interface* é utilizado para que só seja apresentada a informação que é relevante para o administrador. Quando o serviço se destina a um grupo personalizado, os endereços MAC podem ser introduzidos através da caixa de texto presente na *interface* ou através do carregamento de um ficheiro a partir do computador que já contenha a lista de endereços MAC.

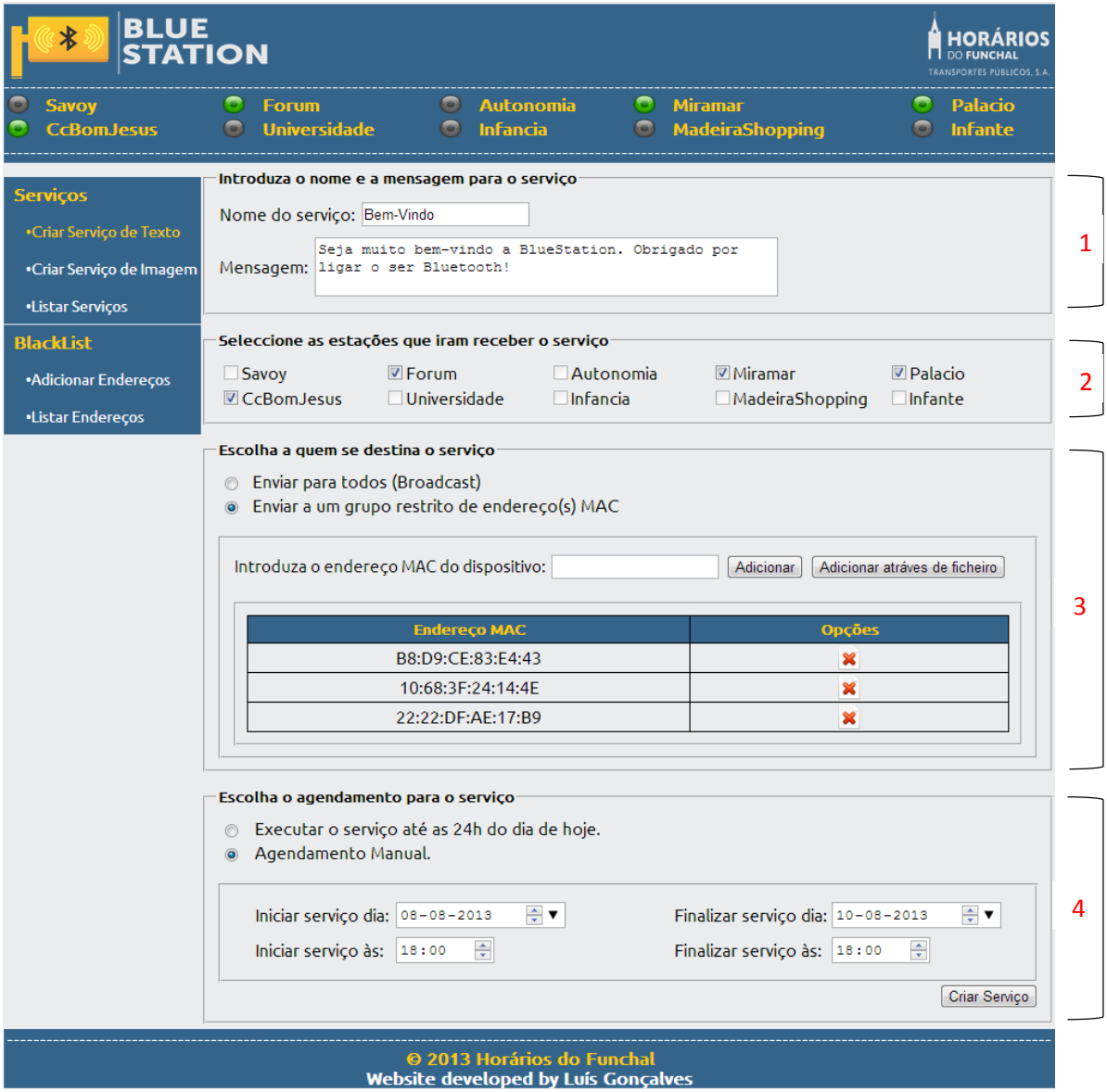

Figura 38 – Print-screen: Interface TextService.html

A possibilidade de ler os endereços através de um ficheiro .txt permite ter um armazenamento de endereços e utilizar esses endereços para futuros serviços sem ter que inserir um a um, de cada vez que o administrador os queira utilizar.

No quarto e último passo é escolhido o período em que o serviço se mantém em execução nas estações. Existem duas opções de agendamento, na primeira opção o serviço executa-se desde o momento em que é enviado para as estações até ao final do dia em que foi criado. A segunda opção o administrador define o dia e hora de início e fim para a execução do serviço (ver Figura 38).

Os ficheiros que fazem parte da *interface* TextService.html encontram-se ilustrados na Figura 39.

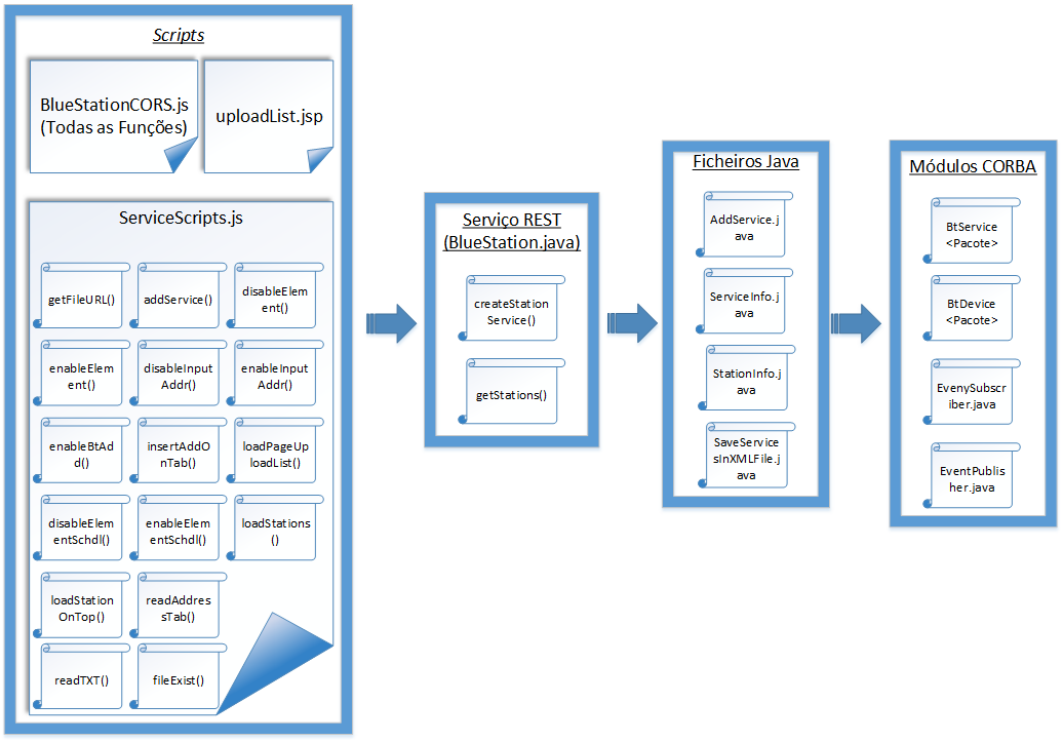

Figura 39 – Ficheiros que compõem a interface TextService.html e suas funcionalidades

6.2.3. FileService.html

Para além das estações suportarem o envio de serviços de texto também suportam o envio de ficheiros para os dispositivos dos utilizadores através do Bluetooth. Esta *interface* (ver Figura 40) tem tudo de semelhante com a *interface* TextService.html, com a exceção do passo número 1. Esta *interface* permite que o administrador da aplicação Web crie um serviço de ficheiro, associando um ficheiro ao serviço de duas formas: por URL (por exemplo, uma imagem que se encontra na internet) ou carregando um ficheiro do computador para o servidor.

Para que uma estação pudesse enviar um ficheiro como serviço, utilizando a anterior aplicação, que corre sobre a linha de comandos, a imagem tinha de ser enviada para a estação e o utilizador tinha de saber o *path* (caminho) completo da imagem e adicionar esse *path* no processo de criação do serviço. O utilizador tinha ainda de utilizar um *software* externo para enviar o ficheiro para a estação.

A estratégia utilizada para contornar esta limitação e tornar o processo muito mais simples, foi recorrer a uma pasta pública no servidor e utilizar o facto de o sistema suportar o envio de ficheiros utilizando uma URL. Quando o administrador carrega um ficheiro utilizando a aplicação Web, o ficheiro não necessita de ser reencaminhado para todas as estações a que se destina o serviço. O administrador apenas tem de fazer o *upload* do ficheiro para a pasta pública do

servidor. Quando o ficheiro é carregado para o servidor, o URL do ficheiro é gerado automaticamente e associado ao serviço que está a ser criado na aplicação Web.

A *interface* está desenhada para que quando a caixa de texto dedicada para o URL não esteja vazia o botão "Carregar Ficheiro" fique bloqueado (ver Figura 40), evitando assim que o administrador cometa erros e crie um serviço inválido. Os restantes passos na criação do serviço são os mesmos que os explicados na secção 6.2.2. A *interface* possui um conjunto de validações do lado do cliente que só permitem que o serviço seja enviado se todos os campos forem preenchidos corretamente.

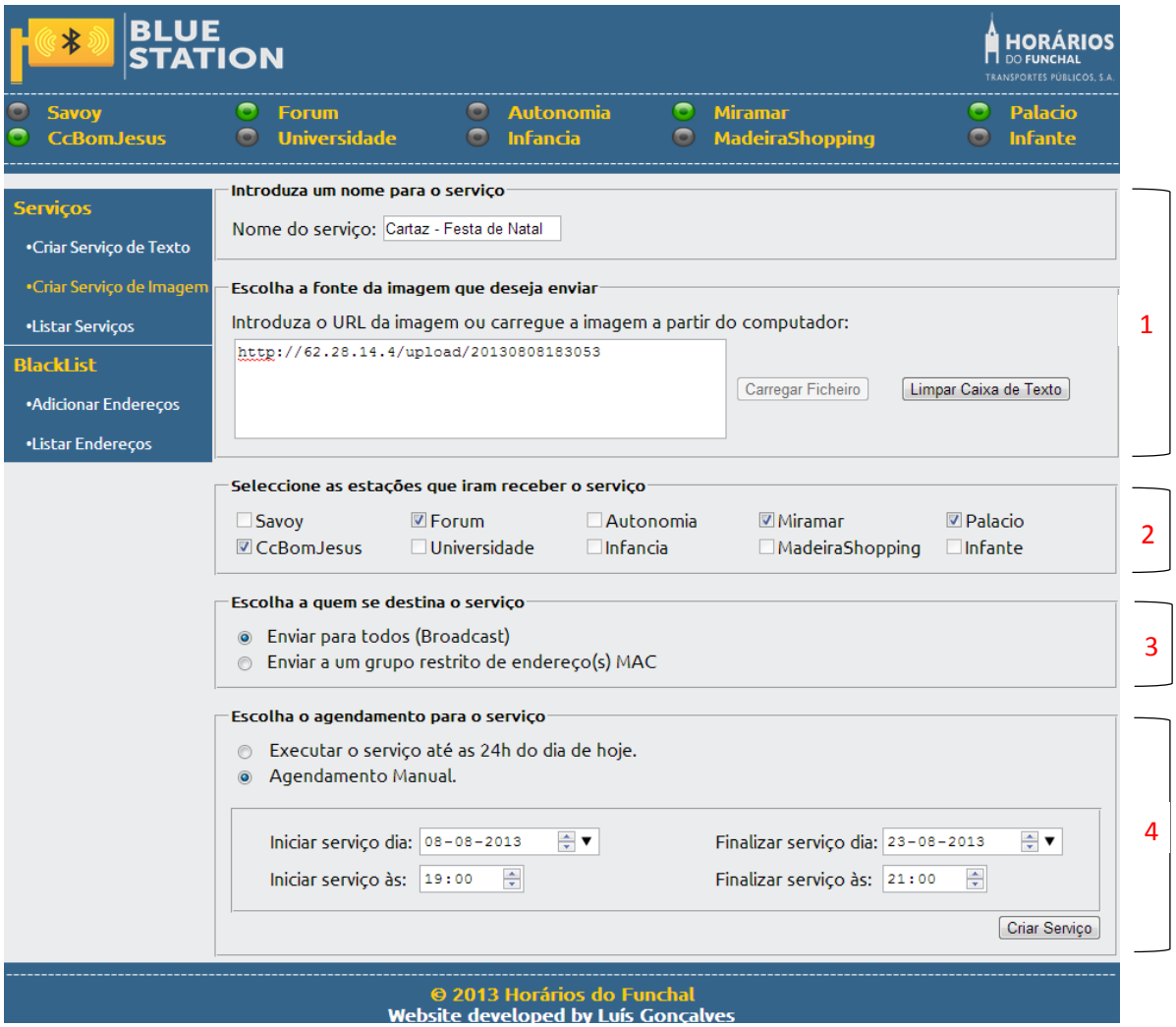

Figura 40 – Print-screen: Interface FileService.html

Os ficheiros que fazem parte da *interface* FileService.html encontram-se ilustrados na Figura 41.

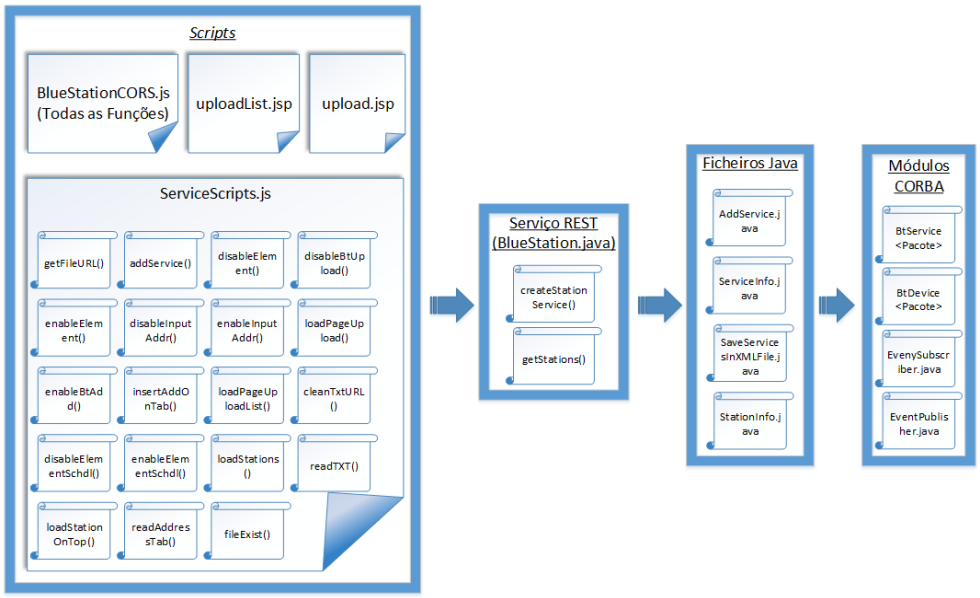

Figura 41 - Ficheiros que compõem a interface FileService.html e suas funcionalidades

6.2.4. ServiceList.html

A *interface* ServiceList.html (ver Figura 42) é a última funcionalidade da categoria dos serviços. Esta *interface* permite visualizar a listagem de todos os serviços que se encontram ativos e inativos numa estação, bem como consultar todos os detalhes dos serviços listados (detalhes como o utilizador que criou o serviço, o nome do serviço, a data e hora de execução, os destinatários, etc.). É nesta *interface* que um serviço pode ser removido da estação bem como é possível obter informações mais detalhadas de um serviço específico, seja ele de texto ou ficheiro.

Figura 42 - Print-screen: Interface ServiceList.html

Os ficheiros que fazem parte da *interface* ServiceList.html encontram-se ilustrados na Figura 43.

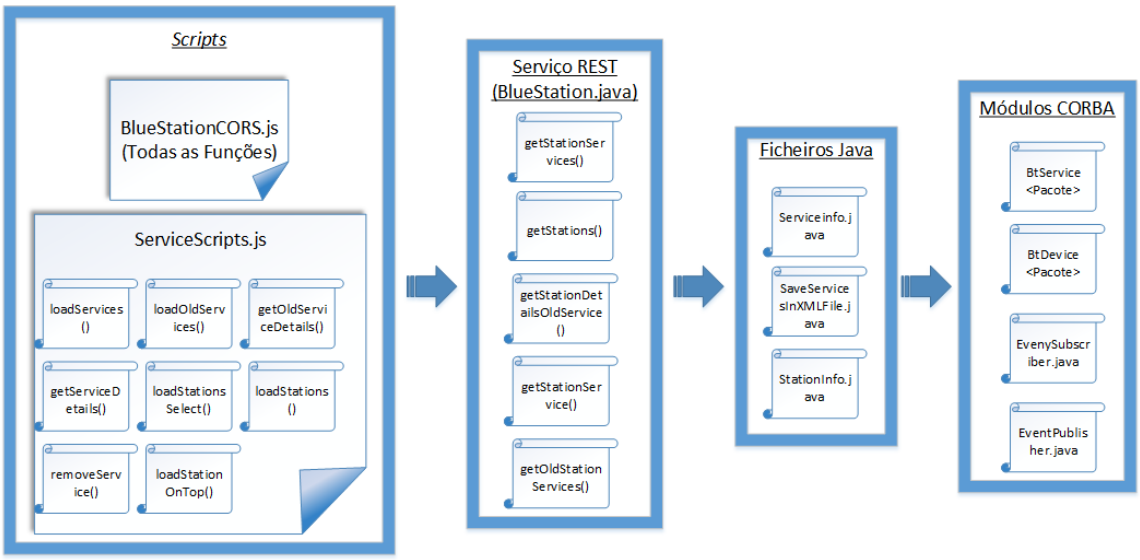

Figura 43 – Ficheiros que compõem a interface ServiceList.html e suas funcionalidades

6.2.5. AddAddressBL.html

Uma das funcionalidades que o sistema instalado nas estações permite é a exclusão de alguns dispositivos através dos endereços MAC. Esta funcionalidade permite excluir determinado dispositivos Bluetooth de receber os conteúdos dos serviços que as estações estejam a transmitir. A *interface* AddAddressBL.html (ver Figura 44) faz parte da categoria *BlackList* e permite adicionar os endereços MAC na *blacklist* das BlueStations.

Figura 44 – Print-screen: Interface AddAddressBL.html

Pode acontecer que algumas pessoas não queiram por alguma razão serem notificadas para receber um serviço por parte de uma ou mais estações. Então esta *interface* permite, através do endereço MAC da placa Bluetooth do equipamento do utilizador, registar o endereço na *blacklist*

de uma ou mais estações e assim quando esse endereço é detetado, a estação automaticamente exclui esse endereço de ser notificado sobre qualquer serviço.

Os ficheiros que fazem parte da *interface* AddAddressBL.html encontram-se ilustrados na Figura 45.

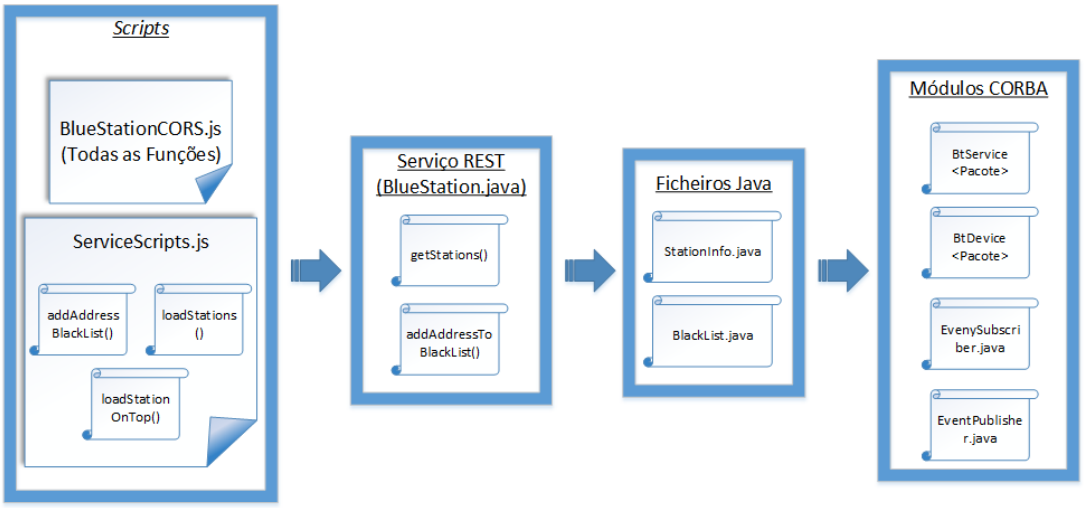

Figura 45 - Ficheiros que compõem a interface AddAddressBL.html e suas funcionalidades

6.2.6. ListAddressBL.html

Nesta *interface* (ver Figura 46) é possível fazer a gestão dos endereços MAC das placas Bluetooth dos equipamentos que se encontram na *blacklist*. É possível listar por estação esses endereços e remover o endereço desejado ou remover de uma só vez todos os endereços da *blacklist* da estação selecionada.

Figura 46 - Print-screen: Interface ListAddressBL.html

Os ficheiros que fazem parte da *interface* ListAddressBL.html encontram-se ilustrados na Figura 47.

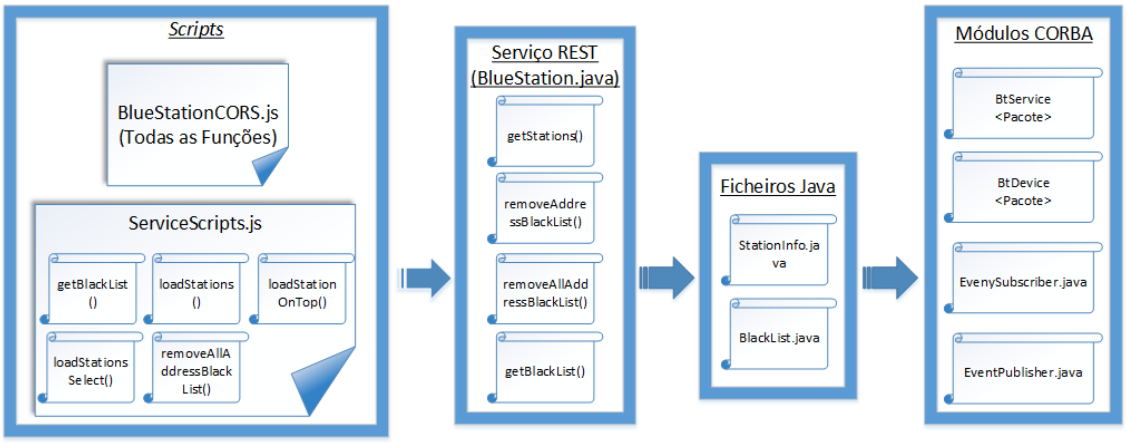

Figura 47 - Ficheiros que compõem a interface ListAddressBL.html e suas funcionalidades

7. COMPARAÇÃO DAS APLICAÇÕES

Um dos principais objetivos deste projeto centrou-se em criar uma aplicação Web que substituísse a aplicação consola de administração e que fossem introduzidos atributos de qualidade que melhorem a utilização do sistema. Estes atributos são apresentados comparando a antiga aplicação consola de administração com a aplicação Web. Esta estratégia visa entender quais as limitações e problemas de usabilidade que a consola de administração apresentava, como essas limitações foram superadas e como os novos atributos de qualidade foram introduzidos com a nova aplicação.

7.1. REQUISITOS NECESSÁRIOS PARA EXECUTAR CADA UMA DAS APLICAÇÕES

- **Aplicação Consola de Administração (Antiga Aplicação)**

    1) Acesso à internet.

    2) *Software* que permita aceder remotamente ao servidor (por exemplo: Putty) onde a aplicação se encontra instalada, para que seja possível executar e utilizar a aplicação.

    3) Conhecimentos básicos de alguns comandos Linux, de forma a encontrar e executar o ficheiro da aplicação.

- **Aplicação Web (Nova Aplicação)**

    1) Acesso à internet.

    2) Utilizar um *browser* (por exemplo: Google Chrome) para aceder à aplicação Web utilizando o URL (http://62.28.14.4:8080/BlueStation/TextService.html).

Com a aplicação Web o utilizador deixa de ter a necessidade de um conhecimento prévio de comandos Linux, de recorrer a um *software* de acesso remoto e de saber em que diretório do servidor se encontra a aplicação. Com a nova aplicação o utilizador apenas necessita de ter um *browser* instalado na sua máquina e de aceder ao URL da aplicação.

7.2. *INTERFACE* E USABILIDADE

- **Aplicação Consola de Administração (Aplicação Existente)**

    1) *Interface* por linha de comandos (ver Figura 48).

    2) Curva de aprendizagem grande.

- **Aplicação Web (Nova Aplicação)**

    1) *Interface* gráfica (ver Figura 49).

    2) Curva de aprendizagem mais pequena.

Na aplicação existente a *interface* por linha de comandos revelava-se um dos grandes obstáculos para a utilização da aplicação. A *interface* não cumpre alguns princípios das heurísticas de Nielsen, nomeadamente falar a linguagem do utilizador, desfazer e refazer ações no sistema, prevenir que o utilizador cometa erros e não utiliza o princípio de reconhecimento invés de lembrança (esta aplicação impõe ao utilizador recordar-se ou fazer a listagem dos comandos para executar uma tarefa).

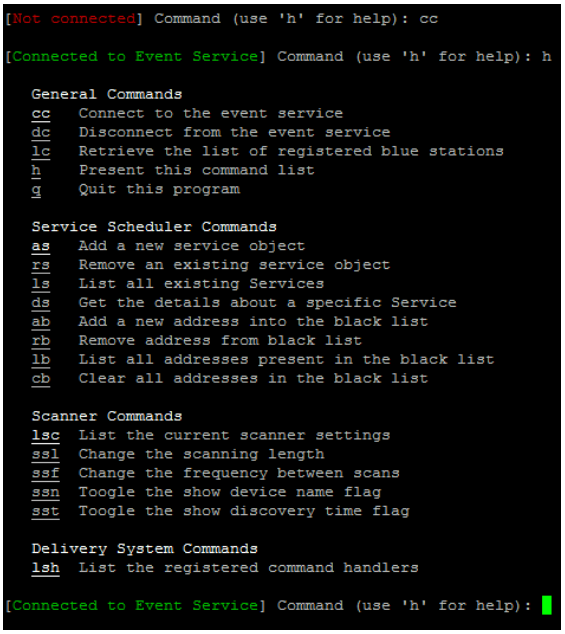

Figura 48 – Print-screen: Interface da Aplicação Consola de Administração

Uma aplicação que utiliza uma *interface* por linha de comandos é pensada mais para o sistema do que para o utilizador. Nos dias que correm os utilizadores já se encontram tão familiarizados com uma *interface* gráfica, que quando se deparam com uma *interface* por linha de comandos a recusa em utilizar este tipo de aplicação é quase imediata.

Na nova aplicação é utilizada uma *interface* gráfica, que implementa todas as funcionalidades que a aplicação existente implementa e acrescenta princípios de qualidade à interação entre o utilizador e a aplicação. Na implementação desta aplicação foram tidos em conta os princípios de usabilidade de Nielsen, para que os problemas de usabilidade mencionados na aplicação existente não se repetissem, e tornassem a aplicação mais adequada ao sistema e ao utilizador.

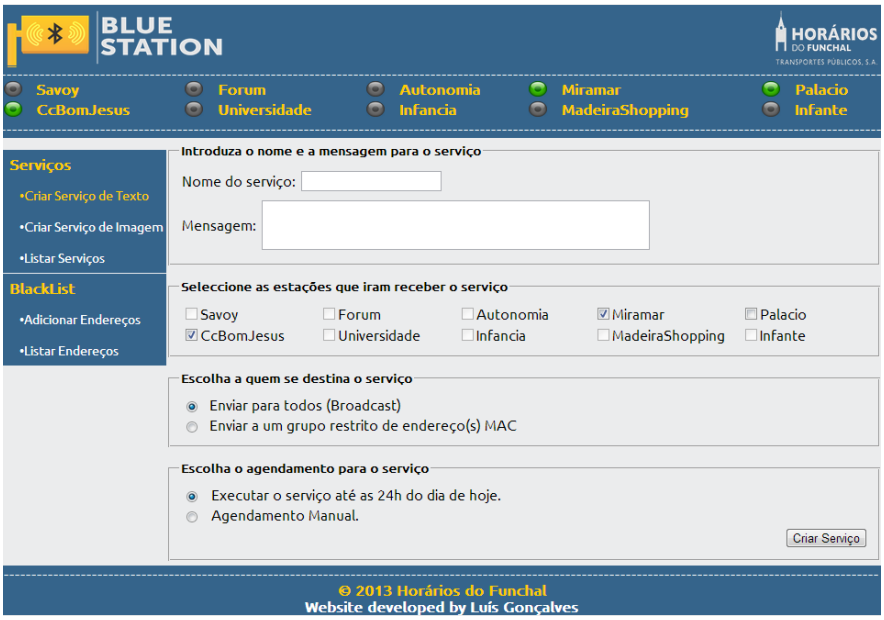

Figura 49 – Print-screen: Interface da Aplicação Web

7.3. FUNCIONALIDADES

- **A Aplicação Consola de Administração permite:**

1) Listar as estações que estão registadas no sistema.

2) Criar um serviço de texto ou de ficheiro.

3) Enviar serviços a um conjunto específico de dispositivos (os endereços MAC da placa Bluetooth dos dispositivos são introduzidos um a um).

4) Remover serviços que estão em execução nas estações.

5) Listar os serviços que estão em execução nas estações.

6) Adicionar um dispositivo Bluetooth nas *blacklist* das estações.

7) Listar os dispositivos Bluetooth que se encontram nas *blacklist* das estações.

8) Retirar os dispositivos Bluetooth que se encontram nas *blacklist* das estações.

- **A Aplicação Web permite:**

    1) Visualizar o estado de funcionamento das estações quase em tempo real.

    2) Listar as estações que estão registadas no sistema.

    3) Criar um serviço de texto ou ficheiro.

    4) Adicionar destinatários para receber o serviço, seja de forma manual ou automatica através de um ficheiro de texto com vários endereços MAC das placas Bluetooth dos equipamentos.

    5) Fazer *upload* de um ficheiro, diretamente da aplicação.

    6) Associar um ficheiro ao serviço criado utilizando o URL do ficheiro.

    7) Remover serviços que estão em execução nas estações.

    8) Listar os serviços que estão em execução nas estações.

    9) Consultar os registos de todos serviços criados.

    10) Adicionar dispositivos Bluetooth nas *blacklist* das estações.

    11) Listar os dispositivos Bluetooth que se encontram nas *blacklist* das estações.

    12) Remover os dispositivos Bluetooth que se encontram nas *blacklist* das estações.

Todas as funcionalidades existentes na consola de administração foram mantidas na aplicação Web e além dessas funcionalidades foram inseridas as seguintes: carregamento de ficheiros para serem associados aos serviços utilizando a própria aplicação, leitura de ficheiros de texto com os endereços MAC das placas Bluetooth evitando assim terem que ser inseridos um a um de cada vez que o utilizador os quisesse utilizar, estado de funcionamento das estações quase em tempo real e por fim o registo de todos os serviços criados em cada uma das estações.

8. TESTE DE USABILIDADE

Neste capítulo serão abordados os passos realizados para os testes de usabilidade e os resultados obtidos nos testes efetuados.

De forma a determinar erros ou dificuldades resultantes da utilização da aplicação, que impeçam o administrador de ter uma experiência satisfatória e de cumprir as metas pretendidas, foram realizados um conjunto de testes de usabilidade.

8.1. PREOCUPAÇÕES

No desenvolvimento de uma aplicação vão surgindo dúvidas e preocupações, nomeadamente se determinada conceção será do agrado dos utilizadores, se as opções de desenho são as mais adequadas para as tarefas que serão desempenhadas e se a aplicação transmite ao utilizador as expectativas para a qual foi desenhada.

Ao longo do desenvolvimento foram surgindo as seguintes preocupações/dúvidas:

- Será que os utilizadores vão ser capazes de navegar com sucesso na aplicação?
- O menu lateral que permite a navegação entre as *interfaces* encontra-se organizado e agrupado de forma que os utilizadores consigam efetuar as operações que pretendem?
- Será que a aplicação fornece toda a informação necessária aos utilizadores e que esta informação ajuda-os durante a realização das tarefas?
- Será que a aplicação transmite aos utilizadores os resultados das suas ações?
- Será que o mecanismo de *pop-up* para o carregamento de ficheiros será intrusivo ou o utilizador se sentirá perdido quando voltar à tarefa que estava a realizar?
- Será que existirão tarefas que os utilizadores queiram realizar e que a aplicação não suporte?
- Será que a combinação de cores das *interfaces* será do agrado dos utilizadores? Será que realçam aspetos essenciais e auxiliam à utilização da aplicação?

8.2. PÚBLICO-ALVO E TAREFAS

A aplicação desenvolvida tem como finalidade controlar um sistema distribuído, como já foi descrito ao longo desta dissertação. Este sistema neste momento é utilizado pela empresa Horários do Funchal e portanto o público-alvo da aplicação é restrito a alguns membros da empresa. Posto isto, foram realizados os testes de usabilidade a duas pessoas que ficaram responsáveis por utilizar a aplicação. As duas pessoas, com idades entre os 30 e 40 anos, tinham um bom nível informático.

Estas duas pessoas, que serão os administradores do sistema, realizaram um conjunto de tarefas, que lhe foram detalhadas sobre a forma de vários cenários que tinham de realizar utilizando a aplicação Web.

Os cenários atribuídos nos testes de usabilidade têm como objetivo que os administradores realizem as seguintes tarefas:

1) Adicionar um serviço de texto numa estação ativa, tendo como destino todos os utilizadores, escolher a opção de agendamento automática, ou seja, que o serviço se execute até às 24h do presente dia;

2) Adicionar um serviço de texto em todas as estações ativas, tendo como destino um conjunto de utilizadores sendo que o endereço MAC da placa Bluetooth desses utilizadores encontra-se num ficheiro com a extensão .txt que o administrador terá que carregar para a aplicação Web;

3) Adicionar um serviço de ficheiro (uma imagem), em todas as estações ativas para ser executado e finalizado num dia específico a uma hora específica à sua escolha, tendo como destino todos os utilizadores;

4) Listar e eliminar todos os serviços que se encontrem em execução numa estação à escolha;

5) Adicionar um dispositivo Bluetooth através do seu endereço MAC, na *blacklist* de todas as estações e listar os dispositivos Bluetooth que se encontram na *blacklist* de uma estação;

6) Listar e eliminar todos os dispositivos Bluetooth que se encontrem na *blacklist* de uma estação.

No final de cada tarefa realizada os administradores foram incentivados a partilharem os seus pensamentos sobre as ações que estavam a desempenhar na aplicação, reportando, por exemplo, dificuldades ou aspetos que lhe fizessem confusão e o que achavam da *interface* da aplicação.

8.3. Avaliação de Usabilidade da Aplicação Web

Dos testes de usabilidade realizados foram retiradas as seguintes anotações:

- Problemas reportados;
- Erros cometidos;
- Estado de conclusão da tarefa (Concluída com sucesso, com dificuldades e não concluída).

De forma a avaliar os problemas e erros cometidos, foi adotada a avaliação de heurísticas de Nielsen.

8.4. Resumo dos Resultados

Concluídos os testes de usabilidade, foram reunidos e analisados os resultados, sendo estes apresentados de seguida, para cada uma das tarefas realizadas:

1) "Adicionar um serviço de texto numa estação ativa, tendo como destino todos os utilizadores, escolher a opção de agendamento automática, ou seja, que o serviço se execute até às 24h do presente dia.";

- Número de participantes que completou a tarefa:
  - 100% (2 em 2).

- Problemas encontrados, aplicando as heurísticas de Nielsen:
  - Prevenção de erros: 100% (2 em 2):
    - Os participantes não visualizavam o estado de funcionamento das estações na barra superior da *interface* (ver Figura 49) e selecionavam estações que não estavam a funcionar para receber os serviços;

- Estado de conclusão da tarefa:
  - Concluído com sucesso: 100% (2 em 2);
  - Concluído com dificuldade: 0% (0 em 2), e;
  - Não concluído: 0% (0 em 2).

2) "Adicionar um serviço de texto em todas as estações ativas com agendamento para 5 anos, tendo como destino um conjunto de utilizadores onde o endereço MAC da placa Bluetooth desses utilizadores encontra-se num ficheiro com a extensão .txt que o administrador terá que carregar para a aplicação Web";

- Número de participantes que completou a tarefa:
  - 100% (2 em 2).

- Problemas encontrados, aplicando as heurísticas de Nielsen:
  - Prevenção de erros: 100% (2 em 2):
    - Aqui, aplicou-se o mesmo problema da tarefa **1)**.
  - Flexibilidade e eficiência de uso: 100% (2 em 2):
    - A tarefa de inserir os endereços MAC um a um, revelou-se pouco eficiente e maçadora para os participantes, que nas suas sugestões referiram que seria importante ter outro método para a inserção destes endereços;
    - Quando o agendamento se prolongava por um período de tempo muito longo, o *widget* da data revelou-se pouco eficiente e levava os participantes a perderem muito tempo.
- Estado de conclusão da tarefa:
  - Concluído com sucesso: 100% (2 em 2);
  - Concluído com dificuldade: 0% (0 em 2), e;
  - Não concluído: 0% (0 em 2).

3) "Adicionar um serviço de ficheiro (uma imagem), em todas as estações ativas para ser executado e finalizado num dia específico a uma hora específica à sua escolha, tendo como destino todos os utilizadores";

- Número de participantes que completou a tarefa:
  - 100% (2 em 2).
- Problemas encontrados, aplicando as heurísticas de Nielsen:
  - Prevenção de erros: 100% (2 em 2):
    - Aqui, aplicou-se o mesmo problema da tarefa **1)**.
- Estado de conclusão da tarefa:
  - Concluído com sucesso: 100% (2 em 2);
  - Concluído com dificuldades: 0% (0 em 2), e;
  - Não concluído: 0% (0 em 2).

4) "Listar e eliminar todos os serviços que se encontrem em execução numa estação à escolha";

- Número de participantes que completou a tarefa:
  - 100% (2 em 2).
- Problemas encontrados, aplicando as heurísticas de Nielsen:
  - Prevenção de erros: 100% (2 em 2):
    - Aqui, aplicou-se o mesmo problema da tarefa **1)**.

- Estado de conclusão da tarefa:
  - Concluído com sucesso: 100% (2 em 2);
  - Concluído com dificuldades: 0% (0 em 2), e;
  - Não concluído: 0% (0 em 2).

5) "Adicionar um dispositivo Bluetooth através do seu endereço MAC, na *blacklist* de todas as estações e listar os dispositivos Bluetooth que se encontram na *blacklist* de uma estação";

- Número de participantes que completou a tarefa:
  - 100% (2 em 2).

- Problemas encontrados, aplicando as heurísticas de Nielsen:
  - Prevenção de erros: 100% (2 em 2):
    - Aqui, aplicou-se o mesmo problema da tarefa **1)**.

- Estado de conclusão da tarefa:
  - Concluído com sucesso: 100% (2 em 2);
  - Concluído com dificuldades: 0% (0 em 2), e;
  - Não concluído: 0% (0 em 2).

6) "Listar e eliminar todos os dispositivos Bluetooth que se encontrem na *blacklist* de uma estação";

- Número de participantes que completou a tarefa:
  - 100% (2 em 2).

- Problemas encontrados, aplicando as heurísticas de Nielsen:
  - Prevenção de erros: 100% (2 em 2):
    - Aqui, aplicou-se o mesmo problema da tarefa **1)**.

- Estado de conclusão da tarefa:
  - Concluído com sucesso: 100% (2 em 2);
  - Concluído com dificuldades: 0% (0 em 2), e;
  - Não concluído: 0% (0 em 2).

8.5. CONCLUSÕES E ALTERAÇÕES

Com a realização dos testes de usabilidade foram identificados problemas que os participantes evidenciaram na utilização da aplicação Web. Com base nos resultados destes testes e com o *feedback* fornecido pelos participantes, foram efetuadas algumas alterações na aplicação, tais como:

- Para que os administradores não enviem serviços para estações que não estão ativas, a solução encontrada foi bloquear as *checkbox* destas estações e assim impedir que o administrador cometa os erros detetados nos testes de usabilidade. Estas estações não são simplesmente excluídas da aplicação, porque estaria a ser desrespeitada a heurística de Nielsen: manter o estado do sistema visível;

- Trocar o *widget* que permite introduzir a data para agendar o tempo de vida de um serviço, por outro que fosse mais eficiente;

- Alterar determinados termos utilizados na aplicação;

- Adicionar a possibilidade de introduzir endereços MAC das placas dos dispositivos Bluetooth, através da leitura de um ficheiro .txt.

- Refazer o *layout* das *interfaces*. De acordo com o *feedback* dos participantes, revelaram que a combinação de cores utilizada nas *interfaces* não era a mais favorável à visualização.

Depois de todas as alterações concluídas, a aplicação Web foi disponibilizada aos elementos dos Horários do Funchal que participaram nos testes de usabilidade e que ficaram como administradores do sistema. Durante o período em que a aplicação foi disponibilizada aos administradores foi fornecido *feedback*, e as falhas reportadas prontamente foram resolvidas.

9. CONCLUSÕES E TRABALHO FUTURO

Ao longo do processo de engenharia reversa, foram encontradas inúmeras dificuldades, as quais foram ultrapassadas com sucesso. As dificuldades sentidas deveram-se:

- À linguagem de programação utilizada para implementar o sistema;
- Ao estado em que o *hardware* se encontrava;
- Ao sistema operativo obsoleto em uso;
- À falta de documentação que explicasse com clareza como o sistema estava implementado;

Deste processo de engenharia reversa foi colmatada a falta de documentação sobre como o sistema foi projetado e implementado, resolvendo uma das dificuldades encontradas inicialmente. Assim futuras implementações e manutenções têm explicado, em detalhe, como o sistema está implementado, poupando tempo e custos associados a essa intervenção.

De realçar o facto de o sistema possuir na sua implementação a arquitetura CORBA, para suportar as comunicações da aplicação de gestão e criação de serviços com as estações, o que foi determinante no processo de engenharia reversa. Está arquitetura permitiu replicar a consola de administração, sem a necessidade de proceder a alterações do código fonte do sistema. Ao longo do processo o componente do sistema que sofreu uma alteração de forma a cumprir com os requisitos propostos pela empresa, foi a aplicação que permite a gestão e criação de serviços para serem adicionados às BlueStations.

Após a modelação e implementação da nova aplicação, foram realizados os testes de usabilidade que evidenciaram o grande entusiamo dos atuais utilizadores do sistema, em verem os resultados obtidos. Destacando a facilidade de utilização da aplicação, a simplicidade em criar um novo serviço para ser enviado para as estações e por fim as novas funcionalidades que foram implementadas.

Com todo o trabalho realizado e os requisitos inicialmente propostos cumpridos, é com enorme satisfação que se pode afirmar que o sistema se encontra de novo em funcionamento e a ser utilizado pelos Horários do Funchal. A empresa ficou muito satisfeita com o resultado final de todo o projeto e preparou inclusive uma divulgação à comunicação social, para apresentar o sistema e sensibilizar os utentes sobre o novo serviço.

De forma a finalizar esta dissertação, segue-se uma lista de futuros trabalhos que podem ser realizados a fim de melhorar o sistema abordado nesta tese de mestrado:

- Ao longo deste trabalho uma das grandes limitações que impediu que se tenha um sistema mais robusto, constituído por um leque mais diversificado de funcionalidades, e que o envio de ficheiros revela alguns problemas quando é dirigido a um grande número de pessoas, deve-se às limitações da tecnologia Bluetooth. Todavia, é de realçar que o sistema funciona perfeitamente para o envio de conteúdos de texto e para as restantes funcionalidades. Posto isto, e para a finalidade com que as estações são utilizadas, a mudança para outra tecnologia (nomeadamente o WiFi) que permita

contornar todas as limitações da tecnologia Bluetooth e ter um leque de novas abordagens, seria muito interessante para este tipo de sistema.

- Devido à falta de verbas não foi possível proceder à manutenção de algumas estações, que necessitam uma intervenção a nível de hardware. Seria essencial colocar estas estações em funcionamento, cobrindo assim novas áreas com o serviço.

- Integração de novas funcionalidades na aplicação, nomeadamente uma componente que permita consultar e gerir a base de dados, que contém toda a informação relacionada com os dados recolhidos pelas estações e os serviços que nelas são executados.

- Dinamizar o sistema através da comunicação social e da publicidade, de forma a sensibilizar as pessoas para a sua utilização.

- Procurar que o sistema seja rentável a nível financeiro para a empresa, através do aluguer/venda do sistema para outras entidades, por exemplo a hotéis.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ATOJI, Rodolpho Iemini. *Bluetooth e NFC: Estudo de Caso.* 2010. 63 f. Dissertação. Universidade de São Paulo, São Paulo. Disponível em: <http://www.ime.usp.br/~cef/mac499-10/monografias/rodolpho/pdf/mac499-monografia.pdf>. Acesso em 14 Novembro. 2013.

[2] BRITO, Alisson Vasconcelos. *Sistema de Transmissão de Vídeo para Vigilância Utilizando Bluetooth*. 2003. 96 f. Dissertação (Pós-Graduação em Informática) - Universidade Federal de Campina Grande, Campina Grande. Disponível em: <http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2003/Dissertacao_AlissonVasconcelosDeBrito.pdf>. Acesso em 4 Março. 2013.

[3] GROUP, Core Specification Working. *Simple Pairing Whitepaper*. Jun. 2006. Disponível em: <http://mclean-linsky.net/joel/cv/Simple%20Pairing_WP_V10r00.pdf>. Acesso em 7 Março. 2013.

[4] ALBUQUERQUE, Hugo Rodrigues. *Estudo sobre Pilhas Bluetooth e as suas Limitações em Sistemas Embarcados*. 2010. 50 f. Dissertação (Mestrado em Engenharia da Computação) - Universidade Federal de Pernambuco, Recife. Disponível em: <http://www.cin.ufpe.br/~tg/2010-2/hra.docx>. Acesso em 9 Março. 2013.

[5] SIQUEIRA, Thiago Senador. *Bluetooth - Características, protocolos e funcionamento*. Institudo de Computação - Universidade Estadual de Campinas, Campinas. Disponível em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf>. Acesso em 14 Março. 2013.

[6] MAIA, Eduardo Mendes de Sousa. *Redes e Protocolos Bluetooth*. Artigo. Disponível em: <http://www.academia.edu/1339122/Desvendando_o_bluetooth>. Acesso em 19 Março. 2013.

[7] BHAGWAT, Pravin. *Bluetooth: Technology for Short-Range Wireless App*. Jun 2001. Artigo - Reefedge Inc, Industry Report, IEEE Internet Computing. Disponível em: <http://erdos.csie.ncnu.edu.tw/~ccyang/WirelessNetwork/Papers/Bluetooth/BTIntro-1.pdf>. Acedido em 24 Março. 2013.

[8] LINK, Eduardo.; ALEXANDRE, Everton.; WOLF, Joe.; STRZYKALSKI, Marcelo. *Uma Introdução ao CORBA*. Disponível em: <http://www.inf.pucrs.br/~gustavo/disciplinas/sd/material/Artigo_Corba.pdf>. Acesso em 1 Abril. 2013.

[9] JALK, Michael. *REST Representational State Transfer*. Artigo - University of Technology Vienna. Disponível em: <http://blog.interlinked.org/static/files/rest.pdf>. Acesso em 4 Abril. 2013.

[10] BIGOLIN. M. *REST*. 2012. Dissertação (Mestrado em Engenharia da Computação) - Universidade de São Paulo, São Paulo. [Online]. Disponível em: <http://saloon.inf.ufrgs.br/twiki-data/Disciplinas/CMP167/TF12MarcioBigolin/Textofinal.pdf>. Acesso em 10 Abril. 2013.

[11] XAVIER, Otávio Calaça. *Serviços Web Semânticos Baseados em RESTful*. 2011. 137 f. Dissertação (Mestrado em Informática) - Universidade Federal de Goiás. Disponível em: <http://www.inf.ufg.br/mestrado/sites/www.inf.ufg.br.mestrado/files/uploads/Dissertacoes/OtavioCala%C3%A7a.pdf>. Acesso em 22 Abril. 2013.

[12] "WINDLEY'S, Philip. *REST: Representational State Transfer*. 2005. Artigo. Disponível em: <http://oreilly.com/catalog/pwebserperl/chapter/ch11.pdf>. Acesso em 25 Abril. 2013.

[13] EXOPLATFORM. *eXo JCR Developer Guide*. 2010. Artigo - Community Projects - JBoss Community. Disponível em: <http://docs.jboss.org/exojcr/1.12.13-GA/developer/en-US/html/ch-introduction-to-rest.html#d0e16289>. Acesso em 10 Maio. 2013

[14] FIELDIG, Roy Thomas. *Architectural Styles and the Design of Network - based Software Architectures*. 2000. 180 f. Dissertação (Mestrado em Informática) - Universidade da Califórnia, Irvine. Disponível em: <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em 15 Maio. 2013.

[15] CAMACHO, Tiago. *Proximity Sensing and Context-Aware Content Dissemination.* 2009. 127 f. Dissertação (Mestrado em Engenharia de Informática) - Universidade da Madeira, Madeira.

[16] CONSTANTINE, Larry. *The Usage-Centered Design Resource.* 2006. Artigo. Disponível em: <http://www.foruse.com/articles/activitymodeling.pdf>. Acessoo em 29 Maio. 2013.