FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


# EFACEC - Atualização Automática de sistemas SCADA em produção

**Luís Miguel Machado Rodrigues**

PARA APRECIAÇÃO POR JÚRI

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Orientador: Prof. Doutor Mário Jorge Sousa

Co-orientador: Eng. Paulo Jorge Santos

25 de Junho de 2018

# Resumo

Esta dissertação consiste na idealização e implementação de um Módulo para a gestão de atualizações automáticas de Sistemas SCADA em produção.

O ScateX# constitui a última geração da solução SCADA da EFACEC e inclui sistemas de treino de operadores e gestão de equipas, entre outros. Este é utilizado para: redes; aplicações DMS e EMS avançadas para transmissão; distribuição ou geração de energia; aplicações para redes ferroviárias e metropolitanos; supervisão de infraestruturas técnicas.

Atualmente, a atualização em ambiente de produção do Sistema SCADA comporta custos consideráveis de mão de obra, bem como tempo de indisponibilidade do Sistema incompatíveis com a eficiência exigida pelo mercado.

Com vista a minimizar estes custos, foi idealizado um Módulo de gestão de atualizações que realize automaticamente o *Upgrade* dos postos de trabalho do Sistema. Este Módulo consistirá no desenvolvimento de uma interface *web* onde será possível: fazer o *upload* das atualizações; listar os postos de trabalho ligados à rede; agendar atualizações para os postos de trabalho pretendidos; realizar o controlo de versões; efetuar as configurações gerais dos postos de trabalho. Para além da interface *web*, será necessário criar uma Aplicação *Windows* que correrá nos postos de trabalho com o intuito de servir como ponte de comunicação entre o posto e o servidor. Esta aplicação será, também, responsável pelo *download* e atualização nos postos.

# Abstract

This dissertation consists on the idealization and implementation of a Module for the Management of the Updates for SCADA Software in the production workstations.

ScateX# includes systems of training of operators and management of teams, among others. This is the latest generation of EFACEC's SCADA solution for networks and it is used: for advanced DMS and EMS applications for transmission, distribution or power generation; on applications for rail and metro networks; for supervision of technical infrastructures.

Currently, the upgrade on a production environment of the SCADA System entails considerable costs of labor as well as time of unavailability of the System incompatible with the efficiency demanded by the market.

In order to minimize these costs, an update management Module has been devised, which automates the upgrade of the System workstations. This module will consist on the development of: a web interface, where you can upload the updates; a list of the workstations connected to the network; a schedule of updates to the desired workstations; a version control; general settings of the workstations. In addition to the web interface, it will also be necessary to create a Windows Application that will run in the workstations. This Application will be useful to provide a communication bridge between the post and the server and will also be responsible for downloading and updating the posts.

# Agradecimentos

Inspirado nos pensamentos de Agostinho da Silva, “O homem não nasce para trabalhar, o homem nasce para criar, para ser o tal poeta à solta.”. No entanto, homem criativo não cria sozinho. Este homem é influenciado pelo ambiente que o rodeia e, principalmente, pelas pessoas que moldam esse ambiente. Assim sendo, gostaria de agradecer:

Ao Professor Mário Jorge Sousa, docente da Faculdade de Engenharia da Universidade do Porto e orientador do meu percurso na realização desta dissertação.

Ao Engenheiro Paulo Santos da EFACEC que se tornou num apoio essencial, orientando-me da melhor forma para que este projeto se tornasse bem sucedido. Representou uma presença imprescindível quer na estruturação do trabalho em si, quer devido à sua disponibilidade em ler e sugerir alterações aquando da elaboração desta dissertação.

À minha namorada pela persistência e motivação para que eu atingisse um patamar que cumprisse o nível de exigência que esta dissertação requer. Pela ajuda que proporcionou na revisão deste trabalho e pelo encorajamento que me deu quando a exaustão se aproximava.

À minha família pelo esforço e investimento que fazem diariamente para que eu me possa instruir, desenvolver intelectualmente e pessoalmente para, futuramente, me tornar num bom profissional e atingir as expetativas e confiança que depositaram em mim.

À EFACEC pelo projeto desafiante e pela oportunidade de o desenvolver nas suas instalações com todas as condições necessárias para realizar um bom trabalho.

Aos meus colegas da EFACEC por me terem proporcionado um ambiente de trabalho adequado, com fácil adaptação e integração na empresa.

Aos meus amigos por me terem aconselhado na realização de certas decisões e incentivado para conseguir cumprir esta tarefa de conclusão de curso.

A todas a pessoas que participaram e me ajudaram a ultrapassar situações adversas.

Miguel Rodrigues

.

*"O homem não nasce para trabalhar,*
*o homem nasce para criar, para ser o tal poeta à solta."*

Agostinho da Silva

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Abreviaturas

**API** Application Programming Interface

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**HTTPS** HyperText Transfer Protocol Secure

**SOAP** Simple Object Access Protocol

**REST** Representational State Transfer

**CORBA** Common Object Request Broker Architecture

**WWW** *World Wide Web*

**XML** Extensible Markup Language

**IDE** Integrated Development Environment

**SQL** Structured Query Language

**RDBMS** Relational Database Management System

**JSON** JavaScript Object Notation

**JSP** JavaServer Pages

**JEE** Java Enterprise Edition

**JDBC** Java Database Connectivity

**SCADA** Supervisory control and data acquisition

**MVC** Model View Controller

**M2E** Maven para Eclipse

**CSS** Cascading Style Sheets

**SSH** Secure Shell

**SCP** Secure Copy Protocol

**OMS** Outage Management System

**DMS** Distribution Management System

**EMS** Energy Management System

**UI** User Interface

**WKS** Workstation

**SCD** SCADA

**SAH** Sistema de Arquivo Histórico

**RBAC** Role-Based Access Control

**RPC** Remote Procedure Call

**SMTP** Simple Mail Transfer Protocol

**FTP** File Transfer Protocol

**WSDL** Web Services Description Language

# Capítulo 1

# Introdução

## 1.1 Apresentação do Grupo Efacec

A EFACEC é o maior grupo eletromecânico Nacional de capitais portugueses na área da energia, engenharia e mobilidade. Em 1948, surgiu a EFME- Empresa Fabril de Máquinas Elétricas cuja evolução deu origem à EFACEC, em 1962, e posteriormente, nasceu a EPS-EFACEC Power Solutions que é formada por um conjunto de sociedades [1]. Este grupo possui várias Unidades de Negócio cujas atividades abrangem a conceção e produção de equipamentos, o design de sistemas e a criação de soluções em 3 grandes áreas:

- Produtos de Energia - Transformadores, Aparelhagem, Service e Automação;
- Sistemas - Energia, Ambiente e Industria, Transportes;
- Mobilidade - Mobilidade Elétrica;

Este leque de atividades de negócio permite que a EFACEC satisfaça as necessidades do mercado não apenas nacional, mas também, internacional, marcando a sua presença em 65 países de 5 continentes diferentes.

O departamento de Automação tem como função projetar, desenvolver e fornecer soluções, utilizando tecnologia inovadora, para redes elétricas, sistemas ferroviários, gestão de infra-estruturas entre outras aplicações. Na área de Automação de Sistemas existem várias soluções para Sistemas de Gestão de Redes (Supervisory control and data acquisition (SCADA)/Distribution Management System (DMS)/Outage Management System (OMS)/Energy Management System (EMS)), sendo esta dissertação focada em sistemas SCADA [2].

As tarefas relativas à gestão operacional de redes de energia constitui um problema devido ao facto das suas infra-estruturas se situarem relativamente dispersas em áreas geograficamente distintas. Nestes sistemas é de extrema importância a utilização de ferramentas que garantem a operação contínua e manutenção da elevada qualidade da concentração de informação relativa aos seus processos e com a possibilidade de intervenção remota nesses mesmos processos. Tais atividades são asseguradas por sistemas altamente especializados: os sistemas SCADA.

Sistemas SCADA são sistemas de supervisão, controlo e aquisição de informação de processos industriais. Estas características fazem dos sistemas SCADA estruturas complexas que integram conceitos e tecnologias diversas. A sua implementação requer a utilização de equipamentos especializados (sensores, atuadores, unidades de aquisição, etc), capacidade de comunicação rápida por utilização de vários protocolos e a existência de um sistema central capaz de receber, gerir e apresentar a informação [3].

A EFACEC propõe um sistema SCADA para redes de longa duração, o *ScateX#*. Este sistema contém:

- Aplicações DMS e EMS avançadas para transmissão, distribuição ou geração de energia;
- Aplicações para redes ferroviárias e metropolitanos ou supervisão de infraestruturas técnicas;
- Sistemas de treino de operadores e gestão de equipas [4].

## 1.2 Enquadramento e Motivação

Atualmente, na empresa EFACEC, existe uma questão que necessita de ser abordada: a instalação automática de novas Versões do *ScateX#* nos postos de trabalho em produção. Esta necessidade de automatizar os processos de atualização dos postos de trabalho é importante para reduzir os custos e horas de trabalho gastas nestes processos. O método praticado encontra-se desatualizado e não é fiel ao projeto de inovação da empresa.

A solução idealizada é a criação de uma Plataforma de gestão de atualizações. Esta Plataforma consistirá no desenvolvimento de uma Interface *web* onde será possível fazer *Upload* das atualizações, listar os postos de trabalho ligados à rede, agendar atualizações para os postos de trabalho pretendidos e realizar o controlo de versões e configurações gerais dos postos de trabalho. Para além da Interface *Web*, será, também, criada uma Aplicação *Windows* que correrá nos postos de trabalho para averiguar se existem notificações ou alterações na Interface *Web* referente ao posto.

Uma das minhas maiores motivações consiste em adquirir conhecimentos, permitindo desenvolver o meu intelecto, tornando-me mais apto para qualquer funcionalidade desta área. Para além disto, saber que o meu contributo foi importante para a automatização de processos com grande impacto numa empresa como a EFACEC, é algo definitivamente gratificante e de concretização pessoal e profissional.

## 1.3 Objetivos

Tendo em conta a importância de tornar este processo de atualizações mais eficiente, torna-se imprescindível desenvolver uma Plataforma que tenha a capacidade de automatizar as instalações de atualização nos postos de trabalho, mediante os critérios do utilizador e independentemente de onde estejam localizados ou do tipo de cliente que vai utilizar o *Software*. Com isto, o desenvolvimento de um Módulo para gestão de atualizações deve contemplar os seguintes objetivos:

- Estudo sobre o *SCADA*, principais falhas, funcionalidades e limitações que possam influenciar o desenvolvimento da Plataforma;
- Estudo e pesquisa sobre as limitações que são impostas por um *Windows Service*;
- Estudo e pesquisa sobre o *Web Service* mais correto a utilizar no projeto;
- Deve permitir que o utilizador possa listar e configurar os postos de trabalho;
- Deve permitir que o utilizador possa agendar as atualizações;
- Deve permitir fazer *Upload* das atualizações na Plataforma *Web*;
- Devem permitir que o utilizador seja notificado de novas atualizações, da instalação de uma atualização ou das duas opções ou de nenhuma;
- Deve permitir que uma atualização seja feita ou não, dependendo do tipo de serviços que irá interromper;
- Ter em conta o limite de tráfego da rede quando se transfere uma atualização para os posto de trabalho;
- Usar os melhores métodos de segurança no desenvolvimento do Módulo;
- Ser possível configurar os diferentes tipos de atualizações e notificações nos postos de trabalho.

## 1.4 Estrutura do documento

Este relatório encontra-se dividido num total de 6 capítulos.

O primeiro capítulo, "Introdução", realiza uma apresentação do grupo EFACEC, um enquadramento do projeto, a motivação para o mesmo e menciona os objetivos.

O segundo capítulo, "Estado da Arte", apresenta as tecnologias existentes e os conceitos necessários possuir e compreender para a perceção do enquadramento do trabalho desenvolvido. Neste capítulo apresentam-se, também, as limitações impostas pelas tecnologias, serve para introduzir o leitor às ferramentas que irão ser utilizadas no capítulo 5.

O terceiro capítulo, "Atualização Automática de sistemas SCADA em produção", aborda 3 pontos fulcrais: definição do problema, requisitos e solução propostos pela empresa. Relativamente à definição do problema, especifica-se e caracteriza-se os principais problemas e desafios existentes. Quanto aos requisitos propostos, estes serão listados, divididos em diferentes componentes e classificados tendo em conta a sua prioridade de realização. Por fim, haverá uma descrição da solução proposta que irá futuramente ser testada, obtendo-se a verificação do cumprimento dos objetivos pretendidos.

O quarto capítulo, "Arquitetura do trabalho", ilustra a arquitetura dos diferentes componentes que constituem o projeto. Este irá incluir as tabelas e relações da base de dados e os diagramas de atividades dos diferentes componentes do projeto.

O quinto capítulo, "Implementação", contém o desenvolvimento de todos os diferentes componentes implementados neste projeto, assim como realça as tecnologias mais relevantes para a concretização do trabalho.

Por fim, o sexto capítulo, "Conclusões e Trabalho Futuro", como o nome indica, consiste nas conclusões finais do trabalho e a idealização do trabalho futuro. Neste capítulo final, abordam-se as vantagens e o impacto que este projeto poderá ter quando implementado num cenário de produção real.

# Capítulo 2

# Estado da Arte

Nesta secção apresenta-se uma análise das tecnologias que poderão vir a ser utilizadas no decorrer do projeto. O objetivo é concluir quais serão as mais adequadas para o desenvolvimento deste trabalho.

## 2.1 Sistemas SCADA

O sistema SCADA permite monitorizar e controlar processos geograficamente dispersos, como também, possibilita a comunicação entre estações remotas e um centro de comando, providenciando informação útil para o controlo do processo. Este tipo de sistema é maioritariamente usado para: Gestão de processos industriais; Distribuição de gás, água e energia; Redes de telecomunicações; Redes de transporte, entre outros [5].

Num sistema SCADA ocorre a constante supervisão da rede, controlo da mesma e aquisição dos dados. Relativamente à supervisão da rede, os operadores de comando analisam a informação simples e intuitiva que é emitida pela rede e recolhida nos postos de trabalho. Desta forma, são alertados, em tempo útil, aquando da ocorrência de uma anomalia do processo. No que diz respeito ao controlo da rede, os operadores do sistema SCADA podem atuar, de modo remoto, sobre um processo físico através da execução de controlos a partir dos seus postos de trabalho. Assim, poderá ocorrer a atuação direta e rápida no sistema partindo de um local centralizado e, aquando do surgimento de potenciais problemas, a sua correção possa ser realizada num curto intervalo de tempo. Por fim, a aquisição de dados constitui uma componente fundamental, pois recolhe informação referente ao sistema físico que controla que é posteriormente enviada para o centro do comando. Esta informação pode ser visualizada de imediato (supervisão) como pode ser arquivada em registo histórico [].

## 2.2 ScateX#

ScateX# é um sistema SCADA para aplicações de supervisão e controlo de redes. Este sistema corresponde ao *software* que é instalado nos postos de trabalho e é considerado a mais recente versão da plataforma de gestão de rede da EFACEC. Numa só plataforma integrada disponibiliza: Múltiplas aplicações para operações; Informações de gestão; Gestão de rede de energia; Gestão de interrupções; Monitorização e controlo; Monitorização de ativos; Supervisão técnica.

A construção do ScateX# é feita numa infraestrutura de cibersegurança, em tempo real. Esta inclui recursos inovadores que: fornecem um suporte de decisão aprimorado; acesso fácil e rápido à informação, permitindo uma melhor a perceção situacional. A otimização e eficiência da rede é determinada pelo conjunto de aplicações de gestão de rede, energia e interrupção [6]. Características como: flexibilidade, capacidade de expansão e desempenho, permitem que o ScateX# possa ser dimensionado para satisfazer as necessidades de grandes redes e infraestruturas. Com isto, os clientes poderão adotar estratégias de investimento graduais. Este software pode ser adaptado para atender às necessidades específicas de cada usuário para oferecer melhores soluções de custo / benefício.

A implementação direta, integração e manutenção são um requisito fundamental para qualquer sistema de redes. Logo, o ScateX# apresenta uma engenharia e gestão completa de toda a infraestrutura, desde a sala de controlo à infraestrutura de comunicação e equipamentos remotos. O ScateX# inclui, também, um programa para formação do cliente, tornando os operadores mais aptos para a utilização.

### 2.2.1 Arquitetura do ScateX#

O ScateX# é desenvolvido em conformidade com os padrões estabelecidos pela indústria relativamente a: comunicações; integração empresarial; interface; troca de dados. Este sistema é sustentado por um barramento distribuído, em tempo real, e uma infraestrutura de várias bases de dados que é flexível e escalável. Proporciona diferentes abordagens de implementação, tendo em conta a estratégia de expansão e investimento dos clientes.

Os módulos da aplicação são agrupados em nós de computação que podem suportar o processamento ou partilha de energia ou a reconfiguração on-line do sistema. Com isto, consegue-se fornecer elevada disponibilidade e escalabilidade. Os nós de computação podem ser: processadores *front-end*; núcleos SCADA; processadores de aplicações de energia; postos de trabalho; interfaces de utilizador; servidores web; adaptadores de aplicações externas. Estes nós podem ser implantados em vários sistemas operativos e em diferentes configurações de equipamento, incluindo virtualização, permitindo diferentes arquiteturas físicas [7].

A infraestrutura do ScateX# providencia um ambiente de distribuição unificado para aplicações que inclui: privilégios de usuário; sincronização de horário; gestão de configurações; recuperação de desastres; auto-supervisão do sistema; gestão redundante. Esta infraestrutura foi projetada com um módulo de gestão de segurança que inclui: Role-Based Access Control (RBAC);

áreas de responsabilidade; autenticação segura; registo e auditoria; segurança ao nível da rede que inclui criptografia e bloqueio de serviço ou mapeamento de porta.

Na figura 2.1 é evidenciado um exemplo da arquitetura física do ScateX#.

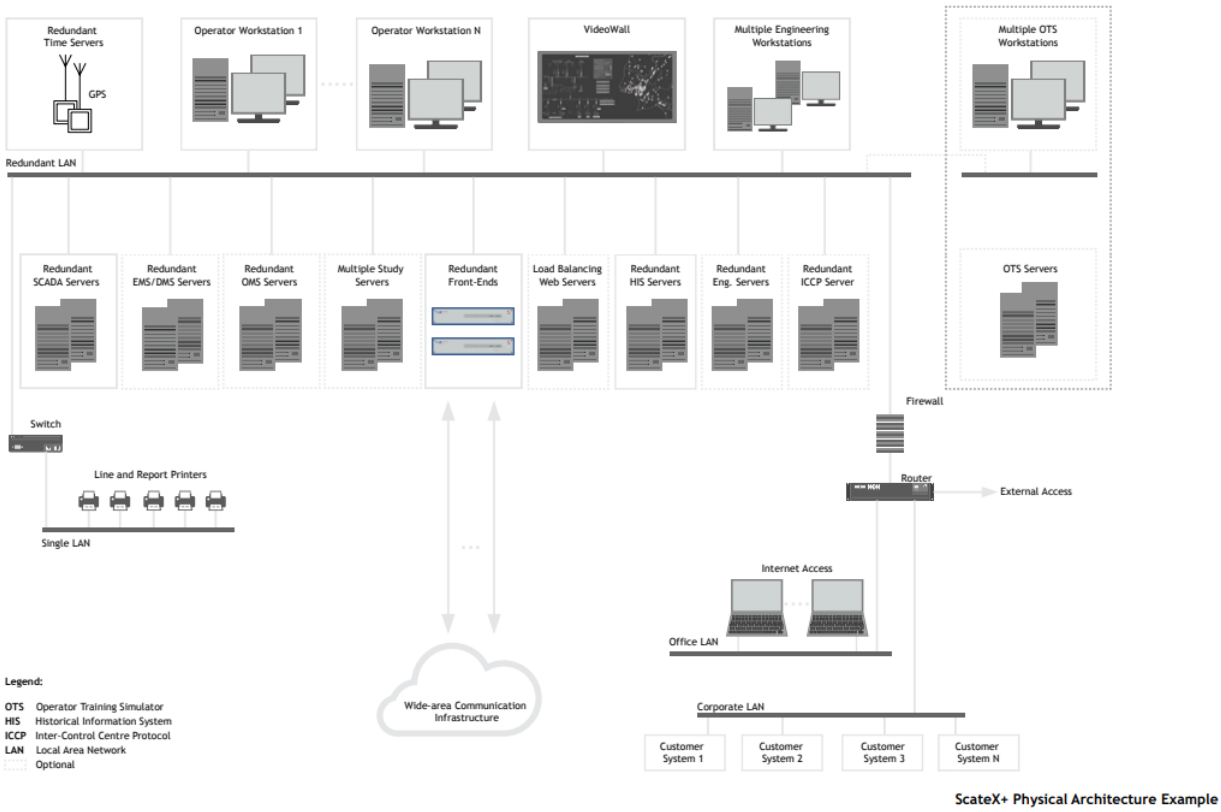



Figura 2.1: Exemplo de Arquitetura Física ScateX#

http://www.efacec.pt/en/wp-content/uploads/2016/10/CS256I1502B1.pdf

## 2.3 WebService

Um *Web Service* é um método de comunicação que tem como objetivo permitir que diferentes sistemas e aplicações consigam comunicar entre eles. Este serviço possibilita a interação de novas tecnologias com outras já existentes, permitindo que os sistemas sejam mais flexíveis e abrangentes [8]. Faz uso das funções e características do *software* e faz a sua distribuição pela rede de forma normalizada, para que todas as outras plataformas as possam entender. Como o próprio nome sugere, é uma aplicação *web* e de um modo geral, todas comunicações do *web services* têm como base o protocolo HTTP/HTTPS para proceder à transmissão de informação. Isto permite que aplicações escritas em qualquer linguagem de programação possam trocar dados através de uma rede, utilizando uma página *web*[9].

A figura 2.2 ilustra uma troca de mensagens entre um *Web Service* e um cliente. Nesta troca, observa-se que o cliente envia uma mensagem de pedido e o *Web Service* envia uma mensagem de resposta. Na maior parte dos casos, é usado JavaScript Object Notation (JSON) ou Extensible

Markup Language (XML) para a troca de informação. O JSON é um padrão aberto, compacto e mais legível para o utilizador e, como é uma linguagem leve, a sua transmissão é mais rápida. Este formato é menos loquaz que o XML e, consequentemente, mais simples. Devido a tais razões descritas, nota-se o aumento da expansão do JSON, surgindo como uma alternativa para XML.

Os *Web Service* mais usados são: o Simple Object Access Protocol (SOAP) e o Representational State Transfer (REST). Seguidamente, proceder-se-á a uma avaliação para concluir qual o *service* mais adequado para o problema tratado nesta dissertação.

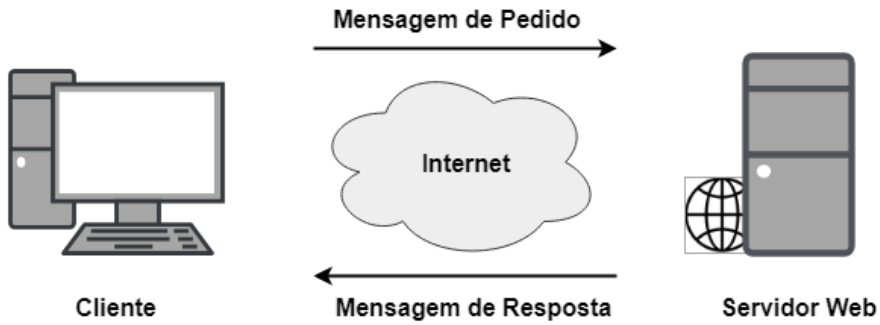



Figura 2.2: Comunicação entre um Cliente e um Servidor *Web*

- **REST** - O REST foi introduzido e desenvolvido em 2000 por Roy Fielding na sua dissertação de doutoramento. É um estilo de arquitetura, que visa o desenvolvimento de sistemas distribuídos para transmissão de dados numa interface padronizada como Hypertext Transfer Protocol (HTTP), através de um conjunto de restrições, como ter uma ligação cliente/servidor, uma interface uniforme e *stateless*(cada pedido do cliente ao servidor, tem de conter toda a informação necessária para perceção do pedido). Apesar de normalmente ser associado a HTTP estes não estão inteiramente relacionados. *RESTful* é tipicamente usado para se referir a *web services* implementados com arquitetura REST.
- **SOAP** - O SOAP é um protocolo leve para invocar procedimentos remotos através do mecanismo Remote Procedure Call (RPC) ou para efetuar troca de mensagens num ambiente descentralizado, distribuído e independente da plataforma e da linguagem de programação. Este utiliza XML para formatar as mensagens (logo, uma mensagem SOAP é um documento XML) e o HTTP para transporte, podendo ser usados outros protocolos (Simple Mail Transfer Protocol (SMTP),File Transfer Protocol (FTP)).

Tabela 2.1: Principais diferenças entre REST e SOAP

| REST | SOAP |
|---|---|
| Permite vários formatos de mensagem | Só permite o uso de XML |
| Rápido e leve devido ao uso de JSON | Mais pesado devido ao uso de XML |
| Sem especificações | Com especificações (Web Services Description Language (WSDL)) |
| Requer o uso de protocolo HTTP | É por si um protocolo |

O REST é, portanto, uma escolha conveniente para ser utilizada com os *Web Services*. Desta forma, pode-se garantir a interoperabilidade e intercomunicação entre diferentes sistemas.

# Capítulo 3

# Atualização Automática de sistemas SCADA em produção

Este capítulo trata da definição do problema, a solução proposta face ao problema e o levantamento dos requisitos propostos. Para contextualização do tema, efetuar-se-á uma explicação pormenorizada sobre o conceito de uma atualização de uma nova Versão num posto de trabalho num Sistema SCADA da EFACEC. Assim, uma atualização pode ser dividida em várias partes:

- Ligação remota à máquina de trabalho que se pretende atualizar;
- Transferência do ficheiro para a máquina;
- Interrupção dos processos;
- Instalação do ficheiro;
- *Reboot* da máquina;

Quando se pretende atualizar os postos de trabalho de um cliente, o departamento de suporte da EFACEC coordena com o cliente um escalonamento de atualizações de conjuntos de postos de trabalho. Durante os períodos acordados, os postos de trabalho não podem ser utilizados pelo cliente e um elemento da EFACEC: acede remotamente ao posto de trabalho; transfere o instalador da nova versão; executa a atualização e eventuais configurações adicionais; reinicia o posto; comunica ao cliente que o posto está de novo disponível.

## 3.1 Definição do problema

Atualmente a atualização do *ScateX#* em ambiente de produção (figura 2.1) do Sistema SCADA da EFACEC comporta custos consideráveis de mão de obra, bem como tempo de indisponibilidade (tempo de interrupção dos processos nos postos de trabalho) do Sistema incompatíveis com a eficiência exigida pelo mercado.

Considerando o seguinte cenário real de um cliente da EFACEC que possui uma rede com 105 postos de trabalho:

Numa instalação de uma nova Versão do *ScateX#*, é necessária a presença de duas pessoas para realizar os trabalhos de *Update* dos postos de trabalho. Estes operários podem efetuar a atualização de duas formas: um posto de trabalho de cada vez que implica a interrupção dos processos, instalação e *Reboot* e só após destas 3 etapas passam para o próximo posto; realizar a interrupção de várias máquinas simultaneamente, atualização encadeada seguida de *Reboot*. O tempo de indisponibilidade de um cenário de produção como o descrito pode ser ilustrado na tabela 3.1.

Tabela 3.1: Tabela de Indisponibilidade dos Postos de Trabalho

| **Nº de Postos de Trabalho** | **Tempo de Indisponibilidade por Posto de Trabalho (minutos)** |
|---|---|
| 1 | 30-40 |
| Várias ao mesmo tempo | 60 |

Como podemos verificar na tabela 3.1, o tempo de indisponibilidade da atualização dos postos de trabalho em produção não vão de encontro com a estratégia de automatização da empresa. Atualmente, encarregando dois operários com a função de efetuar a atualização de 105 postos de trabalho em simultâneo, teria-se uma indisponibilidade de 3150 minutos, cerca de 53 horas. Na realização de um *Update* de Versão de apenas um posto de trabalho, existe uma indisponibilidade de cerca de 30 a 40 minutos. O tempo de atualização do *Software* (apenas instalação e *Reboot* do sistema) seria, aproximadamente, 10 minutos. Outro pormenor que se encontra evidenciado na tabela 3.1, é que não compensa fazer o *Update* em vários postos simultaneamente. No entanto, este cenário é praticado devido ao facto de, por vezes, os postos se encontrarem organizados por grupos de trabalho e a interrupção de alguns processos implica a interrupção dos restantes postos de trabalho pertencentes ao mesmo grupo.

## 3.2 Solução proposta

Com vista a minimizar custos associados à atualização em ambiente de produção do Sistema SCADA, foi idealizado uma Plataforma de gestão de atualizações que realize, de modo automático, a instalação de uma nova Versão do *ScateX#* nos postos de trabalho do Sistema.

A Plataforma consistirá em 3 componentes principais: uma Interface *Web*, alojada num servidor no Sistema SCADA, onde será possível fazer o *upload* das atualizações, listar os postos de trabalho ligados à rede, agendar atualizações para os postos de trabalho pretendidos, realizar o controlo de Versões e configurações gerais dos postos de trabalho; uma base de dados onde serão guardados todos os dados necessários para o funcionamento da Plataforma *Web*; uma Aplicação *Windows* que correrá nos postos de trabalho com o intuito de servir como ponte de comunicação entre o posto e o servidor e que será, também, responsável pelo *Download* e instalação do *ScateX#*

nos postos de trabalho. A solução encontrada para o problema proposto pode ser ilustrada através da Arquitetura de Alto Nível representada no seguinte esquema (Fig. 3.1):

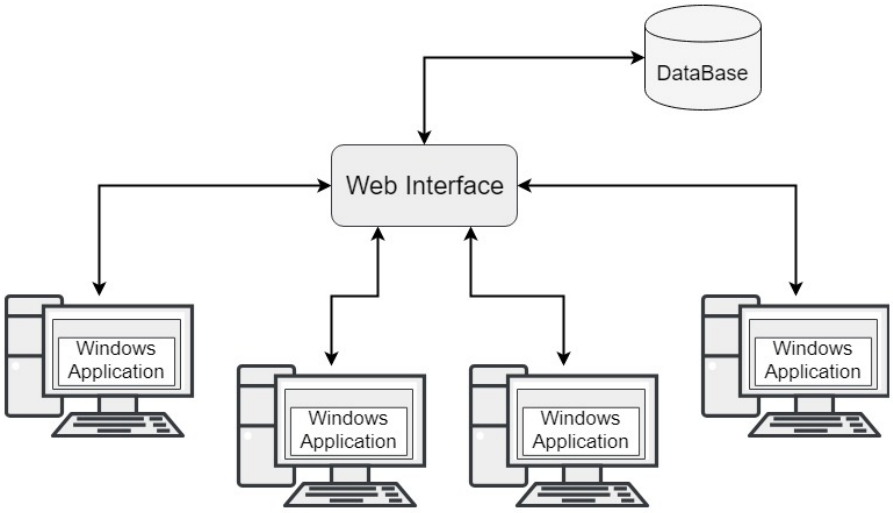



Figura 3.1: Arquitetura de Rede

O agendamento de atualizações não será a única maneira possível de atualizar um posto de trabalho. Irão ser permitidos 3 modos de configuração de um o posto de trabalho:

- *Modo Automático* - Quando uma nova Versão é introduzida na Plataforma *Web*, a Aplicação *Windows* verificará se a Versão que tem instalada é recente ou não. Caso não seja recente, irá surgir a transferência automática do ficheiro para o posto de trabalho seguido de notificação da ocorrência. Após o sucedido, o utilizador terá que decidir quando irá efetuar a instalação da nova Versão no posto;

- *Modo Remoto* - Quando existir um agendamento de atualização, o operador desse posto de trabalho será notificado sobre a instalação de uma atualização no seu posto. Será disponibilizada a opção de "adiar a instalação por um tempo definido", pretendendo precaver o cenário de existir um operador a efetuar manobras na máquina que não possam ser interrompidas. Todavia, se o operador recusar muitas vezes a instalação, está ocorrerá sem a permissão do mesmo.

- *Modo Manual* - Semelhante ao modo Automático, contudo, neste caso, o utilizador tem o poder de decidir se pretende fazer o *Download* e quando é mais oportuno instalar a nova Versão.

  As secções que se seguem irão caracterizar os 3 componentes principais da Arquitetura de Alto Nível: Base de dados, Interface *Web* e Aplicação *Windows*.

### 3.2.1 Base de Dados

A arquitetura da base de dados será definida e ilustrada na secção 4.1. Aquando da idealização da tese, foi acordado que a base de dados seria Oracle 11g | 12c, padrão do departamento *SCADA* da EFACEC. Porém, para a implementação do protótipo, foi usada uma base de dados embebida, *H2 Console*, que será abordada com mais detalhe na secção 5.1.1, referente à implementação da base de dados.

A base de dados irá conter toda a informação necessária para o funcionamento da Interface *Web*, todos os postos de trabalho que estão registados na Interface, terá uma listagem dos *Updates* que foram colocados no servidor para serem instalados nos postos de trabalho, assim como os dados relativos aos utilizadores da Interface. Para usos práticos e de implementação deste protótipo, apenas se dará uso ao perfil de Administrador.

### 3.2.2 Interface web

O desenvolvimento de uma Plataforma *Web* implica a implementação de vários componentes. Nesta secção apenas será feita a descrição das páginas que irão constituir esta Interface. Para aceder a esta Plataforma *web*, o utilizador utilizará um endereço HyperText Transfer Protocol Secure (HTTPS) que dará acesso a uma primeira Interface, de autenticação. Esta página terá: um formulário de autenticação com: *login* e *password*; uma ligação para que os operadores dos postos de trabalho possam descarregar a Aplicação *Windows*. Realizada a autenticação com sucesso, o utilizador será direcionado para a página principal (*Home Page*) da Plataforma *Web*. Em todas as Interfaces, o utilizador terá acesso a um botão de *Logout* e a um *Menu* com os seguintes separadores:

- ***Home*** - constituída por um formulário de inserção de novos *Updates* na Plataforma, como também, por uma listagem dos *Updates* que já se encontram no repositório.
  - No *Formulário de Inserção de Updates*, o utilizador terá de preencher alguns campos antes de inserir o ficheiro, nomeadamente: Revisão; Versão; Data de *Build*[1]; Pequeno texto descritivo sobre as características da nova atualização e para que tipo de máquina será feito a atualização[2].
  - Na *Lista*, o utilizador poderá visualizar todas as Versões do *Scatex#* que se encontram no repositório. A tabela de listagem terá os seguintes campos: ID; Nome; Papel; Versão; Revisão; Data *Build*; Data de *Upload* [3]; Diretório onde se encontra; uma *Flag*. A *Flag* informa se a versão está disponível ou não, algo que o utilizador pode ativar ou desativar. Quando uma Versão é introduzida, o valor *Default* da *Flag* é igual a zero, ou seja, houve introdução da Versão, mas ainda não se encontra disponível.

[1] Refere-se à data que o *Scatex+* foi compilado

[2] De momento só está previsto os postos de trabalho serem atualizáveis, mas esta plataforma poderá ser expandida para acomodar outras atualizações dos postos de trabalho

[3] Data em que o ficheiro foi introduzido na plataforma

A Plataforma terá um mecanismo que verificará quanto tempo um *Update* fica em desuso. Na hipótese de este período ultrapassar os 90 dias, o ficheiro será eliminado automaticamente do repositório. Na listagem, o utilizador tem a possibilidade de eliminar um *Update*. Tal é permitido apenas se o *Update* não estiver a ser usado em nenhum posto de trabalho. Esta tabela de listagem terá apenas as últimas 15 entradas de ficheiros no repositório para não sobrecarregar a página principal.

- ***Upload***- existem 2 opções:

  - *Upload To* - página em que o utilizador poderá agendar instalações de novas Versões do *ScateX#* nos postos de trabalho. Esta página será constituída por: uma lista de todos os ficheiros de atualizações existentes no repositório (apenas Versões disponíveis, isto é, quando o *Default* da *Flag* é igual a 1), podendo ser selecionado apenas um; uma lista de todos os postos de trabalho ligadas em rede e registados na Plataforma, não existindo limite de seleção; um formulário de seleção da data e hora que pretende que as atualizações sejam realizadas; um Módulo para controlo de tráfego de rede.

  - *List* - página que terá uma tabela de listagem semelhante à que se encontra na *Home Page*, mas sem limite de ficheiros listados.

- ***Users List*** - página de administração de utilizadores que será, exclusivamente, acedida por um utilizador com privilégios de Administrador. Este terá oportunidade de alterar a palavra passe de todos os utilizadores registados no servidor.

- ***Computer List*** - página que contém uma lista dos postos de trabalho que se registaram no servidor com os seguintes campos: ID; Nome; IP; *Role*; Modo de Atualização; Data de Registo no servidor; Estado em que se encontra no momento (*Updating*, *Download*, etc); Versão em que se encontra; Botão de configuração. Ao clicar no botão de configuração, surgirá uma nova janela com alguns campos informativos do posto de trabalho que o Administrador poderá alterar, sendo eles: Nome; IP; Role; Modo em que se encontra o posto de trabalho atualmente.

- ***Configurations*** - página em que o utilizador poderá configurar o idioma da Interface *Web* (por *Default* vem em inglês) como poderá, também, consultar uma lista com o histórico da Interface.

Na figura 3.2 é ilustrado uma mapa das páginas da Interface *Web* e as respetivas funcionalidades que foram abordadas nesta secção sobre cada página.

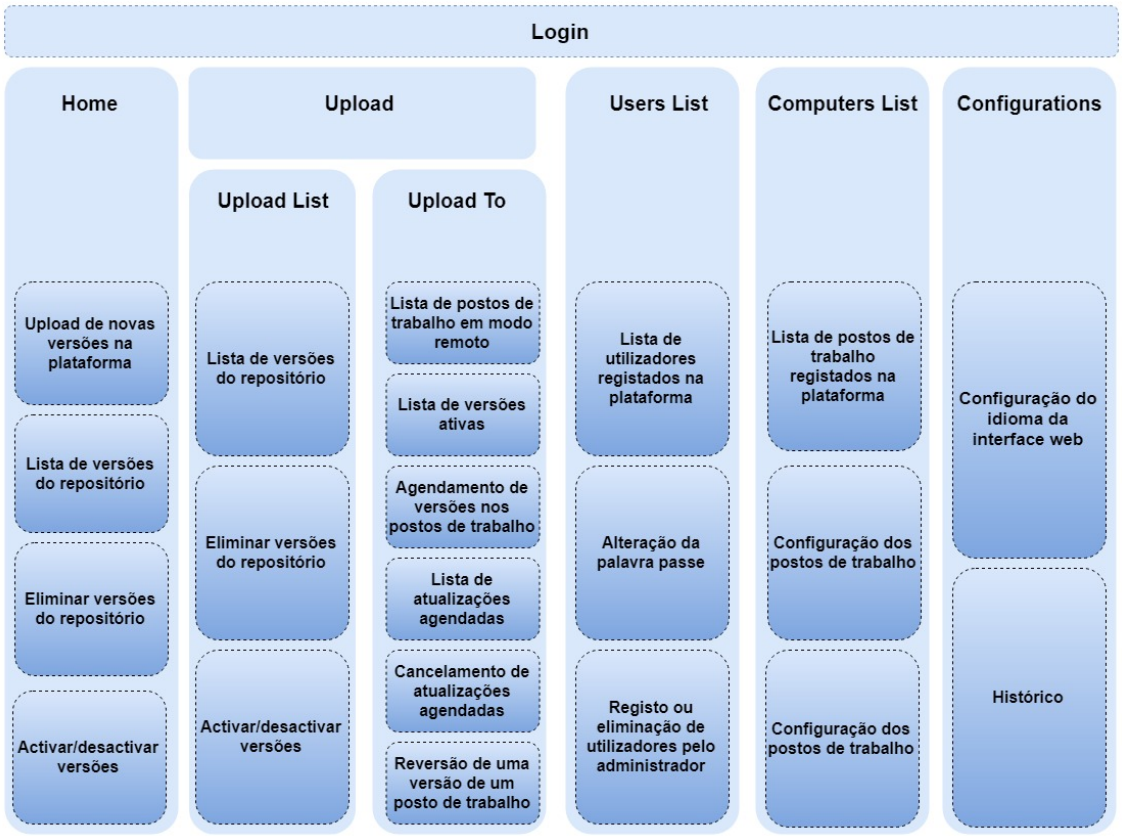



Figura 3.2: Mapa das Páginas da Interface *Web*

### 3.2.3 Aplicação *Windows*

Desenvolvimento de uma aplicação *Windows* que sistematicamente averiguará se o servidor tem atualizações disponíveis, concretizando-as caso se verifique. Será constituída por: uma janela informativa com algumas características da Máquina: IP; Modo; Nome; Versão que tem instalada do *ScateX#*; um *System Tray*[5] que permitirá ao utilizador configurar o modo de atualização do seu posto de trabalho, aceder à janela informativa e permitir a atualização do sistema. A Interface Gráfica da proposta de solução, para a Aplicação *Windows*, é ilustrada na figura 3.3.

[5]Introduzido pela primeira no *Windows 95*, situado na barra de tarefas (no canto inferior direto perto do "relógio"), e que contém alguns ícones de programas do sistema para que o utilizador possa aceder mais facilmente[10] às aplicações

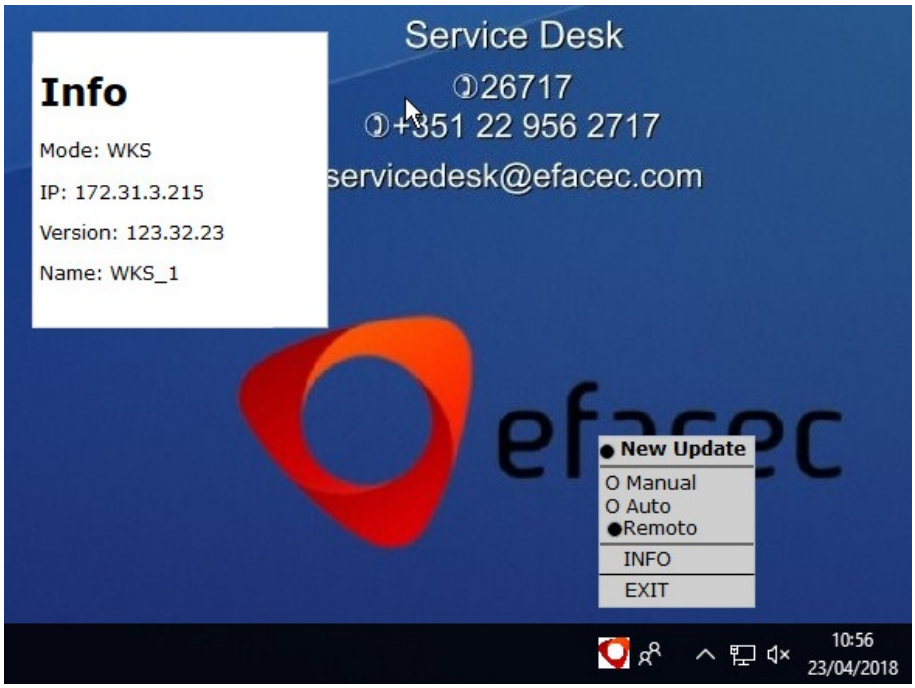



Figura 3.3: Solução proposta para a Interface da Aplicação *Windows*

Caso haja a possibilidade de efetuar atualizações, podem surgir 3 dos seguintes cenários, tendo em conta o modo em que o posto está definido:

*Primeiro cenário: Modo Automático.*

Irá surgir uma notificação no posto de trabalho que informará um possível utilizador da iniciação de uma instalação automática. Esta instalação terá início num determinado período de tempo. A não ocorrência desta atualização apenas será possível caso o utilizador o rejeitar. Deste modo, as instalações agendadas não irão interromper qualquer tipo de atividade que um utilizador possa estar a fazer nos postos de trabalho;

*Segundo cenário: Modo Remoto.*

Surgirá uma notificação a informar acerca de uma atualização agendada que se encontra disponível, dando a oportunidade ao utilizador de a iniciar ou não. Em oposição ao primeiro cenário, este apenas irá acontecer se o utilizador aceitar fazer a instalação;

*O terceiro cenário: Modo Manual.*

Aparecerá uma notificação a comunicar a disponibilidade de realizar o *Download* da atualização, podendo mais tarde um possível utilizador disputar a instalação. Este cenário é gerado como solução ao primeiro cenário, pois caso o utilizador não possa efetuar a instalação devido ao trabalho que esteja a realizar no momento, terá a oportunidade de realizar o *Download* da atualização para mais tarde instalar.

## 3.3 Requisitos

Todos os requisitos para o desenvolvimento deste projeto foram elaborados em conjunto com o Professor Mário Jorge Sousa, orientador interno, e o Engenheiro Paulo Santos, orientador externo. Estes requisitos foram sendo ajustados ao longo do estágio, no sentido de viabilizar e enriquecer o trabalho a concretizar sem colocar em causa um dos principais objetivos propostos inicialmente: "tornar o processo de atualizações mais eficiente".

Para organizar melhor a listagem dos requisitos, estes foram divididos pelos 3 componentes que perfazem o trabalho: Componente Servidor (UI), Componente Servidor (API) e Componente Cliente. De modo a facilitar a sinalização do grau de prioridade dos requisitos, usar-se-á a seguinte nomenclatura:

- **P0 -** Prioridade Máxima (essencial para o funcionamento do projeto);
- **P1 -** Prioridade Média (complementar os P0);
- **P2 -** Prioridade Baixa (facultativos);

A tabela 3.2 representa a lista de requisitos que se encontra estruturada em: Referência (Ref.), Título, Descrição e Prioridade (Pri.).

Tabela 3.2: Tabela de Requisitos

| Ref. | Título | Descrição | Pri. |
|---|---|---|---|
| UI.1 | UI *Web* | Deve disponibilizar uma Interface *web* (UI *Web*) para permitir a interação entre o utilizador e a Plataforma *web*. | P0 |
| UI.2 | Suporte Multi-idioma | O UI deve ter suporte multi-idioma (*default* = US). | P2 |
| UI.3 | *Link* direto para *Download* | Deve disponibilizar um *Link* direto para *Download* do *Setup* de instalação do Componente Cliente. | P0 |
| UI.4 | Restrição das funcionalidades do Componente Servidor | As funcionalidades do Componente Servidor devem ser acedidas apenas por utilizadores credenciados. | P1 |
| UI.5 | Módulo de gestão de Versões | Deve ser disponibilizado um Módulo de gestão de Versões. | P0 |
| UI.5A | *Upload* de uma nova Versão | Deve ter a capacidade de realizar um *Upload* de uma nova Versão. | P0 |

Tabela 3.2: Tabela de Requisitos

| Ref. | Título | Descrição | Pri. |
|---|---|---|---|
| UI.5B | *Update* | Uma nova Versão/*Update* caracteriza-se por: Binário de auto-instalação; Papel WKS \| SCD \| DMS \| SAH \|; Versão MAJOR.MINOR.PATCH; Revisão *Source-Code*; Data de *Build*; Anotações; Data de *Upload*. | P0 |
| UI.5C | Características do *Upload* | As características de uma nova Versão/*Upload* devem ser obtidas automaticamente do binário de auto-instalação. | P2 |
| UI.5D | Ativação/Desativação de uma Versão | Uma Versão pode ser ativada/desativada no momento do *Upload* (*Default*=desativada) ou em bloco após o *Upload*. | P0 |
| UI.5E | Eliminação manual de uma Versão | Deve ser possível eliminar manualmente uma Versão que não esteja atualmente em uso. | P0 |
| UI.5F | Eliminação automática de uma Versão | Deve ser possível eliminar automaticamente uma Versão que não esteja atualmente em uso. A eliminação ocorre apenas a partir de um certo período de tempo desde que deixou de ser usada (*Default*=90 dias). | P2 |
| UI.6 | Módulo de gestão de Máquinas | Deve ser disponibilizado um módulo de gestão de Máquinas. | P0 |
| UI.6A | Caracterização da Máquina | Uma Máquina caracteriza-se por: Endereço IP; Nome; Papel WKS \| SCD \| DMS \| SAH \|; Modo de Atualização: Auto, Manual, Remoto; Progresso da instalação em curso/Versão; Data de registo. | P0 |
| UI.6B | Registo de uma Máquina | As características de uma Máquina devem ser obtidas automaticamente a partir do Componente Cliente. | P0 |

Tabela 3.2: Tabela de Requisitos

| Ref. | Título | Descrição | Pri. |
|---|---|---|---|
| UI.6C | Alteração do Modo de Atualização | Deve ser possível alterar o Modo de Atualização individualmente ou em bloco. | P0 |
| UI.6D | Agendamento de uma Versão | Deve ser possível instalar uma Versão de forma individual ou em bloco e determinar a altura em que se inicia (Modo Remoto). | P0 |
| UI.6E | Manutenção do Modo Remoto | Após a conclusão do agendamento da instalação, deve-se manter o Modo Remoto. | P0 |
| UI.6F | Remoção do agendamento | Deve ser possível remover o agendamento individualmente ou em bloco. | P0 |
| UI.6G | Reversão da Versão | Deve ser possível reverter a Versão, de forma individual ou em bloco, e determinar a altura em que se efetua (Modo Remoto). | P2 |
| UI.7 | Módulo de Gestão de Utilizadores | Deve ser disponibilizado um Módulo de Gestão de Utilizadores. | P2 |
| UI.8 | Histórico | Todas as ações devem ser auditáveis (instante; quem; o quê) | P2 |
| API.1 | *Application Programming Interface* (API) REST | Deve disponibilizar uma API REST para permitir a comunicação entre a Componente Servidor e a Componente Cliente. | P0 |
| API.1A | Registo da Máquina | Deve-se registar a Máquina com as seguintes características: ID; *Name*; IP; *Role*; *Mode*; *Date_Register*; *State*; *Version*. | P0 |
| API.1B | *Download* | Função para *Download* de uma nova Versão | P0 |
| API.1C | Alteração do Modo de Atualização | Função para alternar o Modo de Atualização | P0 |

Tabela 3.2: Tabela de Requisitos

| Ref. | Título | Descrição | Pri. |
|---|---|---|---|
| API.1D | Progresso do *Upload* | Atualização do estado em que a Máquina se encontra no processo do *Upgrade* (*Downloading*; *Waiting*; *Logoff*; *Stopping*; *Upgrading*; *Stratings*; *Done*) | P0 |
| CC.1 | Instalação Manual da Aplicação *Windows* | O componente deve ser instalado manualmente após o *Download*, através do Componente Servidor (UI). | P0 |
| CC.2 | Configuração da Aplicação *Windows* | Durante a instalação do componente, deve ser introduzido o IP/Nome do Componente Servidor, onde se deverá registar. | P0 |
| CC.3 | Registo automático na Plataforma *Web* | O registo no Componente Servidor deverá incluir automaticamente todas as características da Máquina: IP \| Nome \| Papel \| Versão \| Modo. | P0 |
| CC.4 | *System Tray* | O componente deverá funcionar como Serviço de *background* | P0 |
| CC.5 | Configuração do Modo de Atualização | O utilizador deverá poder alterar, localmente, o Modo de Atualização: Auto; Manual, desde que não seja em Modo Remoto. | P0 |
| CC.6 | Modo Remoto | O Modo Remoto, deixando de estar ativo, deverá reverter para o Modo local previamente escolhido. | P0 |
| CC.7 | Modo Automático | Em Modo Manual, deverá ser apresentada notificação da existência de uma Versão mais recente, à qual o utilizador dará permissão para executar o *Upload*, desde que não haja um um *Login* ativo. | P0 |
| CC.8 | Auto atualização da Aplicação *Windows* | O componente deverá auto atualizar-se após disponibilização de uma nova Versão do Componente Servidor. | P2 |

# Capítulo 4

# Arquitetura

A análise dos requisitos propostos permitiu a construção de uma arquitetura do projeto. Esta etapa é de extrema importância para que a concretização do projeto seja possível dentro do prazo estabelecido. Este ponto para além de enriquecer o trabalho, também permite uma organização estruturada e linear, facilitando a compreensão daquilo que se desenvolveu.

Assim sendo, neste capítulo descreve-se a arquitetura dos diferentes componentes que constituem este trabalho: Base de Dados, Interface Web e Aplicação Windows, apresentados, previamente, na figura 3.1. Nesta descrição serão apresentados vários tipos de diagramas: primeiramente, surgirão as tabelas representativas da base de dados; de seguida, os fluxogramas ilustrativos da Interface *Web*; por fim, os diagramas de classes da Aplicação e diagramas de sequência de certos mecanismos da Aplicação *Windows*.

## 4.1 Base de Dados

A base de dados está responsável por armazenar as informações referentes aos diferentes componentes: formulário de *Upload* (composto por vários campos informativos sobre as Versões); dados dos utilizadores (papel que tem na Plataforma *Web*; dados relativos aos postos de trabalho. Após análise dos requisitos, definiu-se que a base de dados deste projeto iria ser formada por 4 tabelas: *User_Roles*, *Users*, *Updates* e *Computers*. A figura 4.1 representa a ilustração das mesmas:

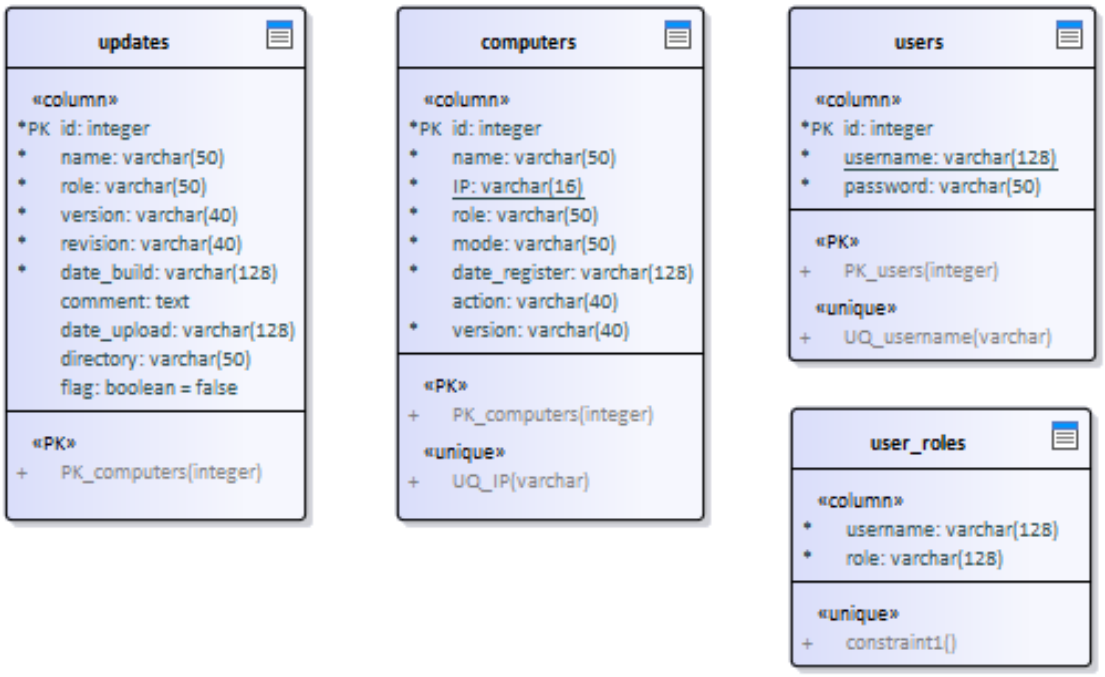



Figura 4.1: Arquitetura da Base de dados

- **User_Roles:** Esta tabela serve para definir o papel que cada entidade tem e os privilégios que esta acarreta. Neste protótipo só se irá fazer uso do *Role* "ROLE_ADMIN"que tem privilégios de Administrador. Logo, tem a capacidade de desempenhar qualquer função.
- **Users:** A tabela "*users*"lista todos os utilizadores registados no servidor. É constituída por 3 colunas:
  - **[ID]** - identificador único para cada utilizador;
  - **[Username]** - nome identificativo do utilizador na Plataforma. Este campo também é único de cada utilizador;
  - **[Password]** - é encriptada por motivos de segurança;
- **Updates:** Esta tabela contém toda a informação relevante para cada *Update*. É formada por 10 colunas:
  - **[ID]** - identificador único para cada *Update*;
  - **[Name]** - nome da Versão introduzida na Plataforma. Ao nome original será adicionado um prefixo com a data e hora de introdução da Versão na Plataforma *Web*;
  - **[Role]** - o tipo de Máquina onde a Versão será instalado. Neste protótipo só se irá atualizar postos de trabalho, mas futuramente, esta Plataforma poderá ser útil para a atualização de outras entidades e *Softwares*;
  - **[Version]**

  - **[Revision]**
  - **[Date_Build]** - data em que a Versão foi concluída;
  - **[Comment]** - o utilizador poderá adicionar algum comentário que ache relevante sobre a Versão que colocou na Plataforma;
  - **[Date_Upload]** - data de inserção na Plataforma;
  - **[Directory]** - diretoria do repositório onde se encontra a nova Versão;
  - **[Flag]** - sinaliza se o *Update* está a ser utilizado ou não;

- **Computers:** A tabela *Computers* lista os postos de trabalho, existentes na rede, que se registaram na Plataforma. É constituída por 8 colunas:

  - **[ID]** - identificador único de cada posto de trabalho;
  - **[Name]** ;
  - **[IP]** - único de cada posto de trabalho;
  - **[Role]** ;
  - **[Mode]** - *Mode* de atualização em que se encontra;
  - **[Date_Register]** - data em que se registou na Plataforma;
  - **[Action]** - pode estar a fazer *Download*; instalação de um *Update*; o estado em que se encontra: a fazer *Download*, instalação de um *Update*, em *Stand-by* (à espera de novas versões).
  - **[Version]** - Versão que está instalada na Máquina;

## 4.2 Interface Web

Nesta secção irá ser apresentada a navegação da Interface *web*. Os principais diagramas de classes referentes à Interface *web* (*Computer, Users, Files e Login*) encontram-se anexados neste documento e podem ser consultados no Anexo A.1. Após a análise dos requisitos decidiu-se proceder à implementação do fluxograma da figura 4.2. Deste modo, foi possível melhorar a perceção do fluxo da Interface *web*.

Um utilizador só terá acesso à página principal se estiver autenticado. Na página principal (*Home*), poderá introduzir novas Versões de *Software*, listar todas as Versões existentes ou eliminar uma Versão. Na página de *Computers List*, o utilizador poderá ver a lista de todos os postos de trabalho, assim como configurar alguns dados sobre os postos. Para além disso, terá, também, acesso a mais duas páginas: *Upload to*, onde procederá ao agendamento de atualizações nos postos de trabalho; e *Upload List*, onde encontrará uma lista completa de todas as Versões existentes no repositório como poderá, também, eliminar as Versões que não estejam instaladas em nenhum posto de trabalho.

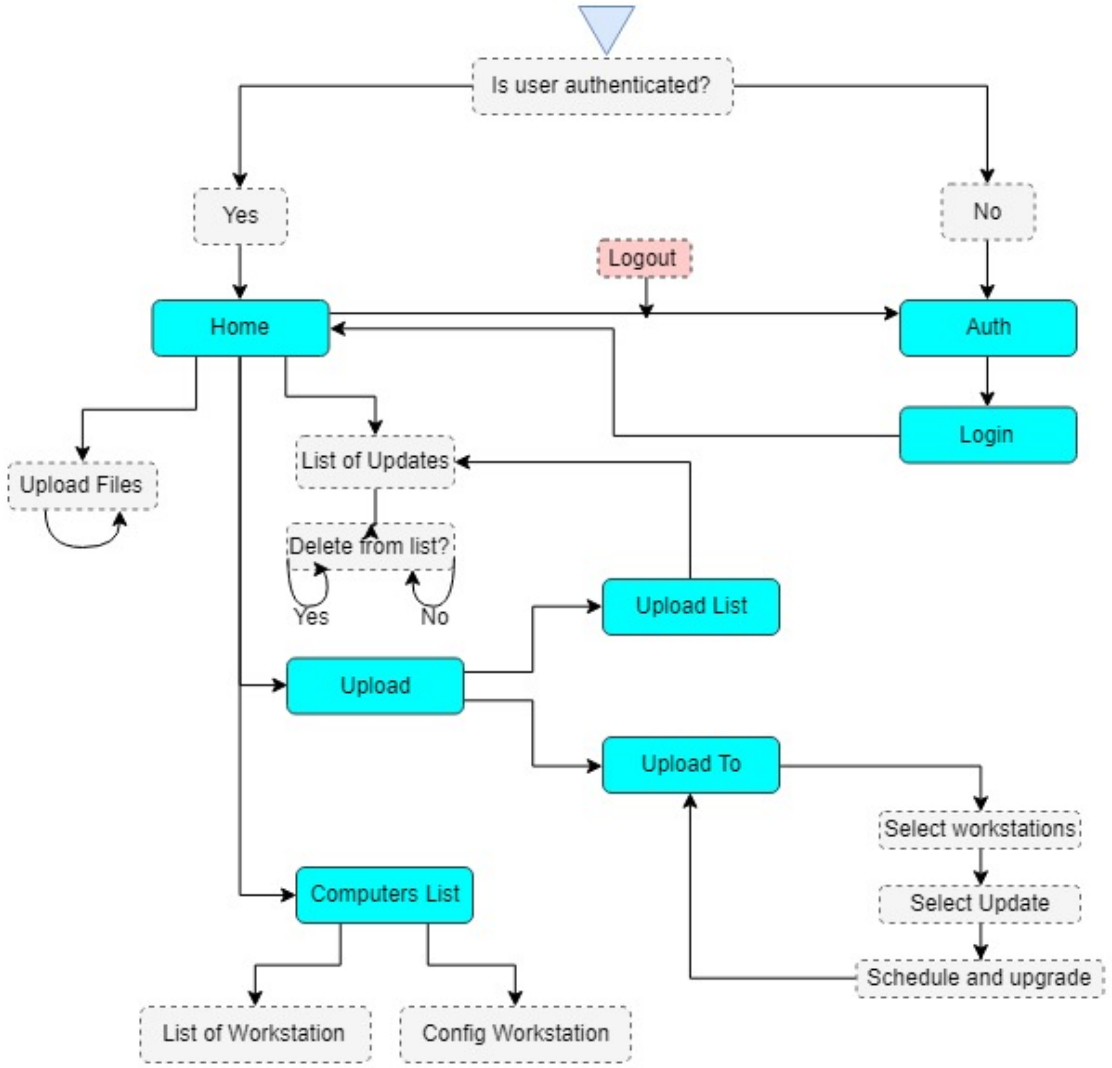



Figura 4.2: Fluxograma da Interface Web

## 4.3 Aplicação Windows

Um ponto essencial para este projeto é a leveza e simplicidade da Aplicação *Windows*. Pretende-se que passe despercebida nos postos de trabalho para não interferir com o trabalho dos operadores. Portanto, foi desenvolvida com o mínimo de complicações de código possível.

A figura 4.3 ilustra a classe de diagramas da aplicação. Toda a Aplicação gira à volta da classe *Form1*, onde são geradas as Interfaces Gráficas finais do *Info Window* (Janela informativa) e o *System Tray*. Para auxiliar este processo, foi criado um Controlador (*SystemController*) que gere e trata toda a informação trocada com o *Web Service*. Esta informação passa primeiro pela classe *SystemApi*, responsável pela comunicação com a REST API. Outro aspeto importante da Aplicação *Windows*, é a função *TImer()* implementada no controlador. Esta função é responsável por

averiguar, periodicamente, se existem novas Versões disponíveis se existem alterações de dados no servidor ou nos postos.

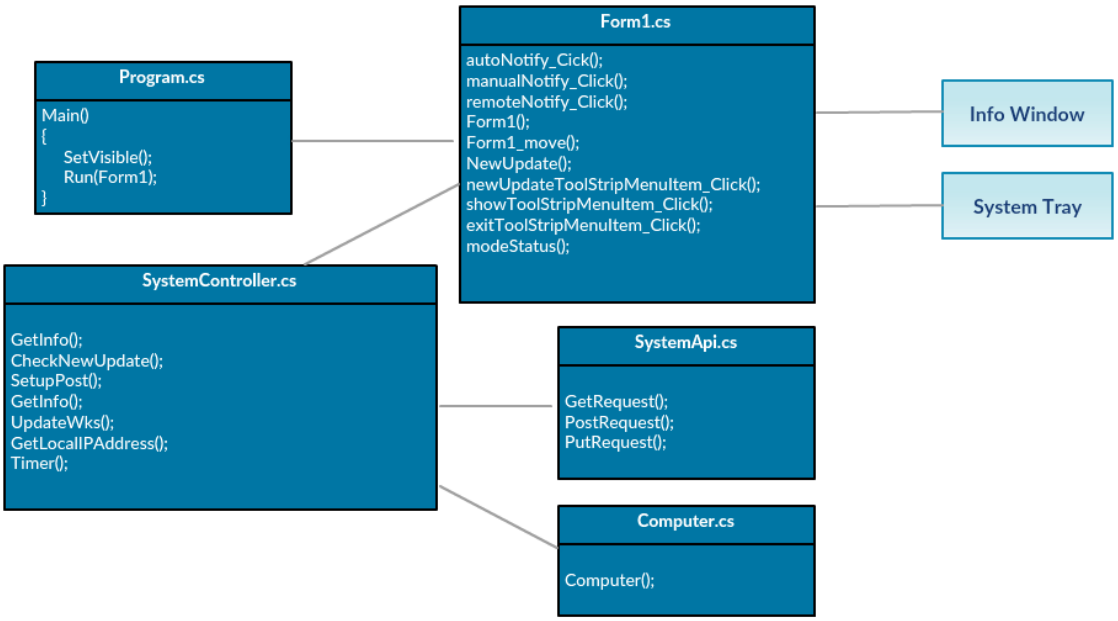



Figura 4.3: Diagrama de classes da Aplicação Windows

O diagrama de sequência da figura 4.4 demonstra o registo do posto de trabalho no servidor. Um dos requisitos da Aplicação *Windows* consta no seu registo automático na Plataforma *Web*.

Após a instalação, a aplicação envia os dados sobre o posto de trabalho para a Plataforma *Web*. A Plataforma valida os dados e insere-os na base de dados, retornando os dados completos, com o ID atribuído, ao posto de trabalho.

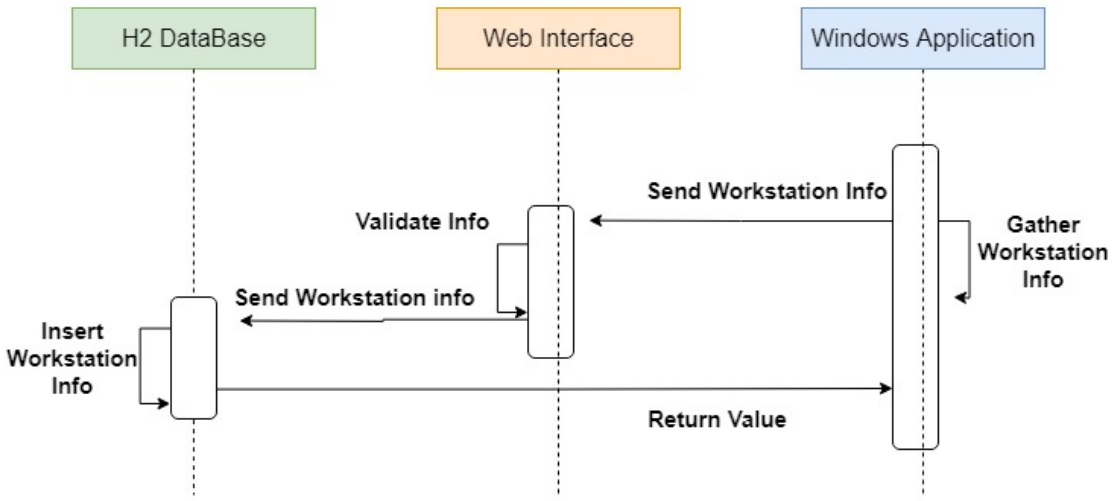



Figura 4.4: Diagrama de sequência de registo da Aplicação *Windows*

# Capítulo 5

# Implementação e Demonstração

Este capítulo será dividido em duas secções: implementação e demonstração. Em cada uma serão detalhados os principais componentes deste projeto (Base de dados, *Interface Web* e Aplicação *Windows*). Na primeira secção, implementação, serão abordados os métodos de desenvolvimento de cada componente como também as tecnologias usadas no projeto. Na segunda secção, demonstração, serão exibidos os diferentes componentes em uso, num cenário de desenvolvimento criado para o projeto.

## 5.1 Implementação

### 5.1.1 Base de Dados

Neste trabalho teve-se a preocupação inicial de isolar o projeto das restantes componentes do Sistema SCADA, isto deve-se ao facto do SCADA ser um sistema complexo e quanto mais alienado e objetivo for este trabalho, mais benefícios trará à empresa. Por estes motivos e querendo ter o menor impacto no Sistema SCADA, foi decidido usar uma base de dados embebida no projeto: H2. Esta base de dados é bastante leve e como a quantidade de dados a reter não é elevada, não se justificava a implementação do programa na base de dados do *ScateX#*. Nesta base de dados foram inseridos todo o tipo de dados relacionados com o projeto como: lista de *Updates*, lista de computadores em rede e utilizadores registados.

Para fazer a conexão do *Software* ao *Driver* da base de dados H2 usou-se uma conexão Java Database Connectivity (JDBC). A configuração Java da ligação à base de dados pode ser vista no seguinte bloco de código em 5.1:

Listing 5.1: Conexão à base de dados

```
public DataSource getDataSource() {
    WebConfig.LOGGER.debug("Starting DataSource");
    DriverManagerDataSource ds = new DriverManagerDataSource();
    ds.setDriverClassName("org.h2.Driver");
    ds.setUrl("jdbc:h2:C:\\workspace\\eclipse-workspace\\WebAppMVC\\
        target/db");
```

```
ds.setUsername("root");
ds.setPassword("******");
return ds;
}
```

Uma das grandes vantagens da base dados H2, é que pode ser acedida através de uma Interface *Web*, como é ilustrado na figura 5.1. Esta simplicidade na manipulação da base de dados, favoreceu o desenvolvimento deste trabalho pois permitia com facilmente alterar qualquer dado necessário, ou verificar se a inserção de dados, pela plataforma, estava correto.

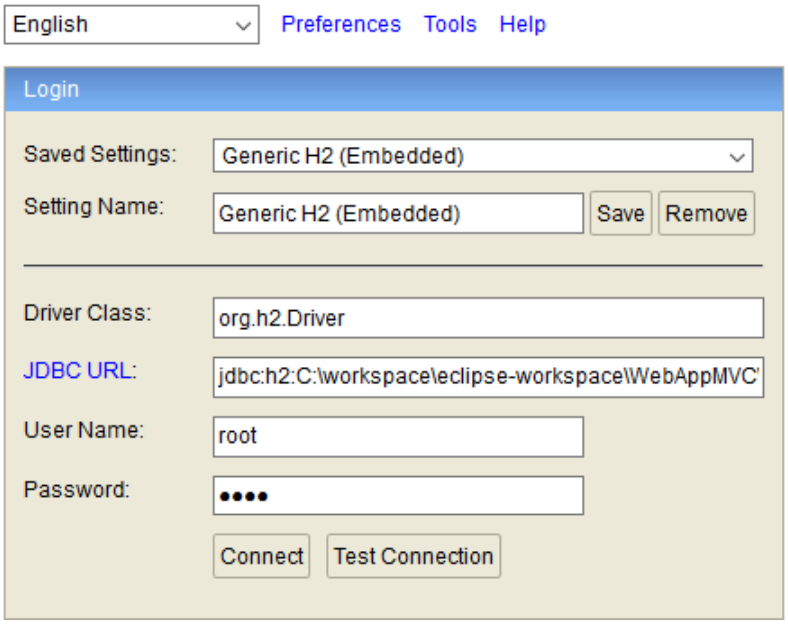



Figura 5.1: Login Interface Web da Base dados H2

Para utilizar esta base de dados, era necessário autenticar, preenchendo um formulário onde eram inseridas as credenciais da base de dados e o endereço JDBC que foi utilizado na conexão que é ilustrada no bloco de código 5.1. Após esta etapa, a Plataforma permitia alterar facilmente a base de dados, desde a criação de tabelas à inserção e alteração de dados.

### 5.1.2 Interface Web

A Plataforma *web* deve ser robusta para poder incorporar elementos como a API REST e a transferência de ficheiros, como também ser de fácil acesso e uso para os utilizadores, através de uma Interface *Web*.

Foi feita uma pesquisa sobre as melhores opções para satisfazer as necessidades do *Client* e do *Server*. Relativamente ao *Client*, foi decidido usar HyperText Markup Language (HTML) por causa do pouco consumo de recursos, compatibilidade e longa duração do *Software*. Para a construção das Interfaces propriamente ditas, foi usado HTML, Cascading Style Sheets (CSS), XML e JavaScript como linguagens de programação. O resto do programa foi desenvolvido em Java.

Relativamente ao *Server*, este deve ser o mais leve e que consuma o menor número de recursos possível para melhorar o desempenho da Aplicação. As tecnologias em Java usadas para este fim foram o JDBC e o Java Enterprise Edition (JEE). Uma das utilizações do JEE foi o uso de *Servlet* para o *Download* do instalador da Aplicação *Windows* para os postos de trabalho.

Utilizou-se uma *Framework* para a implementação do código no lado do servidor: *Spring Framework*. Esta *Framework* foi escolhida, principalmente, por facilitar o uso de um modelo *Model View Controller (MVC)* que ajuda na organização do código e na sua robustez. Para desenvolver este *Software* foi escolhido o Eclipse que irá ser detalhado com mais pormenor na secção 5.1.4, como Plataforma de desenvolvimento. Decidiu-se prosseguir com a seguinte estrutura: as Interfaces estão em */src/main/webapp/WEB-INF/jsp/*; em */src/main/resources/* colocou-se alguns ficheiros de configuração do *Log*, dos validadores e da *Spring Framework*; o código do *Software* encontra-se em */src/main/java/webapp/* e foi dividido em 8 componentes:

- *Config* - configurações de inicialização do *Software*, conexão da base de dados e questões relacionadas com a segurança;
- *Controller* - como o próprio nome indica, trata dos controladores do modelo MVC. Foram implementados controladores para as principais atividades do projeto: *ComputerController*, *RESTController*, *LoginController*, *UploadController* e *UserController*;
- *Dao* - Onde foram implementadas as *Queries* da base de dados;
- *Form* - Criação dos diferentes formulários: *UploadForm e UserForm*;
- *Model* - Onde foram implementados os construtores de Java para as principais classes: *ComputerInfo, FileUpload, UserInfo*;
- *Service* - Tratamento dos serviços que serviam de ligação entre a base de dados e os controladores;
- *Servlet* - Desenvolvimento de um *Servlet* para o *Download* da Aplicação *Windows* para os postos de trabalho;
- *Validator* - Foram desenvolvidos os diferentes validadores para os vários campos dos formulários para garantir que estes sejam introduzidos nos corretos formatos;

#### 5.1.2.1 REST API

Um dos principais componentes da projecto, é o *Web Service*. Depois de alguma pesquisa, decidiu-se usar uma arquitetura REST. O facto de ser usada uma *Spring Framework* facilitou o desenvolvimento deste deste serviço, a *Framework* disponibiliza várias bibliotecas que ajudaram a implementar o serviço *RESTful* [11]:

Listing 5.2: Bibliotecas Spring usados no desenvolvimento do Web Service

```
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;
```

A implementação das funções da API seguem uma estrutura simples e constante. No bloco de código 5.3, é ilustrado uma função, onde foi utilizado o método *PUT* para atualizar os dados alterados pelo postos de trabalho, na Plataforma *Web*. É estabelecido o formato da mensagem de pedido e da mensagem de resposta, ambos JSON. Para identificar o posto que será atualizado é enviado no endereço HTTP o ID do posto. Depois de atualizado, retorna os dados completos de volta para o posto.

Listing 5.3: Função da API que atualiza os dados alterados nos postos de trabalho

```
@PutMapping(value={"/updateComputer/{id}"}, consumes={"application/json"
    , "application/json"}, produces={"application/json", "application/
    json"})
@ResponseBody
public ComputerInfo updateComputer(@PathVariable("id") int id,
    @RequestBody ComputerInfo computer)
{
  LOGGER.debug("updateComputer()");

  computerService.updateById(id, computer.getName(), computer.getIp(),
      computer.getRole(), computer.getMode());
  return computer;
}
```

Na figura 5.2 estão representadas as classes envolvidas no funcionamento da API, bem como todas as funções que foram implementadas nesse serviço. Sendo este um *Web Service* que serve para estabelecer a comunicação e troca de dados entre o servidor e a Aplicação *Windows*, serão apenas manipulados dados referentes aos postos de trabalho (*ComputerService*). No entanto, para averiguar qual é a versão de *Software* instalada no posto e se existem novas Versões disponíveis, é necessário consultar a lista de *Updates* da base dados, sendo por essa razão que se usa a Interface *FileService*.

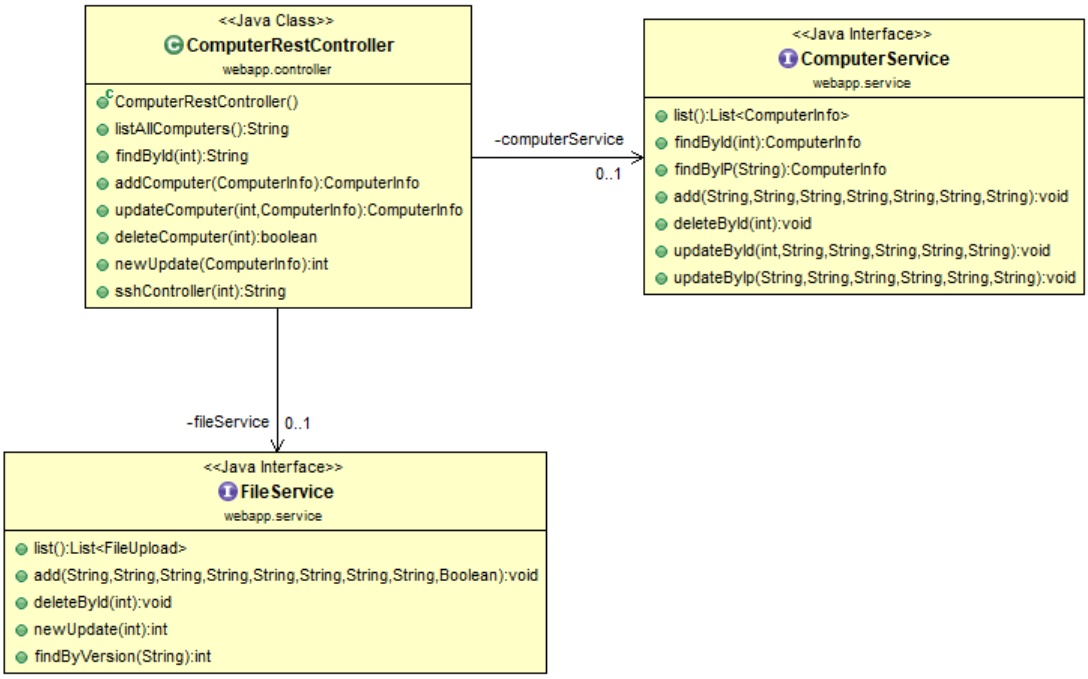



Figura 5.2: Diagrama de classes do Web Services

#### 5.1.2.2 Servlet

Outro elemento que foi desenvolvido, foi o de permitir os operadores dos postos de trabalho de descarregarem Aplicação *Windows* a partir da página de autenticação da Interface *Web*. Portanto, procedeu-se à implementação de um *Servlet* que permitisse o *Download* do ficheiro de instalação da Aplicação *Windows* para os postos de trabalho. Para fins de teste e simplificação do processo de validação, foi usado um ficheiro no formato *.txt*, para descarregar no posto de trabalho. A função responsável pelo *Download* encontra-se ilustrada no bloco de código 5.4.

Listing 5.4: Função que trata da implementação do servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
     PrintWriter out = response.getWriter();
     String filename = "windows_service.txt";
     String filePath = "C:\\workspace\\";
     response.setContentType("APPLICATION/OCTET-STREAM");
     response.setHeader("Content-Disposition", "attachment;filename=\""
         +filename+"\"");
     FileInputStream fi = new FileInputStream(filePath+filename);
     int i;
     while((i=fi.read()) != -1)
         out.write(i);
     out.close();
     fi.close(); }
```

### 5.1.3 Aplicação Windows

A aplicação Windows deve ser o mais leve possível, pois pretende-se que tenha o menor impacto no sistema dos postos de trabalho. Tem que ter um instalador simples, que os utilizadores possam descarregar da Plataforma *Web*, instalem nos postos de trabalho, configurando apenas alguns dados sobre o posto.

Como se trata de uma Aplicação em *Windows*, decidiu-se desenvolver usando a linguagem de programação, C#, realizando a implementação no *Visual Studio 2017*. Esta escolha recai sobre o facto do C# ser uma linguagem de programação criada pela *Microsoft* e o *Software* disponibilizar vários *Templates* que ajudam no desenvolvimento de Aplicações para sistemas operativos *Windows*.

A aplicação *Windows* pode ser dividida em diferentes componentes, para facilitar a sua implementação:

- *Setup Wizard* - Instalador de uma Aplicação *Windows* que, para além de instalar a aplicação nos postos de trabalho, pedirá ao operador para introduzir alguns dados iniciais sobre o posto.
- *Form* - Janela informativa com informação relevante sobre o posto de trabalho.
- *System Tray* - Este componente ficará situado ao lado do relógio, na barra de ferramentas do *Windows*, e ficará disponível sempre que a janela informativa for minimizada. Permitirá que o operador possa alterar o modo de atualização do posto de trabalho, voltar a abrir a janela informativa e encerrar a aplicação.
- *OpenSsh* - É um programa existente em todas os postos de trabalho que tenham o sistema operativo *Windows10*, e que permitirá fazer uma ligação Secure Shell (SSH) entre o servidor e Aplicação *Windows* para transferir ficheiros através de Secure Copy Protocol (SCP);
- *System API* - Este componente será responsável pela comunicação e troca de dados entre a aplicação e o *Web Service*.

### 5.1.4 Tecnologias Usadas

Nesta secção será feito um levantamento e caracterização das tecnologias usadas na implementação e validação do projeto.

Referenciando a figura 5.4 do cenário de desenvolvimento, serão enumeradas as tecnologias usadas pelos componentes da figura. Foi utilizado o *Eclipse* como ambiente de desenvolvimento da Interface *Web*, do *Web Service Rest*, da integração com a base de dados H2 e da comunicação com o servidor *TomCat*. No *Eclipse* foi usado: *Spring Framework*, *Maven* ; *Log4J*.

O *Microsoft Visual Studio* foi o Integrated Development Environment (IDE) usado para o desenvolvimento da Aplicação *Windows*.

- ***Eclipse***

É um ambiente de IDE desenvolvido pela Eclipse Foundation. É uma ferramenta *Open Source* e gratuita, sem custos para quem a utiliza. É suportado em *Windows, Linux and macOS systems*. É um sistema altamente customizável com características como *Debbuging*, *Git Control*, realce de sintaxe, *Pluggins* e *Frameworks* como *Spring*. É a plataforma ideal para programação em Java, pois tem todas a ferramentas essenciais para o desenvolvimento, incluindo: Java IDE, editor XML, integração *Maven*, Apache Server [12].

- ***Microsoft Visual Studio***

  *Microsoft Visual Studio* é um ambiente de IDE criado pela *Microsoft*. O editor foi desenvolvido especialmente para linguagens de programação como .NET, C#, Visual Basic, C++; áreas de desenvolvimento web como ASP.NET. É o IDE aconselhado para desenvolver Aplicações *Windows*. Esta Plataforma permite que o utilizador adicione vários *Plugins* úteis, podendo customizar da melhor forma para o programador. No caso deste projeto, foi ideal para o desenvolvimento do *System Tray* e futuramente para a criação *Setup Wizard* (instalação e configuração da Aplicação *Windows* nos postos de trabalho), pois existem vários *Templates* da Plataforma que facilitam o desenvolvimento.

  Para a concretização deste projeto, foi usado a versão *Microsoft Visual Studio 2017 Comunnity*, gratuita para alunos da Universidade do Porto.

- ***H2 Console***

  *H2 Database* é o sistema de gestão de base de dados relacional escrita em Java que pode ser embebida em Aplicações Java e correr em modo *Client-Server*. Tem a vantagem de ser rápida, *Open Source*, pode ser encontrada no *Eclipse* e suporta JDBC API. É uma base dados que pode ser gerida através de uma interface web e que tem uma pequena imputação no projeto de cerca de 1.5MB no ficheiro jar [13].

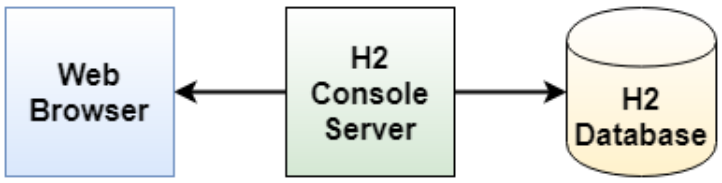



Figura 5.3: Estrutura de uma cenário com H2 Database

Como se trata de uma aplicação do *Cliente/Servidor*, são requeridos um servidor e um cliente (*Browser*) para correr a aplicação, como é ilustrado na figura 5.3.

- ***Postman***

  Postman é uma aplicação gratuita para facilitar o desenvolvimento de API, podendo ser usado em várias plataformas como *Windows, MacOS, Chrome, Linux*. A Plataforma tem várias opções e características que permite ao utilizador simular qualquer requerimento ou

cenário. Neste projeto foi bastante útil para testar e desenvolver a REST API criada em Java no *Eclipse*. À medida que se iam criando funções na API iam sendo testadas com o Postman para averiguar o *HTTP Reponse* e *HTTP Request* bem como a estrutura JSON do conteúdo que era transmitido [14].

- ***Maven***

  O *Maven* é uma ferramenta de gestão e compreensão de projetos de *Software* que tem como base de conceito de um modelo de objeto de projeto (POM). O *Maven* aborda dois aspetos do desenvolvimento de *Software*: a descrição da construção do *Software* e a descrição das suas dependências. Este possui um arquivo XML que descreve: o projeto de Software que é construído; as suas dependências de outros módulos e componentes externos; a ordem de construção; diretórios e *Plugins* essenciais. O ficheiro, *pom.xml*, deste projeto está anexado em A.

  Esta tecnologia é muito utilizada em projetos Java. O *Maven* faz o *Download* dinâmico de bibliotecas Java e *Plug-ins Maven* de um ou mais repositórios (como o Repositório Central Maven 2) e armazena-os numa *Cache* local. Essa *Cache* local de artefactos transferidos por *Download* também pode ser atualizado com artefactos criados por projetos locais.

  Neste projeto, foi utilizado o Maven para Eclipse (M2E) para fazer a gestão das dependências de bibliotecas. O M2E faz uso do *Maven* para efetuar esta gestão. Este tipo de ferramentas é importante para grandes projetos que utilizam várias bibliotecas, porque se responsabiliza pela biblioteca de dependências do projeto [15].

  Na secção A.2 foi anexado o *pom.xml*, usado neste projeto.

- ***Log4j***

  Para um projeto desta dimensão é uma boa prática usar

  Logs para ajudar o programador a desenvolver o *Software*, como também para identificar mais facilmente falhas no programa. Foi decidido usar um *Logging* baseado em Java, o *Log4j*. O *Log4j* é uma biblioteca de Java especializada em *Logging* e foi desenvolvido pela *Apache Software Foundation*. Uma das características do *Log4j* é disponibilizar vários níveis de *Log* que o programador pode configurar, consoante a necessidade. Este permite que o programador crie os seus próprios níveis de *Log* [16].

  O bloco de código 5.5 é o conteúdo do ficheiro de configuração do *Log4j* usado neste projeto.

Listing 5.5: Conteúdo do ficheiro de configuração do Log4j

```
log4j.rootCategory=warn, R
log4j.logger.webapp=debug, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{HH:mm:ss.SSS} [%t] %
    p %F:%L - %m%n
```

```
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=logs/app.log
log4j.appender.R.MaxFileSize=10MB
log4j.appender.R.MaxBackupIndex=10
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{yyyy-MMM-dd HH:mm:ss.SSS}
    [%t] %p %F:%L - %m%n
```

## 5.2 Demonstração

Inicialmente pretendia-se criar um cenário de testes onde fosse possível validar e demonstrar todos os requisitos alcançados. Esse cenário inclui uma simulação de um Sistema SCADA com algumas Máquinas virtuais ligadas em rede com a aplicação *Windows* instalada, de modo a simular da melhor maneira possível um cenário real de produção. Infelizmente, não foi possível desenvolver e implementar o projeto neste tipo de cenário. Contudo, foi possível demonstrar e validar alguns requisitos, com um cenário que é ilustrado na figura 5.4.

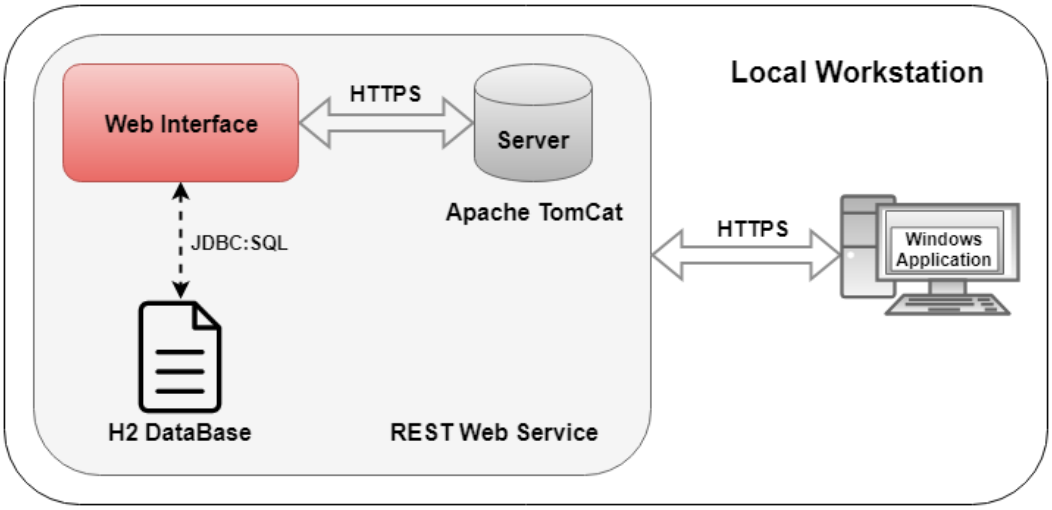



Figura 5.4: Cenário de Desenvolvimento

Como pode ser aferido pela figura 5.4, todo o trabalho foi demonstrado e validado no computador onde este foi desenvolvido. O projeto foi alocado num servidor local, *Apache Tomcat*, acedido através do endereço `https://localhost:8080/WebAppMVC/`. Os dados foram guardados na base de dados embebida, H2, como estava previsto. A aplicação *Windows*, instalada no computador local (com sistema operativo *Windows10*), comunica com a Interface *web* através de uma REST API.

Para a demonstração e validação do projeto, foram criadas, no computador, duas pastas: *Upload Files* e *Download Files*. Para validar o *Upload* de novas Versões para a Plataforma *Web*,

quando introduzido um ficheiro através da Interface *Web*, este iria para a pasta *Upload Files*, alterando automaticamente o nome do ficheiro: com a adição de um prefixo com a data e hora que foi introduzido, respeitando a seguinte nomenclatura *"AA.MM.DD.hh.mm.ss_"*. Quando efetuada uma atualização de um posto de trabalho, esta implicaria a transferência do ficheiro para posto de trabalho por SSH, em termos práticos, a deslocação do ficheiro da pasta *Upload Files* para a pasta *Download Files*. As próximas secções vão detalhar melhor os diferentes componentes, como também, a validação de algumas funcionalidades.

### 5.2.1 Base de Dados

O registo de utilizadores da Plataforma *Web* através da Interface *Web* não era um requisito do projeto, por isso, para registar novos utilizadores na plataforma, era necessário introduzi-los diretamente na base de dados. Como já foi referido em secções anteriores, a base de dados (H2) pode ser acedida através de uma Interface *Web*, o que facilita a manipulação de dados. Com isto, todos os utilizadores, usados no cenário de desenvolvimento, foram introduzidos manualmente na base de dados. Na figura 5.5, pode ser observada uma seleção de todos os dados da tabela *USERS*, apenas com um único utilizador: administrador. Este foi o único utilizador usado durante a demonstração e validação do projeto.

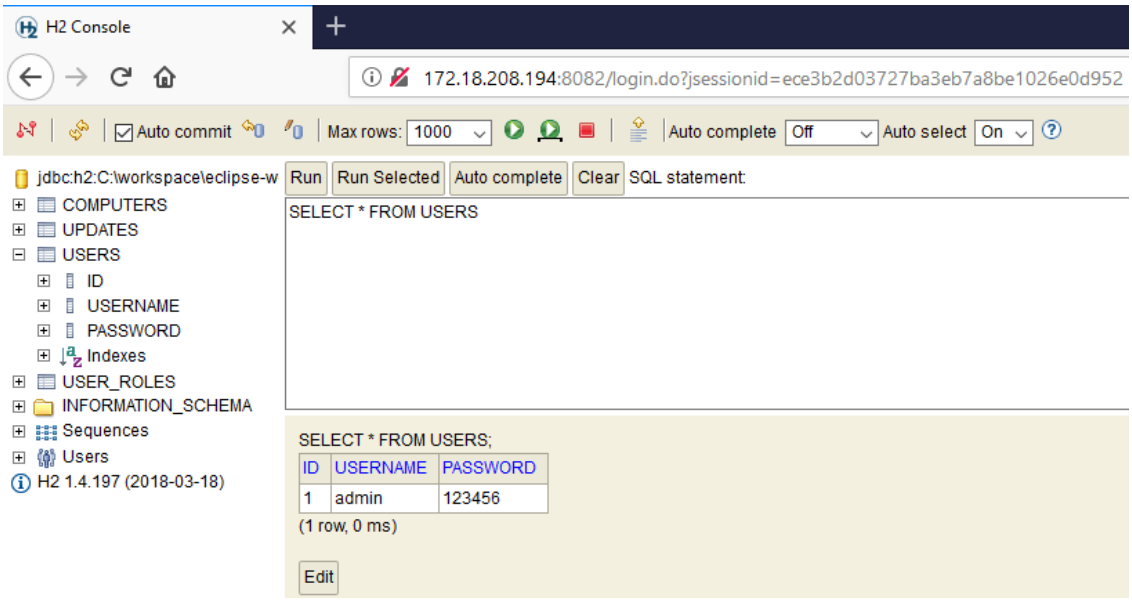



Figura 5.5: Tabela "users"da base de dados

### 5.2.2 Interface Web

Nesta secção, será feita uma demonstração do funcionamento da Interface *Web*.

Como o *Design* das Interfaces Gráficas não constituía um dos requisitos que a empresa estabeleceu inicialmente, o aspeto destas é robusto e objetivo. Desta maneira, manteve-se um formato mais simples e limpo, contribuindo para a facilidade de utilização das mesmas.

- **Autenticação**

  A autenticação é tratada no lado do cliente para um utilizador se registar na Plataforma. Como já foi referido, é necessário a inserção manual na base de dados. Como se trata de uma Plataforma que apenas será gerida por Administradores, o registo de novos utilizadores não foi um dos requisitos do projeto.

  Com o utilizador já registado, ao surgir o formulário de *Login* (figura 5.6) na página inicial da Interface, este introduz as respetivas credenciais e poderá, assim, entrar na Plataforma. Para além disso, na página do formulário de *Login*, pode ser observada a opção de *Download* que serve para os operadores dos postos de trabalho descarregarem a Aplicação *Windows*.

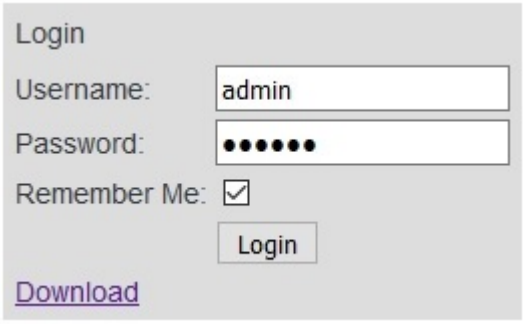



Figura 5.6: Formulário Login

- **Menu**

  O Menu é comum a todas as páginas da Interface *Web*, exceto a página de *Login*. Como no resto das Interfaces, este foi desenvolvido para ser simples e eficaz. É constituído por: Nome da Interface "*Updates Management Module*"; 4 separadores que redirecionam para as diferentes páginas: *Home*, *Upload* (tem 2 opções: *Upload List* e *Upload To*); *Users List*; *Computer List*; um botão de *Logout* (para que o utilizador possa sair da Interface).

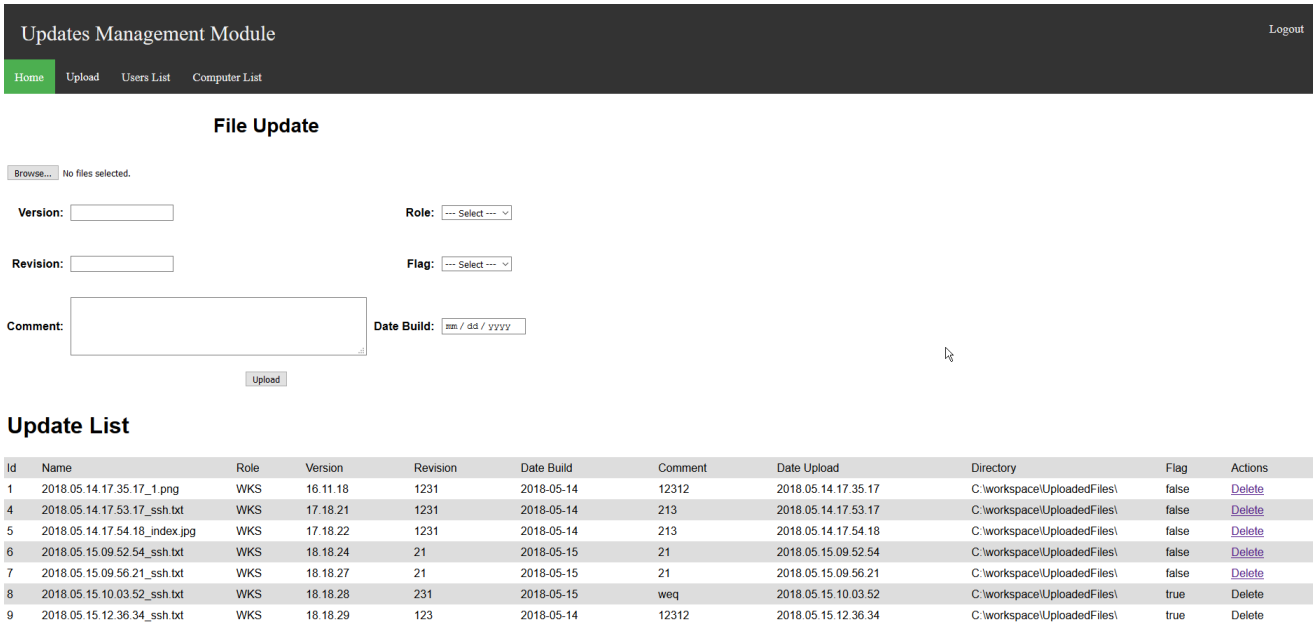
- ***Home Page***

  Após a autenticação o utilizador é redirecionado para esta página (figura 5.7, que se pode considerar como página principal da plataforma]. Nesta Interface é apresentado, ao utilizador, um formulário onde pode inserir novas Versões de *Updates* no servidor e uma lista de Versões existentes no repositório.

  Para validar as funcionalidades desta página, foram inseridos vários ficheiros na Plataforma. Não foi implementado nenhum validador para o formato que é inserido no formulário, tal pode ser verificado no formato dos ficheiros inseridos na lista. A falta de validador deve-se ao facto de a empresa pretender futuramente expandir a Plataforma para a instalação de outros tipos de *Software*.

Como pode ser verificado pela figura 5.7, foi adicionado, automaticamente, o prefixo com a data de inserção a todos os ficheiros que foram inseridos. Este prefixo serve para prevenir a inserção de ficheiros com o mesmo nome.

A lista de *Updates* permite ao utilizador selecionar e eliminar as Versões que deseja. No entanto, só pode proceder à eliminação de Versões se estas tiverem a *Flag* no estado *False* (versão em desuso). Esta *Flag* assinala se a Versão está instalada nalgum posto de trabalho ou não. Não é permitido eliminar ficheiros do repositório que estejam em uso.




| Id | Name | Role | Version | Revision | Date Build | Comment | Date Upload | Directory | Flag | Actions |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2018.05.14.17.35.17_1.png | WKS | 16.11.18 | 1231 | 2018-05-14 | 12312 | 2018.05.14.17.35.17 | C:\workspace\UploadedFiles\ | false | Delete |
| 4 | 2018.05.14.17.53.17_ssh.txt | WKS | 17.18.21 | 1231 | 2018-05-14 | 213 | 2018.05.14.17.53.17 | C:\workspace\UploadedFiles\ | false | Delete |
| 5 | 2018.05.14.17.54.18_index.jpg | WKS | 17.18.22 | 1231 | 2018-05-14 | 213 | 2018.05.14.17.54.18 | C:\workspace\UploadedFiles\ | false | Delete |
| 6 | 2018.05.15.09.52.54_ssh.txt | WKS | 18.18.24 | 21 | 2018-05-15 | 21 | 2018.05.15.09.52.54 | C:\workspace\UploadedFiles\ | false | Delete |
| 7 | 2018.05.15.09.56.21_ssh.txt | WKS | 18.18.27 | 21 | 2018-05-15 | 21 | 2018.05.15.09.56.21 | C:\workspace\UploadedFiles\ | false | Delete |
| 8 | 2018.05.15.10.03.52_ssh.txt | WKS | 18.18.28 | 231 | 2018-05-15 | weq | 2018.05.15.10.03.52 | C:\workspace\UploadedFiles\ | true | Delete |
| 9 | 2018.05.15.12.36.34_ssh.txt | WKS | 18.18.29 | 123 | 2018-05-14 | 12312 | 2018.05.15.12.36.34 | C:\workspace\UploadedFiles\ | true | Delete |

Figura 5.7: Home Page

- ***Upload List***

O separador *Upload* dá acesso a duas páginas, umas delas é a representada na figura 5.8. Esta página ilustrada é bastante simples e semelhante à lista de Versões existentes na primeira página, a única diferença é que a lista da página principal, está limitada a 15 Versões, enquanto esta lista dá acesso a todos os ficheiros que existem no repositório. Este limite de Versões foi implementado para não sobrecarregar a página principal, contudo, esta página deixará de existir, no futuro, e passará a haver uma opção na "*Home Page*"que possibilitará o utilizador de escolher a quantidade de ficheiros que são listados.

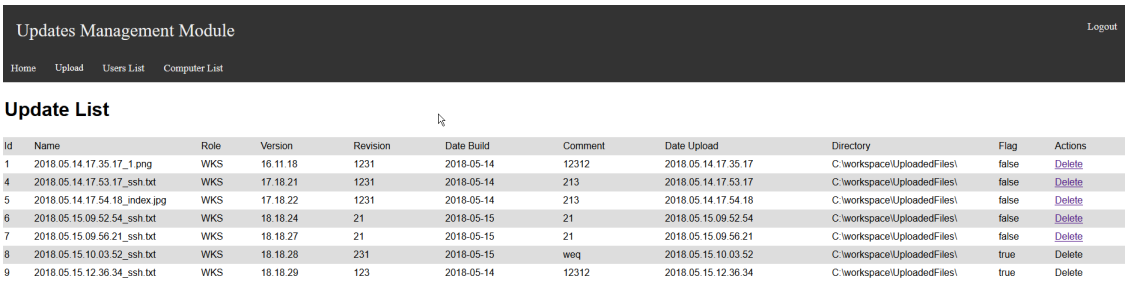



Figura 5.8: Upload List

- ***Upload To***

  O agendamento de atualizações nos postos de trabalho é um dos pontos chaves deste projeto. Na página representada na figura 5.9, o utilizador poderá agendar as versões nos postos de trabalho que pretende atualizar. Para isso, seleciona um ficheiro de atualização na lista de *Updates* e os postos de trabalho que pretende atualizar da lista de computadores. Esta interface permite atualizar vários postos de trabalho ao mesmo tempo. De seguida, basta selecionar, do formulário *Data Upload*, a data de realização da atualização e clicar no botão *Upload*. Depois de agendada, a atualização surgirá numa lista, nesta mesma página, com a opção de a eliminar, se esta não tiver ultrapassado a data de agendamento, nesse caso a atualização continuará.

  Apenas surgirá na lista de computadores, os postos de trabalho que estejam em modo de atualização remoto. Os restantes serão automaticamente filtrados.

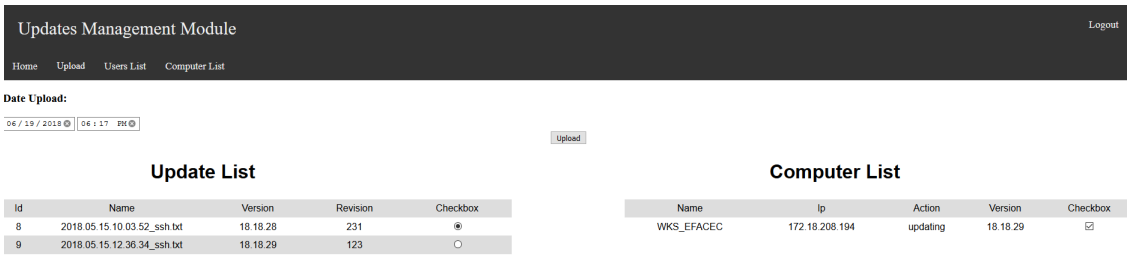



Figura 5.9: Upload To

- ***Users List***

  Nesta página (figura 5.10) é apresentada a lista de todos os utilizadores registados no servidor. Um utilizador com perfil de administrador pode alterar a *Password* dos utilizadores. Reforçando a informação exposta previamente, apenas existe um utilizador na Plataforma: o Administrador. Como a gestão de utilizadores não era um requisito funcional do projeto, não foi dado muito ênfase a este componente. No entanto, a página é funcional e pode ser aproveitada para trabalho futuro.

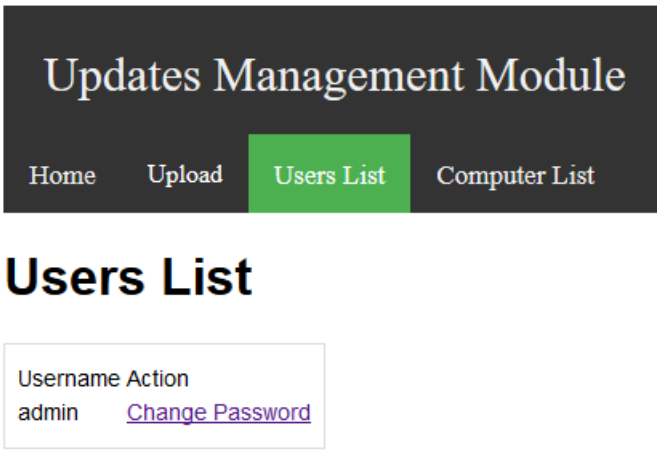



Figura 5.10: Users List

- ***Computer List***

  Na figura 5.11, é apresentada a lista de todos os postos de trabalho que se registaram, através da API na plataforma *Web*. Foi conseguido, com sucesso, registar o computador local na Plataforma. Esta página fornece ao utilizador a capacidade de configurar alguns dados sobre o posto de trabalho selecionado. Ao clicar no botão de configuração, surgirá um *Popup* que permite alterar 4 campos: Modo de atualização; Nome; IP; *Role*. É, também, possível visualizar, no campo *Action*, o estado em que o posto de trabalho se encontra, o que permite seguir todas as etapas de uma atualização.

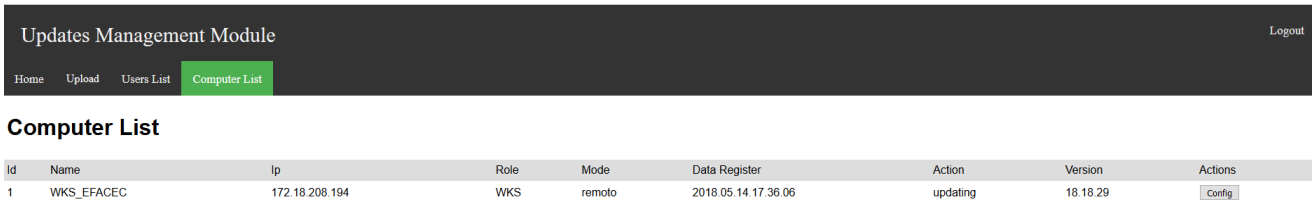



Figura 5.11: Computers List

### 5.2.3 Aplicação Windows

Nesta secção, será feita uma demonstração dos diferentes mecanismos da Aplicação *Windows*: o mecanismo de instalação e registo (surge logo após o *Download* do instalador); o mecanismo de mudança do Modo de Atualização (alteração possível de ser realizada pelo utilizador); o mecanismo de notificação (verifica periodicamente se existe alteração de dados na Plataforma *Web*).

Todos os dados informativos sobre os postos trabalho são guardados num ficheiro de configuração na própria máquina. Se for necessário alterar os dados da máquina, estes podem ser alterados no ficheiro de configuração ou através do modo de configuração da Interface *Web*. O

modo de configuração pela Interface apenas permite a alteração de alguns dados, em oposição ao ficheiro de configuração.

- **Mecanismo de Instalação e Registo**

  Consiste na instalação, registo e primeira ligação da aplicação *Windows*.

  O IDE escolhido para desenvolver a aplicação, disponibiliza diversos *Templates*, para facilitar o desenvolvimento de *Software*. Um desses *Templates* é o *Setup Wizard*, que é utilizado como instalador de aplicações. No instalador criado para o projeto será pedido ao utilizador para preencher alguns campos de dados iniciais, nomeadamente: Nome do posto de trabalho; Versão; Revisão da atualização atualmente em uso. Neste momento, ainda não existe nenhum mecanismo que permita identificar a Versão do *Software* que está instalada no posto de trabalho, portanto, o ponto de partida serão os dados que o utilizador inserir na primeira configuração.

  Após a instalação, a aplicação iniciará automaticamente e será enviada uma mensagem de pedido à plataforma web para verificar se o posto de trabalho está registado no servidor. Caso não esteja, os dados necessários para o registo irão ser enviados para a Plataforma. Neste processo de registo, irá ser atribuído um ID ao posto de trabalho. Apesar do IP da máquina poder ser considerado um identificador único, para simplificar e organizar o trabalho, atribuiu-se um ID a todos os postos de trabalho.

  Após o primeiro registo, a aplicação despoletará uma janela informativa (que pode ser visto na figura 5.12) com: a Versão; Nome; IP; Modo de Atualização da Máquina. Concluindo, assim, o processo de instalação, registo e primeira ligação da aplicação. Quando a janela informativa for minimizada, surgirá o *System Tray* (figura 5.13).

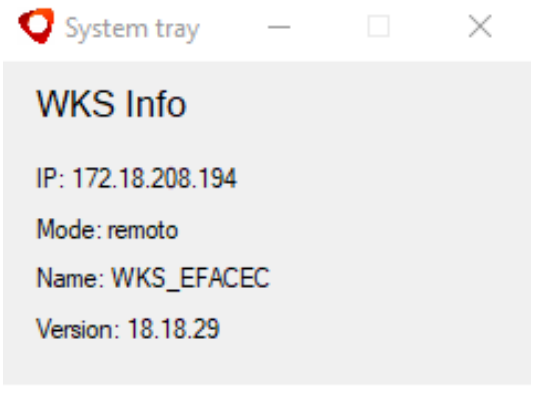



Figura 5.12: Janela Informativa


Figura 5.13: System Tray

- **Mecanismo de Alteração de Modo**

  Sendo uma aplicação dinâmica, o utilizador pode alterar o modo em que o sistema se encontra, podendo escolher entre o modo Manual ou o modo automático. O modo Remoto

apenas pode ser configurado na Plataforma *Web*. No entanto, caso deixe de estar no Modo Remoto, o posto de trabalho passa para o último modo que esteve ativo.

Observando a figura 5.14, a aplicação encontra-se em modo "Manual"e o utilizador apenas pode selecionar a opção "Auto", estando o modo "Remoto"indisponível. Por oposição, na figura 5.15, pode-se verificar que o modo "Remoto"está ativado, impedindo o utilizador de alterar para um Modo diferente.

Para além destas dinâmicas, o utilizador pode ver a janela informativa, ao clicar em "Info", ou ao sair da Aplicação (clicando em "Exit").

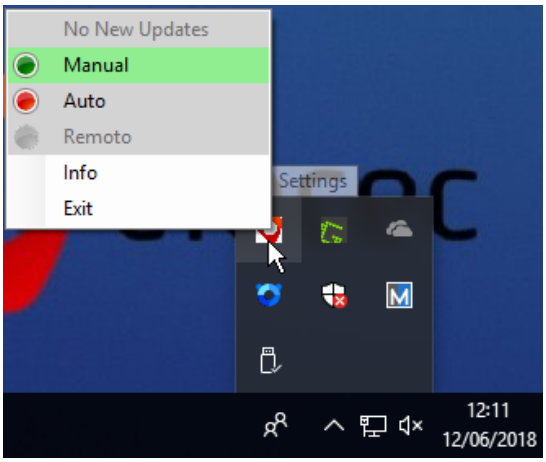



Figura 5.14: Aplicação em Modo Manual

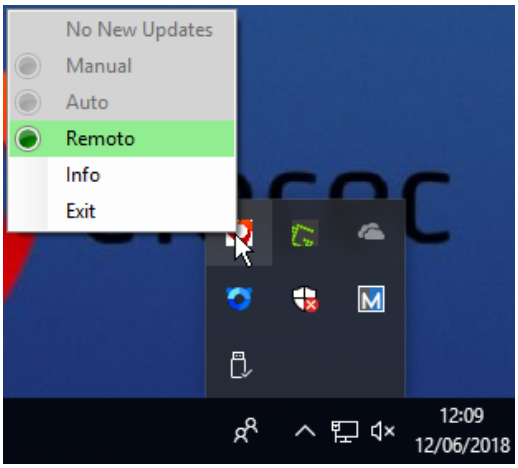



Figura 5.15: Aplicação em Modo Remoto

- **Mecanismo de Notificação**

  Podem existir 2 tipos de notificação: um aviso na Interface do *System Tray* (figura 5.16) ou um alerta do *Windows* (figura 5.17). Se existir uma nova Versão na Plataforma ou uma atualização agendada, estas notificações podem ocorrer nos seguintes cenários:

  - **Modo Manual** - o operador do posto de trabalho será notificado que pode realizar o *download* (figura 5.16). Após a conclusão do respetivo download (figura 5.17), o operador será alertado relativamente à possibilidade de efetuar a instalação da atualização. Quando a instalação estiver concluída, receberá um alerta a informar do sucesso ou insucesso da mesma.
  - **Modo Auto** - o utilizador receberá apenas duas notificações. A primeira diz respeito à possibilidade de realizar uma atualização que o serviço descarregou automaticamente. A segunda informa acerca do sucesso ou insucesso da instalação.
  - **Modo Remoto** - o utilizador receberá um alerta do *Windows* a informar que uma nova versão foi instalada no posto de trabalho. Não foi possível completar na totalidade a implementação deste modo de atualização. Num cenário ideal o utilizador autenticado no posto de trabalho, seria avisado que uma atualização de *software* iria dar inicio. O utilizador teria a oportunidade de recusar e adiar a instalação.

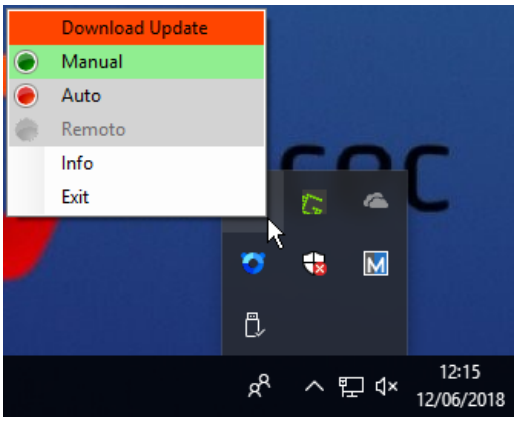



Figura 5.16: Aviso de disponibilidade de nova atualização

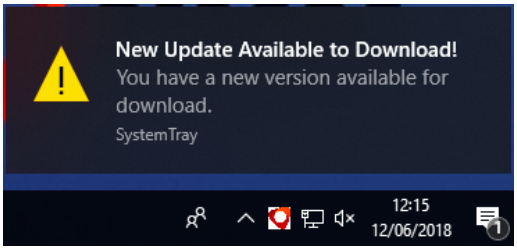



Figura 5.17: Alerta que existe uma Atualização Disponível

# Capítulo 6

# Conclusões e Trabalho Futuro

Este trabalho teve o intuito de automatizar os processos de atualização de software da EFACEC. Colaborar para um protótipo que pretende melhorar o funcionamento de uma empresa como a EFACEC foi um aspeto essencial para o aumento da minha motivação ao longo do trabalho. Este projeto propôs, portanto, uma série de objetivos para o completo funcionamento do mesmo.

Infelizmente, não foi possível cumpri-los como era esperado inicialmente, devido ao reduzido período de tempo que tive comparativamente ao necessário para, de facto, realizar com sucesso todos os objetivos propostos. Apesar do parcial cumprimento destes, afirmo sentir uma concretização pelo trabalho que realizei, pois este projeto exigiu que eu superasse todos os meus conhecimentos e aptidões, promovendo o desenvolvimento das cognições que possuía relativamente às novas tecnologias. Tais conhecimentos que adquiri serão extremamente importantes e representarão um papel fulcral para tornar um futuro profissional mais apto para exercer no mundo do trabalho.

Efetuar-se-á uma breve conclusão relativa à satisfação dos requisitos na Secção 6.1 e ao trabalho futuro que pode ser derivado desta dissertação na Secção 6.2.

## 6.1 Satisfação dos Objetivos

Os objetivos do projeto foram concretizados quase na sua plenitude. A revisão bibliográfica foi fundamental para a pesquisa e aprendizagem de tecnologias e conhecimentos essenciais para o desenvolvimento deste trabalho.

Neste momento, existe um protótipo com uma interface web ligada a uma base de dados que comunica com uma Aplicação *Windows* através de pedidos HTTPS numa REST API. Através de testes, é possível introduzir um ficheiro na plataforma web, a Aplicação *Windows* consegue registar a estação de trabalho na plataforma web e, assim, ocorrer a transferência do ficheiro para a estação de trabalho através de SSH. Portanto, obtém-se o produto mínimo necessário para permitir o bom funcionamento do trabalho.

No entanto existem alguns objetivos que não foram concluídos:

- Controlo de tráfego - num cenário de produção, surgem dezenas ou centenas de máquinas ligadas à rede. Logo, não seria conveniente a transferência simultânea de ficheiros por parte das mesmas, porque tal iria sobrecarregar o servidor, bem como a rede em geral. Para evitar o cenário descrito, é preciso implementar um sistema de controlo de tráfego.

- *Setup Wizard* - como já foi mencionado em capítulos anteriores, trata-se do instalador da Aplicação *Windows* nas estações de trabalho. O *Visual Studio* está preparado com um *template* que facilita o desenvolvimento deste tipo de instalador.

- Interface de agendamento - numa fase já avançada do projeto, houve sugestão de eliminar a página de agendamento (figura 5.9) substituindo-a por uma a lista de estações de trabalho (figura 5.9). Com esta alteração, o utilizador ao selecionar a estação de trabalho que gostaria de atualizar, surgiria um *popup* com a lista de atualização, a data e hora para agendar. Com isto, eliminar-se-ia uma interface, tornando a plataforma web mais simples e leve. Mas, devido à fase já avançada em que o projeto se encontrava, decidiu-se não alterar o que já estava feito e em funcionamento.

- Aplicação *Windows* - esta componente foi a última a ser desenvolvida, por isso, contém pequenos pormenores que necessitam de aperfeiçoamento como a implementação de todas as notificações.

## 6.2 Trabalho Futuro

Durante os quatros meses que estive na EFACEC, houve o destaque do potencial deste protótipo. Este projeto está preparado para transferir e executar qualquer tipo de trabalho nas estações de trabalho, podendo futuramente, adotar a aplicação para atualização de softwares de sistema ou outro tipo de software necessário.

Outras questões ficaram por abordar, como, por exemplo, as questões de segurança. Enquanto programador, existe sempre uma preocupação de trabalhar respeitando boas práticas, no entanto, uns métodos apesar de mais eficientes são, também, mais demorosos. No caso deste protótipo utilizaram-se métodos menos completos, mas mais rápidos e que cumpriam o objetivo que este trabalho requeria.

De seguida, é apresentada uma lista com algumas funcionalidades que poderão ser implementadas:

- Transferência de ficheiros - quando se transfere um ficheiro da plataforma web para uma estação de trabalho, este é transferido através de SCP. De momento, quando a estação verifica que é necessário efetuar-se uma atualização, esta envia uma mensagem por HTTPS no formato JSON contendo as credenciais da estação de trabalho para que a plataforma possa realizar a transferência do ficheiro. Contudo, isto não se trata de uma boa prática. Para um trabalho futuro, será correto implementar a partilha de uma chave encriptada entre os componentes para que estes possam estabelecer a comunicação e transferência de dados.

- Processos - Como já foi referido, não foi possível testar este projeto num cenário de teste adequado. Com isto, nunca surgiu o caso de se instalar, de facto, uma atualização no posto de trabalho.

  Criando o seguinte cenário: se a estação de trabalho estiver em modo remoto, é preciso verificar que não existem outros processos a correr, previamente a uma instalação de software. Não se pode correr o risco de interromper com uma instalação, um utilizador que esteja a meio de um processo. Seria importante analisar o melhor método para a verificação do momento apropriado para um sistema se encontrar num estado em que é propícia a instalação de uma nova atualização.

- Auto-atualizável - este protótipo caso entre em produção, como todos os softwares, irá sofrer melhorias e adição de novas características e funcionalidades. Foi proposto, pela EFACEC, que se colocasse como um possível trabalho futuro, um método que tornaria a Aplicação *Windows* auto-atualizável, isto é, sempre que o sistema verificasse que a aplicação na estação de trabalho se encontra desatualizada, instala-se uma nova atualização da mesma através da plataforma web.

# Anexo A

# Anexo

## A.1 Diagrama de Classes

### A.1.1 Login CLass

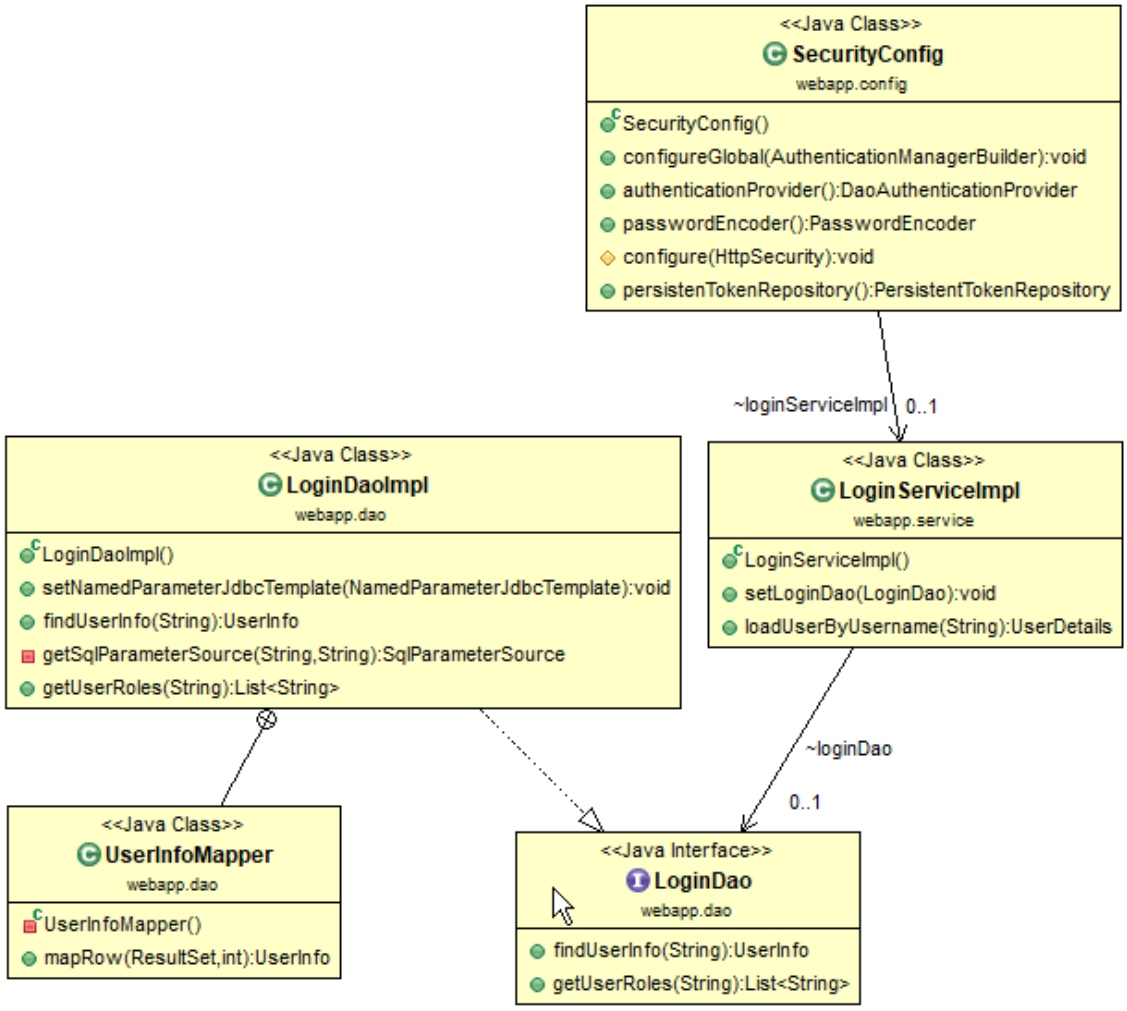



Figura A.1: Diagrama de classes, Login

### A.1.2 User Class

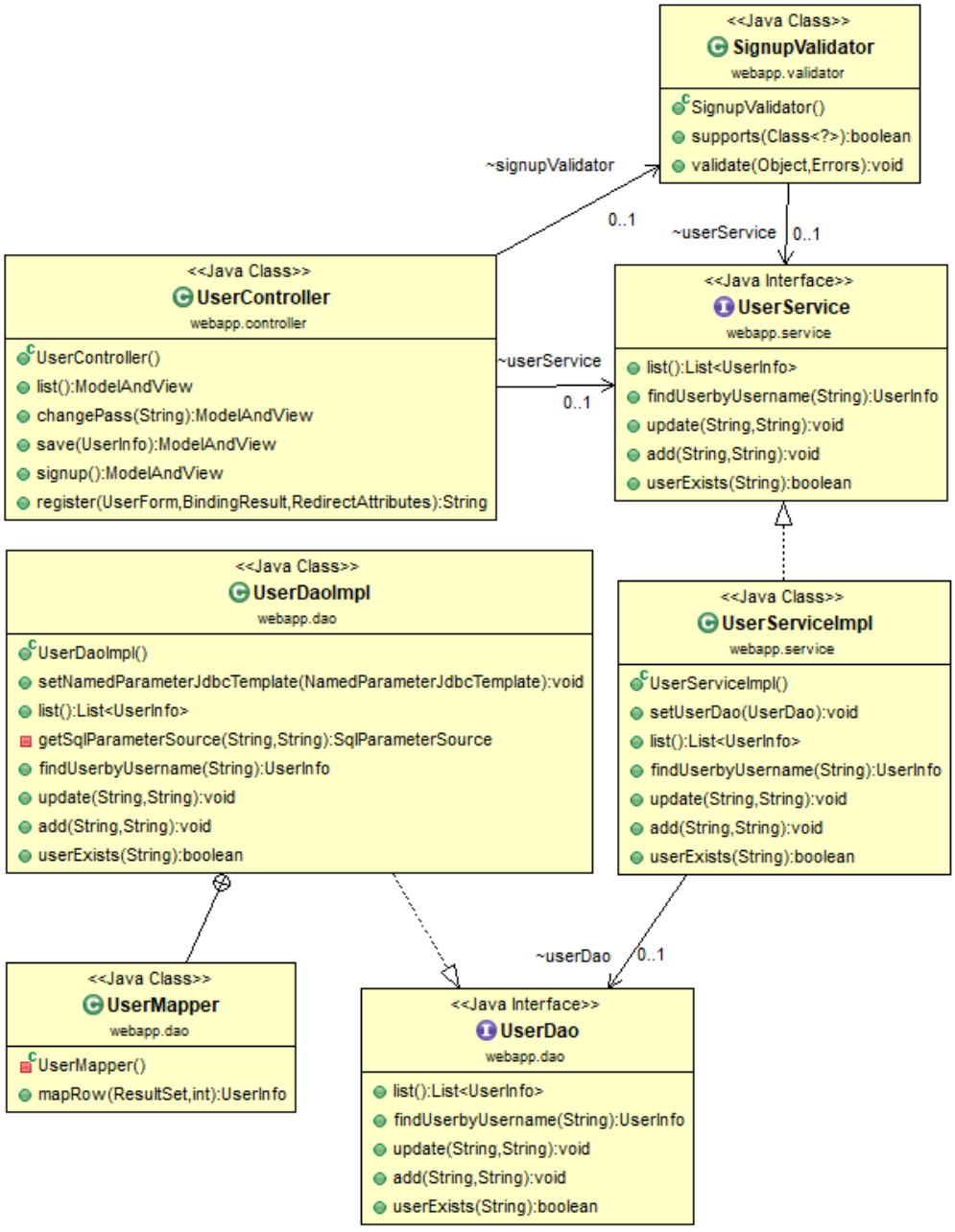



Figura A.2: Diagrama de classes, Users

### A.1.3 File Class

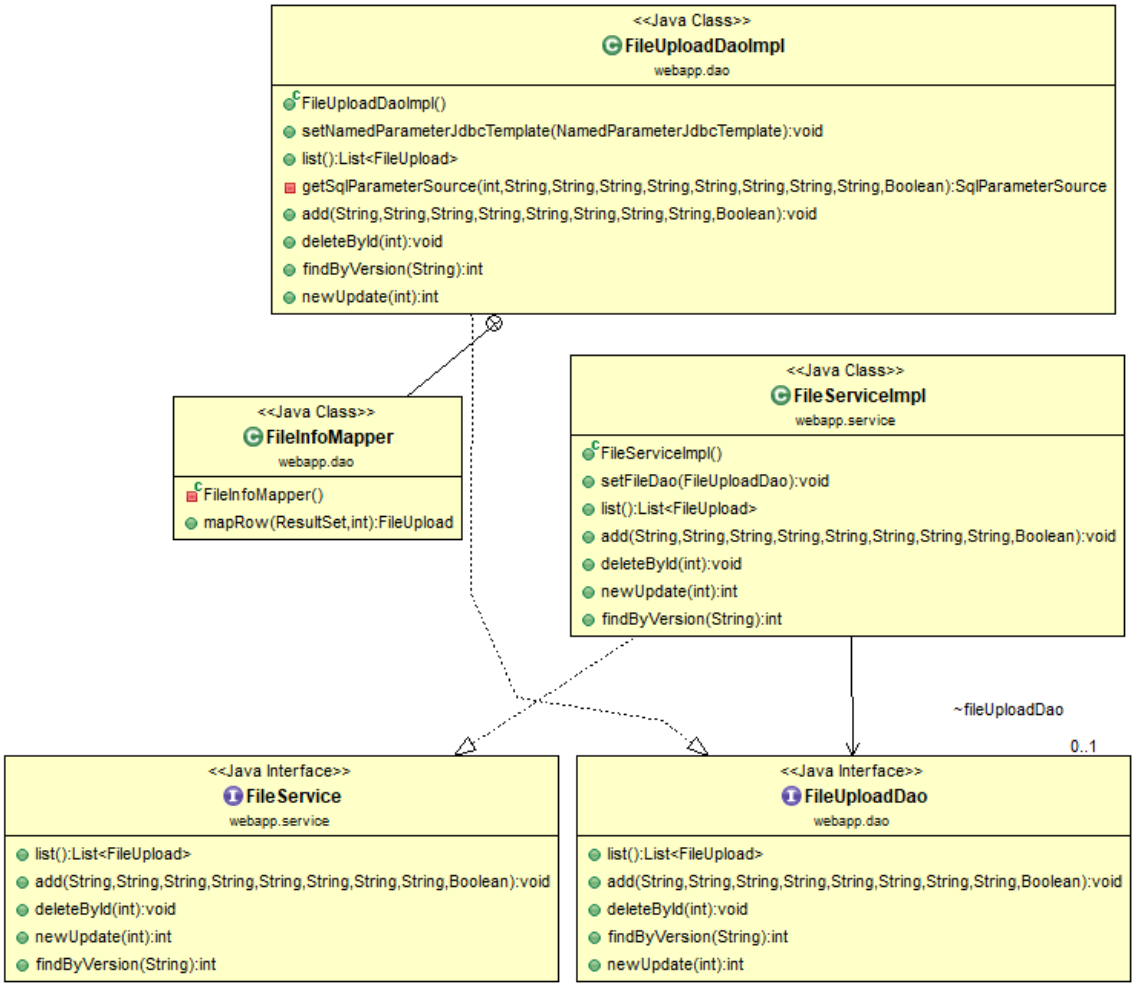



Figura A.3: Diagrama de classes, FileClass

### A.1.4 Computers CLass

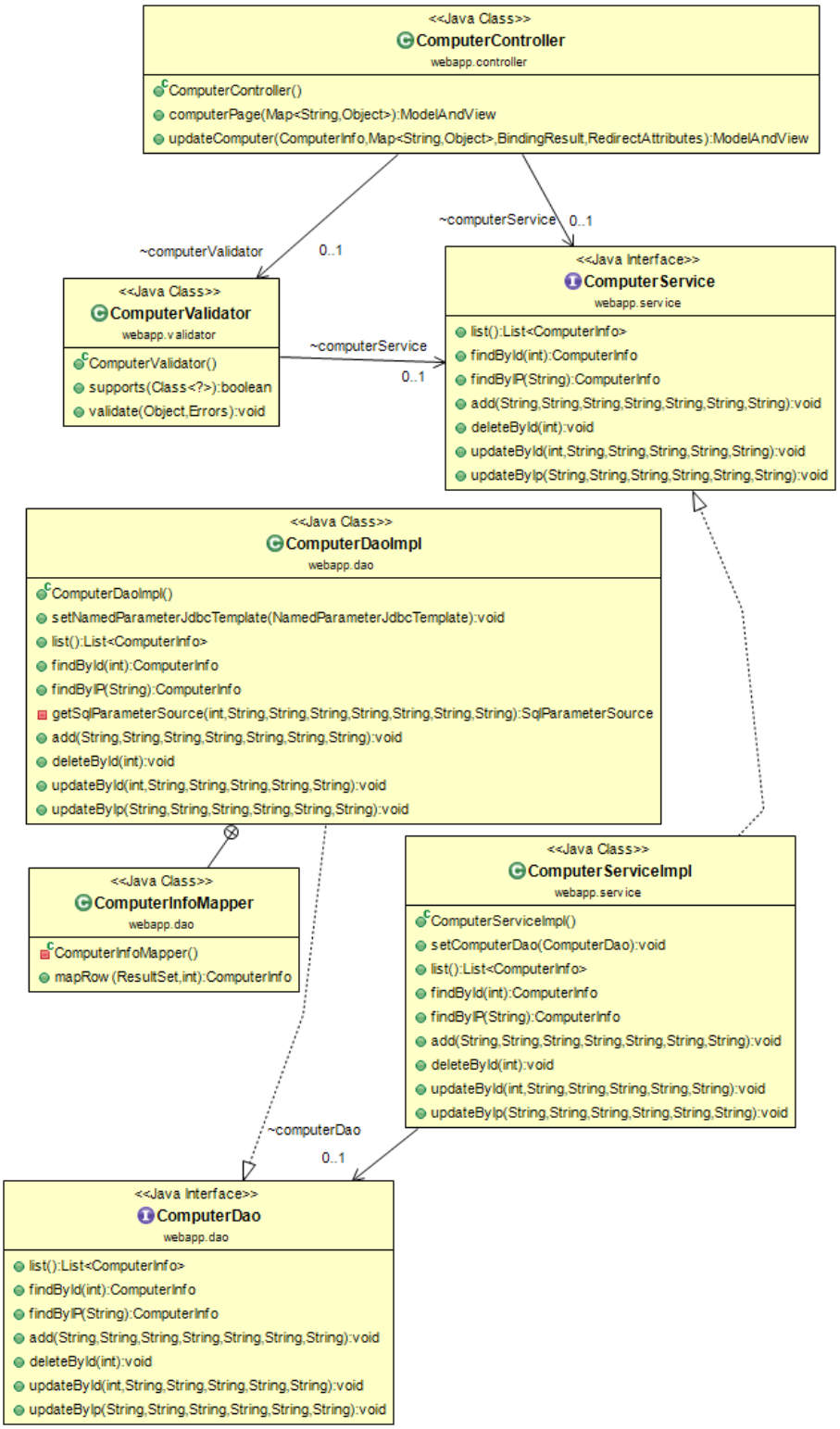



Figura A.4: Diagrama de classes, Computers

## A.2 Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
        apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.efacec.miguel</groupId>
    <artifactId>WebAppMVC</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.source>1.6</maven.compiler.source>
        <maven.compiler.target>1.6</maven.compiler.target>
        <jackson.version>2.7.5</jackson.version>
        <java-version>1.7</java-version>
    </properties>
    <repositories>
        <repository>
            <id>prime-repo</id>
            <name>PrimeFaces Maven Repository</name>
            <url>http://repository.primefaces.org</url>
            <layout>default</layout>
        </repository>
    </repositories>
    <dependencies>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.3.168</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>4.3.0.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.3.0.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
```

```
        <artifactId>spring-security-web</artifactId>
        <version>4.0.4.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
        <version>4.0.4.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>4.3.0.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
    </dependency>
    <!-- Faces Implementation -->
    <dependency>
        <groupId>com.sun.faces</groupId>
        <artifactId>jsf-impl</artifactId>
        <version>2.2.4</version>
    </dependency>
    <!-- Faces Library -->
    <dependency>
        <groupId>com.sun.faces</groupId>
        <artifactId>jsf-api</artifactId>
        <version>2.2.4</version>
    </dependency>
    <!-- JSTL Library -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
```

```
    <version>1.1.2</version>
</dependency>
<dependency>
    <groupId>commons-dbcp</groupId>
    <artifactId>commons-dbcp</artifactId>
    <version>1.4</version>
</dependency>
<dependency>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-core</artifactId>
    <version>5.0.7</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.22</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>1.7.6</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>1.0.0.GA</version>
</dependency>
<!-- Apache Commons FileUpload -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>

<!-- Apache Commons IO -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>3.2.0</version>
</dependency>
```

```
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20090211</version>
        </dependency>
        <dependency>
            <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-mapper-asl</artifactId>
            <version>1.9.13</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>\${jackson.version}</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish</groupId>
            <artifactId>javax.json</artifactId>
            <version>1.0.4</version>
        </dependency>
        <dependency>
            <groupId>com.jcraft</groupId>
            <artifactId>jsch</artifactId>
            <version>0.1.50</version>
        </dependency>
        <dependency>
            <groupId>org.apache.ant</groupId>
            <artifactId>ant-jsch</artifactId>
            <version>1.8.1</version>
        </dependency>
    </dependencies>

    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>2.4</version>
                    <configuration>
                        <source>1.7</source>
                        <target>1.7</target>
                        <warSourceDirectory>src/main/webapp</
                            warSourceDirectory>
```

```
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
</project>
```

# Referências


[1] Carlos Domingo Moreira dos Santos. Apoio na implementação de sistemas de gestão em mercados internacionais na efacec sistemas de gestão, s.a. Relatório técnico, Faculdade de Engenharia da Universidade do Porto, 2012.

[2] Efacec. Quem somos, 2018. Disponível em http://www.efacec.pt/quem-somos/, acedido a última vez em 10 de Junho de 2018.

[3] Ricardo Jorge Nogueira Fernandes. Interfaces web para aplicações scada. Relatório técnico, Faculdade de Ciências da Universidade do Porto, 2003.

[4] Efacec. Scatex+ cc, 2018. Disponível em http://www.efacec.pt/quem-somos/, acedido a última vez em 10 de Junho de 2018.

[5] Filipe Manuel Azevedo Marinho. Framework para visualização de diagramas na web na efacec sistemas de eletrÓnica s.a. Relatório técnico, Faculdade de Ciências da Universidade do Porto, 2004.

[6] Cristiano Rafael da Mota Lima Gomes. Migração scate x - sistemas de supervisão e telecontrolo para hp-ux. Relatório técnico, Faculdade de Engenharia da Universidade do Porto, 2005.

[7] Efacec. Scatex+ network manager, 2018. Disponível em http://www.efacec.pt/en/wp-content/uploads/2016/10/CS256I1502B1.pdf, acedido a última vez em 14 de Junho de 2018.

[8] Pedro Miguel Freitas Moura. Identity management and authorization infrastructure in secure mobile access to electronic health records. Relatório técnico, Universidade da Beira Interior, 2018.

[9] José Joaquim Magalhães Moreira. Wsql uma arquitectura de software baseada em web services. Relatório técnico, Faculdade de Ciências da Universidade do Porto, 2005.

[10] Vangie Beal. system tray, 2018. Disponível em https://www.webopedia.com/TERM/S/system_tray.html, acedido a última vez em 18 de Junho de 2018.

[11] Pivotal Software. Building a restful web service, 2018. Disponível em https://spring.io/guides/gs/rest-service/, acedido a última vez em 22 de Junho de 2018.

[12] Inc Eclipse Foundation. Eclipse ide for java developers, 2018. Disponível em https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/keplersr1, acedido a última vez em 20 de Junho de 2018.

[13] H2. Embedding h2 in an application, 2018. Disponível em http://www.h2database.com/html/quickstart.html, acedido a última vez em 20 de Junho de 2018.

[14] Postdot Technologies. Postman, 2018. Disponível em https://www.getpostman.com/docs/v6/, acedido a última vez em 20 de Junho de 2018.

[15] The Apache Software Foundation. What is maven?, 2018. Disponível em https://maven.apache.org/what-is-maven.html, acedido a última vez em 23 de Junho de 2018.

[16] The Apache Software Foundation. Apache log4j 2, 2018. Disponível em https://logging.apache.org/log4j/2.x/, acedido a última vez em 22 de Junho de 2018.