UNIVERSIDADE da MADEIRA

DEPARTAMENTO DE MATEMÁTICA E ENGENHARIAS

TACREMOTE

Rui Aureliano Ferreira Gonçalves

(Licenciado)

Tese Submetida à Universidade da Madeira para a

Obtenção do Grau de Mestre em Engenharia Informática

Funchal – Portugal

Novembro 2008

Orientador

Mestre Luís Alberto da Silva Gaspar,

Assistente convidado no Departamento de Matemática e Engenharia da Universidade da Madeira

RESUMO

A automatização de tarefas que sejam sistemáticas é um dos objectivos que se pressupõe fazer nesta investigação, pois com a ajuda de computadores conseguimos grande performance bem como eficiência/eficácia na execução dessas tarefas.

Este trabalho tem como objectivo desenvolver uma aplicação para monitorização de temperatura e consequentemente com base nessas temperaturas fazer cálculos de resistências previstas, análise de valores e visualização de gráficos. Esta aplicação diferencia-se das outras aplicações que existem no mercado por utilizar uma tecnologia opensource, ser altamente modular e expansível e pela arquitectura utilizada (MVC) ser multiplataforma.

De forma a tornar esta aplicação mais apelativa optou-se por utilizar a tecnologia FLEX da Adobe, que permite a realização de interfaces ricas e ao mesmo tempo simples de modo a permitir uma fácil utilização do programa mas sem que isso implique perda de beleza estética

Pretende-se que a aplicação seja implementada a nível empresarial, e que se torne num futuro próximo numa aplicação por excelência e extensível o suficiente para cobrir grande parte das necessidades específicas das empresas da área da construção civil.

ABSTRACT

The automation of tasks that are systematically is one of the objectives that requires doing this research, because with the help of computers achieved great performance and efficiency / effectiveness in carrying out these tasks

This paper aims to develop an application for monitoring of temperature and consequently on the basis of these temperatures make calculations of resistance planned, analysis of values and display graphics. This application differentiates itself from other applications on the market by using an opensource technology, is highly modular and expandable and the architecture used (MVC) be multi platform.

In order to make the application more appealing we opted to use the FLEX technology, which enables the achievement of interfaces and rich at the same time simple to enable easy operation of the program but without the loss of aesthetic beauty

It is intended that the application is implemented at the enterprise level, and that becomes in the near future in an application for excellence, which extended enough to cover large parts of the specific needs of businesses in the area of construction.

PALAVRAS-CHAVE

Monitorização, Maturação de betão, Análise de valores, Controlo de temperatura, Cálculo de resistências, Análise de gráficos, Gestão de utilizadores, Geo-referenciação

KEYWORDS

Monitoring, Maturation, Analysis of values, Control of temperature, Calculation of resistance, Graphic analysis, User management, Geo-referencing

AGRADECIMENTOS

Ao Mestre Luís Gaspar pela orientação e apoio ao longo de todo o período do mestrado.

Aos meus colegas de trabalho sem nenhuma ordem em particular João Pereira, Fábio Neves, Rodolfo Abreu, Lisber Pontes, Juan Figueira e Dulce Rocha pelo apoio e ajuda na detecção de erros.

À Mónica por me ter ajudado na revisão final do relatório

À Softventure pela oportunidade de colaboração e parceria no desenvolvimento do sistema

À Universidade da Madeira pela oportunidade de realização do curso de mestrado.

A todos os que ajudaram a fazer os testes necessários.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ACRÓNIMOS

SGBD – Sistema Gestor de Bases de Dados

HVAC – Heating, Ventilating, and Air Conditioning

E/S – Entrada e Saída

LAMP – Linux, Apache, MySQL, PHP / Perl / Python

ODBC – Open Database Connectivity

SDK – Software Development Kit

IDE – Integrated Development Environment

J2EE - Java 2 Platform, Enterprise Edition

W3C - World Wide Web Consortium

XML - eXtensible Markup Language

SGML - Standard Generalized Markup Language

MVC – Model View Controller

1. INTRODUÇÃO

"You'll never need more than 64K of memory"

Bill Gates

Tópicos

1.1 MOTIVAÇÃO E OBJECTIVOS

Automatizar e melhorar a performance das tarefas, sempre foi algo que o homem procurou fazer nas mais diversas áreas e partindo desta necessidade em relação a área da construção civil, e tendo em conta que não se fizeram muitos trabalhos nesta área, procurou-se neste trabalho fazer um sistema que permitisse analisar e monitorizar obras de construção civil sem a necessidade de presença física, ou seja esta análise e monitorização ser feita remotamente. Associada a falta de ferramentas para fazer este tipo de monitorização, identificamos os seguintes objectivos:

- Criar um sistema que permitisse a monitorização de vários parâmetros relacionados com a obra, tais como dureza do betão, resistência do betão e temperatura do betão.
- Permitir a gestão de utilizadores no sistema, bem como os seus privilégios, ou seja poder atribuir a um utilizador permissões para ver ou não certos parâmetros.
- Permitir ao utilizador fazer análises, através dos parâmetros disponibilizados (dureza, resistência, temperatura), com o auxílio de tabelas e gráficos.
- Permitir ao utilizador localizar o local das obras através de mapas dinâmicos.
- Notificar os utilizadores do sistema, sempre que seja atingido um determinado evento.

1.2 PROJECTO TACREMOTE

No âmbito do projecto de mestrado em engenharia informática na Universidade da Madeira no ano lectivo de 2007/2008, vamos desenvolver uma aplicação (**tacRemote**), que consiste em aceder remotamente através da internet a um dispositivo (TAC XENTA 511®), que consegue monitorizar os valores de sensores que estão ligados a este. Neste caso serão sensores de temperatura, mas no futuro poderão ser de outra natureza (pressão, humidade etc.).

O projecto é basicamente dividido em duas partes, a primeira parte consiste em explorar o TAC XENTA 511®, que é o dispositivo que armazena os valores que são lidos pelos sensores e conseguir aceder a estes valores. O TAC XENTA 511® tem um modo de funcionamento como Web Server, e estes dados são acedidos por pedidos http (HTTPRequest) e depois de feita a leitura dos dados, é devolvido esses resultados num formato standard e gratuito (**XML**), de modo a que esta leitura não esteja dependente de tecnologias e/ou soluções proprietárias.

A segunda parte consiste em criar uma aplicação (tacRemote) que consiga aceder a esses valores e faça um tratamento a esses valores, fazendo estatísticas, mostrando gráficos, efectuando cálculos, mostrando tabelas, simulações, previsões etc.

O **tacRemote** foi desenvolvido para auxiliar os utilizadores na monitorização de obras de construção civil, auxiliar a análise e compreensão dos valores lidos através dos sensores instalados nas obras. Com base nesses valores o sistema faz previsões, podendo essas previsões servir para auxiliar a tomada de decisões, como por exemplo se o betão está ou não resistente o suficiente, permite para alem disso a gestão dos utilizadores, para que se possa atribuir/retirar permissões para fazer certas operações.

1.3 Enquadramento da Dissertação

Esta dissertação foi feita em colaboração com a empresa Softventure Consultoria & Tecnologia S.A [SV2008]. A Softventure é uma empresa criada em Dezembro de 2004, tendo o core business assente na concepção, desenvolvimento e implementação de soluções tecnológicas, utilizando as mais recentes metodologias e ambientes de desenvolvimento, de forma a posicionar-se junto dos seus clientes com um grau de inovação cujo foco passa por responder às suas necessidades de curto prazo, mas simultaneamente procurando antecipar, sem condicionar, as suas necessidades no médio e longo prazo.

1.4 Organização da Dissertação

Esta dissertação esta organizada em 7 capítulos, que cobrem todo processo de desenvolvimento do software e estão organizados da seguinte forma:

Capitulo 1 – Introdução: Contextualiza o trabalho de investigação, introduz-se o sistema **tacRemote**, a sua motivação e objectivos, e sintetiza-se a organização da dissertação.

Capitulo 2 – Estado da Arte: Apresentamos o estado da arte da temática dos dispositivos de controlo remoto existentes da empresa TAC, focaliza-se no estudo das linguagens de programação, protocolos de comunicação e SGBD escolhidos para o desenvolvimento do sistema **tacRemote**.

Capitulo 3 – Modelos e Metodologias: Fazemos uma revisão bibliográfica do conjunto de modelos metodologias. Apresenta-se o paradigma escolhido para o

desenvolvimento do sistema e por fim justifica-se a plataforma e as linguagens de programação escolhidas.

Capitulo 4 – Implementação do Sistema: Fazemos o levantamento dos requisitos do sistema, justificam-se os métodos e metodologias utilizadas e apresenta-se a modelação do sistema.

Capitulo 5 – O Sistema tacRemote: Apresentamos a interface do sistema tacRemote, mostrando-se as funcionalidades mais importantes do sistema e a sua relação com o utilizador. Faz-se também três pequenas demonstrações do sistema

Capitulo 6 – Módulos do Sistema: Apresentamos os módulos do sistema, explica-se a relação entre cada um deles, mostra-se os componentes de cada um dos módulos, assim como se relacionam esses componentes, quais os parâmetros que passam e recebem.

Capitulo 7 – Conclusão e Trabalhos Futuros: Apresentamos as conclusões gerais de todo os trabalho de investigação relativo à concepção e implementação do sistema tacRemote e mostra-se ainda alguns aspectos em aberto que merecem trabalho futuro.

2. ESTADO DA ARTE

"The people who are doing the work are the moving force behind the Macintosh. My job is to create a space for them, to clear out the rest of the organization and keep it at bay."

Steve Jobs

Tópicos

2.1 INTRODUÇÃO

Neste capítulo vamos fazer um estudo do estado da Arte, onde vamos nos focalizar em vários dispositivos de controlo desenvolvidos pela empresa TAC, ver as semelhanças e diferenças entre os mesmos, ver o seu propósito de construção. Também serão analisadas várias linguagens de programação, em particular serão analisadas com maior detalhe as que achei mais indicadas para o desenvolvimento deste projecto. Finalmente explorar um pouco o SGBD escolhido bem com os protocolos de comunicação usados pelos controladores da TAC.

2.2 TAC XENTA®

A **TAC** [TACCB] é a empresa com maior crescimento e maior inovação na indústria actual de construção automatizada. O objectivo desta passa por providenciar serviços de controlo de temperatura, segurança e uso de energia recorrendo à mais avançada tecnologia para os utilizadores finais.

A **TAC** estabeleceu o conceito de Open Integrated Systems for Building IT™ no Mercado. Trata-se de uma tecnologia baseada em standards que permite aos utilizadores integrar sistemas de aquecimento/arrefecimento, controlo de acessos, monitorização de segurança e ventilação, controlo de fogos e fumos. Esta aproximação reduz tanto o custo de manutenção, como o consumo de energia.

Nos dias de hoje, os sistemas **TAC** podem ser encontrados em mais de 150,000 instalações em cada continente, numa variedade distinta de empresas, incluindo escritórios comerciais, educação, saúde, ciências vivas, data-centers, transportes, industria e tecnologia, Governo e Militar.

2.2.1 CONTROLADORES TAC XENTA®

Existe uma grande variedade de controladores **TAC Xenta®**, para os mais variados tipos de construção, a seguir apresentamos os controladores disponibilizados pela **TAC Xenta®**.

2.2.1.1 TAC XENTA 100 ®

O **TAC Xenta 100®** [TAC100] consiste num controlador específico que foi desenhado para uma área de aplicações tais como controlo de salas, tectos refrigerados e ventoinhas. Todos os controladores podem ser integrados com outros através da rede LonWorks®, uma indústria standard para comunicações entre redes que permite a integração de diferentes sistemas (será visto com maior pormenor na secção 2.3). O

TAC Xenta 100® providencia uma arquitectura aberta, a prova do futuro. Ao mesmo tempo, providencia um acesso a tecnologia de rede standard suportando um controlo flexível.

2.2.1.2 TAC XENTA 121 ®

O versátil controlador **TAC Xenta 121®** [TAC121] é focado para zonas específicas tais como ventoinhas ou bombas de calor, Esta nova família de controladores são uma parte do TAC Vista Building Management System providenciando uma eficiente gestão quer duma simples construção, como por exemplo uma pequena parede, quer de uma construção grande e complexa como uma ponte.

2.2.1.3 TAC XENTA 300®

O **TAC Xenta 300®** [TAC300] é um controlador programável certificado pela LonMark®, feito para controlar sistemas de pequeno e médio tamanho de calor, ventilação e ar condicionado. O TAC Xenta 300® é desenhado para ser usado em sistemas abertos e integrados via LonWorks®. O **TAC Xenta 300®** providencia uma arquitectura de sistema aberta a prova de futuro, ou seja está previsto funcionar com o passar do tempo. Ao mesmo tempo providencia um acesso a uma rede standard, suportando um sistema de controlo flexível, para cada um dos componentes de outros fabricantes que sejam conectados.

2.2.1.4 TAC XENTA 400®

O controlador **TAC Xenta 400®** [TAC400] é uma unidade programável, certificada pela LonMark® [LNW2008] indicada para o controlo de temperatura, ventilação e sistemas de ar condicionado ou como unidade de controlo num controlo aberto e sistema de supervisionamento. O **TAC Xenta 400®** foi desenvolvido para ser integrado via LonWorks® para permitir que diferentes sistemas tais como **HVAC**[1], iluminação e controlo de acessos, possam ser integrados na mesma rede. O **TAC Xenta 400®** providencia um sistema aberto, a prova do futuro. Ao mesmo tempo providencia um acesso a uma rede standard, suportando um sistema de controlo flexível, para cada um dos componentes de outros fabricantes que sejam conectados.

---

[1] HVAC - Heating, Ventilating, and Air Conditioning

2.2.1.5 TAC XENTA 401®

O **TAC Xenta 401®** [TAC401] pertence a família de controladores programáveis livres, com recursos de comunicações desenhados para sistemas HVAC. **O TAC Xenta 401®** possui todas as funcionalidades dos sistemas HVAC, incluindo controlo de loops curvas, controlo de tempo e alarmes. Este tipo de controladores não possui módulos de entradas ou saídas. Em vez disso os módulos de E/S necessários, são usados a partir do controlador **TAC Xenta 400®.**

2.2.1.6 TAC XENTA 511®

Utilizando o **TAC Xenta 511®** [TAC511-B] podemos controlar e monitorizar dinamicamente o estado do sistema através de um simples e intuitivo standard Web Browser.

Dependentes da autorização, utilizadores conectados podem fazer alterações parâmetros opcionais e set-points, bem como ver os alarmes conhecidos. Utilizadores podem também ler a documentação específica do sistema tais com descrição de funções, documentos técnicos e relatórios de serviço.

2.2.1.7 TAC XENTA 42XA® E TAC XENTA 45XA ®

O **TAC Xenta 421A/422A®** e o **TAC Xenta 451A/452A®** estendem um serie de módulos de E/S, oferecendo entradas 100% universais. Estes foram desenhados para ser usados com os controladores programáveis **TAC Xenta 300®** e **TAC Xenta 401®.** [T420/450].

2.2.1.8 TAC XENTA 527®

O **TAC Xenta 527®** providencia uma motorização livre do sistema a partir de qualquer localização com acesso a internet. Isto pode ser em qualquer lugar da construção, num campus, numa cidade, ou qualquer lugar no mundo. Utilizando uma descoberta automática da rede, podemos estar prontos para a monitorização numa questão de alguns minutos após a instalação. Temos também a flexibilidade de criar apresentações gráficas específicas para cada operador. [TAC400].

2.2.1.9 TAC XENTA 700®

A serie **TAC Xenta 700®** [TAC700] foi o primeiro controlador automático construído, que combinava o controlo das construções, funcionalidades Web, controlo de alarmes e gráficos espantosos, num só pacote poderoso e compacto, Esta solução

compacta é tudo o que é necessário para monitorizar e controlar aplicações num único dispositivo económico.

2.2.1.10 TAC XENTA 911®

O **TAC Xenta 911®** [TAC911] é um dispositivo de comunicação Ethernet que permite expandir uma rede LonWorks sobre TCP/IP. O **TAC Xenta 911®** é uma solução de custo efectivo para quem conectar um sistema baseado em redes LonWorks® a uma infra-estrutura, minimizando uma grande parte dos custos de instalação dos sistemas.

2.2.1.11 TAC XENTA 913®

O **TAC Xenta 913®** [TAC913] é um gateway de custo fixo que permite uma integração com uma grade variedade de produtos numa rede TAC. Permite a construção de um sistema integrado com uma única interface de operador com todas as funcionalidades ao nível de rede. O **TAC Xenta 913®** suporta a maioria dos protocolos abertos usados, tais como LonWorks® e BACnet. O **TAC Xenta 913®** transfere valores através de redes suportadas.

2.2.1.12 TABELA RESUMO

Os controladores **TAC Xenta®** foram desenhados para os mais diferentes tipos de funções. A tabela (Tabela 1) seguinte mostra um resumo dos controladores descritos anteriormente.

TABELA 1 – RESUMO DOS CONTROLADORES TAC XENTA®

Nome	Módulos E/S	Função	Suporte Lonworks®
TAC Xenta 100®	SIM	Controlo de salas, tectos refrigerados, ventoinhas	SIM
TAC Xenta 121®	SIM	Ventoinhas, bombas de calor	SIM
TAC Xenta 300®	SIM	Controlo de temperatura e supervisionamento	SIM
TAC Xenta 400®	SIM	Controlo de temperatura, ventilação, ar condicionado	SIM

TAC Xenta 401®	NÃO (Usa os módulos de E/S do TAC Xenta 400)	Sistemas HVAC	SIM
TAC Xenta 420/450A®	SIM	Usados para controlar os dispositivos TAC Xenta 300® e TAC Xenta 401®	SIM
TAC Xenta 511®	SIM	Controlar e monitorizar dinamicamente o sistema	SIM
TAC Xenta 527®	SIM	Monitorização do sistema remotamente	SIM
TAC Xenta 700®	SIM	Controlo de alarmes, visualização de gráficos	SIM
TAC Xenta 911®	SIM	Permite expandir uma rede Lonworks® através da comunicação Ethernet	SIM
TAC Xenta 913®	SIM	É um gateway que permite a integração com outros dispositivos TAC Xenta®	SIM

2.2.2 TAC VISTA®

O **TAC Vista®** [TACSO] é uma solução de software que controla, verifica e analisa economicamente as operações diárias. O **TAC Vista®** está disponível numa variedade de pacotes desenhados para maximizar a eficiência e economia. O **TAC Vista®** é também modular, fazendo com que seja fácil a expansão deste sistema quando seja necessário. O **TAC Vista®** está disponível numa variedade de idiomas.

A Figura 1 mostra a arquitectura do sistema **TAC Vista**, como os diversos componentes se relacionam entre si.

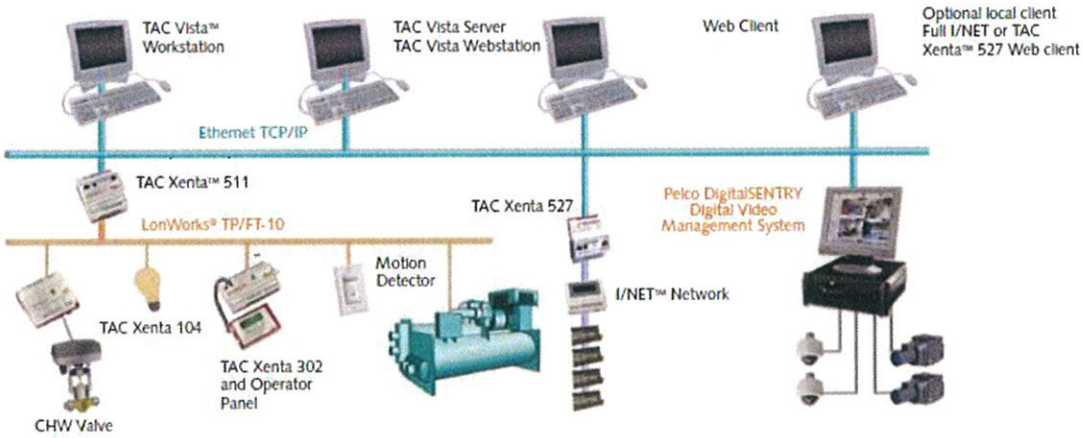


FIGURA 1 – ARQUITECTURA DO SISTEMA TAC VISTA

As Figuras 2 e 3 mostram funcionalidades do **TAC Vista**, quer no aspecto de controlo e monitorização quer no aspecto de análise.

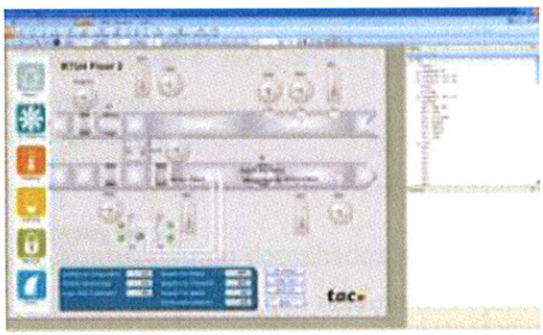


FIGURA 2 – MONITORIZAÇÃO DO MODO DE OPERAÇÃO DA CONSTRUÇÃO

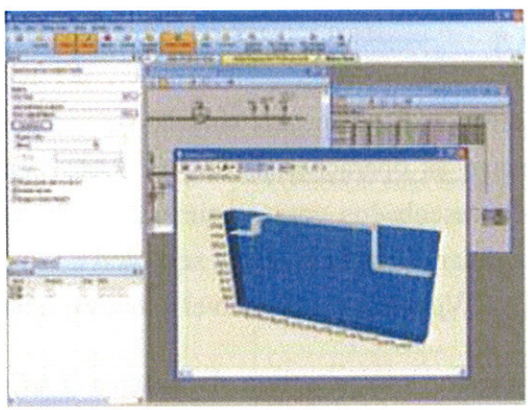


FIGURA 3 – ANÁLISE PARA MELHORAR A PERFORMANCE

2.2.3 TAC VISTA SERVER E WORKSTATION

O **TAC Vista Server** [TACSO] providencia um acesso ao ambiente e controlos de segurança para o operador das workstations, e é a interface primaria do operador para o controlo do sistema. Este mostra as operações diárias através de uma interface gráfica de utilização, providenciando aos operadores acesso aos alarmes, logs de histórico, dados sofisticados, bem com relatórios.

2.2.4 TAC VISTA WEBSTATION

O **TAC Vista Webstation** [TACSO] permite aceder aos controlos do sistema usando um comum Web Browser. Os utilizadores podem navegar no site, ver gráficos, tendências de gráficos, gestão de alarmes. O **TAC Vista Webstation** providencia o acesso ao histórico dos eventos no sistema e o **TAC Vista Webstation Server** providencia o acesso periódico a relatórios automáticos. Utilizando qualquer Web browser, os utilizadores podem navegar no site, ver gráficos, ver tendências e gerir alarmes.

2.2.5 TAC VISTA SCREENMATE

A principal tarefa do **TAC Vista ScreenMate** [TACSO] é substituir as funcionalidades encontradas nos termóstatòs sofisticados. O **TAC Vista ScreenMate** faz com que seja possível os utilizadores ver e alterar as preferências pessoais tais como temperatura e set-point. Para ver a temperatura do ar directamente do computador do utilizador. A solução ScreenMate é baseada em tecnologias web standards e podem ser acedidas a partir de qualquer dispositivo cliente com um Web browser.

2.2.6 TAC MENTA ™

O **TAC Menta™** é uma ferramenta de programação para os controladores **TAC Xenta™**. Reduz o tempo de execução e aumenta fiabilidade das operações com esta ferramenta para aplicações HVAC. O **TAC Menta** providencia blocos de funções pré-programadas e elementos básicos, monitoriza simulações offline e testes online com uma integração de logs de tendências. [TACSO].

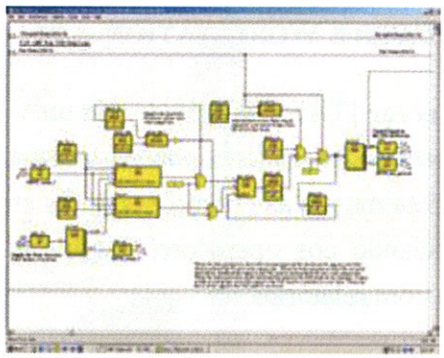


FIGURA 4 - TAC MENTA

2.2 PROTOCOLOS DE COMUNICAÇÃO

O uso difundido e a expansão dos protocolos de comunicação são ao mesmo tempo um pré-requisito e uma contribuição para o sucesso da Internet. O **TCP/IP** é uma referência ao conjunto dos protocolos mais utilizados.

Geralmente apenas os protocolos mais simples são utilizados sozinhos. A maioria dos protocolos, especialmente no contexto da comunicação, são agrupados em pilhas de protocolo, onde as diferentes tarefas que constituem uma comunicação são executadas por níveis especializados da pilha.

Enquanto uma pilha de protocolos denota uma combinação específica de protocolos que trabalham conjuntamente, um modelo de referência é uma arquitectura de software que mostra cada um dos níveis e os serviços que cada um pode oferecer. O modelo clássico OSI, em sete níveis, é utilizado para conceptualizar pilhas de protocolo.

2.2.1 PROTOCOLO LONWORKS®

O protocolo **Lonworks®** [LW2008] é um protocolo de rede especificamente desenhado para obter grandes desempenhos e fiabilidade de aplicações de controlo. Este protocolo é construído sobre um protocolo de baixa largura de banda criado pela empresa Echelon [EC2008], na década de 90, para controlar dispositivos ligados por redes de par entrançado, transmissão de dados através de redes eléctricas, e fibras ópticas. É muito utilizado em sistemas HVAC.

2.3 SISTEMAS GESTORES DE BASES DE DADOS

Um **Sistema Gestor de Bases de Dados (SGBD)**, é um conjunto complexo de programas que controlam a organização, o armazenamento, gestão e obtenção dos dados numa base de dados. OS **SGBDs** estão categorizados de acordo com a sua estrutura de

dados e tipos (anteriormente também eram vistos como Database Manager). È um conjunto de programas pré-escritos que são usados para guardar, obter e alterar os dados num base de dados.

Um **SGBD** inclui uma linguagem de modelação que define o esquema para cada base de dados, estrutura de dados (**campos, ficheiros, objectos etc.**), uma linguagem para fazer perguntas à base de dados (**query**), e um mecanismo de transacção para garantir a integridade dos dados, concorrência e falhas.

2.3.1 MySQL®

O **MySQL** [WSQL2008] é um sistema gestor de bases de dados (SGBD), que utiliza a linguagem **SQL** (Structured Query Language) como interface. É actualmente uma das bases de dados mais populares, com mais de 10 milhões de instalações pelo mundo. [WSQL2008].

É usada em grandes empresas tais como: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S Army, US. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems.

O **MySQL** foi criado na Suécia por dois suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que têm trabalhado juntos desde a década de 1980. Hoje para o seu desenvolvimento e manutenção empregam aproximadamente 70 profissionais no mundo inteiro, e mais de mil contribuem testando o software, integrando-o com outros produtos e documentando.

No dia 16 de Janeiro de 2008, a **MySQL AB**, a empresa criadora do **MySQL** foi adquirida pela **Sun Microsystems**, por mil milhões de dólares, um preço nunca visto no sector das licenças livres.

O sucesso do **MySQL** deve-se em grande parte à fácil integração com o **PHP**. Empresas como Yahoo! Finance, MP3.com, Motorola, NASA, Silicon Graphics e Texas Instruments usam o **MySQL** em aplicações críticas [CSQL2008].

Podemos ver um exemplo de código MySQL neste caso um exemplo de criação de uma tabela

```
CREATE
  TABLE tr_users(
    id_user INT(11) NOT NULL auto_increment
    ,username CHAR(50) NOT NULL DEFAULT ''
    ,pass CHAR(100) NOT NULL DEFAULT ''
    ,firstname CHAR(50) NOT NULL DEFAULT ''
```

```
    ,lastname CHAR(50) NOT NULL DEFAULT ''
    ,id_usertype INT(11) NOT NULL DEFAULT '0'
    ,borndate DATE NOT NULL DEFAULT '0000-00-00'
    ,country CHAR(50) NOT NULL DEFAULT ''
    ,city CHAR(50) NOT NULL DEFAULT ''
    ,zipcode CHAR(50) NOT NULL DEFAULT ''
    ,address CHAR(50) NOT NULL DEFAULT ''
    ,phone CHAR(20) NOT NULL DEFAULT ''
    ,email CHAR(50) NOT NULL DEFAULT ''
    ,webpage CHAR(50) NOT NULL DEFAULT ''
    ,m_msn CHAR(50) NOT NULL DEFAULT ''
    ,y_msn CHAR(50) NOT NULL DEFAULT ''
    ,skype CHAR(50) NOT NULL DEFAULT ''
    ,usergroup CHAR(24) NOT NULL DEFAULT '0'
    ,PRIMARY KEY (id_user)
  ) ENGINE = MyISAM AUTO_INCREMENT = 12 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
```

2.4 LINGUAGENS DE PROGRAMAÇÃO

Desde o aparecimento dos primeiros computadores que ficou claro que seria necessário encontrar um método standard para melhor expressar as instruções que deveriam ser dadas ao computador. Um conjunto de regras sintácticas e semânticas que permitissem definir um programa, passível de ser interpretado pela máquina da forma mais eficiente. Surgem assim as linguagens de programação, que nada mais são do que um conjunto de palavras, compostas segundo essas regras, e que permitem que um programador especifique precisamente sobre quais os dados que a máquina vai actuar, como serão armazenados ou transmitidos e quais as acções que devem ser tomadas sob várias circunstâncias.

Como será fácil compreender, um dos principais objectivos das linguagens de programação é permitir que os programadores possuam uma maior produtividade, criando condições para que estes transmitam mais facilmente e rapidamente as suas intenções à máquina. À medida que caminhamos rumo ao século XXI, o universo programático sofreu diversas evoluções, onde muitas linguagens desadequadas à nova realidade computacional deixaram de existir, cedendo lugar e novas formas de comunicação neste eterno diálogo homem/máquina. Na figura 5 podemos ver essa evolução

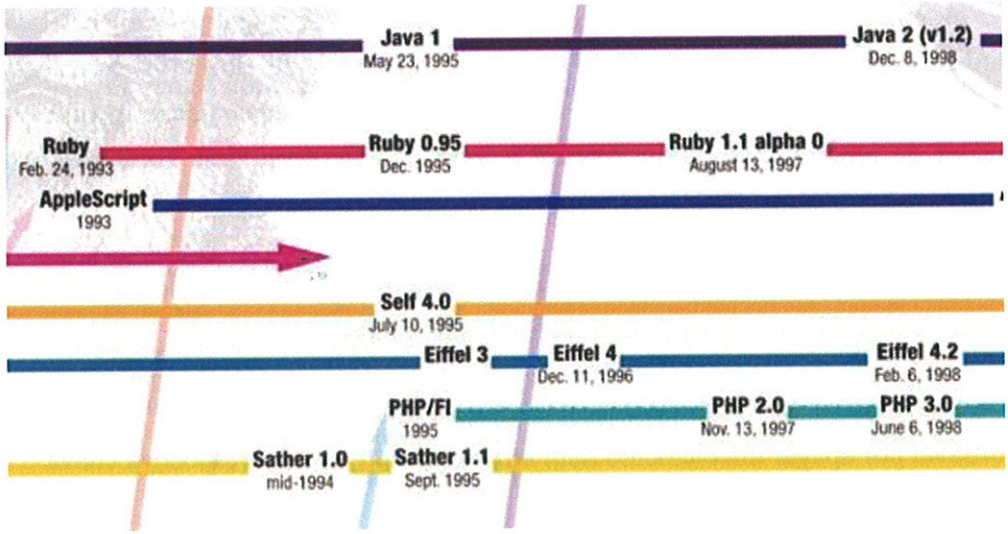


FIGURA 5 - EVOLUÇÃO DAS LINGUAGENS DE PROGRAMAÇÃO [HPL2008]

Nos pontos seguintes vamos mostrar algumas características das linguagens escolhidas para realizar o projecto

2.4.1 PHP

O **PHP** [HISPHP] começou e ainda é usado principalmente como uma linguagem de scripting do lado do servidor embutida no HTML.

PHP, conhecido originalmente como Personal Home Pages, foi primeiramente concebido em 1994 por Rasmus Lerdorf. Esta tinha o intuito de rastrear os visitantes de seu CV online. A primeira versão foi realizada no início de 1995, época na qual Rasmus decidiu tornar seu projecto opensource, sendo assim, as pessoas corrigiriam seus bugs. A primeira versão era muito directa e tinha um parser simples que reconhecia algumas macros especiais e continha algumas utilidades que eram usualmente usadas em páginas Web daquela época.

O parser foi reescrito em meados de 1995 e renomeado para PHP/FI version 2. O "FI" desta versão significava Form Interpreter, Rasmus então adicionou ao **PHP** mais uma série de funções que cobriam as necessidades das páginas Web daquela época. PHP/FI então começou a ter um crescimento massivo, e outras pessoas começaram a contribuir com o código regularmente.

Em meados de 1997 Zeev Suraski e Andi Gutmans reescreveram o parser principal, e o **PHP** deixou de estar sob a tutela de Rasmus' para estar sob a orientação de um grupo. Com isto a formou-se base para o **PHP3**, agora chamado de **PHP**: Hypertext Preprocessor - um acrónimo recursivo.

A versão 4, o **PHP4**, é outra reescrita feita por Suraski e Gutmans e é baseada no mecanismo (tecnologia) Zend. O **PHP** agora tem mais de duzentos contribuintes regulares trabalhando em várias partes do projecto. Tem uma quantidade gigantesca de extensões de terceiros, módulos, suporte a todo tipo de servidores populares, e tem suporte para **MySQL** e **ODBC** embutido.

A última estatística mostra que o **PHP** agora é usado por mais de 5,5 milhões de domínios, e tem tido uma taxa de uso crescente ano pós ano.

Podemos ver um pequeno exemplo de uma função criada em PHP, esta função e responsável por apagar um elemento num array associativo e sua respectiva chave

```
<?php
  function array_delete(&$ary,$key_to_be_deleted){
    $new = array();
    if(is_string($key_to_be_deleted)) {
      if(!array_key_exists($key_to_be_deleted,$ary)) {
        return;
      }
      foreach($ary as $key => $value) {
        if($key != $key_to_be_deleted) {
          $new[$key] = $value;
        }
      }
      $ary = $new;
    }
    if(is_array($key_to_be_deleted)) {
      foreach($key_to_be_deleted as $del) {
        array_delete(&$ary,$del);
      }
    }
  }
?>
```

2.4.2 FLEX

O Adobe **Flex** [FLX2008] (antes chamado de Macromedia) é o nome de uma tecnologia lançada em Março de 2004 pela Macromedia, que suporta o desenvolvimento de aplicações ricas para a Internet, baseadas na plataforma do Macromedia Flash. A versão inicial possuía um SDK, um IDE uma integração com o J2EE também conhecido como **Flex Data Services**. Desde que a Adobe adquiriu a Macromedia em 2005, as versões seguintes do **Flex** começaram a requerer uma licença para o Flex Data Services, que era inicialmente um produto separado e que posteriormente foi rebaptizado como LiveCycle Data Services.

O **Flex** 3 incorpora novos recursos para a criação de **RIAs** para navegador e desktop, é uma tecnologia opensource

Podemos ver um pequeno excerto de código FLEX (MXML e Actionscript), neste caso podemos com dar capacidades à aplicação de arrastar elementos na aplicação (Drag and Drop)

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  width="365" height="225"
  creationComplete="creationCompleteHandler();">
  <mx:Script>
    <![CDATA[
        private function creationCompleteHandler():void{
          srclist.dataProvider = [
                                  'Reading',
                                  'Skating',
                                  'Movies'
                                ];
          destlist.dataProvider = [];
        }
    ]]>
  </mx:Script>
  <mx:Panel title="Select activities" layout="horizontal">
    <mx:VBox width="50%">
      <mx:Label text="Available activities"/>
      <!-- Drag initiator -->
      <mx:List
        id="srclist" width="100%" height="100"
        allowMultipleSelection="true"
        dragEnabled="true"/>
    </mx:VBox>
    <mx:VBox width="50%">
      <mx:Label text="Activities I enjoy"/>
      <!-- Drop target -->
      <mx:List
        id="destlist" width="100%" height="100"
        dropEnabled="true"/>
    </mx:VBox>
  </mx:Panel>
</mx:Application>
```

2.4.3 JAVA

Java [HISJAVA] é uma linguagem de programação objectos desenvolvida na década de 90 por uma equipa de programadores chefiada por James Gosling, na empresa Sun Microsystems. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é executado por uma máquina virtual. A linguagem de programação Java é a linguagem convencional da Plataforma Java, mas não a sua única linguagem.

Podemos ver um pequeno exemplo de código Java, que mediante uma entrada de texto separa esse texto em termos separados por virgulas.

```
import java.util.StringTokenizer;
```

```
public class TokenizerTest {
        public static void main(String[] args) {
                String s = "123, Jones, Bill";
                StringTokenizer t = new StringTokenizer(s, ",");
                while (t.hasMoreTokens()) {
                        System.out.println(t.nextToken());
                }
        }
}
```

2.4.4 JAVASCRIPT

JavaScript [HISJS] é uma linguagem de programação criada pela Netscape em 1995, que a princípio se chamava LiveScript, para atender, principalmente, as seguintes necessidades:

- Validação de formulários no lado cliente
- Interacção com a página. Assim, foi feita como uma linguagem de script. Javascript tem sintaxe semelhante à do Java, mas é totalmente diferente no conceito e no uso.

Esta linguagem oferece tipos dinâmicos, ou seja os tipos de variáveis não são definidos, é uma linguagem interpretada, em vez de compilada, possui boas ferramentas padrão para listagens (como as linguagens de script, de modo geral e oferece bom suporte a expressões regulares (característica também comum a linguagens de script).

A sua união com o **CSS** é conhecida como **DHTML**. Usando o **Javascript**, é possível modificar dinamicamente os estilos dos elementos da página em **HTML**.

Dada a sua enorme versatilidade e utilidade ao lidar com ambientes em árvore (como um documento HTML), foi criado a partir desta linguagem um padrão ECMA, o ECMA-262, também conhecido como **ECMAScript**. Este padrão é seguido, por exemplo, pela linguagem **ActionScript** da Adobe (Antigamente Macromedia, porém a empresa foi vendida à Adobe).

Podemos ver um excerto de código Javascript onde fazemos a validação de um formulário

```
<script language="javascript" type="text/javascript">
        function validateForm(contact){
                if ("" == document.forms.contact.fullName.value) {
                        alert("Please enter your full name.");
                        return false;
                }
                if ("" == document.forms.contact.email.value) {
                        alert("Please enter your email address.");
```

```
                return false;
            }
            if ("" == document.forms.contact.phoneNum.value) {
                alert("Please enter your phone number.");
                return false;
            }
        }
</script>
```

2.4.5 XML

O **XML** [HISXML] é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.

É um subtipo de SGML, capaz de descrever diversos tipos de dados. O seu propósito principal é facilitar a partilha de informações através da Internet. Existem diversas linguagens baseadas em XML tal como XHTML, RDF, SDMX ,SMIL, FLEX, MathML (formato para expressões matemáticas), NCL , XBRL , XSIL e SVG (formato gráfico vectorial).

Podemos ver um pequeno exemplo de código XML, que permite armazenar dados do currículo vitae de uma pessoa.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<curriculo>
  <InformacaoPessoal>
    <DataNascimento>23-07-68</DataNascimento>
    <Nomecompleto>...</Nomecompleto>
    <Contatos>
      <Morada>
        <Rua>R.Topazio</Rua>
        <Num>111</Num>
        <Cidade>nome_cidade</Cidade>
        <Pais>nome_país</Pais>
      </Morada>
      <Telefone>9999-9999</Telefone>
      <CorreioEletronico>email@email.com</CorreioEletronico>
    </Contatos>
    <Nacionalidade>brasileiro</Nacionalidade>
    <Sexo>M</Sexo>
  </InformacaoPessoal>
  <objetivo>Atuar na area de TI</objetivo>
  <Experiencia>
    <Cargo>Suporte tecnico</Cargo>
    <Empregador>Empresa, Cidade - Estado</Empregador>
  </Experiencia>
  <Formacao>Superior Completo</Formacao>
</curriculo>
```

2.5 SUMÁRIO

Neste capítulo podemos concluir que embora os dispositivos desenvolvidos pela **TAC**, sejam bons e robustos, não são escaláveis, devido principalmente ao facto destes controladores ser fechados, ou seja são bons controladores, com um vasto número de funcionalidades, mas caso seja necessário fazer uma outra coisa que não tenha sido prevista pelos seus construtores, e muito difícil pelo facto de ter o seu código fechado. Sendo assim este tipo de controladores só devem ser utilizados no caso das tarefas a efectuar serem coisas previstas, caso contrário deve se optar por outro tipo de solução.

3. Modelos e Metodologias

"Any intelligent fool can make things bigger and more complex... It takes a touch of genius - and a lot of courage to move in the opposite direction. "

Albert Einstein

Tópicos

3.1 INTRODUÇÃO

Neste capítulo apresentamos uma revisão bibliográfica de um conjunto de conceitos e definições. Começa-se com uma abordagem sobre a **Engenharia de Software**, introduz-se o conceito de **Processo de Software** enumerando as suas fases e explicando em que consiste cada uma delas. Introduzimos ainda os modelos do processo de software, enumerando e explicando alguns que considerei mais importantes, introduzimos o conceito de Protótipos Canónicos Abstractos (PAC) e finalmente apresentamos o paradigma orientado a objectos, onde mostramos os conceitos e vantagens da sua utilização, referindo as metodologias mais utilizadas.

3.2 ENGENHARIA DE SOFTWARE

A **Engenharia de Software** é uma área da informática que procura estruturar de uma forma racional e científica (através do uso de modelos matemáticos) a especificação, desenvolvimento e manutenção de sistemas de software aplicando tecnologias e métodos da Ciência da computação, gestão de projectos, das Engenharias e outros campos do conhecimento.

O termo **Engenharia de Software** foi criado na década de 1960 e utilizado oficialmente em 1968 na NATO (**NATO Software Engineering Conference** [NATO68]). A **Engenharia de Software** surgiu numa tentativa de contornar a crise do software e dar um tratamento de engenharia (mais sistemático e controlado) ao desenvolvimento de sistemas de software complexos. Um sistema de software complexo caracteriza-se por um conjunto de componentes abstractos de software (estruturas de dados e algoritmos) encapsulados na forma de procedimentos, funções, módulos, objectos ou agentes ligados entre si, compondo a arquitectura do software, que por sua vez deverão ser executados em sistemas computacionais.

Segundo **Friedrich Ludwig Bauer** [NATO68]., "Engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira económica, que seja confiável e que trabalhe eficientemente em máquinas reais".

A **Engenharia de Software** focaliza-se nos aspectos práticos da produção de um sistema de software, enquanto a ciência da computação estuda os fundamentos teóricos dos aspectos computacionais.

3.3 PROCESSOS DE SOFTWARE

Um **Processo de desenvolvimento de software** é um conjunto de actividades, parcialmente ordenadas, com a finalidade de obter um produto de software. É estudado dentro da área de Engenharia de Software, sendo considerado um dos principais mecanismos para se obter software de qualidade e cumprir correctamente os contratos de desenvolvimento.

Foi uma das respostas para resolver a **Crise do software** [EWD340]. A **crise do software** foi um termo utilizado nos anos 70, quando a engenharia de software era praticamente inexistente. O termo expressava as dificuldades do desenvolvimento de software frente ao rápido crescimento da necessidade de software, da complexidade dos problemas a serem resolvidos e da inexistência de técnicas estabelecidas para o desenvolvimento de sistemas que funcionassem adequadamente ou pudessem ser validados.

As causas da crise do software estão ligadas a complexidade do processo de software e a relativa imaturidade da engenharia de software como profissão. A crise pode se manifestar de várias maneiras [EWD340]:

- Projectos muito acima do orçamento previsto.
- Projectos muito acima do prazo previsto.
- Software de baixa qualidade.
- Softwares muitas vezes não atingem os requisitos.
- Projectos difíceis de gerir e o código difícil de manter.

Consequência dos pontos anteriores, a maior parte dos projectos continuam com estes problemas ainda na actualidade, assim pode se dizer que a crise continua na actualidade.

3.3.1 FASES DO PROCESSO DE SOFTWARE

Existe diversos processos de software, com diferentes fases, mas segundo Schwartz [SCH75] as principais fases de um processo de software são:

1. **Especificação de Requisitos:** tradução da necessidade ou requisito operacional para uma descrição da funcionalidade a ser executada.

2. **Projecto de Sistema:** tradução destes requisitos em uma descrição de todos os componentes necessários para codificar o sistema.

3. **Programação (Codificação):** produção do código que controla o sistema e realiza a computação e lógica envolvida.

4. **Verificação e Integração:** verificação da satisfação dos requisitos iniciais pelo produto produzido.

Ao contrário do que possa parecer não existe uma sequência obrigatória das fases, sendo que diversos autores apontam a natureza não simultânea das fases como uma realidade na aplicação de processos de software, e também defendem que o processo de software é muito mais iterativo e cíclico do que a ideia de fases simples pode sugerir. [CHR2001]

3.4 MODELOS DO PROCESSO DE SOFTWARE

Os **modelos do processos de desenvolvimento de software** surgiram pela necessidade de dar resposta às situações a analisar, porque só na altura em que enfrentamos o problema é que podemos escolher o modelo.

Nos **modelos de processo de software** é dado uma atenção especial à representação abstracta dos elementos do processo e à sua dinâmica, não estabelecendo métodos de desenvolvimento, pois este trabalha num nível mais alto de abstracção do que os modelos de ciclo de vida.

A seguir descrevemos os principais modelos:

3.4.1 O MODELO EM CASCATA

Modelo idealizado por **Royce** em 1970 [WWR1970], também conhecido como abordagem **"top-down"**, tem como principal característica a sequência de actividades onde cada fase transcorre completamente e os seus produtos são vistos como entrada para uma nova fase. Sofreu diversos ajustes e melhoramentos sendo muito utilizado nos dias actuais. A figura 6 exemplifica o modelo em cascata.

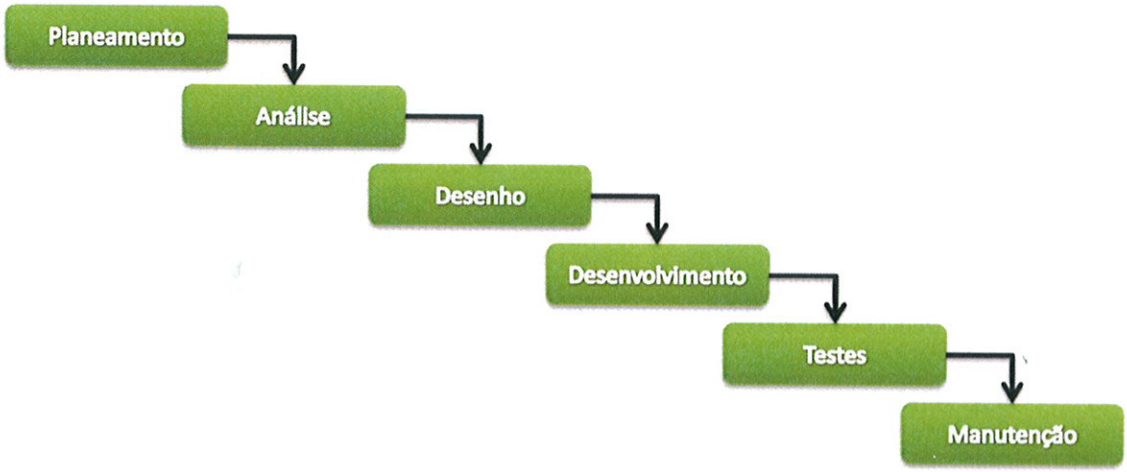


FIGURA 6 – MODELO EM CASCATA

A ideia principal do modelo é que as diferentes etapas de desenvolvimento sigam uma sequência. As actividades a executar são agrupadas em tarefas, executadas sequencialmente, de forma que uma tarefa só poderá ter início quando a anterior tiver terminado. Uma das vantagens do modelo é que só avança para a tarefa seguinte quando o cliente valida e aceita os produtos finais da tarefa actual.

O modelo pressupõe que o cliente participa activamente no projecto e que sabe muito bem o que quer. Este modelo minimiza o impacto da compreensão adquirida no decurso de um projecto, uma vez que se um processo não pode voltar atrás de modo a alterar os modelos e as conclusões das tarefas anteriores, é normal que as novas ideias sobre o sistema não sejam aproveitadas. Numa tentativa de resolver este tipo de problema foi definido um novo tipo de processo baseado no clássico em cascata, designado por modelo em cascata revisto como podemos ver na figura 7, cuja principal diferença consiste em prever a possibilidade de a partir de qualquer tarefa do ciclo se poder regressar a uma tarefa anterior de forma a contemplar alterações funcionais e/ou técnicas que entretanto tenham surgido, em virtude de um maior conhecimento que entretanto se tenha obtido. O risco desta abordagem é que, na ausência de um processo de gestão do projecto e de controlo das alterações bem definido, podemos passar o

tempo num ciclo infinito, sem nunca se atingir o objectivo final, ou seja disponibilizar o sistema a funcionar.

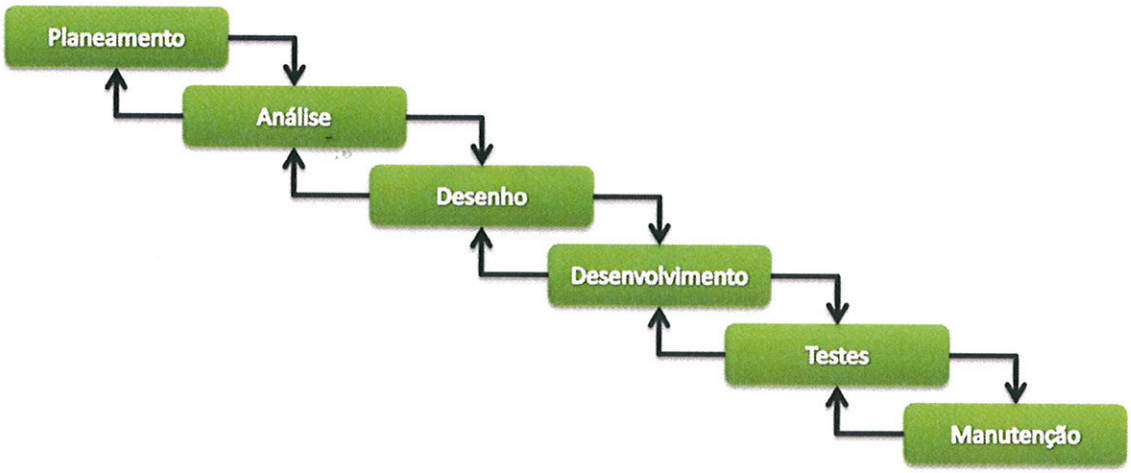


FIGURA 7 – MODELO EM CASCATA REVISTO

3.4.2 O MODELO ESPIRAL

Neste modelo o projecto é desenvolvido como uma série de pequenos ciclos, cada um finalizando uma versão de um software executável.

O modelo em **espiral** foi proposto por **Barry Boehm** em 1988 [BB1988] como forma de integrar os diversos modelos existentes à época, eliminando suas dificuldades e explorando seus pontos fortes.

Este modelo foi desenvolvido para abranger as melhores características tanto do ciclo de vida clássico como da prototipagem, acrescentando, ao mesmo tempo, um novo elemento - a análise de riscos - que falta a esses paradigmas.

Entretanto a integração não se dá através da simples incorporação de características dos modelos anteriores. O modelo em espiral assume que o processo de desenvolvimento ocorre em ciclos, cada um contendo fases de avaliação e planeamento, onde a opção de abordagem para a próxima fase (ou ciclo) é determinada. Estas opções podem acomodar características de outros modelos. A figura 8 mostra o modelo espiral.

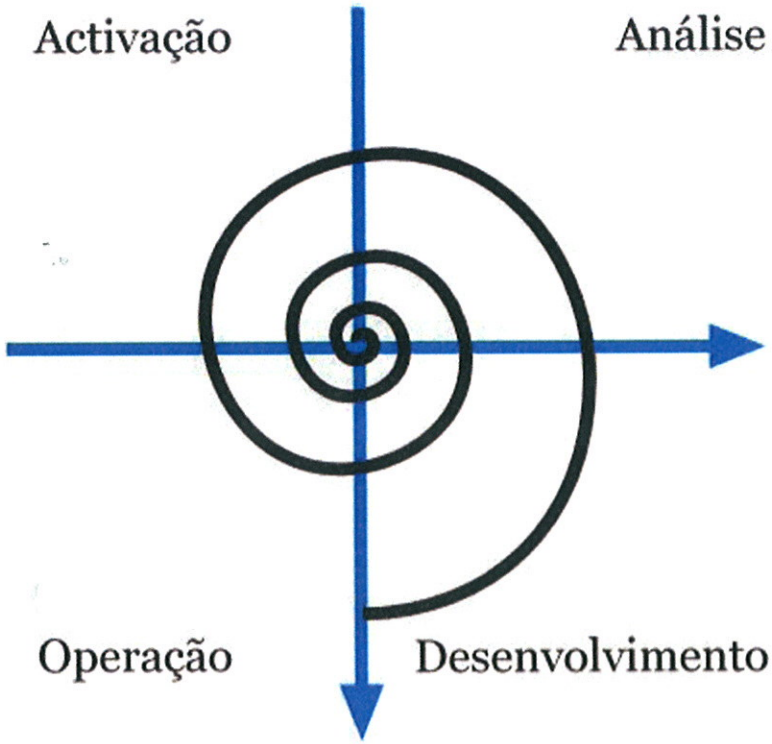


FIGURA 8 – MODELO ESPIRAL

O modelo original em **espiral** organiza o desenvolvimento como um processo iterativo em que vários conjuntos de quatro fases se sucedem até se obter o sistema final. Um ciclo inicia-se com a determinação dos objectivos, alternativas e restrições (primeira tarefa) onde ocorre o comprometimento dos envolvidos e o estabelecimento de uma estratégia para alcançar os objectivos.

Na segunda tarefa, análise e avaliação de alternativas, identificação e solução de riscos, executa-se uma análise de risco. A prototipagem é uma boa ferramenta para tratar riscos. Se o risco for considerado inaceitável, pode-se parar o projecto.

Na terceira tarefa ocorre o desenvolvimento do produto. Neste quadrante pode-se considerar o modelo cascata.

Na quarta tarefa o produto é avaliado e se prepara para iniciar um novo ciclo.

Variações do modelo espiral consideram entre três e seis tarefas ou sectores da espiral, que podem ser:

1. Comunicação com o cliente.

2. Planeamento

3. Análise de risco

4. Engenharia

5. Construção e liberação

6. Avaliação do cliente.

O modelo **espiral** é, actualmente a abordagem mais realística para desenvolvimento de software em grande escala, e usa uma abordagem que capacita a empresa que presta o serviço, e o cliente a entender e reagir aos riscos em cada etapa evolutiva. Este tipo de modelo exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso, pode ser difícil convencer os clientes que uma abordagem evolutiva é controlável.

3.4.3 O MODELO ITERATIVO E INCREMENTAL

Este modelo é uma extensão do modelo espiral sendo porém mais formal e rigoroso [SOMM95].

O desenvolvimento de um produto comercial de software é uma grande tarefa que pode ser estendida por vários meses, possivelmente um ano ou mais. Por isso, é mais prático dividir o trabalho em partes menores ou iterações. Cada iteração resultará num incremento.

Iterações são passos em fluxo de trabalho e incrementos são crescimentos do produto.

O princípio subjacente ao **processo incremental e iterativo** é que a equipa envolvida possa refinar e alargar paulatinamente a qualidade, detalhe e âmbito do sistema envolvido. Por exemplo, numa primeira iteração deve-se identificar a visão global e determinar a viabilidade económica do sistema, efectuar a maior parte da análise, desenho e implementação. Numa segunda iteração, deve-se concluir a análise, fazer uma parte significativa do desenho e um pouco mais de implementação. Numa terceira iteração, deve-se concluir o desenho, fazer-se parte substancial da implementação, testar e integrar um pouco, etc. Ou seja, a principal consequência da aproximação iterativa é que os produtos finais de todo o processo vão sendo amadurecidos e sendo completados ao longo do tempo, mas cada iteração produz sempre um conjunto de produtos finais.

A cada iteração são realizadas as seguintes tarefas:

- **Análise** (refinamento de requisitos, refinamento do modelo conceitual)
- **Projecto** (refinamento da arquitectura do projecto, projecto de baixo nível)
- **Implementação** (codificação e testes)
- **Transição para produto** (documentação, instalação)

A figura 9 mostra o modelo iterativo e incremental

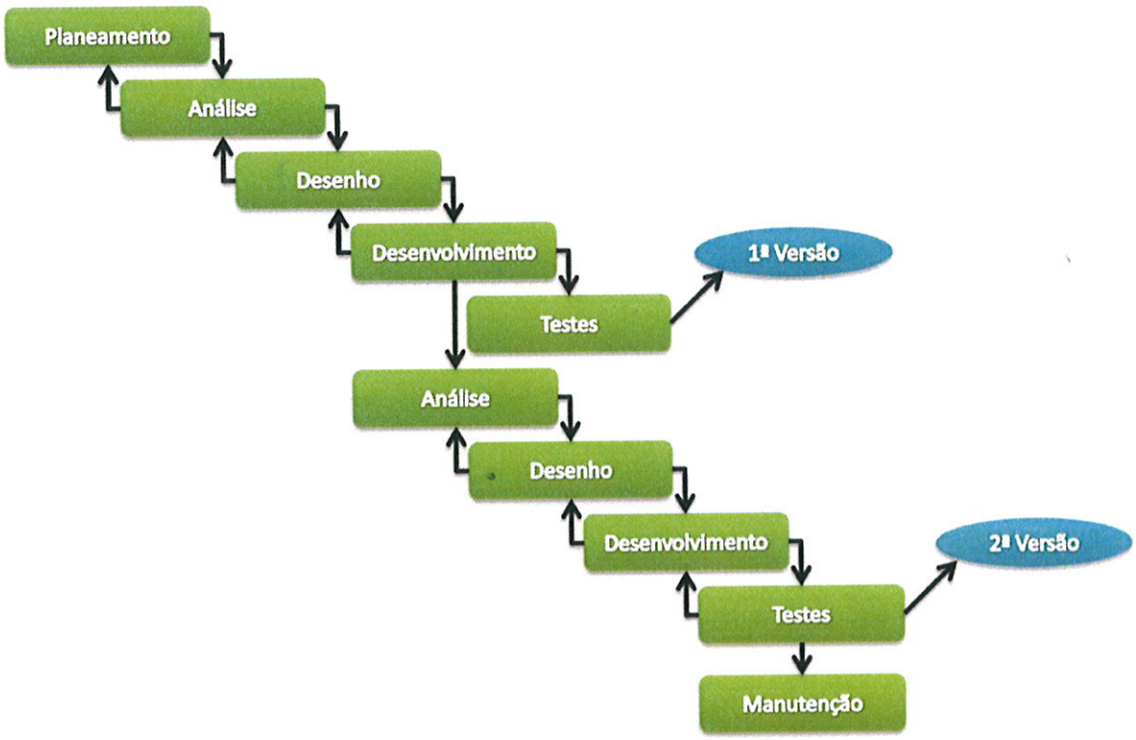


FIGURA 9 – MODELO ITERATIVO E INCREMENTAL

3.4.4 O MODELO DE PROTOTIPAGEM

O modelo de desenvolvimento baseado na prototipagem procura suprir duas grandes limitações do modelo cascata. De acordo com **Jalote** [PJALO97] a ideia básica deste modelo é que em vez de manter inalterado os requisitos durante o projecto e codificação, um protótipo é desenvolvido para ajudar no entendimento dos requisitos. Este desenvolvimento passa por um projecto, codificação e teste, sendo que cada uma destas fases não é executada formalmente. Usando assim os protótipos o cliente pode entender melhor os requisitos do sistema.

A sequência de eventos deste modelo está explicada na figura 10:

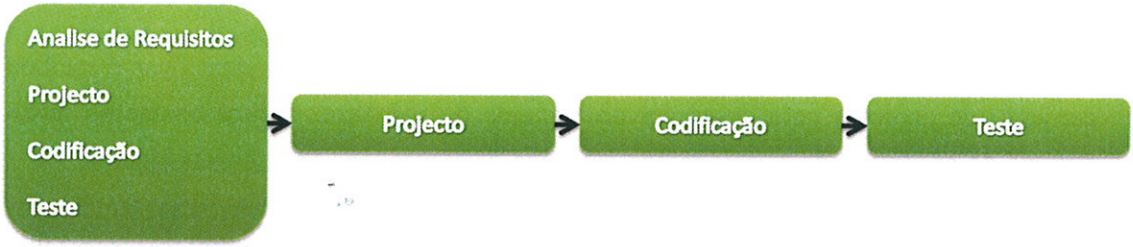


FIGURA 10 – MODELO DE PROTOTIPAGEM

O protótipo é desenvolvido com uma versão inicial do documento de especificação dos requisitos. Depois de estar pronto o cliente utiliza-o, e baseado na avaliação do cliente é fornecido o que precisa ser alterado, o que esta faltando e o que não é preciso. O protótipo é então modificado incorporando as sugestões de mudança e o cliente usa o protótipo novamente repetindo o processo até que o mesmo seja válido em termos de custo e tempo. No final os requisitos iniciais são alterados para produzir a especificação final dos requisitos.

Segundo **Pressman** [PMAN97] e **Jalote** [PJALO97] este modelo pode trazer os seguintes benefícios:

- O modelo é interessante para alguns sistemas de grande porte nos quais representem um certo grau de dificuldade para exprimir rigorosamente os requisitos
- È possível obter uma verão do que será o sistema com um pequeno investimento inicial
- A experiência de produzir o protótipo pode reduzir o custo das fases posteriores
- A construção do protótipo pode demonstrar a viabilidade do sistema.

3.5 PROGRAMAÇÃO ORIENTADA POR OBJECTOS

O **paradigma orientado a objectos** [SUNOOP], também conhecida como **Programação Orientada a Objectos (POO)** ou ainda em inglês **Object-Oriented Programming (OOP)** é um paradigma de análise, projecto e programação de sistemas de software baseado na composição e interacção entre diversas unidades de software chamadas de objectos.

Em alguns contextos, prefere-se usar modelação orientada a objectos, em vez de programação.

A análise e projecto orientados a objectos têm como meta identificar o melhor conjunto de objectos para descrever um sistema de software. O funcionamento deste sistema dá-se através do relacionamento e troca de mensagens entre estes objectos.

Na programação orientada a objectos, implementa-se um conjunto de classes que definem os objectos presentes no sistema de software. Cada classe determina o comportamento (definidos nos métodos) e estados possíveis (atributos) dos seus objectos, assim como o relacionamento com outros objectos.

Smalltalk, **Python Ruby**, **C++**, **Object Pascal**, **Java** e **C#** são alguns exemplos de linguagens de programação orientadas a objectos.

3.5.1 CONCEITOS FUNDAMENTAIS

Vamos agora mostrar alguns conceitos fundamentais da programação orientada por objectos [SUNOOP],

- **Classe:** Representa um conjunto de objectos com características afins. Uma classe define o comportamento dos objectos, através de métodos, e quais estados ele é capaz de manter, através de atributos.

- **Objecto:** é uma instância de uma classe. Um objecto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objectos.

- **Atributos:** São características de um objecto. Basicamente a estrutura de dados que vai representar a classe.

- **Métodos:** Definem as habilidades dos objectos. A acção só ocorre quando o método é invocado através do objecto. Normalmente, uma classe possui diversos métodos.

- **Mensagem:** É uma chamada a um objecto para invocar um de seus métodos, activando um comportamento descrito por sua classe. Também pode ser direccionada directamente a uma classe (através de uma invocação a um método estático).

- **Herança (ou generalização):** É o mecanismo pelo qual uma classe (subclasse) pode estender outra classe (superclasse), aproveitando os seus métodos e atributos. Existe Herança múltipla quando uma subclasse possui mais de uma super-classe.

- **Associação:** É o mecanismo pelo qual um objecto utiliza os recursos de outro. Pode tratar-se de uma associação simples ou composição

- **Encapsulamento:** Consiste na separação de aspectos internos e externos de um objecto. Este mecanismo é utilizado amplamente para impedir o acesso directo ao estado de um objecto (seus atributos), disponibilizando externamente apenas os métodos que alteram estes estados.

- **Abstracção:** É a capacidade de se focar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelação orientada a objectos, uma classe é uma abstracção de entidades existentes no domínio do sistema de software.

- **Polimorfismo:** É o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura (lista de parâmetros e retorno) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objecto do tipo da superclasse.

3.6 PROTÓTIPOS ABSTRACTOS CANÓNICOS (PAC)

A ideia de **Protótipos Canónicos Abstractos**, foi uma abordagem que foi desenvolvida em workshop de profissionais [PAC2003], basicamente é um modelo desenvolvido para testar ideias de design, com isto pretende-se examinar conteúdo, estética e técnicas de interacção nas perspectivas dos designers, clientes e utilizadores, para que se consiga cortar na complexidade da implementação, eliminando partes do sistema total.

Existem três tipos de componentes genéricos, abstractos e extensíveis

- Materiais: contentores, conteúdo, informação □
    - e.g.: resultados de procura, notificações ao utilizador
- Ferramentas: acções, mecanismos que operam sobre os materiais ↗
    - e.g.: selector de cor, comando impressão, botão de 'submit'

- Híbridos (ou materiais Activos) 
    - e.g.: caixa de introdução de texto (mostra informação e manipula-a!)

3.6.1 TÉCNICAS DE PROTOTIPAGEM

Existem várias técnicas de prototipagem, vamos enumerar algumas, explicando o propósito de cada uma delas.

3.6.1.1 MOCK-UPS EM PAPEL

Segundo Carolyn Snyders [CS2003], No início do processo, o designer esboça e cria protótipos de papel, usualmente compostos por desenhos a lápis, screenshots impressos, ou uma combinação de ambos, usando materiais low-tech, a equipa pode rapidamente construir um protótipo de teste

3.6.1.2 CENÁRIOS

Levam a prototipagem ao extremo, reduzindo tanto o nível de funcionalidade como o número de funcionalidades.

Podem ser muito baratos para desenhar e implementar, mas só podem simular a interface de utilização desde que o utilizador siga um caminho previamente definido.

Podem ser implementados como mock-ups em papel ou em simples ambientes de prototipagem rápida, que podem ser mais fáceis de aprender do que ambientes de programação avançados

3.6.1.3 MAPAS DE NAVEGAÇÃO

Diagrama que especifica como os diferentes espaços de interacção estão interligados e como o utilizador pode fluir através da interface de utilização no decurso das tarefas.

- Normalmente evoluem ao longo do ciclo de vida do projecto
- São muito usados para definir a estrutura de informação em sites Web (área de Information Design)

3.6.1.4 GUIÕES

Um guião é uma representação de uma sequência de interacção particular, reflectem detalhe limitado sobre o conteúdo de cada ecrã

Apenas os elos de navegação necessários ao desempenho de uma tarefa são representados

3.6.1.5 ESQUEMAS (SCHEMATICS)

Reapresentações do conteúdo que deveria aparecer num determinado ecrã, não têm imagens, apesar de poderem indicar com uma etiqueta textual onde as imagens apareceriam, ou onde deveriam ser colocadas, não devemos utilizar cor, mas podem fazer uso da cor ou de escalas de cinzentos para dar significado acerca de elementos na interface de utilização

3.7 RESUMO

Neste capitulo fez se um revisão bibliográfica do conjunto de modelos metodologias conceitos e definições. Abordou-se e a Engenharia de Software, introduziu-se o processo de Software, bem com enumerando as sua fases e explicou-se o que consiste cada uma delas. Introduziu-se os Modelos do Processo de Software, explicando os mais importantes, introduziu-se o conceito de Protótipos Canónicos Abstractos e finalmente apresentou-se o paradigma orientado a objectos onde mostramos os seus conceitos e vantagens da sua utilização.

4. IMPLEMENTAÇÃO DO SISTEMA

"I think it's fair to say that personal computers have become the most empowering tool we've ever created. They're tools of communication, they're tools of creativity, and they can be shaped by their user."

Bill Gates

Tópicos

4.1 INTRODUÇÃO

Neste capítulo mostra-se técnicas, métodos para o desenvolvimento da aplicação, para tal optou-se pelo desenvolvimento incremental, utilizando para o efeito

o conceito de programação orientada por objectos (POO), e finalmente usamos o modelo UML (Unified Modeling Language), para fazer toda a parte de modelação do sistema.

4.2 ESPECIFICAÇÃO DO SISTEMA

Uma das etapas mais importantes no processo de desenvolvimento de software e o processo de levantamento de requisitos, esta fase é de extrema importante para saber para que propósitos será desenvolvido o software e para quem será desenvolvido. Existem várias definições para classificar os requisitos, por exemplo requisitos funcionais e requisitos não funcionais.

Um **requisito** [ASCV05] é uma especificação de uma determinada acção, condição ou necessidade que o sistema deverá satisfazer.

Um **requisito funcional** [ASCV05] descreve uma determinada acção (ou função) que o sistema deverá providenciar. Por outro lado, um requisito não funcional descreve um aspecto (não funcional) que o sistema deverá satisfazer.

Um **requisito não funcional** [ASCV05], também designado por "propriedades emergentes", referem-se a aspectos transversais, normalmente aplicáveis a todo o sistema, tais como: desempenho, robustez, fiabilidade, distribuição, segurança, integração, com a internet, facilidade de manutenção, usabilidade ou suporte a standards.

4.2.1 REQUISITOS FUNCIONAIS

Para este projecto identificamos os seguintes requisitos funcionais:

1. O sistema deverá ser capaz de gerir os utilizadores do sistema, ou seja deve ser capaz de **criar**, **editar** e **apagar** utilizadores, bem como **atribuir/retirar permissões** para certas operações no sistema.

2. O sistema deverá permitir a **criação de obras**, bem como poder introduzir alguns detalhes da obra, como morada, localização geográfica, data de inicio e data de fim.

3. O sistema deverá permitir o **manuseamento** dos registadores que são atribuídos a cada uma das obras, bem com poder mostrar no sistema a disposição dos mesmos de acordo com a sua real distribuição na obra.

4. O sistema deverá permitir a **geração** de relatórios automáticos, com os valores lidos pelos registadores bem como outro tipo de informação.

5. O sistema deverá permitir que sejam **introduzidas fotos** relativas a cada uma das obras, criando uma galeria de fotos associada a cada uma das obras.

6. O sistema deverá permitir uma **análise** de estatísticas através de gráficos 2D, que mostrem quer os últimos valores lidos nos sensores bem com médias.

4.2.2 REQUISITOS NÃO FUNCIONAIS

Neste sistema foram identificados vários requisitos não funcionais que se dividem em três classes, requisitos de processo, de produtos e externos

4.2.2.1 REQUISITOS DE PROCESSO

Os requisitos não funcionais de processo identificados neste sistema foram os seguintes:

1. O Sistema teria de ser implementado num prazo determinado pelo utilizador final

2. O sistema teria de ser implementado de maneira que não dependessem de terceiros, ou seja não estivessem dependentes de outros softwares, e caso estivessem fossem de produtos gratuitos.

4.2.2.2 REQUISITOS DE PRODUTOS

Os requisitos não funcionais de produto identificados neste sistema foram os seguintes:

1. **Usabilidade**: o sistema deve ser fácil de utilizar, quer para um utilizador experiente quer para um utilizador comum.

2. **Fiabilidade**: o sistema deve ser fiável, para garantir a veracidade dos dados obtidos.

3. **Eficiência/Eficácia**: o sistema deve ser eficiente e eficaz na realização das suas operações.

4. **Desempenho**: o sistema deve ter um desempenho alto, mesmo quando este está a processar um nível médio/alto de operações

5. **Capacidade**: o sistema deve ter uma grande capacidade de armazenamento de informação.

6. **Escalonável**: o sistema deve estar preparado para ser expandido, quer em termos de capacidade armazenamento, quer de outro tipo

7. **Segurança**: o sistema deve ser seguro e garantir que os utilizadores só vejam aquilo que tem permissões para ver.

4.2.2.3 REQUISITOS EXTERNOS

Os requisitos não funcionais externos identificados neste sistema foram os seguintes:

1. **Interoperabilidade**: o sistema deve estar preparado para interoperar com outros sistemas.

2. **Restrições Legais**: o sistema deve garantir a segurança e confidencialidade dos dados do sistema.

3. **Económicas**: o sistema deve permitir que sejam minimizados custos de execução das obras, a quando da sua introdução

4.3 VISÃO GERAL

De modo a poder dar uma visão geral, escolheu-se a modelação UML, neste caso o diagrama de classes de alto nível.

A figura 11 podemos ter uma visão geral do sistema **tacRemote** e como as diferentes classes se relacionam. Nesta figura apenas serão mostradas as classes principais bem como o seu relacionamento.

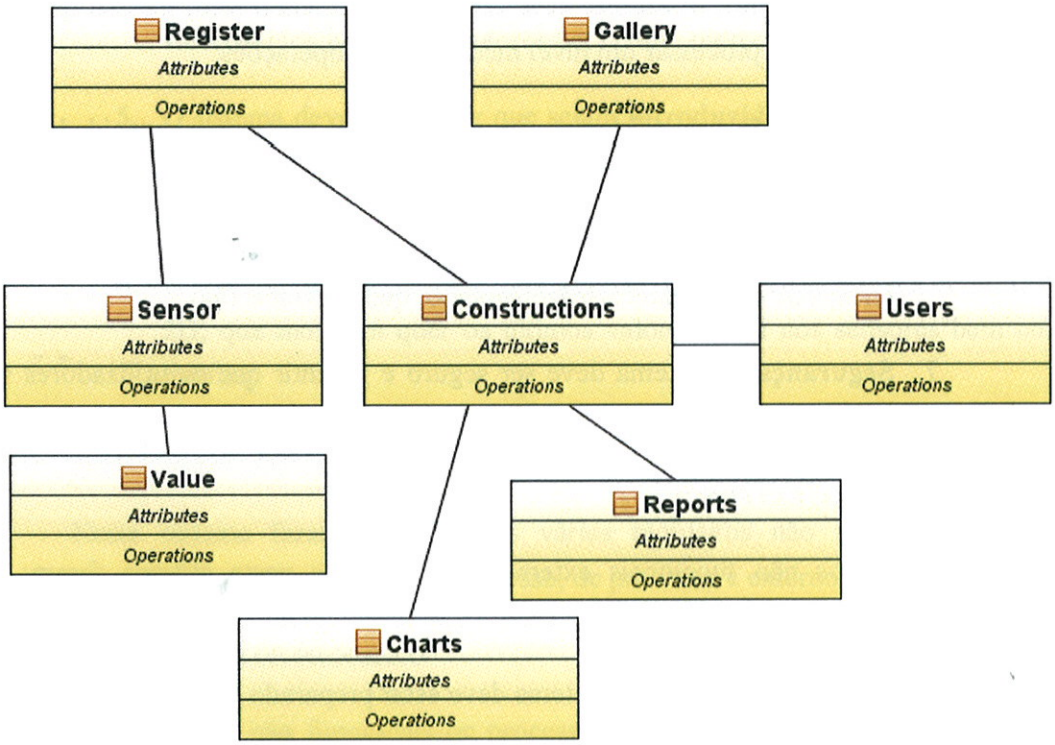


FIGURA 11 – DIAGRAMA DE CLASSES DE ALTO NÍVEL DO SISTEMA TACREMOTE

4.3.1 A CLASSE UTILIZADORES (USERS)

A classe **Utilizadores** é responsável por toda a gestão de utilizadores, isto implica que esta classe é responsável por criar utilizadores, apagar utilizadores e editar utilizadores.

4.3.2 A CLASSE OBRAS (CONSTRUCTIONS)

A classe **Obras** é responsável pela gestão das obras, isto implica que esta classe é responsável por criar obras, apagar obras e editar utilizadores. Esta classe é também responsável por atribuir permissões ao diferentes utilizadores para poderem ou não criar, ver, editar e apagar obras.

4.3.3 A CLASSE REGISTADOR (REGISTER)

A classe **Registador** é responsável por adicionar registadores as obras, editar esses registadores e remover esses registadores.

4.3.4 A CLASSE SENSOR

A classe **Sensor** é responsável por atribuir sensores a cada um dos registadores. Esta classe é também responsável por remover e editar sensores atribuidos.

4.3.5 A CLASSE VALOR (VALUE)

A classe **Valor** é responsável por ler os valores de cada um dos sensores atribuídos aos respectivos registadores.

4.3.6 A CLASSE RELATÓRIOS (REPORTS)

A classe **Relatórios** é responsável por toda a gestão dos relatórios, ou seja todos os relatórios que são gerados, são geridos por esta classe, que também tem funções básicas, como apagar, e ver os relatórios gerados.

4.3.7 A CLASSE GALERIA (GALLERY)

A classe **Galeria** é responsável pele gestão da galeria, ou seja quando é criado uma obra, automaticamente é criada uma galeria de imagens, onde podemos inserir, remover e editar imagens. Estas funções são disponibilizadas pela classe **Galeria**.

4.3.8 A CLASSE GRÁFICOS (CHARTS)

A classe **Gráficos** é responsável pela gestão dos gráficos, esta classe permite a visualização de gráficos, estes que são gerados automaticamente com base nos valões obtidos dos sensores. Esta classe também permite a visualização de gráficos de valores médios e resistências médias.

4.4 VISÃO FUNCIONAL

Depois de ver o diagrama de classes anterior, podemos ver agora de uma maneira mais profunda as funcionalidades propostas para o sistema, através da identificação dos seus actores e descrição dos principais casos de utilização. Neste sistema identificamos 3 tipos de actores (**Cliente**, **Operador** e **Administrador**). Estes utilizadores podem realizar os seguintes casos de utilização como podemos ver na figura 12. De notar que o utilizador **Administrador** estende o utilizador **Operador** que por sua vez estende o utilizador **Cliente**, ou seja todas as funcionalidades do utilizador **Cliente** também são do utilizador **Operador** e consequentemente as funcionalidades do utilizador **Operador** são do utilizador **Administrador**.

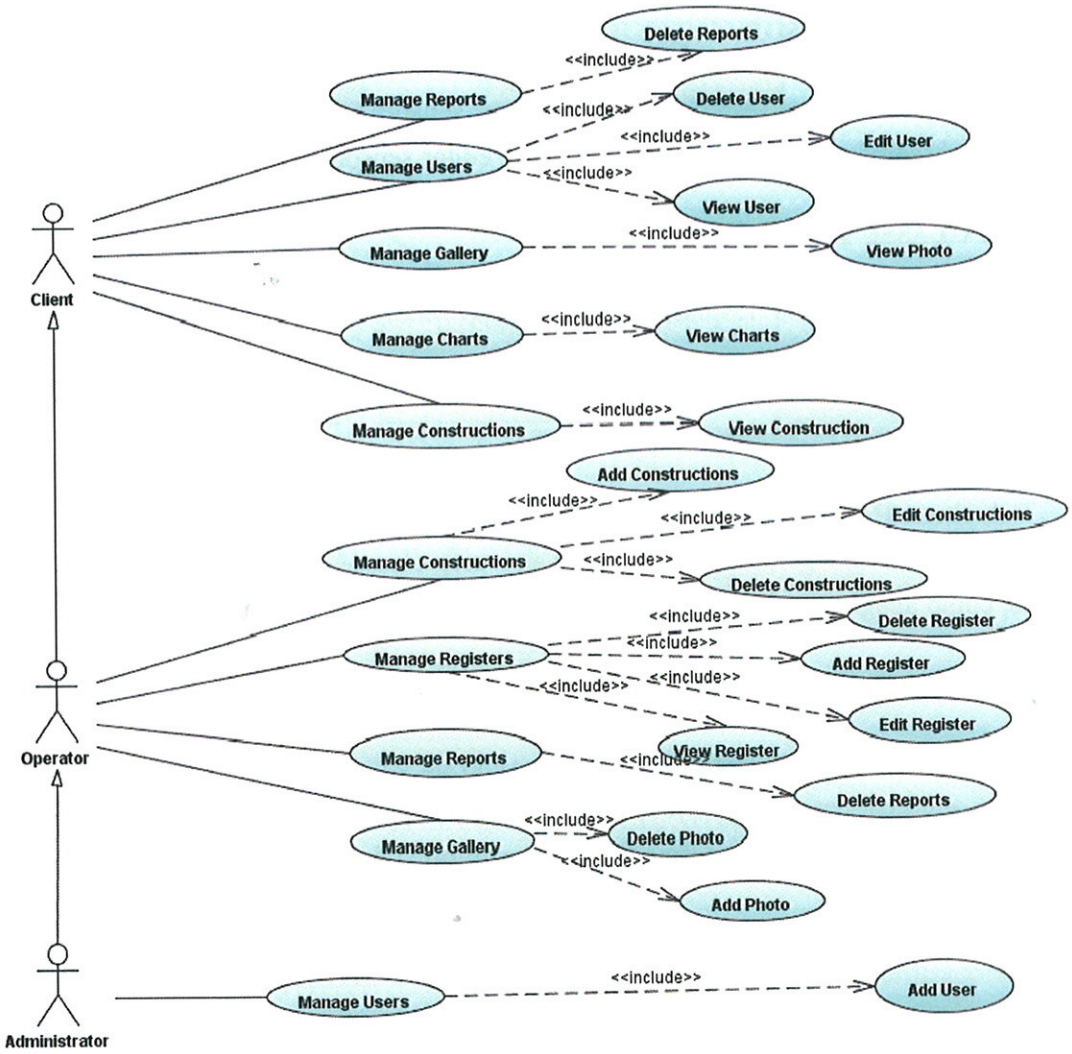


FIGURA 12 - DIAGRAMA DE CASOS DE UTILIZAÇÃO

O actor **Cliente** pode:

- **Fazer download de relatórios**: Este utilizador após escolher o relatório pode fazer o download do relatório para o seu computador.
- **Apagar Utilizador:** Este utilizador pode apagar apenas o seu perfil
- **Editar Utilizador:** Este utilizador pode editar apenas o seu perfil
- **Ver Utilizador:** Este utilizador apenas pode ver o seu perfil
- **Ver Fotos:** Este utilizador apenas pode ver as fotos da galeria que um utilizador do tipo Administrador tenha dado permissões.

- **Ver Gráficos:** Este utilizador pode ver os gráficos gerados relativos a uma determinada obra que um utilizador do tipo Administrador tenha dado permissões.

- **Ver Obras:** Este utilizador pode ver as obras que um utilizador do tipo Administrador ou Operador tenha dado permissões

O utilizador Operador pode fazer todas as operações do utilizador Cliente e ainda as seguintes:

- **Adicionar Obra:** Este utilizador pode criar obras e atribuir permissões aos utilizadores do tipo cliente para vê-la.

- **Editar Obra:** Este utilizador pode editar as obras que tenham sido criadas por si.

- **Apagar Obra:** Este utilizador pode apagar as obras que tenham sido criadas por si.

- **Adicionar Registador:** Este utilizador pode adicionar registadores as obras criadas por si.

- **Apagar Registador:** Este utilizador pode apagar registadores das obras criadas por si.

- **Editar Registador:** Este utilizador pode editar registadores das obras criadas por si.

- **Apagar Relatório:** Este utilizador pode apagar relatórios das obras criadas por si.

- **Adicionar Fotos:** Este utilizador pode adicionar fotos à galeria das obras criadas por si.

- **Apagar Fotos:** Este utilizador pode apagar fotos da galeria das obras criadas por si.

O utilizador Administrador pode fazer todas as operações dos utilizadores Operador e Cliente e ainda as seguintes:

- **Adicionar utilizadores:** Este utilizador pode adicionar utilizadores ao sistema

4.5 RESUMO

Neste capítulo mostramos técnicas, métodos para o desenvolvimento da aplicação, para tal optamos pelo desenvolvimento incremental, utilizando para o efeito o conceito de programação orientada por objectos (POO), e finalmente usamos o modelo UML (Unified Modeling Language), para fazer toda a parte de modelação do sistema.

5. O SISTEMA TACREMOTE

"I'm surprised at the extent of the bigotry. But it really plays out when companies or schools take a side and prohibit the other platform at all. We Mac users should be good even when the other side is bad. We should do what we can to accept the other platforms."

Steve Wozniak

Tópicos

5. O SISTEMA TACREMOTE

"I'm surprised at the extent of the bigotry. But it really plays out when companies or schools take a side and prohibit the other platform at all. We Mac users should be good even when the other side is bad. We should do what we can to accept the other platforms."

Steve Wozniak

Tópicos

5.1 INTRODUÇÃO

Neste capítulo apresenta-se a interface do sistema **tacRemote**, mostrando as funcionalidades mais importantes do sistema e a sua relação com o utilizador. Faz-se também três pequenas demonstrações do sistema

5.2 PROTÓTIPOS ABSTRACTOS CANÓNICOS DO SISTEMA

Como vimos anteriormente (Secção 3.6), os **PACs** são uma maneira de podermos desenhar a aplicação sem que para isso entremos em grandes detalhes, podemos ver nas Tabelas 2,3 e 4 os tipos de materiais que podemos utilizar para fazer a prototipagem do sistema. No ponto seguinte (5.3), vamos mostrar a passagem dos protótipos abstractos canónicos para as interfaces em concreto.

SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
	action/operation*	Print symbol table, Color selected shape
	start/go/to	Begin consistency check, Confirm purchase
	stop/end/complete	Finish inspection session, Interrupt test
	select	Group member picker, Object selector
	create	New customer, Blank slide
	delete, erase	Break connection line, Clear form
	modify	Change shipping address, Edit client details
	move	Put into address list, Move up/down
	duplicate	Copy address, Duplicate slide
	perform (& return)	Object formatting, Set print layout
	toggle	Bold on/off, Encrypted mode
	view	Show file details, Switch to summary

TABELA 2 – FERRAMENTAS ABSTRACTAS [CAP2003]

SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
	container*	Configuration holder, Employee history
	element	Customer ID, Product thumbnail image
	collection	Personal addresses, Electrical Components
!	notification	Email delivery failure, Controller status

TABELA 3 – MATERIAIS ABSTRACTAS [CAP2003]

SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
	active material*	Expandable thumbnail, Resizable chart
	input/accepter	Accept search terms, User name entry
	editable element	Patient name, Next appointment date
	editable collection	Patient details, Text object properties
	selectable collection	Performance choices, Font selection
	selectable action set	Go to page, Zoom scale selection
	selectable view set	Choose patient document, Set display mode

TABELA 4 – HÍBRIDOS ABSTRACTAS OU MATERIAIS ACTIVOS [CAP2003]

5.3 INTERFACES DO SISTEMA

O sistema tacRemote, a partir do momento em que fazemos o login como podemos ver na Figura 22, é composto por uma interface geral que possui **seis** separadores que agregam cada um dos sub-módulos do sistema.

5.3.1 AUTENTICAÇÃO NO SISTEMA

Para fazer a autenticação no sistema basta introduzir um username e uma password válida, como podemos ver na figura 14. A figura 13 mostra nos o PAC que deu origem à figura 14.

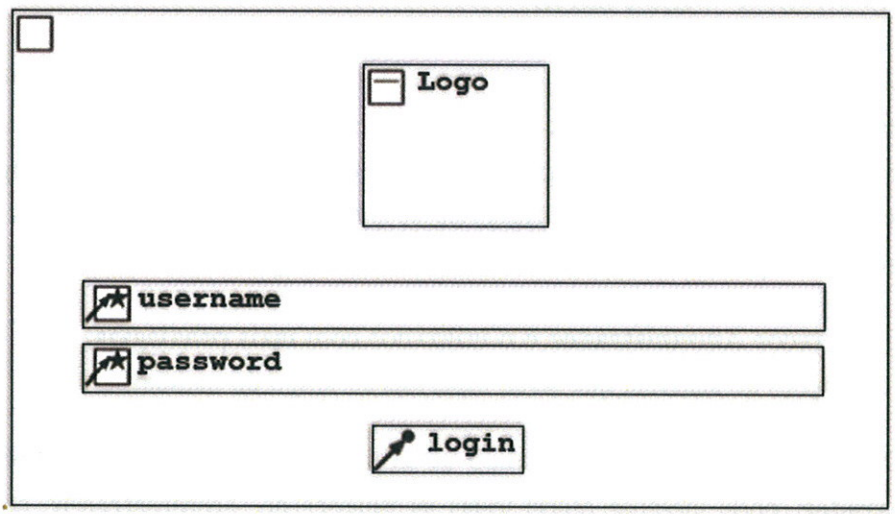


FIGURA 13 - AUTENTICAÇÃO NO SISTEMA (PAC)

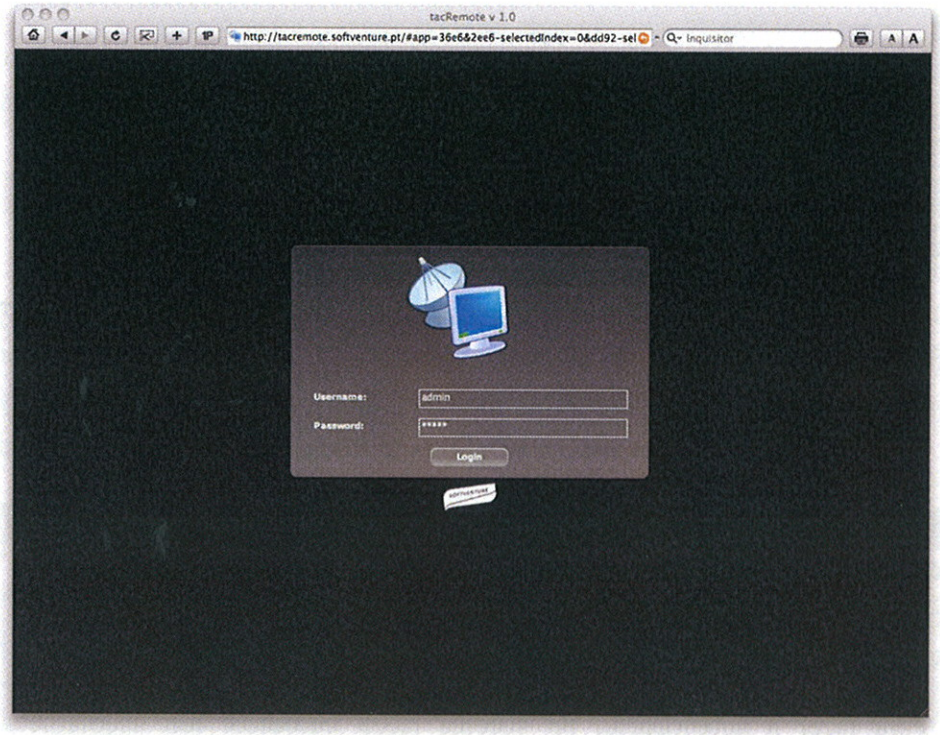


FIGURA 14 – AUTENTICAÇÃO NO SISTEMA

5.3.2 SEPARADOR "MEU PERFIL"

O separador **"Meu perfil"** é o separador onde podemos efectuar todas as operações sobre os utilizadores do sistema, basicamente permite quatro grandes operações (ver, adicionar, alterar e apagar utilizadores), como podemos ver na figura 16, que podem ser efectuadas pelos utilizadores, se tiverem permissões para o fazer. Este sistema admite 3 tipos de utilizadores, **Administrador**, **Operador** e **Cliente**.

O utilizador **Administrador** é o utilizador com mais privilégios, pode efectuar todas as operações do sistema, que serão descritas ao longo do manual.

O utilizador **Operador**, é um tipo de utilizador com menos permissões que o **Administrador**, pode efectuar quase todas as funções do **Administrador**, excepto a gestão de Utilizadores.

Finalmente o utilizador Cliente, possui apenas permissões de leitura, ou seja não tem permissões para alterar ou apagar alguma coisa, excepção feita aos detalhes do seu perfil. A figura 15 mostra-nos o PAC que deu origem a figura 16.

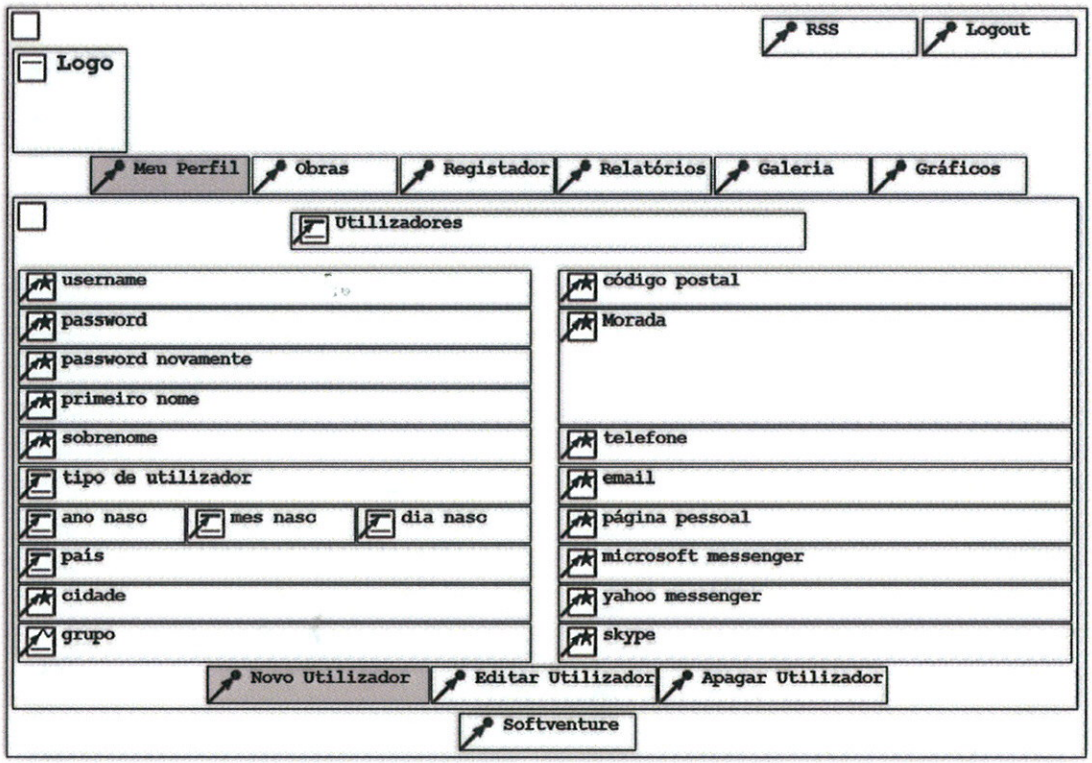


FIGURA 15 - O SEPARADOR "MEU PERFIL" (PAC)

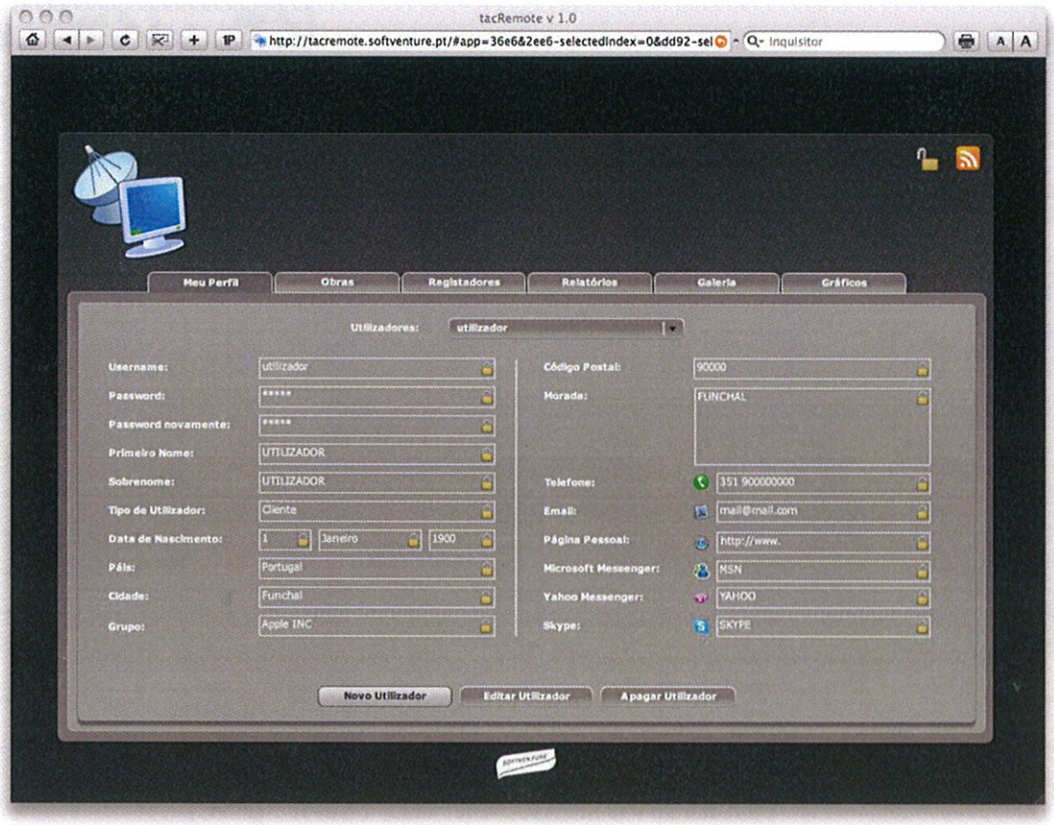


FIGURA 16 – O SEPARADOR "MEU PERFIL"

5.3.3 SEPARADOR "OBRAS"

O separador "Obras" é o separador onde podemos efectuar todas as operações sobre as obras do sistema, basicamente permite quatro grandes operações (adicionar, apagar, editar e ver obras) como podemos ver na figura 18, que podem ser efectuadas pelos utilizadores, se tiverem permissões para o fazer. A figura 17 mostra-nos o PAC que deu origem a figura 18.

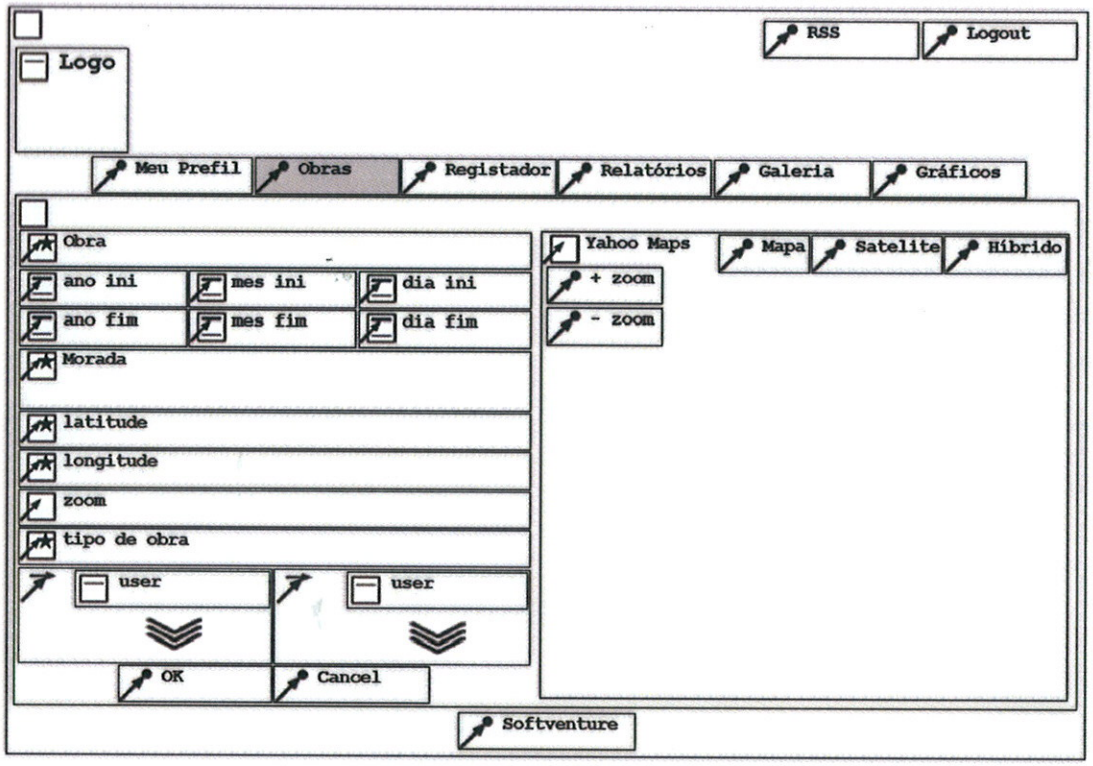


FIGURA 17 - O SEPARADOR "OBRAS" (PAC)

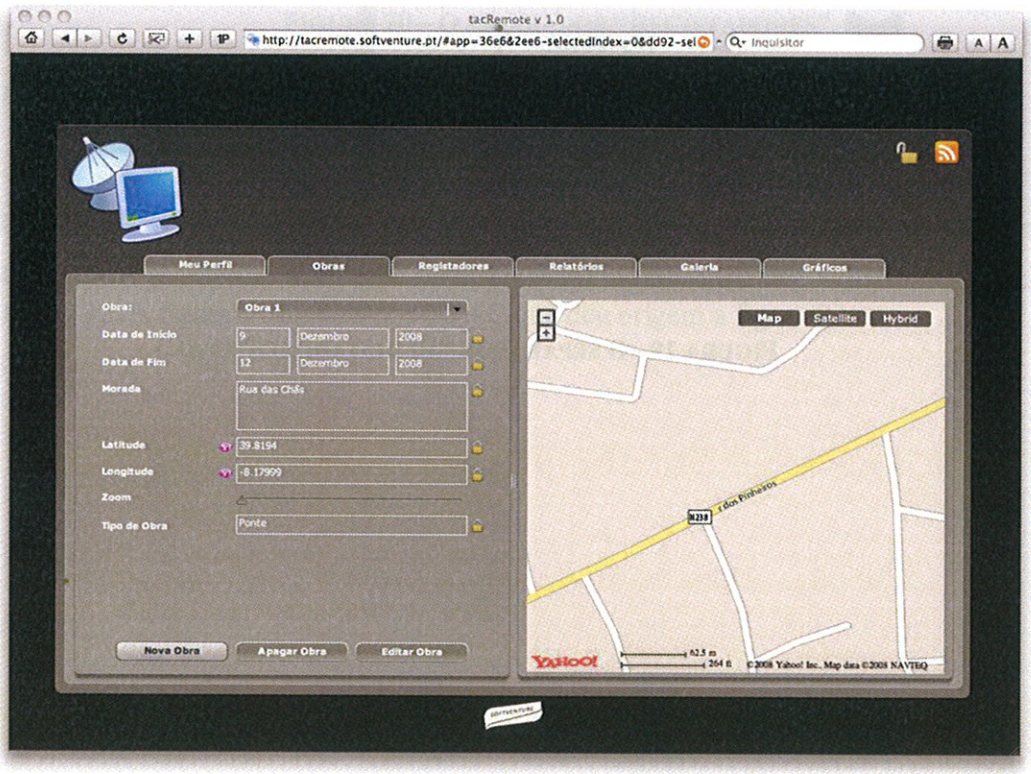


FIGURA 18 – O SEPARADOR "OBRAS"

5.3.4 SEPARADOR "REGISTADORES"

O separador **"Registadores"** é o separador onde podemos efectuar todas as operações sobre os registadores e sensores, basicamente permite 3 grandes operações (adicionar, remover e apagar), como podemos ver na figura 20, que podem ser efectuadas pelos utilizadores, se tiverem permissões para o fazer. A figura 19 mostra-nos o PAC que deu origem a figura 20.

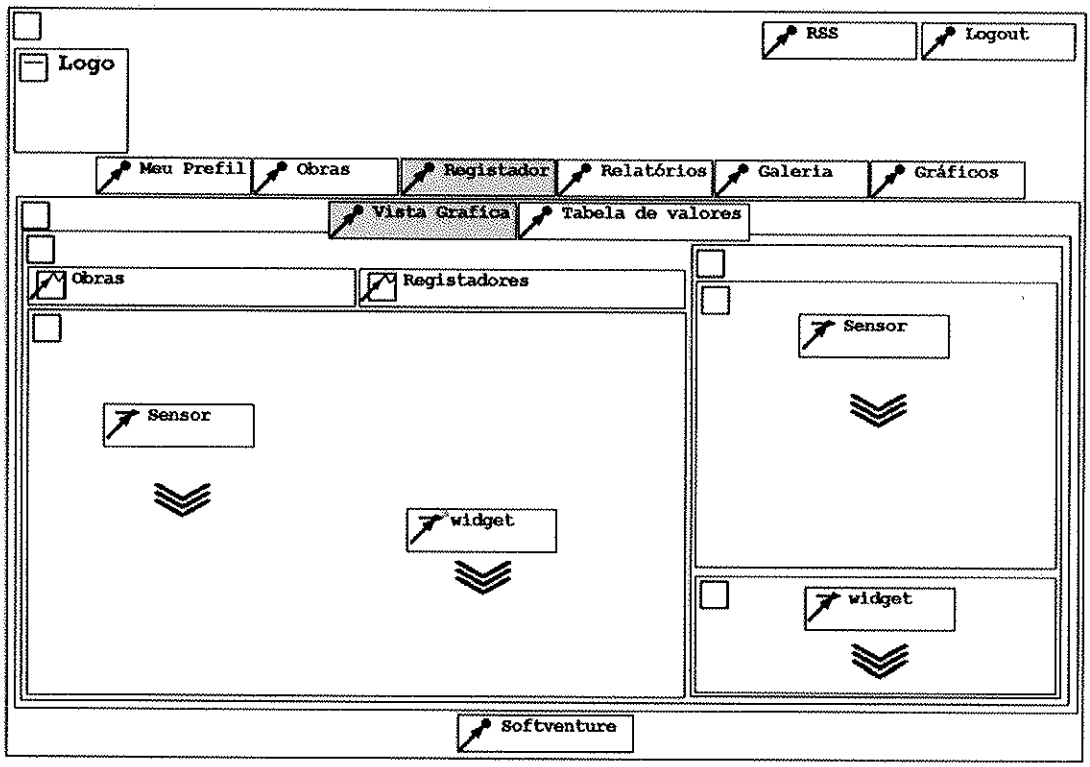


FIGURA 19 - O SEPARADOR "REGISTADORES" (PAC)

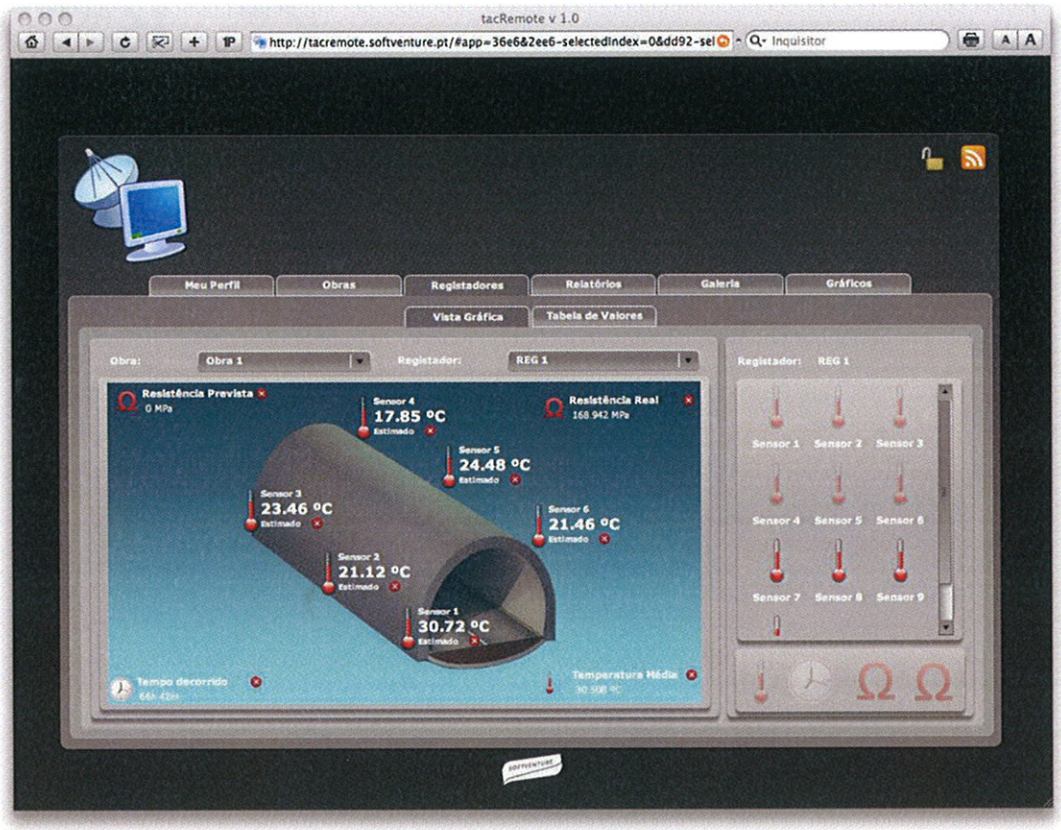


FIGURA 20 – O SEPARADOR "REGISTADORES"

5.3.5 SEPARADOR "RELATÓRIOS"

O separador **"Relatórios"** é o separador onde podemos efectuar todas as operações sobre os relatórios. Os relatórios do tacRemote são gerados duma forma automática. Este separador permite fazer a gestão desses relatórios como podemos ver na figura 22. A figura 21 mostra-nos o PAC que deu origem a figura 22.

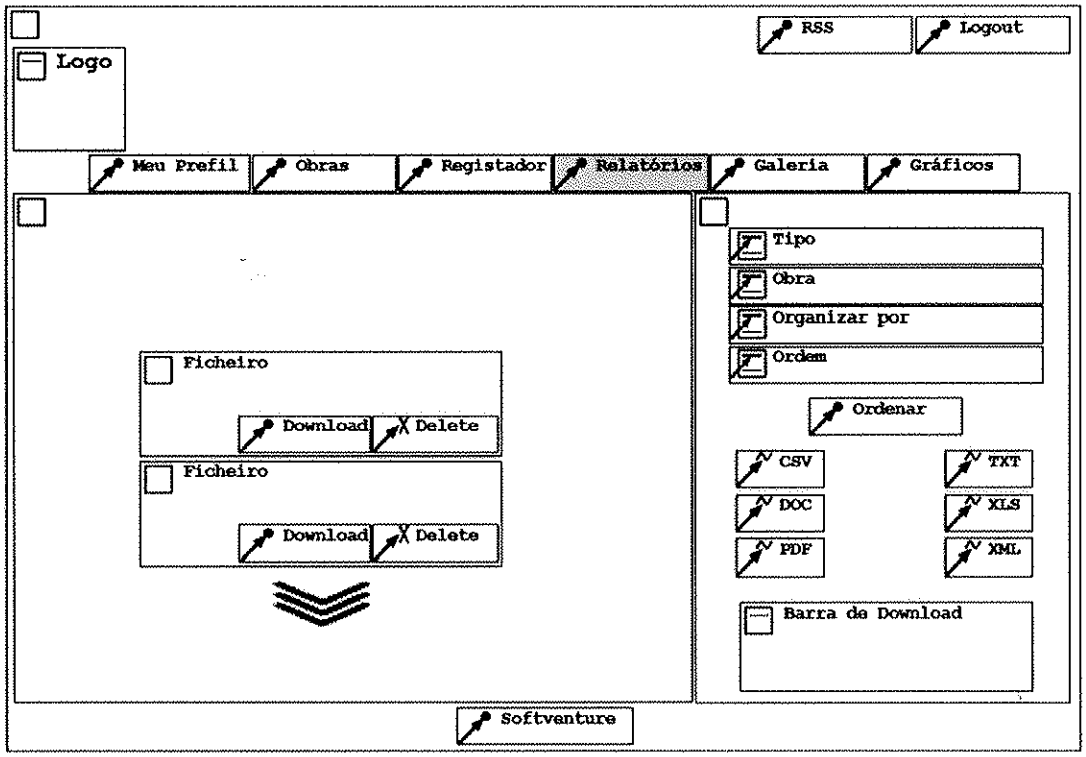


FIGURA 21 - O SEPARADOR "RELATÓRIOS" (PAC)

FIGURA 22 – O SEPARADOR "RELATÓRIOS"

5.3.6 SEPARADOR "GALERIA"

O separador **"Galeria"** é o separador onde podemos efectuar todas as operações sobre a galeria. O sistema tacRemote permite aos utilizadores introduzirem imagens na galeria relativas a cada obra, ou seja sempre que seja criada uma obra, e automaticamente é criada uma galeria para cada obra. O processo de gestão da galeria está explicado na figura 24. A figura 23 mostra-nos o PAC que deu origem a figura 24.

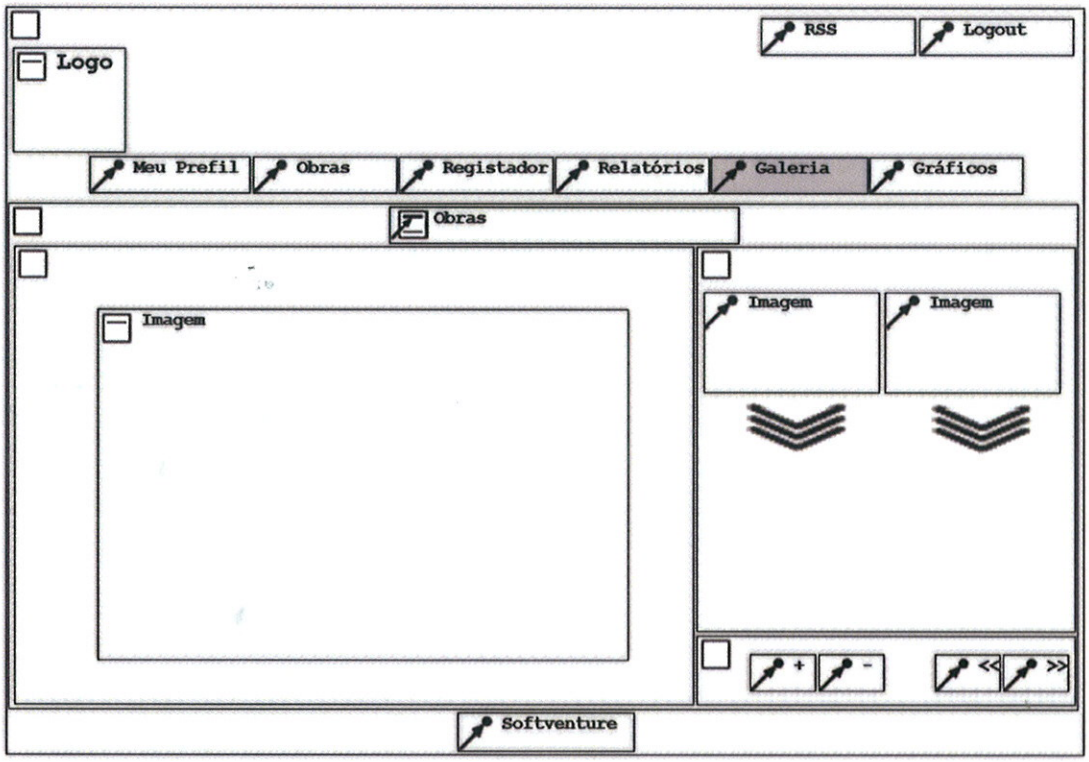


FIGURA 23 - O MENU GALERIA (PAC)

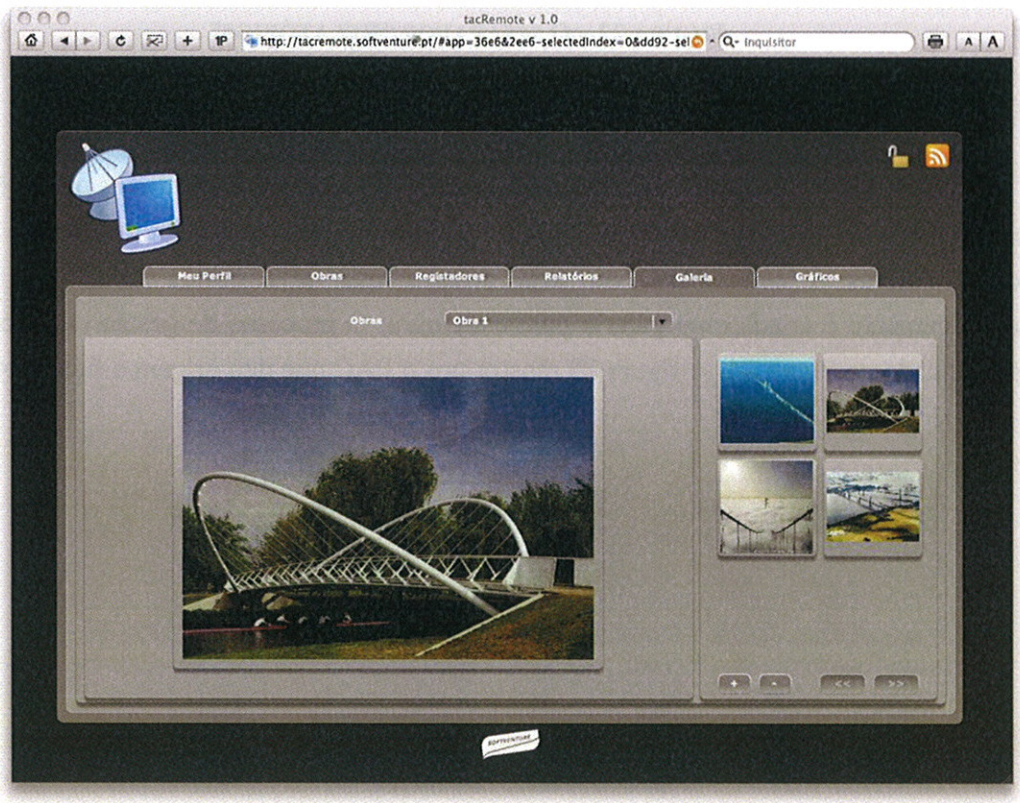


FIGURA 24 – O MENU GALERIA

5.3.7 SEPARADOR "GRÁFICOS"

O separador **"Gráficos"** é o separador onde podemos efectuar todas as operações sobre a galeria. O sistema tacRemote permite a análise estatística dos dados através de vários gráficos disponibilizados, com vários tipos de valores, ou seja permite analisar o cálculo das resistências de cada sensor, através de um gráfico de linhas, podemos ver todos os sensores ou escolher quantos queremos ver seleccionando a checkbox relativa a cada sensor, bem como analisar os valores lidos por cada sensor, através de um gráfico de linhas, podemos ver todos os sensores ou escolher quantos queremos ver seleccionando a checkbox relativa a cada sensor, como podemos ver na figura 26. A figura 25 mostra-nos o PAC que deu origem a figura 26.

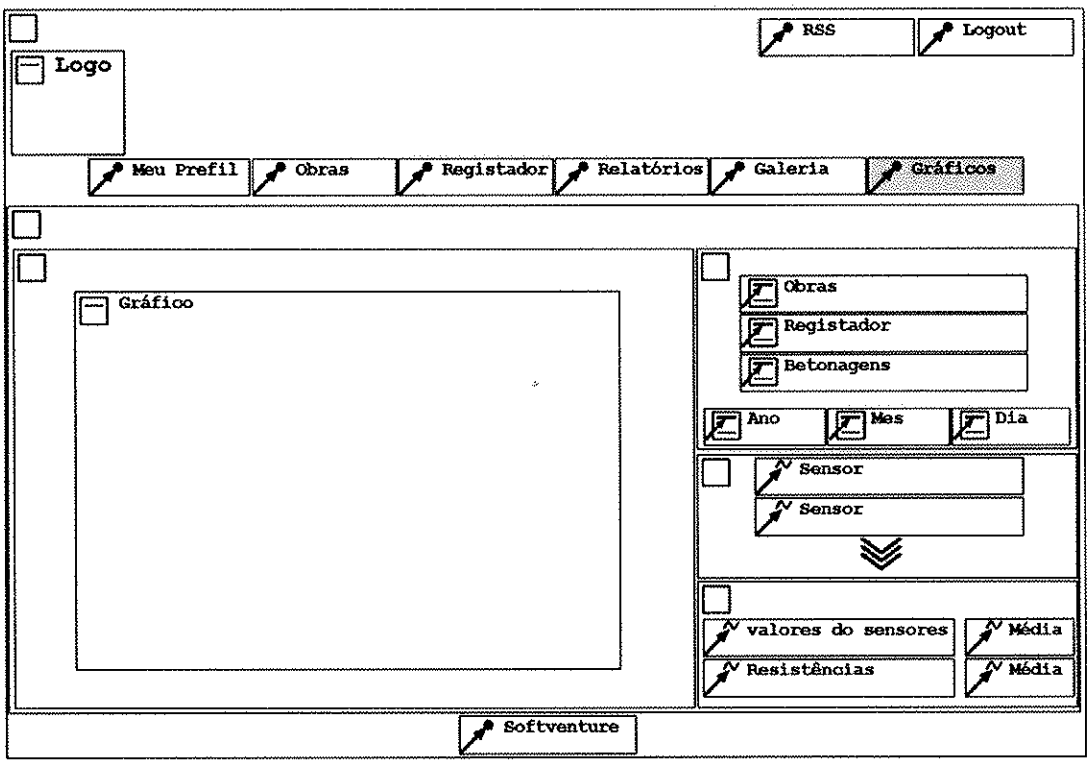


FIGURA 25 - O SEPARADOR "GRÁFICOS" (PAC)

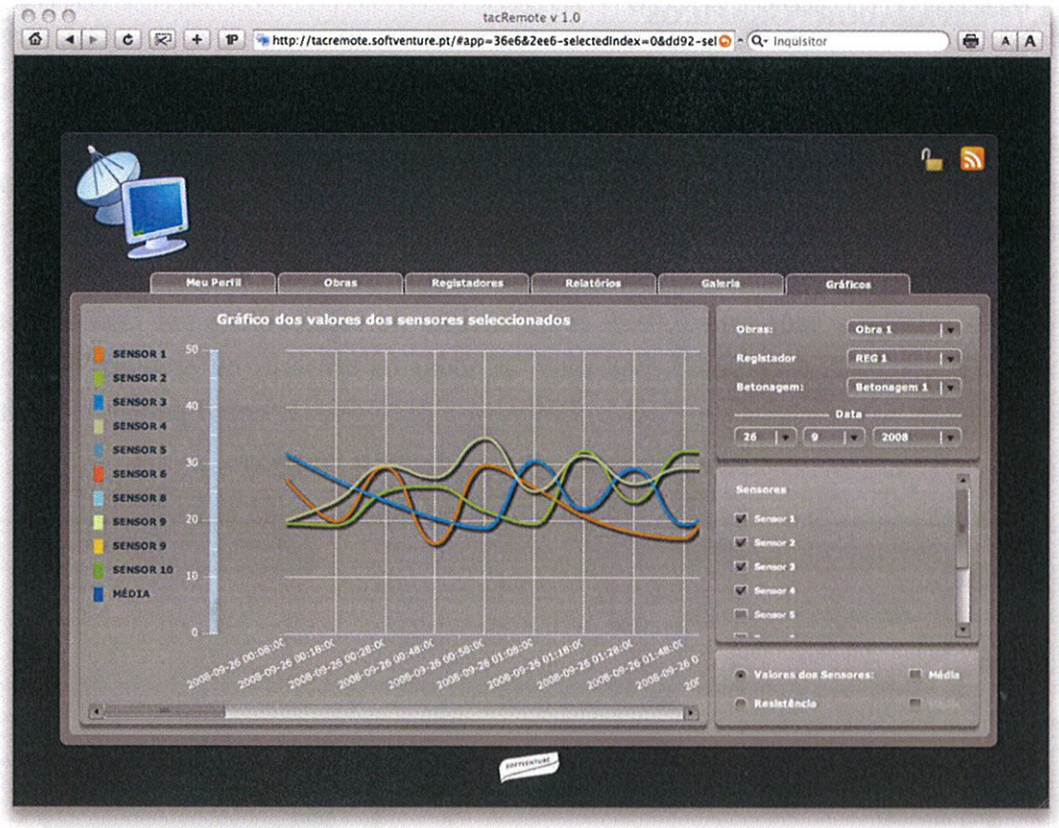


FIGURA 26 – O SEPARADOR "GRÁFICOS"

5.4 DEMONSTRAÇÃO

O sistema **tacRemote** permite uma variedade de funcionalidades, nos próximos pontos vamos explicar 3 cenários de possíveis casos de utilização que são a introdução de um novo utilizador no sistema, a criação de uma nova obra e visualização dos resultados através de gráficos

5.4.1 INTRODUÇÃO DE UTILIZADORES

O sistema permite adicionar utilizadores, embora esta operação seja a que necessita de um nível de privilégios maior, ou seja apenas o utilizador do tipo Administrador pode criar utilizadores. Após escolher o separador "Meu Perfil" e escolher a opção Novo Utilizador será direccionado para a interface de criação de um novo utilizador como podemos ver na figura 27.

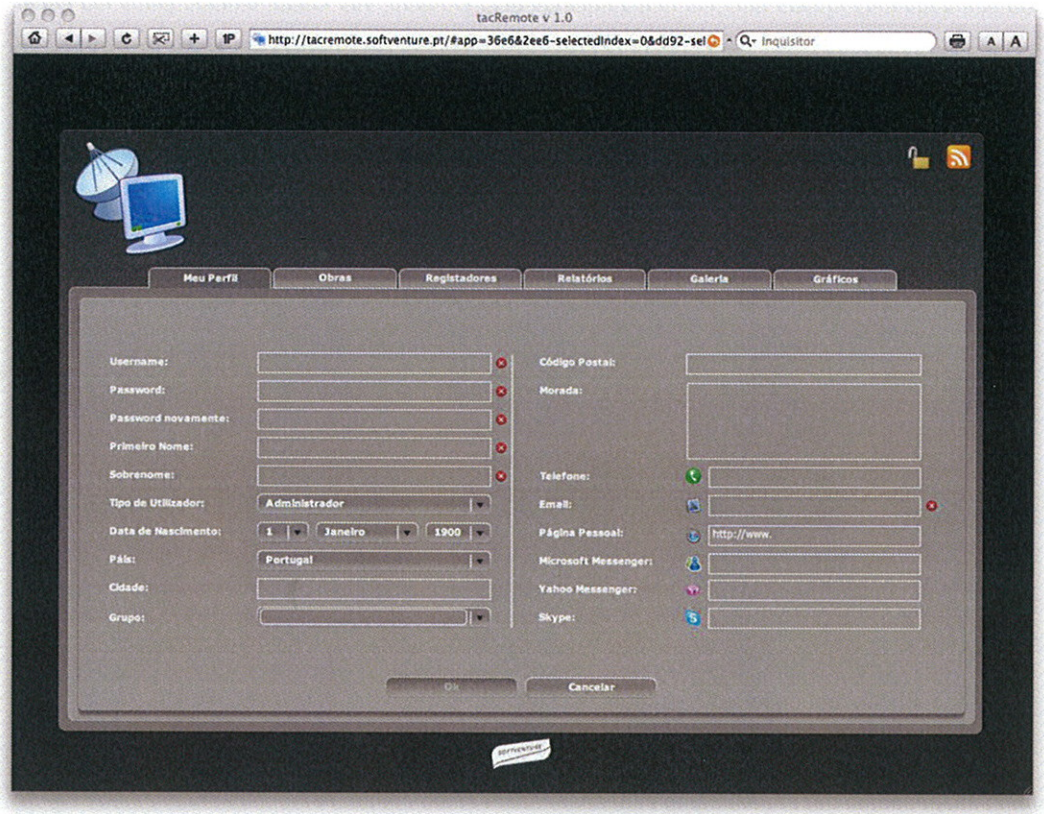


FIGURA 27 – CRIAR UM NOVO UTILIZADOR

A partir desta fase e necessário preencher os dados do utilizador, note-se que os campos com o símbolo (⊗) são obrigatórios e tem de ser preenchidos correctamente, ou seja o nome tem de ser válido ou e-mail também, e este campos só estão válidos se o símbolo (⊗) passar a (✓).

Após a validação dos campos o sistema coloca o botão **OK** disponível para enviar esses dados. Ao pressionar o botão é mostrado uma janela de confirmação para criar ou não o utilizador, se confirmar que sim o utilizador é introduzido no sistema.

5.4.2 CRIAÇÃO DE OBRA

O sistema tacRemote permite adicionar obras, mais uma vez por questões de segurança, apenas os utilizadores do tipo Administrador e Operador podem adicionar obras, para tal basta escolher o separador obras e pressionar o botão NOVA OBRA, como podemos ver na figura 28.

Para criar uma obra existem campos que são obrigatórios, o nome da obra, morada, data de inicio e de fim, os restantes são opcionais embora sejam importantes para caracterizar a obra. Após estes campos estarem todos validos (com o símbolo ✓),

podemos escolher quais são os utilizadores que podem ver estas obras, para tal basta a arrastar o utilizador da lista mais a direita para a da esquerda para remover esse utilizador o processo é inverso.

FIGURA 28 – CRIAR UMA NOVA OBRA

Após preenchermos os dados podemos escolher a localização geográfica da obra para tal basta arrastar-mos o mapa.

Depois de introduzirmos todos os dados da obra, pressionamos o botão **OK**, e é mostrado uma janela de confirmação para criar ou não a obra, se confirmar que sim a obra é introduzido no sistema.

5.4.3 VISUALIZAÇÃO DE RESULTADOS

O sistema tacRemote permite visualização de gráficos e a análise estatística dos dados através de vários gráficos disponibilizados, com vários tipos de valores, como podemos ver na figura 29.

FIGURA 29 – VISUALIZAÇÃO DE RESULTADOS

Podemos escolher qual o tipo de valores que queremos ver (Valores dos sensores ou Resistências) basta seleccionar o radiobutton respectivo, ao fazermos isto, o gráfico vai actualiza-se com os respectivos valores. Também podemos escolher quantos sensores queremos ver de cada vez, para tal basta seleccionar os sensores.

Finalmente podemos também filtrar os valores, escolhendo as várias opções no canto superior esquerdo.

5.5 RESUMO

Neste capítulo apresentamos a interface do sistema **tacRemote**, mostrando as funcionalidades mais importantes do sistema e a sua relação com o utilizador. Fezemos também três pequenas demonstrações do sistema, explicando todos os passos para efectuar essas operações, que forma a introdução de um novo utilizador no sistema, a criação de uma nova obra e visualização dos resultados através de gráficos.

6. MÓDULOS DO SISTEMA

"Computer science is no more about computers than astronomy is about telescopes"

Edsger Dijkstra

Tópicos

6.1 INTRODUÇÃO

Neste capítulo pretende-se explicar a relação entre cada um dos módulos necessário. O sistema completo é composto por **cinco** módulos, como podemos ver na figura na figura 30, marcados com o símbolo 1, cada um deles conecta-se com o outro de uma maneira sequencial através de diferentes tecnologias, podemos ver isso através das setas e vamos explorar cada um deles nos pontos seguintes.

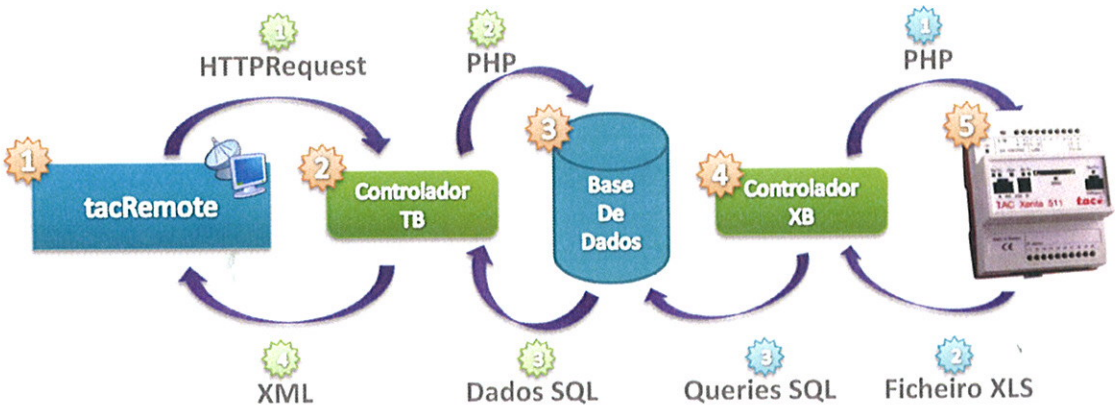


FIGURA 30 – MÓDULOS DO SISTEMA

6.2 MÓDULO TACREMOTE

O sistema tacRemote (para além das funções já explicadas anteriormente) possui também um módulo que é responsável por comunicar com o módulo **Controlador TB**.

Este módulo, como podemos na figura 31, tem a capacidade de enviar e receber mensagens, ou seja o módulo faz pedidos por HTTPRequest, para o módulo **Controlador TB**, e fica a espera de receber os resultados, estes resultados são devolvidos em XML, que depois são transformados para poderem ser apresentados no sistema.

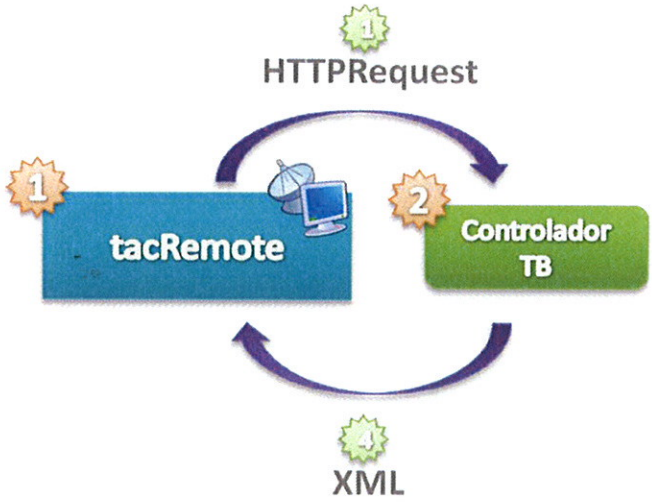


FIGURA 31 – O MÓDULO TACREMOTE

O módulo **tacRemote** é composto por **quatro** componentes que fazem com que este módulo efectue as funções necessárias para o processo descrito na figura 32. Basicamente são quatro os componentes deste módulo (numerados na figura 32), o **primeiro** é responsável por enviar os dados para o **Controlador TB**, neste caso um pedido http, o **segundo** módulo é responsável por apanhar os dados que são retornados pelo **Controlador TB** e reencaminha-los para o **terceiro** módulo que faz a interpretação destes dados e transforma-os numa estrutura standard e entrega-a ao **quarto** módulo que será responsável por mostra estes valores, este processo é desencadeado sempre que seja necessário actualizar algo no sistema.

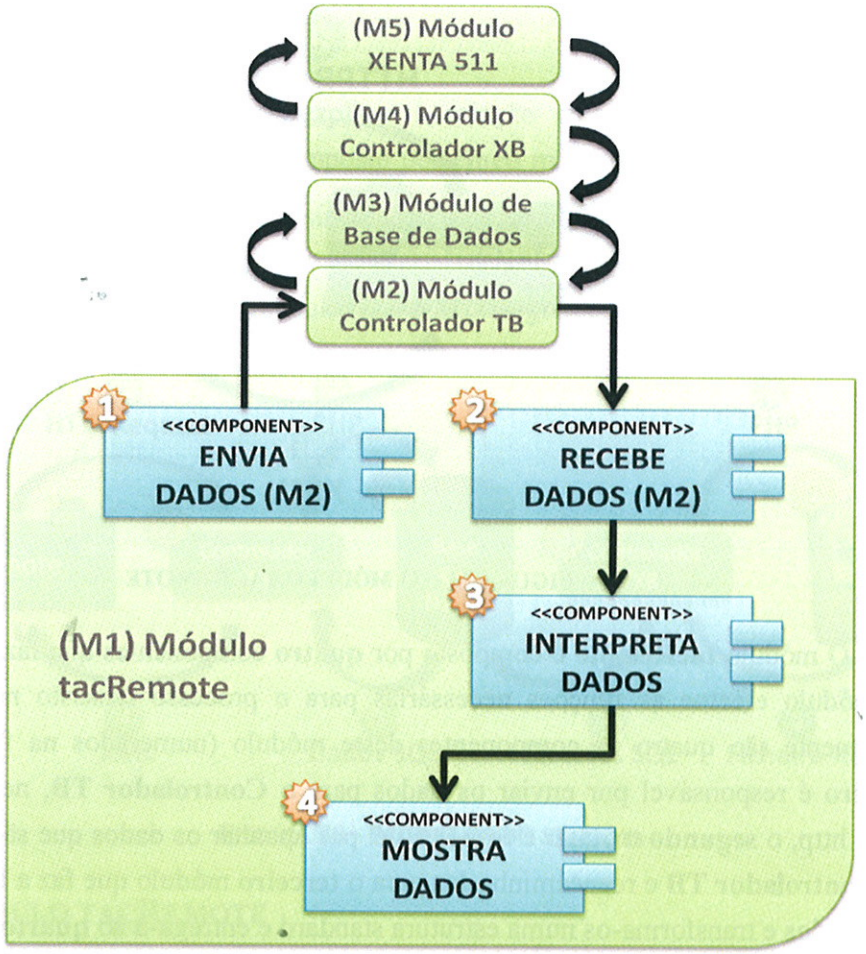


FIGURA 32 - DIAGRAMA DE COMPONENTES DO MÓDULO TACREMOTE

6.3 MÓDULO CONTROLADOR TB

Entre o primeiro módulo (módulo **tacRemote**) e o módulo da **Base de Dados** existe um módulo Contr**olador TB** que é responsável por fazer a comunicação entre ambos, como podemos ver na figura 33, foi visto no ponto anterior que o módulo **controlador TB** recebia um pedido por parte do módulo **tacRemote** (ver figura 31), após receber esse pedido, este módulo (desenvolvido em PHP), após analisar o pedido, faz perguntas à base de dados com base nesses pedidos, após obter a resposta por parte da base de dados normalmente em tabelas, este módulo vai ser responsável por fazer uma transformação dessas tabelas num formato XML, para que o módulo **tacRemote** consiga efectuar a leitura desses valores.

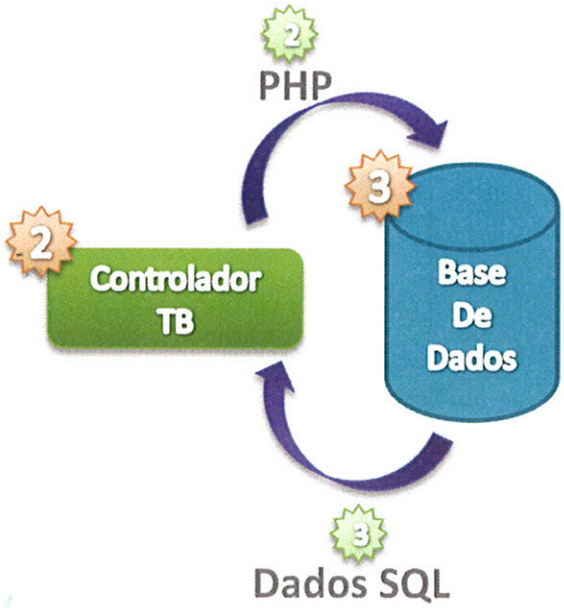


FIGURA 33 - O CONTROLADOR TB

O módulo **Controlador TB** (tacRemote/Base de Dados), é constituído por **seis** componentes, enumerados na figura 34, o **primeiro** módulo tem a responsabilidade de receber os pedidos efectuados pelo módulo **tacRemote**, a partir do momento que recebe esses dados, passa os mesmos para o **segundo** componente que tem a função de interpretar esses dados e transformá-los de maneira que o **terceiro** módulo consiga enviar esses dados para o módulo da base de dados. O **quarto** módulo fica responsável por apanhar o retorno do módulo de **Base de Dados**, neste caso normalmente são tabelas de valores, e passar esses valores para o **quinto** módulo, que vai interpretar esses valores, gerar uma estrutura de dados para que o **sexto** e último módulo consiga enviar esses dados para o módulo **tacRemote**, este processo está descrito na figura 34 e pode ser desencadeado várias vezes.

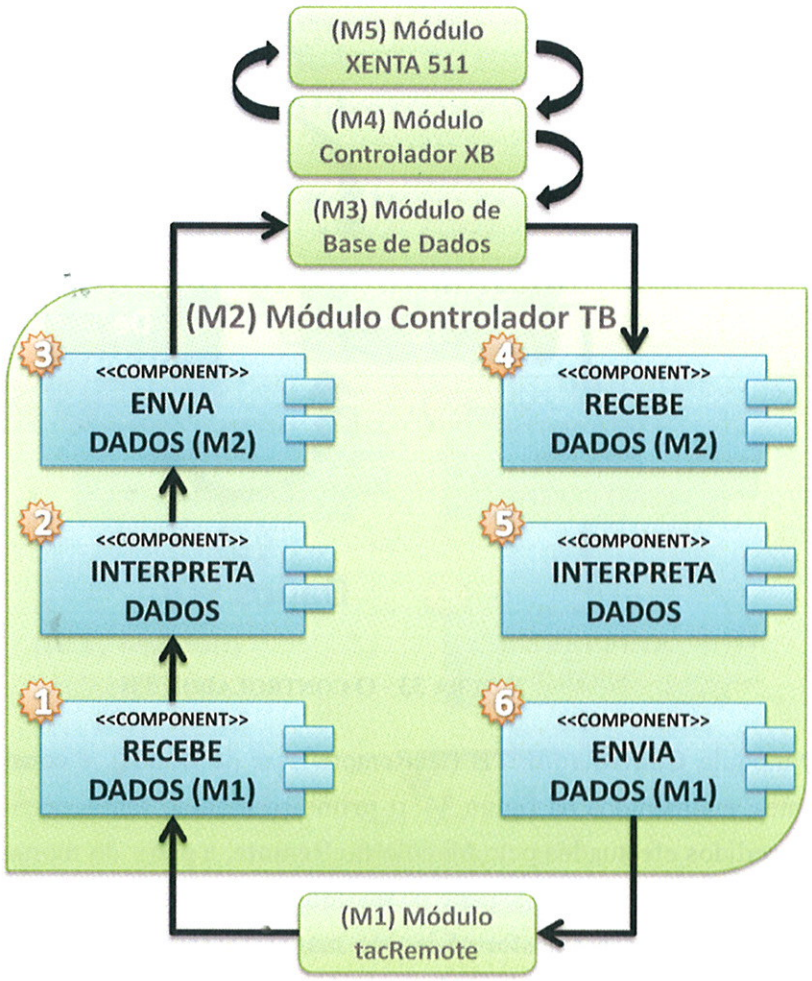


FIGURA 34 - DIAGRAMA DE COMPONENTES DO MÓDULO CONTROLADOR TB

6.4 MÓDULO DE BASE DE DADOS

O módulo da **Base de Dados**, foi implementado fazendo uso da linguagem MySQL, basicamente recebe uma pergunta (query) do módulo **controlador TB** ou do módulo **controlador XB**, analisa essa pergunta e se esta for valida faz a query a base de dados e retorna o resultado dessa query para o **controlador TB** (não retorna qualquer tipo de resposta ao módulo **Controlador XB**), como podemos ver na figura 35.

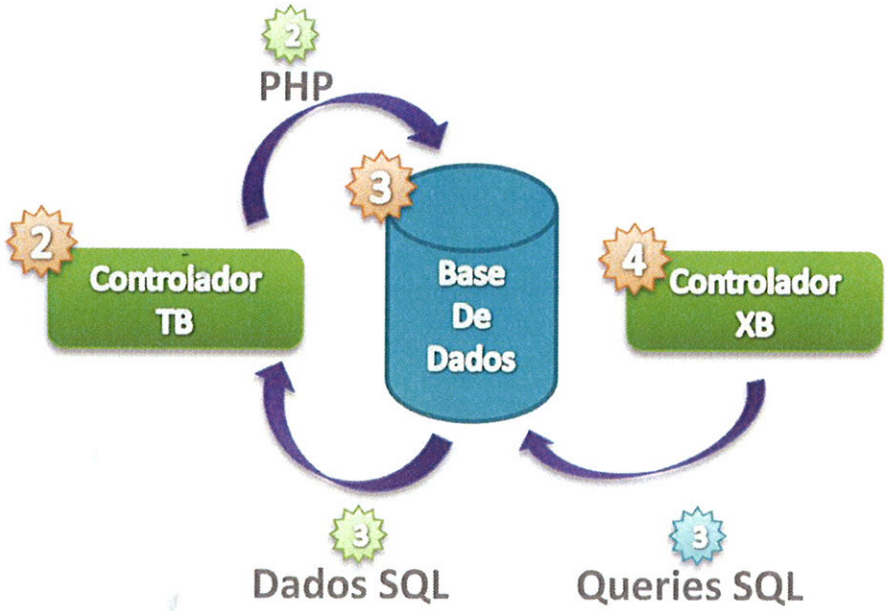


FIGURA 35 - O MÓDULO DA BASE DE DADOS

O módulo de Base de Dados é composto por **cinco** componentes, enumerados na figura 36, o **primeiro** tem como responsabilidade receber dados do módulo **Controlador TB** e enviar esses dados para o **segundo** que tem como função interpretar os dados que recebe do primeiro componente mas também do **terceiro** componente, que recebe dados do módulo **Controlador XB** e após validar esses dados envia-os para o **quarto** componente que tem como função fazer queries à base de dados e enviar o resultado das perguntas feitas pelo **primeiro** componente para o **quinto** componente e finalmente este componente entrega esses resultados ao módulo **Controlador TB**, este processo está descrito na figura 36 e pode ser desencadeado várias vezes.

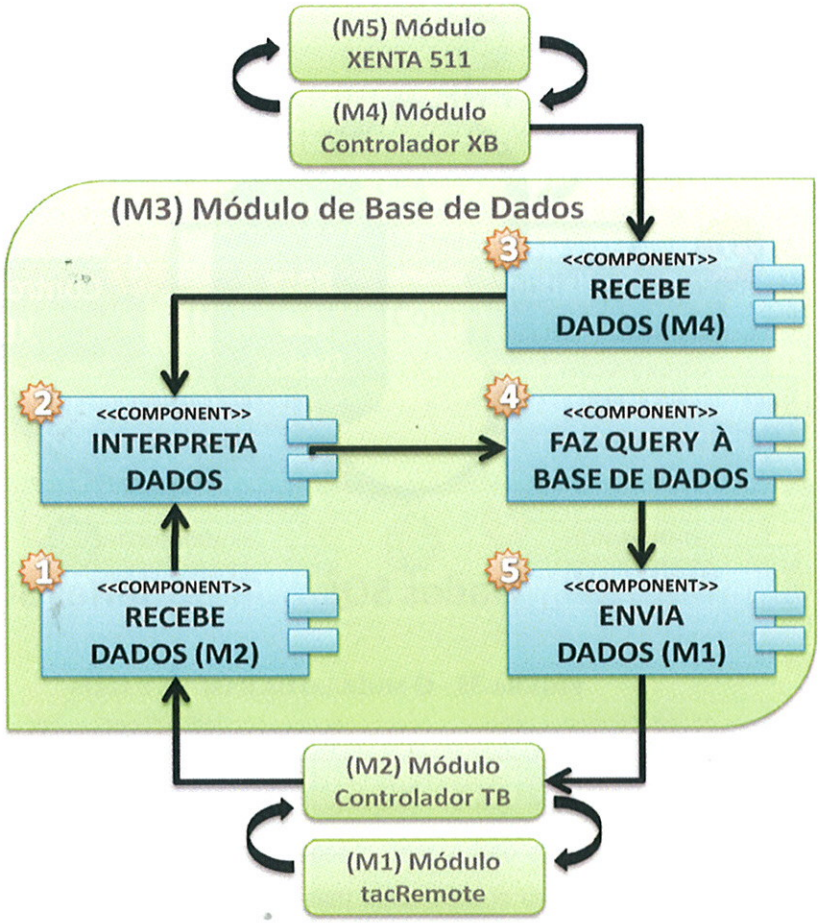


FIGURA 36 - DIAGRAMA DE COMPONENTES DO MÓDULO DE BASE DE DADOS

6.5 MÓDULO CONTROLADOR XB

O módulo **controlador XB** é responsável pela comunicação entre o módulo de **Base de Dados** e o **TAC Xenta 511®**, o processo de funcionamento está descrito na figura 37, ou seja existe um ciclo (feito de minuto a minuto) que vai ler os valores do dispositivo **TAC Xenta 511®**, este gera um ficheiro XLS, que por sua vez é transformado em XML. O módulo **Controlador XB**, interpreta essa estrutura XML cria as queries e introduz esses valores na base de dados.

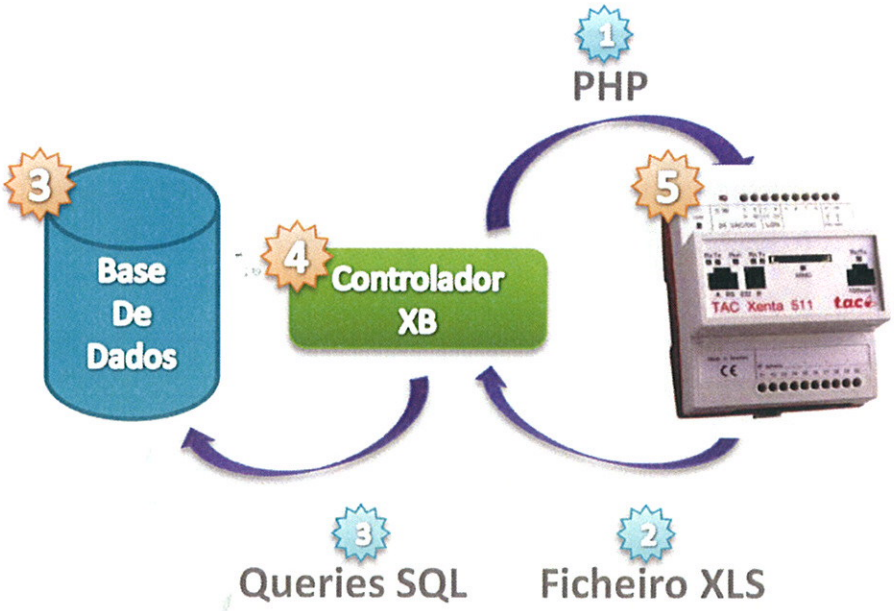


FIGURA 37 – O MÓDULO CONTROLADOR XB

O Módulo **Controlador XB** é composto por **quatro** módulos enumerados na figura 38, o **primeiro** componente tem com responsabilidade enviar pedidos periodicamente (neste caso de minuto a minute) para o dispositivo **TAC Xenta 511®**, o **segundo** tem como função apanhar o retorno do **TAC Xenta 511®**, que neste caso é um ficheiro binário Excel (XLS), e enviar esse ficheiro para o **terceiro** componente, este é capaz de interpretar esse ficheiro e fazer uma transformação para XML e envia esses dados para o **quarto** componente, este cria as queries com base no XML recebido e envia essas queries para o módulo de **Base de Dados**, este processo está descrito na figura 38 e pode ser desencadeado várias vezes.

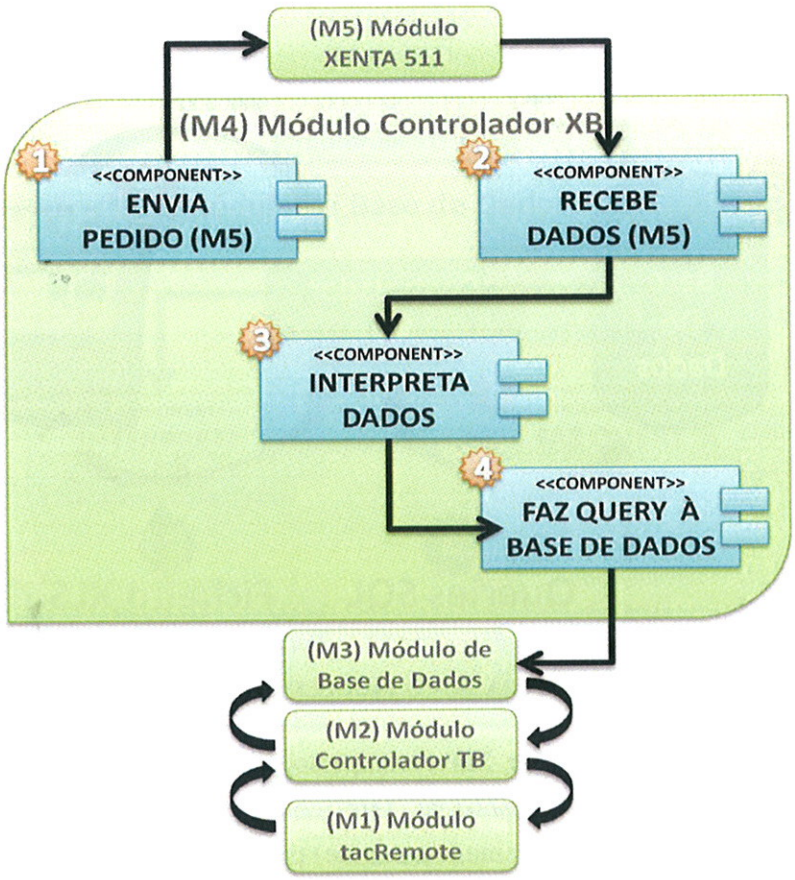


FIGURA 38 - DIAGRAMA DE COMPONENTES DO MÓDULO CONTROLADOR XB

6.6 RESUMO

Neste capítulo demos uma visão geral sobre os módulos existentes no sistema, vimos que era composto por cinco módulos, tentamos também explorar cada um desses módulos, e apresentar a constituição desses módulos, e explicar o que cada um dos componentes que constituem esses módulos fazem, como se interligam entre si, quais são os parâmetros que passam e recebem uns dos outros.

7. Conclusão e Trabalhos Futuros

"I think computer viruses should count as life. I think it says something about human nature that the only form of life we have created so far is purely destructive. We've created life in our own image"

Stephen Hawking

Tópicos

7.1 INTRODUÇÃO

Este capitulo apresenta as conclusões gerais de todo os trabalho de investigação relativo à concepção e implementação do sistema tacRemote. O capítulo identifica e descreve ainda alguns aspectos em aberto que merecem trabalho futuro.

7.2 CONCLUSÃO

No inicio deste trabalho de investigação identificaram-se as motivações que levaram à realização deste trabalho, salientando a importância que uma aplicação deste género terá no contexto da monitorização automática na área da construção civil.

Com o objectivo de criar uma aplicação que inovasse neste campo, recorreu-se a utilização de tecnologias recentes, e nas versões mais actuais possíveis (FLEX, PHP e MySQL), de modo que a aplicação perdure o mais tempo possível, embora sabendo que no contexto de aplicações informáticas isto seja muito difícil devido à constante evolução da informática

Da pesquisa efectuada na literatura existente sobre aplicações idênticas verificou-se que não existe muita variedade neste âmbito, e quanto as aplicações existentes verifica-se que são muito rígidas, ou seja não são flexíveis o suficiente para adaptar-se as necessidades especificas de cada pessoa. Para além disso este sistema são também proprietários das marcas que fabricam os sensores, isto faz com que o uso desses sensores implique que tenhamos necessariamente de usar o software desenvolvido por essas empresas, isto foi um dos problemas que tivemos por causa da pouca flexibilidade dos dispositivos usados.

No que diz respeito à metodologia, verificou-se que devido a dimensão e especificidade do sistema tacRemote, melhor que enquadra-lo numa só metodologia, seria utilizar o UML (Unified Modeling Language). Sendo o UML uma linguagem formal, concisa e compreensível, optou-se por, com base nas ideias apresentadas, recorrer a este, para especificar um conjunto de diagramas que descrevessem de forma clara o problema e a implementação do sistema, nomeadamente diagrama de casos de utilização, diagramas de classes e diagramas de componentes. Finalmente tivemos a necessidade de usar uma metodologia para fazer a representação das interfaces do sistema, para tal utilizou-se a metodologia dos protótipos abstractos canónicos.

Com o objectivo de melhorar a performance dos utilizadores tentou-se fazer um sistema simples e intuitivo, ou seja interfaces consistentes e que fossem fáceis de aprender e utilizar, bem como fazer com que utilizadores mais avançados possam tirar partido desta para aumentar a produtividade.

Temos como resultado uma aplicação robusta, modular e simples que permite aos utilizadores monitorizar e analisar a temperatura do betão e com base nesta temperatura permitir a previsão de resistências, resistência médias para poderem auxiliar os responsáveis das obras a tomar decisões.

Com base neste trabalho de investigação fizemos em colaboração com a empresa Softventure Consultoria & Tecnologia uma aplicação que faz a monitorização da temperatura, cálculo de resistências previstas, geração de gráficos de valores, gestão de utilizadores, esta aplicação foi implementada numa empresa, mas por questões de confidencialidade não podemos divulgar nem o nome da empresa nem o nome do software, apenas podemos adiantar algumas características, que a aplicação foi desenvolvida numa tecnologia diferente da utilizada no âmbito deste trabalho embora os dispositivos de leitura de valores fossem os mesmos.

A grande vantagem deste projecto é sem duvida o facto deste se modular e se adaptara às necessidades da cada pessoa ou empresa indo de encontro às necessidades dessa pessoas na procura de auxiliar e facilitar o desempenho dessas mesma pessoa de maneira eficiente mas também fazer com que o realização dessas tarefas sejam mais eficazes.

7.3 Trabalhos Futuros

A realização de qualquer trabalho de investigação encontra-se necessariamente incompleta e suscita linhas em aberto que poderão ser alvas de trabalho futuro. Numa perspectiva objectiva e pragmática identificam-se os seguintes aspectos principais: **(6.9.1)** Continuação da implementação do sistema tacRemote; **(6.9.2)** Alargadamente do suporte de sensores; **(6.9.3)** Generalização do sistema tacRemote;

7.3.1 Continuação da implementação do sistema tacRemote

Nova iterações poderão ser feitas para melhorar e implementar novos casos de utilização, de acordo com o feedback que vamos obter dos utilizadores do sistema tacRemote, podemos proceder ao refinamento e melhoria no sistema. As sugestões dos utilizadores devem ser tidas em conta porque poderão servir para evoluir para novos casos de utilização e consequentemente novas iterações em termos de desenvolvimento.

7.3.2 INTEGRAÇÃO DE NOVOS SENSORES

O sistema tacRemote foi desenhado para se fácil a integração de novos sensores, esta versão foi testada com sensores de temperatura, por questões temporais compreensíveis não foi possível testar com outro tipos de sensores, será importante fazer os ajustes necessários para que o sistema seja o mais robusto e extensível possível.

7.3.3 GENERALIZAÇÃO DO SISTEMA TACREMOTE

Finalmente pensamos que seria uma boa ideia generalizar mais o sistema tacRemote, ou seja fazer com este sistema cubra uma grande parte das necessidades dos utilizadores que usem este sistema, bem como que seja possível parametrizar o sistema Agosto do utilizador, quer seja em temos estéticos (cores, tipos de letra) quer em termos de poder escolher quais os componente que querem ver ou não.

REFERÊNCIAS BIBLIOGRÁFICAS

[TACCB] TAC CORPORATE BROCHURE 2008:

http://TAC.com/data/internal/data/07/03/1212427722606/_2008TAC_CorporateBrochureSE_A4.pdf

[TAC100] TAC XENTA™ 100:

http://TAC.com/data/internal/data/03/95/1145460081748/SDS_XENTA100_letter.pdf

[TAC121] TAC XENTA™ 121:

http://TAC.com/data/internal/data/04/99/1168894015694/Xenta121_ProgrammableZoneControl_A4.pdf

[TAC400] TAC XENTA™ 400:

http://TAC.com/data/internal/data/05/12/1170864485332/XENTA+400.pdf

[TAC401] TAC XENTA™ 400:

http://www.ibscltd.co.uk/oandm/XENTA/401.pdf

[T420/450] TAC XENTA™420A/450A SERIES I/O MODULES:

http://TAC.com/data/internal/data/05/00/1169148639818/Xenta420_450A_IOmodules.pdf

[TAC527] TAC XENTA™ 527 COMPLETE CONTROL VIA THE INTERNET:

http://TAC.com/data/internal/data/06/75/1203432402395/Xenta_527_A4.pdf

[TAC700] TAC XENTA 700 EVERYTHING YOU NEED TO MONITOR AND CONTROL YOUR BUILDING:

http://TAC.com/data/internal/data/05/95/1184768888249/TAC+Xenta+700+brochure_A4.pdf

[TAC911] TAC XENTA™ 911:

http://TAC.com/data/internal/data/05/00/1169214590202/Xenta911_Controller.pdf

[TAC913] TAC XENTA™ 913:

http://TAC.com/data/internal/data/06/75/1203432308129/Xenta913_Controller_A4.pdf

[TAC300] TAC XENTA™ 300:

http://TAC.com/data/internal/data/05/00/1169214073955/XENTA300_ProgrammableController.pdf

[TACSO] TAC VISTA™ SYSTEM OVERVIEW:

http://TAC.com/data/internal/data/06/81/1205413085077/Vista_System_Overview_A4.pdf

[CLCTL] HTTP API SPECIFICATION V2.3.4 - 18 JULY 2008:

http://www.clickatell.com/downloads/http/Clickatell_HTTP.pdf

[TAC511-A] COMPLETE CONTROL VIA THE INTERNET:

http://www.TAC.com/data/internal/data/05/00/1169146940063/Xenta511_ControllerViaInternet.pdf

[TAC511-B] A WEB SERVER DEVICE THAT MAKES IT EASY TO MONITOR LONWORKS®-BASED NETWORKS VIA THE INTERNET:

http://www.TAC-global.com/pub/products/vista/vistapdfs/Xenta511.pdf

[IBMDB2] DB2 PRODUCT FAMILY

http://www-306.ibm.com/software/data/db2/

[ORACLE] ORACLE 11G, SIEBEL, PEOPLESOFT

http://www.oracle.com/global/pt/index.html

[MSACC] MICROSOFT OFFICE ACCESS 2007

http://office.microsoft.com/pt-pt/access/default.aspx

[MSSQL] MICROSOFT SQL SERVER

http://www.microsoft.com/sqlserver/2008/en/us/default.aspx

[MYSQL] MYSQLTHE WORLD'S MOST POPULAR OPEN SOURCE DATABASE

http://www.mysql.com/

[PGSQL] POSTGRESQL THE WORLD'S MOST ADVANCED OPEN SOURCE DATABASE

http://www.postgresql.org/

[SQLITE] SQLITE HOME PAGE

http://www.sqlite.org/

[EC2008] ECHELON CORPORATION

http://www.echelon.com/company/

[LW2008] LONWORKS CORE TECHNOLOGY

http://www.echelon.com/developers/lonworks/default.htm

[WSQL2008] WHY MYSQL

http://www.mysql.com/why-mysql/

[CSQL2008] CASE STUDIES

http://www.mysql.com/why-mysql/case-studies/

[LNW2008] LONMARK - CERTIFICATIONS

http://www.lonmark.org/certifications/

[HPL2008] THE HISTORY OF PROGRAMMING LANGUAGES

http://oreilly.com/news/graphics/prog_lang_poster.pdf

[HISPHP] PHP: HISTORY OF PHP AND RELATED PROJECTS - MANUAL

http://pt.php.net/history

[HISJAVA] THE HISTORY OF JAVA TECHNOLOGY

http://www.java.com/en/javahistory/

[HISJS] O'REILLY NETWORK -- JAVASCRIPT: HOW DID WE GET HERE?

http://www.oreillynet.com/pub/a/javascript/2001/04/06/js_history.html

[HISXML] EXTENSIBLE MARKUP LANGUAGE (XML)

http://www.w3.org/XML/

[SV2008] SOFTVENTURE CONSULTURIA & TECNOLOGIA

http://www.softventure.pt/index.php?option=com_content&view=article&id=22&Itemid=2

[NATO68] NATO SOFTWARE ENGINEERING CONFERENCE 1968

http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

[EWD340] THE HUMBLE PROGRAMMER

http://www.cs.utexas.edu/users/EWD/ewd03xx/EWD340.PDF

[SCH75] SCHWARTZ, J. I., CONSTRUCTION OF SOFTWARE. IN: PRACTICAL STRATEGIES FOR DEVELOPING LARGE SYSTEMS. MENLO PARK: ADDISON-WESLEY, 1ST. ED., 1975.

[CHR2001] CHRISTIAN REIS., CARACTERIZAÇÃO DE UM MODELO DE PROCESSO PARA PROJECTOS DE SOFTWARE LIVRE. TESTE DE MESTRADO. INSTITUTO DE CIÊNCIAS MATEMÁTICA E COMPUTAÇÃO. SÃO CARLOS, SÃO PAULO ABRIL DE 2001

[WWR1970] MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS

http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

[BB1988] A SPIRAL MODEL OF SOFTWARE DEVELOPMENT AND ENHANCEMENT

http://www.cs.usu.edu/~supratik/CS%205370/r5061.pdf

[SOMM95] SOMMERVILLE, I. SOFTWARE ENGINEERING. 5TH. ED. ADDISON-WESLEY, 1995.

[ASCV05] SILVA A, VIDEIRA C. UML METODOLOGIAS E FERRAMENTAS CASE, 2ª EDIÇÃO, VOL. I, 2005.

[PMAN97] PRESSMAN, R. S., SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH. 4TH. ED. MCGRAW-HILL, 1997.

[PJALO97] P. JALOTE, AN INTEGRATED APPROACH TO SOFTWARE ENGINEERING, SPRINGER, NEW YORK , THIRD EDITION, 2005, SECOND EDITION, 1997

[SUNOOP] LESSON: OBJECT-ORIENTED PROGRAMMING CONCEPTS (THE JAVA™; TUTORIALS > LEARNING THE JAVA LANGUAGE)

http://java.sun.com/docs/books/tutorial/java/concepts/

[CAP2003] CANONICAL ABSTRACT PROTOTYPES FOR ABSTRACT VISUAL AND INTERACTION DESIGN, LARRY L. CONSTANTINE UNIVERSITY OF TECHNOLOGY, SYDNEY, (AUSTRALIA) CONSTANTINE & LOCKWOOD LTD

http://www.foruse.com/articles/abstract.pdf

[CS2003] CAROLYN SNYDER, PAPER PROTOTYPING MORGAN KAUFMANN, 2003

ANEXOS

ANEXO A – GUIA DE INSTALAÇÃO

[Manual De Instalação]

1. Introdução

O presente documento serve para auxiliar a instalação do sistema tacRemote, que segue o padrão de arquitectura de software **MVC** (Model View Controller ou Modelo Vista Controlador).

Com o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação entre os dados (**Model**) e o layout (**View**). Desta forma, alterações feitas no layout não afectam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

O MVC resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interacção com o utilizador, introduzindo um componente entre os dois: o **Controller**. A figura 1 mostra as relações no modelo.

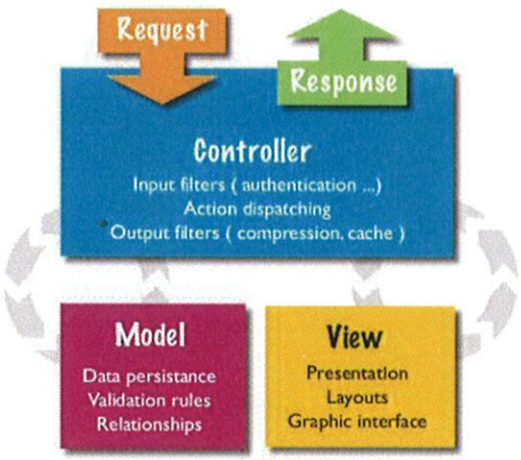


FIGURA 1 – ARQUITECTURA MVC[2]

[2] Imagem retirada de http://akelos.org/docs/tutorials/booklink

2. Instalação Do tacRemote

Como já foi referindo na introdução este software segue a arquitectura de software MVC, nos pontos seguintes mostras como fazer a instalação do sistema tacRemote.

2.1 MODELO MVC

O modelo MVC (Model View Controller), como o seu nome indica é constituído por três componentes que compõem esta arquitectura, vamos mostra quais são esses componentes e como fazer a instalação. A instalação dos componentes não requerem que seja seguida a ordem do documento, ou seja pode primeiro fazer o ponto 2.12 e depois o 2.11, mas aconselha-se a ordem do documento.

2.1.2 MODEL

Model ou **Modelo** são os dados do sistema, normalmente bases de dados, para este caso escolheu-se uma base de dados **MySQL**, embora pudesse ter sido escolhida outro tipo, e futuramente pode ser mudada sem que para isso seja necessário mudar os restantes componentes, de facto isto é uma das principais vantagens da arquitectura MVC.

A figura 2 mostra as tabelas necessárias para o funcionamento do sistema tacRemote.

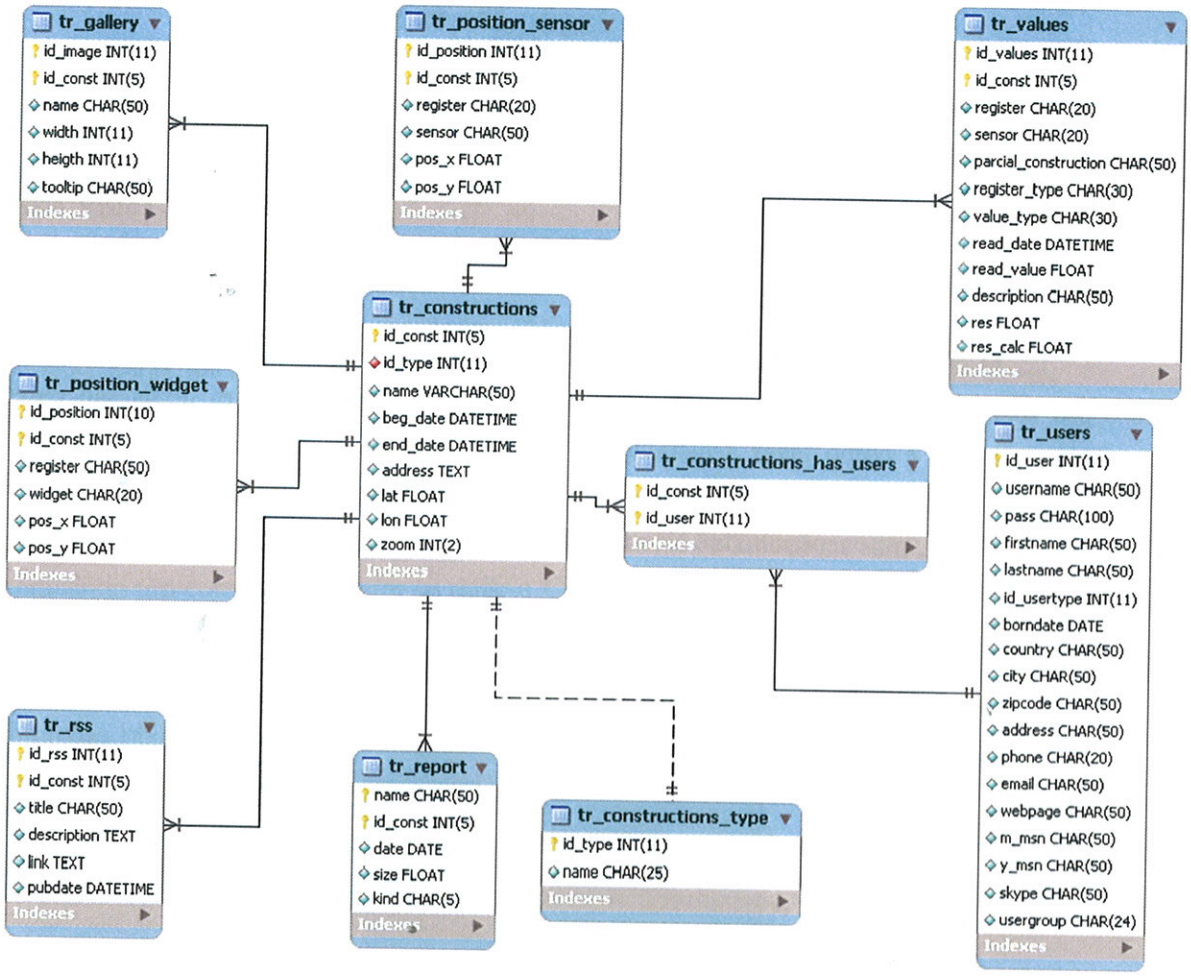


FIGURA 2 – DIAGRAMA DE ENTIDADES E RELACIONAMENTOS DA BASE DE DADOS

Para criar a tabelas basta seguir os passos seguintes, onde temos os comandos **MySQL** necessários para criar as tabelas necessárias.

```
CREATE
  TABLE tr_constructions(
    id_const INT(5) NOT NULL auto_increment
    ,id_type INT(11) NOT NULL DEFAULT '0'
    ,name VARCHAR(50) NOT NULL DEFAULT ''
    ,beg_date datetime NOT NULL
    ,end_date datetime NOT NULL
    ,address text NOT NULL
    ,lat FLOAT NOT NULL DEFAULT '0'
    ,lon FLOAT NOT NULL DEFAULT '0'
    ,zoom INT(2) NOT NULL DEFAULT '0'
    ,PRIMARY KEY (
      id_const
      ,id_type
    )
    ,KEY fk_tr_constructions_tr_constructions_type(id_type)
  ) ENGINE = MyISAM AUTO_INCREMENT = 10 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
```

```
CREATE
   TABLE tr_constructions_type(
      id_type INT(11) NOT NULL DEFAULT '0'
      ,name CHAR(25) DEFAULT NULL
      ,PRIMARY KEY (id_type)
   ) ENGINE = InnoDB DEFAULT CHARSET = latin1
;
CREATE
   TABLE tr_gallery(
      id_image INT(11) NOT NULL auto_increment
      ,id_obra INT(11) NOT NULL DEFAULT '0'
      ,name CHAR(50) NOT NULL DEFAULT ''
      ,width INT(11) NOT NULL DEFAULT '0'
      ,heigth INT(11) NOT NULL DEFAULT '0'
      ,tooltip CHAR(50) NOT NULL DEFAULT ''
      ,PRIMARY KEY (id_image)
   ) ENGINE = MyISAM AUTO_INCREMENT = 114 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
CREATE
   TABLE tr_position_sensor(
      id_position INT(11) NOT NULL auto_increment
      ,id_const INT(3) NOT NULL DEFAULT '0'
      ,register CHAR(20) NOT NULL DEFAULT ''
      ,sensor CHAR(50) NOT NULL DEFAULT ''
      ,pos_x FLOAT NOT NULL DEFAULT '0'
      ,pos_y FLOAT NOT NULL DEFAULT '0'
      ,PRIMARY KEY (id_position)
   ) ENGINE = MyISAM AUTO_INCREMENT = 11624 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
CREATE
   TABLE tr_position_widget(
      id_position INT(10) NOT NULL auto_increment
      ,id_const CHAR(25) NOT NULL DEFAULT ''
      ,register CHAR(50) NOT NULL DEFAULT ''
      ,widget CHAR(20) NOT NULL DEFAULT ''
      ,pos_x FLOAT NOT NULL DEFAULT '0'
      ,pos_y FLOAT NOT NULL DEFAULT '0'
      ,PRIMARY KEY (id_position)
   ) ENGINE = MyISAM AUTO_INCREMENT = 5858 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
CREATE
   TABLE tr_report(
      name CHAR(50) NOT NULL DEFAULT ''
      ,id_const INT(5) DEFAULT NULL
      ,DATE DATE DEFAULT NULL
      ,SIZE FLOAT DEFAULT NULL
      ,kind CHAR(5) DEFAULT NULL
      ,PRIMARY KEY (name)
   ) ENGINE = InnoDB DEFAULT CHARSET = latin1
;
CREATE
   TABLE tr_rss(
      id_rss INT(11) NOT NULL auto_increment
      ,title CHAR(50) NOT NULL
      ,description text NOT NULL
      ,link text
      ,pubdate datetime NOT NULL
```

```
    ,PRIMARY KEY (id_rss)
  ) ENGINE = MyISAM AUTO_INCREMENT = 4 DEFAULT CHARSET = utf8
AUTO_INCREMENT = 1
;
CREATE
  TABLE tr_user_has_constructions(
    id_const INT(5) NOT NULL DEFAULT '0'
    ,id_user INT(5) NOT NULL DEFAULT '0'
    ,PRIMARY KEY (
      id_const
      ,id_user
    )
    ,KEY fk_tr_constructions_has_tr_users_tr_constructions(id_const)
    ,KEY fk_tr_constructions_has_tr_users_tr_users(id_user)
  ) ENGINE = MyISAM DEFAULT CHARSET = latin1
;
CREATE
  TABLE tr_users(
    id_user INT(11) NOT NULL auto_increment
    ,username CHAR(50) NOT NULL DEFAULT ''
    ,pass CHAR(100) NOT NULL DEFAULT ''
    ,firstname CHAR(50) NOT NULL DEFAULT ''
    ,lastname CHAR(50) NOT NULL DEFAULT ''
    ,id_usertype INT(11) NOT NULL DEFAULT '0'
    ,borndate DATE NOT NULL DEFAULT '0000-00-00'
    ,country CHAR(50) NOT NULL DEFAULT ''
    ,city CHAR(50) NOT NULL DEFAULT ''
    ,zipcode CHAR(50) NOT NULL DEFAULT ''
    ,address CHAR(50) NOT NULL DEFAULT ''
    ,phone CHAR(20) NOT NULL DEFAULT ''
    ,email CHAR(50) NOT NULL DEFAULT ''
    ,webpage CHAR(50) NOT NULL DEFAULT ''
    ,m_msn CHAR(50) NOT NULL DEFAULT ''
    ,y_msn CHAR(50) NOT NULL DEFAULT ''
    ,skype CHAR(50) NOT NULL DEFAULT ''
    ,usergroup CHAR(24) NOT NULL DEFAULT '0'
    ,PRIMARY KEY (id_user)
  ) ENGINE = MyISAM AUTO_INCREMENT = 12 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
CREATE
  TABLE tr_values(
    id_values INT(11) NOT NULL auto_increment
    ,id_const INT(5) DEFAULT NULL
    ,register CHAR(20) DEFAULT NULL
    ,sensor CHAR(20) DEFAULT NULL
    ,parcial_construction CHAR(50) DEFAULT NULL
    ,register_type CHAR(30) DEFAULT NULL
    ,value_type CHAR(30) DEFAULT NULL
    ,read_date datetime DEFAULT NULL
    ,read_value FLOAT DEFAULT '0'
    ,description CHAR(50) DEFAULT ''
    ,res FLOAT DEFAULT '0'
    ,res_calc FLOAT DEFAULT '0'
    ,PRIMARY KEY (id_values)
  ) ENGINE = InnoDB AUTO_INCREMENT = 114241 DEFAULT CHARSET = latin1
AUTO_INCREMENT = 1
;
```

Depois de criarmos as tabelas, esta fase fica confluída podendo passar para outra fase.

2.1.2 VIEW

O **View** ou **Vista** como o próprio nome indica refere-se a parte de apresentação dos dados ao cliente, mais uma vez o modelo **MVC** permite que no futuro possa ser mudada esta parte, mesmo o tipo de linguagem, sem que com isto os outros componentes sejam afectados.

A instalação desta parte é a mais complexa, visto que temos de meter vários ficheiros e tem de ser feito respeitando uma hierarquia de ficheiros. Para meter os ficheiros temos de ter em conta que devemos meter os ficheiros numa pasta preparada para funcionar com webserver neste caso na pasta **httdocs,** como podemos ver na figura seguinte.

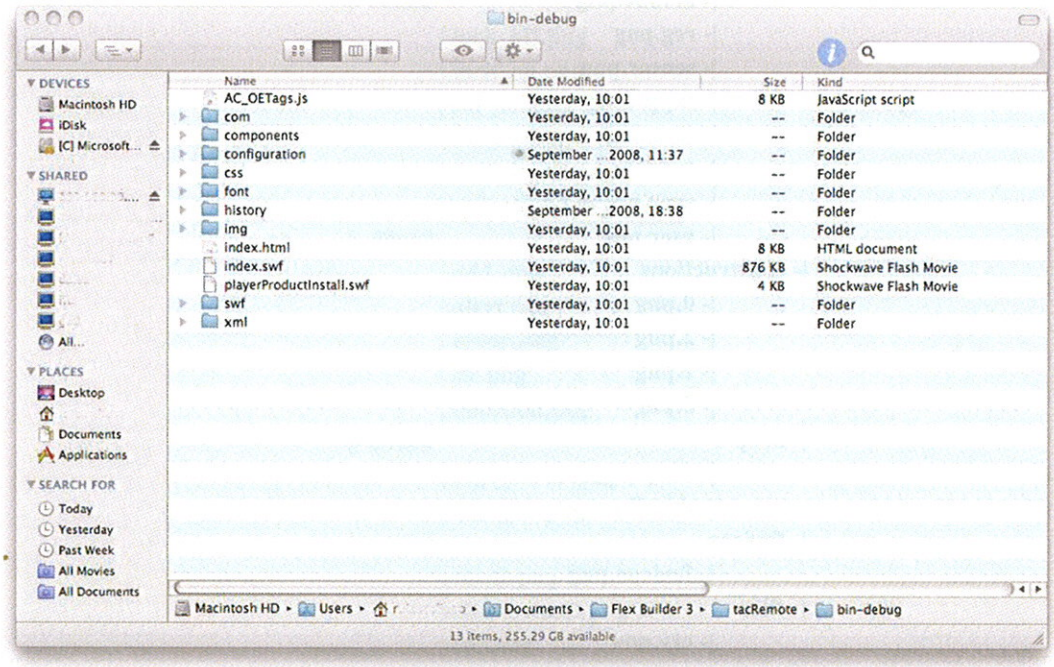


FIGURA 3 – LISTA DE FICHEIROS

A seguinte estrutura mostra todos os ficheiros do módulo View, e a hierarquia entre eles, mais uma vez note-se que é **IMPORTANTE** manter a estrutura e ordem dos ficheiros.

[httdocs]
|- AC_OETags.js

```
|- com
        |- yahoo
                |- maps
                        |-markers
|- components
|- configuration
        |- tacremote.xml
|- css
        |_ style.css
|- font
        |- verdana.ttf
|- history
        |- history.css
        |- history.js
        |- historyFrame.html
|- img
        |- bubbles
                |- bg.png
                |- cal.png
                |- center.png
                |- down.png
                |- mail.png
                |- mobile.png
                |- reg.png
                |- sensor.png
                |- skype.png
                |- text.png
                |- top.png
                |- user-group.png
                |- user.png
        |- constructions
                |- 0.png
                |- 2.png
                |- 1.png
                |- 3.png
        |- drop
                |- drop.png
                |- drop2.png
        |- favicon
                |- favicon.png
        |- files
                |- csv.png
                |- txt.png
                |- doc.png
                |- xls.png
                |- pdf.png
                |- xml.png
        |- logo
                |- 128x128
                        | logo.png
                |- 16x16
                        | 1.png
```

```
                    | 2.png
                    | 3.png
                    | 4.png
                    | 5.png
                    | 6.png
                    | del.png
                    | download.png
                    | favicon.png
                    | find.png
                    | lock.png
                    | mail.png
                    | msn.png
                    | notok.png
                    | ok.png
                    | phone.png
                    | private.png
                    | public.png
                    | site.png
                    | skype.png
                    | yahoo.png
            |- 256x256
                    | logo.png
            |- 32x32
                    | lock_off.png
                    | lock_on.png
                    | rss_off.png
                    | rss_on.png
                    | unlock_off.png
                    | unlock_on.png
            |- 48x48
                    | add.png
                    | alert.png
                    | error.png
                    | ok.png
                    | question.png
            |- 64x64
                    | logo_error.png
                    | logo_ok.png
            |- registers
                    | clock.png
                    | clock_small.png
                    | res.png
                    | res_small.png
                    | term.png
                    | term_small.png
            |- softventure
                    | find.png
                    | logo100x47.png
                    | logo70x33.png
|- index.html
|- index.swf
|- playerProductInstall.swf
```

|- swf
    |- spinner16.swf
    |- spinner22.swf
    |- spinner40.swf
    |- spinner48.swf
|- xml
    |- countries.xml
    |- years.xml

2.1.3 CONTROLLER

O módulo **Controller** ou **Controlador** é o módulo intermediário que fica entre as camadas **View** e **Model**. Esta camada faz a ligação entres as outras camadas, mas um vez devido a modularidade da arquitectura este parte pode ser alterada no futuro sem afectar as outras partes. Neste caso optou-se pela tecnologia **PHP** para implementar o controlador. A figura 4 mostra os ficheiros necessários para implementar o **Controlador**.

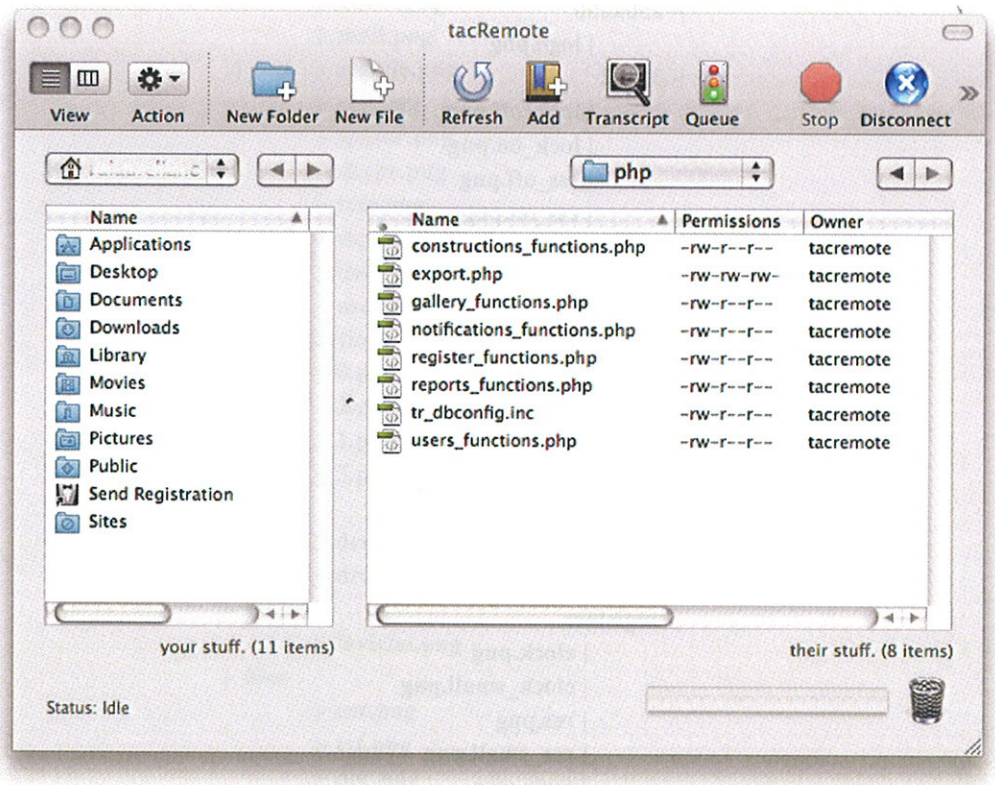


FIGURA 4 – LISTA DE FICHEIROS DO CONTROLADOR

Estes ficheiros têm de ser colocados num servidor que suporte PHP, podendo servidor num local diferente do usado para armazenar os ficheiros do módulo **View**. Depois de colocarmos estes ficheiros temos de efectuar os seguintes passos.

1. Editar o ficheiro **tr_dbconfig.php**, para colocarmos os atributos para fazer a conexão a base de dados.

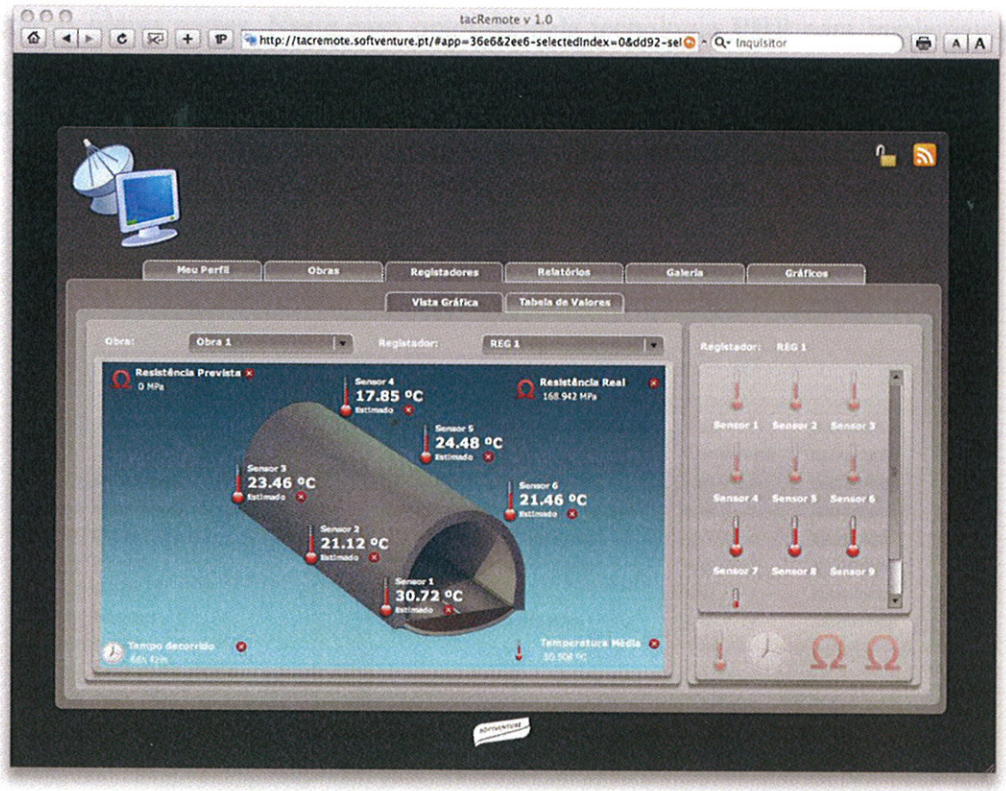
```
<?php
    $host_db="localhost";
    $user_db="xxxxxxxxx";
    $pass_db="xxxxxxxxxx";
    $database="xxxxxxxxxx";
?>
```

2. Depois temos introduzir o caminho completo de onde foi instalado o modulo **Controller** num ficheiro de configuração **XML**, com o nome tacremote.xml que se encontra da directoria httdocs/configuration/, e alteramos o nó **path**.

```
<tacremote>
  <path>http://o.meu.caminho/php</path>
</tacremote>
```

3. Após estes passos o sistema está pronto para funcionar.

ANEXO B – GUIA DE UTILIZAÇÃO



[Manual De Utilização]

1. Introdução

A ferramenta tacRemote, foi desenvolvida com o propósito de controlar e monitorizar obras remotamente, ou seja desde que o utilizador possua uma ligação à internet e um browser compatível, pode aceder ao sistema e efectuar tarefas de acordo com o seu nível de permissões, como vamos poder durante o manual.

Na sua essência o sistema funciona da seguinte forma, são colocados sensores de temperatura (podendo ser de outro tipo) nas obras que estão em constante actualização. Estes valores são lidos pelos sensores e são periodicamente guardados numa base de dados para poderem ser analisados no futuro. O sistema tacRemote permite a monitorização em tempo real destes valores (com atraso máximo de 1 minuto).

Ao longo deste manual será descrita todas as operações que podemos realizar.

2. Autenticar-se

O primeiro passo a fazer é entrar no sistema, para autenticar-se e introduzir o **username** e **password**, como podemos ver na figura 1.

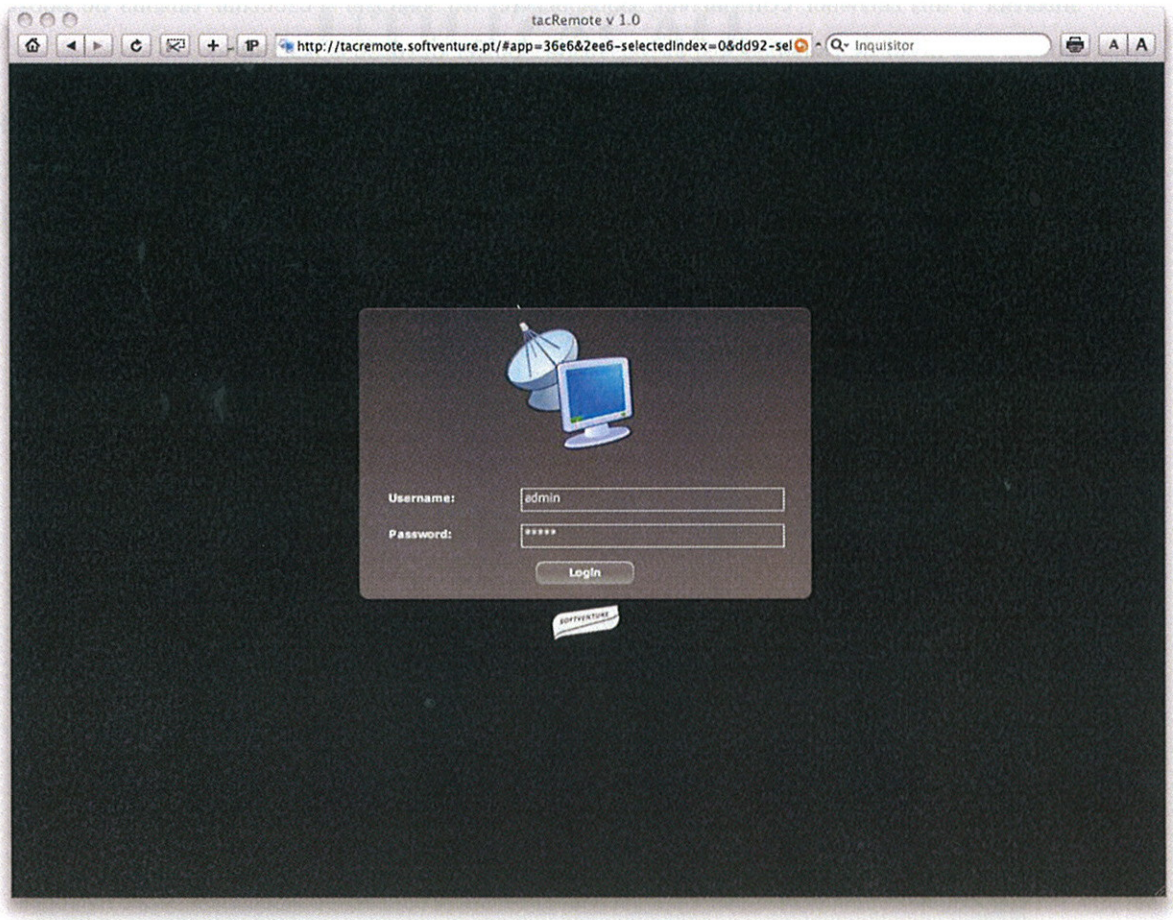


FIGURA 1 – AUTENTICAÇÃO NO SISTEMA

3. Gerir Utilizadores

O Sistema tacRemote permite toda a gestão dos utilizadores do sistema, isto claro dependendo do tipo de utilizador e dos seus privilégios. Basicamente o sistema permite 4 grandes operações (ver, adicionar, alterar e apagar utilizadores), que são descritas nos pontos seguintes, que podem ser efectuadas pelos utilizadores, se tiverem permissões para o fazer. Este sistema admite 3 tipos de utilizadores, **Administrador**, **Operador** e **Cliente**,

O utilizador **Administrador**, é o utilizador com mais privilégios, pode efectuar todas as operações do sistema, que serão descritas ao longo do manual.

O utilizador **Operador**, é um tipo de utilizador com menos permissões que o **Administrador**, pode efectuar quase todas as funções do **Administrador**, excepto a gestão de Utilizadores.

Finalmente o utilizador **Cliente**, possui apenas permissões de leitura, ou seja não tem permissões para alterar ou apagar alguma coisa, excepção feita aos detalhes do seu perfil.

3.1 Adicionar Utilizadores

O sistema permite adicionar utilizadores, embora esta operação seja a que necessita de um nível de privilégios maior, ou seja apenas o utilizador **Administrador** pode criar utilizadores. Na figura 1 podemos ver como criamos um utilizador, para tas basta pressionar a opção **NOVO UTILIZADOR**, como podemos ver na figura 2.

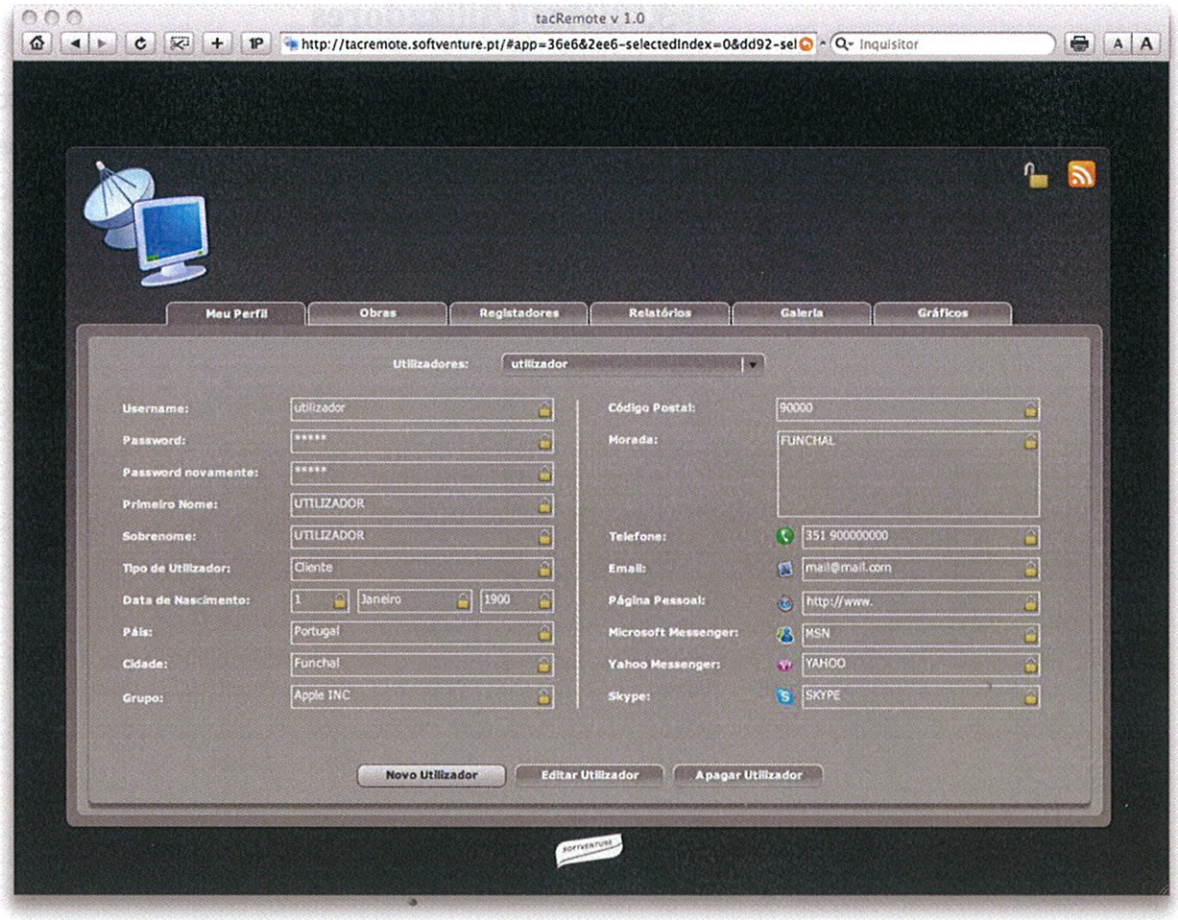


FIGURA 2 – ADICIONAR NOVO UTILIZADOR

Na figura 3, podemos ver que existe uma janela com vários campos, uns são opcionais e outros são obrigatórios, os campos com o símbolo são obrigatórios e tem de verificar algumas condições para serem considerados validos, por exemplo o nome e password tem de possuir pelo menos uma letra entre outros.

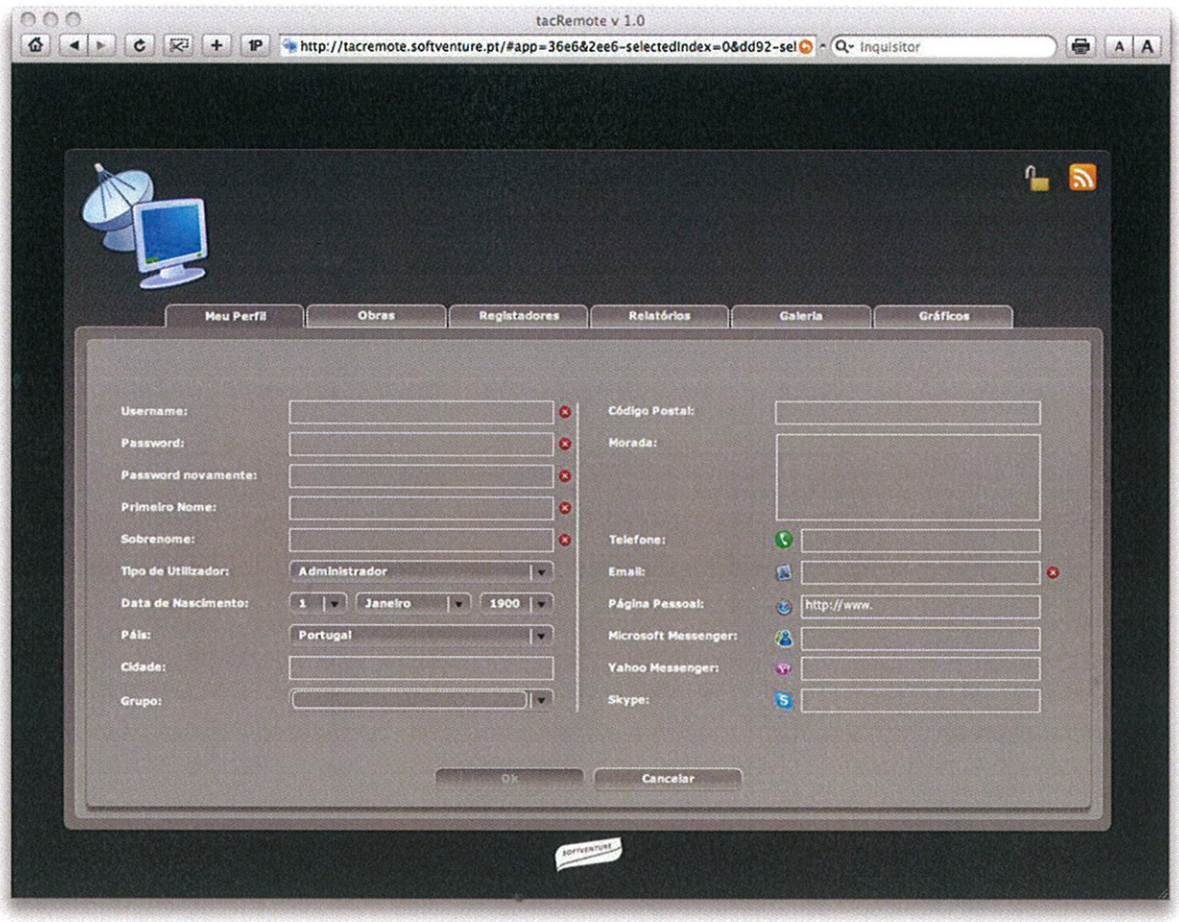


FIGURA 3 – PREENCHER DADOS DO UTILIZADOR

Quando o campo for valido aparece o símbolo , e quando todos os campos estiverem válidos, podemos adicionar o utilizador, pressionando o botão **OK**. Será mostrado um *popup* para confirmar se quer mesmo adicionar um novo utilizador como podemos ver na figura 4.

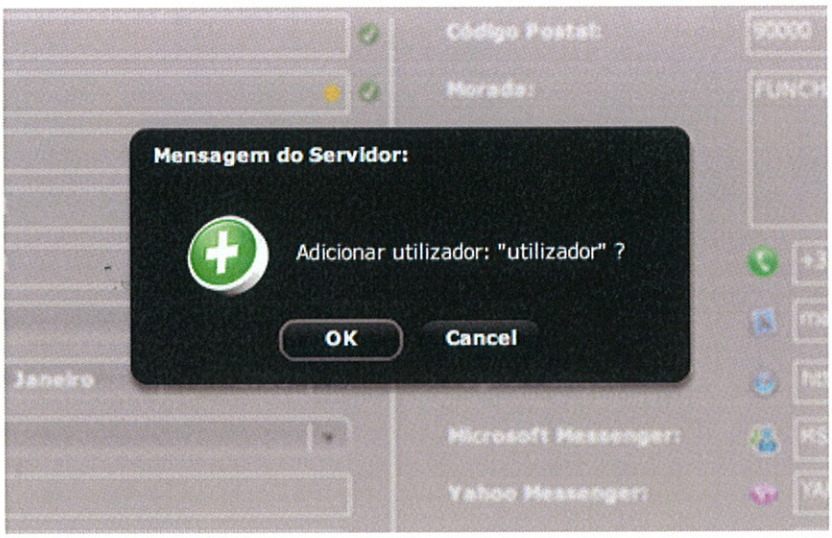


FIGURA 4 – CONFIRMAR A INTRODUÇÃO DE UM NOVO UTILIZADOR

De notar que os campos que são opcionais devem ser preenchidos para uma melhor gestão dos utilizadores, ou seja por exemplo o campo Grupo, indica o grupo a qual pertence o utilizador.

3.2 Apagar Utilizadores

O sistema permite apagar utilizadores criados, os utilizadores do tipo **administrador** podem apagar qualquer tipo de utilizador, enquanto os restantes tipos de utilizadores apenas podem apagar a sua conta, para tal basta pressionar o botão **APAGAR UTILIZADOR**, como podemos ver na figura 5.

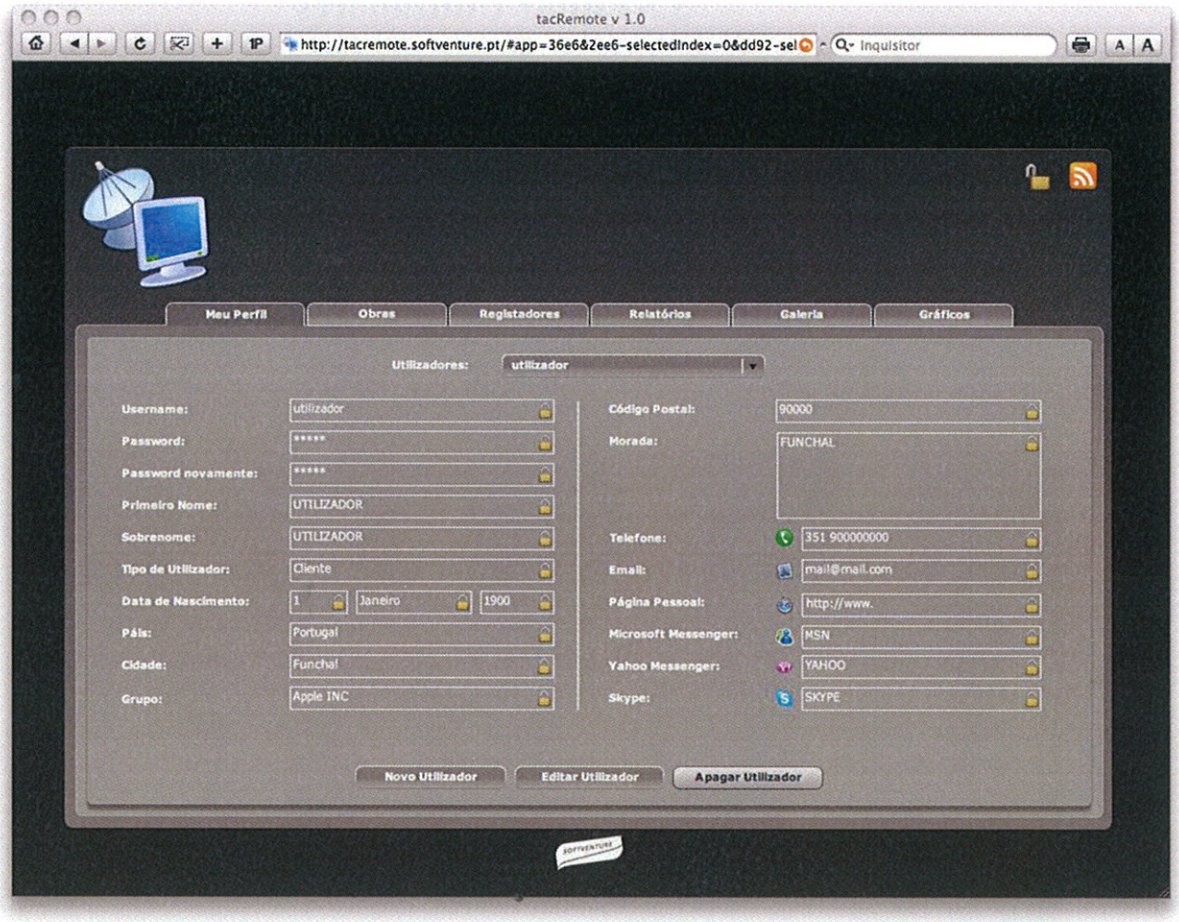


FIGURA 5 – APAGAR UTILIZADOR

Após pressionar o botão, vai aparecer uma janela de confirmação para apagar ou não o utilizador, como podemos ver na figura 6.

FIGURA 6 – CONFIRMAÇÃO PARA APAGAR O UTILIZADOR

3.3 Editar Utilizadores

O sistema permite a alteração dos dados do utilizador, mais uma vez o utilizador do tipo **administrador** pode ver todos os utilizadores como podemos ver na figura 7.

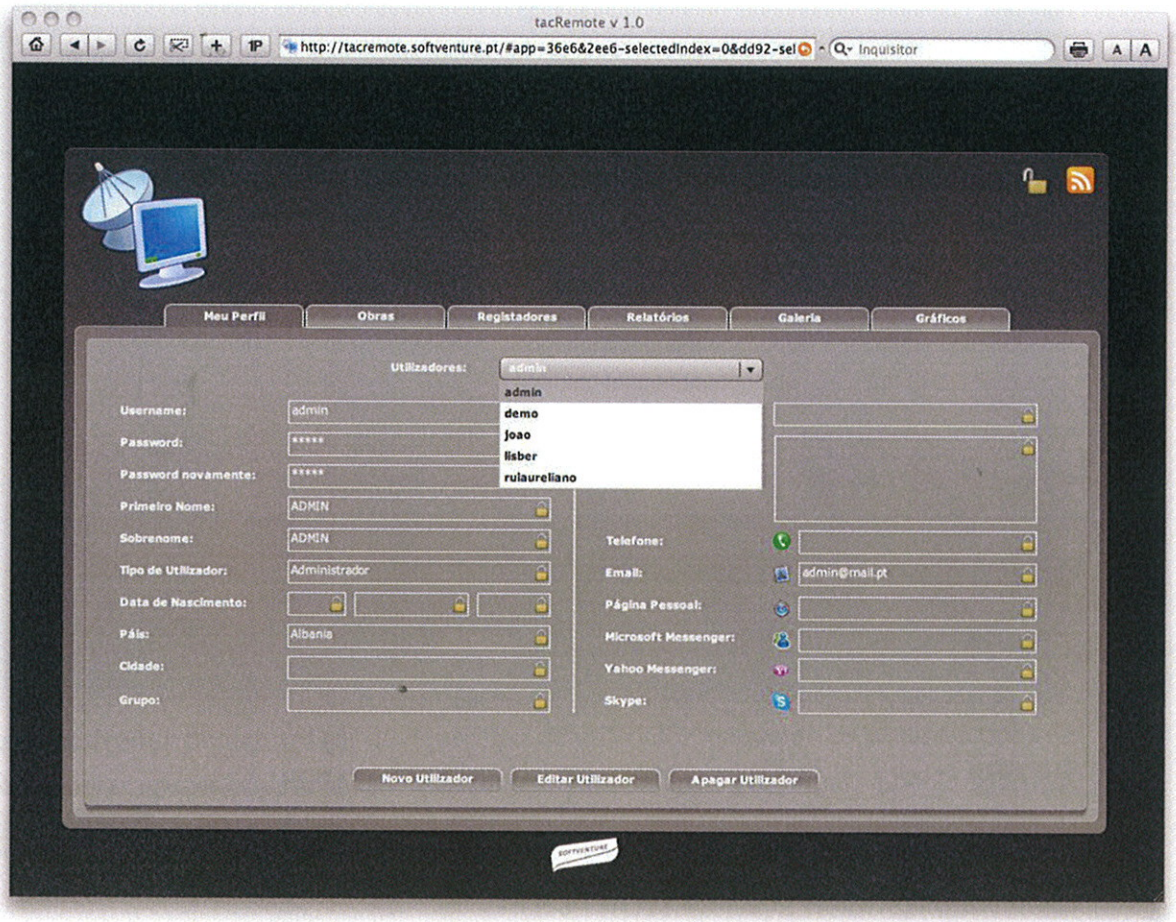


FIGURA 7 – LISTA DE UTILIZADORES

Após seleccionar o utilizador e pressionar o botão **APAGAR UTILIZADOR**, o utilizador será direccionado para uma nova janela como podemos ver na figura 8.

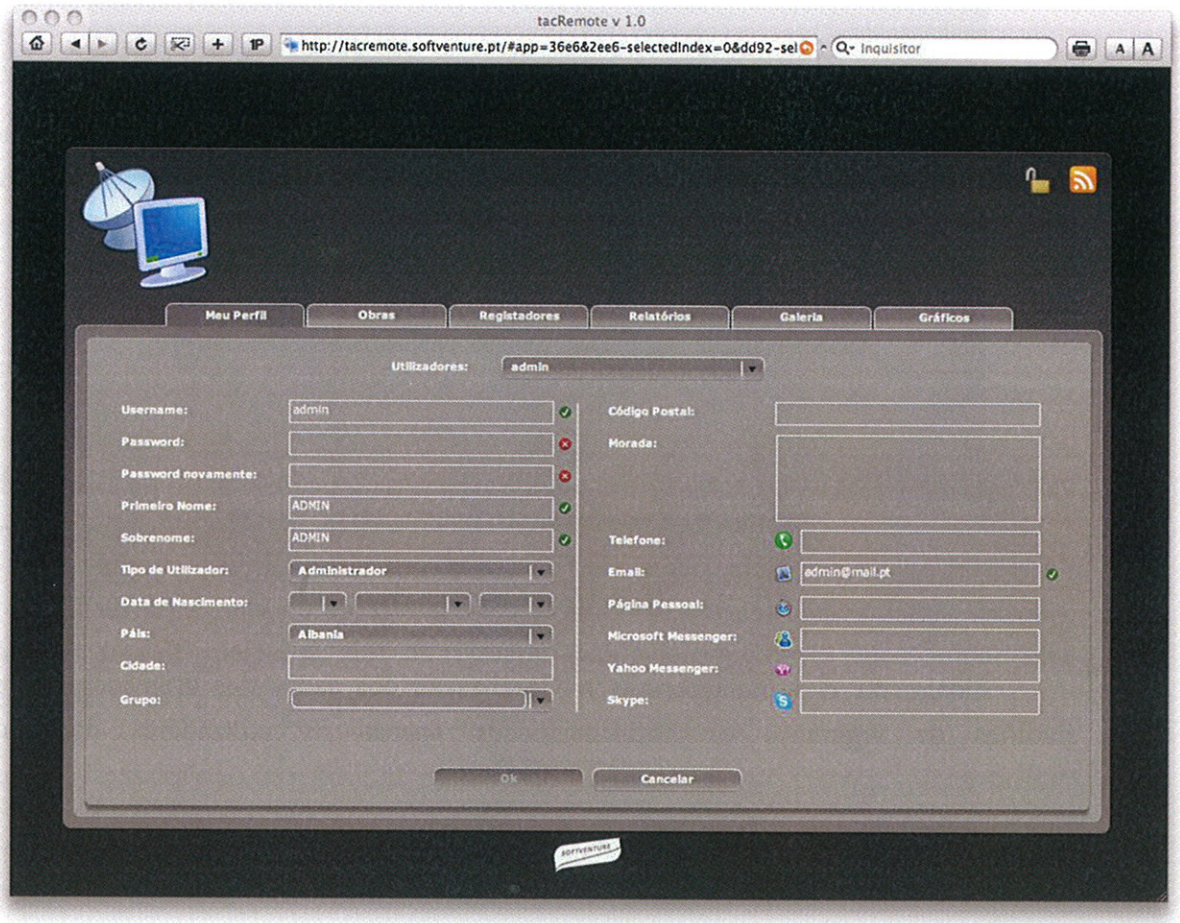


FIGURA 8 – EDITAR CAMPOS DO UTILIZADOR

Note-se que mais uma vez para poder confirmar a edição dos dados, todos os campos obrigatórios tem de ser validos ou seja todos os símbolos deverão passar ao símbolo , e quando os campos estiverem todos válidos, pressionamos o botão **OK**, e será apresentado um *popup* de confirmação para o utilizador decidir se quer ou não editar os dados como podemos ver na figura 9.

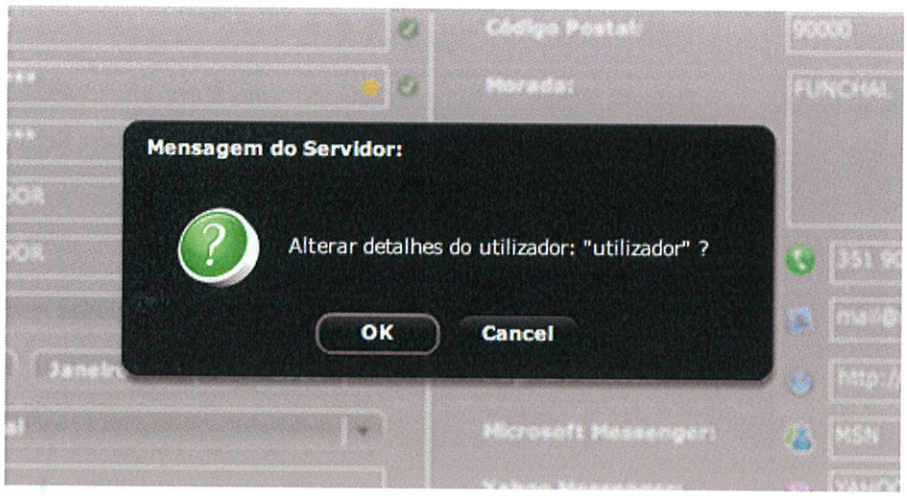


FIGURA 9 – CONFIRMAÇÃO DE EDIÇÃO DO UTILIZADOR

3.4 Ver Utilizadores

Finalmente o sistema permite a visualização dos dados dos utilizadores, por questões de segurança e confidencialidade apenas os utilizadores do tipo **administrador** podem ver os detalhes dos outros utilizadores os restantes apenas podem ver os seus detalhes.

4. Gerir Obras

O sistema tacRemote permite fazer toda a gestão de obras. Basicamente o sistema permite 4 grandes operações (adicionar, apagar, editar e ver obras), que são descritas nos pontos seguintes, que podem ser efectuadas pelos utilizadores, se tiverem permissões para o fazer.

4.1 Adicionar Obras

O sistema *tacRemote* permite adicionar obras, mais uma vez por questões de segurança, apenas os utilizadores do tipo **administrador** e **operador** podem adicionar obras, para tal basta escolher o separador obras e pressionar o botão **NOVA OBRA**, como podemos ver na figura 10.

Para criar uma obra existem campos que são obrigatórios, o nome da obra, morada, data de inicio e de fim, os restantes são opcionais embora sejam importantes para caracterizar a obra.

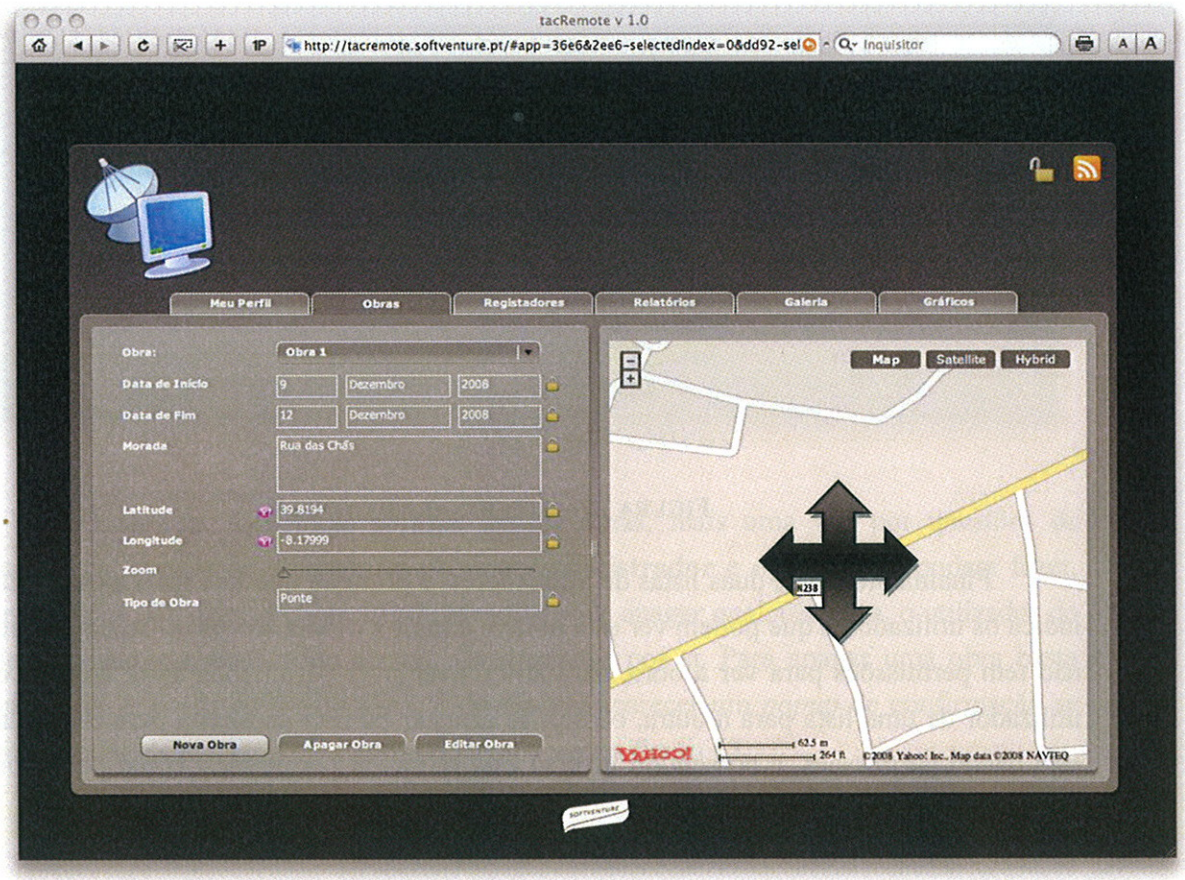


FIGURA 10 – CRIAR NOVA OBRA

Após escolhermos esta opção o utilizador será direccionado para uma janela onde deverá preencher os detalhes da obra, mais uma vez note-se que existem campos obrigatórios e opcionais, mas por uma questão de coerência e objectividade, deve-se preencher todos os campos. Existe também um mapa que permite seleccionar a localização geográfica da obra, podendo aumentar e diminuir o zoom como podemos ver na figura 11. A latitude e longitude é conseguida arrastando o mapa.

FIGURA 11 – ADICIONAR NOVA OBRA

Finalmente existe duas listas de utilizadores, a primeira que fica mais a esquerda indica os utilizadores que podem ver esta obra, e a mais a direita são os utilizadores que não tem permissões para ver a obra, para atribuir/remover permissões basta arrastar o utilizador de uma lista para a outra, ou seja se arrastar da lista da direita para esquerda está dando permissões, e no sentido inverso esta retirando permissões, como podemos ver na figura 12.

FIGURA 12 – ATRIBUIR/RETIRAR PERMISSÕES A UM UTILIZADOR

Após os campos estarem todos válidos, ou seja os símbolos deverão passar a símbolos , pode criar uma nova obra pressionando o botão OK.

4.2 Apagar Obras

O sistema *tacRemote* permite apagar obras, mais uma vez por questões de segurança, apenas os utilizadores do tipo **administrador** e **operador** o podem fazer, enquanto o utilizador do tipo **administrador** pode apagar qualquer obra, o utilizador do tipo **operador** apenas pode apagar obras criadas por si. Para apagar uma obra basta pressionar o botão **APAGAROBRA**, e depois vai aparecer um *popup* de confirmação se quer realmente apagar a obra como podemos ver na figura 13.

FIGURA 13 – CONFIRMAÇÃO SE DESEJA APAGAR A OBRA

4.3 Editar Obras

O sistema *tacRemote* permite editar obras, mais uma vez por questões de segurança, apenas os utilizadores do tipo **administrador** e **operador** podem editar obras, no entanto enquanto o utilizador do tipo **administrador** pode editar todas as obras, o tipo **operador** apenas pode editar as obras criadas por si.

Pode alterar os campos com os detalhes de cada obra, bem com a localização geográfica e dar/retirar permissões aos utilizadores.

Após os campos estarem válidos o utilizador pode confirmar a alteração dos dados pressionando o botão **OK**, como podemos ver na figura 14.

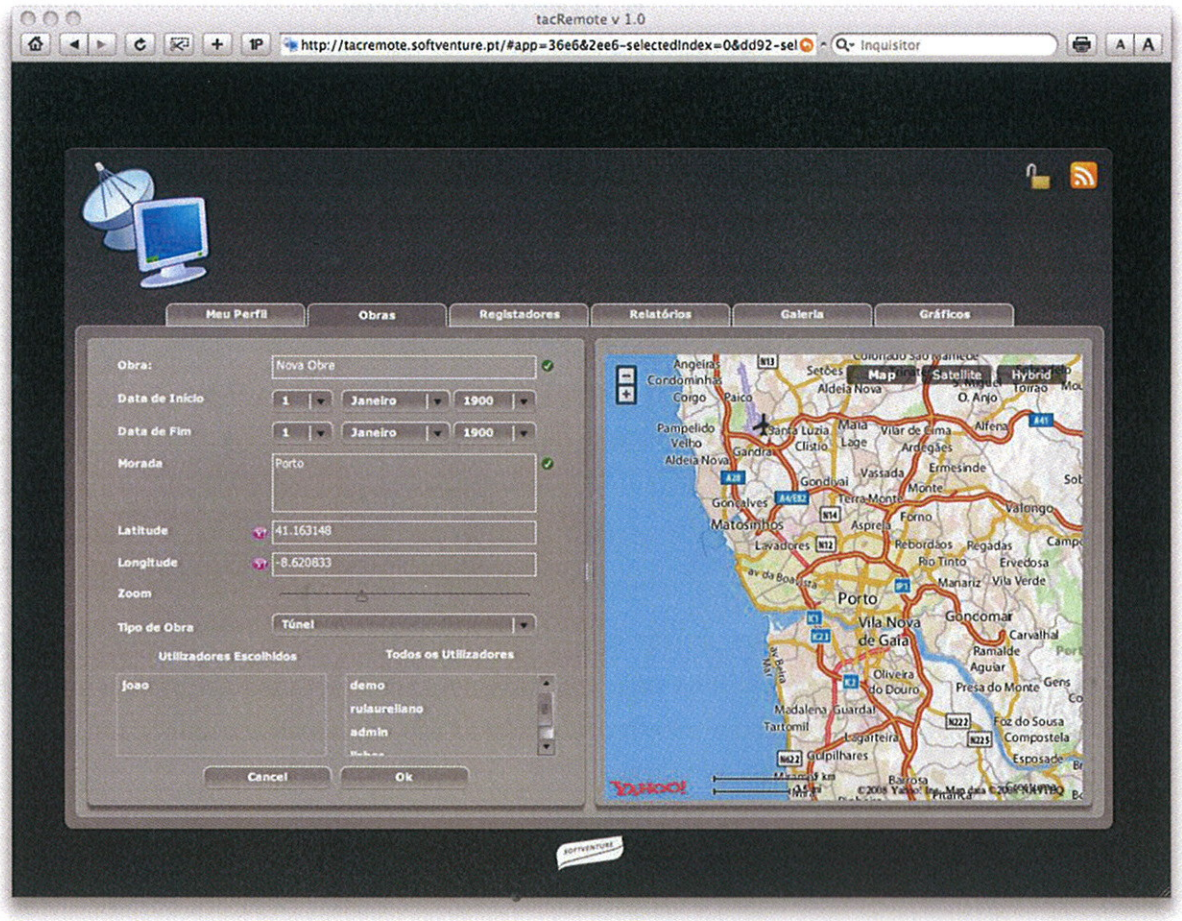


FIGURA 14 - ADITAR OBRA

4.4 Ver Obras

Finalmente o sistema permite a visualização dos dados das obras, por questões de segurança e confidencialidade apenas os utilizadores do tipo **administrador** podem ver todas as obras, os utilizadores do tipo **operador** podem ver as obras criadas por si e os utilizadores do tipo **cliente** podem ver as obras cujo um administrador ou operador tenha dado permissão.

5. Gerir Sensores

O sistema tacRemote permite fazer toda a gestão dos sensores. Basicamente o sistema permite 3 grandes operações (adicionar, remover e apagar), que são descritas nos pontos seguintes, que podem ser efectuadas pelos utilizadores, se tiverem permissões para o fazer.

5.1 Adicionar Sensores

O sistema tacRemote permite aos utilizadores adicionar sensores a cada um dos registadores que tão associados as obras, desde que tenham permissão para faze-lo. Os utilizadores do tipo **administrador** podem adicionar sensores a todas as obras, enquanto os utilizadores do tipo **operador** só podem adicionar sensores as obras criadas por si. Os utilizadores do tipo **cliente** não têm permissões para adicionar registadores. O processo para adicionar um sensor é simples, basta para tal arrastar o sensor da palete de sensores mais a direita para a obra seleccionada com podemos ver na figura 15. Note-se que se o utilizador não tiver privilégios não é possível arrastar os sensores.

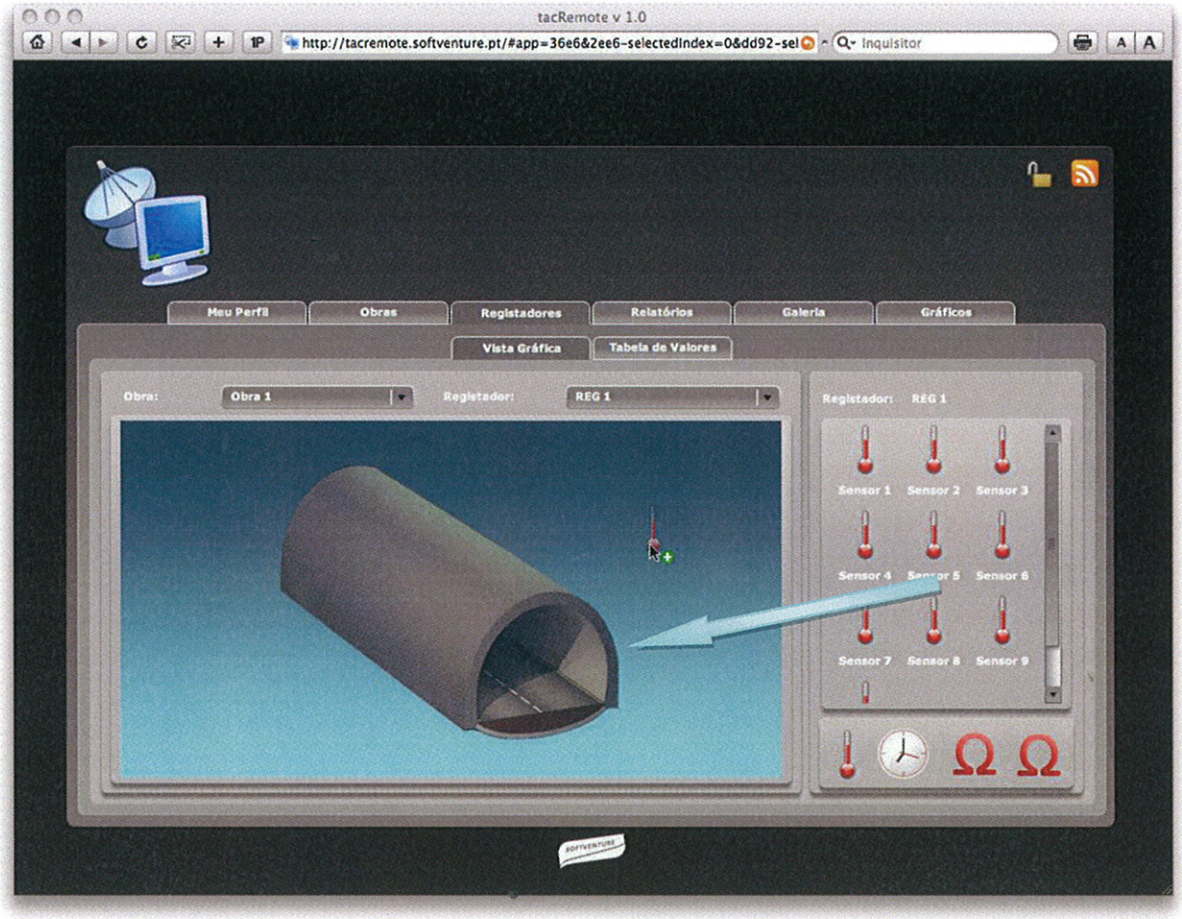


FIGURA 15 – ADICIONAR UM SENSOR A UMA OBRA

5.2 Apagar Sensores

O sistema tacRemote permite aos utilizadores remover sensores de cada um dos registadores que tão associados as obras, desde que tenham permissão para faze-lo. Os utilizadores do tipo **administrador** podem remover sensores de todas as obras, enquanto os utilizadores do tipo **operador** só podem remover sensores das obras criadas por si, os utilizadores do tipo **cliente** não tem permissões para remover sensores. O processo para remover um sensor é simples, basta para tal pressionar o símbolo , e o respectivo sensor será apagado, com podemos ver na figura 16. Note-se que se o utilizador não tiver privilégios não é possível arrastar os sensores.

FIGURA 16 – SENSORES ATRIBUÍDOS A UM REGISTADOR

5.3 Editar Sensores

O sistema *tacRemote* permite aos utilizadores editar a posição dos sensores de cada um dos registadores, desde que tenham permissão para faze-lo. Os utilizadores do tipo **administrador** podem mover os sensores de todas as obras, enquanto os utilizadores do tipo **operador** só podem mover os sensores das obras criadas por si, os utilizadores do tipo **cliente** não tem permissões para mover sensores. Para mover um sensor basta arrastar o sensor para a localização desejada como podemos ver na figura 17.

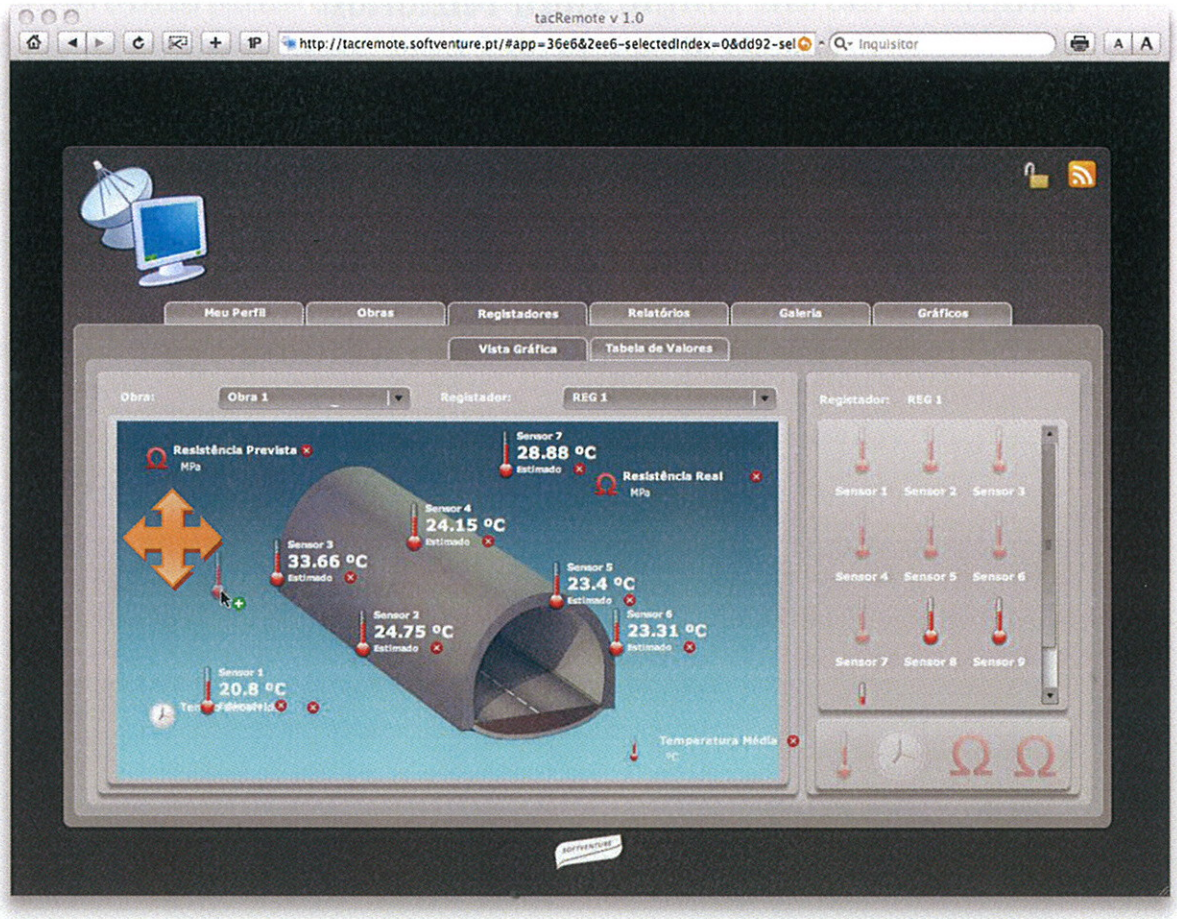


FIGURA 17 – MOVER UM SENSOR

5.4 Ver Sensores

Finalmente o sistema permite a visualização dos sensores de cada uma das obras, por questões de segurança e confidencialidade apenas os utilizadores do tipo **administrador** podem ver todos os sensores, os utilizadores do tipo **operador** podem ver os sensores associados as obras criadas por si e os utilizadores do tipo **cliente** podem ver os sensores cujo um administrador ou operador tenha dado permissão.

6. Gerir relatórios

Os relatórios do tacRemote são gerados duma forma automática, o separador **Relatórios** permite fazer a gestão desses relatórios, os pontos seguintes mostram o que se pode fazer com esses relatórios.

6.1 Fazer download do relatório

O sistema permite que todos os utilizadores registados no sistema possam aceder aos relatórios, note-se que estes relatórios estão associados as obras, assim um utilizador só pode ver esses relatórios se tiver permissões para ver essa obra. Caso tenha permissões o utilizador poderá ver a lista dos relatórios e se pressionar o botão , dará inicio ao processo de download do relatório, como podemos ver na figura 18.

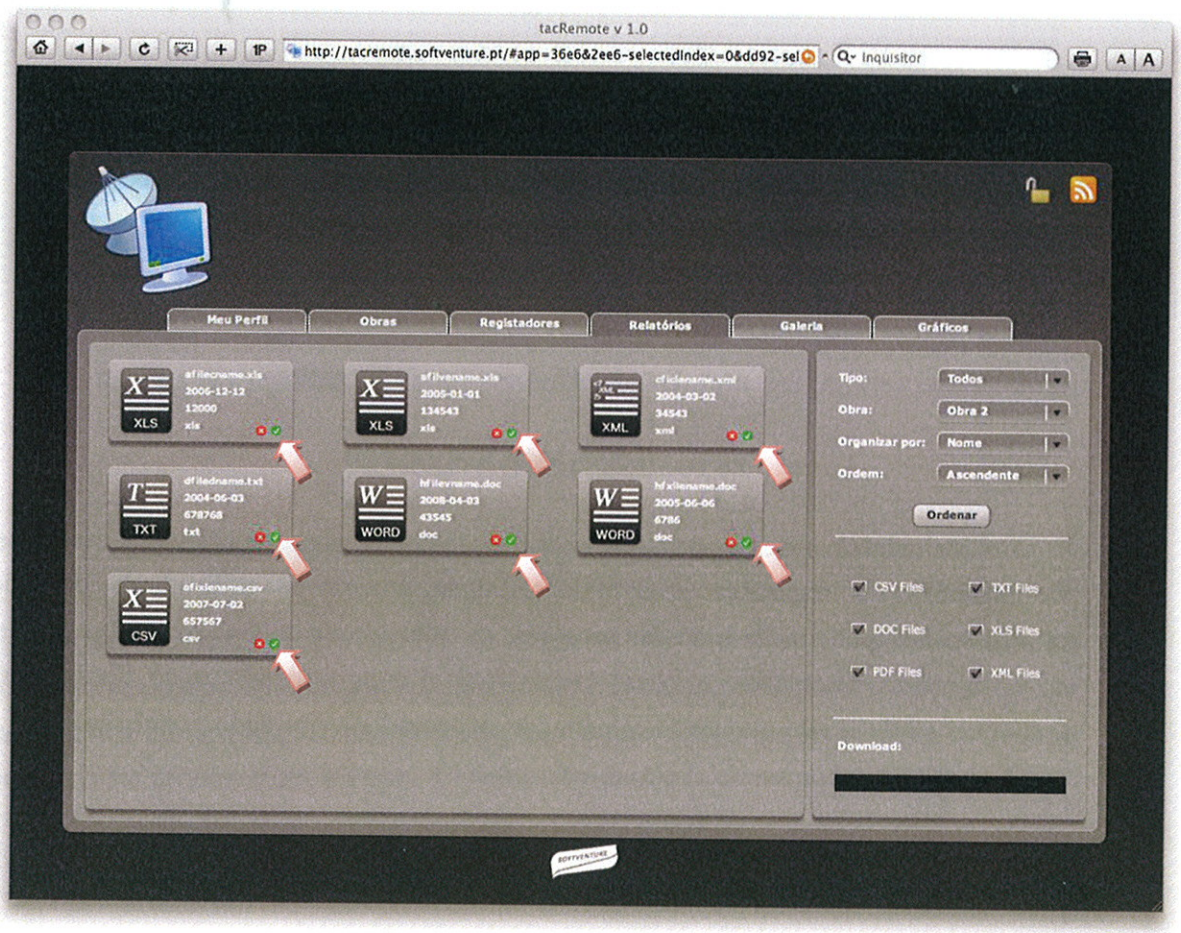


FIGURA 18 – LISTA DE RELATÓRIOS ASSOCIADOS A CADA OBRA

6.2 Apagar relatórios

O sistema permite que os relatórios sejam apagados, neste caso os utilizadores do tipo **Administrador** podem apagar todos os relatórios enquanto os utilizadores do tipo **Operador** apenas podem apagar relatórios associados as obras criadas por sim, os utilizadores do tipo **Cliente** não podem apagar relatórios. O processo para apagar relatórios é simples, basta pressionar o botão e depois confirmar que quer apagar o relatório, como podemos ver na figura 19.

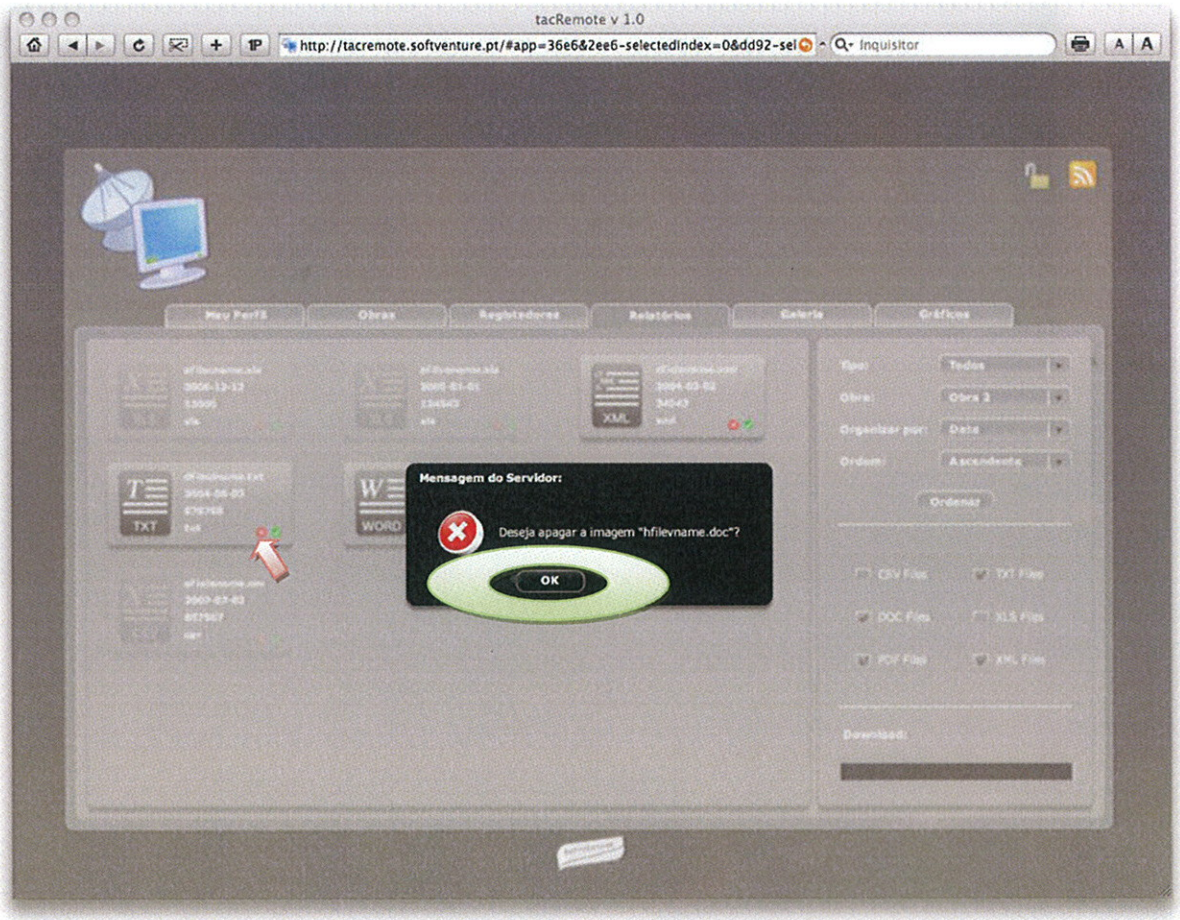


FIGURA 19 – APAGAR UM RELATÓRIO DO SISTEMA

6.3 Ordenar relatórios

O sistema permite a ordenação dos relatórios, esta funcionalidade é de extrema utilidade quando existem muitos relatórios pois permite que possamos ordenar os relatórios por nome, tamanho, tipo e data, para além disso permite que o processo de ordenação seja por ordem crescente ou decrescente. Este processo está explicado na figura 20.

FIGURA 20 – PROCESSO DE ORDENAÇÃO DOS RELATÓRIOS NO SISTEMA

7. Gerir Galeria

O sistema *tacRemote* permite aos utilizadores introduzirem imagens na galeria relativas a cada obra, ou seja sempre que seja criada uma obra, e automaticamente é criada uma galeria para cada obra.

7.1 Ver foto

O sistema *tacRemote* permite a visualização das fotos associadas a cada obra. Como já foi referido, por questões de segurança e confidencialidade apenas os utilizadores do tipo **administrador** podem ver toda as imagens, os utilizadores do tipo **operador** podem ver as imagens associados as obras criadas por si e os utilizadores do tipo **cliente** podem ver as imagens cujo um administrador ou operador tenha dado permissão. O processo é descrito na figura 21.

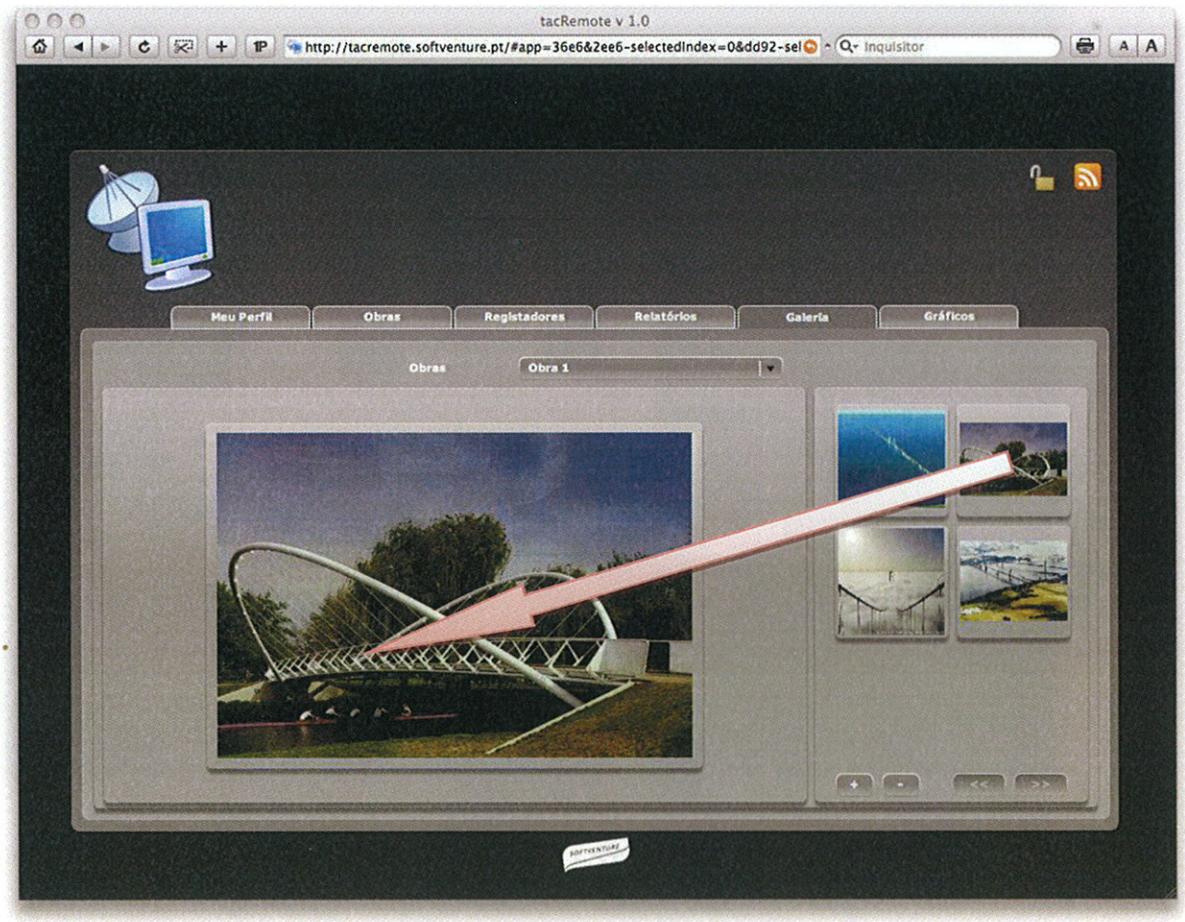


FIGURA 21 – VER IMAGENS NA GALERIA

7.2 Adicionar Foto

O sistema *tacRemote* permite que os utilizadores adicionem imagens a galeria. Mais uma vez por questões de segurança e confidencialidade apenas os utilizadores do tipo administrador podem adicionar imagens, os utilizadores do tipo operador podem adicionar imagens a galeria associadas as obras criadas por si e os utilizadores do tipo cliente não podem adicionar imagens. O processo de inserção de uma imagem é simples, basta pressionar o botão +, depois vai aparecer uma janela para adicionar uma imagem, escolhemos a imagem que queremos introduzir no sistema depois de escolher essa imagem fazemos OK e a imagem é introduzida. Este processo pode ser visto na figura 22.

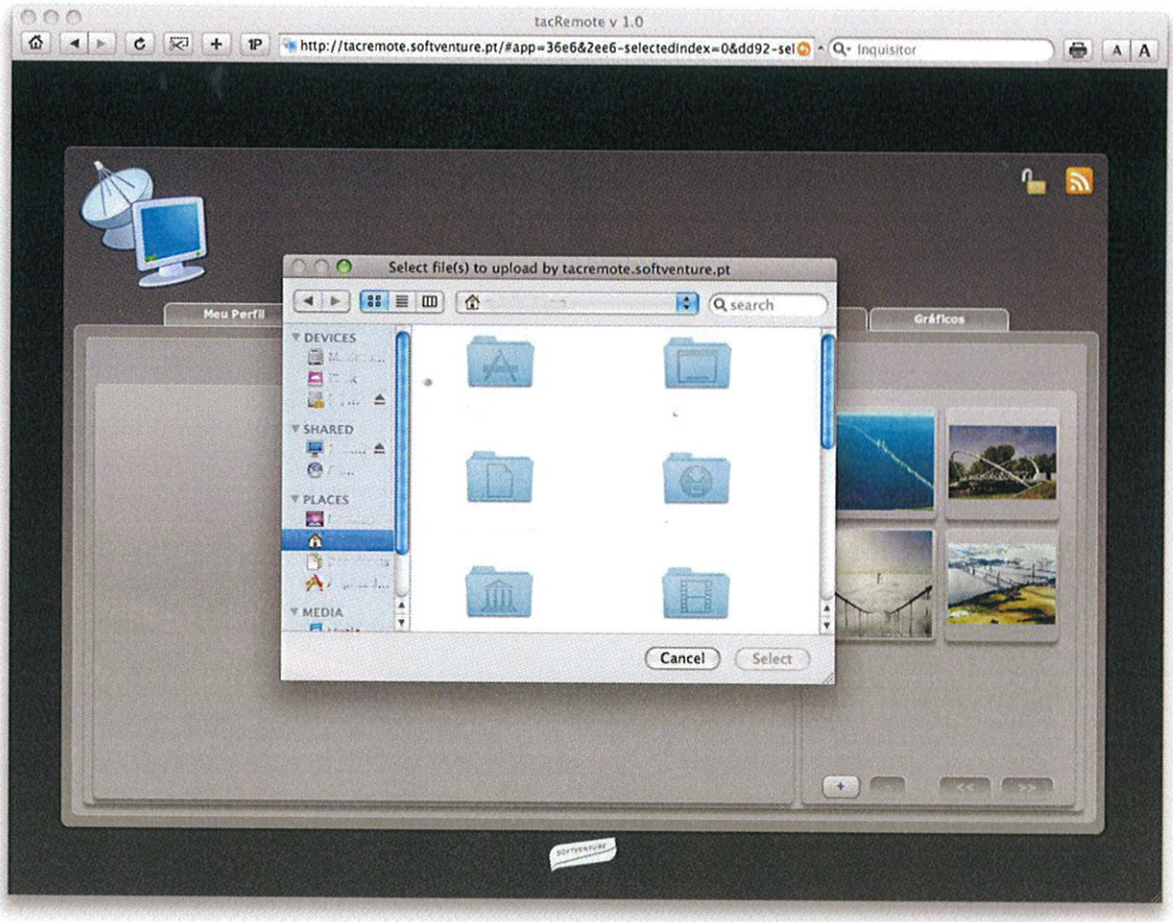


FIGURA 22 – ADICIONAR UMA IMAGEM

7.3 Apagar Foto

O sistema *tacRemote* permite que os utilizadores adicionem imagens da galeria. Por questões de segurança e confidencialidade apenas os utilizadores do tipo **administrador** podem apagar imagens, os utilizadores do tipo **operador** podem apagar imagens da galeria associadas as obras criadas por si e os utilizadores do tipo **cliente** não podem apagar imagens. O processo para remover uma imagem é simples, basta pressionar o botão -, depois vai aparecer uma janela para confirmar se quer mesmo remover a imagem, como podemos ver na figura 23.

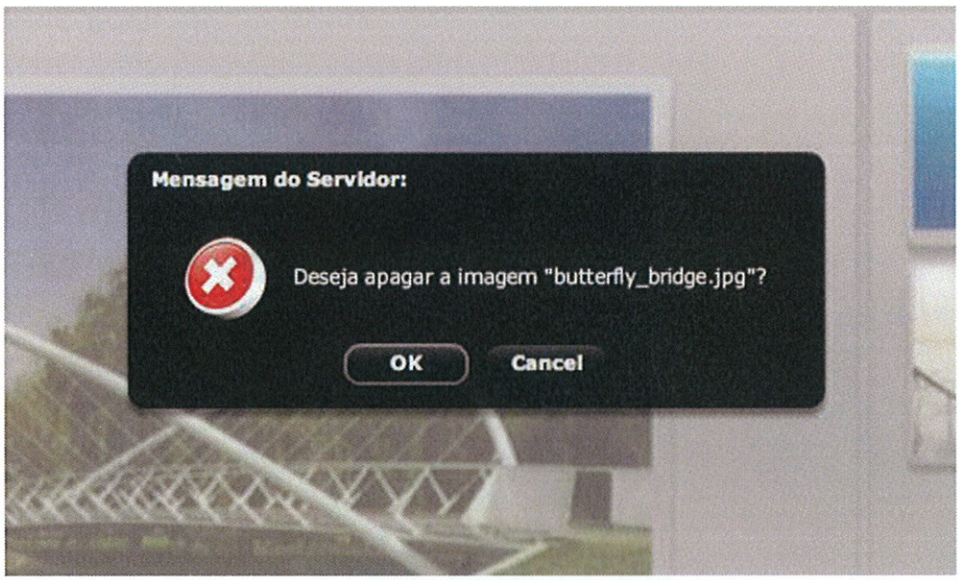


FIGURA 23 – CONFIRMAÇÃO PARA REMOVER A IMAGEM

8. Gerir Gráficos

O sistema *tacRemote* permite a análise estatística dos dados através de vários gráficos disponibilizados, com vários tipos de valores, como podemos ver nos pontos seguintes.

8.1 Ver Gráfico das Resistências

O sistema permite analisar o cálculo das resistências de cada sensor, através de um gráfico de linhas, podemos ver todos os sensores ou escolher quantos queremos ver seleccionando a *checkbox* relativa a cada sensor, como podemos ver na figura 24.

FIGURA 24 – GRÁFICO DAS RESISTÊNCIAS CALCULADAS

8.2 Ver Gráfico da Media das Resistências

O sistema permite também analisar o cálculo das resistências médias de todos os sensores, através de um gráfico de linhas, seleccionando a *checkbox* **Média**, como podemos ver na figura 25.

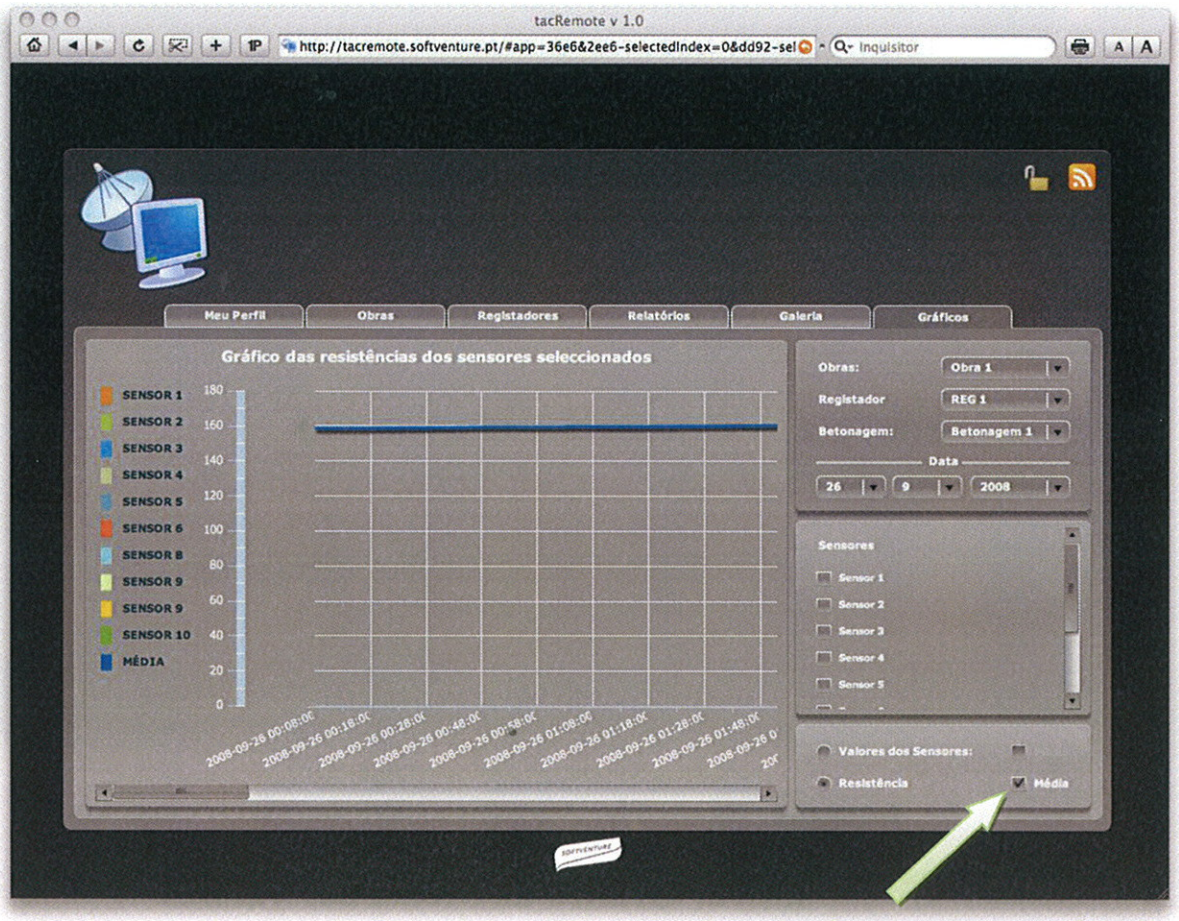


FIGURA 25 – MÉDIA DAS RESISTÊNCIAS CALCULADAS

8.3 Ver Gráfico dos Valores

O sistema permite analisar os valores lidos por cada sensor, através de um gráfico de linhas, podemos ver todos os sensores ou escolher quantos queremos ver seleccionando a *checkbox* relativa a cada sensor, como podemos ver na figura 26.

FIGURA 26 – GRÁFICO DOS VALORES DOS SENSORES SELECCIONADOS

8.4 Ver Gráfico da Media dos Valores

O sistema permite também analisar a média dos valores lidos por todos os sensores, através de um gráfico de linhas, seleccionando a *checkbox* **Média**, como podemos ver na figura 27.

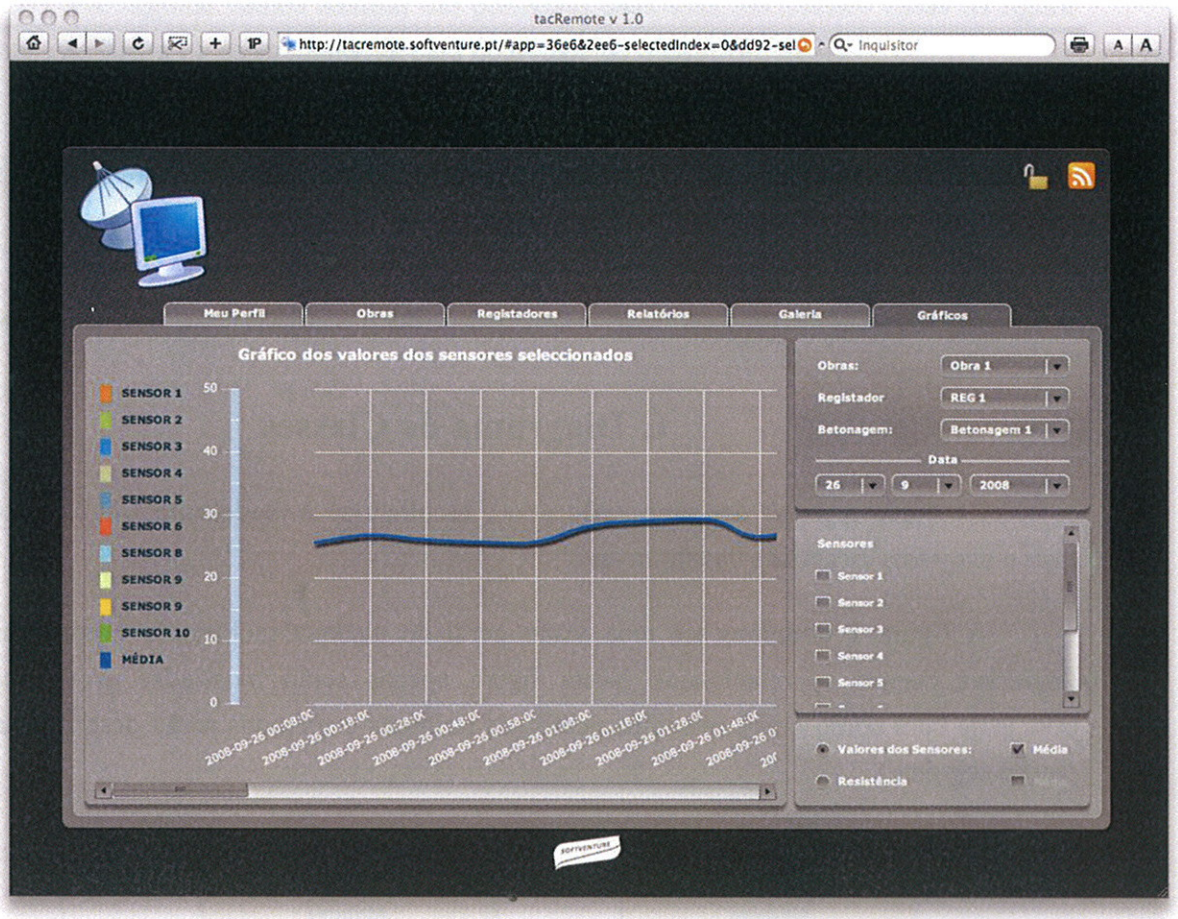


FIGURA 27 – GRÁFICO DA MÉDIA DOS VALORES LIDOS

ANEXO C – DIAGRAMA DE CLASSES

1. Diagrama de Classes

De modo a poder dar uma visão geral, escolheu-se a modelação UML, neste caso o diagrama de classes de alto nível.

A figura 1 podemos ter uma visão geral do sistema tacRemote e como as diferentes classes se relacionam. Nesta figura apenas serão mostradas as classes principais bem como o seu relacionamento (os atributos e métodos serão descritos na secção seguinte)

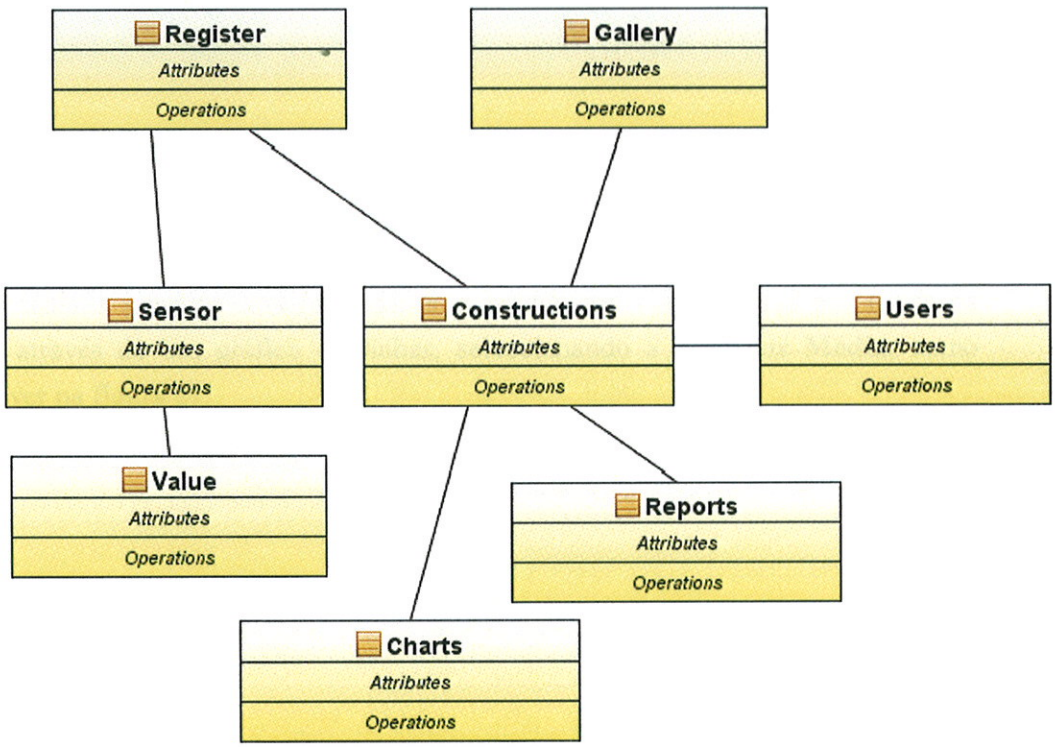


FIGURA 1 – DIAGRAMA DE CLASSES DE ALTO NÍVEL DO SISTEMA TACREMOTE

De seguida vamos analisar em pormenor cada uma das classes vistas na figura anterior (figura 1), separadamente para podermos ter uma ideia melhor.

A figura 2 mostra a classe Utilizadores (Users), onde podemos ver todos os atributos e métodos desta classe.

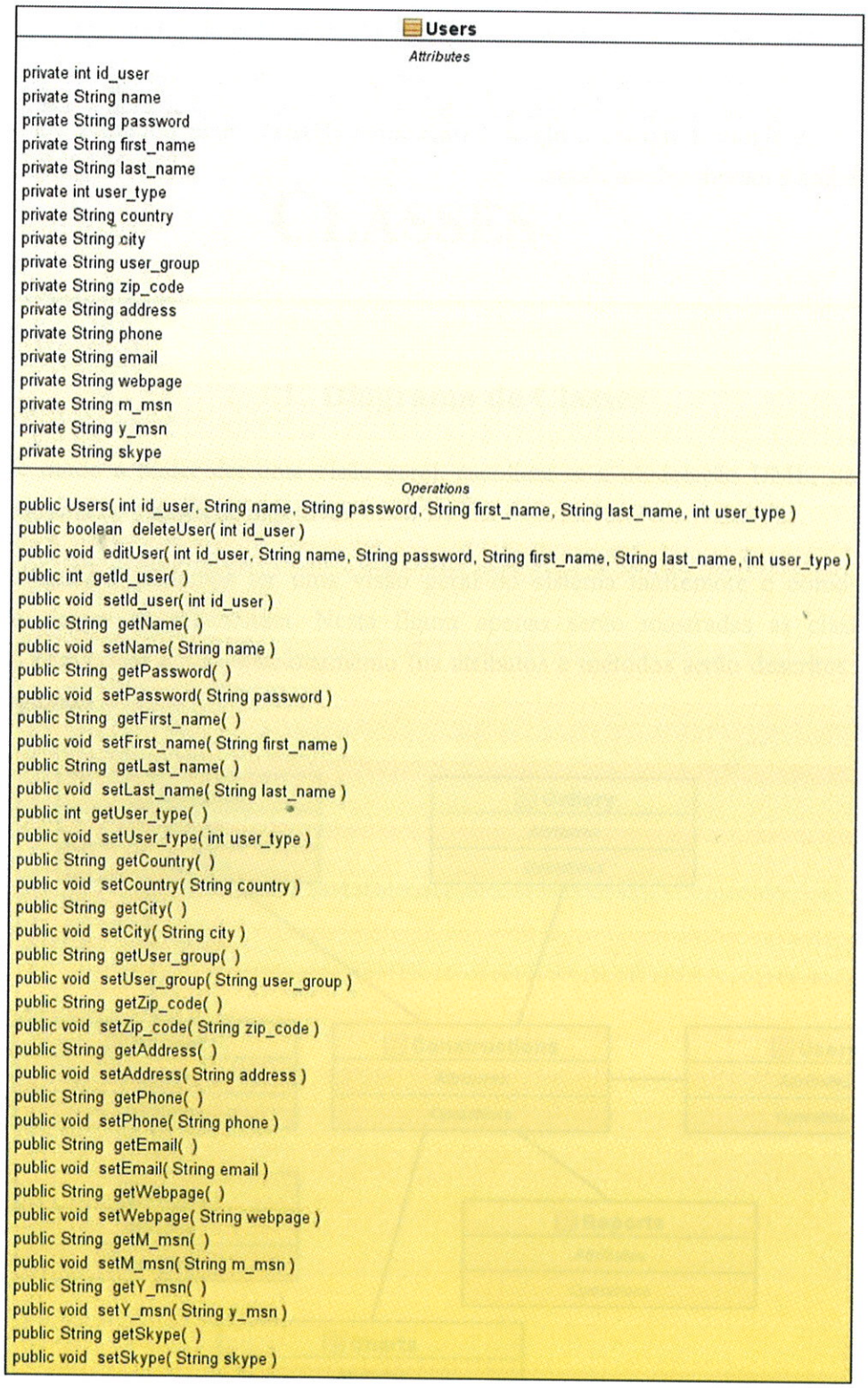


FIGURA 2 - CLASSE UTILIZADOR (USER)

A figura 3 mostra a classe Obras (Constructions), onde podemos ver todos os atributos e métodos desta classe.

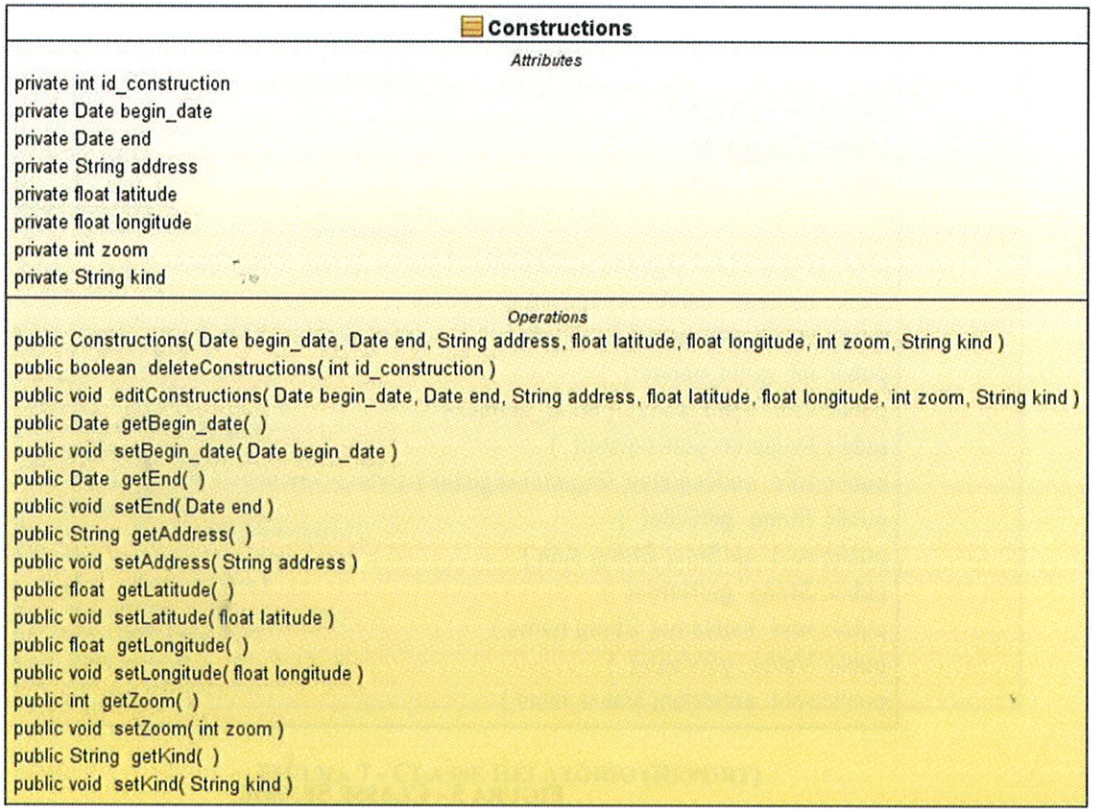


FIGURA 3 - CLASSE OBRAS (CONSTRUCTIONS)

A figura 4 mostra a classe Registador (Register), onde podemos ver todos os atributos e métodos desta classe.

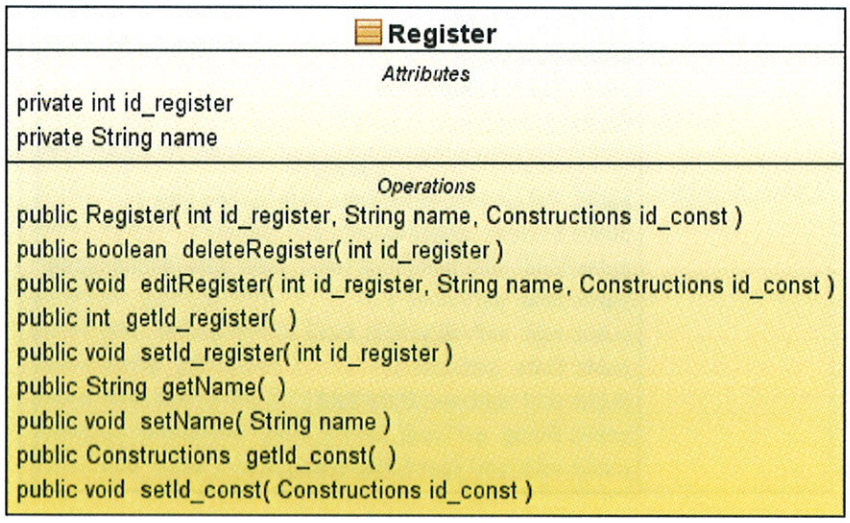


FIGURA 4 - CLASSE REGISTADOR (REGISTER)

A figura 5 mostra a classe Sensor, onde podemos ver todos os atributos e métodos desta classe.

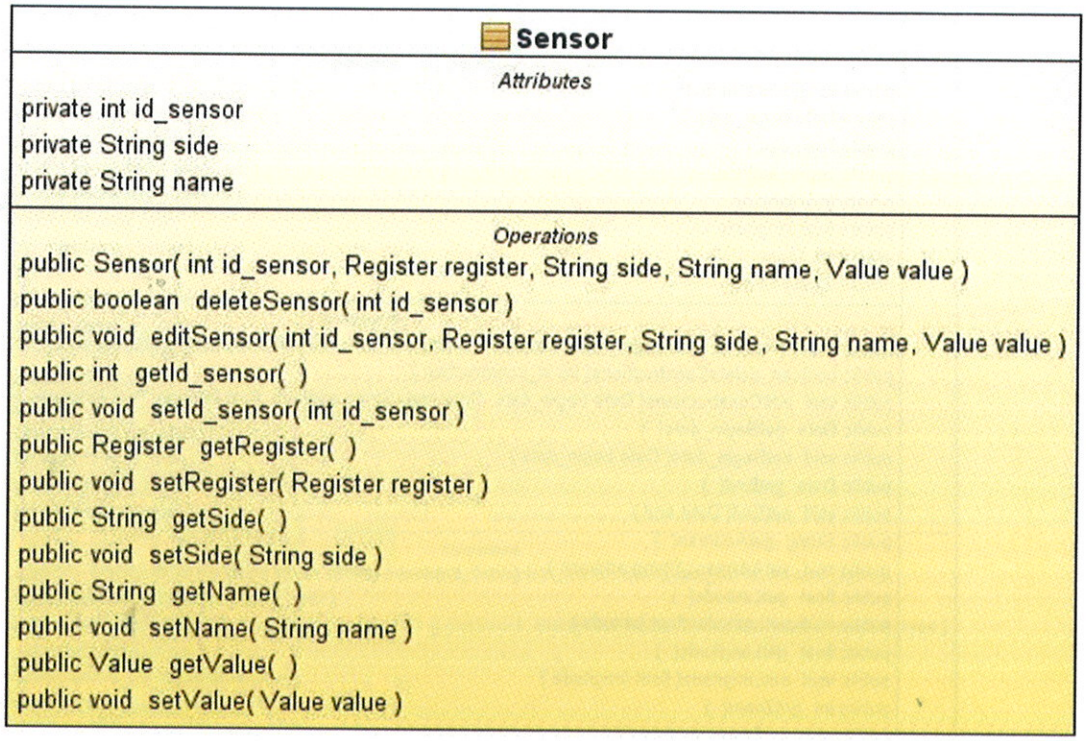


FIGURA 5 - CLASSE SENSOR

A figura 6 mostra a classe Valor (Value), onde podemos ver todos os atributos e métodos desta classe.

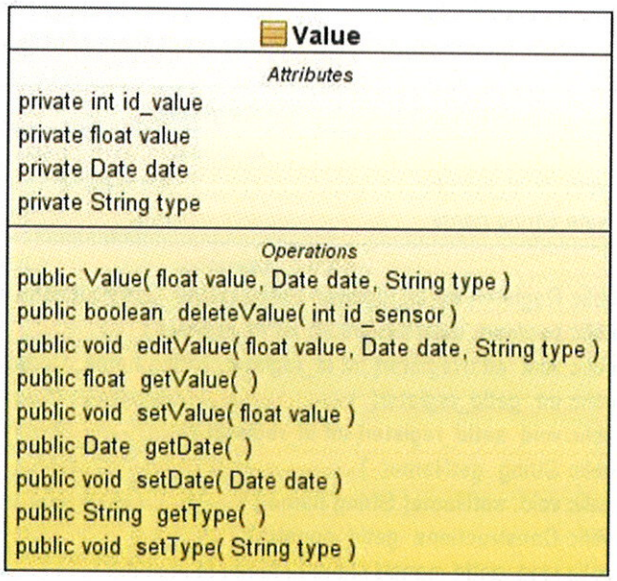


FIGURA 6 - CLASSE VALOR (VALUE)

A figura 7 mostra a classe Relatórios (Reports), onde podemos ver todos os atributos e métodos desta classe.

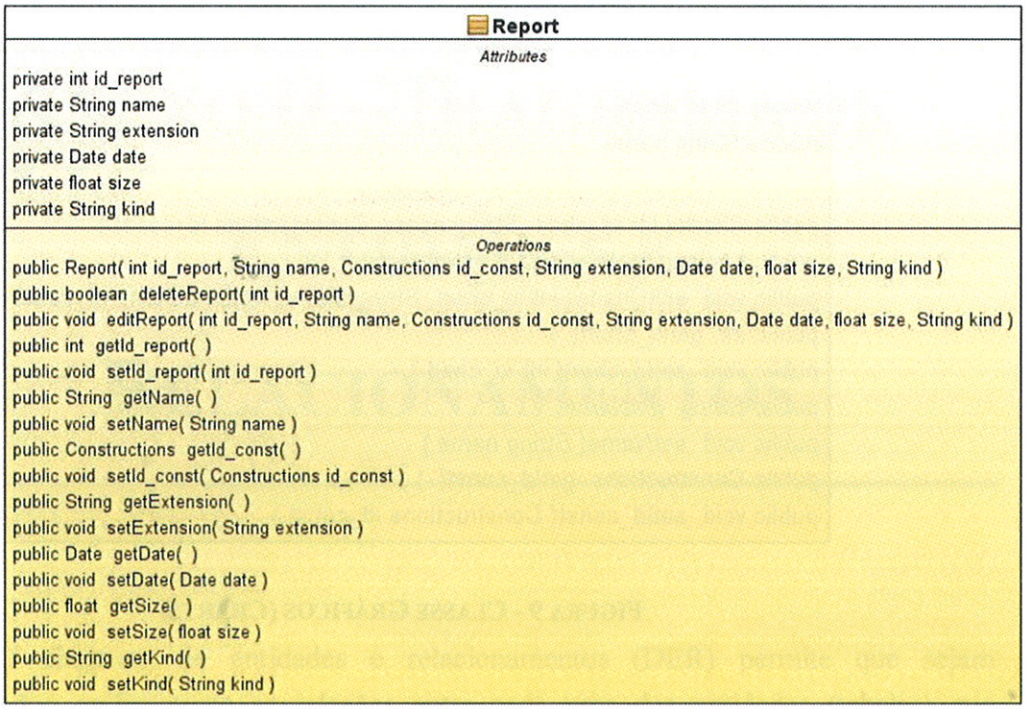


FIGURA 7 - CLASSE RELATÓRIO (REPORT)

A figura 8 mostra a classe Galeria (Gallery), onde podemos ver todos os atributos e métodos desta classe.

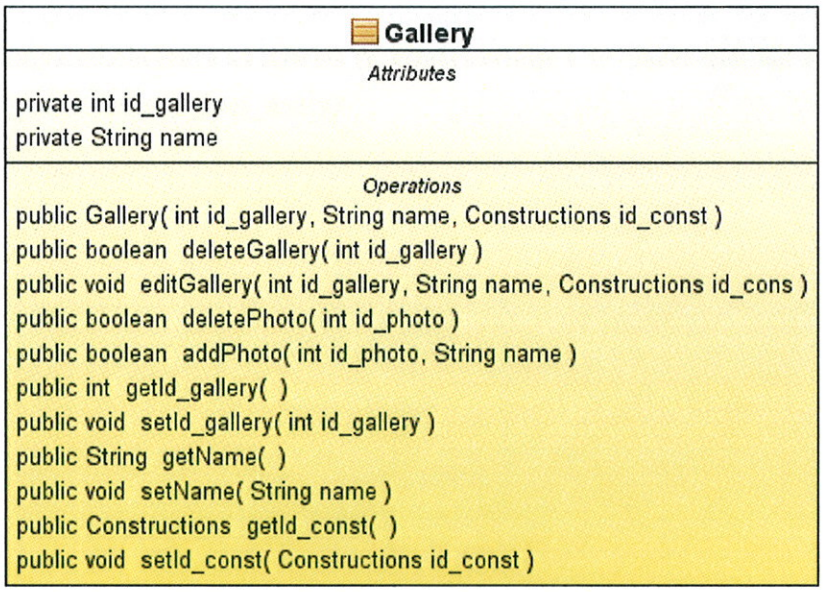


FIGURA 8 - CLASSE GALERIA (GALLERY)

A figura 9 mostra a classe Gráficos (Charts), onde podemos ver todos os atributos e métodos desta classe.

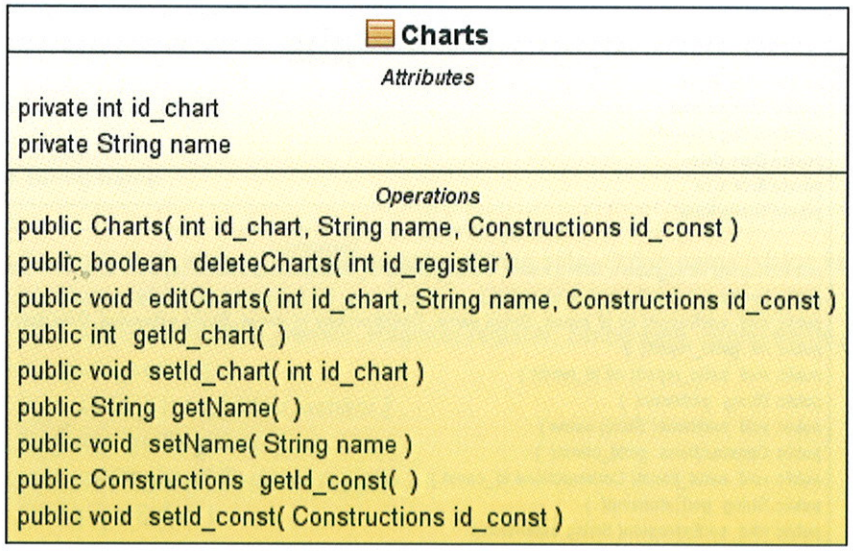


FIGURA 9 - CLASSE GRÁFICOS (CHARTS)

ANEXO D – DIAGRAMA DE ENTIDADES E RELACIONAMENTOS

O diagrama de entidades e relacionamentos (DER) permite que sejam visualizados graficamente as relações entre cada uma das entidades (tabelas) que compõem o modelo de dados do sistema tacRemote.

O modelo é composto por 10 entidades (tabelas), que estão todas relacionadas entre si, como podemos ver na figura 1. Existe uma entidade principal que é a entidade Obras (**tr_contructions**) que se liga a todas as outras excepto a entidade utilizadores que como é necessita de uma ligação de cardinalidade *n* para *m* então foi necessário introduzir uma nova tabela entra as tabelas **tr_contructions e tr_users** que faz a ligação entre ambas (**tr_contructions_has_users**)

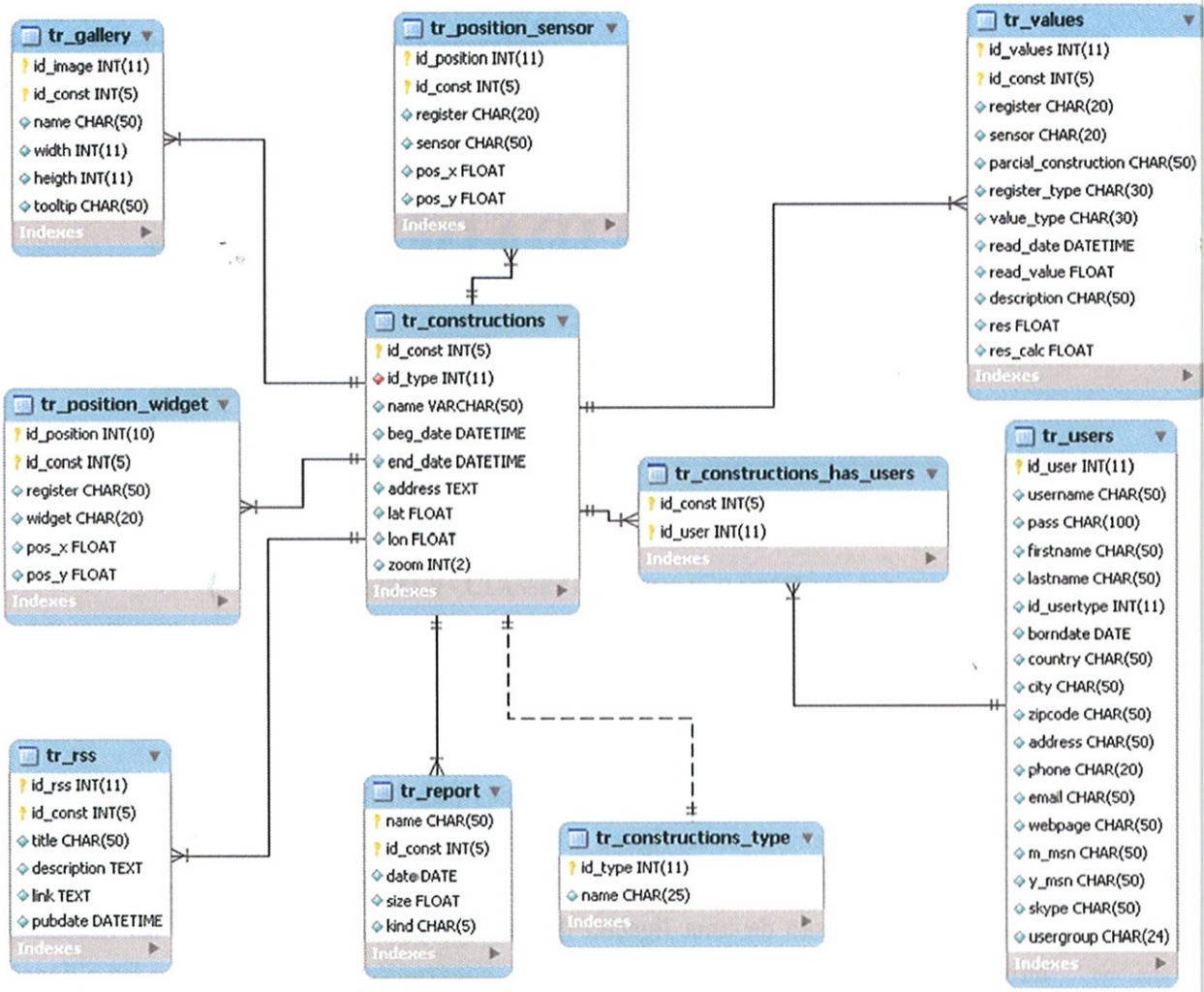


FIGURA 1 – DIAGRAMA DE ENTIDADES E RELACIONAMENTOS