FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


# Tecnologias Abertas de Suporte à Experimentação Remota via Web

**João Filipe Dias Gonçalves**

Relatório de Dissertação

Mestrado Integrado em Engenharia Informática e Computação

Orientador: José Manuel Magalhães Cruz, Prof. Doutor

Co-orientador: José Carlos Alves, Prof. Doutor

30 de Março de 2010

# Tecnologias Abertas de Suporte à Experimentação Remota via Web

**João Filipe Dias Gonçalves**

Relatório de Dissertação

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Francisco José de Oliveira Restivo (Professor Doutor)

---

Arguente: Rui Pedro Sanches de Castro Lopes (Professor Doutor)

Vogal: José Manuel Magalhães Cruz (Professor Doutor)

25 de Março de 2010

# Resumo

As aulas laboratoriais no ensino de cursos como Engenharia são extremamente importantes mas estão sujeitas a um certo conjunto de limitações que podem influenciar o seu número e até a sua existência (disponibilidade do laboratório, disponibilidade de docentes, orçamento). Para contornar um pouco este problema, surgiram nos últimos anos, impulsionados pelo desenvolvimento tecnológico em geral e da Informática, em particular, dois novos tipos de laboratórios: laboratórios simulados e laboratórios remotos.

O laboratório remoto actual existente na Faculdade de Engenharia da Universidade do Porto tem graves problemas de compatibilidades de sistemas informáticos, de gestão e de expansão. Esta dissertação surgiu da necessidade de se desenvolver uma plataforma que possa substituir a plataforma actual, baseada em LabVIEW, e permitir a fácil implementação e desenvolvimento de novas experiências, acima de tudo, a baixo custo.

Neste documento foram identificados alguns projectos desenvolvidos a nível mundial que possibilitam a criação de laboratórios remotos; no entanto, a sua maioria ou é baseada em *software* proprietário ou, é baseada em *software open-source*, mas não se encontra disponível para *download* e utilização noutras instituições de ensino. Outros projectos são soluções específicas, desenvolvidas apenas para um determinado tipo de experiências. O único sistema disponível para *download* é o iLab Project mas tem o inconveniente de apenas poder ser instalado em sistemas operativos Microsoft Windows.

A plataforma aqui desenvolvida é baseada em ferramentas *open-source* e, sob este paradigma, disponibilizada à comunidade. Possui uma arquitectura cliente-servidor e foi desenvolvida utilizando PHP para a aplicação alojada no servidor *web*(Apache) e JavaFX para os painéis de controlo das experiências. Ela permite o agendamento de sessões para utilização das experiências, possibilita a criação, edição e remoção de experiências, utilizadores, sessões agendadas, grupos de utilizadores e grupos de experiências. Paralelamente, também foi desenvolvido um conjunto de mostradores semelhantes aos utilizados nos equipamentos de medição e algumas ferramentas de apoio para facilitarem o desenvolvimento de novos painéis de controlo para experiências.

Relativamente à comunicação entre os vários componentes, a comunicação entre o cliente e o servidor é efectuada sobre o protocolo HTTP e entre o servidor e a experiência sobre um protocolo próprio, desenvolvido exclusivamente para esta plataforma, baseado em texto e assente sobre a pilha de protocolos TCP/IP.

Depois de alguns testes efectuados, esta arquitectura é mais lenta do que uma em que o cliente se ligue directamente à experiência, em cerca de 130 ms. Como tal, terão de ser efectuados mais testes para ver se é possível baixar este valor.

O sistema ainda só foi testado de forma preliminar, para se provar o conceito. Funciona bem, mas carece de algum desenvolvimento adicional no sentido de melhorar os

tempos de resposta de comandos individuais de certos tipos de experiências que necessitam de ser observadas e monitorizadas em “tempo real”.

# Abstract

Hands-on laboratories have a very important role in Science or Engineering education, but are subject to a certain set of limitations that may influence their number and even their existence (availability of laboratory, availability of teachers, budget). To work around this problem somewhat, emerged in recent years, driven by technological development in general and IT in particular, two new types of laboratories: simulated laboratories and remote laboratories.

The current existing remote laboratory in FEUP has serious problems of compatibilities, management and expansion. This dissertation appeared from the need to develop a platform that can replace the current platform, based on LabVIEW, and permits an easy implementation and development of new experiences and, above all, low cost.

This document identified a certain number of projects developed worldwide that allow the creation and development of remote laboratories, however, its majority or is based on proprietary software, or is based on open-source software, but is not available for download and use in other educational institutions. Other projects are specific solutions that were developed only for a certain type of experiments. The only system available for download is the iLab Project but has the drawback that can only be installed on Microsoft Windows operating systems.

The platform developed here is based on open-source tools, and under this paradigm, available to the community. It has client-server architecture and was developed using PHP for the application hosted on web server (Apache) and JavaFX for the experiments control panels. It allows the scheduling of sessions to use the experiments, enables the creation, editing and removal of experiments, users, booked sessions, user groups and experiment groups. In parallel, it was developed a set of gauges similar to those used in measure equipment and a set of other support tools to facilitate the development of new experiment control panels.

The communication between the client and the server is carried over the HTTP protocol and between the server and the experiment, it uses a specific text-based protocol, developed exclusively for this platform, based on the TCP/IP layers.

After some testing, this architecture is slower than the one where the client connects directly to the experiment, in about 130 ms. More tests will have to be carried out to see if it's possible to low this value.

The system has only been tested in a preliminary way, to prove the concept. It works well, but needs some further development to improve the response times of individual controls for certain types of experiments that need to be observed and monitored in a more "real time" way.

# Agradecimentos

Em primeiro lugar, gostaria de agraceder ao Professor José Manuel Magalhães Cruz por ter aceite ser o meu orientador, pelo acompanhamento que fez ao longo desta dissertação, pelas suas ideias e sugestões e, em especial, pela tempo dispendido a rever este documento. Quero agradecer também ao Prof. José Carlos Alves pelo acompanhamento feito ao longo desta dissertação.

Quero também mostrar a minha gratidão para com os meus companheiros de laboratório, especialmente ao Eduardo e ao Miguel, por estarem sempre prontos a ajudar.

Gostaria também de agradecer a toda à minha família, que nas alturas boas e menos boas, me apoiaram, ajudaram e me proporcionaram momentos inesquecíveis. Um agradecimento muito especial para o meu pai e para a minha mãe, que embora já não se encontre entre nós, estará sempre presente no meu pensamento.

Estendo também os meus agradecimentos àqueles (Ferreira, Pires, Rui, Grilo, Nélson, Hélder) que passaram tardes ou noites de trabalho comigo.

Agradeço também a todos os meus amigos, em especial ao Flávio, à Sandra, à Lili, à Xana, ao Renato e ao Otávio pelo seu apoio e amizade.

A todos aqueles, que embora não constem nesta página, também fazem parte da minha vida deixo os meus mais profundos agradecimentos.

João Filipe Dias Gonçalves

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Abreviaturas e Símbolos

| | |
|---|---|
| API | Application Programming Interface |
| ARM | Advanced Risc Machine |
| CD | Create Delete |
| CGI | Common Gateway Interface |
| CRUD | Create Read Update Delete |
| CSS | Cascading Style Sheets |
| DIESEL | Distance Internet-Based Embedded System Experimental Laboratory |
| EPSRC | Engineering and Physical Sciences Research Council |
| FAQ | Frequently Asked Questions |
| FI | Forms Interpreter |
| FPGA | Field-Programmable Gate Array |
| FTP | File Transfer Protocol |
| HDL | Hardware Design Language |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| HTTPS | HyperText Transfer Protocol Secure |
| IDE | Integrated Development Environment |
| IIS | Internet Information Services |
| IP | Internet Protocol |
| ISEL | Intelligent Systems Engineering Laboratory |
| JEHUTY | Java Web Hyper modUlar inTerface sYstem |
| JNLP | Java Network Launch Protocol |
| JRE | Java Runtime Environment |
| JSON | JavaScript Object Notation |
| JVM | Java Virtual Machine |
| LAN | Local Area Network |
| LCD | Liquid Crystal Display |
| LED | Light-Emitting Diode |
| LO | Learning Object |
| MIT | Massachusetts Institute of Technology |
| MVC | Model-View-Controller |
| MPM | Multi-Processing Module |
| ORM | Object-Relational Mapping |
| PC | Personal Computer |
| RTOS | Real time Operating System |
| SO | Sistema Operativo |
| SoC | Systems on Chip |

| | |
|---|---|
| TCP | Transmission Control Protocol |
| TV | Televisão |
| URL | Uniform Resource Locator |
| VNC | Virtual Network Computing |
| XAML | eXtensible Application Markup Language |
| XML | eXtensible Markup Language |

# Capítulo 1

# Introdução

As aulas laboratoriais existentes no ensino superior assumem extrema importância na preparação dos alunos para o mercado de trabalho especialmente na área das engenharias, existindo mesmo quem afirme que "as aulas laboratoriais estão no centro do ensino científico" [Ner89]. No entanto, para as aulas laboratoriais existirem é necessário ter acesso a um laboratório, ao material necessário e ter pelo menos um professor para orientar os alunos na execução das experiências. Isto torna-as caras e extremamente restritas no tempo pois só poderão ter lugar quando o laboratório estiver livre ou o professor estiver disponível.

Com o desenvolvimento das tecnologias e da Internet, novas alternativas têm vindo a surgir de forma a permitir a simulação de experiências ou a sua condução remota. Através destas alternativas, é possível reduzir os custos e/ou optimizar o aproveitamento das experiências e do laboratório.

## 1.1 Contexto/Enquadramento

É com a intenção de possibilitar aos alunos o acesso remoto às experiências à hora que pretenderem e quando o pretenderem que esta dissertação foi elaborada.

A plataforma actualmente utilizada na Faculdade de Engenharia da Universidade do Porto tem sérias limitações e os custos de actualização do *software* e aquisição de novo *hardware* são elevados pelo que se torna imperativo desenvolver e/ou adaptar uma plataforma que tenha um custo menor e que permita acesso remoto fácil dos alunos às experiências.

## 1.2 Motivação e Objectivos

O gosto do autor pela programação e desenvolvimento de aplicações *web* e a sua vontade de entender melhor o desenvolvimento e o funcionamento do *hardware* ligado a experiências de electrotecnia, foram a sua principal motivação.

Esta dissertação tem como objectivo principal o desenvolvimento de uma plataforma de suporte à experimentação remota. Esta plataforma deverá ser desenvolvida utilizando tecnologias *open-source* e ser disponibilizada, também como uma tecnologia *open-source*. Deverá permitir que o utilizador faça a reserva de uma sessão, para utilização de uma experiência, via *web*, e que, quando chegada a hora da sessão, possa aceder ao painel de controlo da experiência e conduzi-la também através da *web*. O painel de controlo da experiência deverá retratar, da melhor forma possível, os instrumentos disponíveis no laboratório real e deverá permitir ao utilizador a visualização, em tempo real, do que está a acontecer no laboratório.

## 1.3 Estrutura da Dissertação

Esta dissertação é composta por 5 capítulos, incluíndo a introdução.

No capítulo 2, serão analisadas outras plataformas de suporte a laboratórios remotos desenvolvidas a nível mundial assim como algumas das tecnologias disponíveis para o desenvolvimento e implementação deste tipo de laboratórios.

No capítulo 3, serão especificados os requisitos da plataforma a desenvolver, seguidos das tecnologias escolhidas e as razões por detrás de cada escolha. Serão também apresentadas as arquitecturas adoptadas para o sistema, incluindo a sua base de dados.

No capítulo 4, serão apresentados alguns dos detalhes do desenvolvimento e implementação do sistema e as suas funcionalidades.

Por fim, no capítulo 5, serão apresentadas as conclusões retiradas da elaboração desta dissertação e serão apresentadas algumas melhorias que se poderão fazer na plataforma desenvolvida.

# Capítulo 2

# Revisão Tecnológica

Este capítulo destina-se principalmente à apresentação de alguns projectos já desenvolvidos a nível mundial no âmbito dos laboratórios remotos. Alguns destes projectos são soluções específicas para um determinado tipo de experiência, que pode ser conduzida remotamente.

De qualquer forma, a análise destes exemplos, a maioria dos quais plenamente funcionais, servirá de orientação para a concretização do sistema especificado nos objectivos desta dissertação.

Pretende-se também apresentar com este capítulo, um conjunto de tecnologias que se pensa serem úteis para o projecto a desenvolver.

## 2.1 Projectos de Laboratórios Remotos

Projectos de laboratórios remotos têm vindo a suscitar grande interesse [MN06a] e devido à suas vantagens [GB09], várias instituições têm vindo a desenvolver plataformas de suporte a este tipo de laboratórios. Nas sub-secções seguintes vão ser analisados os seguintes projectos:

- Remote Controlled Laboratory (RCL);
- DIESEL Project ;
- The iLab Project
- JEHUTY
- Remote Laboratory Access for Hardware Design
- University of South Australia

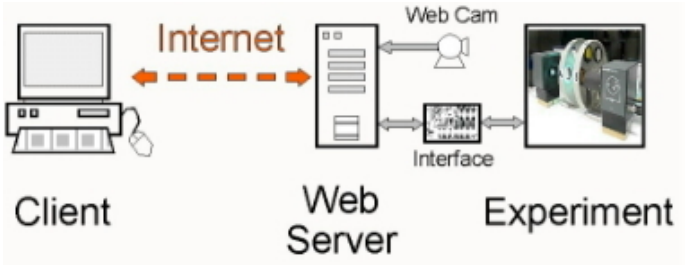



Figura 2.1: Arquitectura do RCL (Fonte: http://rcl.physik.uni-kl.de/eng/about.htm)

  - NetLab
  - MicroLab

- McGill University
- Remote Lab FEUP - Laboratório para a Instrumentação e Medição

### 2.1.1 Remote Controlled Laboratory [GVEJ07]

Esta plataforma, cuja descrição terá como base o artigo [GVEJ07], foi desenvolvida pelo Departamento de Física da Universidade Técnica de Kaiserslautern (DFUTK) e está disponível em http://rcl.physik.uni-kl.de/

O seu desenvolvimento teve início no ano de 2002 através da preparação de um protótipo para analisar a difracção de electrões provocada por uma folha de grafite.

Depois de desenhado o protótipo, foram construídas mais três experiências que estiveram disponíveis numa exposição no "Deutsches Museum", em Munique, cujo tema era "O Clima". Durante o ano que estiveram disponíveis, estas três experiências foram utilizadas por cerca de 20 000 utilizadores.

Actualmente esta plataforma, gerida pelo DFUTK, tem 15 experiências disponíveis *online*, das quais se destacam:

- Tomografia Computorizada;
- Túnel de Vento;
- Velocidade da Luz;

A plataforma tem uma arquitectura cliente-servidor (Fig. 2.1), em que a aplicação existente no servidor foi desenvolvida em PHP. A interface gráfica foi desenvolvida em JAVA, podendo assim ser executada em ambiente Windows, UNIX ou OS X. Basta para isso que o utilizador tenha instalado no seu computador uma máquina virtual JAVA.

Tendo a aplicação existente do lado do servidor sido desenvolvida em PHP, pode-se partir do princípio de também poder ser instalada em diversos sistemas operativos.

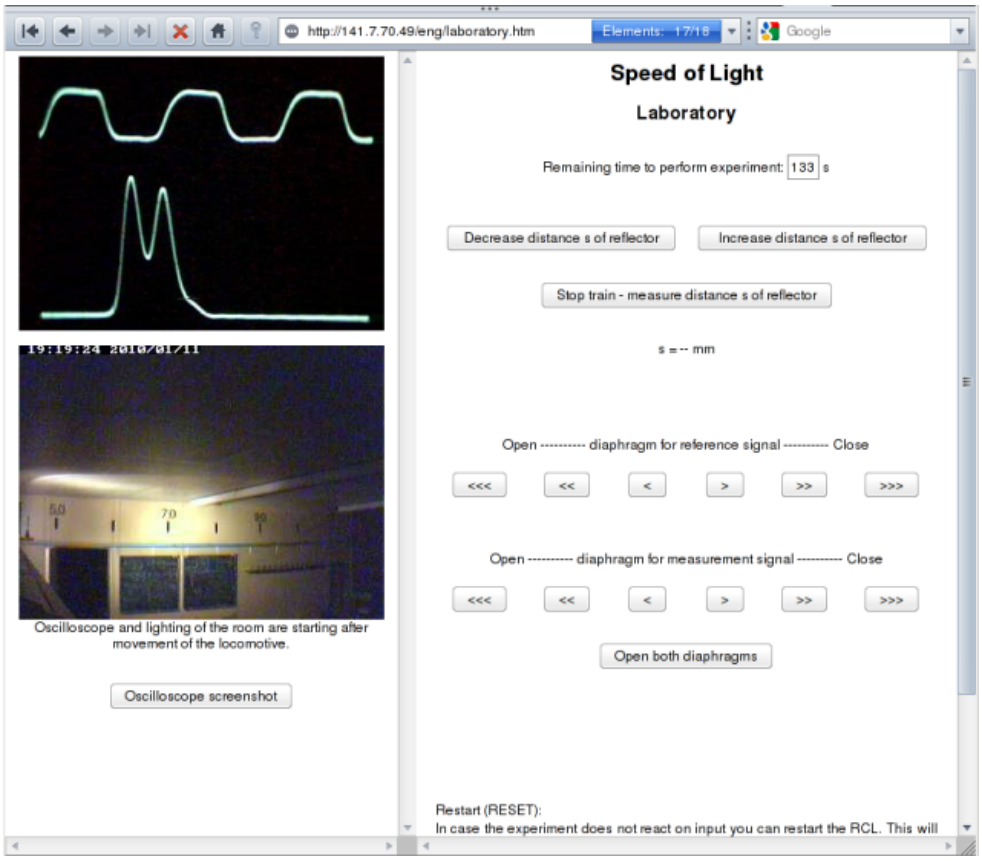



Figura 2.2: Interface gráfica da experiência para medir a velocidade da luz (Fonte: http://141.7.70.49/eng/laboratory.htm)

Claro que isto só poderá ser verdade, no caso da aplicação apenas utilizar funções PHP verdadeiramente multi-plataforma.

A aplicação foi desenvolvida de modo a facilitar a adição de novas experiências ao sistema. Para adicionar uma experiência, basta adicionar ao servidor o respectivo módulo, que é composto por um ou mais *scripts* que irão apresentar a página da experiência e fazer a interpretação dos comandos a enviar para a experiência.

Apesar de nas referências consultadas estar declarado que as interfaces gráficas foram implementadas em JAVA, o que se verificou através da utilização da plataforma, foi que apenas parte de uma das 15 interfaces utiliza a tecnologia JAVA. Em todas as outras, as acções que o utilizador submete são enviadas sob a forma de formulários HTML.

As interfaces gráficas utilizadas neste projecto são muito simples, como se pode ver na Fig. 2.2.

Este facto tem a vantagem do utilizador aprender a controlar a experiência rapidamente, mas tem como desvantagem o facto de não reproduzir minimamente o ambiente que existe quando utilizadores estão perante as experiências, o que poderá levar ao aparecimento de um certo desinteresse por parte do utilizador, no uso da plataforma.

Para que o utilizador possa ver o que está a acontecer no laboratório físico, em algumas das experiências, são fornecidas na interface uma ou duas imagens obtidas por cãmaras de vídeo *web*. Estas imagens podem ser da experiência em si e/ou de algum instrumento de medição nela utilizado (no caso da experiência da velocidade da luz, um osciloscópio).

### 2.1.2 DIESEL Project [CHM+06]

O projecto DIESEL, cuja base para a descrição que vai ser realizada neste sub-capítulo foi o artigo [CHM+06], foi um projecto com duração de 3 anos, fundado pelo EPSRC, desenvolvido no ISEL, localizado na Universidade de Ulster na Irlanda do Norte. O objectivo do projecto era disponibilizar o acesso remoto a experiências para processamento digital de sinal e microcontroladores, programação de FPGA, RTOS e SoC. Este projecto foi desenvolvido em duas fases distintas:

- 1ª Fase - O projecto tinha uma arquitectura semelhante à de outros projectos, o utilizador controlava a experiência remotamente através de comunicação directa com os aparelhos nela utilizados.
- 2ª Fase - A arquitectura do projecto foi alterada de forma a permitir o trabalho colaborativo nas experiências. Ou seja, através de um sistema de convites, vários utilizadores podiam partilhar uma mesma sessão, como se de um trabalho em grupo se tratasse.

#### 1ª Fase

Nesta fase, a arquitectura do projecto (Fig. 2.3) era relativamente simples. O servidor *web* era utilizado para fazer as reservas das sessões e para fazer a autenticação dos utilizadores perante a aplicação cliente e a aplicação servidor, sendo esta última realizada através de um *web service* utilizando o protocolo SOAP e uma ligação cifrada. Depois do utilizador estar autenticado, a comunicação passava a ser efectuada directamente entre a aplicação cliente e a aplicação servidor existente em cada máquina com ligação às experiências, também através de uma ligação cifrada.

Neste projecto temos presentes várias tecnologias proprietárias:

- LabVIEW - Interfaces dos instrumentos virtuais (osciloscópios, multímetro digital e um gerador de sinal);
- Adobe Flash[1] - A área onde o utilizador desenha o circuito que quer implementar e também na transmissão e visualização das imagens provenientes das *webcams*;
- Microsoft C# com .NET - O núcleo da aplicação foi desenvolvido em C# com recurso à *framework* .NET.

[1] Na altura em que o projecto foi desenvolvido a proprietária do Flash ainda era a empresa Macromedia

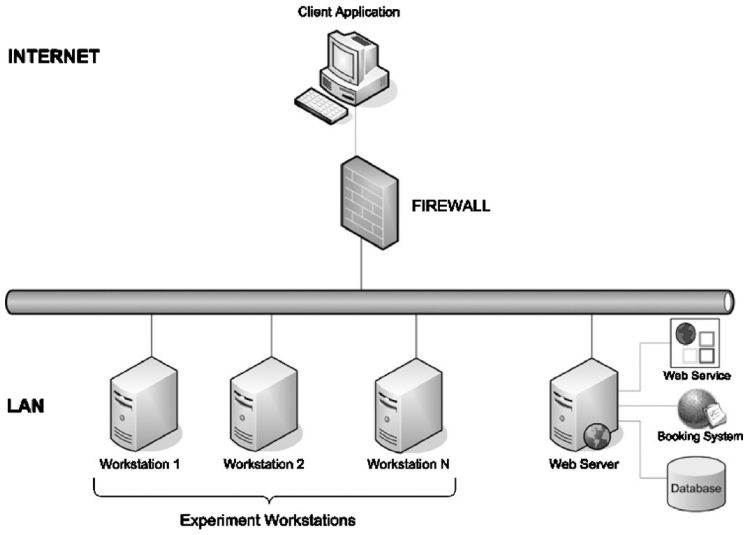



Figura 2.3: Arquitectura 1ª Fase (Fonte: [CHM+06])

### 2ª Fase

A 2ª fase foi onde se começou a pensar na possibilidade de vários alunos trabalharem em grupo numa única sessão, de modo a recriar o ambiente existente nas aulas. Com este objectivo, a arquitectura foi redesenhada e foi adicionado um servidor para gerir o trabalho colaborativo, pelo que a arquitectura resultante pode ser visível na Fig. 2.4.

Esta nova arquitectura foi desenvolvida e implementada em C# e também utiliza serviços *web* para fazer a autenticação dos utilizadores. No entanto, para que outros utilizadores se pudessem conectar à mesma experiência, foi necessário introduzir algumas alterações. Algumas das alterações efectuadas foram, por exemplo, a utilização de objectos remotos e a subscrição de eventos entre todas as comunicações clientes-servidor e a modificação do protocolo utilizado para partilha do ambiente de trabalho remoto (VNC), de modo a suportar vários utilizadores simultâneos. A subscrição de eventos, também utilizada no servidor colaborativo, permite que quando um utilizador provoca uma alteração do lado do servidor, todos os clientes são notificados desta.

A interface gráfica onde se desenhava os circuitos eléctricos virtuais também foi alterada para utilizar a funcionalidade de objectos partilhados (Shared Object) do *Macromedia Flash Communication Server* deforma a suportar a interacção simultânea de vários utilizadores. Com a utilização deste servidor também foi possível adicionar funcionalidades de comunicação entre os utilizadores através de texto e de vídeo e áudio com *webcams*.

**Conclusão** Este projecto tem a vantagem de permitir o trabalho em grupo, permitindo que os utilizadores de uma sessão comuniquem e se ajudem uns aos outros. Uma das des-

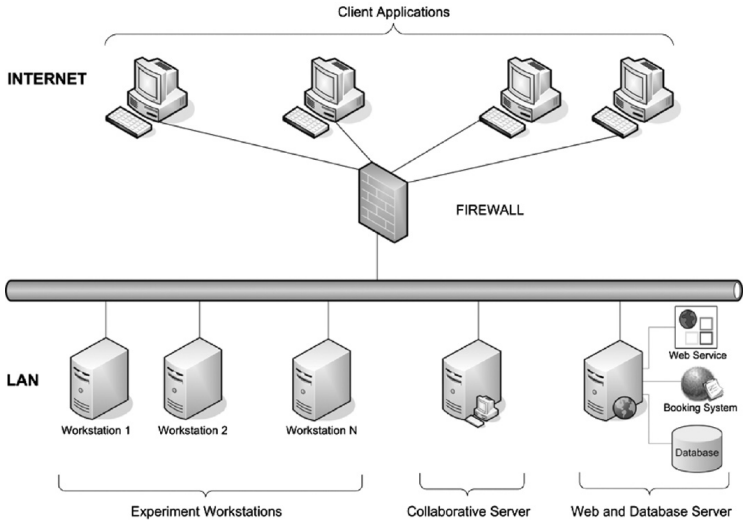



Figura 2.4: Arquitectura 2ª Fase (Fonte: [CHM+06])

vantagens deste projecto é o facto de ter sido implementado em tecnologias proprietárias (e.g. Microsoft), o que vai limitar a sua utilização por cliente e implementação nos servidores a apenas computadores que tenham instalado o sistema operativo Windows. O facto de quase todas as tecnologias implementadas, serem proprietárias, exigirá a aquisição de licenças, cujo custo pode ser um factor limitativo.

### 2.1.3 The iLab Project([MIT], [HMJ06])

A partir dos artigos [MIT] e [HMJ06], vai ser feita uma descrição deste projecto financiado pela Microsoft, através do *iCampus Project* no MIT. Os objectivos principais deste projecto seriam:

- Disponibilizar uma infraestrutura de *software* que permitisse a utilização de laboratórios *online* e, simultaneamente, reduzisse os seus custos de montagem e manutenção;
- A infraestrutura deveria ser suficientemente genérica para suportar vários tipos de experiências sem ser necessário aumentar a carga de trabalho, quer dos técnicos do laboratório quer dos docentes da disciplina associados;
- Proporcionar uma infraestrutura escalável;
- Encorajar a cooperação entre diferentes instituições, de forma a que cada uma disponibilizasse os seus próprios laboratórios remotos. No caso de se utilizar esta infraestrutura, passaria a ser possível os alunos da instituição B acederem aos laboratórios da instituição A, mediante uma determinada política de acesso pré-estabelecida.

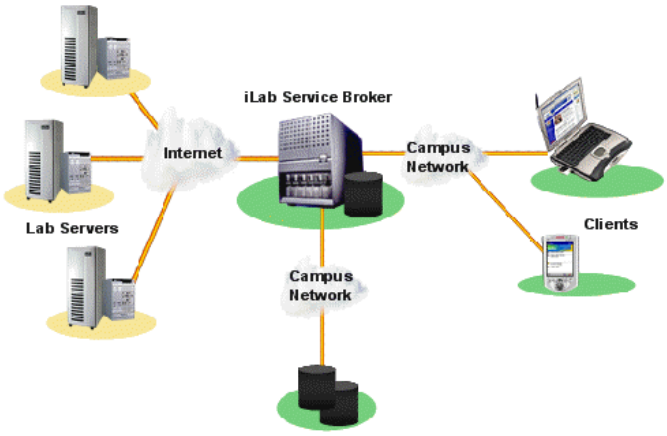



Figura 2.5: Arquitectura - iLab (Fonte: http://icampus.mit.edu/ilabs/architecture/)

Com vista a atingir os objectivos, a arquitectura implementada foi a que se pode observar na Fig. 2.5.

Com esta arquitectura e devido à multiplicidade de *frameworks* existente para desenvolver aplicações distribuídas, a utilização de *web services* foi a solução adoptada para efectuar a comunicação entre o cliente e o *iLab Service Broker* (ISB) e entre o ISB e o *Lab Server*. A utilização de *web services* permite a comunicação entre clientes e servidores num formato uniformizado, independentemente das plataformas e linguagens utilizadas.

O primeiro tipo de experiências a ser suportado por este projecto, foi aquele que não possibilita qualquer tipo de acção ao utilizador durante a realização da experiência (experiências não interactivas). Neste tipo, o utilizador submete ao servidor os parâmetros da experiência a realizar,que são validados e, depois da experiência terminar os resultados são retornados para o utilizador.

Numa fase posterior, foi implementado o suporte a experiências interactivas. No entanto, através da documentação [MIT07] afecta ao projecto, verifica-se que a interacção só suporta experiências que utilizem o *software* LabVIEW.

Enquanto que na utilização de experiências não interactivas, não existia comunicação directa entre o cliente e o *Lab Server* pois esta era mediada pelo ISB, na utilização de experiências interactivas, a comunicação passa a ser entre o *Lab Server* e o cliente, a partir do momento em que este se encontra validado no sistema.

Actualmente, o laboratório *online* disponibilizado pelo MIT encontra-se implementado sobre esta plataforma. O servidor foi desenvolvido em C# e as aplicações cliente foram desenvolvidas em JAVA, dado esta última ser multi-plataforma. O facto do servidor ter sido desenvolvido em C#, vem restringir a instalação deste a apenas sistemas Windows. No que respeita à base de dados, a que é referida no manual de instalação é SQL Server, também proprietária. Relativamente às experiências interactivas, como foi

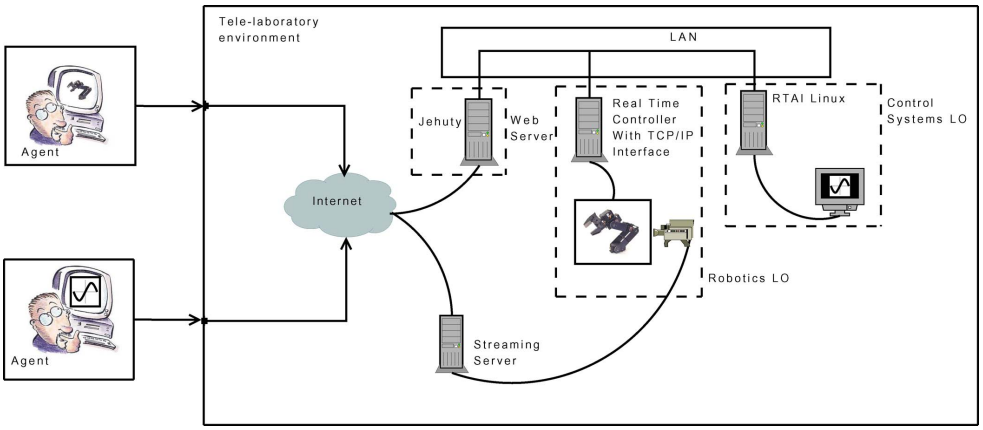



Figura 2.6: Arquitectura do sistema (Fonte: [BCC09])

referido anteriormente, a documentação apenas faz referência que estas são suportadas aquando a utilização do *software* LabVIEW. Os três factores acima enunciados podem ser impeditivos para a implementação do sistema por outras universidades pois são utilizadas ferramentas proprietárias, para as quais é necessária a aquisição de licenças e equipamento específico, cujo custo ainda é considerável.

### 2.1.4 JEHUTY [BCC09]

A descrição deste projecto, desenvolvido na Universidade de Pisa (Itália), será baseada no artigo [BCC09]. Este é uma plataforma de suporte a laboratórios remotos com uma arquitectura cliente-servidor (Fig. 2.6) muito semelhante àquela que acabou por ser utilizada para a realização desta dissertação.

O Jehuty é o servidor que faz de intermediário entre o cliente e os *Learning Objects* (LOs). É este servidor que vai também fazer toda a gestão de utilizadores, permissões, acessos e agendamentos. A ligação entre o servidor e os LOs é assegurada por uma rede local, usando TCP/IP. A comunicação entre os clientes (interfaces gráficas dos LOs) e o servidor *web*, é efectuada recorrendo a *web services*.

O sistema é modular, baseado na utilização de *Java Servlets*, e vai construindo as páginas dinamicamente conforme a informação do utilizador e os dados da base de dados. Está distribuído em cinco camadas:

- JXMLDC - *JEHUTY XML Database Connector*. *Parser* para ler o ficheiro de configuração do sistema.

- JDD - *JEHUTY Database Driver*. Camada que faz a ligação com o servidor de base de dados MySQL.

- JC - *JEHUTY Core*. Através das duas camadas enunciadas anteriormente, liga-se à base de dados, verifica as permissões dos utilizadores e gere as regras de acesso a cada módulo.
- JM - *JEHUTY Modules*. Implementados em *Java Interface*, são inicializados pelo JC. É através destes módulos que se pode interagir com o sistema (e.g. alterar regras de autenticação, variáveis globais, variáveis de sessão).
- JTS - *JEHUTY Template System*. Sistema de modelos que permite manter toda a camada lógica do sistema separada da camada de apresentação da informação. Os módulos podem ser inicializados com ou sem o JTS.

Relativamente ao sistema de permissões, existem 4 grupos de utilizadores:

- *Students* - Depois de registados, escolhem um professor (*Teacher*) para validar o seu registo. Depois do registo ser validado, podem utilizar os LOs que pretenderem através de um sistema de reservas, aceder aos seu histórico, à FAQ, ao material teórico, etc.
- *Teachers* - Validam os registos dos alunos, podem enviar para o servidor material didáctico, adicionar apontadores e ver o histórico e os resultados dos seus alunos.
- *System Administrators* - Podem adicionar e remover professores.
- *Developers* - Fazem a gestão dos LOs disponíveis e podem alterar o código fonte do sistema.

As páginas de apresentação dos LOs seguem uma estrutura padrão (Fig. 2.7). Nestas páginas é possível visualizar os apontadores e todo o material didáctico associado a cada LO. Existe também um botão para iniciar a interface gráfica para o controlo do LO e um apontador que abre uma página onde se pode ver em tempo real o que está a acontecer no LO, através de uma *webcam*.

Actualmente os LOs que se podem utilizar através desta plataforma são o controlo de um braço robótico (GeT) e um laboratório virtual (Virtual Lab) onde o aluno aprende a desenhar um controlador para sistemas lineares invariantes no tempo.

As interfaces gráficas para controlar/utilizar cada LO são implementadas através de *JAVA Applets*. Através do recurso à tecnologia JAVA, é assegurado que a maior parte dos alunos poderá controlar os LOs, dado estar disponível na maioria das plataformas informáticas actuais.

Tomando como exemplo a interface gráfica do primeiro LO (Fig. 2.8), vê-se que é uma interface simples onde o utilizador tem de escrever um programa através do *drag and drop* de comandos, que ao serem interpretados pelo sistema durante a execução da experiência, vão-se traduzir numa sequência de movimentos que o braço vai efectuar.

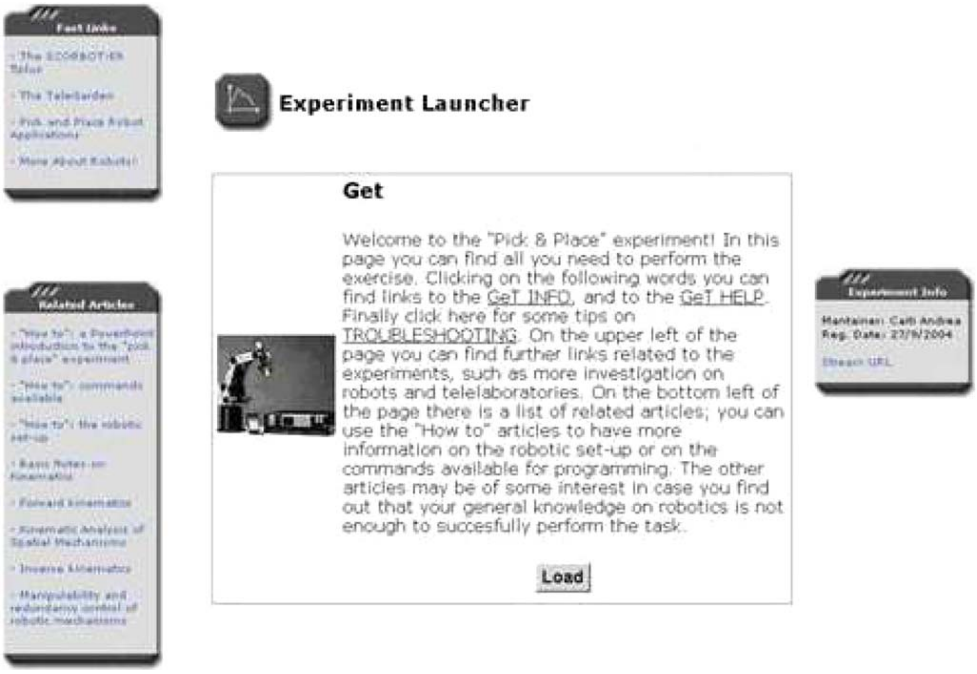



Figura 2.7: Página de apresentação de um LO (GeT) (Fonte: [BCC09])

Para esta interface, foi necessário desenvolver uma nova linguagem de programação relativamente flexível, que fosse transparente para o utilizador e que lhe permitisse focar a sua atenção na aprendizagem do conceito e não na linguagem. A linguagem suporta dois tipos de comandos: os comandos de controlo de fluxo (*if, while, repeat*) e os comandos de acções (abrir garra, fechar garra, movimento, etc).

O facto da estrutura das páginas dos LOs seguir um determinado padrão, obriga os *Developers* a seguirem uma determinada sequência de passos, em que um deles é fornecer o *applet* que irá ser lançado para que o utilizador possa interagir com o LO. No entanto, para ser criado um novo LO, é necessário fazer uma avaliação rigorosa da experiência que se pretende adicionar, nomeadamente quais os benefícios que se poderão ter e se é possível mapear a experiência num LO.

### 2.1.5 Remote Laboratory Access for Hardware Design [HP09]

Este sistema foi desenvolvido e implementado na *Northern Illinois University* e tem como objectivo permitir aos alunos desenhar e testar circuitos através da programação de uma FPGA.

Baseado também numa arquitectura cliente-servidor, o sistema disponibiliza aos alunos dois tipos de servidores: servidores com apenas placas de desenvolvimento e servidores com placas de desenvolvimento e equipamentos de medição. No entanto, convém

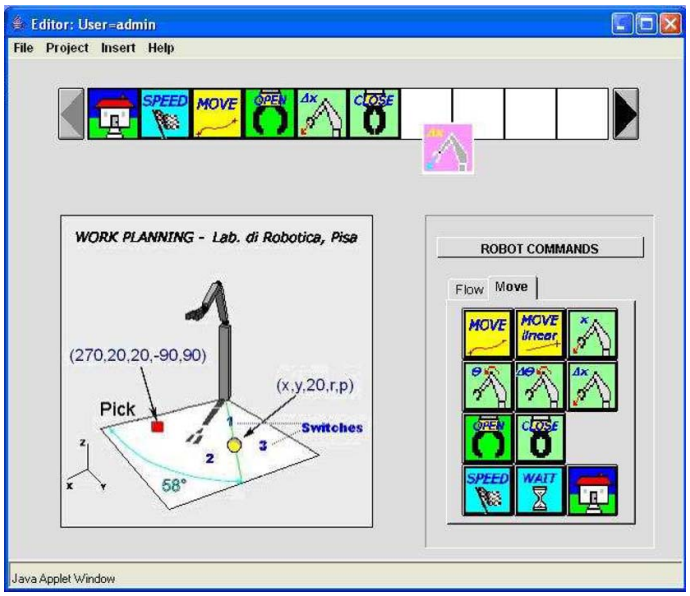



Figura 2.8: Interface Gráfica - Braço Robótico (Fonte: [BCC09])

referir que dos actuais oito servidores disponíveis, apenas três possuem ligação a equipamentos de medição.

Este projecto começou por ser implementado, numa primeira fase, recorrendo à tecnologia LabVIEW, da National Instruments. Nesta fase, só estavam disponíveis para utilização as placas de desenvolvimento e foi necessária a inclusão de algum *hardware* extra para facilitar a comunicação e aquisição de dados.

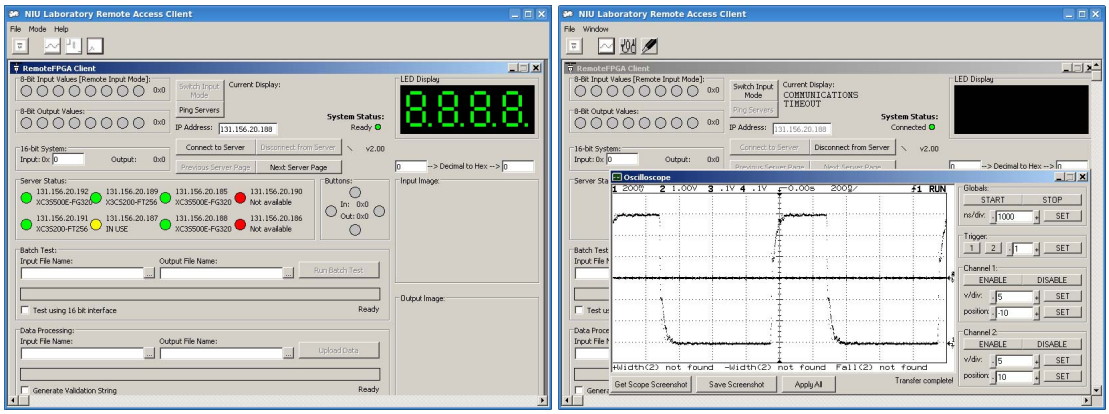
Actualmente a plataforma já funciona sem o *hardware* extra e sem recorrer ao *software* da National Instruments. Esta nova plataforma permite aos alunos ver a taxa de ocupação dos servidores e o seu tipo. A partir do momento que um aluno se liga a um dos servidores, o servidor e todos os seus recursos são reservados durante um intervalo de tempo. O sistema também suporta a prioritização dos acessos, mediante os projectos dos alunos.

O software que suporta este sistema divide-se em três módulos, sendo eles:

- *Remote Access Interface Module (RAIM)* - É uma aplicação escrita em HDL, que se encarrega de gerar um bloco de circuitos na FPGA que irá assegurar a comunicação entre o circuito implementado pelo utilizador na FPGA e o cliente. Esta comunicação é feita através de porta série entre a placa de desenvolvimento e o servidor e através da rede entre o servidor e o cliente.

- *Remote Access Server (RAS)* - Aplicação, em execução no servidor, que faz a ligação entre a placa de desenvolvimento e o cliente (aluno).

- *Remote Access Client (RAC)* - Aplicação que é executada do lado do cliente. Depois do utilizador seleccionar o servidor ao qual se quer conectar, a aplicação estabelece




(a) Vista para controlo da placa de desenvolvimento

(b) Vista do equipamento de medição

Figura 2.9: RAC - Interfaces Gráficas (Fonte: [HP09])

a conexão e o utilizador pode então interagir com a placa de desenvolvimento (Fig. 2.9(a)). Dependendo do tipo de servidor ao qual o utlizador se conectou, também é possível consultar os equipamentos de medição (Fig. 2.9(b)).

Pelo que se entende da leitura de [HP09], onde foi baseada a descrição acima efectuada, a aplicação cliente pode ser executada em ambientes Linux ou Windows. No entanto, a ligação entre o cliente e o servidor é estabelecida directamente entre eles, havendo necessidade de fazer reencaminhamento de portos nas *firewalls* para os respectivos servidores.

### 2.1.6 University of South Australia

Nesta sub-secção vão ser analisadas duas plataformas desenvolvidas na School of Electrical and Information Engineering, University of South Australia (UniSA). Inicialmente vai ser analisada a plataforma Netlab, que é utilizada para analisar circuitos. De seguida irá ser feita a análise da plataforma Microlab, que pode ser utilizada para testar e executar código de microcontroladores em *hardware* real. Em ambas as plataformas, tudo é realizado remotamente, ou seja, sem a necessidade de estarmos perante a experiência física.

#### 2.1.6.1 NetLab ([MN06b, Uni10, NMN08])

Esta plataforma começou a ser desenvolvida em 2002, tendo contribuído para que os seus autores, Jan Machotka e Zorica Nedic, fossem galardoados com o prémio *Excellence in Teaching Award* em 2004, atribuído pela UniSA [MN06b].

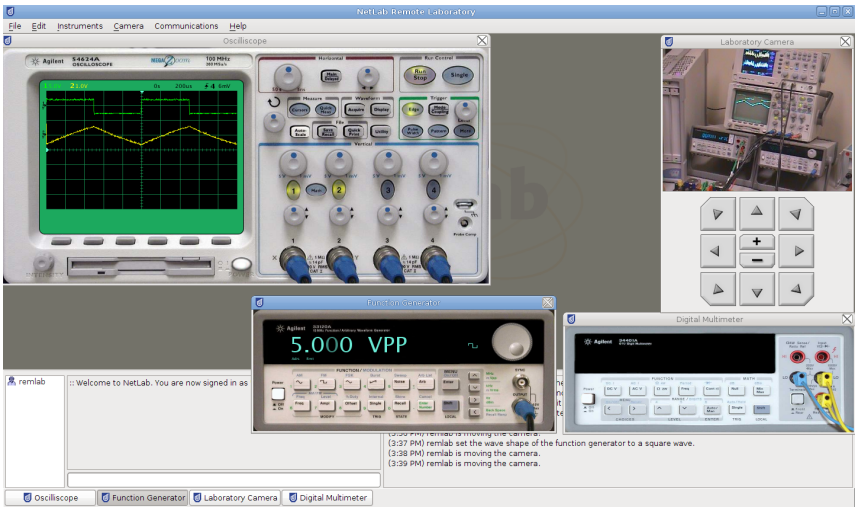



Figura 2.10: Interface Gráfica do NetLab

A plataforma baseia-se numa arquitectura cliente-servidor, sendo que, inicialmente a aplicação que era executada do lado do servidor foi desenvolvida recorrendo ao LabView e a interface gráfica foi desenvolvida em JAVA [MN06b]. No entanto, no ano de 2006, Ben Cloud juntamente com Brett Thredgold, reescreveram o projecto [Uni10], desta vez utilizando JAVA, tanto do lado do cliente como do lado do servidor [NMN08].

A interface gráfica está muito bem conseguida e muito completa, conseguindo passar a ideia ao utilizador de que se encontra perante a experiência real (Fig. 2.10).

Através da interface gráfica é possível:

- Controlar a câmara de vídeo que filma a experiência, movendo-a na vertical, horizontal e fazendo *zoom*;
- Abrir ou fechar as janelas de cada instrumento, câmara ou do *Circuit Builder*;
- Falar com outros utilizadores que tenham agendamentos para a mesma hora, até um máximo de 3 (Trabalho em grupo);
- Ver o que vai acontecendo à medida que se utiliza a aplicação, através da área de notificação;
- Desenhar o circuito que se pretende analisar, podendo-se utilizar diversos componentes eléctricos (resistências, condensadores, bobines e transformadores), e equipamentos como geradores de funções, multímetros e osciloscópios (Fig. 2.11);
- Ver as alterações ao longo do tempo, à medida que se alteram os valores dos vários componentes do circuito.

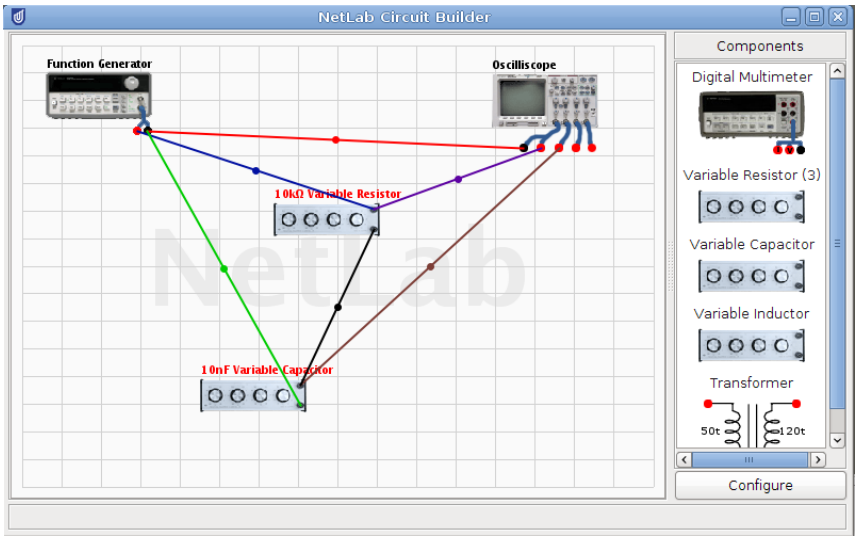



Figura 2.11: *Circuit Builder* - Onde se pode desenhar o circuito a testar

A interface gráfica e o modo de utilização são de muito boa qualidade. Infelizmente, pela leitura da bibliografia existente sobre o projeto e pela procura efectuada no *website* da plataforma, aparentemente só é possível analisar o comportamento de circuitos eléctricos, não havendo referências a outros tipos de experiências.

#### 2.1.6.2 MicroLab [Uni06]

O MicroLab foi um projecto de laboratórios remotos também desenvolvida na UniSA, cujo objectivo era permitir aos seus utilizadores testar código desenvolvido para micro-controladores em *hardware* real.

O dispositivo montado foi projectado para incluir um microcontrolador PIC16F882, alguns componentes de saída (7 LEDs e um *display* LCD) e alguns componentes de entrada (um botão de pressão e um teclado de 16 botões). Para permitir a programação do *chip* e total controlo a partir do computador, inclui um programador PicKit2 e uma unidade de aquisição de dados.

A arquitectura do sistema pode ser visualizada na Fig. 2.12. Neste sistema, a aplicação que é executada do lado do servidor é que trata da ligação ao *hardware* e faz a gestão de todos os serviços e das ligações dos clientes. Do lado do cliente, será executada a interface gráfica que permitirá ao utilizador controlar a experiência e observar seu desenrolar através dos componentes de saída (LEDs e LCD). Existe ainda um servidor *web* onde o utilizador poderá registar-se, fazer a gestão da sua conta, consultar alguma documentação sobre o projecto e criar os seus agendamentos de modo a ter acesso à experiência na hora especificada.

Neste projecto identificam-se duas grandes desvantagens:

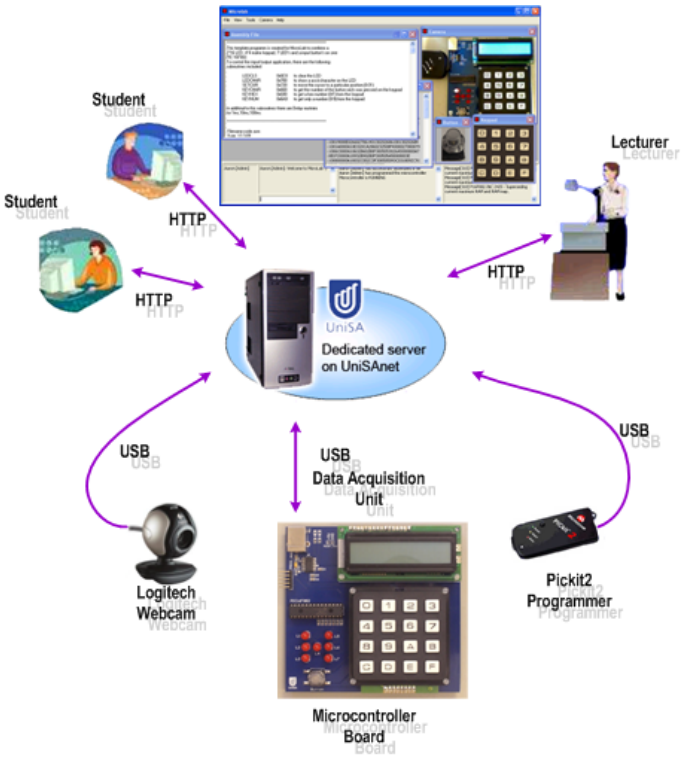



Figura 2.12: Arquitectura utilizada no MicroLab (Fonte: http://microlab.unisa.edu.au/howMicroLabWorks.xhtml)

- O facto dela ter sido construída sobre o software LabVIEW, que, na versão utilizada no projecto, apenas permite a utilização da aplicação em ambiente Windows;
- O facto da utilização da interface, implicar a descarga do *LabVIEW Run-Time Engine 8.6*, que ocupa cerca de 108 MB. Isto, para um utilizador que tenha uma largura de banda reduzida, pode ser um factor impeditivo para utilização da aplicação.

### 2.1.7 McGill University [MZC03]

Existe um projecto desenvolvido na *McGill University*, em Montreal no Canadá [MZC03], que data de 2003 e cuja tecnologia utilizada poderá já não existir ou então, estar desactualizada. No entanto, o projecto merece ser referido, pois demonstra uma outra forma de abordar a implementação de laboratórios remotos.

Este projecto suporta o trabalho colaborativo, ou seja, é possível estarem vários utilizadores a ver o que se está a passar na experiência e a falarem através de um sistema de mensagens instantâneas (*chat*).

Na Fig. 2.13, é possível ver o ecrã de trabalho que o utilizador terá quando estiver a utilizar o laboratório remoto. Na parte de cima da figura, pode-se ver os vídeos dos outros utilizadores, e à direita, está a janela do *chat* que permite a partilha de ideias ou opiniões e,

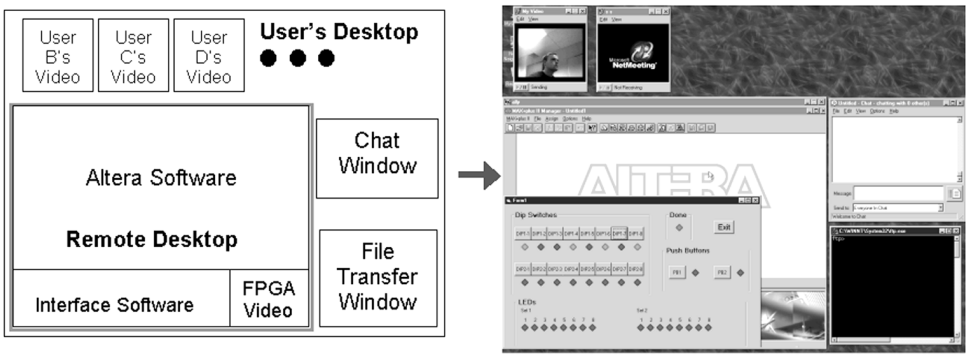



Figura 2.13: Estrutura da Interface gráfica (Fonte: [MZC03])

por baixo dessa janela, temos a janela para a troca de ficheiros através de comandos FTP. Na janela maior, é visualizado o ambiente trabalho remoto, onde se pode interagir com o *software* da Altera e uma outra aplicação auxiliar, através da utilização do protocolo VNC.

A funcionalidade de *chat*, troca de ficheiros e video-conferência é possível devido à utilização do NetMeeting [Mic10a]. A utilização deste *software* acaba por impôr uma limitação no cliente, dado que o NetMeeting só está disponível em sistemas Windows.

A funcionalidade de interacção com a FPGA é assegurada pela utilização do protocolo VNC, que permite ao utilizador o acesso ao ambiente de trabalho do servidor, como se o estivesse a utilizar directamente. Assim, é possível utilizar o *software* da Altera para compilação e programação da FPGA e uma aplicação auxiliar que permite:

- verificar se a FPGA já se encontra ou não programada;
- visualizar o estado das saídas existentes na FPGA;
- alterar o estado das entradas da FPGA, através de botões.

Apesar das vantagens que este projecto tem, ele também exibe limitações entre as quais se destacam a impossibilidade de se ligar algum tipo de equipamento de medição [MZC03] e o facto das tecnologias utilizadas necessitarem de uma quantidade significativa de largura de banda, o que pode ser um factor impeditivo para certos utilizadores.

### 2.1.8 Remote Lab FEUP - Laboratório para a Instrumentação e Medição

Este projecto está actualmente disponível na Faculdade de Engenharia da Universidade do Porto através da *web*[2] (Fig. 2.14). Estão disponíveis 4 experiências:

- Avaliação da Rectitude;

[2] http://remotelab.fe.up.pt/

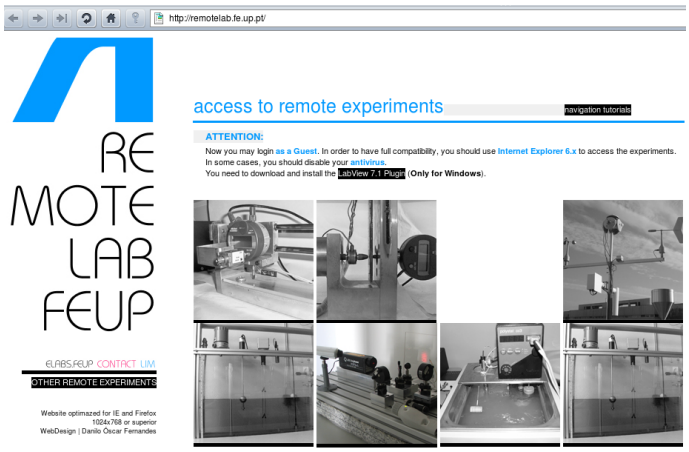



Figura 2.14: *Website* LIM

- Caracterização de Materiais;
- Procedimento de Calibração de Temperatura;
- Estação Meteorológica.

O projecto tem integração com o sistema Moodle[3], mas sofre no entanto de algumas falhas graves:

- Devido a problemas de actualização de *software* do servidor, neste momento só é possível utilizar as experiências como utilizador convidado, e não como utilizador com conta própria;
- Devido à versão do LabVIEW que foi utilizada (7.1) para configurar as experiências e disponibilizá-las *online*, só é possível utilizar o laboratório em sistemas operativos Windows e com o navegador Internet Explorer 6.0;

A experiência relativa à estação meteorológica tem um funcionamento ligeiramente diferente das outras experiências. Nesta experiência, implementada em LabVIEW 8.5, os dados são recolhidos e guardados numa base de dados. Assim, quando alguém acede à página da estação, os últimos dados recolhidos presentes na base de dados são apresentados na página. Também é possível efectuar pesquisas de dados recolhidos num determinado intervalo de tempo.

### 2.1.9 Conclusões

Para ajudar à comparação global dos sistemas descritos, foi elaborada a Tabela 2.1 onde se poderá ver as características gerais dos projectos.

Relativamente ao significado do campo do custo, considerou-se:

[3]Plataforma *open-source* de suporte ao *e-learning*, disponível para download em http://moodle.org/

Tabela 2.1: Tabela das Características Gerais dos Projectos

| | Multi-Plataforma | | | |
|---|---|---|---|---|
| **Projecto** | **Cliente** | **Servidor** | **Solução Específica** | **Custo** |
| RCL | Sim | Sim | Sim | Baixo |
| DIESEL Project | Não | Não | Não | Médio a Alto |
| The iLab Project | Sim | Não | Sim | Médio a Alto |
| JEHUTY | Sim | Sim | Sim | Baixo |
| Remote Laboratory Access for Hardware Design | Sim | - | Sim | Baixo a Médio |
| NetLab | Sim | Talvez | Não | Alto |
| MicroLab | Talvez | Talvez | Não | Alto |
| McGill University | Não | Não | Não | Médio |
| Remote Lab FEUP | Não | Não | Sim | Alto |

- *Baixo* - se se utilizar ferramentas *open-source* e linguagens multi-plataforma;

- *Médio* - se se utilizar ferramentas ou sistemas operativos proprietários, mas que, aparentemente, não exigem grandes investimentos iniciais relativos a licenças de utilização.

- *Alto* - Quando o projecto utiliza ferramentas proprietárias muito específicas, neste caso, software ou hardware da National Instruments.

A utilização de tecnologias proprietárias e, principalmente, fechadas nem sempre é a melhor opção pelo simples facto de depois se correr o risco de se ficar "preso" a uma determinada tecnologia, seja ela *software* e/ou *hardware*. Como tal, a escolha deste tipo de tecnologias tem de ser muito bem ponderado.

Quando a opção recai sobre as tecnologias abertas, normalmente a situação acima descrita tem menos probabilidades de ocorrer dado que, normalmente, este tipo de tecnologias permite uma maior flexibilidade.

Relativamente aos projectos analisados, alguns são de muito boa qualidade, pelo que se pode tentar aplicar algumas das ideias neles apresentadas. Um desses projectos é o NetLab e a sua interface gráfica de excelente qualidade, em que os painéis de controlo dos instrumentos utilizados são muito semelhantes aos reais. A possibilidade de, no projecto iLab, ser possível a partilha de experiências disponíveis entre diferentes instituições é também uma mais valia que poderia ser aplicada no projecto a desenvolver. A utilização de serviços *web* (*web services*) também poderá ser interessante, pois não restringe o desenvolvimento das interfaces gráficas a uma única tecnologia.

## 2.2 Tecnologias de Interesse

Com a diversidade de linguagens de programação e tecnologias existente hoje em dia, é aconselhável realizar um estudo prévio das vantagens e desvantagens de cada uma para o projecto a desenvolver, de forma a que se possa avaliar qual será a mais vantajosa.

Dado pretender-se uma plataforma do tipo cliente-servidor, torna-se necessário separar o estudo em duas partes: na primeira parte, serão avaliadas as tecnologias mais vulgarmente utilizadas para a programação do lado do servidor; na segunda parte, serão analisadas as linguagens que permitem efectuar a programação do lado do cliente e, ao mesmo tempo, permitem criar interfaces apelativas tanto no aspecto como na facilidade de utilização.

### 2.2.1 LabVIEW

O LabVIEW é um *software* proprietário desenvolvido pela *National Instruments*[4] (NI), permite a criação de sofisticados sistemas de controlo, teste e medição através de programação gráfica. Isto é, através da utilização de uma linguagem gráfica própria e de uma representação das ligações entre componentes muito semelhante a um diagrama de fluxo, possibilita que qualquer pessoa possa desenhar e montar uma experiência ou sistema sem ser necessário ter grandes bases de programação. A construção das interfaces gráficas onde se vão visualizar as informações recolhidas pelo *hardware* também segue um procedimento muito semelhante ao do desenho e programação do circuito. Para além da programação gráfica, também é possível programar através da API disponibilizada pela NI, permitindo o alargamento deste *software* a outras linguagens de programação.

No que diz respeito aos requisitos de utilização relativos aos sistemas operativos, é possível executar este *software* na maioria dos sistemas operativos utilizados actualmente assim como em sistemas operativos em tempo real.

Este *software* também tem óptima capacidade de integração com o *hardware* disponibilizado pela própria NI, permitindo também controlar e/ou programar diversos dispositivos (FPGA, processadores ARM, instrumentos de medição).

Relativamente ao suporte existente, a comunidade de utilizadores é significativa (cerca de 110000) e a própria NI também dá suporte ao *software* promovendo também cursos sobre como o utilizar.

A grande desvantagem deste *software* e do *hardware* é o seu elevado custo de aquisição. A título de exemplo, o pacote completo de *software* para desenvolver um projecto de laboratórios remotos em sistemas Windows fica por 2 599 €, sem qualquer extensão. Depois talvez fosse preciso adquirir algum *hardware* necessário para fazer a ligação entre

[4] http://www.ni.com/

a experiência e o PC que serviria de servidor. Desta forma, esta alternativa não é acessível a qualquer instituição.

### 2.2.2 Server-Side

#### 2.2.2.1 Arquitectura Model-View-Controller

O desenvolvimento e manutenção de aplicações pode tornar-se confuso e difícil quando se mistura a lógica de negócio da aplicação com código de apresentação da informação e com código de acesso à informação. A arquitectura Model-View-Controller (MVC) surgiu para dar resposta a este problema [Mic02]. As aplicações desenvolvidas segundo esta arquitectura dividem-se em três camadas:

- *Model* - Esta camada representa a informação e controla o seu acesso e actualização. É nela que também estão definidas as regras de negócio;

- *View* - Esta camada apresenta ao utilizador a informação contida no *Model* e mantém a sua consistência quando existem alterações no *Model*;

- *Controller* - É nesta camada que as acções do utilizador executadas na *View* vão ser convertidas em acções a serem executadas no *Model*. Dependendo do valor retornado pelas acções efectuadas no *Model*, o *Controller* seleccionará a *View* a apresentar ao utilizador.

Na Fig. 2.15 pode-se observar um diagrama genérico desta arquitectura onde também são apresentadas as interacções existentes entre as várias camadas.

Esta arquitectura permite a re-utilização dos *Models*, dado estes normalmente estarem acessíveis a partir de qualquer *controller*, e facilita a expansão de utilização da aplicação a novos dispositivos, pois apenas é necessário criar uma nova *View* e acrescentar algum código aos *Controllers* [Mic02]. A implementação de novas funcionalidades também é relativamente fácil, dependendo do tipo de funcionalidade, pois normalmente basta a criação de uma nova acção no *Controller* e a criação de uma nova *View*.

A desvantagem desta arquitectura é que implica um aumento na complexidade do projecto, devido à separação de classes que a arquitectura requer (*Model, View, Controller*).

O facto de existirem *frameworks* para o desenvolvimento de aplicações *web* que seguem esta arquitectura, facilita o seu desenvolvimento, pois assim o programador consegue focar-se realmente no problema a resolver. Essas *frameworks*, normalmente, fazem automaticamente a ligação à base de dados e permitem fazer o mapeamento directo entre tabelas existentes na base de dados e classes (ORM), entre muitas outras funcionalidades que aceleram o desenvolvimento das aplicações.

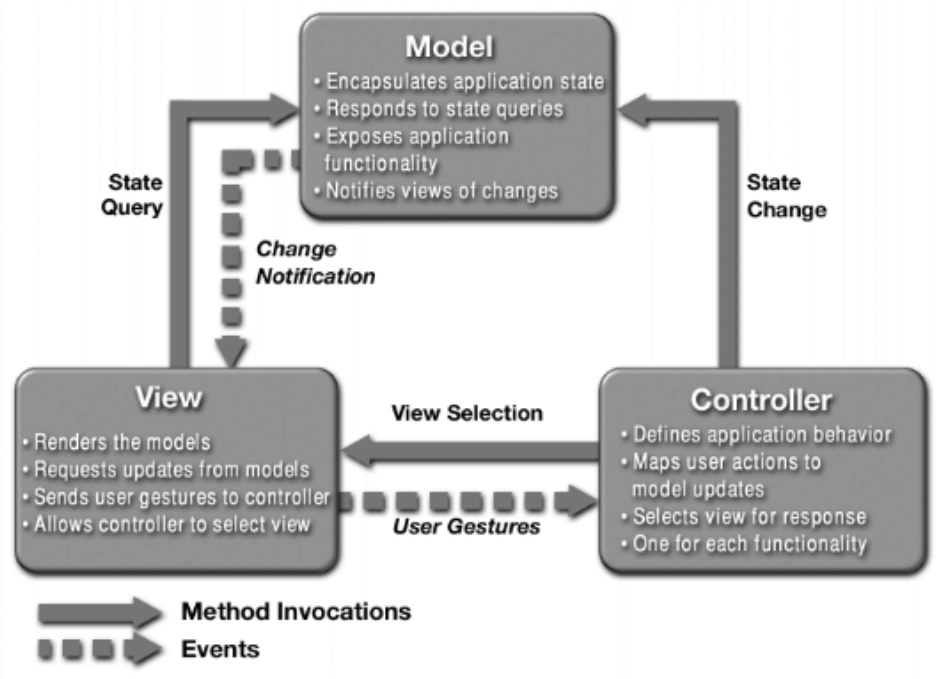



Figura 2.15: Diagrama genérico da Arquitectura MVC (Fonte: [Mic02])

#### 2.2.2.2 PHP

Inicialmente a linguagem de programação PHP/FI começou por ser um conjunto de *scripts* escritos em Perl pelo programador Rasmus Lerdorf em 1995, melhorados relativamente aos que ele utilizava para gerir a sua página *web* pessoal.

Com a necessidade de mais funcionalidades, foi implementada em C uma maior e melhor especificação da linguagem, o que lhe permitiu comunicar com bases de dados, possibilitando a criação de pequenos *websites* dinâmicos. A linguagem foi disponibilizada ao público em Junho de 1995, de forma a acelerar a descoberta de erros e a melhorar o código, sob o nome de PHP *version 2*.

Em 1997 o *parser* da linguagem foi reescrito pelos programadores isrealitas Zeev Suraski e Andi Gutmans, criando-se assim a base para o PHP 3. Desde então, estes dois programadores têm vindo a trabalhar no desenvolvimento da linguagem e do *Zend Engine*[5].

A linguagem PHP, actualmente na versão 5, é orientada a objectos, consistente, poderosa, tem uma síntaxe muito parecida com a da linguagem e é *open-source* Perl[PHP09]. Actualmente é uma das linguagens mais utilizadas para fazer desenvolvimento *web*.

[5]Máquina virtual sobre a qual é interpretada a PHP

#### Symfony

É uma *framework* criada para desenvolver aplicações web em PHP. Esta *framework* nasceu de uma derivação do projecto Mojavi3-DEV, que era uma implementação do modelo MVC, cujo mapeamento objecto-relacional era feito através do Propel[6].

Algumas das funcionalidades que possui foram inspiradas na *framework* Ruby on Rails [Sym10], tais como os *helpers* (utilizados nos *templates*) e o *routing* (utilizado para fazer o mapeamento do *URL* para os respectivos *controllers* e *actions*).

Possui uma vasta comunidade, muita documentação e está em constante desenvolvimento. A sua desvantagem, talvez no início, é que a sua curva de aprendizagem ainda é significativa, quando comparada com outras *frameworks*. Isto também devido ao elevado número de funcionalidades que tem.

#### CakePHP

É uma outra *framework* para desenvolvimento de aplicações *web* em PHP que utiliza a arquitectura MVC. Esta *framework*, tem uma estrutura muito semelhante à da *framework* Ruby on Rails, pelo que talvez seja a mais aconselhada para quem já utilizou Rails. Tem uma documentação muito boa, com uma comunidade com um número de membros significativo e também está em constante desenvolvimento.

A sua curva de aprendizagem é mais curta, dada a sua estrutura ser mais simples e não possuir tantas funcionalidades como a Symfony (não querendo isto dizer que esta seja necessariamente pior do que a anterior).

### 2.2.2.3 Ruby

Ruby é uma linguagem de programação, *open-source*, criada por Yukihiro Matsumoto [Rub10], que tentou juntar nesta linguagem todas as características que mais lhe agradavam em linguagens como Perl, Smalltalk, Eiffel, Ada, e Lisp.

É uma linguagem interpretada onde tudo é visto como sendo um objecto. É muito flexível, pelo que o programador pode redefinir uma classe já existente na linguagem, adicionando-lhe os métodos que pretender. É uma linguagem criada com o intuito de se aproximar um pouco do Inglês, pelo que normalmente se utiliza palavras-chave inglesas. Suporta excepções e o programador não necessita de se preocupar com a necessidade de limpar a memória depois de a utilizar dado ela possuir o seu próprio colector de objectos inacessíveis (*garbage collector*). Facilmente se escrevem extensões em ruby para utilizar em C dada a existência de uma API que facilita a execução de código ruby em C e é multi-plataforma, sendo possível ser utilizada em vários sistemas operativos. [Rub10]

[6] Biblioteca ORM open-source escrita em PHP5

**Ruby On Rails**

É uma *framework* criada em 2003 pelo dinamaquês David Heinemeier Hansson que também segue a arquictetura MVC e foi uma das grandes impulsionadoras da linguagem Ruby[ROR10]. Devido à sua estrutura e às funcionalidades que possui, é vulgarmente utilizada em equipas que utilizam metodologias ágeis de desenvolvimento de software.

Actualmente, existem muitos *websites* desenvolvidos com base nesta framework, nomeadamente [ROR10]:

- BaseCamp[7] - Uma ferramenta online para gestão de projectos;
- Twitter[8] - Uma das redes sociais mais populares, hoje em dia;
- GitHub[9] - Uma ferramenta que facilita a gestão e utilização do sistema de controlo de versões Git (http://git-scm.com/).

Dado que esta *framework* foi desenvolvida na linguagem Ruby, também é multi-plataforma.

#### 2.2.2.4 C#

Esta linguagem foi lançada pela Microsoft e foi projectada para fazer o desenvolvimento de aplicações para serem executadas na *framework* .NET. É uma linguagem com uma síntaxe semelhante à das linguagens JAVA, C e C++ e é orientada a objectos.

Tem como vantagens, o facto de ter um poderoso IDE (Microsoft Visual Studio) e ser de fácil integração com o servidor de bases de dados Microsoft SQL Server e com o IIS. A comunidade de programadores que utilizam esta linguagem também é significativa. Como desvantagens, tem o facto dos *websites* desenvolvidos nesta linguagem apenas poderem ser publicados *online* em servidores com sistemas operativos Windows.

Apesar desta linguagem ser proprietária, actualmente já é possível desenvolver aplicações em C# sem custos, através da suite Express [10] disponibilizada pela Microsoft de forma totalmente gratuita.

### 2.2.3 Client-Side

Nesta sub-secção vai ser feita uma breve análise a linguagens e/ou *frameworks* utilizadas para o desenvolvimento de RIAs. Este tipo de aplicações distingue-se pelo facto de permitir a utilização de vários elementos multimédia (áudio, vídeo, animações, texto) em aplicações *web*, de forma enriquecer a experiência do utilizador, tornando-a muito semelhante à utilização de um ambiente informático local.

[7] http://basecamphq.com/
[8] http://twitter.com/
[9] http://github.com/
[10] http://www.microsoft.com/express/Default.aspx

#### 2.2.3.1 JavaFX

Esta linguagem é relativamente recente pois foi lançada há cerca de 1 ano e meio pela, na altura, Sun Microsystems, actual Oracle. É uma linguagem declarativa relativamente simples de modo a que possa ser utilizada por pessoas que não tenham grandes noções de programação. As aplicações desenvolvidas nesta linguagem são executadas através da JVM, pelo que também são multi-plataforma. Existe ainda a vantagem do programador poder utilizar bibliotecas desenvolvidas em JAVA, em aplicações desenvolvidas em JavaFX. Uma outra vantagem desta linguagem é o facto de ter sido desenhada tendo em mente a sua expansão para utilização num vasto leque de dispositivos electrónicos (dispositivos móveis, TVs, leitores de BluRay). As aplicações desenvolvidas em JavaFX que estejam incorporadas em *websites* podem ser arrastadas para o ambiente de trabalho, continuando a ser executadas mesmo depois do utilizador fechar o navegador *web*.[Sun10]

Por enquanto as desvantagens desta linguagem são:

- Desempenho - Devido ao facto da linguagem ser recente e ainda estar a amadurecer, o seu desempenho não é tão bom como se desejaria.

- Elementos visuais disponíveis - Certos elementos visuais que já se encontram implementados em JAVA (caixas de texto com suporte para várias linhas, caixas de texto para senhas de acesso) ainda não se encontram disponíveis em JavaFX. No entanto, devido a uma comunidade crescente de programadores, já existem bibliotecas que têm vindo a implementar os elementos em falta (JFXtras[11]).

Dado o interesse suscitado pela linguagem, espera-se que estas desvantagens mencionadas, venham a ser ultrapassadas a curto prazo. Está também previsto o lançamento de uma ferramenta para o desenvolvimento visual de aplicações em JavaFX, semelhante à ferramenta da Adobe para o desenvolvimento de aplicações em Flash.

Assim como a linguagem JAVA, esta também é uma linguagem aberta.

#### 2.2.3.2 Microsoft SilverLight

A Microsoft Silverlight é uma *framework* para o desenvolvimento de aplicações *web* lançada pela Microsoft de modo a fazer frente ao Adobe Flash. As interfaces desenhadas em Silverlight são declaradas em XAML e programadas utilizando um subconjunto da *framework* .NET. Actualmente na versão 3.0, permite desenvolver interfaces muito ricas e que melhoram muito o experiência do utilizador. Apesar de incialmente ter sido lançada para aplicações *web*, esta já pode ser executada como se de uma aplicação normal se tratasse, sem necessidade de ser executada dentro de um navegador *web*. Dado que

[11] http://jfxtras.org/

esta framework utiliza também a linguagem de programação C#, também é possível desenvolver aplicações que tirem partido desta de forma totalmente gratuita através da suite Express.

A grande desvantagem desta *framework*, é o facto das suas aplicações apenas poderem ser publicadas em servidores Windows e de apenas poderem ser executadas em sistemas operativos Windows e Mac OSX. Existe um *plugin* (MoonLight) para o Mozilla Firefox que permite a visualização, em sistemas operativos baseados em Linux, de *websites* desenvolvidos em SilverLight, mas que por enquanto ainda não suporta todas as funcionalidades da versão 3.0 e possui alguns problemas de desempenho.

#### 2.2.3.3 Adobe Flash

O Flash, desenvolvido incialmente pela Macromedia, que depois foi comprada pela Adobe, foi uma das primeiras plataformas multimédia, se não a primeira, a ser lançada para permitir o desenvolvimento de *websites* que incorporassem elementos áudio e vídeo e que fossem interactivos. Hoje em dia é largamente utilizada na Internet para a criação de animações, pequenos jogos, leitores multimédia, faixas publicitárias, vídeos e mesmo *websites*.

É suportado na maioria dos sistemas operativos utilizados, através da instalação do *Flash Player*. Apesar da existência do leitor para a maioria dos sistemas operativos existentes, em Linux a performance desta plataforma fica um pouco atrás da performance desta em Windows e Mac OSX[12].

Uma vantagem desta tecnologia é o facto de também estar integrada em dispositivos móveis.

A sua desvantagem é que aplicações que a utilizem apenas podem ser desenvolvidas na ferramenta disponibilizada pela Adobe, dado ser uma plataforma fechada e com um custo considerável.

### 2.2.4 Resumo e Conclusões

Nesta sub-secção foram analisadas algumas das ferramentas disponíveis para serem utilizadas no desenvolvimento do sistema que se pretende. O LabView é uma ferramenta proprietária, que utiliza uma linguagem gráfica para desenhar o fluxo de informação das experiências e para se desenhar o painel de controlo das experiências, tendo como desvantagem o facto de ter um custo elevado. Depois de analisado o LabView, fez-se uma breve introdução às *frameworks* MVC e às suas vantagens.

Relativamente às tecnologias disponíveis para se efectuar o desenvolvimento da aplicação que ficará alojada no servidor, fez-se uma breve análise às linguagens PHP e Ruby

[12] Facto observado pelo autor na utilização diária do sistema operativo Debian e na visualização de *sites* que utilizam a tecnologia Flash

e às *frameworks* MVC disponíveis nelas. Também se analisou a linguagem C#, que é uma linguagem proprietária mas aberta e que tem a desvantagem de apenas poder ser totalmente utilizada em sistemas operativos Windows.

As ferramentas analisadas para o desenvolvimento dos painéis de controlo das experiências foram: JavaFX, Microsoft SilverLight e Flash. JavaFX é uma linguagem aberta que é executada sobre a máquina virtual de Java. Microsoft SilverLight é uma *framework* baseada na plataforma .NET, mas que não é 100% compatível com Linux. O Flash, é uma plataforma utilizada para criar conteúdos multimédia que podem ser disponibilizados online, tendo como vantagens o facto de ser multi-plataforma, mas que em Linux aparenta ter alguns problemas de performance e é uma plataforma proprietária com um custo considerável.

# Capítulo 3

# Escolhas Tecnológicas e Arquitectura do Sistema

Este capítulo começará por fazer uma exposição dos requisitos (funcionais e não funcionais) que o sistema a desenvolver deverá satisfazer. De seguida, serão apresentadas as tecnologias escolhidas a serem utilizadas para o desenvolvimento do sistema, sendo depois apresentada a arquitectura do sistema. Por fim, será apresentada a arquitectura da base de dados utilizada.

## 3.1 Especificação de Requisitos

Qualquer sistema a ser desenvolvido deve satisfazer uma série de requisitos, requisitos esses que irão determinar a forma como o sistema será projectado e o seu modo de funcionamento. Como tal, a fase de levantamento de requisitos é uma das fases cruciais no processo de desenvolvimento de um sistema que, quando mal efectuada ou efectuada de forma deficiente, pode condicionar, de forma irreversível, o sucesso do sistema.

Os requisitos, estes puderam ser divididos em 2 grupos: requisitos não funcionais e requisitos funcionais.

### 3.1.1 Requisitos Não Funcionais

Os requisitos não funcionais dizem respeito às qualidades que o sistema deve possuir. Ou seja, é este tipo de requisitos que determinará se o sistema deverá ser escalável, usável, estar de acordo com os padrões publicados, entre muitas outras características.

Para o sistema a desenvolver nesta dissertação, os requisitos não funcionais especificados foram os seguintes:

- Todo o *software* desenvolvido deverá ser baseado em ferramentas *open-souce* e, ele mesmo, deverá poder ser disponibilizado como *software open-source*;
- A interface *web* a desenvolver deverá ser amigável e estar de acordo com as recomendações do W3C (World Wide Web Consortium);
- Deverá ser garantida uma utilização sem problemas com os navegadores *web* mais comuns utilizados nos 3 sistemas operativos actualmente generalizados na comunidade estudantil (Windows, Linux e Mac OS X);
- Todo o software desenvolvido deverá ser devidamente documentado;
- O sistema de controlo das experiências deverá ser multi-plataforma, de forma a poder ser executado em, praticamente, qualquer sistema operativo;
- O sistema deverá ser minimamente seguro, de modo a evitar que as credenciais dos seus utilizadores possam ser apanhadas durante o percurso efectuado entre cliente e servidor.

### 3.1.2 Requisitos Funcionais

Nos requisitos funcionais são especificados os comportamentos e funcionalidades que o sistema deverá cumprir de forma a ir ao encontro das expectativas do cliente (pessoa ou empresa). De modo a que o sistema a desenvolver vá de encontro às expectativas do cliente, foi feito um levantamento das funcionalidades e comportamentos que ele deve possuir. Dado que ele terá uma arquitectura cliente-servidor, os requisitos foram divididos em duas categorias: requisitos do servidor e requisitos do cliente.

#### Requisitos do servidor

- Autenticar os utilizadores;
- Mediante as permissões do utilizador, limitar as suas acções no sistema (controlo de acessos):
  - CRUD[1] Utilizadores;
  - CRUD Grupos de Utilizadores;
  - CRUD Experiências;
  - CRUD Grupos de Experiências;
  - CD[2] Reservas de condução de experiências;

[1] Por CRUD entende-se *Create, Read, Update, Delete*, ou seja, adicionar, ler, actualizar e apagar
[2] Por CD entende-se *Create, Delete*, ou seja, adicionar e apagar

- Receber os comandos do cliente, enviá-los para a experiência, receber os resultados da experiência e retorná-los para o cliente;

**Requisitos do cliente:**

- Autenticação dos utilizadores, de forma a verificar se estes têm alguma reserva de realização de experiência para o período de utilização pretendido;
- Apresentação dos dados provenientes da experiência, de forma semelhante ao ambiente laboratorial real, através de mostradores (7 segmentos, barra de percentagem, analógico horizontal, díodo electro-luminescente) e gráficos;
- Permitir o controlo da experiência através de botões, *check-boxes*, *sliders*, botões rotativos;
- Permitir a visualização do sinal de vídeo, transmitido a partir de uma ou mais câmaras, para que o utilizador possa observar o que está a acontecer na experiência.

## 3.2 Escolhas Tecnológicas

De forma a cumprir o requisito não funcional de que o sistema desenvolvido deveria utilizar ferramentas *open-source*, todas as tecnologias escolhidas para se efectuar o desenvolvimento serão de utilização livre ou mesmo *open-source*. O facto de se utilizar este tipo de tecnologias tem várias vantagens, entre as quais se destacam:

- Fácil utilização;
- Possibilidade de se fazerem alterações ou adaptações, no caso de ser necessário;
- O facto de não ser necessária a aquisição de licenças, sejam elas de que tipo forem.

### 3.2.1 Linguagem e Framework MVC

Das linguagens analisadas no capítulo anterior, chegou-se à conclusão que a melhor escolha seria a linguagem PHP. As razões por detrás desta escolha são:

- Ser uma linguagem multi-plataforma;
- Ter um bom desempenho;
- Ser uma linguagem de utilização livre;
- Dependendo da configuração do servidor *web*, poder suportar ligações persistentes[3]

[3]Por persistentes, entende-se que a ligação é mantida entre diferentes execuções de *scripts*, isto é, entre vários pedidos.

Em Ruby, a utilização de ligações persistentes não seria tão fácil de implementar como em PHP [Hul09]. Quanto à linguagem C#, o facto dela ser proprietária e dos projectos com ela desenvolvidos só poderem ser alojados em servidores com sistema operativo Windows (IIS)[4], fez com que fosse posta de lado.

Depois de se decidir a linguagem a utilizar, restava escolher a *framework* a utilizar: Symfony ou CakePHP. A Symfony, apesar de ter mais funcionalidades, tem uma curva de aprendizagem mais longa, o que para utilização num projecto com uma prazo relativamente curto, pode fazer toda a diferença. A CakePHP, é uma *framework* com menos funcionalidades, mas de boa qualidade e tem uma curva de aprendizagem mais curta. A sua estrutura é muito semelhante à *framework* Ruby on Rails, o que foi mais uma razão a seu favor dado já se ter alguma experiência na utilização daquela. Com as razões acima mencionadas, optou-se então pela CakePHP.

### 3.2.2 Base de Dados

Actualmente existem 3 principais sistemas de gestão de base de dados (SGBD) *open-source*:

- PostgreSQL [Pos10]
- MySQL [MyS10]
- SQLite [SQL10]

Os dois primeiros, são sistemas mais robustos que necessitam de ser instalados. O terceiro, é um SGBD mais leve e portátil, que não necessita de instalação e a base de dados é apenas um ficheiro. Todos os sistemas são multi-plataforma, pelo que poderão ser utilizados na maioria dos sistemas operativos existentes.

Tanto o PostgreSQL como o MySQL são largamente utilizados para dar apoio a aplicações *web*, pois são sistemas abertos e de muito boa qualidade, pelo que optar por um ou por outro não faria grande diferença. Neste caso, optou-se por utilizar o PostgreSQL, por preferência do autor desta dissertação. No entanto, dado a *framework* CakePHP ter suporte para qualquer um dos SGBDs acima mencionados, a configuração para utilizar qualquer um dos outros pode ser feita com facilidade.

### 3.2.3 Painel de Controlo da Experiência

No que diz respeito às tecnologias para desenvolver/construir os painéis de controlo das experiências, como se viu na secção anterior, estavam disponíveis as seguintes:

[4]Existe a possibilidade de alojar e executar aplicações desenvolvidas na linguagem C# em ambientes Linux e Mac OS X, através do projecto Mono (http://mono-project.com/Main_Page), no entanto a compatibilidade não é assegurada a 100% [Pro10]


Figura 3.1: *Screenshot* dos mostradores

- JavaFX
- Adobe Flash
- Microsoft SilverLight

Existia também outra que se podia utilizar para o efeito desejado (Javascript), mas a única biblioteca com os elementos visuais pretendidos encontrada, foi a Emprise Javascript Charts[5], que tem o inconveniente de ser paga. Ora isto poderia causar problemas, caso o código fonte deste projecto fosse tornado público.

De volta às alternativas analisadas, depois de ter sido feita uma pesquisa nas outras 3 opções e uma apresentação dos elementos visuais encontrados em cada uma das delas, optou-se por escolher JavaFX, devido a ser multi-plataforma e por já existir uma biblioteca com alguns elementos visuais e de controlo (Fig. 3.1) que foram do agrado dos responsáveis pelo projecto.

## 3.3 Arquitectura do Sistema

A arquitectura adoptada para implementação do sistema a desenvolver, como já foi dado a entender no capítulo anterior e mesmo, por características inerentes ao próprio funcionamento de sistemas de controlo remotos, foi uma arquitectura cliente-servidor (Fig. 3.2).

No sistema a desenvolver, o cliente comunica de duas formas distintas com o servidor.

[5] http://www.ejschart.com/

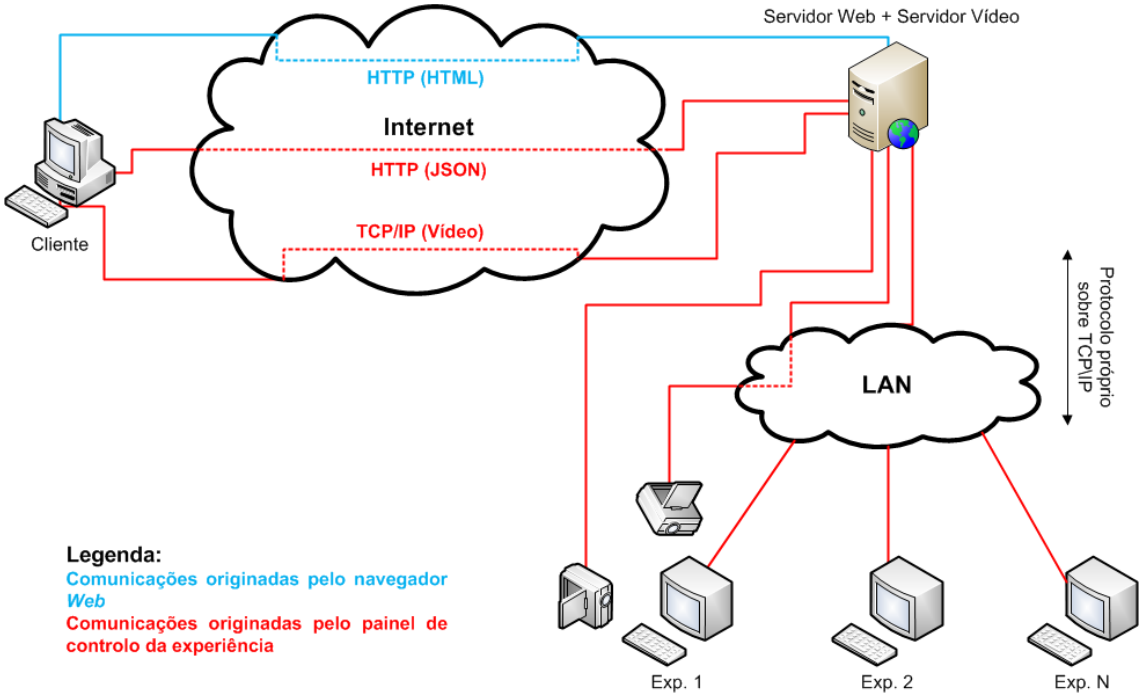



Figura 3.2: Arquitectura do sistema

A primeira forma de comunicação, identificada na Fig. 3.2 pela linha azul, é originada pelo acesso ao servidor *web* através do navegador *web*. Este acesso é efectuado utilizando o protocolo HTTP ou HTTPS e o conteúdo a ser transferido na comunicação é, maioritariamente, texto, mais especificamente, no formato HTML. Através do navegador *web*, o utilizador pode aceder ao *website* onde pode efectuar e apagar reservas, gerir experiências[6], gerir utilizadores[6], etc.

A segunda forma de comunicação é originada pelo painel de controlo da experiência e está identificada na Fig. 3.2 pelas linhas a vermelho. Esta comunicação utiliza, no entanto, duas vias de comunicação distintas:

- Na primeira via de comunicação, o protocolo utilizado é o HTTP ou HTTPS, e os dados transferidos entre o painel de controlo e o servidor *web* é texto no formato JSON [JSO10]. Para esta comunicação são utilizados um conjunto de serviços *web* existentes do lado servidor;

- A segunda via de comunicação, assente sobre a pilha de procolos TCP/IP [Kio09], permite ao utilizador ter uma área no painel de controlo da experiência onde pode observar o que realmente se está a passar do lado da experiência, através de uma transmissão contínua de imagens capturadas por cada câmara que filma a experiência;

[6]Dependendo das permissões do utilizador

Na imagem pode-se ainda observar que as linhas de comunicação entre o servidor e as experiências estão a vermelho. Isto deve-se ao facto da comunicação entre o cliente e as experiências ser indirecta e o servidor só comunicar com as experiências quando receber pedidos provenientes dos painéis de controlo das mesmas.

A comunicação existente entre o servidor e as experiências é efectuada sobre a pilha de protocolos TCP/IP e utiliza um protocolo especificamente desenvolvido para este projecto, baseado no princípio pergunta-resposta, ou seja, sempre que o servidor enviar um comando deve sempre esperar uma resposta.

Uma alternativa a toda esta arquitectura, seria estabelecer uma via directa de comunicação entre o cliente e a experiência; no entanto, isso poderia complicar o dispositivo de segurança da rede interna da instituição, dado que seria necessário fazer o reencaminhamento de portos na *firewall* para cada experiência disponível.

## 3.4 Arquitectura da Base de Dados

Dado ser necessário guardar a informação relativa às experiências, aos utilizadores, aos grupos de utilizadores, aos grupos de experiências e às reservas, e assegurar de que a informação guardada mantém a sua integridade foi necessário desenhar a base de dados com a arquitectura visível na Fig. 3.3. Como é possível observar na figura, teremos as seguintes associações:

- 1 utilizador (tabela *users*) pode ter várias reservas (tabela *schedules*) e 1 reserva apenas pode estar associada a 1 utilizador;
- 1 experiência (tabela *experiments*) pode ter várias reservas (tabela *schedules*) e 1 reserva apenas pode estar associada a 1 experiência;
- 1 grupo de utilizadores (tabela *usergroups*) pode ter vários utilizadores e 1 utilizador apenas pode pertencer a 1 grupo de utilizadores;
- 1 grupo de experiências (tabela *experimentgrps*) pode ter várias experiências e 1 experiência apenas pode pertencer a 1 grupo de experiências;
- 1 grupo de utilizadores pode estar associado a vários grupos de experiências e 1 grupo de experiências pode estar associado a vários grupos de utilizadores. Devido ao facto de esta associação ser de muitos para muitos, é necessário criar uma tabela auxiliar (*experimentgrps_usergroups*).

Convém apenas referir que, as associações acima mencionadas permitiam a um utilizador fazer várias reservas para diferentes experiências, todas a começar ao mesmo tempo, e o mesmo se aplicava às experiências. As condições de unicidade existentes na tabela

*schedules* servem para evitar que essa situação aconteça: garantem que um utilizador apenas tem uma reserva para cada *starttime* e que uma experiência apenas tem uma reserva para cada *starttime*. Todas as outras condições de unicidade são simples e de fácil percepção pelo que não será aqui feita uma análise extensiva de cada uma delas, mencionando-se apenas que foram implementadas de modo a garantir que determinados atributos fossem únicos, dado permitirem a identificação de cada objecto.

A tabela *configs* é apenas uma tabela auxiliar para guardar algumas configurações que assim poderão ser alteradas a partir do *website*, não sendo necessário efectuar alterações em ficheiros de configuração no servidor.

## 3.5 Resumo e Conclusões

Inicialmente, começou-se por fazer uma especificação dos requisitos que o sistema a desenvolver deve satisfazer, dividindo-os em requisitos funcionais e não funcionais. De seguida, explicou-se as razões por detrás da escolha da linguagens de programação PHP e JavaFX (servidor e cliente, respectivamente), da *framework* CakePHP e do sistema de gestão de bases de dados PostgreSQL. Depois, fez-se uma exposição da arquitectura cliente-servidor que o sistema iria utilizar e do modo de comunicação existente entre os seus componentes. Para finalizar, foi feita uma apresentação da arquitectura da base de dados a utilizar.


Figura 3.3: Arquitectura da base de dados

# Capítulo 4

# Implementação

Neste capítulo será feita a descrição da forma como o servidor *web*deverá ser configurado de modo a ser possível a utilização das ligações persistentes. De seguida, será efectuada uma descrição da aplicação que ficará alojada no servidor *web* e das suas funcionalidades. Serão também apresentados os mostradores desenvolvidos e o sistema de sessões que vai ser utilizado nos painéis de controlo das experiências.

Por fim, serão apresentados os resultados de testes efectuados para calcular o tempo médio de duração de um pedido, seguidos de uma breve conclusão.

## 4.1 Servidor *Web*

### 4.1.1 Configuração

A razão pela qual se pretendia utilizar uma linguagem onde as ligações pudessem ser mantidas entre diferentes execuções de *scripts* era para reduzir ligeiramente o tempo de comunicação entre o servidor *web* e a experiência. Desta forma, o *handshake* inicial existente aquando do estabelecimento de uma ligação utilizando o protocolo TCP/IP só seria efectuado uma única vez (no primeiro pedido), ao invés de ter de ser efectuado sempre que um pedido fosse efectuado.

Ora, as ligações persistentes são possíveis em PHP mas dependem da forma como é configurado o servidor Apache[1] e o interpretador do PHP. Normalmente, numa distribuição Linux, quando se instala o servidor Apache juntamente com o PHP, este último é instalado como um módulo. Utilizando este método, as ligações persistentes poderão ou não funcionar. O Apache, por omissão, lança 5 processos-filho que vão servir para receber e processar os pedidos HTTP. Dado que o PHP, nesta configuração, será executado como um módulo do Apache, cada processo-filho do Apache terá o seu próprio interpretador de

[1] http://httpd.apache.org/

PHP e a sua própria memória. Neste caso, a ligação só será persistente se se tiver a sorte do pedido HTTP ser processado pelo mesmo processo-filho que iniciou a ligação. Caso o processo não seja o mesmo, este tentará criar uma nova ligação que irá falhar, devido ao facto de naquele já existir uma ligação aberta e da experiência só permitir uma ligação aberta em cada instante de tempo.

De forma a ultrapassar a limitação acima mencionada, o Apache irá ser configurado para funcionar com o módulo *MPM Worker* e com o módulo *FastCGI*. Ao mesmo tempo que se configurará o Apache, também se irá proceder à configuração do PHP para ser executado como CGI. Com esta configuração [Sil09], o servidor Apache apenas processará os pedidos de conteúdos estáticos (.html, .css, .js) e quando receber um pedido para um conteúdo dinâmico (.php), passará o ficheiro pedido como argumento ao interpretador indicado, neste caso o do PHP. Este último irá interpretar o ficheiro e retornar o resultado da execução ao Apache, que por sua vez o irá retornar para o cliente. Com esta configuração, já será possível a correcta utilização das ligações persistentes dado que o apontador para a ligação mantém-se na memória do processo PHP que irá interpretar o ficheiro passado pelo Apache.

### 4.1.2 Aplicação *Web*

Esta aplicação é a que ficará alojada no servidor *web* e fará a ligação entre o cliente e a experiência. É através desta aplicação que também será efectuada a gestão de utilizadores, de experiências, dos grupos de utilizadores, dos grupos de experiências e das reservas. Nas secções seguintes far-se-á uma descrição do que foi implementado nesta aplicação e dar-se-á uma ideia das suas funcionalidades principais.

#### 4.1.2.1 Gestão de Utilizadores

Na página dos utilizadores, é possível visualizar a lista de todos os utilizadores registados no sistema e ver as informações relativas a cada um deles, nomeadamente, o nome, o *e-mail*, o tipo (aluno, docente ou administrador), o nome de identificação perante o sistema (*login*) e o grupo a que pertence. Relativamente à criação de novos utilizadores, de momento, apenas um administrador os poderá criar.

Um utilizador sem permissões de administração pode alterar todos os seus dados pessoais, excepto as suas permissões e o grupo a que pertence.

#### 4.1.2.2 Gestão de experiências

A gestão de experiências é um dos elementos principais desta aplicação, pelo que sem ela seria impossível ao sistema comunicar com a experiência física.

Aquando da criação de uma experiência, dever-se-ão preencher todos os campos do formulário existentes na página (Fig. 4.1).


Figura 4.1: Página para adicionar uma experiência

O valor dos campos relativos ao endereço IP e o porto são os que vão ser utilizados para, no web service, ser estabelecida a ligação com a experiência. Relativamente ao campo ficheiro, é aqui que deverá ser submetido um ficheiro do tipo ZIP [Wik10b], cujo conteúdo serão os ficheiros contidos na pasta *dist* existente na pasta de um projecto Netbeans, onde foi criado o painel de controlo para a experiência que se está a instalar.

Na sequência de submissão do formulário, o ficheiro enviado será validado e, se a validação tiver sucesso, os ficheiros nele contidos serão extraídos para um caminho construído a partir do nome do grupo ao qual pertence a experiência e do nome da experiência. Desta forma, todos os painéis de controlo ficam localizados numa pasta de acesso público de forma a facilitar o seu lançamento, devido às dependências que estes possam ter e ao facto de se tirar proveito do *Java Network Launch Protocol* (JNLP) [Mic10b]. Todas estas pastas serão colocadas, por omissão, na pasta *'app/webroot/interfaces/'*.

Aquando da criação da experiência, o endereço para o lançamento do painel de controlo da experiência também é construído de modo a apontar para o ficheiro JNLP correcto. Por exemplo, se fosse criada uma experiência com o nome 'Cálculo da aceleração gravítica' pertencente ao grupo de 'Mecânica', o seu painel de controlo iria ficar colocado em:

*'app/webroot/interfaces/mecanica/calculo_da_aceleracao_gravitica'*

Mesmo depois da experiência criada, se um utilizador for alterar o nome ou o grupo da experiência, a localização do painel de controlo também é alterada de forma a manter o endereço para fazer o lançamento do painel de controlo coerente.


Figura 4.2: Estrutura de ficheiros de um projecto NetBeans

No caso de se apagar uma experiência, os ficheiros relativos ao painel de controlo dessa experiência também serão apagados.

É conveniente dizer que apenas um utilizador com permissões de administrador ou de professor poderá criar, editar ou remover experiências. No caso de um utilizador querer editar uma experiência, ele não é obrigado a enviar o ficheiro novamente, pois serão utilizados os ficheiros existentes no servidor, relativos à experiência a editar.

**O Ficheiro ZIP**

O ficheiro ZIP a enviar para o servidor deverá ter uma estrutura relativamente fixa, de forma a que o protocolo JNLP consiga detectar e descarregar as bibliotecas utilizadas pelo painel de controlo da experiência.

Deste modo, far-se-á de seguida uma breve explicação de como criar o ficheiro compactado a enviar, partindo-se do princípio que se utiliza o NetBeans como ambiente de desenvolvimento.

Aquando da criação de um projecto neste ambiente de desenvolvimento, é gerada uma estrutura de ficheiros semelhante à da figura 4.2. No entanto, apenas quando for feita a compilação do projecto criado (*Build*) é que a pasta *dist*, mencionada na secção anterior, será criada.

Depois de compilado o projecto, deverá abrir-se a pasta *dist* e adicionar, principalmente, os ficheiros '[nome_projecto].jar' e [nome_projecto].jnlp e a pasta 'libs', caso exista, ao ficheiro ZIP. Depois de criado o ficheiro, segundo esta estrutura, será possível submetê-lo.

#### 4.1.2.3 Gestão de Grupos de Experiências e Grupos de Utilizadores

Devido ao facto das funcionalidades implementadas nos grupos de experiências e nos grupos de utilizadores serem muito semelhantes, optou-se por juntar a descrição dos dois numa só secção.

O objectivo da criação de grupos de experiências e grupos de utilizadores, será, no caso de se pretender, limitar o acesso de um grupo de utilizadores a apenas alguns grupos de experiências.

Actualmente, é possível a criação, edição e remoção de grupos de utilizadores e experiências por parte de utilizadores com permissões de 'professor' ou de 'administrador'.

No que diz respeito à limitação do acesso dos grupos de utilizadores aos grupos de experiências a que estão associados, esta funcionalidade ainda não se encontra implementada.

#### 4.1.2.4 Gestão de Reservas de Condução de Experiências

A gestão de reservas é efectuada através da página de cada experiência, onde está disponível a opção para criar uma reserva (Fig. 4.3).


Figura 4.3: Opção para visualizar o horário das reservas

Ao clicar sobre essa opção (*link*), é efectuado um pedido HTTP assíncrono (AJAX [w3s10]) e é carregado o horário semanal das reservas para a experiência actual (Fig. 4.4). Nesse mapa, o utilizador poderá visualizar todos os intervalos de tempo da semana escolhida. Esses intervalos de tempo poderão assumir qualquer um dos estados visíveis na fig. 4.5.

Um intervalo de tempo assume o estado livre (Fig. 4.5(a)), quando o seu tempo é futuro e quando não existem reservas para esse mesmo intervalo. Assumirá o estado ocupado (Fig. 4.5(b)) já existir uma reserva ou se o utilizador fizer uma reserva nesse mesmo intervalo. O estado desactivado (Fig. 4.5(c)) será assumido quando tempo do intervalo já tiver passado.

Um intervalo de tempo indicado como ocupado, apenas pode ser libertado pelo utilizador que o reservou ou por um administrador. No caso do intervalo de tempo já ter passado, este será mostrado como inactivo.

Ao utilizador, também é permitido fazer uma navegação entre semanas, de modo a poder escolher a data que mais lhe convém para fazer a sua reserva. Esta navegação é possível clicando nas setas visíveis na fig. 4.4, em que se clicar na seta da direita avançará uma semana e se clicar na seta da esquerda retrocederá uma semana.

Convém ainda referir que todas estas acções são executadas assíncronamente, utilizando-se para o efeito pedidos AJAX e que o horário que é apresentado ao utilizador é desenhado recorrendo apenas à linguagem de estilo CSS [W3C10].

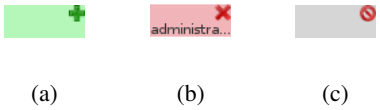

Figura 4.4: Horário das reservas

### 4.1.3 Web-Services

Os serviços *web* são utilizados para fazerem a ligação entre o cliente e a experiência. Actualmente encontram-se implementados dois serviços: *Auth* e *Act*.

O serviço *Auth* é apenas utilizado para fazer a autenticação do utilizador. Quando este serviço é invocado, as credenciais correctas são enviadas pelo cliente no formato utilizado na autenticação básica [Fou10] e validadas. No caso de serem as correctas, o servidor retornará um identificador da sessão no cabeçalho HTTP enviado para o cliente.

O serviço *Act* é utilizado para interagir com a experiência. O cliente quando pretender executar qualquer comando, fará um pedido para este serviço em que deverão ser fornecidos os seguintes parâmetros:




(a) (b) (c)

Figura 4.5: Estados de um intervalo de tempo: (a) Livre; (b) Ocupado; (c) Passado.

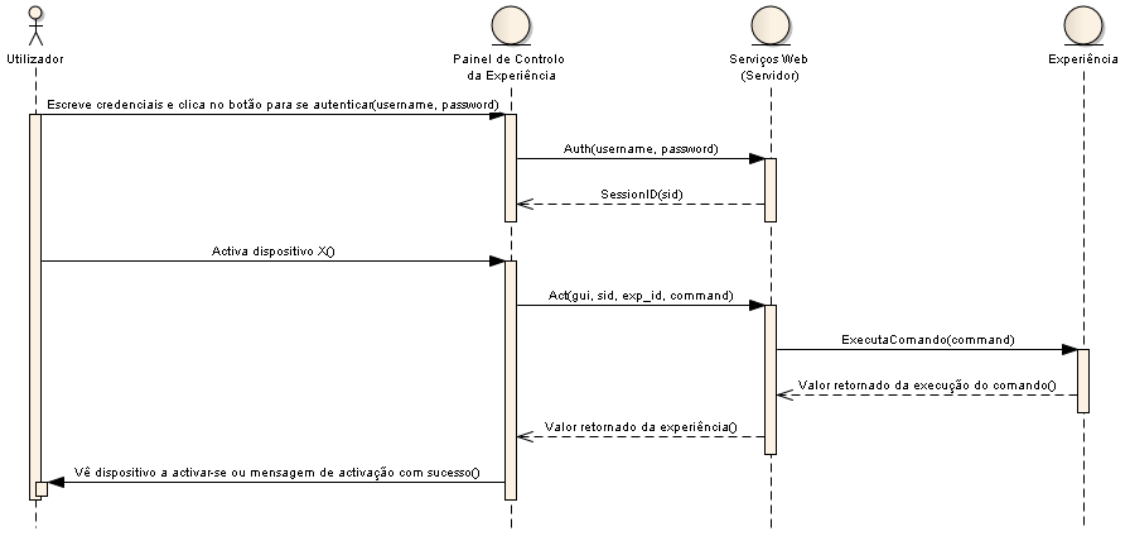



Figura 4.6: Sequência de mensagens com autenticação e envio de um comando

- GUI - Este parâmetro deverá ser definido, servindo apenas para identificar se se o pedido é efectuado por um painel de controlo;
- SID - Este parâmetro é o identificador devolvido pelo serviço *Auth*. É através dele que se evita enviar em cada pedido as credenciais do utilizador;
- ID - Serve para identificar a experiência para a qual se destina o comando a enviar;

Depois de recebido o pedido e verificada a sua validade, ele é enviado para a experiência e a resposta à execução do comando é enviada para o cliente. Na comunicação servidor → cliente, os dados enviados estão sob o formato JSON. Na Fig. 4.6 é possível ver a sequência de autenticação e de envio de um comando.

### 4.1.4 Comunicação com o Dispositivo de Controlo da Experiência

A comunicação com o *hardware* é efectuada através de uma ligação TCP/IP. O protocolo utilizado é um protocolo desenvolvido para o efeito e que se baseia no princípio "pergunta-resposta". Ou seja, a cada comando enviado para o *hardware*, o servidor terá sempre de receber uma resposta, reencaminhando-a depois para o cliente.

Na Fig. 4.7 é possível observar um exemplo de como a comunicação é efectuada.

Os comandos especificados nesta primeira versão do protocolo são:

- START - Dá início à experiência;
- STOP - Pára a experiência e reinicia todos os parâmetros;
- PAUSE - Interrompe momentaneamente a experiência;

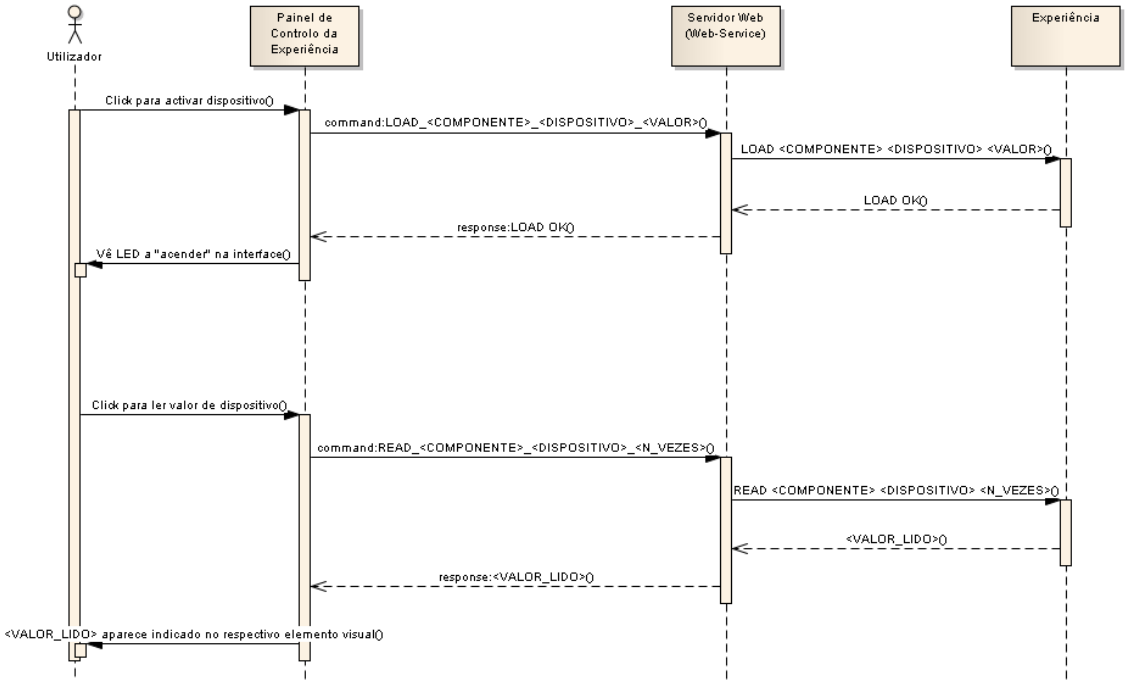



Figura 4.7: Exemplo de sequência de mensagens entre o cliente e o *hardware*

- PLAY - Continua a experiência;
- CONNECT - Utilizado para estabelecer a ligação com a experiência;
- DISCONNECT - Serve para informar a experiência que o utilizador terminou a sessão e que a ligação entre o servidor e a experiência pode ser terminada;
- LOAD (COMP) (IND) (VAL) - Serve para carregar um determinado valor (VAL), no dispositivo com índice (IND) situado no componente (COMP). Devolve como resposta 'LOAD OK' no caso de carregar o valor com sucesso;
- READ (COMP) (IND) (N) - Utilizado para ler N vezes o dispositivo com indice (IND) no componente (COMP). Devolve N valores lidos separados por espaço.

No entanto, actualmente ainda só se encontram implementados e em utilização o 'DISCONNECT', o 'LOAD' e o 'READ'. É possível encadear vários comandos 'LOAD' e 'READ' seguidos, bastando para isso, que estes estejam separados por uma vírgula.

O protocolo também suporta o envio de múltiplos comandos. Neste caso, os comandos deverão ser enviados no seguinte formato:

> READ (COMP) (IND) (N), LOAD (COMP) (IND) (N), LOAD (COMP) (IND) (N)

A resposta ao comando será: X X X X X X X, LOAD OK, LOAD OK.

## 4.2 Painel de Controlo de Experiências

Os painéis de controlo das experiências serão executados do lado do cliente e permitirão ao utilizador interagir com a experiência, através do envio de comandos e os seus respectivos parâmetros, da recepção das respectivas respostas e da "tradução" das respostas em informação visual.

### 4.2.1 Elementos Visuais

De modo a permitir o desenvolvimento de painéis de controlo de experiências semelhantes aos painéis de controlo dos instrumentos reais, foi necessário desenvolver ou pesquisar bibliotecas já existentes em JavaFX que permitissem a utilização de mostradores analógicos semelhantes aos existentes nos instrumentos reais.

Alguns dos mostradores necessários, já se econtravam desenvolvidos numa biblioteca (MemeFX) disponibilizada online[2] por Mauricio Aguilar O. Esta biblioteca já possui uma vasta gama de mostradores analógicos facilmente configuráveis e apelativos. De modo a que a biblioteca pudesse ser utilizada em pleno, foram feitas algumas alterações no código-fonte para a tornar compatível com a versão 1.2 da linguagem JavaFX.

No entanto, a biblioteca utilizada não possuía todos os mostradores desejados. Para dar solução a este problema, foi desenvolvido mais um conjunto de mostradores.

#### Mostrador de 7 Segmentos

O mostrador de 7 segmentos um dos primeiros mostradores a ser desenvolvido. Este mostrador pretende simular os mostradores de 7 segmentos reais e é totalmente configurável. Ou seja, quando utilizado é possível escolher: (entre parêntesis, estão os valores pré-definidos)

- O número de dígitos que terá (9);
- A cor do fundo do mostrador (preto);
- A cor dos segmentos (vermelho);
- O seu tamanho, quer seja em altura, em comprimento ou nos dois.

No que respeita ao mostrador, na Fig. 4.8 é possível ver dois exemplos.

#### Barra Horizontal

Outro dos mostradores a desenvolver foi uma barra de percentagem, ou seja, independentemente do valor máximo, o valor indicado pela barra é sempre uma percentagem, logo a escala será sempre de 0 a 100.

[2] http://code.google.com/p/memefx/

(a) (b)

Figura 4.8: Exemplos do mostrador de 7 segmentos

Esta barra também é altamente configurável, sendo possível ao programador que a utilizar escolher a cor de fundo, a cor da barra, o tamanho do mostrador (altura e largura), se devem ser desenhadas as marcações, a cor das marcações e o intervalo de unidades entre elas e ainda a velocidade de animação da barra, isto é, o tempo que esta demora a chegar ao valor indicado.

Na fig. 4.9 é possível observar um exemplo deste artefacto.

#### Mostrador Horizontal

Apesar deste mostrador também ser horizontal, tem algumas diferenças significativas. Neste mostrador, os valores apresentados já não são entre 0 e 100, mas sim entre os valores que o programador escolher. Outra diferença é o facto de não existir uma barra, mas sim uma agulha que vai deslizando pelo mostrador, na horizontal, entre os valores máximo e mínimo.

Este mostrador também é totalmente configurável, sendo possível alterar a cor da "agulha", a cor de fundo, dos valores apresentados no mostrador, das suas marcações, entre outras. Neste mostrador, para ajudar o utilizador a identificar melhor o valor nele indicado, além da posição da agulha existe também um pequeno espaço a meio onde é apresentado o valor.

Um exemplo deste tipo de mostrador é visível na fig. 4.10.


Figura 4.9: Exemplo da barra horizontal


Figura 4.10: Exemplo do mostrador horizontal

**LED (Díodo Electro-Luminescente)**

Por último, foi necessário o desenvolvimento de um elemento que fosse capaz de simular um LED. De todos os elementos visuais que foram necessários desenvolver, este foi sem dúvida o mais simples. No que diz respeito a configurações possíveis, o programador pode escolher a cor quando o dispositivo se encontrar "desligado" e "ligado" e o seu tamanho, através do raio.

É também possível saber o estado actual e alterar esse mesmo estado através de duas funções definidas na classe. Na fig. 4.11 é possível ver um exemplo de 2 LEDs nos dois estados permitidos.

(a) (b)

Figura 4.11: 2 LEDs nos dois estados possíveis

**Outros Tipos de Controlos e Elementos Visuais**

No respeita ao desenvolvimento de outros tipos de controlos e/ou elementos visuais passíveis de serem utilizados, não houve necessidade de os desenvolver pois a linguagem JavaFX já possui vários tipos de controlos desde botões, *checkboxes*, botões de alternância de estado, gráficos, caixas de texto e muitos outros que poderão ser utéis para o desenvolvimento dos painéis de controlo.

### 4.2.2 Sistema de Sessões

Uma das dificuldades que se encontrou durante o desenvolvimento dos painéis de controlo das experiências foi o facto de, ao implementar a autenticação e validação dos utilizadores nos serviços web utilizados no servidor, existir a obrigação de enviar as credenciais do utilizador em todos os pedidos efectuados. Isto ainda pode ser aceitável aquando

a utilização de uma ligação cifrada, dado que o canal de comunicação entre o cliente e o servidor se encontra protegido, mas aquando a utilização de uma ligação não cifrada, isto representa um risco muito elevado dado que as credenciais são enviadas em texto normal para o servidor, sendo extremamente fácil, para alguém que esteja a capturar pacotes na rede, ter acesso às credenciais do cliente.

De modo a reduzir um pouco o risco deste problema, optou-se por criar um sistema de sessões muito semelhante ao, normalmente, utilizado nos *websites*, com base em *cookies* [Wik10a].

Quando o utilizador abre o painel de controlo, deverá inicialmente fazer a sua autenticação. Neste pedido inicial, as credenciais são enviadas e, no caso de serem válidas, o servidor enviará um elemento extra no cabeçalho da resposta HTTP, que se pode ver na fig. 4.12 na área sombreada.

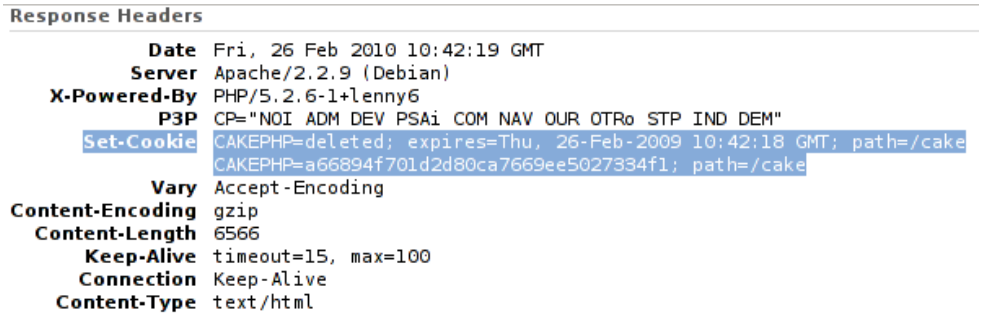



Figura 4.12: Exemplo de um cabeçalho HTTP

O que se optou por fazer então foi, em cada resposta enviada por um serviço web, procura-se no cabeçalho pelo elemento 'Set-Cookie' e guarda-se numa variável a sequência de caracteres que aparece depois de 'CAKEPHP=', dado que é este o identificador da sessão. No pedido seguinte, envia-se o identificador da sessão que é validado do lado do servidor.

Desta forma, reduz-se as probabilidades das credenciais serem apanhadas durante o percurso até ao servidor. Existirá sempre o perigo de captura e utilização indevida dos *cookies*, no entanto a *framework* incorpora alguns mecanismos de prevenção (mudanças de IP, *user-agent* e outros parâmetro invalidam a sessão). Apesar destes mecanismos, poderá haver sempre um pequeno perigo que pode ser resolvido com a utilização de ligações cifradas (HTTPS) em toda a sessão.

### 4.2.3 Outras Bibliotecas

A linguagem JavaFX está a evoluir, pelo que existem algumas funcionalidades que ainda não se encontram implementadas. Por isso, para o desenvolvimento dos painéis de

controlo, recorreu-se a uma biblioteca suplementar, também ela *open-source*, JFXtras[3]. Esta biblioteca, desenvolvida e mantida por uma comunidade activa de programadores, pode quase ser considerada como uma extensão e, em certos casos, melhoria às funcionalidades actuais do JavaFX. Neste projecto, uma das funcionalidades utilizadas desta biblioteca foi a possibilidade de se ter uma *thread* a ser executada em *background* que vai fazendo pedidos ao servidor com um período constante que pode ser escolhido pelo utilizador. Desta forma, é possível manter o painel de controlo actualizado.

## 4.3 Resultados e Testes

No que respeita aos testes efectuados à plataforma, para além dos testes de utilização que iam sendo efectuados à medida que se iam implementando novas funcionalidades, foram também efectuados uns testes para medir o tempo médio de um pedido efectuado a um serviço *web*.

Nestes testes, o tempo começava a contar a partir do momento em que era iniciado o pedido e apenas parava quando fosse recebida a resposta. Os testes também foram efectuados a partir de 2 ambientes diferentes. No primeiro ambiente os pedidos estavam a ser gerados na máquina onde estava o servidor *web* enquanto que no segundo ambiente os pedidos estavam a ser gerados numa máquina que não o servidor *web* mas situada na rede local. Relativamente ao que os pedidos faziam, apenas activavam ou desactivavam um dispositivo consoante o seu estado. Os resultados obtidos podem ser observados na Tabela 4.1.

Para se ter uma forma de comparar os resultados dos testes acima efectuados utilizando o *software* da plataforma desenvolvida com um sistema que não utilizasse a plataforma *web*, efectuaram-se também testes em que os pedidos eram efectuados através de uma ligação directa à experiência, ou seja, o PC de onde iam ser feitos os pedidos ligava a um *switch* que por sua vez, ligava directamente à experiência. Nestes novos testes, os pedidos eram idênticos aos anteriores: activavam ou desactivavam um dispositivo. Efectuando assim, dois testes com 500 pedidos cada um, os tempos médios foram de 231 e 232 ms, respectivamente, o que permitiu concluir que, em média, a diferença entre uma ligação directa e através da arquitectura adoptada é de 130 ms. Isto deve-se ao facto de, na ligação através desta arquitectura, existirem mais algumas camadas de *software* e troca de informações entre diversos processos. A ser demasiado grande este tempo, terá de se procurar meios que permitam baixá-lo.

No anexo A, é possível ver mais ao pormenor os tempos medidos de cada um dos pedidos efectuados e, a forma como foram gerados e calculado o tempo médio em ambos os tipos de testes.

[3] http://jfxtras.org/

Tabela 4.1: Tempos médios dos pedidos gerados: na mesma máquina onde estava instalado o servidor *web*(A); numa máquina diferente do servidor *web*(B); numa máquina ligada directamente à experiência por uma ligação TCP/IP(C). O servidor *web*, a outra máquina e a experiência situam-se na mesma rede local.

| **Máq. Origem** | **Tipo de Ligação** | **Bateria de Teste** | **Nº de pedidos** | **Tempo médio (ms)** |
|---|---|---|---|---|
| Mesma que o servidor *web* (A) | Indirecta | 1 | 101 | 382 |
| | | 2 | 100 | 335 |
| | | 3 | 100 | 373 |
| | | 4 | 100 | 339 |
| | | | **Média:** | $\approx 357$ |
| Distinta do servidor *web* (B) | Indirecta | 1 | 101 | 352 |
| | | 2 | 101 | 383 |
| | | 3 | 101 | 362 |
| | | 4 | 101 | 339 |
| | | | **Média:** | 359 |
| Distinta do servidor *web* (C) | Directa | 1 | 500 | 231 |
| | | 2 | 500 | 232 |
| | | | **Média:** | $\approx 232$ |

Deve, no entanto, ser referido que os testes mediados pelo servidor *web* foram efectuados sem qualquer optimização, ou seja, apesar de existirem módulos em PHP para que seja criada uma memória *cache* e da própria *framework* utilizada também suportar a funcionalidade de guardar em *cache* os dados consultados, estes recursos não foram utilizados.

Foi também construído um painel de controlo muito básico, visível na Figura 4.13, de forma a ser utilizado no teste da plataforma desenvolvida. Neste painel de controlo, o utilizador tem de efectuar a sua autenticação e caso ela seja válida, é iniciada uma *thread* em *background* que trata de actualizar automaticamente, num período definido pelo utilizador, 1 dos mostradores de 7 segmentos e 4 mostradores analógicos. Nesta experiência não é possível actualizar todos os mostradores visíveis de uma só vez devido a limitações no protocolo utilizado no *hardware* para comunicação com os dispositivos físicos, aqui simples led's.

## 4.4 Resumo e Conclusões

Neste capítulo, foi efectuada uma descrição das funcionalidades que houve necessidade de implementar em cada um dos componentes do sistema, assim como em alguns casos, o seu modo de funcionamento.

Foram também apresentados resultados de testes efectuadas ao tempo de execução de um pedido, ou seja, desde o momento em que ele é iniciado pelo cliente até este último

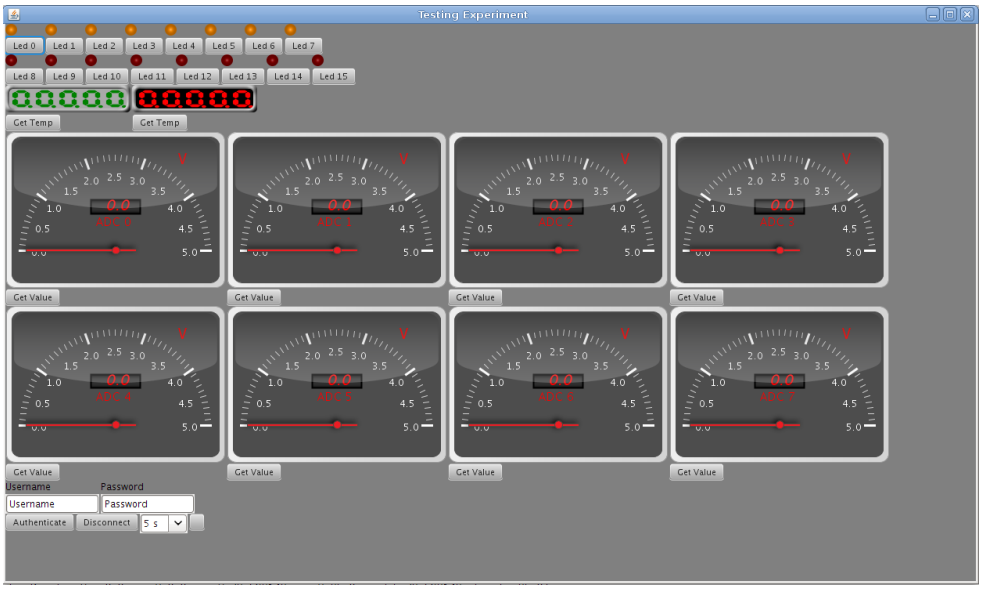



Figura 4.13: Protótipo básico a retratar a experiência utilizada durante o desenvolvimento da plataforma

receber a resposta correspondente. Com os testes, foi possível observar que os testes mediados pelo servidor *web*, em média, têm uma duração maior em cerca de 130 ms.

# Capítulo 5

# Conclusões e Trabalho Futuro

Com a elaboração desta dissertação, pretendeu-se provar que era possível desenvolver uma plataforma de suporte a laboratórios remotos utilizando apenas tecnologias abertas.

Foi possível observar que existem vários artigos publicados sobre o desenvolvimento e implementação de palataformas de suporte aos laboratórios remotos a nível mundial, mas que, de todos os que foram analisados, apenas o código-fonte de um (*The iLab Project* - 2.1.3) é disponibilizado, para quem quiser utilizar a plataforma.

Foi também efectuada uma análise às tecnologias existentes que poderiam ser utilizadas, chegando-se à conclusão de que a utilização das linguagens PHP e JavaFX seria a melhor escolha, devido à primeira permitir ligações persistentes e de já existir uma biblioteca de mostradores desenvolvida na segunda.

No que diz respeito à arquitectura implementada (utilizando o protocolo HTTP para fazer a comunicação entre o servidor e o cliente) verifica-se que, em média, é mais lenta em cerca de 130 ms face a uma ligação directa à experiência. Isto deve-se ao facto de nesta arquitectura existir uma maior número de processos e camadas onde a informação é processada (servidor *web*, processo PHP, base de dados).

Relativamente aos mostradores desenvolvidos, estes encontram-se completamente funcionais e são facilmente configuráveis. O sistema de sessões a ser utilizado pelos painéis de controlo das experiências encontra-se também funcional.

Para finalizar, pode-se concluir que o desenvolvimento desta plataforma foi positivo e bem sucedido pois permitiu à equipa adquirir conhecimentos sobre este tipo de plataformas e as suas necessidades e demonstrar o conceito através de um protótipo funcional.

## 5.1 Trabalho Futuro

Como a plataforma desenvolvida não é perfeita, aqui ficam alguns pontos a melhorar no sistema:

- Apesar da comunicação entre o cliente e a experiência funcionar correctamente utilizando o protocolo HTTP e serviços *web*, deverão ser efectuados testes utilizando um outro tipo de intermediário de forma a verificar-se se é possível baixar um pouco o tempo médio de um pedido efectuado;
- Neste momento, não são utilizadas ligações cifradas pelo painel de controlo da experiência, pelo que seria uma boa medida a implementar de forma a melhorar a segurança de utilização da plataforma. O sistema de sessões implementado também deve ser melhorado de forma a ser mais robusto. Claro que isto deverá ser pesado face ao provável aumento da latência nas comunicações protocolares;
- No que respeita a aplicação *web*, a sua interface gráfica deverá ser melhorada de forma permitir uma utilização ainda mais fácil e intuitiva;
- A utilização das credenciais do SIFEUP para autenticação no sistema desenvolvido, através do LDAP poderá ser uma mais valia dado poupar aos utilizadores o facto de terem de memorizar mais um nome de utilizador e uma senha de acesso;
- Analisar a integração deste sistema com o Moodle, utilizado no âmbito de múltiplas disciplinas dos cursos da FEUP;
- Melhorar o sistema de sessões desenvolvido para os painéis de controlo de forma a torná-lo mais robusto.

# Referências


[BCC09] Aldo Balestrino, Andrea Caiti e Emanuele Crisostomi. From remote experiments to web-based learning objects: An advanced telelaboratory for robotics and crontrol systems. *IEEE Transactions on Industrial Electronics*, 56(12), December 2009.

[CHM+06] M. J. Callaghan, J. Harkin, E. McColgan, T. M. McGinnity e L. P. Maguire. Client-server architecture for collaborative remote experimentation. *Journal of Network and Computer Applications*, 30:1295–1308, September 2006.

[Fou10] The Apache Software Foundation. Authentication, authorization and access control, Fevereiro 2010. Disponível em http://httpd.apache.org/docs/1.3/howto/auth.html#basic, acedido a última vez em Fevereiro de 2010.

[GB09] Luís Gomes e Seta Bogosyan. Current trends in remote laboratories. *IEEE Transactions on Industrial Electronics*, 56(12), December 2009.

[GVEJ07] Sebastian Gröber, Martin Vetter, Bodo Eckert e Hans-Jörg Jodl. Experimenting from a distance - remote controlled laboratory (rcl). *European Journal of Physics*, pages 127–141, Abril 2007.

[HMJ06] Jud Harward, Ting Ting Mao e Imad Jabbout. Ilab interactive services - obverview. Technical report, Massachusetts Institute of Technology, 2006.

[HP09] Reza Hashemian e Timothy R. Pearson. A low-cost server-client methodology for remote laboratory access for hardware design. *39th ASEE/IEEE Frontiers in education Conference*, October 2009.

[Hul09] Steve Hull. How to keep persistent socket connections?, Maio 2009. Disponível em http://www.ruby-forum.com/topic/187517, acedido a última vez em Fevereiro de 2010.

[JSO10] JSON. Introducing json, Fevereiro 2010. Disponível em http://www.json.org/, acedido a última vez em Fevereiro de 2010.

[Kio09] Kioskea. Tcp/ip, Agosto 2009. Disponível em http://pt.kioskea.net/contents/internet/tcpip.php3, acedido a última vez em Fevereiro de 2010.

[Mic02] Sun Microsystems. Java blueprints - j2ee patterns, 2002. Disponível em http://java.sun.com/blueprints/patterns/MVC-detailed.html, acedido a última vez em Fevereiro de 2010.

[Mic10a] Microsoft. Microsoft netmeeting, Fevereiro 2010. Disponível em http://www.microsoft.com/downloads/details.aspx?FamilyID=26c9da7c-f778-4422-a6f4-efb8abba021e&displaylang=en, acedido a última vez em Fevereiro de 2010.

[Mic10b] Sun Microsystems. Java network launch protocol, Fevereiro 2010. Disponível em http://java.sun.com/docs/books/tutorial/deployment/deploymentInDepth/jnlp.html, acedido a última vez em Fevereiro de 2010.

[MIT] MIT. The challenge of building internet accessible labs. Technical report, Massachusetts Institute of Technology.

[MIT07] MIT. ilabs interactivelabserver with labview setup and configuration. online, 2007.

[MN06a] Jing Ma e Jeffrey V. Nickerson. Hands-on, simulated and remote laboratories: A comparative literature review. *ACM Computing Surveys*, 38(3), September 2006.

[MN06b] Jan Machotka e Zorica Nedic. The remote laboratory netlab for teaching engineering courses*. *Global J. of Engineering Education, Vol. 10, No. 2*, 10, 2006.

[MyS10] MySQL. Mysql, Fevereiro 2010. Disponível em http://www.mysql.com/, acedido a última vez em Fevereiro de 2010.

[MZC03] Stuart McCracken, Zeljko Zilic e Hoy Yun Henry Chan. Real laboratories for distance education. *Journal of Computing and Information Technology*, 11(1):67–76, March 2003.

[Ner89] Nancy J. Nersessian. Conceptual change in science and in science education. *Synthese*, 80(1):163–183, July 1989.

[NMN08] Zorica Nedic, Jan Machotka e Andrew Nafalski. Remote laboratory netlab for effective interaction with real equipment over the internet. Maio 2008.

[PHP09] PHP. History of php, 2009. Disponível em http://pt.php.net/manual/en/history.php.php, acedido a última vez em 15 de Dezembro de 2009.

[Pos10] PostgreSQL. Postgresql, Fevereiro 2010. Disponível em http://www.postgresql.org/, acedido a última vez em Fevereiro de 2010.

[Pro10] Mono Project. Faq: Asp .net. Website, February 2010. Disponível em http://mono-project.com/FAQ:_ASP.NET, acedido a última vez em Fevereiro de 2010.

[ROR10] ROR. Ruby on rails, Fevereiro 2010. Disponível em http://rubyonrails.org/, acedido a última vez em Fevereiro de 2010.

[Rub10] Ruby. About ruby, Fevereiro 2010. Disponível em http://www.ruby-lang.org/en/about/, acedido a última vez em Fevereiro de 2010.

[Sch07] Felix Schwarz. Secure php environments with php, suexec and fastcgi (mod fcgid) (en), Fevereiro 2007. Disponível em http://www.felix-schwarz.name/Secure_PHP_environments_with_PHP%2C_suexec_and_FastCGI_(mod_fcgid)_(en), acedido a última vez em Fevereiro de 2010.

[Sil09] Carlos Silvestre. Desempenho – apache com mpm-worker e fast-cgi, Junho 2009. Disponível em http://patch3s.wordpress.com/2009/06/29/alta-performance-apache-com-mpm-worker-e-fast-cgi/, acedido a última vez em Fevereiro de 2010.

[SQL10] SQLite. Sqlitel, Fevereiro 2010. Disponível em http://www.sqlite.org/, acedido a última vez em Fevereiro de 2010.

[Sun10] Sun. Javafx | rich internet applications development | rias java fx, Fevereiro 2010. Disponível em http://pt.php.net/manual/en/history.php.php, acedido a última vez em Fevereiro de 2010.

[Sym10] Symfony. About | symfony | web php framework. Website, Fevereiro 2010. Disponível em http://www.symfony-project.org/about, acedido a última vez em Fevereiro de 2010.

[Uni06] UniSA. Microlab - the online microcontroller remote laboratory. WebSite, 2006. Disponível em http://microlab.unisa.edu.au/, acedido a última vez em Fevereiro de 2010.

[Uni10] UniSA. Netlab - credits and sponsors, 2010. Disponível em http://netlab.unisa.edu.au/credits.xhtml, acedido a última vez em Janeiro de 2010.

[W3C10] W3C. Cascading style sheets, Fevereiro 2010. Disponível em http://www.w3.org/Style/CSS/, acedido a última vez em Fevereiro de 2010.

[w3s10] w3schools. Ajax introduction, Fevereiro 2010. Disponível em http://www.w3schools.com/Ajax/ajax_intro.asp, acedido a última vez em Fevereiro de 2010.

[Wik10a] Wikipedia. Http cookie, Fevereiro 2010. Disponível em http://en.wikipedia.org/wiki/HTTP_cookie, acedido a última vez em Fevereiro de 2010.

[Wik10b] Wikipedia. Zip (file format), Fevereiro 2010. Disponível em http://en.wikipedia.org/wiki/ZIP_(file_format), acedido a última vez em Fevereiro de 2010.

# REFERÊNCIAS

# Anexo A

# Medição dos Tempos de Resposta

## A.1 Pedidos Efectuados Através do Serviço *Web*

Nesta secção, é possível ver o código utilizado para gerar e medir o tempo médio dos pedidos feitos ao serviço *web* para activar ou desactivar um dispositivo.

É também possível visualizar as imagens com os resultados dos testes efectuados nos dois ambientes diferentes. Nessas imagens, os valores presentes no vector são o tempo que cada pedido demorou a ser executado e em *Average* pode ser observado o tempo médio.

### A.1.1 Código utilizado para fazer a geração dos pedidos

```
1  //ciclo que vai de m=1 ate m = 100
   for(m in [1..n]) {
3      //Se for premido o botao stop, a funcao retorna e a tarefa termina
       if(worker.cancelled) {
5          return;
       }
7
       //gerado um numero aleatorio entre zero e 15, que representa o led sobre o
           qual se vai actuar
9      ledNo = Math.round(15 * Math.random());

11     var loc = "";
       //se o estado do led for true, indica que ele esta aceso, pelo que vai
           fazer um load para o valor 1 para o apagar
13     if(status[ledNo]) {
           loc = "{baseLink}act/5/gui:true/command:LOAD_OWIO_{ledNo}_{1}.json";
15     } else {
           loc = "{baseLink}act/5/gui:true/command:LOAD_OWIO_{ledNo}_{0}.json";
17     }

19     //parser da resposta
       var parser = PullParser {
21         documentType: PullParser.JSON;
           onEvent: function(event: Event) {
23           if(event.type == PullParser.END_VALUE) {
                 if(event.name == "response") {
25                     println(event.text);
                       //se recebido LOAD OK significa que o pedido foi executado
                           com sucesso
```

```
27                  if(event.text.equals("LOAD OK\r\n")) {
                        //alteracao do estado do led para o novo estado
29                      if(status[ledNo]) {
                            status[ledNo] = false;
31                      } else {
                            status[ledNo] = true;
33                      }
                    }
35              }
            }
37      }
    };
39
    //construcao do pedido a ser enviado
41  var request: HttpRequest = HttpRequest {
        headers: [
43        header,
          HttpHeader {
45            name: "Accept"
              value: "application/json"
47        }
        ]
49      method: HttpRequest.GET
        location: loc
51      onStarted: function () {
            //startTime igual ao tempo actual, aquando o inicio da conexao com
                o servidor web
53        startTime = System.currentTimeMillis();
        }
55      onReading: function () {
            //aquando o inicio da leitura da resposta, e feita a "publicacao"
                do tempo que passou desde o pedido ser
57        //iniciado ate se comecar a ler a resposta
          worker.publish([System.currentTimeMillis() - startTime]);
59      }
        onInput: function(is: InputStream) {
61        try {
              parser.input = is;
63            parser.parse();
          } finally {
65            is.close();
          }
67      }
    };
69
    request.start(); //inicio do pedido
71
    while(not request.done) {
73      //enquanto a flag done nao for verdadeira, faz um ciclo com um sleep de
            50 ms, de modo a que nao seja
      //efectuado o pedido seguinte sem ter sido recebida a resposta do actual.
75      Thread.sleep(50);
    }
77 }
```

test_code.fx

### A.1.2 Resultados dos testes executados na mesma máquina do servidor *Web*

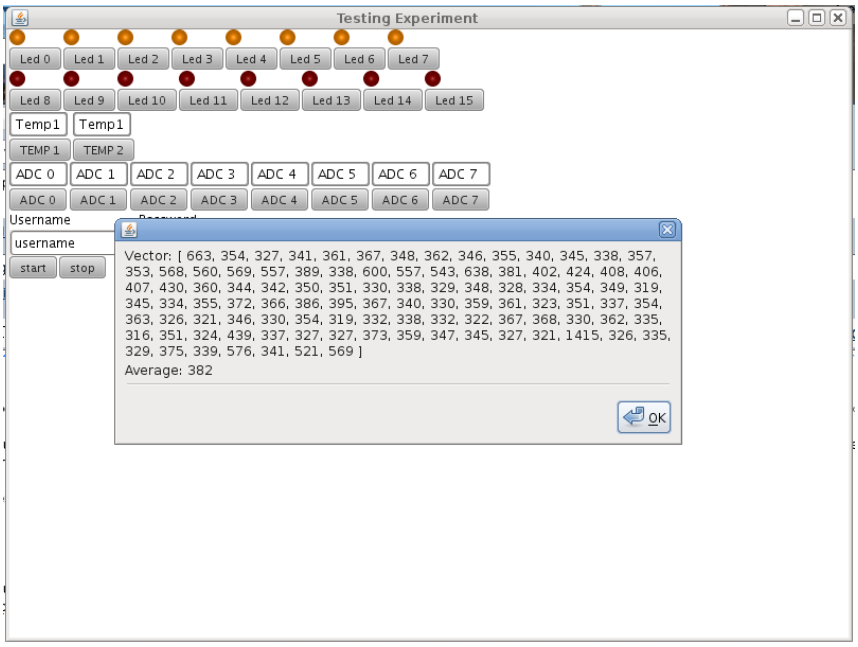



Figura A.1: 1ª Amostra Local - 101 pedidos

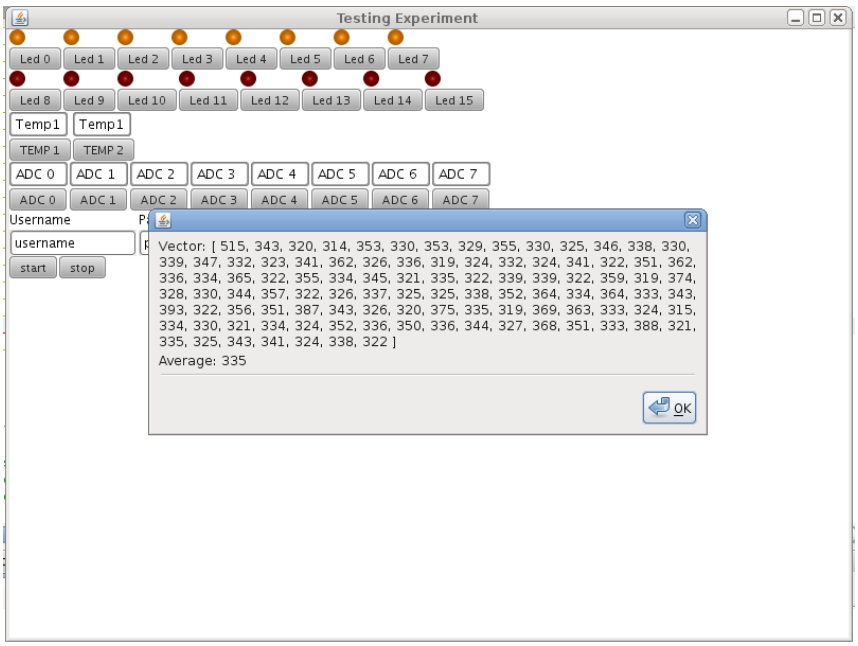



Figura A.2: 2ª Amostra Local - 100 pedidos

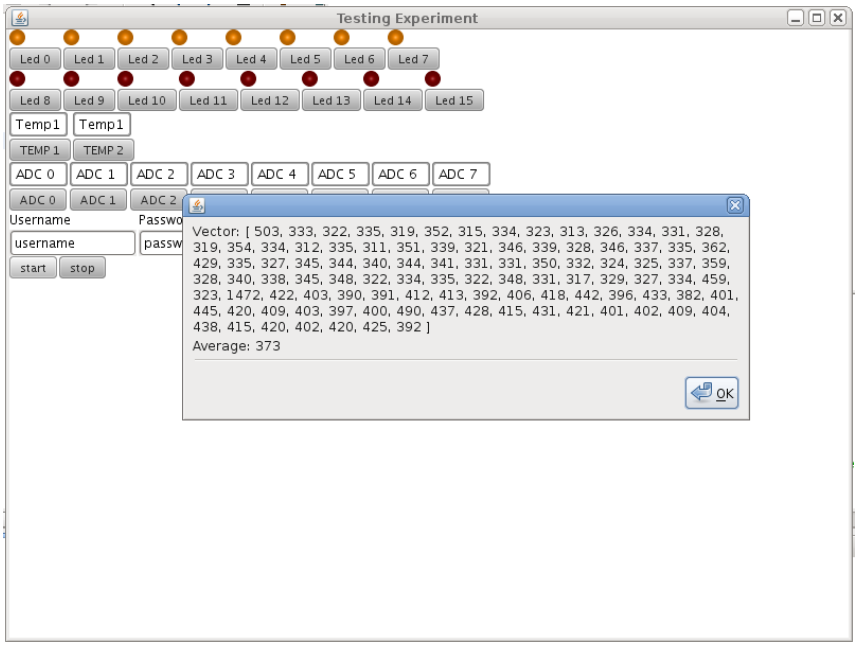



Figura A.3: 3ª Amostra Local - 100 pedidos

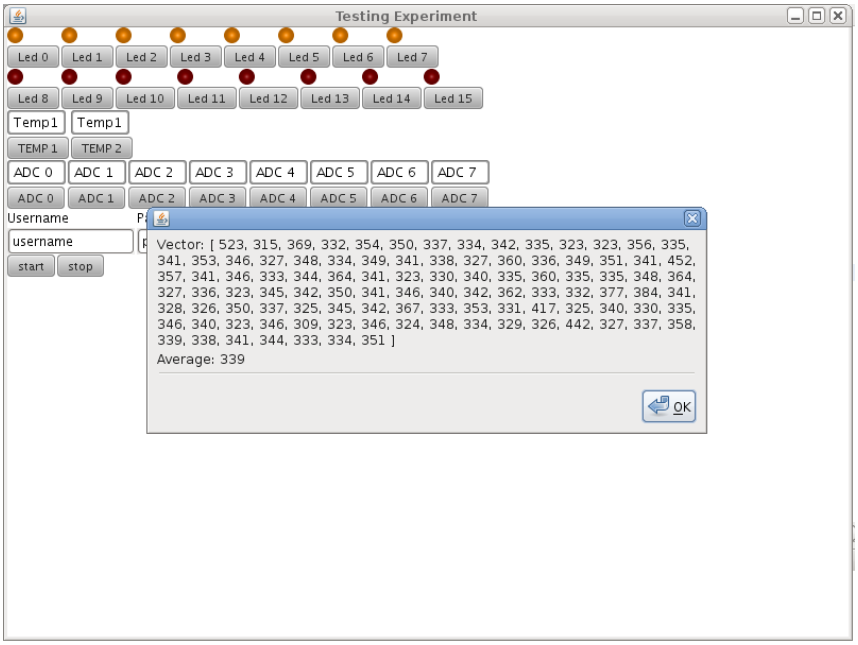



Figura A.4: 4ª Amostra Local - 100 pedidos

### A.1.3 Resultados dos testes executados numa máquina localizada na rede local do do servidor *Web*

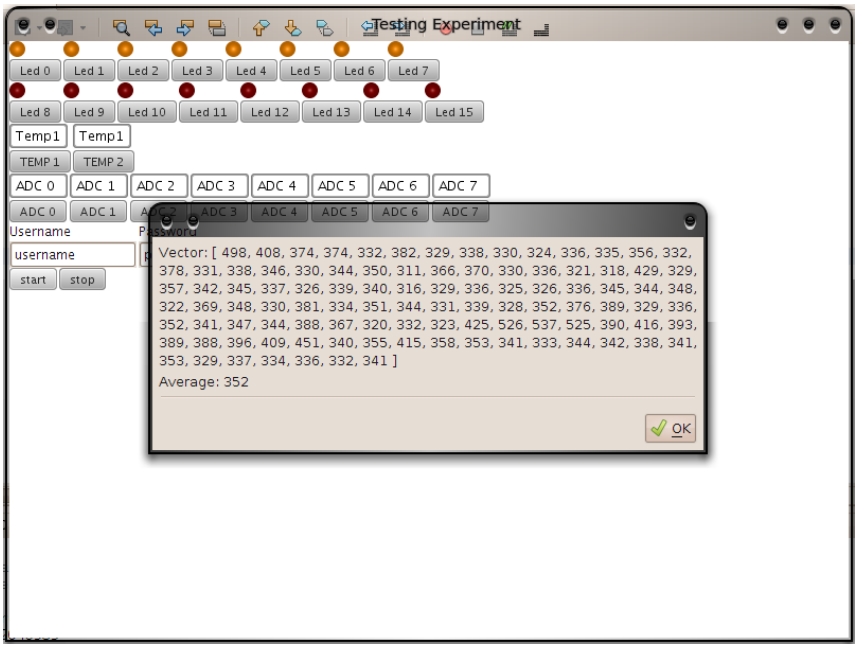



Figura A.5: 1ª Amostra Externa - 101 pedidos

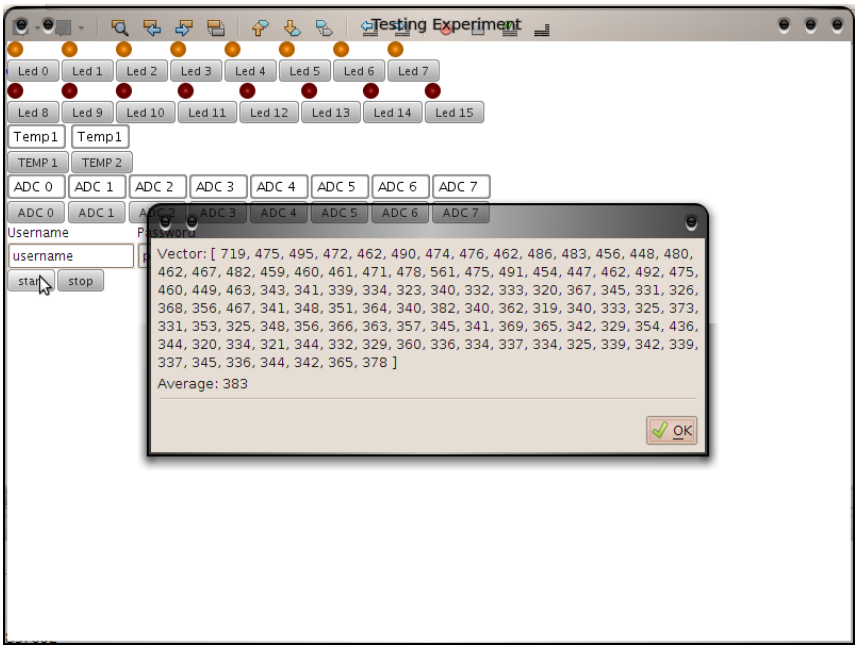



Figura A.6: 2ª Amostra Externa - 101 pedidos

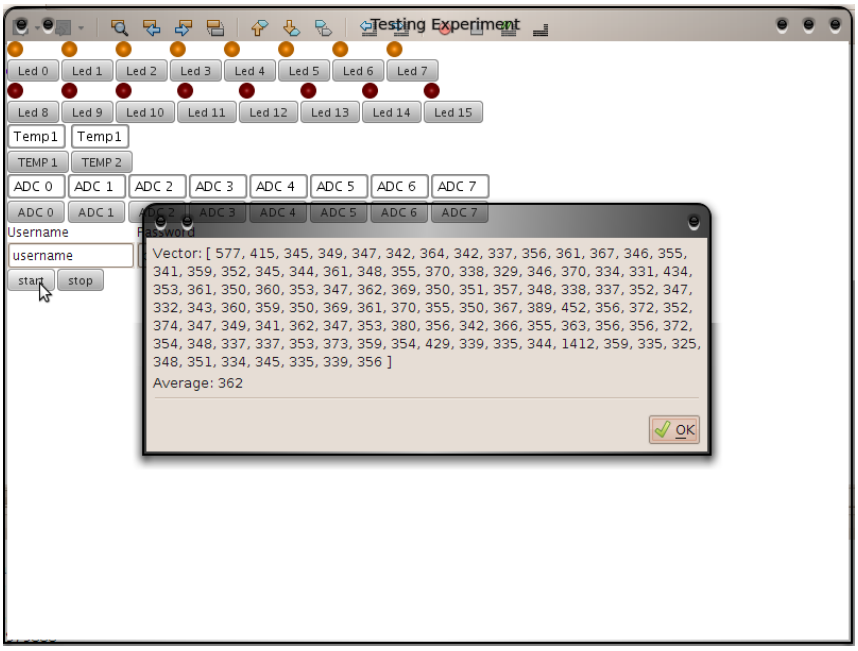



Figura A.7: 3ª Amostra Externa - 101 pedidos

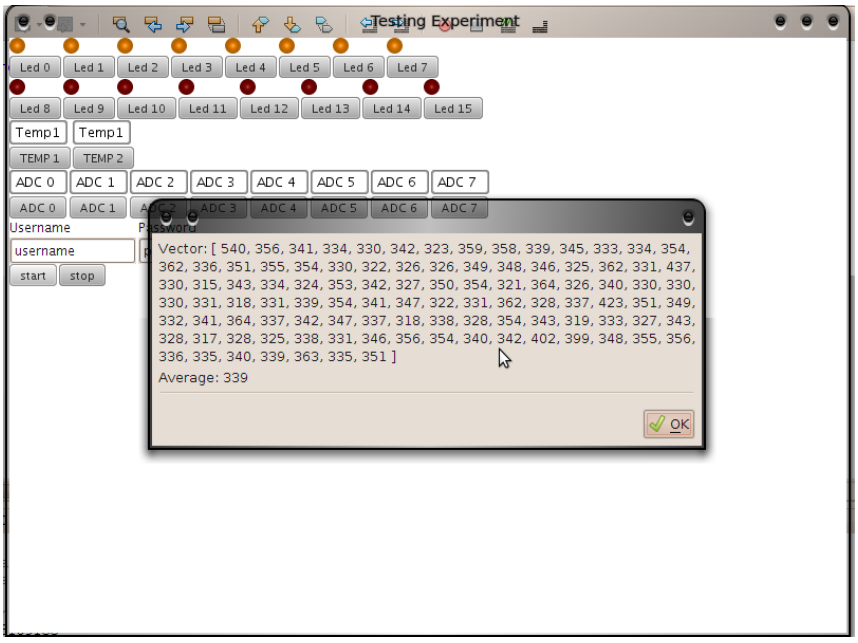



Figura A.8: 4ª Amostra Externa - 101 pedidos

## A.2 Pedidos Efectuados Através de uma Ligação Directa à Experiência

Nesta secção, é possível observar o código utilizado para efectuar os pedidos para activação e desactivação de um dispositivo, através de uma ligação directa à experiência.

Por fim, são apresentados os resultados obtidos, sendo que no vector se encontram os tempos individuais de cada pedido e fora do vector se encontra o tempo médio e o número total de pedidos efectuados.

### A.2.1 Código utilizado para fazer a geração dos pedidos

```
1  //array com os dispositivos para activar ou desactivar
   int[] leds = {0, 1, 6, 7, 10, 11, 12, 13};
3
   //array que guarda do lado do "cliente" o estado dos dispositivos
5  boolean[] state = {false, false, false, false, false, false, false, false};

7  //vector onde serao guardados os tempos
   Vector<Integer> times = new Vector<Integer>();
9
   //dispositivo a activar ou desactivar
11 int slot = 0;

13 //soma total dos tempos
   int average = 0;
15
   //contador de pedidos
17 int counter = 0;

19 //tempo de inicio do pedido
   long startTime = 0;
21
   //duracao do pedido
23 int duration = 0;

25 //comando a enviar
   String command = new String();
27
   Socket echoSocket = null;
29 PrintWriter out = null;
   BufferedReader in = null;
31
   try {
33     echoSocket = new Socket("192.168.103.124", 8080);
       out = new PrintWriter(echoSocket.getOutputStream(), true);
35     in = new BufferedReader(new InputStreamReader(echoSocket.getInputStream()))
           ;

37     //ciclo que a efectuar 499 vezes
       while (counter < 500) {
39       //seleccao aleatoria do dispositivo a activar
           slot = (int) Math.round(Math.random() * (leds.length-1));
41
           //se dispositivo activo, comando para desactivar. se dispositivo
               desactivo, comando para activar
43         if(state[slot]) {
               command = "load owio " + leds[slot] + " 1"; //comando para
                   desactivar dispositivo
45         } else {
               command = "load owio " + leds[slot] + " 0"; //comando para activar
                   dispositivo
47         }
```

```
49          startTime = System.currentTimeMillis(); //tempo de inicio de envio do
                pedido
            out.println(command); //envio do pedido
51          in.readLine(); //recepcao da resposta
            duration = (int) (System.currentTimeMillis() - startTime); //calculo da
                 duracao do pedido
53
            times.add(duration); //adiciona-se a duracao ao vector
55          average += times.lastElement(); //soma-se a duracao do ultimo pedido a
                soma

57          //alteracao do array de estados para manter consistencia dos dados com
                o hardware
            if(state[slot]) {
59              state[slot] = false;
            } else {
61              state[slot] = true;
            }
63
            counter++; //incrmento do contador de pedidos
65      }

67      System.out.println(times.toString());
        System.out.println("Average: " + average/times.size()); //calculo da medio
            do tempo
69
        out.close();
71      in.close();
        echoSocket.close();
73      System.exit(0);

75  } catch (UnknownHostException e) {
        System.err.println("Don't know about host: taranis.");
77      System.exit(1);
    } catch (IOException e) {
79      System.err.println("Couldn't get I/O for "
                           + "the connection to: taranis.");
81      System.exit(1);
    }
```

test_code.java

### A.2.2 Resultados obtidos

```
    [367, 205, 195, 192, 196, 196, 195, 196, 196, 192, 196, 195, 197, 199, 196,
 2  196, 195, 196, 192, 196, 196, 199, 196, 196, 200, 191, 192, 200, 200, 192, 199,
    196, 196, 200, 196, 195, 196, 192, 196, 200, 195, 192, 192, 196, 191, 192, 196,
 4  195, 197, 198, 196, 192, 197, 195, 196, 195, 193, 195, 196, 199, 192, 196, 200,
    199, 199, 192, 192, 200, 191, 200, 196, 196, 192, 199, 200, 196, 192, 196, 199,
 6  200, 192, 192, 196, 199, 192, 196, 193, 199, 199, 200, 192, 197, 194, 197, 195,
    200, 196, 199, 200, 192, 196, 200, 198, 193, 195, 193, 198, 200, 200, 196, 197,
 8  198, 193, 199, 196, 196, 195, 200, 200, 200, 196, 199, 192, 200, 192, 203, 196,
    196, 196, 196, 195, 196, 196, 196, 192, 199, 196, 200, 196, 196, 195, 192, 200,
10  200, 192, 194, 196, 196, 196, 195, 200, 192, 197, 199, 196, 195, 200, 192, 196,
    199, 197, 195, 196, 196, 191, 192, 192, 200, 196, 198, 200, 192, 200, 199, 200,
12  196, 200, 196, 199, 192, 196, 196, 196, 199, 192, 197, 199, 196, 199, 196, 193,
    195, 199, 195, 200, 192, 192, 199, 200, 200, 196, 200, 195, 200, 196, 196, 199,
14  196, 200, 196, 200, 195, 192, 192, 196, 192, 199, 200, 192, 196, 200, 191, 200,
```

```
196, 192, 192, 195, 196, 200, 200, 191, 191, 193, 195, 196, 195, 192, 197, 195,
192, 1336, 264, 273, 280, 272, 277, 289, 253, 259, 280, 267, 258, 284, 260,
279, 270, 283, 283, 276, 288, 272, 284, 274, 276, 294, 281, 260, 270, 267, 286,
261, 276, 287, 278, 278, 287, 284, 285, 288, 287, 276, 276, 283, 282, 297, 278,
283, 280, 296, 280, 280, 272, 269, 285, 271, 292, 280, 281, 270, 282, 279, 293,
277, 281, 288, 294, 279, 282, 278, 299, 284, 276, 288, 285, 278, 288, 291, 280,
294, 281, 292, 298, 283, 289, 299, 287, 293, 290, 273, 282, 285, 282, 302, 295,
290, 291, 275, 281, 287, 351, 340, 331, 338, 320, 334, 324, 331, 337, 313, 332,
339, 325, 320, 341, 337, 318, 331, 321, 339, 334, 339, 319, 307, 322, 336, 345,
325, 340, 329, 342, 360, 332, 323, 346, 331, 326, 334, 345, 347, 337, 346, 334,
344, 344, 326, 316, 344, 372, 323, 332, 329, 333, 350, 345, 358, 322, 327, 328,
268, 191, 196, 193, 195, 192, 199, 196, 192, 200, 196, 199, 192, 196, 192, 195,
191, 188, 193, 195, 191, 192, 200, 196, 188, 195, 192, 192, 197, 196, 194, 196,
196, 196, 196, 187, 192, 196, 196, 191, 199, 192, 192, 196, 195, 192, 196, 196,
196, 196, 199, 196, 193, 191, 192, 191, 193, 199, 196, 195, 192, 192, 196, 195,
199, 192, 197, 199, 191, 196, 196, 200, 189, 194, 197, 195, 192, 193, 198, 196,
192, 200, 191, 200, 196, 196, 196, 195, 200, 200, 196, 192, 199, 192, 200, 200,
196, 191, 200, 192, 200, 195]

Average: 231
Total Requests: 500
```

```
[242, 194, 200, 196, 200, 200, 196, 200, 203, 200, 193, 195, 197, 194, 192,
196, 196, 191, 200, 196, 196, 196, 191, 196, 197, 195, 200, 191, 200, 196, 192,
195, 191, 200, 193, 199, 196, 195, 192, 196, 193, 194, 196, 200, 197, 199, 199,
192, 196, 192, 195, 199, 196, 196, 196, 199, 197, 195, 196, 197, 194, 192, 196,
192, 200, 199, 192, 200, 192, 196, 195, 192, 192, 200, 195, 195, 200, 196, 196,
195, 196, 200, 196, 196, 195, 192, 193, 195, 201, 198, 200, 193, 193, 197, 195,
196, 192, 200, 196, 195, 193, 195, 200, 195, 196, 196, 192, 200, 199, 192, 200,
196, 196, 198, 196, 200, 196, 199, 197, 203, 196, 200, 195, 200, 197, 199, 192,
195, 196, 196, 196, 199, 200, 196, 200, 196, 195, 200, 300, 196, 199, 196, 196,
200, 196, 191, 196, 196, 200, 203, 195, 196, 196, 200, 199, 196, 196, 200, 196,
199, 196, 196, 1356, 264, 270, 272, 260, 266, 267, 276, 272, 274, 271, 274,
274, 282, 275, 275, 282, 264, 275, 269, 264, 276, 280, 282, 272, 263, 281, 277,
280, 268, 281, 265, 287, 275, 276, 282, 266, 271, 286, 291, 281, 275, 277, 277,
283, 272, 279, 275, 270, 273, 284, 277, 294, 277, 297, 268, 271, 293, 274, 259,
279, 277, 289, 277, 274, 268, 274, 252, 289, 278, 277, 267, 348, 334, 317, 330,
328, 333, 333, 327, 311, 329, 323, 327, 334, 337, 331, 310, 325, 344, 327, 320,
316, 311, 317, 335, 328, 335, 330, 347, 336, 337, 330, 325, 324, 350, 325, 279,
196, 192, 196, 192, 192, 195, 196, 196, 192, 191, 196, 200, 192, 191, 195, 192,
193, 200, 191, 191, 192, 196, 196, 195, 196, 196, 192, 196, 199, 196, 192, 192,
192, 195, 196, 200, 197, 199, 199, 196, 192, 196, 191, 199, 192, 192, 196, 203,
192, 196, 192, 196, 195, 192, 200, 192, 197, 198, 192, 193, 199, 195, 199, 196,
196, 200, 199, 196, 196, 192, 197, 199, 198, 192, 192, 196, 195, 200, 200, 192,
191, 196, 195, 196, 192, 200, 191, 196, 196, 192, 199, 196, 196, 192, 197, 194,
196, 196, 200, 191, 191, 196, 200, 192, 195, 196, 192, 200, 196, 191, 192, 192,
197, 199, 199, 200, 192, 200, 200, 191, 195, 193, 196, 191, 196, 192, 192, 200,
195, 196, 196, 192, 192, 199, 188, 192, 192, 193, 194, 200, 196, 200, 191, 191,
200, 196, 196, 197, 198, 197, 195, 192, 195, 200, 196, 192, 197, 198, 200, 192,
196, 192, 195, 1356, 267, 272, 262, 272, 270, 271, 260, 272, 269, 271, 270,
283, 269, 284, 290, 270, 276, 291, 272, 268, 272, 259, 277, 279, 259, 282, 274,
274, 279, 279, 289, 273, 286, 283, 274, 284, 268, 277, 283, 266, 292, 295, 274,
293, 284, 294, 271, 278, 284, 270, 283, 281, 286, 271, 293, 273, 274, 276, 273,
281, 279, 276, 276, 300, 292, 302]

Average: 232
Total Requests: 500
```

# Anexo B

# Processos de Configuração do Servidor

Eventualmente, depois do Apache e do PHP estarem correctamente configurados, o Apache poderá apresentar uma página de erro, com o código 500 (*Internal Server Error*) aquando a execução de ficheiros PHP correctos. Quando o erro ocorrer, nos registos de erros do Apache, deverá aparecer a seguinte sequência de mensagens:

> *'mod_fcgid: read data timeout in 20 seconds*
> *[error] [client ip] Premature end of script headers: index.php, referer: ...*
> *mod_fcgid: process <script>(pid) exit(communication error), get stop signal 15'*

Este erro dever-se-á a um *timeout* na execução do ficheiro PHP. O problema poderá ser resolvido aumentando o prazo máximo de execução do ficheiro. Para mais informações sobre o erro, dever-se-á consultar [Sch07].