UNIVERSIDADE FEDERAL DE OURO PRETO

ESCOLA DE MINAS

COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - CECAU

HENRIQUE DA CUNHA MELO

CONTROLE DE ACESSO DE LABORATÓRIO REMOTO UTILIZANDO O WEBLAB-DEUSTO

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2017

HENRIQUE DA CUNHA MELO

CONTROLE DE ACESSO DE LABORATÓRIO REMOTO UTILIZANDO O WEBLAB-DEUSTO

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Paulo Marcos de Barros Monteiro

Co-orientador: José Alberto Naves Cocota Júnior

Ouro Preto

Escola de Minas – UFOP

Setembro/2017

M528c Melo, Henrique da Cunha.
Controle de acesso de laboratório remoto utilizando o WEBLAB-DEUSTO [manuscrito] / Henrique da Cunha Melo. - 2017.

90f.: il.: color.

Orientador: Prof. Dr. Paulo Marcos de Barros Monteiro.
Coorientador: Prof. MSc. José Alberto Naves Cocota Júnior.

Monografia (Graduação). Universidade Federal de Ouro Preto. Escola de Minas. Departamento de Engenharia de Controle e Automação e Técnicas Fundamentais.

1. Acesso Remoto. 2. Controle de Acesso. 3. Rastreio de Informação. 4. Laboratório Remoto. 5. WebLab-Deusto. I. Monteiro, Paulo Marcos de Barros. II. Cocota Júnior, José Alberto Naves. III. Universidade Federal de Ouro Preto. IV. Titulo.

Catalogação: ficha@sisbin.ufop.br CDU: 681.5

Monografia defendida e aprovada, em 27 de setembro de 2017, pela comissão avaliadora constituída pelos professores:

Prof. Dr. Paulo Marcos de Barros Monteiro – Orientador

Prof. M. Sc. José Alberto Naves Cocota Júnior – Co-orientador.

Prof. Dr. Agnaldo José da Rocha Reis – Professor Convidado.

Prof. M. Sc. João Carlos Vilela de Castro – Professor Convidado.

Dedico este trabalho à Samir, Cristina e Jessica. Obrigado por acreditarem enquanto, em momentos, eu não era capaz. Além disso, agradeço toda a paciência que tiveram durante essa “curta” jornada.

AGRADECIMENTOS

Agradeço ao meu pai, minha mãe e minha namorada por todo o auxílio, paciência, carinho e amor nessa jornada.

Aos meus orientadores, os professores, Paulo e Cocota por transformarem a oportunidade de encerrar à minha graduação em realidade, além de todo o auxílio prestado durante a concepção do trabalho. Além disso, gostaria de agradecer ao Cocota por toda a ajuda durante o curso.

Aos meus familiares pela torcida e por já terem comemorado antecipadamente essa data comigo.

Ao Pablo Orduña, por todo o auxílio em transformar as ideias do projeto em realidade, adaptando as realidades do WebLab-Deusto para a realidade do Laboratório Remoto da Escola de Minas, sendo parte fundamental do processo.

A UFOP e a cidade de Ouro Preto pelos 8 anos de estada, posso olhar para trás e dizer no futuro que realmente aproveitei os melhores anos da minha vida, com uma dose de piores anos também. E também para todo mundo que fez parte da jornada, mesmo que por breves momentos, pois a parte mais importante da chegada é o caminho que trilhamos até ela.

“I am the Special One!”

(José Mourinho)

RESUMO

Esse trabalho trata da aplicação de uma plataforma para controle de acesso e rastreio de informações de uso e de usuários no Laboratório Remoto da Escola de Minas, aperfeiçoando a atual estrutura existente no Laboratório Controle e Automação de Multiusuário (LabCAM) da Escola de Minas – UFOP, através da instalação e adaptação da plataforma WebLab-Deusto. São ferramentas como essa que permitem instituições e docentes transformarem a interação dos estudantes com o aprendizado em seu curso, fornecendo ferramentas relevantes para a sua formação e carreira. Isso foi alcançado através da aplicação de APIs disponibilizadas pela equipe do WebLab-Deusto, permitindo uma perfeita adaptação às necessidades existentes, sem a necessidade de reavaliar ou transformar a tecnologia existente. Além disso, consiste na adaptação de um computador em um *Web Server*, que será responsável por gerenciar separadamente a nova plataforma, fazendo com que a adaptação do servidor do Laboratório Remoto da Escola de Minas seja independente e com o mínimo de riscos possíveis.

Palavras-chave: Acesso Remoto, Controle de Acesso, Rastreio de Informação, Laboratório Remoto, WebLab-Deusto, APIs.

ABSTRACT

The presented work is about the deployment of a platform for access control and users and usage tracking on the Laboratório Remoto da Escola de Minas, improving the current structure existing in the Laboratório Controle e Automação de Multiusuário da Escola de Minas – UFOP, through the deployment and adaptation of WebLab-Deusto platform. Tools like this enable institutions and academic teachers to transform the student interaction with the learning process inside their degrees, supplying relevant knowledge for their professional formation and careers. This was accomplished by the development of APIs by the WebLab-Deusto team, allowing the perfect adaptation to the existing necessities, without the need of revaluating or revamping the current technology. Furthermore, it consisted on the adaptation of a computer to a Web Server, who will be responsible for the separate management of the new platform, making the adaption of the Laboratório Remoto da Escola de Minas independent and with minimum risks.

Key words: Remote Access, Access Control, Information Tracking, Remote Laboratory, WebLab-Deusto, APIs.

Lista de Figuras

Lista de Siglas

Sumário

1. Introdução

A sociedade moderna sempre foi extremamente dependente dos recursos tecnológicos desenvolvidos ao longo da história e a cada novo avanço tecnológico, as ações tomadas em qualquer área do conhecimento ou do dia-a-dia se tornam intrinsicamente ligadas à tecnologia disponível. Hoje, a internet e a engenharia de computação e softwares são os principais fatores em qualquer segmento que envolvam a nossa presença.

O ser humano vive em um mundo onde computação na nuvem, maiores capacidades de processamento e avanços computacionais tornam novas descobertas no ramo obsoletas a cada 6 meses. Hoje, a extensa maioria da população global carrega um computador portátil em suas mãos ou bolsos, muito mais potentes e poderosos do que computadores de cinco anos atrás.

Além disso, todos esses elementos estão conectados diretamente com a rede mundial de computadores e isso torna possível uma dezena de atividades distintas sem que a presença física seja mais necessária. Basta ter um computador e um ponto com acesso à internet e você pode estar em instantes vinculado a outras pessoas que estão em diversas localidades distintas pelo globo terrestre.

Assim como diversas áreas, a educação não poderia deixar de aproveitar as vantagens e confortos que essas tecnologias proporcionam. Com o barateamento de hardwares e softwares ao longo dos anos, é possível que cada pessoa possa montar sua própria unidade de processamento de informações, como podemos ver nas unidades PICs e Arduinos espalhados por quase todos os projetos universitários e acadêmicos atualmente.

Os laboratórios remotos se beneficiaram desse novo *status quo* e tem o potencial para mudar a forma como os estudantes, professores e unidades educacionais se relacionam com atividades práticas. Anteriormente, era necessária uma estrutura física em forma de um laboratório extenso e enorme que pudesse receber uma turma de alunos, hoje, você pode transformar isso em algo com proporções menores e que permitam que cada aluno possa interagir virtualmente com a mesma atividade, de maneira mais próxima e objetiva do que num experimento que envolvessem diversos de seus colegas.

Além das praticidades para o estudante, existem as facilidades financeiras. Um laboratório, na maioria dos casos, costuma ser caro, difícil de manter e precisa de um profissional especializado guiando todo o processo de experimentação para evitar danos à propriedade material e também aos operadores leigos.

Com uma sociedade virtualizada, seria muito mais seguro e interessante utilizar de ambientes simulados, mas nada substitui verdadeiramente o contato físico e próximo com uma situação estudada, além disso, é muito mais educacional que o aluno veja os fundamentos e leis que envolvem o objeto de estudo, para que ele possa ter a necessária capacidade de compreensão ao final da sua carreira estudantil ou acadêmica.

Um laboratório remoto é mais barato, já que não é necessário que seja produzido em grande escala, muito mais simples de se manter, já que envolvem muito menos materiais para serem vistoriados e você pode instruir seguramente os alunos através de informações na página de internet. Sendo assim, você virtualiza a estrutura física, trazendo as vantagens e facilidades de uma simulação, mas sem perder a experimentação real necessária.

Um dos primeiros expoentes na utilização de técnicas remotas de experimentação foi o *The Interactive Nano-Visualization in Science and Engineering Education* (IN-VSEE*)*, um consórcio de instituições de ensino de diversos estados dos Estados Unidos da América, onde diversos experimentos laboratoriais estão disponíveis para fomentar o ensino de nanociência e conceitos tecnológicos, tendo inclusive um prêmio Nobel como consultor da plataforma.

Na Alemanha, existem a *Learnet* e *ControlNet24* que são voltadas para a área de Engenharia de Controle. São os pioneiros nessa área e foram realmente desenvolvidos para servirem como uma comunidade laboratorial para os seus usuários. Porém, como foi desenvolvido em alemão, a necessidade de conhecimento do idioma torna-se necessária para quem deseja se aventurar.

No Brasil, têm-se o *Remote Experimentation Laboratories* (*RexLab)*, da Universidade Federal de Santa Catarina (UFSC) e da Universidade do Sul de Santa Catarina (UNISUL), voltados para a área de experimentação através de sistemas embutidos e micro-controlados. Porém, assim como os laboratórios alemães, são desenvolvidos no idioma materno, nesse caso, o

português brasileiro, dificultando o acesso internacional, mesmo com o *RexLab* fazendo parte de uma rede intercontinental de laboratórios atualmente.

Além desses, existem outros laboratórios importantes, como o *iLabs*, do *Massachussets Institute of Technology* (MIT), desenvolvido para auxiliar os estudantes da universidade com experimentos em tempo real. A *Universidade de Bordeaux*, na França, é pioneira na área de laboratórios remotos para o estudo de eletrônica. Por fim, temos os laboratórios da *Universidade de Deusto*, localizada no País Basco, Espanha e foi através do trabalho desenvolvido por lá, com o desenvolvimento da plataforma *WebLab-Deusto*, que houve a iniciativa e desejo de contar com o Laboratório Remoto da Escola de Minas (LabREM), na Universidade Federal de Ouro Preto (UFOP), buscando auxiliar os discentes a se familiarizarem com as atividades relativas as disciplinas de Teoria de Controle, do curso de Engenharia de Controle e Automação.

O *WebLab-Deusto* é uma plataforma *open-source* de controle e desenvolvimento de laboratórios remotos, fornecendo um conjunto de *Application Programming Interface* (API) para o desenvolvimento de novas instâncias, ferramentas para controle de usuários, permissões, rastreio de informações, coleta de dados, compartilhamento de laboratórios remotos e a possibilidade de utilizar outras instâncias da própria plataforma.

Através das ferramentas disponibilizadas pela plataforma, o LabREM se tornará uma instância do *WebLab-Deusto*, sendo capaz de utilizar toda a estrutura disponível para tornar a atual estrutura mais eficiente, segura e com uma vasta gama de informações que possibilitam uma melhor capacidade de administração e controle das funções disponibilizadas pelo atual laboratório.

1.1 Objetivo

1.1.1 Objetivo Geral

- Realizar a implementação da plataforma *WebLab-Deusto* no Laboratório Remoto da Escola de Minas para o controle de acesso ao site do Laboratório.

1.1.2 **Objetivos Específicos**

- Implementação de um *Web Server* independente para viabilizar o uso da plataforma *WebLab-Deusto.*
- Integrar o software do Laboratório Remoto da Escola de Minas com o software do *WebLab-Deusto.*
- Configurar o *WebLab-Deusto* para realizar as funções de controle de acesso, rastreio de uso pelo usuário e coleta de dados dos experimentos realizados na plataforma.

1.2 Metodologia utilizada

Para o desenvolvimento do trabalho foi utilizada a estrutura existente do LabREM e a implementação de um *Web Server* que permitisse a integração entre o servidor do laboratório remoto com o servidor do *WebLab-Deusto*, localizado no Laboratório de Máquinas Elétricas da Escola de Minas, da UFOP.

Ocorreu a instalação da plataforma, a adaptação do servidor remoto às configurações pertinentes do *WebLab-Deust*o e a configuração da nova plataforma para funcionar dentro dos parâmetros buscados e necessários para o perfeito funcionamento do projeto.

1.3 Justificativa

A atual estrutura do LabREM permite que os usuários acessem simultaneamente através de um endereço web específico, sem gerenciamento de usuário, sem controle de acesso e muito menos sem saber a origem do acesso e como foi utilizado o laboratório.

A plataforma *WebLab-Deusto* permite que se realize funções ausentes na conjuntura atual, fazendo com que sua instalação e configuração se tornem partes cruciais da viabilidade do projeto para o uso futuro por discentes e docentes da Escola de Minas, da UFOP e também por outras instituições interessadas no projeto.

1.4 Estrutura do Trabalho

O trabalho foi dividido em 3 módulos: revisão bibliográfica, desenvolvimento do projeto e resultado e conclusões. A revisão bibliográfica cobre o capítulo 2, onde se debate e analisa tudo que é importante para compreendermos o escopo do projeto. O desenvolvimento do projeto é debatido no capítulo 3, onde discute-se como foi feito para atingirmos o objetivo e todas as partes necessárias para chegarmos até esse fim. Por fim, no último módulo, que compreende os capítulos 4 e 5, discute-se os resultados obtidos e conclui-se o trabalho, sua importância e sugestões de trabalhos que podem continuar o exposto aqui.

2. Revisão Bibliográfica

2.1 Acesso Remoto

A quantidade de informação sendo trocada pela humanidade é imensa e cresce exponencialmente e com isso nos tornamos uma sociedade baseada no compartilhamento de conhecimentos, já que em média a cada 2 ou 3 anos, nosso conhecimento acumulado universalmente duplica e isso traz muitos desafios para quem ensina. Como compartilhar esse tanto de informação disponível de uma maneira que contribua para o desenvolvimento das habilidades, aptidões e conhecimentos dos discentes e docentes envolvidos no processo (HUBA et al, 2004)?

Com o avanço das tecnologias de difusão e compartilhamento de dados e informações, as formas de comunicações e interações também evoluíram e com isso, modificou-se o panorama de diversas áreas. Desde 1998, a *United Nations Education, Scientific and Cultural Organization* (UNESCO) recomenda o uso de tecnologias de compartilhamento de informações e comunicação no ensino, mostrando que a tecnologia deve ser aliada do aprendizado (HUBA et al, 2004).

Com isso, a atividade de ensino passou da interação direta entre professor-aluno para uma abordagem mais centrada no aluno e suas interações com os meios disponíveis para aprendizado à sua volta. Por isso, laboratórios com acesso remoto tem o potencial para transformarem o processo de ensino e estudo. Segundo Huba et al (2004, p.135, tradução do autor), “em ambientes nos quais os estudantes estão engajados, eles se tornam capazes de ter uma maior responsabilidade no seu aprendizado e construírem seu próprio conhecimento”.

Sendo assim, o acesso remoto, ou controle remoto, é uma maneira de realizarmos o controle de longa distância ou remoto de alguma operação ou atividade.

> Segundo Kennepohl *et al.* (2005, p.1, tradução do autor) “(...) controle remoto está muito bem incorporado no nosso crescente mundo tecnológico. Nós temos ignição automática de um carro através do chaveiro, esquadrões antibombas usando regularmente robôs, drones de reconhecimento não tripulados são usados pelos exércitos, e muitos de nós usamos um controle remoto para mudar o canal da televisão ”.

Usualmente, o acesso remoto e controle remoto são utilizados em situações em que é impossível o alcance físico ou até mesmo um ambiente inóspito para a presença de vida. Mas como enfatizado, laboratórios são o núcleo de muitos cursos, principalmente nas áreas de ciências

aplicadas. E replicar essa experiência física num interfaceamento remoto apresenta muitos desafios e nenhuma resposta definitiva e universal (KENNEPOHL *et al.*, 2005).

O acesso remoto permite que qualquer dispositivo, com acesso a outro através de uma rede de servidores, possa se comunicar com outro dispositivo fisicamente distante ou inacessível. Obviamente que é necessário previamente configurar o destino desse acesso. A internet facilitou esse processo, pois existem aplicações e softwares que substituem todo um trabalho de configuração mais complexo.

O método mais comum de utilização dessa configuração é através de uma *Virtual Private Network* (VPN). Essa VPN cria uma ligação segura entre o dispositivo alvo e o dispositivo que irá administrá-lo remotamente.

> "Para criar uma VPN existem duas maneiras: por meio do protocolo SSL ou softwares. Na primeira, a conexão pode ser feita usando somente um navegador e um serviço em nuvem. No entanto, sem o mesmo nível de segurança e velocidade dos programas desktop. Já na segunda e mais comum forma de acesso remoto, é necessário um software que utiliza protocolo IPseg para fazer a ligação direta entre dois computadores ou entre um computador e um servidor. Nesse caso, a conexão tende a ser mais rápida e a segurança otimizada".
>
> (ALVES, Paulo. *O que é acesso remoto? Entenda tudo sobre conexão à distância.)*

Porém, a VPN não é a única forma de se configurar uma rede de acesso remoto: você pode utilizar serviços básicos de internet, softwares de controle remoto, serviço de acesso remoto em uma configuração de rede *Local Area Network* (LAN), Terminais de Serviços, *Wide Area Network* (WAN) e esquemas de sincronização ou replicação de base de dados. A mais simples, obviamente é a combinação de internet e softwares, enquanto redes WAN são inviáveis do ponto de vista pessoal, sendo aplicado apenas em grandes escalas, quando é possível fisicamente e financeiramente a montagem de uma rota privada interligando uma ou mais localidades.

O acesso remoto traz diversas vantagens como a expansão do alcance de determinada atividade, como por exemplo, uma grande empresa pode ter diversos centros comerciais interligados remotamente, facilitando a comunicação e troca de informações. Essa facilitação no acesso à informações e conhecimento é outra vantagem do acesso remoto, que faz com que a produtividade global das atividades interligadas se torne maior, além de ser uma grande economia de tempo e o motivo mais interessante de um mundo global e capitalista, a economia de custos, já que viabiliza formas mais baratas de interação.

Pelas características de agilizarem a troca de informação, permitirem uma interação direta e segura, mas num ambiente de escala muito menor que num ambiente físico e estruturado. Por serem mais simples e com implementação e manutenção menos dispendiosas, o acesso remoto é uma ferramenta interessante na difusão do conhecimento.

2.2 Arquitetura de Laboratórios Remotos

A configuração de hardware de um laboratório remoto pode ser bastante flexível e assumir diversas formas. A existência de alguns sistemas de gestão de laboratórios remotos como o *iLab*, do MIT, *Sahara*, da *University of Technology Sydney* (UTS), em Sydney, Austrália e o *WebLab-Deusto*, da Universidade de Deusto acabam fornecendo ferramentas nas formas de APIs que podem simplificar o processo de montagem. Ainda assim, é necessário um conhecimento avançado de programação, além de um conhecimento de configurações de sistemas mais complexo em todo o processo de elaboração de um laboratório remoto (KALUZ *et al*, 2015).

As arquiteturas mais comuns são as de *computador-servidor com um software de controle*, *computador-servidor com software de controle e placa de aquisição de dados*, *arquitetura ramificada*, *computador-servidor com um sistema SCADA e PLC* e *um computador de baixo-custo com micro controlador ou Field Programmable Gate Array (FPGA)* (KALUZ *et al*, 2015).

A figura 2.1 demonstra esses diferentes tipos de arquiteturas.

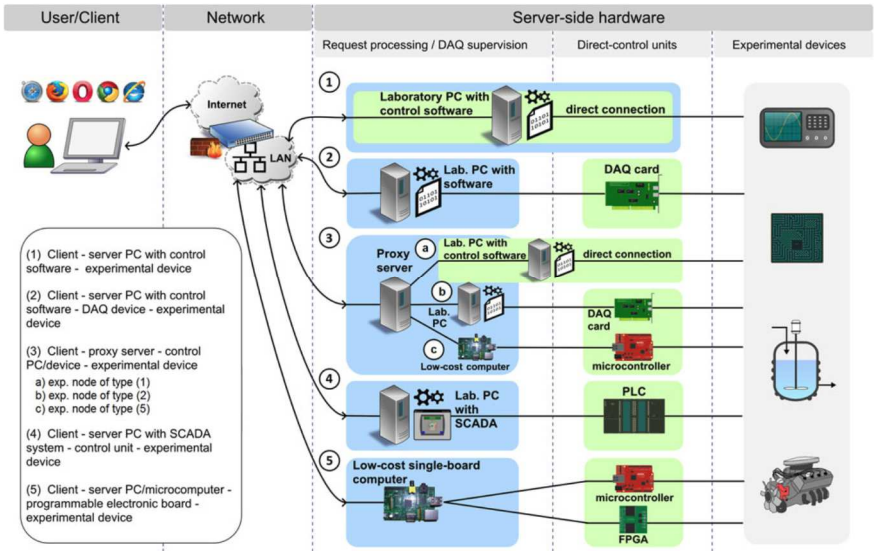

Figura 2.1 - Diferentes tipos de arquitetura de controle

FONTE: KALUZ et al, 2015, p. 300.

2.2.1 Computador-servidor com Software de Controle

É a configuração mais comum, pois utiliza a conexão direta entre um computador (que faz o papel de servidor) e se conecta diretamente ao experimento através de uma conexão USB ou porta serial. É a implementação mais simples dentre as disponíveis e também uma das mais baratas. Usualmente, softwares como MATLAB ou LabView são utilizados para criar o ambiente necessário (KALUZ *et al*, 2015).

O usuário acessa o seu computador e através da internet faz uma conexão com o server do experimento, esse server contém o hardware e software essencial e qualquer equipamento fundamental para representar o sistema real. O server permitirá que o usuário comande o experimento através do seu computador, retransmitindo as ações realizadas para o equipamento (BISTÁK, 2011).

A figura 2.2 representa o funcionamento dessa configuração.

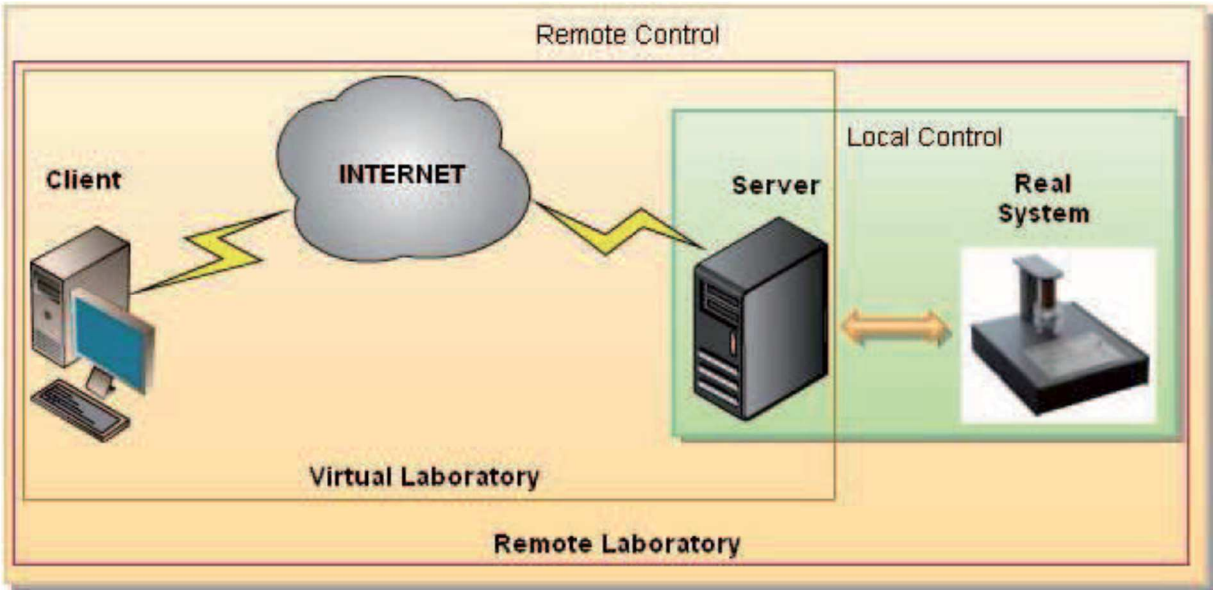

Figura 2.2 - Representação da arquitetura Computador-servidor com Software de Controle

FONTE: BISTÁK, P., 2011, p. 345

2.2.2 Computador-servidor com Software de Controle e placa DAQ

Nesse caso temos um ambiente muito semelhante ao anterior, a diferença é a utilização de uma placa de *Data Acquisition* (DAQ). Geralmente é usada quando não é possível uma conexão direta entre o computador e o experimento (KALUZ *et al*, 2015).

Assim como em 2.2.1, o funcionamento é muito similar, porém, caso não seja possível a ligação direta entre o server e o experimento, uma placa de aquisição de dados é colocada para servir de ponte e fazer o interfaceamento entre as duas entidades, permitindo que o funcionamento ocorra da mesma maneira como numa ligação direta entre as duas partes.

O mercado fornece uma grande variedade de placas de aquisição de dados, mas isso acaba encarecendo um pouco com relação a 2.2.1. Portanto, é uma opção utilizada apenas se é inviável realizar a configuração anterior.

2.2.3 Arquitetura Ramificada

Também alcunhado como *servidor-proxy e suas distintas configurações* provém uma maior possibilidade de configurações. O servidor-proxy é usado para gerenciar diferentes clientes simultaneamente, permitindo que você tenha diferentes tipos de laboratórios conectados numa mesma estrutura de acesso, sem que eles estejam fisicamente localizados no mesmo lugar. Pode ser configurado em conjunto com as duas primeiras configurações e também acoplar a configuração de *computador de baixo-custo com micro controlador* (KALUZ *et al*, 2015).

Essa arquitetura permite que você crie uma rede de laboratórios e experimentos vasta e diversa, onde você pode montar seu sistema em qualquer arquitetura e integrá-lo com outros, criando uma linha de acesso direta com essa rede integrada e através do *servidor-proxy* ocorrerá a redistribuição da atividade desejada pelo cliente.

O usuário irá realizar o seu acesso normalmente e em determinada interface irá decidir qual experimento quer realizar, a partir daí o *servidor-proxy* irá direcioná-lo de acordo com sua escolha inicial. Isso permite que instituições distintas colaborem e formem laboratórios integrados. Por exemplo, uma universidade no Brasil pode-se associar com uma universidade na China e ambos fornecerem os seus laboratórios para os seus estudantes e para os da outra universidade.

O *Sistema de Laboratorios a Distancia* (SLD) criado pelo Departamento de Sistemas Automáticos e Computacionais da *Universidade Central "Marta Abreu" de Las Villas* (UCLV)) em colaboração com a *Universidad Politécnica de Madrid* (UPM) é um exemplo de configuração de arquitetura ramificada. A Figura 2.3 traz a representação do funcionamento do SLD, por consequência, a representação de um laboratório com arquitetura ramificada. O LabREM é baseado nesse tipo de arquitetura.

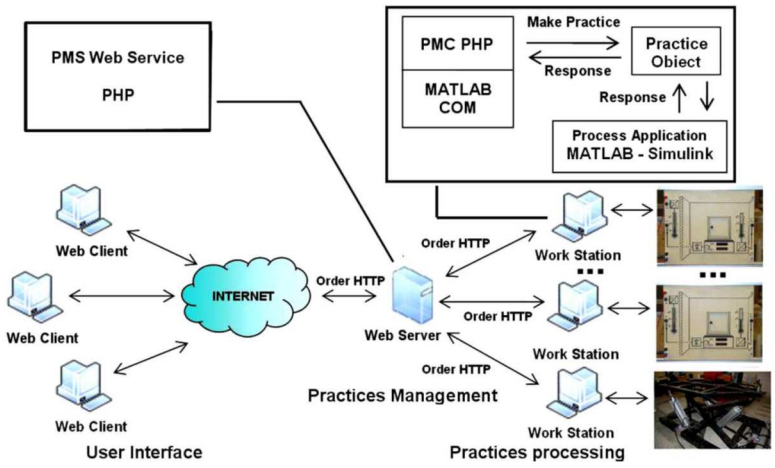

Figura 2.3 - Representação de um laboratório de arquitetura ramificada

FONTE: SANTANA et al, 2013, p. 549

2.2.4 Computador-servidor com um sistema SCADA e PLC

No caso do *computador-servidor com um sistema SCADA*, temos a junção de um sistema SCADA com um PLC, ou qualquer outro tipo de controlador. É mais utilizado em aplicações industriais, porém requer um maior investimento, já que tanto o software quando o hardware tem custos elevados, porém, o alto valor é recompensado com uma implementação mais rápida e simples, além de permitir uma simples *Human-Machine Interface* (HMI) (KALUZ *et al*, 2015).

Em comparação com as arquiteturas anteriores, temos a substituição do *computador-servidor* por um computador com um sistema SCADA e um PLC. O sistema SCADA irá fazer a interface entre o cliente e o experimento, enquanto o PLC irá controlar o experimento de acordo com as instruções dadas pelo usuário na plataforma contendo o SCADA.

Existe uma grande variedade de PLCs e sistemas SCADA no mercado, portanto, a configuração de um sistema depende da marca selecionada para realizar a implementação. Além do custo elevado já mencionado anteriormente, essa configuração não pode ser simplesmente feita através de uma conexão à internet e ligação direta ao experimento, sendo necessária em algumas ocasiões, a preparação de uma rede de conexão local (seja ethernet ou wireless) que interligue todos os componentes necessários para terem acesso à rede mundial de computadores.

Assim como a arquitetura ramificada, permite que você tenha diversos laboratórios conectados ao mesmo tempo, entretanto é limitada pela configuração física necessária. A figura 2.4 representa uma configuração utilizando o sistema SCADA e PLC.

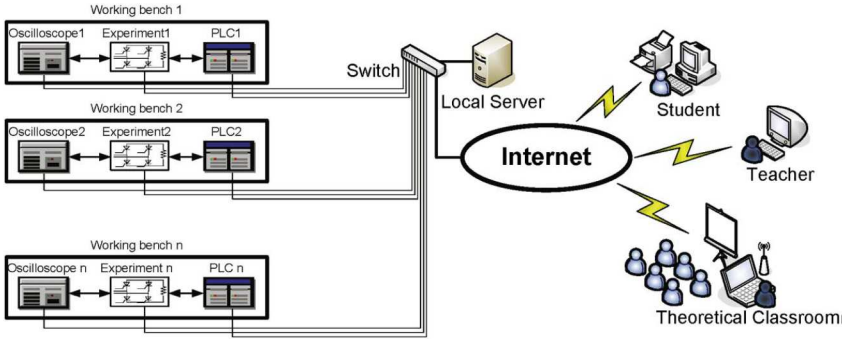

Figura 2.4 - Representação de um laboratório utilizando um sistema SCADA e PLC

FONTE: MARQUES *et al*, 2013, p. 2427.

2.2.5 Computador de baixo-custo com microcontrolador

Por fim, temos o *computador de baixo-custo com microcontrolador*. Essa é uma das configurações mais utilizadas atualmente, ganhando bastante espaço nos últimos anos e é baseada em placas programáveis como Arduino, Intel Galileo, etc. com computadores de baixo-custo como PIC, Raspberry Pi, etc. É o oposto da configuração anterior, já que é bem mais barata, porém requer um maior esforço na implementação (KALUZ *et al*, 2015).

Além do baixo-custo dessa arquitetura, temos a sua flexibilidade como uma de suas principais características, você pode criar diferentes tipos de experimentos e fazê-los funcionar através da implementação correta através da configuração e programação do computador e do microcontrolador.

Além disso, caso um experimento já não sirva mais para o seu propósito ou tenha ficado defasado, a simples reconfiguração da arquitetura permite que você atualize e modernize o atual laboratório ou simplesmente o substitua inteiramente, colocando uma atividade mais adequada em seu lugar, sem precisar recriar todo o ambiente novamente. A Figura 2.5 representa o funcionamento de um laboratório que utiliza essa arquitetura.

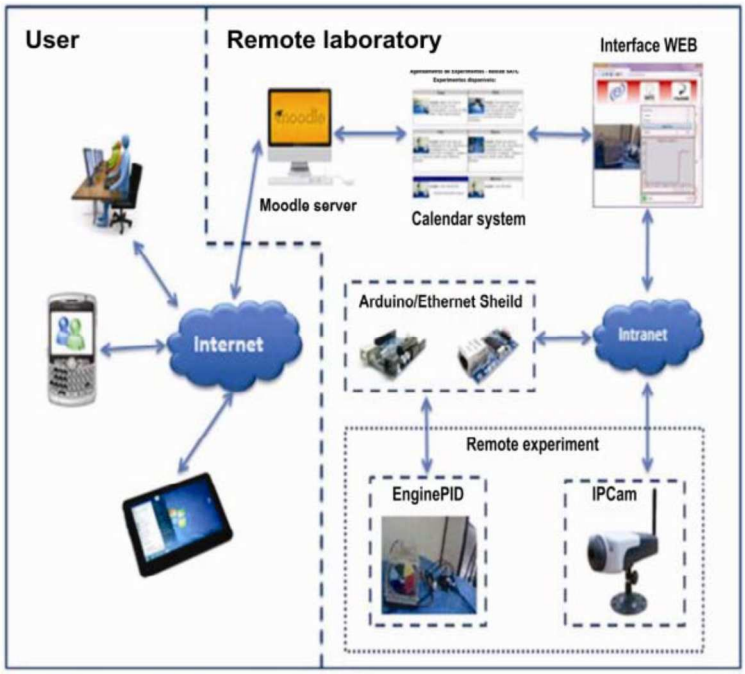

Figura 2.5 - Representação de um laboratório que utiliza um computador de baixo-custo e microcontrolador

FONTE: NETO *et al*, 2012, p. 2

2.3 Sistemas Embutidos

2.3.1 Definição

Um sistema embutido é um sistema computacional aplicado, bem distinto de outros sistemas do gênero como os computadores pessoais e supercomputadores. A verdadeira definição do termo é fluída e tem mais haver com o estado atual dos avanços da tecnologia, já que esse tipo de abordagem acompanha o *status quo* tecnológico e se aproveita da redução de custos que cada nova descoberta gera. Porém, existem algumas características que são comumente atribuídas aos sistemas embutidos e podem ser usadas para defini-los (NOERGAARD, 2012).

Os sistemas embutidos são mais limitados em termos de hardware e software, tendo uma capacidade de processamento menor, consumindo menos energia e memória além de possuir menos aplicações e nenhum sistema operacional específico. Também são mais específicos que suas contrapartes, já que você pode criar e configurar um sistema embutido para realizar apenas determinada tarefa ou função. Por fim, necessita de uma maior qualidade e confiança no seu

processo de design e programação, já que algumas de suas aplicações dependem muito de um resultado final preciso e confiável (NOERGAARD, 2012).

Alguns exemplos de sistemas embutidos presentes no dia-dia são o sistema de ignição de veículos automotores, controle do motor e freios de um carro, televisores (digitais ou analógicos), DVDs, micro-ondas, refrigeradores, GPS, câmeras, sistemas de manufaturas que envolvem o controle de robôs, máquinas de hemodiálise, monitores cardíacos, máquina de fax, impressoras e tantos outros espalhados por diversas áreas de mercado (NOERGAARD, 2012).

A Figura 2.6 traz a esquematização do funcionamento genérico de um sistema embutido.

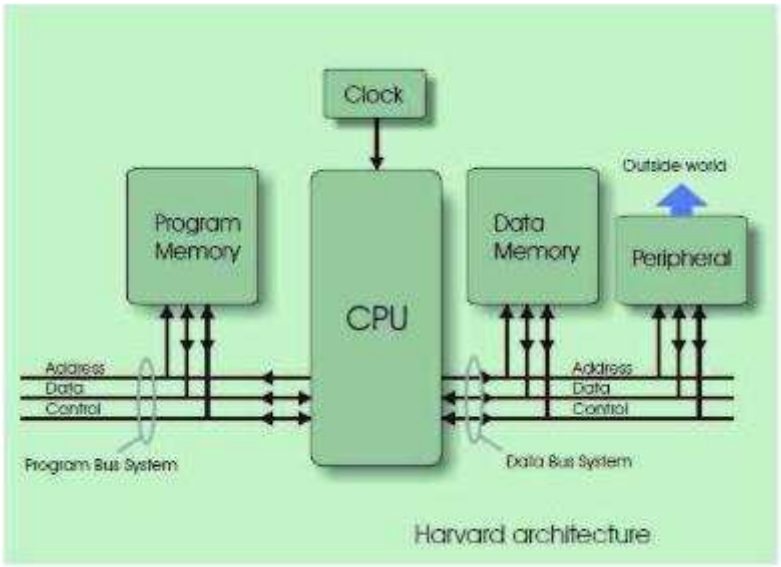

Figura 2.6 - Funcionamento de um sistema embutido

Disponível em http://www.ebah.com.br/content/ABAAAepngAJ/dsp-s. Acesso em 24/07/2017 às 16:19

2.3.2 Principais elementos de um Sistema Embutido

Em termos de hardware, os principais elementos são: a placa do circuito, a alimentação do sistema, o processador, a memória e os dispositivos de entrada e saída (I/O) e seus componentes elétricos, que variam de acordo com as necessidades e especificações do projeto.

2.3.2.1 Alimentação do sistema

Todo sistema eletrônico necessita de alimentação e normalmente, os sistemas embutidos são alimentados através da corrente elétrica, seja ela contínua ou alternada.

> Segundo Noergaard (2012, p. 87, tradução do autor) "O cálculo preciso da potência e energia tem que ser feito para determinar os requerimentos de consumo de todos elementos de determinado circuito. Isso ocorre porque cada um dos elementos pode suportar apenas um tipo de potência, portanto conversores Corrente Alternada (CA) - Corrente Contínua (CC), conversores CC-CA, conversores diretos CA-CA, podem ser necessários. Também, cada elemento tem uma capacidade limitada de potência necessária para funcionar, suportar ou dissipar. Esses cálculos determinam que a fonte de voltagem pode ser usada no circuito e o quão poderosa essa fonte precisará ser."

Ambos os tipos de correntes são utilizados, pois cada uma delas tem suas vantagens e desvantagens. Normalmente, circuitos de sistemas embutidos são conectados ou integrados com algum sistema de alimentação interno conectado a uma fonte externa, que fornece a alimentação necessária para o funcionamento ideal do circuito (NOERGAARD, 2012).

2.3.2.2 Unidade Central de Processamento

É a peça principal de um sistema embutido, sendo responsável por processar instruções e dados. Segundo Noergaard, (2008, p. 129, tradução do autor) "um equipamento eletrônico contém ao menos um processador *mestre*, atuando como o sistema de controle central, e em adição, pode ter processadores *escravos* que trabalham para, e são controlados, o processador *mestre*".

Todo e qualquer sistema embutido deve ser projetado em torno da unidade central de processamento. Para Noergaard, (2008, p. 130, tradução do autor) "a complexidade do processador *mestre* geralmente determina se ele será classificado como um *microprocessador* ou um *microcontrolador*".

A Figura 2.7 traz o exemplo de uma unidade central de processamento (CPU) comumente comercializada em lojas de informática e componentes eletrônicos.

Figura 2.7 - Unidade Central de Processamento (CPU)

Disponível em http://3.bp.blogspot.com/_kiXdoFEX3iQ/SsI_owbrBsI/AAAAAAAAAAc/Vtna8vWWTas/s1600-h/CPU_Processor_Intel_dual_core.jpg. Acesso em 247/07/2017 às 16:21

2.3.2.3 Memória

Um sistema embutido pode usar diversos tipos de memórias, cada uma com sua velocidade, tamanho e uso. Elas podem estar fisicamente integradas ao processador ou a placa. As memórias mais comuns são as *Read-Only Memory* (ROM) e *Random Access Memory* (RAM) (NOERGAARD, 2012).

2.3.2.3.1 MEMÓRIA ROM

Segundo Noergaard, (2008, p. 227, tradução do autor) “a memória ROM é um tipo de memória não-volátil que pode ser utilizada para armazenar dados num sistema embutido permanentemente. (...) O conteúdo da memória ROM tipicamente só pode ser *lido* pelo processador *mestre*; ”. Pelo fato de ser não-volátil, isso evita que os dados sejam perdidos caso a alimentação seja interrompida.

As memórias ROMs estão divididas em diversas categorias com as *Flash Memory*, *Mask Ready-Only Memory* (MROM), OTP, *Erasable Programmable ROM* (EPROM) e *Electrically Erasable Programmable ROM* (EEPROM) sendo as mais comuns. Durante todo o processo de criação e manufatura de um sistema embutidos, diversos tipos de memórias podem ser utilizados sem que elas necessariamente apareçam no produto final (NOERGAARD, 2012).

A Figura 2.8 traz o circuito lógico da memória ROM, enquanto a Figura 2.9 mostra o exemplo de uma memória ROM utilizada em circuitos embutidos.

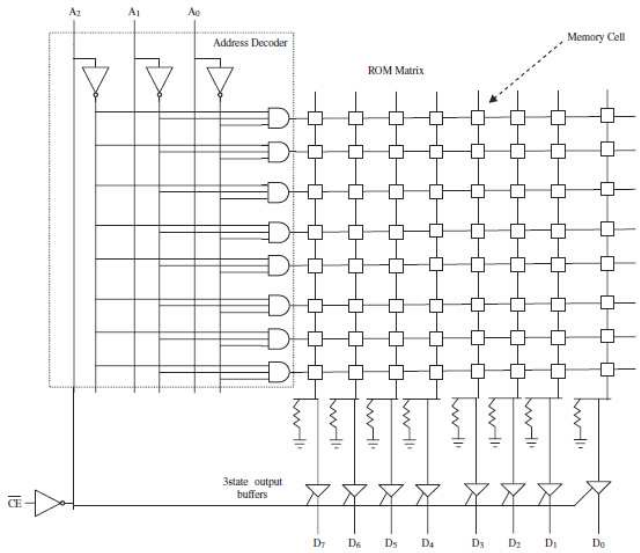

Figura 2.8 - Circuito lógico da Memória ROM

FONTE: NOERGAARD, 2012, p.228

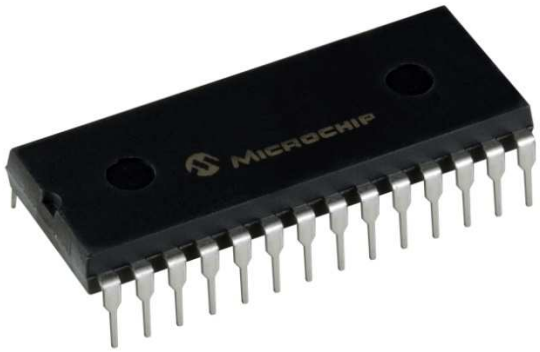

Figura 2.9 - Memória ROM utilizada em sistemas embutidos

Disponível em https://etcbinaria.files.wordpress.com/2011/01/eeprom.jpg. Acesso em 24/07/2017 às 16:36

2.3.2.3.2 MEMÓRIA RAM

É a memória principal e que permite que qualquer dado presente possa ser localizado diretamente e randomicamente, além de permitir que seu conteúdo seja modificado mais do que uma vez. Ao contrário das memórias ROM, que é uma memória não-volátil, seu conteúdo é apagado caso o circuito perca a alimentação. Os principais tipos de RAM são a *Static Random Access Memory* (SRAM) e *Dinamic Random Access Memory* (DRAM). A DRAM é mais confiável e é utilizada como memória principal, enquanto a SRAM é mais rápida e permite que certos tipos de processamento ocorram de maneira mais rápida e eficaz (NOERGAARD, 2012).

A Figura 2.10 traz o circuito lógico de uma DRAM, a Figura 2.11 traz o circuito lógico de uma SRAM e a Figura 2.12 traz exemplos de memórias RAM que são normalmente comercializadas.

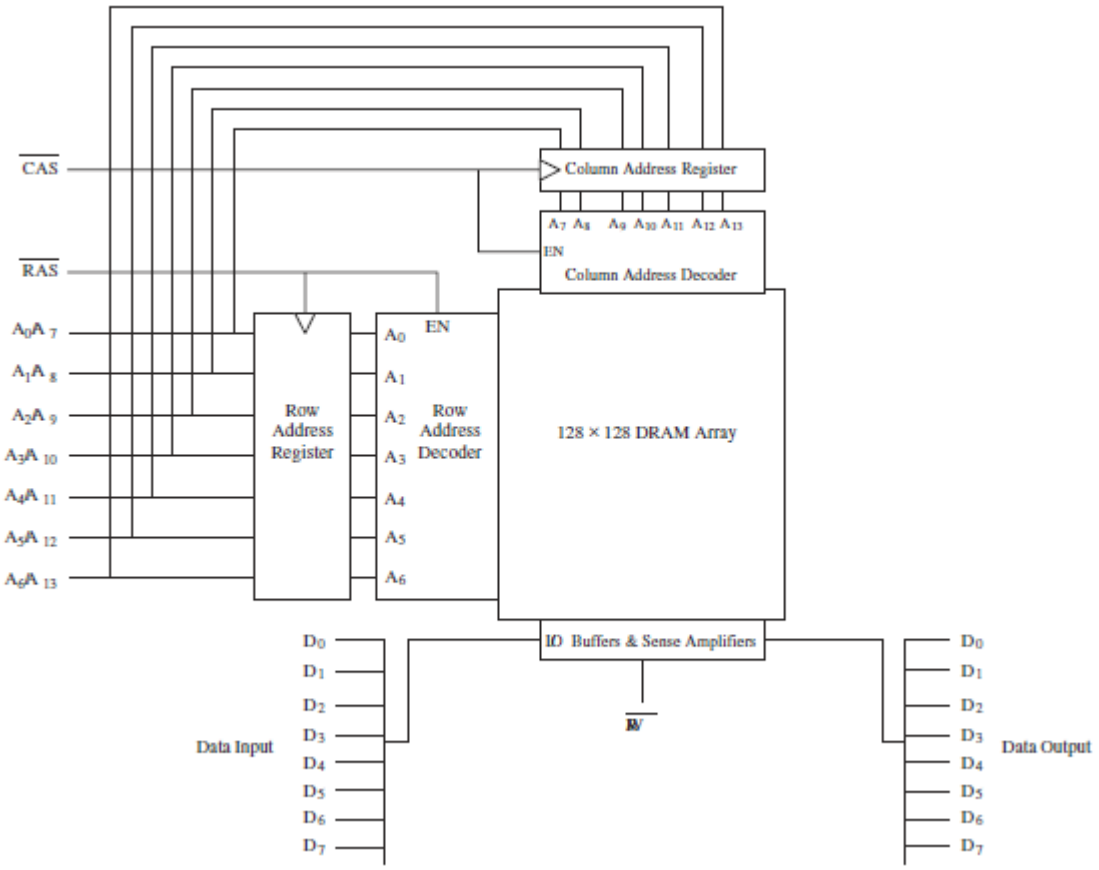

Figura 2.10 - Circuito lógico de uma DRAM

FONTE: NOERGAARD, 2012, p.236

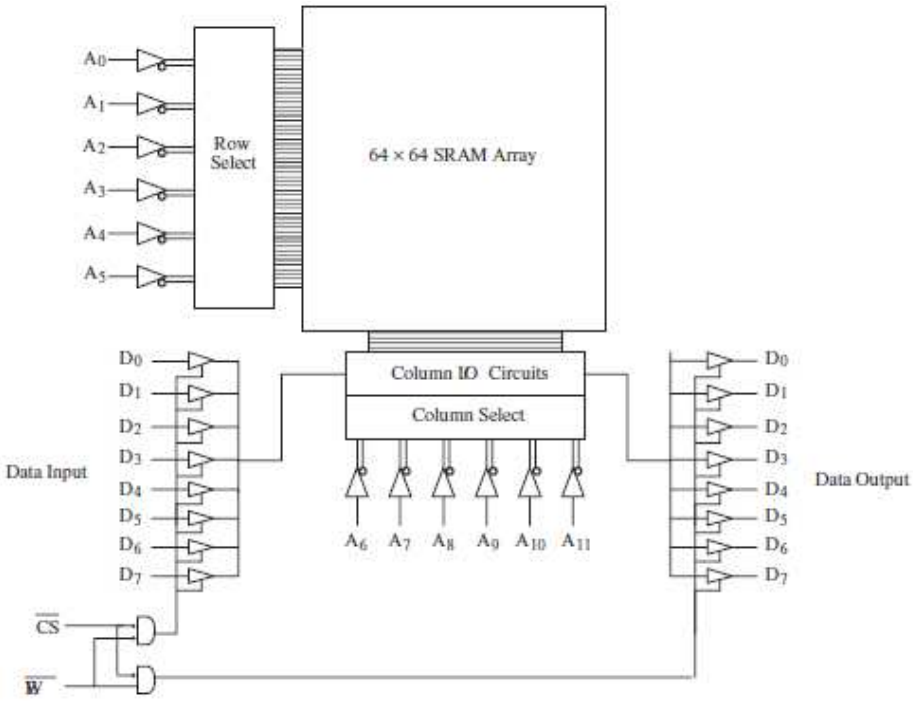

Figura 2.11 - Circuito lógico de uma SRAM

FONTE: NOERGAARD, 2012, p.233

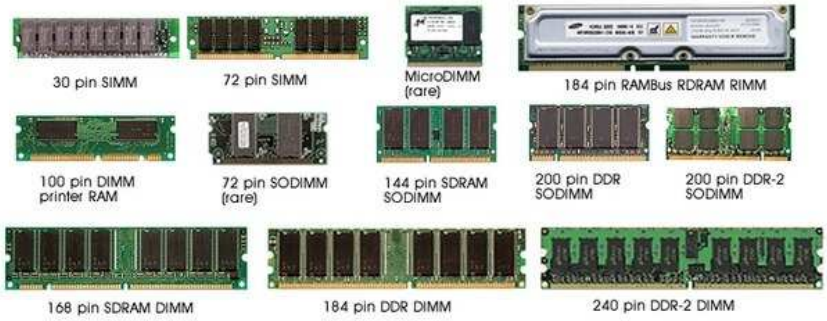

Figura 2.12 - Diversos tipos de memórias RAM comercializadas

Disponível em http://i0.statig.com.br/bancodeimagens/e8/jq/tv/e8jqtvnyp28ny85pipnedrvqh.jpg. Acesso em 24/07/2017 às 16:53

2.3.3 Dispositivos de Entrada e Saída (I/O)

São responsáveis por movimentar a informação para dentro e para fora da placa. Estão divididos em dispositivos de entrada (input), que levam informação para o processador ou dispositivos de saída (output), que levam a informação do processador para algum dispositivo externo (NOERGAARD, 2012).

Os dispositivos I/O estão divididos em diversas categorias, sendo as mais comuns as de comunicação e rede, input de equipamentos (como mouse e teclado), gráficos e equipamentos de output (por exemplo, *touch screen* e LEDs), dispositivos de armazenamento, dispositivos de *debugging* e dispositivos de tempo-real (NOERGAARD, 2012).

2.4 Rede TCP/IP

A utilização de uma rede TCP/IP é uma das formas de conexão de computadores em rede mais utilizadas. O *Transmission Control Protocol* (TCP) contém características de checagem de erros, controle de fluxo e identificação do nível de aplicação de um processo, seja no computador fonte ou no computador de destino (AXELSON, 2003).

> Segundo Axelson (2003, p. 7, tradução do autor), "O TCP realiza todas essas funções. Muitas comunicações via Internet e redes locais, tais quais pedidos por uma página web ou receber um e-mail, usam TCP. Windows e outros sistemas operacionais tem suporte nativo à TCP configurados. Kits de desenvolvimentos para sistemas embutidos com capacidade de conexão à rede muitas vezes contam com *libraries* ou pacotes com suporte ao TCP".

Já o *Internet Protocol* (IP) faz com que os dados que navegam pelas redes cheguem ao seu destino final, mesmo que tenham que passar por diversos outros tipos de conexões e redes. Apesar do nome sugerir que apenas quem tem conexão à rede mundial de computadores possa utilizar IP, é possível utilizá-los em conexões locais. Isso é permitido pois o IP atua em conjunto com o TCP, criando assim uma rede TCP/IP, que é uma junção de camadas de protocolos que formam a estrutura para a conexão à internet (AXELSON, 2003).

> "(...)TCP/IP é um conjunto de protocolos. Esse grupo é dividido em quatro camadas: aplicação, transporte, rede e interface. Cada uma delas é responsável pela execução de tarefas distintas. Essa divisão em camadas é uma forma de garantir a integridade dos dados que trafegam pela rede". (MARTINS, Elaine. *O que é TCP/IP?*)

2.4.1 A divisão das Camadas

Segundo Kurose e Ross (2012, p.49, tradução do autor) “Para providenciar estrutura para o design de protocolos de rede, designers de rede organizaram protocolos – e o hardware e software de rede para implementar esses protocolos – em camadas”. Cada protocolo necessário para o funcionamento da rede TCP/IP pertence a cada uma das camadas.

A divisão das camadas da rede TCP/IP é baseado no modelo do *Open Systems Interconnection* (OSI), criado pela unificação do modelo da *International Organization for Standardization* (ISO) e do *Consultative Committe for International Telephony and Telegraphy* (CCITT). Foi publicado pela primeira vez em 1984, com o objetivo de definir um padrão único para a arquitetura de sistemas de rede (KOSIEROK, 2005).

O OSI tinha como objetivo original ser a base que estabeleceria uma grande quantidade de variedade de protocolos, regulamentando e padronizando as comunicações internacionais inter-redes, o que no futuro viria se tornar a Internet (KOSIEROK, 2005).

> “Entretanto, as coisas não saíram conforme o planejado. O crescimento na popularidade da Internet e seu protocolo TCP/IP bateu de frente com o protocolo OSI, e resumindo, o TCP/IP venceu. Alguns dos protocolos do OSI foram implementados, mas no geral, o OSI acabou perdendo para o TCP/IP quando a Internet começou a crescer”. (KOSIEROK, 2005, p. 82)

Contudo, o OSI não desapareceu e acabou florescendo do ponto de vista educacional, já que era uma ferramenta perfeita para explicar o funcionamento e operação dos protocolos de redes e também para ensinar sobre redes em geral. Além disso, é muito útil para quem trabalha com o desenvolvimento de produtos hardware e software, já que ele demonstra claramente as tarefas e funções que cada camada realiza dentro de uma rede e seus sistemas integrados (KOSIEROK, 2005).

A Figura 2.13 demonstra a comparação entre a arquitetura de uma rede TCP/IP e o modelo de referência OSI, da ISO.

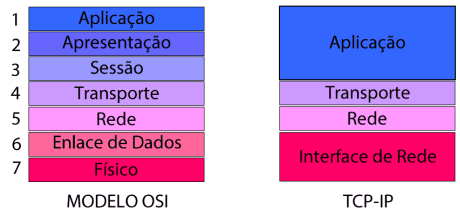

Figura 2.13 - Comparação entre o modelo OSI e TCP/IP

Disponível em https://www.citisystems.com.br/protocolo-tcp-ip/. Acesso em 25/07/2017

A camada de Aplicação é a camada superior do protocolo TCP/IP. É o local onde as aplicações de redes e as camadas dos protocolos de aplicação residem. Além disso, é essa camada que realiza a comunicação entre as duas pontas da conexão de rede, trocando dados e informações entre computadores, dispositivos ou qualquer equipamento capaz de conexões locais ou com a Internet.

A Camada de Transporte realiza a transmissão de dados da camada de transporte de um computador para outro, transformando as informações presentes na Camada de Aplicação em pequenos pacotes de dados que são transmitidos através da conexão existente. Além disso, é responsável por organizar essas informações para que o outro lado possa compreender o que está sendo transmitido. O protocolo TCP encontra-se nessa camada. A Figura 2.14 demonstra como a camada de Aplicação e de Transporte interagem.

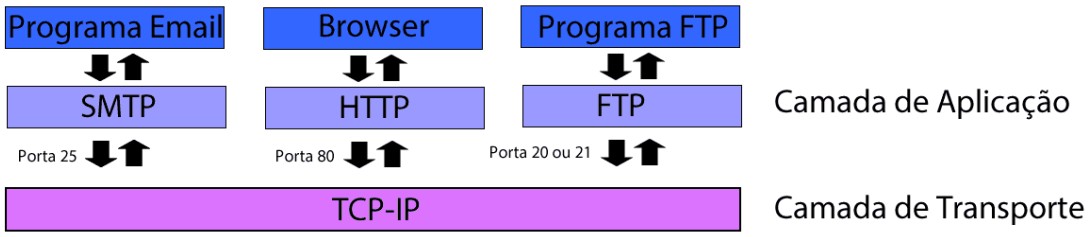

Figura 2.14 - Interação entre as camadas de Aplicação e Transporte

Disponível em https://www.citisystems.com.br/wp-content/uploads/2016/08/tcp-ip-protocolos.png. Acesso em 25/07/2017 às 10:37

A camada de Internet é onde se localiza o endereço virtual do equipamento, o endereço IP. Essa camada é responsável por identificar o caminho dos dados da comunicação entre dois dispositivos. Acaba funcionando como uma camada de identificação. A Figura 2.15 mostra como a Camada de Internet se relaciona com as camadas superiores.

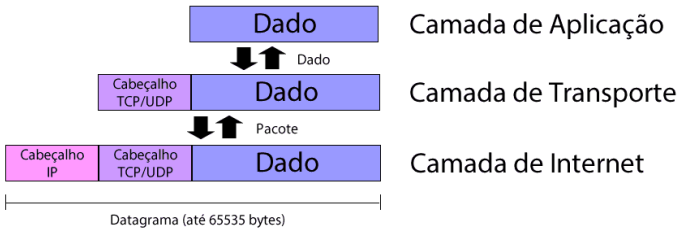

Figura 2.15 - Interação entre a Camada de Internet e as camadas superiores

Disponível em https://www.citisystems.com.br/wp-content/uploads/2016/08/tcp-ip-camada-internet.png. Acesso em 25/07/2014 às 10:54

Por fim, a Camada de Interface de Rede ou Camada Física. Ela é definida pela rede física (a mais tradicional é a rede Ethernet) em que seu equipamento está conectado. É ela que realiza a comunicação entre o computador com a Internet, interagindo com a Camada de Internet e verificando se existe uma conexão ativa para que a transmissão de dados possa ser realizada. A Figura 2.16 demonstra como todas as camadas do protocolo TCP/IP se comunicam.

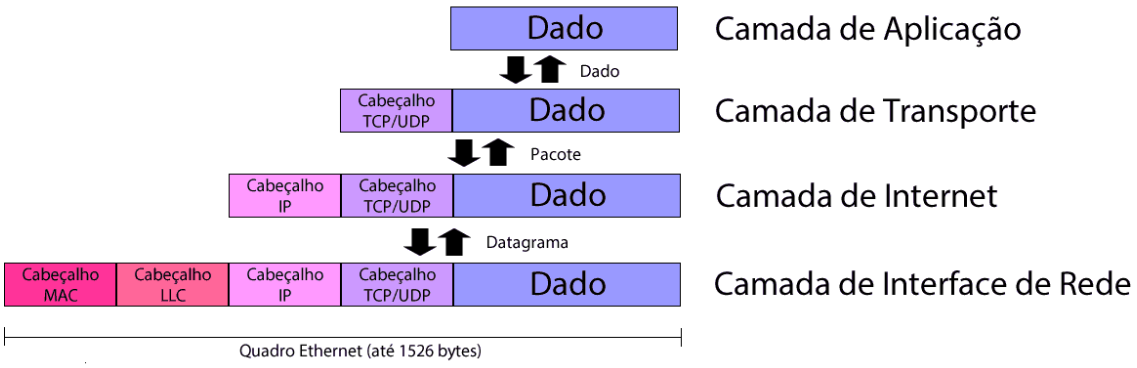

Figura 2.16 - A interação realizada entre as camadas TCP/IP

Disponível em https://www.citisystems.com.br/wp-content/uploads/2016/08/tcp-ip-ethernet-camadas.png. Acesso em 25/07/2017 às 10:59

2.4.2 Principais protocolos das camadas TCP/IP

Os principais protocolos da camada de Aplicação são o *HyperText Transfer Protocol* (HTTP), *Simple Mail Transfer Protocol* (SMTP), *File Transfer Protocol* (FTP), *Simple Network Transfer Protocol* (SNTP), *Domain Name System* (DNS) e *Telnet*.

O HTTP é responsável pelo transporte das páginas da internet para o navegador, o SMTP é o responsável pela transferência de e-mails pela internet, o FTP faz a transferência entre qualquer tipo de arquivos, o SNTP realiza o gerenciamento dos elementos de rede, de acordo com o tráfego de dados, o DNS é responsável pelo registro de domínios através das redes, permitindo o acesso a nomes mais discerníveis do que um IP e o Telnet é utilizada para criação de acesso remoto entre dois pontos de uma rede.

Outros protocolos da camada de Aplicação são o TFTP, NFS, SNMP, BOOTP, DHCP, BGP, EGP, IGP, RIP, OSPF, POP3 e IMAP4.

Além do procotolo TCP mencionado anteriormente, a camada de Transporte conta com o *User Datagram Protocol* (UDP), *Internet Control Protocol* (ICMP) e *Internet Group Management Protocol* (IGMP). O UDP é similar ao TCP, porém cria uma conexão não-confiável, e isso faz

com que seja pouco usado. O ICMP realiza o gerenciamento dos erros na comunicação entre dispositivos, enquanto o IGMP, possui suporte para mensagens *multicast* (transmissão para conexões variadas e específicas) e faz o rastreio desse *multicasting*.

O protocolo IP é o mais utilizado da camada de Internet, porém existem outros dois protocolos comumente utilizados. O *Address Resolution Protocol* (ARP) e o *Reverrse Address Resolution Protocol* (RARP). O ARP habilita a transmissão dos pacotes do computador em um pacote Ethernet, permitindo que o hardware se comunique com o endereço de IP. Já o RARP permite que um computador sem armazenamento de dados permanentes tenha um endereço de IP baseado no seu endereço Ethernet.

Por fim, na camada de Interface de Rede, além da Ethernet, como principal protocolo, temos, em adição, o *Serial Line Internet Protocol* (SLIP), *Compressed Serial Line Internet Protocol* (CSLIP) e o *Point-to-Point Protocol* (PPP). O SLIP realiza o transporte de dados em linhas seriais, é um protocolo sem correção de erros, endereçamento ou identificação de pacotes, utilizando apenas o transporte de dados via IP. O CSLIP é similar ao SLIP, porém os dados transmitidos são comprimidos, reduzindo o tamanho dos pacotes transmitidos. Já o PPP é uma melhoria ao SLIP, onde os dados seriais são encapsulados, fazendo com que a comunicação seja bidirecional.

2.5 Servidor

A comunicação via rede costumeiramente ocorre entre dois computadores ou dois equipamentos com acesso à rede. Essas duas entidades conectadas recebem o nome de cliente e servidor.

> “A maioria das aplicações de rede são projetadas para que um lado seja o cliente e o outro lado seja o servidor. O servidor provê algum tipo de serviço aos clientes, tal qual acesso aos arquivos no host do server. Podemos categorizar servidores em duas classes: iterativos e concorrente”. (FALL & STEVENS, 2011, p. 20, tradução do autor)

Os servidores iterativos esperam algum pedido através da rede feito pelo cliente, processa esse pedido e devolve a resposta esperada para o cliente, e o processo vai se repetindo continuamente, em um loop. Porém, se o servidor demorar demais para processar o pedido feito pelo cliente, isso gera problemas, pois durante esse longo processamento, caso algum outro cliente deseje acessar o servidor, ele não terá seus pedidos atendidos (FALL e STEVENS, 2011).

Já os servidores concorrentes esperam algum pedido feito pelo cliente chegar, iniciam uma nova instância dentro do servidor, que geralmente é a criação de um novo processo, tarefa ou thread. Portanto, isso faz com que cada cliente que acessar o servidor, terá seu "servidor particular". Com essa vantagem, a maioria dos servidores são concorrentes, pois evita-se a perda do tráfego do cliente para o servidor (FALL e STEVENS, 2011).

> "Observe que usamos os termos *cliente* e *servidor* para referir as aplicações e não a um sistema particular de computador onde eles rodam. Os mesmos termos algumas vezes são utilizados para se referirem a peças de hardware que normalmente executam tanto aplicações de cliente quanto aplicações de servidor. Apesar da terminologia ser de certa forma imprecisa, ela funciona suficientemente bem na prática. Como resultado, é comum encontrarmos um servidor (hardware) executando mais de um servidor (aplicação). " (FALL & STEVENS, 2011, p. 20, tradução do autor)

Em termos de hardware, o servidor é uma máquina que está sempre ligada, e conectada a uma rede (seja local ou Internet), executando sempre a mesma função. Para cada aplicação ou tarefa existe um tipo de servidor recomendado. Os mais comuns são servidores web, servidores de arquivos e servidores de impressão.

Apesar de parecer uma peça separada (e em alguns casos realmente é) no mundo da informática, o servidor pode ser um simples computador pessoal, desde que ele seja configurado com a instalação e configuração de softwares que propiciam a criação desse ambiente.

A configuração de computadores pessoais para servidores de acesso remoto é muito comum, já que para esse caso, apenas uma boa capacidade de processamento e memória bastam para termos um servidor confiável e seguro. Com as atuais tecnologias no ramo de processadores, onde a maioria das marcas trabalham com produtos com diversos núcleos, esse tipo de configuração se torna muito mais prática e barata do que a utilização de um hardware específico de servidores.

Na Figura 2.17 podemos ver a configuração de hardware interna de um servidor, enquanto a Figura 2.18 traz um rack de servidores utilizados para tarefas que exigem mais robustez e confiabilidade, além de uma capacidade de armazenamento e processamento superior à que um computador comum pode oferecer.

Figura 2.17 - Hardware interno de um servidor

Disponível em http://e.cdn-hardware.com.br/static/20101203/sr1500al-open-2.jpg.optimized.jpg. Acesso em 25/07/2017 às 16:19

Figura 2.18 - Racks de Servidores

Disponível em http://www.garratech.com.br/fotos/solucao-empresarial-virtualizacao-de-servidores-big.jpg. Acesso em 25/07/2017 às 16:21

2.6 Microsoft .NET

2.6.1 .NET Framework

Um framework é um conjunto de códigos, costumeiramente classes, que é voltado para alguma linguagem de programação, criando um relacionamento entre esses códigos, para disponibilizar funcionalidades para quem está desenvolvendo algum tipo de software baseado naquela linguagem de programação.

> "Em outras palavras, é como uma caixa de ferramentas, um kit que possui diversas funcionalidades devidamente implementadas, testadas e prontas para serem utilizadas na construção de softwares, poupando ao desenvolvedor tempo e trabalho na elaboração de operações básicas, como acesso a banco de dados, sistema de templates, mapeamento de rotas e validação de dados". (JAQUES, Rafael. *O que é um Framework? Para que serve?*)

Basicamente, o framework é um conjunto de classes que interagem entre si, é reutilizável, bem documentado, fácil de usar, extensível, seguro, eficiente e completo. Pois diferentemente de uma biblioteca de classes, o framework realiza uma interação entre essas classes, conforme comparação na Figura 2.19, facilitando o trabalho de desenvolvimento e programação de softwares.

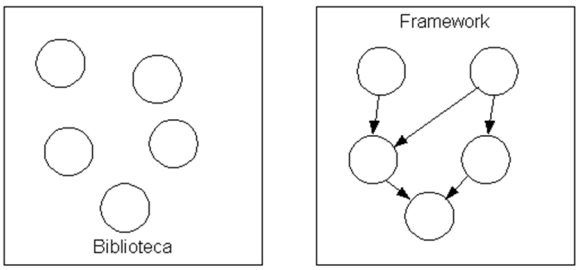

Figura 2.19 - Comparação entre biblioteca de classes e Framework

Disponível em http://3.bp.blogspot.com/_kiXdoFEX3iQ/SsI_owbrBsI/AAAAAAAAAAc/Vtna8vWWTas/s1600-h/CPU_Processor_Intel_dual_core.jpg. Acesso em 247/07/2017 às 16:21

Um framework reduz os custos e o tempo de preparação de um software, além de criar uma consistência interessante no mercado, gerando aplicações compatíveis. Porém, a construção de um framework é complexa e de longo prazo, além de precisar modificar todo o processo de desenvolvimento presente num ambiente para que se incentive o trabalho em um framework.

O .NET Framework, ou Microsoft .NET, é uma iniciativa da Microsoft, criado nos anos 2000, criado para facilitar a programação e desenvolvimento de aplicações para o Windows, principal produto da empresa. Antes de ser lançado, a Microsoft passou cerca de 4 anos trabalhando nele.

Como ferramentas de desenvolvimento criadas para esse framework constam o uso das linguagens de programação C# e VB.NET, o desenvolvimento de um ambiente de programação específico para esse framework, o Visual Studio, e uma vasta biblioteca de códigos para a elaboração de serviços web, aplicações web e aplicações Windows (THAI & LAM, 2003).

> "A Microsoft .NET é uma plataforma para desenvolvimento de serviços Web baseados em XML. Essa é, na verdade, uma definição muito simplista. Como plataforma, a .NET vai muito além de serviços Web. A Microsoft .NET vai permitir desenvolver qualquer tipo de aplicação usando a linguagem de sua preferência. C#, Visual Basic.NET, C++, Cobol, Perl, Fortran, Pascal são apenas algumas das

> linguagens suportadas na plataforma .NET, que não apenas permite o uso de múltiplas linguagens, mas também a completa e perfeita integração entre componentes desenvolvidos em linguagens diferentes". (LIMA & REIS, 2002)

A plataforma .NET é recomendada para o desenvolvimento de aplicações *front end*, aplicações de *middleware*, aplicações para a internet, aplicações gráficas, acesso a bancos de dados via ADO.NET e aplicações multitarefa (LIMA & REIS, 2002).

Além disso, ela é considerada como um ambiente de programação visual.

> "Outro tipo de linguagem de programação, a linguagem visual, é uma subcategoria das linguagens de programação imperativas As linguagens de programação mais populares são as linguagens .NET. Essas linguagens (ou suas implementações) incluem a capacidade de geração de segmentos de códigos através do método arrasta-e-solta. (...). As linguagens visuais oferecem uma maneira simples de gerar interfaces gráficas de usuários para programas". (SEBESTA, 2012, p.22, tradução do autor)

2.6.2 ASP.NET

O ASP.NET faz parte da plataforma .NET da Microsoft e foi criado para trazer o fácil desenvolvimento de aplicações Windows para aplicações web. Segundo Bochicchio et al (2011, p. 3, tradução do autor) "Se você pensa no .NET Framework como uma casa, então o ASP.NET são os cômodos".

Os programas desenvolvidos em ASP.NET são aplicações centralizadas presentes em um ou mais *Web Servers* interagindo dinamicamente com os dados do cliente. Normalmente, essa interação é estática, mas o ASP.NET interage com algumas extensões de arquivos (.aspx ou .ascx) e delega a responsabilidade da resposta para outros arquivos de código que são capazes de realizar essa interação dinamicamente (JONES, 2002).

Essa interação dinâmica entre cliente e servidor que o ASP.NET é capaz de prover, traz inúmeras possibilidades no desenvolvimento de aplicações web. O cliente entrará em contato com o seu *Web Server* e o programador pode decidir como essa interação irá ocorrer. Em uma interação estática, a relação cliente e servidor era em linha reta, o ASP.NET transforma essa linha em um caminho com diversas ramificações (JONES, 2002).

> "(...), quando a programação Web se tornou difundida, existiam poucas ferramentas para ajudar os programadores a escrever aplicações Web. Para criar uma aplicação Web, você tinha que comunicar escrevendo um código de comunicação socket de baixo-nível. Com o passar dos anos, o nível de abstração também aumentou para a programação Web. ASP.NET é a última (e provavelmente a melhor) dessas abstrações, porque lhe permite trabalhar quase que exclusivamente com diversas classes e objetos alto-nível ao invés de diretamente com dados puros. Sem o ASP.NET, desenvolver uma aplicação Web é uma tarefa árdua". (JONES, 2002, n.p.)

2.6.3 C#

Existe uma quantidade razoável de linguagens de programação disponíveis no cenário do desenvolvimento de aplicações e softwares. Então, por qual motivo a Microsoft, ao invés de trabalhar com uma das diversas opções, decidiu apostar numa nova linguagem programação, o C# (pronuncia-se C *sharp*)?

> "Este é um assunto que tem gerado uma ampla discussão não apenas no nível técnico ou de engenharia de software em si, como também no nível de mercado (afinal alguém tem de pagar a conta, não é?). Até certo ponto é fácil convencer pessoas técnicas (...) a usar uma nova linguagem ou tecnologia quando tecnicamente for provado que teremos ganhos consideráveis em relação ao que já existe no mercado, mas as implicações de se acrescentar uma nova linguagem ao parque tecnológico instalado numa corporação são séries. Afinal, será necessário investir em treinamentos e reciclagem de pessoal para essa nova linguagem, ou pior ainda, em contratação de mão de obra especializada que conheça essa nova linguagem". (LIMA & REIS, 2002, p. 16)

A razão dessa aposta está na plataforma .NET. Ela faz com que o desenvolvedor não tenha que mudar de linguagem de programação, fazendo com que o parque tecnológico possa migrar para a plataforma .NET sem grandes problemas. Mas isso ainda não responde qual o motivo da escolha por uma nova linguagem de programação (LIMA & REIS, 2002).

O C# em si só não é uma linguagem de programação extraordinária, que reinventou a roda, o seu ponto forte está no fato de reunir as melhores técnicas de desenvolvimento de programação nas principais linguagens do mercado em uma só. É uma linguagem clara, simples, fácil de aprender, completamente orientada a objetos, não requer ponteiros no gerenciamento de memória, suporta todas as características essenciais de uma linguagem de programação orientada a objetos e é 100 % reutilizável por qualquer outra linguagem de programação (LIMA & REIS, 2002).

A Figura 2.20 faz a demonstração de um exemplo de código "*Hello World*" (Olá Mundo), muito utilizado para introduzir os usuários em novas linguagens de programação.

```
1  using System;
2  class AppPontoNet
3  {
4      static void Main()
5      {
6          // escrevendo no console
7          Console.WriteLine("Olá mundo em C#");
8          Console.ReadLine();
9      }
```

Figura 2.20 - Código Hello World em C#

Fonte: LIMA & REIS, 2002, p.20, elaborado pelo autor no Visual Studio 2015

2.6.4 Internet Information Services

O *Internet Information Services* (IIS) foi desenvolvido pela Microsoft para ser o servidor web de seus sistemas operacionais para servidores. Foi introduzido no Windows NT Server e é responsável por oferecer um ambiente para hospedagem de sites, serviços e aplicativos, integrando diversas tecnologias como ASP.NET, FTP, PHP e WCF, além de contar com especificações tecnológicas próprias e voltadas para o uso com as aplicações Windows e da Microsoft. Foi especificamente desenvolvido para rodar as aplicações do Microsoft .NET e é o responsável por gerenciar o uso de protocolos de acesso à internet para as aplicações desenvolvidos na plataforma .NET.

2.7 Application Programming Interface

As APIs se tornaram uma grande parte do mundo conectado em que vivemos hoje, com muitas empresas investindo nesse novo ramo de negócios. Segundo Cooksey (2014, cap. 1, tradução do autor), “Em 2013 existiam mais de 10.000 APIS publicadas por empresas para consumo livre. Esse valor é quatro vezes maior que a quantidade disponível em 2010". Esse crescimento ao longo dos anos pode ser visto na Figura 2.21.

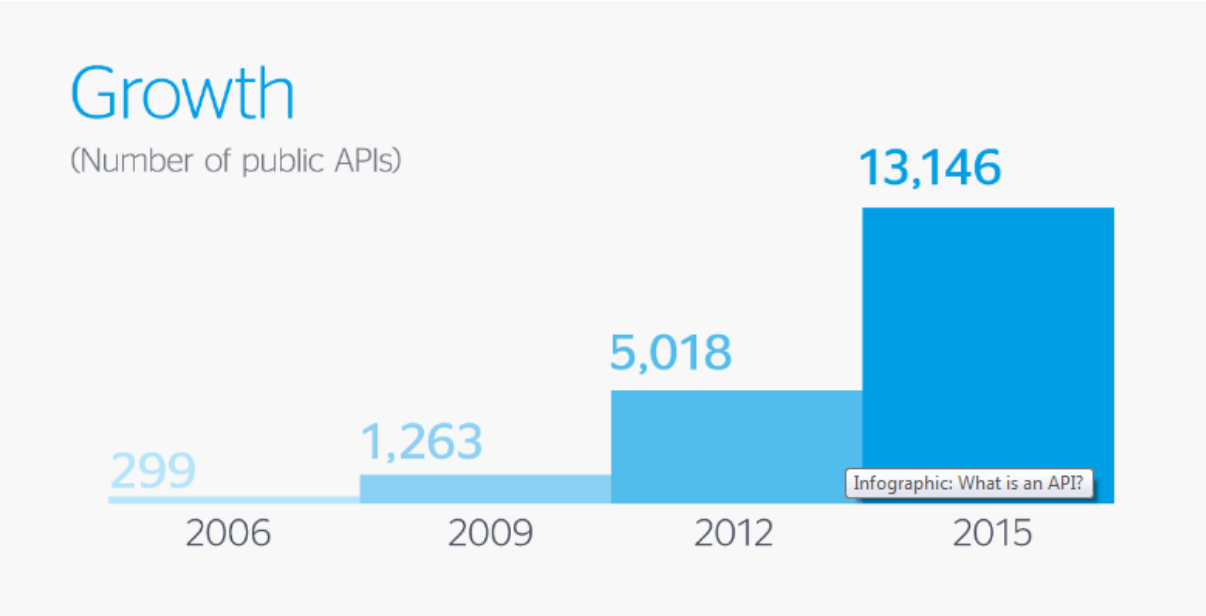

Figura 2.21 - Crescimento do número de APIs na última década

Disponível em https://bbvaopen4u.com/en/actualidad/infographic-what-api. Acessado em 27/07/2017 às 09:37

As APIs são ferramentas que transformam a capacidade de um computador interagir com dados disponíveis na rede, principalmente em sites. Elas permitem que um computador possa ver e editar dados de um site como uma pessoa normal. Como sites são desenhados para facilitar a interação do usuário com as informações disponíveis, os computadores têm dificuldade de utilizar os mesmos dessa maneira.

> "Seres humanos tem uma incrível capacidade de pegar informação visual, combinar com suas experiências e atribuir significado, para então agir sobre o que aquilo significa. É por isso que você pode olhar em um formulário em um site e saber que a pequena caixa com a frase "Primeiro Nome" em cima significa que você tem que digitar uma palavra que você usa para se identificar.
>
> Ainda, o que ocorre quando você tem uma tarefa que gasta bastante tempo, como copiar a informação de contatos de mil clientes de um site para outro? Você adoraria delegar esse trabalho para um computador para que ele seja feito rapidamente e precisamente. Infelizmente, as características que fazem os sites otimizados para seres humanos também o fazem complicados para o computador utilizar. A solução é uma API". (COOKSEY, 2014, cap. 1)

Diversos setores já se beneficiam do seu uso, desde pequenas companhias até as grandes e inclusive as administrações públicas. Como grandes exemplos de setores que as utilizam temos as redes sociais (Facebook, Instagram, Twitter, etc.), aplicativos de geolocalização (Google Maps, Foursquare, etc.), o setor financeiro (Aplicativos de Bancos, Paypal, etc.) e o setor de *e-commerce* (Amazon, eBay, etc) (BBVAOPEN4U, 2015).

Pode parecer que as APIs são aplicações e aplicativos para celular, mas na realidade, elas são aplicações Web que são instaladas em um server para realizar determinadas funções e/ou

tarefas. Segundo Cooksey, (2014, cap.1, tradução do autor) “Ajuda lembrar que uma API é simplesmente mais um programa rodando dentro de um servidor”.

Normalmente, quando um usuário faz um pedido ao servidor, durante a interação, sabe-se o que está disponível através da API (ou das APIs) presentes naquele servidor e manipula-se as informações presentes para atender ao pedido realizado pelo usuário (COOKSEY, 2014).

Para Cooksey (2014, cap.1, tradução do autor), “Um grande exemplo é um aplicativo de smartphone que sincroniza com um site. Quando você aperta o botão atualizar no seu aplicativo, ele conversa com o servidor via API e recolhe a informação mais nova”. Na Figura 2.22, podemos ver como ocorre a interação entre cliente e servidor quando uma API está presente.

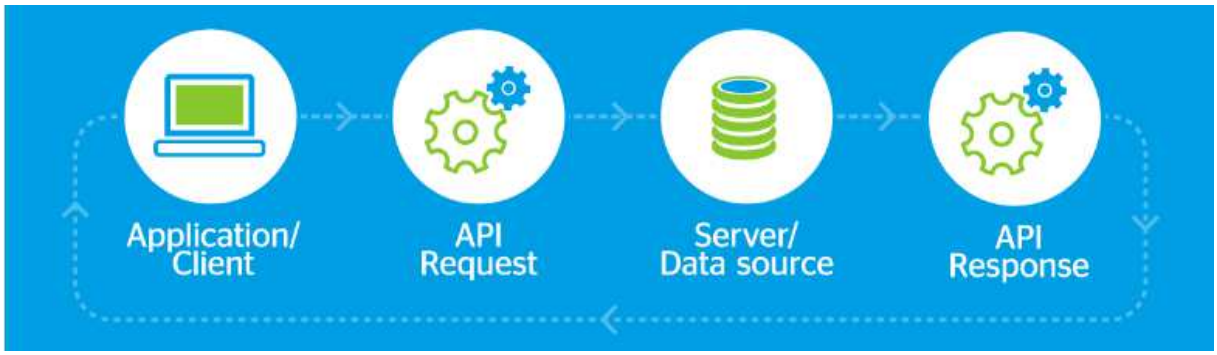

Figura 2.22 - O funcionamento de uma API

Disponível em https://bbvaopen4u.com/en/actualidad/infographic-what-api. Acessado em 27/07/2017 às 10:00

Elas são divididas em duas classes: SOAP (Simple Object Access Protocol) e REST (Representational State Transfer). A SOAP é um protocolo que define como dois objetos diferentes podem se comunicar através da troca de dados via XML. Já a REST é uma maneira simples de receber e enviar dados entre cliente e servidor.

2.8 Laboratório Remoto da Escola de Minas

A bancada com o experimento de Dois Tanques Acoplados desenvolvido no Laboratório de Máquinas Elétricas da Escola de Minas, da UFOP, foi desenvolvida com base no projeto de Tanques Acoplados da *Quanser*, uma empresa canadense focada no desenvolvimento de equipamentos e sistemas de controle voltado para o ensino e pesquisa na área de Engenharia (QUANSER, 2017).

Na Figura 2.23 podemos observar o modelo mais recente desenvolvido pela empresa, enquanto na Figura 2.24 encontra-se a bancada desenvolvida na UFOP.

Figura 2.23 - Sistema de Tanques Acoplados da Quanser

FONTE: QUANSER, 2013, p. 1

Figura 2.24 - Sistema de Tanques Acoplados da UFOP

FONTE: OLIVEIRA, 2017, p.17

A bancada utilizada nesse projeto foi desenvolvida por estudantes dos cursos de Engenharia da Escola de Minas, da UFOP. Ela possui 4 tanques com dimensões de 4,76 cm de diâmetro interno por 25 cm de altura. Essa configuração permite que sejam realizados experimentos com 2 ou 4 tanques acoplados. Além disso, conta com duas bombas motores 12V, que fornecem o fluxo de água para o sistema, enquanto a pressão é medida por quatro sensores MPXM2010 (OLIVEIRA, 2017).

A configuração atual permite que os discentes do curso de Engenharia de Controle realizem atividades práticas na área de Teoria de Controle, podendo visualizar um sistema de controle de nível funcionando e também entender como funcionar os diferentes tipos de controladores. Além de ter um contato físico com tecnologias que podem ser encontradas no mercado de trabalho (OLIVEIRA, 2017).

Dependendo da configuração escolhida, dois ou quatro tanques, a bancada se transforma em um sistema monovariável com ou sem acoplamento ou um sistema multivariável com fase mínima ou não (OLIVEIRA, 2017).

2.9 WebLab-Deusto

A Faculdade de Engenharia da *Universidade de Deusto*, localizada em Bilbao, Espanha, possui, desde o ano de 2001, laboratórios remotos para o ensino de microeletrônica. Desde então, a área de laboratórios remotos é bastante ativa na universidade. Os *WebLabs*, como são chamados pela comunidade acadêmica de lá, se tornaram referência na área de ensino de microeletrônica, e muitas universidades seguiram o exemplo dos espanhóis (GARCÍA-ZUBIA *et al*, 2006).

Para os estudantes, o WebLab-Deusto se mostrou uma ferramenta útil, fácil de se utilizar, que traz o controle da atividade prática e a imersão do estudante num ambiente laboratorial (GARCÍA-ZUBIA *et al*, 2011).

O sistema conta com autenticação de usuário, o gerenciamento de permissões para o usuários utilizarem os experimentos (podendo liberar experimentos de acordo com o interesse de cada professor, aluno, curso, etc.), uma extensa variedade de ferramentas administrativas que melhoram a experiência de usuários e administradores, a capacidade de gerenciamento de filas (permitindo que apenas um estudante por vez acesse o experimento), a capacidade de observar o comportamento de cada estudante durante seu acesso ao experimento, além do fato de que cada nova melhoria no sistema ser automaticamente aplicada para todos os experimentos presentes no servidor do WebLab-Deusto (ORDUÑA *et al*, 2011).

O WebLab-Deusto começou como uma solução proprietária baseada em *Socket* e *Applet*, desenvolvido em linguagem C. O laboratório remoto se comunicava com o server do WebLab através de um Socket BSD. Acabou sendo utilizado apenas para algumas aulas e alguns estudantes convidados (GARCÍA-ZUBIA *et al*, 2006).

Posteriormente se tornou uma solução baseada na Web. A plataforma era composta de um servidor na linguagem Python, comunicando com um PIC que controlava um PLC, enquanto uma *webcam* transmitia as imagens capturadas. O grande problema dessa solução era a segurança. O próximo passo foi se tornar uma solução Web baseada em AJAX, ele incorporava o funcionamento da visão anterior e resolvia o problema de segurança da mesma (GARCÍA-ZUBIA *et al*, 2006).

Depois, todo o progresso realizado anteriormente foi transportado para uma solução Web baseada em AJAX e microservidores, permitindo ao WebLab-Deusto ser acessado por uma grande quantidade de usuários e que esses usuários acessassem diversos tipos de laboratórios integrados ao sistema. Essa configuração substituiu a comunicação física pela comunicação via internet. A Figura 2.25 demonstra como funciona essa arquitetura atualmente. (GARCÍA-ZUBIA *et al*, 2006)

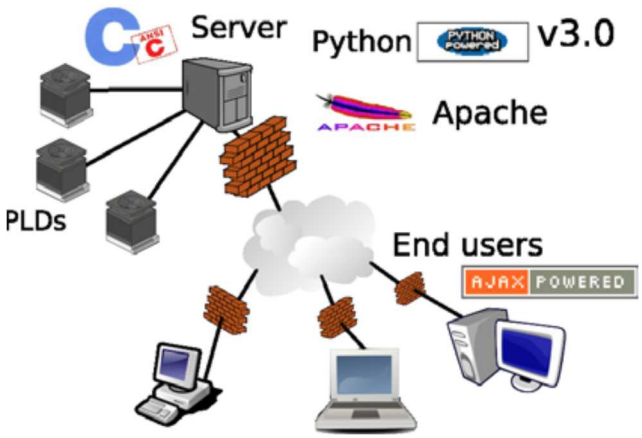

Figura 2.25 - WebLab-Deusto como solução Web baseada em Ajax e microservidores

FONTE: GARCÍA-ZUBIA et al, 2006, cap. 2

Atualmente, o WebLab-Deusto é um sistema de gerenciamento de código aberto para laboratórios remotos. Segundo Kaluz *et al* (2013, p 346, tradução do autor) “tornou-se um sistema de laboratórios remotos onde diversos laboratórios poderiam ser desenvolvidos usando diferentes tecnologias de software”.

Baseado em uma arquitetura cliente/servidor, o usuário que utiliza o WebLab-Deusto irá passar por um server de *login*, para autenticar seu acesso e um outro servidor que irá servir como base de operação para todas as outras atividades realizadas pelo usuário. Esse outro servidor é o núcleo do funcionamento (KALUZ *et al*, 2013).

> "Por trás do Servidor Núcleo, a arquitetura do WebLab-Deusto define mais dois servidores: o servidor Laboratório e o servidor Experimento. O primeiro busca localizar espaço laboratório físico onde múltiplos servidores Experimento são esperados. (...). O servidor Experimento é um servidor que interage fisicamente com o hardware". (KALUZ *et al*, 2013, p. 346, tradução do autor)

Em termos de desenvolvimento, o WebLab-Deusto pode ser desenvolvido de duas maneiras: *managed* e u*nmanaged*.

Os experimentos de categoria *managed* abraçam o claro conceito e funcionamento da arquitetura cliente/servidor. Dessa maneira, o WebLab-Deusto fornece estrutura (em termos de software) para que o cliente e o servidor possam se comunicar tranquilamente, além de fornecer ferramentas intermediárias para essa comunicação, assegurando que não ocorrerão acessos desautorizados (ORDUÑA *et al*, 2011).

Todo o processo que ocorre dentro do experimento *managed* é controlado pelos servidores do WebLab-Deusto, que funcionam como intermediários entre o cliente e o servidor. Para o desenvolvimento de aplicações nessa área, o próprio sistema oferece soluções tanto para o lado do cliente quanto para o lado do servidor (ORDUÑA *et al*, 2011).

Porém, como é necessário o desenvolvimento de códigos para realizar a integração entre usuário, WebLab-Deusto e servidor, muitos experimentos que poderiam ser transmutados para o ramo dos laboratórios remotos acabariam não tendo essa disponibilidade pois pode ocorrer de quem desenvolveu os experimentos físicos não ser capaz de programar ou não ter experiência no desenvolvimento de aplicações Web (ORDUÑA *et al*, 2011).

Para isso, foram desenvolvidos os experimentos *unmanaged*. Nesse caso, o WebLab-Deusto deixa de ser o intermediário que gerencia toda a operação e passa a funcionar como um sistema de gerenciamento de acesso e rastreio e armazenamento de dados do experimento (ORDUÑA *et al*, 2011).

Apesar de auxiliar nessa integração, essa categoria traz desvantagens com relação aos experimentos *managed*, já que essa é uma forma mais segura de acesso e tem uma facilidade melhor no controle de acesso e gerenciamento de informações dos experimentos. A sua grande

vantagem é a capacidade de integrar qualquer tipo de tecnologia que a versão *managed*, fazendo com que o experimento possa ser adaptado a qualquer tipo de tecnologia disponível (ORDUÑA *et al*, 2011).

As Figuras 2.26 e 2.27 trazem a comparação entre um laboratório *managed* e um laboratório *umanaged*, mostrando os caminhos que ocorrem entre o momento que o usuário acessa o WebLab-Deusto e o caminho que a informação faz até chegar no experimento.

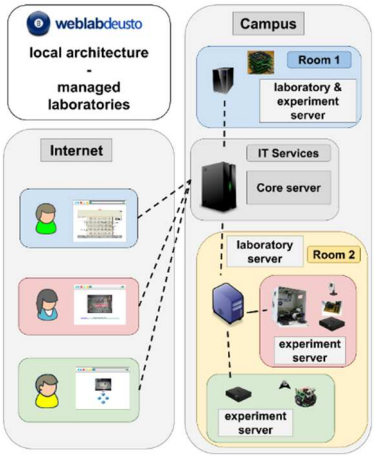

Figura 2.26 - Arquitetura local de um laboratório *managed*

Disponível em https://weblabdeusto.readthedocs.io/en/latest/remote_lab_development.html. Acesso em 28/07/2017 às 09:54

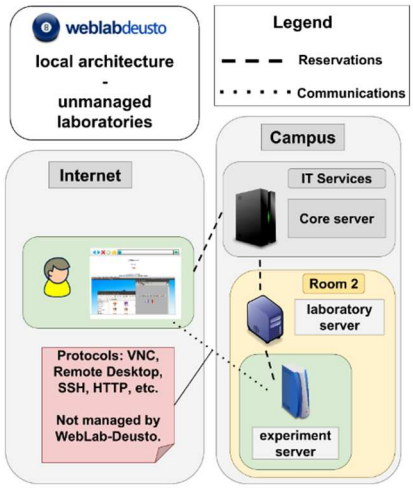

Figura 2.27 - Arquitetura local de um laboratório *unmanaged*

Disponível em https://weblabdeusto.readthedocs.io/en/latest/remote_lab_development.html. Acesso em 28/07/2017 às 10:35

3. Desenvolvimento do Projeto

3.1 Preparando o ambiente para o WebLab-Deusto

O WebLab-Deusto pode ser obtido de duas formas, através de um *GitHub*, ou fazendo o download da aplicação direto do *web browser*. A documentação da plataforma recomenda a utilização da primeira opção. Além disso, alguns pré-requisitos são necessários para o perfeito funcionamento do *Web Server*. Primeiramente, o computador alvo deve contar com uma instalação do Python 2.7, para que seja possível dar prosseguimento à configuração. Já existe uma versão mais avançada do Python, porém, a plataforma é incompatível com ela.

Existem versões do WebLab-Deusto tanto para Linux, Mac OS X e Windows. Para o desenvolvimento desse projeto, como a aplicação web do LabREM foi construída utilizando tecnologias Windows, a versão para Windows foi selecionada pela facilidade em adequar estruturalmente as duas entidades. Os desenvolvedores da plataforma recomendam a utilização da versão Linux, mas, não existem grandes perdas entre as versões de sistemas operacionais diferentes.

O Python é uma linguagem de programação que surgiu em 1991. Seu principal mote é ser uma linguagem de fácil aprendizagem e poderosa capacidade de programação. Com uma sintaxe elegante e dinâmica, é ideal para programadores que desejam um desenvolvimento mais rápido em suas aplicações. Atualmente é muito visada na área de análise de dados e aprendizado de máquinas (PYTHON E PYTHON BRASIL, 2017).

Após a instalação dessas duas ferramentas, deve ser criado um ambiente virtual para que se opere o *Web Server*, essa virtualização é uma característica do Python. Esse ambiente virtual é criado através da inserção de comandos no *cmd* que a instalação via *GitHub* lhe fornece. O ambiente é bastante similar ao Windows, conforme comparação na Figura 3.1, mas é voltado para realizar a interface entre o que foi instalado através do *GitHub* e o computador.

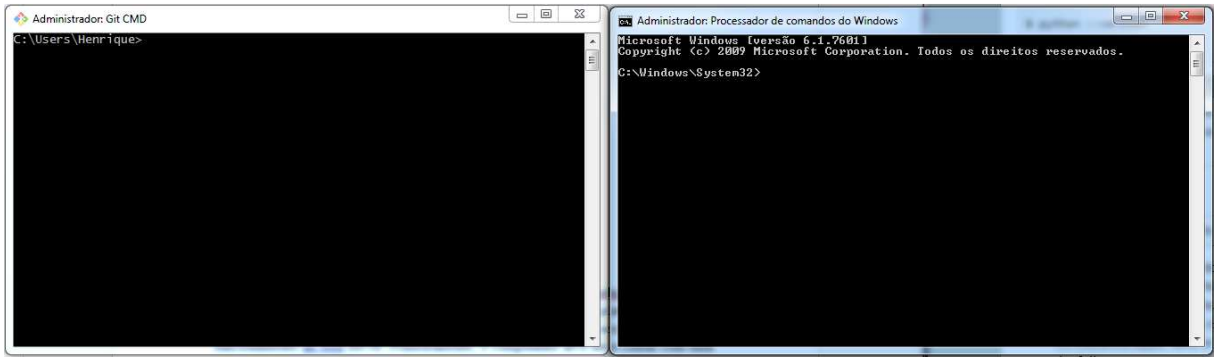

Figura 3.1 - Comparação entre o cmd do GitHub (esquerda) e do Windows 7 (direita)

Fonte: *print screen* adaptado das aplicações no sistema operacional Windows 7

A criação do ambiente virtual necessita que algumas aplicações Python estejam presentes no computador, portanto, é necessário que se instale também as aplicações *setuptools*, que pode ser obtida através de download e as aplicações *virtualenv* e *virtualenvwrapper*, que podem ser obtidas inserindo comandos no *cmd* do *GitHub*.

Após a instalação dos componentes previamente requeridos, pode-se partir para a instalação do WebLab-Deusto. Assim como a instalação de alguns recursos foi realizada no *cmd* do *GitHub*, todo o procedimento de instalação da plataforma é realizado pelo mesmo *cmd*.

Quando realizada a instalação via *GitHub*, é instalada uma pasta no disco rígido do sistema operacional e é importante tomar nota do diretório dessa instalação, pois é relevante para a correta configuração do *Web Server*. O primeiro passo é criar um *virtualenv*, que é um comando que designa um ambiente virtual para a inicialização da plataforma no sistema.

Posteriormente, é necessário checar se as versões instaladas do *setuptools* e *pip* são as mais atuais e eventualmente atualizá-los, caso necessário. Após isso, pode ser dado o comando para a instalação do WebLab-Deusto.

A instalação é realizada através da sequência de comandos abaixo:

pip install virtualenv virtualenvwrapper

cd weblabdeustofolder
mkvirtualenv weblab

pip install –upgrade setuptools
pip install –upgrade pip

python setup.py install

3.2 A instalação de uma instância do WebLab-Deusto

3.2.1 Instância básica

Um computador pode rodar múltiplas instâncias do WebLab-Deusto, o que é muito útil para você testar a configuração e encontrar a opção ideal para sua necessidade. Cada instância é como se fosse uma entidade separada, mesmo estando no mesmo computador. É como se o WebLab-Deusto fosse uma escola e cada instância uma sala de aula.

A primeira coisa a se fazer é criar a instância e selecionar uma porta para o servidor utilizar, para um primeiro teste, é meio irrelevante o número da porta, porém, em configurações avançadas, alguns programas e as recomendações de portas mais seguras devem ser levadas em consideração para a seleção das portas utilizadas na criação do ambiente.

Depois de criada, a instância deve ser iniciada.

Dessa forma, o server estará rodando a partir daquele instante e para fechá-lo, um simples comando encerra as atividades, a tecla *Enter*. Para acessar a plataforma basta inserir no *web browser*, o endereço: http://localhost:númerodaporta/. Esse endereço eletrônico direciona para a tela de *login* do WebLab-Deusto. De forma padrão, qualquer usuário acessa com poderes de administrador ao inserir o *login admin* e a senha *password*.

Após o acesso realizado, alguns laboratórios ficam disponíveis para teste e ambientação. Existe uma opção de laboratório local (*dummy*) e outra opção localizada na Universidade de Deusto, como laboratório exemplo. A Figura 3.2 demonstra a página que se abre após a realização de acesso ao serviço.

Figura 3.2 – Página inicial do *web server*

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Para criar e rodar essa instância básica na porta 8000, a sequência de comandos a serem inseridas no *cmd* do *GitHub* é a seguinte:

weblab-admin create example –http-server-port=8000

weblab-admin start example

3.2.2 Configurando uma instância mais avançada

Com a versão mais simples possível do WebLab-Deusto rodando, é hora de evoluir e trabalhar de maneira mais completa com a ferramenta. Como a opção básica trabalha com o SQLite como *database* e mecanismo de escalonamento de processos, isso faz que apenas um processo possa rodar na máquina e que os processos sejam mais lentos. Além disso, o server HTTP que está sendo utilizado não é real, é embutido, e ele foi feito apenas para testes.

Para amplificar as capacidades do WebLab-Deusto, alguns sistemas externos devem ser instalados. O Apache HTTP server, o MySQL e o Redis são os sistemas recomendados pelos desenvolvedores para aprimorar a experiência. O Apache substitui o servidor HTTP embutido presente na instância básica, que é simples e programado em Python.

O MySQL substitui o SQLite, que é útil em mecanismo de baixo-custo, como o Raspberry Pi, porém, em aplicações mais robustas, como desktops e servers, não é uma ideia interessante. Como o objetivo é atingir uma grande quantidade de pessoas, um sistema de base de dados mais complexos é necessário para que as coisas funcionem de maneira adequada.

O Redis é um sistema de escalonamento de processos, mais utilizado em aplicações no MAC OS X e Linux, mas em um futuro breve, passará a ser importante também em aplicações Windows. Atualmente, é opcional na hora de configurarmos o WebLab-Deusto em qualquer sistema operacional da Microsoft.

Tanto o MySQL quanto o Apache podem ser instalados através de softwares e em sua grande maioria, eles vêm juntos. O XAMPP é a opção recomendada pelos desenvolvedores, mas qualquer aplicação do mesmo estilo tem o mesmo efeito. Como o LabREM foi desenvolvido no WampServer, o WampServer foi selecionado para a implementação do WebLab-Deusto, visando facilitar a integração dos sistemas.

Para realizar a configuração dessas opções mais avançadas, devemos recorrer a criação de uma nova instância, portanto, a seguinte sequência de comandos deve ser inserida no *cmd* do *GitHub*:

Weblab-admin create novoexemplo –coordination-db-engine=mysql –db-engine=mysql

3.2.3 O ambiente interno do WebLab-Deusto

Ao acessarmos com a conta de administrador padrão da plataforma, podemos criar usuários e gerenciar permissões que permitam direcionar o acesso e rastrear o uso das informações dos usuários dentro do laboratório remoto. A página inicial do administrador, conforme Figura 3.3, já traz algumas indicações dos últimos usuários que usaram o laboratório, porém, como acabamos de criar a instância, não existem esses dados. Para acessá-la, basta clicar ícone da chave de fenda e chave inglesa cruzadas, no canto superior direito da tela, na página inicial, representada na Figura 3.2.

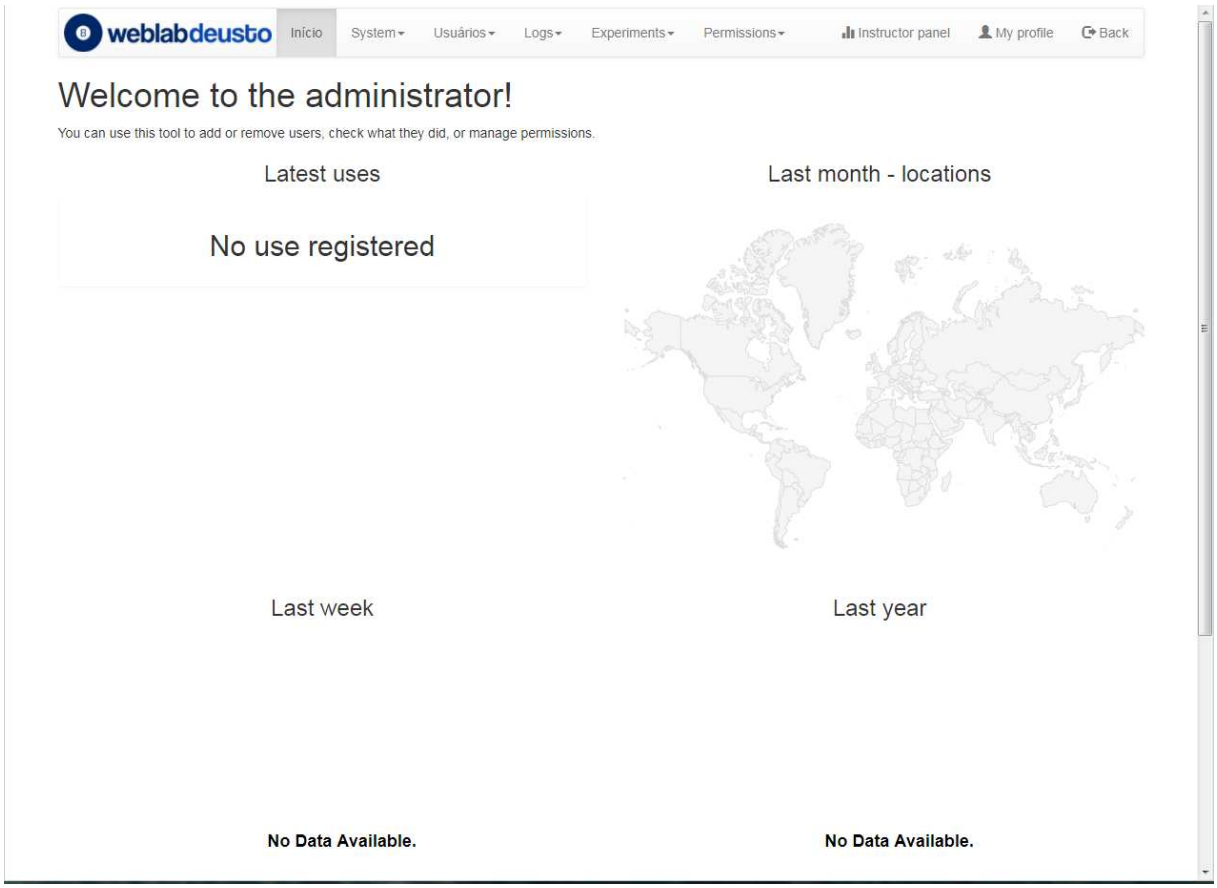

Figura 3.3 – Página inicial do Administrador do WebLab-Deusto

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

No canto superior direito, existe a opção de acessar o painel de controle do administrador e dentro dele, diversas operações estão disponíveis para configuração. O primeiro passo é adicionar um novo usuário. Para isso, basta acessarmos a opção *Usuários* e clicarmos em *Usuários*. Essa opção irá listar todos os usuários registrados naquele *Web Server*, obviamente,

como é o primeiro acesso, apenas o administrador estará listado, como pode ser visto na Figura 3.4.

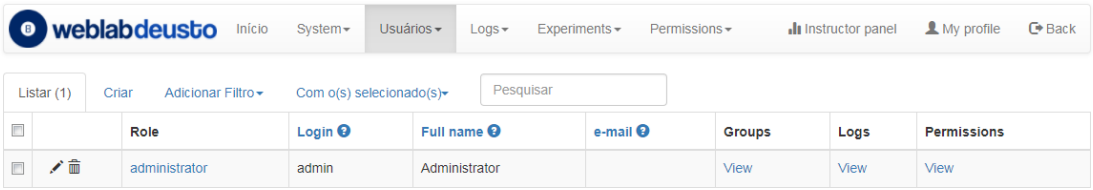

Figura 3.4 – Painel de usuários no *Web Server* criado

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Para adicionar um novo usuário, basta clicar na opção *Criar* e será gerado um formulário, nele existem as opções de definir a função do novo usuário, que se dividem em 4 categorias: *federated*, *administrator*, *instructor* e *student*. Também selecionamos a opção de qual grupo de permissão o novo usuário estará inserido, além de preenchermos as informações para *login*, e-mail e nome completo, conforme com a Figura 3.5.

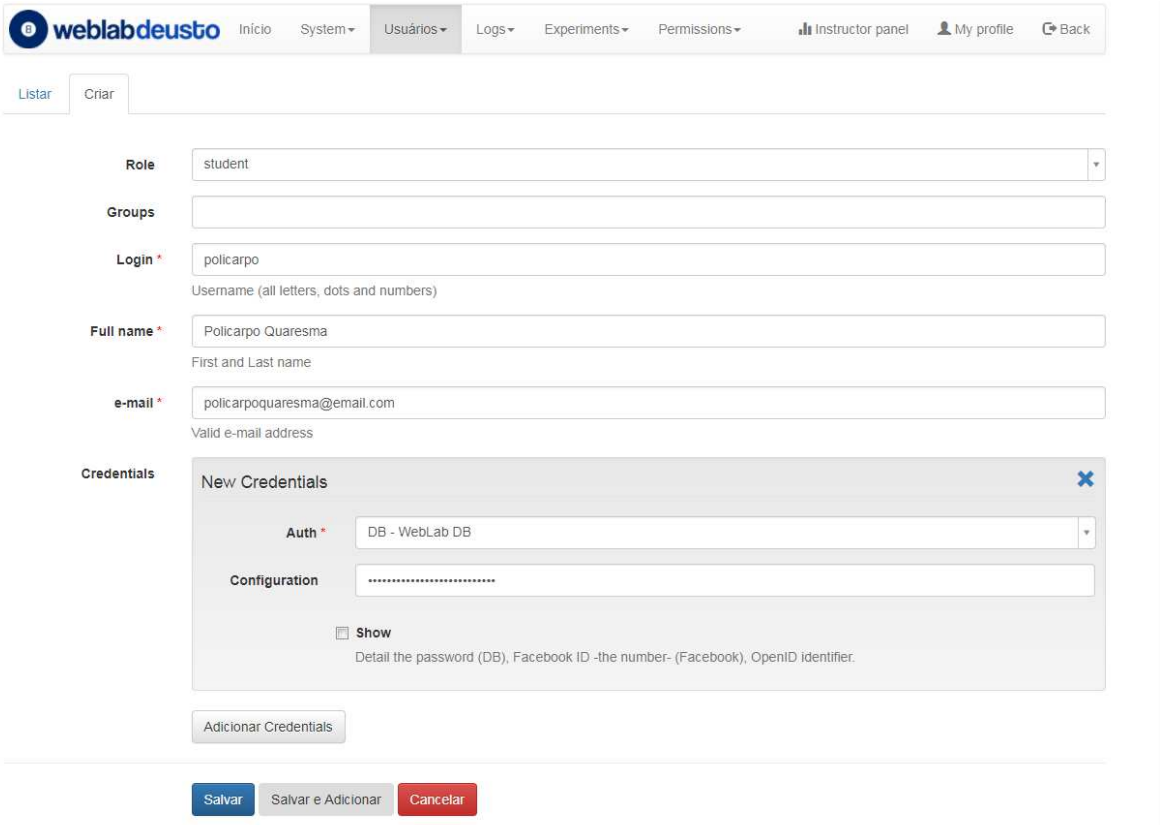

Figura 3.5 – Tela de criação de novo usuário

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Para uma atividade ou professor que tem diversos turmas envolvidas com o laboratório, a criação de Grupos facilita bastante a organização e controle de cada agrupamento de usuários. Para isso, basta acessar *Usuários* e clicar em *Groups*. O panorama dos grupos é o mesmo panorama da tela de *Usuários*, na Figura 3.4. O processo de criação de um novo grupo também é similar, basta clicar em *Criar* e um formulário será gerado para ser preenchido, conforme Figura 3.6

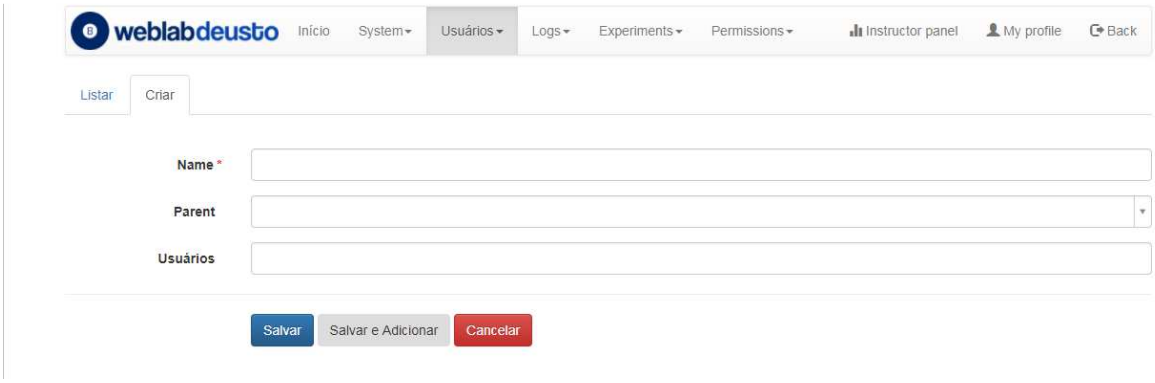

Figura 3.6 – Tela de criação de novo grupo

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Para criar um novo grupo, basta definir um nome para o mesmo e definir se existe um grupo pai, que pode ser muito útil para quando um professor ou instrutor tem diferentes turmas de uma mesma disciplina, além disso, caso os usuários tenham sido criados previamente, você pode adicioná-los direto nessa turma, sem precisar editar as configurações na tela de *Usuários*.

O próximo passo é gerenciar as permissões, seja para algum usuário ou para um grupo. Para isso, basta ir em *Permissions* e clicar em *Create*. Na primeira tela, você escolhe qual o tipo de permissão irá dar ao usuário e qual será o usuário beneficiado. Existem 6 diferentes tipos de permissões, que são explicadas previamente na própria tela, mesmo que em inglês. E essas 6 permissões podem ser aplicadas em 3 diferentes grupos: *Grupo*, *Usuário* e *Role* (função).

A permissão *access_forward* permite que usuários possam repassar reservas para usuários externos, já o *instructor_of_group*, permite que usuários visualize grupos selecionados para essa permissão. O *admin_panel_access* habilita o acesso de um usuário ao painel de controle do administrador e se ele terá ou não privilégios completos da função. O *profile_editing_disabled* desativa a opção de editar perfil dos usuários em questão por eles mesmos. O *access_all_labs* libera o acesso de um usuário a todos os laboratórios disponíveis, mas com uma quantidade fixa de tempo e uma prioridade bem baixa. Por fim, o *experiment_allowed* decide qual tipo de experimento um usuário terá acesso e qual a duração

da sua atividade nesse experimento ou laboratório. A Figura 3.7 traz a tela de seleções e a tela de edição das permissões *experiment_allowed*.

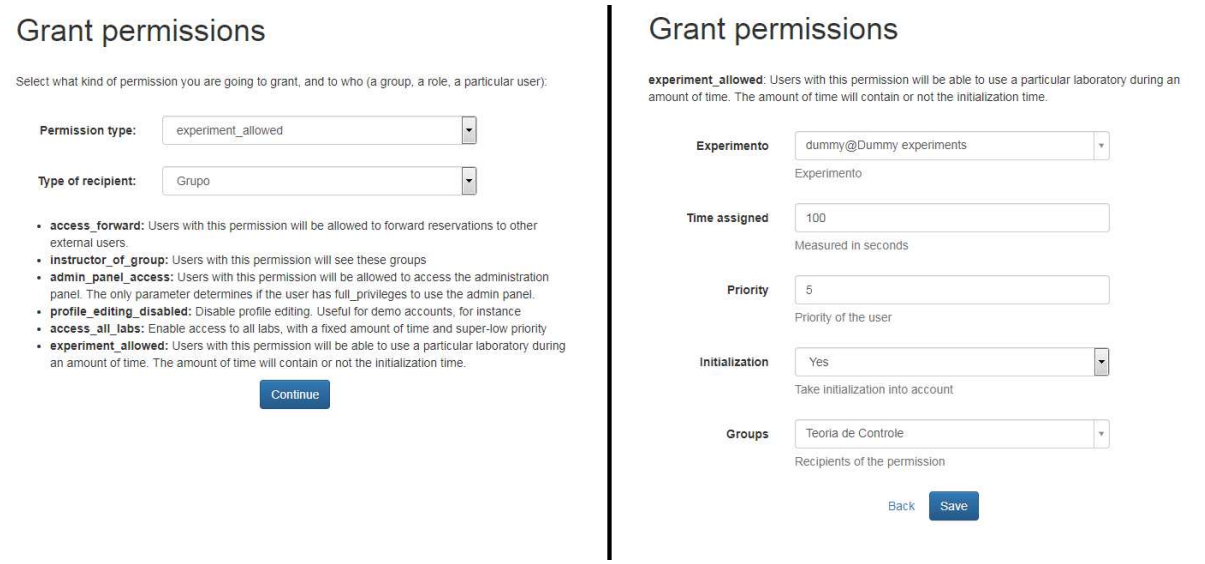

Figura 3.7 – Tela de seleção (esquerda) e tela de edição (direita)

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Além disso, existem opções para a adição de múltiplos usuários, a opção de convidar usuários para se registrarem na plataforma ou em algum grupo, caso ele já esteja na plataforma. Por fim, através da opção *Logs*, os usuários podem ser rastreados e quem tiver a permissão pode observar algumas informações mais simples e mais detalhadas de cada acesso, conforme Figura 3.8.

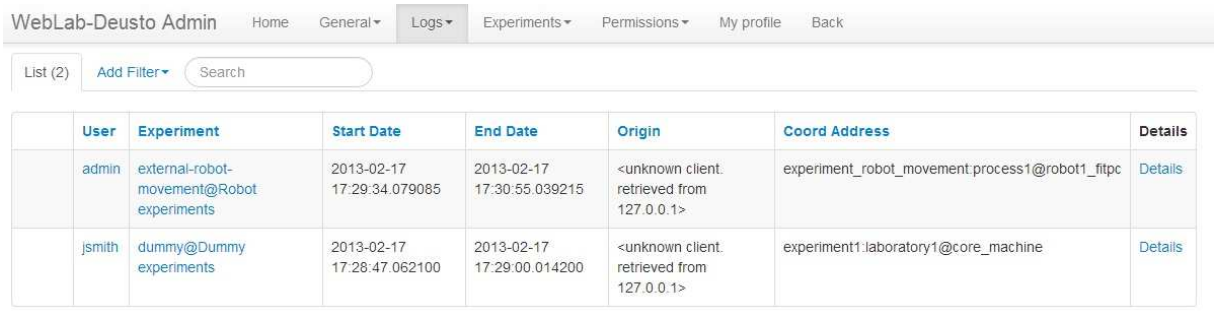

Figura 3.8 – Tela de rastreio dos dados de uso

Disponível em https://weblabdeusto.readthedocs.io/en/latest/first_steps.html#first-steps. Acesso em 10/08/2017 às 11:44

3.3 Desenvolvimento do Web Server

O WebLab-Deusto suporta dois tipos de desenvolvimento de laboratórios: *managed* and *unmanaged*. O assunto foi abordado no item 2.9. Como o LabREM já havia sido desenvolvido

e apenas necessitava de algumas funcionalidades presentes no ambiente WebLab-Deusto, optou-se por trabalhar com o desenvolvimento de um laboratório *unmanaged*.

Esse processo gira em torno do desenvolvimento de APIs para lidar com 5 tarefas: perceber que um novo usuário está acessando o site, checar se o usuário ainda está ativo, notificar o usuário que seu tempo está acabando e ele tem que terminar seu experimento, o desenvolvimento de uma API que se comunica com todas as outras e um serviço de teste. Com isso, pode-se ter duas aplicações web diferentes trabalhando em conjunto para utilizar a estrutura existente de qualquer laboratório com as ferramentas da plataforma.

A primeira tarefa é permitir que o WebLab-Deusto saiba qual a metodologia disponível e qual o formato dela. Isso é necessário, pois conforme o ambiente vai se desenvolvendo e recebendo novas adições, toda e qualquer versão anterior poderia se tornar incompatível. Na Figura 3.9, pode-se observar o processo de informação, onde o *Web Server* se comunicará com o LabREM.

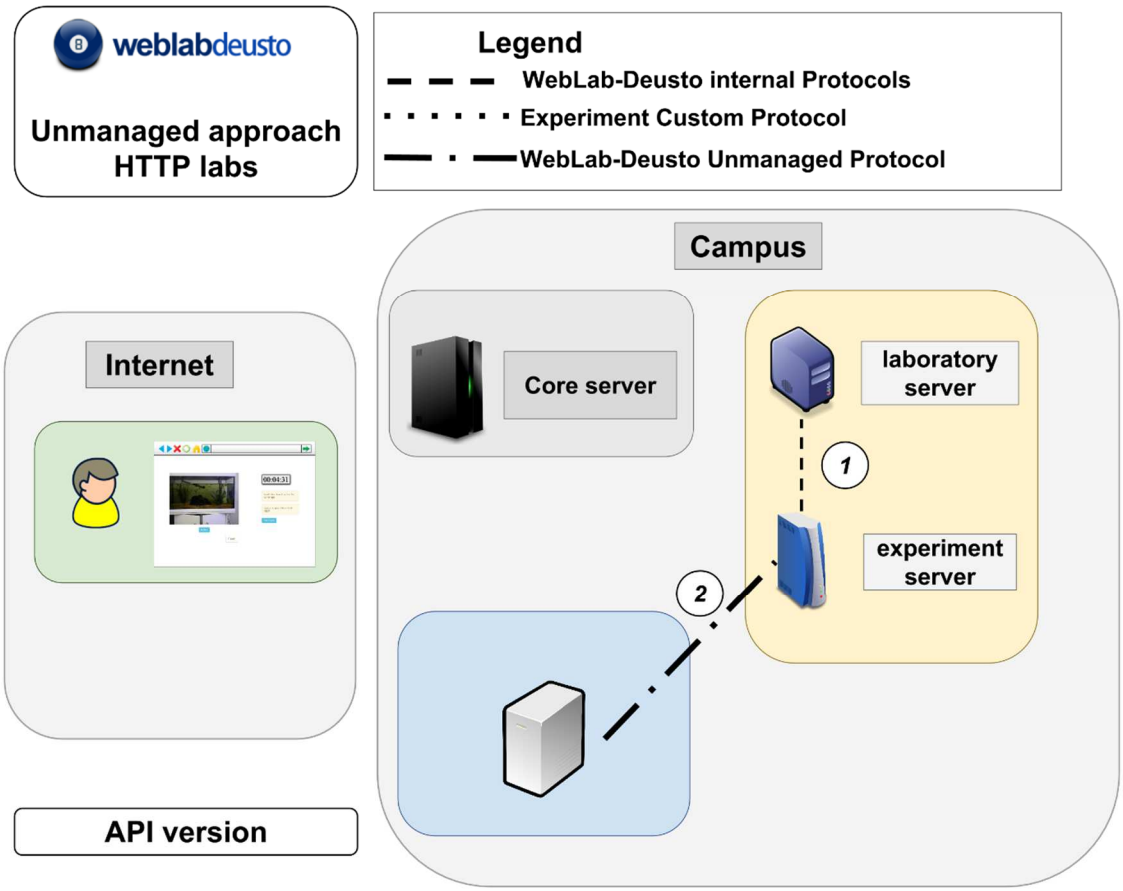

Figura 3.9 – Tarefa de checagem de versão da API

Disponível em https://weblabdeusto.readthedocs.io/en/latest/remote_lab_development.html#remote-lab-devel-unmanaged-laboratories. Acesso em 11/08/2017 às 10:47

A segunda tarefa é testar a conexão, e averiguar se ela é válida ou não de acordo com as informações de *login* fornecidas pelo usuário. Assim como na tarefa anterior, o *Web Server* que se comunica com o LabREM.

A terceira tarefa é permitir que o usuário acesso o laboratório. Na Figura 3.10 vê-se o processo que o sistema faz para dar início a um acesso. Em 3.10(1) o usuário contata o WebLab-Deusto, solicitando se o sistema está disponível, em 3.10(2), o sistema checa a disponibilidade do experimento, passando o usuário para a próxima etapa ou adicionando ele em uma fila. No 3.10(3), quando o usuário estiver liberado para começar, o WebLab-Deusto, em 3.10(4) irá iniciar uma sessão comunicando-se com o server do LabREM, que subsequentemente, vai dar início ao experimento, devolvendo um acesso para o usuário, como em 3.10(5).

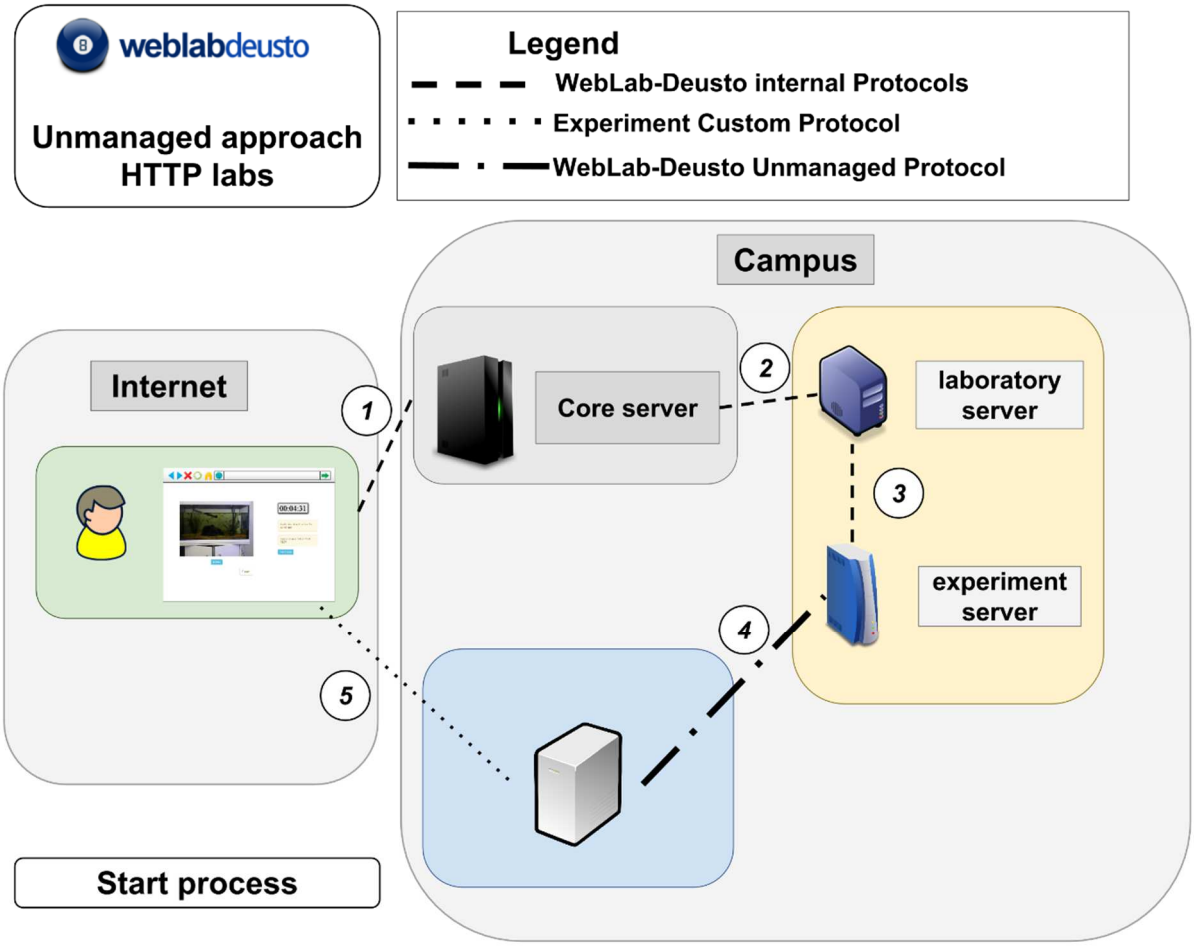

Figura 3.10 – Permitindo acesso ao laboratório e experimento

Disponível em https://weblabdeusto.readthedocs.io/en/latest/remote_lab_development.html#remote-lab-devel-unmanaged-laboratories. Acesso em 11/08/2017 às 10:59

A penúltima tarefa é checar se o usuário ainda está utilizando o laboratório, fazendo com que constantemente ela seja chamada pelo WebLab-Deusto. Nesse caso, o *Web Server* se comunica

com o server do LabREM, que se comunica com o experimento, que checa se o usuário ainda está online.

Por fim, a última tarefa determina se o usuário deve ou não ter seu acesso cortado, por não estar ativo ou ter excedido o tempo do experimento. As APIs desenvolvidas nessa etapa foram desenvolvidas por Pablo Orduña, da equipe do WebLab-Deusto e estão disponíveis no Anexo A.

3.4 Integração entre Web Server e LabREM

Com as APIs desenvolvidas e fornecidas pela equipe do WebLab-Deusto, chega a hora de integrar o *Web Server* e o laboratório remoto. Essa parte do processo é simples e quase poderia ser considerada *plug and play*, não fosse a necessidade de gerar uma *Dynamic Link Library* (DLL) para a comunicação entre o código do laboratório e o código do WebLab.

> "Uma DLL é uma biblioteca que contém código e dados que podem ser usados por mais de um programa ao mesmo tempo. Por exemplo, em sistemas operacionais Windows, a DLL Comdlg32 executa funções comuns relacionadas à caixa de diálogo. Portanto, cada programa pode usar a funcionalidade contida dessa DLL para implementar uma caixa de diálogo Abrir . Isso ajuda a promover a reutilização de código e o uso de memória eficiente". (MICROSOFT SUPPORT, *O que é uma DLL?* Em <https://support.microsoft.com/pt-br/help/815065/what-is-a-dll>. Acessado em 01/09/2017 às 18:45

Para isso, é necessário rodar um ambiente exclusivo com os arquivos de códigos e APIs no computador que está localizado o site do laboratório, esse procedimento serve para fazer a integração entre o *Web Server* e o LabREM, além de permitir que a arquitetura de programação utilizada para o desenvolvimento do laboratório remoto possa se conectar com os arquivos necessários para fazer a comunicação entre as duas partes no processo.

Após gerar a DLL, é só colocá-la dentro da pasta bin do site, adicionar uma pasta com os arquivos necessários para a integração e adicionar um trecho no código principal do site para que a comunicação seja realizada e o LabREM possa se comunicar com o *Web Server* e ser dotado das ferramentas e atributos do WebLab-Deusto.

Depois é só uma questão de configurar os endereços virtuais do laboratório remoto e *Web Server* nos códigos e arquivos de configuração do WebLab-Deusto. Com tudo isso preparado, basta acessar o WebLab-Deusto e criar um novo experimento, acessando a área *Experiments* ao clicando em Experimentos. Uma página com uma lista de experimentos irá abrir e é só clicar em criar para adicionar o experimento desejado. A Figura 3.11 demonstra as opções relevantes

para que o sistema entenda que deve utilizar um laboratório localizado externamente da rede do WebLab-Deusto.

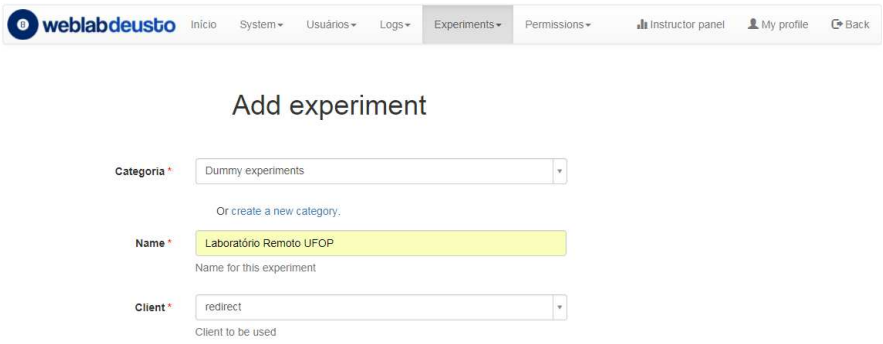

Figura 3.11 – Painel de criação de experimento

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Com isso, basta retornar à página inicial da plataforma e acessar o experimento criado, ao clicar no experimento, o usuário é direcionado para a página demonstrada na Figura 3.12, que contém o acesso ao LabREM através do botão *Reservar*, além de contar com algumas estatísticas de uso do usuário e do laboratório.

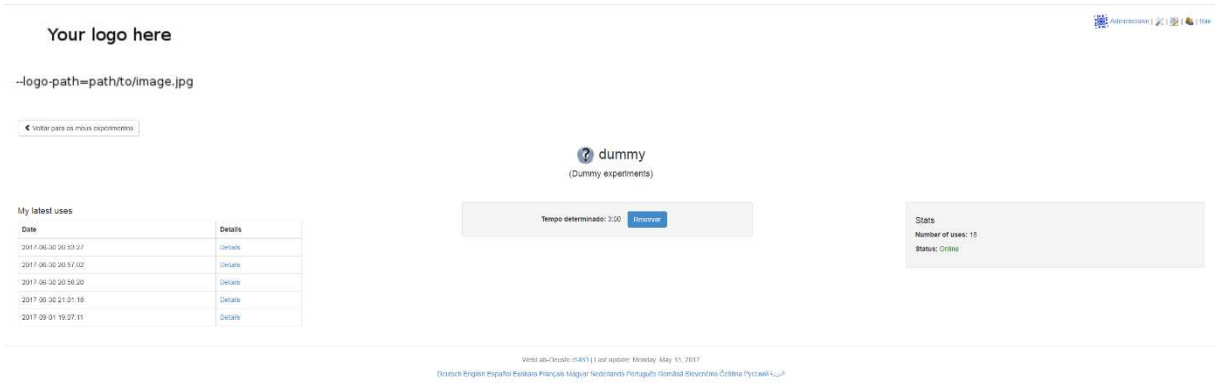

Figura 3.12 – Página de acesso ao LabREM na plataforma

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Após acessar o laboratório, a página exibida na Figura 3.13 será demonstrada pelo *web browser*, contendo a página do laboratório remoto, com algumas simples opções da própria plataforma. A página do laboratório que é exibida para o usuário é a mesma que seria exibida através do acesso direto, porém, agora, só é possível acessá-la através do WebLab-Deusto.

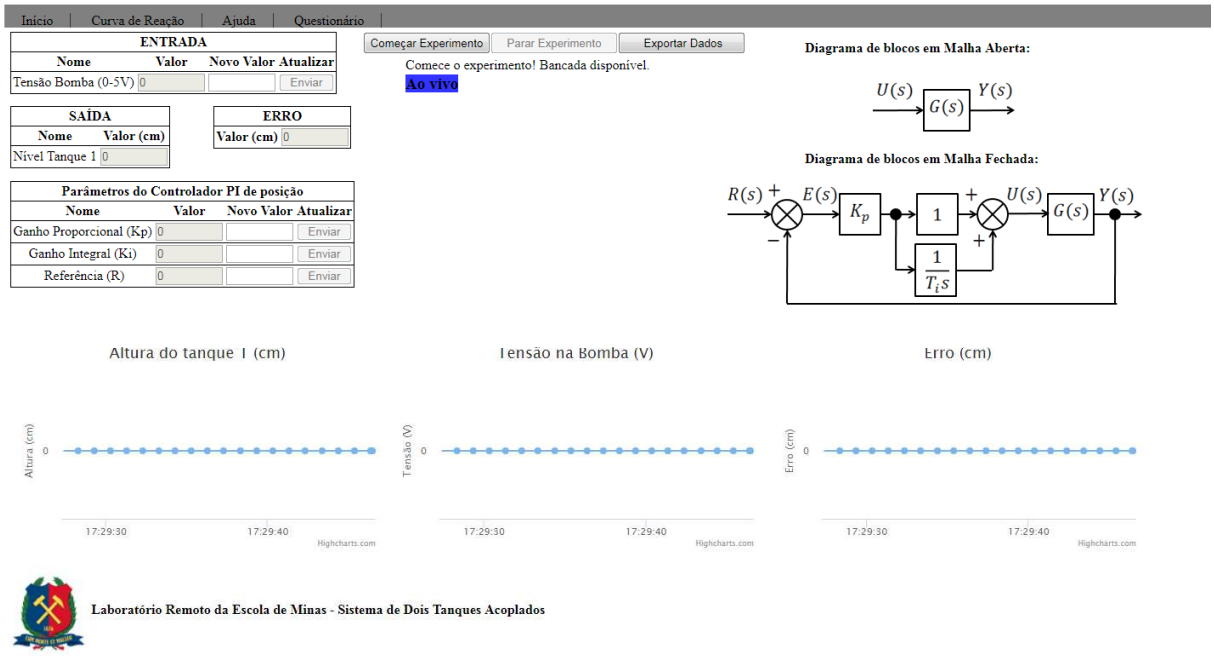

Figura 3.13 – Página do laboratório remoto após acesso via WebLab-Deusto

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

Caso alguém tente acessar o endereço externamente, sem passar pelo procedimento de controle de acesso e verificação de identidade do WebLab-Deusto, a plataforma não permite, pedindo que o usuário forneça uma sessão válida, que é executada apenas pela plataforma, assim, a página acaba exibindo uma mensagem de erro para o usuário, conforme observado na Figura 3.14.

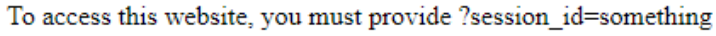

Figura 3.14 – Mensagem retornada por acesso não-autorizado do usuário

Fonte: *print screen* adaptado da página no sistema operacional Windows 7

3.5 Uma instância completamente funcional do WebLab-Deusto

A instância do LabREM em conjunto com o *Web Server* está completamente instalada e funcional, porém, para efeitos de demonstração das ferramentas mais importantes da plataforma

WebLab-Deusto, é impossível utilizar a instância do LabREM, já que a ausência de uso da plataforma, impedem que se demonstre adequadamente essas funcionalidades no âmbito da UFOP.

Contudo, para mostrar a relevância e importância da plataforma para o LabREM, uma análise de uma instância completamente funcional e que está em uso há bastante tempo se faz necessária. Para isso, a observação da plataforma multi-laboratorial da Universidade de Deusto se torna assaz necessária.

Inicialmente, temos a página inicial, onde se localizam os experimentos. Conforme a Figura 3.15, temos a página inicial da plataforma. Nela se encontram todos os experimentos registrados e instalados no *Web Server* do WebLab-Deusto da Universidade de Deusto.

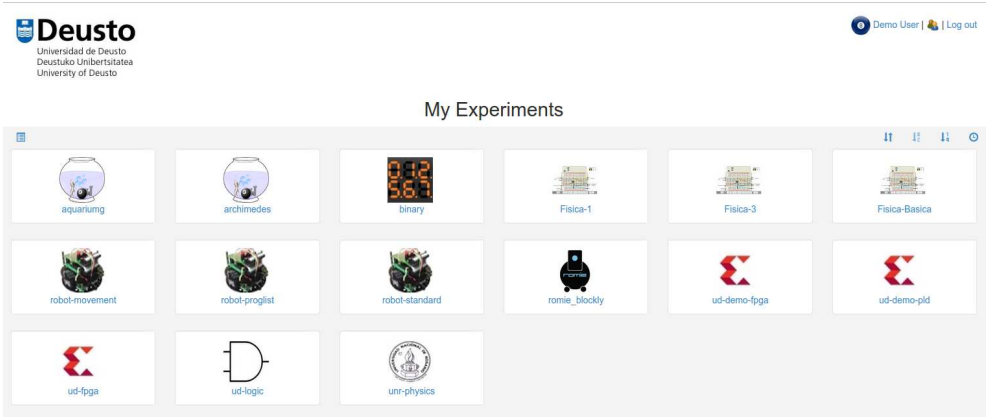

Figura 3.15 – Página dos experimentos presentes no WebLab-Deusto

Disponível em http://weblabdeusto.readthedocs.io/en/latest/_images/experiment-list-desktop.png. Acesso em 01/09/2017 às 20:31

A página inicial do administrador, conforme Figura 3.16, contém algumas informações sobre o uso dos experimentos e dos usuários. Além disso, é possível verificar quais os laboratórios estão sendo mais usados.

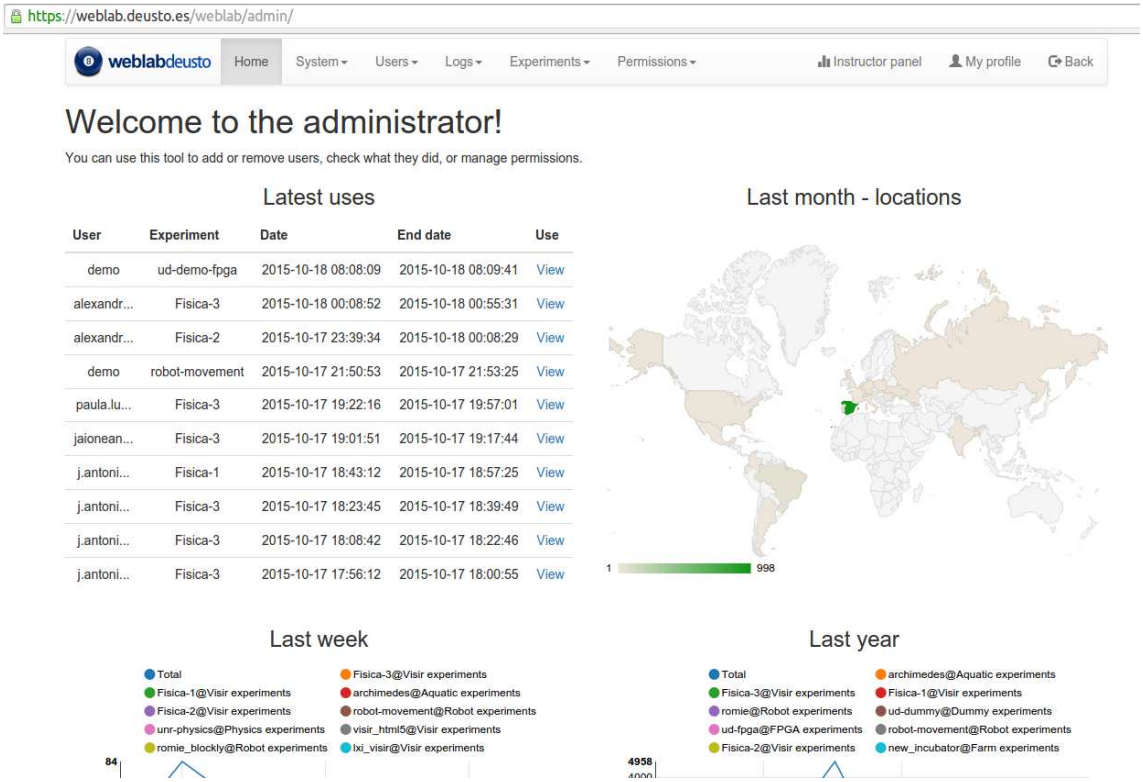

Figura 3.16 – Página inicial do administrador com dados de uso e usuários

Disponível em http://weblabdeusto.readthedocs.io/en/latest/_images/weblab-admin-panel.png. Acesso em 01/09/2017 às 20:38

E também podemos verificar mais de perto o comportamento de cada grupo de alunos. A Figura 3.17 traz informações do uso ao longo dos horários do dia e da semana e a Figura 3.18 traz informações a respeito do uso dos experimentos por um grupo.

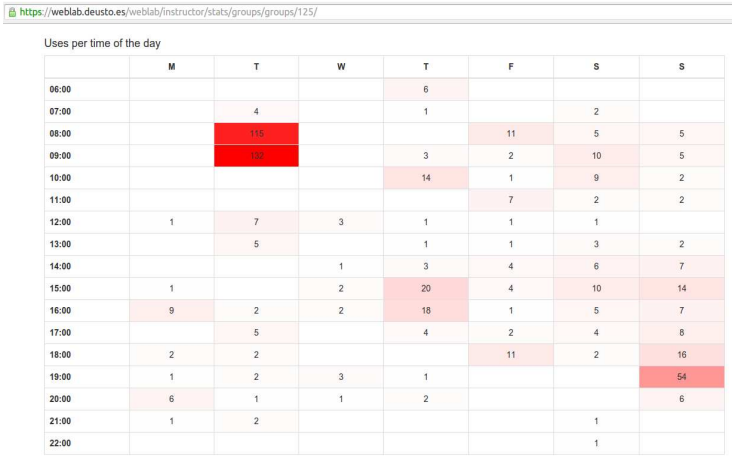

Figura 3.17 – Painel de informações do uso diário dos usuários de cada grupo

Disponível em http://weblabdeusto.readthedocs.io/en/latest/_images/weblab-admin-panel2.png. Acesso em 01/09/2017 às 20:42

Figura 3.18 – Informações a respeito do padrão de uso de um grupo

Disponível em http://weblabdeusto.readthedocs.io/en/latest/_images/weblab-admin-panel3.png. Acesso em 01/09/2017 às 20:44

4. Resultados e Discussões

A integração entre *Web Server* e o Laboratório Remoto da Escola de Minas foi realizada com sucesso, adaptando a tecnologia existente com a do WebLab-Deusto, produzindo um *Web Server* independente que fosse capaz de realizar as funções necessárias para a aquisição de dados de uso e o rastreio dessas informações, além do armazenamento e demonstração simples, compreensível e de fácil acesso a esses dados.

Infelizmente, não é possível avaliar adequadamente esses dados e o comportamento dos usuários do LabREM, já que essas informações só passaram a surgir a partir do momento que a plataforma e o laboratório remoto estejam em constante uso pela comunidade discente.

Porém, esse passo é muito importante para revolucionar a forma como a UFOP se relaciona com suas instalações laboratoriais, sendo possível criar uma extensa rede de laboratórios remotos. O trabalho prévio no LabREM já demonstra a capacidade da comunidade acadêmica local ser capaz de produzir seus próprios laboratórios remotos, sendo talvez necessária agora a difusão entre os próprios professores e alunos da universidade, para que a prática se torne mais corriqueira e seja possível a implementação de um grande coletivo de laboratórios, que permita que a UFOP tenha algo similar à Universidade de Deusto e até mesmo a UFSC, com o *RexLab*.

Obviamente que esse desenvolvimento de uma extensa rede de laboratórios seria dependente da integração com o *Web Server*, porém, após a integração inicial, o processo de repetição se torna mais simples, facilitando esse processo num futuro. A tecnologia e auxílio do pessoal do WebLab-Deusto foi muito importante para que esse passo fosse dado por aqui.

E apesar de ainda não sermos capazes de mensurar a importância desse impacto diretamente, pelas análises e opiniões expressas sobre a utilização desse tipo de dinâmica no ensino prático dos estudantes, pode-se estar a caminho de uma nova maneira de estudantes e professores interagirem no processo de aprendizagem, trazendo benefícios para todos os envolvidos.

5. Conclusão e trabalhos futuros

Analisando-se o panorama presente na área de ensino através do uso de laboratórios remotos e como eles se integram no processo de aprendizagem, pode-se analisar que é uma ferramenta importante num cenário econômico crítico para as universidades brasileiras, além de possibilitar espaço para que uma maior integração entre as diversas modalidades de ensino e também os diversos cursos e disciplinas práticas possam se inter-relacionar de uma maneira que permita a difusão rápida e eficaz do conhecimento, permitindo uma maior disponibilidade de ferramentas que uma instituição de ensino pode fornecer.

Aparenta ser um processo simples, porém, só possível pela implementação de uma tecnologia já existente e que já foi testada, comprovada e está em constante evolução. A área de acesso e controle remoto ganha um novo aliado na UFOP e transforma as capacidades do LabREM. Porém, ainda existe espaço para melhorias.

Inicialmente, apesar de se correlacionarem, as duas entidades apresentam similaridades que acabam gerando uma redundância, portanto, seria necessário um polimento e adaptação ao atual site do LabREM para que as coisas funcionem de uma maneira mais síncrona, retirando funcionalidades que o WebLab-Deusto fornece nativamente e que estão presentes também no site. Felizmente, não é uma questão de funcionalidade, é apenas uma questão de interfaceamento para o usuário.

Além disso, é possível que ambas as partes se beneficiem de uma adaptação ao Linux. O WebLab-Deusto foi planejado dentro do sistema operacional, apesar de não existir desvantagens em utilizá-lo no Windows. Para o futuro, a equipe do WebLab-Deusto irá utilizar um sistema de coordenação que é voltado para o ambiente Linux. O Redis vai passar a se tornar uma ferramenta vital para novas funcionalidades a serem implementadas na plataforma. O programa existe para Windows, mas ele é uma espécie de *port* e poderá não haver uma plena convergência nesse ambiente.

Por fim, o ideal seria que em caso de uma ampliação no uso da plataforma no âmbito da comunidade laboratorial do curso de Engenharia de Controle Automação, ou na Escola de Minas ou na UFOP, que o desenvolvimento de uma instância *managed* seja desenvolvida. Isso permitirá uma capacidade de adaptação e replicação de laboratórios muito grande. No caso do LabREM, a atual situação é bastante confortável, mas para um futuro de integração múltipla de laboratórios, seria interessante uma padronização.

Essa padronização permitiria que mais projetos nessa área sejam desenvolvidos, evitando que se perca tempo em produzir um ambiente do zero, e também facilitaria, ao direcionar os esforços em uma mesma linha de pensamento. Além disso, permitiria que mais energia fosse dispendida em outras partes do processo e do aprendizado, flexibilizando a forma como se interage com esse tipo de ferramenta.

6. Referências Bibliográficas

ALVES, Paulo. **O que é acesso remoto? Entenda tudo sobre conexão à distância**. Em: < http://www.techtudo.com.br/noticias/noticia/2013/10/o-que-e-acesso-remoto-entenda-tudo-sobre-conexao-distancia.html >, acessado em 17/07/2017 às 19:45.

AXELSON, Jan. **Embedded ethernet and internet complete: designing and programming small devices for networking**. Lakeview research llc, 2003.

BBVAOPEN4U. **101: Introduction to the world of APIs**. Disponível em <https://bbvaopen4u.com/en/actualidad/infographic-what-api>, acessado em 27/07/2017 às 09:37

BISTÁK, Pavol. **Remote laboratory server based on Java Matlab interface**. In: Interactive Collaborative Learning (ICL), 2011 14th International Conference on. IEEE, 2011. p. 344-347.

BOCHICCHIO, Daniele; et al. **ASP. NET 4.0 in practice**. Manning Publications Co., 2011.

COOKSEY, Brian. **An Introduction to APIs**. Zapier. 2014. Edição Kindle

COSTA, Ricardo J. et al. **Reconfigurable IEEE1451-FPGA based weblab infrastructure**. In: Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on. IEEE, 2012. p. 1-9.

EGEA, Lucas. **Acesso remoto: o que é e como funciona?** Em: < http://blog.loupen.com.br/acesso-remoto-o-que-e-e-como-funciona/ >, acessado em 17/07/2017 às 19:52

FALL, Kevin R.; STEVENS, W. Richard. **TCP/IP illustrated, volume 1: The protocols**. Addison-Wesley, 2011.

FAUSTO, Pedro. **Conceitos de Acesso Remoto**. Em: < https://www.vivaolinux.com.br/artigo/Conceitos-de-Acesso-Remoto >, acessado em 17/07/2017 às 19:56

GAO, Hongyan. **Development of a remote laboratory for process control experiments**. In: E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on. IEEE, 2010. p. 87-90.

GARCÍA-ZUBIA, Javier et al. **Evolution of the WebLab at the University of Deusto**. EWME 2006, 2006.

GARCIA-ZUBIA, Javier et al. **Application and user perceptions of using the WebLab-Deusto-PLD in technical education**. In: Frontiers in Education Conference (FIE), 2011. IEEE, 2011. p. GOLC1-1-GOLC1-6.

GOMES, Luís; ZUBÍA, Javier Garcia. **Advances on remote laboratories and e-learning experiences**. Universidad de Deusto, 2008.

HUBA, Mikuláš et al. **Remote experiments in control education**. IFAC Proceedings Volumes, v. 37, n. 7, p. 135-140, 2004.

JAQUES, Rafael. **O que é um Framework? Para que serve?** Em <http://www.phpit.com.br/artigos/o-que-e-um-framework.phpit>, acessado em 26/07/2017 às 09:23

JONES, A. Russell. **Mastering Asp. Net with VB**. Net. Sybex, 2002.

KALÚZ, Martin et al. **Sharing control laboratories by remote laboratory management system weblab-deusto**. IFAC Proceedings Volumes, v. 46, n. 17, p. 345-350, 2013.

KALUZ, Martin et al. **A flexible and configurable architecture for automatic control remote laboratories. IEEE Transactions on Learning Technologies**, v. 8, n. 3, p. 299-310, 2015.

KENNEPOHL, Dietmar et al. **Remote access to instrumental analysis for distance education in science**. The International Review of Research in Open and Distributed Learning, v. 6, n. 3, 2006.

KOZIEROK, Charles M. **The TCP/IP guide: a comprehensive, illustrated Internet protocols reference**. No Starch Press, 2005.

KUROSE, Jim; ROSS, Keith. **Computer Networking: A Top Down Approach**, 2012.

LIMA, Edwin; REIS, Eugênio. **C# e. net–Guia do Desenvolvedor**. Rio de Janeiro: Campus, 2002.

MARQUES, Rui et al. **Design and implementation of a reconfigurable remote laboratory, using oscilloscope/PLC network for WWW access**. IEEE transactions on industrial electronics, v. 55, n. 6, p. 2425-2432, 2008.

MARTINS, Elaine. **O que é TCP/IP?** Em <https://www.tecmundo.com.br/o-que-e/780-o-que-e-tcp-ip-.htm>, acessado em 25/07/2017 às 09:29

MICROSOFT. Disponível em <https://support.microsoft.com/pt-br/help/815065/what-is-a-dll>, acessado em 01/09/2017 às 18:45

MORIMOTO, Carlos E. **Hardware de servidores: uma introdução**. Em <http://www.hardware.com.br/dicas/hardware-servidores.html>, acessado em 25/07/2017 às 16:10

NETO, J. M. et al. **Remote educational experiment applied to electrical engineering**. In: Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on. IEEE, 2012. p. 1-5.

NOERGAARD, Tammy. **Embedded systems architecture: a comprehensive guide for engineers and programmers**. Newnes, 2012.

OLIVEIRA, Luiz Otávio Mendes de. **Desenvolvimento de um laboratório remoto para práticas de controle de nível dos tanques acoplados**. 2017.

ORDUÑA, Pablo et al. **Adding New Features to New and Existing Remote Experiments Through Their Integration in WebLab-Deusto**. International Journal of Online Engineering, v. 7, 2011.

PYTHON. Disponível em <www.python.org>, acessado em 09/08/2017 às 15:19

PYTHON BRASIL. Disponível em <www.python.org.br>, acessado em 09/08/2017 às 15:20

QUANSER. **Coupled Water Tank Experiments**, Quanser, EUA, 2013

QUANSER. Disponível em <www.quanser.com>, acessado em 27/07/2017 às 15:20

SANTANA, I. et al. **Remote laboratories for education and research purposes in automatic control systems**. IEEE transactions on industrial informatics, v. 9, n. 1, p. 547-556, 2013.

SEBESTA, R. W. **Concepts of Programming Languages**. Always learning. 2012.

SILVEIRA, Cristiano Bertulucci. **Entenda como funciona o protocolo TCP-Ip**. Em <https://www.citisystems.com.br/protocolo-tcp-ip/>, acessado em 25/07/2017 às 10:28

THAI, Thuan; LAM, Hoang. **. NET framework essentials.** " O'Reilly Media, Inc.", 2003.

WEBLAB-DEUSTO. **WebLab-Deusto Documentation**. Disponível em <https://weblabdeusto.readthedocs.io/en/latest/summary.html>, acessado em 13/07/2017 às 19:17.

7. Anexo A

Todos os códigos e arquivos foram fornecidos por Pablo Orduña, da equipe do WebLab-Deusto.

Código inserido no código principal (Default.apsx) do site:

```
<span id="timer">
        <script type="text/javascript">
        function PollUser() {
            PageMethods.PollUser(function (result, userContext, methodName) {
                if (!result) {
                    alert("Your session is over");
                    location.reload();
                }
            });
        }
        setInterval(PollUser, 5000); // Every 5 seconds, call the method
        var totalSeconds;
        function startTimer(result) {
            totalSeconds = result;
            // document.getElementById("timer").innerHTML = result;
            setInterval(function () {
                totalSeconds = totalSeconds - 1;
                // document.getElementById("timer").innerHTML = totalSeconds;
            }, 1000);
        }
        PageMethods.GetSeconds(startTimer, startTimer);
    </script>
```

Código inserido no código principal (Default.apsx.cs) do site:

```
[System.Web.Services.WebMethod()]
   public static int GetSeconds()
   {
      return WebLabPage.GetSeconds();
   }
```

```
  [System.Web.Services.WebMethod()]
  public static bool PollUser()
 {
     return WebLabPage.PollUser();
  }
```

Arquivo action.apsx.cs:

```
using System;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Json;
namespace weblab {
   public partial class WebLabAction : Page {
      protected void Page_Load(object sender, EventArgs e) {
         string sessionId = WebLabUtils.CheckSessionId(this);
         if (sessionId == null)
            return;
         if (!WebLabUtils.CheckWebLabCredentials()) {
            WebLabUtils.SendInvalidCode(this);
            return;
         }
         UserUtils.Get(Application).ExpireUser(sessionId);
         JsonObject json = new JsonObject();
         Response.Clear();
         Response.ContentType = "application/json; charset=utf-8";
         Response.Write(json.ToString());
         Response.End();
      }
   }
}
```

Arquivo mylab.aspx.cs:

```
using System;
```

```
using System.Web;
using System.Web.Security;
using System.Web.Services;
using System.Web.UI;
using System.Json;

using weblab;

namespace mylab {
   public partial class MyLab : Page {

                protected System.Web.UI.WebControls.TextBox TextBox1;
                protected System.Web.UI.WebControls.TextBox TextBox2;
                protected System.Web.UI.WebControls.Button Button1;
                protected System.Web.UI.WebControls.Label Label1;
                protected System.Web.UI.WebControls.Label Label2;

                #region Web Form Designer generated code
                override protected void OnInit(EventArgs e)
                {
                        //
                        // CODEGEN: This call is required by the ASP.NET Web Form Designer.
                        //
                        InitializeComponent();
                        base.OnInit(e);
                }

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
```

```
            this.Button1.Click += new System.EventHandler(this.Button1_Click);
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

protected void Page_Load(object sender, EventArgs e) {
    if (!WebLabPage.ValidatePageLoad())
        // It generates an error
        return;

    // Your code, for example:
    Label1.DataBind();
}

protected void Button1_Click(object sender, EventArgs e)
{
    // If the user is not active, don't let him make changes
    if (!WebLabPage.CheckUserActive())
        return;

    // Otherwise, run the code
    string buf = TextBox1.Text;
    TextBox2.Text = buf.ToUpper();
}


[System.Web.Services.WebMethod()]
public static int GetSeconds()
{
    return WebLabPage.GetSeconds();
}
```

```
        [System.Web.Services.WebMethod()]
        public static bool PollUser()
        {
            return WebLabPage.PollUser();
        }
    }
}
```

Arquivo newSession.aspx.cs:

```
using System;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Json;

namespace weblab {
    public partial class NewSession : Page {
        protected void Page_Load(object sender, EventArgs e) {

            // This is a POST website, not a GET website
            if (Request.HttpMethod != "POST") {
                JsonObject json = new JsonObject();
                json["error"] = true;
                json["message"] = "This should be a POST call";
                Response.Clear();
                Response.ContentType = "application/json; charset=utf-8";
                Response.Write(json.ToString());
                Response.End();
            }

            // Check that the caller is WebLab-Deusto (and not a user)
            if (!WebLabUtils.CheckWebLabCredentials()) {
                WebLabUtils.SendInvalidCode(this);
```

```
            return;
        }

        // Obtain the user data from the WebLab-Deusto request
        // There are more values than this, check WebLab-Deusto documentation
        string backUrl = Request.Form["back"];
        JsonValue serverData = JsonValue.Parse(Request.Form["server_initial_data"]);
        string username = serverData["request.username"];
        int secondsRemaining = serverData["priority.queue.slot.length"];
        DateTime maxDate = DateTime.UtcNow.AddSeconds(secondsRemaining);

        // Generate a secure secret to identify this session
        string sessionId = WebLabUtils.GenerateSecret();

        // Create a WebLabUser, and add it to Application:
        WebLabUser user = new WebLabUser(username, backUrl, maxDate);
        UserUtils.Get(Application).AddUser(sessionId, user);

        // Create the response: the session_id and the url to go
        JsonObject jsonResponse = new JsonObject();
        jsonResponse["session_id"] = sessionId;
        jsonResponse["url"] = WebLabUtils.CreateUrl(sessionId);

        // Send the JSON object
        Response.Clear();
        Response.ContentType = "application/json; charset=utf-8";
        Response.Write(jsonResponse.ToString());
        Response.End();
      }
    }
}
```

Arquivo status.aspx.cs

```
using System;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Json;

namespace weblab {
    public partial class WebLabStatus : Page {

        protected void Page_Load(object sender, EventArgs e) {
            string sessionId = WebLabUtils.CheckSessionId(this);
            if (sessionId == null)
                return;

            if (!WebLabUtils.CheckWebLabCredentials()) {
                WebLabUtils.SendInvalidCode(this);
                return;
            }

            bool userFound = false;
            bool userExpired = false;
            WebLabUser user = UserUtils.Get(Application).GetCurrentUser(sessionId);
            if (user != null) {
                userFound = true;
                if (user.IsExpired)
                    userExpired = true;
            }

            JsonObject json = new JsonObject();
            if (!userFound) {
                json["should_finish"] = -1;
                json["reason"] = "user session not found";
            } else if (userExpired) {
```

```
            json["should_finish"] = -1;
            json["reason"] = "user expired";
        } else {
            json["should_finish"] = 10;
        }

        Response.Clear();
        Response.ContentType = "application/json; charset=utf-8";
        Response.Write(json.ToString());
        Response.End();
      }
   }
}
```

Arquivo test.aspx.cs:

```
using System;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Json;

namespace weblab {
   public partial class Test : Page {
      protected void Page_Load(object sender, EventArgs e) {

        JsonObject json = new JsonObject();

        if (WebLabUtils.CheckWebLabCredentials()) {
            json["valid"] = true;
        } else {
            if (Request.QueryString["force"] != null) {
                WebLabUtils.SendInvalidCode(this);
                return;
```

```
            }
            json["valid"] = false;
            JsonArray messages = new JsonArray();
            messages.Add("Invalid credentials");
            json["error_messages"] = messages;
        }

        Response.Clear();
        Response.ContentType = "application/json; charset=utf-8";
        Response.Write(json.ToString());
        Response.End();
    }
  }
}
```

Arquivo utils.cs:

```
using System;
using System.Security.Cryptography;
using System.Collections.Generic;
using System.Text;
using System.Json;
using System.Web;
using System.Web.UI;
namespace weblab {
  public class WebLabPage {
    public static bool PollUser() {
      string sessionId = (string)HttpContext.Current.Session["weblabSessionId"];
      if (sessionId == null)
        return false;
      WebLabUser user = UserUtils.Get(HttpContext.Current.Application).GetCurrentUser(sessionId);
      if (user != null) {
        user.Update();
        return !user.IsExpired;
```

```
    } else {
        return false;
    }
}
public static int GetSeconds() {
    string sessionId = (string)HttpContext.Current.Session["weblabSessionId"];
    if (sessionId != null) {
        WebLabUser user = UserUtils.Get(HttpContext.Current.Application).GetCurrentUser(sessionId);
        if (user != null) {
            return (int)user.RemainingSeconds;
        }
    }
    return -1;
}
public static WebLabUser GetCurrentUser() {
    string sessionId = (string)HttpContext.Current.Session["weblabSessionId"];
    if (sessionId == null)
        return null;
    return UserUtils.Get(HttpContext.Current.Application).GetCurrentUser(sessionId);
}
public static string GetBackUrl() {
    string sessionId = (string)HttpContext.Current.Session["weblabSessionId"];
    if (sessionId == null)
        return null;
    WebLabUser user = UserUtils.Get(HttpContext.Current.Application).GetCurrentUser(sessionId);
    if (user != null)
        return user.BackUrl;
    user = UserUtils.Get(HttpContext.Current.Application).GetPastUser(sessionId);
    if (user != null)
        return user.BackUrl;
    return null;
}
public static bool CheckUserActive() {
```

```
        if (HttpContext.Current.Session["weblabSessionId"] == null)
            return false;
        if
(!UserUtils.Get(HttpContext.Current.Application).CheckUser((string)HttpContext.Current.Session["weblabSess
ionId"]))
            return false;
        return true;
    }
    public static bool ValidatePageLoad() {
        string sessionId = HttpContext.Current.Request["session_id"];
        if (sessionId == null) {
            if (HttpContext.Current.Session["weblabSessionId"] == null) {
                HttpContext.Current.Response.Clear();
                HttpContext.Current.Response.Write("To access this website, you must provide
?session_id=something");
                HttpContext.Current.Response.End();
                return false;
            }
            sessionId = (string)HttpContext.Current.Session["weblabSessionId"];
        }
        UserUtils utils = UserUtils.Get(HttpContext.Current.Application);
        WebLabUser currentUser = utils.GetCurrentUser(sessionId);
        if (currentUser != null) {
            // Perfect! It's an active user! Let's establish the session identifier
            // NOTE: it would be better if in this moment we redirected the user to the
            // same website without the token. Given that it's in a cookie, sharing the
            // URL to friends or whatever will not work
            HttpContext.Current.Session["weblabSessionId"] = sessionId;
            HttpContext.Current.Session["weblabUsername"] = currentUser.Username;
            return true;
        }
        // if currentUser is null, it might be an expired session:
        WebLabUser pastUser = utils.GetPastUser(sessionId);
        if (pastUser != null) {
```

```
            // It's here. Let's send the user back to weblab
            HttpContext.Current.Response.Redirect(pastUser.BackUrl);
            return false;
        }
        // if both are null, then the user is invalid:
        HttpContext.Current.Response.Clear();
        HttpContext.Current.Response.Write("You had an invalid session_id. To access this website, you must provide a valid ?session_id=something");
        HttpContext.Current.Response.End();
        return false;
    }
}
public class WebLabUser {
    private readonly string username;
    private readonly string backUrl;
    private readonly DateTime maxDate;
    private DateTime lastPoll;
    public WebLabUser(string username, string backUrl, DateTime maxDate) {
        this.username = username;
        this.backUrl = backUrl;
        this.maxDate = maxDate;
        this.lastPoll = DateTime.UtcNow;
    }
    public void Update() {
        this.lastPoll = DateTime.UtcNow;
    }
    public string Username {
        get {
            return this.username;
        }
    }
    public double RemainingSeconds {
        get {
```

```
            return this.maxDate.Subtract(DateTime.UtcNow).TotalSeconds;
        }
    }
    public string BackUrl {
        get {
            return this.backUrl;
        }
    }
    public bool IsExpired {
        get {
            DateTime now = DateTime.UtcNow;
            if (now > this.maxDate)
                return true;
            TimeSpan difference = now.Subtract(this.lastPoll);
            if (difference.TotalSeconds > WebLabUtils.MAX_POLL_IN_SECONDS)
                return true;
            return false;
        }
    }
}
public class UserUtils {
    private readonly HttpApplicationState application;
    public UserUtils(HttpApplicationState application) {
        this.application = application;
    }
    public static UserUtils Get(HttpApplicationState application) {
        return new UserUtils(application);
    }
    private Dictionary<string, WebLabUser> getCurrentDict() {
        if (this.application["currentWebLabUsers"] == null) {
            this.application["currentWebLabUsers"] = new Dictionary<string, WebLabUser>();
        }
```

```
        Dictionary<string, WebLabUser> currentWebLabUsers = (Dictionary<string,
WebLabUser>)(this.application["currentWebLabUsers"]);
        return currentWebLabUsers;
    }
    private Dictionary<string, WebLabUser> getPastDict() {
        if (this.application["pastWebLabUsers"] == null) {
            this.application["pastWebLabUsers"] = new Dictionary<string, WebLabUser>();
        }
        Dictionary<string, WebLabUser> pastWebLabUsers = (Dictionary<string,
WebLabUser>)(this.application["pastWebLabUsers"]);
        return pastWebLabUsers;
    }
    public void AddUser(string sessionId, WebLabUser user) {
        this.application.Lock();
        try {
            Dictionary<string, WebLabUser> currentWebLabUsers = getCurrentDict();
            currentWebLabUsers[sessionId] = user;
            this.application["currentWebLabUsers"] = currentWebLabUsers;
        } finally {
            this.application.UnLock();
        }
    }
    public WebLabUser GetCurrentUser(string sessionId) {
        this.application.Lock();
        try {
            Dictionary<string, WebLabUser> currentWebLabUsers = getCurrentDict();
            try {
                return currentWebLabUsers[sessionId];
            } catch (KeyNotFoundException) {
                return null;
            }
        } finally {
            this.application.UnLock();
```

```
    }
}
public WebLabUser GetPastUser(string sessionId) {
    this.application.Lock();
    try {
        Dictionary<string, WebLabUser> pastWebLabUsers = getPastDict();
        try {
            return pastWebLabUsers[sessionId];
        } catch (KeyNotFoundException) {
            return null;
        }
    } finally {
        this.application.UnLock();
    }
}
public bool CheckUser(string sessionId) {
    WebLabUser user = GetCurrentUser(sessionId);
    if (user == null)
        return false;
    if (user.IsExpired)
        return false;
    return true;
}
public WebLabUser ExpireUser(string sessionId) {
    this.application.Lock();
    try {
        Dictionary<string, WebLabUser> currentWebLabUsers = getCurrentDict();
        if (!currentWebLabUsers.ContainsKey(sessionId))
            return null;
        WebLabUser userToExpire = currentWebLabUsers[sessionId];
        currentWebLabUsers.Remove(sessionId);
        Dictionary<string, WebLabUser> pastWebLabUsers = getPastDict();
        pastWebLabUsers[sessionId] = userToExpire;
```

```
            this.application["currentWebLabUsers"] = currentWebLabUsers;
            this.application["pastWebLabUsers"] = pastWebLabUsers;
            return userToExpire;
        } finally {
            this.application.UnLock();
        }
    }
}
public class WebLabUtils {
    /*
    *
    * These two variables are the ones that need to be configured in WebLab-Deusto, in
    * the variables:
    *     http_experiment_username
    *     http_experiment_password
    *
    * As described in:
    *     http://weblabdeusto.readthedocs.io/en/latest/remote_lab_deployment.html#unmanaged-server
    */
    private const string WEBLAB_USERNAME = "weblab";
    private const string WEBLAB_PASSWORD = "password";
    private const string LAB_URL = "http://http://200.239.166.71:8085/Default.aspx?session_id=";
    public const int MAX_POLL_IN_SECONDS = 15;
    public static string GenerateSecret() {
        var hmac = new HMACSHA256();
        string converted = Convert.ToBase64String(hmac.Key);
        converted = converted.Replace("/","");
        converted = converted.Replace("+","");
        converted = converted.Replace("=","");
        converted = converted.Substring(0, 12);
        return converted;
    }
    public static string CreateUrl(string sessionId) {
```

```
        return LAB_URL + sessionId;
    }
    public static string CheckSessionId(Page page) {
        string sessionId = page.Request["session_id"];
        if (sessionId == null) {
            JsonObject jsonSessionMissing = new JsonObject();
            jsonSessionMissing["error"] = true;
            jsonSessionMissing["message"] = "session_id missing";
            page.Response.Clear();
            page.Response.ContentType = "application/json; charset=utf-8";
            page.Response.Write(jsonSessionMissing.ToString());
            page.Response.End();
            return null;
        }
        return sessionId;
    }
    public static bool CheckWebLabCredentials() {
                        HttpContext httpContext = HttpContext.Current;
                        string authHeader = httpContext.Request.Headers["Authorization"];
                        if (authHeader != null && authHeader.StartsWith("Basic")) {
                                string encodedUsernamePassword = authHeader.Substring("Basic
".Length).Trim();
                                Encoding encoding = Encoding.GetEncoding("iso-8859-1");
                                string usernamePassword =
encoding.GetString(Convert.FromBase64String(encodedUsernamePassword));
                                int seperatorIndex = usernamePassword.IndexOf(':');
                                string username = usernamePassword.Substring(0, seperatorIndex);
                                string password = usernamePassword.Substring(seperatorIndex + 1);
            if (username == WEBLAB_USERNAME && password == WEBLAB_PASSWORD)
                return true;
                        }
        return false;
    }
```

```
        public static void SendInvalidCode(Page page) {
            page.Response.StatusCode = 401;
            page.Response.Headers["WWW-Authenticate"] = "Basic realm=\"Login Required\"";
            page.Response.SuppressFormsAuthenticationRedirect = true;
            page.Response.End();
        }
    }
}
```