



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

PLATAFORMA WEB PARA ACESSO REMOTO A PLANTAS DIDÁTICAS DE AUTOMAÇÃO E CONTROLE

Victor Carvalho Galvão de Freitas

Orientador: Prof. Dr. Fábio Meneghetti Ugulino de Araújo

Natal/RN
Junho de 2013

PLATAFORMA WEB PARA ACESSO REMOTO A PLANTAS DIDÁTICAS DE AUTOMAÇÃO E CONTROLE

Victor Carvalho Galvão de Freitas

Orientador: Prof. Dr. Fábio Meneghetti Ugulino de Araújo

Monografia apresentada à Banca Examinadora do Trabalho de Conclusão do Curso de Engenharia de Computação, em cumprimento às exigências legais como requisito parcial à obtenção do título de Engenheiro de Computação.

Natal/RN
Junho de 2013

*Aos meus pais, José Galvão de
Freitas e Alzira Carvalho Galvão de
Freitas, que jamais mediram
esforços para me proporcionar a
melhor educação possível.*

Agradecimentos

Agradeço a Deus pela sua generosidade ao prover o homem de inteligência capaz de compreender as maravilhas criadas por Ele.

A Antônio Modesto Oliveira Macedo e Gleide Oliveira Macedo, verdadeiros tios adotivos, que tão bem me acolheram em sua casa.

Aos professores do Departamento de Engenharia de Computação e Automação, pelos ensinamentos, desafios propostos e, principalmente, pela dedicação.

Aos amigos e amigas que se acostumaram com minha ausência e me perdoaram nas inúmeras vezes que ouviram de mim um "não posso, preciso estudar", e ainda assim se fizeram presentes quando eu precisei deles.

Finalmente, meu sincero agradecimento aos colegas de curso dessa verdadeira família "EngComp", em especial aos "manolos", companheiros de madrugadas e fins de semana de estudo no DCA. Vocês foram essenciais para a minha formação.

[...]

"The brick walls are there for a reason. The brick walls are not there to keep us out. The brick walls are there to give us a chance to show how badly we want something. Because the brick walls are there to stop the people who don't want it badly enough. They're there to stop the other people."

Resumo

Este trabalho apresenta o desenvolvimento de uma aplicação *Web* utilizada no controle de um sistema de tanques acoplados da Quanser, que permite acesso remoto controlado e o armazenamento de informações relativas à execução de experimentos de controle na planta, tais como sintonia do controlador, resposta da planta e demais variáveis medidas e calculadas. É possível controlar a planta manualmente ou de forma automática através de controladores do tipo PID, portanto uma fundamentação teórica sobre esses controladores também é apresentada. A comunicação entre a aplicação desenvolvida e a planta faz uso de um *gateway* disponível no laboratório e a implementação foi realizada na linguagem Java, utilizando os *frameworks* Java Server Faces, Spring e Hibernate.

Palavras-chave: Laboratório Remoto; Controlador PID; Java Web; Spring.

Abstract

This work presents the development of a Web application applied on the control a Quanser coupled tank system, which allows remote access and the storage of information, related to the execution of level control experiments, such as the controller's tuning, plant response and other measured and calculated signals. It is possible to control the plant manually or automatically using PID controllers, so the basic theory of these controllers is also presented on this work. The communication between the application and the plant is made through a gateway available in the laboratory and the implementation was done in the Java language, making use of Java Server Faces, Spring and Hibernate frameworks.

Keywords: Remote Laboratory; PID Controller; Java Web; Spring.

Sumário

Sumário	i
Lista de Figuras	iii
1 Introdução	1
1.1 Objetivos	2
1.2 Estrutura do documento	2
2 Arquitetura de Software	4
2.1 Decisões de projeto	4
2.1.1 Java Web	4
2.2 Tecnologias utilizadas	5
2.2.1 Tomcat 7.0	5
2.2.2 Java Server Faces 2.1	5
2.2.3 Primefaces 3.4	6
2.2.4 Spring 3.1	6
2.2.5 Spring Security	6
2.2.6 PostgreSQL 9.2	6
2.2.7 Hibernate 4.1	7
2.3 Divisão em camadas	7
2.3.1 Principais classes	11
2.4 Controle de acesso	12
2.5 O banco de dados	14
2.5.1 Configuração do <i>data source</i>	15
3 Arquitetura de Controle	17
3.1 Sistema em malha aberta	18
3.2 Controladores	19
3.2.1 Controladores PID	19

4	Descrição da Planta	22
4.1	Descrição da planta	22
4.2	Esquema de conexão	23
5	O WebLab	25
5.1	Experimentos	25
5.1.1	Cadastro	25
5.1.2	Listagem	25
5.1.3	Exclusão	25
5.1.4	Execução	26
5.1.5	Visualização de resultados	26
5.2	Usuários	28
5.2.1	Cadastro	28
5.2.2	Listagem	28
6	Conclusão	30
6.1	Dificuldades encontradas	30
6.2	Trabalhos futuros	30
6.2.1	Aprimoramento do supervisor	30
6.2.2	Diferentes técnicas de agendamento	31
6.2.3	Análise de resultados	31
6.2.4	Aprimoramento do <i>hardware</i> e do esquema de conexão	31
6.2.5	Calibração automática dos sensores	32
6.2.6	<i>Upload</i> de sinais de controle	32
	Referências bibliográficas	33
A	Criação de novas classes	34
A.1	Managed Beans	34
A.2	Objetos persistentes e DAO's	35

Lista de Figuras

1.1	Arquitetura típica de um laboratório remoto.	2
2.1	Arquitetura em camadas.	7
2.2	Arquitetura em camadas modificada.	8
2.3	O padrão <i>Model View Controller</i>	9
2.4	Diagrama UML das classes de domínio.	10
2.5	Tela de autenticação do WebLab.	13
2.6	Tela de acesso negado.	15
2.7	Diagrama entidade-relacionamento.	16
4.1	Foto da planta de nível.	22
4.2	Configurações disponíveis.	23
4.3	Representação do esquema de comunicação.	23
5.1	Cadastro de experimento.	26
5.2	Execução de experimento.	27
5.3	Visualização dos resultados de um experimento.	27
5.4	Cadastro de usuário.	28
5.5	Listagem de usuários.	29

Capítulo 1

Introdução

A utilização de laboratórios é essencial para um melhor entendimento sobre os diversos conceitos da teoria de controle de sistemas dinâmicos. Além de melhorar o entendimento sobre a teoria, as práticas de laboratório apresentam diversos desafios presentes em aplicações reais, tais como problemas numéricos, imperfeições no modelo da planta, ruído dos sensores e atrasos de transporte.

Contudo, os custos da aquisição e manutenção dos diversos equipamentos necessários a um laboratório são elevados. Isto pode acarretar na utilização de laboratórios virtuais [Gomes e García-Zubía 2007], que consistem de modelos computacionais que visam simular processos reais. Além do custo tipicamente menor, outra vantagem da utilização de ambiente de simulação é a não existência da possibilidade do usuário danificar os equipamentos, e a operação incorreta do sistema não traz riscos ao próprio operador. Entretanto, principalmente em algumas áreas específicas, como na teoria de sistemas de controle, a substituição de experimentos em processos físicos reais por simulações computacionais traria prejuízos ao processo de aprendizagem.

Nesse contexto, uma alternativa razoável é o uso de laboratórios remotos, cujos controle e supervisão dos processos físicos são mediados via um computador servidor conectado a uma rede específica de comunicação, disponibilizando acesso remoto aos experimentos reais do laboratório, como ilustrado na figura 1.1. Uma vantagem no quesito custo é que um mesmo laboratório pode ter seu uso compartilhado por diversas instituições de ensino, reduzindo o custo por aluno. Além disso, laboratórios remotos não possuem as restrições típicas de tempo e lugar, apresentando assim maior disponibilidade.

Vários laboratórios remotos com implementações e recursos diversos têm sido propostos [Gomes e García-Zubía 2007], por exemplo, o Laboratório de Experimentação Remota (RExLab) da Universidade Federal de Santa Catarina (disponível em <http://rexlab.ararangua.ufsc.br/>) e o WebLab-Deusto da Universidade de Deusto na Espanha (disponível em <https://www.weblab.deusto.es/weblab/client/>).

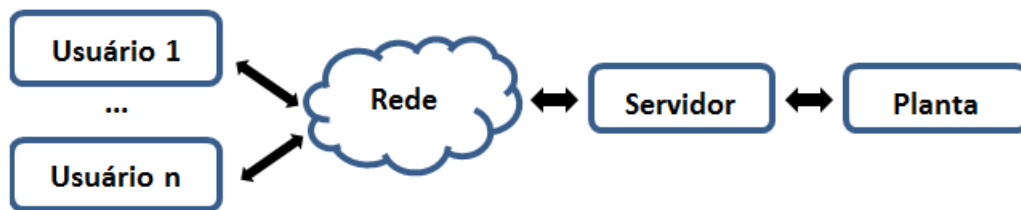


Figura 1.1: Arquitetura típica de um laboratório remoto.

No entanto, vale ressaltar que a experimentação remota não é necessariamente uma atividade de propósito educacional. Na indústria, assim como em centros de pesquisa, o controle remoto de dispositivos através da *internet* também é utilizado. Um exemplo é a Volkswagen que desenvolve laboratórios remotos para testes de motores visando a redução de custos por meio do compartilhamento de sua estrutura laboratorial entre a matriz na Alemanha e uma filial no México [Álvares e Ferreira 2003].

1.1 Objetivos

Os principais objetivos deste trabalho são:

- Desenvolver uma aplicação *web*, denominada "WebLab", que permite o acesso remoto controlado ao sistema de tanques Quanser, disponível no Laboratório de Controle do DCA-UFRN;
- Implementar os controladores a serem utilizados no controle da planta;
- Utilizar um banco de dados para armazenar dados de usuários e dados relativos à execução dos experimentos.
- Utilizar técnicas de engenharia de *software* com o objetivo de desenvolver um sistema com arquitetura suficientemente flexível para adaptar o sistema para plantas diferentes da usada no desenvolvimento deste trabalho;
- Documentar os aspectos relevantes no desenvolvimento da aplicação a fim de que aprimoramentos do sistema e novas funcionalidades possam ser implementadas em trabalhos futuros até mesmo por outros alunos.

1.2 Estrutura do documento

Este trabalho está estruturado em seis capítulos. Nesse primeiro capítulo, foi realizada uma breve discussão sobre a utilização de diferentes tipos de laboratórios, além

disso, foram apresentados os principais objetivos do trabalho. O capítulo 2 descreve aspectos relacionados ao desenvolvimento do *software*, como tecnologias e técnicas de projeto adotadas; o capítulo 3 apresenta basicamente uma fundamentação teórica sobre os controladores implementados; já no capítulo 4, é descrita a planta utilizada e o esquema de comunicação entre a aplicação desenvolvida e a planta; o capítulo 5 ilustra as principais funcionalidades do sistema. Por fim, o capítulo 6 apresenta a conclusão e algumas propostas de trabalhos futuros.

Capítulo 2

Arquitetura de Software

2.1 Decisões de projeto

Optou-se por utilizar somente tecnologias de licença *open source*.

Diversas tecnologias podem ser utilizadas na implementação de clientes para laboratórios remotos. Pode-se dividir essas tecnologias em:

- Aplicações *desktop*: executadas no computador do cliente.
- Aplicações *Web*: executadas no servidor e acessadas por um navegador *Web* no computador do cliente.

As aplicações *desktop* geralmente são capazes de implementar mais funcionalidades que aplicações *Web*, porém esse tipo de aplicação requer que o usuário instale o *software* em seu computador, motivo pelo qual optou-se por descartar a utilização desta abordagem. Dentre as diversas tecnologias disponíveis para o desenvolvimento de aplicações *Web* [Gomes e García-Zubía 2007], Java Web foi a escolhida.

Java é uma linguagem de programação orientada a objetos de alto nível. Segundo o índice TIOBE¹, é uma das linguagens de programação mais utilizadas no mundo. Possui uma extensa biblioteca padrão e vasta documentação. As aplicações desenvolvidas em Java podem ser executadas na maioria dos sistemas operacionais.

2.1.1 Java Web

Uma aplicação Java Web consiste de um conjunto de arquivos de configuração, classes compiladas, bibliotecas e de recursos estáticos e dinâmicos, sendo todos tratados como uma unidade em um *container* de *servlets*.

¹O índice TIOBE é um indicador da popularidade de linguagens de programação. Pode ser acessado em <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

Uma *servlet* é uma classe Java que lê e processa os dados da requisição (*request*) de um cliente e devolve uma resposta (*response*), como uma página HTML. As *servlets* possibilitam a geração de conteúdo dinâmico a partir de classes Java.

A especificação Java Servlet define um *Web Container*, uma aplicação *Web* e uma API *Wervlet*, que une o *container* à aplicação.

Um *container* de *servlets* é análogo a um servidor *Web*, porém esse também é capaz de gerenciar aplicações *Web*. O container provê serviços que dão suporte à API *servlet* que é usada pela aplicação para interagir com requisições e respostas HTTP.

O *Web Container* é o componente do servidor que interage com as *servlets* Java. Ele é responsável por gerenciar o ciclo de vida de *servlets* e mapear uma URL para uma determinada *servlet*. É ele que implementa o componente *Web* do contrato da especificação Java EE.

Tipicamente, uma aplicação *Web* é "empacotada" em um arquivo WAR (*Web ARchive*), que nada mais é do que um arquivo compactado com uma estrutura de diretórios bem definida. O diretório "WEB-INF" contém os arquivos de configuração, o diretório "lib" contém as bibliotecas (empacotadas em arquivos JAR) e o diretório "classes" contém as classes compiladas.

2.2 Tecnologias utilizadas

As principais tecnologias utilizadas são brevemente descritas nas seções seguintes.

2.2.1 Tomcat 7.0

O Apache Tomcat é um servidor *Web* e *container* de *servlets* desenvolvido pela Apache Software Foundation, que implementa a especificação Java Servlet da Sun Microsystems e provê um ambiente *Web* para executar códigos Java.

2.2.2 Java Server Faces 2.1

JavaServer Faces (JSF) é um *framework* padrão Java para construção de interfaces de usuário em aplicações *Web*. JSF simplifica o desenvolvimento da interface de usuário, fazendo uso de componentes gráficos reutilizáveis em uma página, facilitando a implementação de uma das partes mais difíceis e tediosas no desenvolvimento de uma aplicação *Web* [Burns e Schalk 2010]. A partir da versão 2.0 o JSF incorpora suporte nativo a Ajax

e sua especificação define um conjunto padrão de componentes e fornece ainda uma API para o desenvolvimento de novos componentes.

2.2.3 Primefaces 3.4

PrimeFaces é uma popular biblioteca de componentes JSF que possui fácil instalação, além de contar com uma comunidade ativa de usuários e incluir um rico conjunto de componentes, tais como editores HTML (*HyperText Markup Language*), gráficos dinâmicos e suporte a recursos multimídia.

2.2.4 Spring 3.1

Spring é um *framework* de aplicação para ambientes Java que inclui módulos para controle de transação, interação com ferramentas de persistência, desenvolvimento *Web*, entre outros. Suas principais características são injeção de dependência e programação orientada a aspectos [Walls 2011].

2.2.5 Spring Security

O Spring Security é um *framework* baseado no Spring que provê ferramentas para o gerenciamento de autorização e autenticação tanto no nível de requisições *Web* quanto no nível de invocação de métodos [Walls 2011].

2.2.6 PostgreSQL 9.2

O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional, utilizado nas mais diversas aplicações, desde as mais simples até grandes aplicações corporativas, possuindo alta escalabilidade. É multiplataforma (roda nos principais sistemas operacionais) e possui interface nativa de programação para a linguagem Java, além de vasta documentação. Possui diversos recursos sofisticados [<http://www.postgresql.org/docs/9.2/static/index.html>] , tais como:

- Controle de concorrência multiversionado;
- Recuperação em um ponto no tempo;
- Suporte a conjunto de caracteres internacionais;
- Replicação assíncrona;
- Herança múltipla.

Mais informações podem ser encontradas em: .

2.2.7 Hibernate 4.1

O Hibernate é uma ferramenta de mapeamento objeto-relacional para ambientes Java, ou seja, realiza o mapeamento de classes Java em tabelas de um banco de dados relacional e tipos de dados Java em tipos de dados SQL, além de fornecer facilidades de consulta. Mais informações em: <http://docs.jboss.org/hibernate/orm/4.1/manual/en-US/html/>.

2.3 Divisão em camadas

Na programação orientada a objetos, uma camada pode ser definida como um conjunto de classes que são responsáveis pela realização de tarefas específicas e bem definidas. A disposição de um programa em camadas pode ser entendida conceitualmente como uma pilha, representada pela Figura 2.1.

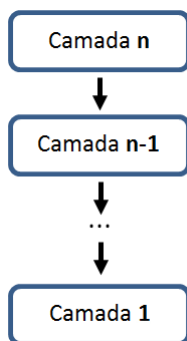


Figura 2.1: Arquitetura em camadas.

A camada n utiliza os serviços providos pela camada $n - 1$, mas a camada $n - 1$ independe e não precisa ter conhecimento da camada n . Contudo, variações desse esquema de comunicação inter-camadas podem ser adotadas permitindo, por exemplo, os seguintes cenários (Figura 2.2):

- Uma camada $n - 1$ pode enviar notificações para uma camada n .
- Uma camada n pode se comunicar com uma camada que é mais de um nível abaixo dela.

A separação de um *software* em camadas é uma maneira de reduzir a complexidade do sistema, proporcionando diversas vantagens [Buschmann et al. 1996], tais como:

- Reduzir a dependência de implementações específicas, pois é possível substituir camadas por implementações alternativas que forneçam os mesmos serviços;

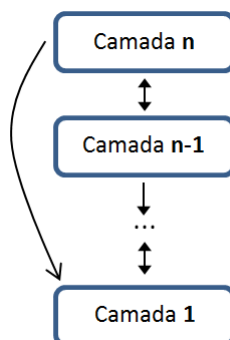


Figura 2.2: Arquitetura em camadas modificada.

- Reuso de *software*. Uma camada bem definida e bem documentada pode ser utilizada em diversos sistemas;
- Facilidade no entendimento do sistema.

Por outro lado, também pode-se identificar algumas desvantagens nessa abordagem, como por exemplo:

- Uma mudança no comportamento de uma camada pode implicar mudanças em diversas outras camadas;
- Maior custo computacional.

O padrão *Model View Controller*

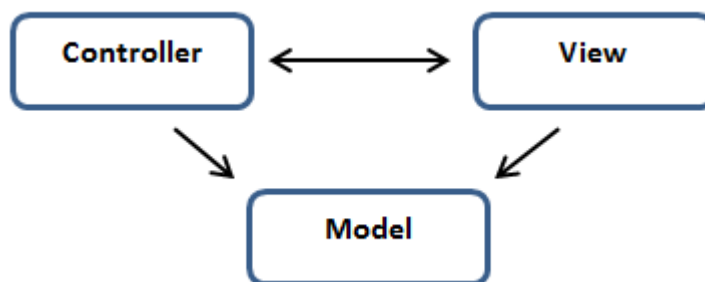
A arquitetura do WebLab utiliza o padrão *Model View Controller* (MVC), que pode ser definido como um padrão de design que separa um software em três camadas:

- 1) **Model ou Modelo** - contém informações sobre o domínio, incluindo seus dados e comportamentos.
- 2) **View ou Apresentação** - responsável pela visualização dos dados do *model*.
- 3) **Controller ou Controle** - realiza a interação entre a camada *Model* e a *View*, manipulando o *model* e atualizando a *view*.

Esse padrão provê uma clara separação entre as camadas de apresentação e modelo, ilustrada na Figura 2.3.

Camada de domínio

É a camada de modelo do MVC. Contém as classes que modelam o domínio do sistema, representados pelas seguintes entidades:

Figura 2.3: O padrão *Model View Controller*.

- Usuário;
- Experimento;
- Dados de execução do experimento;
- Dados de visualização do experimento;
- *Slot* de tempo;
- Controlador manual (controle em malha aberta);
- Controlador PID;
- Planta.

Um diagrama de classes UML² simplificado - incluindo as principais classes, métodos e atributos - é apresentado na Figura 2.4.

Camada de persistência

Responsável pelas operações relativas à interação entre as entidades do sistema e o banco de dados, tais como consulta e persistência de dados. Implementa o padrão DAO³, que tem como objetivo separar as regras de negócio das regras de acesso a dados.

Além disso, essa camada possui uma interface chamada "GenericDAO" que deve ser implementada por todas as classes da camada DAO do sistema. Ela define métodos genéricos que são aplicáveis a qualquer objeto persistente⁴.

A interface "GenericDAO" é implementada pela classe "GenericDAOHibernate", que é estendida pelas demais classes DAO do sistema, como "ExperimentDAO" e "UserDAO", que necessitam de operações mais específicas, como validar o cadastro de um experimento ou usuário. A classe "GenericDaoHibernate" faz uso do Hibernate para a consulta e persistência dos dados.

²Do inglês, "Unified Modeling Language".

³do inglês, "Data Access Objects".

⁴Objeto persistente é aquele que deve ser armazenado no banco de dados.

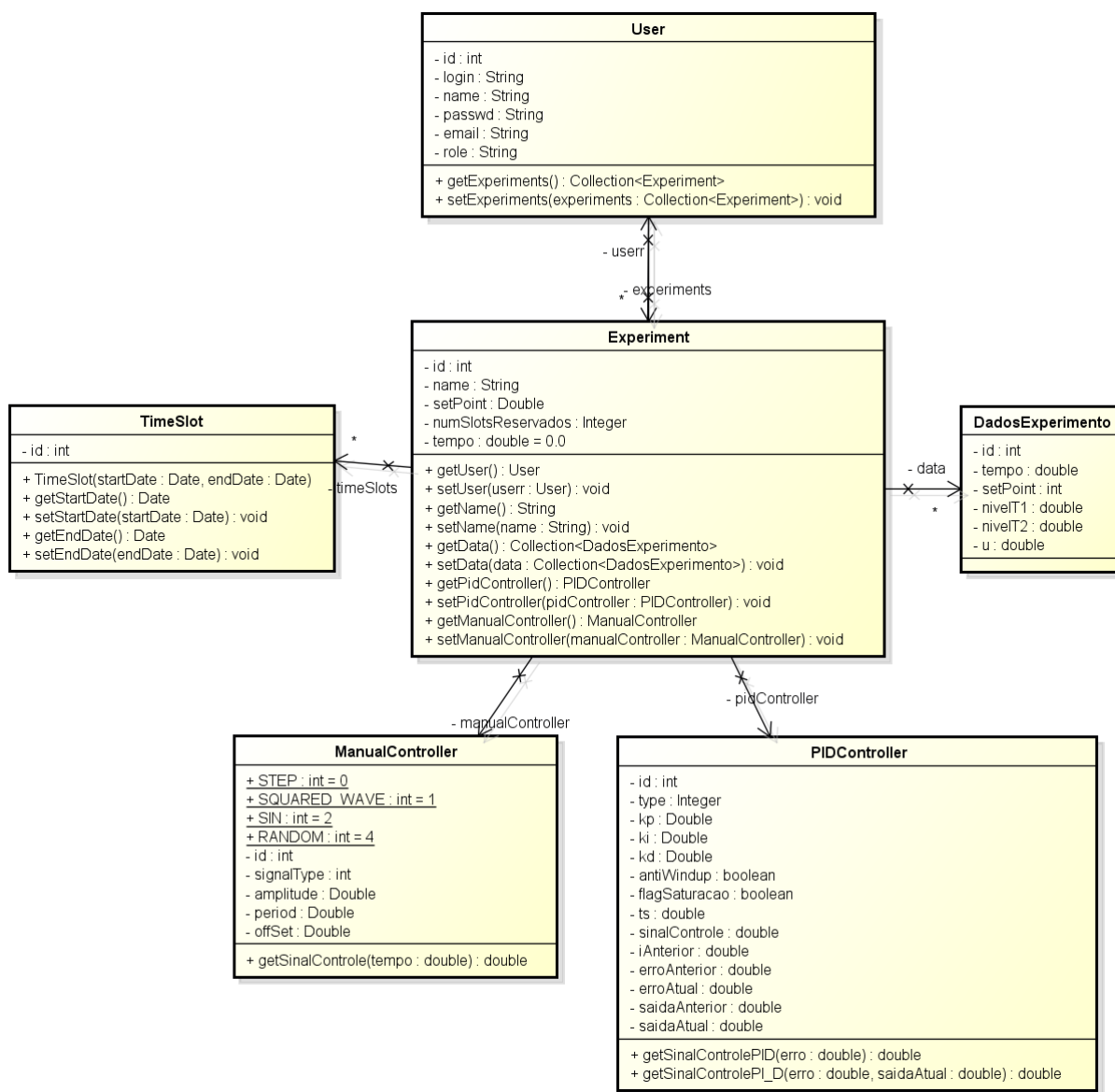


Figura 2.4: Diagrama UML das classes de domínio.

Camada de serviços

A camada de serviços contém a classe "ServiceCadastro", que é responsável por fornecer funcionalidades de acesso à camada DAO em operações transacionais, utilizando o controle de transação fornecido pelo Spring.

Contém ainda a classe "ServiceCadastraAdmin". Essa classe é utilizada por uma *servlet* que verifica, na inicialização da aplicação no servidor, se existe pelo menos um usuário administrador cadastrado no banco de dados. Caso contrário, é criado um usuário padrão com *login* "admin" e senha "admin".

Camada de controle

Contém os *Managed Beans* "UserMBean" e "ExperimentMBean", responsáveis pelo CRUD⁵ de usuário e de experimento, respectivamente. É equivalente à camada de controle do MVC. Essa camada contém ainda uma classe "AbstractMBean" que possui comportamento comum a todos os *Managed Beans* do sistema e, portanto, é estendida por todos eles.

Camada de apresentação

Essa é a camada de apresentação do MVC e contém as páginas XHTML da aplicação e os arquivos de folhas de estilo (CSS). Utiliza diversos componentes do PrimeFaces.

2.3.1 Principais classes

A seguir, serão descritas algumas classes do sistema que modelam características específicas da planta utilizada e do esquema de conexão apresentados neste trabalho.

Classe "TanquesAcoplados"

Representa a planta, um sistema de tanques acoplados, provendo métodos para obtenção de informações sobre a planta, como níveis dos tanques e sinal de tensão. É responsável por saturar o sinal de tensão a ser enviado à planta, além de implementar travas de segurança de nível que consistem em desligar a bomba em situações onde ocorrem comportamentos inadequados, tais como bombear água quando o tanque já está cheio ou drenar água quando o tanque estiver vazio.

⁵Do inglês, "Create-Retrieve-Update-Delete", sigla comumente utilizada para indicar as operações de inserção, busca, atualização e exclusão de uma entidade no banco de dados

Classe "ThreadControle"

A classe "ThreadControle" utiliza a classe "Timer"⁶ para implementar um timer responsável pelas ações de controle que, quando ativado, é executado a cada período de amostragem. Essa classe ainda é responsável pela interação entre supervisor, controlador e planta.

Classe "GerenciadorConexaoPlanta"

Essa classe foi criada com o objetivo de gerenciar a interação entre a aplicação *Web* e a placa de aquisição, através de um *gateway*. Fornece métodos para leitura e escrita na placa de aquisição de dados e garante que só existe uma conexão entre a aplicação *Web* e placa de aquisição, evitando assim que mais de um usuário do sistema possa se comunicar com a planta simultaneamente.

2.4 Controle de acesso

O controle de acesso da aplicação é baseado em papéis, que definem um conjunto de responsabilidades ou permissões atribuídas a um usuário. No WebLab, foram definidos dois papéis: usuário administrador e usuário comum. Ao usuário comum, é permitido:

- Gerenciar seus próprios experimentos: criar, editar, executar e visualizar resultados;
- Cadastrar e executar seus próprios experimentos;
- Salvar e visualizar os dados de execução de seus experimentos;
- Editar dados pessoais.

Ao usuário administrador, além das mesmas permissões de um usuário comum, lhe é possível:

- Cadastrar e remover usuários;
- Listar e/ou excluir qualquer experimento;
- Visualizar os resultados qualquer experimento.

O *framework* Spring Security é utilizado para realizar o controle de autenticação e autorização. Autenticação é o processo que busca verificar a identidade digital de um usuário, em geral, através de um nome de usuário e senha, contudo outros métodos de validação podem ser utilizados, tais como impressão digital e padrão de voz. Por sua vez,

⁶Mais informações em <http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Timer.html>

a autorização verifica se o usuário, uma vez autenticado, possui permissão apropriada para executar a operação desejada.

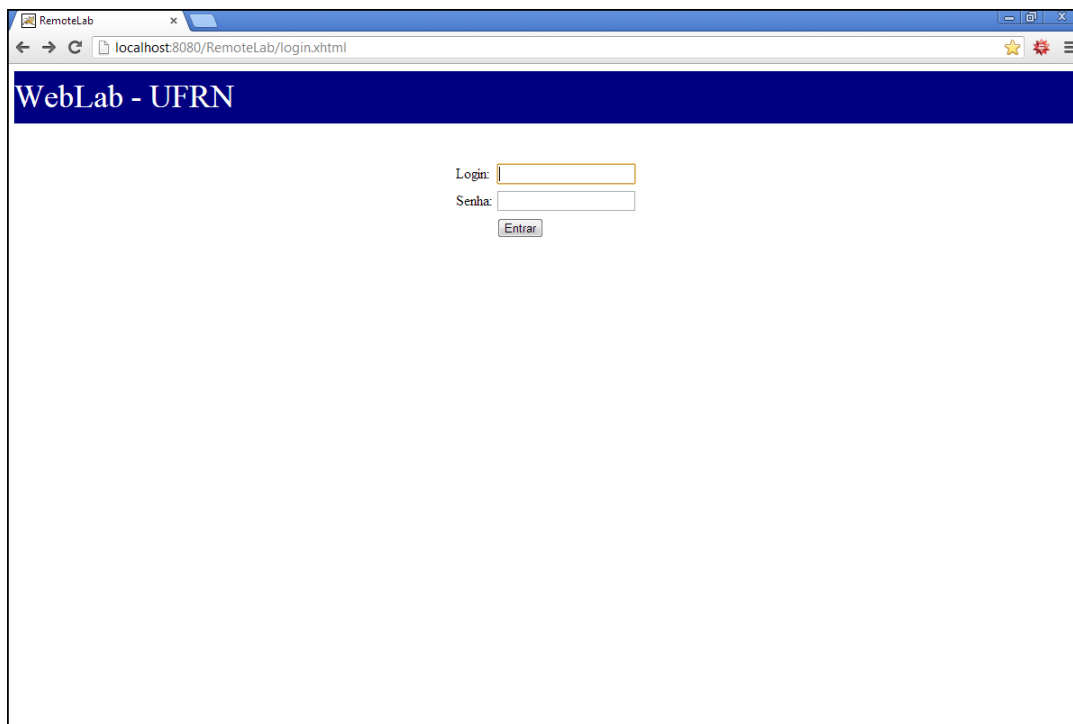


Figura 2.5: Tela de autenticação do WebLab.

Para entrar no sistema, o usuário deve autenticar-se fornecendo *login* e senha (Figura 2.5), que são comparados com os dados que foram informados no cadastro desse usuário, armazenados no banco de dados. A listagem 2.1 mostra como é feito o controle de autorização no Spring Security.

Listagem 2.1: Trecho do arquivo "applicationContext-security.xml"

```
<authentication-manager>
  <authentication-provider>
    <password-encoder hash="md5" />
    <jdbc-user-service id="userService"
5    data-source-ref="dataSource"
      users-by-username-query="SELECT login AS username, passwd AS
        password, 'true' AS enable FROM usuario WHERE login=?"
      authorities-by-username-query="SELECT login AS username, role AS
        authority FROM usuario WHERE login=?" />
    </authentication-provider>
  </authentication-manager>
```

Ainda no arquivo "applicationContext-security.xml", são definidos os padrões de URL que podem ser acessados por cada usuário a partir do papel que lhe foi atribuído, listados a seguir:

- A página de autenticação (/login.xhtml) pode ser acessada por qualquer usuário, autenticado ou não.
- As páginas sob o diretório /admin só podem ser acessadas por usuários administradores.
- Demais páginas podem ser acessadas por qualquer usuário autenticado.

A seguir, um trecho do arquivo de configuração que permite definir o controle de autorização no Spring Security é apresentado:

Listagem 2.2: Trecho do arquivo "applicationContext-security.xml"

```
<http auto-config="true" use-expressions="true">
  <form-login login-page="/login.xhtml" default-target-url="/index.
    xhtml"
    authentication-failure-url="/login.xhtml?login_error=1" />
  <logout logout-success-url="/login.xhtml" invalidate-session="true" /
    >
5  <intercept-url pattern="/login.xhtml*" access="permitAll" />
  <intercept-url pattern="/admin/**" access="hasRole('ADMIN')" />
  <intercept-url pattern="/**" access="hasAnyRole('USUARIO', 'ADMIN')"
    />
</http>
```

Caso um usuário tente acessar uma página sem possuir a permissão adequada, ele é direcionado para uma página que informa que ele não pode acessar o recurso desejado (Figura 2.6).

2.5 O banco de dados

Um banco de dados é utilizado para armazenar diversas informações da aplicação:

- Dados dos usuários;
- Informações sobre sintonia dos controladores cadastrados;
- Resultados da execução de experimentos.

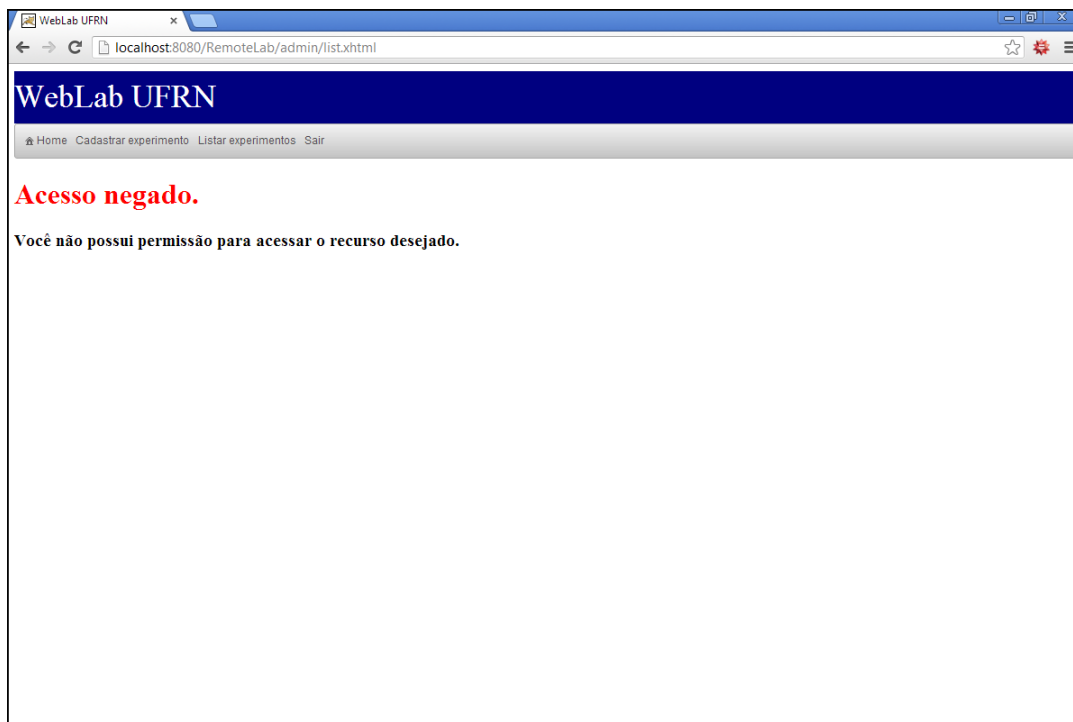


Figura 2.6: Tela de acesso negado.

2.5.1 Configuração do *data source*

Um *data source* é um mecanismo que permite que o servidor gerencie o acesso à conexão com o banco de dados. Para utilizar as funcionalidades de persistência que o Spring oferece, é necessário declarar e configurar um *data source* em um arquivo de configuração "applicationContext.xml", mostrado na listagem 2.3. O arquivo "db-info.properties" contém os parâmetros de configuração necessários à conexão.

Listagem 2.3: Trecho do arquivo "applicationContext.xml"

```
<bean id="propertiesPlaceholder"
    class="org.springframework.beans.factory.config.
        PropertyPlaceholderConfigurer" lazy-init="false">
    <property name="location" value="/WEB-INF/db-conf/db-info.
        properties" />
</bean>
s <bean id="dataSource" class="org.springframework.jdbc.datasource.
    DriverManagerDataSource">
    <property name="driverClassName">
        <value>${driver}</value>
    </property>
    <property name="url">
```

```

10         <value>${ banco }</ value>
    </ property>
        <property name="username">
            <value>${ usuario }</ value>
        </ property>
15    <property name="password">
        <value>${ senha }</ value>
    </ property>
</ bean>

```

Por fim, o diagrama entidade-relacionamento da Figura 2.7 representa as tabelas do banco de dados.

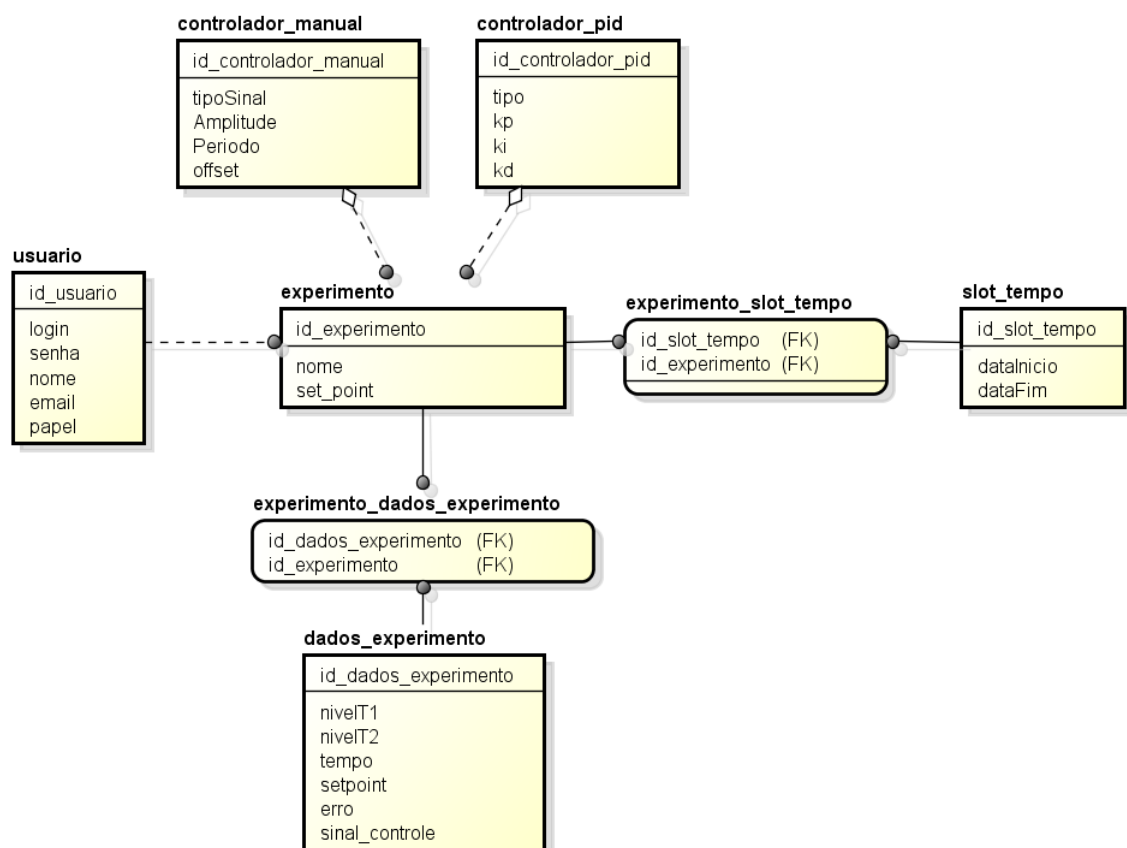


Figura 2.7: Diagrama entidade-relacionamento.

Capítulo 3

Arquitetura de Controle

Devido à possibilidade de ocorrer atraso ou falha na transferência de dados e até mesmo a perda de comunicação entre cliente e servidor durante a execução do experimento, o sinal de controle é calculado localmente no servidor.

Uma requisição feita ao servidor a cada segundo, durante a execução do experimento, possibilita que os dados de execução localizados no servidor sejam obtidos para a visualização no navegador *Web* do cliente em tempo real.

Apesar de não haver prejuízos para a planta, é desejável que um experimento tenha sua execução interrompida no caso de perda de conexão entre cliente e servidor. Para assegurar isso, foi implementado um mecanismo de desligamento automático baseado na ideia de *watchdog*.

Esse mecanismo funciona da seguinte forma: existe uma variável denominada "watchdog" cujo valor é atribuído no início da execução do experimento. A cada período de amostragem, o valor dessa variável é decrementado em uma unidade, funcionando como um contador regressivo. Se o valor dessa variável chegar a zero, é enviado um sinal para que a planta seja desligada. Assim, para evitar o desligamento, o cliente deve, a cada requisição feita durante a execução, reinicializar o valor dessa variável. Para calcular seu valor inicial, considera-se o período de amostragem da planta e o tempo entre cada requisição ao servidor.

Como o período de amostragem da planta é de 0.1 segundos, a variável "watchdog" é decrementada em uma unidade nesse período. Esse período de amostragem é equivalente a um décimo do tempo de cada requisição entre cliente e servidor. Assim, um valor inicial igual a vinte para a variável watchdog garante que após dois segundos sem que o servidor receba uma requisição do cliente, a variável atinja o valor zero, o que ocasiona o desligamento da planta.

Os passos realizados no laço do algoritmo de controle são basicamente:

1. Ler valores da planta

2. Calcular ação de controle
3. Aplicar sinal de controle à planta
4. Enviar dados ao supervisor
5. Decrementar valor da variável "watchdog" e verificar se esse valor é maior que zero

3.1 Sistema em malha aberta

O sistema pode operar em modo malha aberta. Nesse caso, um sinal específico é gerado e enviado à planta. A seguir, são descritos os sinais que podem ser gerados.

Degrau

Por definição, o sinal degrau é dado pela equação:

$$u(t) = \begin{cases} A, & \text{se } t \geq 0; \\ 0, & \text{se } t < 0. \end{cases} \quad (3.1)$$

onde A é uma constante que determina a amplitude do sinal. No caso particular em que $A = 1$, esse sinal é denominado **degrau unitário**.

Senoide

Uma senoide é um sinal periódico que pode ser expresso na forma:

$$u(t) = A \sin(\omega t + \phi) \quad (3.2)$$

onde A é a amplitude do sinal e $\omega = 2\pi f$ é a frequência angular.

Onda quadrada

A onda quadrada é um sinal de período T que pode ser representado pela equação abaixo:

$$u(t) = \begin{cases} -A, & \text{se } -\frac{T}{2} \leq t \leq 0; \\ A, & \text{se } 0 \leq t \leq \frac{T}{2}. \end{cases} \quad (3.3)$$

Para cada um desses sinais, é possível configurar um valor de *offset*, que é um valor a ser somado na amplitude do sinal. Outros tipos de sinal, como sinal aleatório ou dente de serra, podem ser facilmente acrescentados posteriormente, se necessário.

3.2 Controladores

Existem diversas técnicas de controle, tais como controle adaptativo, controle preditivo e controle inteligente. O tipo de controlador utilizado neste trabalho é o Proporcional-Integral-Derivativo, ou simplesmente PID. Os controladores PID possuem grande utilidade, pois são capazes de resolver diversos problemas de controle, e são os mais empregados nos problemas de controle industrial [Aström e Murray 2012].

3.2.1 Controladores PID

O controlador PID gera uma ação de controle u que depende de três termos:

- Um termo proporcional ao erro;
- Um termo proporcional à integral do erro;
- Um termo proporcional à derivada do erro.

A equação abaixo apresenta uma descrição matemática desse tipo de controlador.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (3.4)$$

onde o erro $e(t)$ é definido como a diferença entre o sinal de referência e a saída do sistema. As constantes k_p , k_i e k_d são denominadas, respectivamente, ganho proporcional, ganho integral e ganho derivativo.

Aplicando a transformada de Laplace em (3.4), obtém-se a função de transferência $G_c(s)$:

$$G_c(s) = \frac{U(s)}{E(s)} = k_p + \frac{k_i}{s} + k_d s$$

Essa equação deve ser discretizada a fim de ser implementada em um computador digital. Foram feitas aproximações discretas para os termos derivativo e integrativo. Para o termo derivativo, utilizou-se a discretização conhecida como *backward*, que é feita da seguinte forma:

$$u(kT) = \frac{1}{T} (e(kT) - e[(k-1)T])$$

cuja transformada Z é:

$$\frac{z-1}{Tz} E(z)$$

O termo integrativo foi discretizado pelo método conhecido como *forward*:

$$u(kT) = u[(k-1)T] + Te(kT)$$

onde $u(kT)$ é a saída do integrador no instante $t = kT$ e cuja transformada Z é dada por

$$U(z) = z^{-1}U(z) + TE(z)$$

e sua função de transferência:

$$\frac{U(z)}{X(z)} = \frac{Tz}{z-1}$$

Portanto, a função de transferência discreta do controlador PID é dada por:

$$G_c(z) = k_p + \frac{k_i T z}{z-1} + k_d \frac{z-1}{T z} \quad (3.5)$$

Seja $x(kT) = x(k)$. Aplicando a transformada Z inversa em (3.5), obtém-se uma equação de diferenças que representa o controlador PID:

$$u[k] = K_p e[k] + K_i (u[k-1] + T e[k]) + K_d \left(\frac{e[k] - e[k-1]}{T} \right) \quad (3.6)$$

Existem diversos aspectos práticos relacionados ao modo como o PID é implementado em computadores digitais que podem influenciar significativamente o desempenho do sistema de controle, como filtragem do termo derivativo e transferência suave de modo manual para automático.

Além disso, existem limites de operação relacionados aos atuadores como, por exemplo, o nível máximo de tensão que pode ser aplicado à bomba no sistema de tanques acoplados. Logo, o sinal de controle calculado pelo controlador pode ter seu valor saturado de acordo com a faixa de operação do atuador. Essa saturação pode ser descrita matematicamente pela equação 3.7.

$$u(t) = \begin{cases} u_{min}, & \text{se } u(t) < u_{min}; \\ u(t), & \text{se } u_{min} \leq u(t) \leq u_{max}; \\ u_{max}, & \text{se } u(t) > u_{max}. \end{cases} \quad (3.7)$$

onde u é o sinal de controle calculado e u_{min} e u_{max} representam os limites de operação do atuador.

Quando ocorre a saturação, o valor do termo integral pode se tornar muito grande, gerando respostas oscilatórias. É o chamado efeito *wind-up*. Uma estratégia simples para evitar o efeito *wind-up*, adotada neste trabalho, consiste em desabilitar a soma do termo integral quando o sinal de controle sofre saturação.

Outro problema está relacionado a variações do sinal de referência. Essas variações implicam em valores muito altos para o termo derivativo devido à mudança praticamente instantânea do erro. O esquema adotado, denominado PI-D, consiste em calcular o termo derivativo levando em conta a saída ao invés do erro, considerando que nessas situações a saída geralmente varia mais lentamente que o sinal de erro.

Capítulo 4

Descrição da Planta

4.1 Descrição da planta

A planta utilizada para a realização dos testes neste trabalho é o Sistema de Tanques Acoplados da Quanser Consulting Inc. que consiste de dois tanques acoplados, uma bomba elétrica e um reservatório de água.

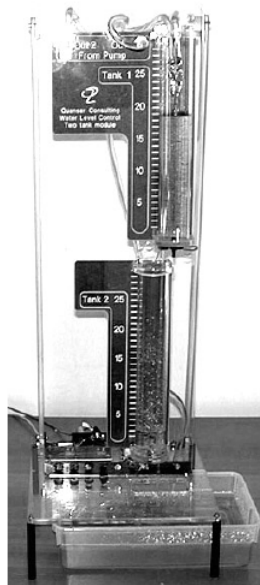


Figura 4.1: Foto da planta de nível.

Um exemplar deste sistema está disponível no Laboratório de Controle do Departamento de Engenharia de Computação da Universidade Federal do Rio Grande do Norte e ele é utilizado pelos professores e alunos do departamento no desenvolvimento e aplicação de diversas técnicas de controle, servindo tanto para fins de pesquisa quanto para o melhor aprendizado proporcionado pela realização de um experimento prático.

A água é bombeada do reservatório para dois orifícios normalmente fechados, "Out1" e

"Out2" que possuem diâmetros diferentes. Os dois tanques estão dispostos na parte frontal do sistema, conectados aos orifícios através de tubos de borracha e são dispostos de maneira que o fluxo de saída do tanque superior flua para o tanque inferior e que o fluxo de saída do tanque inferior flua para o reservatório de água. A vazão de saída dos tanques e a forma de controle da bomba sobre os mesmos podem ser configuradas de três formas diferentes, conforme apresentadas na Figura 4.2.

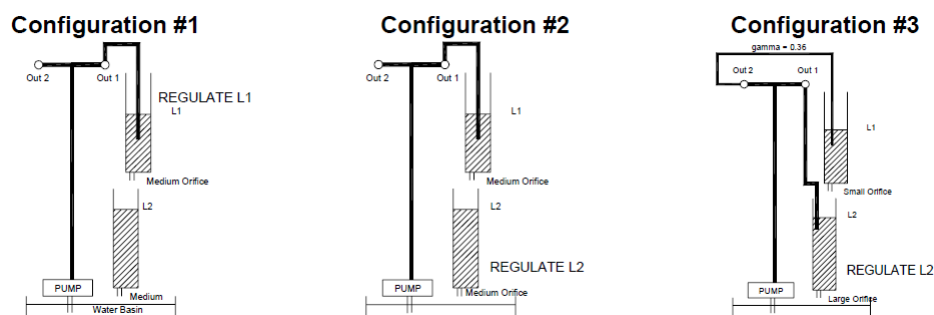


Figura 4.2: Configurações disponíveis.

As configurações que podem ser utilizadas neste trabalho são as duas primeiras, que se caracterizam como um sistema de primeira ordem e segunda ordem, respectivamente.

4.2 Esquema de conexão

O esquema de comunicação do usuário com a planta pode ser representado pela Figura 4.3.

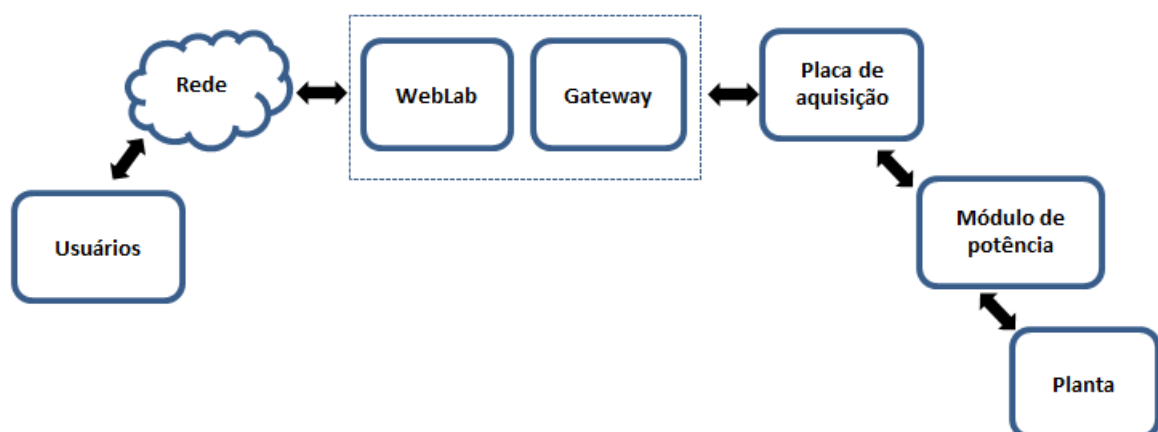


Figura 4.3: Representação do esquema de comunicação.

Rede

Rede baseada em tecnologia TCP/IP, pode ser local ou até mesmo a Internet.

Gateway

Provê um serviço de conexão TCP via *socket* capaz de receber comandos em alto nível e traduzi-los para comandos de baixo nível responsáveis pela comunicação com a placa de aquisição. Mais detalhes disponíveis em [Oliveira 2008].

Placa de aquisição

Conectada ao computador onde o gateway é executado, é responsável pela conversão AD/DA.

Módulo de potência

Fornece energia para os sensores e amplifica o sinal de saída da placa de aquisição, que será utilizado para acionar a bomba.

Capítulo 5

O WebLab

Este capítulo ilustra as funcionalidades do sistema desenvolvido, denominado WebLab.

5.1 Experimentos

5.1.1 Cadastro

Para cadastrar um novo experimento (ver Figura 5.1), o usuário deve acessar o *link* "Cadastrar experimento" no menu principal da aplicação.

É necessário informar um nome para o experimento, o modo de controle, que pode ser "manual" (malha aberta) ou "automático" (malha fechada), e escolher um horário para sua execução. É possível visualizar os horários disponíveis e os já reservados por outro usuário.

5.1.2 Listagem

Para listar os experimentos, o usuário deve acessar o *link* "Listar experimentos". Se o usuário possuir papel de administrador, é possível visualizar todos os experimentos cadastrados. Caso ele possua papel de usuário comum, só é possível visualizar seus próprios experimentos.

5.1.3 Exclusão

Para excluir um experimento, basta acessar o *link* para listar experimentos e clicar na opção "Excluir experimento". Usuários administradores podem excluir qualquer experimento; usuários comuns só podem excluir seus próprios experimentos.

WebLab UFRN

Home Cadastrar experimento Listar experimentos Cadastrar usuário Listar usuários Sair

Tipo de controlador

Nome:

Modo: Automático

Parâmetros do controlador

Setpoint: 15

Tipo de controlador: Seleccione... Kp: Ki: Kd:

Reserva de horário

Jun 2 — 8 2013

	Dom 6/2	Seg 6/3	Ter 6/4	Qua 6/5	Qui 6/6	Sex 6/7	Sab 6/8
08:00		Reservado: 08:00-08:59					
09:00		Reservado: 09:00-09:59					
10:00							
11:00			Reservado: 11:00-11:59				
12:00				Reservado: 12:00-12:59			
13:00				Reservado: 13:00-13:59			
14:00			Reservado: 14:00-14:59				
15:00							
16:00							
17:00							
18:00							
19:00							
20:00							
21:00							
22:00							

Cancelar Salvar

Figura 5.1: Cadastro de experimento.

5.1.4 Execução

No horário reservado, o usuário deve acessar o sistema e escolher o experimento a ser executado. Os sinais medidos e calculados são apresentados de forma gráfica durante a execução do experimento e é possível modificar a sintonia do controlador de forma *online*.

Ao clicar no botão "Salvar"(Figura 5.2), é possível armazenar os dados de execução do experimento em banco de dados para posterior visualização. Esses dados consistem dos parâmetros do controlador escolhido e de sinais lidos e calculados, como níveis dos tanques, sinal de referência e sinal de controle aplicado à planta.

5.1.5 Visualização de resultados

É possível visualizar os dados de execução de um experimento (Figura 5.3) ao clicar no botão "Visualizar dados" disponível no *link* de listagem de experimentos, para cada experimento cujos dados tenham sido salvos pelo usuário.

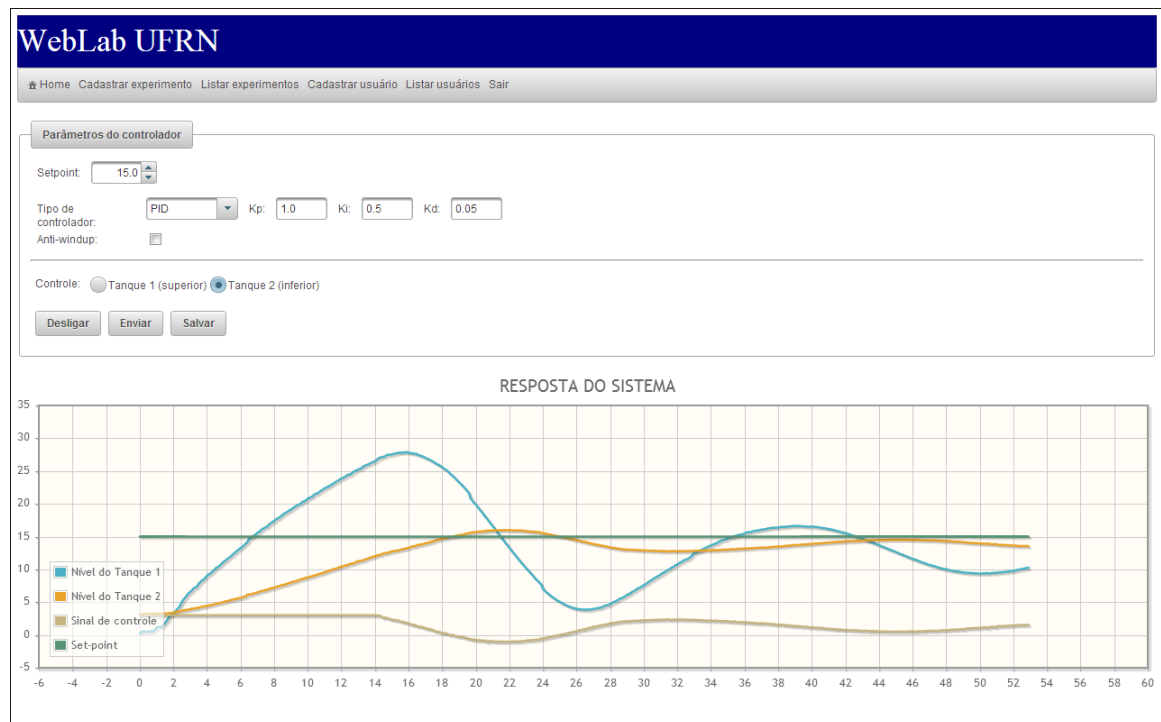


Figura 5.2: Execução de experimento.

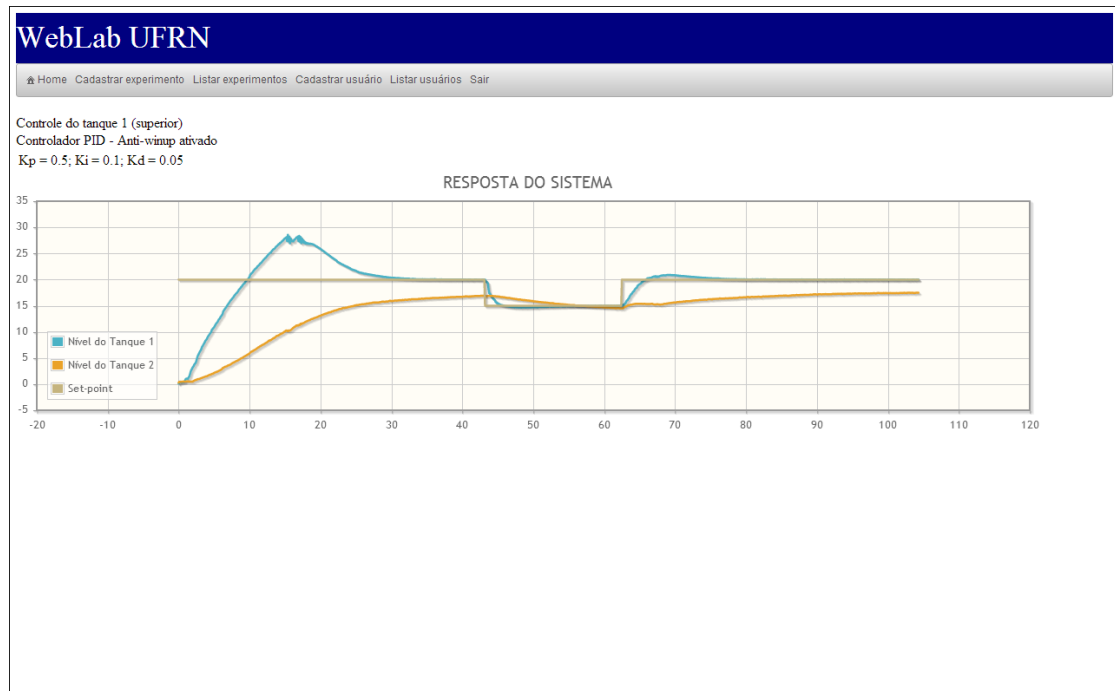
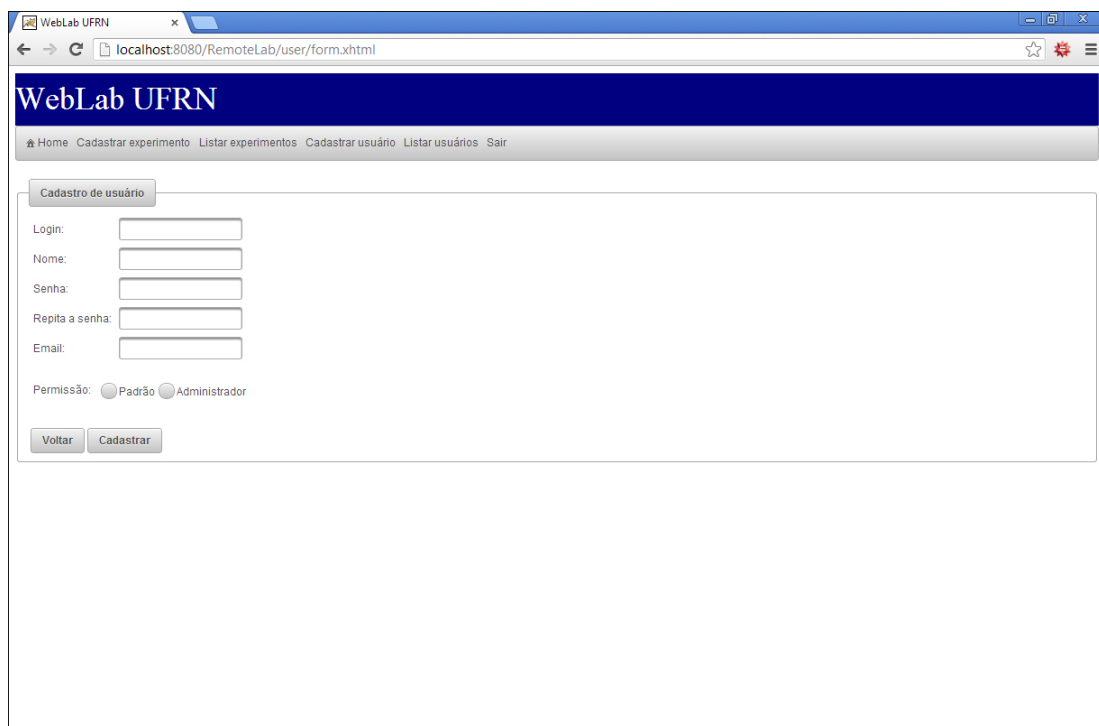


Figura 5.3: Visualização dos resultados de um experimento.

5.2 Usuários

5.2.1 Cadastro

Para cadastrar um usuário, o administrador deve acessar o *link* "Cadastrar usuário" do menu principal (Figura 5.4). É necessário informar *login*, senha e o tipo de permissão do novo usuário.

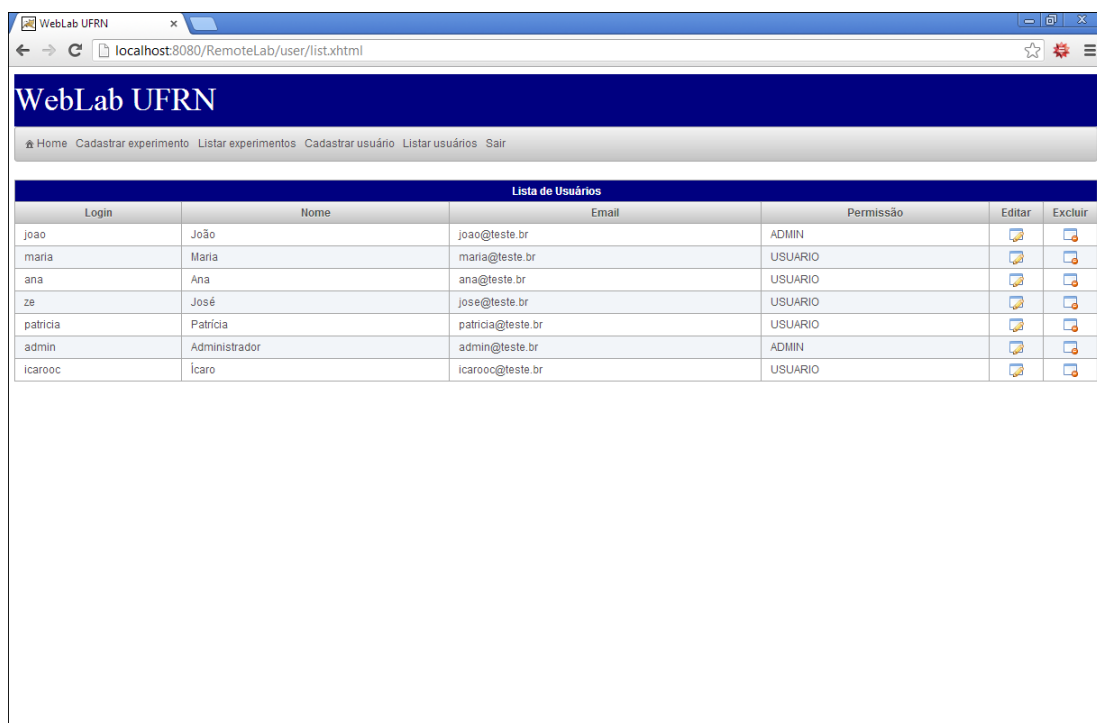


The screenshot shows a web browser window titled "WebLab UFRN" with the address bar displaying "localhost:8080/RemoteLab/user/form.xhtml". The page features a dark blue header with the "WebLab UFRN" logo and a navigation menu with links: Home, Cadastrar experimento, Listar experimentos, Cadastrar usuário, Listar usuários, and Sair. The main content area is titled "Cadastro de usuário" and contains a registration form with the following fields: Login, Nome, Senha, Repita a senha, and Email. Below these fields is a "Permissão" section with two radio buttons: "Padrão" (selected) and "Administrador". At the bottom of the form are two buttons: "Voltar" and "Cadastrar".

Figura 5.4: Cadastro de usuário.

5.2.2 Listagem

Para listar os usuários, o administrador deve acessar o *link* "Listar usuários" do menu principal (Figura 5.5).

















Lista de Usuários					
Login	Nome	Email	Permissão	Editar	Excluir
joao	João	joao@teste.br	ADMIN		
maria	Maria	maria@teste.br	USUARIO		
ana	Ana	ana@teste.br	USUARIO		
ze	José	jose@teste.br	USUARIO		
patricia	Patrícia	patricia@teste.br	USUARIO		
admin	Administrador	admin@teste.br	ADMIN		
icaroor	Ícaro	icaroor@teste.br	USUARIO		

Figura 5.5: Listagem de usuários.

Capítulo 6

Conclusão

Ao analisar este trabalho, pode-se observar que o principal objetivo foi cumprido. A aplicação desenvolvida disponibiliza acesso remoto controlado ao sistema de tanques disponível no laboratório, permitindo realizar o controle da planta e armazenar os dados de execução dos experimentos.

A maior contribuição deste trabalho foi o desenvolvimento de uma arquitetura de *software* que pode ser facilmente reaproveitada em trabalhos futuros, graças à preocupação em documentar os aspectos relevantes da implementação, à aplicação de técnicas de reuso de *software* e à utilização de *frameworks* bem consolidados no mercado.

6.1 Dificuldades encontradas

Um tempo consideravelmente alto foi gasto na instalação, configuração e utilização dos *frameworks* usados na implementação deste trabalho, tarefas que demandam um nível de conhecimento relativamente alto, principalmente com relação ao Spring, que aplica técnicas como injeção de dependência e programação orientada a aspectos.

6.2 Trabalhos futuros

Durante a implementação do sistema, pôde-se observar diversas necessidades apresentadas por um ambiente de experimentação remota. As principais necessidades identificadas são apresentadas nesta seção.

6.2.1 Aprimoramento do supervisor

Um ponto muito importante a ser melhorado é a inclusão de uma funcionalidade de *streaming* de vídeo, para que seja possível exibir uma representação visual da planta du-

rante a execução do experimento, aumentando assim a experiência com o usuário.

6.2.2 Diferentes técnicas de agendamento

Como o gerenciamento de uso da planta é fundamental, em trabalhos futuros uma maior atenção deve ser dada à questão do agendamento de experimentos. Algumas estratégias consideradas, porém não implementadas, são as seguintes:

- Incluir no sistema uma área onde o administrador possa escolher a duração de cada slot de tempo, o número máximo permitido por usuários, entre outros parâmetros;
- Excluir automaticamente uma reserva de horário caso o usuário que fez a reserva não execute o experimento no horário determinado;
- Punir usuários que reservam horário, mas não executam os experimentos, impedindo que esses reservem novos horários durante um determinado período de tempo. Essa medida teria como objetivo incentivar o bom uso do laboratório.

6.2.3 Análise de resultados

Uma funcionalidade importante a ser disponibilizada ao administrador do sistema, do ponto de vista da avaliação dos resultados obtidos pelos usuários, seria uma área no site onde pudessem ser visualizadas medidas de desempenho dos controladores como, por exemplo, menor tempo de acomodação, menor erro e menor sobressinal.

Além da análise dos resultados, também seria interessante apresentar estatísticas de uso do laboratório, tais como horários de maior demanda e usuários que mais acessam o sistema.

6.2.4 Aprimoramento do *hardware* e do esquema de conexão

Embora a rigor a aplicação seja capaz de ser acessada a partir de qualquer cliente com acesso à *internet*, pode-se afirmar que, com o esquema de comunicação disponível atualmente, não é recomendado o uso remoto da planta sem que sua operação seja supervisionada localmente por operador humano.

Entre os problemas identificados, pode-se destacar frequentes falhas de conexão do *gateway* com a rede e descalibração dos sensores da planta. Este último é uma questão crítica, já que as travas de segurança implementadas em *software* que evitam o transbordamento da água nos tanques dependem exclusivamente da informação de nível fornecida por esses sensores.

6.2.5 Calibração automática dos sensores

Com a utilização de técnicas de processamento de imagem, é possível implementar um algoritmo de calibração "virtual" dos sensores da planta. Isso pode ser feito através de um ajuste linear (mínimos quadrados, por exemplo), utilizando informações da leitura dos sensores e medições visuais dos níveis em pelo menos dois pontos distintos.

6.2.6 Upload de sinais de controle

Tanto no modo de operação em malha aberta quanto em malha fechada, atualmente só é possível utilizar os sinais previamente definidos no desenvolvimento da aplicação. O ideal seria adicionar uma funcionalidade que permitisse ao usuário fazer o *upload* de seus próprios sinais de controle, aumentando a flexibilidade do sistema.

Referências Bibliográficas

- Agrawal, Awkash e Sanjay Srivastava [2007], ‘Weblab: A generic architecture for remote laboratories’, *15th International Conference on Advanced Computing and Communications* .
- Álvares, Alberto José e Joao Carlos Espíndola Ferreira [2003], ‘Metodologia para implantação de laboratórios remotos via internet na área de automação da manufatura’, *2º Congresso Brasileiro de Engenharia de Fabricação* .
- Aström, Karl Johan e Richard M. Murray [2012], *Feedback Systems*, Princeton University Press.
- Burns, Ed e Chris Schalk [2010], *JavaServer Faces 2.0: The Complete Reference*, McGraw-Hill.
- Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sornmerlad e Michael Stal [1996], *Pattern-Oriented Software Architecture*, John Wiley & Sons.
- Dorf, Richard C. e Robert H. Bishop [2011], *Modern Control Systems*, Pearson Education.
- Gomes, Luís e Javier García-Zubía, eds. [2007], *Advances on remote laboratories and e-learning experiences*, Deusto Publicaciones.
- Oliveira, Leonardo Dantas [2008], ‘Desenvolvimento de um gateway de comunicação para interação com plataformas de aquisição e escrita de dados’.
- Walls, Craig [2011], *Spring in Action*, Manning Publications.

Apêndice A

Criação de novas classes

Este apêndice tem o objetivo de documentar detalhes relevantes à criação de objetos em diversas camadas do sistema. Espera-se, assim, facilitar a implementação de novas funcionalidades do sistema em trabalhos futuros utilizando a arquitetura já desenvolvida.

A.1 Managed Beans

A listagem A.1 apresenta um exemplo de Managed Bean com escopo de sessão. A anotação `@ManagedProperty` é utilizada para permitir que os objetos "ServiceCadastro" e "ExperimentDAO" sejam instanciados através de injeção de dependência. Os *setters* desses atributos são obrigatórios.

Listagem A.1: Exemplo de Managed Bean

```
import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
// Outros import..

5
@ManagedBean
@SessionScoped
public class ExemploMBean extends AbstractMBean implements Serializable
{

10    @ManagedProperty(value = "#{serviceCadastro}")
    private ServiceCadastro serviceCadastro;

    @ManagedProperty(value = "#{experimentDAO}")
    private ExperimentDAO dao;

15    public void setServiceCadastro(ServiceCadastro serviceCadastro) {
```

```

        this.serviceCadastro = serviceCadastro;
    }

20    public void setDao(ExperimentDAO dao) {
        this.dao = dao;
    }

}

```

A.2 Objetos persistentes e DAO's

O container do *framework* Spring é responsável por obter e gerenciar uma sessão do Hibernate, portanto é necessário informar no arquivo de configuração desse *framework*, o "applicationContext.xml", os pacotes que contém os objetos persistentes e DAO's do sistema.

A listagem A.2 apresenta uma forma de declarar as unidades persistentes, nesse caso, todas as classes contidas no pacote br.ufrn.engcomp.domain.

Listagem A.2: Exemplo de Managed Bean

```

<bean id="sessionFactory" class="org.springframework.orm.hibernate3.
    annotation.AnnotationSessionFactoryBean">
    <property name="packagesToScan">
        <list>
            <value>br.ufrn.engcomp.domain</value>
5        </list>
    </property>

    // ...
</bean>

```

Para criar um classe DAO, a tag **context:component-scan** é utilizada no applicationContext e a anotação **@Repository** é incluída na classe, conforme exemplificado nas listagens A.3 e A.4.

Listagem A.3: Exemplo de Managed Bean

```

<context:component-scan base-package="br.ufrn.engcomp.dao" />

```

Listagem A.4: Exemplo de Managed Bean

```

import org.springframework.stereotype.Repository;
// Outros import...

```

```
@Repository
5 public class ExemploDAO extends GenericDAOHibernate {
    // ...
}
```
