UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM SISTEMA DE GERÊNCIA DOS DADOS COLETADOS EM EXPERIMENTOS MICROMETEOROLÓGICOS – SGDM

TRABALHO DE GRADUAÇÃO

Leandro de Almeida Rodrigues

Santa Maria, RS, Brasil

2007

DESENVOLVIMENTO DE UM SISTEMA DE GERÊNCIA DOS DADOS COLETADOS EM EXPERIMENTOS MICROMETEOROLÓGICOS - SGDM

por

Leandro de Almeida Rodrigues

Trabalho de Graduação apresentado ao Curso de Bacharelado em
Ciência da Computação, Área de Concentração em Banco de Dados,
da Universidade Federal de Santa Maria (UFSM, RS),
como requisito para obtenção do grau de
Bacharel em Ciência da Computação

Orientadora: Profª. Roseclea Duarte Medina
Co-orientadores: Prof. Osvaldo L. L. Moraes e Prof. Otávio Costa Acevedo

Trabalho de Graduação Nº 236
Santa Maria, RS, Brasil

2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UM SISTEMA DE GERÊNCIA DOS DADOS COLETADOS EM EXPERIMENTOS MICROMETEOROLÓGICOS - SGDM

elaborado por
Leandro de Almeida Rodrigues

como requisito final para obtenção do grau de
Bacharel em Ciência da Computação

Comissão Examinadora:

Roseclea Duarte Medina, Drª.
(Presidente/Orientadora)

Iria Brucker Roggia, Msc. (UFSM)

Oni Reasilvia Sichonany, Msc. (UFSM)

Santa Maria, 24 de agosto de 2007.

AGRADECIMENTOS

Dedico este trabalho de graduação a todas as pessoas, em especial a meus familiares e amigos, que me apoiaram na trajetória de estudante, desde os tempos de ensino médio e principalmente nos anos de curso superior.

RESUMO

Trabalho de Graduação
Curso de Bacharelado em Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM SISTEMA DE GERÊNCIA DOS DADOS COLETADOS EM EXPERIMENTOS MICROMETEOROLÓGICOS - SGDM

AUTOR: Leandro de Almeida Rodrigues
ORIENTADOR: Roseclea Duarte Medina
CO-ORIENTADORES: Osvaldo L. L. Moraes e Otávio C. Acevedo
Data e Local da Defesa: Santa Maria, 24 de Agosto de 2007.

Este trabalho de graduação consiste em desenvolver um sistema para gerenciar os dados coletados em experimentos micrometeorológicos, com freqüência de 1 e 10 hertz, através de sensores instalados em torres, onde são realizados os experimentos do Grupo de Modelagem Atmosférica (GRUMA) da Universidade Federal de Santa Maria. O estudo de caso do GRUMA apontou as necessidades de criar uma aplicação que fosse capaz de armazenar, proteger e disponibilizar os dados de forma simples e eficaz, com a finalidade de minimizar o tempo de seleção e consulta dos dados pelos pesquisadores vinculados ao grupo. Para isso foi utilizado um conjunto de ferramentas que melhor se adequaram às necessidades apontadas pelo estudo de caso. Entre elas, foram utilizados a linguagem PHP na versão 5.0, o sistema de gerenciamento de banco de dados MySQL na versão 5.0, instalados num servidor *web* Apache na versão 2.0, além de outros recursos necessários para a criação da aplicação. O gerenciamento que foi desenvolvido na aplicação descrita neste trabalho, consiste em gerenciar o banco de dados do GRUMA através da criação de novas tabelas para armazenar os dados dos experimentos já realizados, bem como adicionar novos dados aos experimentos em andamento, e ainda, prover um sistema eficaz de seleção e recuperação dos dados contidos no banco de dados.

Palavras-chave: banco de dados; PHP; MySQL; micrometeorologia

ABSTRACT

Graduation Work
Course of Computer Science
Federal University of de Santa Maria

DEVELOPMENT OF A MANAGEMENT SYSTEM FOR DATA COLLECTED AT MICROMETEOROLOGICAL EXPERIMENTS - SGDM

AUTHOR: Leandro de Almeida Rodrigues
ADVISOR: Roseclea Duarte Medina
CO-ADVISOR: Osvaldo L. L. Moraes e Otávio C. Acevedo
Local and Defense Date: Santa Maria, August 27, 2007.

This paper is focused on the development and management system for micrometeorological experimental data. This micrometeorological data is collected in frequencies of 1 and 10 Hertz by high response meteorological sensors located at the tower which belongs to the Atmospheric Modeling Group of Santa Maria/Federal University of Santa Maria (GRUMA/UFSM), and are used in their field experiments. The large amount of high frequency observational data, demands a fast and safe data storage system able to disseminate and promote easy data exchange between the GRUMA researchers and associates researchers. In this sense, some tools as PHP 5.0 language, data management system MySQL 5.0. These tools are installed at an Apache 2.0 web server. To supply the data management demand on GRUMA was developed a micrometeorological data management able to create new table database, and select and append data to the old database existent.

Keywords: database; PHP; MySQL; micrometeorology

Lista de Figuras

Figura 1: Esquema e funcionamento do SGBD em um sistema de banco de dados.....15
Figura 2: Integração PHP, servidor *web* e o retorno na forma de html.....20
Figura 3: Desenho esquemático de uma torre de coleta.....27
Figura 4: Fotografia de uma torre Micrometeorológica de 15m.....28
Figura 5: Datalogger - Campbell 21XL 28
Figura 6: Exemplo dos dados coletados em experimentos micrometeorológicos 29
Figura 7: Autenticação no servidor.....30
Figura 8: Demonstração da seqüência de cadastro de um novo experimento.....31
Figura 9: Adicionando dados em um experimento já cadastrado 32
Figura 10: Consulta de dados passo a passo 34
Figura 11: Arquivo contendo os dados que foram consultados.....35
Figura 12: Diagrama de casos de uso completo do SGDM 36
Figura 13: Diagrama de seqüência para o caso de uso cadastrar experimento 37
Figura 14: Diagrama de seqüência para o caso de uso adicionar dados 37
Figura 15: Diagrama de seqüência para o caso de uso consultar dados.....38
Figura 16: Programação do datalogger utilizado 39
Figura 17: Arquivo de dados utilizados para testar o sistema.....39
Figura 18: Criação do banco de dados 40
Figura 19: *Script* que reconhece os tipos das variáveis 41
Figura 20: Interface *html* gerada a partir da execução do *script* reconhecer_tipos.....42
Figura 21: Comando SQL executado para criação da tabela 43
Figura 22: *Script* de inclusão dos dados em experimentos já cadastrados no banco de dados 44
Figura 23: Códigos que listam das tabelas de experimentos cadastrados no banco de dados 44
Figura 24: Interface *html* gerada a partir da execução do *script* lista_tabela.....45
Figura 25: Códigos do *script* que produz a interface de consulta dos dados 46

Figura 26: Interface mostrada ao usuário após a execução do script selecionar_variavel 47
Figura 27: Códigos PHP do *script* "*selecionar*_dados" 48
Figura 28: Interface *html* apenas com os dados selecionados 49
Figura 29: Esquema de funcionamento do sistema 50

Lista de Abreviaturas

SQL	*System Query Language*
ANSISQL	Padronização textual com notações SQL
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
WWW	*World Wide Web*
PHP	*Hypertext Processor*
HTML	*Hypertext Markup Language*
GRUMA	Grupo de Modelagem Atmosférica
XML	*eXtensible Markup Language*
SGML	*Standa*rd Generalized Markup Language
GPL	*General Public Licence*
SGDM	Sistema Gerenciador de Dados Micrometeorológicos

SUMÁRIO

1 – INTRODUÇÃO 12

2 – REVISÃO DE LITERATURA 14

2.1 – O BANCO DE DADOS 14

2.1.1 - O gerenciador e o administrador do banco de dados 14

2.1.2 - Sistema de Gerenciamento de Banco de Dados - SGBD 15

2.2 - BANCO DE DADOS RELACIONAL 16

2.2.1 - Os SGBD's disponíveis 17

2.2.2 – O SGBD escolhido para este trabalho 17

2.2.3 – Os tipos de tabela do MySQL 17

2.2.4 – O tipo de tabela escolhido 18

2.3 – A INTERFACE DO SISTEMA DE ACESSO AO BANCO DE DADOS 18

2.3.1 – Hypertext Markup Language (HTML) 19

2.3.2 – PHP 19

2.3.2.1 - Uma breve história do PHP 21

2.3.2.2 - Principais características do PHP 21

2.4 – O SERVIDOR WEB APACHE 21

2.5 – O SISTEMA DE SEGURANÇA 22

2.6 – MODELAGEM DO SISTEMA 23

2.6.1 – UML 23

2.6.2 - Fases do Desenvolvimento de um Sistema em UML 24

2.6.2.1 - Análise de Requisitos 24

2.6.2.2 - Análise 24

2.6.2.3 - Design do projeto 25

2.6.2.4 - Programação 25

2.6.2.5 - Testes 25

3 – ESTUDO DE CASO: SISTEMA DE GERÊNCIA DOS DADOS COLETADOS EM EXPERIMENTOS MICROMETEOROLÓGICOS - SGDM 26

3.1 – O GRUMA 26

3.2 – OS EXPERIMENTOS MICROMETEOROLÓGICOS E OS DADOS COLETADOS 26

3.2.1 – A coleta dos dados 27

3.2.2 – A padronização dos dados 28

3.3 – SGDM 29

3.3.1 – Os grupos de usuários *29*
3.3.2 – O portal dos usuários do grupo administrador *30*
3.3.2.1 – Cadastro de um novo experimento no banco de dados 30
3.3.2.2 – Adição dos dados na tabela do experimento 32
3.3.3 – O portal dos usuários do grupo pesquisador *32*
3.4 – ETAPAS DE DESENVOLVIMENTO DO SISTEMA 35
3.4.1 – Modelagem do sistema *35*
3.4.2 – Os dados utilizados nos testes do sistema *38*
3.4.3 – Criação do banco de dados *40*
3.4.4 – Programação da interface de criação da estrutura das tabelas de armazenamento *40*
3.4.5 – Programação do script de inclusão dos dados *43*
3.4.6 – Programação do sistema de acesso *44*
3.5 – RESULTADO FINAL DA MODELAGEM DO SISTEMA 50
4 – CONCLUSÕES 51

1 – Introdução

O Grupo de Modelagem Atmosférica (GRUMA) realiza experimentos micrometeorológicos em diversas cidades do estado do Rio Grande do Sul. Segundo Acevedo (2007), uma das fases destes experimentos consiste em adquirir dados atmosféricos que são coletados através de sensores instalados nas torres, que são montadas nos locais onde os experimentos se realizarão.

Na torre existe um equipamento conhecido como datalogger, que é permanentemente conectado aos sensores e tem a função de catalogar os dados coletados pelos sensores, conforme foi definido em Datalogger (2007). Entretanto, os dataloggers não possuem grandes capacidades de armazenamento, logo, é necessário que os dados sejam descarregados em computadores também instalados nos locais dos experimentos. O próximo passo é guardar os dados de forma segura, consistente e persistente, de acordo com Guimarães (2001), para que posteriormente eles sejam disponibilizados aos pesquisadores vinculados ao grupo. Para isso, surge a necessidade de armazená-los definitivamente de forma adequada, o que nos remete instantaneamente a dois fatores relevantes:

- O primeiro deles é que na maioria das vezes, ao final de cada experimento, estes dados são armazenados em mídias removíveis, entre eles, CD's e DVD's, e em alguns casos guardados em discos rígidos sem qualquer forma de organização. É visível que estes tipos de armazenamento não são adequados, segundo Date (1994). No caso das mídias removíveis, tem como problema principal a durabilidade que é pequena, pois leva em conta o uso e o local onde são guardadas comprometendo a persistência, além do pouco espaço de armazenamento que faz com que os dados de um mesmo experimento fiquem armazenados em diversas mídias diferentes. Fator que pode levar a perda de parte dos dados caso uma ou mais mídias sejam extraviadas, além de aumentar o tempo para reagrupar os dados de todas as mídias quando for necessário. Já no caso dos discos rígidos podemos citar o fato de ser impossível conter a replicação de dados, bem como prever uma possível falha de disco que provocaria a perda dos dados, de acordo com Kroenke (1999).
- O segundo é o fato de ser necessário garantir a segurança do acesso e da disponibilização dos dados, segundo Molina (2001), pois em muitos casos estes dados não são de ordem pública, visto que, para a coleta houve custos e

demanda. Assim, para disponibilizá-los aos pesquisadores, é necessário obter um controle sobre os dados.

Para conseguir suplantar estas duas necessidades, o presente trabalho de graduação apresenta a criação de um sistema de banco de dados, denominado SGDM, com uma interface voltada ao usuário, que na maioria dos casos não detém conhecimentos específicos sobre sistemas de banco de dados. Na maior parte dos casos este usuário será um pesquisador cuja necessidade é usar o sistema de banco de dados apenas para aquisição dos dados. Isto remete a necessidade de agilidade na seleção dos dados, pois percebe-se que, na forma de armazenamento através de mídias, os pesquisadores perdem um tempo considerável para localizar e selecionar um período de dados com apenas as variáveis que são relevantes a sua pesquisa. Este tempo é gasto através de programas e planilhas que muitas vezes são complexos, limitados e/ou demorados de usar, dependendo da quantidade de dados que serão manipulados, afirmou Guimarães (2001).

Portanto, esta aplicação fornecerá aos seus usuários todos os módulos necessários para que, após a coleta dos dados, eles tenham um destino confiável e seguro quanto ao seu armazenamento com o objetivo de disponibilizá-los aos pesquisadores de uma forma simples, eficaz e transparente, visando facilitar a separação dos dados que serão úteis em suas pesquisas, definiu Navathe (2001).

Para isso, foi pré-selecionado um conjunto de ferramentas que melhor atende as necessidades do trabalho, levando em conta o desempenho do banco de dados, a portabilidade da linguagem de programação, a segurança do sistema operacional e o menor custo de utilização, ou seja, a licença gratuita da utilização dos softwares, conforme definido em GPL (2000).

2 – Revisão de Literatura

Este trabalho baseia-se nos fundamentos e conceitos da programação voltada a aplicações *web*, integrada com sistema de banco de dados.

2.1 – O banco de dados

O banco de dados, segundo Guimarães (2001), é um recurso valioso que tem levado ao desenvolvimento de uma enorme gama de conceitos e técnicas para o eficiente gerenciamento de informações. São inúmeras e infinitas suas aplicações e usos.

Inicialmente é preciso saber o que se quer realizar e, em seguida, elaborar uma lista de itens (ou dados) que deverão ser incluídos. Depois, é preciso projetar como será o crescimento dos dados que deverão ser mantidos no banco de dados permanentemente para serem manipulados sempre que for preciso.

Outra preocupação é a da eficiência, tanto no nível de abstração físico, como no conceitual e no de visão. São níveis que determinam à exposição dos dados armazenados, sob diferentes estruturas.

Todas essas informações necessitam de um grande espaço de armazenamento. Para administrá-lo usamos um gerenciador de banco de dados. A medida usada é gigabyte. Um gigabyte corresponde a 1.000 megabytes ou 1 bilhão de bytes.

2.1.1 - O gerenciador e o administrador do banco de dados

O gerenciador do banco de dados é responsável por uma série de tarefas: interação com o gerenciador de arquivos, garantia da integridade e de segurança dos dados, recuperação e backup para correção de falhas e controle de concorrência para preservação dos dados.

Mas, para termos um gerenciador de bancos de dados, precisamos também possuir um controle central das informações e dos programas que os acessam. Para isso, precisamos contar com o papel do Administrador do Banco de Dados.

As funções de um administrador do banco de dados, segundo Silberschatz (1995) devem incluir:

- definição do esquema original do próprio banco;
- definição da estrutura de armazenamento e do método de acesso;
- modificação da organização física e do esquema; concessão de autorização para acesso a dados;

- e especificação de restrições de integridade para preservar as informações.

2.1.2 - Sistema de Gerenciamento de Banco de Dados - SGBD

Para Teorey (1999), um sistema gerenciador de banco de dados (SGBD) é um software genérico para manipulação de banco de dados. Um SGBD é capaz de tratar os aspectos de visão lógica e física dos dados, suportar linguagens para definição e manipulação de dados, bem como prover utilitários para o gerenciamento de transações e controle de concorrência. Leite (1980) resume os requisitos fundamentais que um SGBD deve cumprir:

- Permitir a independência de dados;
- controlar a redundância de dados;
- oferecer estruturas de dados compatíveis com as características dos diversos tipos de usuários;
- permitir o acesso concorrente;
- oferecer facilidades que permitam estabelecer o controle de acesso;
- permitir e manter o relacionamento entre os dados;
- garantir a integridade das informações armazenadas;
- ter bom desempenho.

Segundo Silberschatz (1995), o grande objetivo de um SGBD é prover os usuários com uma visão abstrata dos dados. Similarmente aos conceitos de abstração e encapsulamento em sistemas orientados a objeto, cabe ao SGBD omitir certos detalhes de como os dados são armazenados e mantidos, para esconder a complexidade das estruturas de dados utilizadas internamente pelo sistema.

A figura 1 apresenta um esquema de funcionamento do SGBD em um sistema de banco de dados.

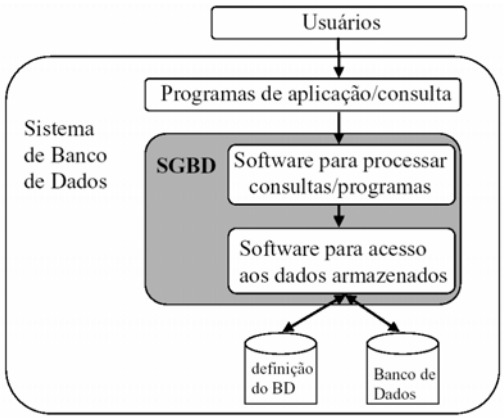


Figura 1: Esquema e funcionamento do SGBD em um sistema de banco de dados

Segundo Ferreira (2003), toda a complexidade de um sistema de banco de dados divide-se em níveis crescentes de abstração. O físico é o nível mais baixo de abstração no qual descreve-se detalhadamente as estruturas de dados que armazenam os dados fisicamente. No nível conceitual abstrai-se as complexidades do nível físico e determina-se apenas quais dados são armazenados no banco e quais as relações entre eles. Por último, no nível de visões, parte-se do princípio que os usuários de um SGBD não estão interessados em todos os dados armazenados e, portanto descreve-se apenas parte dos dados estruturados no nível conceitual.

2.2 - Banco de dados relacional

O modelo relacional de banco de dados foi definido inicialmente por Codd (1982) na década de 1970 e posteriormente bastante estendido por outros autores. Este modelo pode ser definido por pelo menos três características significativas:

- Suas estruturas de dados são simples. Tratam-se de relações que são tabelas de duas dimensões cujos elementos são os itens de dados. Essa característica permite um alto grau de independência da representação física dos dados (arquivos e índices);
- Provê uma fundação sólida para a consistência de dados. O projeto de banco de dados é auxiliado pelo processo de normalização que elimina anomalias de dados. O estado consistente do banco de dados pode ser definido e mantido através de regras de integridade;
- Permite a manipulação das relações sob a forma de conjuntos. Esta característica levou ao desenvolvimento de poderosas linguagens não procedurais baseadas na teoria de conjuntos (álgebra relacional) ou na lógica (cálculo relacional).

Para Ferreira (2003), o sucesso dos sistemas de banco de dados relacionais pode ser atribuído a vários fatores. O modelo relacional é de fácil compreensão e possui uma fundamentação teórica forte. Além disso, foi adotada a linguagem de consultas padronizadas ANSISQL que é completamente integrada aos sistemas de banco de dados. A arquitetura dos SGBDR abrange a gama completa de funcionalidades: definição de dados, manipulação de dados, consulta, controle dos dados, gerenciamento de transações etc. Atualmente, os sistemas de gerenciamento de banco de dados relacionais estão consolidados no mercado e há diversos fabricantes disponibilizando seus produto, afirmou Ferreira (2003).

2.2.1 - Os SGBD's disponíveis

Desde o lançamento do System/R da IBM em 1980, os sistemas de banco de dados relacionais vêm dominando o segmento de gerenciamento de dados. Nos dias de hoje há diversos produtos comerciais de bancos de dados relacionais tais como IBM DB2, Oracle, Informix, Microsoft Access, Microsoft SQL Server, Borland Interbase, MySQL, PostgreSQL.

2.2.2 – O SGBD escolhido para este trabalho

A implementação deste trabalho utilizará um SGBD que seja de livre distribuição e licenciado pela GPL (2000). Na comunidade de software livre, há duas opções de bancos de dados que são amplamente utilizados: MySQL (2004) e PostgreSQL (2005).

Neste trabalho foi utilizado o MySQL por diversos fatores, dentre os quais é possível destacar:

- desempenho favorável;
- boa documentação;
- boa portabilidade;
- grande utilização no mercado;
- grande quantidade de recursos;
- iteração com a linguagem PHP;
- iteração com o servidor *web*.

2.2.3 – Os tipos de tabela do MySQL

O MySQL trabalha de uma maneira diferente de outros bancos de dados ele possui o recurso *"Tipo de tabela"* ou *"Table handler"*. Para Carmo (2003), escolher corretamente o tipo de tabela é importante para que se possa utilizar poder total do MySQL.

O MySQL suporta dois tipos diferentes de tabelas: tabelas seguras com transação (InnoDB and BDB) e tabelas não seguras com transação HEAP, ISAM, MERGE, e MyISAM), de acordo com Mysql (2007).

Vantagens de tabelas seguras com transação (TST):

- É mais segura, ou seja, mesmo se o MySQL falhar ou se você tiver problemas com hardware, você pode ter os seus dados de volta, ou através de recuperação automática (*crash recovery*) ou de um backup incluindo o log de transação.
- É possível combinar muitas instruções concorrentemente e gravar todas de uma só vez com o comando `COMMIT`.

- Você pode ignorar suas mudanças (se você não estiver rodando em modo auto-commit) com o comando `ROLLBACK`.
- Se uma atualização falhar, todas as suas mudanças serão restauradas.
- Este tipo de tabela fornece um melhor esquema de concorrência no caso dos dados da tabela serem atualizados concorrentemente com leituras.

Vantagens de tabelas não seguras com transação (NTST):

- Além de ser muito mais rápida, não há nenhuma sobrecarga de transação.
- Este tipo de tabela usa muito menos espaço em disco já que não há nenhuma sobrecarga de transação.
- Usará menos memória para as atualizações.

2.2.4 – O tipo de tabela escolhido

No MySQL é possível combinar tabelas TST e NTST na mesma instrução para obter o melhor dos dois tipos de transação, entretanto, neste trabalho foi utilizado como padrão o tipo InnoDB que é um tipo de TST, pois possui o suporte a transação e a integridade referencial (*cascade refers*) bem como o suporte a "*crash recovery*" no caso de uma queda de luz ou falha física, e também a maneira de gravação e manipulação dos dados, são fatores relevantes que levaram a escolha das TST e não das NTST, mesmo estas sendo mais rápidas, ocupando menos espaço em disco e usando menos memória durante as atualizações.

2.3 – A interface do sistema de acesso ao banco de dados

Como não há local determinado de onde o sistema do banco de dados será utilizado, optamos por utilizar uma interface *web*, para que qualquer usuário que precise utilizar o sistema possa se conectar através da internet, onde quer que esteja, possibilitando total acesso aos recursos de consulta do sistema.

A interface deste banco de dados se enquadra nos moldes da página de acesso do GRUMA, sendo que para implementar o design foi utilizado o *grumaweb template,* foram usadas técnicas XML (*eXtensible Markup Language*), que, segundo W3 (2006), é uma linguagem de marcação definida pela W3C e pode ser definida como um subtipo da linguagem padrão de marcação generalizada, a SGML (*Standard Generalized Markup* Language), cujo objetivo é facilitar o compartilhamento de informações na Internet. Esta técnica permite importar toda a interface usada no portal do GRUMA como forma de padronizar o conteúdo do banco de dados com os demais conteúdos do site.

2.3.1 – Hypertext Markup Language (HTML)

O padrão da *web* para armazenamento e distribuição de documentos eletrônicos, na forma de hipermídia, é a linguagem HTML.

O HTML pode ser definido como um formato de texto com marcações (tags) utilizado para apresentar conteúdo da *web* num navegador (browser). Uma vez que é uma linguagem bastante simples, com apenas alguns “comandos”, o HTML rapidamente tornou-se popular para apresentar documentos formatados independentes de plataforma e de processador de texto.

Devido às limitações do HTML 1.0, muitas extensões foram criadas nos últimos anos. Além das extensões do HTML em si, um conjunto considerável de tecnologias foi criado para dar apoio às aplicações *web* tal como o Javascript, Java, CGI, ASP, JSP, PHP, Shockwave Flash, MP3, CSS e XML por exemplo.

Dentre estas destacam-se as soluções mais recentes para a disponibilização de conteúdo dinâmico na *web* que são linguagens para processamento no servidor tal como o Active Server Pages (ASP), JavaServer Pages (JSP) e Hypertext Processor (PHP) sendo que, segundo Camargo (2005), esta última é atualmente a opção mais amplamente empregada nas aplicações *web*, conforme afirmou.

2.3.2 – PHP

O PHP é uma linguagem baseada na linguagem C que é processada no servidor, ou seja, é aquela que é executada no servidor *web* antes da página ser enviada através da Internet ao cliente. As páginas que executam no servidor podem realizar acessos a banco de dados, conexões em rede, e outras tarefas para criar a página final que será vista pelo cliente. O cliente somente recebe uma página com o código HTML resultante da execução da PHP. Como a página resultante contém apenas código HTML, é compatível com todos os navegadores, conforme Criar Web (2005).

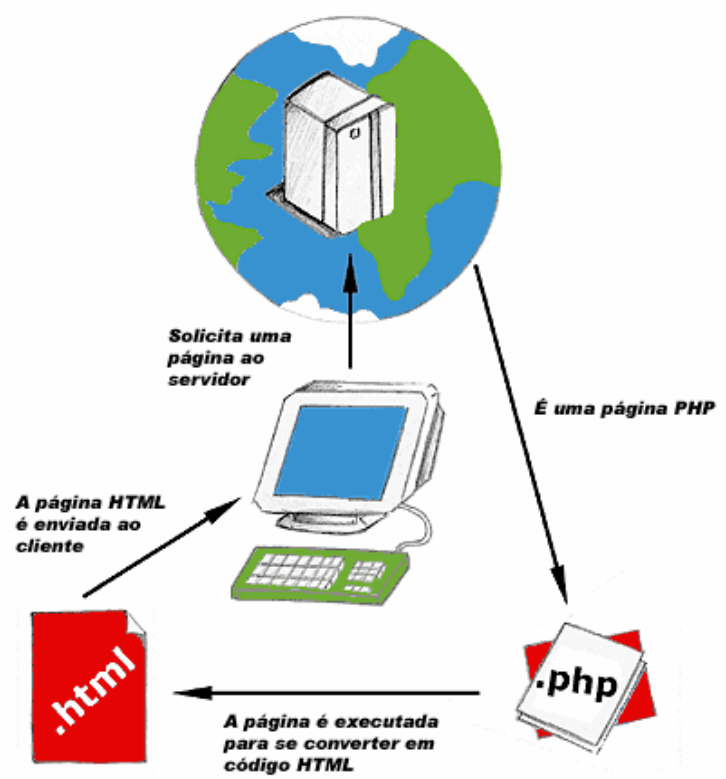


Figura 2: Integração PHP, servidor *web* e o retorno na forma de html

O PHP, atualmente, é a sigla para Hypertext Preprocessor, mas originalmente significou Personal Home Page, e se destaca entre as linguagens citadas anteriormente por ser multiplataforma (enquanto outras rodam somente em uma plataforma), ou seja, aceita vários sistemas operacionais, como Windows, Unix, Linux. Além disso, ela é de fácil aprendizado, pois permite a conexão direta com uma grande quantidade de bancos de dados relacionais, enquanto outras ferramentas precisam de drivers ODBC para realizar a mesma tarefa. Entre os bancos de dados com conexão direta podemos citar: Oracle, Sybase, Informix, Postgresql, MySQL, mSQL. Para outros bancos de dados, segundo Achour (2007), o PHP disponibiliza acesso via ODBC. Como vantagem adicional, o PHP é totalmente gratuito e pode ser baixado por meio de seu site oficial.

A linguagem PHP é uma combinação de linguagem de programação e servidor de aplicações. É possível programar em PHP como em qualquer outra linguagem, definindo variáveis, criando funções, realizando loops, enfim, fazer tudo que é necessário e usado no mundo da programação.

2.3.2.1 - Uma breve história do PHP

O PHP foi criado originalmente por *Rasmus Lerdorf* em meados de 1994 e escrito em Perl, sendo reescrito depois em C para incluir acesso a bancos de dados. Com a propagação dessa ferramenta pelo mundo virtual, Rasmus disponibilizou alguma documentação do software e batizou-o oficialmente de PHP v.1.0, conforme afirmou Achour (2007). Com a crescente utilização do PHP, segundo Alecrim (2003), mais e mais recursos foram incluídos como loops e arrays, por exemplo, tornando a linguagem cada vez mais potente. Nessa época, outros programadores juntaram-se a Rasmus, contribuindo sensivelmente para o aprimoramento da linguagem, entre os quais podemos citar os israelenses Zeev Suraski e Andi Gutmans, e foi assim que nasceu o PHP versão 3.0, segundo Achour (2007), que deu origem a versão 5.0 que foi usado para implementação dos *scripts* deste trabalho.

2.3.2.2 - Principais características do PHP

Podemos citar como uma das principais características do *PHP* o fato desta linguagem consumir poucos recursos do servidor, permitindo que programas complexos sejam desenvolvidos, sem que isto implique em grande demora na sua execução. Além disso, o PHP como módulo nativo do servidor *web*, evita chamadas externas, o que o torna ainda mais eficiente.

Além desta característica, é possível citar ainda a capacidade de ler informação do padrão XML, processamento de arquivos (leitura e gravação, tanto no formato texto quanto binário), a manipulação de variáveis complexas, a utilização de funções e classes e geração de código JavaScript, ou outro qualquer para processamento no lado cliente, a manipulação de e-mails, o gerenciamento de documentos PDF e muitas outras características que tornam o PHP uma linguagem realmente potente e indicada para a construção de sites dinâmicos, conforme afirmou Alecrim (2003).

Portanto, devido a seu grande número de recursos e sobretudo pela sua forte interação com o SGBD selecionado através de uma ampla biblioteca de funções, ela foi escolhida como linguagem de programação para o sistema de acesso ao banco de dados que foi implementado neste trabalho e que será demonstrado nas próximas seções.

2.4 – O servidor *web* Apache

O Apache (2007) é atualmente o servidor *web* mais utilizado no mundo e tem como a principal vantagem o custo, pois é disponibilizado gratuitamente. Ele surgiu em 1995, a partir

da evolução de um servidor *web* chamado NCSA, com a contribuição de voluntários que criaram a *Apache Software Foundation*, Apache (2007). Esse grupo tem sido responsável pelo desenvolvimento constante do software, ao mantê-lo com o status de servidor *web* mais utilizado no mundo. Grande parte desta popularidade deve-se ao fato de ser um software de código aberto, obtido gratuitamente, que permite implementar e adaptar novas características para atender necessidades próprias de cada projeto.

Para Camargo (2005), uma outra importante característica do servidor Apache é a sua arquitetura modular, que aceita a inclusão de praticamente qualquer funcionalidade desejada em um servidor *web*. A sua modularidade associada a um núcleo bastante simples, faz com que ele seja bastante flexível e possa atender necessidades muito específicas de cada página *web*.

2.5 – O sistema de segurança

Os bancos de dados se modificam com o passar do tempo. Neles, as informações podem ser removidas ou acrescidas, atualizadas e incrementadas. O gerenciador do banco de dados é responsável pela iteração com o gerenciador de arquivos, pela garantia de integridade, segurança, pelo *backup* e recuperação de informações e pelo controle de concorrência que garante que o mesmo dado não será modificado por dois processos ao mesmo tempo, de acordo com Carmo (2003).

Para garantir segurança e integridade em um banco de dados foi usado mecanismos que protejam as informações, guardadas no banco de dados, de acessos não-autorizados, de destruição ou alteração intencional, ou inclusão acidental de inconsistências, conforme foi definido por Date (1994).

É impossível a proteção absoluta do Banco de Dados contra abusos intencionais. Mas o custo e a penalidade ao contraventor podem ser aumentados para deter suas ações. Para proteger os bancos de dados podem ser adotadas medidas de segurança em diferentes níveis: no físico, no humano, no sistema operacional e no sistema de banco de dados, afirmou Kroenke (1999).

O primeiro nível, o físico, consiste em manter o servidor num local adequado e seguro; o segundo, o humano, prevê o critério cuidadoso de autorização a usuários para evitar que pessoas estranhas possam ter acesso aos dados.

No nível de sistema operacional, a segurança consiste em evitar qualquer debilidade que permita a entrada de pessoas estranhas aos serviços. Dessa forma, o nível de segurança do software dentro do sistema operacional é tão importante quanto à segurança física. Já o

sistema de banco de dados pode prever a segurança restringindo as informações para alguns usuários e habilitando-as para outros. É responsabilidade do sistema de banco de dados assegurar que essas restrições não sejam violadas.

A segurança dentro do sistema operacional é implementada em diversos níveis, desde códigos para acesso ao sistema até o isolamento de outros processamentos em curso. O sistema de arquivo também permite certo grau de proteção aos dados. É possível obter segurança usando mecanismos que restrinjam a capacidade de visões ou de ocultação de dados aos usuários.

A segurança do sistema de banco de dados relacionais pode ser aplicada em dois níveis: relação, que permite ou nega o acesso direto do usuário a uma relação; visão, que impede ou permite o acesso a dados que apareçam em uma visão. Assim, o usuário pode acessar parte das informações por meio de uma visão. Para Guimarães (2001), uma combinação de segurança em níveis relacionais e de visão pode ser utilizada para limitar o acesso do usuário apenas aos dados necessários a ele. Todos esses serviços podem ser facilmente obtidos de forma simples, prática e econômica.

2.6 – Modelagem do Sistema

Com o objetivo de criar modelos de qualidade para ajudá-los a construir e manter sistemas cada vez mais eficazes foi utilizado a *Linguagem Unificada de Modelagem*, originalmente em inglês "Unified Modeling Language" (UML) conforme foi definido em Apostilando (2007).

2.6.1 – UML

Segundo Schmuller (2004), quando a UML foi lançada, muitos desenvolvedores ficaram entusiasmados, já que essa padronização proposta pela UML era exatamente o que eles sempre esperaram.

Ainda segundo Schmuller (2004), a UML pode ser caracterizada como muito mais que a padronização de uma notação. É também o desenvolvimento de novos conceitos, que anteriormente, não eram usados normalmente. Por isso e por muitas outras razões, que o bom entendimento da UML não é apenas aprender a simbologia e o seu significado, mas também significa aprender a modelar sistemas de forma padronizada e organizada.

A UML foi desenvolvida por Grady Booch, James Rumbaugh, e Ivar Jacobson. Eles possuem um extenso conhecimento na área de modelagem já que as três mais conceituadas metodologias de modelagem foram eles que desenvolveram, sendo a UML uma junção do que

havia de melhor nestas três metodologias adicionado a novos conceitos e visões da linguagem de acordo com Apostilando (2007).

Para Stadzisz (2002), a UML aborda o caráter estático e dinâmico do sistema a ser analisado, levando em consideração, já no período de modelagem, todas as futuras características do sistema em relação à linguagem que será utilizada, a utilização do banco de dados, bem como as diversas especificações do sistema a ser desenvolvido de acordo com as métricas finais do sistema.

2.6.2 - Fases do Desenvolvimento de um Sistema em UML

Neste trabalho foram utilizadas cinco fases no desenvolvimento do sistema: análise de requisitos, análise, design do projeto, programação e testes. Estas cinco fases não precisam ser executadas na ordem descrita, mas sim à medida que os problemas detectados numa certa fase, modifiquem e melhorem as fases desenvolvidas anteriormente de forma que o resultado global gere um produto de alta qualidade e performance. Nas próximas subseções será caracterizada cada fase do desenvolvimento de um sistema em UML.

2.6.2.1 - Análise de Requisitos

Esta fase captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas "casos de uso". Através do desenvolvimento de cada caso de uso, as entidades externas ao sistema (em UML chamados de "atores externos") que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem, conforme definido em Apostilando (2007). Os atores externos e os casos de uso são modelados com relacionamentos que possuem comunicação associativa entre eles ou então, desmembrados de acordo com suas hierarquias. Cada caso de uso modelado é descrito através de um texto, e este especifica os requerimentos do ator externo que utilizará este caso de uso, segundo Stadzisz (2002).

O diagrama de casos de uso mostrará o que os atores externos, ou seja, tudo que os futuros usuários do sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade, sem se importar como esta será implementada, segundo Vilalta (2001).

2.6.2.2 - Análise

A fase de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de Classe,

afirmou Vilalta (2001). Os relacionamentos entre classes também são mostrados neste diagrama para desenvolver os casos de uso modelados anteriormente, estas colaborações são criadas através de modelos dinâmicos em UML. Na análise, só serão modeladas classes que pertençam ao domínio principal do problema do software, ou seja, classes técnicas que gerenciem banco de dados, interface, comunicação, concorrência e outros não estarão presentes neste diagrama, afirmou Schmuller (2004).

2.6.2.3 - Design do projeto

Na fase de design, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros, definiu Stadzisz (2002). As classes do domínio do problema modeladas na fase de análise são mescladas nessa nova infra-estrutura técnica, tornando possível alterar tanto o domínio do problema quanto a infra-estrutura. O design resulta no detalhamento das especificações para a fase de programação do sistema, de acordo com Vilalta (2001).

2.6.2.4 - Programação

Na fase de programação, as classes provenientes do design são convertidas para o código da linguagem escolhida. Dependendo da capacidade da linguagem usada, essa conversão pode ser uma tarefa fácil ou muito complicada. Nas fases anteriores, os modelos foram criados a partir do entendimento e da estrutura do sistema. Já no momento da geração do código, os modelos criados são convertidos em código, de acordo com Schmuller (2004).

2.6.2.5 - Testes

Um sistema normalmente é executado através de testes de unidade, integração, e aceitação. Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador. Os testes de integração são aplicados já usando as classes e componentes integrados, para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos. Os testes de aceitação observam o sistema como uma “caixa preta" e verificam se o sistema está funcionando como o especificado nos primeiros diagramas de "casos de uso", segundo Schmuller (2004).

3 – Estudo de caso: Sistema de gerência dos dados coletados em experimentos micrometeorológicos - SGDM

Este capítulo tem como objetivo caracterizar os experimentos micrometeorológicos realizados pelo GRUMA, os dados que são coletados e como eles serão armazenados no SGDM, e também, demonstrar o sistema implementado, começando por suas funcionalidades e em seguida explicando como procedeu a modelagem do sistema.

3.1 – O GRUMA

O Grupo de Modelagem Atmosférica da Universidade Federal de Santa Maria atua tanto nas áreas de modelagem teórica de fenômenos turbulentos, como na sua observação experimental. Para Acevedo (2007), mais especificamente, do ponto de vista teórico, este grupo tem desenvolvido parametrizações para o transporte turbulento em diferentes modelos de dispersão atmosférica, levando em conta efeitos de forçantes térmicos e mecânicos. Da mesma maneira, o grupo tem também se dedicado ao estudo de processos de decaimento da turbulência bem desenvolvida na ausência dos forçantes que a originam.

Estas linhas de pesquisa pertencem à área de modelagem atmosférica, que, neste caso, faz uso de dados micrometeorológicos para gerar modelos que explicam como os fenômenos citados anteriormente estão acontecendo na atmosfera. Mais informações sobre estas pesquisas e como elas são desenvolvidas, podem ser encontradas no site do GRUMA, localizado no endereço: *http://www.gruma.ufsm.br* .

3.2 – Os experimentos micrometeorológicos e os dados coletados

Concomitantemente, com a modelagem teórica de fenômenos turbulentos, o grupo realiza campanhas observacionais, chamados de experimentos micrometeorológicos, que têm o objetivo de entender as transferências de energia e espécies entre a superfície e a atmosfera em diferentes ecossistemas. Estes experimentos envolvem regiões de terrenos complexos, florestas e diferentes coberturas superficiais afirmou Acevedo (2007).

Estes experimentos coletam, através de sensores instalados em torres, grandes volumes de dados, fato que levantou a necessidade de um armazenamento seguro, persistente e eficiente. E também, que sejam disponibilizados de forma simples, rápida e intuitiva como forma de otimizar a obtenção de resultados nos estudos mencionados acima.

3.2.1 – A coleta dos dados

A coleta dos dados em experimentos micrometeorológicos, é realizada de duas formas:

- em curtos períodos de coleta, que podem durar dias, semanas ou até alguns meses, onde geralmente são coletados dados com freqüência de 1 e 10 hz;
- em longos períodos de coleta, que podem durar anos, onde geralmente são utilizado dados com freqüências menores que 1 e 10 hz.

Durante cada experimento, os dados são coletados através de pequenas bases instaladas nos locais da coleta. Estas bases, geralmente, são equipadas com os seguintes aparelhos:

- uma torre, onde os sensores estão instalados;
- os sensores que fazem a coleta (figuras 3 e 4);

Figura 3: Desenho esquemático de uma torre de coleta

Figura 4: Fotografia de uma torre Micrometeorológica de 15m

- um datalogger que cataloga os dados dos sensores (figura 5);

Figura 5: Datalogger - Campbell 21XL

- um computador que recebe os dados do datalogger.

3.2.2 – A padronização dos dados

Os dados, normalmente, seguem o mesmo padrão. Os quatro primeiros campos, que são escritos nos arquivos de dados, são necessariamente nesta ordem:

1. código, ex: 21;
2. ano, ex: 2000;
3. dia juliano, ex: entre 1 e 365;
4. hora e minuto, ex: entre 0 e 2359.

Os demais campos possuem os valores dos sensores e estão na ordem em que o datalogger foi programado para escrever no arquivo de dados, como mostra a figura 6:

```
21,2000,211,1744,31.2,9.8051,11.431,88.496,90.8,3.2,3.95,3.95,58.888,55.231,46.571,-99999,2051
21,2000,211,1744,32.2,9.7374,9.4579,88.496,88.564,3.95,4.6999,3.95,27.664,31.801,36.179,-99999,2163.4
21,2000,211,1744,33.2,9.7374,11.364,88.159,90.937,3.95,3.95,4.6999,20.303,26.649,34.351,-99999,2266.5
```

Figura 6: Exemplo dos dados coletados em experimentos micrometeorológicos

No exemplo apresentado na figura 6, tem-se o 21 que representa o código do experimento, o 2000 que representa o ano, o 211 que representa o dia juliano e o 1744 que representa a hora e o minuto em que o dado foi coletado. Os demais campos representam o tipo de dado que foi coletado por cada sensor. Nos campos que aparecem o número -9999, significa que naquele instante o dado não foi coletado, ou seja, o número -9999 representa o valor nulo.

3.3 – SGDM

Nesta seção será demonstrado o sistema implementado descrevendo detalhadamente suas funcionalidades e as permissões de acesso que cada grupo de usuários possui. São demonstradas algumas telas de exemplo, como as de criação da estrutura, as de adição de dados em tabelas de experimentos já cadastrados e as telas de consulta que são padrões em todo o sistema.

3.3.1 – Os grupos de usuários

O sistema foi implementado com dois grupos de usuários que tem suas permissões diferenciadas e suas opções disponibilizadas após autenticação no servidor. São eles:

- os administradores: que tem permissão de criar novas tabelas para armazenar dados de um experimento ainda não cadastrado no banco de dados, bem como adicionar novos dados em um experimento já cadastrado;
- os pesquisadores: que possuem permissão apenas de consulta aos dados de um experimento já cadastrado.

A figura 7 apresenta a tela de “autenticação no servidor” que é mostrada quando o usuário acessa o banco de dados.

Figura 7: Autenticação no servidor

3.3.2 – O portal dos usuários do grupo administrador

Caso o usuário autenticado no sistema pertença ao grupo dos administradores, será exibida uma tela com as seguintes opções:

- Consultar experimento cadastrado no banco de dados;
- Cadastrar novo experimento no banco de dados;
- Adicionar dados em experimento cadastrado no banco de dados.

3.3.2.1 – Cadastro de um novo experimento no banco de dados

A figura 8 demonstra o cadastro, passo a passo, de um novo experimento no banco de dados. Esta seqüência de opções permitirá estruturar e criar uma tabela que armazenará os dados de um experimento de maneira simples e organizada.

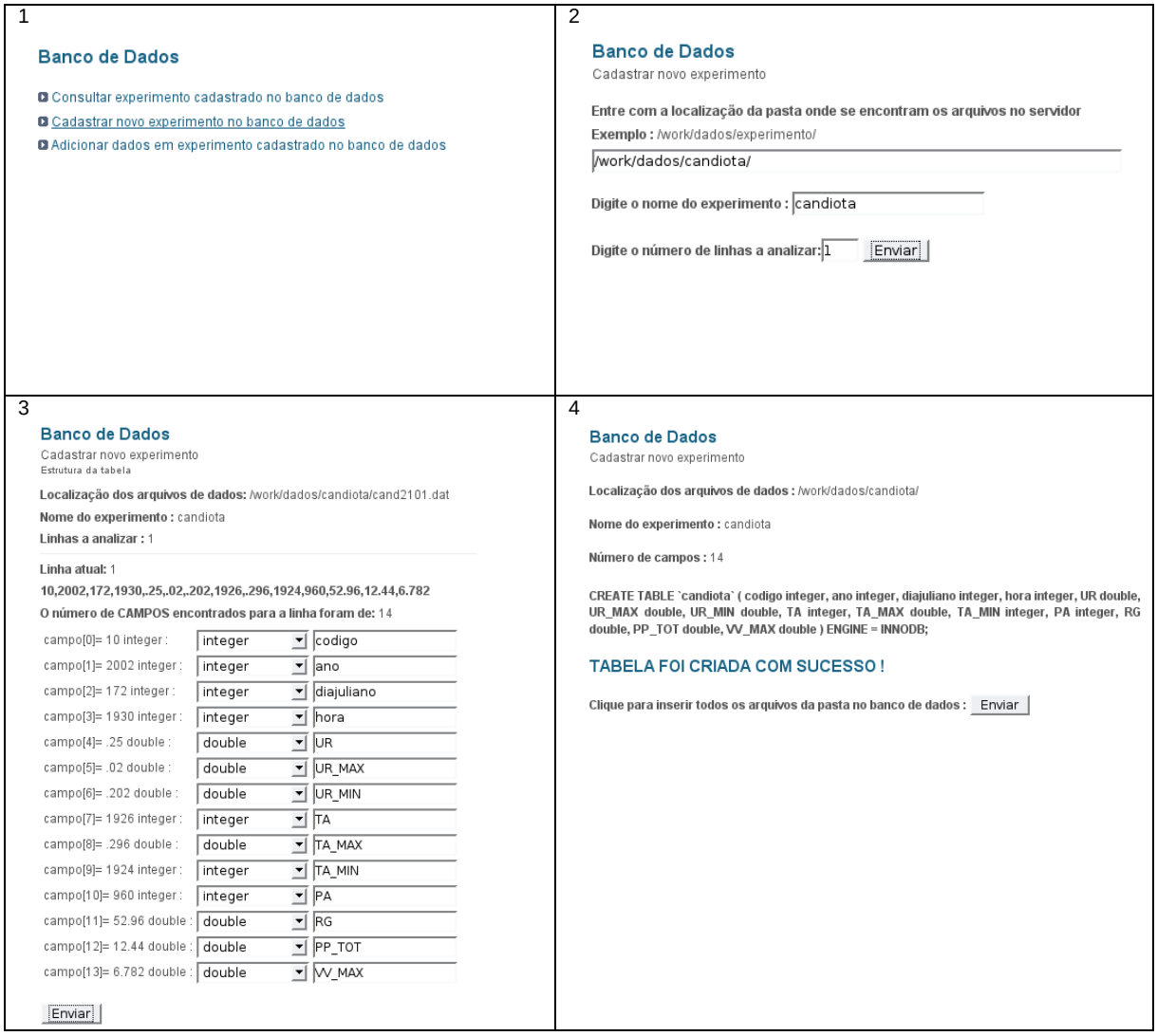


Figura 8: Demonstração da seqüência de cadastro de um novo experimento

No passo 1 o usuário, com permissão de administrador, seleciona a opção “Cadastrar novo experimento no banco de dados” e logo em seguida ele é encaminhado ao passo 2. Neste passo o usuário deverá informar ao sistema a pasta onde se encontram os dados e a quantidade de linhas dos arquivos dessa pasta que ele deseja analisar a consistência e só então, ele é encaminhado ao próximo passo.

No passo 3, se a pasta dos dados informada anteriormente estiver correta, o sistema retornará os tipos das variáveis que foram reconhecidas pelo sistema. Ele também dará a sugestão do nome dos campos para Campo_1, ..., Campo_n e permitirá que o usuário altere, se assim desejar.

A partir dos passos anteriores, o sistema agora está capacitado para criar a tabela que armazenará os dados de um novo experimento, como demonstra a figura 8 no passo 4.

3.3.2.2 – Adição dos dados na tabela do experimento

Para adicionar os dados na tabela do experimento, é absolutamente necessário que os dados estejam em alguma pasta temporária do servidor. Sendo assim, a adição pode ser feita de duas maneiras:

- a primeira, vem logo após a criação da tabela (conforme o a sub-seção anterior), onde a localização da pasta já foi informada ao sistema desde o *passo 2* da figura 8**;**
- a segunda é a adição de dados em uma tabela já existente no banco de dados. Para isto, é necessário selecionar a terceira opção, na lista de opções do grupo de usuários administradores, que é a opção denominada: "*Adicionar dados em experimento cadastrado no banco de dados*". Desta forma, será necessário informar ao sistema a localização da pasta onde os arquivos dos dados se encontram no sistema, bem como selecionar o experimento na qual pertencem os dados a serem inseridos, conforme demonstra a figura 9.

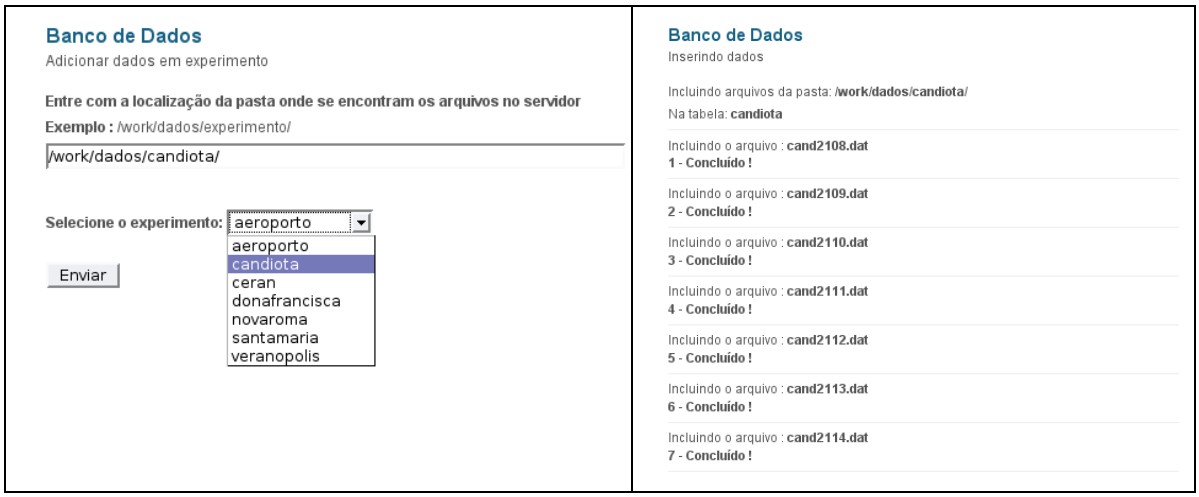


Figura 9: Adicionando dados em um experimento já cadastrado

A figura 9 demonstra como um grupo de dados de um experimento já existente no banco de dados pode ser adicionado com apenas 2 passos.

3.3.3 – O portal dos usuários do grupo pesquisador

A opção disponível para os usuários pertencentes ao grupo pesquisador é análoga a primeira opção da lista de opções dos usuários pertencentes ao grupo administrador, uma vez

que os administradores possuem as permissões de administração e também as permissões do grupo pesquisador.

A figura 10 demonstra os passos que devem ser seguidos pelo usuário, neste caso o pesquisador, pode efetuar uma consulta em um grupo de dados contendo apenas o período e as variáveis desejadas.

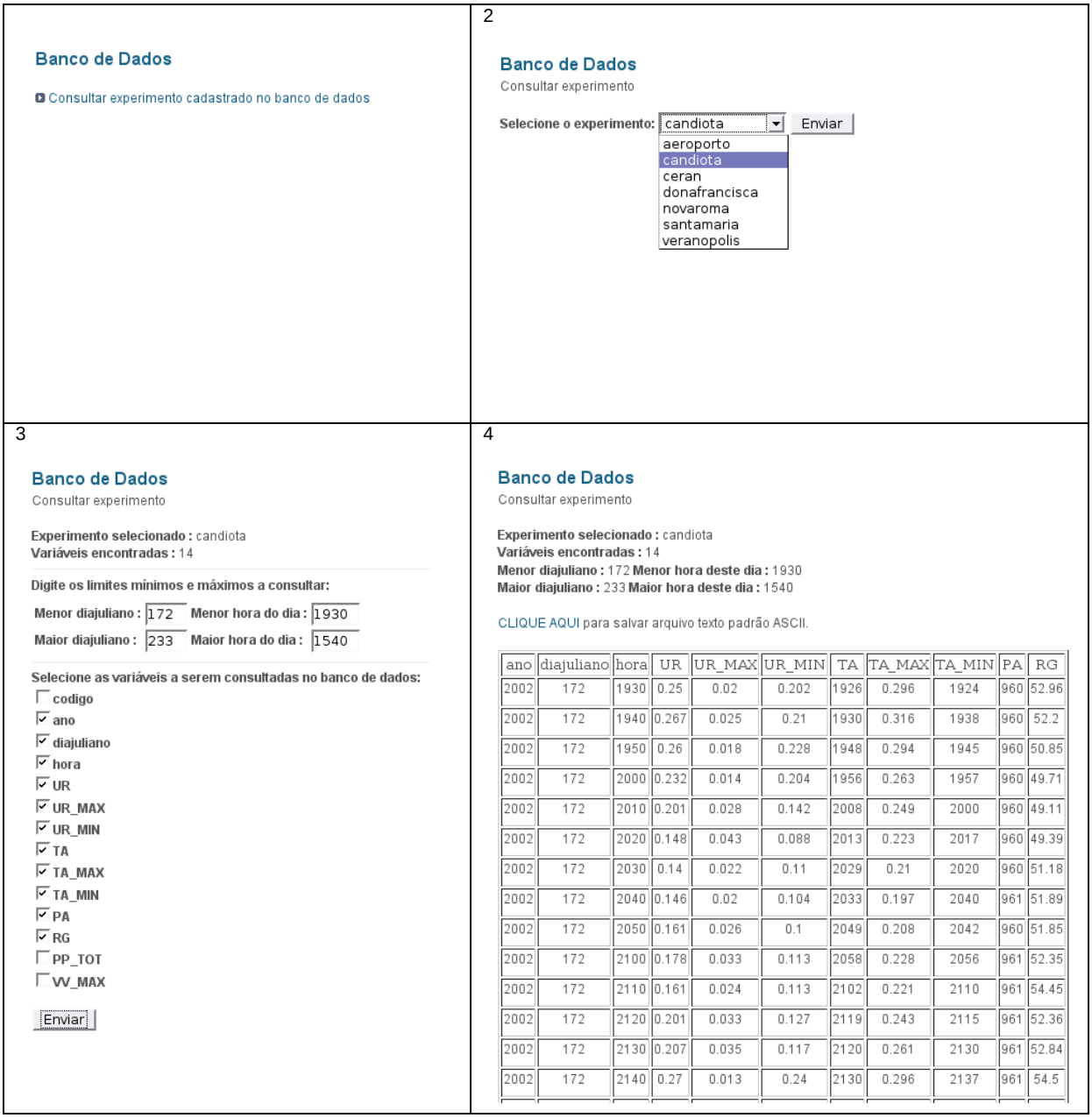


Figura 10: Consulta de dados passo a passo

O pesquisador após selecionar o experimento no passo 2, deverá informar ao sistema o limite mínimo e máximo dos dias juliano e da hora, e também, marcar quais as variáveis que ele deseja obter na consulta. Vale ressaltar que, quanto maior o período de consulta, maior será o tempo de consulta dos dados. Esta diferença será perceptível principalmente quando os dados forem de freqüência de 10 hz. Entretanto, o sistema é capaz de recuperar grandes quantidades de dados, e para todos os casos, a saída será sempre a mesma, divergindo apenas no tamanho do arquivo gerado.

O *passo 4* da figura 10 mostrou um *link* denominado *"CLIQUE AQUI (...)"* que disponibiliza para *download* um arquivo texto codificado no formato ASCII Padrão contendo os dados que foram consultados a partir dos filtros informados no *passo 3* da mesma figura.

Este arquivo encontra-se estruturado internamente conforme a figura 11, ou seja, pronto para ser manipulado por diversas aplicações como *Matlab*, *Fortran* e *Excel*, por exemplo.

```
ano,diajuliano,hora,UR,UR_MAX,UR_MIN,TA,TA_MAX,TA_MIN,PA,RG,
2002,172,1930,0.25,0.02,0.202,1926,0.296,1924,960,52.96,
2002,172,1940,0.267,0.025,0.21,1930,0.316,1938,960,52.2,
2002,172,1950,0.26,0.018,0.228,1948,0.294,1945,960,50.85,
2002,172,2000,0.232,0.014,0.204,1956,0.263,1957,960,49.71,
2002,172,2010,0.201,0.028,0.142,2008,0.249,2000,960,49.11,
2002,172,2020,0.148,0.043,0.088,2013,0.223,2017,960,49.39,
2002,172,2030,0.14,0.022,0.11,2029,0.21,2020,960,51.18,
2002,172,2040,0.146,0.02,0.104,2033,0.197,2040,961,51.89,
2002,172,2050,0.161,0.026,0.1,2049,0.208,2042,960,51.85,
2002,172,2100,0.178,0.033,0.113,2058,0.228,2056,961,52.35,
2002,172,2110,0.161,0.024,0.113,2102,0.221,2110,961,54.45,
2002,172,2120,0.201,0.033,0.127,2119,0.243,2115,961,52.36,
2002,172,2130,0.207,0.035,0.117,2120,0.261,2130,961,52.84,
2002,172,2140,0.27,0.013,0.24,2130,0.296,2137,961,54.5,
2002,172,2150,0.265,0.031,0.185,2149,0.316,2146,961,53.14,
2002,172,2200,0.154,0.034,0.081,2157,0.218,2151,962,53.06,
2002,172,2210,0.165,0.023,0.125,2208,0.24,2202,962,54.76,
2002,172,2220,0.085,0.041,0.028,2217,0.158,2211,962,54.94,
2002,172,2230,0.015,0.014,0,2222,0.056,2230,962,55.06,
2002,172,2240,0.153,0.086,0.02,2231,0.316,2240,962,55.76,
2002,172,2250,0.373,0.043,0.281,2240,0.453,2245,962,56.23,
2002,172,2300,0.373,0.042,0.296,2258,0.463,2253,963,57.14,
2002,172,2310,0.327,0.039,0.25,2301,0.415,2303,963,57.47,
2002,172,2320,0.261,0.035,0.205,2314,0.358,2310,963,58.07,
2002,172,2330,0.368,0.04,0.275,2320,0.453,2327,963,58.56,
2002,172,2340,0.374,0.026,0.319,2337,0.426,2335,963,59.72,
2002,172,2350,0.301,0.042,0.226,2345,0.397,2340,963,60.26,
2002,173,0,0.276,0.021,0.236,2353,0.345,2354,963,60.6,
2002,173,10,0.227,0.02,0.171,8,0.276,7,963,60.85,
2002,173,20,0.26,0.03,0.199,10,0.354,14,963,61.11,
```

Figura 11: Arquivo contendo os dados que foram consultados

3.4 – Etapas de desenvolvimento do sistema

Esta fase do trabalho foi divida nas seguintes etapas:

1. Modelagem do sistema;
2. Coleta dos dados;
3. Criação do banco de dados;
4. Programação do *script* de criação automática da estrutura da tabela de armazenamento dos dados;
5. Programação do *script* de adição automática dos dados;
6. Programação do sistema de acesso.

3.4.1 – Modelagem do sistema

Como citado na revisão de literatura, foram utilizado as técnicas de UML para modelar o sistema. A figura 12 mostra a modelagem dos casos de uso do sistema através do diagrama de casos de uso.

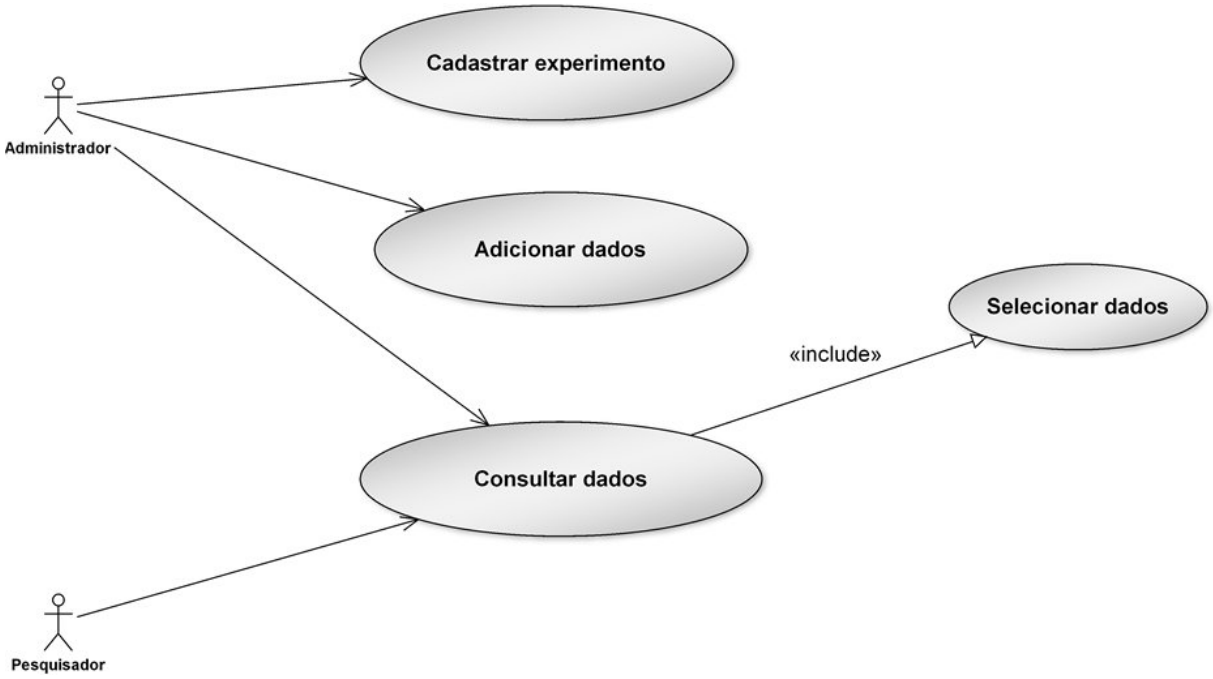


Figura 12: Diagrama de casos de uso completo do SGDM

A figura 12 demonstra o diagrama de casos de uso do SGDM. Nele existem 2 atores, o Administrador, que tem acesso a todas as opções do sistema, e o Pesquisador que tem acesso apenas à consulta de dados.

A figura 13, 14 e 15 mostra o diagrama de seqüência para os três casos de uso do SGDM, cadastrar experimento, adicionar dados e consultar dados, respectivamente.

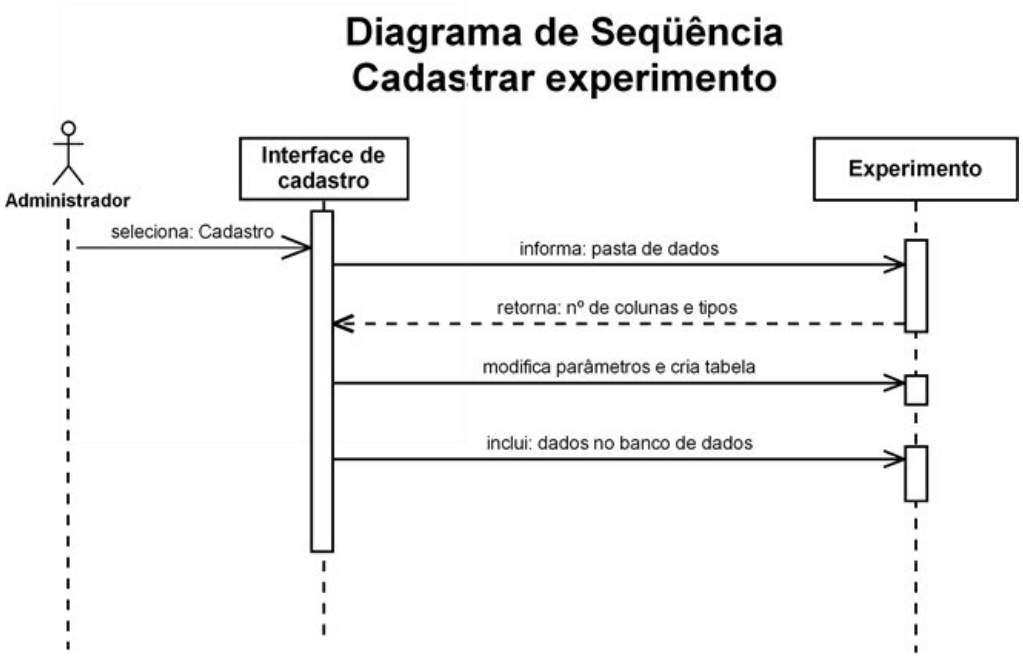


Figura 13: Diagrama de seqüência para o caso de uso cadastrar experimento

A figura 13 demonstrou os casos de uso da parte de cadastro dos experimentos. Inicialmente o usuário informa à interface que deseja cadastrar um novo experimento, e então, as mensagens são trocadas no diagrama: "*informa: pasta de dados*", "*retorna: nº de colunas e tipos*", "*modifica parâmetros e cria tabela"*, e "*inclui: dados no banco de dados*".

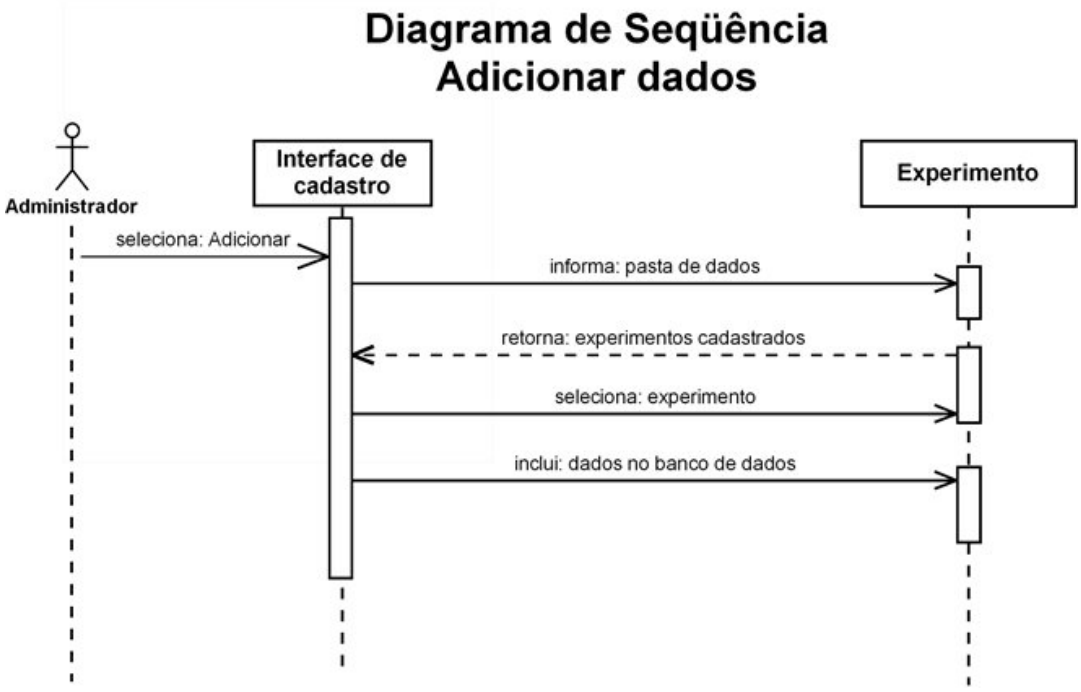


Figura 14: Diagrama de seqüência para o caso de uso adicionar dados

A figura 14 demonstrou os casos de uso da parte de adicionar dados nos experimentos. Inicialmente o usuário informa à interface que deseja adicionar dados a um experimento cadastrado, e então, são trocadas as mensagens no diagrama: "*informa: pasta de dados*", "*retorna: experimentos cadastrados*", "*seleciona: experimento"*, e "*inclui: dados no banco de dados*".

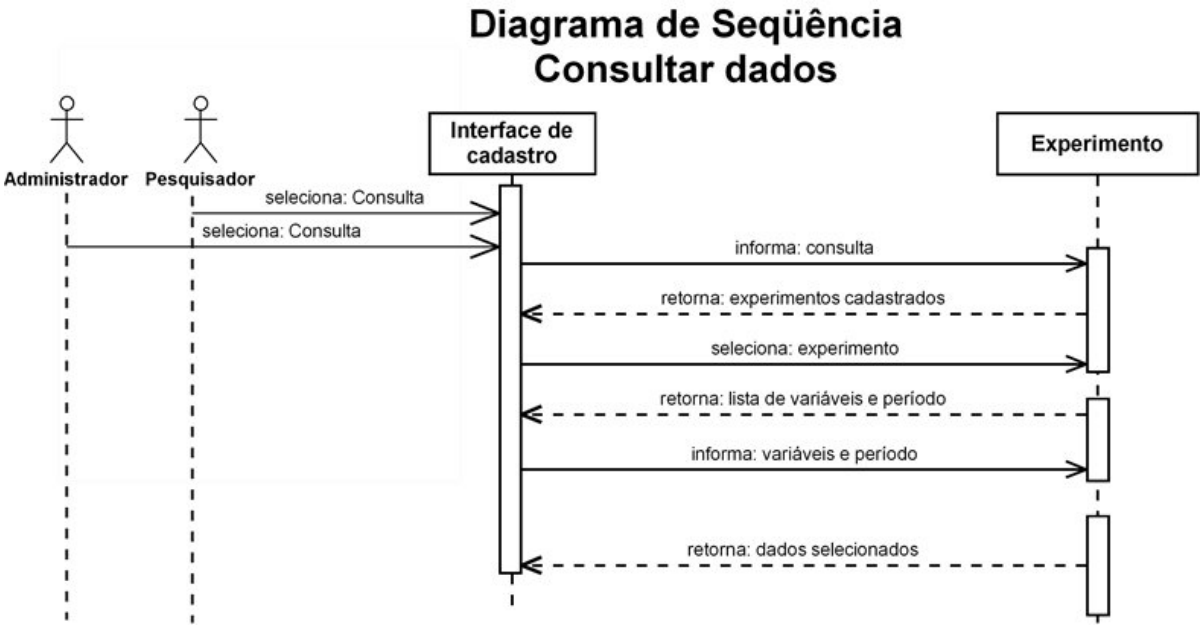


Figura 15: Diagrama de seqüência para o caso de uso consultar dados

A figura 15 demonstrou os casos de uso da parte de consultar dados nos experimentos. Inicialmente o usuário informa à interface que deseja consultar dados a um experimento cadastrado, e então, são trocadas as mensagens no diagrama: "*informa: consulta*", "*retorna: experimentos cadastrados*", "*seleciona: experimento"*, e "*retorna: lista de variáveis e períodos*", "*informa: variáveis e período*", "*retorna: dados selecionados*".

3.4.2 – Os dados utilizados nos testes do sistema

Os dados selecionados para testar o sistema foram dados coletados em um experimento realizado na cidade de Candiota no ano de 2002. O período usado para incluir no banco de dados do sistema piloto totalizou 1457 arquivos de dados, cada um com cerca de 4500 linhas de dados com 14 colunas (variáveis) para cada linha de dados, totalizando cerca de 6.556.500 linhas.

A figura 16 mostra como o datalogger utilizado neste experimento foi programado.

```
Final Storage Label File for:  CGTEEAR.CSI
Date:  8/22/2002
Time:  13:55:41

10 Output_Table  10.00 Min
1 10 L
2 Year_RTM  L
3 Day_RTM  L
4 Hour_Minute_RTM  L
5 DV_AVG  L
6 DV_STD  L
7 VV_AVG  L
8 VV_MAX  L

60 Output_Table  60.00 Min
1 60 L
2 Year_RTM  L
3 Day_RTM  L
4 Hour_Minute_RTM  L
5 UR  L
6 UR_MAX  L
7 UR_MIN  L
8 TA  L
9 TA_MAX  L
10 TA_MIN  L
11 PA  H
12 RG  L
13 PP_TOT  L


Estimated Total Final Storage Locations used per day 1488.0
```

Figura 16: Programação do datalogger utilizado

Isso significa dizer que os dados ficarão dispostos no arquivo de saída na seguinte ordem: Código, Year, Day, Hour_Minute, UR, UR_MAX , UR_MIN, TA, TA_MAX, TA_MIN, PA, RG, PP_TOT.

Que representa: o código do experimento, ano, dia juliano, hora e minuto, umidade relativa, umidade relativa máxima, umidade relativa mínima, temperatura atmosférica, temperatura atmosférica máxima, temperatura atmosférica mínima, pressão atmosférica, radiação e precipitação total. A figura 17 exemplifica o arquivo de dados utilizado para testar o sistema.

```
10,2002,172,1930,.25,.02,.202,1926,.296,1924,960,52.96,12.44,6.782
10,2002,172,1940,.267,.025,.21,1930,.316,1938,960,52.2,12.35,6.782
10,2002,172,1950,.26,.018,.228,1948,.294,1945,960,50.85,12.27,6.782
10,2002,172,2000,.232,.014,.204,1956,.263,1957,960,49.71,12.35,6.406
10,2002,172,2010,.201,.028,.142,2008,.249,2000,960,49.11,12.3,6.783
10,2002,172,2020,.148,.043,.088,2013,.223,2017,960,49.39,12.28,6.406
10,2002,172,2030,.14,.022,.11,2029,.21,2020,960,51.18,12.32,6.783
10,2002,172,2040,.146,.02,.104,2033,.197,2040,961,51.89,12.32,6.407
10,2002,172,2050,.161,.026,.1,2049,.208,2042,960,51.85,12.3,6.407
10,2002,172,2100,.178,.033,.113,2058,.228,2056,961,52.35,12.34,6.784
10,2002,172,2110,.161,.024,.113,2102,.221,2110,961,54.45,12.4,6.407
10,2002,172,2120,.201,.033,.127,2119,.243,2115,961,52.36,12.41,6.784
10,2002,172,2130,.207,.035,.117,2120,.261,2130,961,52.84,12.39,6.407
10,2002,172,2140,.27,.013,.24,2130,.296,2137,961,54.5,12.33,6.407
10,2002,172,2150,.265,.031,.185,2149,.316,2146,961,53.14,12.26,6.407
10,2002,172,2200,.154,.034,.081,2157,.218,2151,962,53.06,12.46,6.784
10,2002,172,2210,.165,.023,.125,2208,.24,2202,962,54.76,12.43,6.407
10,2002,172,2220,.085,.041,.028,2217,.158,2211,962,54.94,12.46,6.407
10,2002,172,2230,.015,.014,0,2222,.056,2230,962,55.06,12.33,6.784
10,2002,172,2240,.153,.086,.02,2231,.316,2240,962,55.76,12.29,6.784
10,2002,172,2250,.373,.043,.281,2240,.453,2245,962,56.23,12.3,6.784
10,2002,172,2300,.373,.042,.296,2258,.463,2253,963,57.14,12.36,6.407
10,2002,172,2310,.327,.039,.25,2301,.415,2303,963,57.47,12.32,6.784
10,2002,172,2320,.261,.035,.205,2314,.358,2310,963,58.07,12.29,6.784
10,2002,172,2330,.368,.04,.275,2320,.453,2327,963,58.56,12.41,6.784
10,2002,172,2340,.374,.026,.319,2337,.426,2335,963,59.72,12.35,6.784
10,2002,172,2350,.301,.042,.226,2345,.397,2340,963,60.26,12.36,6.407
10,2002,173,0,.276,.021,.236,2353,.345,2354,963,60.6,12.33,6.784
10,2002,173,10,.227,.02,.171,8,.276,7,963,60.85,12.33,6.407
10,2002,173,20,.26,.03,.199,10,.354,14,963,61.11,12.39,6.784
10,2002,173,30,.254,.02,.211,28,.308,22,963,61.54,12.38,6.407
10,2002,173,40,.176,.028,.113,39,.235,30,963,61.57,12.31,6.407
10,2002,173,50,.169,.022,.108,42,.209,46,963,61.86,12.24,6.784
10,2002,173,100,.146,.016,.111,53,.182,57,963,61.88,12.34,6.784
10,2002,173,110,.099,.016,.074,103,.143,101,963,61.83,12.34,6.784
10,2002,173,120,.123,.02,.073,111,.151,117,963,62.03,12.33,6.785
10,2002,173,130,.106,.018,.078,123,.145,130,963,61.64,12.38,6.408
10,2002,173,140,.135,.011,.112,140,.161,130,963,61.46,12.36,6.408
```

Figura 17: Arquivo de dados utilizados para testar o sistema

3.4.3 – Criação do banco de dados

A criação do banco de dados foi feita no SGBD MySQL 5.0 *Server* instalado no servidor GRUMA, *on-line* no endereço http://www.gruma.ufsm.br.

Foram usadas as técnicas clássicas para criação de um novo banco de dados no MySQL, através de comandos SQL. Exemplo mostrado pela figura 18.

```
CREATE DATABASE `grumadb` ;
```

Figura 18: Criação do banco de dados

3.4.4 – Programação da interface de criação da estrutura das tabelas de armazenamento

O *script* de criação das tabelas foi escrito na linguagem PHP, e seu funcionamento ocorre de acordo com os seguintes passos:

1. O usuário do grupo administrador informa o local do servidor onde se encontram os dados a serem inclusos;
2. O *script* de reconhecimento dos tipos analisa uma ou mais linhas, determinado pelo administrador, com a finalidade de certificar que a tabela seja estruturada de forma correta.

Para exemplificar, a figura 19 mostra a o *script* chamado de “*reconhecer*_tipos”, que reconhece os tipos das variáveis que serão estruturadas na tabela a ser criada.

```php
<?php require("bd_topo800.php"); ?>
<?php
  function tipo($s) {
    eval('$temp=' . $s . ";");
    return gettype($temp);
  }
$pasta=$_POST[pasta];
$linhas=$_POST[linhas];
$experimento=$_POST[experimento];
$current_dir = $pasta;
$dir = opendir($current_dir);
        while ($file = readdir($dir))
        {
          if ($file!="." and $file!=".."){
            $arquivo=$pasta.$file;
          }
        }
        closedir($dir);
echo "<p><b>Localização : </b>".$arquivo;
echo "<p><b>Nome do experimento : </b>".$experimento;
echo "<p><b>Linhas a analizar : </b>".$linhas;
if($fdb= fopen($arquivo,"r")) {
  $n=1;
  while ($n<=$linhas) {
        echo "<hr><p><b>Linha atual: </b>".$n;
        $buffer= fgets($fdb, 1024);
        echo "<p><b>".$buffer."</b><br>";
        $campo= explode(',', $buffer);
        $num_campos= count($campo);
        echo "<p><b> O número de CAMPOS encontrados para a linha foram de: </b>".$num_campos."<br>";
        $i= 0;
        $t=sizeof($campo);

        ?><table border=0 >
        <tr>
        <form enctype="multipart/form-data" action="criar_tabela.php" method="post"><?php
             ?><input type="hidden" name="t" value="<?php echo $t ?>"></input><?php
             while ( $i<$t ) {
                ?><td><?php echo("campo[$i]= " . $campo[$i] .  " " . tipo($campo[$i])  . " : "); ?></td><?php
                // Cria automaticamente para:
                $nome_campo= "campo ".$i;

                if($nome_campo=='campo_0'){$nome_campo='codigo';}
                if($nome_campo=='campo_1'){$nome_campo='ano';}
                if($nome_campo=='campo_2'){$nome_campo='diajuliano';}
                if($nome_campo=='campo_3'){$nome_campo='hora';}
                $typecampo= tipo($campo[$i]);
                if($typecampo=='string'){$typecampo='char(20)';}
                $typevalor= "typevalor_".$i;
                $nomevalor= "nomevalor_".$i;

                ?><td>
                    <select name="<?php echo $typevalor ?>">
                        <option value="<?php echo $typecampo ?>"><?php echo $typecampo ?></option>
                        <?php require("typeoptions.php"); ?>
                    </select>
                  </td>
                  <td>
                    <input maxLength="20" name="<?php echo $nomevalor ?>" type="text" id="<?php echo $nomevalor
?>" size="20" value="<?php echo $nome_campo ?>"></input>
                  </td>
        </tr>
                  <?php
                $i++;
              }
        ?><?php

        $campos_da_tabela_sem_virgula=substr($campos_da_tabela, 1);
        $tabela=$experimento;
        $n++;
        $campos_da_tabela='';
        ?></table><p><br>
        <input type="hidden" name="pasta" value="<?php echo $pasta ?>"></input>
        <input type="hidden" name="experimento" value="<?php echo $experimento ?>"></input>
        <input type="submit" value="Enviar" /><br><br>
        </form><?php

  } //fim do primeiro while
}  //fim do if

?>
<?php require("bd_bottom800.php"); ?>
```

Figura 19: *Script* que reconhece os tipos das variáveis

A figura 20 mostra a interface *html* gerada a partir da execução dos códigos *PHP* da figura 19.

Com esta interface será possível alterar a estrutura da tabela que será criada modificando, caso necessário, os tipos de variáveis, bem como o nome das variáveis, que inicialmente foi atribuído *"campo_<nº do campo>"* exemplo: *campo_1 ... campo_n* .

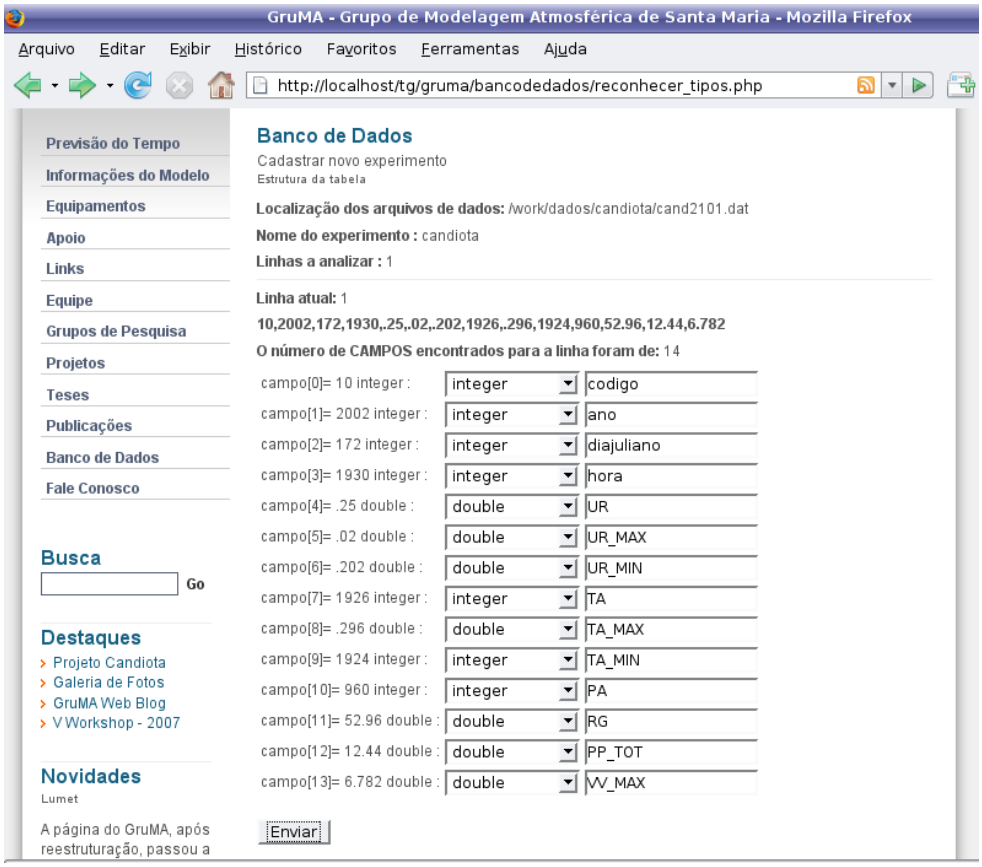


Figura 20: Interface *html* gerada a partir da execução do *script* reconhecer_tipos

Após apertar o botão "*enviar*", os dados da estrutura da tabela serão repassados a um segundo *script*, chamado de "*cria_tabela*", que terá a função de executar o comando SQL, de criação de tabela, gerado a partir dos parâmetros definidos no *script* anterior, como mostra a figura 21.

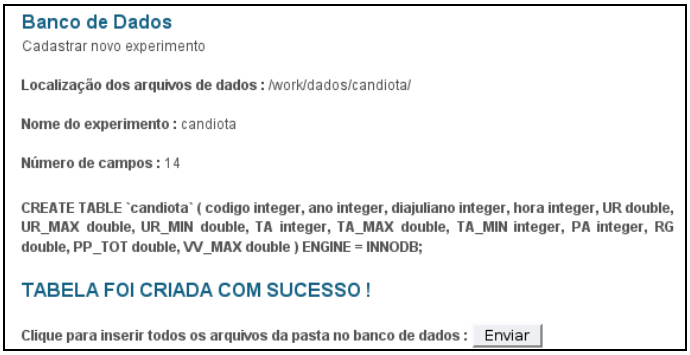


Figura 21: Comando SQL executado para criação da tabela

Para a execução deste comando, é necessário que o sistema continue seguindo os passos:

3. O *script* cria_tabela conecta no servidor de banco de dados;
4. O *script* cria_tabela acessa o banco de dados "*grumadb*" criado anteriormente;
5. E através de técnicas SQL e com os resultados obtidos no passo 2, ele cria a tabela que posteriormente receberá os dados do experimento que está sendo incluído no banco de dados. Para realização dos testes do sistema, nomeamos a tabela com o nome da cidade onde o experimento foi realizado, ou seja, *candiota.*

3.4.5 – Programação do *script* de inclusão dos dados

O *script* de inclusão, chamado de " *bd_import* ", também foi escrito na linguagem PHP. Sua função é adicionar os dados dos arquivos de dados contidos na pasta indicada pelo usuário. Seu funcionamento ocorre de acordo com os seguintes passos:

1. O *script* conecta no servidor de banco de dados
2. O *script* acessa o banco de dados "*grumadb*";
3. O *script* acessa a tabela "*candiota*" criada ao final da execução do *script* de criação das tabelas*;*
4. Inclui todas as linhas de todos os arquivos da pasta informada pelo usuário do grupo administrador, neste caso, a pasta:
   /work/dados/candiota/

A figura 22 demonstra a programação dos passos descritos acima.

```php
<?php require("bd_topo800.php"); ?>

<?php
require("conect_db2.php");
$pasta=$_POST[pasta];
$current_dir=$pasta;
echo "<p>Incluindo arquivos da pasta: <b>".$current_dir."<br></b>";
$tabela=$_POST[experimento];
echo "<p>Na tabela: <b>".$tabela."</b><br><hr>";
$dir = opendir($current_dir);
$c_num = 0;
while ($file = readdir($dir))
        {
          if ($file!="." and $file!=".."){

              $sql = 'LOAD DATA LOCAL INFILE '."'".$current_dir.$file."'".' INTO TABLE '.'`'.$tabela.'`'.' FIELDS
TERMINATED BY '."'".','."'".' ';

              echo "<p>Incluindo o arquivo : <b>$file</b>";
              mysql_query($sql);
              $c_num = $c_num + 1;
              echo "<br><b>$c_num</b> - <b> Concluído ! </b><hr>";

          }
        }
        closedir($dir);

?>

<?php require("bd_bottom800.php"); ?>
```

Figura 22: *Script* de inclusão dos dados em experimentos já cadastrados no banco de dados

3.4.6 – Programação do sistema de acesso

O objetivo do sistema de acesso é permitir uma interface entre os usuários do grupo pesquisador e as tabelas dos experimentos existentes no banco de dados "*grumadb*".

Podemos definir o sistema de acesso através dos seguintes passos:

1. usuário faz *login* no sistema;
2. o sistema verifica suas permissões de acesso;
3. o sistema retorna as tabelas dos experimentos cadastrados no banco de dados "*grumadb*" através do *script* chamado de "*lista_tabelas*"; A figura 23 demonstra os códigos que listam das tabelas dos experimentos.

```php
<?php

require("conect_db.php");

$tables = mysql_list_tables($banco);

?>
<form enctype="multipart/form-data" action="selecionar_variavel.php" method="post">
        <B>Selecione o experimento:</B>
        <select name=experimento>
            <?php
              $tables = mysql_list_tables($banco);
              while ($row = mysql_fetch_row($tables))
              {
                $experimento = $row[0];
                ?><option value="<?php echo $experimento;?>"><?php echo $experimento;?></option><?php
              }
            ?>
        </select>
        <input type="submit" value="Enviar" /><br><br>
</form>


<?php require("bd_bottom800.php"); ?>
```

Figura 23: Códigos que listam das tabelas de experimentos cadastrados no banco de dados

Após a execução destes códigos, a interface gerada permite ao usuário escolher qual experimento deseja acessar para consultar os dados. A figura 24 mostra a interface *html* gerada a partir da execução dos códigos *PHP* da figura 23.

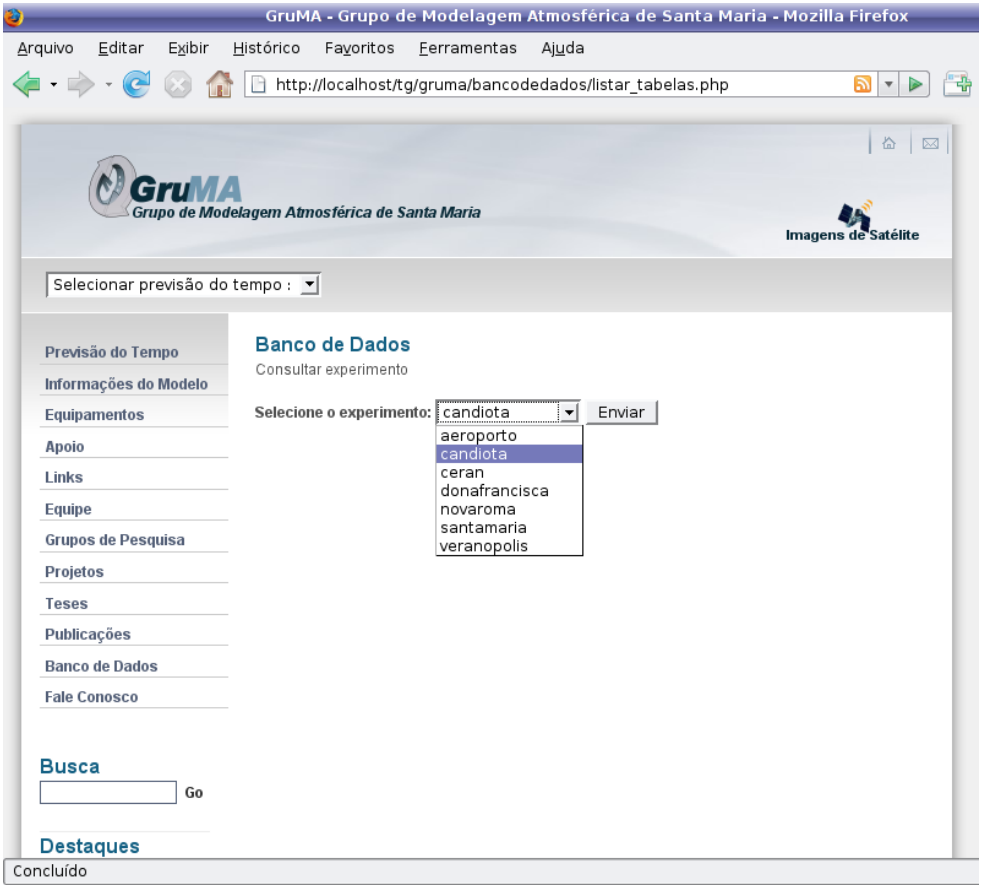


Figura 24: Interface *html* gerada a partir da execução do *script* lista_tabela

Após escolha do experimento, o sistema encaminha o usuário ao próximo passo:

4. o sistema lista as variáveis contidas na tabela do experimento escolhido, através do *script* chamado de "*selecionar_variavel*". A figura 25 demonstra os códigos PHP do *script* que produzem a interface que, através de recursos de formulários *html*, permite determinar o período e as variáveis a serem consultadas.

```php
<?php require("bd_topo800.php");
$table = $_POST[experimento];
echo "<b>Experimento selecionado : </b>".$table;
require("conect_db.php");
$fields = mysql_list_fields($banco,$table,$db);
$columns = mysql_num_fields($fields);
echo "<br><b>Variáveis encontradas : </b>".$columns;

if (!$db) {
    die('Não foi possível conectar: ' . mysql_error());
}

mysql_select_db('grumadb');

$dado_min = "SELECT min(diajuliano), min(hora) FROM ".$table;
?><form enctype="multipart/form-data" action="selecionar_dados.php" method="post">
<hr>
<p><b>Digite os limites mínimos e máximos a consultar:</b>
<table border="0">
<?php
$min=mysql_query($dado_min);
        while ($row_min=mysql_fetch_array($min)) {
            $diajuliano_min = $row_min[0];
            $hora_min =  $row_min[1];
            ?>
            <tr><td><center><b>Menor diajuliano : </b></td>
            <td><input maxLength="3" name="diajuliano_min" type="text" id="diajuliano_min" size="3"
value="<?php echo $diajuliano_min; ?>"></input></td>
            <td><center><b>Menor hora do dia : </b></td>
            <td><input maxLength="4" name="hora_min" type="text" id="hora_min" size="4" value="<?php echo
$hora_min; ?>"></p></br></input></td></tr>
            <?php
        }

$dado_max= "SELECT max(diajuliano), max(hora) FROM ".$table;
$max=mysql_query($dado_max);
        while ($row_max=mysql_fetch_array($max)) {
            $diajuliano_max = $row_max[0];
            $hora_max =  $row_max[1];
            ?>
            <tr><td><b>Maior diajuliano : </b></td>
            <td><input maxLength="3" name="diajuliano_max" type="text" id="diajuliano_max" size="3"
value="<?php echo $diajuliano_max; ?>"></input></td>
            <td><b>Maior hora do dia : </b></td>
            <td><input maxLength="4" name="hora_max" type="text" id="hora_max" size="4" value="<?php echo
$hora_max; ?>"></p></br></input></td></tr>
            <?php
        }
?></table><hr>

<?php

$query="select * from ".$table." LIMIT 0 , 1";
$result = mysql_query("$query");
if (!$result) {
    die('A consulta falhou: ' . mysql_error());
}
?>
        <input type="hidden" name="experimento" value="<?php echo $table;?>">
        <B>Selecione as variáveis a serem consultadas no banco de dados:</B><br><p>
            <?php
                $i = 0;
                while ($i < mysql_num_fields($result)) {
                    $meta = mysql_fetch_field($result, $i);
                    if (!$meta) { echo "Sem informação disponível<br />\n"; }
                    $campo="$meta->name";
                    ?><input name="<?php echo $campo;?>" type="checkbox" value="sim"><b><?php echo
$campo;?></b><br><?php
                    $i++;
                }
                mysql_free_result($result);
            ?>
        <br><input type="submit" value="Enviar" /><br><br>
</form>
<?php require("bd_bottom800.php"); ?>
```

Figura 25: Códigos do *script* que produz a interface de consulta dos dados

A interface gerada é demonstrada através da figura 26 que permitirá ao usuário limitar o período das variáveis a serem consultadas na tabela do experimento selecionado no passo anterior.

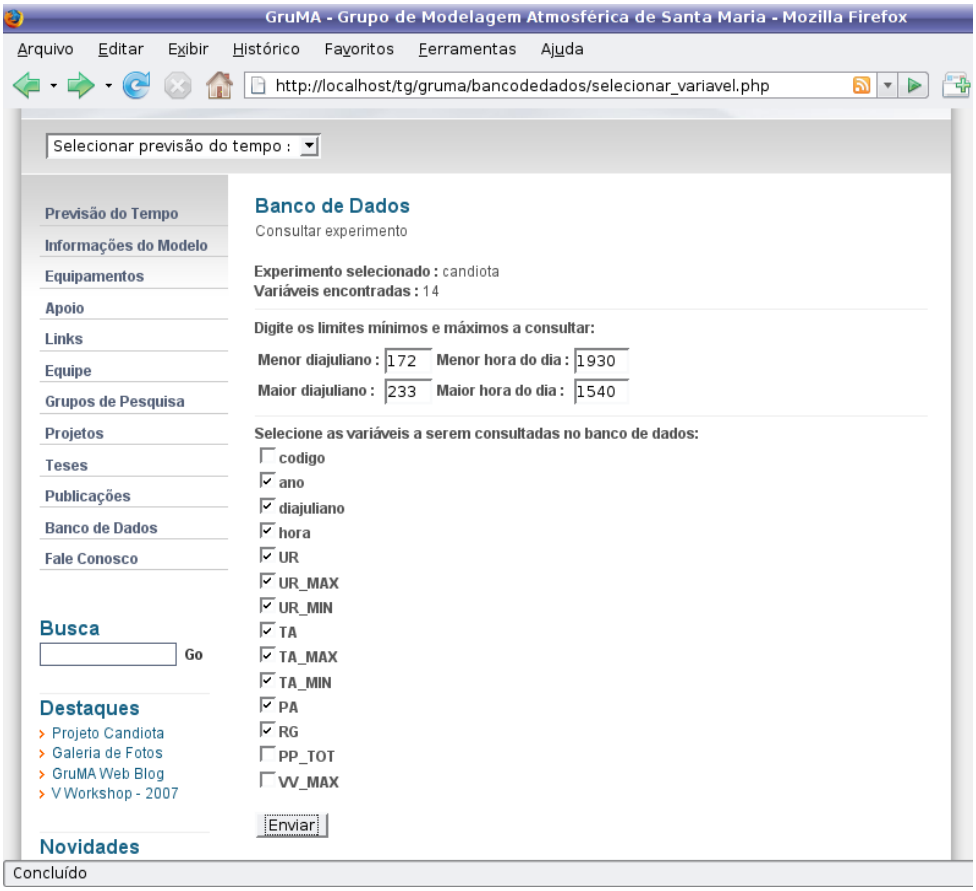


Figura 26: Interface mostrada ao usuário após a execução do script selecionar_variavel

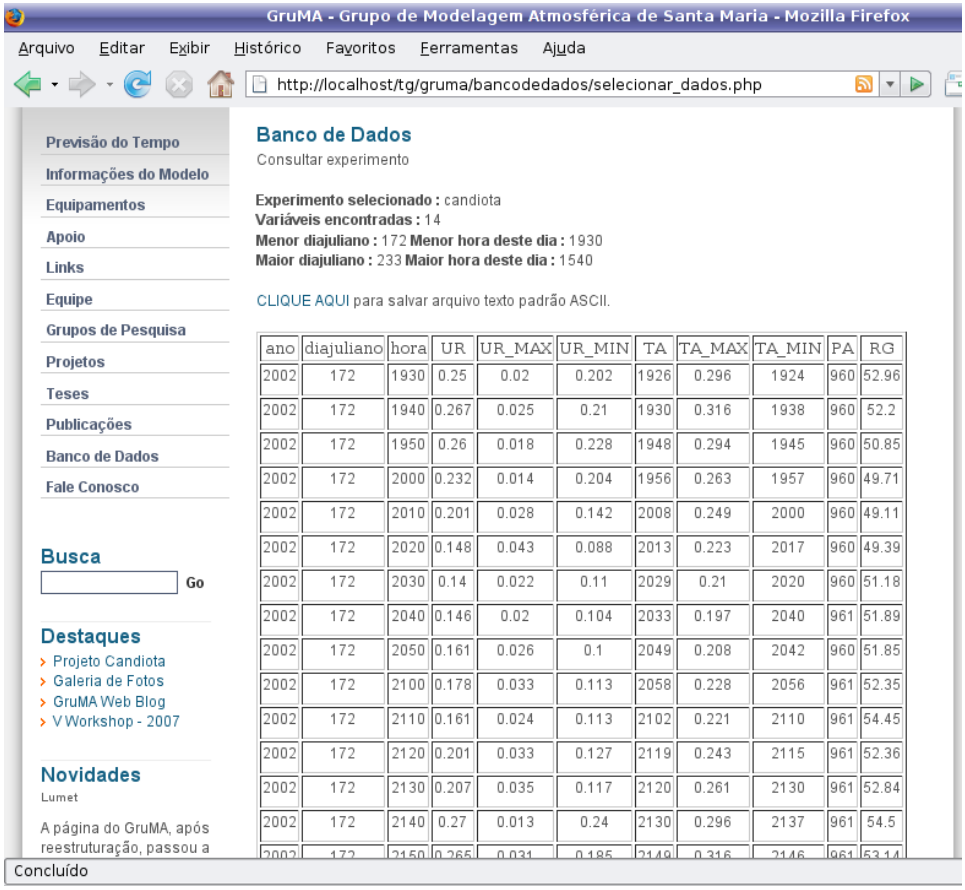
Por fim, após definição das opções de consulta, o usuário envia ao sistema através do botão "*Enviar*" e a execução prossegue ao próximo e último passo:

5. o sistema faz a consulta no banco de dados e retorna ao usuário uma interface *html* disponibilizando o link para um arquivo texto apenas com os dados selecionados através da execução dos códigos PHP do *script* "*selecionar*_dados" demonstrado através da figura 27.

```php
<?php require("bd_topo800.php");
$table = $_POST[experimento];
$diajuliano_min = $_POST[diajuliano_min];
$hora_min = $_POST[hora_min];
$diajuliano_max = $_POST[diajuliano_max];
$hora_max = $_POST[hora_max];
if ( $diajuliano_min < '1' or $diajuliano_min > '365' or $diajuliano_min > $diajuliano_max ) { echo " Não foi possível cosultar, por favor, verifique o campo <b>diajuliano mínimo</b> que obrigatoriamente deve estar entre 1 e 365 e ser menor que o <b>diajuliano máximo</b>."; require("bd_bottom800.php"); exit;}
if ( $diajuliano_max > '365' or $diajuliano_max < '1' or $diajuliano_max < $diajuliano_min ) { echo " Não foi possível cosultar, por favor, verifique o campo <b>diajuliano máximo</b> que obrigatoriamente deve estar entre 1 e 365."; require("bd_bottom800.php"); exit;}
if ( $hora_min < '0' or $hora_min > '2359') { echo " Não foi possível cosultar, por favor, verifique o campo <b>hora mínima</b> que obrigatoriamente deve estar entre 0 e 2359."; require("bd_bottom800.php"); exit;}
if ( $hora_max < '0' or $hora_max > '2359') { echo " Não foi possível cosultar, por favor, verifique o campo <b>hora máxima</b> que obrigatoriamente deve estar entre 0 e 2359."; require("bd_bottom800.php"); exit;}
echo "<br><b>Experimento selecionado : </b>".$table;
require("conect_db.php");
$fields = mysql_list_fields($banco,$table,$db);
$columns = mysql_num_fields($fields);
echo "<br><b>Variáveis encontradas : </b>".$columns;
echo "<br><b>Menor diajuliano : </b>".$diajuliano_min." <b>Menor hora deste dia : </b> ".$hora_min;
echo "<br><b>Maior diajuliano : </b>".$diajuliano_max." <b>Maior hora deste dia :</b> ".$hora_max;
echo "<br><br>";
$filename = 'temp/'.$table.'.txt';
$fp = fopen ($filename, 'w');
$fimcabecalho = "\n";
?><a href="<?php echo $filename ?>" target="_blank">CLIQUE AQUI</a>  para salvar arquivo texto padrão
ASCII.</font></center></p><br><br><?php
if (!$db) {
    die('Não foi possível conectar: ' . mysql_error());
}
mysql_select_db('grumadb');
$query="select * from ".$table." LIMIT 0 , 1";
$result = mysql_query("$query");
if (!$result) {
    die('A consulta falhou: ' . mysql_error());
}
?>
<table border="1"><tr>
<?php
$i = 0;
while ($i < mysql_num_fields($result))
{
        $meta = mysql_fetch_field($result, $i);
        if (!$meta) { echo "Sem informação disponível<br />\n"; }
        $campo="$meta->name";
         switch ($_POST[$campo]) {
               case "sim":
              ?><td><center><font size="2" face="Arial Narrow"><?php echo $campo;?></td><?
                fwrite($fp, "$campo,");
                break;
                }
    $i++;
}
fwrite($fp, $fimcabecalho);
mysql_free_result($result);
?></tr><?php // 2a Parte => Conexão com o banco de dados ?>
<tr>
<?php
$query="select * from ".$table." where $diajuliano>=$diajuliano_min and $diajuliano<=$diajuliano_max and $hora>=$hora_min and $hora<=$hora_max";
$res=mysql_query($query);
        while ($row=mysql_fetch_array($res)) {
            $query2="select * from ".$table." LIMIT 0 , 1";
            $result = mysql_query("$query2");
            if (!$result) { die('A consulta falhou: ' . mysql_error()); }
            $i = 0;
            while ($i < mysql_num_fields($result)) {
                $meta = mysql_fetch_field($result, $i);
                if (!$meta) { echo "Sem informação disponível<br />\n"; }
                $campo="$meta->name";
                switch ($_POST[$campo]) {
                 case "sim":
                  ?><td><p align="center"><? echo $row[$campo]; ?></td><?
                  fwrite($fp, $row[$campo].",");
                   break;
                 }
                 $i++;
            }
            mysql_free_result($result);
            fwrite($fp, $fimcabecalho);
            ?><tr><?
        }
?></table><?php require("bd_bottom800.php"); ?>
```

Figura 27: Códigos PHP do *script* "*selecionar*_dados"

A interface gerada é demonstrada através da figura 28 que permitirá ao usuário visualizar os dados selecionados através de uma tabela *html* e fazer *download* do arquivo texto no padrão ASCII.



ano	diajuliano	hora	UR	UR_MAX	UR_MIN	TA	TA_MAX	TA_MIN	PA	RG
2002	172	1930	0.25	0.02	0.202	1926	0.296	1924	960	52.96
2002	172	1940	0.267	0.025	0.21	1930	0.316	1938	960	52.2
2002	172	1950	0.26	0.018	0.228	1948	0.294	1945	960	50.85
2002	172	2000	0.232	0.014	0.204	1956	0.263	1957	960	49.71
2002	172	2010	0.201	0.028	0.142	2008	0.249	2000	960	49.11
2002	172	2020	0.148	0.043	0.088	2013	0.223	2017	960	49.39
2002	172	2030	0.14	0.022	0.11	2029	0.21	2020	960	51.18
2002	172	2040	0.146	0.02	0.104	2033	0.197	2040	961	51.89
2002	172	2050	0.161	0.026	0.1	2049	0.208	2042	960	51.85
2002	172	2100	0.178	0.033	0.113	2058	0.228	2056	961	52.35
2002	172	2110	0.161	0.024	0.113	2102	0.221	2110	961	54.45
2002	172	2120	0.201	0.033	0.127	2119	0.243	2115	961	52.36
2002	172	2130	0.207	0.035	0.117	2120	0.261	2130	961	52.84
2002	172	2140	0.27	0.013	0.24	2130	0.296	2137	961	54.5
2002	172	2150	0.265	0.031	0.185	2149	0.316	2146	961	53.14

Figura 28: Interface *html* apenas com os dados selecionados

3.5 – Resultado final da modelagem do sistema

O resultado final da modelagem do sistema é demonstrado através do esquema mostrado na figura 29, que ilustra o funcionamento do SGDM.

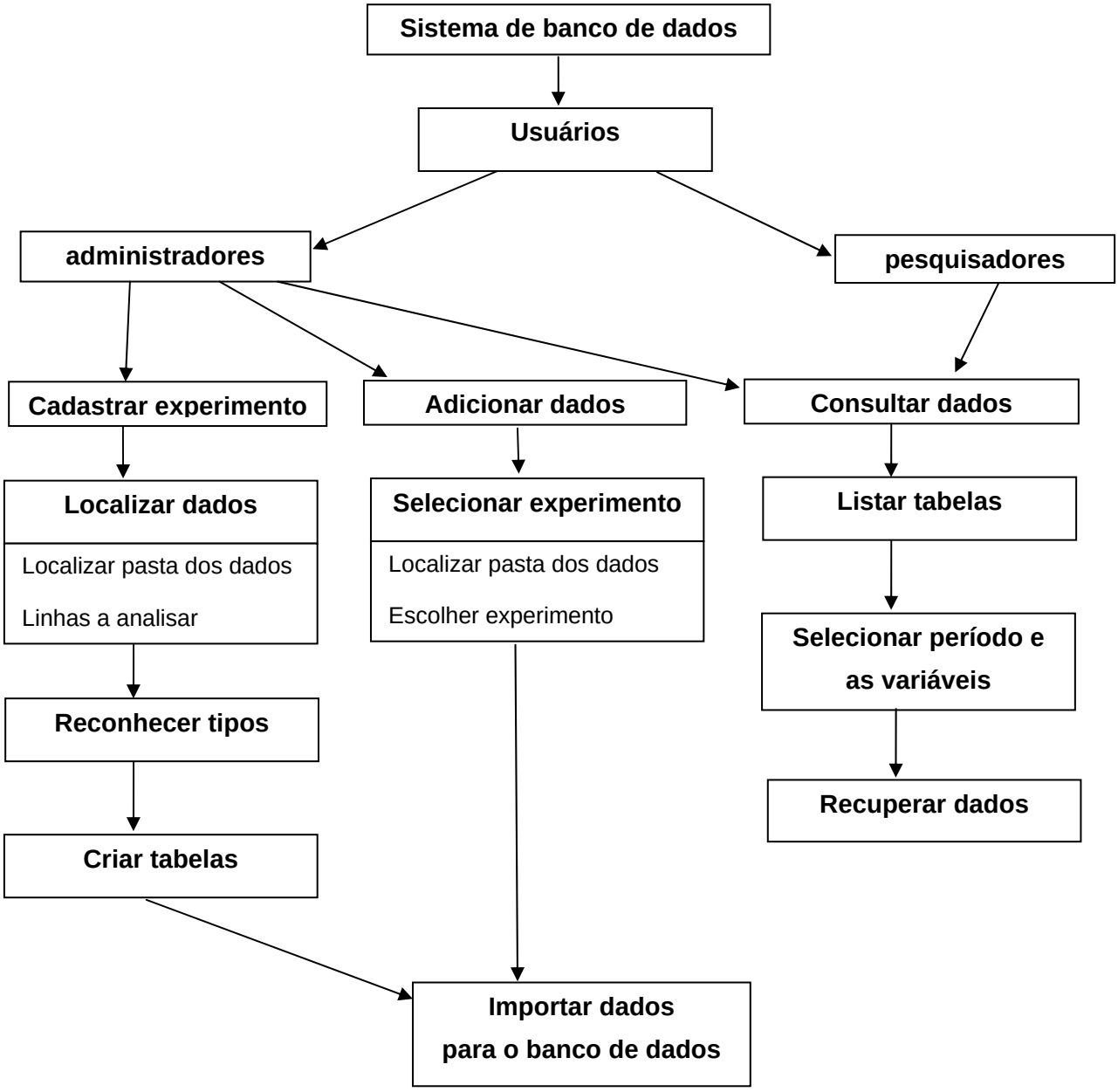


Figura 29: Esquema de funcionamento do sistema

4 – Conclusões

As conclusões obtidas na implementação do SGDM a partir do uso da tecnologia PHP como linguagem de programação padrão dos *scripts* que compõe o sistema, do MySQL como SGBD padrão SQL, ambos integrados com um servidor *web* Apache foram notoriamente satisfatórias.

Os resultados obtidos com a linguagem PHP foram muito favoráveis, pois ela apresentou inúmeras vantagens, dentre as quais pode-se destacar:

- a possibilidade de programar scripts que são executados direto no servidor permitiu que o sistema mantivesse uma forte interação com o servidor de banco de dados, além de permitir o uso do SGDM em qualquer lugar do mundo, via *Internet.*
- a vantagem de ser uma linguagem gratuita isentou este trabalho de custos em relação a licença de software;
- por ser uma linguagem multiplataforma, permitiu que o SGDM possa ser acessado por diversos sistemas operacionais, como Linux e Windows por exemplo;
- ela se mostrou rápida e eficiente nas execuções favorecendo um bom desempenho no sistema como um todo;
- além de ter servido para criação da interface do sistema, contribuiu intensamente na criação dos *scripts*, pois a vasta biblioteca de funções dessa linguagem, principalmente as relacionadas com o SGBD, permitiu que programas complexos fossem desenvolvidos, sem que isto implicasse em uma busca externa por recursos de programação, propiciando um ganho considerável de tempo no decorrer da implementação.

A implementação do SGDM baseado no MySQL 5.0, demonstrou que esta foi uma ótima escolha, pois mesmo com o grande volume de dados utilizados durante os testes, e que pretende-se trabalhar, pode-se verificar que foi consumido apenas a banda necessária a cada requisição, ou seja, somente os dados necessários para obtenção do resultado desejado, foram buscados no servidor.

A combinação da dupla MySQL + PHP mostrou que, durante os testes, a manipulação de dados apresentou baixos tempos de resposta o que resultou em uma ótima performance para o sistema.

A escolha das ferramentas para implementação deste trabalho foi criteriosa, portanto, o resultado foi o sucesso obtido no desenvolvimento do sistema, que durante toda sua

implementação e principalmente na fase de testes, se comportou de maneira satisfatória e esperada. Isto deixou claro que a escolha deste conjunto de aplicações foi notoriamente acertada para o desenvolvimento deste trabalho.

Vale ressaltar que, apesar do SGDM já estar pronto para o funcionamento, ele ainda passará por inúmeras atualizações, pois, ele ainda necessita de um desenvolvimento na parte de segurança e tolerância à falhas. Ele também passará por correções a pequenas falhas que só poderão ser detectadas com o uso da aplicação e com o crescimento do banco de dados, visto que, teoricamente, não haverá limite para a quantidade de dados a serem incluídos em um experimento, e nem limite para a quantidade de experimentos a serem realizados. Pois, se um experimento foi realizado, seus dados deverão persistir no banco de dados por tempo indeterminado.

Como projeto futuro, é possível destacar a intenção de criar um *link* via *internet* entre o SGDM e as torres de coleta, com a finalidade de adicionar novos dados ao banco de dados com o menor tempo possível em relação à hora de sua coleta, mantendo o banco de dados atualizado praticamente em tempo real. Para isso, será necessário que o SGDM passe por adaptações com a finalidade de obter o suporte a tais funcionalidades.

Por fim, o emprego deste sistema para o gerenciamento dos dados coletados em experimentos micrometeorológicos passará a ser de fundamental importância para o GRUMA, pois a partir da implantação e funcionamento do sistema, não será mais necessária a preocupação com o armazenamento e controle dos dados, uma vez que, se submetidos ao sistema, terão um destino confiável e persistente.

6 – Bibliografia

ACEVEDO, Otávio Costa. **V Workshop 2007** – Documento de Justificativa de realização do evento. http://www.gruma.ufsm.br/workshop, 11 de junho, Ano 2007.

ACHOUR, Mehdi; et al. **Manual do PHP** – Site Oficial do PHP, http://www.php.net, Ano 2007.

ALECRIM, Emerson. **Linguagem PHP** – Artigo do site Infowester, http://www.infowester.com, Ano 2003.

APOSTILANDO. **UML – Unified Modeling Language**. Apostila disponibilizada pelo site http://www.apostilando.com.br, Ano 2007.

APACHE. **Apache Software Foundation** – http://www.apache.org, Ano 2007.

CAMARGO, Wladimir Pena. **Desenvolvimento de um ambiente *web* para a interação entre participantes de projetos de agricultura de precisão** – Tese de mestrado, USP, Piracicaba, SP, Março, Ano 2005.

CARMO, Dyego Souza. **Tutorial sobre o novo MySQL 4** – Publicado no site Br.Linux.Org. http://brlinux.linuxsecurity.com.br/tutoriais, Ano 2003.

CODD, E. F. **Relational databases: a practical foundation for productivity** – New York, ACM Press, Ano 1982.

CRIAR WEB. **Introdução às linguagens da *web*** – Artigo do site Criar Web, manuais e recursos para desenvolvimento *web*, http://www.criarweb.com/artigos/202.php, Ano 2005.

DATALOGGER. **Campbell Scientific** – http://www.campbell.com, Ano 2007.

DATE, C.J. **Introdução a Sistemas de Bancos de Dados** - tradução (4ª edição americana). Editora Campus, Rio de Janeiro, Ano 1994.

FERREIRA, Cláudio Luiz. **Framework para gerenciar dados de interação do usuário em ambientes hipermídia de aprendizagem** – Dissertação de mestrado, UFSC, Florianópolis, Ano 2003.

GPL – **General Public License** – http://www.gpl.org, Ano 2000.

GUIMARÃES, Ana Paula; et al. **O mundo on-line - banco de dados** – Rev. PEC, Curitiba, v.1., n.1, p.15-22, julho, Ano 2001.

KROENKE, David M. **Banco de Dados: Fundamentos, projeto e implementação**. Sexta edição (tradução). LTC – Livros Técnicos e Científicos, Ano 1999.

LEITE, L. L.P. **Introdução aos sistemas de gerência de banco de dados** – Editora Edgard Blücher, São Paulo, Ano 1980.

MOLINA, Hector Garcia. **Implementação de Sistemas de Banco de Dados** – Editora Campus, Rio de Janeiro – Ano 2001.

MYSQL. **Manual de Referência do MySQL 4.1** – Site Oficial do MySQL, http://www.mysql.com, Ano 2007.

NAVATHE, Elmari. **Sistema de Banco de Dados** – 4ª Edição, Editora Pearson, Adison Wesley, Ano 2001.

SCHMULLER, Joseph. **Aprendendo UML em 24 horas** – Editora Prentice Hall, Ano 2004.

SILBERSCHATZ, Abraham; Korth, Henry F. **Sistema de banco de dados** – Editora Makron Books. 2ª.ed. São Paulo:, Ano 1995.

STADZISZ, Paulo Cezar. **Projeto de Software usando a UML** – CEFET-PR, Curitiba, PR, Ano 2002.

TEOREY, T. J. **Database modeling & design** – San Francisco, Morgan Kaufman, Ano 1999.

VILALTA, Josep. **UML Guía Visual: Cómo crear formas de vida organizativa** – Publicado por Vico.org - http://www.vico.org/UMLguiavisual, 03 de setembro, Ano 2001.

W3. **Extensible Markup Language (XML) 1.0.** Publicado em: http://www.w3.org/TR/2004/REC-xml-20040204, 17 de julho 2006.