UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE HAUBER DA SILVA

SensoRem: um sistema Web para Armazenamento e Análise de Dados de Sensoriamento Remoto Geológico

Trabalho de Graduação.

Prof. Dr. Leandro Krug Wives
Orientador

Profa. Dra. Silvia Beatriz Alves Rolim
Coorientadora

Porto Alegre, dezembro de 2012.

AGRADECIMENTOS

Primeiramente agradeço à minha família, que nunca deixou de acreditar em mim e de me incentivar, dando apoio e conforto nos momentos difíceis e compartilhando da alegria nos momentos de felicidade.

Agradeço aos amigos, por estarem sempre ao meu lado, dispostos a ajudar, a ouvir e a dividir não só os momentos bons. Por compreenderem a ausência devido aos trabalhos e pelos constantes convites para celebrar as conquistas e esquecer os problemas.

Agradeço também àquelas pessoas que por algum motivo não puderam estar sempre presentes, mas que quando estiveram, foram de vital importância para mim.

Agradeço à Profa. Sílvia Beatriz Alves Rolim por disponibilizar o projeto, ajudar sempre que necessário, estar presente na apresentação deste projeto no Salão de Iniciação Científica e, principalmente, por me incentivar e acreditar no meu trabalho.

Por fim, agradeço ao Prof. Leandro Krug Wives, meu orientador, que foi incrivelmente solícito, compreensivo com minhas outras atividades, sempre disposto a compartilhar seu tempo e conhecimento. Agradeço por jamais ter negado ajuda, por ter me apresentado ao grupo do projeto e, por, principalmente, ter sido um mentor e amigo.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
CASE	Computer-Aided Software Engineering
CRUD	Create, Read, Update, Delete
CSS	Cascade Style Sheet
DB	Data Base
HTML	HyperText Markup Language
PHP	PHP Hypertext Preprocessor
UFRGS	Universidade Federal do Rio Grande do Sul
UML	Unified Modeling Language
INF	Instituto de Informática

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A Web disponibiliza uma grande quantidade de conteúdo, muitas vezes sem a garantia da qualidade da informação, nas mais diversas áreas. Entretanto, na área de sensoriamento remoto aplicado à Geologia, essa disponibilidade ainda é esparsa. O projeto aqui apresentado tem como objetivo criar uma base de dados online (um repositório) que seja de acesso livre para o público, garantindo a qualidade da informação através do cadastro de usuários controlados, que poderão adicionar dados referentes à análise de assinaturas espectrais de amostras de rochas e solos obtidas através técnicas de espectroscopia de reflectância e de emissividade. São abordadas as técnicas utilizadas para desenvolvimento do sistema, assim como toda sua concepção e modelagem, além da arquitetura e softwares utilizados. São exibidas telas demonstrando a versão atual do sistema, explicando sua utilização, como um tutorial, com o intuito de demonstrar que o objetivo traçado foi atingido.

Palavras-Chave: Web, internet, sensoriamento remoto, geologia, repositório digital.

SensoRem – A Web System to Store and Analyze Geological Remote Sensing Data

ABSTRACT

There is a major quantity of content available on the web, frequently without the warranty of the information's quality, on the most diverse fields. However, in the remote sensing field, applied to Geology, this availability is reduced. The project presented here has the objective to create an online database (i.e., a repository) with free public access, guaranteeing the quality of the information through controlled registry of users that are allowed to add data related to the analysis of rocks and soils samples' spectral signatures through techniques of spectroscopy of reflectance and emissivity. The techniques used to develop the system, as well as its conception, modeling, architecture e software are described and commented. Screenshots of the system's actual version explains its usability, with the intention to demonstrate that the goals were reached.

Keywords: Web, internet, remote sensing, geology, online database.

1 INTRODUÇÃO

O armazenamento e o compartilhamento de informações de análises de dados espectrais sobre sensoriamento remoto na Geologia de forma digital é limitado. Um exemplo é o sistema web disponibilizado e mantido pela NASA, chamado ASTER Spectral Library (http://speclib.jpl.nasa.gov/), que mantém dados de inúmeras amostras. Porém, em tal sistema, a adição de novas amostras e a manutenção das informações é realizada por apenas uma única pessoa, o administrador do sistema. A necessidade de um sistema menos restrito e com a possibilidade de adição de conteúdo de modo mais dinâmico gerou o interesse no desenvolvimento de uma aplicação web com foco acadêmico que suprisse essa necessidade. Esses sistemas com características colaborativas têm grande importância na web, aumentando a quantidade de informações disponíveis na rede e sendo de grande ajuda para atividades de ensino e pesquisa.

O objetivo deste trabalho consiste no projeto e no desenvolvimento de um sistema intitulado "SensoRem", que é um sistema web cujo objetivo é cadastrar dados de assinaturas espectrais de amostras de rochas e solos obtidas através técnicas de espectroscopia de reflectância e de emissividade. Vale mencionar que o presente trabalho é parte integrante de um projeto de mapeamento geológico das rochas vulcânicas da Bacia do Paraná, RS, com técnicas de sensoriamento remoto, cuja contribuição será proporcionar condições para formação de recursos humanos de alto nível em meio à grande quantidade de informações na web.

Para a realização do trabalho, houve o estudo do sensor ASTER e a compreensão do que é o sensoriamento remoto e como ele se aplica à Geologia, discutido no segundo capítulo deste trabalho, junto à análise de requisitos do sistema. A tradução dessa análise se deu na forma de diagramas, sendo eles o Diagrama de Casos de Uso, Diagrama de Classes, Diagrama Entidade-Relacionamento, Diagrama de Navegação, Diagrama de Acesso e Diagrama de Apresentação, que serão explicados no capítulo terceiro, assim como as tecnologias utilizadas no desenvolvimento do sistema.

Por fim, no capítulo 4 será demonstrada a utilização do sistema através de imagens referentes às telas do sistema acompanhadas de suas respectivas explicações, além de possíveis *features* novas. As conclusões, resultados, limitações do sistema e trabalhos futuros se encontram no quinto e último capítulo.

2 ANÁLISE DE REQUISITOS

Neste capítulo é apresentada a definição de sensoriamento remoto e como é dada sua utilização na Geologia, para facilitar a compreensão do funcionamento geral do sistema. É apresentado um estudo do sistema ASTER citado anteriormente e, a partir disso, são definidos os requisitos do sistema, tanto os funcionais como os não funcionais.

2.1 Sensoriamento Remoto

"O sensoriamento remoto é a ciência e arte de receber informações sobre um objeto, uma área ou um fenômeno pela análise dos dados obtidos de uma maneira tal que não haja contato direto com este objeto, esta área ou este fenômeno" (LILLESAND; KIEFER, 1987). Um de seus objetivos é o de medir quantidades físicas na superfície terrestre, assim como temperatura, radiância, emissividade e reflectância. Na Geologia e solos procuram-se as assinaturas espectrais de minerais, tipos de solo, entre outros (VAN DER MEER; DE JONG, 2002).

Os produtos obtidos com o sensoriamento remoto orbital e proximal têm auxiliado no mapeamento geológico e exploração mineral desde escalas regionais até microscópicas. Dados multiespectrais e hiperespectrais apresentam informações sobre as propriedades físico-químicas resultantes da interação dos alvos (rochas, minerais, solos, vegetação, etc.) com a radiação eletromagnética. Os principais produtos obtidos são imagens de satélite e curvas espectrais, estas utilizadas para identificação e mapeamento detalhado de rochas e minerais. Tais curvas podem ser obtidas tanto de imagens quanto de espectrorradiômetros de campo e laboratório. Neste trabalho serão apresentadas curvas de reflectância e emissividade obtidas do espectrorradiômetro uFTIR Modelo 102 (HOOK; KAHLE, 1996).

Como explicado, os dados que são importantes para o sistema e serão utilizados na geração de gráficos de visualização são os resultados obtidos através de técnicas de espectroscopia de reflectância e emissividade. Tais técnicas de sensoriamento remoto permitem registrar informações espectrais de alvos a partir da emissão de energia de uma fonte de radiação eletromagnética (REM). A principal fonte é o sol, mas também podem ser utilizadas lâmpadas em medições de laboratório. A fonte de REM, ao interagir com os alvos, pode ser parcialmente refletida, absorvida, transmitida e emitida. As taxas de reflectância e emissividade são utilizadas neste trabalho.

O espectrorradiômetro funciona conectado a um computador que gera um arquivo texto contendo duas colunas com várias linhas. A primeira coluna contém o comprimento da onda emitida pelo espectrorradiômetro e representa o eixo *x* do gráfico, e a segunda a porcentagem de reflectância ou emissividade da amostra, representando o

eixo *y*. O gráfico traçado com esses valores representa a assinatura espectral dessa amostra.

Abaixo, na Tabela 2.1, encontra-se um exemplo com apenas cinco linhas de valores representando os dados presentes no arquivo texto gerado pelo espectrorradiômetro.

Tabela 2.1: Exemplo de dados gerados pelo espectrorradiômetro

A	*B*
2,99945	8,51958
3,00206	7,08235
3,00468	8,19624
3,0073	7,36897
3,00993	6,9919

Fonte: tabela criada pelo autor com dados de uma amostra de quartzo gerado por um espectrorradiômetro.

2.2 Estudo do Sistema ASTER

O estudo do sistema mantido pela NASA ocorreu com o intuito de entender o que é esperado de um sistema de armazenamento de informações de sensoriamento remoto geológico.

A partir desse estudo e da análise de requisitos foi elaborada a Tabela 2.2, encontrada abaixo, que explicita funcionalidades presentes no sistema, e funcionalidades que não estão presentes e justificam a criação de um novo sistema.

Tabela 2.2: Funcionalidades presentes e ausentes do sistema ASTER

Funcionalidade	*Presente no Sistema ASTER*	*Descrição*
Busca detalhada	Sim	Possibilidade de filtrar busca por nome, classe, subclasse e valores inicial e final do eixo *x* da amostra.
Informações sobre a amostra	Sim	Exibe várias informações sobrea a amostra. Exemplo na Figura 2.1, após a tabela.
Gráfico manipulável da assinatura espectral	Sim	É possível alterar o tamanho e o nome do título e dos eixos do gráfico e aplicar zoom. Exemplo nas Figuras 2.2, 2.3 e 2.4, após a tabela.
Possibilidade de salvar alterações no gráfico	Não	-
Possibilidade de inserir novas amostras	Não	-

Fonte: tabela criada pelo autor.

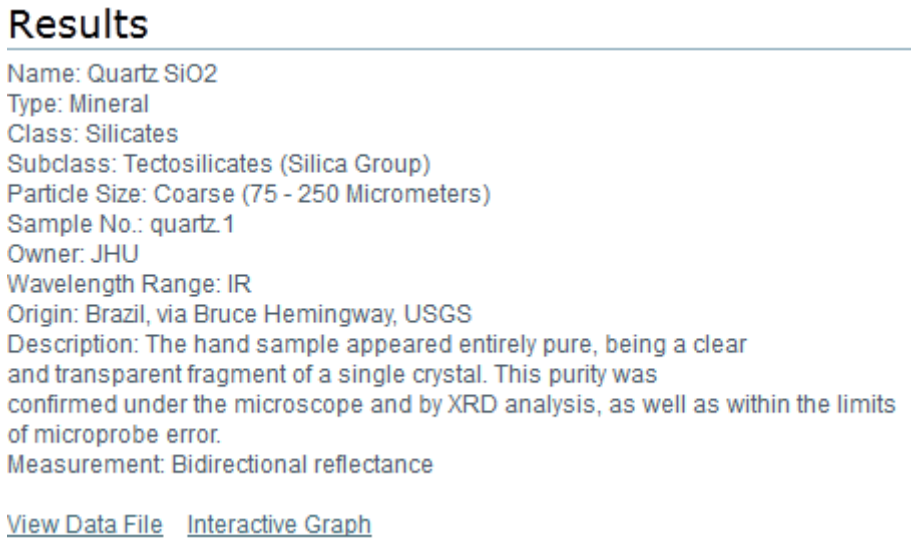


Figura 2.1 - Informações exibidas sobre uma amostra no sistema ASTER

Fonte: imagem salva pelo autor utilizando o sistema ASTER

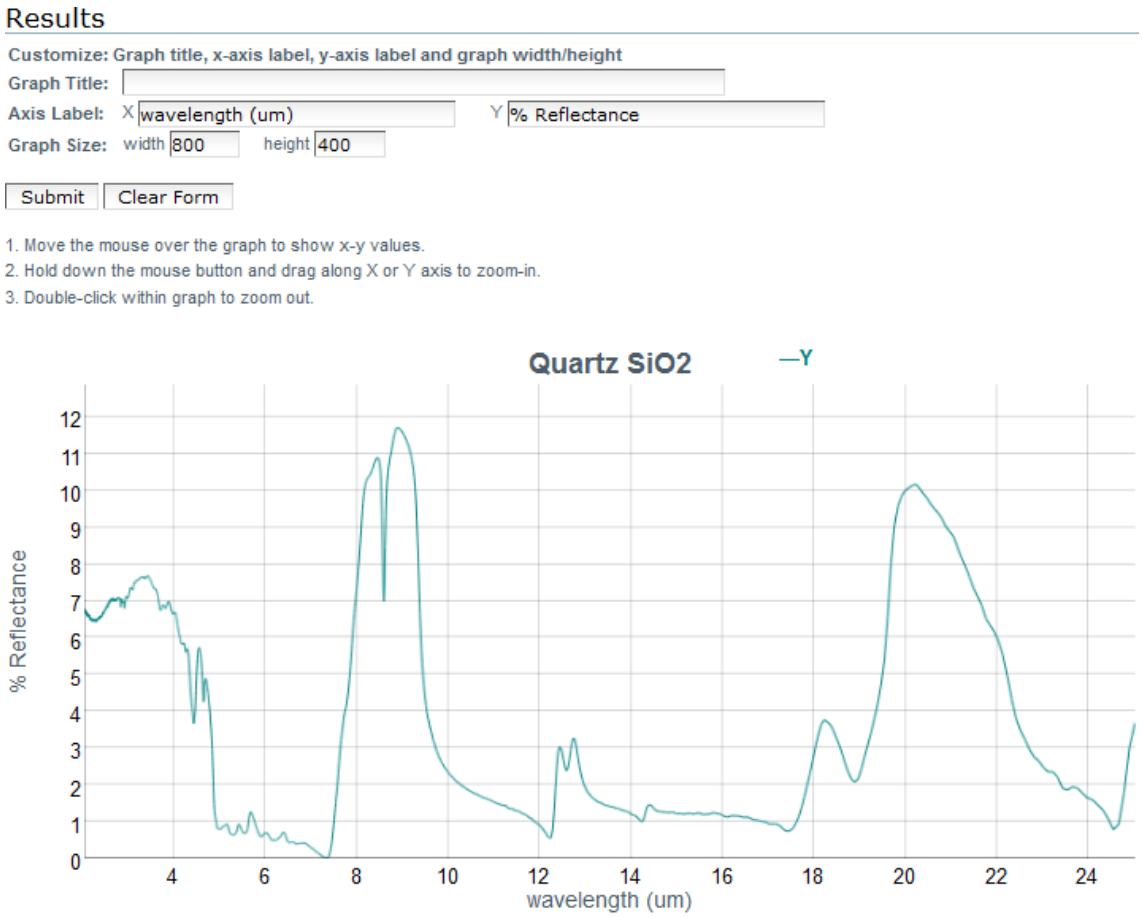


Figura 2.2 - Informações de uma amostra exibidas sem alterações no sistema ASTER

Fonte: imagem salva pelo autor utilizando o sistema ASTER

Abaixo, na Figura 2.3, alterou-se o título do gráfico, dos eixos e também o seu tamanho. Essas operações são comuns durante a análise dos dados, assim como a aplicação de zoom (aproximação).

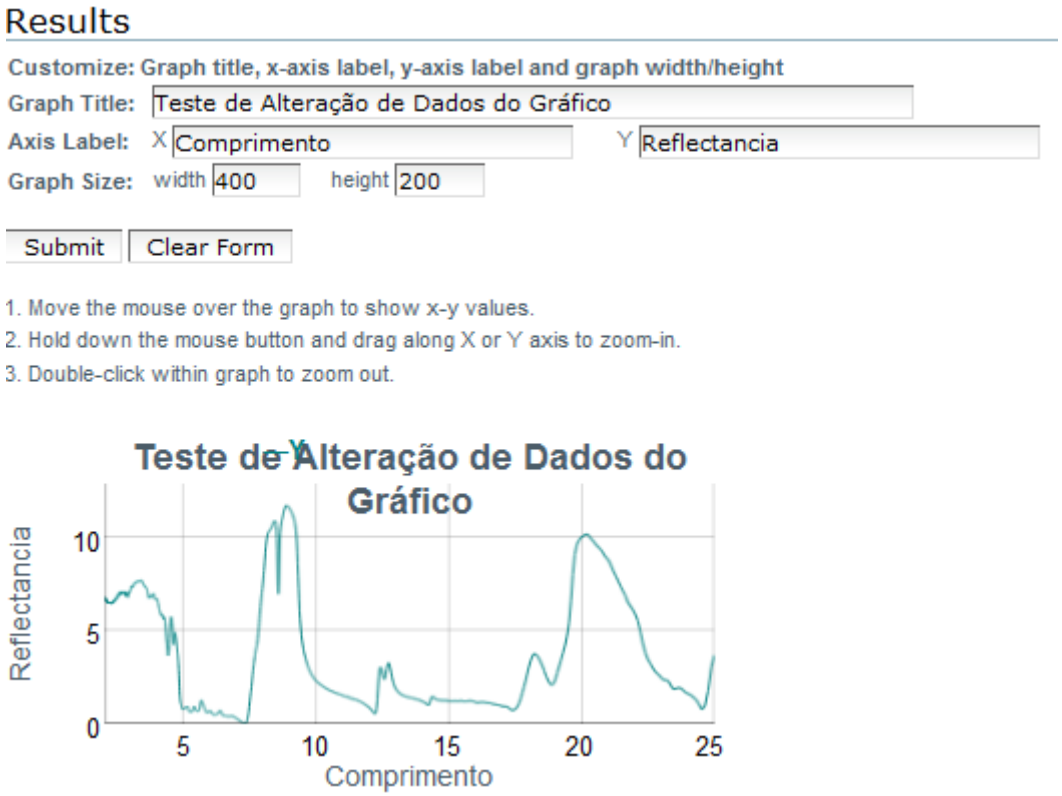


Figura 2.3 - Informações de uma amostra exibidas com alterações no sistema ASTER

Fonte: imagem salva pelo autor utilizando o sistema ASTER

Abaixo, na Figura 2.4, apresenta-se a aplicação de uma operação de *zoom* no gráfico modificado anteriormente.

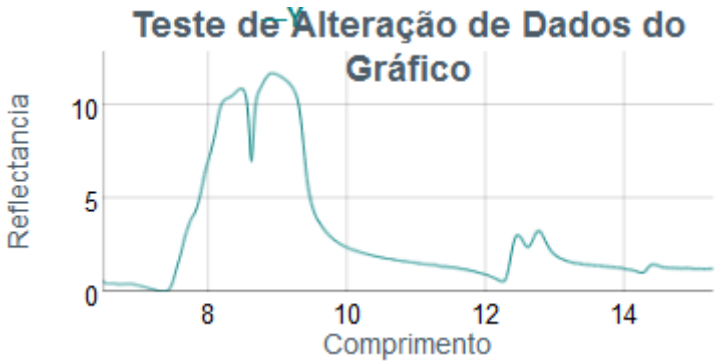


Figura 2.4 - Gráfico com *zoom* aplicado

Fonte: imagem salva pelo autor utilizando o sistema ASTER

2.3 Requisitos do Sistema

A partir da análise do sistema ASTER e de reuniões para definir os objetivos e funcionalidades presentes no sistema a ser desenvolvido, foi feito o levantamento dos requisitos funcionais e não funcionais do sistema aqui proposto.

2.3.1 Requisitos Funcionais

Serão listadas abaixo as funcionalidades que devem estar presentes no sistema.

- O acesso aos dados de amostras cadastradas no sistema é livre, sem a necessidade de ser um usuário registrado;
- Um usuário administrador gerencia o cadastro de novos usuários que possuem permissão para validar ou inserir amostras;
- Usuários sem cadastro devem possuir acesso a todas as informações de amostras disponíveis no sistema e poder fazer alterações no gráfico e salvar essas alterações localmente em seu computador sem alterar o conteúdo do banco de dados;
- O software deve permitir inserção, edição e exclusão de amostras, sendo que sempre que uma amostra for inserida ou editada, deve passar por validação;
- O software deve permitir aplicação de zoom no gráfico referente à assinatura espectral da amostra;
- O software deve permitir visualização de mais de uma assinatura espectral ao mesmo tempo no gráfico, para efeitos de comparação;

2.3.2 Requisitos Não Funcionais

Serão listadas questões de projeto, como plataformas e tecnologias escolhidas para desenvolvimento do sistema.

- O sistema deve ser disponibilizado livremente na web, sem interferência no funcionamento em qualquer sistema operacional desenvolvido para dispositivos não portáteis, assim como funcionar em qualquer navegador;
- O sistema não deve utilizar nenhuma ferramenta de desenvolvimento que envolva custos;
- O sistema deve ser desenvolvido nas linguagens PHP e HTML, por serem linguagens bem estabelecidas na comunidade online;
- O desenvolvedor deve utilizar um framework de desenvolvimento para padronizar o código e facilitar alterações posteriores por outros desenvolvedores;
- O sistema, se possível, deve utilizar tecnologias em ascensão, como HTML 5;
- O sistema deve utilizar banco de dados MySql Server, por ser gratuito e pela integração com a linguagem PHP;
- O sistema deve utilizar o servidor web Apache;
- O sistema deve possuir uma interface de fácil utilização e curva de aprendizagem pequena.

3 SISTEMA SENSOREM

Neste capítulo serão descritas a modelagem do sistema, sua arquitetura e as tecnologias utilizadas no seu desenvolvimento.

É importante salientar que não foi utilizada uma metodologia padrão para o desenvolvimento do sistema, porém foram adotados princípios de metodologia ágil (ABRAHAMSSON et al., 2002) e diagramas UML, com enfoque no desenvolvimento e modelagem para web (KAPPEL; PRÖLL; REICH; RETSCHTZEGGER, 2003), resultando em uma abordagem mista.

3.1 Visão Geral do Sistema

Detalhando o sistema através da visão dos requisitos, pode-se dizer que o mesmo realizará o gerenciamento de informações de amostras provenientes de análises de rochas e solos com a técnica de sensoriamento remoto geológico, garantindo a qualidade desses dados através da manutenção das informações por usuários com níveis de acesso diferentes que realizarão o cadastro e validação desses dados. As informações devem ser livres para o público em geral e devem ficar disponíveis na web sem restrição de plataforma de acesso, sistema operacional ou navegador de acesso à web utilizado.

Pela ótica do fluxo de informações, é correto dizer que cadastradores do sistema inserem dados referentes a uma amostra, que, além de dados básicos como descrição, nome e origem, possui um atributo muito importante, que é o arquivo de texto de valores semelhantes ao exemplo da seção 2.1. O passo seguinte é essa amostra ser validada por um validador do sistema. Quando isso ocorre, todos dados referentes a essa amostra já ficam disponíveis no repositório digital para o público.

Com os dados cadastrados e validados, o sistema pode exibir para os usuários todas as informações referentes à amostra e gerar um gráfico que representa a assinatura espectral dessa amostra. Esse gráfico é interativo, podendo sofrer manipulações de qualquer usuário, sem que isso acarrete em alterações nas informações do sistema. O comportamento do sistema ficará mais claro após o estudo da sua modelagem e do seu desenvolvimento.

3.2 Modelagem do Sistema

A seguir será detalhado o modelo do banco de dados para a aplicação proposta, assim como os casos de uso do sistema através de seus diagramas e suas descrições. Além destes, serão representados os diagramas de classe, entidade-relacionamento, navegação, acesso e apresentação.

3.2.1 Visão Geral

Há algumas propostas para modelagem de sistemas web, e não há um consenso sobre qual abordagem seguir, por isso adotou-se neste projeto a *UWE* (UML-based Web Engineering) (http://uwe.pst.ifi.lmu.de/) por ser semelhante às representações de sistemas tradicionais. A grande diferença e a motivação para se abordar de um modo diferente a modelagem para web é a ausência do conceito de hyperlinks nas ferramentas padrão.

Com esse conceito presente nos sistemas web, percebe-se uma navegação não linear através desses sistemas, onde é possível acessar uma mesma informação através de inúmeros caminhos diferentes. Por essas razões, a modelagem web possui geralmente três níveis, que são o conteúdo, ou seja, informações e aplicações, o hipertexto, que se refere à estrutura que liga os conteúdos aos seus links e entre si, e a apresentação, que se resume ao layout e interface do usuário (KAPPEL; PRÖLL; REICH; RETSCHTZEGGER, 2003). Para representar todos os níveis, utilizar-se-á a representação do sistema através dos diagramas casos de uso, do modelo do hipertexto, que engloba os diagramas de navegação e acesso, e do diagrama de apresentação.

3.2.2 Diagrama de Casos de Uso

Através da análise do diagrama de casos de uso, representado pela Figura 3.1, verifica-se que o sistema possui quatro tipos de atores: administrador, validador, cadastrador e visitante. Essa divisão existe, pois o sistema deve permitir com que qualquer pessoa tenha acesso às informações de amostras adicionadas nele, porém deve garantir a qualidade da informação nele armazenada. Através da descrição dos casos de usos será possível entender as funcionalidades disponíveis para cada ator.

Na abordagem de representação utilizada, há a indicação de quais requisitos são funcionais e quais são de navegação através de uma *flag* <<*navegação*>> (KAPPEL; PRÖLL; REICH; RETSCHTZEGGER, 2003). As *flags* também podem ser chamadas de estereótipos.

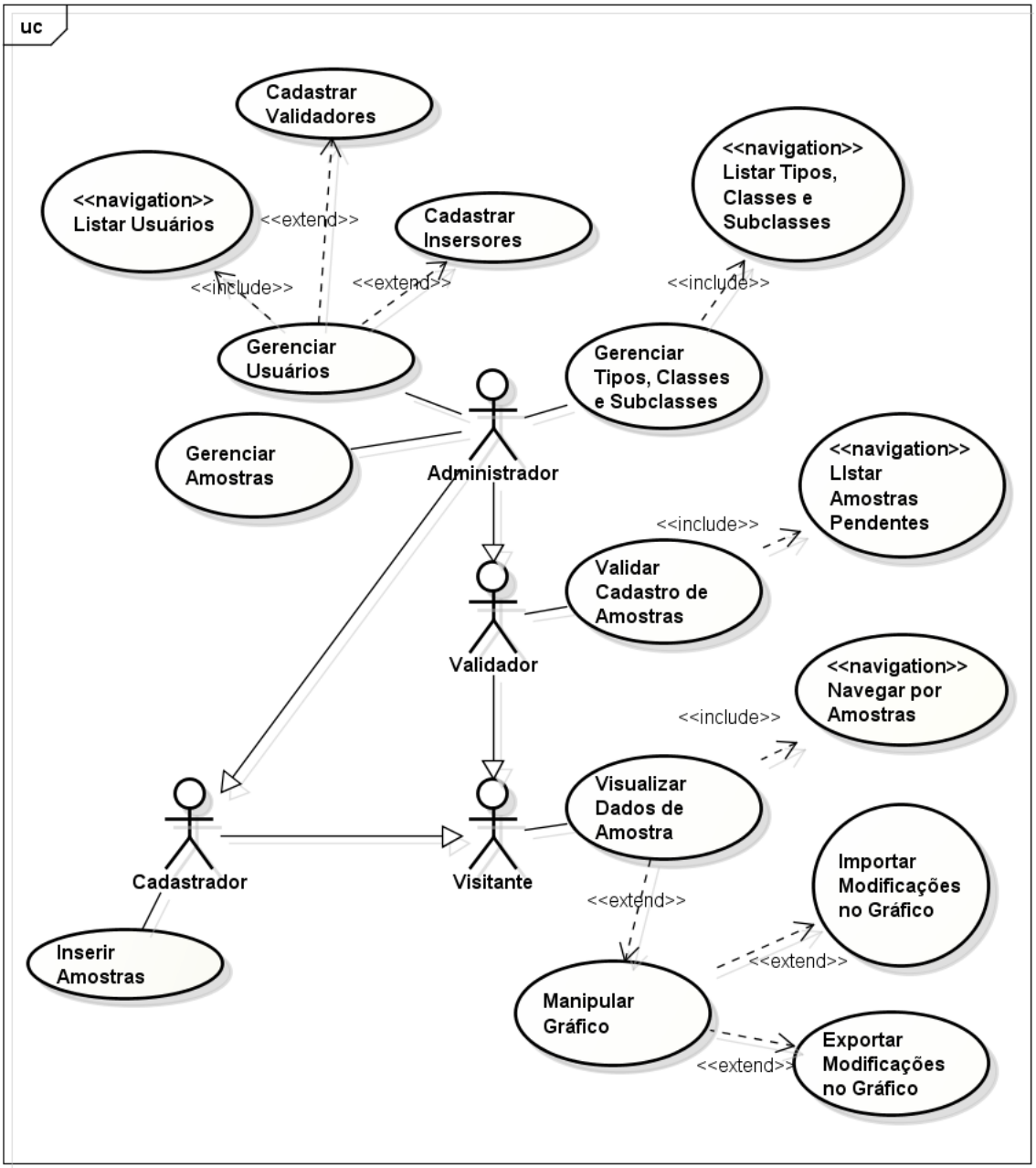


Figura 3.1 - Diagrama de casos de uso

Fonte: diagrama criado pelo autor

3.2.3 Descrição dos casos de uso

O sistema possui várias operações do tipo CRUD, de modo que a descrição de cada uma dessas operações seria bem semelhante. Para evitar a repetição de informações, será utilizado um caso de uso parametrizado (COCKBURN, 2005), que permite representar casos de uso semelhantes através de um *template*. Dessa forma, o caso de uso "Gerenciar Tipos, Classes e Subclasses" não será descrito e "Gerenciar Amostras" será descrito de forma menos detalhada, pois apresentam comportamento semelhante a "Gerenciar Usuários". Como não há uma definição formal sobre como proceder para descrever os requisitos de navegação, foi adotada a mesma forma utilizada para os requisitos funcionais descritas acima.

3.2.3.1 Caso de Uso: Gerenciar Usuários

Atores: Administrador.

Descrição Resumida: O administrador do sistema pode adicionar, editar e excluir do sistema usuários do tipo "validador" ou "cadastrador".

Fluxo de Eventos:

Fluxo Básico:

1. Ator seleciona opção Usuários no menu
2. Sistema apresenta uma lista com os usuários cadastrados no sistema.

Fluxo Alternativo:

A. Visualizar Usuário

1. Ator seleciona um usuário
2. Sistema apresenta as informações do usuário, se ele possuir amostras cadastradas, essa informação também é exibida

B. Adicionar Usuário

1. Ator seleciona a opção de Adicionar usuários no sistema
2. Sistema apresenta um formulário com os dados a serem preenchidas
3. Ator preenche os dados do usuário, incluindo a escolha da função de validador ou cadastrador
4. Se todos dados estiverem corretamente preenchidos, o sistema adiciona o usuário à base de dados, caso contrário retorna uma mensagem de erro para que o ator corrija os dados

C. Editar Usuário

1. Ator seleciona a opção de Editar usuário
2. Sistema apresenta informações atuais do usuário
3. Ator edita os dados do usuário e salva as alterações
4. Sistema atualiza dados na base de dados

D. Excluir Usuário

1. Ator seleciona opção de Excluir usuário
2. Sistema pede uma confirmação da ação e exclui da base de dados o usuário

3.2.3.2 Caso de Uso: Gerenciar Amostras

Atores: Administrador, Cadastrador

Descrição Resumida: O ator pode inserir uma amostra, que fica com sua validação pendente, e executar operações CRUD sobre uma amostra inserida por ele que seja válida.

3.2.3.3 Caso de Uso: Validar Amostras

Atores: Administrador, Validador

Descrição Resumida: O ator verifica os dados de uma amostra previamente inserida, podendo validá-la ou excluí-la

Fluxo de Eventos:

Fluxo Básico:

1. Ator seleciona opção Amostras no menu e a opção Validar Amostras no submenu
2. Sistema apresenta uma lista com as amostras a serem validadas
3. Ator escolhe amostra a ser verificada
4. Ator valida ou exclui amostra

3.2.3.4 Caso de Uso: Visualizar Amostras

Atores: Administrador, Validador, Cadastrador e Visitante.

Descrição Resumida: Os atores podem visualizar os dados de amostras e manipular os gráficos destas, com a opção de exportar e importar localmente essas modificações para posteriores visualizações.

Fluxo de Eventos:

Fluxo Básico:

1. Ator seleciona opção Amostras no menu e a opção Visualizar Amostras no submenu
2. Sistema lista em ordem alfabética as vinte primeiras amostras e exibe campo para filtrar a busca
3. Ator escolhe a amostra e visualiza seus dados, incluindo o gráfico que representa a assinatura espectral da amostra

Fluxo Alternativo:

A. Buscar Amostra

1. Ator entra com informações de filtragem de amostras, como tipo, classe, subclasse, localidade e amostras inseridas pelo ator
2. Sistema lista as amostras que respeitam os filtros impostos pelo ator

B. Manipular Gráfico

1. Ator pode dar *zoom* e alterar valores iniciais de exibição dos eixos x e y do gráfico

C. Importar Gráfico Manipulado

1. Ator importa dados do gráfico manipulado
2. Sistema exibe gráfico

C. Exportar Gráfico Manipulado

1. Ator exporta dados do gráfico manipulado
2. Sistema gera arquivo com os dados de acordo com a manipulação do ator para permitir importação posterior

3.2.4 Modelo Conceitual

Kappel (2003) define um modelo conceitual, cujo foco é representar as informações e requisitos funcionais do sistema, não levando em conta suas características de hipertexto.

A representação é separada em um diagrama de classes, que representa a estrutura do conteúdo do sistema, e diagramas de estados, que representa o comportamento do sistema, ou seja, seus estados e interações. O diagrama de estados não é considerado obrigatório para todos os sistemas.

3.2.4.1 Diagrama de Classes

O diagrama de classes, que está detalhado na Figura 3.2 a seguir, que, por não levar em conta os hyperlinks, é o mesmo modelo utilizado nas abordagens tradicionais de modelagem. Percebe-se que as principais classes do sistema são Amostras e Usuários, que possuem papel fundamental no funcionamento do mesmo.

Como esse diagrama foi utilizado posteriormente para mapear o diagrama de entidade-relacionamento de banco de dados, foram utilizados nomes e atributos em inglês para respeitar a indicação de uso adotada pelo CakePHP e para facilitar o uso de algumas de suas *features*. O framework não obriga a fazer o banco de dados em inglês, mas respeitar essa sugestão facilita muito o desenvolvimento. Alguns motivos serão abordados ao longo de outros capítulos.

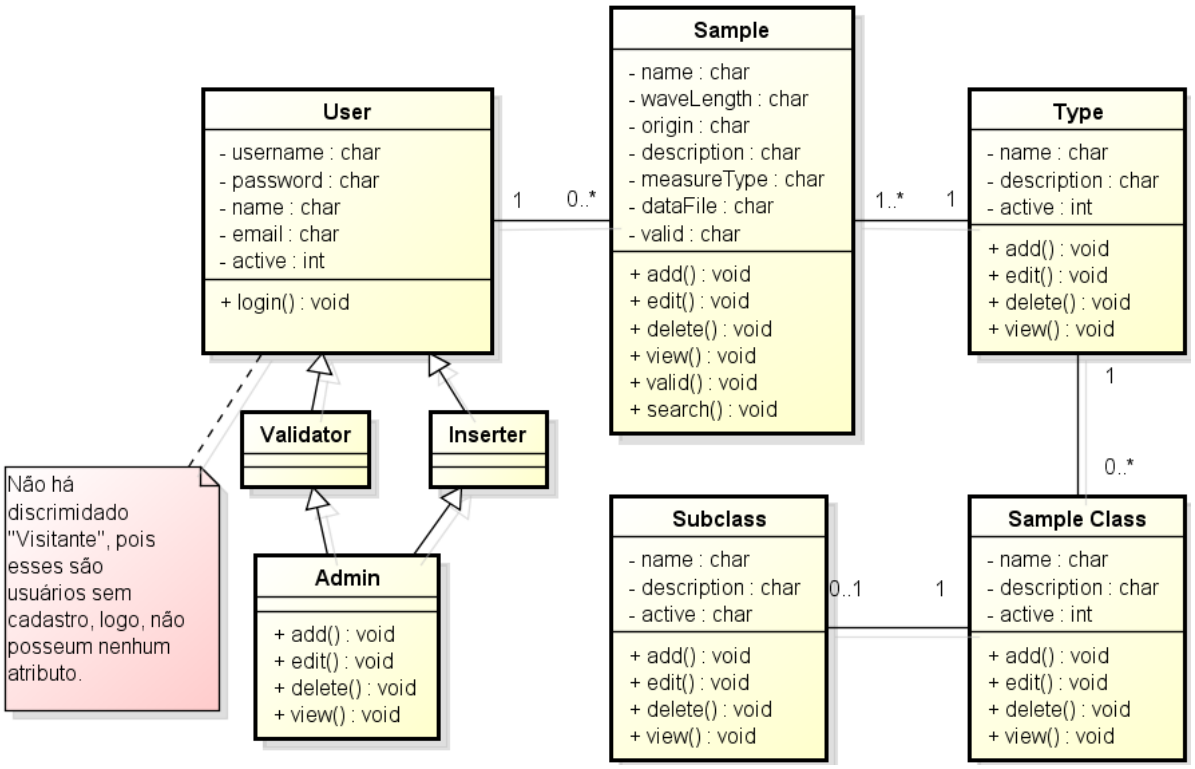


Figura 3.2 - Diagrama de classes

Fonte: diagrama criado pelo autor.

Pela análise, fica fácil perceber que o usuário Administrador (*Admin*) é o único que pode inserir novos usuários no sistema, não sendo possível um visitante entrar no site e optar por se cadastrar.

3.2.5 Banco de Dados

A partir do Modelo Conceitual, é possível chegar ao diagrama de entidade-relacionamento, representando o banco de dados do sistema. Para sua criação foi utilizado o software MySQL Workbench 5.2 (http://www.mysql.com/products/workbench/).

A configuração da base de dados é bem simples, sendo necessário apenas substituir um arquivo chamado *databaset.php.default* por um *database.php*. Abaixo segue a estrutura do arquivo.

```
class DATABASE_CONFIG {

      public $default = array(
            'datasource' => 'Database/Mysql',
            'persistent' => false,
            'host' => 'NomeDoHost',
            'login' => 'NomeDoUsuario',
            'password' => 'Senha',
            'database' => 'BaseDeDados',
            'prefix' => '',
            'encoding' => 'latin1',
               'collation' => 'latin1_swedish_ci'
      );
   }
```

Para realizar operações CRUD, por exemplo, não é necessário chamar uma função de conexão com o banco a cada acesso, o CakePHP conecta-se automaticamente sempre que uma *query* for executada. O framework dá liberdade para realizar *querys* utilizando a sintaxe padrão do MySql, ou sua própria sintaxe. Abaixo segue um exemplo de uma busca simples na base de dados, utilizando os dois modos de conexão.

```
  /* Conexão com query com sintaxe padrão */
       $this->set('samples',  $this->Sample->query("select  *  from  samples
where valid = 'sim' "));

  /* Conexão sugerida pelo framework */
$this->set('samples', $this->Sample->find('all',
      array('conditions' => array('Sample.valid' => 'sim'))));
```

A base de dados, representada pela Figura 3.3, é bem simples, sendo composta de usuários (tabela *users*), que podem possuir zero ou várias amostras (tabela *samples*). Essas amostras obrigatoriamente possuem um tipo (tabela *types*), que não necessariamente possui uma classe (tabela *sample_classes*) que, por fim, pode ou não possuir uma subclasse (tabela *subclasses*).

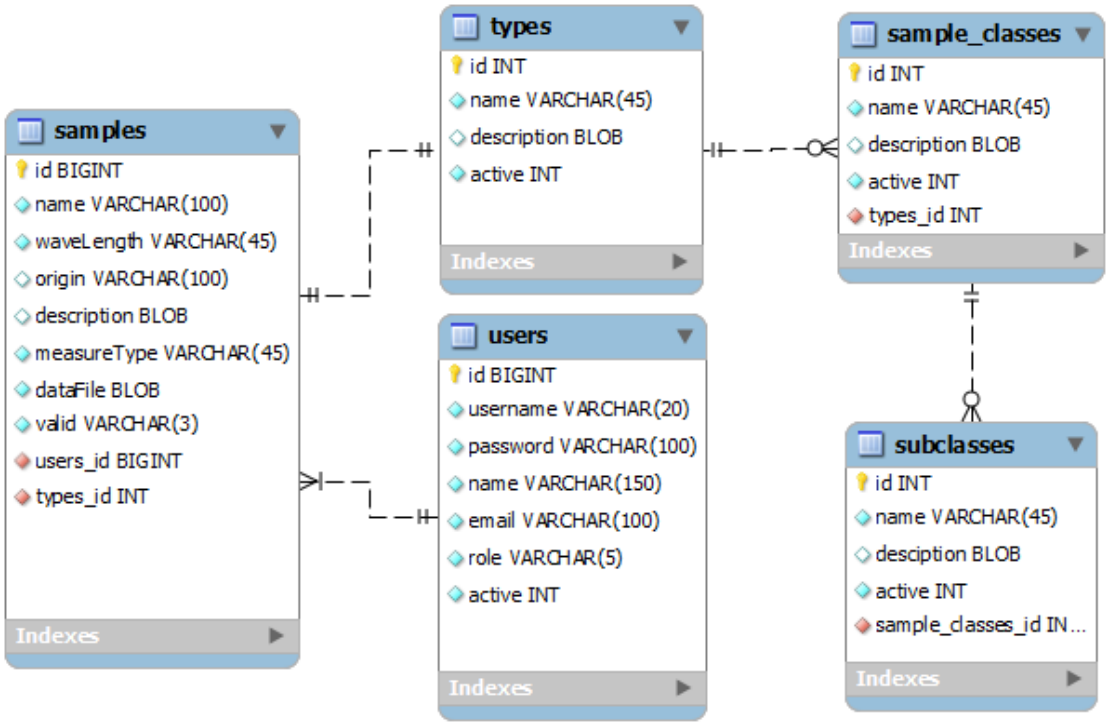


Figura 3.3 - Diagrama de entidade-relacionamento (ER)

Fonte: diagrama criado pelo autor

O campo “dataFile”, na tabela “samples”, representa o arquivo de dados exemplificado na seção 2.1 deste trabalho, e o campo “role” em “users”, representa o tipo de usuário, ou seja, se ele é administrador, validador ou cadastrador. Todos os outros campos das tabelas estão explicados no Apêndice A deste trabalho, onde há um dicionário de dados descrevendo cada um.

O banco de dados está em inglês, pois desse modo é mais fácil aproveitar certas características e funcionalidades do CakePHP. Um exemplo é a tabela de usuários, que se for chamada *users* e possuir os campos *username* e *password*, permite utilizar as funções de *login*(), *logout*() e *auth*() do framework. Essas são funções que gerenciam os controles de acessos e permissões do sistema, que são muito úteis e completas, de modo que não é necessário perder tempo desenvolvendo funções próprias com esses objetivos.

Outra imposição do framework é que todo campo identificador da tabela, que seja chave primária, deve ser chamar *id*. Se tentar utilizar, por exemplo, *users_id* como chave primária, o CakePHP não acusa erro de conexão, mas também não retorna nenhuma informação do banco. Essa situação não é detalhada na documentação do framework, e só utilizando o *debugger* do framework foi possível visualizá-la. Ele ignora a declaração da chave primária e procura pelas informações no campo *id* da tabela, mesmo ele inexistindo. É obrigatório também que as tabelas sejam nomeadas no plural, pois, de outro modo, a conexão falha.

O framework possui alguns nomes protegidos, que não podem ser utilizados pelo desenvolvedor, como o nome “*class*”. Por esse motivo, a tabela de classes ficou nomeada como *sample_classes*, pois o sistema simplesmente não funciona caso o modelo, controlador e tabela do banco de dados sejam nomeados como “class”. E, mais

uma vez, isso não é explicitado na documentação, e os erros reportados pelo framework não são explicativos.

3.2.6 Modelo do Hipertexto

Tendo como foco representar a estrutura de hipertexto da aplicação e a maneira como acessar os seus nodos, o seu objetivo principal é especificar os caminhos de navegação para o usuário, visto que o acesso à informação não é linear, devido à presença dos hyperlinks. Sua concepção é feita focando nos nodos, que são as páginas e documentos, e nas ligações entre esses nodos. O modelo de hipertexto possui dois componentes principais: o diagrama de navegação e o diagrama de acesso.

3.2.6.1 Diagrama de Navegação

É fortemente baseado no diagrama de classes e é sugerido que se faça um diagrama para cada tipo de usuário. Porém, para facilitar a compreensão do sistema como um todo, será criado apenas um diagrama contendo todos os papéis de usuários, como pode ser visto na Figura 3.4 seguinte.

Utiliza-se a *flag* <<*navigation class*>> para haver uma diferenciação das classes de conteúdo, e <<*navigation link*>> para explicitar a ligação entre as classes. Como ele é baseado no diagrama de classes que foi modelado em inglês, para manter um padrão esse diagrama e o de acesso também serão modelados em inglês.

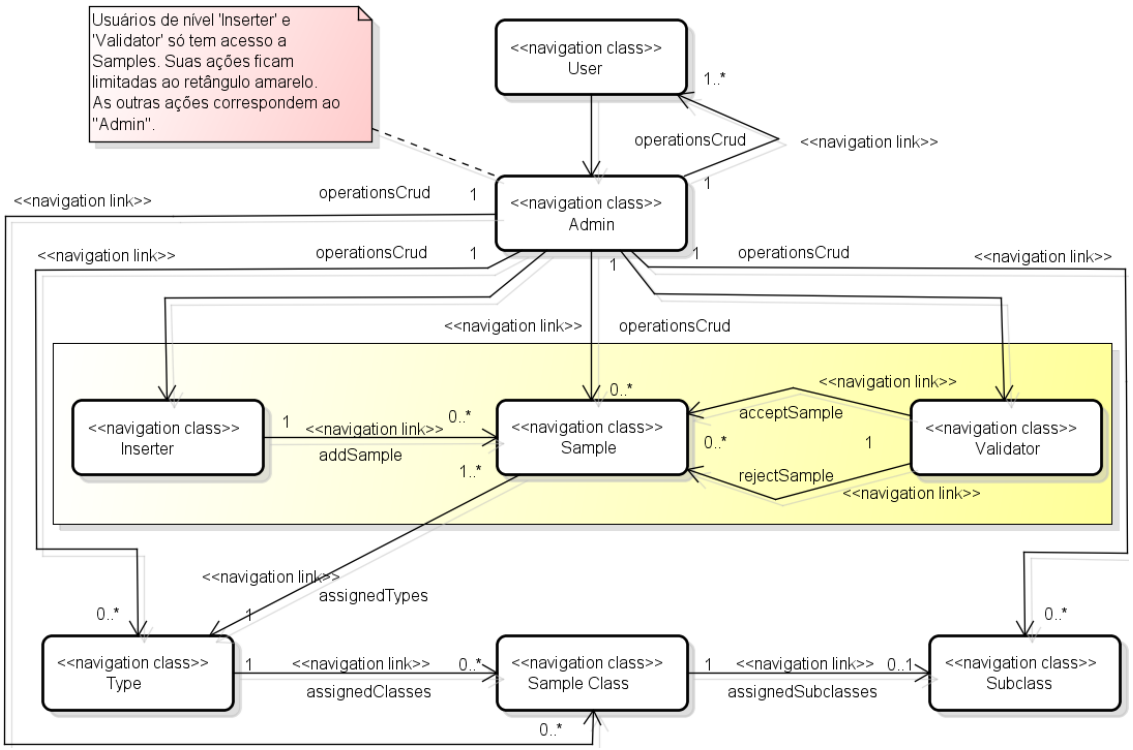


Figura 3.4 - Diagrama de navegação

Fonte: diagrama criado pelo autor.

3.2.6.2 Diagrama de Acesso

Visto que o diagrama de navegação só leva em consideração as classes, utiliza-se um diagrama mais completo que indica uma sequência de funcionamento do sistema. Para isso, o diagrama de acesso permite a utilização de outras *flags* que explicitem as ações do usuário. Utilizar-se-á <<*query*>> para indicar, por exemplo, que o usuário está navegando em um nodo em que ele deve realizar uma busca no sistema através de entrada de dados, <<*menu*>> significando a escolha de uma opção da página e <<*index*>> para uma listagem de objetos, como usuários cadastrados ou amostras.

Existem outros tipos de *flags*, mas não serão utilizados nem explicados neste trabalho.

Como esse é um diagrama grande, será feita apenas modelagem como usuário Administrador, para evitar excesso de nodos. Como se pode ser visto na Figura 3.5, o Administrador pode acessar qualquer nodo de qualquer lugar.

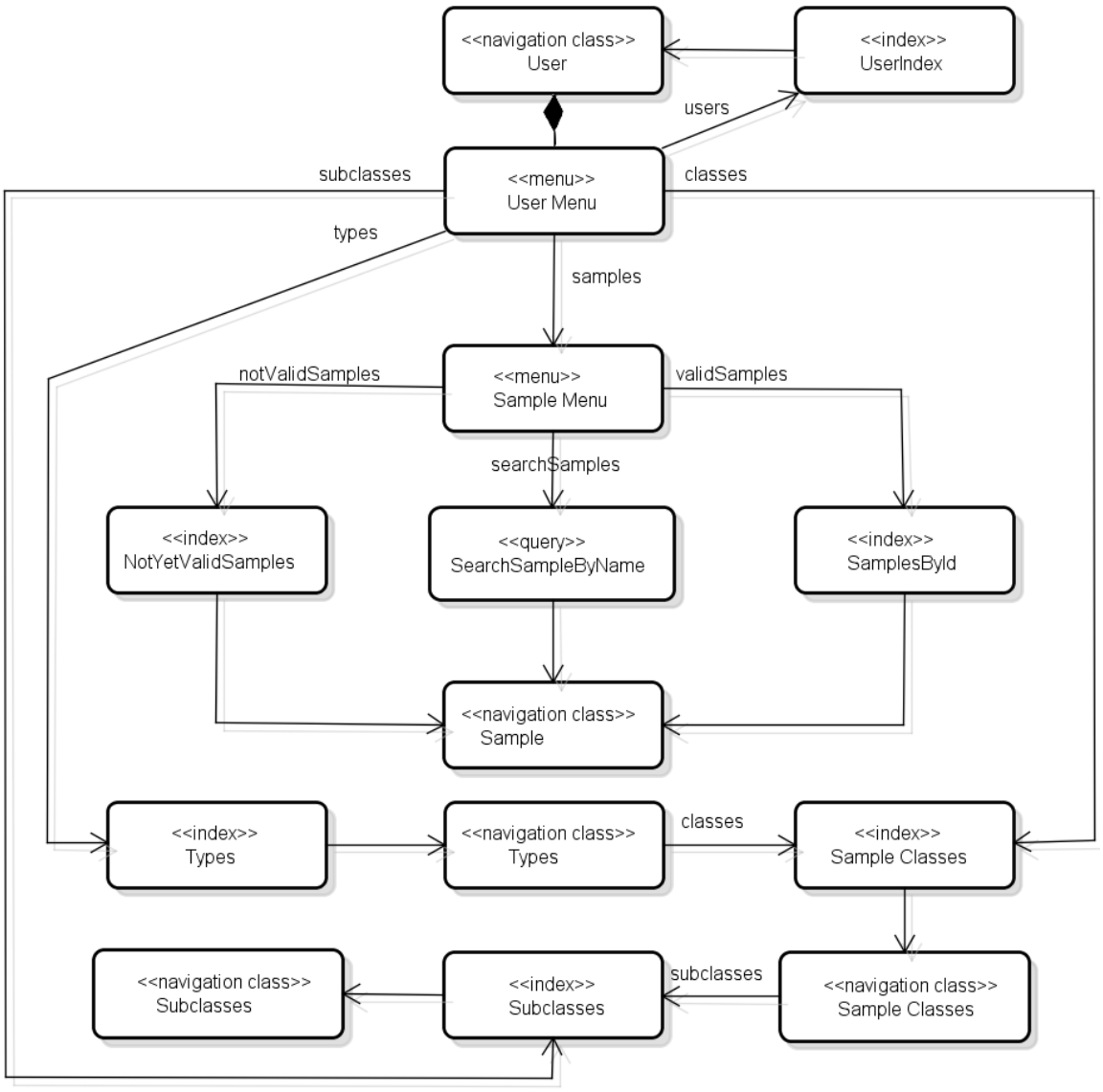


Figura 3.5 - Diagrama de acesso

Fonte: diagrama criado pelo autor.

3.2.7 Diagrama de Apresentação

O diagrama de apresentação está relacionado com a interface do sistema, focando na representação da estrutura e o seu comportamento. É um modelo da composição da tela do sistema vista pelo usuário, exibindo os elementos estáticos, como cabeçalhos e rodapés, e os elementos variáveis para cada página, podendo ser estes os botões, campos, imagens e textos, entre outros. Sua outra função é descrever os aspectos orientados pelo comportamento da página, como qual botão se pressionar para se obter certa ação (KAPPEL; PRÖLL; REICH; RETSCHTZEGGER, 2003).

Utilizando a abordagem do UWE, flags são usadas para separar o conteúdo, como *<<page>>*, que representa toda a estrutura da página, e *<<presentation unit>>*, que engloba o conteúdo presente na tela e agrupa elementos de apresentação. Os elementos básicos são:

- *<<text>>* representa um conteúdo padrão, contendo texto e imagem;
- *<<button>>* é uma ação, um *submit* de alguma função;
- *<<anchor>>* é um hyperlink que realiza a navegação entre as páginas, levando para um conteúdo específico de um nodo;
- *<<anchor collection>>* é um hyperlink que leva para um conjunto de índices, como, por exemplo, a listagem das amostras do sistema.

Como no capítulo seguinte deste trabalho serão exibidas as guias das telas do sistema, aqui será exibido apenas um diagrama de apresentação, representado na Figura 3.6, que é referente à tela de página inicial do sistema, visto pelo usuário com acesso de validador/cadastrador.

Com exceção do diagrama de entidade-relacionamento, que foi criado utilizando o MySQL Workbench, todos os diagramas foram criados utilizando a ferramenta CASE Astah Community (http://astah.net/editions/community).

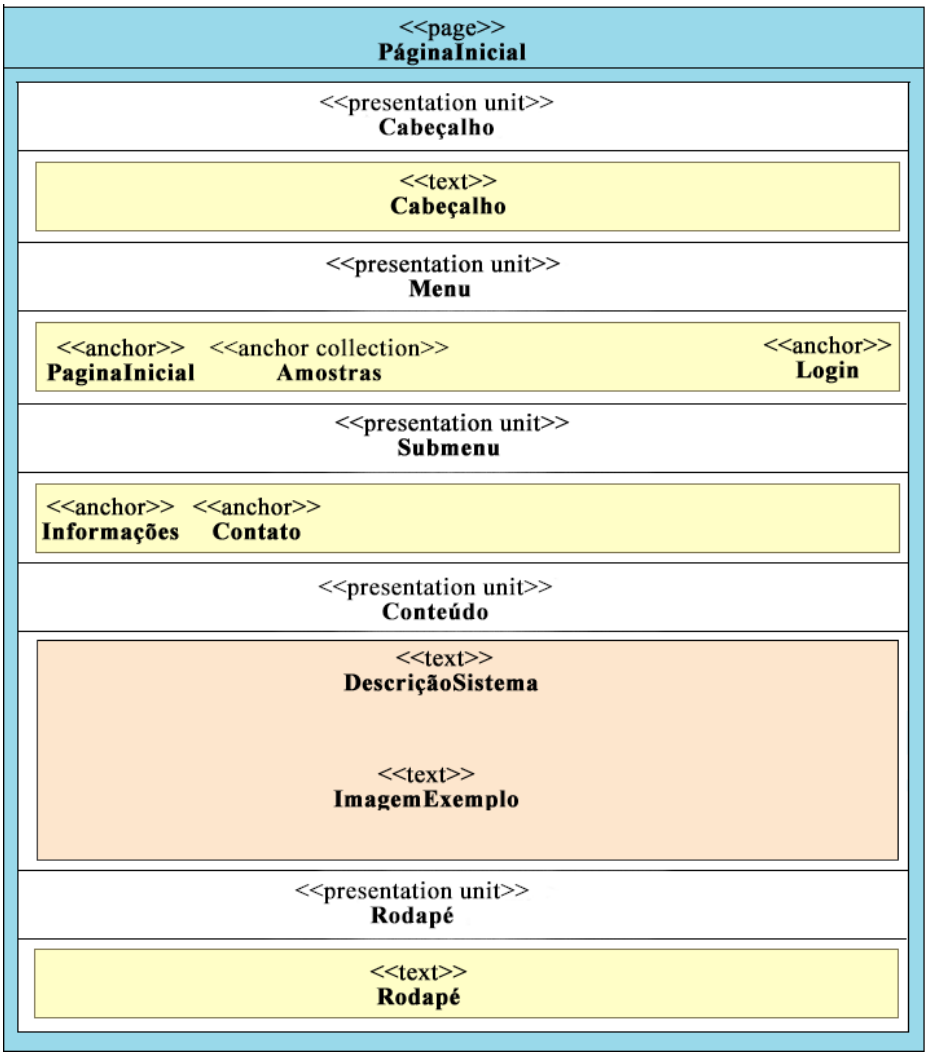


Figura 3.6 - Diagrama de apresentação

Fonte: diagrama criado pelo autor

3.3 Desenvolvimento do sistema

O desenvolvimento do sistema foi feito principalmente com as linguagens HTML, PHP e JavaScript, com o auxílio do Framework CakePHP (http://cakephp.org/), para criar o ambiente de interação via navegador web com o usuário. Adicionalmente, foiutilizada uma biblioteca chamada HighCharts (http://www.highcharts.com/) para realizar a geração de gráficos que possuem e exibem informações das amostras. Essa biblioteca utiliza as linguagens JavaScript, jQuery e recursos de HTML 5.

O banco de dados escolhido para armazenar as informações foi o MySQL, por ser gratuito, assim como as outras tecnologias escolhidas pelo sistema, e por possuir uma ótima integração com a linguagem PHP.

3.4 Tecnologias e Arquitetura do Sistema

"Uma aplicação web é um sistema de software baseado em tecnologias e padrões da World Wide Web Consortium (W3C) que provém recursos específicos de web tal como

conteúdo e serviços através de uma interface de usuário, o navegador web." (KAPPEL; PRÖLL; REICH; RETSCHTZEGGER, 2003).

Vale salientar que cada tecnologia separadamente, como a linguagem HTML ou a linguagem PHP, não é considerada uma aplicação web, apenas ferramentas para se alcançar uma aplicação. Algumas dessas ferramentas serão descritas abaixo.

3.4.1 Framework CakePHP

Um Framework de aplicação web é um conjunto de códigos fontes organizado em uma arquitetura que visa facilitar e acelerar o desenvolvimento de aplicações web. Os frameworks incentivam a prática de boas técnicas de programação, como por exemplo, a reusabilidade, além de sugerirem a padronização do código e o respeito por suas regras de desenvolvimento e nomenclaturas (POREBSKI; PRZYSTALSKI; NOWAK, 2011).

O framework utilizado nesse sistema foi o CakePHP, na versão 2.1 (http://www.cakephp.org), que adota essa postura citada no parágrafo anterior de incentivar práticas de boa programação, com o objetivo de padronizar o desenvolvimento do código, de modo que facilite o entendimento, revisão e edição desse código, não só pelo seu criador original, como por outros desenvolvedores.

A escolha desse framework se deu pelo fato de ele possuir uma comunidade de usuários considerada grande e ativa, o que é bastante importante, pois geralmente os frameworks de aplicação web não possuem uma documentação oficial muito completa, além de versões estáveis e de ser licenciado pelo *MIT License*, que é uma permissão para software livres que consente ao usuário o direito de utilização total do software, incluindo a sua utilização com fins comerciais.

A utilização o framework é simples, basta fazer o seu download, que é um arquivo compactado, descompactar esse arquivo no servidor, ou pasta local (localhost), renomear para o nome do sistema e acessá-lo através de endereço no navegador, por exemplo, *localhost/NomeDoSistema*. A única configuração posterior que ele exige, é a criação de um banco de dados, diferente do *default* que vem com o sistema, e a alteração dos padrões *salt e hash*, sequências aleatórias de bits utilizadas para criptografar informações, caso o desenvolvedor opte por utilizar suas funções de autenticação e login.

Ao longo do trabalho serão salientadas algumas características do framework, como imposição de regras de utilização, ou facilidades que ele provê. A seguir é descrita sua arquitetura de funcionamento.

3.4.2 Modelo MVC

Aplicações desenvolvidas com Model-View-Controller (MVC) são fatoradas em três camadas, onde a primeira se refere ao domínio das informações e é denominada modelo, a segunda, à exibição visual do estado da aplicação, chamada de visão, e a última, o controlador, trata da interação entre as duas primeiras (KRASNER; POPE, 1988). O objetivo é separar a lógica de funcionamento de negócio da interação com o usuário, diminuindo a complexidade do sistema e a quantidade de dados disponíveis ao usuário (REENSKAUG, 2007), assim aumentando a proteção dos dados e isolamento de erros. Na Figura 3.7 é possível visualizar o fluxo de funcionamento do MVC.

O *model* (modelo) trata das informações do sistema. Gerencia a lógica de negócio e obtém e trata as informações requisitadas pelo usuário a partir de uma conexão com o banco de dados.

A *view* (visão) exibe os dados que foram requeridos do modelo para o usuário ao criar uma interface de apresentação. Os modelos podem possuir mais de uma visão, e cada uma dessas visões tem que ser capaz de representar de pelo menos um modo o seu modelo associado.

O *controller* (controlador) é a interface entre o modelo, a visão e o dispositivo de entrada, como mouse ou teclado. Ele coordena o fluxo da informação ao requerê-la ao modelo e transmití-la à visão.

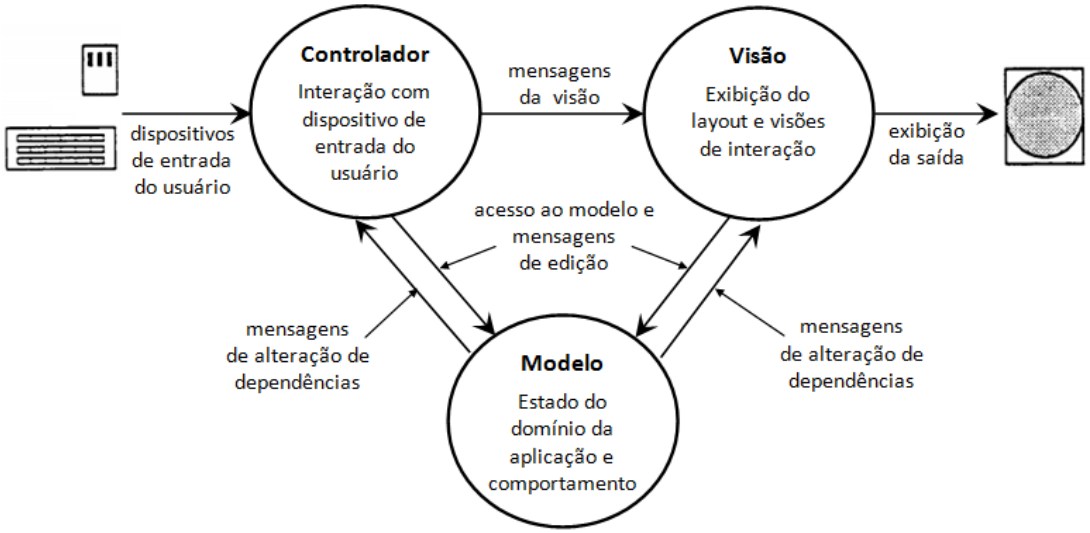


Figura 3.7 - MVC e envio de mensagens

Fonte: imagem traduzida e adaptada de Krasner e Pope (1988)

O Framework CakePHP trabalha com mais uma camada, o *Dispatcher*, que tem como função selecionar o controlador correto para cada requisição.

A Figura 3.8 seguinte exemplifica o funcionamento normal do MVC dentro do CakePHP. O *client* faz a requisição e o *dispatcher* escolhe o *controller* correto para o pedido, e esse comunica ao *model* os dados a serem processados. Com o resultado, o *controller* delega ao *view* correto a visualização dos dados tratados para o *client*.

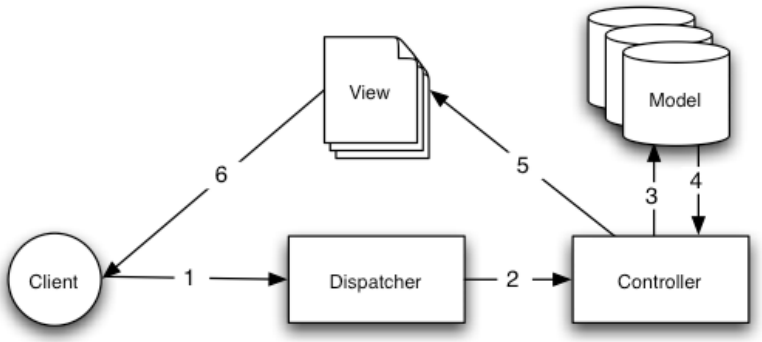


Figura 3.8 - MVC request dentro do CakePHP

Fonte: Cake Software Foundation (2012).

Para deixar claro o que é cada uma das camadas descritas, abaixo será transcrito o código de desenvolvimento do programa do modelo, controlador e de uma das visões da classe usuário (*User*). Cabe salientar que cada uma das classes descritas no diagrama de conteúdo possuirá um modelo, uma visão e um controlador. A estrutura de acesso às visões é definida no modelo de navegação já descrito, definindo, portanto, a arquitetura de acesso ao conteúdo da aplicação Web.

3.4.2.1 Modelo

No código abaixo pode-se ver que a classe *User* define o nome da classe para o sistema, e, logo abaixo, trata das regras do banco de dados. O diagrama contendo a definição do banco de dados será explicado posteriormente. No momento o importante é perceber que o código define quais campos são obrigatórios e suas possíveis mensagens de erro. Várias outras definições são possíveis, como explicitar que um campo é alfanumérico, apenas numérico, um e-mail, como é visto no campo *email* abaixo, entre outras.

Existe uma função dentro da definição do modelo, chamada *beforeSave*(), que já vem definida pelo CakePHP, que serve para encriptar a senha do usuário antes de salvá-la no banco de dados, por isso o nome *beforeSave*(). Essa função só pode ser utilizada se o desenvolvimento do código respeitar algumas regras impostas pelo CakePHP, que nesse caso seriam, o modelo ser chamado *User*, a tabela no banco de dados ser chamada *users* e os campos com os nomes de usuário e senha dessa tabela serem definidos como *username* e *password*, respectivamente. De outro modo, o framework acusa erro, sendo necessário que o desenvolvedor crie sua própria função se quiser utilizar outros nomes. Essa imposição é justificada com a intenção de manter um padrão de desenvolvimento de código.

```
<?php
App::uses('AuthComponent', 'Controller/Component');
class User extends Model {
/* O Framework faz em todas classes essa extensão de Model*/

  public $name = 'User';

  public $validate = array(
   'username' => array(
      'required' => array(
        'rule' => array('notEmpty'),
        'message' => "Login é obrigatório."
      )
    ),
    'password' => array(
      'required' => array(
        'rule' => array('notEmpty'),
        'message' => "Senha é obrigatória."
      )
    ),
    'name' => array(
      'required' => array(
        'rule' => array('notEmpty'),
        'message' => "Nome é obrigatório."
      )
    ),
    'email' => array(
```

```
        'required' => array(
          'rule' => array('email', true),
          'message' => "E-mail é obrigatória."
        )
      ),
      'role' => array(
      'required' => array(
        'rule' => array('notEmpty'),
        'message' => "Função do usuário é obrigatória."
      )
      ),
      'active' => array(
      'required' => array(
      'rule' => array('notEmpty'),
      'message' => "Atividade do usuário é obrigatória."
      )
      )
    );

    public function beforeSave($options = array()) {
      if (isset($this->data[$this->alias]['password'])) {
          $this->data[$this->alias]['password'] =
             AuthComponent::password($this->data[$this->alias]['password']);
      }
      return true;
    }
  } ?>
```

3.4.2.2 Controlador

O código abaixo está incompleto, pois não é necessário ver todas as funções para compreender o funcionamento do controlador. Para cada uma dessas funções, existe um arquivo correspondente de mesmo nome, com a extensão *ctp* (*cake template*), que representa a sua visão, exceto *logout*(), visto que não é necessário exibir nenhuma informação ou campo para o usuário, apenas redirecioná-lo para algum nodo do sistema.

As funções *login*() e *logout*() têm o código disponibilizado pelo CakePHP e são prontas para uso, porém, o framework permite sua edição. Um pequeno exemplo é no caso da primeira função, onde foi adicionado um campo de sessão que gravasse o papel que o usuário que está fazendo o login possui, podendo este ser, administrador, visitante ou cadastrador.

Toda parte de autenticação de usuário já vem elaborada pelo framework, e se forem respeitadas as duas regras, é de fácil utilização. Como se pode ver toda validação de login e logout é feita pelo componente *Auth*, que trata da verificação se um usuário está cadastrado e de sua permissão de acesso. Na função *add()*, que foi totalmente criado pelo autor do trabalho, cuja função é a de adicionar novos usuários ao sistema, usa-se o *Auth* para testar se o usuário é um administrador e permitir ou não a adição de novos usuários.

```
<?php
class UsersController extends AppController {

  public $helpers = array('Html', 'Form');
  public $name = 'Users';
  public $components = array('Session');
```

```
  public function login() {
    if ($this->Auth->login()) {
      $this->Session->write('usuRole', $this->Auth->user('role'));
      $this->redirect($this->Auth->redirect());
    } else {
      $this->Session->setFlash('Nome de usuário ou senha incorretos. Por
favor, tente novamente.', 'flash_erro');
    }
  }

  public function logout() {
    if ($this->Session->valid()) {
      $this->Session->destroy(); // Destrói
    }
    $this->redirect($this->Auth->logout());
  }
...   public function add() {
    if ($this->Auth->user('role') === 'admin') {
      if ($this->request->is('post')) {
        $this->User->create();
        if ($this->User->save($this->request->data)) {
          $this->Session->setFlash('Usuário salvo com sucesso.',
'flash_sucesso');
          $this->redirect(array('action' => 'index'));
        } else {
          $this->Session->setFlash('Não foi possível salvar o usuário. Por
favor, tente novamente.', 'flash_erro');
        }
      }
    } else {
      $this->Session->setFlash('Usuário sem permissão para adicionar novos
usuários. Por favor, entre em contato com o administrador do
sistema.','flash_erro');
    }
  }
...
}?>
```

3.4.2.3 Visão

Nas visões, além das informações pertinentes, são exibidos os menus com o qual o usuário vai interagir para chegar aos nodos desejados. Como é visível no código abaixo, para cada hyperlink criado, é passado o controlador correspondente àquele nodo, e qual ação ele deve tomar, ou seja, qual visão ele deve carregar.

A primeira parte do código é o menu geral, que nesse caso conta com *Página Inicial*, *Amostras*, *Usuários*, *Tipos*, *Classes*, *Subclasses* e *Logout*. O menu é diferente para cada usuário com nível de acesso diferente e também para usuário logados ou não. No caso do menu da visão de criação de usuários não é necessário fazer nenhum teste sobre quais informações exibir no menu, pois foi passado para o componente *Auth()* que apenas usuários de nível administrador podem acessar essa página do sistema.

A segunda parte do código é o submenu, que tem as opções referentes ao grande grupo selecionado no menu geral, e, por fim, na terceira parte do código é onde se monta o formulário, que possui todos os campos vistos no modelo. Esse padrão de escrita de código visto na terceira parte é utilizado nos arquivos de extensão *.ctp*, sendo

então um estilo de escrita de código do CakePHP. Porém, se montarmos um formulário com escrita padrão, utilizando *tags* da linguagem HTML, a visão funcionará perfeitamente. O framework impõe algumas regras, mas também dá certa liberdade para o desenvolvedor.

Na questão de ajustar o design, fica um pouco mais complexo utilizar o modo sugerido pelo CakePHP, pois quando há muitos elementos, divisões e efeitos de visualização, fica mais fácil programar e compreender *tags* indentadas com abertura e fechamento do que objetos com múltiplos *arrays* de definições. Porém, de qualquer forma, no trabalho foi seguido o modo sugerido pelo framework, para padronizar ao máximo o código.

```
<?php
/* Menu Navegação */
$this->start('menu_nav');
$list = array(
    $this->Html->link('Página Inicial', array('controller' => 'samples',
'action' => 'index')),
    $this->Html->link('Amostras', array('controller' => 'samples', 'action'
=> 'index')),
    $this->Html->link('Usuários', array('controller' => 'users', 'action' =>
'index'), array('class' => 'current_page_item')),
    $this->Html->link('Tipos', array('controller' => 'types', 'action' =>
'index')),
    $this->Html->link('Classes', array('controller' => 'sampleclasses',
'action' => 'index')),
    $this->Html->link('Subclasses', array('controller' => 'subclasses',
'action' => 'index')),
    $this->Html->link('Logout', array('controller' => 'users', 'action' =>
'logout'), array('style' => 'float:right'))
);

echo $this->Html->nestedList($list);
$this->end();

/* Menu Sub Navegacional, que serve de BreadCrumb, ou seja, 'caminho atual'*/

$this->start('menu_sub_nav');

$this->Html->addCrumb('Listar', array('controller' => 'users', 'action' =>
'index'));
$this->Html->addCrumb('Adicionar', array('controller' => 'users', 'action' =>
'add'), array('class' => 'current_page_item'));

$this->end();
?>

/* Formulário */

<fieldset>
  <div class="legend"><h3>Adicionar Usuário</h3></div>
<?php
echo $this->Form->create('User', array('action' => 'edit'));
echo $this->Form->input('username',
      array('label' =>
```

```
            array('text' => 'Nome de Usuário', 'class' => 'form_property
                  form_required'
)));
echo $this->Form->input('password',
      array('label' =>
           array('text' => 'Senha', 'class' => 'form_property form_required'
)));
echo $this->Form->input('name',
      array('label' =>
           array('text' => 'Nome', 'class' => 'form_property form_required'
)));
echo $this->Form->input('email',
      array('label' =>
          array('text' => 'Email', 'class' => 'form_property form_required'
)));
$role = array('valid' => 'Validador', 'cad' => 'Cadastrador');
echo $this->Form->input('role',
        array('options' => $role, 'label' =>
          array('text' => 'Função', 'class' => 'form_property form_required'
)));

$active = array('0' => 'Sim', '1' => 'Nao');
echo $this->Form->input('active',
        array('options' => $active, 'label' =>
           array('text' => 'Ativo', 'class' => 'form_property form_required'
)));
echo $this->Form->end('Adicionar Usuário');

?>
   </fieldset>
```

3.4.3 Biblioteca HighCharts

É uma biblioteca para criação de gráficos, que podem ter funções interativas, como a aplicação de *zoom*, e são apresentados em vários formatos, como gráfico em linha, em pizza, por área entre outros. Foi escolhida por ser totalmente desenvolvida com JavaScript e JQuery, tecnologias que funcionam facilmente com PHP, e por possuir uma licença que permite o uso gratuito de todos os recursos quando estes forem voltados para educação e realizados por instituições de ensino. Gráficos simples são fáceis de realizar, porém para atividades mais complexas foram encontradas algumas dificuldades. Mesmo com essas dificuldades, o uso da biblioteca é justificado, pois outras bibliotecas procuradas não satisfaziam as necessidades do projeto, seja por não serem gratuitas, ou por não possuírem a possibilidade de interação.

Sua incorporação no código é bem simples, só sendo necessário adicionar três linhas de código na visão desejada, conforme o exemplo seguinte.

```
<script type="text/javascript" src="../../app/webroot/js/jquery.js"></script>
<script type="text/javascript" src="../../app/webroot/js/highcharts.js">
</script>
<script type="text/javascript" src="../../app/webroot/js/NomeDoArquivo.js">
</script>
```

Elas servem para permitir a utilização dos arquivos *jquery.js*, que é uma biblioteca de JavaScript com várias funções, *highcharts.js*, que é a biblioteca HighCharts, e o último, chamado genericamente de *NomeDoArquivo.js* (para facilitar a explicação), que é o arquivo contendo os dados que o usuário passa como parâmetro para as funções que

geram os gráficos. É possível adicionar outros arquivos da biblioteca HighCharts, como por exemplo, *exporting.js*, que permite ao usuário exportar o gráfico no formato de uma imagem.

A geração de um gráfico com muitas informações não é tão simples, pois o HighCharts oferece muitos exemplos de utilização, e pouca documentação. O problema reside no fato de que cada exemplo é desenvolvido de um modo, utilizando funções diferentes do HighCharts, e geralmente com um dos eixos do gráfico com valores fixos, no sentido de que não é possível atribuir variações no valor do eixo, apenas escolher um intervalo. No caso do projeto, os dois eixos possuem valores variáveis e decimais, e para conseguir alcançar o resultado desejado, foi necessário estudar e experimentar vários exemplos.

Utilizando os dados de um arquivo de texto igual ao demonstrado na seção 2.1, que é resultante da técnica de espectrometria, também explicada na seção 2.1, obtém-se um gráfico como o da Figura 3.9.

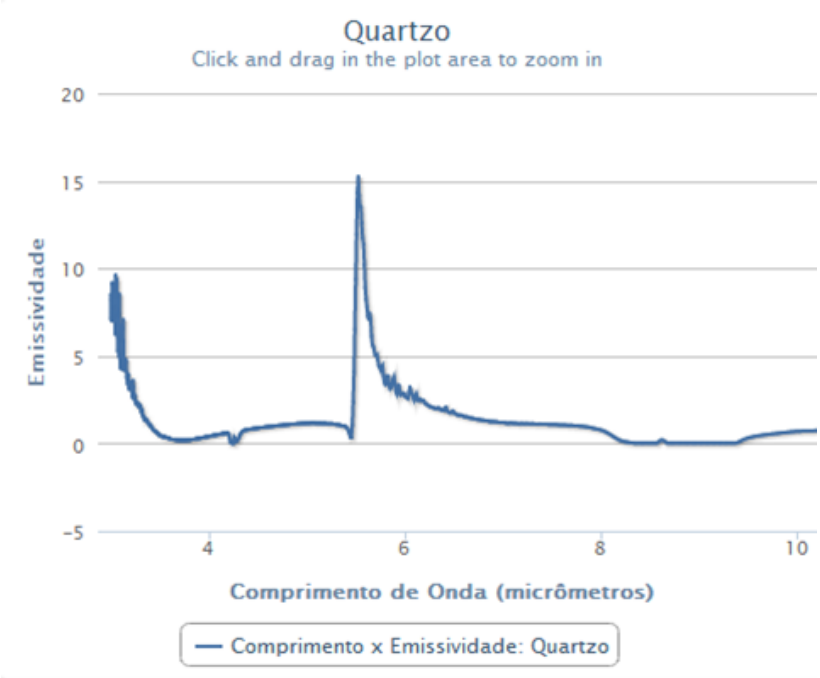


Figura 3.9 - Exemplo de gráfico do sistema

Fonte: gráfico gerado pelo autor utilizando o sistema

Com base na modelagem descrita nesta seção, no modelo MVC e nas tecnologias apresentadas, a aplicação foi desenvolvida. O resultado encontra-se na próxima seção, onde são descritas as visões (telas) do sistema e o seu funcionamento.

4 VISÕES DO SISTEMA

São apresentadas as principais visões do sistema, com explicação de sua funcionalidade. O design foi adaptado de um *template*, desenvolvido por Arcsin Web Templates (http://templates.arcsin.se/), de uso ilimitado, desde que seja inserido no rodapé de cada página um link que direcione para a página do desenvolvedor.

Buscou-se um design que possuísse menus de navegação em dois níveis, com a intenção de se ter o percurso navegacional (*BreadCrumbs*) do usuário, de modo que ele possa identificar em qual nível da página está navegando.

4.1 Visão da Página Inicial

Todos os usuários tem acesso à página inicial, mas será exibida na Figura 4.1 a página inicial do modo como é vista por um usuário de nível de acesso de validador/cadastrador, para haver a comparação com o diagrama de apresentação. Para facilitar essa comparação, foi destacado na imagem onde se encontra cada *<<presentation unit>>*, seguindo o formato de apresentação proposto na seção 3.2, no capítulo anterior (Figura 3.6). É uma página apenas informativa, sendo a única interação as possíveis trocas de nodos.

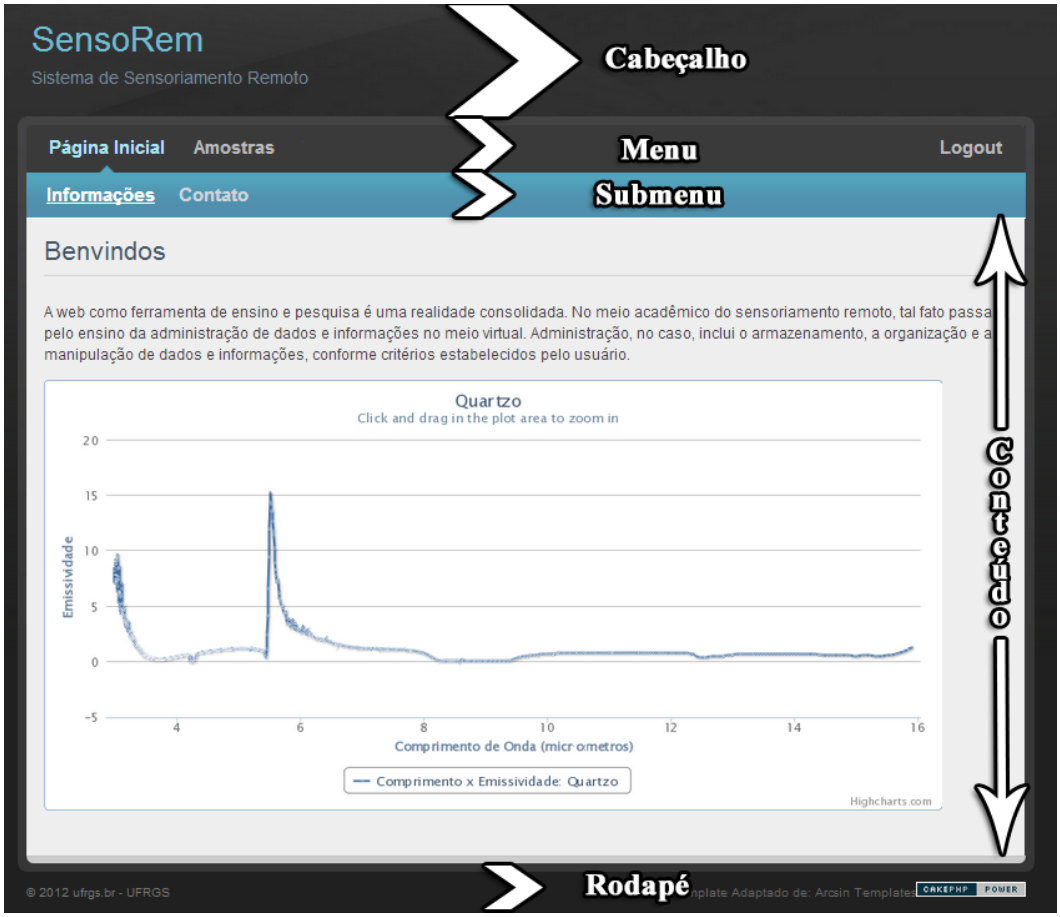


Figura 4.1 - Visão da página inicial

Fonte: imagem salva pelo autor utilizando o sistema

4.2 Visão do Login

A seguir encontra-se a visão da tela de login do usuário, já exibindo uma mensagem de erro caso o login não funcione. Caso haja sucesso, o usuário é redirecionado para a página de listagem de amostras. Como pode ser visto na figura 4.2, no menu, bem à direita, existe a opção “Login”, quando o usuário já está logado, essa opção vira “Logout”, para o usuário poder se desconectar do sistema.

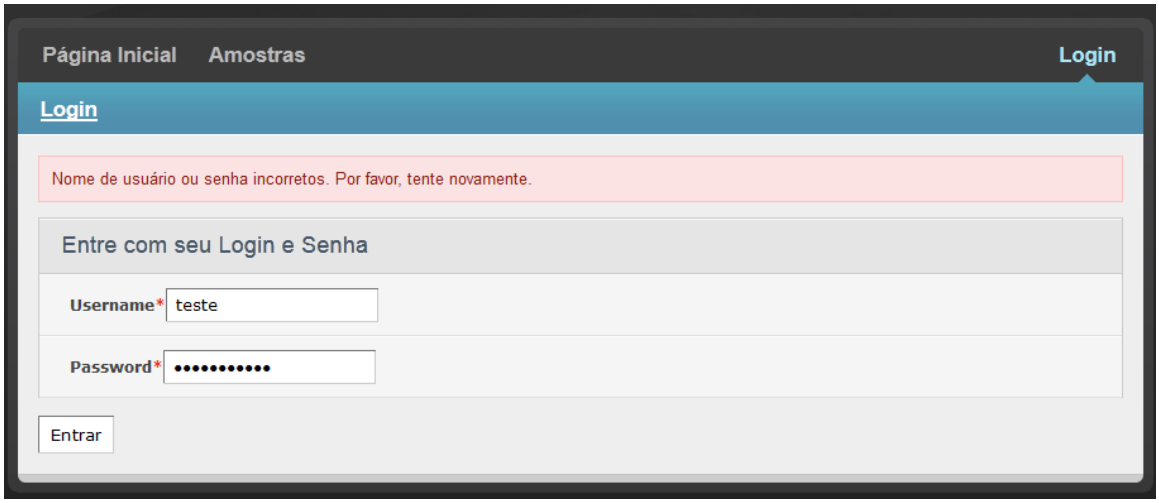


Figura 4.2 - Visão do login

Fonte: imagem salva pelo autor utilizando o sistema

4.3 Visão da Lista de Amostras

A partir de agora serão exibidas as visões como usuário Administrador, para mostrar o sistema de um modo mais completo.

A listagem de amostras permite visualizar as amostras já cadastradas e fazer uma busca simples por nome. Pretende-se adicionar no sistema uma busca mais completa, por qualquer tipo de campo, que será explicada no capítulo seguinte.

Clicando sobre o nome da amostra, abre-se a visão que exibe detalhes dessa amostra, como visto na figura 4.3. Na última coluna da tabela, à direita, existe a opção de e*ditar* a amostra, que redireciona para um formulário em que se é possível alterar os dados da amostra. Por fim temos a opção *apagar*, que apenas exibe um pop-up pedindo a confirmação do usuário. Essas ações de edição e exclusão só podem ser feitas pelo administrador ou pelo usuário que criou a amostra.

Figura 4.3 - Visão da listra de amostras

Fonte: imagem salva pelo autor utilizando o sistema

4.4 Visão de Confirmação de Exclusão

A exclusão não possui uma visão específica, ela apenas tira o foco da visão que lista as amostras, e exibe um pop-up pedindo a confirmação da exclusão, conforme visto na figura 4.4. Todas as opções de exclusão no sistema funcionam desse modo, com um pop-up de confirmação, sem a adição de uma visão específica.

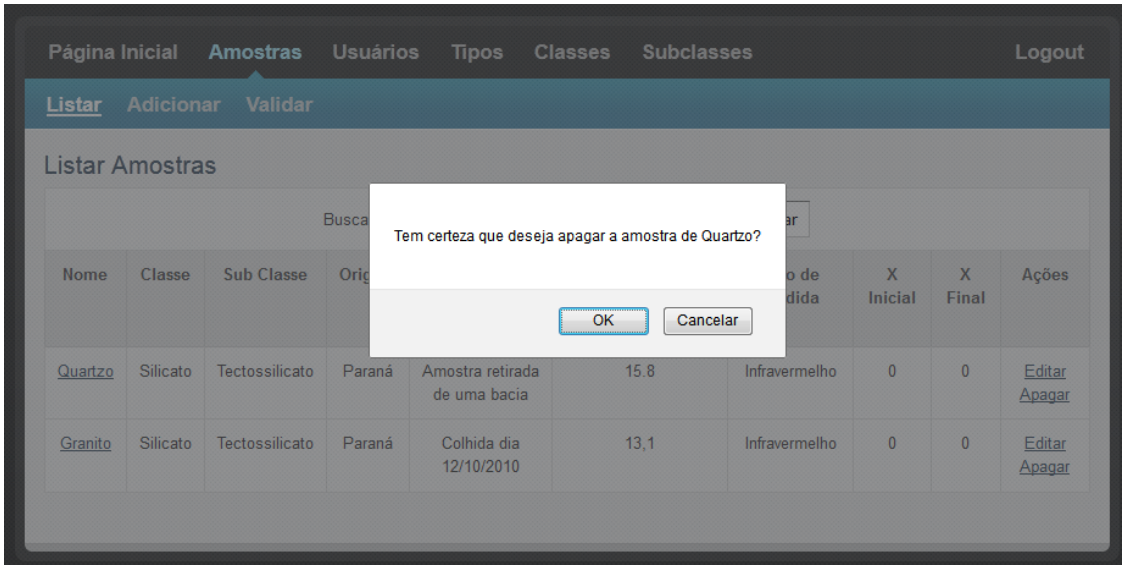


Figura 4.4 - Visão de confirmação de exclusão

Fonte: imagem salva pelo autor utilizando o sistema

4.5 Visão de uma Amostra em Detalhes

A visualização em detalhes mostra todos os campos da amostra, e também as opções de exclusão e edição presentes na listagem completa de amostras.

O gráfico criado a partir dos dados fica visível e é possível a aplicação de *zoom*, para visualizar partes do gráfico com mais detalhes. Além disso, pode-se exportá-lo no formato de uma imagem ou imprimi-lo. Adicionalmente ao *zoom* pretende-se adicionar outras manipulações ao gráfico, como limitar os valores inicial e final do eixo x, a possibilidade de visualização conjunta de outra assinatura espectral de outra amostra e poder alternar entre gráfico de comprimento de onda por reflectância e comprimento de onda por emissividade. Como visto na figura 4.5 e na figura 4.6, há uma legenda explicitando sobre o que se refere aquela curva, no caso Comprimento de Onda x Emissividade. Isso será muito útil quando for possível a geração de mais de uma assinatura espectral por gráfico.

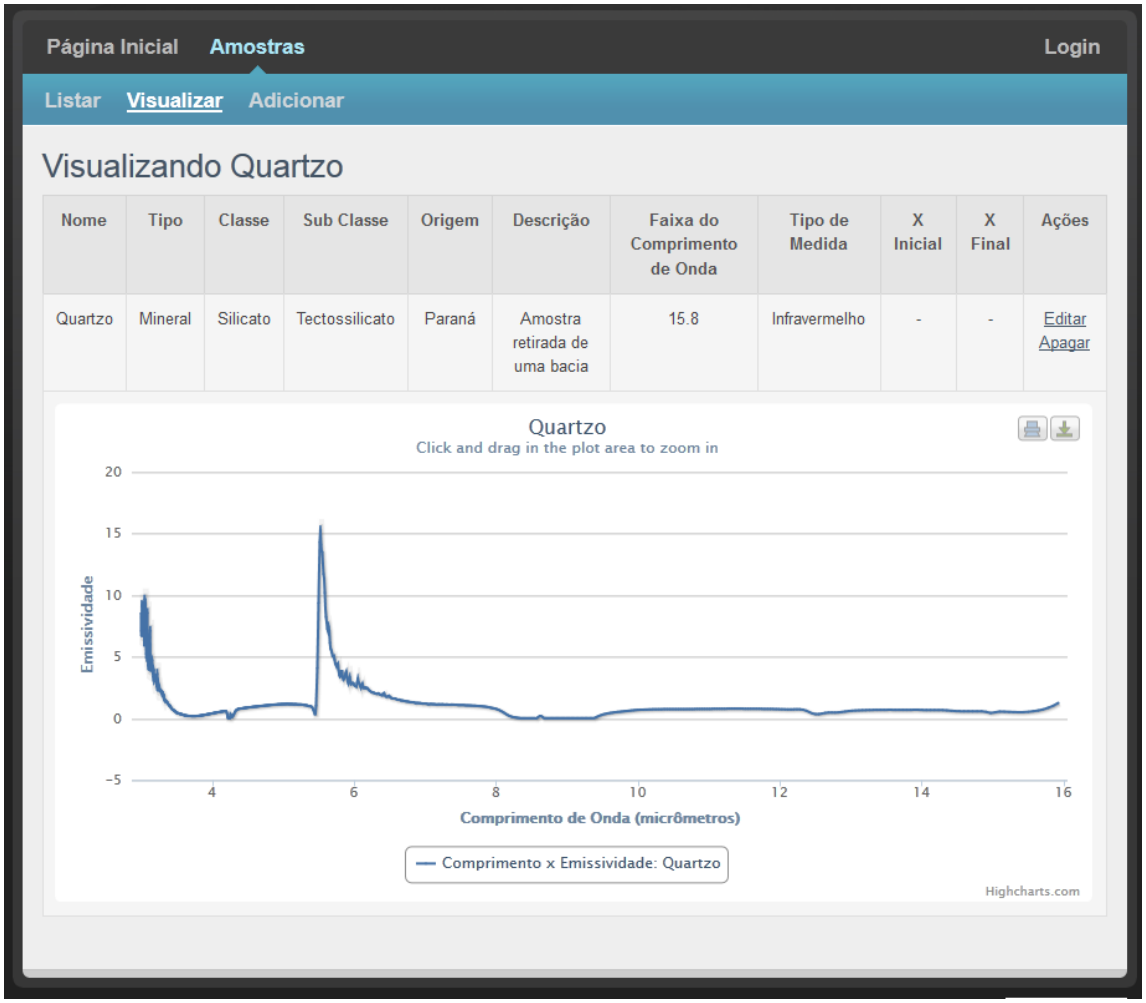


Figura 4.5 - Visão de uma Amostra em Detalhes

Fonte: imagem salva pelo autor utilizando o sistema

Figura 4.6 - Gráfico com zoom

Fonte: imagem salva pelo autor utilizando o sistema

4.6 Visão da Adição de uma Amostra

Essa é a visão de um formulário padrão do sistema. Os campos em negrito e com asteriscos ao lado são obrigatórios, enquanto os outros são opcionais. É possível entender os campos do formulário visualizando a tabela A.3, no apêndice A, que contém o detalhamento dos campos de uma amostra.

É possível ver na figura 4.7 que os campos “classe” e “subclasse” são opcionais, pois nem todo tipo de amostra contém classes e subclasses, então no modelo da amostra é preciso defini-los como campos opcionais.

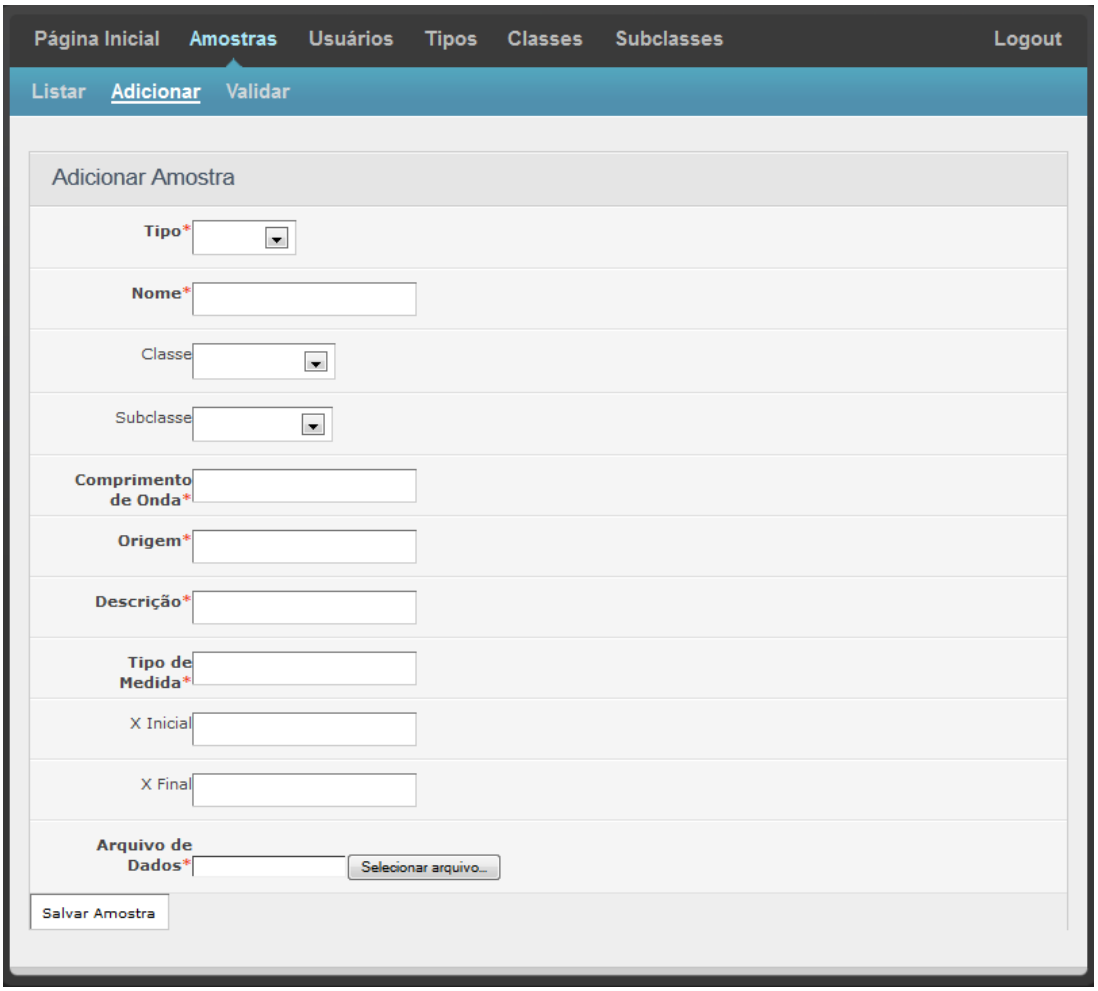


Figura 4.7 - Visão da adição de uma amostra

Fonte: imagem salva pelo autor utilizando o sistema

Os outros formulários, de adição/edição de usuário, tipo, classe e subclasse não serem exibidos por serem semelhantes, apenas mudando os seus campos de entrada.

4.7 Visão da Validação de uma Amostra

A validação exibe uma lista das amostras a serem validadas, como no caso das amostras já inseridas no sistema. Para validar uma amostra, é necessário selecionar uma, visualizar suas informações em detalhes, como visto na visão de amostras em detalhes, e, então, escolher validar, editar ou excluir esta amostra. Como é uma visão de apresentação muito semelhante às outras, não será adicionada uma imagem dessa tela.

5 CONCLUSÃO

A necessidade de se armazenar e analisar dados gerados através da técnica de espectrometria motivou a criação deste trabalho, que visou desenvolver um sistema web que tornasse possível a visualização com detalhes do resultado destas análises, assim como criasse um repositório digital que disponibilizasse online as informações para o público em geral.

O desenvolvimento começou após a análise dos requisitos do sistema, que foram definidos através de reuniões e discussões com as partes interessadas na criação deste. Com essa definição bem encaminhada, começou-se a modelagem do sistema e o estudo de tecnologias e metodologias ainda não conhecidas pelo autor. O estudo de metodologia de desenvolvimento voltado para web e a utilização de um framework de desenvolvimento, para facilitar a manutenção posterior do sistema, foram necessários e demandaram um tempo inicial.

Algumas reuniões posteriores foram realizadas para sanar dúvidas e mostrar o estado atual do sistema. Essas foram bastante importantes, pois foi necessário realizar algumas mudanças na estrutura do banco de dados do sistema, e como foram apresentadas ao longo do desenvolvimento, não foi necessário alterar todo um sistema pronto.

5.1 Resultados

Através do estudo dos requisitos funcionais bem definidos, das constantes reuniões e do estudo do sistema da NASA de semelhante funcionalidade, foi possível atingir vários objetivos traçados do trabalho com relativa facilidade, sendo o principal a exibição das informações das amostras já com a possibilidade de aplicação de *zoom* nos gráficos que representam suas assinaturas espectrais.

5.2 Limitações

Com o tempo demandado para aprender as novas tecnologias e com a metodologia de implementar o projeto após toda a modelagem, não foi possível alcançar todos objetivos traçados. O grande limitador foi a biblioteca gráfica, cujo uso básico é bem simples, porém para realizar tarefas mais completas, é complicado devido à pouca documentação.

5.3 Trabalhos Futuros

Em relação aos objetivos iniciais traçados, faltou incluir uma busca mais detalhada para as amostras, com a opção de utilizar filtros para direcionar melhor a busca.

Atualmente o sistema só conta com uma busca por nome. Os outros objetivos não alcançados são vistos na parte de manipulação do gráfico de dados das amostras. Deseja-se visualizar mais de um gráfico ao mesmo tempo para realizar comparações de amostra, assim como importar e exportar alterações realizadas nos gráficos, de modo que o usuário possa guardar localmente essas mudanças sem alterar os dados originais do sistema. Também se deseja permitir indicar os valores inicial e final do eixo x, respeitando os valores obtidos no arquivo de texto da amostra, limitando a visualização à área de interesse do usuário. Por último, deseja-se poder alternar entre o gráfico de comprimento de onda por reflectância com o gráfico de comprimento de onda por emissividade.

Como foi salientado no trabalho, devido a certa dificuldade de uso da biblioteca gráfica alguns objetivos não foram alcançados, então, será feito um estudo maior de outras biblioteca. Visto que todas anteriormente estudadas ou não eram gratuitas ou não possuíam as interações necessárias, será feito o estudo de uma nova, a Data-Driven Documents (http://d3js.org/), que, apesar de não possuir inicialmente as interações procuradas, é gratuita e possui várias outras bibliotecas criadas a partir de seu código.

Durante o desenvolvimento do sistema, surgiu uma nova ideia que foi aceita pelas partes interessadas no projeto, e por ser necessário realizar mudanças no banco de dados, ficou programada para ser desenvolvida em trabalhos futuros, de modo que o protótipo atual seja finalmente testado por alguns usuários. A nova funcionalidade consiste em adicionar a latitude e longitude de onde a amostra foi colhida, e com isso o sistema exibir um link que direcione para o *Google Latitude* (https://www.google.com.br/latitude/b/0?hl=pt-BR), que exibe em um mapa a localização exata do ponto escolhido.

Outra ideia que surgiu após a finalização da versão atual do projeto foi a possiblidade de anexar arquivos nas amostras, como fotos ou arquivos no formato *.pdf*, que poderiam conter descrições e informações dessa amostra, podendo haver níveis diferentes de confidencialidade e segurança.

Foi criado um questionário para avaliar a opinião de usuários após acessarem o sistema uma primeira vez, com a ideia de medir a satisfação quanto à facilidade de utilizar o site, de encontrar as informações procuradas e de adicionar uma amostra, por exemplo. Porém, faltou tempo hábil para aplicar os testes com a versão final do projeto. O questionário atual está incluído no apêndice B do trabalho e foi criado utilizando formulários do Google Drive (https://drive.google.com/).

REFERÊNCIAS

ABRAHAMSSON, Pekka; SALO, Outi; RONKAINEN, Jussi; WARSTA, Juhani. **Agile Software Development Methods**. Review and Analysis. Espoo 2002. VTT Publications 478. 107 p.

ASTER Spectral Library. **NASA**. Disponível em: < http://speclib.jpl.nasa.gov/>

CAKEPHP. **The Manual.** Disponível em: < http://book.cakephp.org/1.3/view/876/The-Manual >. Acesso entre março 2012 e novembro 2012.

COCKBURN, Alistar. **Escrevendo Casos de Uso Eficazes**. Porto Alegre: Bookman, 2005.

HOOK, S.J., KAHLE, A. B.; **The Micro Fourier Transform Interferometer (uFTIR).** A New Field Spectrometer for Acquisition of Infrared Data of Natural Surfaces, *Remote Sensing of Environment*, v.56, p.172-181, 1996.

KAPPEL, Gerti; PRÖLL, Birgit; REICH, Siegfried; RETSCHITZEGGER, Werner. **Web Engineering**. The Discipline of Systematic Development of Web Applications. Alemanha: John Wiley & Sons Ltd, 2003.

KRASNER, Glenn E.; POPE, Stephen T. **A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80**. Journal for Object-Oriented Programming, v. 1, n. 3, p. 26-49, 1988.

LILLESAND, T.M. ; KIEFER, R.W. **Remote Sensing and Image Interpretation**. . 2ª Edição. New York. John Wiley& Sons. 1987. 721p.

POREBSKI, B.; PRZYSTALSKI, K.; NOWAK, L. **Building PHP Applications**. With Symphony™, CakePHP, and Zend® Framework. Indianapolis, Wiley Publishing, Inc.

VAN DER MEER, F.; DE JONG, S.; **Imaging Spectrometry.** Basic Principles and Prospective Applications. Kluwer Academic Publishers, Estados Unidos, Holando, Inglaterra, Rússia, p. XXI - XXIII 2002.

Referências dos produtos utilizados

ARCSIN WEB TEMPLATES. **Cerulean Elegance**. Disponível em < http://templates.arcsin.se/cerulean-elegance-website-template/>

ASTAH. **Astah Community**. Disponível em < http://astah.net/editions/community >

CAKEPHP. **CakePHP Framework**. Disponível em: < http://cakephp.org/ >

HIGHCHARTS JS. **HighCharts**. Dísponível em: < http:// http://www.highcharts.com/>

MYSQL. **MySQL Database**. Disponível em < http://www.mysql.com/ >

MYSQL. **MySQL Workbench**. Disponível em < http://www.mysql.com/products/workbench/ >

WAMPSERVER. **WampServer**. < http://www.wampserver.com/en/ >

APÊNDICE A DICIONÁRIO DE DADOS

Nas tabelas abaixo se encontra o dicionário de dados referente à base de dados desenvolvida neste trabalho. Cada uma delas contêm os atributos da entidade, seu tipo de dado, descrição e alguma observação a seu respeito, caso necessário.

Tabela A.1: Dicionário de dados da tabela *users*

Coluna	*Tipo do Dado*	*Descrição*	*Observações*
id	bigint	Identificador do usuário	-
username	varchar(20)	Nome de acesso do usuário	-
password	varchar(100)	Senha de acesso	-
name	varchar(150)	Nome do usuário	-
email	varchar(100)	E-mail do usuário	-
role	varchar(5)	Papel do usuário no sistema	Valores: admin, valid, inser
active	int	Flag se usuário está ativo no sistema	0 = sim 1 = não

Fonte: tabela criada pelo autor

Tabela A.2: Dicionário de dados da tabela *types*

Coluna	*Tipo do Dado*	*Descrição*	*Observações*
id	int	Identificador do tipo	-
name	varchar(45)	Nome do tipo	Tipos aceitos pelo sistema: mineral, rocha, solo, feita pelo homem
description	blob	Descrição breve do tipo	Opcional
active	int	Flag se tipo está ativo no sistema	0 = sim 1 = não

Fonte: tabela criada pelo autor

Tabela A.3: Dicionário de dados da tabela *samples*

Coluna	*Tipo do Dado*	*Descrição*	*Observações*
id	bigint	Identificador da amostra	-
name	varchar(100)	Nome da amostra	-
waveLenght	varchar(100)	Senha de acesso	-
origin	varchar(100)	Local de origem da amostra	Opcional; Ex: Paraná
description	blob	Descrição da amostra	Opcional
measureType	varchar(45)	Tipo de medida utilizada na amostra	Valor: Infravermelho; Pode adicionar outros
dataFile	blob	Arquivo gerado pelo espectrorradiômetro	Dados que geram o gráfico
x_start	int	Ponto inicial de exibição do gráfico	Opcional
x_inicial	int	Ponto final	Opcional
valid	varchar(3)	Se a amostra já foi validada	Valores: sim, não
types_id	int	Identificador do tipo da amostra	Chave estrangeira
users_id	bigint	Identificador do usuário que criou a amostra	Chave estrangeira

Fonte: tabela criada pelo autor

Tabela A.4: Dicionário de dados da tabela *sample_classes*

Coluna	*Tipo do Dado*	*Descrição*	*Observações*
id	int	Identificador da classe da amostra	-
name	varchar(45)	Nome da classe da amostra	-
description	blob	Descrição breve da classe	Opcional
active	int	Flag se classe está ativa no sistema	0 = sim 1 = não
types_id	int	Identificador do tipo à qual a classe pertence	Chave estrangeira

Fonte: tabela criada pelo autor

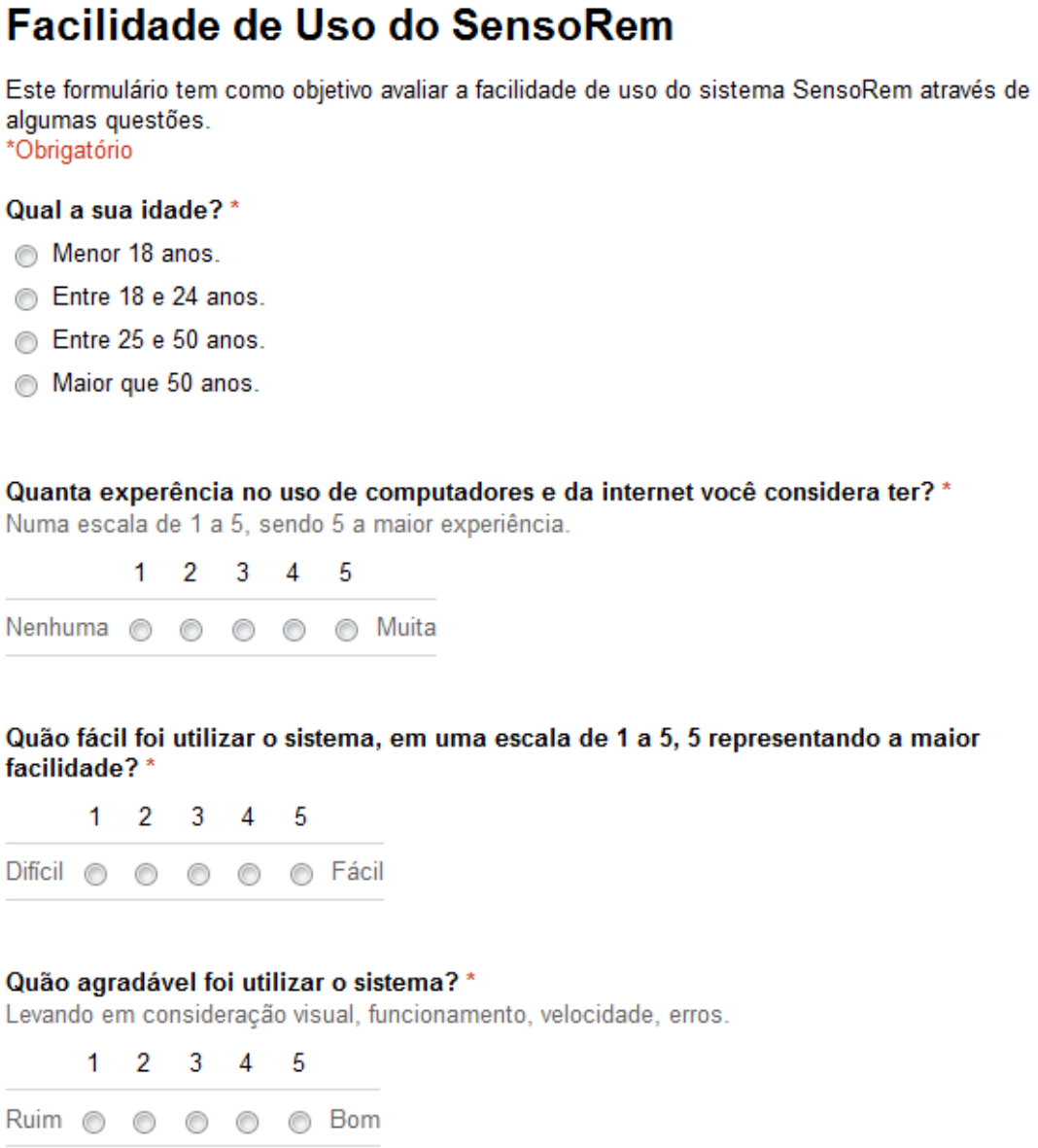
Tabela A.5: Dicionário de dados da tabela *subclasses*

Coluna	*Tipo do Dado*	*Descrição*	*Observações*
id	int	Identificador da subclasse da amostra	-
name	varchar(45)	Nome da subclasse da amostra	-
description	blob	Descrição breve da subclasse	Opcional
active	int	Flag se subclasse está ativa no sistema	0 = sim 1 = não
sample_classes_id	int	Identificador da classe à qual a subclasse pertence	Chave estrangeira

Fonte: tabela criada pelo autor

APÊNDICE B QUESTIONÁRIO USABILIDADE

Nas imagens abaixo se visualizam as questões propostas para o usuário.



Facilidade de Uso do SensoRem

Este formulário tem como objetivo avaliar a facilidade de uso do sistema SensoRem através de algumas questões.
*Obrigatório

Qual a sua idade? *

- Menor 18 anos.
- Entre 18 e 24 anos.
- Entre 25 e 50 anos.
- Maior que 50 anos.

Quanta experência no uso de computadores e da internet você considera ter? *
Numa escala de 1 a 5, sendo 5 a maior experiência.

	1	2	3	4	5	
Nenhuma						Muita

Quão fácil foi utilizar o sistema, em uma escala de 1 a 5, 5 representando a maior facilidade? *

	1	2	3	4	5	
Difícil						Fácil

Quão agradável foi utilizar o sistema? *
Levando em consideração visual, funcionamento, velocidade, erros.

	1	2	3	4	5	
Ruim						Bom

Figura B.1 - Primeira parte quiestionário usabilidade

Fonte: imagem salva pelo autor

Quão fácil foi encontrar as informações desejadas? *

	1	2	3	4	5	
Difícil	○	○	○	○	○	Fácil

Quão completo considera o sistema? *

Referente ao tipo de informações que abrange, e não quantidade de amostras inseridas.

	1	2	3	4	5	
Incompleto	○	○	○	○	○	Completo

Utilizando o sistema como um usuário "Insersor", quão fácil foi inserir uma nova amostra?

	1	2	3	4	5	
Difícil	○	○	○	○	○	Fácil

Utilizando o sistema como um usuário "Validador", quão fácil foi avaliar uma amostra?

Por avaliar entende-se validar, editar ou apagá-la.

	1	2	3	4	5	
Difícil	○	○	○	○	○	Fácil

Utilizando o sistema como um usuário "Administrador", quão fácil foi adicionar/editar /excluir um usuário/tipo/classe/subclasse?

Pode ser respondido tendo realizado só uma das opções.

	1	2	3	4	5	
Difícil	○	○	○	○	○	Fácil

Possui alguma crítica, sugestão ou elogio? Deixe-nos saber respondendo no espaço abaixo.

Muito obrigado!

Figura B.2 - Segunda parte questionário usabilidade

Fonte: imagem salva pelo autor