UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

POLIANE DE SOUZA

PROTÓTIPO DE UM SISTEMA PARA GERAÇÃO DE LAUDOS DE ANÁLISES FÍSICO-QUÍMICAS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2015

POLIANE DE SOUZA

PROTÓTIPO DE UM SISTEMA PARA GERAÇÃO DE LAUDOS DE ANÁLISES FÍSICO-QUÍMICAS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Beatriz Terezinha Borsoi

PATO BRANCO
2015

ATA Nº: **265**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DA ALUNA **POLIANE DE SOUZA.**

Às 15:00 hrs do dia 18 de junho de 2015, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Soelaine Rodrigues Ascari (Convidada) e Lucilia Yoshie Araki (Convidada), para avaliar o Trabalho de Diplomação da aluna Poliane De Souza, matrícula 1209752, sob o título **Protótipo de um sistema para geração de laudos de análises físico-químicas**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação a candidata foi entrevistada pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 15:55 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

Profa. Lucilia Yoshie Araki, M.Sc.
Convidada

Coordenadora do Trabalho de Diplomação
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

SOUZA, Poliane de. Protótipo de um sistema para geração de laudos de análises físico-químicas. 2015. 77. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

O processo que envolve as atividades realizadas em laboratórios de análises físico-químicas, independentemente da sua especialidade, é semelhante. Nesse processo, em resumo, amostras para análise são recebidas, o procedimento de análise é realizado, com processo específico para cada tipo de material e análise, e um laudo com resultados é gerado. Embora o processo possa ser simples, o registro dos dados da análise pode ser complexo. Isso ocorre porque para cada material analisado, dados específicos são registrados. Outra característica comum a esses ambientes, além dos cuidados para não a contaminação de pessoas e dos materiais analisados, é a necessidade de rastreabilidade e identificação sem falhas do material analisado. Isso no sentido de que seja assegurado que o laudo de referida amostra é o que efetivamente produziu os resultados registrados. Para facilitar essa identificação, sugere-se, neste trabalho o uso de código de barras para acompanhamento da amostra. Visando contribuir para o registro das amostras, dos resultados das análises realizadas e emissão de laudos e laboratórios de análises físico-químicas, neste trabalho é proposto o protótipo funcional de um sistema para gerenciamento das atividades realizadas neste tipo de ambiente. E, também, apresentado algumas funcionalidades que foram codificadas visando exemplificar o uso de tecnologias para desenvolvimento *web* definidas para implementar o sistema para o qual o protótipo foi criado.

Palavras-chave: Gerenciamento de laboratórios. Análises físico-químicas. Laudos de análises. Protótipos de software.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

ADF	*Application Development Framework*
AJAX	*Asynchronous Javascript and XML*
API	*Application Programming Interface*
ASP	*Active Server Page*
CNPJ	Cadastro Nacional da Pessoa Jurídica
CSS	*Cascading Style Sheets*
DDL	*Data Definition Language* ou *Data Description Language*
GUI	*Graphical User Interface*
GWT	*Google Web Toolkit*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
J2EE	*Java Enterprise Edition*
JCP	*Java Community Process*
JDBC	*Java Database Connectivity Technology*
JDO	*Java Data Objects*
JPA	J*ava Persistence API*
JSF	*JavaServer Faces*
JSP	*JavaServer Page*
MVC	*Model View Controller*
ORM	*Object-Relational Mapping*
PDF	*Portable Document Format*
PHP	*PHP HyperText Preprocessor*
POM	*Project Object Model*
RIA	*Rich Internet Application*
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
UTFPR	Universidade Tecnológica Federal do Paraná
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo estão as considerações inicias do trabalho, os seus objetivos e a justificativa. Por fim está a organização do texto por meio da apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

As análises físico-químicas possuem diversas aplicações (CETAL, 2015), seja na indústria para, por exemplo, atestar a qualidade de produtos ou determinar a sua composição; seja na agricultura para determinar elementos componentes de amostras de solos; seja com finalidades sanitárias para atestar qualidade ou identificar agentes contaminantes de água e efluentes.

Para cada elemento analisado como por exemplo: alimento, água, efluente ou solo, , pode ser medida uma série de componentes ou identificada a presença ou ausência de determinados elementos e agentes contaminantes. Considerando a quantidade dos materiais manipulados nesse tipo de ambiente e a diversidade de análises realizadas, um sistema para o gerenciamento desses laboratórios e das análises que realizam é amplo e complexo. Assim, verificou-se a importância de realizar a prototipagem completa de um sistema para o gerenciamento da realização das análises, antes da sua implementação. Permitindo, dessa forma, ao usuário navegar pelos menus e visualizar os campos dos formulários e os modelos de laudos gerados. Por meio dessa visualização o usuário pode avalizar se os requisitos pretendidos (representados pelos formulários e seus campos de entrada e relatórios, incluindo laudos, gerados) estão sendo atendidos.

A solução proposta terá como base o laboratório de análises físico-químicas da Universidade Tecnológica Federal do Paraná (UTFPR), Câmpus Pato Branco, mas pode ser aplicado para qualquer laboratório que realiza esse tipo de análise.

1.2 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos do desenvolvimento do trabalho.

1.2.1 Objetivo Geral

Implementar o protótipo de um sistema para gerenciamento de um laboratório de análises físico-químicas, o controle das análises realizadas e a emissão dos respectivos laudos.

1.2.2 Objetivos Específicos

- Possibilitar a visualização das funcionalidades do sistema por meio do uso de um protótipo funcional.
- Desenvolver a modelagem de um sistema para laboratórios que realizam análises físico-químicas para o cadastro de amostra, registro de análises realizadas e emissão de laudos.
- Apresentar o uso das tecnologias no desenvolvimento do sistema como forma de exemplificar a aplicação dessas tecnologias na implementação do protótipo.

1.3 JUSTIFICATIVA

As análises físico-químicas de elementos como alimentos, água e solos, entre outros , têm o objetivo de verificar a presença ou ausência de determinados componentes ou elementos. Considerando o crescimento da indústria e da agricultura, a contaminação do solo, da água e dos efluentes é muito frequente a necessidade de realizar análises em amostras desses elementos.

O desenvolvimento de um protótipo experimental (funcional, no sentido de permitir navegação pelos menus e opções), facilita o entendimento do sistema por fornecer uma visão geral das suas funcionalidades e a possibilidade de acessar formulários, itens de menu e exemplos de relatórios. O desenvolvimento do protótipo utilizando uma ferramenta de modelagem, ao invés de uma linguagem de programação, torna o modelo mais genérico (no sentido que ele poderá ser implementado em qualquer linguagem de programação que atenda os requisitos da aplicação) e mais simples em termos de entendimento pelo usuário. Simples, porque o usuário não precisa entender especificidades da tecnologia para interagir com o sistema. Os componentes que compõem a interface são tratados como desenhos em formato que representa itens comumente encontrados em aplicações de amplo uso, como aplicativos para automação de escritórios e páginas *web*.

A escolha da ferramenta Balsamiq para a modelagem do protótipo funcional decorre de a mesma permitir representar de maneira esquemática (como se fossem desenhos manuais) todos os elementos pretendidos para o sistema. Fornecendo, assim, a ideia de desenhos obtidos a partir de reuniões utilizando telas ou mesmo folhas de papel para esquematizar a interação do usuário com o aplicativo à medida que o modelo é discutido entre usuários e a equipe de projeto.

Para a implementação do aplicativo foram utilizadas tecnologias para desenvolvimento *web* baseadas em Java com a interface caracterizada como rica e, para tanto, foram utilizados *frameworks* como o *JavaServer Faces* (JSF) e a biblioteca de componentes PrimeFaces. JSF é o *framework* Java amplamente utilizado para desenvolvimento de aplicações *web*. Alguns dos principais motivos para esse uso intensivo são: oferecer um excelente conjunto de funcionalidades, possuir bons materiais de estudo e exigir pouco conhecimento inicial para a construção de interfaces de usuários (FRANCO, 2011).

Neste trabalho de conclusão de curso apenas alguns formulários desenvolvidos foram implementados visando apresentar o uso das tecnologias. A ênfase do trabalho está na elaboração do protótipo.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. No Capítulo 2 está o referencial teórico que apresenta sobre desenvolvimento de aplicações para *web* com interface denominada rica. Esse é o tipo de aplicação desenvolvido. No Capítulo 3 estão os materiais e o método utilizados no desenvolvimento do trabalho. No Capítulo 4 são apresentados os resultados obtidos com a realização do trabalho. O Capítulo 5 apresenta a conclusão. E por fim estão as referências bibliográficas utilizadas na composição do texto.

2 REFERENCIAL TEÓRICO

Aplicação Internet rica é a caracterização do aplicativo desenvolvido como resultado deste trabalho, portanto, o referencial teórico é sobre esse tipo de aplicação e a tecnologia *JavaServer Faces* que é utilizada na implementação da mesma.

2.1 APLICAÇÕES INTERNET RICA

Nos últimos anos tem havido uma migração crescente de uma multiplicidade de aplicações tradicionalmente *desktop* para aplicações *web* (POWELL; NAKAMURA; AKAMA, 2009). A essa migração está vinculado ao interesse e, de certa forma, a necessidade dos usuários que as aplicações *web* ofereçam os mesmos recursos de interação que as aplicações *desktop*. As aplicações *web* tradicionais, assim definidas as baseadas em *HyperText Markup Language* (HTML) com *links* e formulários simples, são bem diferentes em termos de recursos de interação das aplicações *desktop* que possuem menus e botões diferenciados, efeitos de arrastar-e-soltar, entre outros.

A necessidade de aprimoramento da interface das aplicações *web* e sem afetar negativamente o tráfego de rede e outros requisitos técnicos, vem sendo atendida pelas aplicações denominadas *Rich Internet Application* (RIA). Essas aplicações são definidas visando unificar o melhor das aplicações *desktop* tradicionais - tais como a interatividade de interface com o usuário e o tempo de resposta sem a necessidade de recarregamento da página que é típico nessas aplicações - com o melhor das aplicações *web* – como o uso de *download* progressivo para obtenção de conteúdo (DUHL, 2003).

As RIAs apresentam as seguintes características (DUHL, 2003; PRECIADO *et al*., 2005):

a) *offloading* de camadas de apresentação e interatividade a partir do servidor para o cliente;

b) redução da taxa de transferência de dados entre cliente e servidor pelo carregamento/atualização parcial das páginas;

c) distribuição do processamento das páginas entre o cliente e o servidor;

d) comunicação assíncrona e concorrente entre o servidor e o cliente.

Essas características também têm mostrado que modelos de comportamento padrão usados para especificar aplicações *web* tradicionais são inadequados para as RIAs uma vez

que elas podem exibir comportamento mais rico e flexível, tais como a computação envolvendo somente fragmentos parciais da interface (COMAI; CARUGHI, 2007). Além disso, há uma diversidade de tecnologias (tais como JavaScript, HTML e *Cascading Style Sheets* (CSS) para o cliente; *JavaServer Page* (JSP), Perl, PHP (*PHP HyperText Preprocessor)*, *Active Server Page* (ASP) para o servidor; *Structured Query Language* (SQL), XQuery para o banco de dados) que são necessárias para implementar aplicações *web* tradicionais e RIAs em particular (COOPER *et al*., 2007). Com o uso de Java como linguagem de programação para o servidor, a interface com o cliente pode ser desenvolvido por um conjunto bastante amplo de *frameworks*, tais como JavaServer Faces(JSF) (JAVASERVER…, 2014), Spring (JOHNSON *et al*., 2011) e Grails (GRAILS..., 2014), aos quais são agregadas bibliotecas de componentes, como PrimeFaces, IceFaces e RichFaces para JSF.

Essa diversidade de tecnologias resulta em um amplo conjunto de dificuldades, tais como problemas de segurança, questões de assincronismo e falhas de implementação tecnológica (POWELL; NAKAMURA; AKAMA, 2009). Além disso, adicionam-se dificuldades no desenvolvimento de RIAs, pelo fato de elas serem construídas com tecnologias distintas, não há um caminho de comunicação claro e/ou fácil entre elas (BOZZON et al., 2006; STEARN, 2007).

2.2 FRAMEWORKS JAVA

Nas palavras de Pree (1994), um *framework* de software é composto por "*frozen spots*" e "*hot spots*". Os *frozen spots* definem a arquitetura geral ou básica do sistema de software, ou seja, os componentes básicos da arquitetura e o relacionamento entre eles. São definidos como *frozen* porque não podem ser alterados nas instanciações de aplicação do *framework*. Os *hot spots* representam as partes do *framework* que são especificadas em cada aplicação desenvolvida. São definidos de *hot* porque são projetados para serem genéricos e especializados na definição das aplicações.

Buschmann *et al*. (1996) também se refere aos *hot spots* em um *framework* com "locais" nos quais adaptações são realizadas para implementar funcionalidades específicas. E os autores ressaltam que em um ambiente orientado a objetos, um *framework* consiste de classes concretas e abstratas e instanciar um *framework* consiste em compor uma aplicação a partir de classes existentes e criar subclasses a partir dessas classes.

De acordo com a definição de Buschmann *et al*. (1996), um *framework* é um software projetado para ser instanciado. Esse software pode ser parcialmente completo e define uma arquitetura para uma família de subsistemas e oferece os construtores básicos para criá-los. Os *frameworks* também possuem pontos de extensão definidos, os chamados *hot spots*. Nesses pontos são feitas as adaptações do código para um funcionamento específico do sistema implementado a partir da instanciação e complementação do *framework*. Um *framework* é caracterizado pela certa dependência entre as suas classes, conhecida como modelo de colaboração (OLIVEIRA, 2014).

A linguagem Java apresenta uma série de *frameworks* que auxiliam no desenvolvimento de sistemas (OLIVEIRA, 2014). Fanzini (2013) organizou 35 categorias de *frameworks* Java para desenvolvimento de aplicações. Algumas dessas categorias, as consideradas mais pertinentes ao escopo deste trabalho são listadas nos Quadros 1 e 2. A composição desses quadros é realizada a partir de dados apresentados por Fanzini (2013) que são complementados com dados constantes em Oliveira (2014) e Shan e Hua (2006).

Especificação	**Framework**	**Descrição**	**Disponível em**
Contêiner de inversão de controle	Spring	Baseado em orientação a aspectos. Possibilidade de uso em conjuntos com outros *frameworks Model View Controller* (MVC), como o Struts e JSF.	http://www.springsource.org/
	Java Data Objects (JDO)	Componente do lado servidor que encapsula a lógica de negócio de uma aplicação.	http://www.oracle.com/technetwork/java/javaee/ejb/index.html
Persistência	Spring Data	Usa conceitos de banco de dados, além do mapeamento objeto-relacional (classes Java para tabelas de databases).	http://www.hibernate.org/
	Commons Validator	Interface que provê uma camada de abstração aplicação - armazenamento de dados.	**http://www.java.sun.com/products/JDO**
	Java Bean Validator	Usa uma JVM invulnerável logicamente com uso de uma camada de prevalência de objetos.	**http://www.prevayker.org**
	Swing	Implementação da camada de persistência de dados. Oferece funcionalidades sofisticadas e comuns à maioria dos métodos de acesso a banco de dados.	http://www.springsource.org/
Validação	Ruby On Rails	Associado ao Structs Validator.	http://commons.apache.org/validator/
	Jbanana	Aplicação dos dados a serem persistidos.	http://docs.oracle.com/javaee/6/tutorial/doc/gircz.html
Kit de interface gráficas desktop	JBoss	Criação de aplicações *desktop.*	http://docs.oracle.com/javase/tutorial/uiswing/
	iReport	Desenvolvimento de aplicações *desktop.*	http://www.jgoodies.com/
	Jasper Report	*Toolkit para widgets.*	http://www.eclipse.org/swt/
	JQuery	Composto por SWT designer e Swing designer para criação de interface	http://www.eclipse.org/windowbuilder/

RAD interface gráficas desktop		gráfica.	
	JSON	Para criação de interface gráfica com a IDE NetBeans.	http://netbeans.org/features/java/swing.html
Framework web RIA	Maven	Inclui contêiner para gerenciar componentes e um conjunto de serviços para interfaces de usuário, transações e persistência da *web.*	**http://www.springsource.org/**
	Struts	Usa o modelo MVC e é caracterizado por uma camada de controle com uso de *Java Enterprise Edition* (J2EE) e Extensible Markup Language (XML).	http://struts.apache.org/
	JSF	Tecnologia de *servlets* e JSP que pode ser usado como uma opção ao Struts.	**http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html**
	VRaptor	Solução brasileira para inversão de controle e injeção de dependências. Implementa MVC.	http://vraptor.caelum.com.br/
	Google Web Toolkit (GWT)	Conjunto de APIs, ferramentas e componentes visuais para criação de interface.	https://developers.google.com/web-toolkit/
	Ruby On Rails	*Framework* escrito em Ruby.	http://rubyonrails.org/
J2EE	Jbanana	*Framework* brasileiro e *open-source* para desenvolvimento *web*.	**http://www.jbanana.org**
Autenticação e autorização web	Spring Security	Criar um mecanismo de autenticação e autorização para sua aplicação *web*.	http://www.springsource.org/
	J2EE Container Managed Security	Define um modelo padrão JEE de regras de como a aplicação *web* pode definir e gerenciar controle de acesso.	http://tomcat.apache.org/tomcat-7.0-doc/realm-howto.html
JEE web container standalone	Tomcat	Servidor *web* Java, mais especificamente um contêiner de *servlets.*	http://tomcat.apache.org/
	JBoss	Servidor de aplicações.	http://www.jboss.org/
	Glassfish	Servidor de aplicações.	http://glassfish.java.net/
Relatórios e gráficos	iReport	Para geração gráfica de relatórios	http://jasperforge.org/projects/ireport
	Jasper Report	Framework para geração de modo dinâmico de relatórios. Compatível com formatos XML, Portable Document Format (PDF) e HTML	**https://www.jaspersoft.com/reporting-software**
JavaScript	DOJO Toolkit	Biblioteca em JavaScript.	http://dojotoolkit.org/
	JQuery	Biblioteca JavaScript *cross-browser* para os *scripts* do lado cliente que interagem com HTML.	http://jquery.com/
JSON	JSON	Subconjunto da notação de objeto de JavaScript.	http://www.json.org/java/
Gerais	Maven	Gerencia as dependências do projeto e o seu ciclo de vida: limpar, compilar, testar, empacotar, distribuir.	http://maven.apache.org/
	JLicense	Gerenciador de licenciamento de uso de software.	http://freecode.com/projects/jlicense

Quadro 1 – Frameworks Java
Fonte: composto a partir de Fanzin (2013) e Oliveira (2014).

2.2.1 JavaServer Faces

JavaServer Faces foi proposto pela *Java Community Process* (JCP), publicada em 2004, com o objetivo de padronizar um *framework* para desenvolvimento da camada de apresentação em aplicações *web* (PAGANINI, 2014).

JSF é um *framework* baseado em componentes. Com isso, para criar uma tabela, por exemplo, ao invés de criar um *loop* para adicionar linhas e colunas com *tags* HTML, um componente do tipo tabela é adicionado à página. Os principais objetivos desse *framework* são maximizar a produtividade, minimizar a complexidade de manutenção, evoluir a experiência do usuário com uso de técnicas *Asynchronous Javascript and XML* (AJAX) e proporcionar melhor integração com outras tecnologias *web* (FRANCO, 2011).

Os dois principais componentes da tecnologia *JavaServer Faces* são (ORACLE, 2013):

a) Uma *Application Programming Interface* (API) para representar componentes de interface com o usuário (*Graphical User Interface (GUI)*) e gerenciar seu estado; manipulação de eventos, validação do lado servidor e conversão de dados; definição de navegação de página; suporte à internacionalização e acessibilidade; e possibilidade de extensão para todas essas características.

b) Uma biblioteca de *tags* para adicionar componentes às páginas *web* e conectar componentes a objetos do lado do servidor.

Uma aplicação JavaServer Faces típica inclui (ORACLE, 213):

a) um conjunto de páginas *web* nos quais os componentes são definidos;

b) um conjunto de *tags* para adicionar componentes à página *web*;

c) um conjunto de *backing beans*, que são componentes JavaBeans que definem propriedades e funções para componentes de uma página;

d) um descritor de *deploy web* (arquivo web.xml);

e) opcionalmente, um ou mais arquivos de recursos de configuração da aplicação, como o arquivo faces-config.xml, os quais podem ser usados para definir uma página de regras de navegação e configurar *beans* e outros objetos e componentes customizados;

f) opcionalmente, um conjunto de objetos customizados, que podem incluir componentes customizados, validadores, conversores, ou *listeners*, criados pelo desenvolvedor da aplicação;

g) um conjunto de *tags* customizadas para representar objetos customizados na página.

JSF é uma especificação, portanto, é necessário escolher uma implementação. A implementação de referência da Oracle é a Mojarra, mas existem várias outras, como (FRANCO, 2011): MyFaces, *Application Development Framework* (ADF) e Shale. Além disso, existem extensões de implementações com componentes visuais e *kits* de componentes. JSF também permite ainda integração com outras tecnologias e *frameworks*. O Quadro 2 apresenta tecnologias vinculadas a JSF para desenvolvimento Java. Os dados desse Quadro são provenientes de Fanzin (2013), Oliveira (2014) e Franco (2011).

Especificação	**Framework**	**Disponível em**
Provedores de JSF	Mojorra	http://javaserverfaces.java.net/
	MyFaces	http://myfaces.apache.org/
	ADF	http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html
	Shale	http://shale.apache.org/
Componentes JSF	RichFaces	http://www.jboss.org/richfaces
	PrimeFaces	http://primefaces.org/
	OpenFaces	http://openfaces.org/
	IceFaces	http://www.icesoft.org/
	EasyFaces	http://www.easyfaces.com.br
	Gmaps4jsf	http://code.google.com/p/gmaps4jsf/
Extensões JSF	PrettyFaces	http://ocpsoft.org/prettyfaces/
	Omnifaces	https://showcase-omnifaces.rhcloud.com/
Soluções web mobile	JSF RichFaces Mobile	http://www.jboss.org/richfaces
	JSF PrimeFaces Mobile	http://primefaces.org/
	JSF OpenFaces Mobile	http://openfaces.org/
	JSF IceFaces Mobile	http://www.icesoft.org/

Quadro 2 – Frameworks JSF
Fonte: composto a partir de Fanzin (2013), Oliveira (2014) e Franco (2011).

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a modelagem e o desenvolvimento do protótipo e a implementação básica do aplicativo para laboratórios de análises físico-químicas que é objeto da realização deste trabalho.

3.1 MATERIAIS

No Quadro 3 estão as ferramentas e as tecnologias utilizadas para a modelagem e a implementação do sistema.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
Astah* Community	6.9.0 (model version : 33)	http://astah.net/editions/community	Documentação da modelagem baseada na UML.
Case Studio 2	2.25	http://www.casestudio.com	Modelagem do diagrama de entidades e relacionamentos do banco de dados.
Balsamiq Mockups	3.0.7	https://balsamiq.com/products/mockups/	Prototipagem da interface (telas do sistema)
Linguagem Java	JDK 1.7	http://www.oracle.com	Linguagem de programação.
NetBeans	8.0.1	http://www.netbeans.org/	Ambiente de desenvolvimento.
MySQL	5	http://www.mysql.com/	Banco de dados.
MySQL WorkBench	6.2 CE	http://www.mysql.com/products/workbench/	Administrador do banco de dados.
Apache Tomcat	8.0.15	http://www.apache.org/	Servidor *web* para a aplicação.
PrimeFaces	5.1	http://primefaces.org/	Biblioteca de componentes para *web*.
JSF	2.0	https://javaserverfaces-spec-public.java.net/	*Framework* para *web.*
Hibernate	3	http://www.hibernate.org/	Efetuar o mapeamento objeto-relacional e a persistência dos dados.
Maven	3.3.3	https://maven.apache.org/	Controle e gerenciamento de dependências no projeto.
JPA		http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html	Persistência de dados.

Quadro 3 – Ferramentas e tecnologias de modelagem e implementação

3.2 MÉTODO

Para o método foi utilizado o modelo sequencial linear de Pressman (2008) composto das atividades: análise, projeto, codificação e testes. Além das atividades propostas no modelo de Pressman (2008) uma etapa de planejamento e estudo das tecnologias foi realizada após o levantamento dos requisitos. Assim houve necessidade de incluir essas atividades, adaptando esse modelo de ciclo de vida clássico. As principais atividades desenvolvidas na realização do trabalho foram:

a) Verificação do domínio da aplicação

Para o entendimento dos processos envolvidos nos laboratórios de análise de solos, alimentos, água e efluentes da UTFPR foram realizadas conversas com as pessoas responsáveis por esses ambientes e que realizam as atividades de recepção de amostras e emissão dos laudos. Também foi realizado o estudo de laudos produzidos pelos laboratórios.

Documentos utilizados nesses laboratórios foram coletados (exemplos nos anexos deste texto). Os documentos são os gerados a partir das análises e dos laudos, ou seja, o resultado da análise que é entregue para o cliente. Esses documentos forneceram a ideia dos dados necessários para a emissão dos laudos e, portanto, dos campos para os formulários que alimentam as tabelas para gerar os laudos.

O entendimento da forma de obtenção dos dados das análises realizadas e do processo manual que é realizado para registro dos dados foi essencial para a definição dos requisitos do sistema. Foi possível identificar a forma geral de definição dos campos dos dados e assim estabelecer o formato para as tabelas em banco de dados que armazenam os campos dos formulários.

b) Levantamento de requisitos

Com base no entendimento do processo foram definidos os requisitos que compõem o escopo do sistema a ser desenvolvido. Também foi realizada uma primeira versão das permissões para os usuários do sistema.

c) Análise e Projeto

Na fase de planejamento inicialmente foram prototipadas todas as telas do aplicativo utilizando a ferramenta Balsamiq. As telas foram ligadas de maneira a permitir a navegação pelas telas do protótipo. Permitindo, assim, além de visualizar o leiaute e os campos dos formulários, a navegação e o fluxo de trabalho.

Em seguida o banco de dados foi modelado visando identificar as entidades persistentes para os cadastros e as operações de análise realizadas com as amostras cadastradas.

d) Estudo das tecnologias

Tecnologias candidatas foram listadas e a linguagem Java para *web* foi a escolhida para implementar o sistema. Em seguida foi escolhido o *framework* e a biblioteca de componentes para a definição da interface. Alguns exemplos de teste foram realizados visando o entendimento das tecnologias.

e) Implementação

Na fase de implementação foram desenvolvidas as funcionalidades de inclusão, exclusão, consulta e alteração de cadastros básicos.

4 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho, que é a modelagem, incluindo a prototipação, de um aplicativo para registro de resultados de análises físico-químicas realizadas em laboratórios. Também são apresentados exemplos da implementação de cadastros.

4.1 ESCOPO DO SISTEMA

Um laboratório de análises físico-químicas pode realizar diferentes tipos de análises para um mesmo material e analisar materiais distintos. O que será desenvolvido como resultado deste trabalho está relacionado aos processos de cadastro da amostra à emissão do laudos de análises bromatológicas, físico-químicas e microbiológicas realizadas pelo Laboratório Laqua, vinculado a UTFPR Câmpus de Pato Branco.

O sistema proverá o cadastro de amostras, o registro dos dados obtidos com a realização da análise das amostras e a emissão dos laudos a partir dos dados da análise realizada.

4.2 MODELAGEM DO SISTEMA

O Quadro 4 apresenta os requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF01	Cadastro de clientes	Cadastro dos clientes que solicitam análises no laboratório.
RF02	Cadastro de cidades	Cadastro de cidades utilizadas em outros cadastros.
RF03	Cadastro de estados	Cadastro de estados utilizados em outros cadastros.
RF04	Cadastro de fornecedores	Cadastro de fornecedores.
RF05	Cadastro de usuários	Cadastro de usuários com acesso ao sistema
RF06	Cadastro de amostras	Cadastro de amostras a serem analisadas.
RF07	Cadastro de tipo de amostras	Cadastro dos tipos de amostras.
RF08	Recibo de entrega da amostra	Recibo emitido para o cliente na entrega da amostra. Um recibo pode referi-se à várias

		amostras.
RF09	Análise	Registro dos resultados coletados durante as análises realizadas com as amostras.
RF10	Laudo	Emissão dos resultados das análises.

Quadro 4 – Requisitos funcionais

O sistema possuirá três tipos (em termos de acesso às funcionalidades do sistema) de usuário:

a) Administrador – com acesso a todas as funcionalidades do sistema.

b) Laboratorista – inclusão dos dados das análises.

c) Atendente – cadastro de amostra e emissão de laudos.

Os tipos de usuários (os atores) e os casos de uso são apresentados na Figura 1.

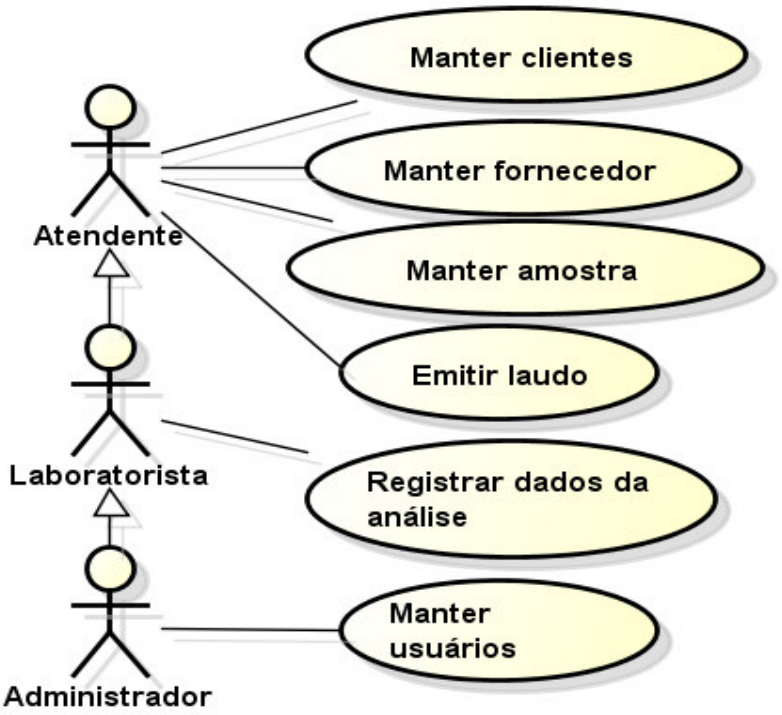


Figura 1 - Diagrama de casos de uso

Os Quadros 5 a 8 apresentam a descrição do caso de uso cadastrar. Essa descrição se refere à operação de inclusão de todos os casos de uso identificados como “manter”.

Caso de uso:
Incluir (refere-se à operação de inclusão de todos os casos de uso identificados como “manter”).
Descrição:
Inclusão dos dados cadastrais de um registro no sistema.
Evento Iniciador:
O usuário solicita a inclusão de um registro no sistema.
Atores:

Atendente, laboratorista ou administrador, no que couber.
Pré-condição:
Usuário logado no sistema.
Sequência de Eventos:
1. Ator acessa a tela para cadastro incluindo as informações necessárias.
2. O sistema insere os dados no banco de dados e informa o usuário que o referido cadastro foi realizado com sucesso.
Pós-Condição:
Registro inserido no banco de dados.
Extensões:
Informações provenientes de outros cadastros como, por exemplo, cidade e estado não cadastrados.

Quadro 5 – Operação "incluir" dos casos de uso de cadastro

No Quadro 6 é apresentada a expansão da operação excluir dos casos de uso de cadastro.

Caso de uso:
Excluir (refere-se à operação de exclusão de todos os casos de uso identificados como "manter").
Descrição:
Exclusão de dados cadastrais de um registro no sistema.
Evento Iniciador:
O usuário solicita a exclusão de um registro no sistema.
Atores:
Atendente, laboratorista ou administrador, no que couber.
Pré-condição:
Usuário logado no sistema.
Cadastro sem vínculos que impeçam a sua exlcusão.
Sequência de Eventos:
1. Ator acessa a tela para Usuário logado no sistema.
exclusão de cadastros.
2. Ator seleciona registro a ser excluído.
3. O sistema exclui o registro indicado e informa ao usuário que a referida exclusão foi realizada com sucesso.
Pós-Condição:
Registro excluído no banco de dados.
Extensões:
Registro sendo utilizado em outros cadastros.

Quadro 6 - Operação "excluir" dos casos de uso de cadastro

A operação de atualização de dados de cadastros já realizados é apresentada no Quadro 7 como a expansão do respectivo caso de uso.

Caso de uso:
Atualizar (refere-se à operação de atualização de todos os casos de uso identificados como "manter").
Descrição:
Atualização dos dados cadastrais de um registro no sistema.

Evento Iniciador:
O usuário solicita a atualização de um registro no sistema.
Atores:
Atendente, laboratorista ou administrador.
Pré-condição:
Usuário logado no sistema.
Cadastro a ser alterado incluído no sistema.
Sequência de Eventos:
1. Ator acessa a tela para visualizar o conteúdo de um registro.
2. O sistema apresenta o registro selecionado para edição.
3. O usuário altera os dados e solicita alteração do registro.
4. O sistema inclui os dados alterados e informa ao usuário que a operação foi realizada conforme solicitado.
Pós-Condição:
Dados do registro alterados no banco de dados.

Quadro 7 - Operação "excluir" dos casos de uso de cadastro

No Quadro 8 é apresentada a operação de consulta de dados de cadastros, como expansão de caso de uso.

Caso de uso:
Consultar (refere-se à operação de consulta de todos os casos de uso identificados como "manter").
Descrição:
Consulta dos dados cadastrais de um registro no sistema.
Evento Iniciador:
O usuário solicita a consulta de um registro no sistema.
Atores:
Atendente, laboratorista ou administrador, no que couber.
Pré-condição:
Usuário logado no sistema.
Existência de dados cadastrados que atendam os critérios de consulta.
Sequência de Eventos:
1. Ator acessa a tela para consultar o conteúdo de um registro.
2. O ator indica o parâmetro de consulta.
3. O sistema apresenta os resultados da consulta.
Pós-Condição:
Dados da consulta apresentados ao usuário.

Quadro 8 - Operação "consultar" dos casos de uso de cadastro

O Quadro 9 apresentada a descrição do caso de uso emitir laudo.

Caso de uso:
Emitir laudo.
Descrição:
Emissão de um laudo que contém resultados das análises realizadas para uma determinada amostra.
Evento Iniciador:
O usuário solicita a emissão de laudo de uma amostra.

Atores: Atendente, laboratorista ou administrador. **Pré-condição**: Análise ter sido realizada para a referida amostra. **Sequência de Eventos**: 1. Ator acessa a tela para emissão de laudo. 2. O ator seleciona a amostra da qual o laudo deve ser emitido. 3. O sistema gera o relatório (laudo) e o apresenta permitindo a sua impressão.. **Pós-Condição**: Relatório com o laudo emitido.

Quadro 9 - Caso de uso "emitir laudo"

As Figuras a seguir apresentam o protótipo das telas desenvolvidas utilizando a ferramenta Balsamiq Mockup. As telas foram criadas dentro de um projeto e ligadas por meio de *links*. Desse modo é possível navegar pelas telas do protótipo e testar o fluxo de trabalho.

O sistema foi modelado com base nos dados obtidos do funcionamento e documentos manipulados pelo laboratório Laqua vinculado à UTFPR, Câmpus Pato Branco (modelos de documentos nos anexos deste texto), mas isso não impede que o sistema seja utilizado pelos demais laboratórios pertencentes à Universidade e localizados em outros Câmpus e mesmo por outros laboratórios. Os requisitos do sistema foram definidos de maneira a atender as funcionalidades principais desses ambientes. Na modelagem foi considerada a possibilidade de usos e também na futura implementação completa da aplicação (incluindo módulos de controle de estoque, módulo financeiro e emissão de laudos *on line* pelo cliente). A Figura 2 apresenta a tela inicial, com o *login* do usuário no sistema.

Figura 2 - Página inicial

A tela de *login* do sistema (Figura 2) foi projetada para que o usuário (seja cliente ou um funcionário do laboratório) selecione o Câmpus ao qual está vinculado o laboratório e efetue *login*. Essa forma de organização visa facilitar o gerenciamento de usuários e de acessos.

Na Figura 3 está a tela principal que é apresentada após o usuário estar logado no sistema. Os menus estão organizados pelas funcionalidades: cadastros, amostras, análises e usuários. Ao lado esquerdo estão localizados os atalhos das principais funcionalidades da aplicação.

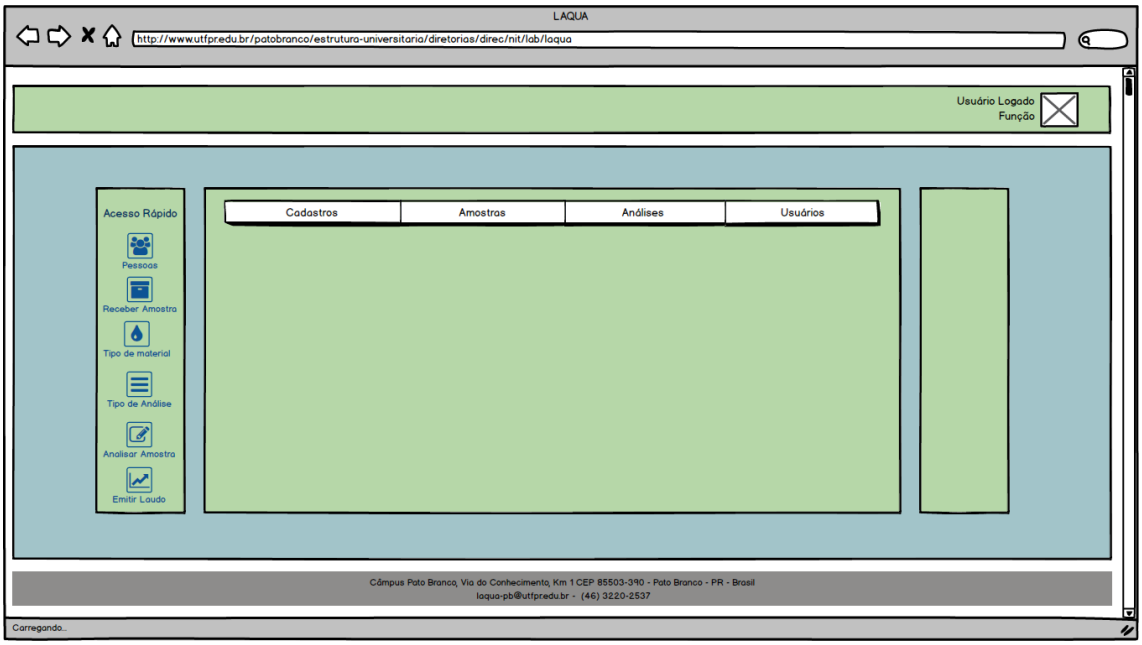


Figura 3 - Página principal

O menu cadastro é composto pelas opções de Cadastro de Pessoas, de Produtos e do submenu Localidades. Além dos dados comuns de um cadastro de pessoa, esse cadastro terá outros dados importantes, como os da aba Estabelecimentos e da aba Cotas de análise.

A aba Estabelecimentos (Figura 4) registra os locais de amostragem utilizados pelo coletor, não sendo necessariamente uma propriedade particular.

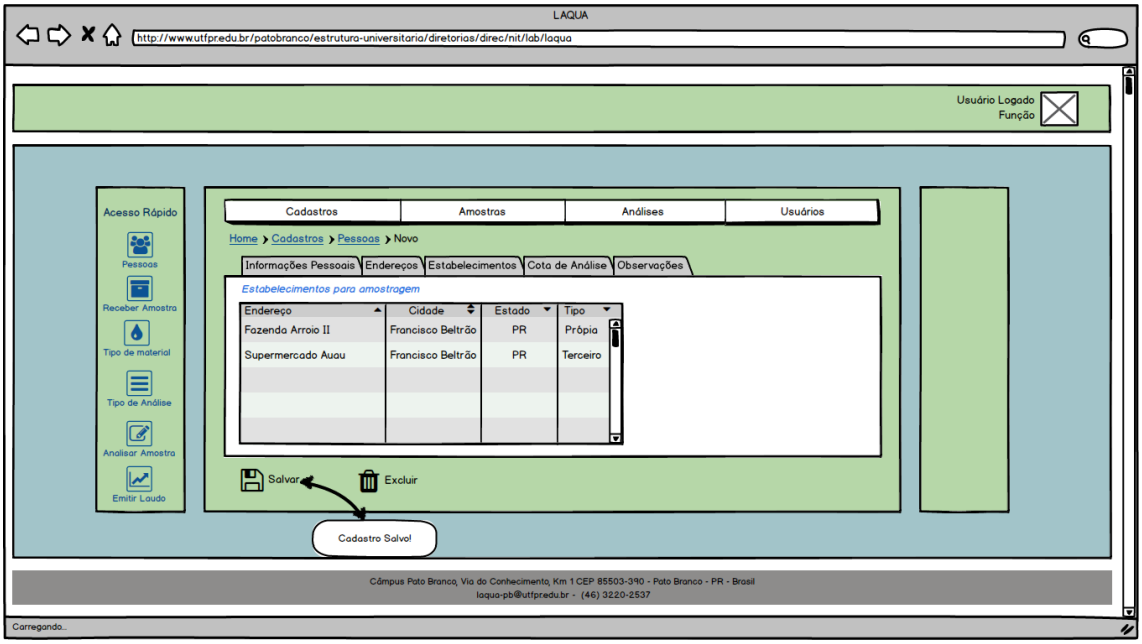


Figura 4 - Cadastro de Pessoas - aba Estabelecimentos

A aba Cotas de análise (Figura 5) servirá para controle das cotas dos servidores internos. Essas cotas são destinadas, por exemplo, a alunos de graduação e pós-graduação e a professores e/ou alunos vinculados a projetos de pesquisa.

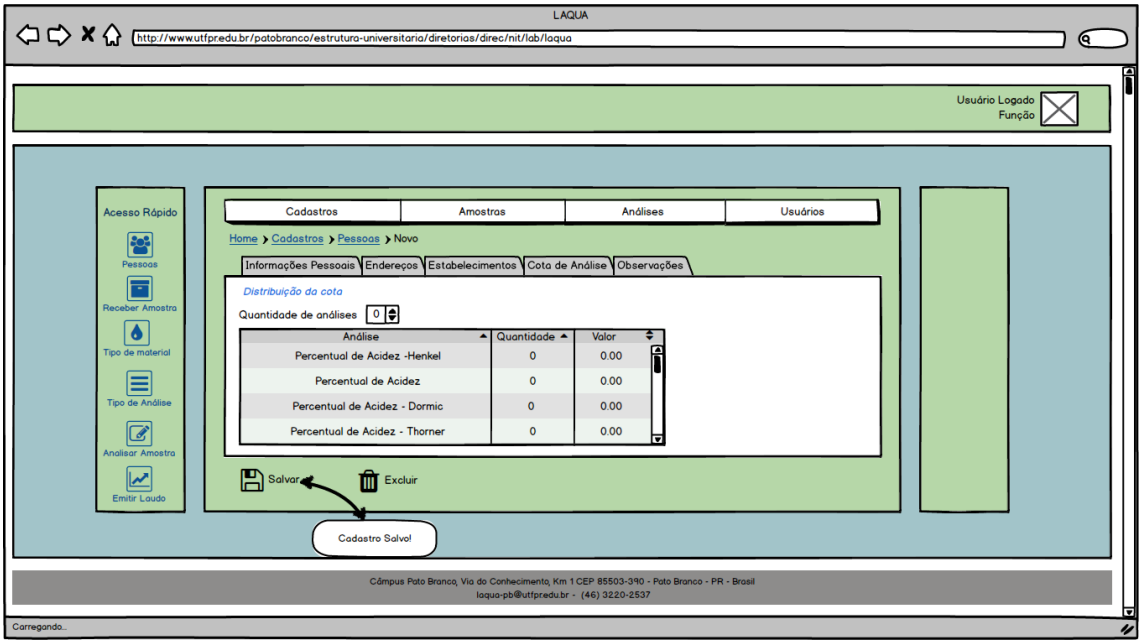


Figura 5 - Cadastro de Pessoas - aba Cota de Análise

O Cadastro de Produtos (Figura 6) possui o campo Tipo que indica se o produto é matéria-prima, manipulado no laboratório ou outras categorias pré-cadastradas.

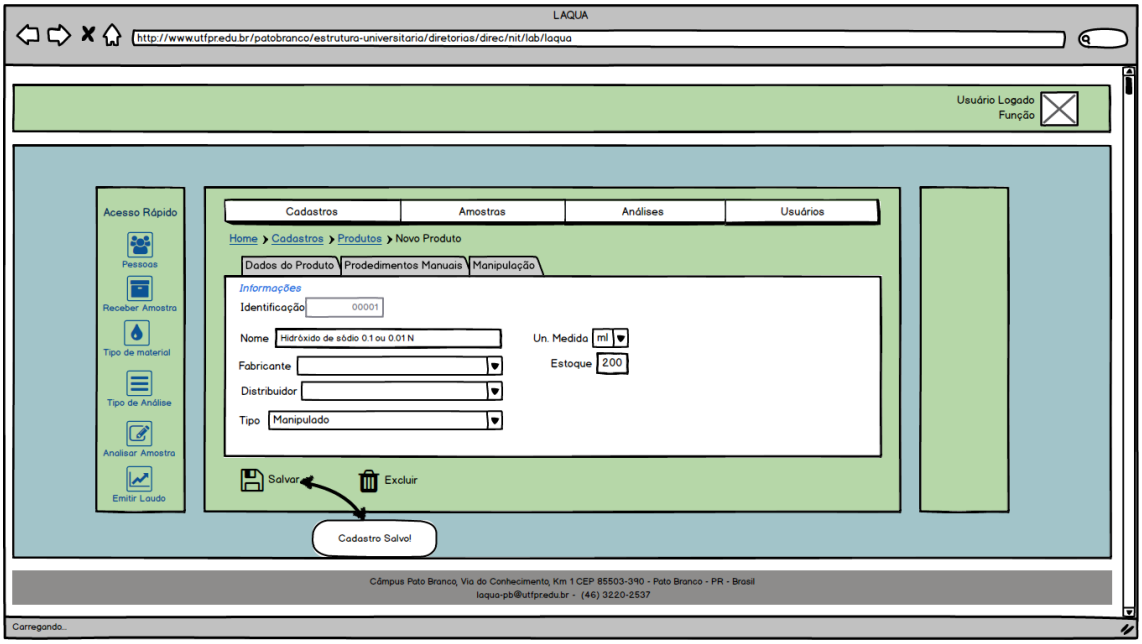


Figura 6 - Cadastro de Produtos - aba Dados do Produto

Ao selecionar a opção Manipulado do Cadastro de Produtos, a aba Manipulação é habilitada para indicar a composição do produto. A ideia é que os produtos manipulados também tenham controle de estoque. A Figura 7 apresenta a aba manipulação do formulário de produtos.

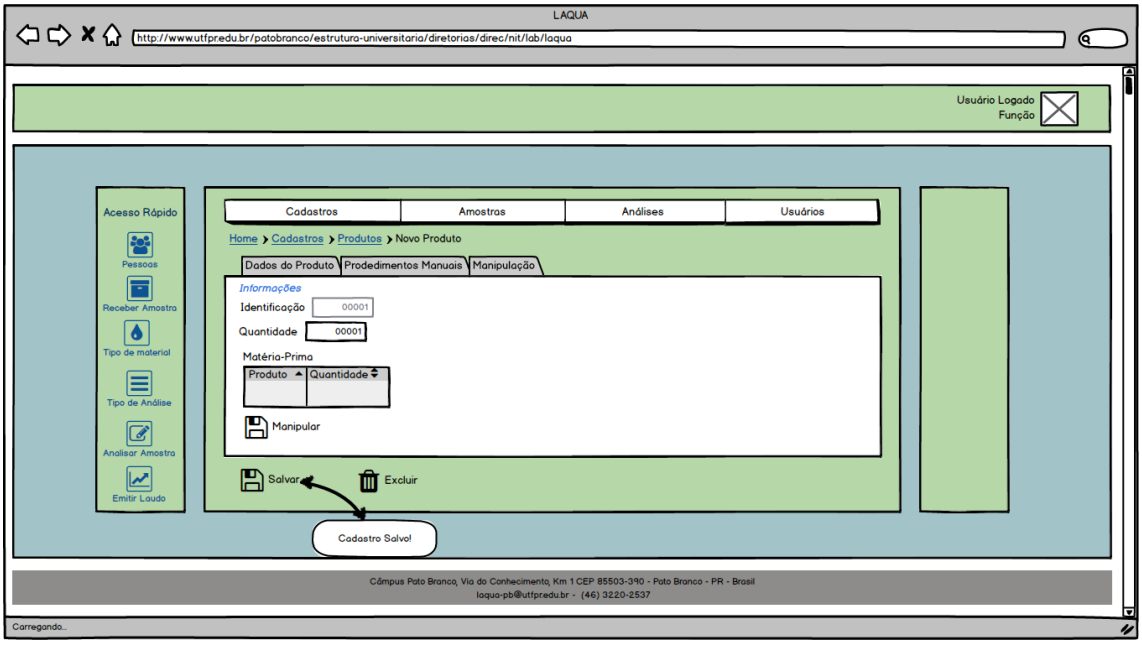


Figura 7 - Cadastro de Produto - aba Manipulação

Os cadastros básicos como os de Cidade, Estados e Países seguem o padrão da Figura 8. Esses são os cadastros mais simples da aplicação.

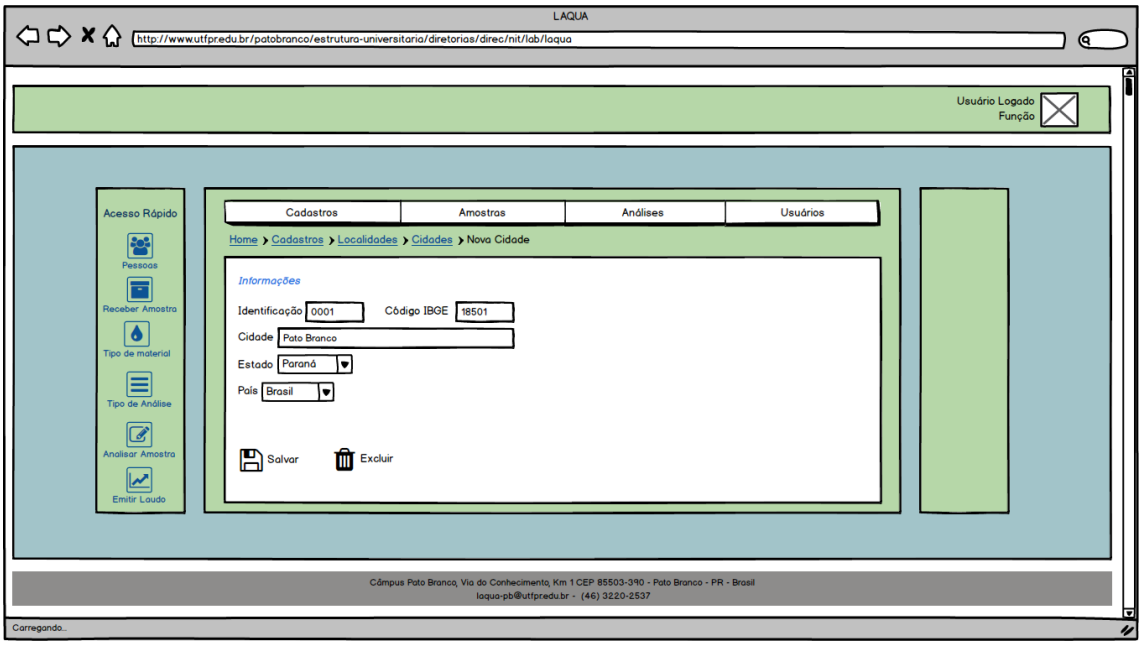


Figura 8 - Cadastro de Cidade

A sequência de Figuras 9, 10 e 11 referem-se às telas de Cadastro das Amostras. Além dos dados da amostra são informados dados sobre as análises a serem realizadas com o referido material.

A Figura 9 contém os dados de identificação da amostra entregues no laboratório para análise. Além de um identificador interno ao sistema, a amostra também vai possuir um código de barras gerado e impresso no momento do recebimento, esse código permitirá a rastreabilidade da amostra e está relacionado à segurança de identificação adequada dos dados da análise à sua respectiva amostra.

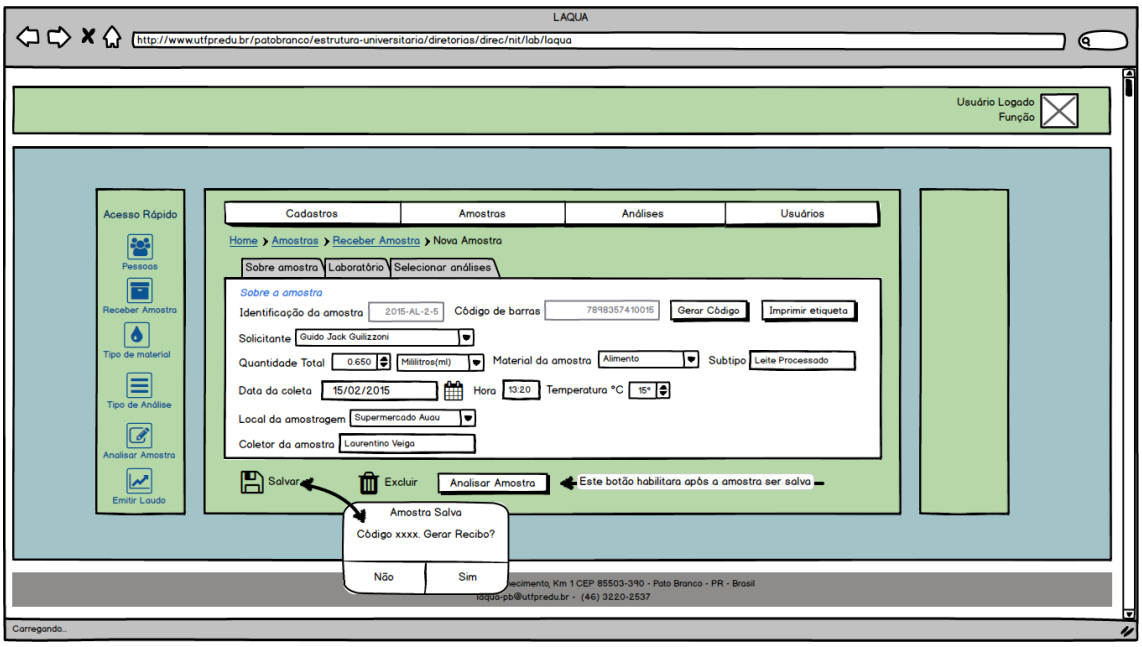


Figura 9 - Receber Amostra - aba Sobre a Amostra

A tela apresentada na Figura 10 é utilizada para registrar os dados referentes à entrada da amostra no laboratório e a previsão de entrega dos resultados. A previsão de entrega é calculada por meio da média do tempo informado no cadastro de tipos de análises.

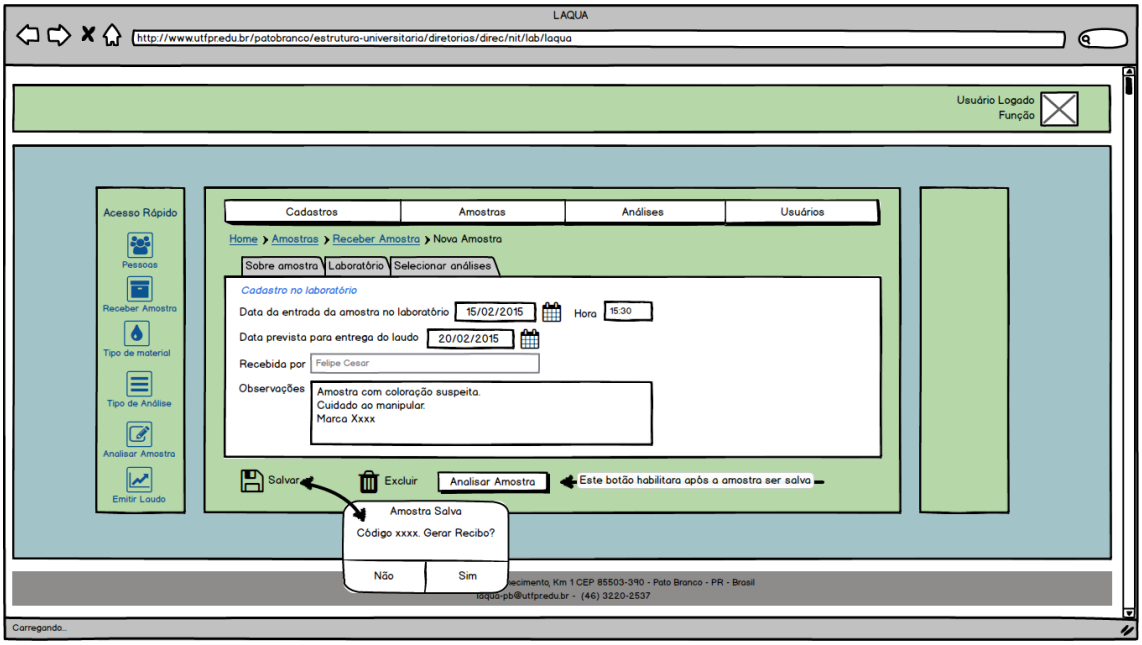


Figura 10 - Receber Amostra - aba Laboratório

A tela apresentada na Figura 11 registra as análises solicitadas para cada amostra, bem como os valores parciais e totais a serem pagos, descontos e acréscimos. Considerou-se que a forma de pagamento será apenas em espécie.

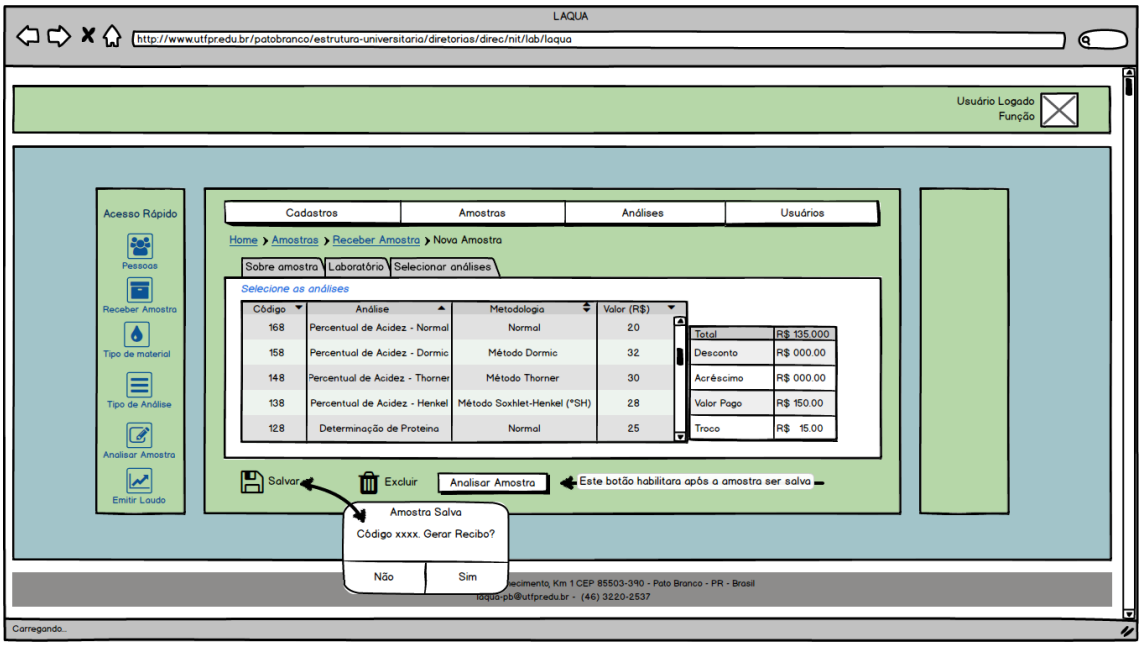


Figura 11 - Receber Amostra - aba Selecionar Análises

Após salvar o cadastro da amostra é possível gerar o Recibo de Entrega de Amostra. Esse recibo será portado pelo cliente como comprovante do recebimento do material. Pode-se, ainda, na mesma tela iniciar a análise do material. O cadastro de uma amostra só pode ser excluído caso nenhuma análise tenha sido realizada.

Na Figura 12 está o protótipo da tela de cadastro dos tipos de material a serem analisados. Esse cadastro é utilizado para classificar o tipo de material da amostra que chega ao laboratório.

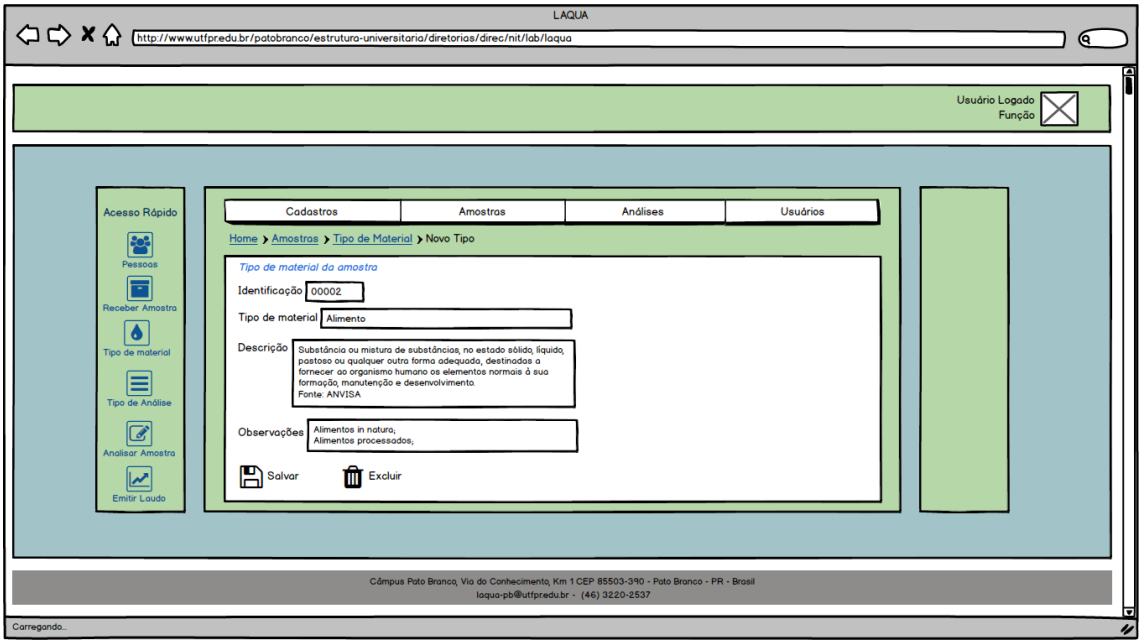


Figura 12 - Cadastro de Tipo de Material

O menu análises (Figura 13) apresenta as opções relacionadas à análise. Analisar amostra permite registrar os dados obtidos com a análise laboratorial do material. Em emitir laudo, o relatório com os resultados da análise realizada é gerado. Metodologias, parâmetros e referências são utilizados para compor o laudo final.

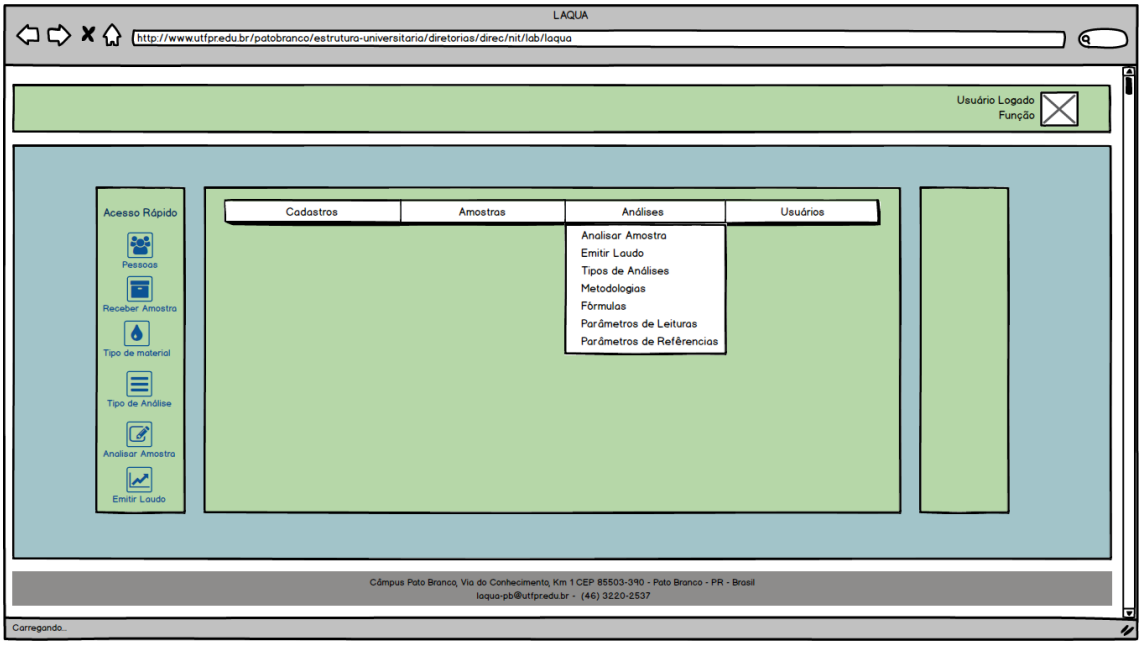


Figura 13 - Menu Análises

Nas Figuras 14 e 15 está o protótipo do formulário para análise da amostra. Nesse formulário serão indicadas as análises que serão feitas com cada amostra.

Na Figura 14 está a tela que é apresentada na primeira aba da análise de amostra. Ao selecionar o nome do solicitante, o campo Amostra é preenchido com as amostras do respectivo cliente que ainda não foram analisadas. Selecionando a amostra, o campo Código de Barras é preenchido. O usuário pode optar também por usar um leitor de código de barras e ler o código diretamente.

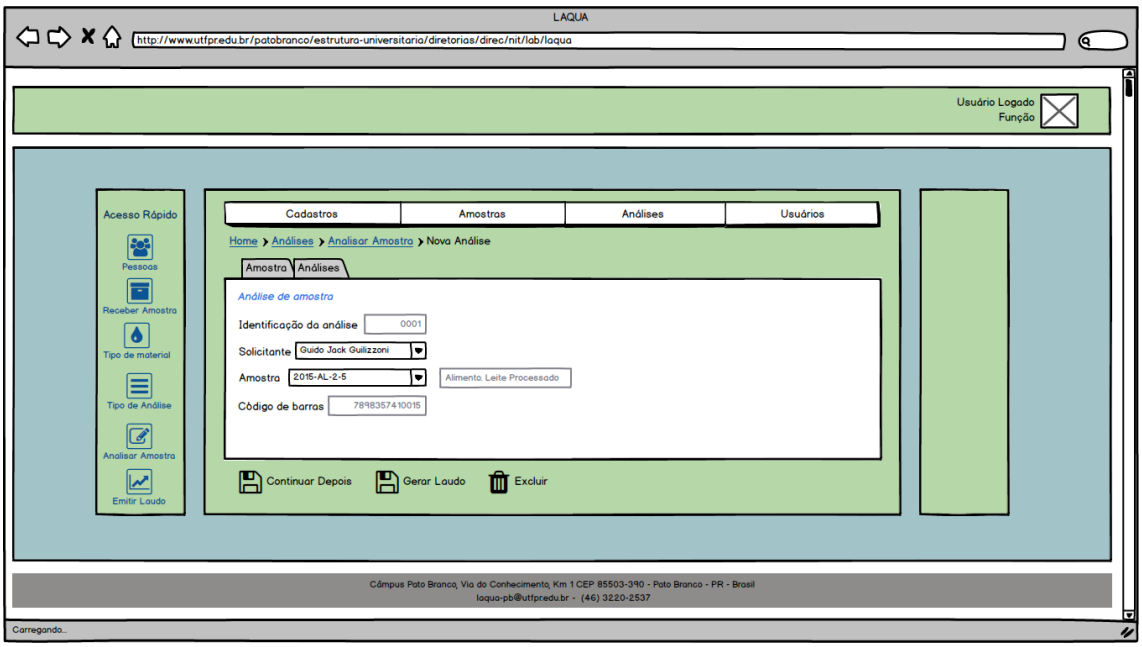


Figura 14 - Análise da amostra - aba Amostra

Na tela da Figura 15 é apresentada a segunda aba do cadastro de amostras. Ao selecionar uma análise os campos para registrar as leituras são automaticamente criados seguindo os parâmetros pré-definidos no cadastro do tipo de análise.

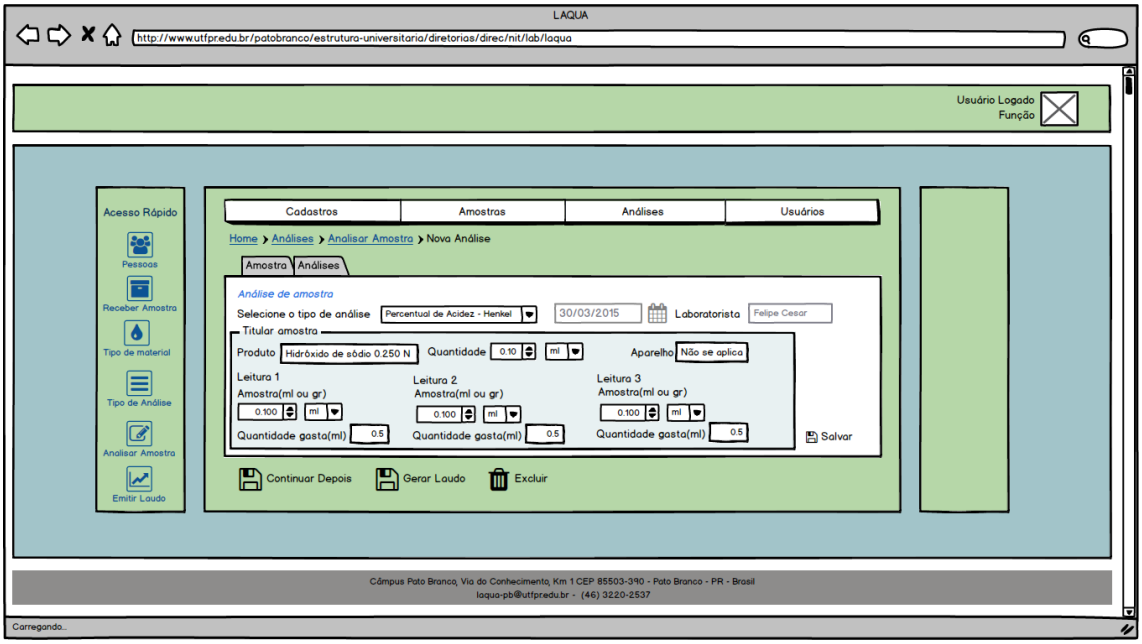


Figura 15. Análise da amostra - aba Análises

As Figuras 16, 17 e 18 referem-se ao cadastro de tipos de análise. Na imagem apresentada na Figura 16 é indicado o tipo de análise (físico, química etc.) e o tempo médio necessário para realizar este tipo de análise. Esse tempo servirá como base para o cálculo da data de previsão de emissão do laudo.

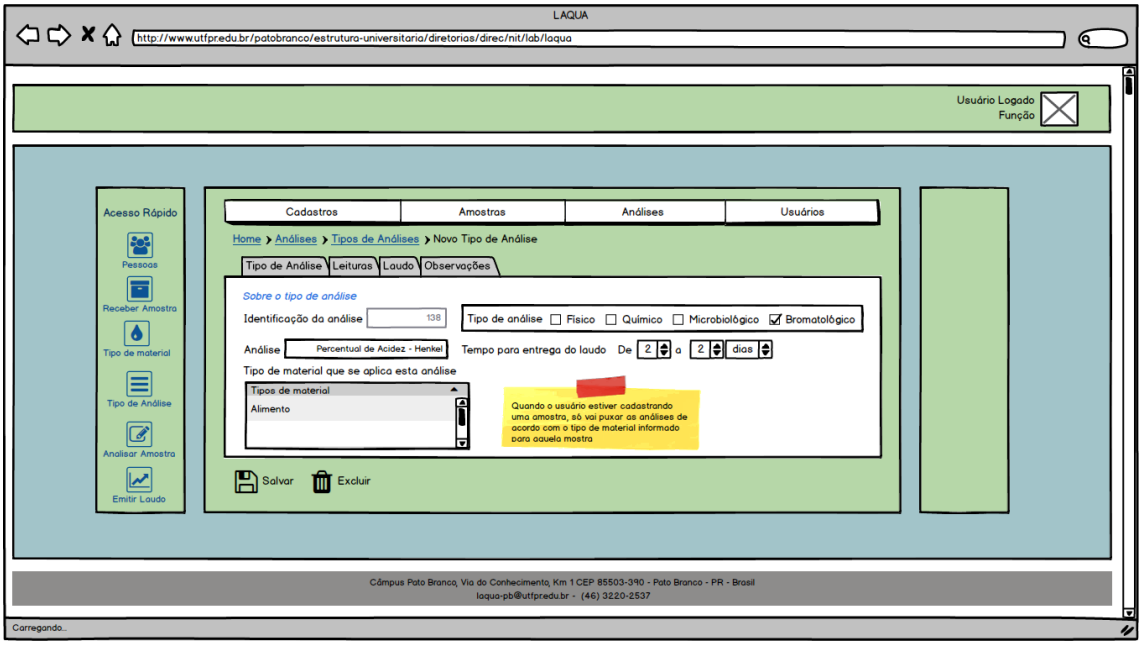


Figura 16 - Cadastro de Tipo de Análise - aba Tipo de Análise

Além disso, o usuário pode definir a que tipo de material esta análise pode utilizará, isso permitindo que no momento do cadastro de uma amostra sejam listadas apenas as que

forem pertinentes aquele tipo de material. A Figura 17 mostra a tela de configuração de leituras para aquela análise. Esse parâmetro vem dos dados cadastrados de Parâmetros de Leituras e refere-se aos dados que serão coletados durante a análise.

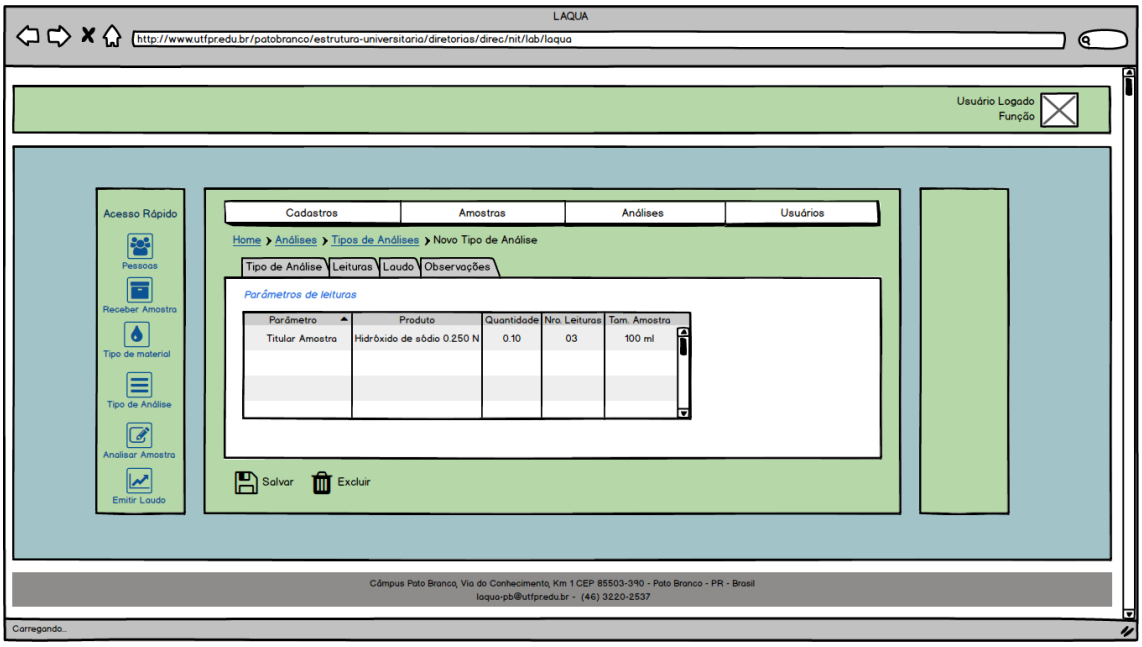


Figura 17 - Cadastro de Tipo de Análise - aba Leituras

Na imagem da Figura 18 é indica a fórmula para a análise e o material de referência para emissão de resultado.

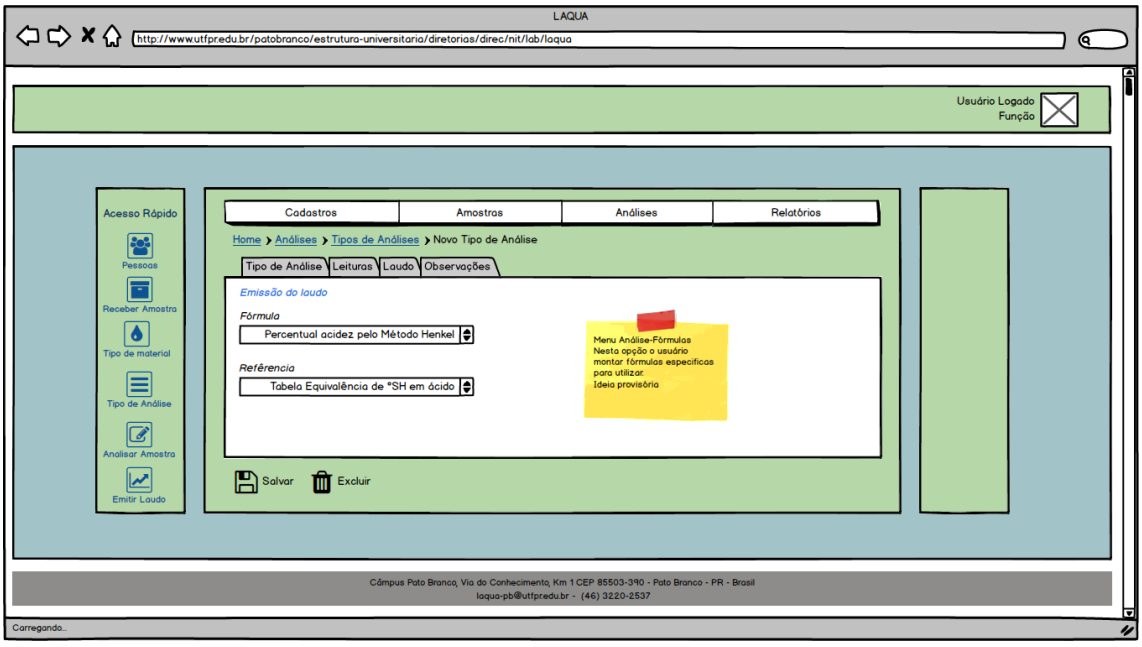


Figura 18 - Cadastro de Tipo de Análise - aba Laudo

A Figura 19 mostra a tela para cadastro de referências fornecidas por órgão ou fonte competente. Esses parâmetros são utilizados em recomendações das análises a partir dos resultados obtidos no laboratório.

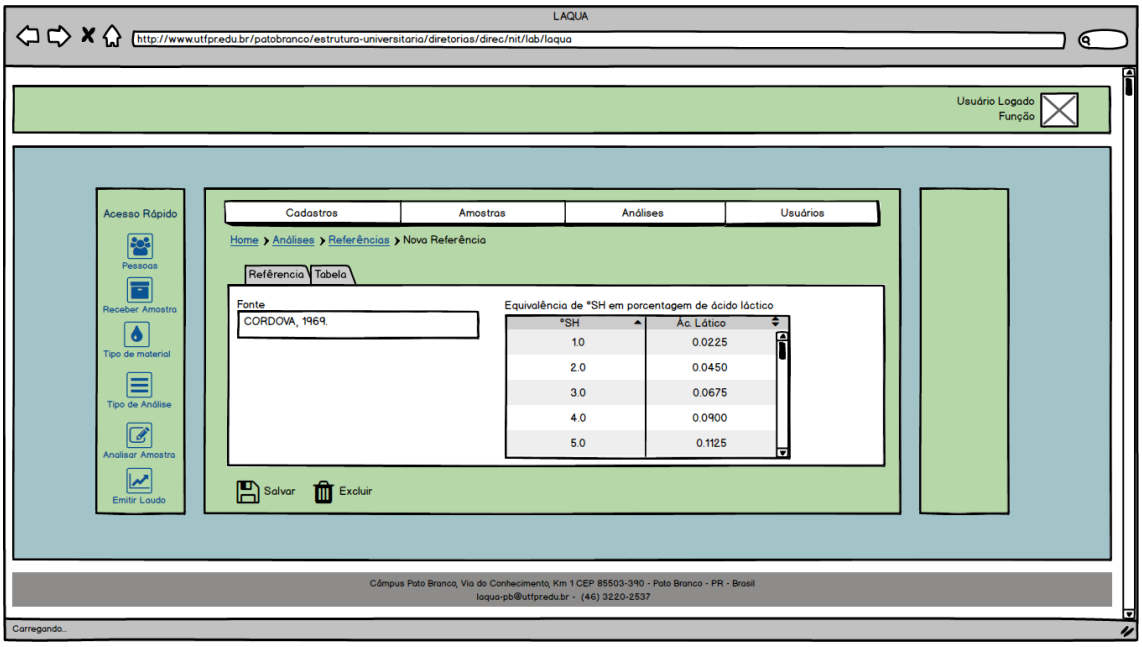


Figura 19 - Cadastro de Parâmetro de Referência, aba Tabela

O cadastro de fórmulas (Figura 20) permite ao usuário criar as próprias fórmulas e vinculá-las aos tipos de análises. Essa tela é uma das mais importantes, pois fornece a autonomia do usuário e desvincula a dependência de acréscimos e inclusões em linhas de códigos da aplicação.

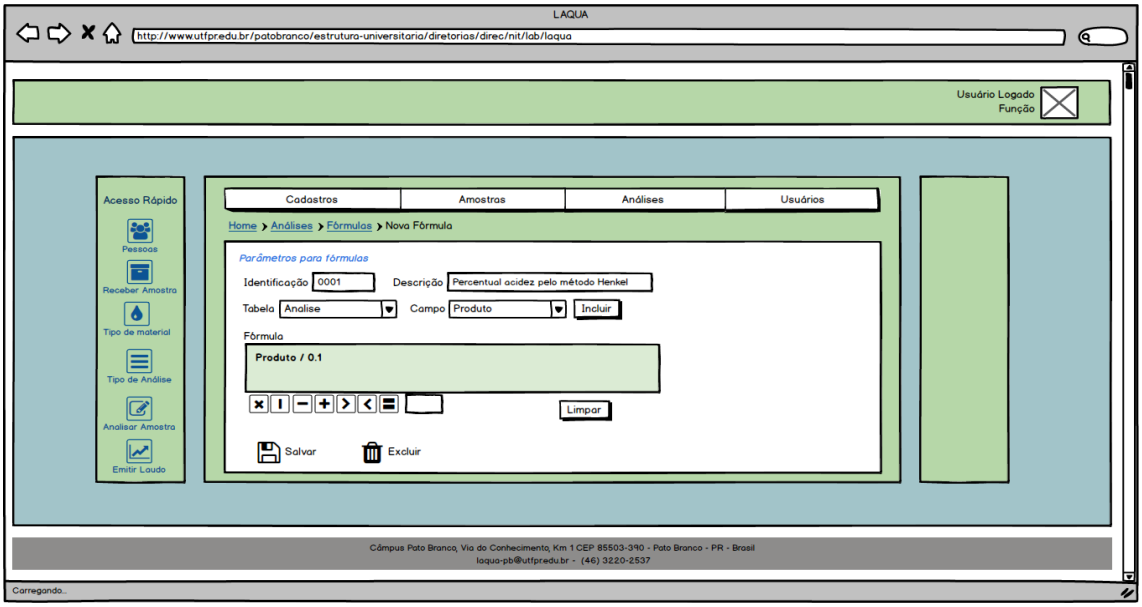


Figura 20 - Cadastro de Fórmulas

A tela de cadastro de usuários (Figura 21) é simples, pode ser vinculado a um cadastro de pessoa existente ou não. No cadastro do usuário é indicado o tipo de permissão e o Câmpus da UTFPR que este terá acesso. Quando é realizado o cadastro de Pessoa, se o campo web for marcado para o recebimento do laudo, essa tela é carregada automaticamente após salvar o registro de Pessoa e vinculará o cliente a um usuário.

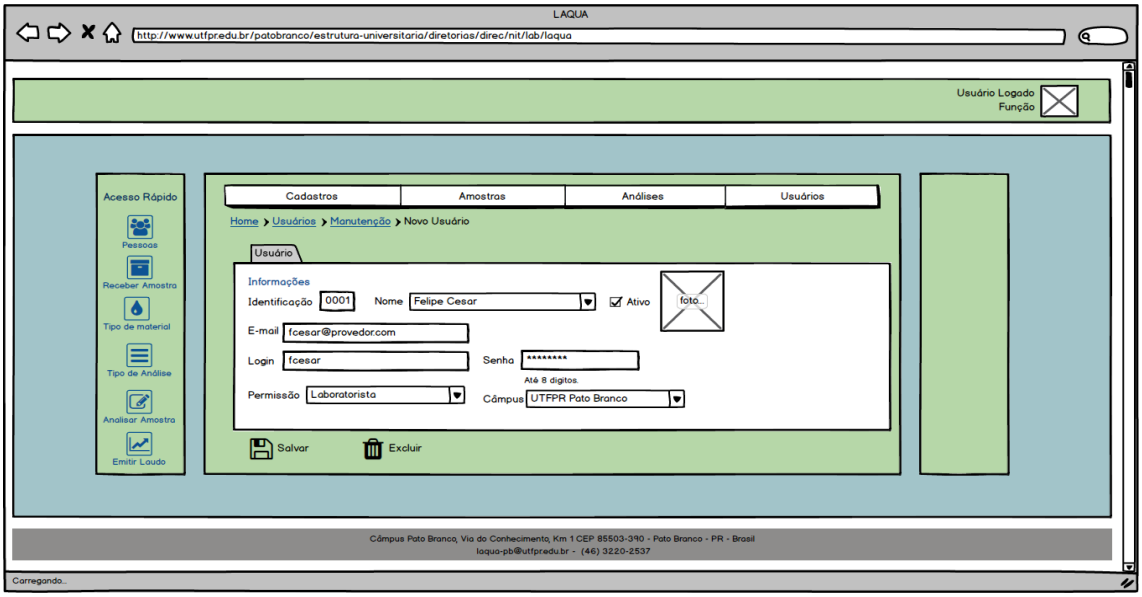


Figura 21 - Cadastro usuário

As Figuras 22 e 23 apresentam o diagrama de entidades e relacionamentos do banco de dados.

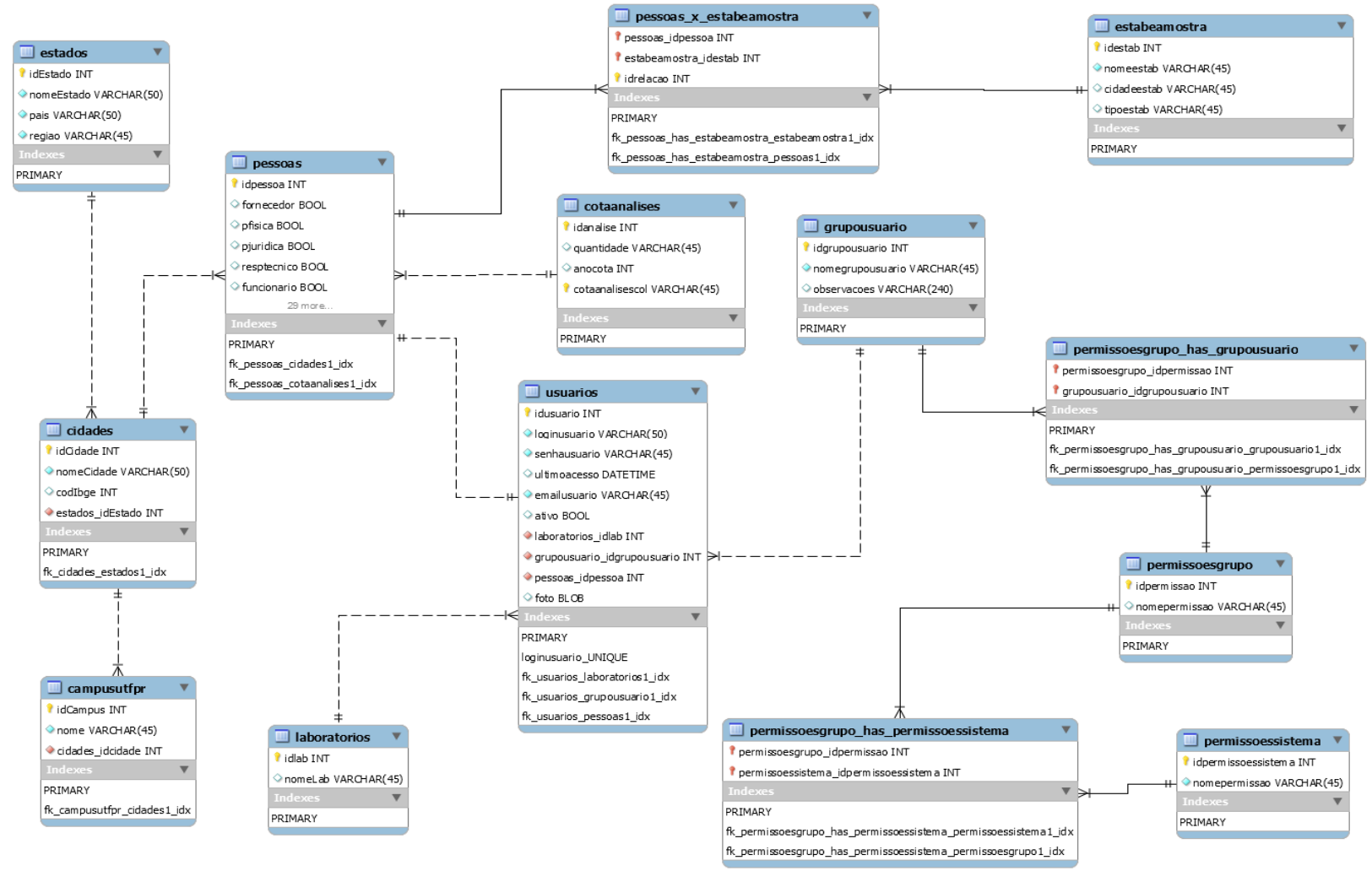


Figura 22 - Diagrama de entidades e relacionamentos do banco de dados

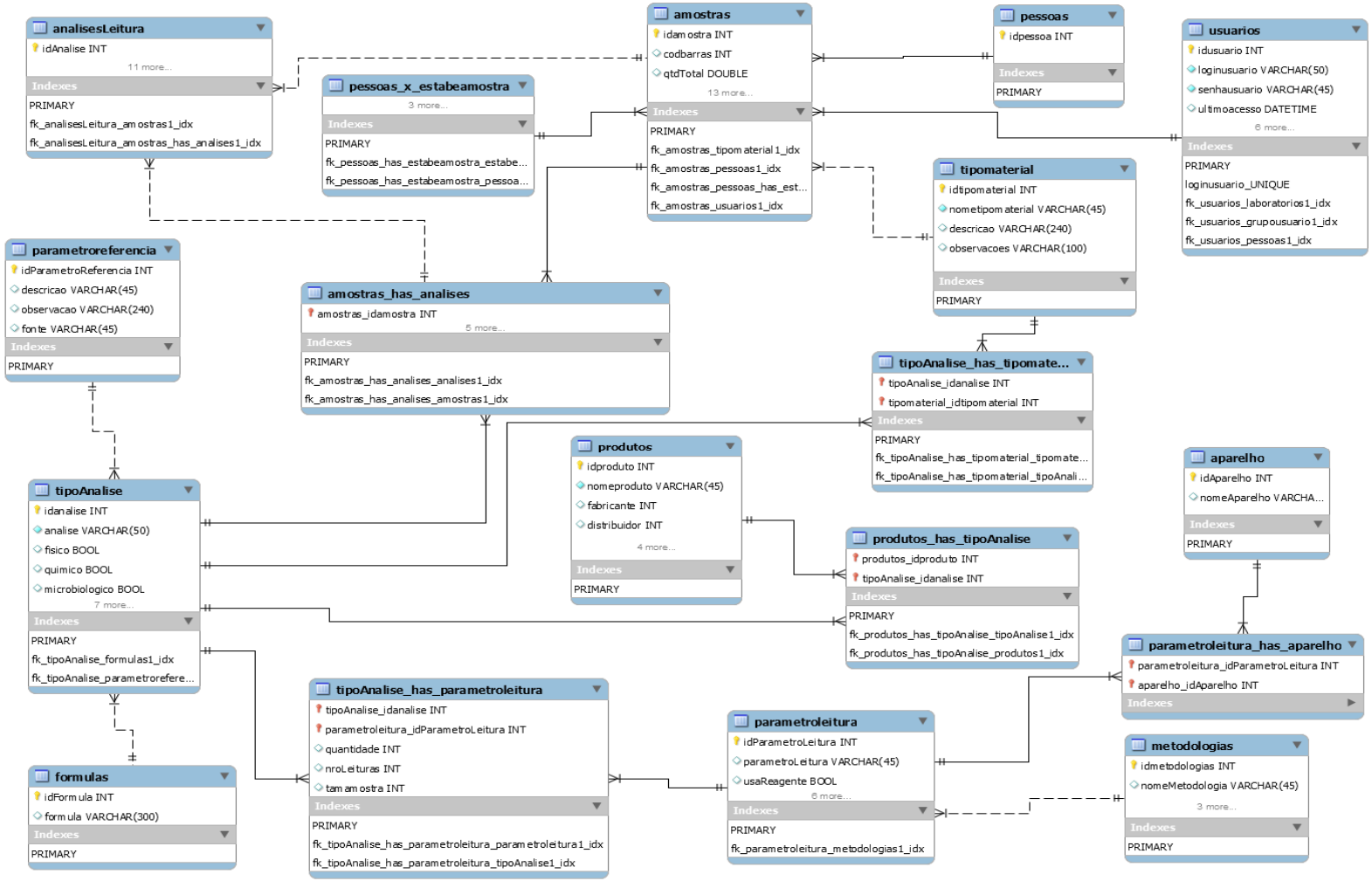


Figura 23 - Diagrama de entidades e relacionamentos do banco de dados

A seguir, os campos de cada uma das tabelas apresentadas nos diagramas das Figuras 22 e 23 são descritos. O digrama de entidades e relacionamentos do banco de dados foi apresentado em duas figuras para facilitar a visualização.

O Quadro 10 apresenta os campos da Tabela de Estados.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idEstado	Numérico	Não	Sim	Não	
nomeEstado	Texto	Não	Não	Não	
pais	Texto	Não	Não	Não	
regiao	Texto	Não	Não	Não	

Quadro 10 – Campos da tabela estados

Os campos da Tabela Cidades são apresentados no Quadro 11.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idCidade	Numérico	Não	Sim	Não	
nomeCidade	Texto	Não	Não	Não	
estados_idEstado	Numérico	Não	Não	Sim	

Quadro 11 – Campos da tabela cidades

Os campos da Tabela Câmpus da UTFPR, apresentados no Quadro 12, armazenam dados relacionados ao Câmpus da Universidade. Esses dados permitem utilizar um sistema centralizado, mas com acesso definido por Câmpus.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idCampus	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
cidades_idCidade	Numérico	Não	Não	Sim	

Quadro 12 – Campos da tabela campusutfpr

Os campos da Tabela de Pessoas são apresentados no Quadro 13.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idpessoa	Numérico	Não	Sim	Não	
fornecedor	lógico	Sim	Não	Não	
pfisica	lógico	Sim	Não	Não	
pjuridica	lógico	Sim	Não	Não	
resptecnico	lógico	Sim	Não	Não	
funcionario	lógico	Sim	Não	Não	

crq	Texto	Sim	Não	Não	
regiao	Texto	Não	Não	Não	
nome	Texto	Não	Não	Não	
profissao	Texto	Não	Não	Não	
cpf	Texto	Não	Não	Não	
rg	Texto	Sim	Não	Não	
cadpro	Texto	Sim	Não	Não	
Pessoascol	Texto	Sim	Não	Não	
razaosocial	Texto	Sim	Não	Não	
cpfcnpj	Texto	Não	Não	Não	
Inscricaoest	Texto	Sim	Não	Não	
Obseervacoes	Texto	Sim	Não	Não	
rua	Texto	Não	Não	Não	
Numero	Texto	Sim	Não	Não	
complemento	Texto	Sim	Não	Não	
bairro	Texto	Sim	Não	Não	
cep	Texto	Sim	Não	Não	
celular	Texto	Sim	Não	Não	
email	Texto	Sim	Não	Não	
fax	Texto	Sim	Não	Não	
telefonecom	Texto	Sim	Não	Não	
telefoneres	Texto	Sim	Não	Não	
telefonecontato	Texto	Sim	Não	Não	
nomecontato	Texto	Sim	Não	Não	
cidades_idcidades	Numérico	Não	Não	sim	
cotanalise	Numérico	Sim	Não	Não	
cotaanalises_idanalise	Numérico	Sim	Não	sim	
cotaanalises_ cotaanalisescol	Texto	Sim	Não	Não	

Quadro 13 – Campos da tabela pessoas

Os campos da Tabela Laboratórios são apresentados no Quadro 14.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idlab	Numérico	Não	Sim	Não	
nomeLab	Texto	Não	Não	Não	

Quadro 14 – Campos da tabela laboratorios

O Quadro 15 apresenta os campos da Tabela Usuários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idusuario	Numérico	Não	Sim	Não	
loginusuario	Texto	Não	Não	Não	
senhausuario	Texto	Não	Não	Não	
ultimoacesso	data	Não	Não	Não	
emailusuario	Texto	sim	Não	Não	
ativo	Lógico	Não	Não	Não	
laboratórios_idlab	Numérico	Não	Não	Sim	
grupousuarios-idgrupousuario	Numérico	Não	Não	Sim	
pessoa_idpessoa	Numérico	Não	Não	Sim	
foto	Imagem	Sim	Não	Não	

Quadro 15 – Campos da tabela usuarios

Os campos da Tabela Grupos de Usuários são apresentados no Quadro 16.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idgrupousuario	Numérico	Não	Sim	Não	
nomegrupousuario	Texto	Não	Não	Não	
observacoes	Texto	Sim	Não	Não	

Quadro 16 – Campos da tabela grupousuario

Os campos da Tabela de Permissões de Grupos de Usuários são apresentados no Quadro 17.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idpermissao	Numérico	Não	Sim	Sim	
nomepermissao	Texto	Não	Sim	Sim	

Quadro 17 – Campos da tabela permissoesgrupo

Os campos da Tabela associação de Permissões a Grupos de Usuários são apresentados no Quadro 18.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
Permissoesgrupo_ idgrupousuario	Numérico	Não	Sim	Sim	
Grupousuarios-idgrupousuario	Numérico	Não	Sim	Sim	

Quadro 18 – Campos da tabela permissoes_has_grupousuario

Os campos da Tabela de Permissões do Sistema no Quadro 19.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idpermissaosistema	Numérico	Não	Sim	Sim	
nomepermissao	Texto	Não	Sim	Sim	

Quadro 19 – Campos da tabela permissoessistema

Os campos da Tabela associação de Permissões de Grupos de Usuários às Funcionalidades do Sistema são apresentados no Quadro 20.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
Permissoesgrupo_ idpermissao	Numérico	Não	Sim	Sim	
permissaosistema-idpermissaosistema	Numérico	Não	Sim	Sim	

Quadro 20 – Campos da tabela permissoesgrupo_has_permissoessistema

Os campos da Tabela Cotas de Análise são apresentados no Quadro 21.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idanalise	Numérico	Não	Sim	Não	
quantidade	Texto	Sim	Não	Não	
anacota	int	Sim	Não	Não	
cotanalisecol	Texto	Não	Não	Sim	

Quadro 21 – Campos da tabela cotaanalise

Os campos da Tabela Estabelecimentos são apresentados no Quadro 22.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idestab	Numérico	Não	Sim	Não	
nomeestab	Texto	Não	Não	Não	
cidadeestab	Numérico	Não	Não	Sim	
tipoestab	Numérico	Não	Não	Sim	

Quadro 22 – Campos da tabela estabeleamostra

Os campos da Tabela de relacionamento entre Pessoas e Estabelecimentos são apresentados no Quadro 23.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
iderlacao	Numérico	Não	Sim	Não	
idstab	Texto	Não	Não	Não	

pessoas_idpessoa	Numérico	Não	Não	Sim	
estabeamostra_ idstab	Numérico	Não	Não	Sim	

Quadro 23 – Campos da tabela pessoa_x_estabeamostra

Os campos da Tabela de Fornecedores estão no Quadro 24.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idfornecedor	Numérico	Não	Sim	Não	
idcidade	Numérico	Sim	Não	Sim	Pato Branco
idestado	Numérico	Sim	Não	Sim	Paraná
nome	Texto	Não	Não	Não	
razaosocial	Texto	Sim	Não	Não	
inscricaoestadual	Texto	Sim	Não	Não	
cpf_cnpj	Texto	Não	Não	Não	
endereço	Texto	Não	Não	Não	
cep	Texto	Sim	Não	Não	
país	Texto	Não	Não	Não	Brasil
telefone	Texto	Não	Não	Não	
telefone celular	Texto	Não	Não	Não	
fax	Texto	Sim	Não	Não	
email	Texto	Não	Não	Não	
pessoacontato	Texto	Sim	Não	Não	

Quadro 24 – Campos da tabela fornecedores

O Quadro 25 apresenta os campos da Tabela de Amostras.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idamostra	Numérico	Não	Sim	Não	
codbarras	Texto	Sim	Não	Não	
qtdeTotal	Numérico	Sim	Nao	Sim	
subtipo	Numérico	Sim	Não	Sim	
datacoleta	Numérico	Não	Não	Sim	
temperatura	Data	Não	Não	Não	
coletor	Data	Não	Não	Não	
dataentrada	Data	Não	Não	Não	
dataentrega					
recebedor					

observacoes					
tipomaterial_ idtipomaterial					
Pessoas_idpessoa					
Pessoas_has_estabeamostra_ idestabe					
Usuarios_idusuario					

Quadro 25 – Campos da tabela amostras

Os campos da Tabela de Parâmetros de Leitura estão apresentados no Quadro 26.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idParametroLeitura	Numérico	Não	Sim	Não	
parametroLeitura	Texto	Não	Não	Não	
usaReagente	Texto	Sim	Não	Não	
leituraAparelho	Numérico	Não	Não	Sim	
peso					
volume					
unMedida					
descricao					
Metodologias_ idmetodologia					

Quadro 26 – Campos da tabela parametroleitura

Os campos da Tabela relacionamentos entre Parâmetros de Leitura e Aparelhos estão apresentados no Quadro 27.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
ParametroLeitura_ idParametroLeitura	Numérico	Não	Sim	Não	
Aparelho_idAparelho	Texto	Não	Não	Não	

Quadro 27 – Campos da tabela parametroleitura_has_aparelho

Os campos da Tabela de relacionamento entre Amostras e Análises estão apresentados no Quadro 28.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
amostras_idamostra	Numérico	Não	Sim	Não	
amostras_pessoas_ idamostra	Texto		Não	Não	
amostras_pessoas_has_ estabeamostra_pessoas_ idpessoa	Texto		Não	Não	

amostras_pessoas_has_ estabeamostra_idestabe					
Amostras_usuarios_ idusuario					
Analises_idanalises					

Quadro 28 – Campos da tabela amostras_has_analises

No Quadro 29 estão os campos da Tabela de Produtos.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idproduto	Numérico	Não	Sim	Não	
nomeproduto	Texto	Não	Não	Não	
fabricante	Texto	Não	Não	Não	
distribuidor	Texto	Não	Não	Não	
produtoscol	Texto	Não	Não	Não	
tipofabricado	Lógico	Sim	Não	Sim	
Tipocomprado	Lógico	Sim	Não	Sim	
Procedimentos	Texto	Não	Não	Não	

Quadro 29 – Campos da tabela produtos

O Quadro 30 apresenta os campos da Tabela de Fórmulas.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idFormula	Numérico	Não	Sim	Não	
formula	Texto	Não	Não	Não	
tabela	Texto	Não	Não	Não	
campo	Texto	Sim	Não	Não	

Quadro 30 – Campos da tabela formulas

O Quadro 31 apresenta os campos da Tabela de Aparelhos.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
idAparelho	Numérico	Não	Sim	Não	
nomeAparelho	Texto	Não	Não	Não	

Quadro 31 – Campos da tabela aparelhos

No Quadro 32 estão os campos da Tabela de Análises realizadas.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
IdAnalise	Numérico	Não	Sim	Não	
amostra_idamostra	Numérico	Não	Não	Sim	

amostras_pessoas_ idpessoa	Numérico	Não	Não	Sim	
amostras_pessoas_ hasestabe_amostra_ pesoas_idpessoa	Texto	Sim	Não	Sim	
amostras_pessoas_ hasestabe_amostra_ estabeamostra_idestab	Texto	Sim	Não	Sim	
Amostras_has_analise_ Amostras_has_pessoas_ Hás_estabeamostra_pessoas_ idpessoa					
Amostras_has_analise_ Amostras_has_pessoas_ Hás_estabeamostra_idstabe					
amostras_usuarios_ idusuario	Texto	Não	Não	Não	
Amostras_has_analises _analises_idanalise	Texto	Não	Não	Não	

Quadro 32 – Campos da tabela analisesLeitura

Os campos da Tabela de Tipo de Análise estão no Quadro 33.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**
IdAnalise	Numérico	Não	Sim	Não	
analise	Texto	Não	Não	Não	
fisico	Lógico	Sim	Não	Não	
quimica	Numérico	Não	Não	Sim	
Bromatologica					
Microbiológica					
entregaDe					
entregaA					
observaacoes					
valor					
Formulas_idFormula					
Parametroreferencia_ idParametroReferencia					

Quadro 33 – Campos da tabela tipoAnalise

A Tabela de relacionamento entre Pessoas e estabelecimentos da Amostra tem seus campos apresentados no Quadro 34.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
pessoas_idpessoa	Numérico	Não	Não	sim	
estabelamostra_idestab	Numérico	Não	Não	Sim	
idrelacao	Numérico	Não	Sim	Não	

Quadro 34 – Campos da tabela pessoas_has_estabeamostra

Os campos da Tabela de Metodologias são apresentados no Quadro 35.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idmetodologias	Numérico	Não	Sim	Não	
nomeMetodologia	Texto	Não	Não	Não	
fonte	texto	Sim	Não	Não	
descProcedimento	texto	Sim	Não	Não	
Observacoes	texto	Sim	Não	Não	

Quadro 35 – Campos da tabela metodologias

Os campos da Tabela de Parâmtros de Referência estão no Quadro 36.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
idParametroReferencia	Numérico	Não	Sim	Não	
descricao	Texto	Não	Não	Não	
fonte	texto	Sim	Não	Não	
observacao	texto	Sim	Não	Não	
fonte	texto	Sim	Não	Não	

Quadro 36 – Campos da tabela parametrosreferencia

Os campos da Tabela de Tipo de Material estão no Quadro 37.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
Idtipomaterial	Numérico	Não	Sim	Não	
nometipomaterial	Texto	Não	Não	Não	
descricao	texto	Sim	Não	Não	
observacao	texto	Sim	Não	Não	

Quadro 37 – Campos da tabela tipomaterial

O Quadro 38 apresenta os campos da Tabela de relacionamento entre Tipo de Análise e Parâmetros de Leitura.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
dtipoanalise_idanalise	Numérico	Não	Sim	Não	
parametroleitura_ idParametroLeitura	Numérico	Não	Não	Sim	
quantidade	Numérico	Sim	Não	Não	
nroLeituras	Numérico	Sim	Não	Não	
tamamostra	Numérico				

Quadro 38 – Campos da tabela tipoanalise_has_parametroleitura

O Quadro 39 apresenta os campos da Tabela de relacionamento entre Tipo de Análise e Tipo de Material.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão
dtipoanalise_idanalise	Numérico	Não	Sim	Sim	
tipomaterial_ idParametroLeitura	Numérico	Não	Sim	Sim	

Quadro 39 – Campos da tabela tipoanalise_has_tipomaterial

4.3 APRESENTAÇÃO DO SISTEMA

Para exemplificar do uso das tecnologias, citadas no capitulo 3 (item 3.1) a seguir são apresentados alguns trecho de código de algumas telas e classes do sistema.

A Figura 24 mostra a tela de login da aplicação. Nessa tela o usuário selecionará o laboratório que deseja acessar e informará respectivamente o login e senha de acesso. O campo de autenticação chamado “Laboratório” foi incluso para possibilitar que futuramente outros Câmpus da UTFPR possam utilizar o mesmo sistema. A base de dados é única, mas as entidades terão um atributo identificador com o código do laboratório. Desse modo, um determinado usuário com permissão para acessar o laboratório Laqua do Câmpus Pato Branco, não poderá, por exemplo, logar e acessar os dados do laboratório do Câmpus Curitiba.

Figura 24 - Tela de login da aplicação

Se o usuário informar os dados incorretos para o login ou para a senha, ou mesmo, informar corretamente, mas eles forem inválidos para o laboratório selecionado, a aplicação limpa os campos e informa uma mensagem de erro (Figura 25).

Figura 25 - Mensagem de erro na autenticação do usuário

No caso dos dados de login estarem informados corretamente e válidos para o laboratório selecionado, a tela inicial da aplicação é carregada, Figura 26.

Figura 26 - Página inicial da aplicação

O leitaute padrão do sistema é composto por um rodapé estático com os endereços para contato com o laboratório, ao lado esquerdo possui um menu expansível (Figura 27). No centro está a área de conteúdo dinâmico, e por fim, a área do cabeçalho, que como sugerido no protótipo, carregará uma foto (armazenada previamente no cadastro de usuário) e informações como a função do usuário e o laboratório.

Figura 27 - Menu expandido

A Figura 28 mostra o *grid* no qual serão carregados dados das pessoas cadastradas no sistema. Esse *grid* permite filtrar pessoas por dados específicos, como Tipo (Física, Jurídica etc.) pelo código (ID), pelo nome e pelo *status*. A barra de menu permite incluir um novo registro, editar um registro existente ou excluir possibilita também exportar os dados do *grid* para o Excel.

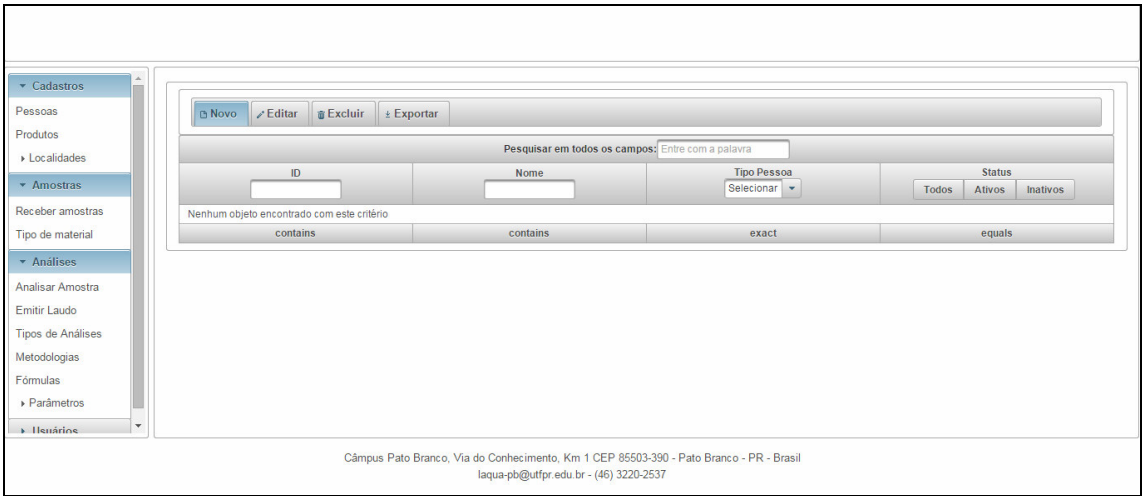


Figura 28 - Grid de Dados

A Figura 29 mostra a aba Dados Pessoais do formulário de Cadastro de Pessoas. A ideia é que os campos deste formulário sejam todos desabilitados no carregamento da página. A habilitação dos campos para preenchimento se dará de acordo com a marcação de um ou mais *checkbox* dos tipos de Pessoa (Funcionário, Física, Jurídica, etc.).

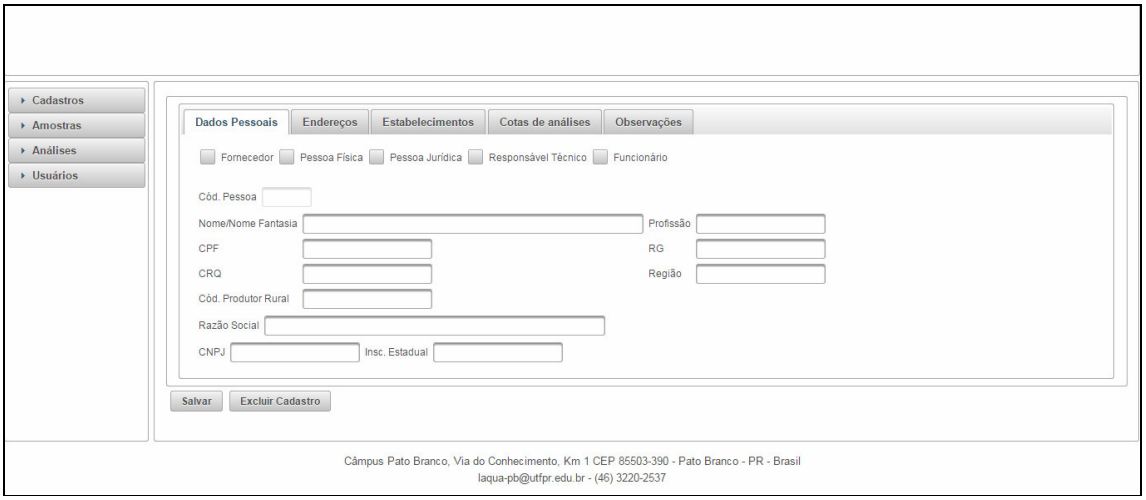


Figura 29 - Cadastro de Pessoa, aba Dados Pessoais

A Figura 30 mostra o formulário para fazer o recebimento de amostras. Com as opções para geração de código de barras e impressão deste código. O campo Solicitante utiliza um recurso de *AutoComplet*e da tecnologia Primefaces. Essa tecnologia permite que ao ser iniciada a digitação no campo sejam listados os dados do cadastro de pessoa compatível com o que está sendo informado.

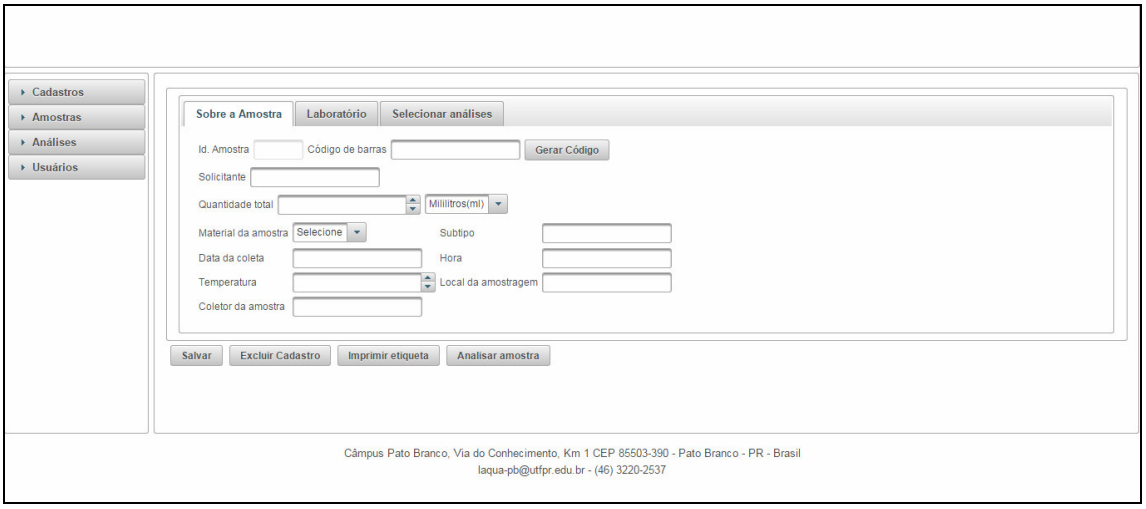


Figura 30 - Formulário para Receber Amostra, aba Sobre a Amostra

Os campos de data e hora são implementados com o componente de input *Calendar* do Primefaces. Esse componente abre um *popup* do tipo *date* para seleção da data e um *popup* do tipo *time* pra seleção da hora. A Figura 31 apresenta esse componente.

Figura 31 - Componente *Calendar*

4.4 IMPLEMENTAÇÃO DO SISTEMA

Para implementação do sistema foi utilizado a IDE de desenvolvimento NetBeans versão 8.02 e a linguagem de programação Java (JDK 1.8.0), associada com outras tecnologias baseadas na mesma linguagem, como mostra o Quadro 1 das ferramentas e tecnologias. Para a construção e o gerenciamento do projeto foi utilizada a ferramenta Maven. A Figura 32 mostra a estrutura geral da codificação da aplicação.

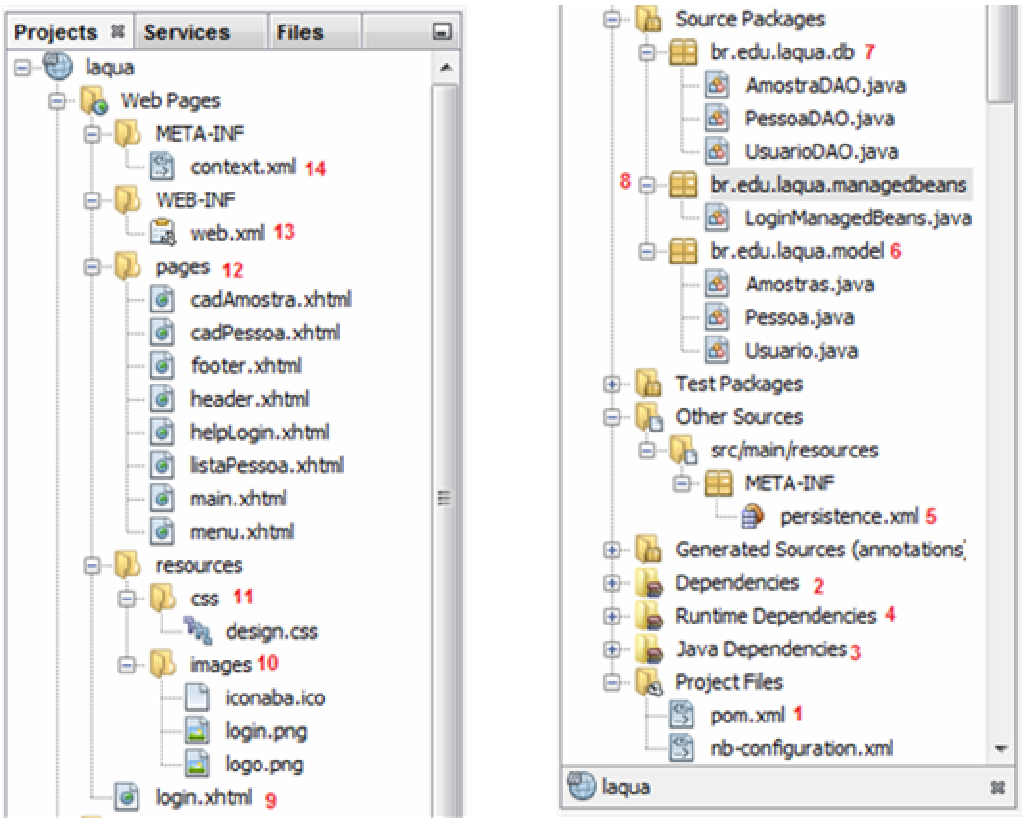


Figura 32 - Estrutura e organização do código do projeto

Explicação dos itens numerados na Figura 32:

1 - Arquivo pom.xml: *Project Object Model* (POM). É a unidade básica de configuração do Maven que fica na raiz do projeto. Ele declara a estrutura, as dependências e as características do projeto;

2 - 3 e 4: Diretórios de armazenamento das dependências, repositórios e *plugins* do projeto indicadas no arquivo pom.xml;

5 - persistence.xml: arquivo de persistência dos dados. Contém os dados de conexão com o banco, como o *drive*, usuário, senha e *Uniform Resource Locator* (URL), bem como as propriedades do Hibernate;

6 - 7 e 8: Pacotes de implementações das classes Java;

9 - login.xml: arquivo *index* do projeto ou *home* da aplicação;

10 - Diretório de recursos de imagens: contém as imagens e ícones utilizados na aplicação;

11 - Diretório de recursos css: contém os arquivos de configuração de estilo das páginas;

12 - pages: Diretório de páginas da aplicação;

13 - web.xml: Descritor das características da aplicação, como mapeamento dos *servlets*.

14 - context.xml: arquivo de contexto da aplicação, contém as configurações das conexões.

Após criar o arquivo do projeto no NetBeans foi configurado o *container* de *servlets* Apache Tomcat e os seguintes arquivos de configuração do projeto:

- Arquivo pom.xml: é um arquivo extenso e construído manualmente pois é nele que são indicadas as dependências, *plugins*, artefatos e repositórios desejo usar no projeto.

A Listagem 1, comentada, é o cabeçalho do pom.xml e identifica as propriedades que o projeto vai obedecer.

```
<Project xmlns=http://maven.apache.org/POM/4.0.0
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion> //Versão do Maven
  <groupId>br.edu</groupId> //Pacote da aplicação
  <artifactId>laqua</artifactId> //Nome do artefato
  <version>1.0-SNAPSHOT</version>//Versão da aplicação
  <packaging>war</packaging> //War indica que esta é uma aplicação web
  <name>laqua</name> //Nome do projeto
  <properties>
    <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>     //Propriedades do projeto, vai seguir o Encoding UTF-8
```

Listagem 1 - pom.xml

Abaixo do cabeçalho seguem as *tags* das dependências do projeto, assim que as *tags* são escritas e o arquivo salvo, o Maven se encarrega de fazer o *download*. Exemplo de dependência usada no projeto é apresentada na Listagem 2.

```
<dependency>
        <groupId>org.primefaces</groupId>
        <artifactId>primefaces</artifactId>
        <version>5.2</version>
 </dependency>
<repository>
        <id>prime-repo</id>
        <name>PrimeFaces Maven Repository</name>
        <url>http://repository.primefaces.org</url>
        <layout>default</layout>
 </repository>
 <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-war-plugin</artifactId>
          <version>2.6</version>
          <configuration>
             <failOnMissingWebXml>false</failOnMissingWebXml>
          </configuration>
</plugin>
```

Listagem 2 - pom.xml

O Arquivo persistence.xml é apresentado na Listagem 3. Esse arquivo também é configurado manualmente e nele são indicados os dados da conexão e as propriedades do J*ava Persistence API* (JPA) e do Hibernate.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="usuarios" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>br.edu.laqua.model.Usuario</class>
    <properties>
       <!--Dados da conexão -->
    1  <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
     2 <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/laqua?zeroDateTimeBehavior=convertToNull"/>
     3 <property name="javax.persistence.jdbc.user" value="admin"/>
     4 <property name="javax.persistence.jdbc.password" value="admin"/>
      <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
      <!-- propriedades do hibernate -->
     5 <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect"/>
     6 <property name="hibernate.show_sql" value="true"/>
     7<property name="hibernate.format_sql" value="true"/>
      <!-- atualiza o banco, gera as tabelas se for preciso -->
     8  <property name="hibernate.hbm2ddl.auto" value="update"/>
    </properties>
  </persistence-unit>
</persistence>
```

Listagem 3 - persistence.xml

Explicação das linhas numeradas na Listagem 3:

Propriedades do JPA:

1 driver – nome completo da classe do *driver* de conexão;

2 url – definição da localização do banco e o nome da base;

3 user – definição do usuário do banco de dados;

4 password – definição da senha do usuário;

Propriedades do Hibernate:

5 dialect – define qual variação do SQL será utilizada;

6 show_sql – define quando mostrar no *log* as consultas SQL geradas;

7 format_sql – formata o log das consultas SQL geradas;

8 hbm2ddl.auto – valida e exporta o esquema *Data Definition Language* ou *Data Description Language* (DDL) para a base de dados.

No persistence.xml a tag *provider* comunica que a implementação de JPA sendo utilizada é o Hibernate e os nomes das classes que definem as entidades a serem persistidas estão marcadas com a tag *class*. Como é permitido utilizar mais outras ferramentas *Object-Relational Mapping* (ORM) na mesma aplicação e é preciso distinguir umas das outras, cada unidade de persistência leva um nome. O nome dado para a unidade é definido pela propriedade *name* da tag *persistence-unit,* que neste caso é usuários.

O arquivo context.xml, conteúdo apresentado na Listagem 4, é lido pelo Tomcat quando a aplicação é iniciada pela primeira vez e disponibiliza ao Tomcat o gerenciamento da conexão com o banco de dados, usando o conceito de recurso (*resource*).

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true"
        path="/laqua">
  <Resource auth="Container"
          driverClassName="com.mysql.jdbc.Driver"
          maxActive="20"
          maxIdle="10"
          maxWait="-1"
          name="laqua"
          password="admin"
          type="javax.sql.DataSource"
          url="jdbc:mysql://localhost:3306/laqua?zeroDateTimeBehavior=convertToNull"
          username="admin"/>
</Context>
```

Listagem 4- context.xml

Propriedades da tag *Resource*:

- *auth:* foi preenchida como *Container*, deste modo é o servidor Tomcat, o responsável pelo processo de abertura e fechamento da conexão;

- *driverClassName*: *Namespace* do pacote do *driver Java Database Connectivity Technology* (JDBC) específico do banco de dados;

- *maxActive*: Número máximo de conexões do banco de dados no *pool* de conexões;

- *maxIdle*: Número máximo das conexões inativas;

- *maxWait*: Tempo máximo (em milissegundos) de espera por uma conexão, se este intervalo de parada for excedido, uma exceção é lançada. O "-1" indica para esperar indefinidamente.

- *name*: Definição do nome que servirá como forma de acesso ao nosso pool pela aplicação;

- *type*: Tipo da classe *dataSource*, que implementará um objeto com a conexão que será compartilhada por várias sessões da aplicação;

- *url*: Caminho de acesso ao serviço do banco de dados;

- *username* e *password*: Respectivamente usuário e senha para acessar o servidor de banco de dados;

A Listagem 5 apresenta o Arquivo web.xml que contém a configuração do projeto e suas respectivas *tags* comentadas com a função que cada uma exerce dentro do projeto.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee                http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd">
   <context-param>
      <param-name>javax.faces.PROJECT_STAGE</param-name>//Nome do parametro da aplicação
      <param-value>Development</param-value>//Valor do parametro
   </context-param>
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>//Nome do meu servlet
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>//Pacote da classe
      <load-on-startup>1</load-on-startup>
   </servlet>
<servlet-mapping>//Mapeamento do servlet
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>/faces/*</url-pattern>
</servlet-mapping>

<session-config>// indica em minutos o tempo de timeout de cada sessão criada, no caso 30 minutos;
      <session-timeout>
         30
   </session-timeout>
 </session-config>
<welcome-file-list> //arquivos que serão procurados pelo servidor quando executar no diretório da aplicação
      <welcome-file>faces/login.xhtml</welcome-file>
   </welcome-file-list>
 <error-page>//Controle de erros para a página de login
        <exception-type>javax.faces.application.ViewExpiredException</exception-type>
        <location>/login.xhtml</location>
   </error-page>
</web-app>
```

Listagem 5 - web.xml

Após a criação dos arquivos de configuração foi, efetivamente, iniciado o desenvolvimento da aplicação. A página de login, a login.xhtml, é apresentada na Listagem 6.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <h:head>
    <title>Laqua</title>
    <link rel="shortcut icon" type="image/x-icon" href="#{request.contextPath}/resources/images/iconaba.ico"
/>
    <link rel="stylesheet" type="text/css" href="#{request.contextPath}/resources/css/design.css"/>
  </h:head>

  <h:body>
    <p:layout fullPage="true" >
      <p:layoutUnit  style="border: none" position="south" size="100"  >
        <ui:include src="/pages/footer.xhtml"></ui:include>
      </p:layoutUnit>

      <p:layoutUnit style="border: none" position="center"  >
        <h:form >
          <h:panelGrid class="lucenter" id="basic" columns="2" cellpadding="80">
            <p:graphicImage value="/resources/images/logo.png" height="320"/>
            <p:panel class="lucenter" id="login">
              <p:messages id="messages" />
              <p:outputLabel value="Laboratório" /><br/>
              <p:selectOneMenu id="console" value="#{LoginMB.usuario.idLab}"
style="width:230px"><br/>
                <f:selectItem itemLabel="Selecione o Laboratório" itemValue="" />
                <f:selectItem itemLabel="Lab. UTFPR/PB" itemValue="1"/>
              </p:selectOneMenu><br/>
              <p:outputLabel for="nomeUsuario" value="Usuário" /> <br/>
              <p:inputText id="nomeUsuario" value="#{LoginMB.usuario.loginUsuario}" /> <br/>
              <p:outputLabel for="senha" value="Senha" /> <br/>
              <p:password id="senha" value="#{LoginMB.usuario.senhaUsuario}" /><br/>
              <h:link outcome="pages/helpLogin.xhtml" value="Não consigo acessar" /><br/>
              <p:commandButton value="Enviar" action="#{LoginMB.envia}" ajax="false"><br/>
              </p:commandButton>
            </p:panel>
          </h:panelGrid>
        </h:form>
      </p:layoutUnit>
    </p:layout>
  </h:body>
</html>
```

Listagem 6 - login.xhtml

A tela de login (Listagem 6), assim como as demais telas do sistema utilizam componentes JSF, mais especificadamente do Primefaces e Facelets, além de HTML. As *tags* iniciadas com *p:* indicam que este é um componente Primefaces, as *tags* iniciadas com *h:*

indica que é um componente HTML básico e as *tags* iniciadas com *ui:* indica que este é um componente Facelets.

No cabeçalho do arquivo é indicado na *tag* <link rel="stylesheet" type="text/css" href="#{request.contextPath}/resources/css/design.css"/> o path da folha de estilo. O uso do css deste modo, melhora a visualização do código e separa a formatação.

A tag <link rel="shortcuticon" type="image/x-icon" href="#{request.contextPath}/resources/images/iconaba.ico" /> indica o caminho do ícone que será carregado na aba da pagina no navegador (Figura 33). É apenas um detalhe, mas importante para a identidade do sistema.

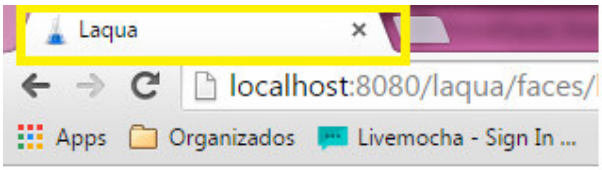


Figura 33 - Ícone da aba

O componente *p:selectOneMenu* possibilita ao usuário selecionar o laboratório que deseja logar-se, por questões de implementação, aqui foi utilizado um dado estático, com o valor 1 para o laboratório Laqua de Pato Branco. No caso da futura codificação completa do sistema, este componente será preenchido com um *List* de laboratórios vindo da Entidade 'Laboratório' do banco de dados.

Em seguida foi utilizado o componente de <*p:inputText*> para inserção do *login* do usuário e o componente <*p:password*> para a inserção da senha, este último componente é utilizado por questões de segurança, pois não mostra os caracteres digitados. O componente <*p:messages*> é utilizado caso ocorra algum erro no *login*, como por exemplo, se o usuário não existir, a mensagem de erro é tratada na classe LoginMB.java.

Ao selecionar o laboratório, informar o login e a senha, o usuário clica no botão Enviar para que os dados sejam validados.

Antes mesmo de criar a tela de *login* foi desenvolvido a classe que implementa os objetos que serão persistidos no banco de dados. Por questões de entendimento do projeto a explicação da tela de *login* foi realizada antes.

Como está sendo utilizado JPA, é necessário utilizar as anotações dessa especificação, para que seja feito o mapeamento objeto-relacional dos atributos da classe.

A Listagem 7 mostra o código da classe Usuario.java, que é o objeto que será persistido no banco de dados. A classe possui uma série de atributos. As principais anotações

utilizadas nela são: a @*Entity* no início da classe, ela indica que esta será uma Entidade do banco; o @*Id* define que o atributo é identificador único da classe; o @*GeneratedValue*, declara que este é um campo de autoincremento. Os campos com a anotação @*Column* definem que esses serão os atributos (coluna) no banco de dados, a anotação @*Temporal* é utilizada para atributos com o tipo Data.

Os campos iD e loginUsuario entre outros, por exemplo, são únicos, por isso o atributo *unique* tem o valor *true*, a senha e a data de ultimo acesso não precisam ser únicos, já que dois usuários diferentes podem acessar a aplicação ao mesmo tempo e também podem ter senhas iguais. Alguns dos dados do usuário são obrigatórios, por isso os dados obrigatórios possuem o atributo *nullable* como *false* e os não obrigatórios como *true*. Além da declaração dos atributos foram gerados os métodos de acesso a eles, os *getters* e *setters,* respectivamente.

```
package br.edu.laqua.model;

import java.io.Serializable;
import java.sql.Blob;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 *
 * @author Poliane
 */
@Entity
public class Usuario implements Serializable {

    @Id
    @GeneratedValue
     @Column(name = "idUsuario", nullable = false, unique = true)
     private int idUsuario;
     @Column(name = "loginUsuario", nullable = false, unique = true)
      private String loginUsuario;
@Column(name = "senhaUsuario", nullable = false, unique = false)
    private String senhaUsuario;

    @Column(name = "lastAccess", unique = true)
    @Temporal(TemporalType.DATE)
    private Date lastAccess;

    @Column(name = "emailUsuario", nullable = false)
    private String emailUsuario;

    @Column(name = "ativo", nullable = false)
    private Boolean ativo;
      @Column(name = "idGrpUsu")
    private int idGrpUsu;
```

```
    public String getEmailUsuario() {
        return emailUsuario;
    }

    public void setEmailUsuario(String emailUsuario) {
        this.emailUsuario = emailUsuario;
    }

    public Boolean getAtivo() {
        return ativo;
    }
public void setAtivo(Boolean ativo) {
        this.ativo = ativo;
    }

    public int getIdGrpUsu() {
        return idGrpUsu;
    }

    public void setIdGrpUsu(int idGrpUsu) {
        this.idGrpUsu = idGrpUsu;
    }

    public int getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(int idPessoa) {
        this.idPessoa = idPessoa;
    }
public Blob getFotoUsuario() {
        return fotoUsuario;
    }

    public void setFotoUsuario(Blob fotoUsuario) {
        this.fotoUsuario = fotoUsuario;
    }

    @Column(name = "idPessoa")
    private int idPessoa;

    @Column(name = "fotoUsuario")
    private Blob fotoUsuario;

    @Column(name = "idlab", nullable = false, unique = false)
    private int IdLab;

    public int getIdUsuario() {
        return idUsuario;
}

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getLoginUsuario() {
        return loginUsuario;
    }
```

```
    public void setLoginUsuario(String loginUsuario) {
        this.loginUsuario = loginUsuario;
    }

    public String getSenhaUsuario() {
        return senhaUsuario;
    }
public void setSenhaUsuario(String senhaUsuario) {
        this.senhaUsuario = senhaUsuario;
    }

    public Date getLastAccess() {
        return lastAccess;
    }

    public void setLastAccess(Date lastAccess) {
        this.lastAccess = lastAccess;
    }

    public int getIdLab() {
        return IdLab;
    }

    public void setIdLab(int IdLab) {
        this.IdLab = IdLab;
    }

}
```

Listagem 7 - Usuario.java

Depois de criar a classe do objeto que será persistido, foi definida a classe que faz as operações no banco de dados, como incluir, excluir e recuperar dados. A Listagem 8 mostra o código da classe UsuarioDAO que tem o método getUsuario recebendo os parâmetros loginUsuario, senha e laboratório. Nesse método é feita uma busca no banco e caso esse usuário exista, ele é retornado; caso contrário, é retornado *null*. Quando uma *query* não retorna resultados é lançada uma exceção do tipo *NoResultException*, por isso foi utilizado um *try/catch*. Caso essa exceção seja lançada, o método retorna *null*, indicando que não existe um usuário com o nome de usuário e senha passados como parâmetro.

```
package br.edu.laqua.db;

import br.edu.laqua.model.Usuario;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.NoResultException;
import javax.persistence.Persistence;

/**
 *
 * @author Poliane
 */
public class UsuarioDAO {

    private EntityManagerFactory factory = Persistence.createEntityManagerFactory("usuarios");
```

```
    private EntityManager em = factory.createEntityManager();

    public Usuario getUsuario(String loginUsuario, String senhaUsuario, int isLab) {
        try {
            Usuario usuario = (Usuario) em.createQuery("SELECT u from Usuario u where u.loginUsuario = :name
and u.senhaUsuario =  :senha and u.idLab= :idLab" ).setParameter("name", loginUsuario).setParameter("senha",
senhaUsuario).setParameter("idLab", idLab).getSingleResult();
            return usuario;
        } catch (NoResultException e) {
            return null;
        }
    }

    public boolean inserirUsuario(Usuario usuario) {
        try {
            em.persist(usuario);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean deletarUsuario(Usuario usuario) {
        try {
            em.remove(usuario);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

}
```

Listagem 8 - UsuarioDAO.java

Com o acesso aos dados implementado é possível desenvolver o ManagedBean do JSF, que é a classe que recebe os dados que o usuário envia pela tela da aplicação para receber os dados, é utilizado o atributo *usuário*. Com esses dados, o método envia getUsuario da classe UsuarioDAO, caso esse método retorne *null* é enviada uma tela de erro para o usuário e ele continua na mesma página. Caso contrário, o usuário é redirecionado para a tela *main.xhtml*. A Listagem 9 mostra o código dessa classe.

```
package br.edu.laqua.managedbeans;

import br.edu.laqua.db.UsuarioDAO;
import br.edu.laqua.model.Usuario;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.context.FacesContext;

/**
 *
 * @author Poliane
 */
@ManagedBean(name = "LoginMB")
@ViewScoped
public class LoginManagedBeans {

    private UsuarioDAO usuarioDAO = new UsuarioDAO();
    private Usuario usuario = new Usuario();

    public String envia() {
        usuario = usuarioDAO.getUsuario(usuario.getLoginUsuario(), usuario.getSenhaUsuario(),
usuario.getID());
        if (usuario == null) {
            usuario = new Usuario();
            FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário não encontrado!", "Erro no Login!"));
            return null;
        } else {
            return "/pages/main";
        }

    }

    public Usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }
}
```

Listagem 9- LoginMB.java

O arquivo *main.xhtml* (Listagem 10) é o arquivo *template* da aplicação, é este arquivo que é chamado após a autenticação do usuário. Esse arquivo foi construído com base no componente <*p:layoutUnit*> do Primefaces. Esse componente divide o leiaute da aplicação em quatro posições: *north, south, west* e *center*.

Para tornar mais fácil a manutenção da aplicação, essas posição serão respectivamente preenchidas com os dados vindos de arquivos específicos, são eles: header.xhtml arquivo que construirá o cabeçalho; footer.xhtml refere-se ao arquivo de composição do rodapé; menu.xhtml refere-se ao arquivo de conteúdo do menu lateral, este arquivo foi codificado com

conteúdo estático, ou seja, diretamente dentro do componente <*p:panelMenu* >, mas a ideia é coloca-lo em um arquivo xml a parte e apenas chama-lo no arquivo .xhtml para construção do menu, deste modo a manutenção do menu será mais fácil.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">

  <f:view contentType="text/html">
    <h:head>
      <f:facet name="first">
        <meta content='text/html; charset=UTF-8' http-equiv="Content-Type"/>
        <meta http-equiv="Cache-Control" content="no-cache" />
        <title> Laqua </title>
        <link rel="shortcut icon" type="image/x-icon"
href="#{request.contextPath}/resources/images/iconeaba.ico" />
        <link rel="stylesheet" type="text/css" href="#{request.contextPath}/resources/css/design.css"/>
      </f:facet>
    </h:head>

    <h:body>
      <p:layout fullPage="true">
        <p:layoutUnit position="north" size="100" resizable="true" closable="true" collapsible="true">
          <ui:include src="/pages/header.xhtml"/>
        </p:layoutUnit>
        <p:layoutUnit style="border: none" position="south" size="100" closable="true" collapsible="true">
          <ui:include src="/pages/footer.xhtml"/>
        </p:layoutUnit>
        <p:layoutUnit position="west" size="175" closable="true" collapsible="true">
          <ui:include src="/pages/menu.xhtml"/>
        </p:layoutUnit>
        <p:layoutUnit position="center">
          <ui:insert name="content"/> //Aqui digo o nome do unidade insert para invocação
        </p:layoutUnit>
      </p:layout>
    </h:body>
  </f:view>
</html>
```

Listagem 10 - Main.xhtml

O arquivo main.xhtml é a unidade central, foi configurada com o Facelet *ui:insert* e nomeado como "*content*" (poderia ser qualquer outro nome), pois este trecho da *unit* será preenchido de forma dinâmica, ou seja, será a área de trabalho da aplicação. O componente (*ui:insert*) recebe um nome porque cada componente que deva ser mostrado nesta parte da tela, terá que invocara-lo chamando por este nome. Ou seja, é o componente que diz onde ele será mostrado na tela (Listagem 11).

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition template="/pages/main.xhtml" //Deve ser composition e não um HTML, pois este .xhtml sera
parte de outra tela
        xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://xmlns.jcp.org/jsf/html"
        xmlns:f="http://xmlns.jcp.org/jsf/core"
        xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
        xmlns:p="http://primefaces.org/ui">

  <ui:define name="content">// Aqui digo em qual parte da tela este componente deve ser carregado quando
invocado
    <h:form class="formcadPessoa">
```

Listagem 11- Cabeçalho do arquivo cadPessoa.xhtml.

Para que a invocação do conteúdo seja feita e preenchida na região correta da tela, além de atribuir um *name* para o trecho da *unit* e ser invocada no componente, a unidade de conteúdo deve estar dentro da tag <*ui:composition*> e não de um <*html*> normal, além disso deve-se indicar na propriedade *tamplate* o endereço do *main.xhtml*. É deste modo que os conteúdos dos arquivos cadAmostra.xhtml, cadPessoa.xhtml e listaPessoa.xhtml serão construídos e invocados.

A lógica de organização apresentada até o momento (das classes Java) será a mesma para a implementação dos cadastros de Pessoa e Amostra, ou seja, com uma classe model para Pessoa e Amostra, bem como os respectivos Beans e Daos. Segue alguns códigos referente ao Cadastro de Pessoa.

Quando a tela do formulário do Cadastro de Pessoas é aberto, todos os campos estarão desabilitados. De acordo com tipo de pessoa marcado nos *checkbox*, campos específicos são acionados, por exemplo, ao marcar o *checkbox* Fornecedor os campos de Nome/Nome Fantasia, Razão Social, Cadastro Nacional da Pessoa Jurídica (CNPJ) e Inscrição Estadual da guia Dados Pessoais se tornarão editáveis, os demais permanecerão desabilitados. Essa configuração reduz a possibilidade de preenchimento incorreto do formulário. A Listagem 12 apresenta um trecho de código do cadastro de pessoas.

```
<h:panelGrid columns="2" cellspacing="5">
    <p:outputLabel value="Fornecedor"/>
    <h:selectBooleanCheckbox id="forn" value="#{PessoaMB.inLimiteAvaliacao}">
        <p:ajax listener="#{PessoaMB.limiteAvaliacao}" update="nomePessoa, rzsocial,inscestad,cnpj"/>
    </h:selectBooleanCheckbox>
</h:panelGrid>

<p:outputLabel value="Nome/Nome Fantasia"/>
<p:inputText id="nomePessoa" style="width: 400px" disabled="#{PessoaMB.mostrar}"
value="#{PessoaMB.pnome}"/>

<p:outputLabel value="Razão Social" />
```

```
<p:inputText id="rzsocial" style="width: 400px" disabled="#{PessoaMB.mostrar}"
value="#{PessoaMB.przsocial}"/>

<p:outputLabel value="CNPJ" />
<p:inputText id="cnpj" disabled="#{PessoaMB.mostrar}" value="#{PessoaMB.pcnpj}"/>

<p:outputLabel value="Insc. Estadual" />
<p:inputText id="inscestad" disabled="#{PessoaMB.mostrar}" value="#{PessoaMB.pinscest}"/>

<p:commandButton value="Salvar" actionListener="#{PessoaMB.salvar}"></p:commandButton>
```

Listagem 12 - Trecho do código da tela CadPessoa

A Listagem 12 mostra o exemplo de codificação para habilitar os campos quando o usuário selecionar o tipo de Pessoa Fornecedor. Para tornar mais fácil a codificação foi usado Ajax, e indicado na propriedade *update* os IDs dos campos que devem ser habilitados quando este *checkbox* for acionado. A mesma lógica é utilizada para os demais tipos de pessoas (física, jurídica, etc.).

A Listagem 13 mostra a lógica em Java para ativar os campos. Quando o usuário clica no componente é feito uma chamada para o MB dizendo “marquei/desmarquei” o *checkbox*, o MB recebe o dado e trata, enviando uma resposta para o Ajax e dizendo o que deve ser feito nos campos de *update*.

```
package br.edu.laqua.managedbeans;

import javax.faces.bean.ManagedBean;

/**
 *
 * @author Poliane
 */
@ManagedBean(name = "PessoaMB")
public class PessoaManagedBeans {

   private boolean inLimiteAvaliacao;

   private boolean mostrar = true;

   public void limiteAvaliacao() {
      if (inLimiteAvaliacao) {
         mostrar = false;
      } else {
         mostrar = true;
      }
   }

   public boolean isInLimiteAvaliacao() {
      return inLimiteAvaliacao;
   }

   public void setInLimiteAvaliacao(boolean inLimiteAvaliacao) {
      this.inLimiteAvaliacao = inLimiteAvaliacao;
   }
```

```
    public boolean isMostrar() {
        return mostrar;
    }

    public void setMostrar(boolean mostrar) {
        this.mostrar = mostrar;
    }

}
```

Listagem 13 - Trecho da classe java PessoaManagedBeans

Observando a Listagem 12, os componentes acessam a classe do Bean (quem faz controle da camada de visão) pelo nome passado pelo ManagedBean, no caso PessoaMB. Ele utiliza os *getters* e *setters* para atribuir ou exibir valores. O método para salvar os dados é invocado através do botão Salvar que contém a propriedade actionListener="#{PessoaMB.salvar}.

O método *PessoaMb.salvar* (Listagem 14) por sua vez aciona o método *inserir* do PessoaDAO (camada de persistência), passando por parâmetro o objeto pessoa.

```
Pessoa pessoa = new Pessoa();
public void salvar (ActionEvent actionEvent){
                    new PessoaDao().inserir(pessoa);
}
```

Listagem 14 - Método salvar da classe PessoaManagedBean

O método *inserir* do PessoaDAO é apresentado na Listagem 15.

```
package br.edu.laqua.db;
import org.hibernate.Criteria;
import org.hibernate.classic.Session;
import util.HibernateUtil;
/**
 *
 * @author Poliane
 */
public class PessoaDAO {
 private Session session;
 public void inserir(Pessoa pessoa){
                                session = HibernateUtil.getSessionFactory().openSession();
                                try{
                                        session.beginTransaction();
                                        session.save(pessoa);
                                        session.getTransaction().commit();

                                }finally{
                                        session.close();
                                }
        }
}
```

Listagem 15 - Método inserir da classe PessoaDAO

Na listagem 15 é possível verificar que o método inserir abre uma sessão no banco, cria uma nova transação, insere o objeto pessoa, comitta a inserção e encerra a sessão. A classe "PessoaDao" utilizou a API *Criteria*, do Hibernate, uma maneira de não utilizar nenhum código SQL para efetuar a persistência no banco.

Do mesmo modo como foi realizada a inserção no banco é realizada a exclusão e a listagem, ou seja, usando a API para não precisar gerir códigos SQL (Listagem 16).

```
public void excluir(Pessoa pessoa){
                        session = HibernateUtil.getSessionFactory().openSession();
                        try{
                                session = HibernateUtil.getSessionFactory().openSession();
                                session.beginTransaction();
                                session.delete(pessoa);
                                session.getTransaction().commit();

                        }finally{
                                session.close();
                        }
                }
```

Listagem 16 - Método excluir da classe PessoaDAO

Observando a Listagem 16 e 17, nota-se que assim como na inserção é preciso abrir uma sessão no banco, cria uma nova transação, dizer o que se deseja fazer com o objeto (inserir, excluir, retornar um *list*), na sequência faz *commit* e encerra a sessão.

```
@SuppressWarnings("unchecked")
                public List  listar(){
                        session = HibernateUtil.getSessionFactory().openSession();
                        try{
                                Criteria cri = session.createCriteria(Pessoa.class);
                                return cri.list();
                        }finally{
                                session.close();
                        }
                }
}
```

Listagem 17 - Método para retornar uma lista de objetos Pessoa

5 CONCLUSÃO

O uso do Balsamiq Mockups para o desenvolvimento do protótipo do sistema Laqua foi muito satisfatório, trata-se de uma tecnologia completa e simples que permitiu a modelagem da aplicação com riqueza de detalhes. A ferramenta possui diversos componentes para criação de protótipos *web*, *mobile* e *desktop*, além de permitir o *download* de componentes extras e a inclusão de imagens externas. A possibilidade de criação do projeto, integrando todas as telas, facilitou a compreensão do fluxo de trabalho que a implementação da aplicação deve contemplar.

Por se tratar de uma aplicação complexa, durante a realização de testes com o protótipo do Laqua pode-se perceber a necessidade de uma nova revisão, buscar simplificar e detalhar mais a geração dinâmica dos formulários. Com essa constatação, percebe-se o quanto o protótipo é útil, pois a alteração do protótipo é mais simples do que se tivesse que alterar a aplicação já codificada totalmente.

Além do protótipo, foi codificada uma pequena parte da aplicação com as ferramentas descritas no Capítulo 3 (item 3.1). O uso das tecnologias citadas foi satisfatório, com algumas ressalvas. O uso do Hibernate, apesar de "eximir" a implementação manual do banco de dados e aumentar a produtividade do trabalho, tornou a aplicação um pouco lenta, se comparada com os métodos convencionais de gerenciamento da camada de persistência.

O Primefaces demonstrou ser um *framework* completo, com seus muitos componentes fáceis de manusear e com uma documentação clara e compreensível, reduziu o tempo de trabalho na elaboração do layout da aplicação. Além de possuir muitos componentes, a maior parte deles já contempla o *design responsive* permitindo futuramente a adaptação do sistema para diferentes dispositivos.

O uso do Maven auxiliou consideravelmente na gestão do projeto e a controlar as dependências e repositórios do Java, do Primefaces e do *container* de *servelt*.

A IDE de desenvolvimento NetBeans, mostrou-se limitada, por se tratar de um projeto complexo com uso de diferentes tecnologias. A ferramenta demanda muito tempo para construir/reconstruir as classes, além de "travar" frequentemente. Por diversas vezes não conseguia executar o *Clean* e *Build*, sendo necessário apagar manualmente os arquivos .class para que o projeto pudesse ser reconstruído.

Quanto à elaboração do protótipo não houve dificuldades, pois o Balsamiq é uma ferramenta intuitiva. Quanto à codificação da aplicação, a falta de conhecimento aprofundado

e experiência com a tecnologia Java tornaram a codificação demorada, lenta se comparada com uma implementação realizada por um programador experiente.

Como trabalhos futuros destaca-se a implementação do restante das funcionalidades e a colocação do sistema em produção como fase de testes. Pela quantidade de funcionalidades e elevada complexidade de algumas delas é certo que ajustes na modelagem serão necessários.

REFERÊNCIAS

BOZZON, Alessandro; COMAI, Sara; FRATERNALI, Piero; CARUGHI, Giovanni T. **Conceptual modeling and code generation for rich internet applications**. In: International Conference on Web Engineering (ICWE), ACM Press, New York, 2006, p. 353–360.

BUSCHMANN, Frank; MEUNIER, Régine; ROHNERT, Hans; SOMMERLAD, Peter; STAHL, Michael. **Pattern-oriented software architecture - a system of patterns**. Nova York:John Wiley and Sons, Inc, 1996.

CETAL. **Análise físico-química**. Disponível em: <http://www.cetal.com.br/analises_fisico_quimico.asp?s=3&ss=2>. Acesso em: 11 jun. 2015

COMAI, Sara; CARUGHI, Giovanni T**. A behavioral model for rich internet applications**. In: Web Engineering, LNCS v.4607, 2007, p.364-369.

COOPER, Ezra; LINDLEY, Sam; WADLER, Philip; YALLOP, Jeremy. **Links: web programming without tiers**. In: Formal Methods for Components and Objects, LNCS, v.4709, 2007, p.266-269.

DUHL, Joshua. **Rich Internet Applications**. White paper. Technical report, IDC, November 2003.

FANZINI, Fernando. **Listagem de frameworks Java**. 2013. Disponível em: < http://imasters.com.br/linguagens/java/listagem-de-frameworks-java/ >. Acesso em: 27 set. 2014

FRANCO, Rebeca S. T. **Estudo comparativo entre frameworks Java para desenvolvimento de aplicações web: JSF 2.0, Grails e Spring Web**. Monografia de Especializção. Unversidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Tecnologia. Especialização em Tecnologia Java. Disponível em: < http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/492/1/CT_JAVA_VI_2010_16.PDF>. Aceso em: 27 set. 2014.

GRAILS. **Grails framework**. Disponível em: <https://grails.org/>. Acesso em: 0 out. 2014.

JAVASERVER FACES. **JavaServer Faces technology**. 2014. Disponível em: <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>. Acesso em: 20 out. 2014.

JOHNSON, Rod; et al. **Reference documentation – Spring Framework 3.0**. 2011. Disponível em: <http://static.springsource.org/spring/docs/3.1.0.M1/spring-framework-reference/pdf/ spring-framework-reference.pdf>. Acesso em: 20 out. 2014.

OLIVERIA, Eric C. M. O **Universo dos frameworks Java**. Disponível em: < http://www.linhadecodigo.com.br/artigo/758/o-universo-dos-frameworks-java.aspx>. Acesso em: 27 set. 2014.

ORACLE. **The Java EE 6 tutorial. Chapter 4 JavaServer Faces Technology**. 2013. Disponível em: <http://download.oracle.com/javaee/6/tutorial/doc/ bnaph.html>. Acesso em: 27 set. 2014.

PAGANINI, Silvio. JSF 2.0. Aprendendo JSF 2.0 com ScrumToys. **Java Magazine**, n. 78. 2014.

POWELL, Courtney; NAKAMURA, Keisuke; AKAMA, Kiyoshi**. Towards a formal behavioral model for rich internet applications**. International Conference on Computational Intelligence and Software Engineering (CiSE 2009), p, 1-5, 2009.

PRECIADO, Juan C.; LINAJE, Marino; SANCHEZ-FIGUEROA, Fernando; COMAI, Sara. **Necessity of methodologies to model rich internet applications**. In: 7th IEEE International Symposium on Web Site Evolution, 2005, p. 7–13.

PREE, Wolfgang. **Meta patterns - a means for capturing the essentials of reusable object-oriented design**. Tokoro, M and Pareschi, R. (eds). In: European Conference Object-Oriented Programming (*ECOOP)*, Springer-Verlag, 1994, p. 150-162.

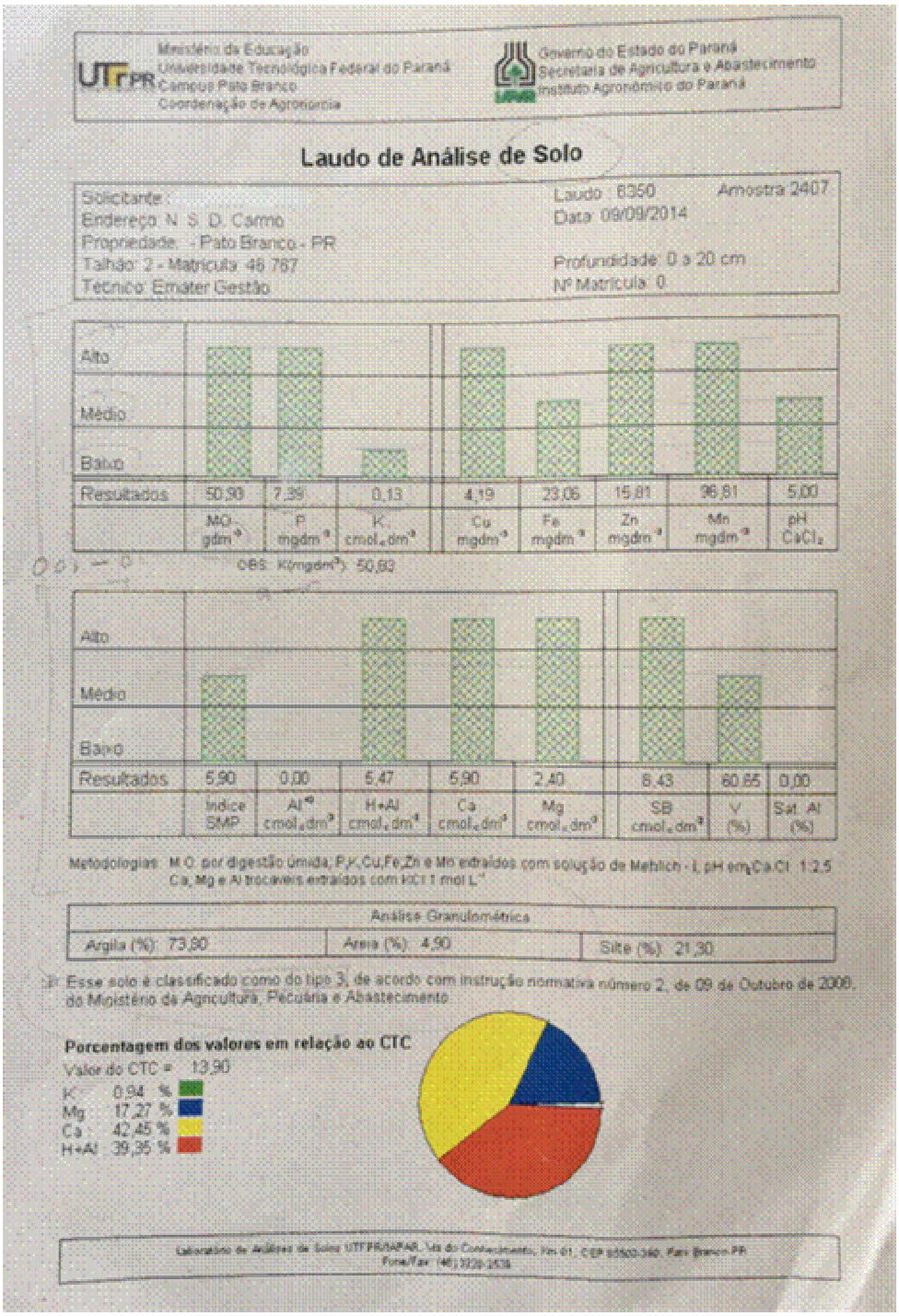
PRESSMAN, Roger. **Engenharia de software**. Rio de Janeiro: MacGraw-Hill, 2008.

SHAN, Tony C.; HUA, Winnie W. **Taxonomy of java web application frameworks.** IEEE International Conference on e-Business Engineering (ICEBE'06), p. 2006.

STEARN, Brent. **XULRunner: a new approach for developing rich internet applications**, IEEE Internet Computing, v.11, n.3, 2007, p. 67-73.

ANEXOS

ANEXO A - Exemplo de Laudo de análise de solo



Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Pato Branco
Coordenação de Agronomia

Governo do Estado do Paraná
Secretaria de Agricultura e Abastecimento
Instituto Agronômico do Paraná

Laudo de Análise de Solo

Solicitante :
Endereço: N. S. D. Carmo
Propriedade: - Pato Branco - PR
Talhão: 2 - Matrícula: 48.787
Técnico: Emater Gestão

Laudo: 8350 Amostra: 2407
Data: 09/09/2014

Profundidade: 0 a 20 cm
Nº Matrícula: 0

Resultados	50,93	7,39	0,13	4,19	23,06	15,81	96,81	5,00
	MO $g dm^{-3}$	P $mg dm^{-3}$	K $cmol_c dm^{-3}$	Cu $mg dm^{-3}$	Fe $mg dm^{-3}$	Zn $mg dm^{-3}$	Mn $mg dm^{-3}$	pH $CaCl_2$

OBS: K(mg dm⁻³): 50,93

Resultados	5,90	0,00	5,47	5,90	2,40	8,43	60,65	0,00
	Índice SMP	Al^{+3} $cmol_c dm^{-3}$	H+Al $cmol_c dm^{-3}$	Ca $cmol_c dm^{-3}$	Mg $cmol_c dm^{-3}$	SB $cmol_c dm^{-3}$	V (%)	Sat. Al (%)

Metodologias: M.O. por digestão úmida; P,K,Cu,Fe,Zn e Mn extraídos com solução de Mehlich - 1; pH em $CaCl_2$ 1:2,5
Ca, Mg e Al trocáveis extraídos com KCl 1 mol L^{-1}

Análise Granulométrica		
Argila (%): 73,80	Areia (%): 4,90	Silte (%): 21,30

Esse solo é classificado como do tipo 3, de acordo com instrução normativa número 2, de 09 de Outubro de 2008, do Ministério da Agricultura, Pecuária e Abastecimento.

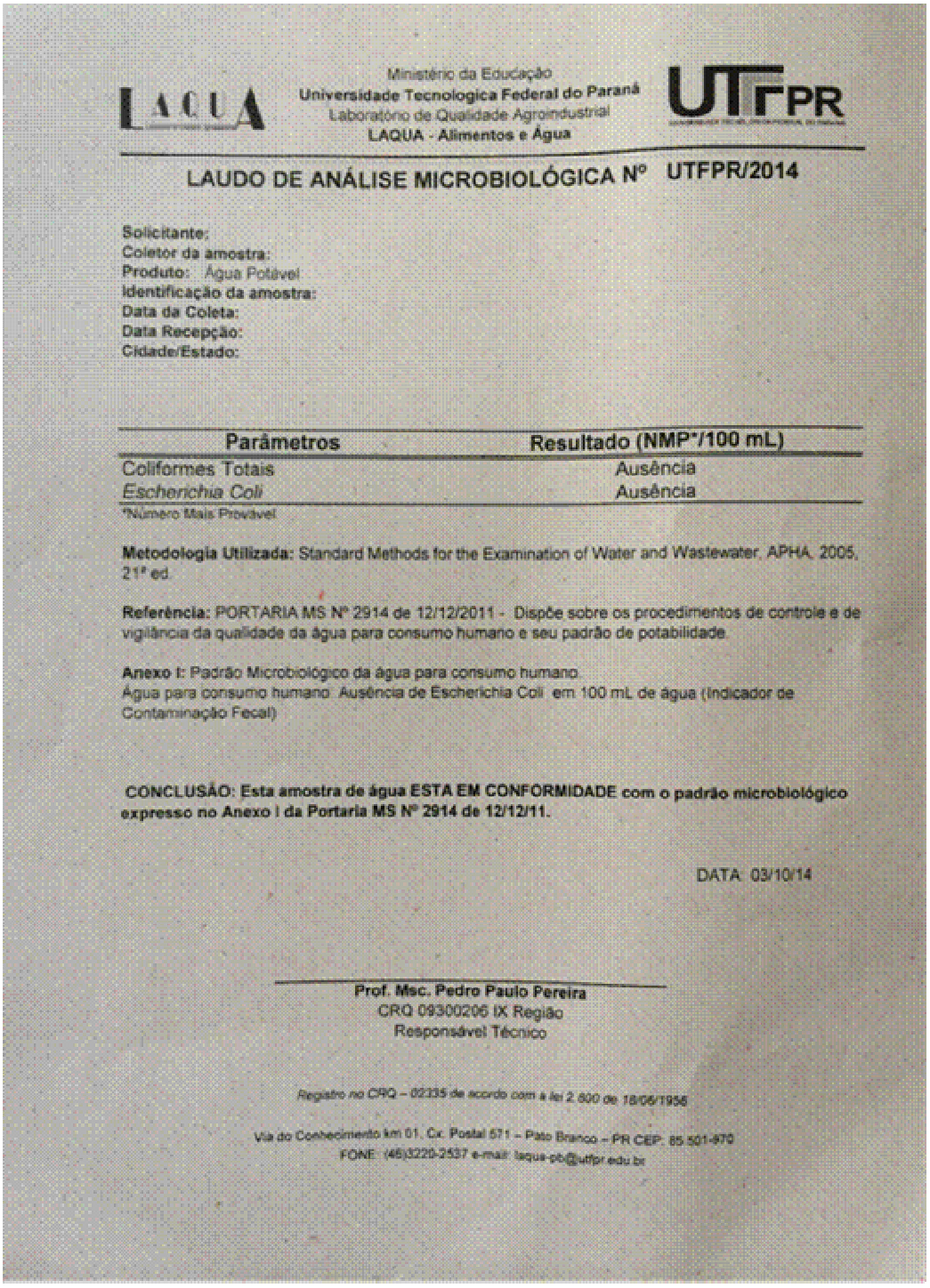
Porcentagem dos valores em relação ao CTC

Valor do CTC = 13,90

K: 0,94 %
Mg: 17,27 %
Ca: 42,45 %
H+Al: 39,35 %

Laboratório de Análises de Solos UTFPR/IAPAR, Via do Conhecimento, Km 01, CEP 85503-390, Pato Branco-PR
Fone/Fax: (46) 3220-2528

ANEXO B - Exemplo de Laudo de análise microbiológica



Ministério da Educação
Universidade Tecnologica Federal do Paraná
Laboratório de Qualidade Agroindustrial
LAQUA - Alimentos e Água

LAUDO DE ANÁLISE MICROBIOLÓGICA Nº UTFPR/2014

Solicitante:
Coletor da amostra:
Produto: Água Potável
Identificação da amostra:
Data da Coleta:
Data Recepção:
Cidade/Estado:

Parâmetros	Resultado (NMP*/100 mL)
Coliformes Totais	Ausência
Escherichia Coli	Ausência

*Número Mais Provável

Metodologia Utilizada: Standard Methods for the Examination of Water and Wastewater, APHA, 2005, 21ª ed.

Referência: PORTARIA MS Nº 2914 de 12/12/2011 - Dispõe sobre os procedimentos de controle e de vigilância da qualidade da água para consumo humano e seu padrão de potabilidade.

Anexo I: Padrão Microbiológico da água para consumo humano.
Água para consumo humano: Ausência de Escherichia Coli em 100 mL de água (Indicador de Contaminação Fecal)

CONCLUSÃO: Esta amostra de água ESTA EM CONFORMIDADE com o padrão microbiológico expresso no Anexo I da Portaria MS Nº 2914 de 12/12/11.

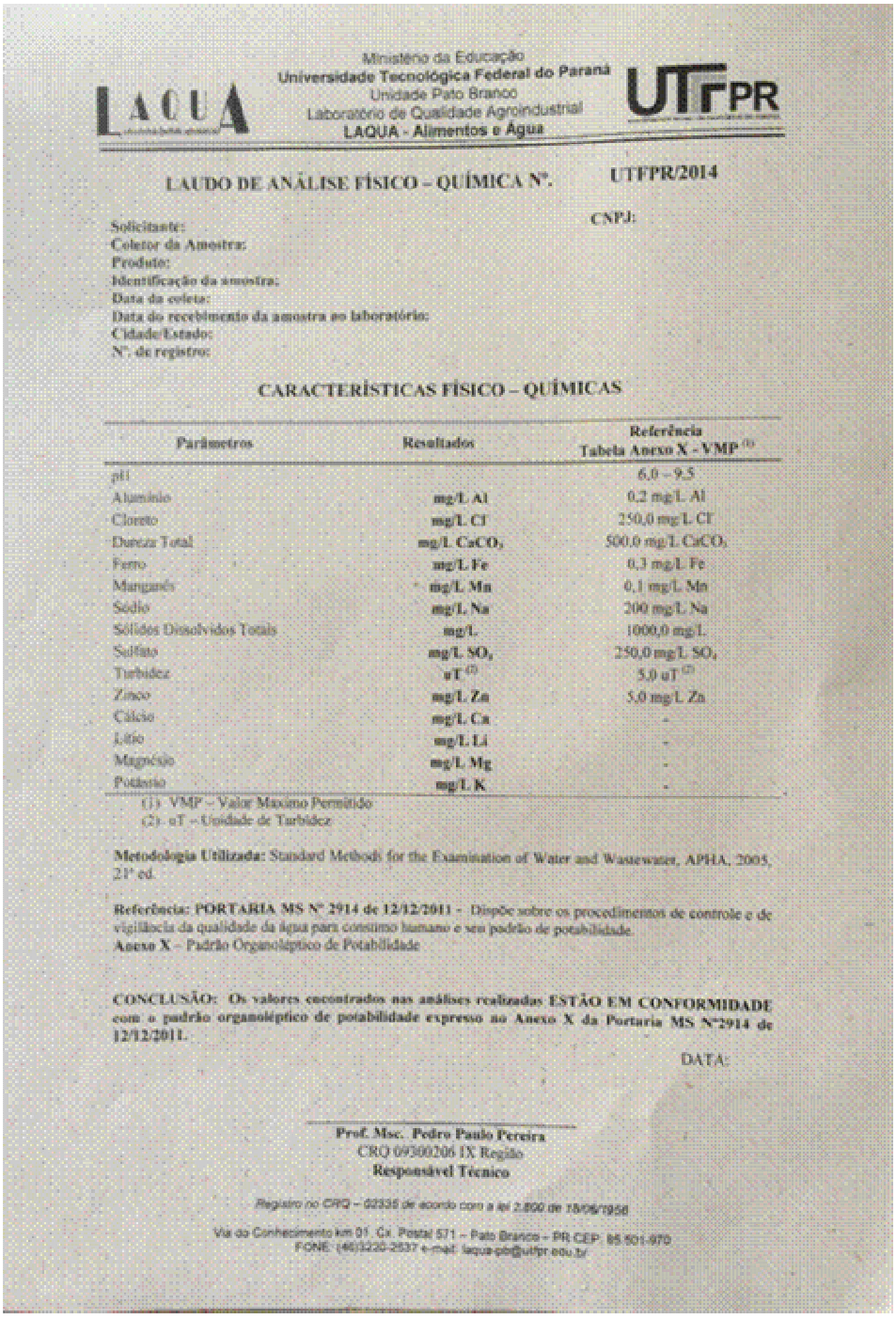
DATA: 03/10/14

Prof. Msc. Pedro Paulo Pereira
CRQ 09300206 IX Região
Responsável Técnico

Registro no CRQ – 02335 de acordo com a lei 2.800 de 18/06/1956

Via do Conhecimento km 01, Cx. Postal 571 – Pato Branco – PR CEP: 85 501-970
FONE: (46)3220-2537 e-mail: laqua-pb@utfpr.edu.br

ANEXO C - Exemplo de laudo de análise físico-química



Ministério da Educação
Universidade Tecnológica Federal do Paraná
Unidade Pato Branco
Laboratório de Qualidade Agroindustrial
LAQUA - Alimentos e Água

LAUDO DE ANÁLISE FÍSICO – QUÍMICA Nº. UTFPR/2014

CNPJ:

Solicitante:
Coletor da Amostra:
Produto:
Identificação da amostra:
Data da coleta:
Data do recebimento da amostra ao laboratório:
Cidade/Estado:
Nº. de registro:

CARACTERÍSTICAS FÍSICO – QUÍMICAS

Parâmetros	Resultados	Referência Tabela Anexo X - VMP (1)
pH		6,0 – 9,5
Alumínio	mg/L Al	0,2 mg/L Al
Cloreto	mg/L Cl	250,0 mg/L Cl
Dureza Total	mg/L $CaCO_3$	500,0 mg/L $CaCO_3$
Ferro	mg/L Fe	0,3 mg/L Fe
Manganês	mg/L Mn	0,1 mg/L Mn
Sódio	mg/L Na	200 mg/L Na
Sólidos Dissolvidos Totais	mg/L	1000,0 mg/L
Sulfato	mg/L SO_4	250,0 mg/L SO_4
Turbidez	uT (2)	5,0 uT (2)
Zinco	mg/L Zn	5,0 mg/L Zn
Cálcio	mg/L Ca	-
Lítio	mg/L Li	-
Magnésio	mg/L Mg	-
Potássio	mg/L K	-

(1): VMP – Valor Máximo Permitido
(2): uT – Unidade de Turbidez

Metodologia Utilizada: Standard Methods for the Examination of Water and Wastewater, APHA, 2005, 21ª ed.

Referência: PORTARIA MS Nº 2914 de 12/12/2011 - Dispõe sobre os procedimentos de controle e de vigilância da qualidade da água para consumo humano e seu padrão de potabilidade.
Anexo X – Padrão Organoléptico de Potabilidade

CONCLUSÃO: Os valores encontrados nas análises realizadas ESTÃO EM CONFORMIDADE com o padrão organoléptico de potabilidade expresso no Anexo X da Portaria MS Nº2914 de 12/12/2011.

DATA:

Prof. Msc. Pedro Paulo Pereira
CRQ 09300206 IX Região
Responsável Técnico

Registro no CRQ – 02335 de acordo com a lei 2.800 de 18/06/1956

Via do Conhecimento km 01, Cx. Postal 571 – Pato Branco – PR CEP: 85.501-970
FONE: (46)3220-2537 e-mail: laqua-pb@utfpr.edu.br