INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ANDERSON DE AZEREDO RIBEIRO

SISTEMA WEB DE CONTROLE DE VACINAS

Jorge Luis Boeira Bavaresco

PASSO FUNDO
2017

ANDERSON DE AZEREDO RIBEIRO

SISTEMA WEB DE CONTROLE DE VACINAS

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Jorge Luis Boeira Bavaresco

PASSO FUNDO

2017

ANDERSON DE AZEREDO RIBEIRO

SISTEMA WEB DE CONTROLE DE VACINAS

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Jorge Luis Boeira Bavaresco

_____

Carmen Vera Scorsatto

_____

André Fernando Rollwagen

_____

Coordenação do Curso

PASSO FUNDO

2017

DEDICATÓRIA

A minha filha,
Pelo tempo em que
Deixei de estar presente,
Fazendo as tarefas do curso.

AGRADECIMENTOS

Agradeço a toda minha família, em especial minha esposa Lisandra Porto dos Santos Comunello e minha filha Laura Comunello Azeredo pela compreensão em momentos que foi necessário me dedicar e deixar um pouco de lado um programa em família, além disso, pelo apoio e motivação nos momentos mais difíceis.

Agradeço muito ao Professor Jorge Luis Boeira Bavaresco, que acreditou e com muita atenção contribuir com suas excelentes orientações auxiliando muito para o desenvolvimento do trabalho. Estendo o agradecimento a todos os professores que compartilharam o conhecimento em aula no decorrer do curso e aos servidores que sempre prestativos sanaram diversas dúvidas. Sendo assim agradeço o IFSul câmpus Passo Fundo pela oportunidade de aprendizado.

Finalizo agradecendo os colegas, pois ficaram lembranças dos momentos em sala de aula e a amizade conquistada para a vida inteira. E também agradecer a Deus por me abençoar e me proteger.

EPÍGRAFE

“Não há derrota que derrote quem nasceu para vencer. ”

Desconhecido

RESUMO

A vacinação em massa nas campanhas pode erradicar uma doença, sendo assim a vacina é de extrema importância na promoção da saúde. O Ministério da Saúde criou em 1973 o Programa Nacional de Imunizações para diminuir ou erradicar doenças através da vacinação da população brasileira. Diante deste contexto, este estudo implementou um sistema *web* de controle de vacinas informatizado, gerenciando os registros de imunizações que ainda são realizados de modo obsoleto, impreciso, inseguro, de forma manual com caneta em uma caderneta. O sistema foi desenvolvido com tecnologias da plataforma Java EE como JSF, JPA, EJB, sendo acessível pelo usuário por navegador de internet, fazendo uso de conceitos de orientação a objetos e banco de dados Postgresql para armazenar as informações. O resultado obtido foi a implementação do sistema acessado através de *login* e que ao ter um usuário logado como administrador permite acesso as manutenções de vacinas, idades, crianças e usuários, podendo cadastrar, editar e excluir informações. Já quando o usuário logado não for administrador, ele apenas poderá cadastrar crianças e também visualizar as suas crianças cadastradas.

Palavras-chave: Vacinas. Gerenciamento. Sistema *web*.

ABSTRACT

Bulk vaccination in campaigns can eradicate an illness, so the vaccine is of utmost importance in promoting health. In 1973, the Ministry of Health created the National Immunization Program to reduce or eradicate diseases through the vaccination of the Brazilian population. In view of this context, this study proposes to implement a web-based computerized vaccine control system, managing immunization records that are still obsolete, inaccurate, and insecure, in a manual way with a pen in a notebook. The developed with Java EE platform technologies such as JSF, JPA, EBJ, being accessible by the user through Internet browser, making use of concepts of object orientation and Postgresql database to store the information. The result obtained was the implementation of the system accessed through login and that having a user logged in as administrator allows access to the maintenance of vaccines, ages, children and users, being able to register, edit and delete information. Already when the logged in user is not an administrator, he can only register children and also view their registered children.

Keywords: Vaccines. Management. Web System.

LISTA DE FIGURAS

Figura 1 - Aplicativo Vacinação em Dia 21
Figura 2 - Cadastro de cartões 22
Figura 3 - Cartões 22
Figura 4 - Adicionar vacina 23
Figura 5 - Vacinas 23
Figura 6 - Lembrete 24
Figura 7 - Tela inicial do aplicativo Vacinas em Dia 25
Figura 8 - Vacinas de uma faixa etária 26
Figura 9 - Descrição da vacina 26
Figura 10 - Diagrama de casos de uso 30
Figura 11 - Diagrama de classes 33
Figura 12 - Diagrama de atividades cadastro 34
Figura 13 - Diagrama de atividades lembrete 34
Figura 14 - Projetos no NetBeans 35
Figura 15 - Banco de Dados 35
Figura 16 - Bibliotecas 37
Figura 17 - Arquivo persistence.xml 38
Figura 18 - Classe Crianca.java 39
Figura 19 - DAOGenerico 40
Figura 20- Classe IdadeDAO 41
Figura 21 - Controle Idade 42
Figura 22 - Métodos Controle Idade 43
Figura 23 - Template 44
Figura 24 - Menu do sistema 45
Figura 25- listar.xhtml 45
Figura 26 - ConverterCalendar 46
Figura 27 - Tela inicial 47
Figura 28 - Login 47
Figura 29 - Listagem de crianças 48
Figura 30 - Dados da criança 48
Figura 31 - Tela de edição de criança 49
Figura 32 - Tela de exclusão 49

Figura 33 - Opção para sair do sistema 50

LISTA DE TABELAS

Tabela 1 - Cadastrar criança 31
Tabela 2 - Enviar lembrete 31

LISTA DE ABREVIATURAS E SIGLAS

EJB – Enterprise Java Beans

IFSUL – Instituto Federal Sul-rio-grandense

IPPES – Instituto de Pesquisa, Planejamento e Promoção da Educação e Saúde.

JPA – Java Persistence API

JSF – Java ServerFaces

MVC – Model-View-Controller

SUS – Sistema Único de Saúde

TCC – Trabalho de Conclusão de Curso

UML – Unified Modeling Language

URL – Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO 13
1.1 TEMA 14
1.1.1 DELIMITAÇÃO DO TEMA 14
1.2 PROBLEMA 14
1.3 HIPÓTESE 14
1.4 OBJETIVOS 14
1.5 OBJETIVO GERAL 15
1.6 OBJETIVOS ESPECIFICOS 15
1.7 JUSTIFICATIVA 15

2 REFERENCIAL TEÓRICO 17
2.1 JAVA 17
2.2 Java EE 17
2.3 JSF 18
2.4 JPA 18
2.5 UML 18

3 TRABALHOS CORRELATOS 20
3.1 SIVAC 20
3.2 Aplicativo Vacinação em Dia 20
3.3 Aplicativo Vacinas em Dia 24

4 METODOLOGIA 27
4.1 Estudo de caso 27
4.2 Contexto atual 27
4.3 Analise do contexto atual 28

5 MODELAGEM 29
5.1 Descrição do novo sistema 29
5.2 Requisitos funcionais 29
5.3 Requisitos não funcionais 29
5.4 Diagramas de casos de uso 30
5.5 Diagrama de Classes 32

5.6 Diagrama de atividades 33

6 DESENVOLVIMENTO DO SISTEMA 35
6.1 Camada de Modelo 38
6.2 Camada DAO 40
6.3 Camada de controle 41
6.4 Camada de Visão 43
6.4.1 Conversores 46
6.5 Apresentação de execução do sistema 46

7 CONSIDERAÇÕES FINAIS 51

8 REFERÊNCIAS 52

1 INTRODUÇÃO

A saúde da população é uma preocupação dos órgãos municipais, estaduais e nacionais de saúde, que tentam de várias formas manter a sociedade livre de qualquer doença.

Uma forma de a sociedade ficar imune a doença é a vacinação, pensando nisso desde 1973 existe o Programa Nacional de Imunizações. Porém existem certas dificuldades no controle preciso da vacinação de uma criança.

Diante dessa importância da vacinação e também da dificuldade de um controle, o trabalho vem proporcionar um gerenciamento adequado das vacinações de uma criança, desenvolvendo um sistema *web* de controle de vacinas para os pais ou responsáveis pela criança, gerenciar as vacinas cadastrando a criança no sistema, assim podendo gerenciar as vacinações de mais de uma criança da família.

O desenvolvido do trabalho foi realizado devido à experiência de ser pai e ao convívio frequente em um posto de vacinação levando minha filha para ser vacinada nos primeiros meses de vida. A análise da forma de registro contribuiu para definir os objetivos do projeto. A partir disto foi realizada uma modelagem do sistema para uma melhor visualização da estrutura. Após foi realizado o desenvolvimento de uma aplicação em Java com uso da estrutura MVC com banco de dados PostgreSql para atender os requisitos.

1.1 TEMA

Desenvolvimento de sistema web para controle de vacinação.

1.1.1 DELIMITAÇÃO DO TEMA

Desenvolvimento de sistema web para controle de vacinas que possibilitará o cadastro de crianças e gerenciamento das vacinas realizadas, e enviará lembretes para os pais ou responsáveis conforme as datas de vacinação se aproximam.

1.2 PROBLEMA

Como controlar de forma mais precisa a vacinação de uma criança possibilitando avisar os pais ou responsáveis quando é necessário realizar alguma vacina?

1.3 HIPÓTESE

Com o desenvolvimento de um sistema *web* pode-se gerenciar de forma mais adequada à vacinação de uma criança, possibilitando aos pais ou responsáveis pela criança receberem um lembrete conforme a aproximação da data da vacinação.

1.4 OBJETIVOS

Esta seção apresenta os objetivos gerais e específicos que conduziram o desenvolvimento do sistema.

1.5 OBJETIVO GERAL

Desenvolver um sistema de controle de vacinas de crianças para que seus responsáveis possam gerenciar as vacinas com envio de lembrete conforme a data da vacinação se aproxima.

1.6 OBJETIVOS ESPECIFICOS

- Estudar linguagem Java;
- Pesquisar sobre vacinação assim alimentando a base de dados;
- Realizar a modelagem da aplicação que permite cadastrar a criança para gerenciamento das vacinas;
- Desenvolver um sistema para gerenciar a vacinação de mais de uma criança da família.

1.7 JUSTIFICATIVA

Devido à importância da vacinação para diminuir ou erradicar uma doença foi criado em 1973 pelo Ministério da Saúde o Programa Nacional de Imunizações para vacinar a população brasileira, Fundação Oswaldo Cruz. Porém o registro ainda é realizado de modo obsoleto, impreciso, inseguro, de forma manual com caneta em uma caderneta de vacinação disponibilizada para os pais logo após o nascimento da criança.

Sendo assim para saber a data e o tipo da próxima vacina a ser realizada, os pais ou responsáveis pela criança devem conferir o lembrete escrito a lápis na última vez que a criança compareceu em um posto de saúde para se vacinar. Sabendo-se que cada criança possui uma caderneta de vacinação, e que as vacinas obrigatórias se diferem conforme a faixa etária, se a família tiver mais de uma criança essa verificação deve ser feita na caderneta de cada uma delas.

Ao nascerem todas as crianças recebem uma carteira de vacinação. Nesta carteira é realizado o registro de forma manual de todas as anotações referentes a uma vacina que foi realizada e também um lembrete a lápis da próxima vacina a ser feita. No momento da vacinação é obrigatória a apresentação desta carteira para

que além de ser registrada a data e qual vacina está sendo feita, também seja possível realizar uma revisão de alguma vacina que possa ter sido esquecida.

Com o objetivo de evitar transtornos com esquecimento de alguma vacina o trabalho com viés social que foi desenvolvido para controle de vacinas mantendo atualizado o calendário de imunização com todas as vacinas que a criança já fez. O sistema enviará um lembrete quando a data da vacinação estiver se aproximando e salva as informações de maneira digital em um banco de dados com fácil acesso.

2 REFERENCIAL TEÓRICO

Este capítulo contém embasamento teórico dos principais conceitos das tecnologias e ferramentas utilizados para dar suporte no desenvolvimento do sistema.

2.1 JAVA

O desenvolvimento de *softwares* só é possível devido à existência de tecnologias de desenvolvimento, dentre as diversas linguagens de programação disponíveis, Java começou a ser desenvolvida pela Sun Microsystems e foi lançada em 1995. É uma linguagem orientada a objetos e permite produção de softwares robustos e multiplataformas. Segundo DEITEL (2002) Java também é utilizado para criação de páginas web com conteúdo interativo e dinâmico, desenvolvimento de aplicativos de grande porte e para várias outras funcionalidades.

O desenvolvimento em Java consiste em dividir as partes com criação de classes, essas por sua vez possuem métodos que realizarão tarefas ou retornarão informações e, além disso, uma classe possui atributos que servem para guardar uma informação da aplicação.

Java é uma linguagem portável, pois os programas nela escritos podem ser executados em vários computadores diferentes que tenham capacidade de rodar a máquina virtual do Java que é chamada de JVM (DEITEL, 2002). Essa característica de multiplataforma só é possível porque após o processo de compilação que traduz o código-fonte é gerado um arquivo **.class** contendo os *bytecodes* linguagem da máquina virtual Java que são interpretados durante a execução.

2.2 Java EE

A plataforma Java EE que se originou próximo dos anos 1999 devido à grande aceitação do Java para resolver problemas empresariais (FURGERI, 2008), sendo que o avanço do *Enterprise Java Beans* (EJB) proporcionou o desenvolvimento do Java EE.

Elaborada segundo FURGERI (2008) para padronizar o desenvolvimento e distribuição de aplicações *web,* possui um padrão para criação de aplicações distribuídas que são executadas em vários sistemas de computador.

2.3 JSF

O JavaServer Faces (JSF) surgiu na versão 1.0 em meados de 2004, e possui as seguintes versões: JSF 1.0, JSF 1.1, JSF 1.2 estas primeiras versões lançadas em 2004, JSF 2.0 em 2009, JSF 2.1 lançada em 2010 e por fim JSF 2.2 em 2013.

JSF é um *framework* baseado em componentes que segue um padrão MVC (*Model-View-Controller)* para construção de interfaces ricas em recursos visuais, disponibilizando interatividade para as aplicações *web* superior ao obtido ao usar HTML puro, usando um ambiente é possível elaborar um formulário arrastando e soltando componentes visuais pré-prontos, pois o JSF contém código necessário para manipulação e organização possibilitando que programadores se dediquem apenas na lógica da aplicação GEARY, HOSTMAN (2012).

2.4 JPA

Ao utilizar uma aplicação, dentre várias funcionalidades do sistema, o usuário necessita de armazenamento das informações em um banco de dados para futuramente realizar consultas e até mesmo gerar relatórios.

Conforme GEARY, HOSTMAN (2012) a arquitetura disponibiliza um mapeador do tipo objeto/relacional (O/R) padrão para Java EE que converte tabelas do banco de dados e objetos Java. Sendo assim a aplicação não interage diretamente com o banco de dados. Com JPA usa-se anotações para indicar classes (entidades) que podem ser armazenadas no banco de dados.

2.5 UML

O surgimento da UML teve origem devido à necessidade de padronização do desenvolvimento de aplicações, pois eram usados processos e notações distintos

entre as diferentes empresas e elas desejavam ferramentas que suportassem os seus processos (DEITEL, 2002).

A UML é um esquema de representação gráfica complexa e completa de recursos para modelagem de sistemas orientados a objetos, que teve versões preliminares liberadas em 1996. Segundo DEITEL (2002) ela unificou os diversos esquemas de notação e tem como característica atraente a flexibilidade.

3 TRABALHOS CORRELATOS

Nesta seção são apresentados alguns trabalhos semelhantes encontrados durante a pesquisa para realização deste trabalho.

3.1 SIVAC

Conforme a página do conselho federal de odontologia, o SIVAC é uma solução disponibilizada pelo IPPES – Instituto de Pesquisa, Planejamento e Promoção da Educação e Saúde que o define como um sistema *on-line* de vacinação com finalidade de registrar e monitorar a aplicação de vacinas.

O sistema possibilita controle de estoque de vacinas, análise de informações de unidades de saúde que realiza aplicação além de disponibilizar uma caderneta de vacinação eletrônica. Atende os padrões dos mais variados serviços públicos de imunização, como por exemplo: unidade básica de saúde, unidade saúde da família, policlínicas, etc. Os gestores têm a possibilidade de realizar consultas através do acesso remoto por navegador. Porém para acessar o SIVAC é necessário estar cadastrado no Portal do Software Público

.

O SIVAC possui as seguintes características técnicas:

- Sistemas operacionais: Linux, Windows e MAC OS;
- Navegadores: IE, FIREFOX, CHROME, SAFARI e OPERA;
- Linguagem de Programação: Sistema Orientado a Objetos com linguagens de programação PHP e JAVASCRIPT, com CSS, HTML em conjunto ao AJAX;
- Banco de Dados: MYSQL.

3.2 Aplicativo Vacinação em Dia

Vacinação em Dia[1], conforme o site do programa nacional de imunizações, é um aplicativo para *smartphones* e *tablets* disponibilizado pelo Ministério da Saúde

[1] Disponível em https://play.google.com/store/apps/details?id=com.digital.vacinacao&hl=en

para o gerenciamento de cadernetas de vacinação castradas pelo usuário. Também possui informações completas das vacinas disponibilizadas pelo Sistema Único de Saúde (SUS) e função de lembretes de campanhas sazonais de vacinação.

A tela inicial do aplicativo, demonstrada na Figura 1, apresenta para o usuário um menu com as opções, cartões, vacinas, lembrete e *info*.

Figura 1 - Aplicativo Vacinação em Dia

Fonte: Aplicativo Vacinação em Dia, 2017

O aplicativo possibilita o usuário adicionar cartões informando os dados: nome, e-mail, celular, data de nascimento, sexo, receber notificações e caso desejar pode inserir uma foto, a Figura 2 demonstra a tela de cadastro de cartões.

Figura 2 - Cadastro de cartões

Fonte: Aplicativo Vacinação em Dia, 2017

Após realizar o cadastro de cartões, ao clicar no botão “Adicionar vacina” poderá adicionar as vacinas informando manualmente a vacinação realizada inserindo dados como: data, vacina, lote, local, e assim fazer o gerenciamento das vacinas. Figura 3 e Figura 4 demonstram as respectivas telas.

Figura 3 - Cartões

Fonte: Aplicativo Vacinação em Dia, 2017

Figura 4 - Adicionar vacina

Fonte: Aplicativo Vacinação em Dia, 2017

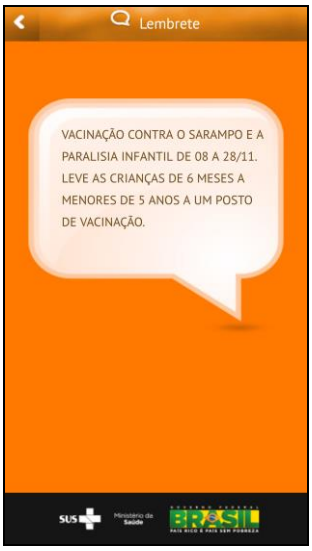
Para obter informações sobre uma determinada vacina, ao clicar no botão vacinas o usuário terá acesso a uma lista de vacinas que estão organizadas por faixa etária (criança, adolescente, adulto e idoso), conforme a Figura 5 apresenta esta tela do aplicativo.

Figura 5 - Vacinas

Fonte: Aplicativo Vacinação em Dia, 2017

Outra funcionalidade do aplicativo é o lembrete de campanhas de vacinação, usuário consegue acessar esta informação clicando no botão “Lembrete”. É apresentado um texto contendo orientações da(s) vacina(s) que deve ser realizada, o prazo para realizar a vacina e a faixa etária do público alvo da campanha. A Figura 6 ilustra esta tela.

Figura 6 - Lembrete



Fonte: Aplicativo Vacinação em Dia, 2017

O aplicativo é uma alternativa além da caderneta de vacinação para os pais e responsáveis realizarem o gerenciamento das vacinações de mais de uma criança da família, porém a função lembrete apenas traz aos usuários a informação da campanha de vacinação somente quando a tela do aplicativo é aberta.

3.3 Aplicativo Vacinas em Dia

Vacinas em dia é um aplicativo para *android* disponível na Google play[2], que disponibiliza informações de todas as vacinas necessárias para fazer durante a vida, possui funcionalidades de exibir as vacinas por faixa etária, vacinas da gravidez, vacinas obrigatórias, vacinas opcionais e informações completas das vacinas.

[2] Disponível em https://play.google.com/store/apps/details?id=br.com.minhavida.vacinas&hl=pt_BR

A Figura 7 demonstra que ao abrir o aplicativo na tela inicial tem seis opções de perfil disponíveis para o usuário escolher:

- 0 a 23 meses;
- 2 a 9 anos;
- 10 a 19 anos;
- 20 a 59 anos;
- 60 ou mais;
- Gestantes.

Figura 7 - Tela inicial do aplicativo Vacinas em Dia

Fonte: Aplicativo Vacinas em Dia, 2017

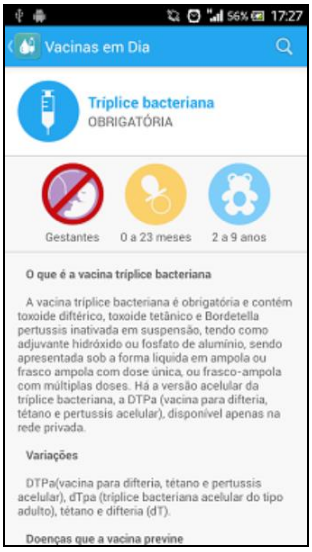
Quando o usuário escolhe um perfil são exibidas todas as vacinas obrigatórias e opcionais, estas podendo ser ocultadas tocando o botão “ocultar opcionais”, conforme ilustra a Figura 8.

Figura 8 - Vacinas de uma faixa etária

Fonte: Aplicativo Vacinas em Dia, 2017

Selecionando uma vacina o aplicativo disponibiliza uma tela com todas as informações da vacina, a Figura 9 apresenta esta tela.

Figura 9 - Descrição da vacina



Fonte: Aplicativo Vacinas em Dia, 2017

Este aplicativo serve de auxílio para os pais ou responsáveis verificarem a idade correta para realização de uma vacina, porém a diferença entre ele e o trabalho proposto é que este aplicativo não tem a funcionalidade de aviso.

4 METODOLOGIA

Neste capítulo serão apresentados os métodos utilizados para realização do trabalho, o detalhamento do contexto atual, os requisitos funcionais e não funcionais e os diagramas de caso de uso, classes e atividades do sistema proposto.

4.1 Estudo de caso

Estudo de caso do atual controle da vacinação da criança em postos de saúde da rede pública. Realizado na experiência de ser pai e na observação do atendimento ao levar minha filha para ser vacinada várias vezes nos primeiros anos de vida

4.2 Contexto atual

De acordo com informações contidas na Caderneta de Saúde da Criança, no Brasil quando nasce uma criança, nasce também um novo cidadão brasileiro ou cidadã brasileira, sendo assim terá ao longo de sua vida vários documentos, tais como: RG, CPF, entre outros. A Caderneta de Saúde da Criança é o primeiro documento que os pais recebem logo após o nascimento do filho, muito importante e único para cada criança, pois contém informações e orientações para acompanhar a saúde, o crescimento e o desenvolvimento.

Começando pela identificação da criança, nome, data de nascimento, nome dos pais e endereço, a caderneta também tem um espaço para os profissionais de saúde realizar registro de dados sobre a gravidez, parto e puerpério com informações do pré-natal (data, número de consultas, gravidez simples ou múltipla), parto e pós-parto (local e tipo). Também podem ser registrados dados do recém-nascido entre elas data e horário do nascimento, maternidade, peso e comprimento ao nascer, exames/triagem neonatal e dados da alta (data e alimentação) e outras informações importantes sobre a criança. Além disso, contém gráficos de crescimento, instrumento de vigilância do desenvolvimento e tabelas para registro de vacinas aplicadas.

Toda vez que os pais levarem seus filhos aos serviços de saúde e nas campanhas de vacinação para receber a dose de uma vacina, esta caderneta deve

ser levada para que o profissional da saúde, que além de realizar o registro dos campos, data, lote, unidade e assinatura de forma manual com uma caneta possibilita também fazer uma revisão das vacinas já realizadas a fim de identificar alguma vacina em atraso. Após isso o profissional escreve a lápis uma espécie de lembrete do ano ou idade da próxima imunização.

4.3 Analise do contexto atual

Analisando-se o contexto atual foi identificado que o controle de vacinação ainda é realizado de maneira manual, portanto apresenta falhas no gerenciamento adequado como, por exemplo, não existir outra forma do pai ou responsável saber quando deve ser realizada a vacinação sem ser verificando a caderneta de vacina de cada criança, além disso, para a vacina ser devidamente realizada é necessário a apresentação da caderneta.

Ao se deslocar até o posto de saúde sem portar a caderneta de vacinação os agentes de saúde podem se recusar aplicar a vacina, ou fornecer uma nova para efetuar o registro, sendo assim nesta nova caderneta pode conter somente uma vacina registrada e a pessoa deverá portar as duas cadernetas de vacinação. Outra falha identificada é a dificuldade de recuperar as informações nela registradas caso ocorra perda ou extravio da caderneta.

5 MODELAGEM

Com a análise do contexto atual foram identificadas as necessidades que o sistema proposto deve suprir. Suas funcionalidades serão ilustradas nos diagramas presentes nesta seção.

5.1 Descrição do novo sistema

O acesso ao novo sistema será realizado através de *login* inserindo usuário e senha válidos. Após estar autenticado será possível realizar o cadastro da criança inserindo todas as informações preenchendo os campos obrigatórios: nome, nascimento, sexo e nome dos pais. Ao cadastrar uma criança o sistema possibilita o usuário importar vacinas, sendo que essa importação será realizada com base no calendário de nacional vacinação do Ministério da Saúde.

O sistema vai exibir as vacinas obrigatórias por faixa etária e também uma tabela de vacinas realizadas. Além disso, conforme a data de vacinação se aproxima o sistema enviará um lembrete.

5.2 Requisitos funcionais

Nesta seção serão apresentados os requisitos funcionais da aplicação.

- **RF1**: Realizar *login*;
- **RF2**: Manter o cadastro da criança;
- **RF3:** Importar vacinas;
- **RF4**: Exibir vacinas obrigatórias por faixa etária;
- **RF5**: Exibir tabela de vacinas realizadas;
- **RF6:** Enviar lembrete.

5.3 Requisitos não funcionais

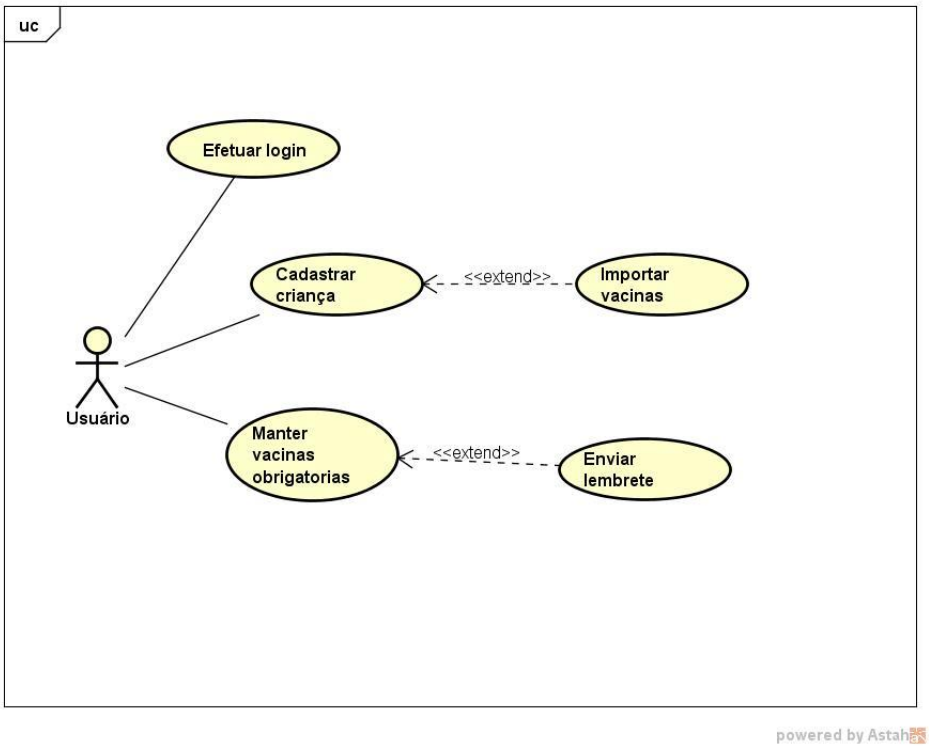
Seção em que os requisitos não funcionais do sistema são apresentados em forma de itens.

- **RNF1**: Sistema multiplataforma;
- **RNF2**: Acessível pelo usuário por navegador de internet;
- **RNF3**: Controle de acesso.

5.4 Diagramas de casos de uso

O diagrama de casos de uso (Figura 10) foi elaborado com base no estudo de caso para apresentar a ação do ator usuário e as funcionalidades do sistema. Sendo que o usuário deverá realizar o *login* e após isso terá acesso ao sistema possibilitando cadastrar criança, no momento do cadastro caso o usuário assim optar, o sistema permite realizar importação das vacinas pelo fato que cada faixa etária tem diferentes vacinas obrigatórias, que são mantidas pelo sistema.

Figura 10 - Diagrama de casos de uso



Fonte: Próprio autor

A baixo segue a descrição detalhada do caso de uso, cadastrar criança Tabela 1.

Tabela 1 - Cadastrar criança

Nome do caso de uso	Cadastrar criança
Ator principal	Usuário
Resumo	Registrar os dados da criança.
Pré-condições	Efetuar login
Pós-condições	Armazenar criança
Fluxo principal	
Ações do ator	Ações do sistema
1. Selecionar opção de cadastro	
	2. Exibir formulário
3. Preenche os dados da criança	
	4. Salva as informações.
Restrições de integridade	Os campos nome, nascimento, sexo e nomePais são obrigatórios, portanto o usuário deverá inserir todos os dados da criança no cadastro.

Na Tabela 2 pode-se visualizar a descrição detalhada do caso de uso enviar lembrete.

Tabela 2 - Enviar lembrete

Nome do caso de uso	Enviar lembrete
Ator principal	
Resumo	O caso de uso descreve o envio de lembretes de vacinas a serem realizadas para os pais ou responsáveis da criança conforme a proximidade da data.
Pré-condições	As crianças possuírem vacinas registradas no sistema.
Pós-condições	O lembrete foi enviado.
Fluxo principal	
Ações do autor	Ações do sistema

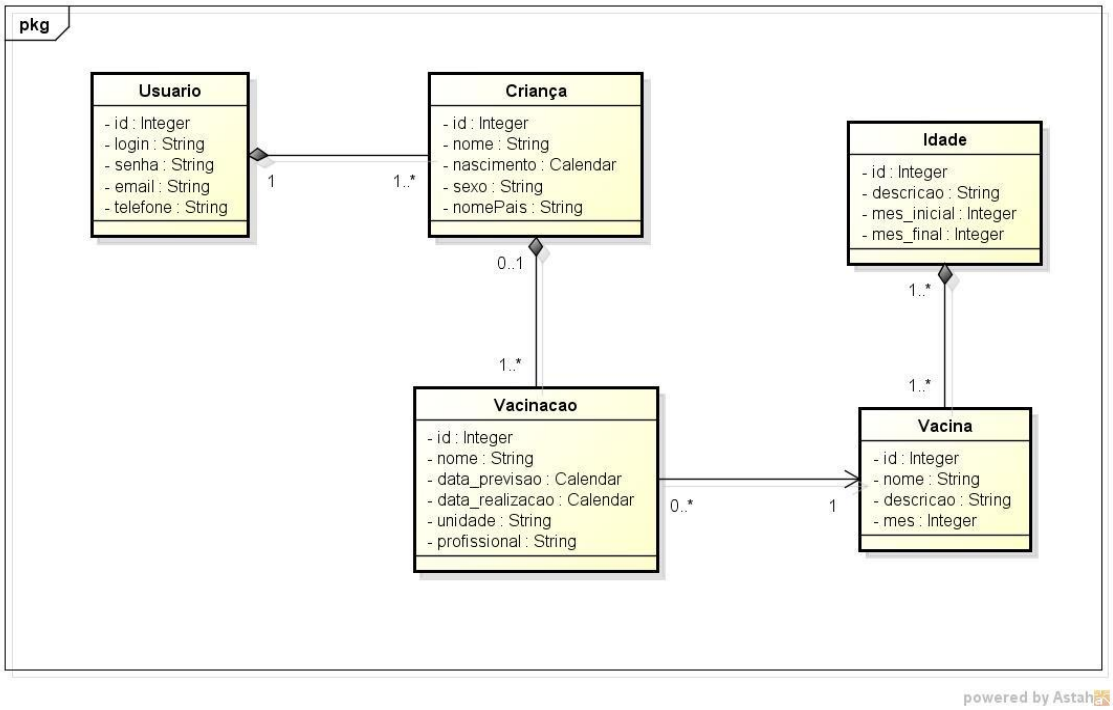
	1. Consultar vacinas não realizadas. 2. Verificar o campo data de realização da vacina. 3. Comparar com a data atual do sistema. 4. Verificar se a diferença de quantidade de dias é menor que dez. 5. Enviar lembrete.
Restrições de integridade	Verificar se existe vacina com o campo data_realizacao não preenchido.

5.5 Diagrama de Classes

O diagrama de classes apresentado na Figura 11 foi desenvolvido para demonstrar as classes do sistema, atributos e seus relacionamentos. A classe “Usuário” possui os atributos *login* e senha, que são necessários para acessar o sistema, e e-mail e telefone para um possível contato.

A classe “Criança” possui uma lista de crianças no sistema, para as quais possuem os atributos nome e nascimento, o qual é o importante para que o sistema possa obter a idade e com esta informação verificar as vacinas necessárias, além desses, a classe também possui os campos sexo e nomePais, esse último campo é importantíssimo para identificação do responsável pela criança, sendo assim esta pessoa é quem receberá o lembrete. A classe “Vacinação” contém os atributos necessários para verificação da vacinação realizada ou em atraso com o campo data de realização. Já a classe “Idade” com os atributos descrição, mês_inicial e mês_final importantes para diferenciar as idades, possui uma lista de vacinas através do relacionamento com a classe “Vacina” sendo que o campo mês tem a função de identificar em qual idade a vacina deve ser realizada.

Figura 11 - Diagrama de classes

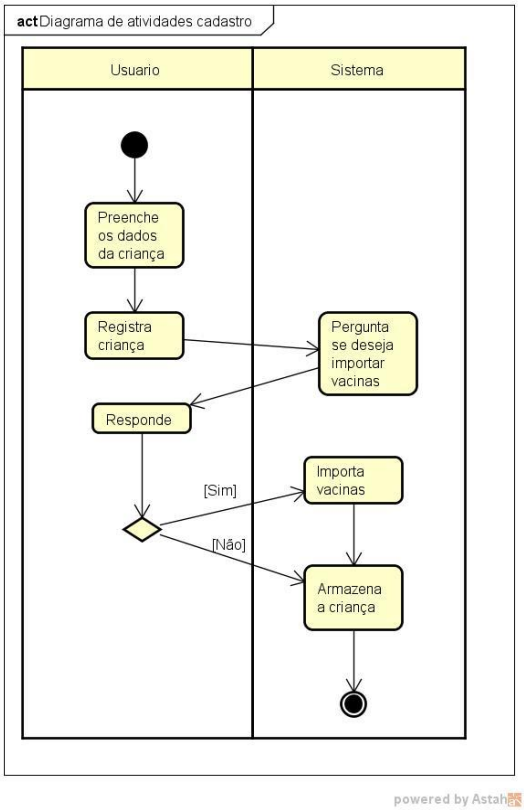


Fonte: Próprio autor

5.6 Diagrama de atividades

O diagrama representado na Figura 12, tem a finalidade de apresentar os fluxos dos casos de uso cadastrar criança, sendo que o usuário deverá inserir os dados da criança e no momento do registro o sistema irá perguntar para o usuário se deseja importar vacinas, com uma resposta positiva o sistema importa vacinas e armazena a criança, caso a resposta for não o sistema armazena a criança sem a importação.

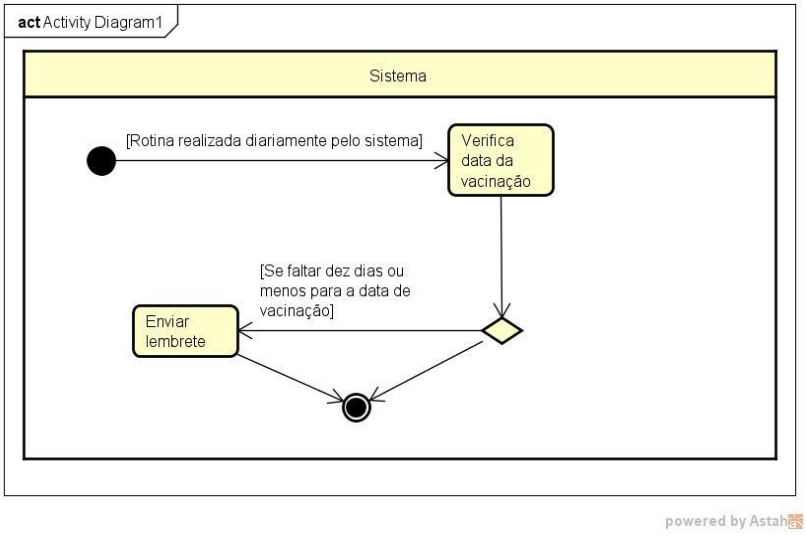
Figura 12 - Diagrama de atividades cadastro



Fonte: Próprio autor

O diagrama exibido na Figura 13 apresenta a atividade diária que o sistema deverá realizar para fazer a verificação da data de vacinação e caso a condição for satisfeita enviar um lembrete para os pais ou responsáveis da criança.

Figura 13 - Diagrama de atividades lembrete

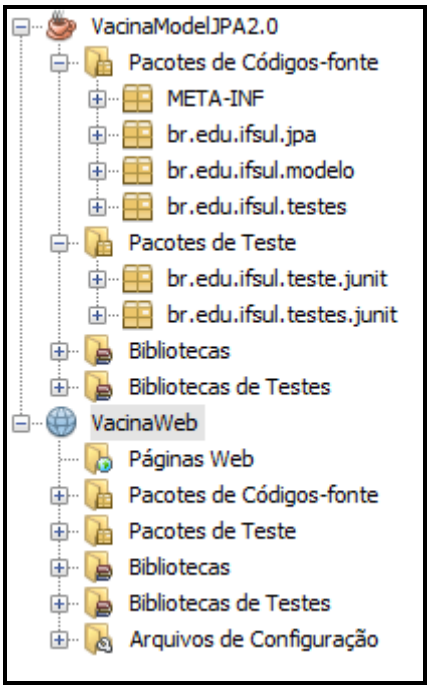


Fonte: Próprio autor

6 DESENVOLVIMENTO DO SISTEMA

O desenvolvimento do sistema proposto foi dividido em dois projetos para o uso da arquitetura MVC possibilitando o reuso e baixo acoplamento. A camada de modelo foi criada com um projeto chamado VacinaModelJPA2.0, e contém as classes das entidades que realizam o mapeamento objeto-relacional com o banco de dados. Já a camada web foi criada com o projeto chamado VacinaWeb, e contém controle e visão com as telas do sistema para o acesso do usuário. Ambos os projetos foram desenvolvidos com a *IDE NetBeans* versão utilizada 8.2. A Figura 14 traz a ilustração da estrutura dos projetos.

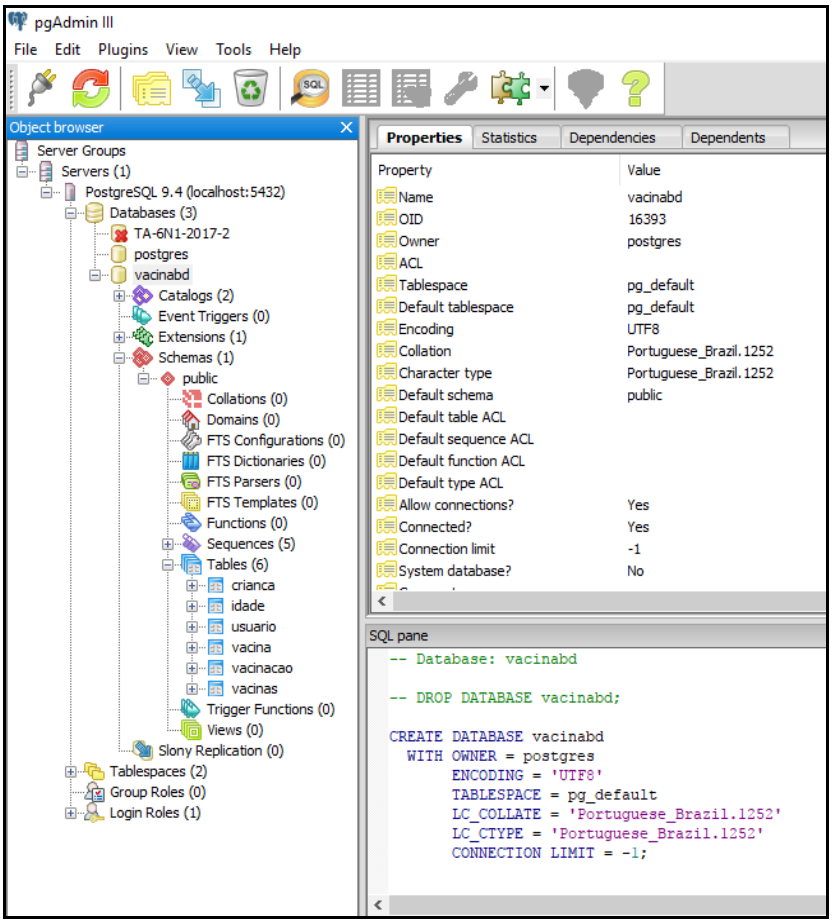
Figura 14 - Projetos no NetBeans



Fonte: Próprio autor

Na aplicação foi utilizado o Sistema Gerenciador de Bando de Dados PostgreSql para armazenamento de informações. Na Figura 15 apresenta-se o banco de dados da aplicação e sua estrutura por meio da ferramenta pgAdminIII.

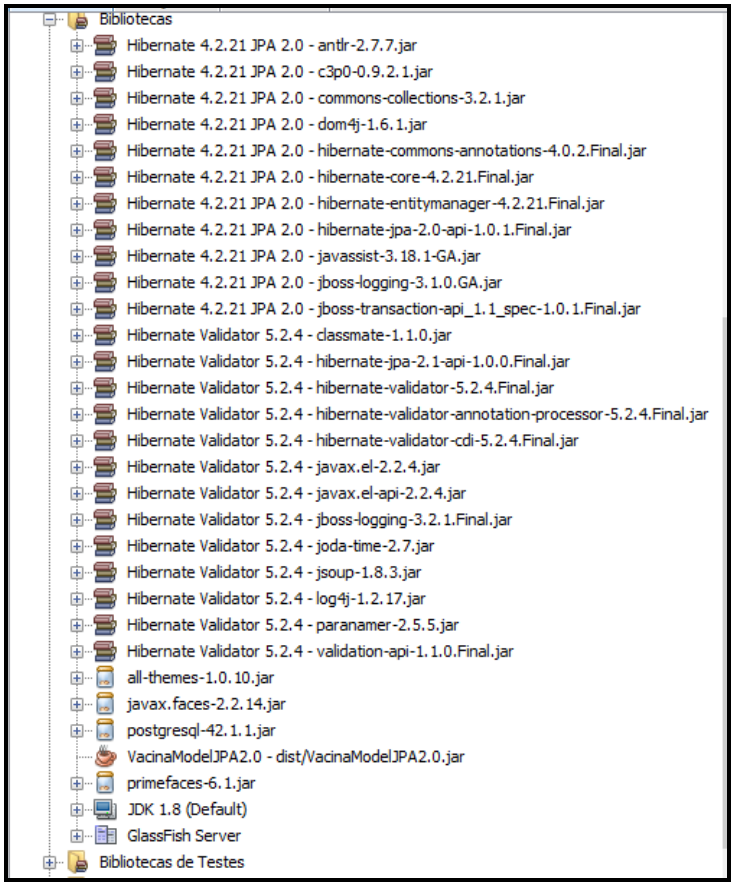
Figura 15 - Banco de Dados



Fonte: Próprio autor

Para o desenvolvimento da camada de modelo foi criado um projeto que utiliza as bibliotecas Hibernate 4.2.21 (biblioteca de persistência de dados que implementa Java Persistence API versão 2.0), utilizada para o mapeamento objeto-relacional e a biblioteca Hibernate Validator 5.2.4 utilizada para validação das informações dos objetos persistidos, além do driver do PostgreSQL para conexão da aplicação com o banco de dados, possibilitando manipulação e manutenção das tabelas. A Figura 16 apresenta as bibliotecas utilizadas.

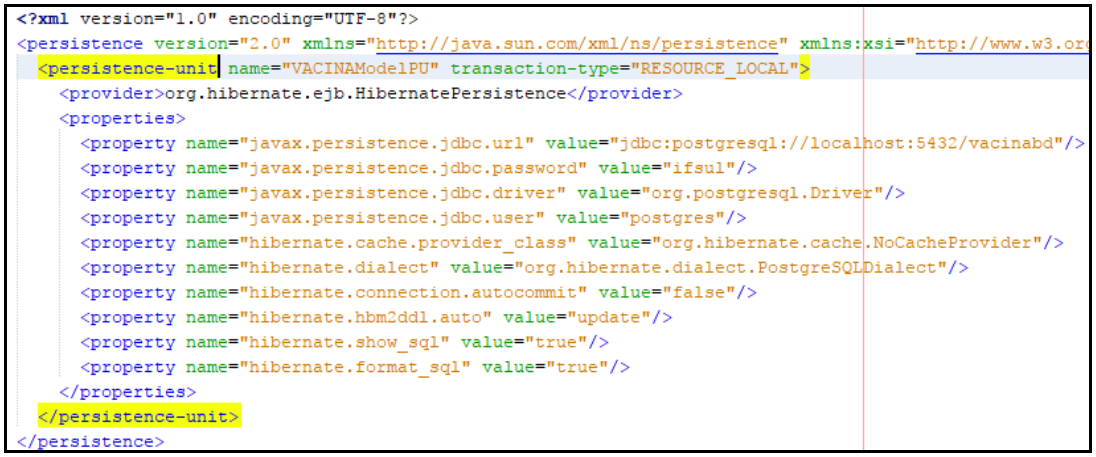
Figura 16 - Bibliotecas



Fonte: Próprio autor

O arquivo ***persistence.xml*** apresentado na Figura 17, contém as propriedades da unidade de persistência e as configurações necessárias para comunicação com banco de dados. No arquivo a *tag <persistence-unit>* armazena o nome da unidade de persistência, já a definição do *Hibernate* como provedor de persistência de dados é através do elemento *<provider>.* A propriedade *<property name="hibernate.hbm2ddl.auto" value="update"/>* define que a atualização das tabelas do banco de dados é realizada pelo método automático.

Figura 17 - Arquivo persistence.xml



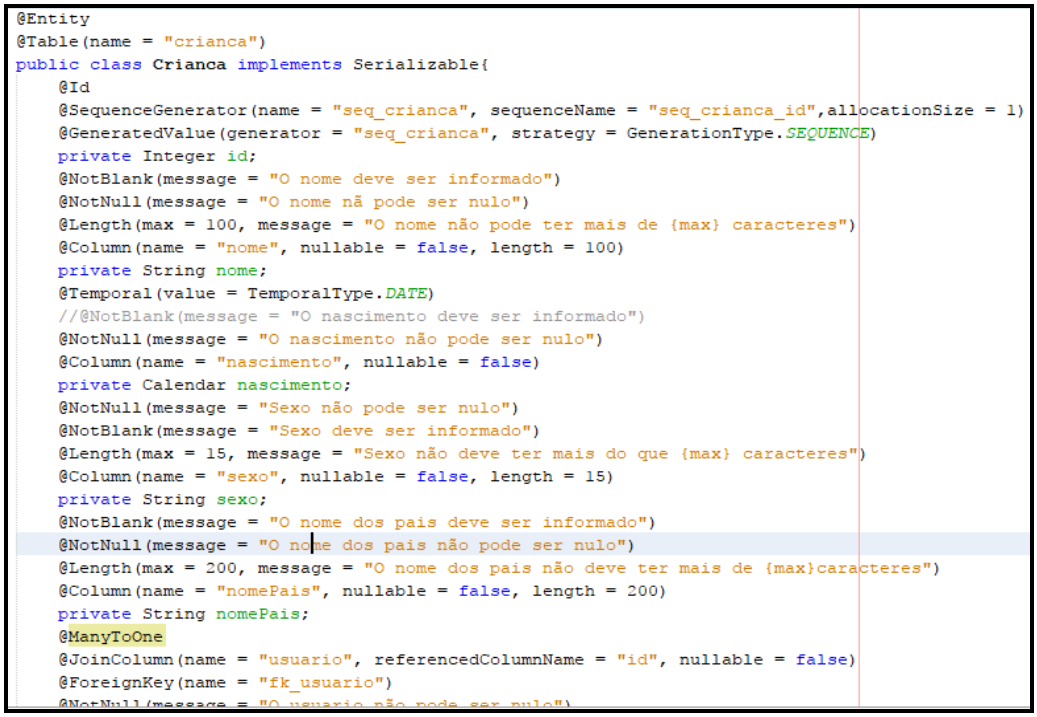
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.or
  <persistence-unit name="VACINAModelPU" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/vacinabd"/>
      <property name="javax.persistence.jdbc.password" value="ifsul"/>
      <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>
      <property name="javax.persistence.jdbc.user" value="postgres"/>
      <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
      <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
      <property name="hibernate.connection.autocommit" value="false"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

Fonte: Próprio autor

6.1 Camada de Modelo

A camada de modelo contém as entidades persistentes, realiza o mapeamento objeto-relacional pela JPA que é responsável por criar correspondência entre os objetos e as tabelas da aplicação. Para fazer parte da persistência as classes devem seguir o padrão *JavaBeans*, para isso é necessário implantar a interface *Serializable,* possuir campos encapsulados e métodos *getter* e *setter.* Um trecho de código da classe Crianca.java é ilustrado na Figura 18.

Figura 18 - Classe Crianca.java



```
@Entity
@Table(name = "crianca")
public class Crianca implements Serializable{
    @Id
    @SequenceGenerator(name = "seq_crianca", sequenceName = "seq_crianca_id",allocationSize = 1)
    @GeneratedValue(generator = "seq_crianca", strategy = GenerationType.SEQUENCE)
    private Integer id;
    @NotBlank(message = "O nome deve ser informado")
    @NotNull(message = "O nome nã pode ser nulo")
    @Length(max = 100, message = "O nome não pode ter mais de {max} caracteres")
    @Column(name = "nome", nullable = false, length = 100)
    private String nome;
    @Temporal(value = TemporalType.DATE)
    //@NotBlank(message = "O nascimento deve ser informado")
    @NotNull(message = "O nascimento não pode ser nulo")
    @Column(name = "nascimento", nullable = false)
    private Calendar nascimento;
    @NotNull(message = "Sexo não pode ser nulo")
    @NotBlank(message = "Sexo deve ser informado")
    @Length(max = 15, message = "Sexo não deve ter mais do que {max} caracteres")
    @Column(name = "sexo", nullable = false, length = 15)
    private String sexo;
    @NotBlank(message = "O nome dos pais deve ser informado")
    @NotNull(message = "O nome dos pais não pode ser nulo")
    @Length(max = 200, message = "O nome dos pais não deve ter mais de {max}caracteres")
    @Column(name = "nomePais", nullable = false, length = 200)
    private String nomePais;
    @ManyToOne
    @JoinColumn(name = "usuario", referencedColumnName = "id", nullable = false)
    @ForeignKey(name = "fk_usuario")
    @NotNull(message = "O usuario não pode ser nulo")
```

Fonte: Próprio autor

Anotações contidas no código fonte da classe possibilitam que a biblioteca de persistência realize o mapeamento objeto relacional dos elementos persistidos. A anotação *@Entity* apresenta a classe como uma entidade representada no banco de dados, o nome da tabela é definido pela anotação *@Table (name = "nome da tabela")*. Cito também a anotação *@Column* que define as colunas da tabela no banco de dados. Outra anotação é *@Id* que define o identificador, a chave primária das tabelas que é gerado automaticamente com o uso da anotação *@GeneratedValue* com geração sequencial pela estratégia *GenerationType.SEQUENCE*.

As anotações *@NotBlank* e *@NotNull* são usadas para verificação de *string*, possibilitando que a biblioteca verifique os dados da classe para não armazenar valor em branco e não aceitação de valores nulos. Ainda referente a atributos com *string* a anotação *@Length* define o tamanho máximo permitido pelo campo. Para atributos relacionados a tempo, *@Temporal* é anotação que define se o atributo armazenará datas ou horário.

Para a representação de relacionamentos muitos para um são utilizadas as anotações *@ManyToOne* e *@JoinColumn,* que definem o nome da coluna que será criada com a referência a chave primária do outro lado da relação.

6.2 Camada DAO

A camada DAO é responsável pelo gerenciamento da comunicação entre a camada de controle e a camada de modelo. Classes DAO utilizam um objeto *EntityManager* como gestor de entidades, com a conexão estabelecida entre o banco de dados e o sistema é possível executar as principais operações: inserção, exclusão, leitura e atualização.

Foi criado um DAO genérico permitindo a reutilização de código, pois quando estendido, pode ter seus métodos utilizados pelas classes da camada de modelo. Ele fornece recursos de paginação, busca, filtro e ordenação de resultados de pesquisa. A Figura 19 apresenta um fragmento do código fonte da classe DAOGenerico.java.

Figura 19 - DAOGenerico

```
public class DAOGenerico <T> implements Serializable {

    private List<T> listaObjetos;
    private List<T> listaTodos;
    protected Class classePersistente;
    protected String mensagem = "";
    protected EntityManager em;
    protected String ordem = "id";
    protected String filtro = "";
    protected Integer maximoObjetos = 8;
    protected Integer posicaoAtual = 0;
    protected Integer totalObjetos = 0;

    public DAOGenerico(){
        em = EntityManagerUtil.getEntityManager();
    }

    public List<T> getListaObjetos() {
            String jpql = "from " + classePersistente.getSimpleName();
        String where = "";
        filtro = filtro.replaceAll("[';-]", "");
        if (filtro.length() > 0){
            if (ordem.equals("id")){
                try {
                    Integer.parseInt(filtro);
                    where += " where " + ordem + " = '" + filtro + "' ";
```

Fonte: Próprio autor

Para utilizar o DAO Genérico as classes DAO estendem do *DAOGenerico* e recebem como parâmetro uma classe persistente da camada de modelo e implementa interface *Serializable.* A classe *IdadeDAO* é apresentado na Figura 20.

Figura 20- Classe IdadeDAO

```
package br.edu.ifsul.dao;

import br.edu.ifsul.modelo.Idade;
import java.io.Serializable;

/**
 *
 * @author Anderson
 */
public class IdadeDAO<T> extends DAOGenerico<Idade> implements Serializable{
    public IdadeDAO(){
        super();
        classePersistente = Idade.class;
        ordem = "descricao";
    }

}
```

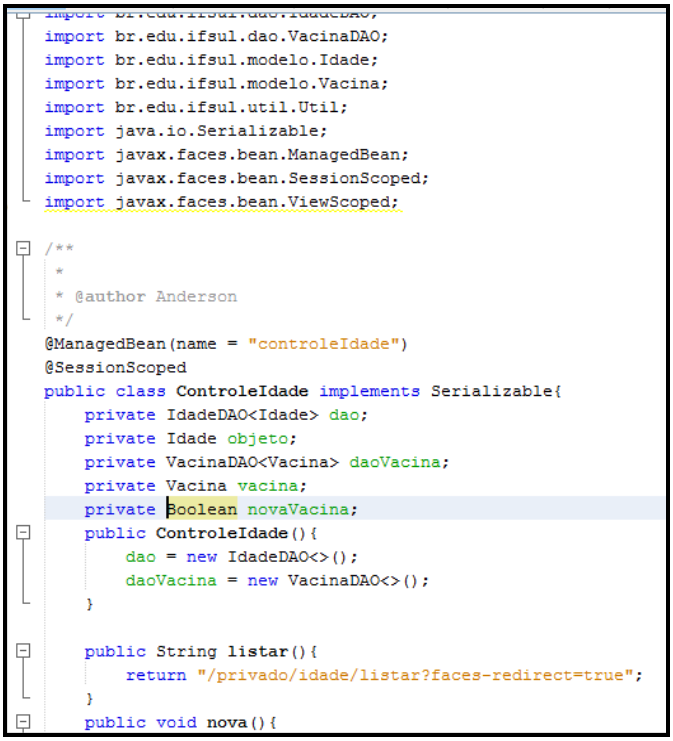
Fonte: Próprio autor

6.3 Camada de controle

A camada de controle existe para a comunicação de componentes da interface gráfica com a camada de persistência, fornecendo métodos para que estes componentes possam requisitar a camada DAO para manipular dados informados em listagens e formulários. É necessário fazer uso da anotação *@ManagedBean* e o nome para requisição é definido no atributo *name.*

A verificação para a inclusão de novos registros, alteração, exclusão, paginação além de preenchimento de campos com data são as ações atribuídas ao controlador. *A*baixo um trecho de código fonte de um controlador da aplicação, representado na Figura 21.

Figura 21 - Controle Idade



```
import br.edu.ifsul.dao.VacinaDAO;
import br.edu.ifsul.modelo.Idade;
import br.edu.ifsul.modelo.Vacina;
import br.edu.ifsul.util.Util;
import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.bean.ViewScoped;

/**
 *
 * @author Anderson
 */
@ManagedBean(name = "controleIdade")
@SessionScoped
public class ControleIdade implements Serializable{
    private IdadeDAO<Idade> dao;
    private Idade objeto;
    private VacinaDAO<Vacina> daoVacina;
    private Vacina vacina;
    private Boolean novaVacina;
    public ControleIdade(){
        dao = new IdadeDAO<>();
        daoVacina = new VacinaDAO<>();
    }

    public String listar(){
        return "/privado/idade/listar?faces-redirect=true";
    }
    public void nova(){
```

Fonte: Próprio autor

A classe que tem anotação *@SessionScoped* que define o escopo de sessão, e possui os métodos *nova(), salvar()* para armazenar os dados da Idade. E os métodos *editar* e *remover* recebem o *ID* como parâmetro permitindo respectivamente realizar edição e exclusão de registro. O método *listar()* retorna uma *string* que possui a *URL* para direcionar até uma página *web.*

O método *nova()* apenas instancia um novo objeto. O método *salvar()* realiza o teste se o objeto já existe no banco de dados verificando se o *ID* é nulo e chama o objeto definido como *dao* e com o método *persist(objeto)* faz a persistência no banco de dados. Caso o *ID* não for nulo significa que o objeto já existe no banco de dados, sendo assim o controlador faz a chamada do atributo *dao* e com o método *merge(objeto)* altera no banco de dados, conforme apresentado na Figura 22.

Figura 22 - Métodos Controle Idade

```
public void nova(){
    objeto = new Idade();
}
public void salvar(){
    boolean persistiu = false;
    if(objeto.getId()== null){
        persistiu = dao.persist(objeto);
    }else{
        persistiu = dao.merge(objeto);
    }
    if(persistiu){
        Util.mensagemInformacao(dao.getMensagem());
    }else {
        Util.mensagemErro(dao.getMensagem());
    }
}
public void editar(Integer id){
    objeto = dao.localizar(id);
}

public void remover(Integer id){
    objeto = dao.localizar(id);
    if(dao.remove(objeto)){
        Util.mensagemInformacao(dao.getMensagem());
    }else{
        Util.mensagemErro(dao.getMensagem());
    }
}
public void novaVacina(){
    vacina = new Vacina();
    novaVacina = true;
```

Fonte: Próprio autor

6.4 Camada de Visão

A camada de visão implementa a interface gráfica e é composta por páginas *.xhtml* desenvolvidas a partir do *framework Java Server Faces.* O arquivo *template.xhtml* é o corpo do *layout* básico com menu e áreas editáveis onde é apresentado o conteúdo das páginas com informações do banco de dados, tela de edição e formulário. A Figura 23 apresenta o arquivo *template.xhtml.*

Figura 23 - Template

```
<!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:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">
    <f:view encoding="ISO-8859-1" contentType="text/html">
        <h:head>
            <title><ui:insert name="titulo">titulo da pagina</ui:insert></title>
            <h:outputStylesheet library="css" name="estilos.css"/>
            <h:outputScript library="js" name="scripts.js"/>
        </h:head>
        <p:layout fullPage="true">
            <p:layoutUnit position="north" header="Sistema de controle de vacinas" style="text-align: center">
                <h:form ...36 linhas />
            </p:layoutUnit>
            <p:layoutUnit position="center">
                <ui:insert name="conteudo">

                </ui:insert>
            </p:layoutUnit>
            <ui:insert name="dialogos">

            </ui:insert>
            <p:layoutUnit position="south">
                <ui:include src="/templates/ajaxstatus.xhtml"/>
                <div align="center">Anderson de Azeredo Ribeiro - Aluno do IFSul</div>
            </p:layoutUnit>
        </p:layout>
    </f:view>
```

Fonte: Próprio autor

O sistema possui um menu e cada um dos seus itens executa uma ação no controlador e redireciona a navegação para tela correta. A propriedade *rendered* do elemento *submenu* realiza testes para saber se o usuário está logado, e se este usuário tem permissão de administrador para mostrar todas as manutenções disponíveis, caso contrário o usuário terá acesso somente a página de listagem de criança. A Figura 24 apresenta o trecho de código.

Figura 24 - Menu do sistema

```
<h:form id="menu">
    <p:megaMenu>
        <p:submenu label="Basicos" rendered="#{controleLogin.usuarioLogado != null and (
                                              controleLogin.usuarioLogado.administrador == true)}">
            <p:column>
                <p:submenu label="Cadastros">
                    <p:menuitem action="#{controleVacina.listar()}" value="Manutençao de Vacinas"
                                ajax="false"/>
                    <p:menuitem action="#{controleUsuario.listar()}" value="Manutençao de Usuarios"
                                ajax="false"/>
                    <p:menuitem action="#{controleIdade.listar()}" value="Manutençao de Idades"
                                ajax="false"/>
                    <p:menuitem action="#{controleCrianca.listar()}" value="Manutençao de Crianças"
                                ajax="false"/>
                    <p:menuitem action="#{controleVacinacao.listar()}" value="Manutençao de Vacinação"
                                ajax="false"/>
                </p:submenu>
            </p:column>
        </p:submenu>
        <p:submenu label="Sua(s) Criança(s)" rendered="#{controleLogin.usuarioLogado != null and
                                                        controleLogin.usuarioLogado.administrador == false}">
            <p:column>
                <p:submenu label="Cadastros">
                    <p:menuitem action="#{controleCrianca.listar()}" value="Manutençao de Crianças"
                            ajax="false"/>
                </p:submenu>
            </p:column>
        </p:submenu>
            <p:menuitem value="Entrar no sistema"
                        rendered="#{controleLogin.usuarioLogado == null}"
                        ajax="false"
```

Fonte: Próprio autor

O arquivo listar.xhtml apresentado na Figura 25 é um arquivo que contém paginação, filtro e ordenação dos registros, é o arquivo responsável por listar na tela todas as informações contidas no banco de dados. Também possuem botões para alteração e exclusão de registros. Além de também ter botões de navegação entre as páginas.

Figura 25- listar.xhtml

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:p="http://primefaces.org/ui"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <ui:composition template="/templates/template.xhtml">
        <ui:define name="titulo">Listagem de Usuarios</ui:define>
        <ui:define name="conteudo">
            <h:form rendered="#{controleLogin.usuarioLogado != null and (controleLogin.usuarioLogado.administrador == true)}">
                <p:messages/>
                <p:growl/>
                <p:commandButton value="Novo" action="#{controleUsuario.novo()}"
                                 ajax="false" icon="ui-icon-plus"/>
                <div align="center">
                    <p:panelGrid columns="11">
                        <p:outputLabel value="Ordem: "/>
                        <p:selectOneMenu value="#{controleUsuario.dao.ordem}">
                            <f:selectItem itemLabel="Id" itemValue="id"/>
                            <f:selectItem itemLabel="Login" itemValue="login"/>
                            <f:selectItem itemLabel="Senha" itemValue="senha"/>
                            <f:selectItem itemLabel="E-mail" itemValue="email"/>
                            <f:selectItem itemLabel="Telefone" itemValue="telefone"/>
                            <f:selectItem itemLabel="Crianças" itemValue="criancas"/>
                            <p:ajax update="listagem"/>
                        </p:selectOneMenu>
                        <p:outputLabel value="Filtro"/>
                        <p:inputText value="#{controleUsuario.dao.filtro}" size="15">
                            <p:ajax update="listagem" event="keyup"/>
                        </p:inputText>
                        <p:commandButton value="Filtrar" update="listagem" icon="ui-icon-search"/>
```

Fonte: Próprio autor

6.4.1 Conversores

Viabilizando a conversão de objetos de *Java* para *String* e vice-versa foi necessário implementar conversores que implementam a interface *Converter*. Ao criar o conversor que implementa os métodos abstratos, o método *getAsObject* recebe como parâmetro uma *string* e retorna um objeto da classe do conversor. Por sua vez o método *getAsString* recebe um objeto e retorna uma *string*. A Figura 26 apresenta o código fonte do conversor de *Calendar.*

Figura 26 - ConverterCalendar

```
@FacesConverter(value = "converterCalendar")
public class ConverterCalendar implements Serializable, Converter{

    private SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

    @Override
    public Object getAsObject(FacesContext fc, UIComponent uic, String string) {
        try {
            Calendar retorno = Calendar.getInstance();
            retorno.setTime(sdf.parse(string));
            return retorno;
        } catch (Exception e){
            return null;
        }
    }

    @Override
    public String getAsString(FacesContext fc, UIComponent uic, Object o) {
        if (o == null){
            return null;
        }
        Calendar obj = (Calendar) o;
        return sdf.format(obj.getTime());
    }
}
```

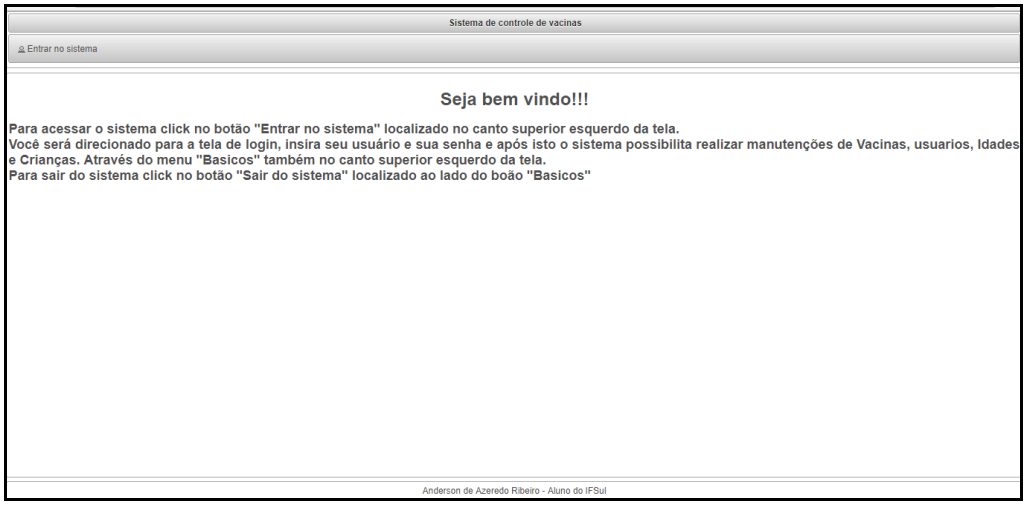
Fonte: Próprio autor

6.5 Apresentação de execução do sistema

Esta seção apresenta o funcionamento do sistema de controle de vacinas, partindo do acesso a página inicial, autenticação do usuário através do *login*, visualização das informações na tela, realização de cadastro, edição e exclusão.

Acessando o sistema o usuário recebe as boas vindas, então a tela é apresentada na Figura 27, sendo o mesmo orientado a realizar o *login*, inserindo usuário e senha válidos.

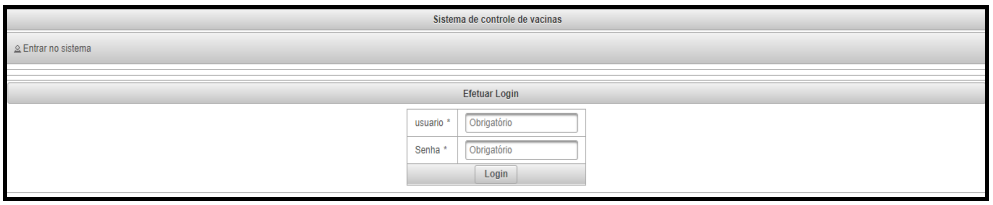
Figura 27 - Tela inicial



Fonte: Próprio autor

Ao clicar no botão “Entrar no sistema” o usuário é direcionado para a tela de *login* representada na Figura 28.

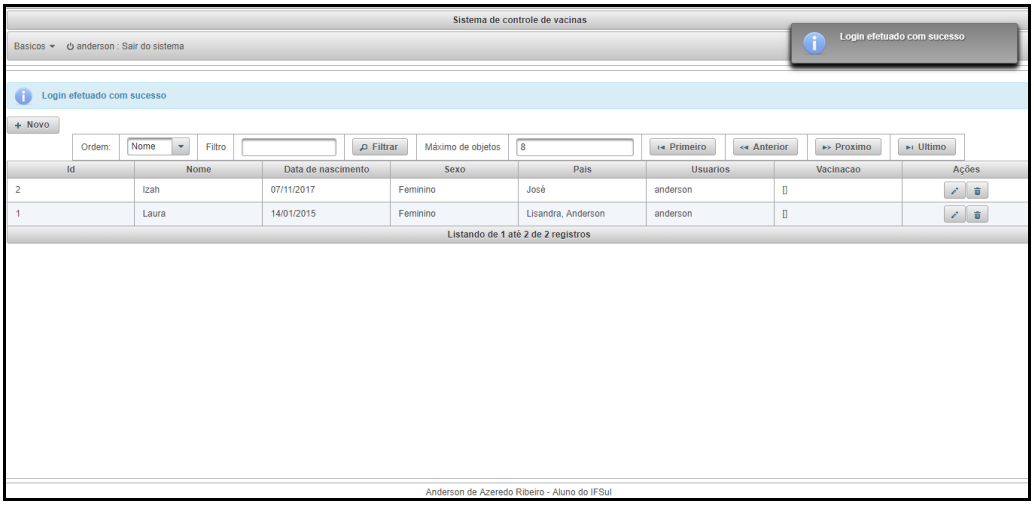
Figura 28 - Login



Fonte: Próprio autor

Ao realizar o *login* com sucesso, o usuário é direcionado para a tela de listagem de crianças, sendo exibido somente as crianças cadastradas pelo usuário logado. Além da visualização também é possível cadastrar uma nova criança, editar uma criança já cadastrada e até mesmo excluir um registro. A Figura 29 apresenta esta tela.

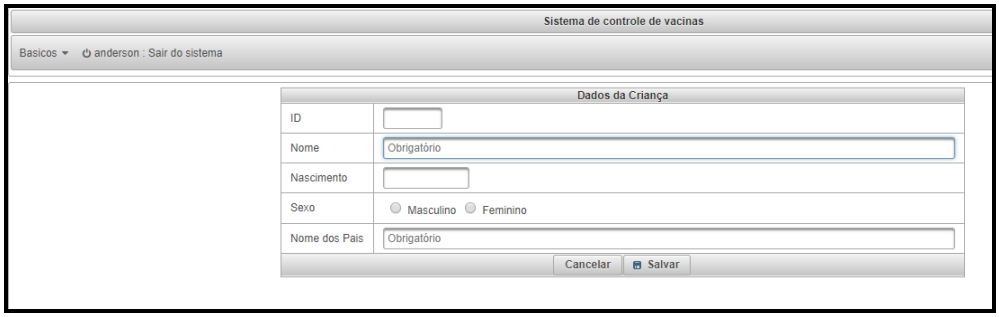
Figura 29 - Listagem de crianças



Fonte: Próprio autor

O cadastro de uma nova criança inicia com um clique no botão “+ Novo”, será exibida uma tela para o usuário inserir os dados da criança que será cadastrada. A Figura 30 ilustra esta tela.

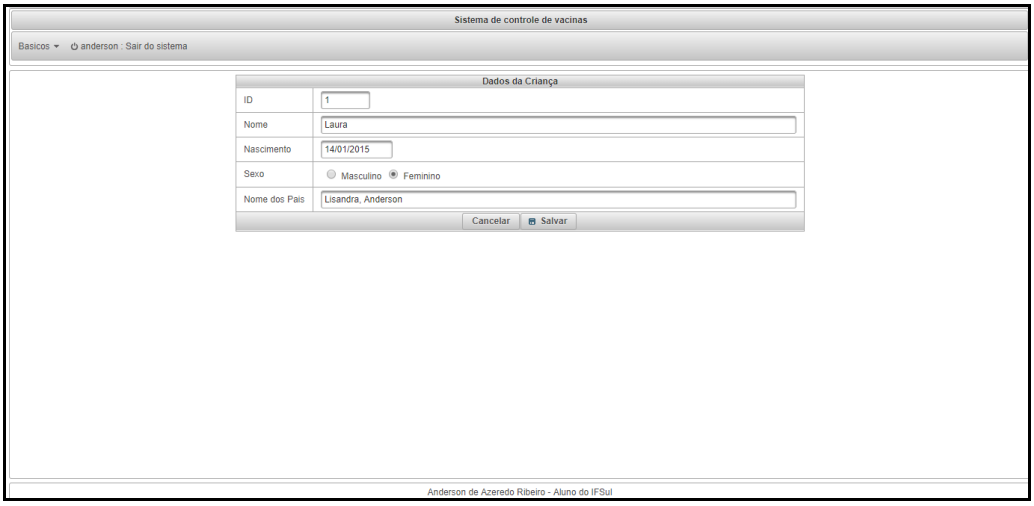
Figura 30 - Dados da criança



Fonte: Próprio autor

A edição de um registro é possível clicando no botão “Editar” representado pelo ícone com desenho de um lápis na coluna “Ações”, que direciona para uma tela já com todos os dados do objeto a ser editado carregados, conforme exibido na Figura 31.

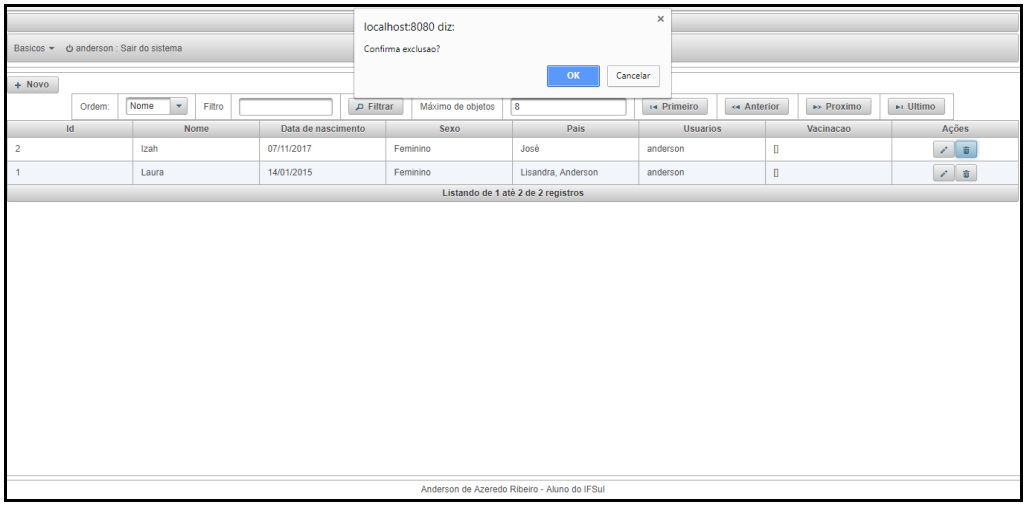
Figura 31 - Tela de edição de criança



Fonte: Próprio autor

Além de inserir e editar a aplicação também permite o usuário excluir registros, está funcionalidade é acessada clicando no botão “Remover” representado pelo ícone com o desenho de uma “Lixeira”, segundo botão da coluna “Ações”. Esta ação pode ser vista na Figura 32.

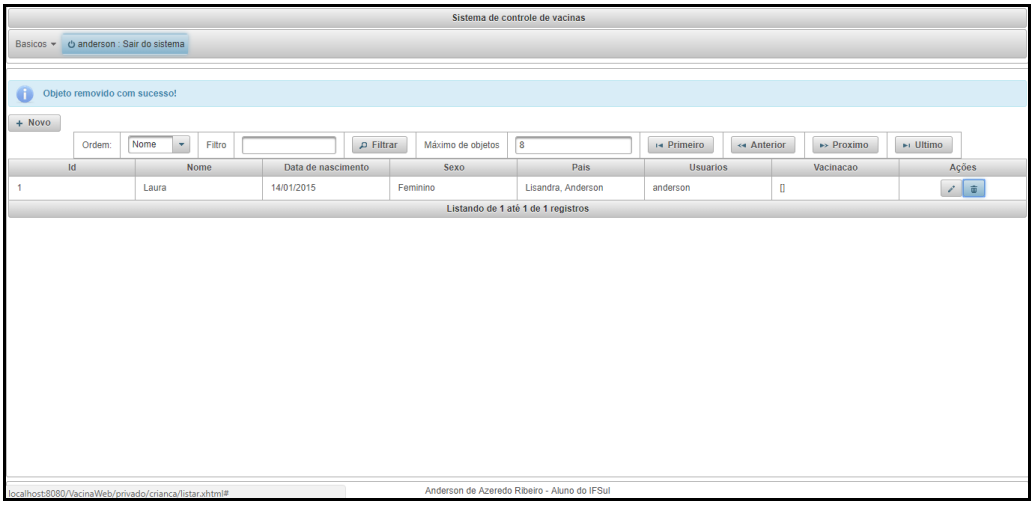
Figura 32 - Tela de exclusão



Fonte: Próprio autor

Caso o usuário desejar sair do sistema deverá clicar no botão “Sair do sistema” localizado ao lado do botão de menu “Básicos”. Realizando este procedimento o sistema apresenta a tela de *login* conforme ilustra a Figura 33.

Figura 33 - Opção para sair do sistema



Fonte: Próprio autor

7 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo o desenvolvimento de uma aplicação para controle de vacinas, com o intuito de um dia ser adotado para acabar com a necessidade de apresentação da carteirinha de vacinação no ato de realizar uma vacina e, além disso, apresentando facilidade na recuperação das informações. Para o seu desenvolvimento, foi necessário a utilização da plataforma Java EE, a JPA para persistência de dados com *Hibernate*, o framework JSF e a biblioteca do *Primefaces.*

Durante o desenvolvimento do trabalho houveram diversas dificuldades, mas a maior delas foi a falta de tempo para o desenvolvimento completo da aplicação. Porém mesmo com as dificuldades a versão atual do sistema possibilita realizar funções básicas de cadastro, visualização, alteração e exclusão, sendo possível realizar manutenção de vacinas, usuários, idades, crianças e vacinação. Além disto, foi implementado um sistema de *login* para acesso das funcionalidades do sistema.

Ao fim do trabalho pode-se concluir boa parte dos objetivos do projeto que contribuiu muito para um maior aprendizado das tecnologias utilizadas no desenvolvimento. Para trabalhos futuros sugiro integração da aplicação com o Google agenda, assim permitindo o usuário poder acompanhar pela sua agenda as datas para a vacinação. Também o desenvolvimento de novas funcionalidades, como por exemplo, o crescimento da criança registrando peso e altura, possibilitando geração de gráficos.

8 REFERÊNCIAS

CONSELHO FEDERAL DE ODONTLOGIA. **Software publico para gestão de vacina é lançado**. Disponível em: <http://cfo.org.br/todas-as-noticias/software-publico-para-gestao-de-vacinacao-e-lancado/#>. Acesso em: 27 mar. 2017.

SILVEIRA, A. S. D. A et al. Controle de vacinação de crianças matriculadas em escolas municipais da cidade de São Paulo. **Revista da Escola de Enfermagem da USP**, v. 41, n. 2, p. 299-305, 2007.

FURGERI, Sérgio. **Java 6 - ensino didático**: Desenvolvendo e Implementando Aplicações. 2 ed. São Paulo: Érica, 2008.

GEARY, David; HORSTMANN, Cay. **Core JavaServer Faces:** Tradução da 3ª edição. 3 ed. Rio de Janeiro: Starlin Alta, 2012.

GOOGLE PLAY. **Vacinação em dia**. Disponível em: <https://play.google.com/store/apps/details?id=com.digital.vacinacao&hl=en>. Acesso em: 21 fev. 2017.

GOOGLE PLAY. **Vacinas em dia**. Disponível em: <https://play.google.com/store/apps/details?id=br.com.minhavida.vacinas&hl=pt_BR>. Acesso em: 26 fev. 2017.

INSTITUTO DE TECNOLOGIA EM IMUNOBIOLÓGICOS BIO-MAGUINHOS. **Calendário de vacinação**. Disponível em: <https:// https://www.bio.fiocruz.br/index.php/calendario-de-vacinacao>. Acesso em: 06 dez. 2017.

DEITEL, M. Harvey; DEITEL, J. Paul. **Java Como Programar**. 4. ed. Porto Alegre: Bookman, 2002. 1386 p.

PROGRAMA NACIONAL DE IMUNIZAÇÕES. **Aplicativo vacinação em dia**. Disponível em: <http://portalarquivos.saude.gov.br/campanhas/pni/>. Acesso em: 25 mar. 2017.

SLIDE SHARE. **Jsf e outras tecnologias**. Disponível em: <https://pt.slideshare.net/eduardo.bregaida/jsf-e-outras-tecnologias-java-web-imesjava>. Acesso em: 02 jun. 2017.