UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
SISTEMAS DE INFORMAÇÃO

RELATÓRIO DE ESTÁGIO SUPERVISIONADO
DESENVOLVIMENTO WEB COM JAVA SERVER FACES E O FRAMEWORK HIBERNATE

MURILO RODRIGUES MOREIRA

CUIABÁ – MT
2015

UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
SISTEMAS DE INFORMAÇÃO

RELÁTORIO DE ESTÁGIO SUPERVISIONADO DESENVOLVIMENTO WEB COM JAVA SERVER FACES E O FRAMEWORK HIBERNATE

MURILO RODRIGUES MOREIRA

Relatório apresentado ao Instituto de Computação da Universidade Federal de Mato Grosso, para obtenção do título de Bacharel em Sistemas de Informação.

CUIABÁ – MT
2015

UNIVERSIDADE FEDERAL DE MATO GROSSO

COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

MURILO RODRIGUES MOREIRA

Relatório de Estágio Supervisionado apresentado à Coordenação do Curso de Sistemas de Informação como uma das exigências para obtenção do título de Bacharel em Sistemas de Informação da Universidade Federal de Mato Grosso.

Aprovado por:

Prof. Msc. Nilton Hideki Takagi
Instituto de Computação
(Coordenador de Estágios)

Prof. Dr. Thiago Meirelles Ventura
Instituto de Computação
(ORIENTADOR)

Jardel Ribeiro
Gerente de Sistema e Banco de Dados
SEJUDH - MT
(SUPERVISOR)

Prof. Dr. Raphael de Souza Rosa Gomes
Instituto de Computação
(CONVIDADO)

DEDICATÓRIA

Dedico esse trabalho primordialmente a Deus "Porque dele e por ele, e para ele, são todas as coisas; glória, pois, a ele eternamente. Amém." (Romanos 11:36). À minha esposa, Talita, que de forma especial e carinhosa me deu força e coragem, me apoiando nos momentos de dificuldade, e, aos meus pais, pois me proporcionaram o estudo de base para que eu pudesse obter êxito nessa caminhada.

AGRADECIMENTOS

Agradeço aos meus professores, dentre eles ao professor Thiago Meirelles Ventura que me orientou na elaboração deste trabalho, aos meus colegas de trabalho que contribuíram com o meu desenvolvimento como profissional compartilhando conhecimento e experiências e, aos meus colegas de classe que também colaboraram com o meu crescimento acadêmico.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

SEJUDH	Secretaria de Estado de Justiça e Direitos Humanos
UFMT	Universidade Federal de Mato Grosso
SEVOCAB	Systems e Software Enginnering Vocabulary
JSF	Java Server Faces
IDE	Integrated Development Environment
JPA	Java Persistence API
JVM	Java Virtual Machine
API	Application Programming Interface
SGBD	Sistema Gerenciador de Banco de Dados
JDBC	Java Database Connectivity
EJB	Enterprise Java Beans
OO	Orientação a Objeto
MVC	Model-View- Controller
CSS	Cascading Style Sheets
XHTML	eXtensible Hypertext Markup Language
ORM	Object Relational Mapper
SQL	Structured Query Language
EL	Expression Language

RESUMO

Este trabalho tem o objetivo de apresentar as atividades desenvolvidas no estágio supervisionado na Secretaria de Estado de Justiça e Direitos Humanos (SEJUDH). O intuito principal do estágio foi estruturar um ambiente de desenvolvimento para elaborar sistemas web flexível e de baixo custo com a intenção de atender a(s) demanda(s) da SEJUDH. Foram abordados temas como Banco de Dados, Desenvolvimento Web com Java Server Faces (JSF) utilizando o padrão de projeto MVC, framework de persistência com Hibernate e Primefaces como *framework front-end*. Ao final, foram apresentados os resultados obtidos no desenvolvimento de uma aplicação web com o uso das tecnologias citadas.

Palavras chaves: Ambiente de desenvolvimento Java web, Orientação a Objeto, Persistência em Banco de Dados, Primefaces.

Introdução

A cada ano a Internet vem evoluindo e com o surgimento da Web 2.0, o dinamismo e a interatividade dos sites com os usuários ganharam uma importância fundamental. Cada vez mais pessoas tem adquirido o acesso à rede mundial de computadores, através de seus computadores pessoais ou de dispositivos móveis (Tokarski, 2010).

Em razão disso, existe uma demanda crescente por desenvolvimento para web, como a criação ou reformulação de sites, sistemas para a Intranet de uma organização. Neste contexto, a Gerência de Sistemas de Banco de Dados (GESB) da Secretaria de Estado de Justiça e Direitos Humanos (SEJUDH) se propôs a pesquisar e estruturar um ambiente de desenvolvimento com a intenção de poder entregar soluções relacionadas ao desenvolvimento de sistemas web.

A GESB procurou escolher tecnologias no mercado que sejam flexíveis, seguras e de custo reduzido. Assim, a equipe decidiu escolher a linguagem Java como sendo o elemento principal do software, pois por conta disso a escolha das outras ferramentas que irão compor o ambiente de desenvolvimento será pensando na integração com a linguagem principal do sistema. Para o desenvolvimento de aplicações web utilizando a linguagem Java foi utilizado a *framework* Java Server Faces (JSF), a qual é destinada à criação de componentes de interface gráfica. Juntamente com o JSF, o framework PrimeFaces também é usado para estruturar a visualização das páginas do sistema de forma fácil e prática (Tokarski, 2010).

Existem *frameworks* responsáveis pela integração entre o sistema Java e o banco de dados, tais como o Hibernate e a especificação JPA (*Java Persistence API*). Desde a sua criação, o Hibernate exerce a função de ponte entre a Orientação a Objeto (OO) e a persistência de dados em um banco de dados relacional. O uso dessas ferramentas traz produtividade para o desenvolvimento de aplicações web (Saab, 2011).

O desafio do trabalho foi abordar as ferramentas e metodologias mais utilizadas para o desenvolvimento de aplicações web, as quais foram usadas para implementar o Sistema de Gestão Penitenciária (SIGEPEN), que tem o objetivo de auxiliar no gerenciamento das atividades administrativas das unidades prisionais da

SEJUDH. O software foi planejado para ser um sistema web, pois dessa forma irá atender todas as unidades que possuem a capacidade de ter acesso à Internet.

1. REVISÃO DE LITERATURA

Nesta seção foram abordados os conceitos que serviram como referência para o desenvolvimento das atividades desempenhadas durante o estágio. Os conhecimentos adquiridos durante as disciplinas da faculdade, foram de extrema importância para a compreensão e evolução no desenvolvimento das atividades realizadas no decorrer do estágio.

1.1 Engenharia de Software

Para a IEEE Systems e Software Enginnering Vocabulary (SEVOCAB) a Engenharia de Software é uma área do conhecimento da computação focada na aplicação de uma sistemática, abordagem quantificável para o desenvolvimento, operação e manutenção de sistemas de software aplicando tecnologias e práticas de sistemas de informação. A prática da Engenharia de Software é composta por um conjunto de conceitos, princípios, metodologias e ferramentas que precisam ser levados em consideração para planejar e desenvolver um sistema (Pressman,2011).

Há diversos tipos de modelos de ciclo de vida do software. Um modelo denominado Cascata é um modelo clássico, utilizado desde o final da década de 70, que possui seis fases que são seguidas quando se cria um produto de software. O modelo Cascata é composto por etapas que servem como base para diversas variações de modelos que podem mudar de empresa para empresa, porém elas são suficientes para alcançar a satisfação do produto desenvolvido. Da mesma forma, a nomenclatura exata de cada etapa pode sofrer alterações de empresa para empresa. Geralmente os nomes adotados nestas etapas são escolhidos para serem os mais genéricos possíveis, para que seja de fácil entendimento dos envolvidos no projeto (Schach, 2009).

Segundo Stephen R. Schach (2009) as etapas que compõem o Modelo Cascata normalmente incluem as seguintes fases:

1 - *Fase de levantamento de necessidades:* Nesta fase as necessidades e requisitos do cliente são elencados após os conceitos serem explorados e lapidados.

2 - *Fase de análise (especificação)*: Durante esta fase os requisitos do cliente são analisados e validados para que seja definido as especificações daquilo que é esperado que o produto execute. Nesta etapa é criado o documento de especificações e também o plano de gerenciamento do projeto de software o qual descreve de forma detalhada o desenvolvimento do software.

3 - *Fase de projeto*: Nesta fase as especificações são subdivididas em módulos que em seguida cada módulo é detalhadamente projetado. Assim a junção desses processos descreve como o software realizará aquilo que se espera.

4 - *Fase de implementação*: Durante essa fase ocorre a codificação e teste unitários ou modular e em seguida os módulos são combinados e testados como um todo, fazendo assim a integração dos componentes. Após a aprovação dos desenvolvedores em termos de funcionalidade do software, o produto é enviado para que seja feito os testes de aceitação. Esta etapa de implementação é finalizada quando o produto é validado pelo cliente e configurado para uso dele.

5 - *Manutenção pós-entrega*: Esta fase ocorre já com o produto em uso pelo cliente. Durante a utilização do software, ele pode sofrer modificação para atender às necessidades exigidas pelo cliente. Essas manutenções abrangem diversos tipos de alterações que pode ser:

- *Manutenções Corretivas*, a qual consiste na correção de possíveis erros, permanecendo inalteradas as especificações do produto;
- *Manutenções de Aperfeiçoamento*, que são modificações solicitadas pelo cliente no intuito de melhorar uma determinada funcionalidade e/ou na diminuição do tempo de resposta durante a execução em um procedimento do sistema;
- *Manutenções Adaptativas*, que estão relacionadas às mudanças feitas em resposta a modificações que ocorrem ao ambiente no qual o software é executado. Isso pode ser ocasionado por uma nova normalização da própria empresa e/ou regulamentações governamentais.

6 - *Retirada do produto*: Nesta etapa ocorre quando o software é retirado de serviço, pelo fato de não mais atender às expectativas do cliente.

Para garantir que a execução do projeto esteja sendo realizado de forma correta, é preciso revisar os requisitos com os interessados para ter a exatidão de que aquilo que o analista/desenvolvedor entendeu é realmente o que os clientes desejam. Por mais que as partes envolvidas estejam em consentimento, as especificações podem sofrer alterações e isso poderá ocorrer ao longo de todo o projeto. Apesar das dificuldades durante os processos de software, essas etapas são muito importantes para o desenvolvimento do projeto, pois possibilita a todos os envolvidos uma visão do caminho a ser trilhado para alcançar o objeto principal com sucesso (Pressman, 2011).

1.2 Banco de dados

Conforme Date (2004), o sistema de banco de dados é considerado um sistema computadorizado de manutenção de registros. Pode ser visto também como um repositório que armazena um conjunto de arquivos de dados informatizados, os quais os usuários de uma organização podem solicitar ao sistema que realize diferentes operações que envolva tais arquivos, como por exemplo:

- Inclusão de novos registros ao banco de dados;
- Acrescentar dados em arquivos existentes;
- Pesquisar dados de arquivos existentes;
- Remover dados de arquivos existentes;
- Editar dados de arquivos existentes;
- Excluir arquivos existentes do banco de dados;

Basicamente um banco de dados é um conjunto de dados que, geralmente, descreve as atividades de uma ou mais organizações relacionadas. Um exemplo básico é o banco de dados de uma escola que pode conter informações sobre (Gehrke, 2008):

- *Entidade como aluno, professores, turmas, disciplinas.*
- *Relacionamentos entre as entidades, como a matrícula dos alunos nas disciplinas, as disciplinas ministradas pelos professores, e até mesmo o uso de salas por disciplinas, entre outros.*

Na Figura 1 é possível observar os principais componentes que envolvem um sistema de banco de dados. São eles: dados, software, hardware e usuários (Date, 2004).

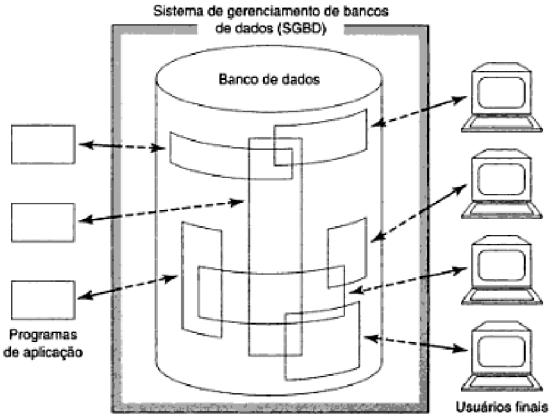

Figura 1 - Representação de um sistema de banco de dados. Fonte - C. J. Date

No contexto de sistemas de banco de dados, o termo dados refere-se ao que realmente é armazenado no banco de dados, ou seja, o dado é um elemento que mantém a sua forma bruta que pode ser um texto, uma imagem, vídeo, etc. Já o termo informação é utilizado para referenciar ao significado de um determinado dado. Os dados são gerenciados por um Sistema Gerenciador de Banco de Dados (SGBD) (Date, 2004).

Um SGBD é um conjunto de software responsável por executar com maior facilidade as tarefas que permite gerenciar os dados armazenados de um sistema. Os SGBDs possuem recursos que permitem gerenciar os dados de forma mais robusta, sendo assim mais eficiente do que uma coleção de arquivos do sistema operacional. Assim o suporte de um SGBD torna-se indispensável à medida que cresce o volume de dados e o número de usuários (Gehrke, 2008).

Gehrke (2008) destaca diversas vantagens ao se usar um SGBD para gerenciar dados, as quais são:

- *Independência de Dados:* Proporciona a independência dos dados em relação à aplicação, assim ocultando os detalhes de representação e armazenamento

de dados. O SGBD retorna apenas uma visão abstrata dos dados sem os detalhes.

- *Acesso Eficiente aos Dados:* O SGBD utiliza-se de diversos recursos computacionais sofisticados para a manipulação dos dados a fim de recuperá-los ao usuário de forma eficiente.
- *Integridade e Segurança dos Dados:* Quando os dados são acessados através do SGBD, o software tem a capacidade de forçar restrições de integridades dos dados no intuito de retornar uma determinada informação. Além disso, é possível manter o controle de acesso aos dados para os diferentes perfis de usuários.
- *Administração de Dados:* São mecanismos de gestão de como os dados podem ser manipulados pelos diversos grupos de usuários do sistema.
- *Acesso Concorrente e Recuperação de Falha:* O acesso corrente aos dados é implementado pelo SGBD de tal forma que os usuários podem ter a impressão que os dados estão sendo acessados por um único usuário de cada vez. O SGBD permite também que os usuários não sofram com os efeitos de falhas de sistema.
- *Tempo reduzido de Desenvolvimento de Aplicações:* Os SGBDs possuem funções que ao juntar-se com uma interface de alto nível aos dados, acaba tornando o desenvolvimento mais rápido do aplicativo, pois essas funções podem ser desenvolvidas diretas no SGBD, e assim não há a necessidade de depurar e testar na aplicação.

Na Figura 2 é possível observar as funções do SGBD de forma mais detalhada.

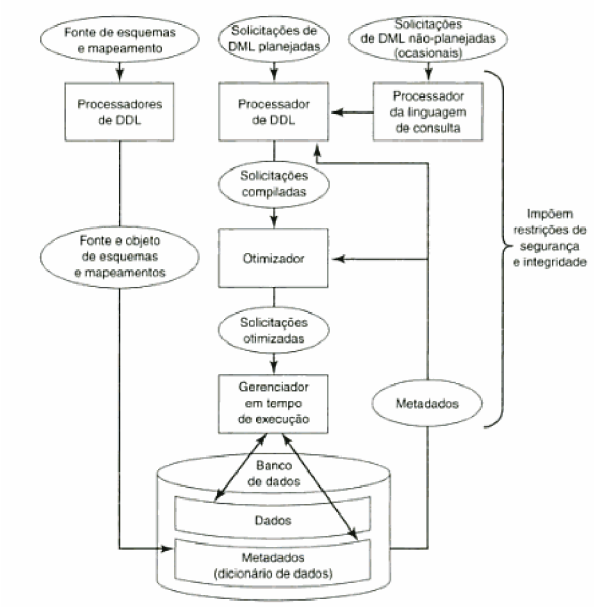

Figura 2 - Principais funções e componentes de um SGBD. Fonte - C. J. Date

A linguagem de consulta estrutura, ou SQL (Structured Query Language) foi originalmente desenvolvida como linguagem de consulta do SGBD relacional pioneiro da IBM, o System-R. Assim o SQL tornou-se a linguagem mais utilizada para criar, manipular e consultar SGBDs relacionais. Para a construção e definição dos esquemas externos e conceituais em um SGBD é utilizado a Linguagem de Definição de Dados (DDL - Data Definition Language). Os usuários conseguem criar, modificar e consultar dados em um SGBD através de uma Linguagem de Manipulação de Dados (DML - Data Manipulation Language) (Date, 2004).

1.3 Fundamentos da Programação Orientada a Objeto

A dificuldade dos sistemas de informação motivou a criação de metodologias que transformassem o processo de desenvolvimento de sistemas menos custoso e mais produtivo. Uma das metodologias que se destaca é a orientada a objeto, a qual aborda os seguintes objetivos:

1. A reutilização de códigos existentes pode proporcionar o aumento de produtividade do desenvolvimento do software;
2. Minimizar o custo da elaboração do software;
3. Código fonte do sistema de fácil manutenção;

A utilização do conceito de Orientação a Objetos teve início amplamente na década de 90, apesar de ter surgido nos anos 60. A partir disso, diversas linguagens de programação foram elaboradas para suportar os conceitos relacionados à metodologia orientada a objetos. Dentre várias linguagens, é possível destacar a linguagem de programação Java (Aragão, 2013).

A plataforma Java tornou bastante popular com o surgimento de um componente de software denominado Máquina Virtual Java, tradução de Java Virtual Machine – JVM. A JVM possibilita que o Java seja executado independente do sistema operacional ou dispositivo. Para que isso ocorra a JVM precisa estar disponível para o dispositivo ou sistema operacional no qual deseja executar o sistema Java (Figura 3) (Aragão, 2013).

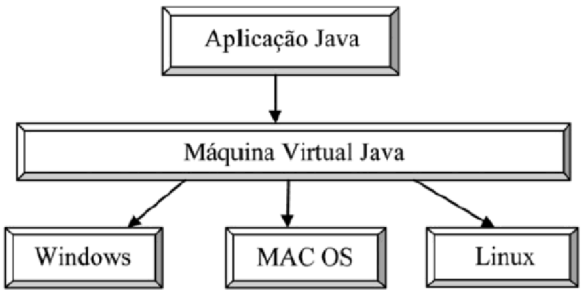

Figura 3 - Execução da aplicação Java em diversos Sistemas Operacionais. Fonte - Autor

O termo "Java" faz referência tanto para à plataforma computacional quanto para linguagem de programação. Em relação à linguagem de programação Java podem-se destacar as seguintes características:

- Orientação a Objetos
- Portável
- Gratuita
- Conjunto de Bibliotecas

A Orientação a Objetos é um modelo que representa toda uma filosofia para a elaboração de um software. Para entender melhor esta filosofia é preciso compreender os conceitos de classes e objetos. Uma classe é a representação de uma estrutura que define um tipo de dados, o qual pode conter atributos (variáveis) e também comportamentos (métodos) para utilizar esses atributos. Um Objeto por sua vez é uma instancia da classe, ou seja, ele é a execução da classe. A programação

Orientada a Objetos proporciona a organização durante o desenvolvimento do software (Aragão, 2013).

2. MATERIAS, TÉCNICAS E MÉTODOS.

Para que os projetos de sistemas web fossem construídos, a Gerência de Sistemas e Banco de Dados utilizou a ferramenta Eclipse IDE como ambiente de desenvolvimento. O ambiente de desenvolvimento foi composto pelos frameworks JSF (Java Server Faces), Java EE 7, Primefaces, Hibernate, pelo servidor de aplicação WildFly e o SGBD Oracle 11g.

2.1 Balsamiq Mockups

Balsamiq Mockups é uma ferramenta de design de interface do usuário que possibilita a criação de *wireframes* (também chamados de moldes ou protótipos de baixa fidelidade). A interface do usuário Balsamiq Mockups é composta de cinco áreas principais: a barra de ferramentas (Toolbar), a biblioteca de interface do usuário (UI Library), o Canvas, o painel de navegação (Navigator) e o painel de propriedades (Properties), como podem ser visualizados na Figura 4.

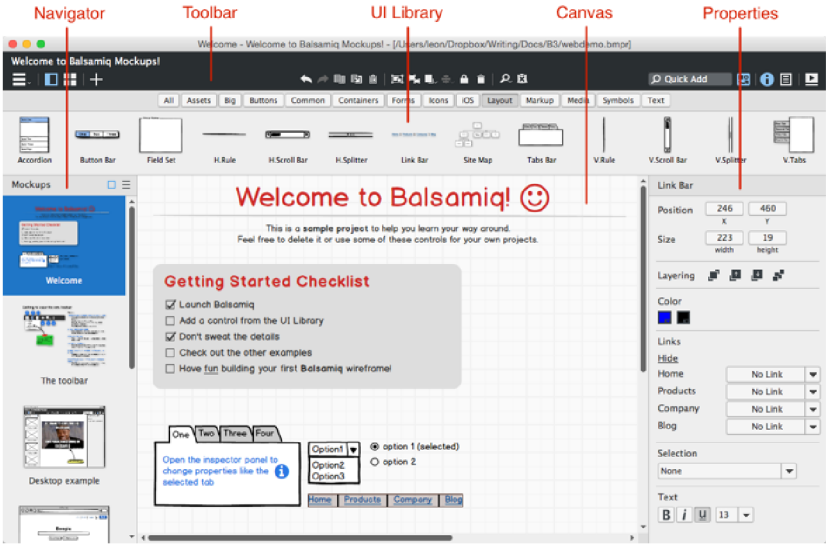

Figura 4 - Interface Gráfica do programa. Fonte - Autor

Essa ferramenta pode ser utilizada para gerar desenhos digitais das ideias do produto para facilitar a discussão e compreensão antes que qualquer código seja escrito. Isso facilita na etapa de concepção do negócio, representando os fluxos de telas e processos do sistema.

2.2 Eclipse IDE

O Eclipse é um Ambiente de Desenvolvimento Integrado (IDE) Java, baseado em software livre extensível. Basicamente é uma estrutura composta por um conjunto de serviços para desenvolvimento de aplicativos. Para desenvolvedores de software, o Eclipse é uma ferramenta importante para que possam ser criadas soluções para diversas finalidades tais como: aplicações Desktop, Web, JPA, JSF, XHTML, entre outras. Embora o Eclipse seja conhecido mais para o desenvolvimento com a linguagem Java, seu uso não se limita apenas a essa linguagem de programação, pois há também plugins que incluem suporte para linguagens de programação como C/C++, COBOL, e versões para PHP.

O Projeto Eclipse foi originalmente criado pela IBM em novembro de 2001 e suportado por um consórcio de fornecedores de software. A Eclipse Foundation foi criada em janeiro de 2004 como uma organização sem fins lucrativos independente para atuar como organizadora da comunidade Eclipse. Hoje, a comunidade do Eclipse é composta por pessoas e organizações de uma seção transversal do segmento de mercado de software.

O Eclipse é uma comunidade de código aberto cujos projetos são concentrados na criação de uma plataforma de desenvolvimento aberta composta de estruturas, ferramentas e tempos de execução extensíveis para desenvolvimento, implementação e gerenciamento de software por todo o ciclo de vida.

2.3 JSF

Atualmente a framework Java Server Faces (JSF) é a principal tecnologia para o desenvolvimento de aplicações web em Java que utiliza um modelo de interface gráficas baseado em eventos (Faria, 2015). O JSF segue o padrão arquitetural Model-View-Controller (MVC), o qual tem por objetivo separar a lógica

de negócio da lógica de apresentação da aplicação. O padrão MVC divide uma aplicação em três tipos de responsabilidades: modelo, visão e controlador.

- Modelo: encapsula os dados e as funcionalidades da aplicação. É responsável por representar os objetos de negócio, manter o estado da aplicação e fornecer ao controlador o acesso aos dados.
- Visão: É responsável pela interface do usuário. Ela que define a forma como os dados são apresentados e encaminha as ações do usuário para o controlador.
- Controlador: É responsável por ligar o modelo e a visualização, interpretando as solicitações do usuário, traduzindo para uma operação no modelo (onde são realizadas efetivamente as mudanças no sistema) e retornando a visualização adequada à solicitação.

MVC aborda a codificação de sistemas em camadas explícitas. Na camada de Modelo (Model) encontra-se a classes Java, que na maioria das vezes são classes que representam uma tabela do banco de dados. A camada de Visualização (View) é composta pelas ferramentas Primefaces, XHTML, JSF, JavaScript, Ajax e CSS. Já na camada de Controle (Controller) é possível destacar as classes e anotações do JSF, onde tais classes são responsáveis por direcionar o fluxo da aplicação fazendo a intermediação entre a Visualização e o Modelo.

Na Figura 5 é possível visualizar a arquitetura do MVC implementada pelo JSF, e compreender que o cliente/usuário realiza uma requisição ao sistema através de uma página JSF (View), a qual irá enviar essa requisição para os arquivos do sistema responsáveis por receber, tratar e retornar resposta às solicitações enviadas, que no caso da especificação JSF são os Managed Beans (Controller) que tem a função de controladores os quais podem buscar e retornar dados do banco de dados, acessar classes de negócio e classes modelos.

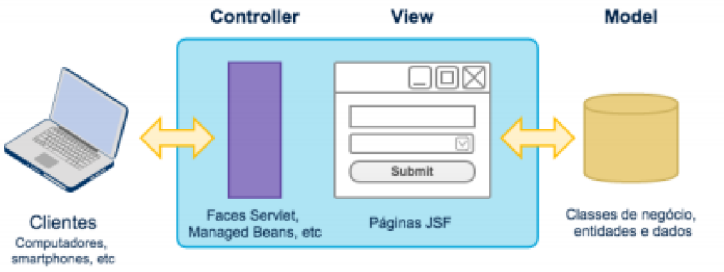

Figura 5 - Arquitetura MVC com JSF. Fonte – Faria, 2015

Uma das vantagens do JSF está no modelo de componentes de interface do usuário, o qual possibilita alta produtividade aos desenvolvedores, pois o framework proporciona a construção de interfaces para web utilizando um conjunto de componentes pré-construídos. Os principais componentes que a implementação de referência do JSF fornece são: formulário, campos de entrada de texto e senhas, rótulos, links, botões, mensagens, painéis, tabela de dados, entre outros.

As páginas criadas com os componentes JSF passarão por um ciclo de vida de processamento bem definido toda vez que forem executadas. As fases que compõe o ciclo de vida do JSF são:

- Fase 1: Restore View (Restauração da Visão)
- Fase 2: Apply Request Values (Aplicar Valores da Requisição)
- Fase 3: Process Validation (Processar as Validações)
- Fase 4: Update Model Values (Atualizar Valores de Modelo)
- Fase 5: Invoke Application (Invocar Aplicação)
- Fase 6: Render Response (Renderizar a Resposta)

As fases do ciclo de vida do JSF podem ser observadas na Figura 6.

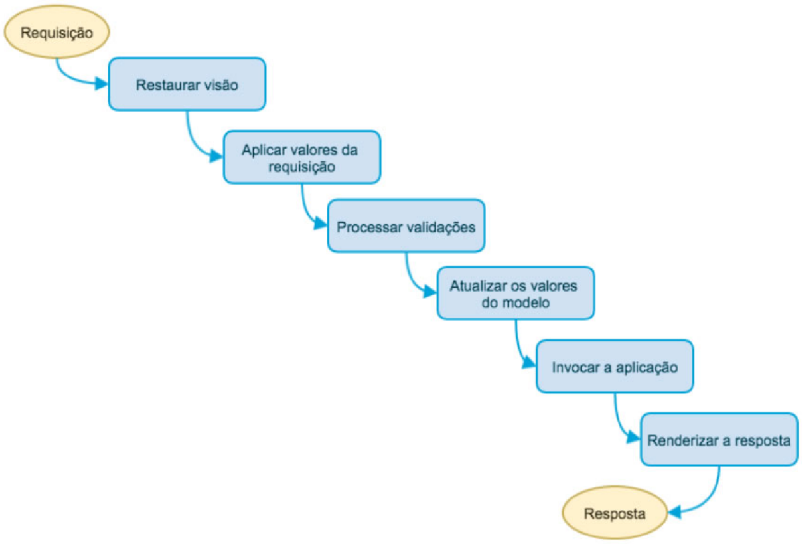

Figura 6 - Ciclo de vida do JSF. Fonte – Faria, 2015

Fase 1 - Restore View (Restauração da Visão): Nesta fase a visão que está sendo requisitada, proveniente de uma requisição do servlet FacesServlet, é identificada por meio do ID desta que é determinado pelo nome da página JSF. Após a página ter sido identificada, a mesma é salva no FacesContext(caso não tenha sido salva ainda) e sua respectiva árvore de componentes é construída.

Fase 2 - Apply Request Values (Aplicar Valores da Requisição): Nesta etapa, os componentes da página, criados ou recuperados, passam a ter o valor que foi inserido pelo usuário.

Fase 3 - Process Validation (Processar as Validações): Aqui ocorre a execução das validações definidas pelo servidor em cada componente da página. Não encontrando valores inválidos, o ciclo segue para a próxima fase, Fase 4. Caso exista algum erro durante a validação, uma mensagem de erro será gerada e o componente é marcado como inválido. Nesta situação, o ciclo segue para a Fase 6 onde será renderizada a resposta.

Fase 4 - Update Model Values (Atualizar Valores de Modelo): Após a validação dos valores na Fase 3, agora esses mesmos valores são atualizados em seus respectivos atributos localizados no *managed beans*.

Fase 5 - Invoke Application (Invocar Aplicação): Nesta fase os valores enviados pelo usuário estão validados e convertidos, estando assim prontos para serem utilizados pela aplicação a fim de atender as necessidades das regras de negócio.

Fase 6 - Render Response (Renderizar a Resposta): A última fase consiste na apresentação da página referente ao resultado da aplicação ao usuário após a execução dos processos que envolvem a aplicação.

2.4 WildFly 8

O WildFly é o servidor de aplicação Open Source da Red Hat baseado no Java EE 7. Ele é a sucessão do projeto JBoss Application Server. O Wildfly tem uma arquitetura modular e vem com ótimas ferramentas de gerenciamento e recursos para criação de clusters (alta disponibilidade).

2.5 PrimeFaces

O PrimeFaces é uma biblioteca OpenSource de componentes JSF, o qual é composto por diversos componentes de interface personalizados. Além dos componentes básicos do JSF, o PrimeFaces possui componentes interessantes tais como: tabelas de dados avançadas, menus suspensos, botões, barras de progressão, janelas de diálogos, layout e componentes para captura de datas e cores. Os componentes do framework tem suporte ao desenvolvimento responsivo para dispositivos móveis. A tecnologia foi criada e é mantida pela organização Prime Technology.

2.6 Hibernate

O Hibernate é uma ferramenta de mapeamento objeto-relacional, tradução de *Object Relational Mapper* (ORM), onde o principal objetivo é facilitar a comunicação entre as aplicações Java que seguem o modelo orientado a objeto e os SGBDs que utilizam o modelo relacional. O sucesso do Hibernate serviu de inspiração para a especificação Java Persistence API (JPA), a qual foi elaborada para padronizar as ferramentas ORM para aplicações Java no intuito de minimizar a complexidade do desenvolvimento. O Hibernate é responsável por abstrair o código SQL, toda a camada JDBC e o SQL será gerado em tempo de execução. Mais que isso, ele vai gerar o SQL que serve para um determinado banco de dados, já que cada banco tem um "dialeto" diferente dessa linguagem. Dessa forma é possível trabalhar

com diferentes SGBDs, basta trocar as configurações necessárias e escolher qual banco de dados deseja utilizar sem ter de alterar a codificação Java, já que isso fica de responsabilidade da ferramenta.

2.7 Oracle 11g

O Oracle armazena dados logicamente em tablespaces e fisicamente em arquivos de dados (datafiles). Apesar dos arquivos de dados e os tablespaces estarem muito "inter-relacionados", os mesmos possuem diferenças importantes:

- Um banco de dados Oracle consiste em uma ou mais unidades de armazenamento lógicas denominadas tablespaces, que armazenam coletivamente todos os dados do banco de dados.
- Cada tablespace em um banco de dados Oracle consiste em um ou mais arquivos denominados arquivos de dados (datafiles), que são estruturas físicas compatíveis com o sistema operacional no qual o Oracle é executado (Legatti, 2011).

O banco de dados Oracle utiliza várias estruturas de armazenamento físico no disco para conservar e gerenciar os dados de transações de usuário (Bryla; Loney, 2009).

2.8 Oracle Linux

Oracle Linux é um sistema operacional de código aberto disponível mediante a GNU General Public License (GPLv2). O sistema operacional Oracle Linux fornece inovações, com modernas ferramentas e recursos, que permitem inovar, colaborar e criar soluções em ambientes tradicionais, virtuais e baseados na nuvem (Oracle).

2.9 GIT

GIT é um dos diversos sistemas para controle de versão que tem a responsabilidade de gerenciar diferentes versões de um mesmo projeto. O GIT possui um design simples, gratuito, fácil instalação, configuração e baixa curva de aprendizado, e é possível obter históricos das modificações, permitir que vários

programadores trabalhem no mesmo projeto e permitir um comparativo entre várias versões do projeto (Ramos, 2015).

3. RESULTADOS

No Estado de Mato Grosso não há um sistema de software integrado entre todas as unidades prisionais da SEJUDH para que haja um controle dos dados de cada recuperando desde sua entrada até sua saída. Atualmente há apenas o sistema SIAPEN que faz o controle dos dados de gerenciamento dos presos, porém é um software que não está sendo utilizado por todas as unidades prisionais do estado. O Sistema de Gestão Penitenciária, SIGEPEN, está sendo desenvolvido para suprir a grande necessidade da secretaria que é manter o controle da gestão dos recuperandos do sistema prisional do estado. Dessa forma, os gestores terão o acompanhamento das etapas do processo de entrada do recuperando bem como suas visitas, dias trabalhados, dias estudados, incluindo registro disciplinar, movimentações internas e externas, contabilização da alimentação, entre outros módulos.

O SIGEPEN é um sistema que irá permitir o controle do sistema penitenciário de Mato Grosso, o registro detalhado e individualizado do prontuário do recuperando e dos servidores de plantões, com controle dos processos e rotinas internas e externas no que diz respeito à administração da unidade carcerária. O software será responsável pela integração dos diversos setores da unidade, tais como controle de entrada e saída de visitantes, módulos específicos para o controle do mapa de alimentação, enfermaria, transferência de reeducando, entre outros. Os *stackholders* do sistema são os servidores que compõem a equipe da Gerência de Inteligência da SEJUDH, a qual tem o conhecimento dos requisitos e processos que o sistema deverá realizar para que possa atender às expectativas dos usuários. A Gerência de Inteligência terá acesso aos mecanismos de configurações do SIGEPEN, tais como permissão de usuário, perfis de acessos, e, além disso, o sistema disponibilizará a visualização de diversos relatórios conforme a necessidade do usuário, fornecendo assim informações para análise e tomadas de decisões.

As tecnologias empregadas para o desenvolvimento de sistemas web foram selecionadas para serem usadas durante o estágio supervisionado com a intenção de desenvolver o sistema web SIGEPEN, o qual terá as seguintes características:

- Sistema Java Web utilizando a framework Java Server Faces
- Framework Front-End: Primefaces
- Utilização do framework de persistência Hibernate e JPA
- Utilização do banco de dados Oracle 11g
- Operações de cadastro, listagem, alteração e exclusão, autenticação, auditoria, entre outros.

Atualmente o projeto web está estruturado com cerca de 35 classes modelos, as quais fazem a representação de uma tabela no banco de dados. Na Figura 7 observamos as classes que compõem a camada de modelos.

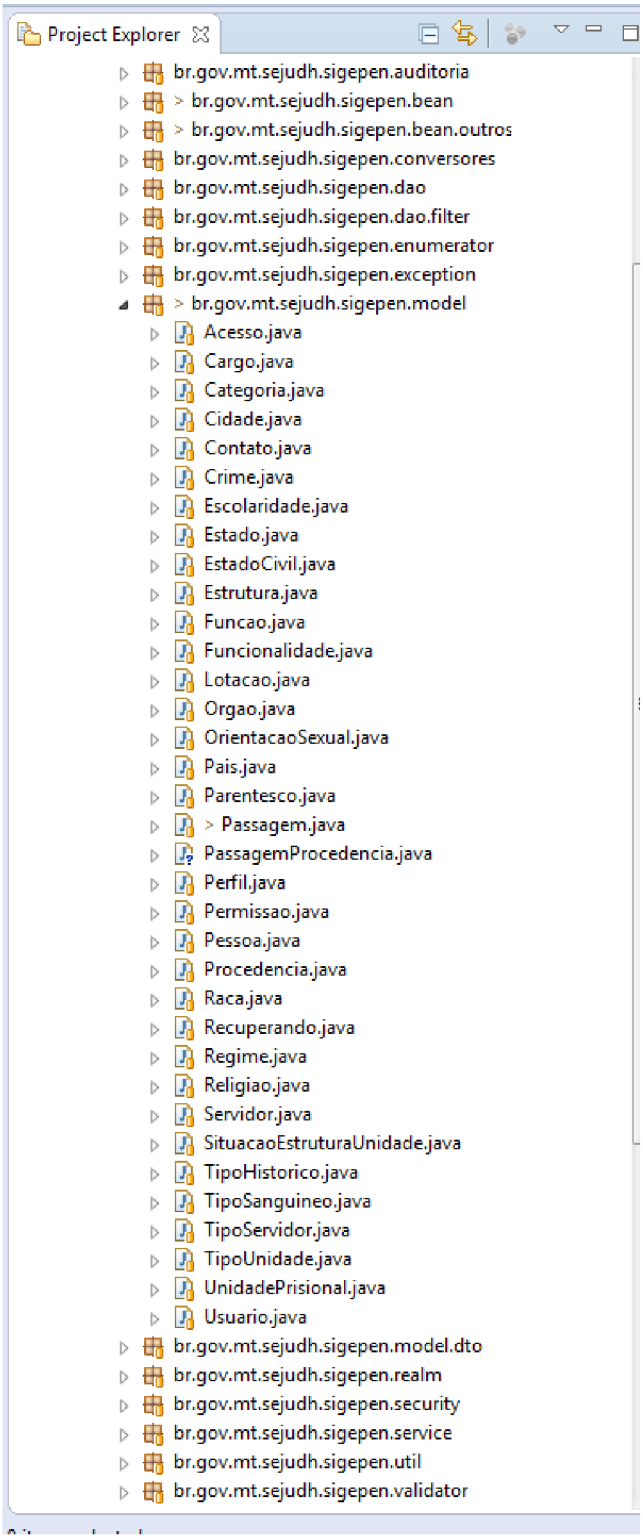

Figura 7 - Classes modelos

O projeto web está organizado no Eclipse IDE em diretórios os quais são denominados pacotes no desenvolvimento Java. Esta organização facilita a implementação do conceito MVC e entre outros padrões de projeto. Na Figura 8

pode-se observar a organização dos pacotes que armazenam as classes em Java e os diretórios que armazenam as páginas XHTML responsáveis por apresentar os dados ao usuário.

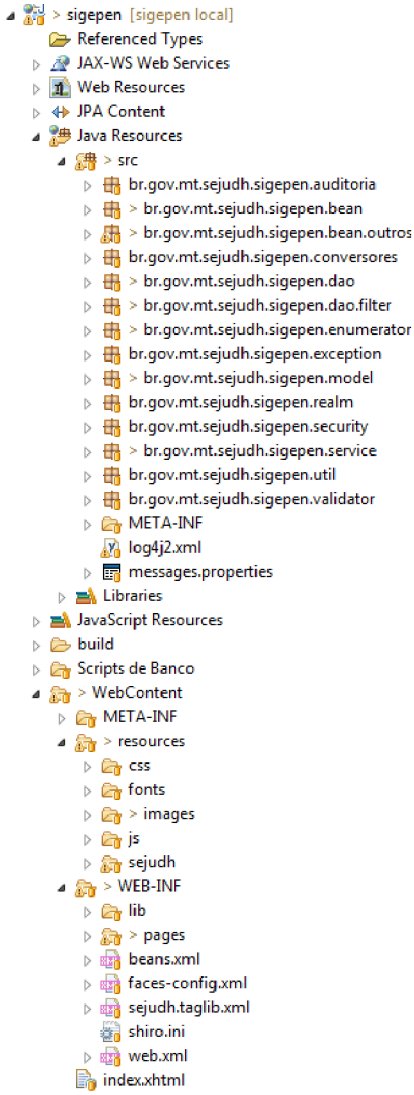

Figura 8 - Estrutura do projeto no Eclipse IDE

3.1 Classes Modelos

As classes que implementam o conceito de *Model* estão localizadas no pacote *br.gov.mt.sejudh.sigepen.model*. Estas classes são a representação das tabelas do banco de dados e seus respectivos atributos. As tabelas são criadas a partir do mapeamento feito no Java através das anotações codificadas nas classes modelos.

Para que o mapeamento objeto/relacional funcione é necessário informar à implementação do JPA, além de informações sobre como as classes devem se tornar persistentes, ou seja, como as instâncias dessas classes podem ser gravadas e consultadas no banco de dados. Para isso, devem-se anotar os *getters* ou os atributos, além das próprias classes. A seguir observar-se as cofigurações das anotações necessárias para que a tabela, da classe em questão, seja criada.

```
package br.gov.mt.sejudh.sigepen.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import org.hibernate.envers.Audited;
import br.gov.mt.sejudh.sigepen.enumerator.Status;

@Audited
@Entity
@Table(name = "CRIME")
public class Crime {

	@SequenceGenerator(name = "SEQ_CRIME_GENERATOR", sequenceName
	= "SEQ_CRIME", initialValue = 1, allocationSize = 1)
	@Id
	@GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
	"SEQ_CRIME_GENERATOR")
	@Column(name = "ID")
	private long id;

	@Column(name = "NOME", length = 80, nullable = false)
	private String nome;

	@Column(name = "DESCRICAO", length = 250, nullable = false)
	private String descricao;

	@Enumerated
	@Column(name = "STATUS")
	private Status status;

	public Crime() {
	}

	public Crime(Status status) {
		this.setStatus(status);
	}

	public long getId() {
		return id;
	}
```

```
	public void setId(long id) {
		this.id = id;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome.trim();
	}

	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		descricao = descricao.replaceAll("\n", " ");
		descricao = descricao.replaceAll("\r", "");
		descricao = descricao.replaceAll("\t", "");
		this.descricao = descricao.trim();
	}

	public Status getStatus() {
		return status;
	}

	public void setStatus(Status status) {
		this.status = status;
	}

	@Override
	public String toString() {
		return "Crime [id=" + id + ", nome=" + nome + "]";
	}
}
```

As anotações @Entity, @Id e @GeneratedValue são as configurações mais importantes para que o JPA possa identificar e gerar no banco de dados a tabela da classe. A anotação @Entity informa que a classe é uma entidade, e que representa uma tabela do banco de dados. As anotações @Id e @GeneratedValue são usadas para declarar o identificador da tabela no banco de dados, e se esse identificador deve ter um valor gerado no momento de inserção (auto-incremento). Outras anotações podem ser usadas para especificar as configurações da tabela, como o @Table e @Column. O @Table define detalhes da tabela no banco de dados, como por exemplo, o nome da tabela, o schema que ela pertence caso seja necessário. O @Column serve para especificar as configurações de um atributo da tabela, como o

nome do atributo, o tamanho, se ele aceita inserções nulas (restrição *not null*), entre outras configurações.

Para mapear um atributo que representa uma chave estrangeira, são utilizadas as anotações @ManyToOne, @OneToOne, @OneToMany e @ManyToMany para diferentes tipos de relacionamento. No SIGEPEN os tipos de relacionamentos mais utilizados são os @ManyToOne e @OneToOne. A anotação @ManyToOne é usado para referenciar o relacionamento de cardinalidade Muito-para-Um como pode ser visto na Figura 9, por exemplo, muitas Cidades em uma Unidade Prisional podem ser relacionadas, ou seja, várias instâncias de uma entidade pode estar relacionado a uma única instância da outra entidade. A anotação @OneToOne é semelhante à anotação @ManyToOne que acabamos de ver, a principal diferença é que na chave estrangeira terá a restrição de unicidade (*unique*).

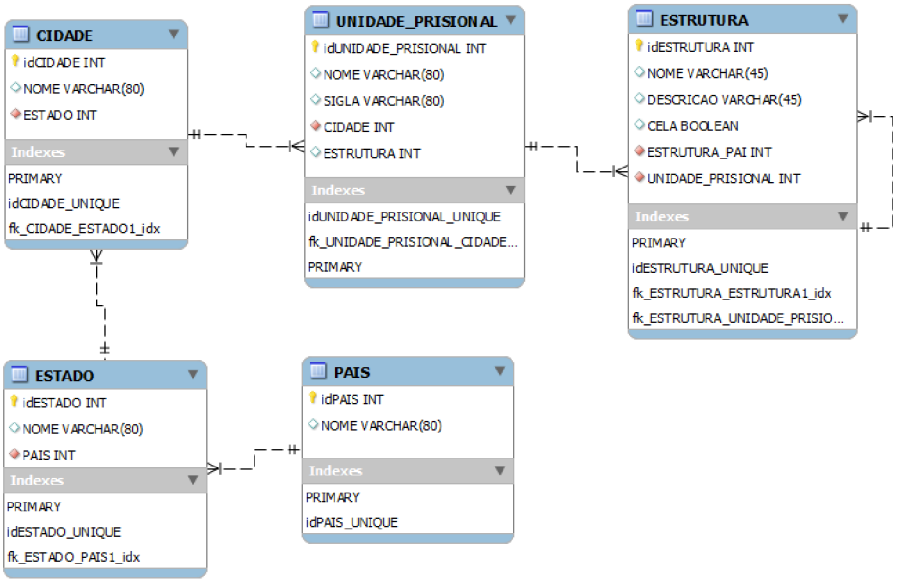

Figura 9 - Relacionamento entre entidades do SIGEPEN

3.2 Managed Bean

Os *Managed Beans* são Java Beans que servem como pontes entre o *back-end* da aplicação (regras de negócio, acesso ao banco de dados, entre outros) e a interface gráfica (a página XHTML). Os *beans* gerenciados do JSF podem receber dados digitados pelos usuários através de alguma página, processar métodos através de ações dos usuários e fornecer dados para apresentação na página.

Para um *bean* ser reconhecido como um *Managed Bean* JSF, precisamos registrá-lo. A maneira mais fácil de fazer isso é através da anotação @ManagedBean, do pacote javax.faces.bean. Por padrão, todas as classes do projeto serão escaneadas para encontrar *beans* anotados. Além disso, cada *Managed Bean* possui um tempo de vida, dentro do ciclo de vida do JSF, o qual é definido conforme o escopo utilizado no *Managed Bean*.

Os principais escopos de *Managed Bean* JSF são:

- @NoneScoped: será criada uma nova instância a cada vez que o *bean* for referenciado.
- @RequestScoped (padrão): inicia-se quando é referenciada em uma única requisição HTTP e é finalizada quando a resposta é enviada de volta ao usuário.
- @ViewScoped: a referência da instância mantém-se ativa até que o usuário navegue para uma outra página.
- @SessionScoped: armazena a instância durante diferentes requisições e até mesmo navegações entre diversas páginas, até que atinja o tempo limite da sessão ou que a sessão do usuário seja invalidada. Os objetos não são compartilhados entre os usuários, pois cada usuário possui sua sessão de navegação.
- @ApplicationScoped: este é um escopo que compartilha os objetos para todos os usuários do sistema, e mantém a instância durante todo o tempo de execução da aplicação.

No trecho de código abaixo se pode visualizar a implementação do *Managed Bean* "CrimeBean" o qual possuiu o escopo de @ViewScope. O método *init()* do *Managed Bean* CrimeBean será invocado toda vez que for criado uma nova instância desse bean, pois esse método possui a anotação @PostConstruct que indica que o método é chamado logo após a instância do *bean* ser criada.

```
@ManagedBean
@ViewScoped
@URLMappings(mappings = {
@URLMapping(id = "crime", pattern = "/crime", viewId = "/WEB-
INF/pages/crime/ListarCrime.xhtml")
,@URLMapping(id = "190tj8vbjigj248u", pattern = "/190tj8vbjigj248u", viewId =
"/WEB-INF/pages/crime/CadastrarCrime.xhtml")})
```

```
public class CrimeBean extends MetodosDePaginaBean {

	private FiltroCrime filtro;
	private Crime crime;
	private LazyDataModel<Crime> crimes;

	@Inject
	private CrimeService service;

	@Inject
	private FacesUtil facesUtil;

	@Inject
	private Logger log;

	public CrimeBean() {

	}

@PostConstruct
public void init() {
	try {
		this.setCrime((Crime) facesUtil.getFlashParameter("crime"));

		if(this.crime == null) {
		this.crime = new Crime(Status.ATIVO);
		}

		this.setIsVisualizacao((Boolean)
		facesUtil.getFlashParameter("isVisualizacao"));

		modificarTituloPaginaCadastro(Crime.class, this.crime);

		pesquisar();
	} catch (Exception e) {
		facesUtil.exibirMensagem(FacesMessage.SEVERITY_FATAL, "ERRO",
		e.getMessage());
		LoggerUtil.logErro(e, log);
		e.printStackTrace();
	}
}

public String cadastrar() {
	return "CadastrarCrime";
}

public String salvar() {
	String retorno = null;

	try {
		service.salvar(this.crime);
		facesUtil.exibirMensagemInfo("Crime " + this.crime.getNome() + "
		salvo com sucesso!");
		retorno = "ListarCrime";
		LoggerUtil.logInfo(Acao.SALVAR, this.crime, log);
	} catch (Exception e) {
```

```
                facesUtil.exibirMensagem(FacesMessage.SEVERITY_FATAL, "ERRO",
                "Erro ao salvar crime (" + this.crime.getNome() + "): " +
                e.getMessage());
                LoggerUtil.logErro(Acao.SALVAR, this.crime, e, log);
                e.printStackTrace();
        }

        return retorno;
}

public void excluir(Crime crime) {
        try {
                service.excluir(crime);
                facesUtil.exibirMensagemInfo("Crime " + crime.getNome() + "
                excluído com sucesso!");
                LoggerUtil.logInfo(Acao.EXCLUIR, crime, log);
        } catch (Exception e) {
                facesUtil.exibirMensagem(FacesMessage.SEVERITY_FATAL, "ERRO",
                "Erro ao excluir crime (" + crime.getNome() + "): " +
                e.getMessage());
                LoggerUtil.logErro(Acao.EXCLUIR, crime, e, log);
                e.printStackTrace();
        }
}
```

3.3 Classes DAO

Data Access Object (DAO) são as classes que proporcionam para a aplicação as operações de acesso os dados no SGBD, como leitura, inserção, alteração e exclusão. No SIGEPEN essas classes são criadas utilizando os padrões do JPA, o qual encapsula as operações como abrir uma conexão com o banco de dados, preparar uma transação, e até mesmo cancelar a execução de alguma operação, invocando assim o *roll-back* da transação.

Utilizando o JPA, os objetos ficam vinculados à interface *EntityManager* a qual é responsável por controlar o objeto dentro do seu ciclo de vida no JPA, mantendo-o atualizado com o banco de dados. A Figura 10 demonstra o ciclo de vida de um objeto na JPA e os possíveis estados que ele pode assumir, e quais métodos da interface *EntityManager* são utilizados para realizar tais transições.

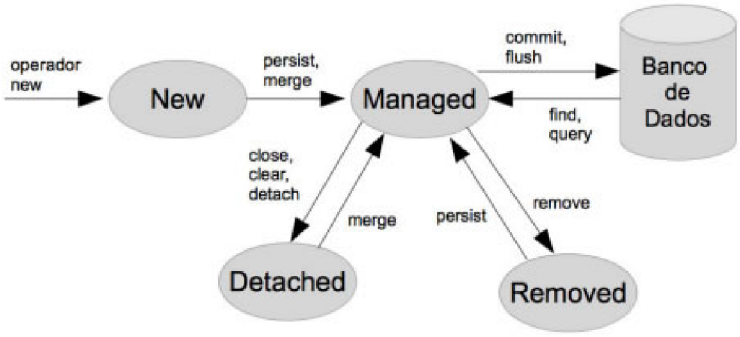

Figura 10 - Ciclo de vida de um objeto na JPA. Fonte – Faria, 2015

A classe CrimeDAO, que segue abaixo no trecho de código, está implementada utilizando os conceitos da JPA. Além da execução de simples SQLs, a JPA mantém a organização entre os objetos através do ciclo de vida deles.

```
@RequestScoped
public class CrimeDAO {

	@Inject
	private EntityManager entityManager;

	public CrimeDAO() {
	}

	public EntityManager getEntityManager() {
		return entityManager;
	}

	public void setEntityManager(EntityManager entityManager) {
		this.entityManager = entityManager;
	}

	public void salvar(Crime crime) {
		if (crime.getId() > 0) {
			crime = entityManager.merge(crime);
		} else {
			entityManager.persist(crime);
		}
	}

	public void excluir(Crime crime) {
		entityManager.remove(entityManager.merge(crime));
	}

	@SuppressWarnings("unchecked")
	public List<Crime> buscaPorFiltro(FiltroCrime filtro) {
		Criteria criteria = filtro.criarCriteriaParaBusca(Crime.class,
		entityManager);
```

```
            return criteria.list();
        }

        public int quantidadeFiltrados(FiltroCrime filtro) {
            Criteria criteria =
            filtro.criarCriteriaParaQuantidade(Crime.class,
            entityManager);
            criteria.setProjection(Projections.rowCount());
            return ((Number) criteria.uniqueResult()).intValue();
        }

    }
```

3.4 Hibernate

No arquivo persistence.xml é feito as configurações da JPA, tais como o nome da unidade de persistência, o provedor de persistência. Existem algumas propriedades que especificam determinadas característica para a criação e manipulação da base de dados.

As principais propriedades utilizadas no arquivo de configuração do projeto SIGEPEN são:

- *jta-data-source*: nome do datasource, qual está localizado no servidor de aplicação e armazena os dados de acesso (login e senha) para banco de dados.
- *hibernate.dialect*: dialeto a ser usado na construção de comandos SQL.
- *hibernate.show_sql*: informa se os comandos SQL devem ser exibidos na console (muito usado para debug, porém é recomendável desabilitar em ambiente de produção).
- *hibernate.format_sql*: indica se os comandos SQL exibidos na console devem ser formatados, assim facilitando a compreensão, porém pode gerar textos longos na saída.
- *hibernate.hbm2ddl.auto*: cria ou atualiza automaticamente a estrutura das tabelas no banco de dados.

O trecho de código a seguir demonstra as configurações realizadas no arquivo de configuração padrão do Hibernate (*persistence.xml*) no projeto web SIGEPEN.

```
<?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="sigepen" transaction-type="JTA">

	<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
		<jta-data-source>java:/sigepen-dev</jta-data-source>
		<properties>
			<property name="hibernate.dialect"
			value="org.hibernate.dialect.Oracle10gDialect" />
			<property name="hibernate.hbm2ddl.auto"
			value="update"/>
			<property name="hibernate.show_sql" value="true" />
			<property name="hibernate.format_sql" value="true" />
			<property name="hibernate.connection.charSet"
			value="UTF-8" />
			<property name="org.hibernate.envers.audit_table_prefix"
			value="AUD_" />
			<property name="org.hibernate.envers.audit_table_suffix"
			value="" />
			<property
			name="org.hibernate.envers.revision_type_field_name"
			value="ACAO"/>
		</properties>
	</persistence-unit>
</persistence>
```

Para maior segurança do projeto, os dados de acesso ao banco de dados ficam armazenados no arquivo de configuração do servidor de aplicação WildFly, denominado *standalone.xml*. Dessa forma facilita a alteração, caso necessário, desses dados de acessos.

Para acessar os recursos do M*anaged Bean* é usado a técnica de Expression Language (EL) que proporciona o acesso rápido a managed beans. O avaliador de expressões é responsável por tratar expressões EL que estão entre os caracteres #{ }. Abaixo segue o código fonte da tela de cadastro de Tipo de Crime, onde podemos observar a utilização dos métodos e atributos localizados no Managed Beans CrimeBean.

```
<ui:composition 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"
	xmlns:sgp="http://java.sun.com/jsf/composite/sejudh/composite"
	template="/WEB-INF/pages/_templates/principal.xhtml">

	<ui:define name="conteudo">
		<h3>#{crimeBean.tituloPagina}</h3>
		<h:form>
```

```
                <p:panel>
                    <sgp:formInputText
                        id="nome"
                        labelValue="Nome"
                        value="#{crimeBean.crime.nome}"
                        validationRequired="true"
                        disabled="#{crimeBean.isVisualizacao}"
                        maxlength="80"
                        display="text"/>

                    <sgp:formInputTextarea
                        id="descCrime"
                        labelValue="Descrição"
                        value="#{crimeBean.crime.descricao}"
                        validationRequired="true"
                        disabled="#{crimeBean.isVisualizacao}"
                        rows="10"
                        columns="50"
                        maxlength="250"
                        autoResize="true"
                    size="#{crimeBean.crime.descricao.length()+5}"
                        display="text"/>

                    <sgp:formSelectOneMenu
                        id="status"
                        labelValue="Status"
                        value="#{crimeBean.crime.status}"
                        itemsValue="#{comboHelperBean.status}"
                        itemValue="#{item}"
                        itemLabel="#{item.descricao}"
                        disabled="#{crimeBean.isVisualizacao}"/>
                    <br/>
                    <br/>
                    <p:commandButton value="Salvar"
                    action="#{crimeBean.salvar()}"    update="@form"
                    icon="ui-icon-disk"
                    rendered="#{(securityUtil.possuiPermissao('CadastrarCrime') and !crimeBean.isVisualizacao) or (securityUtil.possuiPermissao('AlterarCrime') and !crimeBean.isVisualizacao)}" />

                    <p:button outcome="ListarCrime" value="Voltar"
                    icon="ui-icon-arrowreturnthick-1-w"/>
                </p:panel>
            </h:form>
        </ui:define>
    </ui:composition>
```

4. DIFICULDADES ENCONTRADAS

4.1 PrimeFaces + Twitter Bootstrap

No início do projeto, janeiro de 2015, houve uma preocupação em criar o layout da aplicação com características do conceito responsivo, ou seja, layout que se adapta para diferentes tipos de dispositivos móveis. Como o PrimeFaces já havia sido definido pela equipe para ser a tecnologia de front-end, vimos que poderia arrumar uma forma de implementar o layout responsivo já que o PrimeFaces não possuía (na versão utilizada, 5.1) componentes nativos que fizesse esse mecanismo. Dessa forma, precisou ser pesquisada uma ferramenta que trabalhasse em conjunto com o PrimeFaces para atender a necessidade apesar dela não ser um requisito estabelecido para a criação do projeto.

Foi escolhido pela equipe de desenvolvimento a API Twitter Bootstrap para ser a ferramenta responsável para estruturar de forma responsiva o layout da aplicação SIGEPEN. Porém, na medida em que o software foi crescendo, a integração entre o PrimeFaces e o Twitter Bootstrap não proporcionou um layout 100% responsivo para a aplicação. Assim, foi necessário fazer ajustes em configurações do CSS do PrimeFaces e/ou do Bootstrap que sobrescrevesse configurações das implementações originais, para que em algumas telas do sistema fosse possível estruturar de forma que atendesse as expectativas da "responsividade".

Uma recente e esperançosa notícia foi a adição de recursos nativos na nova versão do PrimeFaces, 5.3, que implementam a "responsividade" de forma semelhante já utilizada pelo Bootstrap. Cabe agora à equipe de desenvolvedores realizarem testes para validar os novos recursos e tomar a decisão se convém ou não à migração de tecnologia.

4.2 Lazy e Eager Loading

Outra dificuldade encontrada durante o desenvolvimento foi quanto à performance do software. Ao desenvolver e executar o sistema foi notado uma lentidão no carregamento de entidades que possuía uma quantidade considerável de relacionamentos.

Como exemplo dessa lentidão existia a entidade Unidade Prisional a qual está relacionada com diversas outras entidades. Ao requisitar a listagem dos registros dessa entidade, ocorria que o objeto da classe era retornado com os dados de seus atributos e suas respectivas dependências, chaves estrangeiras.

O Hibernate possui duas configurações que são utilizadas para tratar o carregamento dos dados de atributos que fazem referência à outra entidade, que são o Lazy e Eager Loading. As configurações podem ser feitas através de anotações na classe modelo ou por meio do Hibernate Query Language (HQL). A configuração Eager Loading carrega os dados da entidade mesmo que não vá utilizá-los. Isso pode ser um problema para a perfomance da aplicação, pois isso fará com que haja lentidão na aplicação e problemas de memória. Já a configuração Lazy Loading é o oposto da Eager, a qual faz com que determinados objetos não sejam carregados do banco de dados até que ele seja solicitado pela aplicação, ou seja, são carregados sob demanda.

4.3 Composites Components JSF

Para resolver o problema de repetição dos códigos JSF nas páginas foi utilizado o recurso de Composites do JSF. Inicialmente as páginas foram criadas sem serem pensadas na reutilização das tags JSF, porém as páginas do sistema, principalmente as páginas de cadastro, estavam com uma quantidade grande de códigos repetidos responsáveis por estruturar o formulário de cadastro de uma determinada entidade.

Com o auxílio do recurso *Composite Component*, disponível na versão 2 do JSF, foi definido uma nova maneira de elaborar as páginas do sistema. Os composites foram criados para customizar os componentes de interface nativos do JSF, tais como o `<p:OutputLabel/>`, `<p:inputText/>` e entre outras tags que são utilizadas para que o usuário envie dados pelo sistema. Essa técnica dispensa o uso de classe

para a criação de componentes customizados. Nela os componentes são definidos em páginas XHTML que utilizam a biblioteca *composite* para a customização. Essa biblioteca possui diversas tags que definem interface, atributos da interface, implementações, e até mesmo tratamento de eventos e *listeners* através de composites components.

Com o uso da técnica de *Composite Components* a customização de interface ficou mais fácil, a partir da qual se tornou possível elaborar novos componentes sem a necessidade de utilizar a codificação Java. A customização pode gerar um ganho de produtividade do desenvolvimento com JSF. A medida em que os componentes começam a ser reutilizados, os desenvolvedores podem se preocupar mais com outras partes críticas da aplicação.

4.4 Combos aninhados de cidade e estado

Na aplicação os campos de `<selectOneMenu/>` foram transformados em componentes customizados para serem reutilizados de forma inteligente. Na tela de cadastro de Unidade Prisional encontra-se a implementação de dois campos que utilizam o componente de `<selectOneMenu/>` customizados para o formulário em questão. Para esta página os combos tiverem que trabalhar em conjunto onde inicialmente o combo de cidade é inicializado vazio e é preenchido conforme o estado que é selecionado pelo usuário no combo de estado, ou seja, ao selecionar um determinado estado, o combo de cidade é alimentado com as cidades correspondente ao estado que foi selecionado.

A dificuldade estava na chamada do evento após a seleção do estado para que pudesse executar o carregamento das devidas cidades. Para isso foi utilizado à técnica do Ajax, o qual invoca um método no M*anaged Bean* que faça a implementação necessária para carregar o combo de cidade. Dessa forma, o mecanismo funciona sem a necessidade de atualizar a página do navegador. Na Figura 11 é possível observar a implementação dos componentes de seleção de estado e cidade.

Figura 11 - Combo de Estado e Cidade

5. CONCLUSÕES

A utilização de ferramentas como o JSF, Hibernate e PrimeFaces deixa evidente a produtividade e flexibilidade quanto ao desenvolvimento da aplicação SIGEPEN. Tais tecnologias sendo implementadas de maneira correta podem proporcionar um sistema cada vez mais robusto e seguro a fim de atender da melhor forma as necessidades esperadas. Outro ponto importante é que as ferramentas utilizadas para o desenvolvimento do sistema web SIGEPEN possuem a licença *Open Source* podendo assim ser adquiridas sem nenhum custo pela GESB.

O primeiro marco de desenvolvimento do sistema é composto pelos cadastros básicos que irão servir como base para a aplicação como um todo. Durante os avanços do sistema teremos a preocupação de melhorar a perfomance da aplicação para que satisfaça as expectativas do usuário final. E ainda no primeiro marco do sistema temos o desenvolvimento do cadastro de Recuperandos, o qual possui diversas outras entidades relacionadas e regras de negócio para serem implementadas, tais como: lotação do recuperando, transferências internas e externas (entre unidades prisionais) do recuperando. Assim, é necessário uma atenção quanto ao mapeamento das dependências e seus respectivos relacionamentos para que não venha afetar de maneira negativa o software e as atividades administrativas das unidades prisionais.

Para trazer maior confiabilidade e segurança dos dados de cadastro, está em estudo o uso do cadastro biométrico para criar os registro dos recuperandos, visitantes e advogados. Assim, tanto o cadastro quanto a busca de registro serão mais exatos. Além disso, a busca semântica faz parte dos planos de implementação no projeto SIGEPEN. Tais mecanismos trarão grandes avanços para os usuários, proporcionando à Gerência de Inteligência de SEJUDH, informações precisas para análises e tomada de decisões.

Contudo, a faculdade de Sistemas de Informação da UFMT contribuiu na construção de toda a base de conhecimento necessária para o desenvolvimento deste trabalho. Assim, proporcionou-me condições para participar no desenvolvimento do projeto de sistema web, no qual pude aplicar a maioria dos conceitos adquiridos em sala de aula, tais como a Engenharia de Software, Orientação a Objeto, Análise e

Desenvolvimento Web, Banco de Dados, Projeto de Banco de Dados, Administração de Sistema Gerenciador de Banco de Dados.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ARAGÃO, Helder Guimarães; 2013. Java e Programação Orientada a Objetos: uma abordagem didática: Java. Salvador.

Balsamiq Mockup; 2015. Mockups Application Overview. Disponível em <http://support.balsamiq.com/customer/portal/articles/109151> Acesso em Agosto 2015

BOURQUE, Pierree; FAIRLEY, Richard E.; SWEBOK V3.0. 2014. Guide to the Software Engineering Body of Knowledge - Project of the IEEE Computer Society

CAELUM. Apostila Java para Desenvolvimento Web. Disponível em <http://www.caelum.com.br/apostila-java-web/> Acesso em Maio 2015

Caelum, Introdução ao JSF e PrimeFaces. Disponível em < http://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/introducao-ao-jsf-e-primefaces/> Acesso em Julho 2015.

Caelum; Uma introdução prática ao JPA com Hibernate. Disponível em <http://www.caelum.com.br/apostila-java-web/uma-introducao-pratica-ao-jpa-com-hibernate/#14-2-java-persistence-api-e-frameworks-orm>

ÇIVICI, Çağatay; 2015. Primefaces user guide - 5.1. 1ª Edição, Editora PrimeFaces

CORDEIRO, Gilliard; 2012. Aplicações Java para a web com JSF e JPA – Editora Casa do Código

DATE, C. J.; 2004. Introdução a sistemas de bancos de dados. 8ª Edição. Editora Elsevier.

GEHRKE, Johannes, RAMAKRISHNAN, Raghu; 2008. Sistemas de Gerenciamento de Banco de Dados . 3ª edição. Editora Eletrônica.

FARIA, Thiago; 2015. Java EE 7 com JSF, PrimeFaces e CDI. 2ª Edição. AlgaWorks Softwares, Treinamentos e Serviços Ltda

IC-UFG. Engenharia de Software. Disponível em <http://inf.ufg.br/engenharia-de-software> Acesso em: Abril 2015.

IBM; Introdução à Plataforma Eclipse IDE. Disponível em <http://www.ibm.com/developerworks/br/library/os-eclipse-platform/> Acesso em Agosto 2015

LEGATTI, Eduardo; Introdução ao conceito de Tablespaces. Disponível em <http://www.oracle.com/technetwork/pt/articles/database-performance/introducao-conceito-de-tablespaces-495850-ptb.html> Acesso em Agosto 2015

LONEY, Kevin; BRYLA, Bob; 2009. Oracle Database 11G: Manual do DBA. Editora Bookman

ORACLE; Sistema Operacionais Oracle Linux. Disponível em <http://www.oracle.com/br/technologies/linux/overview/index.html> Acesso em Outubro 2015.

PRESSMAN, Roger S.; 2011. Engenharia de Software - Uma Abordagem Profissional. 7ª Edição, Editora McGrawHill.

RAMOS, Allan; Controle de Versão, Git, Github e Bitbucket – Afinal, o que é tudo isso? Disponível em < https://aprendendowww.wordpress.com/2015/02/20/controle-de-versaogit-github-e-bitbucket-afinal-o-que-e-tudo-isso/> Acesso em Setembro 2015.

SAAB, Felipe Nisrallah; 2011. Hibernate Annotations – Mapeamento Objeto-Relacional através de Annotations. Revista Java Magazine, edição 90 - 2011

SCHACH, Stephen R.; 2009. Engenharia de Software - Os Paradigmas Clássico e Orientado a Objetos. 7ª edição. - Por 2009

TOKARSKI, Marcel; 2010. Anotações e Navegação no JSF 2.0. Revista Java Magazine, edição 86 - 2010

UNB. Engenharia de Software. Disponível em <http://www.unb.br/aluno_de_graduacao/cursos/engenharia_de_software> Acesso em: Abril de 2015.

WildFly; Servidor de Aplicação Java. Disponível em <http://soujava.org.br/2014/02/17/lancamento-mundial-do-wildfly-8/> Acesso em Setembro 2015.

7. Apêndices e/ou Anexos

Anexo I – Telas do sistema SIGEPEN

Figura 12 - Tela de login

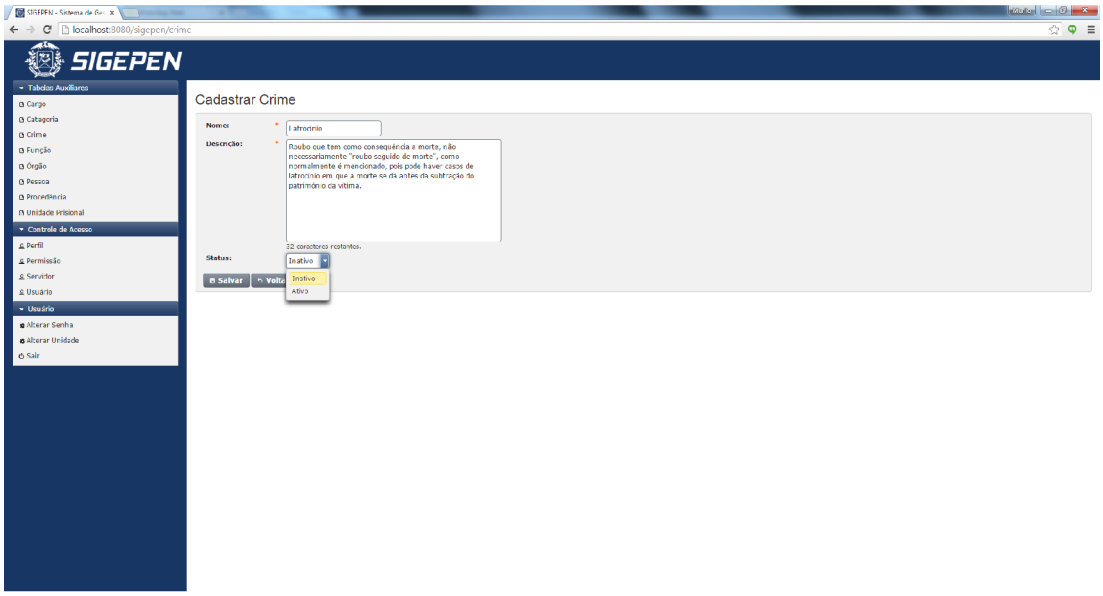

Figura 13 - Tela de cadastro e alteração do registro Tipo de Crime

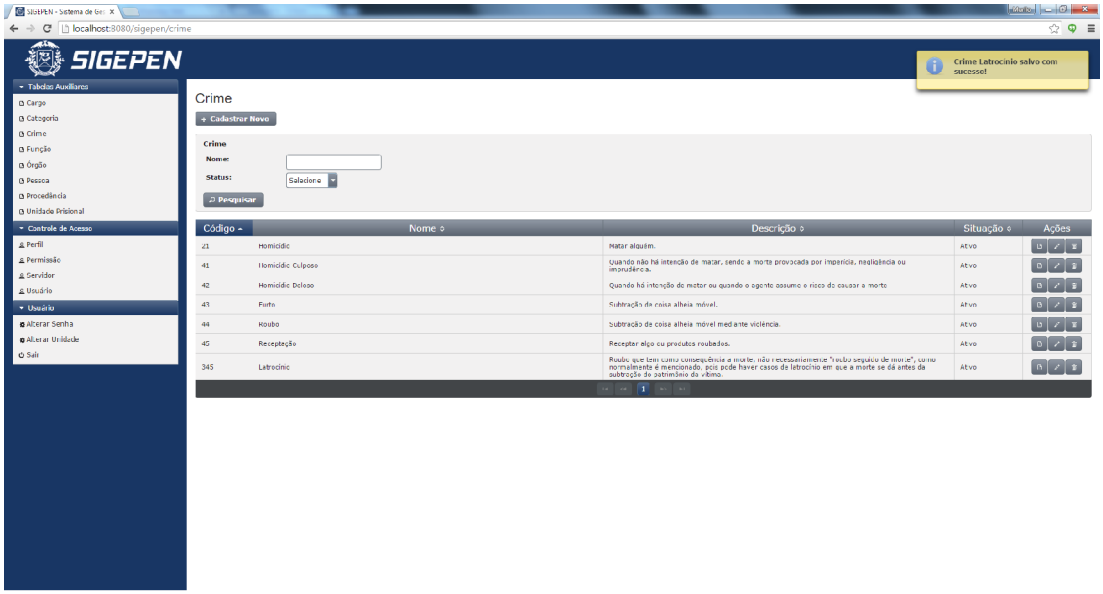

Figura 14 - Tela de listagem dos registros Tipo de Crime