UNIVERSIDADE PAULISTA

DANILO DE OLIVEIRA DOROTHEU

DIEGO DA SILVA SANTANA

MARCIO FERNANDES CRUZ

THIAGO GUY MOZOL VICENTE

UM SISTEMA DISTRIBUÍDO PARA CONTROLE DE OCORRÊNCIAS DE TRÂNSITO

SÃO PAULO

2015

B408FA-3 DANILO DE OLIVEIRA DOROTHEU

B56734-1 DIEGO DA SILVA SANTANA

B22816-4 MARCIO FERNANDES CRUZ

B303GD-9 THIAGO GUY MOZOL VICENTE

UM SISTEMA DISTRIBUÍDO PARA CONTROLE DE OCORRÊNCIAS DE TRÂNSITO

Dados obtidos através dos métodos e aplicações estudados em aulas e com pesquisas para fins de obtenção de nota para o curso de Ciência da Computação 7/8° Semestre (Noturno) da Universidade Paulista (UNIP) sendo entrega ao Professor/Dirceu.

Professor Dirceu.

Ciência da Computação 7° e 8º Semestre – Noturno

SÃO PAULO

2015

OBJETIVO DO TRABALHO

Este trabalho tem como objetivo elaborar as atividades Práticas Supervisionadas (APS), realizando uma pesquisa sobre os principais conceitos de sistemas distribuídos, através de pesquisas na internet e disciplina ministrada no semestre, sendo criado um projeto de software para desenvolver uma aplicação que possa ser acessa via *Web*.

Essa ferramenta tem como foco a colaboração de alimentação de uma base de dados central de problemas encontrados no trânsito. Estes problemas podem ser desde número grande de assaltos, iluminação ou sinalização ausente, ou mesmo, desrespeito de próprios condutores ou pedestres.

Com tudo, o trabalho tem a finalidade de apresentar o contexto sobre sistemas distribuídos e suas tecnologias e o desenvolvimento e as funcionalidades de um programa que possa ser acessado via web.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÂO

O desenvolvimento de sistemas distribuídos é a tendência mundial de desenvolvimento de novos sistemas. Salvo o caso da manutenção de sistemas legados, as equipes de desenvolvimento tendem a criar soluções baseadas em conceitos de computação distribuída.

Porém, o desenvolvimento deste tipo de sistema exige muitos fatores como escolher um padrão arquitetural em função da necessidade da aplicação e escolher uma gama de ferramentas disponíveis com o objetivo de se criar uma solução adequada.

Os conceitos gerais deste trabalho apresentam conceitos sobre sistemas distribuídos e adentra detalhes de especificações e *frameworks* utilizados na plataforma Java.

Os elementos arquiteturais esclarecem sem muitos detalhes os componentes arquiteturais escolhidos para o desenvolvimento deste trabalho acadêmico, que é um cadastro de ocorrências de trânsito colaborativos alimentados via um sistema que funciona em *browser* da Internet.

No capítulo do plano de desenvolvimento, é apresentado as imagens extraídas do sistema em “produção”, bem como é apresentado a aplicação *desktop* que consome o *webservice* criado neste trabalho.

Na conclusão é ressaltado a importância de se entender a complexidade, conceitos e as ferramentas necessárias para se desenvolver qualquer sistema distribuído, seja ela de qual padrão arquitetural.

2 CONCEITOS GERAIS

Sobre sistemas distribuídos, Sommerville (2011 apud TANEMBAUM 2007, p.333) diz “… uma coleção de computadores independentes que aparece para o usuário como um único sistema coerente”.

Qualquer projeto deste tipo de sistema deve se basear em conhecimento extraído em modelos consagrados, ou seja, baseado em normas geralmente aceitas. Para isso, a engenharia de software nos traz uma subárea chamada engenharia de software distribuído para apoiar o projeto e implementação destes sistemas.

2.1 Engenharia de software distribuídos

Segundo Sommerville (2011), praticamente todos os grandes sistemas hoje são distribuídos. Os preceitos da engenharia de software é totalmente aplicável no projeto de um sistema distribuído, mas, devido a algumas particularidades, esta área foi estendida, criando uma subárea de engenharia de software distribuídos.

A principal particularidade deste tipo de sistemas se reside que seus componentes de *hardware* e *software* estarem localizados em unidades computacionais distintas dentro de uma rede. Esta pode ser local ou operando a grandes distâncias. A isso, acrescentam-se outros fatores, como problemas de sincronização, latência ou perda de comunicação. Tudo isso deve ser considerado num projeto.

Outra peculiaridade de um sistema distribuído é a escalabilidade. Diferente de um sistema centralizado, onde todos os recursos de hardware e software funcionam em uma mesma unidade computacional, um sistema distribuído deve ser projetado prevendo a adição de novos recursos de hardware e software.

Por exemplo, pode ser que o componente do sistema distribuído que cuida do banco de dados não ser suficiente ou falhar muito. A adição, remoção ou troca de um novo componente para cuidar deste componente deve ser transparente ao usuário final.

2.2 Padrões arquiteturais

Ao se projetar um sistema distribuído, deve ser escolhido um padrão de projeto. Em tese, o padrão de projeto é algo imutável durante todo o ciclo de vida do *software,* pois todos os componentes irão se utilizar de recursos para troca de mensagens e chamadas de procedimentos remotos serão feitas embasadas sobre determinado padrão.

Existem vários fatores que podem escolher um padrão ou outro. Os principais requisitos é sua aceitabilidade no mercado e o fato de componentes de hardware e software de vários fabricantes, podem se integrar e funcionar integrados. Outro fator, também importante, como aponta Sommerville (2011) é um equilíbrio de fatores.

> […] os projetistas de sistemas distribuídos precisam organizar seus projetos de sistema para encontrar um equilíbrio entre desempenho, confiança, proteção e capacidade de gerenciamento do sistema. Não existe um modelo universal de organização de sistema distribuído que seja apropriado para todas as circunstâncias. […] (SOMMERVILLE 2011, p. 341).

O autor aponta os seguintes padrões arquiteturais: mestre escravo, cliente-servidor de duas camadas, cliente-servidor multicamadas, ponto a ponto e por último a arquitetura de componentes distribuída.

Sobre a arquitetura de componentes distribuídos, segundo Sommerville (2011), o primeiro padrão de projeto que tentou se firmar no mercado foi o Corba, desenvolvido ainda na década de 1990 e, pelo fato de não ter sido muito aceito pelos usuários, foi substituído por outros padrões baseados em serviços. Outras empresas como SUN e Microsoft também criaram seus padrões proprietários para sistemas distribuídos, com várias formas e tecnologias. Na parte prática deste trabalho, será usado alguns frameworks e discutidos algumas especificações da plataforma Java. O aplicativo desenvolvido é um sistema distribuído que funciona em *browser* e também disponibiliza serviço para ser utilizado por outra aplicação cliente.

2.3 Software como um serviço

É difícil imaginar os *softwares* hoje trabalhando como ilhas isoladas. Geralmente, acabam por integrar com outros sistemas. Um exemplo clássico disso é a consulta remota de CEP da base dos correios, a emissão de documentos fiscais e mesmo integração com sistemas contábeis. Mesmos novos sistemas desenvolvidos numa empresa, acabam por precisar se comunicar com sistemas legados da mesma organização.

É desafio dos projetistas saber lidar com essas tendências e buscar soluções para integrar seus sistemas. Dar acesso a uma base de dados nem sempre é o melhor caminho, um sistema externo pode danificar os dados existentes, por exemplo. Este sistema, ao invés disso, deve prover uma interface que possibilite a consulta e a alimentação de dados e, isso é chamado de serviço.

> De uma perspectiva de desenvolvimento de software, o processo de desenvolvimento de serviços tem muito em comum com outros tipos de desenvolvimento de software. No entanto, a construção de serviços normalmente não é conduzida pelos requisitos do usuário, mas por suposições do provedor de serviços sobre o que os usuários precisam. (SOMMERVILLE, 2011, p. 350).

Os grandes sistemas geralmente disponibilizam interfaces para acesso aos dados internos do sistema. Conforme citado acima, o Correios disponibiliza um serviço que consulta dados como endereços. Quanto a Receita Federal, esta disponibiliza outra gama de serviços para alimentação de notas fiscais, consulta sobre situações fiscais, etc.

Dado a importância neste tema, surgiu uma nova área dentro da engenharia de software, chamada engenharia de serviços.

> A engenharia de serviços é o processo de desenvolvimento de serviços para reúso em aplicações orientadas a serviços. Ela tem muito em comum com a engenharia de componentes. Os engenheiros de serviço precisam garantir que o serviço represente uma abstração reusável que poderia ser útil em diferentes sistemas. (SOMMERVILLE, 2011, p. 361).

Para sistemas existentes que funcionam de forma isoladas e principalmente sistemas que ainda não foram implementados, deve-se pensar em prover serviços. O projetista deve ter noção clara de conceitos como

heterogeneidade dos diversos componentes de hardware e software e também noção que os módulos de um sistema geralmente possuem alto acoplamento entre si, mas, quando se trata de conversa entre sistemas, a bandeira é o baixo acoplamento. Tudo deve ser abstraído o máximo possível para que o usuário, geralmente outro sistema, possa através de uma interface interagir com o primeiro sistema.

Os serviços em si são chamados pela literatura e pelo mercado de *web services* e, estes, geralmente usam padrões abertos para operação, como o XML e o *SOAP.*

2.4 WSDL

Segundo Sommerville (2011, p. 357) “A linguagem de definição de web service é um padrão para a definição de interface de serviço. Define como as operações de serviço (Nome de operação, parâmetros e seus tipos) e a associações de serviço devem ser definidas”.

2.5 REST

REST (do inglês *Representation State Transfer*) é um padrão arquitetural que se torna cada vez mais uma opção ao padrão SOAP. É uma tecnologia que não veio para substituir o SOAP, mas sim, para trazer a comunidade mais uma alternativa.

O REST é em sim mais leve que o SOAP. Este último, cria uma camada de abstração a comunicação. Enquanto que o *REST*, se utiliza dos protocolos existentes, no caso o HTTP. Tornando uma ótima alternativa a conexões lentas ou aparelhos móveis de baixo poder de processamento.

Construir serviços utilizando o protocolo *REST* traz muito mais eficiência a comunicação. Mas, em grandes projetos, ou quando for disponibilizar um serviço a um sistema externo, talvez convenha manter o padrão SOAP e todo seu formalismo de definição e envelopamento padrão de mercado.

2.6 Web Services

Segundo Josuttis (2008), o termo *Web Services* foi criado em 2000 pela Microsoft que definia padrões para computadores se comunicarem dentro da mesma rede.

> Hoje em dia, *Web Services* são um grande movimento baseado em diversos padrões e dirigido por muitas empresas e organizações de padronização […] Esses padrões de Web Services cobrem quase todas as áreas de computação distribuída e as chamadas de procedimentos remotos/funções/serviços, tais como segurança, transações, confiabilidade, modelagem de processos e gerenciamento de serviços. (JOSUTTIS, 2008, p. 182).

Segundo Hurwitz et. al. (2009), os padrões de *Web Services* são aceitos pela indústria, pois possibilitam o reúso de software. Um projetista de um sistema pode reusar as funcionalidades de outro sistema, sem a necessidade de criar a funcionalidade novamente. Ainda segundo o autor, esses serviços foram essenciais para a aceitação dos padrões atuais da Internet.

Os *Web Services* trabalham executando troca de mensagens. Uma aplicação cliente que deseja um serviço de algum outro componente, acessa este serviço, enviando uma mensagem. Em seguida, este serviço “responde” com outra mensagem.

O formato desta mensagem é também um padrão conforme será visto na próxima sessão.

2.7 XML: Um padrão aceito para representar uma mensagem

XML em inglês significa (*extensible markup language)* e é uma linguagem baseado em metadados onde criamos rótulos que representam informações e, dentro destes rótulos, estão os dados em si.

Segundo Pulier e Taylor (2008), a criação do XML é um marco na história da Informática pelo fato de criar um formato de mensagem, padronizada, no formato de arquivo-texto, que pode ser interpretada por humanos e máquinas. É um padrão de troca de mensagens.

> O XML é um padrão com um vasto número de aplicações potenciais. Um documento XML pode definir um *site web*, compartilhar dados entre dois bancos de dados, transportar um documento Word para dentro do Microsoft Power Point, comunicar um conjunto de instruções de programação de *software* e muito mais. (PULIER; TAYLOR, 2008, p. 20).

A mensagem em si precisa ser transportada e, para isso, deve-se usar um padrão de protocolo para troca de mensagens.

2.8 SOAP: Um conjunto de regras para descrever um XML

SOAP significa (*Simple Object Access Protocol)*. É um protocolo para descrever em estrutura XML os dados de uma mensagem de comunicação de *Web Services.*

> […] é o protocolo de mensagem que Web Services usam para conversar uns com os outros embora o uso seja restrito por Web Services. Foi inventado pela Microsoft para tornar mais fácil para softwares criados com ferramentas de desenvolvimento da Microsoft interagirem com outros softwares. Simples e flexível, SOAP pode ser usado por quaisquer dos programas que queiram trocar mensagens. (HURWITZ, 2008, p. 124)

O SOAP possui 4 componentes principais, sendo o envelope, o cabeçalho, o corpo e a falha e, este trabalho não tem como objetivo se adentrar aos detalhes sobre cada um deles.

2.9 Segurança com *XML* e *Web Services*

Como foi descrito em sessões anteriores, o arquivo XML tem uma linguagem simples e clara para representação de dados e pode ser facilmente interpretado por humanos e máquinas. Para um arquivo XML representado no formato do SOAP para funcionamento de Web Services, pode ser aplicado padrões de segurança nos dados. Josuttis (2008) confirma que aplicar segurança em XML e Web Services não significa apresentar novas tecnologias, mas sim, aplicar técnicas de segurança de outras áreas dentro do serviço.

Sem adentrar em detalhes, para o arquivo XML, o projetista pode aplicar o padrão de segurança XML *Dsig* para assinatura de alguns dados do arquivo.

Outro padrão é o XML Enc que permite que este arquivo seja criptografado de forma integral ou parcial.

Para a segurança de *Web Service*, existem os padrões *WS-Security, WS-SecurityPolicy, WS-Trust,* etc.

Foi tratado em sessões anteriores os conceitos de computação distribuída e padrões envolvendo *Web Services,* agora serão retomados as ideias envoltas na criação de um sistema distribuído usando uma série de elementos arquiteturais em função da linguagem Java.

3 ELEMENTOS ARQUITETURAIS DO PROJETO

Neste capítulo será apresentado breve resumo dos elementos arquiteturais utilizados na aplicação prática desenvolvido na plataforma Java com o objetivo de estudar parte dos conceitos de sistemas distribuídos.

3.1 Java Persistence API e Spring Data-JPA

Os vários *frameworks*[1] disponibilizados nativamente em uma plataforma ou mesmo desenvolvido e disponibilizados por terceiros visam facilitar a vida do projetista e da equipe de desenvolvimento, fazendo-os focar no que é principal, que é a implementação das regras de negócio.

Uma importante especificação é a JPA, do inglês *Java Persistence API*. Ela cuida e procura facilitar da conversa entre a aplicação e o banco de dados. Em si, esta especificação também se preocupa na mudança de paradigma, ou seja, da orientada a objetos para a relacional, que é do banco de dados.

JPA como foi dito, é apenas uma especificação. Uma importante implementação do JPA é o *Spring Data-JPA.* Este *framework* facilita acesso aos dados e a criação de CRUDs. Flexível, podemos definir várias configurações, como exemplo o *framework* de mapeamento objeto relacional preferido.

3.2 Hibernate

> O Hibernate é um framework ORM - Object Relational Mapping. É uma ferramenta que nos ajuda a persistir objetos Java em um banco de dados relacional. O trabalho do desenvolvedor é definir como os objetos são mapeados nas tabelas do banco e o Hibernate faz todo o acesso ao banco, gerando inclusive os comandos SQL necessários. (CAELUM, Site[2]).

Conforme apontado no material *online* da Caelum (2015), este *framework* é um projeto livre mantido pelo grupo Jboss e é líder neste tipo de aplicação (ORM) no mercado.

---

[1] *Framewok* de aplicações é definido por Sommerville (2011) como uma estrutura genérica estendida para se criar uma aplicação ou sub-sistema mais específico.

[2] Disponível em: http://www.caelum.com.br/apostila-vraptor-hibernate/persistindo-os-dados-com-ohibernate/#4-2-sobre-o-hibernate. Acesso em 25 out. 2015.

3.3 PostGreSql

Todos os usuários de sistemas gerenciadores de banco de dados buscam em comum alguns fatores, como segurança, atomicidade nas transações e rapidez de acesso. Existem no mercado opções pagas e muitas outras livre. Foi escolhido o PostGreSql, uma opção de banco de dados relacional gratuita e que possui muitas fontes de consulta na Internet.

3.4 Controlando a versão com o GIT

Controlar a versão dos códigos fontes é algo importante, não importa o tamanho do projeto e da equipe de desenvolvimento. Mesmo um sistema desenvolvido apenas por uma pessoa, vale a pena ter um controle de versão.

A maior diferença entre o GIT e outros controles de versão é que ele trata *snapshots* e não diferenças entre fontes.

> [...] o Git considera que os dados são como um conjunto de snapshots (captura de algo em um determinado instante, como em uma foto) de um mini-sistema de arquivos. Cada vez que você salva ou consolida (commit) o estado do seu projeto no Git, é como se ele tirasse uma foto de todos os seus arquivos naquele momento e armazenasse uma referência para essa captura. Para ser eficiente, se nenhum arquivo foi alterado, a informação não é armazenada novamente - apenas um link para o arquivo idêntico anterior que já foi armazenado. A figura 1-5 mostra melhor como o Git lida com seus dados. (GIT-SCM, Site[3]).

O GIT veio depois de outros sistemas de controle de versões como o *Subversion* e, é bem aceito pela comunidade Java. Ele reduz o uso do servidor, mantendo quase todas as operações locais. Visualizar histórico de alterações de todo o projeto são operações que, não necessariamente, precisam ser processadas a todo o momento no servidor.

Outra coisa importante que vale a pena citar é que no GIT é a integridade dos dados.

---

[3] Disponível em: https://git-scm.com/book/pt-br/v1/Primeiros-passos-No%C3%A7%C3%B5es-B%C3%A1sicas-de-Git. Acesso em: 25 out. 2015.

> Tudo no Git tem seu checksum (valor para verificação de integridade) calculado antes que seja armazenado e então passa a ser referenciado pelo checksum. Isso significa que é impossível mudar o conteúdo de qualquer arquivo ou diretório sem que o Git tenha conhecimento. Essa funcionalidade é parte fundamental do Git e é integral à sua filosofia. Você não pode perder informação em trânsito ou ter arquivos corrompidos sem que o Git seja capaz de detectar. (GIT-SCM, Site[4]).

Por fim, usamos esta ferramenta de controle de versão junto com o site http://xp-dev.com, para controlar os códigos fontes do projeto prático deste trabalho.

3.5 EhCache

É preferível que a aplicação utilize o menos possível o acesso ao SGBD que tem um custo alto de tempo de processamento comparado aos outros componentes do sistema. Pensando nisso, a comunidade desenvolveu soluções em cache, ou seja, que apresentam os dados persistidos a outros componentes, buscando-os em uma camada intermediária de cache e não no banco diretamente.

Uma implementação deste conceito é o banco EhCache. Totalmente adaptável e configurada através do JPA na configuração da aplicação prática deste trabalho.

3.6 Java Transaction API

Toda operação de persistência no banco de dados deve ser atômica, a fim de manter a integridade dos dados armazenados. Porém, quando um sistema distribuído possuiu vários componentes e várias formas de compor as entidades para orquestrar a organização, fica difícil manter um padrão aceito pela comunidade. Um sistema isolado pode fazer seu controle de transação diretamente através dos drivers do JDBC, por exemplo, mas quando existem necessidades maiores como manter a consistência entre dois sistemas diferentes, por exemplo, o projetista deve recorrer a literatura e procurar a

[4] Disponível em: https://git-scm.com/book/pt-br/v1/Primeiros-passos-No%C3%A7%C3%B5es-B%C3%A1sicas-de-Git. Acesso em: 25 de out. 2015.

melhor implementação desta.

Assim, surgiu o JTA. É uma especificação que visa padronizar o uso de transações em sistemas distribuídos. Vale citar neste texto duas importantes implementações, que é a *Web Logic* e a *JOTM.*

3.7 CDI – Injeção de Dependência

Um grande desafio para um desenvolvedor de software é garantir que seus objetos quando solicitados, estejam criados durante o ciclo de vida da aplicação. Grande parte dos erros acontece durante a operação e não são pegos durante o tempo de desenvolvimento. Por exemplo, em dado momento, um usuário pode fazer uma operação no sistema que, necessita de um objeto que foi destruído pelo coletor no servidor. Pensando nisso, a Sun a partir da especificação JSR-299, criou o CDI.

> A Injeção de Dependência e Contextos (CDI), especificada por JSR-299, é parte integrante do Java EE 6 e fornece uma arquitetura que permite aos componentes do Java EE, como os servlets, enterprise beans e JavaBeans, existirem dentro do ciclo de vida de uma aplicação com escopos bem definidos. Além disso, os serviços CDI permitem que os componentes do Java EE, como beans de sessão EJB e beans gerenciados do JavaServer Faces (JSF), sejam injetados e interajam de maneira acoplada flexível, disparando e observando eventos. (NETBEANS 2015, Site[5]).

3.8 Java Server Faces

Uma das grandes dificuldades no desenvolvimento *web* é trazer as funcionalidades normalmente encontradas em sistemas de janelas. Estes usam componentes ricos, baseados em eventos e, torna a experiência do usuário rica e atrativa.

Era difícil até então para os desenvolvedores criarem as mesmas facilidades para o ambiente *web*, validações de fácil implementação, soluções de internacionalização e, por fim componentes ricos de excelente acoplamento aos componentes do negócio. Isto é o que a especificação JSP propõe e uma excelente opção de implementação no mercado, será visto na próxima sessão.

[5] Disponível em: https://netbeans.org/kb/docs/javaee/cdi-intro_pt_BR.html. Acesso em: 25 de out. 2015.

3.9 PrimeFaces

Baseado em JSF, é um *framework* que oferece uma quantidade enorme de componentes ricos para ser adaptados nas aplicações web desenvolvidos em Java. Existem outras bibliotecas como o *RichFaces* e o *IceFaces*, porém, não foram estudada a sua aplicação na parte deste trabalho prática.

Além de abstrair tecnologia assíncrona Ajax para o programador, dispõe de *templates* e componentes também voltados para as plataformas móveis.

Por fim, vale dizer que é um projeto *opensource,* totalmente atento às novidades e especificações criadas pelo W3C (World Wide Web Consortium).

3.10 Validando conteúdo com expressões regulares

A expressão regular é uma forma de tratar e validar textos padronizados.

> É uma composição de símbolos, caracteres com funções especiais, chamados "metacaracteres" que, agrupados entre si e com caracteres literais, formam uma seqüência, uma expressão. Essa expressão é testada em textos e retorna sucesso caso esse texto obedeça exatamente a todas as suas condições. Diz−se que o texto "casou" com a expressão. (JARGAS, 2011, p. 2. Site[6]).

É suportada pelo Java a partir da versão 1.4 e está disponível no pacote java.util.regex.

Expressões regulares enfim, é implementada pelo pacote Regex e, possui extensa literatura. Geralmente, é utilizada para validação de campos do usuário, mas pode ter muito mais funcionalidades como fazer pesquisas avançadas em banco de dados e mesmo em estrutura de diretórios.

3.11 Jodatime

Trabalhar em aplicações com cálculo de datas e horas é uma tarefa difícil em qualquer linguagem. Normalmente quando precisamos de tal recurso, é algo a parte de nossas regras de negócio e, testar cálculos de data e hora, pode ser algo custoso e propenso a erros.

---

[6] Disponível em: http://aurelio.net/regex/apostila-conhecendo-regex.pdf. Acesso em: 25 de out. 2015.

O Java em sua API nativa oferece o pacote *Calendar*. Mas, quando precisa aprimorar alguns cálculos como calcular diferença entre datas, desprezando sábados e domingos, por exemplo, pode ser uma tarefa não tão intuitiva. Um desenvolvedor consegue desenvolver classes para isso, mas, para que inventar a roda, a SUN, através da especificação JSR 310, criou a Joda Time a partir da Java 8.

> Vale lembrar que JodaTime possui integração com Hibernate e pode ser facilmente configurada no seu projeto. Também é fundamental pensar nos casos particulares quando escrever seus testes de unidade. É muito fácil você cair em situações estranhas, em especial se sua data possui horas e minutos, além de anos bissextos, feriados que caem em fins de semana, etc. (CAELUM, Site[7]).

3. 12 Processo de build com Maven

Gerenciar um processo de *build* de uma aplicação pode ser muito complicado. Geralmente um sistema, não importa o tamanho, depende de bibliotecas de terceiros, etc. Para isso surgiram as ferramentas de *build* de uma aplicação.

A Apache criou no passado a ferramenta Ant (http://ant.apache.org/), citada neste trabalho, pois ainda é muito utilizada pela comunidade. Basicamente a configuração é feita tudo via arquivo *XML*.

O Maven, também criado pelo Apache, veio depois do Ant e, possui várias funções acopladas, tornando mais eficiente que seu antecessor.

> Diferentemente do ant, a idéia do maven é que você não precise usar tarefas e criar targets. Já existem um monte de tarefas pré-definidas, chamadas goals, que são agrupadas por plugins. Por exemplo, o plugin compiler possui goals como compile, que compila o código fonte, e o goal testCompile, que compila nossos unit tests.
> (CAELUM, Site[8]).

O Maven, também possui um repositório central, acessado pelo endereço http://www.mvnrepository.com/ onde a comunidade pode consultar toda uma gama de softwares existentes.

---

[7] Disponível em: http://blog.caelum.com.br/o-eterno-problema-de-calcular-a-diferenca-de-dias-entre-duas-datas-em-java/. Acesso em 25 de out. 2015.

[8] Disponível em: http://www.caelum.com.br/apostila-java-testes-xml-design-patterns/apendice-o-processo-de-build-ant-e-maven/#10-4-o-maven. Acesso em: 27 de out. 2015.

4 DESENVOLVIMENTO DA APLICAÇÃO

4.1 Motivação

Hoje o número de carros que trafegam em vias em grandes cidades está cada vez maior. Aumentando a quantidade, aumenta-se a probabilidade de ocorrer inflações, assaltos em determinadas regiões, etc.

Nem sempre as autoridades de trânsito estão presentes em todos os pontos. Sendo assim, este trabalho propõe uma ferramenta colaborativa para que os usuários possam apontar problemas.

Estes apontamentos de ocorrências serão feitas através de uma aplicação *web*, acessada via *browser* e será alimentada numa base de dados central.

Criamos também um *webservice* para que se consulte algumas estatísticas como exemplo total de ocorrências por ano, por exemplo.

4.2 Um sistema web responsivo

Conforme citada em capítulo anterior, foi utilizado a biblioteca *PrimeFaces* com o objetivo de criar um sistema que pode ser acessado por diferentes dispositivos, se adaptando a cada tela.

Obviamente um computador é mais rápido alimentar informações no sistema do que um celular pela facilidade de digitação. Mas, para o aspecto do sistema aqui apresentado, este fator é irrelevante. O importante é testar e aplicar conceitos de sistemas distribuídos.

Foi desenvolvido uma tela inicial onde o usuário tem acesso a alguns menus de cadastros. Foi inserido um menu de endereços, onde o usuário poderá alimentar país, unidade federativa, cidade e bairro. Com o objetivo de testar relacionamentos de pai-filho usando o *JPA* e o *Hibernate*, ambos citados no capítulo anterior.

4.3 Cadastros do menu endereço

As tabelas de país, unidade federativa, cidade, bairro e logradouro são tabelas auxiliares do sistema. Em tese são informações públicas e poderíamos utilizar a base dos Correios dentro da aplicação. Porém, com o objetivo de fixar conceitos estudados na sessão anterior como conceitos gerais e frameworks, criou-se uma estrutura de CRUD (*Create, Retrieve, Update e Delete)* em separado.

Todas as telas foram criadas utilizando o XHTML (*extensible hypertext markup language)* com o JPA através da biblioteca *PrimeFaces* na camada de apresentação.

Figura 1: Cadastro de País

Figura 2: Cadastro de unidade federativa

Figura 3: Cadastro de cidade

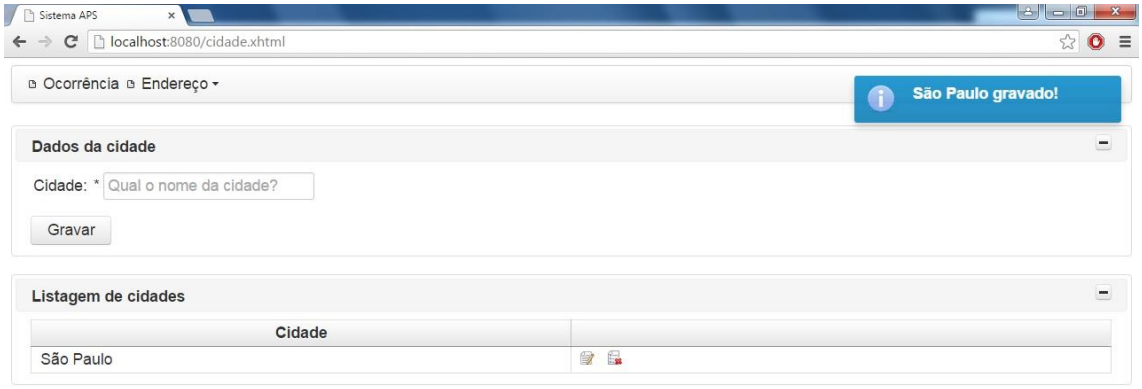


Figura 4: Cadastro de bairro

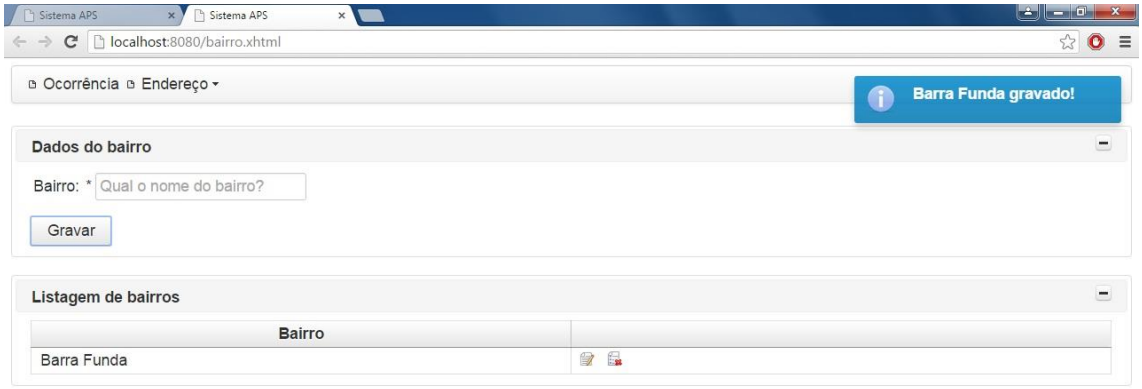
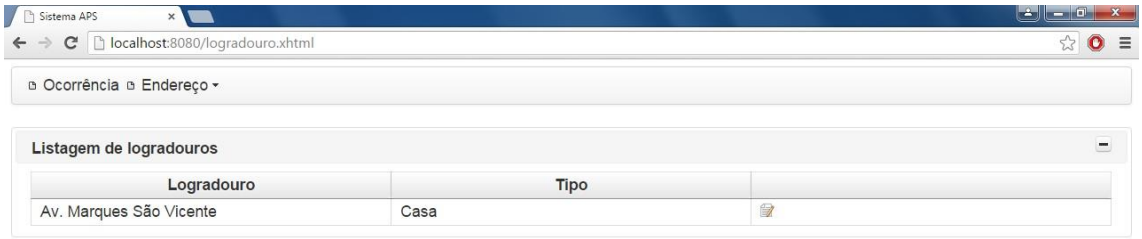


Figura 5: Cadastro de logradouro



4.4 Cadastros de ocorrências

Em termos técnicos, é apenas mais uma tabela que possui colunas que ligam diretamente as tabelas auxiliares do endereço, mas foi uma oportunidade para testar a funcionalidade do cadastro *wizard,* onde é possível fazer um cadastro gradativo, alimentando informações e clicando no botão “próximo”, onde a última fase é o botão “enviar” que realmente faz com que o controle proceda com o processo de persistência através do *JPA* e *Hibernate*.

Figura 6: Detalhes da ocorrência

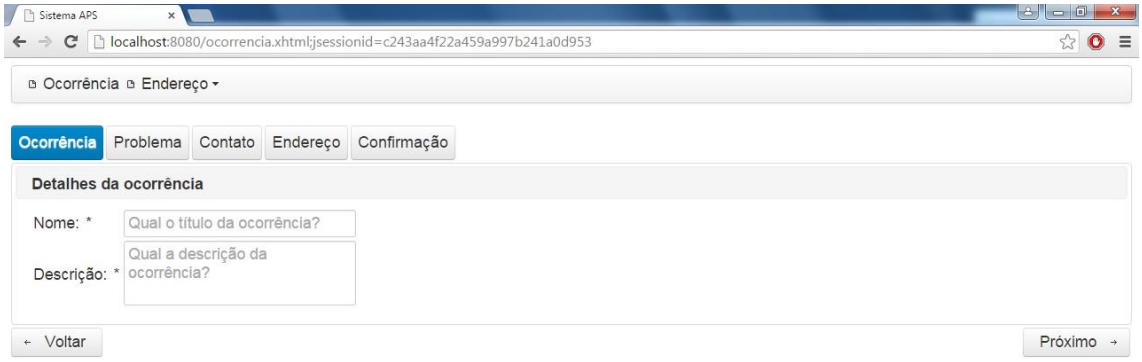


Figura 7: Detalhes do problema

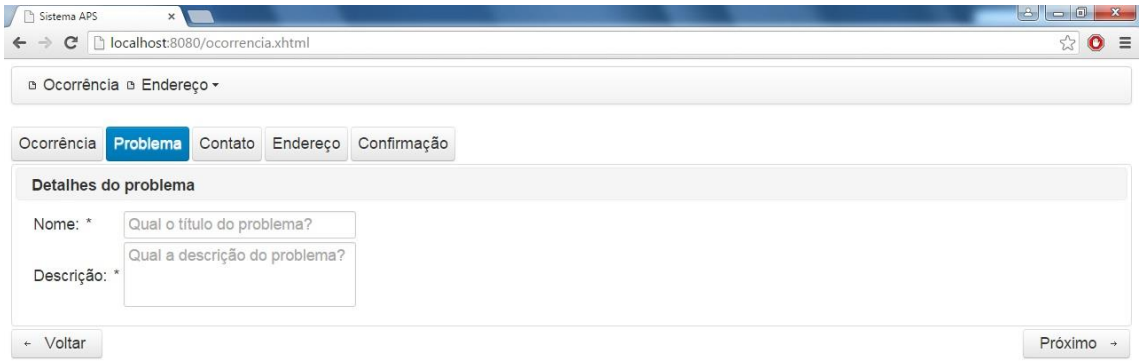


Figura 8: Detalhes do contato

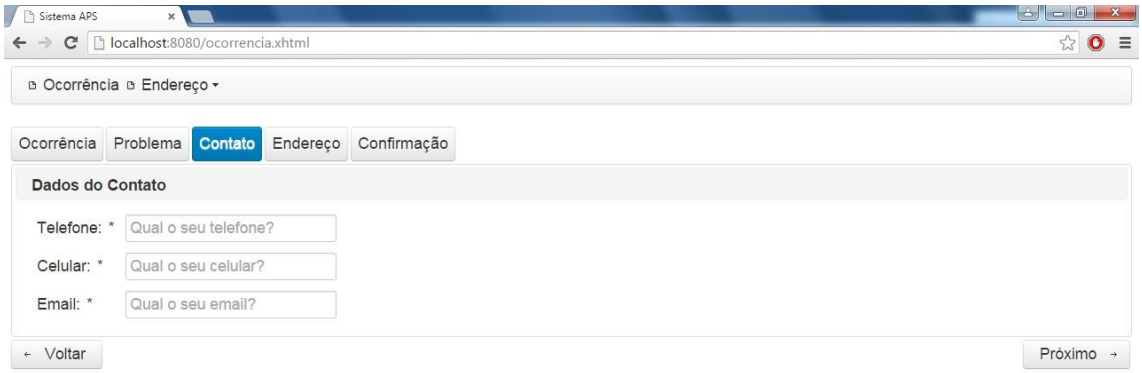


Figura 9: Seleção do endereço da ocorrência

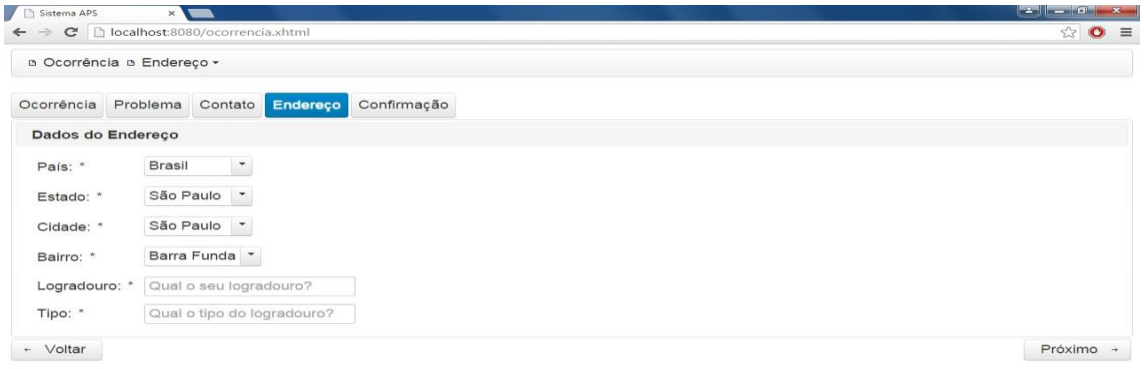
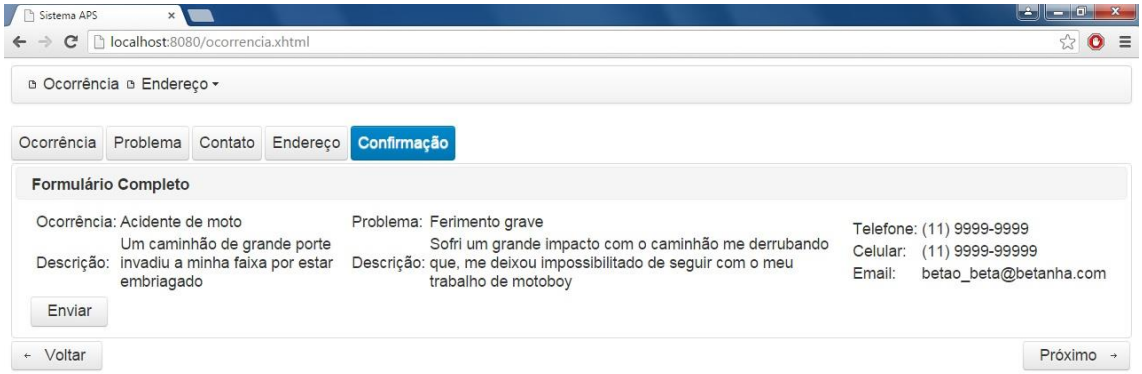


Figura 10: Tela de confirmação e envio de informações ao servidor



4.5 Consulta ao *webservice* de ocorrências

Com o objetivo de estudar e aplicar conceitos sobre *webservices* foi criado um serviço que tem como objetivo consultar ocorrências cadastradas. Para isso, foi criado uma outra aplicação, usando uma aplicação *desktop* utilizando componentes *swing* e *awt* mas, acessando dados do servidor remoto, onde está a aplicação desenvolvida.

Figura 11: Tela inicial da aplicação que consome o Webservice

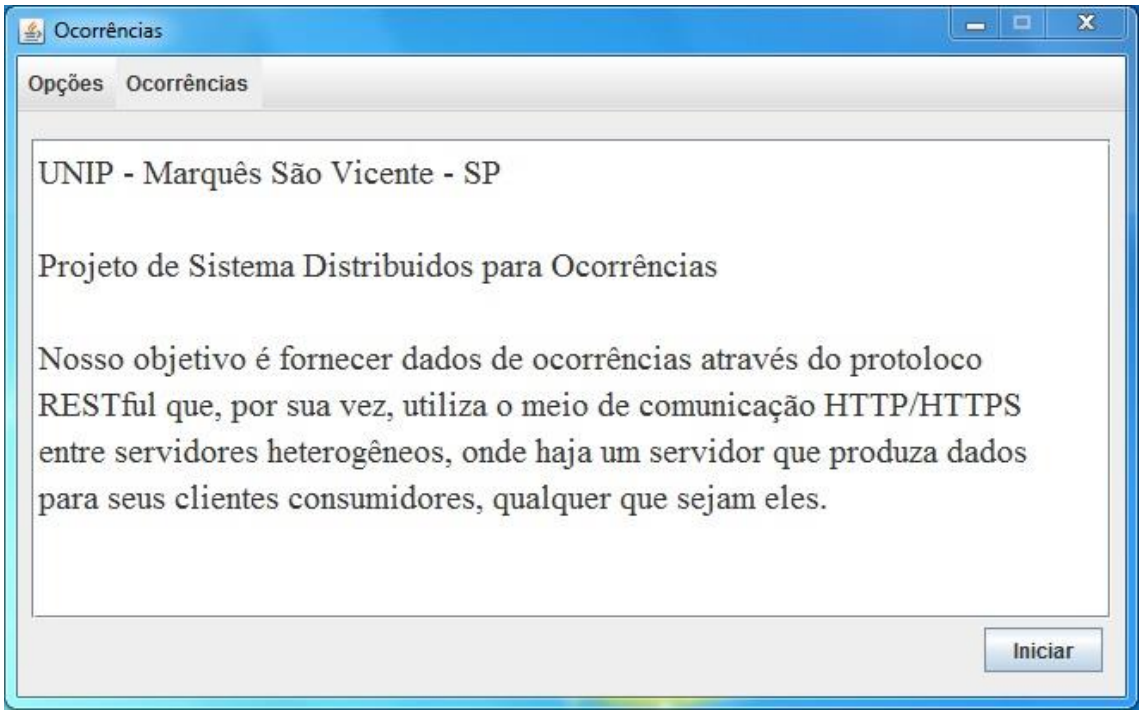
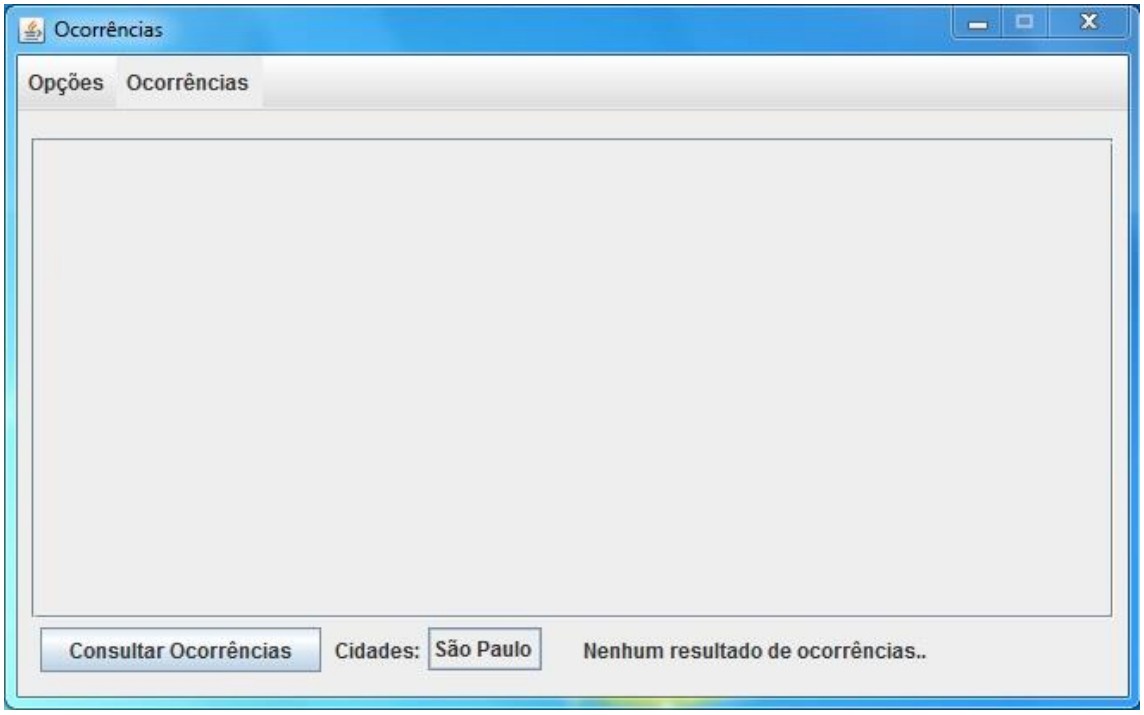
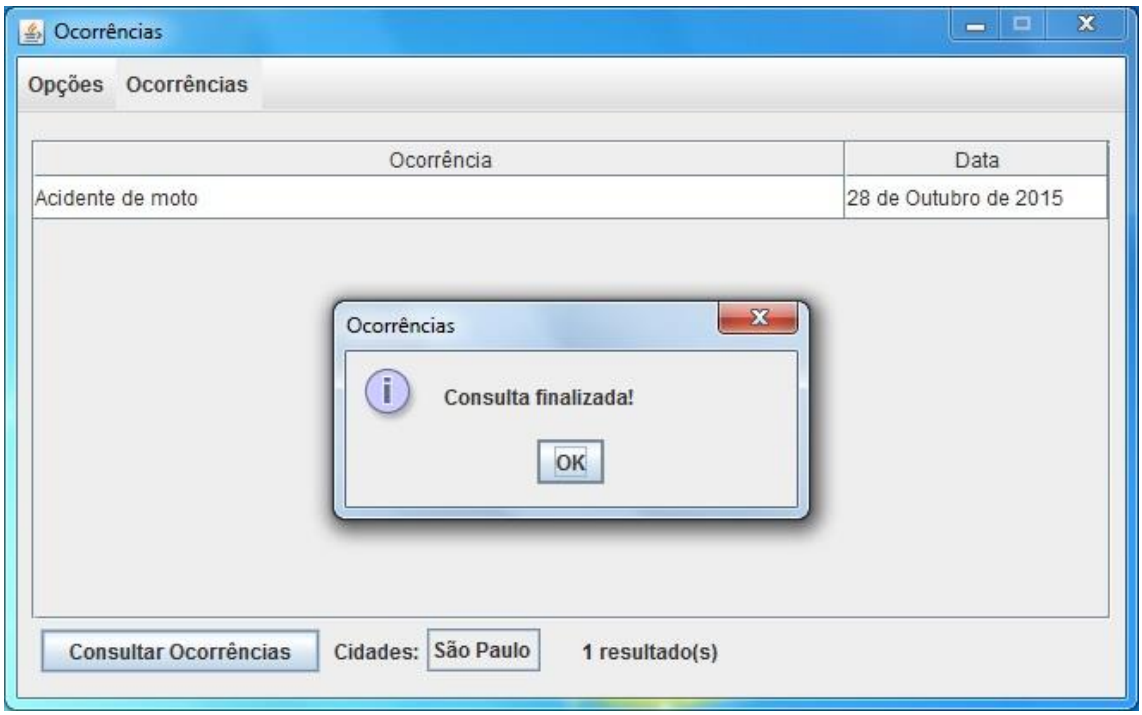


Figura 12: Tela de consulta de ocorrências por cidade



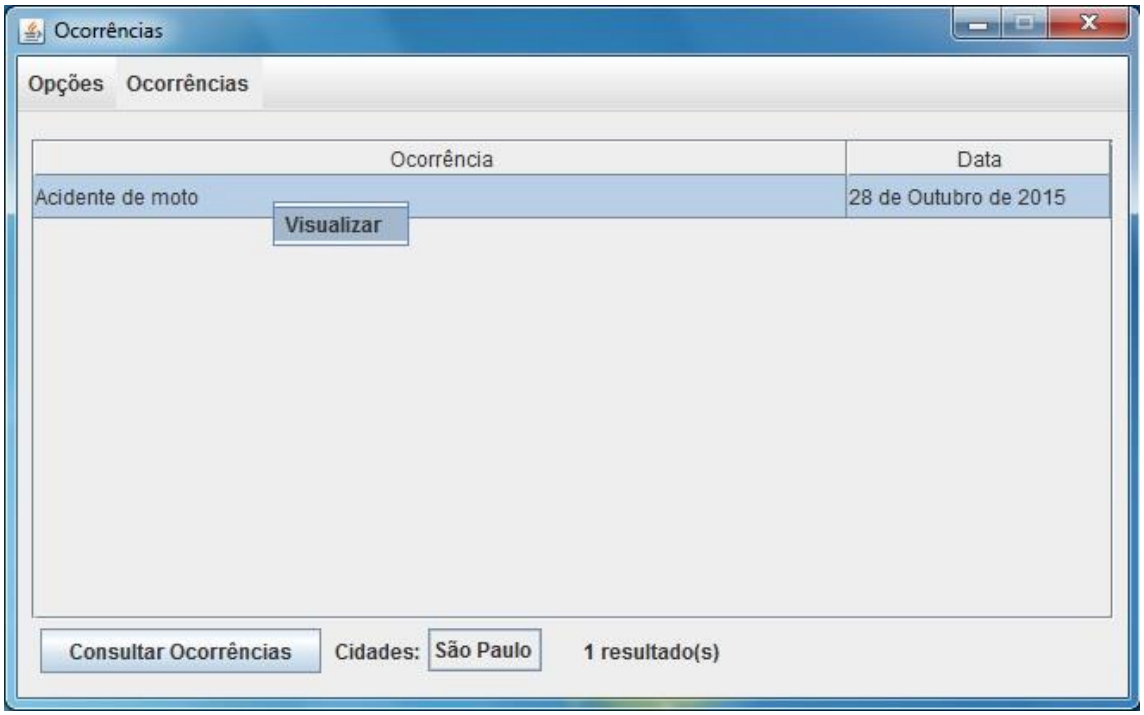
Ao clicar em Consultar ocorrências, o programa *desktop* envia uma solicitação ao servidor e traz as ocorrências gravadas no banco, dentro da cidade de São Paulo, por exemplo.

Figura 13: Resultado do consumo de Webservice de consulta de ocorrências



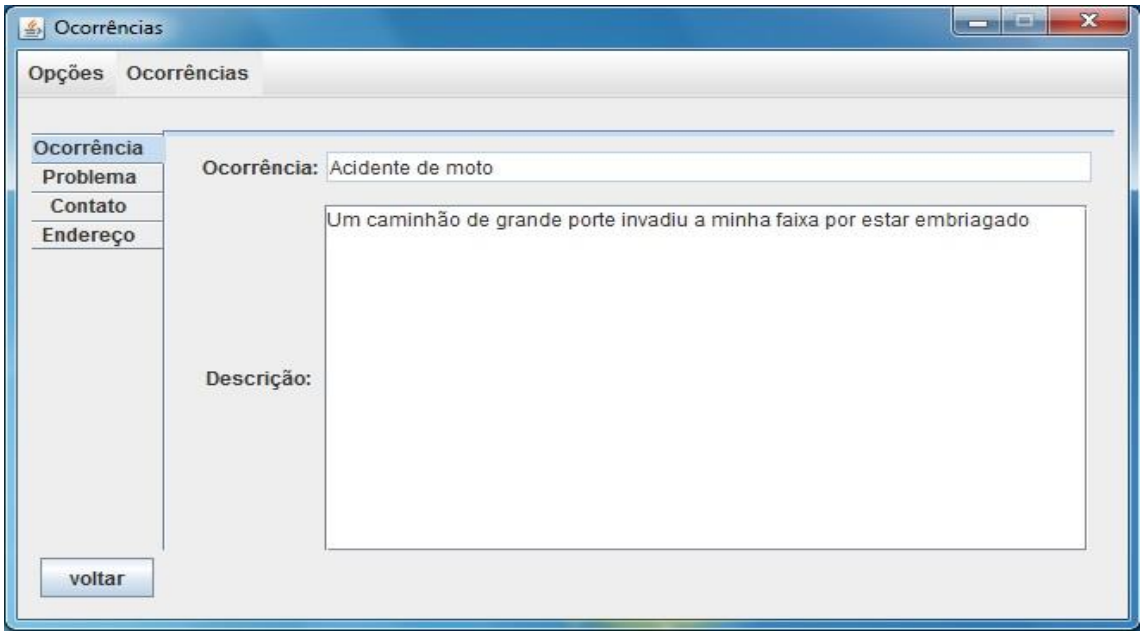
Foi inserido uma opção para consultar os detalhes das ocorrências, acessado pelo botão direito do mouse.

Figura 14: Acessando o menu de detalhes de determinada ocorrência



Esta aplicação, como foi dito em sessão anterior, foi desenvolvida com o objetivo de testar as funcionalidades de consumo do *webservice.* Assim, é exibido um menu para possibilitar ao usuário ter acesso a detalhes de ocorrência.

Figura 15: Menu de opções dos detalhes de determinada ocorrência



4.6 XMLs utilizados pelo webservice

Além de aplicar conceitos do *webservice*, este trabalho se preocupou em testar conceitos de consulta gradativa sob demanda. Uma ocorrência contém grupos de informações que não necessariamente precisam ser trazidas ao usuário no primeiro consumo. Assim sendo, é apresentado as imagens dos arquivos XMLs acessados conforme o usuário solicita determinada informação.

Figura 16: XML inicial de consulta da ocorrência

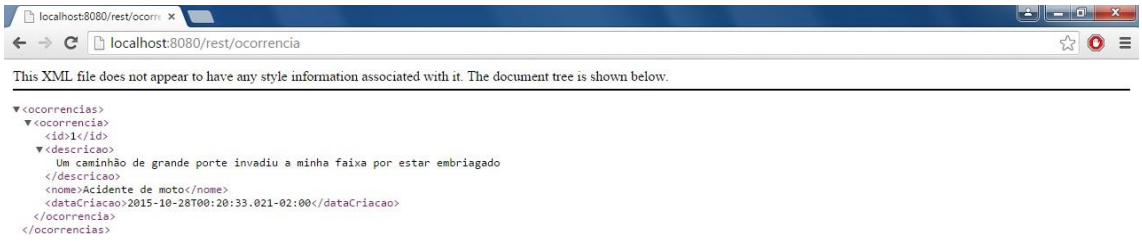


Figura 17: XML de consulta do problema de determinada ocorrência

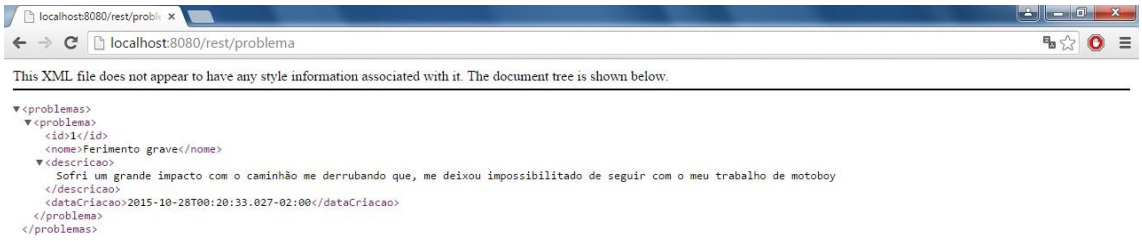


Figura 18: XML dos dados de contato de determinada ocorrência

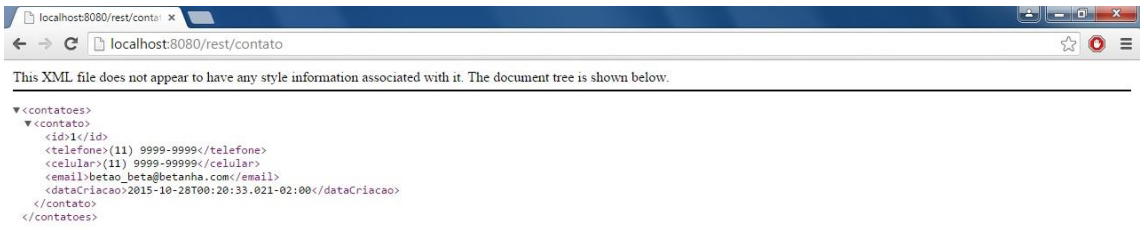
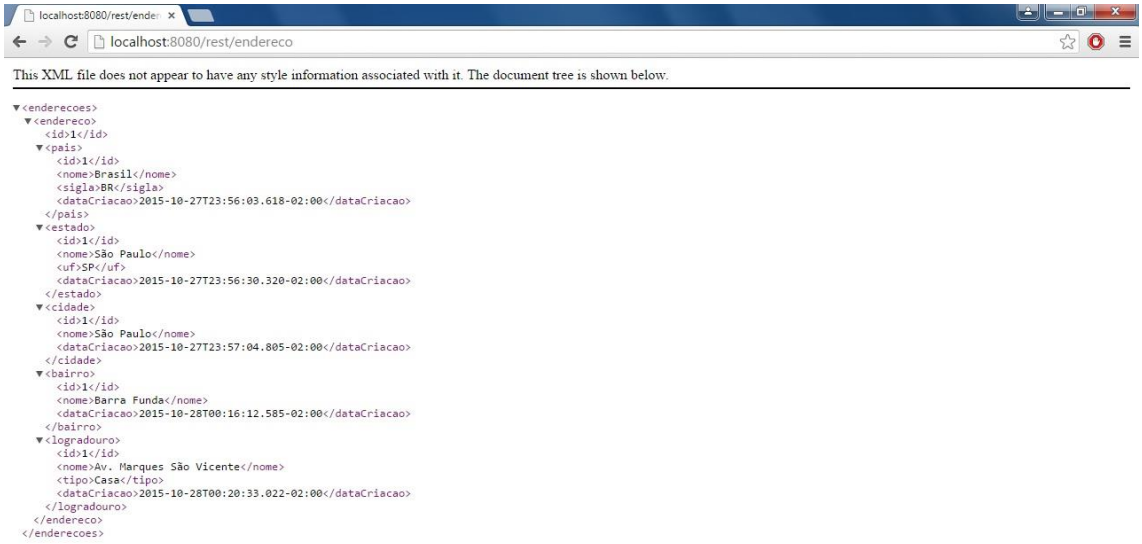


Figura 19: XML de dados de endereço de determinada ocorrência



As figuras acima demonstram que uma ocorrência possui dados separados em grupo. Se estas informações fossem inseridas em um único arquivo, aumentaria o custo de processamento no tráfego. Isto é o espelho da realidade, demonstra a necessidade dos projetistas a desenhar *webservices* que consomem dados sob demanda, a fim de não onerar o processamento geral do sistema distribuído.

5 CONCLUSÂO

O estudo apresentado sobre o desenvolvimento de um sistema distribuído, bem como a criação de um exemplo trouxe uma experiência engrandecedora ao grupo. Foi reforçado a importância apresentada da literatura de se entender e isolar muito bem as camadas do sistema, sendo a apresentação, controle, do modelo de negócio e a do banco de dados.

Este trabalho trouxe a oportunidade de serem compreendidos os conceitos principais que regem um sistema web multicamadas. Foi estudado e aplicado estudo de especificações, bibliotecas e *frameworks* utilizados no mercado como Hibernate e JSF.

Foi possível também criar um *webservice*, onde foi possível fazer com que o sistema desenvolvido não se tornasse uma “ilha”, podendo se comunicar com outros sistemas, como o também desenvolvido e apresentado sistema de consulta de ocorrências criado com as bibliotecas *swing* e *awt.*

Por fim, ao leitor, convidamos a utilizar este trabalho como referência a trabalhos futuros, aprofundando em questões como autenticação, segurança e aplicação de criptografia.

Os códigos fontes do trabalho do sistema web, em todas suas camadas, bem como do segundo sistema, o *desktop*, podem ser verificados junto à sessão anexos deste trabalho.

REFERÊNCIAS

SOMMERVILLE, Ian. **Engenharia de Software.** 9.a edição. São Paulo: Pearson Prentice Hall, 2011.

JOSUTTIS, Nicolai. **SOA NA PRÁTICA: A arte de modelagem de sistemas distribuídos**; 1ª. Edição. Rio de Janeiro: Alta Books, 2008.

PULIER, Eric; TAYLOR, Hugh. **Compreendendo SOA Corporativa**.1ª. Edição. Rio de Janeiro: Ciência Moderna, 2008.

HURWITZ, Judith; BLOOR, Robin; KAUFMAN, Marcia; HALPER, Fern. **Arquitetura orientada a serviços – SOA para leigos.** 2.a edição. Rio de Janeiro: Alta Books, 2009.

Anexo A: Código fonte do servidor e aplicação cliente de acesso via browser

```
package br.com.unip.aps.ws.config;

import java.util.HashSet;
import java.util.Set;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("rest")
public class ApplicationConfig extends Application {

	@Override
	public Set<Class<?>> getClasses() {
		Set<Class<?>> resources = new HashSet<>();
		return resources;
	}

}
package br.com.unip.aps.ws.model;

import java.io.Serializable;
import java.util.Collections;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "bairro")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
```

```
public class Bairro implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -6323385116926791393L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 100, message = "*O tamanho requerido é de 2 ao 100.")
	@Column(length = 100, nullable = false, insertable = true, updatable = true, unique =
true)
	private String nome;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"bairro")
	private List<Endereco> enderecos = Collections.<Endereco> emptyList();

	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;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
```

```
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public List<Endereco> getEnderecos() {
		return enderecos;
	}

	public void setEnderecos(List<Endereco> enderecos) {
		this.enderecos = enderecos;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((nome == null) ? 0 : nome.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Bairro other = (Bairro) obj;
		if (dataCriacao == null) {
			if (other.dataCriacao != null)
				return false;
		} else if (!dataCriacao.equals(other.dataCriacao))
			return false;
		if (dataModificacao == null) {
			if (other.dataModificacao != null)
				return false;
		} else if (!dataModificacao.equals(other.dataModificacao))
			return false;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (!id.equals(other.id))
			return false;
		if (nome == null) {
			if (other.nome != null)
				return false;
```

```
		} else if (!nome.equals(other.nome))
			return false;
		return true;
	}

	@PrePersist
	public void prePersist() {
		dataCriacao = DateTime.now();
	}

	@PreUpdate
	public void preUpdate() {
		dataModificacao = DateTime.now();
	}

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.BairroDao;
import br.com.unip.aps.ws.model.Bairro;

@ManagedBean
@RequestScoped
public class BairroBean {

	@Inject
	private BairroDao dao;
	private Bairro bairro = new Bairro();
	private List<Bairro> bairros;

	public void grava() {

		try {
			if (bairro.getId() == null)
				dao.adiciona(bairro);
			else
				dao.atualiza(bairro);

			addMessage(bairro.getNome() + " gravado!",
					FacesMessage.SEVERITY_INFO);

			bairro = new Bairro();
			bairros = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar gravar o bairro!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public void remove(Bairro bairro) {
		try {
```

```
                        dao.remove(bairro);
                        addMessage(bairro.getNome() + " removido!",
                                        FacesMessage.SEVERITY_INFO);
                        bairros = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar remover o bairro!",
                                        FacesMessage.SEVERITY_ERROR);
                }
        }

        public List<Bairro> getBairros() {

                if (bairros == null)
                        bairros = dao.listaTodos();

                return bairros;
        }

        public Bairro getBairro() {
                return bairro;
        }

        public void setBairro(Bairro bairro) {
                this.bairro = bairro;
        }

        public void addMessage(String summary, Severity severity) {
                FacesMessage message = new FacesMessage(severity, summary, null);
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import br.com.unip.aps.ws.model.Bairro;

public class BairroDao extends Dao<Bairro> implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = -8699341393158039947L;

        public BairroDao() {
                super(Bairro.class);
        }

        public Bairro buscaPorNome(String nome) {

                String query = "select * from bairro b where b.nome=:nome";

                List<Bairro> lista = em.createQuery(query, Bairro.class)
                                .setParameter("nome", nome).getResultList();

                return lista == null ? null : lista.get(0);
        }
```

```
}
package br.com.unip.aps.ws.adapter;

import java.util.Calendar;

import javax.xml.bind.DatatypeConverter;
import javax.xml.bind.annotation.adapters.XmlAdapter;

public class CalendarAdapter extends XmlAdapter<String, Calendar> {

	@Override
	public String marshal(Calendar c) throws Exception {
		return DatatypeConverter.printDate(c);
	}

	@Override
	public Calendar unmarshal(String s) throws Exception {
		return DatatypeConverter.parseDate(s);
	}

}package br.com.unip.aps.ws.model;

import java.io.Serializable;
import java.util.Collections;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "cidade")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Cidade implements Serializable {

	/**
```

```
	 *
	 */
	private static final long serialVersionUID = 8199089019463651429L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 100, message = "*O tamanho requerido é de 2 ao 100.")
	@Column(length = 100, nullable = false, insertable = true, updatable = true, unique =
true)
	private String nome;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"cidade")
	private List<Endereco> enderecos = Collections.<Endereco> emptyList();

	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;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}
```

```
	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public List<Endereco> getEnderecos() {
		return enderecos;
	}

	public void setEnderecos(List<Endereco> enderecos) {
		this.enderecos = enderecos;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((nome == null) ? 0 : nome.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Cidade other = (Cidade) obj;
		if (dataCriacao == null) {
			if (other.dataCriacao != null)
				return false;
		} else if (!dataCriacao.equals(other.dataCriacao))
			return false;
		if (dataModificacao == null) {
			if (other.dataModificacao != null)
				return false;
		} else if (!dataModificacao.equals(other.dataModificacao))
			return false;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (!id.equals(other.id))
			return false;
		if (nome == null) {
			if (other.nome != null)
				return false;
		} else if (!nome.equals(other.nome))
			return false;
		return true;
```

```
	}

	@PrePersist
	public void prePersist() {
		dataCriacao = DateTime.now();
	}

	@PreUpdate
	public void preUpdate() {
		dataModificacao = DateTime.now();
	}

	@Override
	public String toString() {
		return nome;
	}

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.CidadeDao;
import br.com.unip.aps.ws.model.Cidade;

@ManagedBean
@RequestScoped
public class CidadeBean {

	@Inject
	private CidadeDao dao;
	private Cidade cidade = new Cidade();
	private List<Cidade> cidades;

	public void grava() {

		try {
			if (cidade.getId() == null)
				dao.adiciona(cidade);
			else
				dao.atualiza(cidade);

			addMessage(cidade.getNome() + " gravado!",
					FacesMessage.SEVERITY_INFO);

			cidade = new Cidade();
			cidades = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar gravar a cidade!",
					FacesMessage.SEVERITY_ERROR);
		}
	}
```

```
	public void remove(Cidade cidade) {
		try {
			dao.remove(cidade);
			addMessage(cidade.getNome() + " removido!",
					FacesMessage.SEVERITY_INFO);
			cidades = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar remover a cidade!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public List<Cidade> getCidades() {

		if (cidades == null)
			cidades = dao.listaTodos();

		return cidades;
	}

	public Cidade getCidade() {
		return cidade;
	}

	public void setCidade(Cidade cidade) {
		this.cidade = cidade;
	}

	public void addMessage(String summary, Severity severity) {
		FacesMessage message = new FacesMessage(severity, summary, null);
		FacesContext.getCurrentInstance().addMessage(null, message);
	}

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import br.com.unip.aps.ws.model.Cidade;

public class CidadeDao extends Dao<Cidade> implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = 6905511793057488872L;

	public CidadeDao() {
		super(Cidade.class);
	}

	public Cidade buscaPorNome(String nome) {

		String query = "select * from cidade c where c.nome=:nome";

		List<Cidade> lista = em.createQuery(query, Cidade.class)
				.setParameter("nome", nome).getResultList();

		return lista == null ? null : lista.get(0);
```

```
	}

}
package br.com.unip.aps.ws.rest;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.unip.aps.ws.dao.CidadeDao;
import br.com.unip.aps.ws.model.Cidade;

@Path("/cidade")
public class CidadeRest {

	@Inject
	private CidadeDao dao;

	@GET
	@Produces(MediaType.APPLICATION_XML)
	public List<Cidade> getCidades() {
		return dao.listaTodos();
	}

	@GET
	@Path("/{id}")
	@Produces(MediaType.APPLICATION_XML)
	public Cidade getCidade(@PathParam("id") long id) {
		return dao.buscaPorId(id);
	}

}package br.com.unip.aps.ws.model;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
```

```
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "contato")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Contato implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -5272202378740191163L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "contato_id", nullable = false, insertable = true, updatable = false,
unique = true)
	private Long id;

	@XmlElement(name = "telefone", required = true)
	@NotEmpty
	@Pattern(regexp = "^$|[(]\\d{2}[)]\\s\\d{4}[-]\\d{4}", message = "Formato inválido!")
	@Size(min = 14, max = 14, message = "*O tamanho requerido é 14.")
	@Column(length = 14, nullable = false, insertable = true, updatable = false, unique =
false)
	private String telefone;

	@XmlElement(name = "celular", required = true)
	@NotEmpty
	@Pattern(regexp = "^$|[(]\\d{2}[)]\\s\\d{4}[-]\\d{4,5}", message = "Formato inválido!")
	@Size(min = 14, max = 15, message = "*O tamanho requerido é de 14 ao 15.")
	@Column(length = 15, nullable = false, insertable = true, updatable = false, unique =
false)
	private String celular;

	@XmlElement(name = "email", required = true)
	@NotEmpty
	@Email(message = "Formato inválido!")
	@Size(min = 5, max = 120, message = "*O tamanho requerido é de 5 ao 120.")
	@Column(length = 120, nullable = false, insertable = true, updatable = false, unique =
false)
	private String email;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
```

```
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToOne(optional = false, fetch = FetchType.LAZY)
	@JoinColumn(name = "ocorrencia_id", nullable = false, insertable = true, updatable =
false, unique = false)
	private Ocorrencia ocorrencia = new Ocorrencia();

	public Long getId() {
		return id;
	}

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

	public String getTelefone() {
		return telefone;
	}

	public void setTelefone(String telefone) {
		this.telefone = telefone;
	}

	public String getCelular() {
		return celular;
	}

	public void setCelular(String celular) {
		this.celular = celular;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public Ocorrencia getOcorrencia() {
```

```
		return ocorrencia;
	}

	public void setOcorrencia(Ocorrencia ocorrencia) {
		this.ocorrencia = ocorrencia;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((celular == null) ? 0 : celular.hashCode());
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result + ((email == null) ? 0 : email.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result
				+ ((telefone == null) ? 0 : telefone.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Contato other = (Contato) obj;
		if (celular == null) {
			if (other.celular != null)
				return false;
		} else if (!celular.equals(other.celular))
			return false;
		if (dataCriacao == null) {
			if (other.dataCriacao != null)
				return false;
		} else if (!dataCriacao.equals(other.dataCriacao))
			return false;
		if (dataModificacao == null) {
			if (other.dataModificacao != null)
				return false;
		} else if (!dataModificacao.equals(other.dataModificacao))
			return false;
		if (email == null) {
			if (other.email != null)
				return false;
		} else if (!email.equals(other.email))
			return false;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (!id.equals(other.id))
			return false;
		if (telefone == null) {
			if (other.telefone != null)
```

```
                        return false;
                } else if (!telefone.equals(other.telefone))
                        return false;
                return true;
        }

        @PrePersist
        public void prePersist() {
                dataCriacao = DateTime.now();
        }

        @PreUpdate
        public void preUpdate() {
                dataModificacao = DateTime.now();
        }

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.ContatoDao;
import br.com.unip.aps.ws.model.Contato;

@ManagedBean
@RequestScoped
public class ContatoBean {

        @Inject
        private ContatoDao dao;
        private Contato contato = new Contato();
        private List<Contato> contatos;

        public void grava() {

                try {
                        if (contato.getId() == null)
                                dao.adiciona(contato);
                        else
                                dao.atualiza(contato);

                        addMessage(contato.getEmail() + " gravado!",
                                        FacesMessage.SEVERITY_INFO);

                        contato = new Contato();
                        contatos = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar gravar o contato!",
                                        FacesMessage.SEVERITY_ERROR);
                }
        }

        public void remove(Contato contato) {
```

```
                try {
                        dao.remove(contato);
                        addMessage(contato.getEmail() + " removido!",
                                        FacesMessage.SEVERITY_INFO);
                        contatos = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar remover o contato!",
                                        FacesMessage.SEVERITY_ERROR);
                }
        }

        public List<Contato> getContatos() {

                if (contatos == null)
                        contatos = dao.listaTodos();

                return contatos;
        }

        public Contato getContato() {
                return contato;
        }

        public void setContato(Contato contato) {
                this.contato = contato;
        }

        public void addMessage(String summary, Severity severity) {
                FacesMessage message = new FacesMessage(severity, summary, null);
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Restrictions;

import br.com.unip.aps.ws.model.Contato;

public class ContatoDao extends Dao<Contato> implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = -2595353112067142139L;

        public ContatoDao() {
                super(Contato.class);
        }

        public Contato buscaPorIdOcorrencia(Long id) {
                Session session = (Session) em.getDelegate();
                Criteria criteria = session.createCriteria(Contato.class, "c")
                                .createAlias("c.ocorrencia", "o")
                                .add(Restrictions.eq("o.id", id)).setMaxResults(1)
```

```
                .setFetchSize(1);
        return (Contato) criteria.uniqueResult();
    }

    public Contato buscaPorEmail(String email) {

        String query = "select * from contato c where c.email=:email";

        List<Contato> lista = em.createQuery(query, Contato.class)
                .setParameter("email", email).getResultList();

        return lista == null ? null : lista.get(0);
    }

}
package br.com.unip.aps.ws.rest;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.unip.aps.ws.dao.ContatoDao;
import br.com.unip.aps.ws.model.Contato;

@Path("/contato")
public class ContatoRest {

    @Inject
    private ContatoDao dao;

    @GET
    @Produces(MediaType.APPLICATION_XML)
    public List<Contato> getContatos() {
        return dao.listaTodos();
    }

    @GET
    @Path("/{id}")
    @Produces(MediaType.APPLICATION_XML)
    public Contato getContato(@PathParam("id") long id) {
        return dao.buscaPorIdOcorrencia(id);
    }

}package br.com.unip.aps.ws.dao;

import java.util.List;

import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.criteria.CriteriaQuery;

import br.com.unip.aps.ws.tx.Transactional;

public class Dao<T> {
```

```
	private final Class<T> clazz;

	@Inject
	protected EntityManager em;

	public Dao(Class<T> clazz) {
		this.clazz = clazz;
	}

	@Transactional
	public void adiciona(T t) {
		em.persist(t);
	}

	@Transactional
	public void remove(T t) {
		em.remove(em.merge(t));
	}

	@Transactional
	public void atualiza(T t) {
		em.merge(t);
	}

	public T buscaPorId(Long id) {
		T instancia = em.find(clazz, id);
		return instancia;
	}

	public int contaTodos() {
		long result = (Long) em.createQuery(
				"select count(n) from " + clazz.getSimpleName() + " n")
				.getSingleResult();

		return (int) result;
	}

	public List<T> listaTodos() {
		CriteriaQuery<T> query = em.getCriteriaBuilder().createQuery(clazz);
		query.select(query.from(clazz));

		List<T> lista = em.createQuery(query).getResultList();

		return lista;
	}

	public List<T> listaTodosPaginada(int firstResult, int maxResults) {
		CriteriaQuery<T> query = em.getCriteriaBuilder().createQuery(clazz);
		query.select(query.from(clazz));

		List<T> lista = em.createQuery(query).setFirstResult(firstResult)
				.setMaxResults(maxResults).getResultList();
		return lista;
	}

}
package br.com.unip.aps.ws.adapter;

import javax.xml.bind.annotation.adapters.XmlAdapter;
```

```
import org.joda.time.DateTime;

public class DateTimeAdapter extends XmlAdapter<String, DateTime> {

	public DateTime unmarshal(String s) throws Exception {
		return new DateTime(s);
	}

	public String marshal(DateTime dt) throws Exception {
		return dt.toString();
	}

}package br.com.unip.aps.ws.model;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

@XmlRootElement(name = "endereco")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Endereco implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = 1932543763283037 25L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@XmlElement(name = "pais", required = true)
	@ManyToOne(fetch = FetchType.EAGER, optional = false)
	@JoinColumn(name = "pais", nullable = false, insertable = true, updatable = true,
unique = false)
	private Pais pais = new Pais();

	@XmlElement(name = "estado", required = true)
	@ManyToOne(fetch = FetchType.EAGER, optional = false)
	@JoinColumn(name = "estado", nullable = false, insertable = true, updatable = true,
unique = false)
```

```
	private Estado estado = new Estado();

	@XmlElement(name = "cidade", required = true)
	@ManyToOne(fetch = FetchType.EAGER, optional = false)
	@JoinColumn(name = "cidade", nullable = false, insertable = true, updatable = true,
unique = false)
	private Cidade cidade = new Cidade();

	@XmlElement(name = "bairro", required = true)
	@ManyToOne(fetch = FetchType.EAGER, optional = false)
	@JoinColumn(name = "bairro", nullable = false, insertable = true, updatable = true,
unique = false)
	private Bairro bairro = new Bairro();

	@XmlElement(name = "logradouro", required = true)
	@ManyToOne(fetch = FetchType.EAGER, optional = false, cascade =
CascadeType.ALL)
	@JoinColumn(name = "logradouro", nullable = false, insertable = true, updatable =
true, unique = false)
	private Logradouro logradouro = new Logradouro();

	@XmlTransient
	@OneToOne(optional = false, fetch = FetchType.LAZY)
	@JoinColumn(name = "ocorrencia_id", nullable = false, insertable = true, updatable =
false, unique = false)
	private Ocorrencia ocorrencia = new Ocorrencia();

	public Long getId() {
		return id;
	}

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

	public Pais getPais() {
		return pais;
	}

	public void setPais(Pais pais) {
		this.pais = pais;
	}

	public Estado getEstado() {
		return estado;
	}

	public void setEstado(Estado estado) {
		this.estado = estado;
	}

	public Cidade getCidade() {
		return cidade;
	}

	public void setCidade(Cidade cidade) {
		this.cidade = cidade;
	}

	public Bairro getBairro() {
```

```
		return bairro;
	}

	public void setBairro(Bairro bairro) {
		this.bairro = bairro;
	}

	public Logradouro getLogradouro() {
		return logradouro;
	}

	public void setLogradouro(Logradouro logradouro) {
		this.logradouro = logradouro;
	}

	public Ocorrencia getOcorrencia() {
		return ocorrencia;
	}

	public void setOcorrencia(Ocorrencia ocorrencia) {
		this.ocorrencia = ocorrencia;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((bairro == null) ? 0 : bairro.hashCode());
		result = prime * result + ((cidade == null) ? 0 : cidade.hashCode());
		result = prime * result + ((estado == null) ? 0 : estado.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result
				+ ((logradouro == null) ? 0 : logradouro.hashCode());
		result = prime * result + ((pais == null) ? 0 : pais.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Endereco other = (Endereco) obj;
		if (bairro == null) {
			if (other.bairro != null)
				return false;
		} else if (!bairro.equals(other.bairro))
			return false;
		if (cidade == null) {
			if (other.cidade != null)
				return false;
		} else if (!cidade.equals(other.cidade))
			return false;
		if (estado == null) {
			if (other.estado != null)
				return false;
		} else if (!estado.equals(other.estado))
```

```
                return false;
            if (id == null) {
                if (other.id != null)
                    return false;
            } else if (!id.equals(other.id))
                return false;
            if (logradouro == null) {
                if (other.logradouro != null)
                    return false;
            } else if (!logradouro.equals(other.logradouro))
                return false;
            if (pais == null) {
                if (other.pais != null)
                    return false;
            } else if (!pais.equals(other.pais))
                return false;
            return true;
        }

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.EnderecoDao;
import br.com.unip.aps.ws.model.Endereco;

@ManagedBean
@RequestScoped
public class EnderecoBean {

        @Inject
        private EnderecoDao dao;
        private Endereco endereco = new Endereco();
        private List<Endereco> enderecos;

        public void grava() {

            try {
                if (endereco.getId() == null)
                    dao.adiciona(endereco);
                else
                    dao.atualiza(endereco);

                addMessage("Endereco gravado!", FacesMessage.SEVERITY_INFO);

                endereco = new Endereco();
                enderecos = dao.listaTodos();
            } catch (Exception ex) {
                addMessage("Houve um erro ao tentar gravar o endereco!",
                        FacesMessage.SEVERITY_ERROR);
            }
        }
```

```
	public void remove(Endereco endereco) {
		try {
			dao.remove(endereco);
			addMessage("Endereco removido!", FacesMessage.SEVERITY_INFO);
			enderecos = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar remover o endereco!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public List<Endereco> getEnderecos() {

		if (enderecos == null)
			enderecos = dao.listaTodos();

		return enderecos;
	}

	public Endereco getEndereco() {
		return endereco;
	}

	public void setEndereco(Endereco endereco) {
		this.endereco = endereco;
	}

	public void addMessage(String summary, Severity severity) {
		FacesMessage message = new FacesMessage(severity, summary, null);
		FacesContext.getCurrentInstance().addMessage(null, message);
	}

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Restrictions;

import br.com.unip.aps.ws.model.Endereco;

public class EnderecoDao extends Dao<Endereco> implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = 164475287795558614 8L;

	public EnderecoDao() {
		super(Endereco.class);
	}

	public Endereco buscaPorIdOcorrencia(Long id) {
		Session session = (Session) em.getDelegate();
		Criteria criteria = session.createCriteria(Endereco.class, "e")
				.createAlias("e.ocorrencia", "o")
```

```
                .add(Restrictions.eq("o.id", id)).setMaxResults(1)
                .setFetchSize(1);
        return (Endereco) criteria.uniqueResult();
    }

    public List<Endereco> buscaPorCidade(String nome) {

        String query = "select * from endereco e "
                + "inner join cidade c on c.id = e.cidade "
                + "where c.nome=:nome";

        List<Endereco> lista = em.createQuery(query, Endereco.class)
                .setParameter("nome", nome).getResultList();

        return lista;
    }

}
package br.com.unip.aps.ws.rest;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.unip.aps.ws.dao.EnderecoDao;
import br.com.unip.aps.ws.model.Endereco;

@Path("/endereco")
public class EnderecoRest {

    @Inject
    private EnderecoDao dao;

    @GET
    @Produces(MediaType.APPLICATION_XML)
    public List<Endereco> getEnderecos() {
        return dao.listaTodos();
    }

    @GET
    @Path("/{id}")
    @Produces(MediaType.APPLICATION_XML)
    public Endereco getEndereco(@PathParam("id") long id) {
        return dao.buscaPorIdOcorrencia(id);
    }

}package br.com.unip.aps.ws.model;

import java.io.Serializable;
import java.util.Collections;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
```

```
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "estado")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Estado implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = 3737269507365671603L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 100, message = "*O tamanho requerido é de 2 ao 100.")
	@Column(length = 100, nullable = false, insertable = true, updatable = true, unique =
true)
	private String nome;

	@XmlElement(name = "uf", required = true)
	@NotEmpty
	@Pattern(regexp = "[a-zA-Z]{2}", message = "Formato inválido!")
	@Size(min = 2, max = 2, message = "*O tamanho requerido é 2.")
	@Column(length = 2, nullable = false, insertable = true, updatable = true, unique =
true)
	private String uf;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
```

```
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"estado")
	private List<Endereco> enderecos = Collections.<Endereco> emptyList();

	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;
	}

	public String getUf() {
		return uf;
	}

	public void setUf(String uf) {
		this.uf = uf;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public List<Endereco> getEnderecos() {
		return enderecos;
	}
```

```
	public void setEnderecos(List<Endereco> enderecos) {
		this.enderecos = enderecos;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result
				+ ((enderecos == null) ? 0 : enderecos.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((nome == null) ? 0 : nome.hashCode());
		result = prime * result + ((uf == null) ? 0 : uf.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Estado other = (Estado) obj;
		if (dataCriacao == null) {
			if (other.dataCriacao != null)
				return false;
		} else if (!dataCriacao.equals(other.dataCriacao))
			return false;
		if (dataModificacao == null) {
			if (other.dataModificacao != null)
				return false;
		} else if (!dataModificacao.equals(other.dataModificacao))
			return false;
		if (enderecos == null) {
			if (other.enderecos != null)
				return false;
		} else if (!enderecos.equals(other.enderecos))
			return false;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (!id.equals(other.id))
			return false;
		if (nome == null) {
			if (other.nome != null)
				return false;
		} else if (!nome.equals(other.nome))
			return false;
		if (uf == null) {
			if (other.uf != null)
				return false;
		} else if (!uf.equals(other.uf))
			return false;
```

```
		return true;
	}

	@PrePersist
	public void prePersist() {
		dataCriacao = DateTime.now();
	}

	@PreUpdate
	public void preUpdate() {
		dataModificacao = DateTime.now();
	}

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.EstadoDao;
import br.com.unip.aps.ws.model.Estado;

@ManagedBean
@RequestScoped
public class EstadoBean {

	@Inject
	private EstadoDao dao;
	private Estado estado = new Estado();
	private List<Estado> estados;

	public void grava() {

		try {
			if (estado.getId() == null)
				dao.adiciona(estado);
			else
				dao.atualiza(estado);

			addMessage(estado.getNome() + " gravado!",
					FacesMessage.SEVERITY_INFO);

			estado = new Estado();
			estados = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar gravar a estado!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public void remove(Estado estado) {
		try {
			dao.remove(estado);
			addMessage(estado.getNome() + " removido!",
```

```
                                        FacesMessage.SEVERITY_INFO);
                        estados = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar remover o estado!",
                                        FacesMessage.SEVERITY_ERROR);
                }
        }

        public List<Estado> getEstados() {

                if (estados == null)
                        estados = dao.listaTodos();

                return estados;
        }

        public Estado getEstado() {
                return estado;
        }

        public void setEstado(Estado estado) {
                this.estado = estado;
        }

        public void addMessage(String summary, Severity severity) {
                FacesMessage message = new FacesMessage(severity, summary, null);
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import br.com.unip.aps.ws.model.Estado;

public class EstadoDao extends Dao<Estado> implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = -1807410302737862360L;

        public EstadoDao() {
                super(Estado.class);
        }

        public Estado buscaPorNome(String nome) {

                String query = "select * from estado e where e.nome=:nome";

                List<Estado> lista = em.createQuery(query, Estado.class)
                                .setParameter("nome", nome).getResultList();

                return lista == null ? null : lista.get(0);
        }

}
package br.com.unip.aps.ws.util;
```

```
import javax.annotation.PreDestroy;
import javax.enterprise.context.ApplicationScoped;
import javax.enterprise.context.RequestScoped;
import javax.enterprise.inject.Disposes;
import javax.enterprise.inject.Produces;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

@ApplicationScoped
public class JPAUtil {

	private static EntityManagerFactory emf =
Persistence.createEntityManagerFactory("aps-web-service");

	@RequestScoped
	@Produces
	public EntityManager getEntityManager() {
		return emf.createEntityManager();
	}

	public void close(@Disposes EntityManager em) {
		em.close();
	}

	@PreDestroy
	public void fechaFactory() {
		emf.close();
	}
}package br.com.unip.aps.ws.model;

import java.io.Serializable;
import java.util.Collections;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;
```

```
import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "logradouro")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Logradouro implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -7190069287271927273L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 150, message = "*O tamanho requerido é de 2 ao 150.")
	@Column(length = 150, nullable = false, insertable = true, updatable = true, unique =
true)
	private String nome;

	@XmlElement(name = "tipo", required = true)
	@NotEmpty
	@Size(min = 2, max = 30, message = "*O tamanho requerido é de 2 ao 30.")
	@Column(length = 30, nullable = false, insertable = true, updatable = true, unique =
true)
	private String tipo;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"logradouro")
	private List<Endereco> enderecos = Collections.<Endereco> emptyList();

	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;
	}

	public String getTipo() {
		return tipo;
	}

	public void setTipo(String tipo) {
		this.tipo = tipo;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public List<Endereco> getEnderecos() {
		return enderecos;
	}

	public void setEnderecos(List<Endereco> enderecos) {
		this.enderecos = enderecos;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((nome == null) ? 0 : nome.hashCode());
		result = prime * result + ((tipo == null) ? 0 : tipo.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
```

```
                if (obj == null)
                        return false;
                if (getClass() != obj.getClass())
                        return false;
                Logradouro other = (Logradouro) obj;
                if (dataCriacao == null) {
                        if (other.dataCriacao != null)
                                return false;
                } else if (!dataCriacao.equals(other.dataCriacao))
                        return false;
                if (dataModificacao == null) {
                        if (other.dataModificacao != null)
                                return false;
                } else if (!dataModificacao.equals(other.dataModificacao))
                        return false;
                if (id == null) {
                        if (other.id != null)
                                return false;
                } else if (!id.equals(other.id))
                        return false;
                if (nome == null) {
                        if (other.nome != null)
                                return false;
                } else if (!nome.equals(other.nome))
                        return false;
                if (tipo == null) {
                        if (other.tipo != null)
                                return false;
                } else if (!tipo.equals(other.tipo))
                        return false;
                return true;
        }

        @PrePersist
        public void prePersist() {
                dataCriacao = DateTime.now();
        }

        @PreUpdate
        public void preUpdate() {
                dataModificacao = DateTime.now();
        }

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.LogradouroDao;
import br.com.unip.aps.ws.model.Logradouro;

@ManagedBean
@RequestScoped
```

```
public class LogradouroBean {

	@Inject
	private LogradouroDao dao;
	private Logradouro logradouro = new Logradouro();
	private List<Logradouro> logradouros;

	public void atualiza() {

		try {
			dao.atualiza(logradouro);
			addMessage(logradouro.getNome() + " atualizado!",
					FacesMessage.SEVERITY_INFO);

			logradouro = new Logradouro();
			logradouros = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar atualizar o logradouro!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public List<Logradouro> getLogradouros() {

		if (logradouros == null)
			logradouros = dao.listaTodos();

		return logradouros;
	}

	public Logradouro getLogradouro() {
		return logradouro;
	}

	public void setLogradouro(Logradouro logradouro) {
		this.logradouro = logradouro;
	}

	public void addMessage(String summary, Severity severity) {
		FacesMessage message = new FacesMessage(severity, summary, null);
		FacesContext.getCurrentInstance().addMessage(null, message);
	}

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import br.com.unip.aps.ws.model.Logradouro;

public class LogradouroDao extends Dao<Logradouro> implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -8912381038480363582L;

	public LogradouroDao() {
		super(Logradouro.class);
```

```
	}

	public Logradouro buscaPorNome(String nome) {

		String query = "select * from logradouro l where l.nome=:nome";

		List<Logradouro> lista = em.createQuery(query, Logradouro.class)
				.setParameter("nome", nome).getResultList();

		return lista == null ? null : lista.get(0);
	}

}
package br.com.unip.aps.ws.model;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrePersist;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "ocorrencia")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Ocorrencia implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -5272202378740191163L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ocorrencia_id", nullable = false, insertable = true, updatable =
false, unique = true)
	private Long id;
```

```
	@XmlElement(name = "descricao", required = true)
	@NotEmpty
	@Size(min = 10, max = 500, message = "*O tamanho requerido é de 10 ao 500.")
	@Column(length = 500, nullable = false, insertable = true, updatable = false, unique =
false)
	private String descricao;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 100, message = "*O tamanho requerido é de 2 ao 100.")
	@Column(length = 100, nullable = false, insertable = true, updatable = false, unique =
false)
	private String nome;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@OneToOne(fetch = FetchType.LAZY, mappedBy = "ocorrencia", cascade =
CascadeType.ALL, optional = false, orphanRemoval = true)
	private Problema problema;

	@OneToOne(fetch = FetchType.LAZY, mappedBy = "ocorrencia", cascade =
CascadeType.ALL, optional = false, orphanRemoval = true)
	private Contato contato;

	@OneToOne(fetch = FetchType.LAZY, mappedBy = "ocorrencia", cascade =
CascadeType.ALL, optional = false, orphanRemoval = true)
	private Endereco endereco;

	public Long getId() {
		return id;
	}

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

	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}

	public String getNome() {
		return nome;
	}

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

	public DateTime getDataCriacao() {
		return dataCriacao;
```

```
}

public Date getDataCriacaoDate() {
	return dataCriacao.toDate();
}

public void setDataCriacao(DateTime dataCriacao) {
	this.dataCriacao = dataCriacao;
}

public Problema getProblema() {
	return problema;
}

public void setProblema(Problema problema) {
	this.problema = problema;
}

public Contato getContato() {
	return contato;
}

public void setContato(Contato contato) {
	this.contato = contato;
}

public Endereco getEndereco() {
	return endereco;
}

public void setEndereco(Endereco endereco) {
	this.endereco = endereco;
}

@Override
public int hashCode() {
	final int prime = 31;
	int result = 1;
	result = prime * result
			+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
	result = prime * result
			+ ((descricao == null) ? 0 : descricao.hashCode());
	result = prime * result + ((id == null) ? 0 : id.hashCode());
	result = prime * result + ((nome == null) ? 0 : nome.hashCode());
	return result;
}

@Override
public boolean equals(Object obj) {
	if (this == obj)
		return true;
	if (obj == null)
		return false;
	if (getClass() != obj.getClass())
		return false;
	Ocorrencia other = (Ocorrencia) obj;
	if (dataCriacao == null) {
		if (other.dataCriacao != null)
			return false;
	} else if (!dataCriacao.equals(other.dataCriacao))
```

```
                        return false;
                if (descricao == null) {
                        if (other.descricao != null)
                                return false;
                } else if (!descricao.equals(other.descricao))
                        return false;
                if (id == null) {
                        if (other.id != null)
                                return false;
                } else if (!id.equals(other.id))
                        return false;
                if (nome == null) {
                        if (other.nome != null)
                                return false;
                } else if (!nome.equals(other.nome))
                        return false;
                return true;
        }

        @PrePersist
        public void prePersist() {
                dataCriacao = DateTime.now();
        }

}
package br.com.unip.aps.ws.mb;

import java.io.Serializable;
import java.util.List;

import javax.faces.application.Application;
import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.application.ViewHandler;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.component.UIViewRoot;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import org.primefaces.event.FlowEvent;

import br.com.unip.aps.ws.dao.OcorrenciaDao;
import br.com.unip.aps.ws.model.Contato;
import br.com.unip.aps.ws.model.Endereco;
import br.com.unip.aps.ws.model.Ocorrencia;
import br.com.unip.aps.ws.model.Problema;

@ManagedBean
@ViewScoped
public class OcorrenciaBean implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = 3765206373544000150L;
        @Inject
        private OcorrenciaDao dao;
        private Ocorrencia ocorrencia = new Ocorrencia();
        private Problema problema = new Problema();
```

```
private Contato contato = new Contato();
private Endereco endereco = new Endereco();
private List<Ocorrencia> ocorrencias;

public void gravaTudo() {

	try {
		dao.adicionaTudo(ocorrencia, problema, contato, endereco);

		addMessage(ocorrencia.getNome() + " gravado!",
				FacesMessage.SEVERITY_INFO);

		refresh();
	} catch (Exception ex) {
		addMessage("Houve um erro ao tentar gravar a ocorrência!",
				FacesMessage.SEVERITY_ERROR);
	}
}

public void removeTudo(Ocorrencia ocorrencia) {
	try {
		dao.removeTudo(ocorrencia);
		addMessage(ocorrencia.getNome() + " removido!",
				FacesMessage.SEVERITY_INFO);
		ocorrencias = dao.listaTodos();
	} catch (Exception ex) {
		addMessage("Houve um erro ao tentar remover a ocorrência!",
				FacesMessage.SEVERITY_ERROR);
	}
}

public List<Ocorrencia> getOcorrencias() {

	if (ocorrencias == null)
		ocorrencias = dao.listaTodos();

	return dao.listaTodos();
}

public Ocorrencia getOcorrencia() {
	return ocorrencia;
}

public void setOcorrencia(Ocorrencia ocorrencia) {
	this.ocorrencia = ocorrencia;
}

public Problema getProblema() {
	return problema;
}

public void setProblema(Problema problema) {
	this.problema = problema;
}

public Contato getContato() {
	return contato;
}

public void setContato(Contato contato) {
```

```
                this.contato = contato;
        }

        public Endereco getEndereco() {
                return endereco;
        }

        public void setEndereco(Endereco endereco) {
                this.endereco = endereco;
        }

        public void addMessage(String summary, Severity severity) {
                FacesMessage message = new FacesMessage(severity, summary, null);
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

        public String onFlowProcess(FlowEvent event) {
                return event.getNewStep(); // passa para o proximo passo do cadastro
        }

        public void refresh() {
                FacesContext context = FacesContext.getCurrentInstance();
                Application application = context.getApplication();
                ViewHandler viewHandler = application.getViewHandler();
                UIViewRoot viewRoot = viewHandler.createView(context, context
                                .getViewRoot().getViewId());
                context.setViewRoot(viewRoot);
                context.renderResponse();
        }

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Restrictions;

import br.com.unip.aps.ws.model.Contato;
import br.com.unip.aps.ws.model.Endereco;
import br.com.unip.aps.ws.model.Ocorrencia;
import br.com.unip.aps.ws.model.Problema;
import br.com.unip.aps.ws.tx.Transactional;

public class OcorrenciaDao extends Dao<Ocorrencia> implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = 4198787839468917237L;

        public OcorrenciaDao() {
                super(Ocorrencia.class);
        }

        @Transactional
        public void adicionaTudo(Ocorrencia t, Problema p, Contato c, Endereco e) {
                t.setProblema(p);
```

```
                t.setContato(c);
                t.setEndereco(e);
                p.setOcorrencia(t);
                c.setOcorrencia(t);
                e.setOcorrencia(t);
                em.persist(t);
                em.persist(p);
                em.persist(c);
                em.persist(e.getLogradouro());
                em.persist(e);
        }

        @Transactional
        public void removeTudo(Ocorrencia t) {
                em.remove(t.getProblema());
                em.remove(t.getContato());
                em.remove(t.getEndereco());
                em.remove(t);
        }

        public List<Ocorrencia> listaTodosPorCidade(String cidade) {

                Session session = (Session) em.getDelegate();
                Criteria criteria = session.createCriteria(Ocorrencia.class, "o")
                                .createAlias("o.endereco", "e").createAlias("e.cidade", "c")
                                .add(Restrictions.eq("c.nome", cidade).ignoreCase());

                @SuppressWarnings("unchecked")
                List<Ocorrencia> lista = (List<Ocorrencia>) criteria.list();

                return lista;
        }

}
package br.com.unip.aps.ws.rest;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.unip.aps.ws.dao.OcorrenciaDao;
import br.com.unip.aps.ws.model.Ocorrencia;

@Path("/ocorrencia")
public class OcorrenciaRest {

        @Inject
        private OcorrenciaDao dao;

        @GET
        @Produces(MediaType.APPLICATION_XML)
        public List<Ocorrencia> getOcorrencias() {
                return dao.listaTodos();
        }
```

```
	@GET
	@Path("/{id}")
	@Produces(MediaType.APPLICATION_XML)
	public Ocorrencia getOcorrencia(@PathParam("id") long id) {
		return dao.buscaPorId(id);
	}

	@GET
	@Path("/cidade/{cidade}")
	@Produces(MediaType.APPLICATION_XML)
	public List<Ocorrencia> getOcorrenciasPorCidade(
			@PathParam("cidade") String cidade) {
		return dao.listaTodosPorCidade(cidade);
	}

}package br.com.unip.aps.ws.model;

import java.io.Serializable;
import java.util.Collections;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;

@XmlRootElement(name = "pais")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Pais implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = 5530920272683174712L;
```

```
	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 80, message = "*O tamanho requerido é de 2 ao 80.")
	@Column(length = 80, nullable = false, insertable = true, updatable = true, unique =
true)
	private String nome;

	@XmlElement(name = "sigla", required = true)
	@NotEmpty
	@Pattern(regexp = "[a-zA-Z]{2}", message = "Formato inválido!")
	@Size(min = 2, max = 2, message = "*O tamanho requerido é 2.")
	@Column(length = 2, nullable = false, insertable = true, updatable = true, unique =
true)
	private String sigla;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"pais")
	private List<Endereco> enderecos = Collections.<Endereco> emptyList();

	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;
	}

	public String getSigla() {
		return sigla;
	}
```

```
	public void setSigla(String sigla) {
		this.sigla = sigla;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public List<Endereco> getEnderecos() {
		return enderecos;
	}

	public void setEnderecos(List<Endereco> enderecos) {
		this.enderecos = enderecos;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((nome == null) ? 0 : nome.hashCode());
		result = prime * result + ((sigla == null) ? 0 : sigla.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Pais other = (Pais) obj;
		if (dataCriacao == null) {
			if (other.dataCriacao != null)
				return false;
		} else if (!dataCriacao.equals(other.dataCriacao))
			return false;
		if (dataModificacao == null) {
			if (other.dataModificacao != null)
				return false;
```

```
                } else if (!dataModificacao.equals(other.dataModificacao))
                        return false;
                if (id == null) {
                        if (other.id != null)
                                return false;
                } else if (!id.equals(other.id))
                        return false;
                if (nome == null) {
                        if (other.nome != null)
                                return false;
                } else if (!nome.equals(other.nome))
                        return false;
                if (sigla == null) {
                        if (other.sigla != null)
                                return false;
                } else if (!sigla.equals(other.sigla))
                        return false;
                return true;
        }

        @PrePersist
        public void prePersist() {
                dataCriacao = DateTime.now();
        }

        @PreUpdate
        public void preUpdate() {
                dataModificacao = DateTime.now();
        }

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.PaisDao;
import br.com.unip.aps.ws.model.Pais;

@ManagedBean
@RequestScoped
public class PaisBean {

        @Inject
        private PaisDao dao;
        private Pais pais = new Pais();
        private List<Pais> paises;

        public void grava() {

                try {
                        if (pais.getId() == null)
                                dao.adiciona(pais);
                        else
```

```
                        dao.atualiza(pais);

                        addMessage(pais.getNome() + " gravado!",
FacesMessage.SEVERITY_INFO);
                        pais = new Pais();
                        paises = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar gravar o país!",
                                FacesMessage.SEVERITY_ERROR);
                }
        }

        public void remove(Pais pais) {
                try {
                        dao.remove(pais);
                        addMessage(pais.getNome() + " removido!",
                                FacesMessage.SEVERITY_INFO);
                        paises = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar remover o país!",
                                FacesMessage.SEVERITY_ERROR);
                }
        }

        public List<Pais> getPaises() {

                if (paises == null)
                        paises = dao.listaTodos();

                return paises;
        }

        public Pais getPais() {
                return pais;
        }

        public void setPais(Pais pais) {
                this.pais = pais;
        }

        public void addMessage(String summary, Severity severity) {
                FacesMessage message = new FacesMessage(severity, summary, null);
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import br.com.unip.aps.ws.model.Pais;

public class PaisDao extends Dao<Pais> implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = 2926177340279832639L;
```

```
	public PaisDao() {
		super(Pais.class);
	}

	public Pais buscaPorNome(String nome) {

		String query = "select * from pais p where p.nome=:nome";

		List<Pais> lista = em.createQuery(query, Pais.class)
				.setParameter("nome", nome).getResultList();

		return lista == null ? null : lista.get(0);
	}

}
package br.com.unip.aps.ws.model;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

@Entity
@Table
public class Parametro implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -4791813859257525874L;

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(nullable = false, insertable = true, updatable = false, unique = true)
	private Long id;

	@NotEmpty
	@Size(min = 36, max = 36, message = "*O tamanho requerido e 36.")
	@Column(length = 100, nullable = false, insertable = true, updatable = false, unique =
true)
	private String chave;

	@NotEmpty
	@Size(min = 1, max = 100, message = "*O tamanho requerido e de 1 ao 100.")
	@Column(length = 100, nullable = false, insertable = true, updatable = true, unique =
false)
	private String valor;
```

```
	@NotEmpty
	@Size(min = 2, max = 70, message = "*O tamanho requerido e de 2 ao 100.")
	@Column(length = 100, nullable = false, insertable = true, updatable = true, unique =
false)
	private String descricao;

	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	public Long getId() {
		return id;
	}

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

	public String getChave() {
		return chave;
	}

	public void setChave(String chave) {
		this.chave = chave;
	}

	public String getValor() {
		return valor;
	}

	public void setValor(String valor) {
		this.valor = valor;
	}

	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}
}
```

```
	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((chave == null) ? 0 : chave.hashCode());
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result
				+ ((descricao == null) ? 0 : descricao.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((valor == null) ? 0 : valor.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Parametro other = (Parametro) obj;
		if (chave == null) {
			if (other.chave != null)
				return false;
		} else if (!chave.equals(other.chave))
			return false;
		if (dataCriacao == null) {
			if (other.dataCriacao != null)
				return false;
		} else if (!dataCriacao.equals(other.dataCriacao))
			return false;
		if (dataModificacao == null) {
			if (other.dataModificacao != null)
				return false;
		} else if (!dataModificacao.equals(other.dataModificacao))
			return false;
		if (descricao == null) {
			if (other.descricao != null)
				return false;
		} else if (!descricao.equals(other.descricao))
			return false;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (!id.equals(other.id))
			return false;
		if (valor == null) {
			if (other.valor != null)
				return false;
		} else if (!valor.equals(other.valor))
```

```
                return false;
            return true;
        }

        @PrePersist
        public void prePersist() {
            dataCriacao = DateTime.now();
        }

        @PreUpdate
        public void preUpdate() {
            dataModificacao = DateTime.now();
        }

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.ParametroDao;
import br.com.unip.aps.ws.model.Parametro;

@ManagedBean
@RequestScoped
public class ParametroBean {

        @Inject
        private ParametroDao dao;
        private Parametro parametro = new Parametro();
        private List<Parametro> parametros;

        public void grava() {

            try {
                if (parametro.getId() == null)
                    dao.adiciona(parametro);
                else
                    dao.atualiza(parametro);

                addMessage("Parametro gravado!", FacesMessage.SEVERITY_INFO);

                parametro = new Parametro();
                parametros = dao.listaTodos();
            } catch (Exception ex) {
                addMessage("Houve um erro ao tentar gravar o parametro!",
                        FacesMessage.SEVERITY_ERROR);
            }
        }

        public void remove(Parametro parametro) {
            try {
                dao.remove(parametro);
```

```
                        addMessage("Parametro removido!",
FacesMessage.SEVERITY_INFO);
                        parametros = dao.listaTodos();
                } catch (Exception ex) {
                        addMessage("Houve um erro ao tentar remover o parametro!",
                                        FacesMessage.SEVERITY_ERROR);
                }
        }

        public List<Parametro> getParametros() {

                if (parametros == null)
                        parametros = dao.listaTodos();

                return parametros;
        }

        public Parametro getParametro() {
                return parametro;
        }

        public void setParametro(Parametro parametro) {
                this.parametro = parametro;
        }

        public void addMessage(String summary, Severity severity) {
                FacesMessage message = new FacesMessage(severity, summary, null);
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import br.com.unip.aps.ws.model.Parametro;

public class ParametroDao extends Dao<Parametro> implements Serializable {

        /**
         *
         */
        private static final long serialVersionUID = 1320201162555558845L;

        public ParametroDao() {
                super(Parametro.class);
        }

        public Parametro buscaPorChave(String chave) {

                String query = "select * from parametro p where p.chave=:chave";

                List<Parametro> lista = em.createQuery(query, Parametro.class)
                                .setParameter("chave", chave).getResultList();

                return lista == null ? null : lista.get(0);
        }

}
```

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
      http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
        version="2.0">

        <persistence-unit name="aps-web-service"
                transaction-type="RESOURCE_LOCAL">

                <!-- implementacao do JPA -->
                <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

                <properties>
                        <property name="javax.persistence.jdbc.driver"
value="org.postgresql.Driver" />
                        <property name="javax.persistence.jdbc.url"
                                value="jdbc:postgresql://localhost:5432/aps-web-service" />
                        <property name="javax.persistence.jdbc.user" value="usuario" />
                        <property name="javax.persistence.jdbc.password" value="senha" />
                        <property name="javax.persistence.validation.mode" value="none" />
                        <property name="hibernate.hbm2ddl.auto" value="create-drop" />
                        <property name="hibernate.show_sql" value="true" />
                        <property name="hibernate.format_sql" value="true" />
                        <property name="hibernate.transaction.factory_class"
                                value="org.hibernate.transaction.JDBCTransactionFactory" />
                        <property name="hibernate.current_session_context_class"
                                value="thread" />

                </properties>
        </persistence-unit>
</persistence>package br.com.unip.aps.ws.model;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Type;
import org.hibernate.validator.constraints.NotEmpty;
import org.joda.time.DateTime;

import br.com.unip.aps.ws.adapter.DateTimeAdapter;
```

```
@XmlRootElement(name = "problema")
@XmlAccessorType(XmlAccessType.NONE)
@Entity
@Table
public class Problema implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -3191002908800479635L;

	@XmlElement(name = "id", required = true)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "problema_id", nullable = false, insertable = true, updatable = false,
unique = true)
	private Long id;

	@XmlElement(name = "nome", required = true)
	@NotEmpty
	@Size(min = 2, max = 80, message = "*O tamanho requerido é de 2 ao 80.")
	@Column(length = 80, nullable = false, insertable = true, updatable = true, unique =
false)
	private String nome;

	@XmlElement(name = "descricao", required = true)
	@NotEmpty
	@Size(min = 10, max = 150, message = "*O tamanho requerido é de 10 ao 150.")
	@Column(length = 150, nullable = false, insertable = true, updatable = true, unique =
false)
	private String descricao;

	@XmlElement(name = "dataCriacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = false, insertable = true, updatable = false, unique = false)
	private DateTime dataCriacao;

	@XmlElement(name = "dataModificacao", required = true)
	@XmlJavaTypeAdapter(DateTimeAdapter.class)
	@Temporal(TemporalType.TIMESTAMP)
	@Type(type = "org.jadira.usertype.dateandtime.joda.PersistentDateTime")
	@Column(nullable = true, insertable = false, updatable = true, unique = false)
	private DateTime dataModificacao;

	@XmlTransient
	@OneToOne(optional = false, fetch = FetchType.LAZY)
	@JoinColumn(name = "ocorrencia_id", nullable = false, insertable = true, updatable =
false, unique = false)
	private Ocorrencia ocorrencia = new Ocorrencia();

	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;
	}

	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}

	public DateTime getDataCriacao() {
		return dataCriacao;
	}

	public void setDataCriacao(DateTime dataCriacao) {
		this.dataCriacao = dataCriacao;
	}

	public DateTime getDataModificacao() {
		return dataModificacao;
	}

	public void setDataModificacao(DateTime dataModificacao) {
		this.dataModificacao = dataModificacao;
	}

	public Ocorrencia getOcorrencia() {
		return ocorrencia;
	}

	public void setOcorrencia(Ocorrencia ocorrencia) {
		this.ocorrencia = ocorrencia;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((dataCriacao == null) ? 0 : dataCriacao.hashCode());
		result = prime * result
				+ ((dataModificacao == null) ? 0 :
dataModificacao.hashCode());
		result = prime * result
				+ ((descricao == null) ? 0 : descricao.hashCode());
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((nome == null) ? 0 : nome.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
```

```
                return true;
            if (obj == null)
                return false;
            if (getClass() != obj.getClass())
                return false;
            Problema other = (Problema) obj;
            if (dataCriacao == null) {
                if (other.dataCriacao != null)
                    return false;
            } else if (!dataCriacao.equals(other.dataCriacao))
                return false;
            if (dataModificacao == null) {
                if (other.dataModificacao != null)
                    return false;
            } else if (!dataModificacao.equals(other.dataModificacao))
                return false;
            if (descricao == null) {
                if (other.descricao != null)
                    return false;
            } else if (!descricao.equals(other.descricao))
                return false;
            if (id == null) {
                if (other.id != null)
                    return false;
            } else if (!id.equals(other.id))
                return false;
            if (nome == null) {
                if (other.nome != null)
                    return false;
            } else if (!nome.equals(other.nome))
                return false;
            return true;
        }

        @PrePersist
        public void prePersist() {
            dataCriacao = DateTime.now();
        }

        @PreUpdate
        public void preUpdate() {
            dataModificacao = DateTime.now();
        }

}
package br.com.unip.aps.ws.mb;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

import br.com.unip.aps.ws.dao.ProblemaDao;
import br.com.unip.aps.ws.model.Problema;

@ManagedBean
```

```
@SessionScoped
public class ProblemaBean {

	@Inject
	private ProblemaDao dao;
	private Problema problema = new Problema();
	private List<Problema> problemas;

	public void grava() {

		try {
			if (problema.getId() == null)
				dao.adiciona(problema);
			else
				dao.atualiza(problema);

			addMessage(problema.getNome() + " gravado!",
					FacesMessage.SEVERITY_INFO);

			problema = new Problema();
			problemas = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar gravar o problema!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public void remove(Problema problema) {
		try {
			dao.remove(problema);
			addMessage(problema.getNome() + " removido!",
					FacesMessage.SEVERITY_INFO);
			problemas = dao.listaTodos();
		} catch (Exception ex) {
			addMessage("Houve um erro ao tentar remover o problema!",
					FacesMessage.SEVERITY_ERROR);
		}
	}

	public List<Problema> getProblemas() {

		if (problemas == null)
			problemas = dao.listaTodos();

		return problemas;
	}

	public Problema getProblema() {
		return problema;
	}

	public void setProblema(Problema problema) {
		this.problema = problema;
	}

	public void addMessage(String summary, Severity severity) {
		FacesMessage message = new FacesMessage(severity, summary, null);
		FacesContext.getCurrentInstance().addMessage(null, message);
	}
```

```
}
package br.com.unip.aps.ws.dao;

import java.io.Serializable;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Restrictions;

import br.com.unip.aps.ws.model.Problema;

public class ProblemaDao extends Dao<Problema> implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -3063480185278011735L;

	public ProblemaDao() {
		super(Problema.class);
	}

	public Problema buscaPorIdOcorrencia(Long id) {
		Session session = (Session) em.getDelegate();
		Criteria criteria = session.createCriteria(Problema.class, "p")
				.createAlias("p.ocorrencia", "o")
				.add(Restrictions.eq("o.id", id)).setMaxResults(1)
				.setFetchSize(1);
		return (Problema) criteria.uniqueResult();
	}

	public Problema buscaPorNome(String nome) {

		String query = "select * from problema p where p.nome=:nome";

		List<Problema> lista = em.createQuery(query, Problema.class)
				.setParameter("nome", nome).getResultList();

		return lista == null ? null : lista.get(0);
	}

}
package br.com.unip.aps.ws.rest;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.unip.aps.ws.dao.ProblemaDao;
import br.com.unip.aps.ws.model.Problema;

@Path("/problema")
public class ProblemaRest {
```

```
	@Inject
	private ProblemaDao dao;

	@GET
	@Produces(MediaType.APPLICATION_XML)
	public List<Problema> getProblemas() {
		return dao.listaTodos();
	}

	@GET
	@Path("/{id}")
	@Produces(MediaType.APPLICATION_XML)
	public Problema getProblema(@PathParam("id") long id) {
		return dao.buscaPorIdOcorrencia(id);
	}

}package br.com.unip.aps.ws.tx;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

import javax.interceptor.InterceptorBinding;

@InterceptorBinding
@Target({ElementType.METHOD, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface Transactional {

}
package br.com.unip.aps.ws.tx;

import java.io.Serializable;

import javax.inject.Inject;
import javax.interceptor.AroundInvoke;
import javax.interceptor.Interceptor;
import javax.interceptor.InvocationContext;
import javax.persistence.EntityManager;

@Interceptor
@Transactional
public class TransactionInterceptor implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -4645152753746946317L;

	@Inject
	private EntityManager manager;

	@AroundInvoke
	public Object intercept(InvocationContext ctx) throws Exception {

		try {
			System.out.println("Abrindo a transacao");
			manager.getTransaction().begin();
```

```
                    // Passando para o JSF tratar a requisicao
                    // e pegando o retorno da logica
                    Object resultado = ctx.proceed();

                    System.out.println("comitando a transacao");
                    manager.getTransaction().commit();

                    return resultado;
                } catch (Exception ex) {

                    ex.printStackTrace();

                    if (manager.getTransaction().isActive()) {
                        System.out.println("voltando a transacao");
                        manager.getTransaction().rollback();
                    } else {
                        System.out.println("Houve um erro no Transaction Interceptor");
                    }

                    throw ex;
                }
            }

        }
```

Anexo B: Código fonte da aplicação cliente utilizando Web Service

```
package br.com.unip.aps.wsc.converter;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

import com.thoughtworks.xstream.converters.Converter;
import com.thoughtworks.xstream.converters.MarshallingContext;
import com.thoughtworks.xstream.converters.UnmarshallingContext;
import com.thoughtworks.xstream.io.HierarchicalStreamReader;
import com.thoughtworks.xstream.io.HierarchicalStreamWriter;

public class CalendarConverter implements Converter {

	private SimpleDateFormat formatador = new SimpleDateFormat(
			"yyyy-MM-dd-hh:mm");

	public boolean canConvert(final Class clazz) {
		return Calendar.class.isAssignableFrom(clazz);
	}

	public void marshal(final Object valor,
			final HierarchicalStreamWriter escritor,
			final MarshallingContext context) {
		Date data = ((Calendar) valor).getTime();
		escritor.setValue(formatador.format(data));
	}

	public Object unmarshal(final HierarchicalStreamReader leitor,
			final UnmarshallingContext contexto) {
		GregorianCalendar calendar = new GregorianCalendar();
		try {
			calendar.setTime(formatador.parse(leitor.getValue()));
		} catch (final Exception ex) {
			throw new RuntimeException(String.format(
					"Erro ao tentar deserializar objetos Calendar: %s",
					contexto.getRequiredType().getSimpleName()), ex);
		}
		return calendar;
	}
}package br.com.unip.aps.wsc.rest;

import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;

public class ConsumidorRest<T> {

	private final DadosRest dadosRest;
	private HttpURLConnection conexaoHTTP = null;

	public ConsumidorRest(Class<T> clazz) {
		dadosRest = DadosRest.valueOf(clazz.getSimpleName().toUpperCase());
	}
```

```
public T getConteudoEmObjeto(Long id) throws NullPointerException,
                IllegalArgumentException, ClassCastException, IOException {
        try {
                LeitorXML<T> leitorXML = new LeitorXML<T>(dadosRest);
                URL url = new URL(dadosRest.getURLObjeto(id));
                T objeto = leitorXML.carregaObjeto(getConteudo(url));
                return objeto;
        } catch (IOException ex) {
                throw new IOException();
        } catch (NullPointerException | IllegalArgumentException
                        | ClassCastException ex) {
                throw new RuntimeException(ex);
        } finally {
                conexaoHTTP.disconnect();
        }
}

public List<T> getConteudoEmLista() throws NullPointerException,
                IllegalArgumentException, ClassCastException, IOException {
        try {
                LeitorXML<T> leitorXML = new LeitorXML<T>(dadosRest);
                URL url = new URL(dadosRest.getURLLista());
                List<T> lista = leitorXML.carregaLista(getConteudo(url));
                return lista;
        } catch (IOException ex) {
                throw new IOException();
        } catch (NullPointerException | IllegalArgumentException
                        | ClassCastException ex) {
                throw new RuntimeException(ex);
        } finally {
                conexaoHTTP.disconnect();
        }
}

public List<T> getConteudoEmListaComPesquisa(String pesquisa)
                throws NullPointerException, IllegalArgumentException,
                ClassCastException, IOException {
        try {
                LeitorXML<T> leitorXML = new LeitorXML<T>(dadosRest);
                URL url = new URL(dadosRest.getURLListaPesquisa(pesquisa));
                List<T> lista = leitorXML.carregaLista(getConteudo(url));
                return lista;
        } catch (IOException ex) {
                throw new IOException();
        } catch (NullPointerException | IllegalArgumentException
                        | ClassCastException ex) {
                throw new RuntimeException(ex);
        } finally {
                conexaoHTTP.disconnect();
        }
}

private InputStream getConteudo(URL url) throws IOException {

        conexaoHTTP = (HttpURLConnection) url.openConnection();
        InputStream conteudo = conexaoHTTP.getInputStream();

        return conteudo;
}
```

```
}
package br.com.unip.aps.wsc.controller;

import java.io.IOException;
import java.util.List;

import javax.swing.JOptionPane;

import br.com.unip.aps.ws.model.Cidade;
import br.com.unip.aps.ws.model.Contato;
import br.com.unip.aps.ws.model.Endereco;
import br.com.unip.aps.ws.model.Ocorrencia;
import br.com.unip.aps.ws.model.Problema;
import br.com.unip.aps.wsc.rest.ConsumidorRest;
import br.com.unip.aps.wsc.view.PainelInicial;
import br.com.unip.aps.wsc.view.PainelOcorrencias;
import br.com.unip.aps.wsc.view.PainelVisualizadorOcorrencia;
import br.com.unip.aps.wsc.view.TelaPrincipal;

public class Controlador {

	private TelaPrincipal telaPrincipal;
	private PainelInicial painelInicial;
	private PainelOcorrencias painelOcorrencias;
	private PainelVisualizadorOcorrencia painelVisualizadorOcorrencia;
	private Ocorrencia ocorrencia;

	public Controlador(TelaPrincipal telaPrincipal) {
		this.telaPrincipal = telaPrincipal;
		painelInicial = new PainelInicial(this);
		telaPrincipal.setPainelPrincipal(painelInicial);
	}

	public List<Ocorrencia> consultaOcorrencias(String pesquisa)
			throws NullPointerException, IllegalArgumentException,
			ClassCastException, IOException {
		ConsumidorRest<Ocorrencia> cr = new ConsumidorRest<Ocorrencia>(
				Ocorrencia.class);
		List<Ocorrencia> ocorrencias;
		if (pesquisa == null || pesquisa.isEmpty())
			ocorrencias = cr.getConteudoEmLista();
		else
			ocorrencias = cr.getConteudoEmListaComPesquisa(pesquisa);
		return ocorrencias;
	}

	public void carregaTabelaOcorrencias(String pesquisa) {
		try {
			List<Ocorrencia> ocorrencias = consultaOcorrencias(pesquisa);
			painelOcorrencias.carregaTabela(ocorrencias);
			painelOcorrencias.atualizaTotalResultados();
			JOptionPane.showMessageDialog(painelOcorrencias,
					"Consulta finalizada!", "Ocorrências",
					JOptionPane.INFORMATION_MESSAGE);
		} catch (IOException ex) {
			JOptionPane
					.showMessageDialog(
							painelOcorrencias,
```

```
                                                        "Houve uma falha na comunicação com
o web service de ocorrências",
                                                        "Ocorrências",
JOptionPane.ERROR_MESSAGE);
                        ex.printStackTrace();
                } catch (Exception ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
                                                        "Houve um erro ao tentar iniciar a
consulta das ocorrências",
                                                        "Ocorrências",
JOptionPane.WARNING_MESSAGE);
                        ex.printStackTrace();
                }
        }

        public Problema consultaProblema() {
                if (ocorrencia == null)
                        return null;
                ConsumidorRest<Problema> cr = new ConsumidorRest<Problema>(
                                Problema.class);
                Problema problema = null;
                try {
                        problema = cr.getConteudoEmObjeto(ocorrencia.getId());
                } catch (NullPointerException | IllegalArgumentException
                                | ClassCastException ex) {
                        JOptionPane.showMessageDialog(painelOcorrencias,
                                        "Houve um erro ao tentar iniciar a consulta do
problema",
                                        "Problema da Ocorrência",
JOptionPane.WARNING_MESSAGE);
                        throw new RuntimeException(ex);
                } catch (IOException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
                                                        "Houve uma falha na comunicação com
o web service do problema da ocorrência",
                                                        "Problema da Ocorrência",
JOptionPane.ERROR_MESSAGE);
                        throw new RuntimeException(ex);
                }
                return problema;
        }

        public Contato consultaContato() {
                if (ocorrencia == null)
                        return null;
                ConsumidorRest<Contato> cr = new
ConsumidorRest<Contato>(Contato.class);
                Contato contato = null;
                try {
                        contato = cr.getConteudoEmObjeto(ocorrencia.getId());
                } catch (NullPointerException | IllegalArgumentException
                                | ClassCastException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
```

```
                                                        "Houve um erro ao tentar iniciar a
consulta do contato da ocorrência",
                                                        "Contato da Ocorrência",
                                                        JOptionPane.WARNING_MESSAGE);
                        throw new RuntimeException(ex);
                } catch (IOException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
                                                        "Houve uma falha na comunicação com
o web service do contato da ocorrência",
                                                        "Contato da Ocorrência",
JOptionPane.ERROR_MESSAGE);
                        throw new RuntimeException(ex);
                }
                return contato;
        }

        public Endereco consultaEndereco() {
                if (ocorrencia == null)
                        return null;
                ConsumidorRest<Endereco> cr = new ConsumidorRest<Endereco>(
                                Endereco.class);
                Endereco endereco = null;
                try {
                        endereco = cr.getConteudoEmObjeto(ocorrencia.getId());
                } catch (NullPointerException | IllegalArgumentException
                                | ClassCastException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
                                                        "Houve um erro ao tentar iniciar a
consulta do endereço da ocorrência",
                                                        "Endereço da Ocorrência",
                                                        JOptionPane.WARNING_MESSAGE);
                        throw new RuntimeException(ex);
                } catch (IOException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
                                                        "Houve uma falha na comunicação com
o web service do endereço da ocorrência",
                                                        "Endereço da Ocorrência",
JOptionPane.ERROR_MESSAGE);
                        throw new RuntimeException(ex);
                }
                return endereco;
        }

        public List<Cidade> consultaCidades() {
                ConsumidorRest<Cidade> cr = new ConsumidorRest<Cidade>(Cidade.class);
                List<Cidade> cidades = null;
                try {
                        cidades = cr.getConteudoEmLista();
                } catch (NullPointerException | IllegalArgumentException
                                | ClassCastException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
```

```
                                                        "Houve um erro ao tentar iniciar a
consulta de cidades para pesquisas",
                                                        "Endereço da Ocorrência",
                                                        JOptionPane.WARNING_MESSAGE);
                        throw new RuntimeException(ex);
                } catch (IOException ex) {
                        JOptionPane
                                        .showMessageDialog(
                                                        painelOcorrencias,
                                                        "Houve uma falha na comunicação com
o web service de cidades para pesquisas",
                                                        "Endereço da Ocorrência",
JOptionPane.ERROR_MESSAGE);
                        throw new RuntimeException(ex);
                }
                return cidades;
        }

        public void visualizaOcorrencia(Ocorrencia ocorrencia) {
                this.ocorrencia = ocorrencia;
                iniciaPainelVisualizadorOcorrencia();
        }

        public void iniciaPainelOcorrencias() {
                telaPrincipal
                                .setPainelPrincipal(painelOcorrencias = new
PainelOcorrencias(
                                                this));
        }

        public void iniciaPainelVisualizadorOcorrencia() {
                telaPrincipal
                                .setPainelPrincipal(painelVisualizadorOcorrencia = new
PainelVisualizadorOcorrencia(
                                                this));
        }

        public void iniciaPainelInicial() {
                telaPrincipal
                                .setPainelPrincipal(painelInicial = new PainelInicial(this));
        }

        public void sair() {
                System.exit(0);
        }

        public TelaPrincipal getTelaPrincipal() {
                return telaPrincipal;
        }

        public PainelOcorrencias getPainelOcorrencias() {
                return painelOcorrencias;
        }

        public PainelVisualizadorOcorrencia getPainelVisualizadorOcorrencia() {
                return painelVisualizadorOcorrencia;
        }

        public PainelInicial getPainelInicial() {
                return painelInicial;
```

```
	}

	public Ocorrencia getOcorrencia() {
		return ocorrencia;
	}

}
package br.com.unip.aps.wsc.rest;

import br.com.unip.aps.ws.model.Cidade;
import br.com.unip.aps.ws.model.Contato;
import br.com.unip.aps.ws.model.Endereco;
import br.com.unip.aps.ws.model.Ocorrencia;
import br.com.unip.aps.ws.model.Problema;

public enum DadosRest implements DadosRestImpl {

	OCORRENCIA {

		@Override
		public String getURLObjeto(Long id) {
			return "http://localhost:8080/rest/ocorrencia/" + id;
		}

		@Override
		public String getURLLista() {
			return "http://localhost:8080/rest/ocorrencia";
		}

		@Override
		public String getURLListaPesquisa(String pesquisa) {
			return "http://localhost:8080/rest/ocorrencia/" + pesquisa;
		}

		@Override
		public String getAlias() {
			return "ocorrencia";
		}

		@Override
		public String getAliasLista() {
			return "ocorrencias";
		}

		@Override
		public Class<?> getClazz() {
			return Ocorrencia.class;
		}

	},

	PROBLEMA {

		@Override
		public String getURLObjeto(Long id) {
			return "http://localhost:8080/rest/problema/" + id;
		}

		@Override
		public String getURLLista() {
```

```
                return "http://localhost:8080/rest/problema";
        }

        @Override
        public String getURLListaPesquisa(String pesquisa) {
                return "http://localhost:8080/rest/problema/" + pesquisa;
        }

        @Override
        public String getAlias() {
                return "problema";
        }

        @Override
        public String getAliasLista() {
                return "problemas";
        }

        @Override
        public Class<?> getClazz() {
                return Problema.class;
        }

},

CONTATO {

        @Override
        public String getURLObjeto(Long id) {
                return "http://localhost:8080/rest/contato/" + id;
        }

        @Override
        public String getURLLista() {
                return "http://localhost:8080/rest/contato";
        }

        @Override
        public String getURLListaPesquisa(String pesquisa) {
                return "http://localhost:8080/rest/contato/" + pesquisa;
        }

        @Override
        public String getAlias() {
                return "contato";
        }

        @Override
        public String getAliasLista() {
                return "contatos";
        }

        @Override
        public Class<?> getClazz() {
                return Contato.class;
        }

},

ENDERECO {
```

```
        @Override
        public String getURLObjeto(Long id) {
                return "http://localhost:8080/rest/endereco/" + id;
        }

        @Override
        public String getURLLista() {
                return "http://localhost:8080/rest/endereco";
        }

        @Override
        public String getURLListaPesquisa(String pesquisa) {
                return "http://localhost:8080/rest/endereco/" + pesquisa;
        }

        @Override
        public String getAlias() {
                return "endereco";
        }

        @Override
        public String getAliasLista() {
                return "enderecos";
        }

        @Override
        public Class<?> getClazz() {
                return Endereco.class;
        }

},

CIDADE {

        @Override
        public String getURLObjeto(Long id) {
                return "http://localhost:8080/rest/cidade/" + id;
        }

        @Override
        public String getURLLista() {
                return "http://localhost:8080/rest/cidade";
        }

        @Override
        public String getURLListaPesquisa(String pesquisa) {
                return "http://localhost:8080/rest/cidade/" + pesquisa;
        }

        @Override
        public String getAlias() {
                return "cidade";
        }

        @Override
        public String getAliasLista() {
                return "cidades";
        }
```

```
		@Override
		public Class<?> getClazz() {
			return Cidade.class;
		}

	};

}
package br.com.unip.aps.wsc.rest;

public interface DadosRestImpl {

	public String getURLObjeto(Long id);

	public String getURLListaPesquisa(String pesquisa);

	public String getURLLista();

	public String getAliasLista();

	public String getAlias();

	public Class<?> getClazz();

}
package br.com.unip.aps.wsc.converter;

import java.lang.reflect.Constructor;

import org.joda.time.DateTime;

import com.thoughtworks.xstream.converters.Converter;
import com.thoughtworks.xstream.converters.MarshallingContext;
import com.thoughtworks.xstream.converters.UnmarshallingContext;
import com.thoughtworks.xstream.io.HierarchicalStreamReader;
import com.thoughtworks.xstream.io.HierarchicalStreamWriter;

public final class DateTimeConverter implements Converter {

	@Override
	public boolean canConvert(final Class clazz) {
		return (clazz != null)
				&& DateTime.class.getPackage().equals(clazz.getPackage());
	}

	@Override
	public void marshal(final Object valor,
			final HierarchicalStreamWriter escritor,
			final MarshallingContext context) {
		escritor.setValue(valor.toString());
	}

	@Override
	public Object unmarshal(final HierarchicalStreamReader leitor,
			final UnmarshallingContext contexto) {
		try {
			final Class<?> requiredType = contexto.getRequiredType();
			final Constructor<?> constructor = requiredType
					.getConstructor(Object.class);
			return constructor.newInstance(leitor.getValue());
```

```
                } catch (final Exception ex) {
                        throw new RuntimeException(String.format(
                                        "Erro ao tentar deserializar objetos Joda Time: %s",
                                        contexto.getRequiredType().getSimpleName()), ex);
                }
        }

}package br.com.unip.aps.wsc.rest;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import br.com.unip.aps.wsc.converter.DateTimeConverter;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class LeitorXML<T> {

        private final DadosRest dadosRest;

        public LeitorXML(DadosRest dados) {
                this.dadosRest = dados;
        }

        public List<T> carregaLista(InputStream inputStream) {

                XStream stream = getXStream();
                stream.alias(dadosRest.getAliasLista(), new ArrayList<T>().getClass());

                @SuppressWarnings("unchecked")
                List<T> lista = (List<T>) stream.fromXML(inputStream);
                return lista;
        }

        public T carregaObjeto(InputStream inputStream) {

                XStream stream = getXStream();

                @SuppressWarnings("unchecked")
                T objeto = (T) stream.fromXML(inputStream);
                return objeto;
        }

        private XStream getXStream() {
                XStream stream = new XStream(new DomDriver());
                stream.registerConverter(new DateTimeConverter());
                stream.alias(dadosRest.getAlias(), dadosRest.getClazz());
                return stream;
        }

}
package br.com.unip.aps.wsc.main;

import javax.swing.SwingUtilities;

import br.com.unip.aps.wsc.view.TelaPrincipal;

public class Main {
```

```
	public static void main(String[] args) {
		SwingUtilities.invokeLater(new Runnable() {
			@Override
			public void run() {
				new TelaPrincipal();
			}
		});
	}

}
package br.com.unip.as.wsc.adapter;

import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.JPopupMenu;
import javax.swing.JTable;
import javax.swing.SwingUtilities;

public class MouseTableAdapter {

	public MouseAdapter getAction(final JTable tb, final JPopupMenu popupMenu) {

		MouseAdapter action = new MouseAdapter() {

			public void mouseReleased(MouseEvent e) {
				showPopup(e);
			}

			private void showPopup(MouseEvent e) {

				if (SwingUtilities.isRightMouseButton(e)) {
					try {
						Point p = e.getPoint();
						int rowNumber = tb.rowAtPoint(p);
						if (rowNumber >= 0) {
							tb.setRowSelectionInterval(rowNumber,
rowNumber);
							popupMenu

	.show(e.getComponent(), e.getX(), e.getY());
						}
					} catch (Exception ex) {
						ex.printStackTrace();
					}

				}
			}
		};
		return action;
	}
}
package br.com.unip.aps.wsc.model;

import java.util.List;

import javax.swing.table.AbstractTableModel;
```

```
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;

import br.com.unip.aps.ws.model.Ocorrencia;

public class OcorrenciaTableModel extends AbstractTableModel {

	/**
	 *
	 */
	private static final long serialVersionUID = -6851534836762748785L;

	DateTimeFormatter formatador = DateTimeFormat
			.forPattern("dd 'de' MMMM 'de' yyyy");
	private String[] columnNames = { "Ocorrência", "Data" };
	private List<Ocorrencia> data;

	public OcorrenciaTableModel(List<Ocorrencia> data) {
		this.data = data;
	}

	public int getColumnCount() {
		return columnNames.length;
	}

	public int getRowCount() {
		return data.size();
	}

	public String getColumnName(int col) {
		return columnNames[col];
	}

	public Object getValueAt(int row, int col) {
		switch (col) {
		// case 0: return data.get(row).getId();
		case 0:return data.get(row).getNome();
		case 1:return formatador.print(data.get(row).getDataCriacao());
		}
		return null;
	}

	public Ocorrencia getClasseAt(int row) {
		return data.get(row);
	}

	public synchronized Class<?> getColumnClass(int col) {
		try {
			return getValueAt(0, col).getClass();
		} catch (NullPointerException e) {
		}
		return java.util.Date.class;
	}

	public boolean isCellEditable(int row, int col) {

		if (col == 3 || col == 4 || col == 5) {
			return true;
		} else {
```

```
                        return false;
                }
        }

        public void setValueAt(Object value, int row, int col) {
                try {
                        switch (col) {
                        // case 0: data.get(row).setId( (long) value);break;
                        case 0:data.get(row).setNome((String) value);break;
                        case 1:data.get(row).setDataCriacao((DateTime) value);break;
                        }
                        this.fireTableRowsUpdated(row, row);
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        public void addLista(List<Ocorrencia> dados) {
                int oldSize = getRowCount();
                data.addAll(dados);
                int newSize = getRowCount() - 1;
                fireTableRowsInserted(oldSize, newSize);
        }

        public long getIdAt(int row) {
                return data.get(row).getId();
        }

        public void setIdAt(int row, long id) {
                data.get(row).setId(id);
        }

        public void remove(int row) {
                data.remove(row);
                fireTableRowsDeleted(row, row);
        }

        public void add(Ocorrencia conta) {
                data.add(conta);
                int lastIndex = getRowCount() - 1;
                fireTableRowsInserted(lastIndex, lastIndex);
        }

        public void limpar() {
                data.clear();
                fireTableDataChanged();
        }

        public boolean isEmpty() {
                return data.isEmpty();
        }
}
package br.com.unip.aps.wsc.view;

import java.awt.Color;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
```

```
import br.com.unip.aps.ws.model.Contato;
import br.com.unip.aps.wsc.controller.Controlador;

public class PainelContato extends JPanel implements PainelOcorrenciaImpl {

	/**
	 *
	 */
	private static final long serialVersionUID = 470985078583024164L;

	private JPanel painelTelefone, painelCelular, painelEmail;
	private JLabel labTelefone, labCelular, labEmail;
	private JTextField txtTelefone, txtCelular, txtEmail;
	private Contato contato;
	private Controlador controlador;

	public PainelContato(Controlador controlador) {
		this.controlador = controlador;
		carregaComponentes();
		customizaComponentes();
		adicionaComponentes();
	}

	private void carregaComponentes() {
		labTelefone = new JLabel("Telefone:");
		labCelular = new JLabel("Celular:  ");
		labEmail = new JLabel("E-mail:   ");
		txtTelefone = new JTextField(40);
		txtCelular = new JTextField(40);
		txtEmail = new JTextField(40);
		painelTelefone = new JPanel();
		painelCelular = new JPanel();
		painelEmail = new JPanel();
	}

	private void adicionaComponentes() {
		painelTelefone.add(labTelefone);
		painelTelefone.add(txtTelefone);
		painelCelular.add(labCelular);
		painelCelular.add(txtCelular);
		painelEmail.add(labEmail);
		painelEmail.add(txtEmail);
		add(painelTelefone);
		add(painelCelular);
		add(painelEmail);
	}

	private void customizaComponentes() {
		txtTelefone.setBackground(Color.white);
		txtCelular.setBackground(Color.white);
		txtEmail.setBackground(Color.white);
		txtTelefone.setEditable(false);
		txtCelular.setEditable(false);
		txtEmail.setEditable(false);
	}

	@Override
	public void preencheCampos() {
		if (contato == null)
			contato = controlador.consultaContato();
```

```
                txtTelefone.setText(contato.getTelefone());
                txtCelular.setText(contato.getCelular());
                txtEmail.setText(contato.getEmail());
        }

        public String getTitulo() {
                return "Contato";
        }

}
package br.com.unip.aps.wsc.view;

import java.awt.Color;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import br.com.unip.aps.ws.model.Endereco;
import br.com.unip.aps.wsc.controller.Controlador;

public class PainelEndereco extends JPanel implements PainelOcorrenciaImpl {

        /**
         *
         */
        private static final long serialVersionUID = 4709850785830241 64L;

        private JPanel painelPais, painelEstado, painelCidade, painelBairro,
                        painelLogradouro, painelTipo;
        private JLabel labPais, labEstado, labCidade, labBairro, labLogradouro,
                        labTipo;
        private JTextField txtPais, txtEstado, txtCidade, txtBairro, txtLogradouro,
                        txtTipo;
        private Endereco endereco;
        private Controlador controlador;

        public PainelEndereco(Controlador controlador) {
                this.controlador = controlador;
                carregaComponentes();
                customizaComponentes();
                adicionaComponentes();
        }

        private void carregaComponentes() {
                labPais = new JLabel("País:          ");
                labEstado = new JLabel("Estado:       ");
                labCidade = new JLabel("Cidade:       ");
                labBairro = new JLabel("Bairro:       ");
                labLogradouro = new JLabel("Logradouro:");
                labTipo = new JLabel("Tipo:           ");
                txtPais = new JTextField(40);
                txtEstado = new JTextField(40);
                txtCidade = new JTextField(40);
                txtBairro = new JTextField(40);
                txtLogradouro = new JTextField(40);
                txtTipo = new JTextField(40);
                painelPais = new JPanel();
                painelEstado = new JPanel();
                painelCidade = new JPanel();
```

```
		painelBairro = new JPanel();
		painelLogradouro = new JPanel();
		painelTipo = new JPanel();
	}

	private void adicionaComponentes() {
		painelPais.add(labPais);
		painelPais.add(txtPais);
		painelEstado.add(labEstado);
		painelEstado.add(txtEstado);
		painelCidade.add(labCidade);
		painelCidade.add(txtCidade);
		painelBairro.add(labBairro);
		painelBairro.add(txtBairro);
		painelLogradouro.add(labLogradouro);
		painelLogradouro.add(txtLogradouro);
		painelTipo.add(labTipo);
		painelTipo.add(txtTipo);
		add(painelPais);
		add(painelEstado);
		add(painelCidade);
		add(painelBairro);
		add(painelLogradouro);
		add(painelTipo);
	}

	private void customizaComponentes() {
		txtPais.setBackground(Color.white);
		txtEstado.setBackground(Color.white);
		txtCidade.setBackground(Color.white);
		txtBairro.setBackground(Color.white);
		txtLogradouro.setBackground(Color.white);
		txtTipo.setBackground(Color.white);
		txtPais.setEditable(false);
		txtEstado.setEditable(false);
		txtCidade.setEditable(false);
		txtBairro.setEditable(false);
		txtLogradouro.setEditable(false);
		txtTipo.setEditable(false);
	}

	@Override
	public void preencheCampos() {
		if (endereco == null)
			endereco = controlador.consultaEndereco();
		txtPais.setText(endereco.getPais().getNome());
		txtEstado.setText(endereco.getEstado().getNome());
		txtCidade.setText(endereco.getCidade().getNome());
		txtBairro.setText(endereco.getBairro().getNome());
		txtLogradouro.setText(endereco.getLogradouro().getNome());
		txtTipo.setText(endereco.getLogradouro().getTipo());
	}

	public String getTitulo() {
		return "Endereço";
	}

}
package br.com.unip.aps.wsc.view;
```

```
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextPane;

import br.com.unip.aps.wsc.controller.Controlador;

public class PainelInicial extends JPanel {

	/**
	 *
	 */
	private static final long serialVersionUID = 5846952790736669985L;

	private JTextPane txtIntroducao;
	private JButton botaoIniciar;
	private Controlador controlador;
	private JPanel painelInferior;

	public PainelInicial(Controlador controlador) {
		this.controlador = controlador;
		carregaComponentes();
		customizaComponentes();
		adicionaLayouts();
		adicionaComponentes();
		adicionaComandos();
	}

	private void carregaComponentes() {
		txtIntroducao = new JTextPane();
		botaoIniciar = new JButton("Iniciar");
		painelInferior = new JPanel();
	}

	private void adicionaComponentes() {
		add(new JScrollPane(txtIntroducao));
		add(painelInferior);
		painelInferior.add(botaoIniciar);
	}

	private void customizaComponentes() {
		txtIntroducao.setFont(new Font("Serif", Font.PLAIN, 20));
		txtIntroducao.setEditable(false);
		txtIntroducao.setPreferredSize(new Dimension(0, 300));
		txtIntroducao
				.setText("UNIP - Marquês São Vicente - SP\n\n"
						+ "Projeto de Sistema Distribuidos para
Ocorrências\n\n"
						+ "Nosso objetivo é fornecer dados de
ocorrências através do protoloco RESTful que, por sua vez, utiliza o meio de comunicação
HTTP/HTTPS entre servidores heterogêneos,"
						+ " onde haja um servidor que produza dados
para seus clientes consumidores, qualquer que sejam eles.");
```

```
	}

	private void adicionaLayouts() {
		setLayout(new BoxLayout(this, BoxLayout.PAGE_AXIS));
		painelInferior.setLayout(new FlowLayout(FlowLayout.RIGHT));
	}

	private void adicionaComandos() {
		botaoIniciar.addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent evt) {
				controlador.iniciaPainelOcorrencias();
			}
		});
	}

}
package br.com.unip.aps.wsc.view;

import java.awt.Color;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import br.com.unip.aps.ws.model.Ocorrencia;
import br.com.unip.aps.wsc.controller.Controlador;

public class PainelOcorrencia extends JPanel implements PainelOcorrenciaImpl {

	/**
	 *
	 */
	private static final long serialVersionUID = 8185061952407117525L;

	private JPanel painelOcorrencia, painelDescricao;
	private JLabel labOcorrencia, labDescricao;
	private JTextField txtOcorrencia;
	private JTextArea txtDescricao;
	private Ocorrencia ocorrencia;
	private Controlador controlador;

	public PainelOcorrencia(Controlador controlador) {
		this.controlador = controlador;
		carregaComponentes();
		customizaComponentes();
		adicionaComponentes();
	}

	private void carregaComponentes() {
		labOcorrencia = new JLabel("Ocorrência:");
		labDescricao = new JLabel("Descrição: ");
		txtOcorrencia = new JTextField(40);
		txtDescricao = new JTextArea(14, 40);
		painelOcorrencia = new JPanel();
		painelDescricao = new JPanel();
	}
```

```
	private void adicionaComponentes() {
		painelOcorrencia.add(labOcorrencia);
		painelOcorrencia.add(txtOcorrencia);
		painelDescricao.add(labDescricao);
		painelDescricao.add(new JScrollPane(txtDescricao));
		add(painelOcorrencia);
		add(painelDescricao);
	}

	private void customizaComponentes() {
		txtOcorrencia.setBackground(Color.white);
		txtOcorrencia.setEditable(false);
		txtDescricao.setEditable(false);
		txtDescricao.setLineWrap(true);
		txtDescricao.setWrapStyleWord(true);
	}

	@Override
	public void preencheCampos() {
		if (ocorrencia == null)
			ocorrencia = controlador.getOcorrencia();
		txtOcorrencia.setText(ocorrencia.getNome());
		txtDescricao.setText(ocorrencia.getDescricao());
	}

	public String getTitulo() {
		return "Ocorrência";
	}

}
package br.com.unip.aps.wsc.view;

public interface PainelOcorrenciaImpl {

	void preencheCampos();

}
package br.com.unip.aps.wsc.view;

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.TableColumnModel;

import br.com.unip.aps.ws.model.Cidade;
import br.com.unip.aps.ws.model.Ocorrencia;
import br.com.unip.aps.wsc.controller.Controlador;
import br.com.unip.aps.wsc.model.OcorrenciaTableModel;
import br.com.unip.as.wsc.adapter.MouseTableAdapter;
```

```
public class PainelOcorrencias extends JPanel {

	/**
	 *
	 */
	private static final long serialVersionUID = 7675328101711477410L;

	private JTable tb;
	private JScrollPane sp;
	private JButton botaoConsultaOcorrencias;
	private JComboBox<Cidade> listaCidades;
	private JLabel labCidades, visualizadorResultados;
	private OcorrenciaTableModel tm;
	private Controlador controlador;
	private JPanel painelInferior;
	private JPopupMenu menuTabela;
	private JMenuItem itemVisualizar;

	public PainelOcorrencias(Controlador controlador) {
		this.controlador = controlador;
		carregaComponentes();
		adicionaComponentes();
		adicionaLayouts();
		customizaComponentes();
		adicionaComandos();
		preencheCampos();
	}

	private void carregaComponentes() {
		tb = new JTable();
		sp = new JScrollPane();
		botaoConsultaOcorrencias = new JButton("Consultar Ocorrências");
		listaCidades = new JComboBox<Cidade>();
		labCidades = new JLabel(" Cidades:");
		visualizadorResultados = new JLabel("Nenhum resultado de ocorrências..");
		painelInferior = new JPanel();
		menuTabela = new JPopupMenu();
		itemVisualizar = new JMenuItem("Visualizar");
	}

	private void adicionaComponentes() {
		add(sp);
		add(painelInferior);
		painelInferior.add(botaoConsultaOcorrencias);
		painelInferior.add(labCidades);
		painelInferior.add(listaCidades);
		painelInferior.add(visualizadorResultados);
		menuTabela.add(itemVisualizar);
	}

	private void customizaComponentes() {
		tb.setRowHeight(25);
		tb.setSelectionMode(0);
		tb.setAutoResizeMode(1);
		tb.setEnabled(true);
		tb.getTableHeader().setReorderingAllowed(false);
		sp.setViewportView(tb);
	}
```

```
	private void adicionaLayouts() {
		setLayout(new BoxLayout(this, BoxLayout.PAGE_AXIS));
		painelInferior.setLayout(new FlowLayout(FlowLayout.LEFT));
	}

	private void adicionaComandos() {
		botaoConsultaOcorrencias.addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent evt) {
				String pesquisa = null;
				if (listaCidades.getSelectedIndex() != -1)
					pesquisa = "cidade/"
							+
listaCidades.getSelectedItem().toString()
									.replace(" ", "%20");
				controlador.carregaTabelaOcorrencias(pesquisa);
			}
		});
		itemVisualizar.addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent evt) {
				int row = tb.getSelectedRow();
				Ocorrencia ocorrencia = tm.getClasseAt(row);
				controlador.visualizaOcorrencia(ocorrencia);
			}
		});
	}

	public void carregaTabela(List<Ocorrencia> ocorrencias) {
		tm = new OcorrenciaTableModel(ocorrencias);
		tb.setModel(tm);

		TableColumnModel tcm = tb.getColumnModel();
		tcm.getColumn(0).setPreferredWidth(400);
		tcm.getColumn(1).setPreferredWidth(80);

		tb.addMouseListener(new MouseTableAdapter().getAction(tb, menuTabela));
	}

	public OcorrenciaTableModel getTableModel() {
		return tm;
	}

	public void atualizaTotalResultados() throws NullPointerException {
		visualizadorResultados.setText(tm.getRowCount() + " resultado(s)");
	}

	public void preencheCampos() {
		List<Cidade> cidades = controlador.consultaCidades();
		listaCidades.removeAll();
		listaCidades.addItem(null);
		for (Cidade cidade : cidades)
			listaCidades.addItem(cidade);
	}

}
package br.com.unip.aps.wsc.view;

import java.awt.Color;
```

```
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import br.com.unip.aps.ws.model.Problema;
import br.com.unip.aps.wsc.controller.Controlador;

public class PainelProblema extends JPanel implements PainelOcorrenciaImpl {

	/**
	 *
	 */
	private static final long serialVersionUID = -1744447449410175680L;

	private JPanel painelOcorrencia, painelDescricao;
	private JLabel labOcorrencia, labDescricao;
	private JTextField txtProblema;
	private JTextArea txtDescricao;
	private Problema problema;
	private Controlador controlador;

	public PainelProblema(Controlador controlador) {
		this.controlador = controlador;
		carregaComponentes();
		customizaComponentes();
		adicionaComponentes();
		preencheCampos();
	}

	private void carregaComponentes() {
		labOcorrencia = new JLabel("Problema:   ");
		labDescricao = new JLabel("Descrição: ");
		txtProblema = new JTextField(40);
		txtDescricao = new JTextArea(14, 40);
		painelOcorrencia = new JPanel();
		painelDescricao = new JPanel();
	}

	private void adicionaComponentes() {
		painelOcorrencia.add(labOcorrencia);
		painelOcorrencia.add(txtProblema);
		painelDescricao.add(labDescricao);
		painelDescricao.add(new JScrollPane(txtDescricao));
		add(painelOcorrencia);
		add(painelDescricao);
	}

	private void customizaComponentes() {
		txtProblema.setBackground(Color.white);
		txtProblema.setEditable(false);
		txtDescricao.setEditable(false);
		txtDescricao.setLineWrap(true);
		txtDescricao.setWrapStyleWord(true);
	}

	@Override
	public void preencheCampos() {
		if (problema == null)
```

```
                problema = controlador.consultaProblema();
            txtProblema.setText(problema.getNome());
            txtDescricao.setText(problema.getDescricao());
        }

        public String getTitulo() {
            return "Problema";
        }

}
package br.com.unip.aps.wsc.view;

import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import br.com.unip.aps.wsc.controller.Controlador;

public class PainelVisualizadorOcorrencia extends JPanel {

        /**
         *
         */
        private static final long serialVersionUID = -8483572668516460107L;

        private Controlador controlador;
        private JPanel painelPrincipal, painelInferior;
        private JButton botaoVoltar;
        private JTabbedPane menuOcorrencia;
        private PainelOcorrencia painelOcorrencia;
        private PainelProblema painelProblema;
        private PainelContato painelContato;
        private PainelEndereco painelEndereco;

        public PainelVisualizadorOcorrencia(Controlador controlador) {
            this.controlador = controlador;
            carregaComponentes();
            adicionaComponentes();
            adicionaLayouts();
            customizaComponentes();
            adicionaComandos();
            painelOcorrencia.preencheCampos();
        }

        private void carregaComponentes() {
            botaoVoltar = new JButton("voltar");
            painelPrincipal = new JPanel();
            painelInferior = new JPanel();
            menuOcorrencia = new JTabbedPane(2);
            painelOcorrencia = new PainelOcorrencia(controlador);
            painelProblema = new PainelProblema(controlador);
            painelContato = new PainelContato(controlador);
```

```
                painelEndereco = new PainelEndereco(controlador);
        }

        private void adicionaComponentes() {
                add(painelPrincipal);
                add(painelInferior);
                menuOcorrencia.add(painelOcorrencia, painelOcorrencia.getTitulo());
                menuOcorrencia.add(painelProblema, painelProblema.getTitulo());
                menuOcorrencia.add(painelContato, painelContato.getTitulo());
                menuOcorrencia.add(painelEndereco, painelEndereco.getTitulo());
                painelPrincipal.add(menuOcorrencia);
                painelInferior.add(botaoVoltar);
        }

        private void customizaComponentes() {
                menuOcorrencia.setPreferredSize(new Dimension(650, 400));
        }

        private void adicionaLayouts() {
                setLayout(new BoxLayout(this, BoxLayout.PAGE_AXIS));
                painelInferior.setLayout(new FlowLayout(FlowLayout.LEFT));
        }

        private void adicionaComandos() {
                botaoVoltar.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent evt) {
                                controlador.iniciaPainelOcorrencias();
                        }
                });
                menuOcorrencia.addChangeListener(new ChangeListener() {
                        public void stateChanged(ChangeEvent changeEvent) {

                                JTabbedPane menuOcorencia = (JTabbedPane) changeEvent
                                                .getSource();

                                int indice = menuOcorencia.getSelectedIndex();
                                PainelOcorrenciaImpl painel = (PainelOcorrenciaImpl)
menuOcorencia
                                                .getComponentAt(indice);
                                painel.preencheCampos();
                        }
                });
        }
}
package br.com.unip.aps.wsc.view;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
```

```
import br.com.unip.aps.wsc.controller.Controlador;

public class TelaPrincipal extends JFrame {

	/**
	 *
	 */
	private static final long serialVersionUID = -3286457306857076837L;

	private JPanel painelPrincipal, painelMenu;
	private JMenuBar menu;
	private JMenu menuOpcoes;
	private JMenuItem itemInicio, itemOcorrencias, itemSair;
	private GridBagConstraints gbc;
	private Controlador controlador;

	public TelaPrincipal() {
		super("Ocorrências");
		carregaComponentes();
		adicionaLayouts();
		customizaComponentes();
		adicionaComponentes();
		adicionaComandos();
		controlador = new Controlador(this);
		setVisible(true);
	}

	private void carregaComponentes() {
		gbc = new GridBagConstraints();
		painelMenu = new JPanel();
		painelPrincipal = new JPanel();
		menu = new JMenuBar();
		menuOpcoes = new JMenu("Opções");
		itemInicio = new JMenuItem("Início");
		itemOcorrencias = new JMenuItem("Ocorrências");
		itemSair = new JMenuItem("Sair");
	}

	private void adicionaLayouts() {
		getContentPane().setLayout(new GridBagLayout());
		painelMenu.setLayout(new BorderLayout());
		painelPrincipal.setLayout(new BorderLayout());
	}

	private void customizaComponentes() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setResizable(false);
		setSize(650, 400);
		painelMenu.setMinimumSize(new Dimension(0, 20));
		itemOcorrencias.setMaximumSize(new Dimension(85, 70));
		setLocationRelativeTo(null);
	}

	private void adicionaComponentes() {

		painelMenu.add(menu, BorderLayout.CENTER);
		menu.add(menuOpcoes);
		menu.add(itemOcorrencias);
		menuOpcoes.add(itemInicio);
```

```
            menuOpcoes.add(itemSair);

            gbc.fill = GridBagConstraints.BOTH;
            gbc.anchor = GridBagConstraints.NORTHWEST;

            gbc.weightx = 2;
            gbc.weighty = 1;
            gbc.gridwidth = 2;
            gbc.gridheight = 1;
            gbc.gridx = 0;
            gbc.gridy = 0;
            gbc.insets = new Insets(0, 0, 8, 0);
            getContentPane().add(painelMenu, gbc);

            gbc.weightx = 2;
            gbc.weighty = 20;
            gbc.gridwidth = 1;
            gbc.gridheight = 1;
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.insets = new Insets(8, 8, 8, 8);
            getContentPane().add(painelPrincipal, gbc);
        }

        private void adicionaComandos() {
            itemInicio.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent evt) {
                    controlador.iniciaPainelInicial();
                }
            });
            itemOcorrencias.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent evt) {
                    controlador.iniciaPainelOcorrencias();
                }
            });
            itemSair.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent evt) {
                    controlador.sair();
                }
            });
        }

        public void setPainelPrincipal(JPanel painel) {
            try {
                painelPrincipal.remove(0);
            } catch (ArrayIndexOutOfBoundsException ex) {
            }
            painelPrincipal.add(painel, BorderLayout.CENTER);
            painelPrincipal.revalidate();
            validate();
        }

}
```

Anexo C: Fichas APS

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, Trabalhos Individuais e em grupo, práticas de ensino e outras)

Nome do aluno: MARCIO FERNANDES CRUZ Turma: CC7P13 RA: B22816-4
Curso: 15801 CIÊNCIA DA COMPUTAÇÃO Campus: MARQUÊS Semestre: 7 Turno: NOTURNO

			ASSINATURA	
ATIVIDADE	DATA	TOTAL DE HORAS	ALUNO	PROFESSOR
Reunião com grupo para iniciar projeto	08/09/2015	03:00		
Definição do programa e suas funcionalidades com grupo	09/09/2015	04:00		
Estudo sobre as tecnologia de Sistemas distribuidos	16/09/2015	03:00		
Concluindo estudos sobre as tecnologias	17/09/2015	05:00		
Desenho das classes da camada Adapter	18/09/2015	03:00		
Criação das classe da camada dao	23/09/2015	04:00		
Criação da classe fonfig	24/09/2015	06:00		
Escrita do capítulo conceitos gerais	25/09/2015	05:00		
Concluindo escrita do capítulo conceitos gerais	06/10/2015	04:00		
Escrita do capítulo elementos arquiteturiais	07/10/2015	03:00		
Concluindo escrita do capítulo elementos arquiteturiais	13/10/2015	04:00		
Revisão da escrita dos capítulos	14/10/2015	05:00		
Revisão geral do programa com o grupo	20/10/2015	03:00		
Testes e verificação do desenvolvimento com grupo	21/10/2015	05:00		
Término do desenvolvimento com grupo	01/11/2015	06:00		

TOTAL DE HORAS: 63 HORAS
Data: 09/11/2015

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, Trabalhos Individuais e em grupo, práticas de ensino e outras)

Nome do aluno: Danilo de Oliveira Dorotheu Turma: CC8P13 RA: B408FA-3
Curso: 15801 CIÊNCIA DA COMPUTAÇÃO Campus: MARQUÊS Semestre: 8 Turno: NOTURNO

			ASSINATURA	
ATIVIDADE	DATA	TOTAL DE HORAS	ALUNO	PROFESSOR
Reunião com grupo para iniciar o projeto	08/09/2015	03:00		
Definição do programa e suas funcionalidades com grupo	09/08/2015	04:00		
Estudo sobre Rest	10/08/2015	03:00		
Estudo sobre WSDL	14/08/2015	05:00		
Estudo sobre Java Transaction API	15/08/2015	03:00		
Criação das classe do pacote Rest	22/08/2015	04:00		
Criação das classe do pacote Tx	23/08/2015	06:00		
Estudo sobre Prime Faces	24/08/2015	05:00		
Estudo sobre Joda Time	02/09/2015	03:00		
Estudo sobre Hibernate	08/09/2015	03:00		
Criando e configurando Banco de Dados	14/09/2015	04:00		
Concluindo as configuraçoes do Banco de dados	15/09/2015	04:00		
Revisão geral do programa com grupo	20/09/2015	03:00		
testes e verificação do desenvolvimento com grupo	21/10/2015	04:00		
Término do desenvolvimento com grupo	01/11/2015	06:00		

TOTAL DE HORAS: 60 HORAS
Data: 09/11/2015

UNIP
UNIVERSIDADE PAULISTA

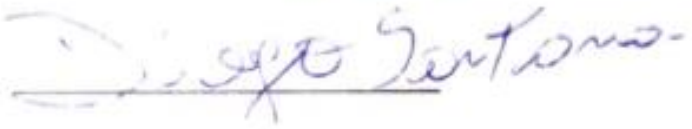
FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, Trabalhos Individuais e em grupo, práticas de ensino e outras)

Nome do aluno: Diego da Sila Santana Turma: CC7P13 RA: B56734-1
Curso: 15801 CIÊNCIA DA COMPUTAÇÃO Campus: MARQUÊS Semestre: 7 Turno: NOTURNO

			ASSINATURA	
ATIVIDADE	DATA	TOTAL DE HORAS	ALUNO	PROFESSOR
Reunião com grupo para iniciar projeto	08/09/2015	03:00		
Definição do programa e suas funcionalidades com grupo	09/08/2015	04:00		
Escrita do capítulo desenvolvimento da aplicação	15/08/2015	03:00		
Concluindo escrita desenvolvimento da aplicação	16/08/2015	05:00		
Escrita da conclusão	17/08/2015	03:00		
Escrita da introdução	22/08/2015	02:00		
Elaboração do sumário	24/08/2015	02:00		
Elaboração do objetivo do trabalho	25/08/2015	02:00		
Revisão da escrita dos capítulos	06/09/2015	04:00		
Concluindo parte escrita do trabalho	06/09/2015	03:00		
Formatando trabalho	13/09/2015	04:00		
Concluindo formatação	14/09/2015	05:00		
Revisão geral do programa com grupo	20/10/2015	03:00		
Testes e verificação do desenvolvimento com grupo	21/10/2015	05:00		
Término do desenvolvimento com grupo	01/11/2015	06:00		

TOTAL DE HORAS: 53 HORAS
Data: 09/11/2015



UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, Trabalhos Individuais e em grupo, práticas de ensino e outras)

Nome do aluno: Thiago Guy Mozol Vicente Turma: CC8P13 RA: B303GD-9
Curso: 15801 CIÊNCIA DA COMPUTAÇÃO Campus: MARQUÊS Semestre: 8 Turno: NOTURNO

			ASSINATURA	
ATIVIDADE	DATA	TOTAL DE HORAS	ALUNO	PROFESSOR
Reunião com grupo para iniciar projeto	08/09/2015	03:00		
Definição do programa e suas funcionalidades com grupo	09/09/2015	04:00		
Configurando bild Maven	16/08/2015	03:00		
Estudo sobre Web Services	17/08/2015	06:00		
Criando a classe util	18/08/2015	05:00		
Criando as classes do pacote Mb	23/08/2015	04:00		
Deselvolvendo as classes do pacote controller	24/08/2015	06:00		
Desenvolvendo as classes do pacote converter	25/08/2015	05:00		
Desenvolvendo as classes do pacote man	06/09/2015	04:00		
Desenvolvendo as classes do pacote view	07/09/2015	03:00		
Desenvolvendo as classes do pacote adapter	12/09/2015	04:00		
Revisando projeto e fazendo acertos	14/10/2015	05:00		
Revisão geral do programa com grupo	20/10/2015	03:00		
Testes e verificação do desenvolvimento com grupo	21/10/2015	05:00		
Término do desenvolvimento com grupo	01/11/2015	06:00		

TOTAL DE HORAS: 67 HORAS
Data: 09/11/2015

Thiago G M Vicente