



**MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
CAMPUS PALMAS
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET**

LEYSSON MURIEL TAVARES GUIMARÃES BARROS

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB UTILIZANDO RECURSOS DA
API JDOM PARA O GERENCIAMENTO DE DADOS XML EM JAVA**

**PALMAS
2016**

LEYSSON MURIEL TAVARES GUIMARÃES BARROS

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB UTILIZANDO RECURSOS DA
API JDOM PARA O GERENCIAMENTO DE DADOS XML EM JAVA**

Trabalho de Conclusão de Curso apresentado à
Coordenação do Curso Superior de Tecnologia em
Sistemas para Internet do Instituto Federal do
Tocantins - *Campus* Palmas, como exigência à
obtenção do grau de Tecnólogo em Sistemas para
Internet.

Orientador: Prof. MsC. Marinaldo Oliveira Santos.

**PALMAS
2016**

Barros, Leysson Muriel Tavares Guimarães
Desenvolvimento de uma aplicação web utilizando
recursos da API JDOM para o gerenciamento de dados XML em JAVA /
Leysson Muriel Tavares Guimarães Barros. – Palmas, 2016
66 f.

Trabalho de Conclusão de Curso
(Graduação em Tecnologia em Sistemas para
Internet) – Instituto Federal de Educação do
Tocantins, Campus Palmas, 2016.

Orientador(a): Prof. Marinaldo Oliveira Santos
1. XML 2. JDOM 3. JAVA I. Título

LEYSSON MURIEL TAVARES GUIMARÃES BARROS

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB UTILIZANDO RECURSOS DA
API JDOM PARA O GERENCIAMENTO DE DADOS XML EM JAVA**

Trabalho de Conclusão de Curso apresentado à
Coordenação do Curso Superior de Tecnologia em
Sistemas para Internet do Instituto Federal do
Tocantins - *Campus* Palmas, como exigência à
obtenção do grau de Tecnólogo em Sistemas para
Internet.

Aprovado em: ____/____/2016

BANCA AVALIADORA

Prof^o MsC. Marinaldo Oliveira Santos - Orientador
IFTO – *Campus* Palmas

Prof. MsC. Fernando Jorge Ebrahim Lima e Silva - Membro
IFTO – *Campus* Palmas

Prof^o MsC. Paulo da Silva Paz Neto - Membro
IFTO – *Campus* Palmas

AGRADECIMENTOS

Primeiramente quero agradecer a Deus por mais essa conquista, por toda a graça que tem feito em minha vida e ter trilhado o meu caminho, por estar sempre ao meu lado principalmente nos momentos mais difíceis é sua presença que me dá forças.

Aos meus familiares, em especial a minha esposa Rayce Cristina Monteiro Parente que sempre me deu apoio durante minha caminhada, passando carinho, incentivo, motivação e princípios fizeram com que me sentisse mais motivado para chegar ao tão sonhado momento.

Aos professores do Curso Superior de Tecnologia em Sistemas para Internet do Instituto Federal do Tocantins - Campus Palmas, com quem tive a oportunidade de aprender novos conceitos e construir novos valores, aos amigos do curso, em especial, ao professor MsC. Marinaldo Oliveira Santos por exercer tão bem o dom de repassar seu conhecimento além de ser uma referência aos alunos e pela orientação e suporte ao longo da realização deste TCC.

RESUMO

Este trabalho visa mostrar o desenvolvimento de uma aplicação web utilizando recursos da API JDOM para facilitar as operações de manipulação de dados semiestruturados em XML. A aplicação visa auxiliar no cadastro do Plano de Trabalho Docente dos professores PEBTT do IFTO e avaliar o emprego da API JDOM para manipulação de dados XML. O sistema foi desenvolvido na linguagem Java, utilizando *JavaServer Faces* (JSF) para design de páginas Web. Os dados são gerenciados por meio da API JDOM que otimiza as operações XML em Java. O trabalho ainda expõe conceitos úteis sobre a linguagem XML, a utilização do Java, o uso das APIs e especificações do sistema. A linguagem XML é uma linguagem eficiente, que permite desenvolver aplicações *Web* flexíveis e manipular os dados localmente. Enquanto que, a API JDOM permite manipular arquivo XML mais eficientemente e sem a necessidade de trabalhar com banco de dados. Dessa forma, através desse estudo, pretende-se mostrar os recursos da API JDOM para facilitar as operações de manipulação de dados semiestruturados em XML. Para trabalhos futuros propõem-se trabalhar com as classes *SAXBuilder* e o *XSL Transformation*.

Palavras-chaves: XML. JDOM. JAVA. JSF.

ABSTRACT

This work aims to demonstrate the development of a web application using the API JDOM resources, in order to facilitate the semi-structured data handling operations in XML. The purpose of the application is to help in the registration of the Plano de Trabalho Docente of the EBTT teachers of the IFTO and also to evaluate the utilization of the API JDOM in the administration of the XML data. The system was developed in the Java Programming Language, employing the JavaServer Faces (JSF) for Web pages design. The data are conducted through the API JDOM that optimizes the XML operations in Java. The project still exposes useful concepts about the XML language, the use of Java, the utilization of the APIs and the system specifications. The XML language is an efficient language that allows the development of Web flexible applications and the management of data in a local way, whereas the API JDOM facilitates the handling of XML archives more efficiently, without the need of working with databank, through this study, one intends to show the API JDOM resources to ease the operations of semi-structures data manipulation in XML. For future researches, works with SAXBuilder and the XLS Transformation are suggested.

Keywords: XML. JDOM. JAVA. JSF.

LISTA DE ILUSTRAÇÕES

Figura 1: Elementos de um arquivo xml	13
Figura 2: Ilustra uma Máquina Virtual Java (JVM).....	15
Figura 3: Arvore JDOM	18
Figura 4: Processo de entrada e saída JDOM	19
Figura 5: Exemplo de diagrama de caso de uso	22
Figura 6: Exemplo de diagrama de classe	22
Figura 7: Plano de trabalho docente – atividades	25
Figura 8: Plano de trabalho docente – extrato.....	26
Figura 9: Diagrama de caso de uso interação com o usuário	30
Figura 10: Diagrama de classes do sistema.....	31
Figura 11: Configuração do eclipse com jsf.....	32
Figura 12: Configuração do eclipse com a API JDOM	33
Figura 13: Tela de cadastro do sistema	33
Figura 14: Funcionalidade <i>tags</i> do JSF	34
Figura 15: Funcionalidade da <i>tag p:inputText</i> em JSF	35
Figura 16: Atributo <i>action</i> para executar o método.....	35
Figura 17: Elemento root da árvore JDOM.....	37
Figura 18: Elemento recebendo valor	37
Figura 19: Criação de um atributo em uma lista.....	38
Figura 20: Elementos filhos vinculados ao elemento raiz.....	38
Figura 21: Criação do arquivo XML	39
Figura 22: Pegando o elemento raiz do documento XML	39
Figura 23: Recebendo o valor dos elementos do XML.....	40
Figura 24: Acesso às lista do documento XML	40
Figura 25: Filtros usando XPath	41

LISTA DE SIGLAS

API	<i>Application Programming Interfaces</i>
CSS	<i>Cascading Style Sheets</i>
EE	<i>Enterprise Edition</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IFTO	Instituto Federal de Educação, Ciência e Tecnologia do Tocantins
IDE	Integrated Development Environment
IP	Internet Protocol
JDK	Java Development Kit
JRE	Java Runtime Environment
JSF	<i>JavaServer Faces</i>
JVM	Java Virtual Machine
LDAP	<i>Lightweight Directory Access Protocol</i>
PEBTT	Professor do Ensino Básico, Técnico e Tecnológico
SE	<i>Standard Edition</i>
TCP	<i>Transmission Control Protocol</i>
UML	<i>Unified Modeling Language</i>
XHTML	<i>eXtensible Hypertext Markup Language</i>
XML	<i>eXtensible Markup Language</i>
XPath	<i>XML Path Language</i>
XSL	<i>eXtensible Stylesheet Language</i>
XSLT	<i>eXtensible Stylesheet Language for Transformation</i>
W3C	<i>World Wide Web Consortium</i>

SUMÁRIO

1	INTRODUÇÃO.....	9
1.1	Objetivos.....	10
1.1.1	Objetivo Geral.....	10
1.1.2	Objetivos Específicos	10
1.2	Justificativa	10
2	REFERENCIAL TEÓRICO	12
2.1	A linguagem XML	12
2.1.1	Xpath	14
2.2	Linguagem de programação JAVA.....	14
2.2.1	JavaServer Faces (JSF)	16
2.3	Autenticação do usuário com LDAP	17
2.4	API JDOM.....	17
2.5	Linguagem UML	21
2.6	Processo de software	23
2.7	Plano de Trabalho docente.....	24
3	METODOLOGIA.....	28
4	ESPECIFICAÇÃO DO SOFTWARE	29
4.1	Diagrama de Caso de Uso	29
4.2	Diagrama de Classe	30
5	PROJETO E IMPLEMENTAÇÃO	32
5.1	Funcionalidades do sistema	33
5.2	Implementação do Código JSF	34
5.3	Implementação do Código JDOM.....	35
6	RESULTADOS E DISCUSSÃO.....	42
7	CONSIDERAÇÕES FINAIS.....	43
8	REFERÊNCIAS	44
9	APÊNDICE	46
10	ANEXOS.....	47

1 INTRODUÇÃO

O desenvolvimento de ferramentas que auxiliam a realização de tarefas de forma mais ágil, tornou-se uma realidade com o advento da tecnologia, pois permite o processamento de mais informações e de maneira mais precisa. Nas organizações o custo que se tem para organizar e gerenciar documentos, armazenar, recuperar ou acompanhar processos é muito alto e difícil de ser realizado com agilidade. Os softwares auxiliam essas atividades garantindo mais segurança, economia e agilidade, facilitando as decisões, barateando os custos e a mão-de-obra (LOPES, 2009).

O uso de novas tecnologias que vão surgindo constantemente geram benefícios incontáveis tanto para os que utilizam de forma direta bem como os que utilizam de forma indireta (LOPES, 2009). A administração pública também tem se beneficiado da tecnologia da informação para melhorar a prestação de seus serviços, eliminando o acúmulo de documentos gerados pelo processo burocrático.

O Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO) regulamentou o regime de trabalho dos Professores do Ensino Básico, Técnico e Tecnológico (PEBTT), através da Resolução nº 74/2013/CONSUP/IFTO, de 16 de dezembro de 2013. Entre as atribuições dos professores, está a necessidade de elaborar o plano de trabalho, pelo qual ele deverá explicitar e descrever as atividades seguindo o modelo contido no regulamento (ANEXO B). O desenvolvimento de uma aplicação web que auxilie na elaboração do plano de trabalho é usado como estudo de caso do presente projeto.

A Linguagem XML (*eXtensible Markup Language*) é caracterizada por prover independência de dados bem como separar conteúdo de apresentação, e tem sido utilizada por desenvolvedores de aplicações por ser portátil e oferecer um grau de interoperabilidade (MCLAUGHLIN, 2001, p. 6). A XML pode ser combinada com diversas tecnologias objetivando desenvolver aplicações orientadas para Web, como a linguagem de programação Java.

Java possui uma coleção de APIs (bibliotecas) padrão que podem ser usadas para construir aplicações (SANTOS, 2014), uma API (*Application Programming Interfaces*) é um conjunto padrões de programação para acesso a um aplicativo de *software* ou plataforma baseado na Web.

Existem APIs para processamento XML, como o DOM e SAX, no entanto eles apresentam certa complexidade. Pensando em facilitar a manipulação de documentos XML em Java foi criada a API JDOM, ela melhora o aprendizado pois é mais simples e é específica para a linguagem JAVA, sendo utilizada para facilitar a criação e atualização de documentos XML pelas aplicações JAVA (TODD, 2003). Ela é uma API de código-fonte aberto que é, fornece um meio para otimizar operações XML em Java através de uma estrutura de árvore, nesse aspecto é um pouco semelhante ao DOM, mas foi construído especificamente para Java e, por isso é em muitos aspectos mais intuitivos para um desenvolvedor Java do que outras APIs.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma aplicação web em java utilizando recursos da API JDOM para facilitar as operações de manipulação de dados semi-estuturados em XML.

1.1.2 Objetivos Específicos

- Estudar a aplicabilidade dos principais recursos da API JDOM para manipulação de dados em XML;
- Desenvolver uma aplicação web para automatizar a elaboração do plano de trabalho do PEBTT.

1.2 Justificativa

De acordo com o Regulamento (IFTO, 2013) os professores devem elaborar o plano de trabalho no início de cada semestre letivo e protocolar junto à chefia imediata até quinze dias úteis após o início do semestre letivo. Plano de Trabalho é o documento administrativo-pedagógico onde são descritas as atividades didáticas e administrativas para planejamento e orientação das atividades exercidas pelo professor no IFTO.

Considera-se Professor do Ensino Básico, Técnico e Tecnológico – PEBTT do IFTO – o profissional que estiver em efetivo exercício no cargo de professor, após

aprovação, nomeação e posse em concurso público (IFTO, 2013). De acordo com a Diretoria de Gestão de Pessoas, o IFTO possui 605 (seiscentos e cinco) professores efetivos divididos em 11 (onze) *campi*.

Na elaboração do plano de trabalho pelos professores, surgem dificuldades em organizar, armazenar e recuperar, gerando acúmulo de documentos. Por ter sido regulamentado recentemente e não existir um sistema que auxilie na elaboração do plano de trabalho dos professores, a fim de propiciar mais segurança, economia e agilidade neste processo, o presente sistema visa atender a resolução vigente e facilitar o trabalho dos professores do IFTO.

Um sistema web que use a linguagem XML para descrever e armazenar informações é muito útil, como é uma linguagem de marcação possibilita a formatação de elementos por meio de *tags* e atributos, como as linguagens HTML e XHTML; e por ser extensível possibilita a criação de elementos, ou seja, você mesmo pode inventar as suas *tags*. Fatores esses que segundo Silva (2001) tornam a linguagem preferida na transformação e armazenamento de dados.

A linguagem XML é útil para a transferência de dados, no entanto, durante a transferência pode ser importante registrar os dados que estão sendo enviados, ou retê-los em um arquivo onde poderão ser pesquisados e alterados. Uma vantagem da XML é que sua estrutura é mais expressiva do que os relacionamentos usados em bancos de dados relacionais (GRAVES, 2003). Através da XML é possível representar com extrema facilidade atributos multivalorados, relações de hierarquia e informações opcionais, entre outras coisas. Por esta razão, é muito mais fácil mapear um objeto para XML do que fazer o mapeamento para tabelas relacionais. Os dados podem ser representados mais claramente em um repositório XML. Os dados XML possuem natureza auto descritiva, as próprias *tags* já são capazes de explicar o significado dos dados, podendo facilmente identificar as informações.

2 REFERENCIAL TEÓRICO

Nesta seção são abordados todos os conceitos que fundamentam a criação da aplicação web utilizando recursos da API JDOM. Consiste em realizar uma revisão dos trabalhos já existentes sobre o tema abordado, possibilitando fundamentar o estudo das tecnologias utilizadas e nortear esta pesquisa.

2.1 A linguagem XML

Linguagem de marcação compreendem todas aquelas que possibilitam a formatação de elementos por meio de *tags* e atributos como o HTML e XML.

A linguagem XML (*eXtensible Markup Language*) é uma linguagem de marcação para descrição de informação semiestruturada, mantida pelo comitê do consórcio internacional W3C (*World Wide Web Consortium*), disponível em <http://www.w3.org/>, responsável por definir os padrões de diretrizes de sua estrutura.

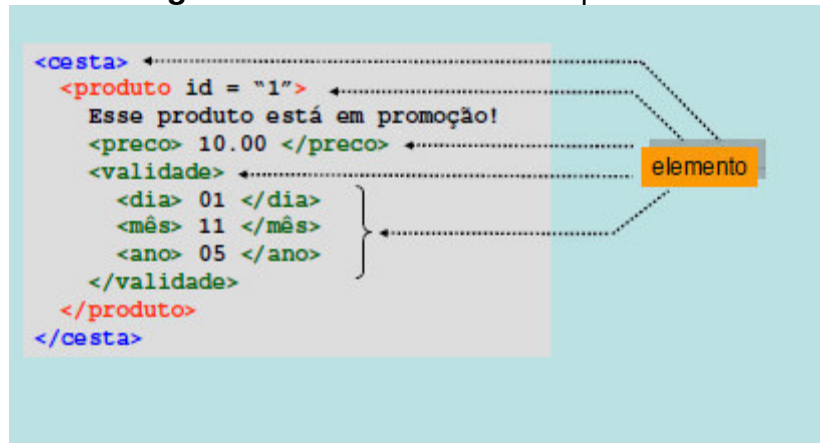
Um documento XML é formado por uma sequência de elementos aninhados e delimitados por *tags*, que servem principalmente para descrever a estrutura dos documentos, permitindo ainda que o desenvolvedor crie seus próprios elementos livremente (SILVA, 2001, p.17). A XML é usada para descrever e armazenar informações e possibilita realizar busca estruturada de informação baseada em *tags*, em vez de apenas em textos.

A formatação e apresentação dos elementos de um documento XML é feito com o uso de folhas de estilo como a XSL (*eXtensible Stylesheet Language*) ou a CSS (*Cascading Style Sheets*). A separação da descrição da estrutura de um documento de como ele deve ser formatado torna o processo de desenvolvimento e manutenção mais efetivo, uma vez que pode se trabalhar nas duas partes de maneira independente. Com o uso de tecnologias como XSL ou XSLT, Xpath, DOM e outras, o usuário será capaz de classificar os dados de diversas maneiras, filtrar, efetuar cálculos, entre outras funcionalidades.

A XML é considerada uma meta-linguagem, pois oferece recursos para a definição de gramáticas que caracterizam linguagens para classes de documentos específicos, com conjunto de elementos, atributos e regras de composição bem determinados.

O elemento é um componente básico, sendo composto por uma *tag* e seu conteúdo. Ele pode possuir atributos e também pode ser vazio. Todo elemento possui uma *tag* de início e uma *tag* de fim, mesmo que não possua conteúdo algum, tudo que estiver entre as *tags* de início e fim de um elemento será considerado seu conteúdo. A figura 1 mostra como de elementos em um arquivo xml.

Figura 1: Elementos de um arquivo xml



Fonte: pt.slideshare.net

Um elemento pode ser composto de:

- Outros elementos;
- Caracteres;
- Outros elementos e caracteres;

Um elemento raiz XML possui as seguintes características:

- Representa o elemento que se encontra no topo da hierarquia de um documento XML;
- Todos os demais elementos são descendentes dele (elementos filhos);

O atributo, por sua vez, é colocado obrigatoriamente dentro da *tag* de início, logo após o nome do elemento. Geralmente um atributo é usado para qualificar ou complementar a informação marcada pelo elemento ao qual pertence.

- Um elemento pode possuir um ou mais atributos;
- São especificados no marcador de início ou no marcador que define um elemento vazio;

Segundo Oliveira (2003) as aplicações que utilizam XML como formato de dados se classificam de acordo com o tipo de documentos que armazenam, que

podem ser documentos centrados em dados e documentos centrados em documentos. O primeiro utiliza a XML para transporte de dados e são modelados para o consumo de uma aplicação. Já o segundo é caracterizado por uma estrutura irregular ou menos regular que a dos documentos centrados em dados. São geralmente modelados para que pessoas – e não máquinas – os leiam, como livros, e-mail, etc.

Por armazenar documentos XML em sua forma pura e nativa, sua utilização é maior quando a aplicação possui dados semiestruturados ou documentos centrados em documentos, que dá importância à apresentação final das informações para o usuário. A estrutura de um elemento XML é mais expressiva que a do relacionamento, porque ela pode conter outros elementos, variar a ordem e a quantidade de atributos, e ter vários elementos do mesmo tipo. Isso torna mais fácil representar dados mais complexos (GRAVES, 2003). Com a XML é possível representar com extrema facilidade atributos multivalorados, relações de hierarquia e informações opcionais.

2.1.1 Xpath

Recomendado pela W3C em 16 de novembro de 1999, a XPath (*XML Path Language*) é uma linguagem de consulta que define como os nós dentro dos documentos XML podem ser acessados (DAUM, 2002). XSLT usa XPath para navegar através de elementos e atributos em documentos XML.

Documentos XML podem ser representados como árvores de nós. XPath usa uma expressão padrão para identificar nós em um documento XML. Um padrão XPath é uma lista nomes de elementos filhos separada por barras que descreve o caminho através do documento XML. O padrão seleciona elementos que combinam com o caminho. Segundo o site W3Schools, a XPath é um elemento importante no padrão XSLT, pois sem o seu conhecimento não será capaz de criar documentos XSLT.

2.2 Linguagem de programação JAVA

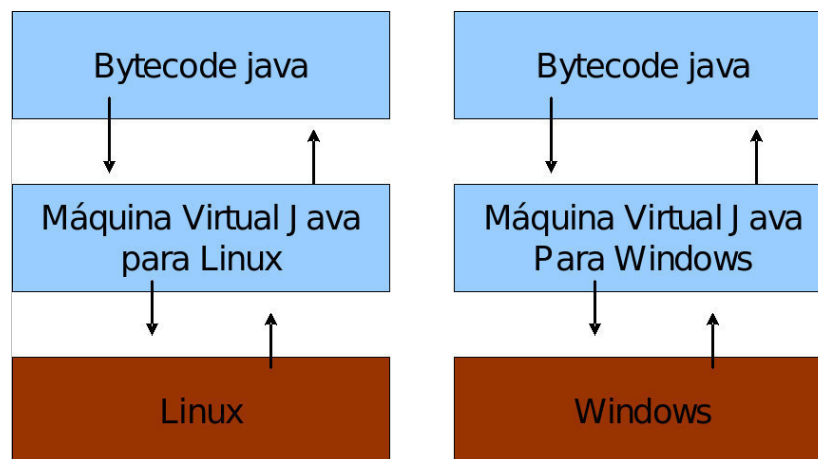
Java foi criado pela empresa *Sun Microsystems* e anunciado formalmente em uma conferência em maio de 1995 (DEITEL, 2010, p.6), em 2009 foi comprado

pela Oracle Corporation (ORACLE, 2016). Para o consumidor de produtos e usuários leigos, não desenvolvedores, é usado um site mais institucional java.com.

É uma linguagem multiplataforma, pois permite que as aplicações desenvolvidas sejam executadas em diferentes hardwares e sistema operacionais. O Java utiliza do conceito de máquina virtual, a JVM (*Java Virtual Machine*), que existe entre o sistema operacional e a aplicação. É uma camada responsável por traduzir o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento. Dessa forma, não é preciso se preocupar em qual sistema operacional sua aplicação está rodando, nem em que tipo de máquina ou configurações.

A aplicação roda sem nenhum envolvimento com o sistema operacional, apenas conversando com a JVM. O código-fonte de aplicação Java é traduzido para um código de máquina, conhecido por “bytecode”, que é gerado por um compilador java, como o javac. Os bytecodes então, são interpretados pela JVM que traduz para o sistema operacional, conforme a figura 2.

Figura 2: Ilustra uma Máquina Virtual Java (JVM)



Fonte: www.caelum.com.br

Para ter a Máquina Virtual Java em seu sistema é necessário instalar o JRE (*Java Runtime Environment*), um ambiente de execução Java, formado pela JVM e bibliotecas que permitem ao usuário rodar aplicativos Java em seu computador.

Para os desenvolvedores é necessário o JDK (*Java Development Kit*) do Java SE (*Standard Edition*), ele é formado pela JRE somado a ferramentas para escrever aplicações e compilador. Para o desenvolvimento de *softwares* corporativos, incluindo serviços de rede e web é utilizada a plataforma Java EE (*Enterprise Edition*).

Segundo Todd (2003) o Java EE se baseia na Standard Edition com um número significativo de outros pacotes que são utilizados especificamente dentro de aplicações corporativas distribuídas.

Como o XML é simplesmente texto, ele pode ser movido entre diferentes plataformas (MCLAUGHLIN, 2001). Assim como o Java que pode ser executado em qualquer sistema operacional, precisando apenas de uma Máquina Virtual Java, sendo JRE ou JVM. A combinação Java e XML oferecem um código portátil e dados portáteis.

2.2.1 JavaServer Faces (JSF)

O Java EE possui várias especificações, entre elas, algumas específicas para lidar com o desenvolvimento de uma aplicação Web, como a JSF.

JavaServer Faces (JSF) é uma especificação Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Por ser uma especificação do Java EE, todo servidor de aplicações Java tem que vir com uma implementação de referência dela (SILVEIRA, 2015).

O JSF fornece um conjunto de componentes de interface com o usuário que simplificam o design de páginas Web. O JSF fornece um conjunto de componentes de interface de usuário, ou componentes JSF, que simplificam o design de páginas Web, basta adicionar os componentes (calendários, tabelas, formulários) e eles serão renderizados e exibidos em formato web. Segundo Deitel (2010) JSF também inclui APIs para tratar eventos de componentes (como o processamento de modificações de estado dos componentes e validação de entrada de usuário), navegação entre páginas de aplicativos Web.

Existem diversas outras implementações disponíveis, como Oracle Mojarra disponível em <http://javaserverfaces.java.net/>, e a MyFaces da Apache Software Foundation em <http://myfaces.apache.org/>. No entanto essas são implementações básicas e tratam do que é fundamental para que ele seja estável, pois as possibilidades das interfaces como usuário crescem rapidamente. JSF é extensível para que se use bibliotecas de terceiros, que são essas bibliotecas de componentes, que possuem componentes mais avançados, como a PrimeFaces, disponível no endereço <http://www.primefaces.org>.

2.3 Autenticação do usuário com LDAP

O Lightweight Directory Access Protocol (LDAP) é um protocolo de aplicação aberto, uma especificação que define um modo padronizado pelo qual as organizações armazenam e acessam as informações de diretório sobre uma rede TCP/IP (BROOKS-BILSON, 2001, p.490). Esse protocolo é muito útil, pois permite o compartilhamento de informações sobre usuários, sistemas, redes, serviços e aplicações através da rede. Uma utilização para LDAP é fornecer um acesso em que uma senha para um usuário é compartilhada entre muitos serviços. O IFTO já utiliza esse serviço em suas aplicações, a facilidade é que não há necessidade de fazer um banco de dados para cadastrar todos os usuários, podendo utilizar as informações já existente em outras aplicações.

2.4 API JDOM

Ambiente de desenvolvimento Java provê APIs para processamento de documentos XML, como as APIs DOM e SAX. Entretanto, essas APIs apresentam certa complexidade que tende a dificultar o seu aprendizado.

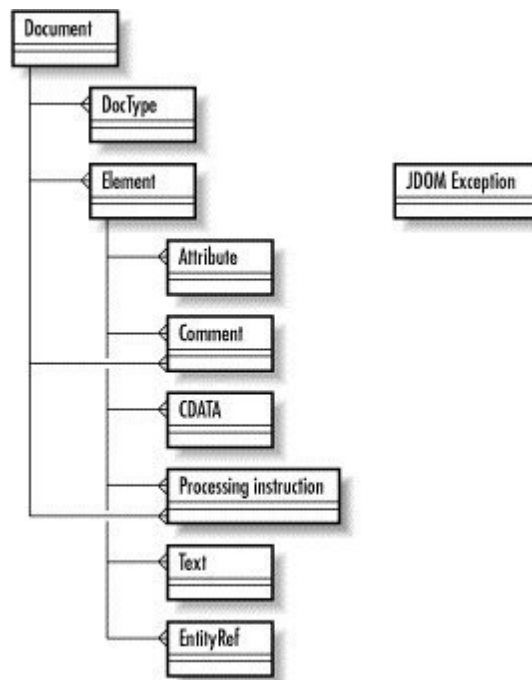
De modo a facilitar o aprendizado, Jason Hunter e Brett McLaughlin criaram uma API mais simples, chamada JDOM. Ela é uma API de código-fonte aberto que fornece um meio para otimizar operações XML em Java através de uma estrutura de árvore, nesse aspecto é um pouco semelhante ao DOM, mas foi construído especificamente para Java e, por isso, é em muitos aspectos mais intuitivos para um desenvolvedor Java do que outras APIs, como o DOM (MCLAUGHLIN, 2001, p.137).

O JDOM foi criado porque o SAX e o DOM não eram suficientemente capazes de atender suas necessidades dos programadores Java. De acordo com TODD (2003) o SAX não oferece uma forma padrão para descrever um documento XML, ele é baseado em fluxo e decompõe as informações em um documento XML em sequências de chamadas de métodos, acontece que ao utilizar o SAX não se pode modificar o documento XML original nem pular no fluxo. O DOM é uma API baseada em árvore, ela constrói uma estrutura em árvore na memória, que representa as informações no documento XML, desta forma, a árvore fica inteira na memória. Uma vez carregado na memória o documento JDOM é possível modifica-lo, pois a árvore

é totalmente leitura e escrita, todas as partes podem ser movido, excluído e adicionado, ao contrário do DOM que sua árvore é somente leitura (HAROLD, 2002).

A figura 3 que dá uma melhor compreensão de como se comporta o JDOM, que representa um documento XML como uma árvore composta de elementos, atributos, comentários, instruções de processamento, nós de texto, seções CDATA, e assim por diante.

Figura 3: Arvore JDOM



Fonte: MCLAUGHLIN, 2011

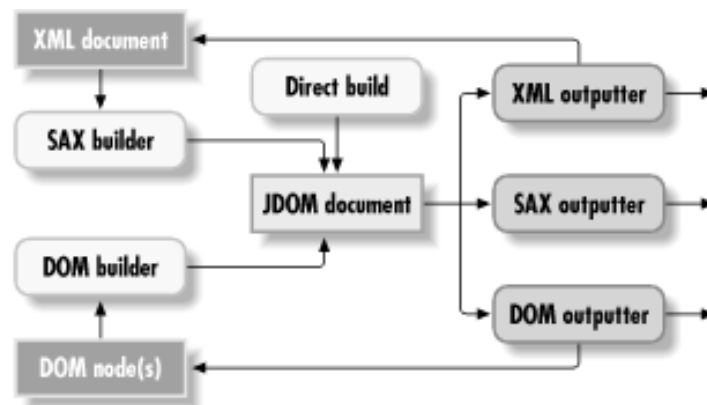
A utilização do JDOM é mais amigável do que SAX ou DOM para a maioria de tarefas. Segundo Santos (2014) ele possui ainda a conveniência de uma árvore baseada na API DOM e a familiaridade de convenções do padrão Java para SAX. Por ser uma API baseada em Java, o JDOM aproveita-se dos recursos da linguagem de programação e das bibliotecas. Além de representar o documento XML de uma maneira orientada a objetos tornando mais familiar aos programadores Java.

O que o JDOM tenta fazer é aproveitar o que existe de melhor do SAX e do DOM e criar uma API baseada em Java de peso leve, com poucos *overheads* quanto ao uso da memória e de processamento (TODD, 2003, p.276). O objetivo expresso no site do JDOM é:

Construir com uma solução completamente passeada em Java para acessar, manipular e gerar a saída dos dados XML a partir de código Java (HUNTER, 2000)

Segundo Harold (2002) toda a entrada e saída da API JDOM devem contar com um analisador para ler o documento XML e construir um documento JDOM. Ele também pode aceitar eventos SAX ou uma árvore DOM como entrada. Para facilitar o processo entrada, JDOM fornece um pacote `org.jdom.input` que fornece classes que serão usados para construir o XML, são as classes `SAXBuilder` e `DOMBuilder`. A classe `SAXBuilder` busca uma implementação SAX para gerar um documento JDOM. A `DOMBuilder` é a classe que utiliza um analisador de DOM para a mesma tarefa. O processo de saída o JDOM possui o pacote `org.jdom.output` e suas classes mais importantes são `XMLOutputter`, `DOMOutputter`, `SAXOutputter`. A classe `XMLOutputter` é a mais útil segundo site do JDOM (HUNTER, 2000). Ele produz o documento JDOM2 como um fluxo de bytes que podem ser gravados em um arquivo. A figura 4 ilustrar bem esse processo de entrada e saída.

Figura 4: Processo de entrada e saída JDOM



Fonte: MCLAUGHLIN, 2011

A versão mais recente da API é a JDOM 2.0, os seus principais pacotes são (HUNTER, 2000):

`org.jdom2` - Este pacote contém as classes centrais que representam os componentes XML:

- *Document* - Representa o documento XML completo. Ele fornece acesso ao elemento raiz e também o `docType`.
- *DocType* - Representa um DOCTYPE XML.

- *Element* - Representa um elemento XML. Tem métodos para manipular os elementos filho, o seu conteúdo textual, atributos e namespaces.
- *Attribute* - o que representa um atributo XML. Existem métodos para obter e definir o valor do atributo e também o namespace. O atributo tem um pai e um *AttributeType*.
- *Namespace* - Representa um espaço de nomes XML.
- *CDATA* - Isto representa uma seção CDATA XML.
- *Text* - o que representa um texto XML.
- *Comment* - Representa um comentário XML. Os usuários podem obter e definir o texto do comentário.
- *EntityRef* - Representa um XML EntityRef.
- *ProcessingInstruction* - Representa uma instrução de processamento XML
- *Content* - Esta é uma classe abstrata que se estende por essas classes que podem ser uma criança de uma classe *org.jdom2.Parent* (*Element*, *CDATA*, *Comentário*, *DocType*, *EntityRef*, *ProcessingInstruction*, *texto*)

org.jdom2.filter - Estes pacote tem filtros que permite a filtragem com base no tipo, nome, valor ou outros parâmetros.

org.jdom2.input - Este pacote tem as classes principais responsáveis pela construção do documento JDOM2 do DOM, SAX ou StAX.

org.jdom2.output - Estes pacote tem classes para OUPUT o documento JDOM2 para vários destinos.

org.jdom2.transform - Ajuda com a transformação XML usando classes JAXP Trax.

org.jdom2.xpath - Suporte para XPath em JDOM2. Por padrão, ele usa a biblioteca de xpath Jaxen.

A filosofia do JDOM tem a intenção de tornar simples a manipulação em Java de documentos XML e está baseada nas seguintes afirmações (SANTOS, 2014, p. 18):

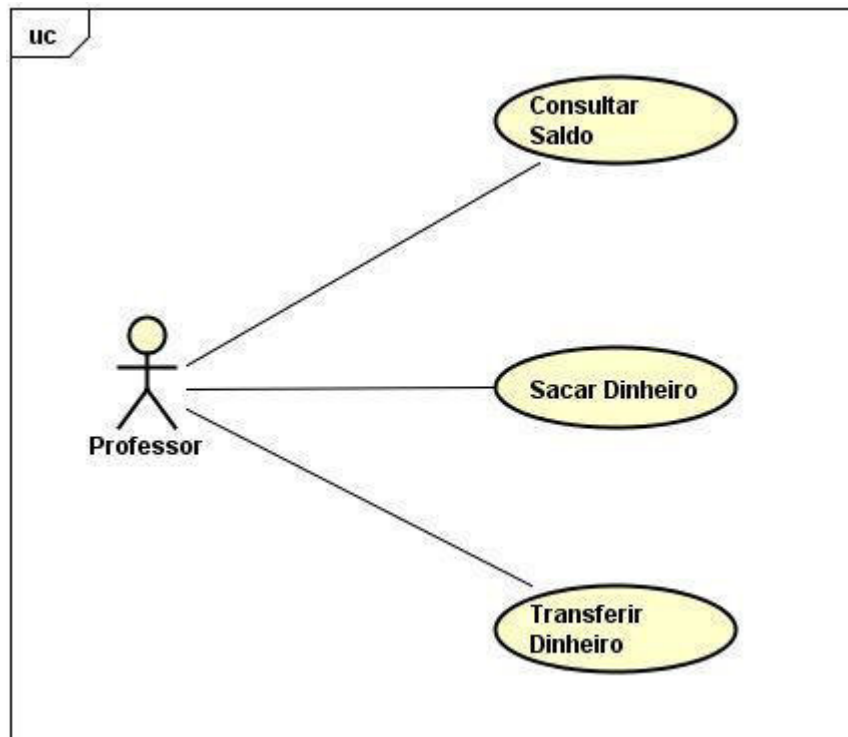
- JDOM é direcionado para programadores Java.
- JDOM deve suportar a modificação fácil e eficiente de um documento XML.
- JDOM deve esconder as complexidades do XML sempre que possível.
- JDOM deve integrar-se com DOM e SAX.
- JDOM deve ser leve e rápido.

- JDOM deve resolver 80% (ou mais) de problemas de Java/XML com 20% (ou de menos) do esforço.

2.5 Linguagem UML

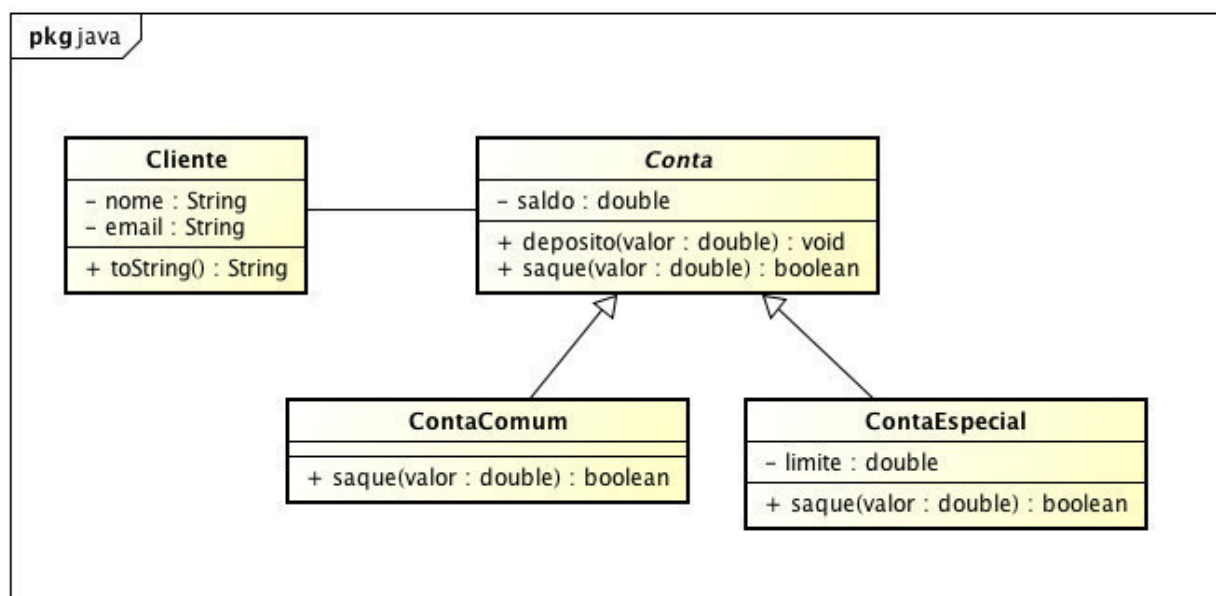
A UML (Unified Modeling Language) é uma linguagem padrão para modelagem orientada a objeto, tem a função de auxiliar na tarefa de modelar e documentar os sistemas. Construímos modelos para comunicar a estrutura e o comportamento desejado do sistema, para visualizar e especificar a arquitetura do sistema, para compreender melhor o sistema que estamos elaborando, para gerenciar os riscos, construir e documentar um sistema complexo de software (BOOCH, 2012). A UML usa notação gráfica por meio de diagramas que apoia na criação de muitos tipos de diferentes modelos de sistema. Segundo Sommerville (2011) pesquisas mostram que a maioria dos usuários de UML acredita que os diagramas de casos de uso e os diagramas de classe estão entre os tipos de diagramas podem representar a essência de um sistema.

Os diagramas de casos de uso, que mostram as interações entre um sistema e seu ambiente. Sommerville (2011) ressalta que os casos de uso permitem apresentar requisitos que vão interagir diretamente com o sistema. Serve como uma ferramenta de comunicação que resume o comportamento do sistema e seus atores. A figura 5 é um exemplo de um diagrama de caso de uso da interação que o usuário com um sistema bancário.

Figura 5: Exemplo de diagrama de caso de uso

Fonte: do autor, 2016

Os diagramas de classe, que mostram as classes de objeto no sistema e as associações entre as classes (figura 6). Eles são responsáveis por ilustrar classes, interfaces e suas associações (LARMAN, 2007, p.265).

Figura 6: Exemplo de diagrama de classe

Fonte: do autor, 2016

2.6 Processo de software

Sommerville (2011) define processo de *software* como um conjunto de atividades relacionadas que resultam em um produto de *software*. Segundo o autor, não existe um processo ideal, as organizações geralmente desenvolvem seus próprios processos de desenvolvimento de *software*. Entretanto, todos devem incluir quatro atividades fundamentais: especificação do *software*; projeto e implementação; validação e evolução do *software*. As atividades de especificação, desenvolvimento, validação e evolução são distintas, isto é, é necessário planejar e programar todas as atividades do processo antes de se iniciar os trabalhos.

A modelagem e a implementação do sistema tem como modelo o ciclo de vida cascata. Segundo Larman (2007) em um processo de ciclo de vida cascata há uma tentativa de definir, em detalhes, todos ou a maioria dos requisitos antes da programação. A especificação consiste no processo de compreensão e definição dos requisitos e restrições sobre as operações e desenvolvimento do sistema.

“Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferecem e as restrições a seu funcionamento” (SOMMERVILLE, 2011, p. 57). Os requisitos fornecem informações mais específicas sobre os serviços e funções que deverão ser implementadas. Estes requisitos são frequentemente classificados como requisitos funcionais e não-funcionais.

Os requisitos funcionais especificam ações que um sistema deve ser capaz de executar, sem levar em consideração restrições físicas. Descrevem as funções do sistema e especificam o comportamento de entrada e saída (SOMMERVILLE, 2011). Sendo assim, os requisitos funcionais descrevem o funcionamento e as operações que irá realizar.

Os requisitos não-funcionais, são os requisitos que não estão diretamente ligados com os serviços oferecidos pelo sistema. Para Sommerville (2011) os requisitos não-funcionais normalmente especificam ou restringem as características do sistema como um todo, tais como confiabilidade, desempenho, proteção ou disponibilidade.

Os Diagramas de Caso de Uso foram utilizados os elementos de modelagem UML. E servem para representação das funcionalidades interna e externa do sistema. Larman (2007) diz que um diagrama de caso de uso é uma excelente

imagem do contexto do sistema; ele é bom diagrama de contexto, ou seja, mostra a fronteira de um sistema, o que está fora dele e como o sistema é usado.

Para Sommerville (2011) a fase de projeto e implementação é a fase de codificação de um sistema. Conforme o autor, a fase de projeto é uma atividade criativa que identifica-se os componentes de softwares e os relacionamentos com base nos requisitos. Enquanto a implementação é o processo de concretização como um programa.

2.7 Plano de Trabalho docente

Segundo regulamento (IFTO, 2013), os professores devem elaborar o plano de trabalho no início de cada semestre letivo e protocolar junto à chefia imediata até quinze dias úteis após o início do semestre letivo.

(Art. 15, 16 e 17º do Regulamento dos regimes de trabalho docente, 16/12/13):

Art. 15. São atribuições gerais do PEBTT:

II - elaborar e cumprir o plano de trabalho, segundo a proposta pedagógica do estabelecimento de ensino;

Art. 16. Plano de Trabalho é o documento administrativo-pedagógico de planejamento e orientação das atividades exercidas pelo PEBTT no IFTO.

Art. 17. O Plano de Trabalho deverá explicitar as atividades contidas no Apêndice A e descritas de acordo com o formulário do Apêndice B.

Art. 18. O Plano de Trabalho deverá ser elaborado no início de cada semestre letivo e protocolizado junto à chefia imediata em até quinze dias úteis após o início do semestre letivo. (IFTO 2013)

Nesse plano de trabalho deverão ser descritas as atividades didáticas e administrativas para planejamento e orientação das atividades exercidas pelo professor. Está disponível um formulário próprio para preenchimento do plano de trabalho, *Apêndice B* do regulamento (IFTO, 2013) (ANEXO B), conforme as figura 7 e 8.

Figura 7: Plano de trabalho docente – atividades**APENDICE B – PLANO DE TRABALHO DO PEBTT**

PLANO DE TRABALHO DO PEBTT – SEMESTRE LETIVO: _____			
Tipo:			
☐ Inicial. ☐ Retificador. Data do último plano de trabalho: ____/____/____			
Identificação do Servidor			
Nome:			Matrícula SIAPE:
Efetivo:	20h ☐	40h ☐	DE ☐ Situação: Estável ☐ Em Estágio Probatório ☐
Substituto:	20h ☐	40h ☐	Fone:
Temporário:	20h ☐	40h ☐	E-mail:
Campus de Lotação:			
Departamento de Lotação:			
Atividade: Aulas			
Cód.	Disciplina	Cód. do Curso/Turma	C.H.
A1			
A2			
Atividade: Manutenção do Ensino			
Cód.	Descrição da Atividade	C.H.	
M1			
M2			
Atividade: Apoio ao Ensino			
Cód.	Descrição da Atividade	C.H.	
AP1			
AP2			
Atividade: Extensão			
Cód.	Descrição da Atividade	C.H.	
E1			
E2			
Atividade: Pesquisa			
Cód.	Descrição da Atividade	C.H.	
P1			
P2			
Atividade: Gestão			
Cód.	Descrição da Atividade	C.H.	
G1			
G2			
Atividade: Comissão			
Cód.	Descrição da Atividade	Portaria	C.H.
C1			

Fonte: IFTO, 2013

Figura 8: Plano de trabalho docente – extrato

EXTRATO DO PLANO DE TRABALHO DO PEBTT – SEMESTRE LETIVO: _____		
Atividades	Quantidade	Carga Horária Semanal
Aulas		
Atividades de Manutenção do Ensino		
Atividades de Apoio ao Ensino		
Atividades de Extensão		
Atividades de Pesquisa		
Atividades de Gestão		
Atividades de Comissão		
Carga Horária Semanal Total:		
_____ de _____ de _____ Local/Data		
_____	_____	
Assinatura do Servidor	Assinatura e Carimbo da Chefia Imediata/	

Fonte: IFTO, 2013

No formulário deve-se preencher os dados pessoais, funcionais do professor e explicitar as atividades contidas no Apêndice A do regulamento (IFTO, 2013) (ANEXO A), que são divididas em atividades didáticas: aulas, manutenção do ensino, apoio ao ensino, pesquisa e extensão; e atividades administrativas: gestão e comissão. As atividades são descritas no regulamento (IFTO, 2013):

Art. 21. As Atividades de Ensino compreendem as ações do PEBTT diretamente vinculadas aos cursos e programas regulares autorizados, em todos os níveis e modalidades de ensino, ofertados pelo IFTO.

Parágrafo único. As atividades de ensino compreendem as aulas, as atividades de manutenção e as atividades de apoio, conforme Apêndice A.

Art. 22. As aulas são atividades didáticas letivas, sistemáticas, teóricas e práticas, dispostas nos projetos de cursos.

(...)

Art. 27. As Atividades de Manutenção do Ensino são as ações didáticas do PEBTT relacionadas ao estudo, planejamento, preparação, desenvolvimento e correção das avaliações ministradas nos cursos e programas regulares autorizados do IFTO.

Art. 28. Serão consideradas Atividades de Apoio ao Ensino as ações do PEBTT diretamente vinculadas às matrizes curriculares e programas dos cursos regulares autorizados do IFTO, compreendendo:

I - atendimento ao Estudante;

II - orientação e coorientação de Trabalho de Conclusão de Curso de graduação e pós-graduação. Até duas horas por orientando;

III - orientação de Estágio Curricular Supervisionado, de Atividades Complementares Curriculares, entre outras. Até uma hora por orientando;

IV - orientação e coordenação de treinamentos esportivos, e de ensaios artísticos e culturais;

V - orientação em Programas Institucionais ou Governamentais de Bolsas. Até duas horas por projeto/programa;

VI - participação em Reuniões Ordinárias e/ou Extraordinárias (de Conselhos de Análise de Turma, de Núcleo Docente Estruturante – NDE –, de Colegiado, Pedagógico-administrativas, entre outras);

VII - Atividades em laboratórios e/ou unidades educativas de produção;

VIII - Atividades de apoio ao ensino não contempladas neste documento e que porventura sejam configuradas como tal pela gestão máxima de ensino (Gerência/Direção de Ensino).

(...)

Art. 29. As atividades de pesquisa são aquelas de natureza teórica, metodológica, prática ou empírica, desempenhadas em ambientes tecnológicos ou em campo visando à produção técnica, científica ou tecnológica.

(...)

Art. 32. As atividades de extensão são aquelas de transferência mútua de conhecimento produzido, desenvolvido ou instalado no âmbito do IFTO, ou na relação deste com a comunidade externa.

(...)

Art. 38. As atividades administrativas compreendem o exercício em cargos comissionados, funções gratificadas ou não gratificadas, e comissões não gratificadas, providos por atos administrativos. (IFTO, 2013)

Após a descrição de todas as atividades é feito o extrato do plano de trabalho com a quantidade de cada atividade e as respectivas carga horária e feito o quadro de horário com o dia da semana e horário que essas atividades serão desempenhadas.

3 METODOLOGIA

O método adotado na elaboração do trabalho foi pautada pela pesquisa bibliográfica, fundamentando o presente estudo, e o desenvolvimento de uma aplicação web em Java utilizando recursos da API JDOM, objeto desse estudo.

A primeira etapa teve como objetivo fazer o levantamento das contribuições científicas sobre o tema e fornecer uma solução para o problema proposto neste trabalho, para isso foram escolhidos como método de pesquisa levantamento referencial teórico através de abordagens bibliográficas e documentais, bem como a pesquisa experimental e descritiva, por meio de artigos, livros e sites especializados no tema. Diversos sites e tutoriais foram usados para testar e entender o funcionamento das tecnologias, como no site do JDOM (HUNTER, 2000) que disponibiliza toda documentação e tutoriais sobre JDOM.

O desenvolvimento do projeto foi feito na segunda etapa. Primeiro teve o planejamento e especificação das funcionalidades da aplicação, sendo baseado no regulamento (IFTO, 2013), onde foram feitos os diagramas de classe e de caso de uso.

Foram realizadas todas as configurações de ambiente necessárias para o desenvolvimento, instalando e configurando as ferramentas e ambiente de desenvolvimento, estes baseadas em ferramentas gratuitas. Utilizou-se a IDE (Integrated Development Environment) ECLIPSE para desenvolvimento, pois possui um editor de código completo e tem uma velocidade de compilação equiparada a outras ferramentas existentes.

O Eclipse possui muitos plugins para auxílio no desenvolvimento das aplicações e também é considerada uma IDE leve. Na criação do Web Service foi utilizado o software Apache Tomcat e configurado a versão Tomcat 8.0 como servidor web.

A implementação foi iniciada com a construção da interface utilizando a especificação JSF, passando em seguida para a codificação da API JDOM para fazer a manipulação do documento XML.

4 ESPECIFICAÇÃO DO SOFTWARE

A aplicação desenvolvida durante este projeto é uma proposta para auxiliar os professores do IFTO na elaboração do plano de trabalho que deve ser feito no início de cada semestre letivo e protocolado junto à chefia imediata.

Através de uma autenticação o usuário tem acesso à aplicação onde e poderá preencher todas as informações descritas no *Apêndice B* do regulamento (ANEXO B).

O levantamento dos requisitos levou em consideração o Regulamento do Regime de trabalho Docente e conversa com usuários em potencial, realizadas através da compreensão e definição de quais serviços e restrições seriam necessários para o desenvolvimento.

Sendo assim, os requisitos funcionais descrevem o funcionamento e as operações que irá realizar. Na análise os seguintes requisitos funcionais foram encontrados:

- O sistema deve permitir gravar, consultar e alterar;
- O sistema deve permitir gerar o plano de Trabalho docente;

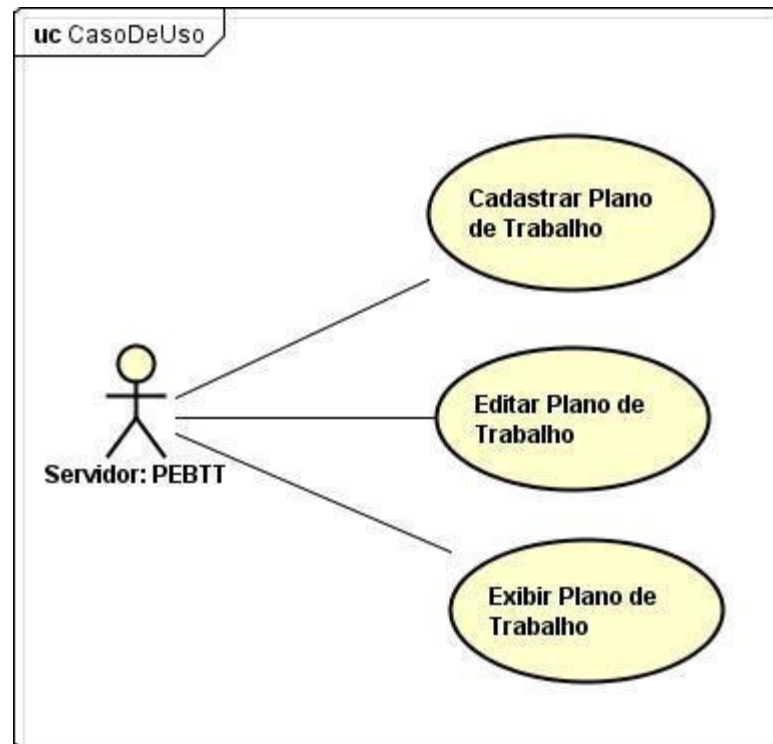
O levantamento de requisitos possibilitou conhecer e estudar os processos envolvidos na fase de elaboração do trabalho e identificar as necessidades dos docentes.

4.1 Diagrama de Caso de Uso

A modelagem do sistema foi realizada através da elaboração de diagramas. O processo de modelagem consistiu em avaliar quais dados deveriam ser armazenados, processados e retornados ao usuário final.

Sommerville (2011) ressalta que os casos de uso permitem apresentar requisitos que vão interagir diretamente com o sistema.

Figura 9: Diagrama de caso de uso interação com o usuário



Fonte: do autor, 2016

A figura 9 ilustra os Casos de Uso pertinentes ao sistema e o usuário irá gerenciar o Plano de Trabalho Docente, ou seja, o caso de uso mapeia a interação do sistema com o usuário.

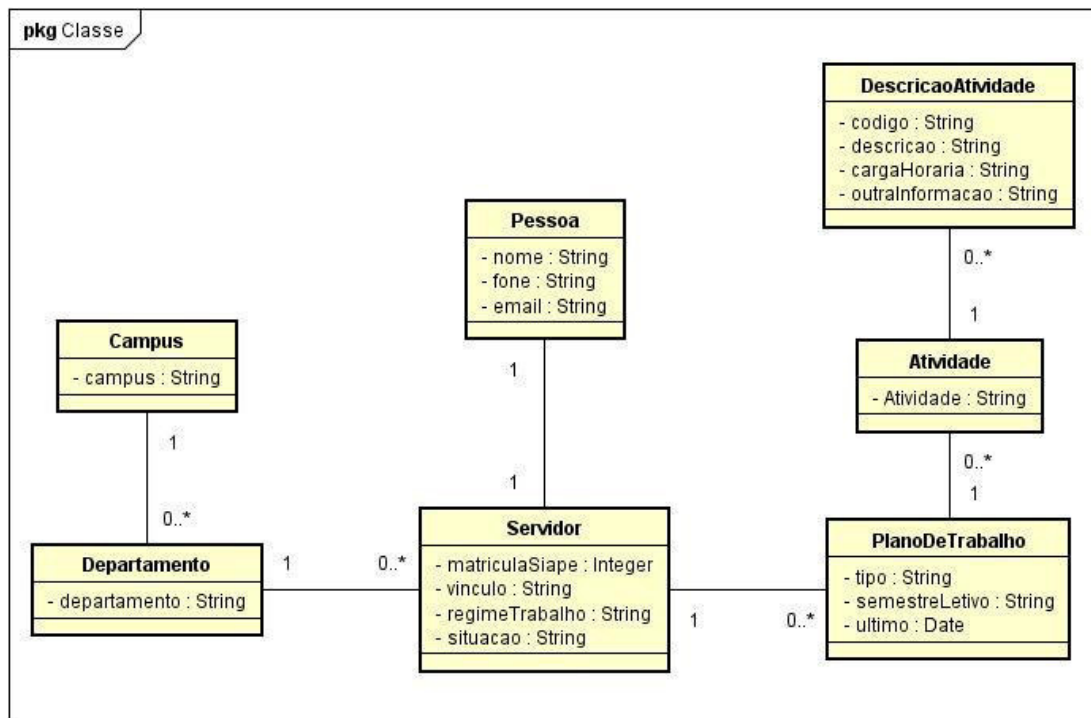
4.2 Diagrama de Classe

As especificações da atividade de projeto foram realizadas com a elaboração do diagrama de classe com especificação dos relacionamentos. A figura 10 apresenta o Diagrama de Classe, que descreve o objeto e informações de estrutura e comunicação com usuários e sistema. Na criação da aplicação existem sete classes que representam os objetos responsáveis pelo gerenciamento dos *campi* e seus departamentos, pessoas, servidores, plano de trabalho, as respectivas atividades e descrições das atividades.

Após o cliente/usuário ter acesso ao sistema ele poderá cadastrar seus dados pessoais e funcionais, *campus* e o departamento de lotação. Após se cadastrar o usuário deve descrever todas as atividades que desempenham. O plano de trabalho divide as atividades desempenhada em sete tipos, sendo: atividades didáticas: aulas,

manutenção do ensino, apoio ao ensino, pesquisa e extensão; e atividades administrativas: gestão e comissão. Cada tipo de atividade possui várias atividades relacionadas com um código, descrição, carga horário e uma informação adicional, com é o caso da atividade: aulas, que possui o código do curso e turma; atividade: comissão, que possui a informação da portaria. Para ilustrar melhor, a Figura 10 demonstra o diagrama de classes do sistema.

Figura 10: Diagrama de classes do sistema



Fonte: do autor, 2016

5 PROJETO E IMPLEMENTAÇÃO

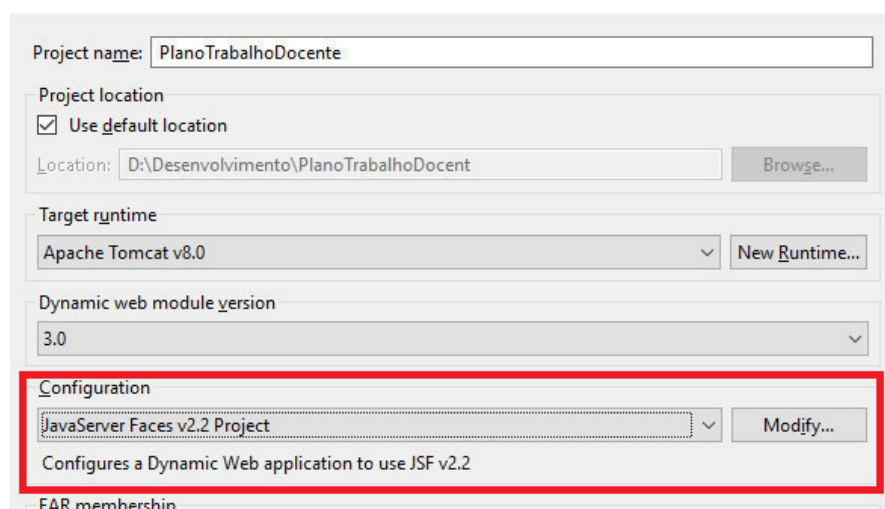
Para a implementação deste projeto, inicialmente foram efetuadas pesquisas sobre as tecnologias a serem usadas e os requisitos a serem implementados no projeto.

Para iniciar o desenvolvimento do projeto, foi necessário instalar e configurar as ferramentas e o ambiente de desenvolvimento, estes já apresentados na seção 3 em que introduz sobre quais ferramentas foram utilizadas.

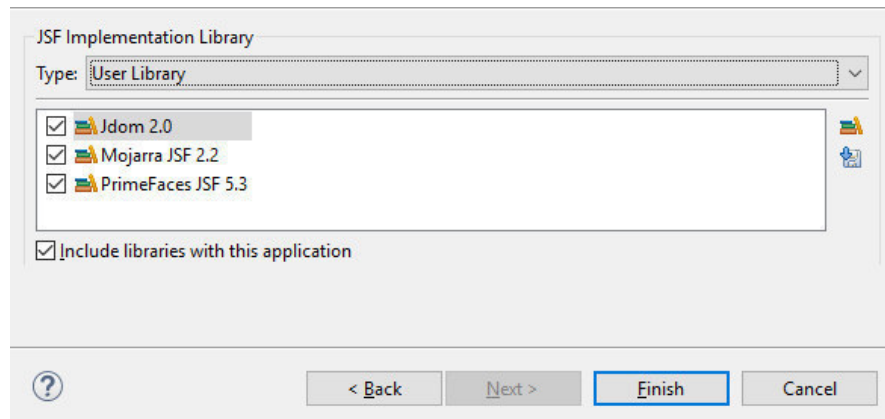
Durante a criação do novo projeto, na IDE Eclipse, foi adicionado a especificação JSF e as bibliotecas Mojarra e PrimeFaces, para construção das interfaces web. Nessa fase também adicionamos o biblioteca JDOM responsável por realizar as operações XML em Java, objeto dessa pesquisa.

Como o JSF é uma especificação do Java EE, todo servidor de aplicações Java tem que vir com uma implementação de referência, foi necessário apenas selecionar a especificação JSF, como pode ser vista na figura 11. As bibliotecas são arquivos com extensão .jar, e após serem baixadas dos seus respectivos sites Mojarra, disponível em: <https://javaserverfaces.java.net/>, PrimeFaces, em: <http://www.primefaces.org> e JDOM, em <http://www.jdom.org/>; foram adicionadas no último passo da criação do novo projeto, conforme ilustra a figura 11 e 12.

Figura 11: Configuração do eclipse com jsf



Fonte: do autor, 2016

Figura 12: Configuração do eclipse com a API JDOM

Fonte: do autor, 2016

De maneira a organizar melhor a codificação desse projeto, foram criadas dois pacotes, um é responsável pelas classes descritas na seção 4.2 e o outro responsável por instanciar os objetos e gerenciar o documento XML através da API JDOM.

5.1 Funcionalidades do sistema

O sistema permite as seguintes funcionalidades: cadastrar Plano de Trabalho Docente; Alterar os dados cadastrados; Exibir; Imprimir e Sair do sistema.

A funcionalidade Cadastrar Plano de Trabalho, apresentada na figura 13, o usuário poderá inserir os dados pessoais, funcionais e explicitar as atividades contidas no ANEXO A, referentes ao plano de atividades a serem desenvolvidas no semestre letivo.

Figura 13: Tela de cadastro do sistema

Cadastro Plano de Trabalho do PEBTT

PlanoTrabalho Identificação Atividades Extrato

Plano de Trabalho

Semestre Letivo:

Tipo:

☐ Inicial.

☐ Retificador.

→ Next

Fonte: do autor, 2016

5.2 Implementação do Código JSF

O JSF possui componentes que são renderizados e exibidos em formato HTML. Para utilizar todas as funcionalidades desses componentes é preciso utilizar, preferencialmente, as *tags* do próprio *framework*, já que, à medida que utilizarmos componentes mais avançados o JSF precisará gerenciar as *tags*, por exemplo, adicionar CSS e javascript que um componente requisitar. A figura 14 mostra como são utilizadas as *tags* do *framework* de HTML provida pelo próprio JSF.

Figura 14: Funcionalidade *tags* do JSF

```

7  <h:head>
8      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
9      <title><ui:insert name="titulo">Plano de Trabalho do PEBTI</ui:insert></title>
10
11      <h:outputStylesheet library="css" name="style.css"/>
12 </h:head>
13 <h:body>
14     <div id="header">
15         <ui:insert name="cabecalho">
16             <ui:include src="/template/Cabecalho.xhtml"/>
17         </ui:insert>
18     </div>

```

Fonte: do autor, 2016

Para remover um pouco do código Java que fica na página JSP, a Sun desenvolveu uma linguagem chamada *Expression Language* que é interpretada pelo *servlet container*.

O JSF tem a proposta de abstrair todo o protocolo HTTP (*Hypertext Transfer Protocol*), o JavaScript e o CSS. Assim o formulário é marcado apenas como um agregador de campos e cada um dos botões internos podem ter funções diferentes, podendo adicionar a configuração da ação ao próprio botão. A figura 15 demonstra essa função na *tag* `<p:inputText>` utiliza para chamar os métodos de uma classe.

Figura 15: Funcionalidade da tag *p:inputText* em JSF

```

17 <h:form id="frm">
18   <p:growl id="growl" sticky="true" showDetail="true"/>
19   <p:wizard >
20
21     <p:tab id="trabalho" title="PlanoTrabalho">
22       <p:panel header="Plano de Trabalho">
23         <p:messages />
24         <h:panelGrid columns="2">
25           <h:panelGroup>
26             <h:outputText value="Semestre Letivo: " for="semestre"/>
27             <p:inputText value="#{gestaoArquivoXML.planoDeTrabalho.semestreLetivo}" id="semestre">
28               <f:convertDateTime pattern="yyyy/MM"/>
29             </p:inputText>
30           </h:panelGroup>
31           <h:outputText/>

```

Fonte: do autor, 2016

Ao colocar o código da figura 15 estamos pegando os dados do usuário e trazendo para dentro da aplicação, usando a tag *<p:inputText>*. Ela fará a ligação entre o atributo da classe e o valor digitado no campo. No código, a classe gerenciada pelo JSF chamada *GestaoArquivoXML*, tem um atributo *PlanoDeTrabalho*, da classe *PlanoDeTrabalho*, que por sua vez tem um atributo *SemestreLetivo*, e o JSF chama o *setter* do atributo. Dessa forma todos os dados informados pelo usuário são instanciados nos objetos das classes para então serem acessados pelo código JDOM.

O atributo *action* faz a ligação com um método e assim como na figura 16, executa a ação do método *incluir* da classe *gestaoArquivoXML*. O método *incluir* é responsável por chamar o método da classe *PlanoDeTrabalhoDAO* que irá implementar o código JDOM para gerar o documento XML.

Figura 16: Atributo *action* para executar o método

```

124
125 <p:commandButton value="Cadastrar" action="#{gestaoArquivoXML.incluir}"/>
126

```

Fonte: do autor, 2016

5.3 Implementação do Código JDOM

Para desenvolver o projeto foi usado a versão da API JDOM 2.0, os seus principais pacotes são:

org.jdom2 - Este pacote contém as classes centrais que representam os componentes XML:

- *Document* - Representa o documento XML completo. Ele fornece acesso ao elemento raiz e também o *docType*.

- *DocType* - Representa um DOCTYPE XML.
- *Element* - Representa um elemento XML. Tem métodos para manipular os elementos filho, o seu conteúdo textual, atributos e namespaces.
- *Attribute* - o que representa um atributo XML. Existem métodos para obter e definir o valor do atributo e também o namespace. O atributo tem um pai e um *AttributeType*.
- *Namespace* - Representa um espaço de nomes XML.
- *CDATA* - Isto representa uma seção CDATA XML.
- *Text* - o que representa um texto XML.
- *Comment* - Representa um comentário XML. Os usuários podem obter e definir o texto do comentário.
- *EntityRef* - Representa um XML EntityRef.
- *ProcessingInstruction* - Representa uma instrução de processamento XML
- *Content* - Esta é uma classe abstrata que se estende por essas classes que podem ser uma criança de uma classe *org.jdom2.Parent* (*Element*, *CDATA*, *Comentário*, *DocType*, *EntityRef*, *ProcessingInstruction*, *texto*)
- *org.jdom2.filter* - Estes pacote tem filtros que permite a filtragem com base

no tipo, nome, valor ou outros parâmetros.

org.jdom2.input - Este pacote tem as classes principais responsáveis pela construção do documento JDOM2 do DOM, SAX ou StAX.

org.jdom2.output - Estes pacote tem classes para OUPUT o documento JDOM2 para vários destinos.

org.jdom2.transform - Ajuda com a transformação XML usando classes JAXP Trax.

org.jdom2.xpath - Suporte para XPath em JDOM2. Por padrão, ele usa a biblioteca de xpath Jaxen.

Para salvar as informações em um documento XML as classes mais importantes são: **XMLOutputter**, **DOMOutputter**, **SAXOutputter**, **StAXEventOutputter**, **StAXStreamOutputter** e **StAXStreamReader**, todas pertencentes ao pacote *org.jdom2.output*. Foi utilizada a classe *XMLOutputter*, que provavelmente é o *outputter* mais útil segundo site do JDOM (HUNTER, 2000). Ele produz o documento JDOM2 como um fluxo de *bytes* que podem ser gravados em um arquivo.

Uma forma de se escrever documentos XML é construir uma árvore DOM, começando pelo elemento raiz e depois adicionando os elementos filhos, e então gerar o documento.

Primeiro criou-se o elemento raiz da aplicação através da classe *Element*, que chamamos de *root*, e passamos o nome do elemento, no caso do projeto que está sendo desenvolvido “*PlanoTrabalhoDocente*”, como pode ser observado na linha 131 da Figura 17. Os demais elementos são criados da mesma forma.

Figura 17: Elemento root da árvore JDOM

```
131 Element root = new Element("PlanoTrabalhoDocente");
132
133 Element dadosPlanoTrabalho = new Element("dadosPlanoTrabalho");
```

Fonte: do autor, 2016

O conteúdo do elemento é inserido através do método *setText*, que recebe o valor do objeto de uma classe, figura 18, que mostra o elemento *semestre* recebendo o valor do objeto *SemestreLetivo* da classe *planoDeTrabalho*.

Figura 18: Elemento recebendo valor

```
133 Element dadosPlanoTrabalho = new Element("dadosPlanoTrabalho");
134
135 Element semestre = new Element("semestreLetivo");
136 semestre.setText(simpleDateFormat.format(this.planoDeTrabalho.getSemestreLetivo()));
137
138 Element tipo = new Element("tipo");
139 tipo.setText(this.planoDeTrabalho.getTipo());
```

Fonte: do autor, 2016

No plano de trabalho docente existem várias atividades e em cada atividade diversos itens. Sendo assim, a aplicação gerou uma lista de objetos da classe *Atividade* onde devem ser incluídos nos elementos para adicionar ao documento. Para isso foi criado uma estrutura de repetição, como é possível verificar na linha 143 da figura 19, que é usada uma lista do objeto *Atividade* para percorrer os objetos e inseri-los a um elemento. O atributo é criado com a classe *Attribute* e inserido pelo método *setAttribute* do elemento, como podem ser vistas nas linhas 146 da figura 19.

Figura 19: Criação de um atributo em uma lista

```

141 Element atividadePlanoTrabalho = new Element("Atividades");
142
143 for(Atividade varAtividade : this.atividades){
144
145     Element tagAtividade = new Element("Atividade");
146     Attribute nomeAtividade = new Attribute("nome", varAtividade.getAtividade());
147     tagAtividade.setAttribute(nomeAtividade);
148
149     for(DescricaoAtividade varDescAtividade: varAtividade.getDescricoesAtividades()){
150         Element descAtividade = new Element("Descricoes");
151         Attribute codigoDescricao = new Attribute("codigo", varDescAtividade.getCodigo());
152         descAtividade.setAttribute(codigoDescricao);
153
154         Element tagDescricao = new Element("Descricao");
155
156         tagDescricao.setText(varDescAtividade.getDescricao());
157
158         Element tagCargaHoraria = new Element("CargaHoraria");
159         tagCargaHoraria.setText(varDescAtividade.getCargaHoraria());
160
161         Element tagOutraInfo = new Element("OutraInformacao");
162         tagOutraInfo.setText(varDescAtividade.getOutraInformacao());
163
164         descAtividade.addContent(tagDescricao);
165         descAtividade.addContent(tagCargaHoraria);
166         descAtividade.addContent(tagOutraInfo);
167
168         tagAtividade.addContent(descAtividade);
169     }
170
171     atividadePlanoTrabalho.addContent(tagAtividade);
172 }

```

Fonte: do autor, 2016

Os elementos filhos são vinculados aos elementos pai pelo método *addContent* do elemento pai, e assim, são vinculados os elementos filhos do elemento raiz, veja a seguir:

Figura 20: Elementos filhos vinculados ao elemento raiz

```

201 root.addContent(dadosPlanoTrabalho);
202 root.addContent(pessoa);
203 root.addContent(servidor);

```

Fonte: do autor, 2016

Depois de criados todos os elementos do documento, criado pela classe *Document*, o objeto *doc* que representa o documento XML. O elemento *root* do documento é adicionado através do método *setRootElement*.

Foi criado um objeto *XMLOutputter* que é responsável por gerar o XML através do método *output*, que recebe como parâmetro o objeto do documento a ser escrito e um *OutputStream* pelo qual será enviado o XML para um arquivo. A classe *Format* é responsável por deixar o documento XML bem formatado, realizando o aninhamento, sem espaços em branco e criando as quebras de linhas entre os elementos, e define a codificação UTF-8.

Figura 21: Criação do arquivo XML

```

205     Document doc = new Document();
206
207     doc.setRootElement(root);
208
209     OutputStream out = new FileOutputStream( new File(this.servidor.getMatriculaSiape()+".xml"));
210
211     XMLOutputter xout = new XMLOutputter();
212
213     xout.setFormat(Format.getPrettyFormat());
214
215     xout.output(doc , out);

```

Fonte: do autor, 2016

O JDOM fornece dois construtores que podem ser utilizados para carregar XML, são eles: as classes *SAXBuilder* e *DOMBuilder*. Utilizamos o *SAXBuilder* porque segundo Todd (2003) ela é mais rápida.

Foi criado o objeto pela classe *SAXBuilder* que é responsável por ler o arquivo e converter em um objeto do tipo *Document* da biblioteca JDOM. É construído o objeto *Document*, através do método *build* do *SAXBuilder*, que recebe o conteúdo do arquivo, que pega o elemento raiz do documento utilizando o método *getRootElement*, como demonstra a figura 22.

Figura 22: Pegando o elemento raiz do documento XML

```

226     SAXBuilder pt = new SAXBuilder();
227
228     Document doc = pt.build(this.servidor.getMatriculaSiape()+".xml");
229
230     Element root = doc.getRootElement();

```

Fonte: do autor, 2016

Utilizando o objeto *root*, é chamado o método *getChildren* que retorna as *tags* filhas, no caso da aplicação *<dadosPlanoTrabalho>*, *<pessoa>* e *<servidor>*, e adiciona a um novo elemento criado pra receber todos os respectivos elementos filhos. Os elementos filhos da tag *<pessoa>* são: *<Nome>*, *<E-mail>* e *<Fone>*, e para receber o conteúdo desse elemento foi usado o método *getChildText*, como demonstra a figura 23.

Figura 23: Recebendo o valor dos elementos do XML

```

233 Element pessoa = root.getChild("pessoa");
234 Element servidor = root.getChild("servidor");
235 Element planoTrabalho = root.getChild("dadosPlanoTrabalho");
236
237 this.servidor.getPessoa().setNome(pessoa.getChildText("Nome"));
238 this.servidor.getPessoa().setEmail(pessoa.getChildText("E-mail"));
239 this.servidor.getPessoa().setFone(pessoa.getChildText("Fone"));
240
241 this.servidor.setMatriculaSiape(servidor.getChildText("siape"));
242
243 this.planoDeTrabalho.setSemestreLetivo(simpleDateFormat.parse(planoTrabalho.getChildText("semestreLetivo")));
244 this.planoDeTrabalho.setTipo(planoTrabalho.getChildText("tipo"));

```

Fonte: do autor, 2016

Uma lista foi criada, como a figura 24 demonstra, para acessar os valores das *tags* <Atividades> e <Descricoes>, que possui uma lista de elementos.

Figura 24: Acesso às lista do documento XML

```

246 List<Element> atividades = planoTrabalho.getChild("Atividades").getChildren();
247
248 for(Element atividade : atividades){
249     if(atividade.getAttributeValue("nome").equals("Aula")){
250         this.atividadeAula.setAtividade(atividade.getAttributeValue("nome"));
251
252         for(Element descricao : atividade.getChildren()){
253             this.descAtividadeAula.setCodigo(descricao.getAttributeValue("codigo"));
254             this.descAtividadeAula.setDescricao(descricao.getChildText("Descricao"));
255             this.descAtividadeAula.setCargaHoraria(descricao.getChildText("CargaHoraria"));
256             this.descAtividadeAula.setOutraInformacao(descricao.getChildText("OutraInformacao"));
257
258             this.atividadeAula.getDescricoesAtividades().add(this.descAtividadeAula);
259             this.descAtividadeAula = new DescricaoAtividade();
260         }
261     }
262     else if(atividade.getAttributeValue("nome").equals("Manutenção do Ensino")){

```

Fonte: do autor, 2016

Dessa forma todas as informações do documento XML são instanciadas pelos seus respectivos objetos e assim podem ser exibidos pela aplicação.

Para acessar e recuperar informações de nós específicos de um documento XML, usamos a sintaxe da linguagem XPath. XPath é uma especificação de linguagem de consulta que é usado para consultar um caminho XML. Para consultar o documento JDOM primeiro é preciso compilar um objeto da classe *XPathExpression* usando a classe *XPathFactory*, ambos do pacote *org.jdom2.xpath*, como demonstrado nas linhas 358 a 363 da figura 25. O código demonstra como realizar uma consulta no documento para encontrar o valor do elemento <semestreLetivo>. Outra forma de fazer o filtro é informando o caminho através nós do documento como demonstra as linhas 364 a 365 da figura 25.

Figura 25: Filtros usando XPath

```

356 XPathFactory xFactory = XPathFactory.newInstance();
357
358 XPathExpression<Element> expr = xFactory.compile("//semestreLetivo", Filters.element());
359 List<Element> result = expr.evaluate(jdomDocument);
360 for (Element resultElement : result) {
361     this.planoDeTrabalho.setSemestreLetivo(simpleDateFormat.parse(resultElement.getValue()));
362 }
363
364 Element firstTitle = xFactory.compile("//servidor/child::siape", Filters.element()).evaluateFirst(jdomDocument);
365 this.servidor.setMatriculaSiape(firstTitle.getValue());

```

Fonte: do autor, 2016

Nesse sessão foi apresentado como criar as tags, inserir valores nelas e salvar em um arquivo XML. Também foi exibido como recuperar essas informações de um arquivo completo ou apenas de uma parte dele. Nos apêndices desse trabalhos estão as imagens das páginas do sistema, a classe responsável pela implementação do código JDOM e um modelo de arquivo XML gerado pelo sistema.

6 RESULTADOS E DISCUSSÃO

A API JDOM facilita trabalhar com documentos XML usando código Java, pois não é apenas escrever em um arquivo, mas poder estruturar o documento seguindo os padrões e diretrizes definidos pelo comitê do consórcio internacional W3C, podendo descrever e armazenar informações, possibilitando realizar busca estruturada de informação baseada em *tags*, em vez de apenas em textos.

A busca por tutoriais permitiu encontrar exemplos que demonstram o funcionamento da API. Como os exemplos mostravam apenas o funcionamento básico, foi preciso aprofundar mais nas pesquisas, trabalhando com atributos multivalorados, relações de hierarquia e listas.

Com a implementação concluída, foi iniciada a etapa de testes e validação. Os testes abordam o armazenamento, consulta e remoção de dados em arquivos XML.

Ao término deste trabalho, pode-se afirmar que os objetivos propostos foram alcançados. O estudo da tecnologia empregada neste trabalho apresentou uma importante ferramenta para a manipulação de documentos XML usando a linguagem Java. São empregadas as principais ferramentas da API JDOM para criar, carregar e filtrar as informações de um arquivo XML. Dessa forma, através desse estudo, procuramos mostrar os recursos da API JDOM para facilitar a manipulação de dados semiestruturado em XML.

Além de entregar uma aplicação web que pode auxiliar a elaboração do plano de trabalho do PEBTT. De acordo com a Diretoria de Gestão de Pessoas o IFTO possui 605 (seiscentos e cinco) professores efetivos divididos em 11 (onze) *campi*, que poderão ser beneficiados com a utilização do sistema, proporcionando mais segurança, economia e agilidade neste processo, atendendo à resolução vigente e facilitando o trabalho dos professores do IFTO.

7 CONSIDERAÇÕES FINAIS

Este projeto demonstra a utilização do XML como ferramenta para manipulação de arquivos. A escolha da linguagem XML e da API se deu em razão que, a XML é uma linguagem eficiente, que permite desenvolver aplicações *Web* flexíveis e manipular os dados localmente. Enquanto que, a utilização da API JDOM permite manipular arquivo XML mais eficientemente e sem a necessidade de trabalhar com banco de dados.

É preciso salientar que se trabalhar com XML torna-se mais trabalhoso, pois as informações devem ser geradas e carregadas por completo, diferentemente de um banco de dados, em que se pode salvar cada informação separadamente. Vale lembrar que arquivos XML é, geralmente, modelados para que pessoas, e não máquinas, os leiam, pois dá importância à apresentação final das informações para o usuário.

De acordo com as pesquisas realizadas, as quais não esgotam o assunto, para trabalhos futuros, propõem-se trabalhar com as classes *SAXBuilder* utilizando para validar documentos XML e o *XSLTransformation*, usado para formatar um documento XM utilizando folhas de estilo.

A classe *SAXBuilder* valida o XML usando o DTD (Document Type Definition) antes da JDOM criar o documento. Uma DTD pode ser associada a um documento XML e define o vocabulário (nomes de tags e atributos) e a estrutura dos documentos, restringindo e deixando claro como deve ser estruturado o documento.

A classe *XSLTransformation* realiza a transformação de um documento XML associando a uma folha de estilo. As folhas de estilo são usadas com sucesso na formatação de documentos HTML, mas também é usado para formatar documentos XML. Os documentos XML fornecem uma maneira de processar os dados, e as folhas de estilo como a XSL (*eXtensible Stylesheet Language*) cuidam da apresentação, transformando e manipulando os dados de forma que eles podem ser facilmente visualizados pelo usuário final.

8 REFERÊNCIAS

- BOOCH, Grady; Rumbaugh, James; Jacobson, Ivar. **UML: guia do usuário**. Rio de Janeiro: Elsevier, 2012. – 12ª reimpressão, 2nd ed. ISBN 978-85-352-1784-1.
- BROOKS-BILSON, Rob. Programação ColdFusion: criando aplicações dinâmicas para Web. Rio de Janeiro: Campus, 2001. ISBN 85352-0888-7.
- DAUM, Berthold; MERTEN, Udo (2002). **Arquitetura de sistemas com XML: conteúdo, processo e apresentação**. 2002. ISBN 85-352-1069-5.
- DEITEL, Paul; Deitel, Harvey. **Java: como programar**. Tradução de Edson Furmankiewicz; revisão técnica Fábio Luis Picelli Lucchini. São Paulo: Pearson Prentice Hall, 2010.
- GRAVES, Mark. **Projeto de Banco de Dados com XML**. tradução Aldair José Coelho Corrêa da Silva; revisão técnica Marcos Jorge. São Paulo: Pearson Education do Brasil, 2003.
- HAROLD, Eliotte Rusty (2002). **Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX – Edition Paperback, 2002**. ISBN-13: 978-0201771862. ISBN-10: 0201771861
- HUNTER J., MCLAUGHLIN B. (2000) “**JDOM API Documentation**”. Disponível em: <http://www.jdom.org/>. Acesso em: 30 de maio de 2016.
- IFTO – Instituto Federal do Tocantins. Resolução nº 74/2013/CONSUP/IFTO, de 16 de dezembro de 2013. **Regulamento dos Regimes de Trabalho, suas Alterações e as Atribuições dos Professores do Ensino Básico, Técnico e Tecnológico do IFTO**. 2013.
- LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientado a objeto e ao desenvolvimento iterativo**. Porto Alegre: Bookman, 2007.
- LOPES, J. C. dos S. (2009). **Políticas públicas para TIC a adoção de soluções abertas e softwares públicos O caso DATAPREV**. Disponível em: <http://portal.dataprev.gov.br/wp-content/uploads/2009/12/JoaoLopes_GestPublica_ENAP_2008_Monografia.pdf> Acesso em 12 de abril de 2014.
- MCLAUGHLIN, B. **Java & XML**. 2 ed. O'Reilly: 2001. ISBN: 0-596-00197-5.
- MCLAUGHLIN, Brett. **Java & XML, 2nd Edition**. O'Reilly, 2001. Second Edition September 2001. ISBN: 0-596-00197-5.
- MOJARRA. **Mojarra JavaServer Faces**. Disponível em: <https://javaserverfaces.java.net/>. Acesso em: 30 de maio de 2016.
- OLIVEIRA, Alex Mariano Costa de. **Sistema de Atendimento ao Vestibulando utilizando Banco de Dados Nativo xml**. Curso de Ciências da Computação. Disponível em : <https://projetos.inf.ufsc.br/> . Acesso em: 12 de março de 2015

ORACLE Corporation. Disponível em: <http://www.oracle.com/technetwork/java/>. Acesso em: 29 de maio de 2016.

PRIMEFACES, Disponível em: <http://www.primefaces.org/>. Acesso em: 30 de maio de 2016.

SANTOS, Celso Alberto Saibel; Passos, Carol; Marques, Manoel (2014). **Tecnologias para Processamento de Documentos XML: Uma Abordagem JAVA**. Disponível em: <http://www.academia.edu/>. Acesso em: 1º de junho de 2015.

SILVA FILHO, M. C. de; Lira Gondim, P. R. de (2011). NCLua SOAP: **Acesso a Web Services em aplicações de TVDi**. Workshop de Computação Aplicada em Governo Eletrônico (WCGE).

SILVA, J. Osmar (2001). **XML: Aplicações Práticas**. São Paulo: Editora ÉRICA , 2001, edição 1. ISBN 85-7194-848-8

SILVEIRA, Paulo e Guilherme. **Java e Orientação a Objetos**. Caelum Ensino e Inovação. Disponível em: <http://www.caelum.com.br>. Acesso em: 27 de maio de 2015.

SILVEIRA, Paulo e Guilherme. **Laboratório Java com Testes, JSF, Web Services e Design Patterns**. Caelum Ensino e Inovação. Disponível em: <http://www.caelum.com.br>. Acesso em: 27 de junho de 2015.

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

TODD, Nick; Szolkowski, Mark. **JavaServer Pages: O Guia do Desenvolvedor**. Tradução de Edson Furmankiewicz. – Rio de Janeiro: Elsevier, 2003 – 3ª reimpressão.

W3Schools. **XPath Tutorial**. Disponível em: <http://www.w3schools.com/xpath/>. Acesso em: 1º de junho de 2015.

9 APÊNDICE

APÊNDICE A – Página de Login do sistema

APÊNDICE B – Página inicial do sistema

APÊNDICE C – Página cadastro 1 - informações do plano de trabalho

APÊNDICE D – Página cadastro 2 - informações do usuário

APÊNDICE E – Página cadastro 3 - informações do das atividades

APÊNDICE F – Página cadastro 4 - informações do extrato

APÊNDICE G – Página exibir

APÊNDICE H – Página impressão

APÊNDICE I – XML gerado pelo sistema

APÊNDICE J – Classe responsável pela implementação do código JDOM

10 ANEXOS

ANEXO A - APÊNDICE A – TABELAS REFERENCIAIS

ANEXO B - APÊNDICE B – PLANO DE TRABALHO DO PEBTT

APÊNDICE A – Página de Login do sistema

Login

Usuário	
Senha	
ENTRAR	

Home Cadastrar Exibir

SISTEMA DE GERAÇÃO DO PLANO DE TRABALHO DO PEBTT

Siape	Semestre Letivo	Editar
1811972	2016/01	EDITAR

Home Cadastrar Exibir

Cadastro Plano de Trabalho do PEBTT

PlanoTrabalho Identificação Atividades Extrato

Plano de Trabalho

Semestre Letivo: 2016/01

Tipo:

☒ Inicial.

☐ Retificador.

→ Next

Home Cadastrar Exibir

Cadastro Plano de Trabalho do PEBTT

PlanoTrabalho Identificação Atividades Extrato

Identificação do Servidor

Nome:	Leysson Muriel Tavares Guimarães	Matrícula SIAPE:	18***72
Regime de Trabalho	Efetivo	Telefone:	639*****12
Jornada de Trabalho	20h	E-mail:	*****m@ifto.edu.br
Situação	Estável		
Campus de Lotação:	Palmas		
Departamento de Lotação:	Coordenação de Informática		

← Back

→ Next

[Home](#) [Cadastrar](#) [Exibir](#)

Cadastro Plano de Trabalho do PEBTT

[PlanoTrabalho](#) [Identificação](#) [Atividades](#) [Extrato](#)

Atividades

Atividade:Aulas

Disciplina:

Cód. do Curso/Turma: C.H.:

Código	Disciplina	Cód. do Curso/Turma	C.H.	Excluir
A1	matemática	2023/2	4	☒
A2	informática	34/3	8	☒
A3	História	509/2	4	☒

Atividade: Manutenção do Ensino

Atividade: Apoio ao Ensino

Atividade: Extensão

Atividade: Pesquisa

Atividade: Gestão

Atividade: Comissão

[Home](#) [Cadastrar](#) [Exibir](#)**Cadastro Plano de Trabalho do PEBTT**[PlanoTrabalho](#) [Identificação](#) [Atividades](#) [Extrato](#)**Extrato do Plano de Trabalho**

Atividade	Quantidade	Carga Horário Semanal
Aulas	3	16
Atividades de Manutenção do Ensino	1	2
Atividades de Apoio ao Ensino	1	6
Atividades de Extensão	2	14
Atividades de Pesquisa	2	12
Atividades de Gestão	1	8
Atividades de Comissão	3	12

[Cadastrar](#)[+ Back](#)

Exibir Plano de Trabalho do PEBTT - [Imprimir Tela](#)

PLANO DE TRABALHO DO PEBTT - SEMESTRE LETIVO: 2016/01			
Tipo: Inicial			
Identificação do Servidor			
Nome: Leysson Muriel Tavares Guimarães			Siape: 1811972
Vínculo: Estável: Carga horária: 20h		Situação: Fone: 63992291912 E-mail: leyssonm@ifto.edu.br	
Campus de Lotação: Campus Palmas			
Departamento de Lotação: Coordenação de Informática			
Atividade: Aula			
Cód.	Disciplina	Cód. do Curso/Turma	C.H.
A1	matemática	2023/2	4
A2	informática	34/3	4
A3	História	509/2	4
Atividade: Manutenção do Ensino			
Código	Disciplina	C.H.	
M1	Revisando provas	2	
Atividade: Apoio ao Ensino			
Código	Disciplina	C.H.	
AP1	Atendimento ao Aluno	2	
Atividade: Extensão			
Código	Disciplina	C.H.	
E1	Arranjos produtivos	2	
E2	Curso de informática	4	
Atividade: Pesquisa			
Código	Disciplina	C.H.	
P1	Projeto de pesquisa	2	
P2	Pesquisa em Arranjos Produtivos Locais	4	
Atividade: Gestão			
Código	Disciplina	C.H.	
G1	Coordenador de Curso	6	
Atividade: Comissão			
Código	Disciplina	Portaria	C.H.
C1	Comissão do PAE	343/2016	2
C2	Comissão do Processo Seletivo	345/2016	2
C3	Comissão de Avaliação	456/2016	2
EXTRATO DO PLANO DE TRABALHO DO PEBTT – SEMESTRE LETIVO: 2016/01			
Atividade	Quantidade	Carga Horário Semanal	
Aulas	3	12	
Atividades de Manutenção do Ensino	1	2	
Atividades de Apoio ao Ensino	1	2	
Atividades de Extensão	2	6	
Atividades de Pesquisa	2	6	
Atividades de Gestão	1	6	
Atividades de Comissão	3	6	
Carga Horária Semanal Total: 40			
_____, _____ de _____ de _____ Local/Data			

APÊNDICE H – Página impressão



MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
REITORIA

APÊNDICE B – PLANO DE TRABALHO DO PEBTT

PLANO DE TRABALHO DO PEBTT - SEMESTRE LETIVO: 2016/01			
Tipo: Inicial			
Identificação do Servidor			
Nome: Leysson Muriel Tavares Guimarães		Siape: 1811972	
Vínculo: Estável: Carga horária: 20h		Situação: Fone: 63992291912 E-mail: leyssonm@ifto.edu.br	
Campus de Lotação: Campus Palmas			
Departamento de Lotação: Coordenação de Informática			
Atividade: Aula			
Cód.	Disciplina	Cód. do Curso/Turma	C.H.
A1	matemática	2023/2	4
A2	informática	34/3	4
A3	História	509/2	4
Atividade: Manutenção do Ensino			
Código	Disciplina	C.H.	
M1	Revisando provas	2	
Atividade: Apoio ao Ensino			
Código	Disciplina	C.H.	
AP1	Atendimento ao Aluno	2	
Atividade: Extensão			
Código	Disciplina	C.H.	
E1	Arranjos produtivos	2	
E2	Curso de informática	4	
Atividade: Pesquisa			
Código	Disciplina	C.H.	
P1	Projeto de pesquisa	2	
P2	Pesquisa em Arranjos Produtivos Locais	4	
Atividade: Gestão			
Código	Disciplina	C.H.	
G1	Coordenador de Curso	6	
Atividade: Comissão			
Código	Disciplina	Portaria	C.H.
C1	Comissão do PAE	343/2016	2
C2	Comissão do Processo Seletivo	345/2016	2
C3	Comissão de Avaliação	456/2016	2
EXTRATO DO PLANO DE TRABALHO DO PEBTT – SEMESTRE LETIVO: 2016/01			
Atividade	Quantidade	Carga Horário Semanal	
Aulas	3	12	
Atividades de Manutenção do Ensino	1	2	
Atividades de Apoio ao Ensino	1	2	
Atividades de Extensão	2	6	
Atividades de Pesquisa	2	6	
Atividades de Gestão	1	6	
Atividades de Comissão	3	6	
Carga Horária Semanal Total: Fri Jan 01 00:00:00 GMT-03:00 2016			
_____, _____ de _____ de _____ Local/Data			

```
<?xml version="1.0" encoding="UTF-8"?>
<PlanoTrabalhoDocente>
  <dadosPlanoTrabalho>
    <semestreLetivo>2016/01</semestreLetivo>
    <tipo>Inicial</tipo>
    <Atividades>
      <Atividade nome="Aula">
        <Descricoes codigo="A1">
          <Descricao>matemática</Descricao>
          <CargaHoraria>4</CargaHoraria>
          <OutraInformacao>2023/2</OutraInformacao>
        </Descricoes>
        <Descricoes codigo="A2">
          <Descricao>informática</Descricao>
          <CargaHoraria>4</CargaHoraria>
          <OutraInformacao>34/3</OutraInformacao>
        </Descricoes>
        <Descricoes codigo="A3">
          <Descricao>História</Descricao>
          <CargaHoraria>4</CargaHoraria>
          <OutraInformacao>509/2</OutraInformacao>
        </Descricoes>
      </Atividade>
      <Atividade nome="Manutenção do Ensino">
        <Descricoes codigo="M1">
          <Descricao>Revisando provas</Descricao>
          <CargaHoraria>2</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
      </Atividade>
      <Atividade nome="Apoio ao Ensino">
        <Descricoes codigo="AP1">
          <Descricao>Atendimento ao Aluno</Descricao>
          <CargaHoraria>2</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
      </Atividade>
      <Atividade nome="Extensão">
        <Descricoes codigo="E1">
          <Descricao>Arranjos produtivos</Descricao>
          <CargaHoraria>2</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
        <Descricoes codigo="E2">
          <Descricao>Curso de informática</Descricao>
          <CargaHoraria>4</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
      </Atividade>
      <Atividade nome="Pesquisa">
        <Descricoes codigo="P1">
          <Descricao>Projeto de pesquisa</Descricao>
          <CargaHoraria>2</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
        <Descricoes codigo="P2">
          <Descricao>Pesquisa em Arranjos Produtivos Locais</Descricao>
          <CargaHoraria>4</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
      </Atividade>
      <Atividade nome="Gestão">
        <Descricoes codigo="G1">
          <Descricao>Coordenador de Curso</Descricao>
          <CargaHoraria>6</CargaHoraria>
          <OutraInformacao />
        </Descricoes>
      </Atividade>
      <Atividade nome="Comissão">
        <Descricoes codigo="C1">
          <Descricao>Comissão do PAE</Descricao>
        </Descricoes>
      </Atividade>
    </Atividades>
  </dadosPlanoTrabalho>
</PlanoTrabalhoDocente>
```

```
<CargaHoraria>2</CargaHoraria>
<OutraInformacao>343/2016</OutraInformacao>
</Descricoes>
<Descricoes codi go="C2">
  <Descricao>Comiss2o do Processo Seletivo</Descricao>
  <CargaHoraria>2</CargaHoraria>
  <OutraInformacao>345/2016</OutraInformacao>
</Descricoes>
<Descricoes codi go="C3">
  <Descricao>Comiss2o de Avalia2o</Descricao>
  <CargaHoraria>2</CargaHoraria>
  <OutraInformacao>456/2016</OutraInformacao>
</Descricoes>
</Atividade>
</Atividades>
</dadosPlanoTrabalho>
<peessoa>
  <Nome>Leysson Muriel Tavares Guimar2es</Nome>
  <Fone>63992291912</Fone>
  <E-mail>leyssonm@fio.edu.br</E-mail>
</peessoa>
<servidor>
  <si ape>1811972</si ape>
  <vinculo>Estavel</vinculo>
  <jornadaTrabalho>20h</jornadaTrabalho>
  <situacao />
  <lotacao>Campus Palmas</lotacao>
  <departamento>Coordena2o de Inform2ica</departamento>
</servidor>
</PlanoTrabalhoDocente>
```

PlanoDeTrabalhoDAO.java

```

1 package br.com.dao;
2
3 import java.io.File;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.OutputStream;
7 import java.text.ParseException;
8 import java.text.SimpleDateFormat;
9 import java.util.List;
10
11 import org.jdom2.Attribute;
12 import org.jdom2.Document;
13 import org.jdom2.Element;
14 import org.jdom2.JDOMException;
15 import org.jdom2.filter.Filters;
16 import org.jdom2.input.SAXBuilder;
17 import org.jdom2.output.Format;
18 import org.jdom2.output.XMLOutputter;
19 import org.jdom2.xpath.XPathExpression;
20 import org.jdom2.xpath.XPathFactory;
21
22 import br.com.dominio.Atividade;
23 import br.com.dominio.DescricaoAtividade;
24 import br.com.dominio.PlanoDeTrabalho;
25
26 public class PlanoDeTrabalhoDAO {
27
28     SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy/MM");
29
30     public void criarXML(PlanoDeTrabalho planoDeTrabalho) throws IOException, JDOMException{
31
32         Element root = new Element("PlanoTrabalhoDocente");
33
34         Element dadosPlanoTrabalho = new Element("dadosPlanoTrabalho");
35
36         Element semestre = new Element("semestreLetivo");
37         semestre.setText(simpleDateFormat.format(planoDeTrabalho.getSemestreLetivo()));
38
39         Element tipo = new Element("tipo");
40         tipo.setText(planoDeTrabalho.getTipo());
41
42         Element atividadePlanoTrabalho = new Element("Atividades");
43
44         for(Atividade varAtividade : planoDeTrabalho.getAtividades()){
45
46             Element tagAtividade = new Element("Atividade");
47             Attribute nomeAtividade = new Attribute("nome", varAtividade.getAtividade());
48             tagAtividade.setAttribute(nomeAtividade);
49
50             for(DescricaoAtividade varDescAtividade: varAtividade.getDescricoesAtividades()){
51                 Element descAtividade = new Element("Descricoes");
52                 Attribute codigoDescricao = new Attribute("codigo", varDescAtividade.getCodigo());
53                 descAtividade.setAttribute(codigoDescricao);
54
55                 Element tagDescricao = new Element("Descricao");
56
57                 tagDescricao.setText(varDescAtividade.getDescricao());
58
59                 Element tagCargaHoraria = new Element("CargaHoraria");
60                 tagCargaHoraria.setText(varDescAtividade.getCargaHoraria());
61
62                 Element tagOutraInfo = new Element("OutraInformacao");
63                 tagOutraInfo.setText(varDescAtividade.getOutraInformacao());
64

```

PlanoDeTrabalhoDAO.java

```

65         descAtividade.addContent(tagDescricao);
66         descAtividade.addContent(tagCargaHoraria);
67         descAtividade.addContent(tagOutraInfo);
68
69         tagAtividade.addContent(descAtividade);
70     }
71
72     atividadePlanoTrabalho.addContent(tagAtividade);
73 }
74
75 dadosPlanoTrabalho.addContent(semestre);
76 dadosPlanoTrabalho.addContent(tipo);
77 dadosPlanoTrabalho.addContent(atividadePlanoTrabalho);
78
79 Element pessoa = new Element("pessoa");
80
81 Element nome = new Element("Nome");
82 nome.setText(planoDeTrabalho.getServidor().getPessoa().getNome());
83
84 Element fone = new Element("Fone");
85 fone.setText(planoDeTrabalho.getServidor().getPessoa().getFone());
86
87 Element email = new Element("E-mail");
88 email.setText(planoDeTrabalho.getServidor().getPessoa().getEmail());
89
90 pessoa.addContent(nome);
91 pessoa.addContent(fone);
92 pessoa.addContent(email);
93
94 Element servidor = new Element("servidor");
95
96 Element siape = new Element("siape");
97 siape.setText(planoDeTrabalho.getServidor().getMatriculaSiape());
98
99 Element vinculo = new Element("vinculo");
100 vinculo.setText(planoDeTrabalho.getServidor().getVinculo());
101
102 Element regime = new Element("regimeTrabalho");
103 regime.setText(planoDeTrabalho.getServidor().getRegimeTrabalho());
104
105 Element jornada = new Element("jornadaTrabalho");
106 jornada.setText(planoDeTrabalho.getServidor().getJornadaTrabalho());
107
108 Element situacao = new Element("situacao");
109 situacao.setText(planoDeTrabalho.getServidor().getSituacao());
110
111 Element lotacao = new Element("lotacao");
112 lotacao.setText(planoDeTrabalho.getServidor().getDepartamento().getCampus().getCampus());
113
114 Element departamento = new Element("departamento");
115 departamento.setText(planoDeTrabalho.getServidor().getDepartamento().getDepartamento());
116
117 servidor.addContent(siape);
118 servidor.addContent(vinculo);
119 servidor.addContent(regime);
120 servidor.addContent(jornada);
121 servidor.addContent(situacao);
122 servidor.addContent(lotacao);
123 servidor.addContent(departamento);
124
125 root.addContent(dadosPlanoTrabalho);
126 root.addContent(pessoa);
127 root.addContent(servidor);
128

```

```

129         Document doc = new Document();
130
131         doc.setRootElement(root);
132
133         OutputStream out = new FileOutputStream( new
File(planoDeTrabalho.getServidor().getMatriculaSiape()+".xml"));
134
135         XMLOutputter xout = new XMLOutputter();
136
137         xout.setFormat(Format.getPrettyFormat());
138
139         xout.output(doc , out);
140
141     }
142
143     public PlanoDeTrabalho lerXML(String arquivo) throws JDOMException, IOException, ParseException{
144
145         SAXBuilder pt = new SAXBuilder();
146
147         Document doc = pt.build(arquivo+".xml");
148
149         Element root = doc.getRootElement();
150
151         Element pessoa = root.getChild("pessoa");
152         Element servidor = root.getChild("servidor");
153         Element planoTrabalho = root.getChild("dadosPlanoTrabalho");
154
155         PlanoDeTrabalho planoDeTrabalho = new PlanoDeTrabalho();
156
157         planoDeTrabalho.getServidor().getPessoa().setNome(pessoa.getChildText("Nome"));
158         planoDeTrabalho.getServidor().getPessoa().setEmail(pessoa.getChildText("E-mail"));
159         planoDeTrabalho.getServidor().getPessoa().setFone(pessoa.getChildText("Fone"));
160
161         planoDeTrabalho.getServidor().setMatriculaSiape(servidor.getChildText("siape"));
162         planoDeTrabalho.getServidor().setVinculo(servidor.getChildText("vinculo"));
163         planoDeTrabalho.getServidor().setRegimeTrabalho(servidor.getChildText("regimeTrabalho"));
164         planoDeTrabalho.getServidor().setJornadaTrabalho(servidor.getChildText("jornadaTrabalho"));
165         planoDeTrabalho.getServidor().setSituacao(servidor.getChildText("situacao"));
166
167         planoDeTrabalho.getServidor().getDepartamento().getCampus().setCampus(servidor.getChildText("lotacao"));
168
169         planoDeTrabalho.getServidor().getDepartamento().setDepartamento(servidor.getChildText("departamento"));
170
171         planoDeTrabalho.setSemestreLetivo(simpleDateFormat.parse(planoTrabalho.getChildText("semestreLetivo")));
172         planoDeTrabalho.setTipo(planoTrabalho.getChildText("tipo"));
173
174         List<Element> atividades = planoTrabalho.getChild("Atividades").getChildren();
175
176         for(Element atividade : atividades){
177
178             Atividade dadosatividade = new Atividade();
179
180             dadosatividade.setAtividade(atividade.getAttributeValue("nome"));
181
182             for(Element descricao : atividade.getChildren()){
183
184                 DescricaoAtividade descricaoAtividade = new DescricaoAtividade();
185
186                 descricaoAtividade.setCodigo(descricao.getAttributeValue("codigo"));
187                 descricaoAtividade.setDescricao(descricao.getChildText("Descricao"));
188                 descricaoAtividade.setCargaHoraria(descricao.getChildText("CargaHoraria"));
189
190                 descricaoAtividade.setOutraInformacao(descricao.getChildText("OutraInformacao"));
191             }
192         }
193     }

```

```
188
189         dadosatividade.getDescricoesAtividades().add(descricaoAtividade);
190
191     }
192     planoDeTrabalho.getAtividades().add(dadosatividade);
193 }
194
195     return planoDeTrabalho;
196 }
197
198 public PlanoDeTrabalho filtrar(String arquivo) throws JDOMException, IOException, ParseException{
199
200     PlanoDeTrabalho planoDeTrabalho = new PlanoDeTrabalho();
201
202     SAXBuilder jdomBuilder = new SAXBuilder();
203     Document jdomDocument = jdomBuilder.build(arquivo+".xml");
204
205     XPathFactory xFactory = XPathFactory.instance();
206
207     XPathExpression<Element> expr = xFactory.compile("//semestreLetivo", Filters.element());
208     List<Element> result = expr.evaluate(jdomDocument);
209     for (Element resultElement : result) {
210         planoDeTrabalho.setSemestreLetivo(simpleDateFormat.parse(resultElement.getValue()));
211     }
212
213     Element firstTitle = xFactory.compile("//servidor/child::siape",
Filters.element()).evaluateFirst(jdomDocument);
214     planoDeTrabalho.getServidor().setMatriculaSiape(firstTitle.getValue());
215
216     return planoDeTrabalho;
217 }
218 }
219
220 }
221
```

ANEXO A - APÊNDICE A – TABELAS REFERENCIAIS



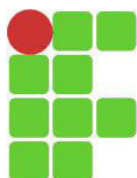
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
REITORIA

APÊNDICE A – TABELAS REFERENCIAIS

Distribuição da Carga Horária do PEBTT – Regime de Trabalho Tempo Parcial (20 h/Semana)								
Atividades Didáticas						Atividades Administrativas		Total
Ensino					Pesquisa e Extensão	Gestão	Comissão	
Aulas/Semana	Manutenção do Ensino	Apoio ao Ensino						
		Reunião	Atendimento	Outras Atividades				
8	3	2	02 -04	1 – 5	0 - 4	*	0 - 4	20h
9	3	2	02 - 04	1 – 4	0 - 3	*	0 - 3	20h
10	3	2	02 -04	1 – 3	0 - 2	*	0 - 2	20h
11	4	2	02 - 03	1	*	*	*	20h
12	4	2	2	*	*	*	*	20h

* → não se aplica.

Distribuição da Carga Horária do PEBTT – Regime de Trabalho Tempo Integral e Dedicção Exclusiva (40 h/Semana)								
Atividades Didáticas						Atividades Administrativas		Total
Ensino					Pesquisa e Extensão	Gestão	Comissão	
Aulas/Semana	Manutenção do Ensino	Apoio ao Ensino						
		Reunião	Atendimento	Outras Atividades				
8	4	2	02 -04	2 – 24	0 - 22	0 - 22	0 – 22	40h
9	5	2	02 - 04	2 – 22	0 - 20	0 - 20	0 – 20	40h
10	5	2	02 -04	2 – 21	0 - 19	0 - 19	0 – 19	40h
11	6	2	02 - 04	2 – 19	0 - 17	0 - 17	0 – 17	40h
12	6	2	02 - 04	2 – 18	0 - 16	0 - 16	0 – 16	40h
13	7	2	02 - 06	2 – 16	0 - 14	0 - 14	0 – 14	40h
14	7	2	02 - 06	2 – 15	0 - 13	0 - 13	0 – 13	40h
15	8	2	02 - 06	2 – 13	0 - 11	0 - 11	0 – 11	40h
16	8	2	02 - 06	2 – 12	0 - 10	0 - 10	0 – 10	40h
17	9	2	02 - 06	2 – 10	0 - 8	0 - 8	0 – 8	40h
18	9	2	02 - 06	2 – 09	0 - 7	0 - 7	0 – 7	40h
19	10	2	02 - 06	2 – 7	0 - 5	0 - 5	0 – 5	40h
20	10	2	02 - 06	2 – 6	0 - 4	0 - 4	0 – 4	40h



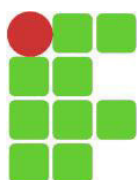
Av. Joaquim Teotônio Segurado
 Quadra 202 Sul, Conjunto 01, Lote 08, Centro
 77.020-450, Palmas - TO
 (063) 3229-2200
reitoria@ifto.edu.br - www.ifto.edu.br



MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
REITORIA

APÊNDICE B – PLANO DE TRABALHO DO PEBTT

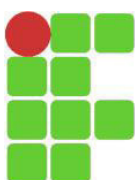
PLANO DE TRABALHO DO PEBTT – SEMESTRE LETIVO: _____			
Tipo: () Inicial. () Retificador. Data do último plano de trabalho: __/__/__			
Identificação do Servidor			
Nome:			Matrícula SIAPE:
Efetivo: 20h ☐	40h ☐	DE ☐	Situação: Estável ☐ Em Estágio Probatório ☐
Substituto: 20h ☐	40h ☐	Fone:	
Temporário: 20h ☐	40h ☐	E-mail:	
Campus de Lotação:			
Departamento de Lotação:			
Atividade: Aulas			
Cód.	Disciplina	Cód. do Curso/Turma	C.H.
A1			
A2			
A3			
A4			
A5			
A6			
A7			
A8			
A9			
A10			
Atividade: Manutenção do Ensino			
Cód.	Descrição da Atividade	C.H.	
M1			
M2			
M3			
M4			
M5			
M6			
M7			
Atividade: Apoio ao Ensino			
Cód.	Descrição da Atividade	C.H.	
AP1			
AP2			
AP3			
AP4			





MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
REITORIA

AP5			
Atividade: Extensão			
Cód.	Descrição da Atividade	C.H.	
E1			
E2			
E3			
Atividade: Pesquisa			
Cód.	Descrição da Atividade	C.H.	
P1			
P2			
P3			
Atividade: Gestão			
Cód.	Descrição da Atividade	C.H.	
G1			
G2			
Atividade: Comissão			
Cód.	Descrição da Atividade	Portaria	C.H.
C1			
C2			
C3			
C4			
C5			
EXTRATO DO PLANO DE TRABALHO DO PEBTT – SEMESTRE LETIVO: _____			
Atividades	Quantidade	Carga Horária Semanal	
Aulas			
Atividades de Manutenção do Ensino			
Atividades de Apoio ao Ensino			
Atividades de Extensão			
Atividades de Pesquisa			
Atividades de Gestão			
Atividades de Comissão			
Carga Horária Semanal Total:			
_____, ____ de _____ de _____ <i>Local/Data</i>			





**MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
REITORIA**

<i>Assinatura do Servidor</i>	<i>Assinatura e Carimbo da Chefia Imediata</i>
-------------------------------------	--

