Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUCIANO RODRIGUES CATAPAN

PADRÕES DE PROJETO, SEGURANÇA DE DADOS COM JAAS E INJEÇÃO DE DEPENDÊNCIAS EM APLICATIVOS WEB

Assis

2017

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUCIANO RODRIGUES CATAPAN

PADRÕES DE PROJETO, SEGURANÇA DE DADOS COM JAAS E INJEÇÃO DE DEPENDÊNCIAS EM APLICATIVOS WEB

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de graduação.

Orientando: Luciano Rodrigues Catapan

Orientador(a):Domingos de Carvalho Villela Junior

Assis/SP

2017

FICHA CATALOGRÁFICA

CATAPAN, Luciano Rodrigues

Padrões de projetos, segurança de dados com jaas e injeção de dependências. Fundação educacional do Município de Assis – FEMA – Assis, 2017

45p

Orientador: Domingos de Carvalho Villela Junior

Trabalho de Conclusão de curso – Instituto Municipal de Ensino Superior de Assis - IMESA

1.JAVA. 2.JAAS. 3.CDI. 4.Padrões de projetos

CDD:001.61

Biblioteca da FEMA

PADRÕES DE PROJETO, SEGURANÇA DE DADOS COM JAAS E INJEÇÃO DE DEPENDÊNCIAS EM APLICATIVOS WEB

LUCIANO RODRIGUES CATAPAN

Trabalho de conclusão de curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação em Ciência da Computação analisado pela seguinte comissão examinadora

Orientador:_____

Analisador(1):_____

Assis/SP

2017

Dedicatória

Dedico este trabalho a Deus, que sempre me ajudou nos momentos difíceis através de suas bênçãos; dedico também este trabalho a minha família que sempre me apoiou muito, a minha namorada que sempre esteve comigo mesmo não podendo dar toda atenção devido ao pouco tempo livre que tenho.

AGRADECIMENTOS

Agradeço ao meu orientador Prof.º Domingos de Carvalho Villela Junior, que sempre me ajudou e me orientou de modo que pudesse terminar este trabalho.

Agradeço a todos os professores que ao longo desta jornada puderam me auxiliar e me ensinar da melhor forma tudo que me era e é necessário para uma carreira de sucesso na área de TI.

Agradeço aos meus colegas de sala e de faculdade que sempre me animaram para nunca desistir de meus sonhos.

"Eu Acredito, que ás vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar."

Alan Turing (1912 - 1954)

Resumo

A necessidade de otimizar um projeto através de baixo acoplamento vem crescendo muito nos últimos anos, bem como a necessidade de encontrar soluções definitivas para erros recorrentes em programas orientados a objetos e a confiabilidade da segurança dos dados em um sistema. Com injeção de dependências pode-se diminuir o nível de acoplamento em um sistema diminuindo os riscos de erro em uma possível reutilização do código e unindo seu conceito com padrões de projetos pode-se alcançar melhores resultados quanto a reutilização e diminuição de erros, além de que Padrões de projetos podem solucionar problemas recorrentes em programas. Garantindo a segurança dos dados em um sistema, pode-se aumentar o nível de confiabilidade passado ao cliente em um devido sistema ou aplicativo.

Palavras-chaves: Injeção de Dependências, segurança de dados com Jaas, Padrões de Projetos.

ABSTRACT

The need to optimize a project through low coupling has been growing a lot in recent years, as well as the need to find definitive solutions to recurring errors in object-oriented programs and the reliability of data security in a program. With dependency injection, we can reduce the level of coupling in a program by reducing the risks of error in a possible reuse of the code and joining its concept with Design Patterns, we can achieve better results regarding the reuse and reduction of errors, besides that Design Patterns can solve recurring problems in programs. By ensuring data security in a program, we can increase the level of reliability passed on to the client in a proper system or program.

Keywords: injection dependency, Design Patterns, Data security with Jaas.

Lista de Ilustrações

LISTA DE TABELAS

Sumário

1. Introdução

Com o avanço da tecnologia nos dias atuais, as empresas vêm investindo em novas tecnologias cada vez mais seguras e de alta confiabilidade na taxa de precisão em evitar erros e perdas de dados, que pode aumentar a qualidade do software e também sua eficácia e segurança. Será abordado neste projeto o uso de algumas dessas tecnologias e como elas influenciam em projetos de sistemas computacionais, obtendo uma segurança de dados confiável e utilizando de mecanismo e técnicas de padrões de projeto e padrões de desenvolvimentos, visando como objetivo principal a criação de um aplicativo web utilizando de todos os temas abordados.

A segurança da informação e dados das empresas tem sido motivo de preocupação por parte de muitos empresários, que se questionam sobre a segurança que o ambiente virtual fornece contra-ataques de vírus e pessoas maliciosas.

De acordo com MAIA(2013) o tema Segurança da Informação vem captando muita atenção desde executivos e gerentes até técnicos. Fato decorrente por que a segurança da informação engloba diversos segmentos, quais são: a infraestrutura tecnológica, aplicações e conscientização organizacional, segurança física, sendo que cada uma contém seus riscos, ameaças a integridade do usuário, sendo o objetivo maior da segurança da informação garantir a integridade dos dados destas empresas.

Ainda existe muita vulnerabilidade em sistemas empresariais que ainda não conseguem fazer a sua segurança de dados de maneira eficiente, Não podendo passar confiança ao cliente e ao empresário de que seus dados estão 100% seguros.

Websense, Inc.(2014) destaca em um artigo os resultados de um levantamento global realizado pelo Instituto Ponemon “Obstáculos, Renovação e Aumento da Educação em Segurança”, que existe um desafio entre empresários e profissionais da área de TI em reformular os atuais sistemas de segurança de dados e tentar elevar o conhecimento limitado de muitos funcionários sobre a importância da segurança de dados.

Devido à grande demanda de empresas buscando cada vez mais a segurança de seus dados e dados de seus clientes, será visado neste projeto a melhor forma de proteger dados utilizando Jaas, pretendendo passar adiante o resultado obtido sobre o tema Segurança da Informação.

Outro tema que será abordado é que as empresas de software também compartilham de um problema em comum, que é a dificuldade em reaproveitar específicos projetos, códigos- fonte e protótipos antigos aos sistemas orientados a objetos.

Devido a esta complicação, uma solução geral de como resolver este problema em diversas ocasiões começou a ser desenvolvida pelos programadores Kent Beck e Ward Cunningham em 1987, tendo como base as ideias de padrões de projeto de Christopher Alexander (1977/1979 Notes on the Synthesis of Form, The Timeless Way of Building e A Pattern Language), propuseram os primeiros padrões de projeto na área da ciência da computação.

Atualmente as empresas desenvolvedoras de software tem utilizado muito os desígnios dos Padrões GoF ('*Gang of Four*').

Neste Trabalho de Conclusão de Curso será estudado o tema de padrões de projetos, especificamente o padrão Singleton, buscando a ciência do assunto e mostrando opções e métodos eficazes que visam solucionar os problemas das empresas na reciclagem de antigos softwares e na melhor forma de se manter e estruturar um aplicativo web, para o caso de uma necessidade em se reutilizar alguma parte do mesmo em outro(s) software(s).

Outro ponto que será mostrado neste Trabalho de Conclusão de Curso é a solução do problema de alto acoplamento de classes e módulos em aplicativos web; podemos dizer que um projeto está altamente acoplado quando suas classes e módulos conhecem muito um do outro, criando uma dependência desnecessária que pode levar a erros e problemas caóticos ao sistema, gerando dificuldades à empresa e a seus desenvolvedores.

A solução de tal problema pode ser encontrada na utilização de injeção de Dependências, a tecnologia CDI: “O Contexto e a Injeção de Dependência ou *Contexts and Dependency Injection* (CDI) está definido na especificação JSR 346 que pode ser baixada no site da Oracle. A especificação CDI basicamente define um mecanismo seguro de injeção de dependência na plataforma Java EE”. (MEDEIROS, 2015).

1.1 OBJETIVOS GERAIS

Um dos objetivos deste Trabalho de Conclusão de Curso é possibilitar o uso do mesmo como forma de auxílio a comunidade JAVA, empresas e desenvolvedores voltados a aplicativos web sobre os temas abordados.

Outro objetivo é abranger o conhecimento nos temas abordados, visando alavancar uma possível carreira na área de TI.

1.2 OBJETIVOS ESPECÍFICOS

Explorar os conceitos e recursos de Injeção de Dependências, explorar os conceitos e recursos de Segurança da Informação e explorar os conceitos e recursos de Padrões de Projetos.

1.3 JUSTIFICATIVA DO TEMA

Segurança de dados, injeção de dependências e padrões de projetos vem sendo requisitados por praticamente todas as empresas de softwares no Brasil e no mundo, sendo sempre bom ter a ciência de novas tecnologias disponíveis no mercado. Será buscado neste Trabalho de Conclusão do Curso abranger os conhecimentos em segurança de dados, injeção de dependências e padrões de projeto.

O tema também foi abordado devido à necessidade de aprender o conteúdo citado neste Trabalho de Conclusão do Curso, já que um sistema utilizando-se de todas as tecnologias citadas será desenvolvido, tendo base nos temas abordados.

1.4 MOTIVAÇÃO

A motivação em se seguir com este projeto é devido à necessidade de ter a ciência dos temas abordados para realização de um aplicativo web à instituição Centro Paula Souza; todos os temas que serão abordados neste Trabalho de Conclusão de Curso são geralmente requisitados em entrevistas de emprego e utilizados em grande parcela de empresas que desenvolvem com a linguagem JAVA, que utilizam do conceito de segurança dos dados com Jaas, injeção de dependências e a necessidade de ter um padrão de projeto para evitar complicações futuras.

1.5 PERSPECTIVA DE CONTRIBUIÇÃO

Pretende-se contribuir com a comunidade de desenvolvedores JAVA para com os aplicativos web, utilizando os estudos feitos sob os levantamentos bibliográficos, utilizando os conceitos abordados para a realização de um aplicativo web, pretendendo deixar o trabalho com o suficiente para que se entenda como realizar os mesmos procedimentos realizados neste trabalho de conclusão de curso e na análise do sistema.

Pretende-se contribuir ao Centro Paula Souza de Assis Etec Pedro D'Arcádia Neto com um sistema de atendimento provido deste Trabalho.

1.6 METODOLOGIA DE PESQUISA

Será realizado um levantamento bibliográfico sobre os temas abordados neste Trabalho de Conclusão do Curso, utilizando-se de livros referentes aos temas de segurança de dados, CDI e padrões de projetos e etc.

Também haverá uso de caso dos segmentos retirados do estudo destas bibliografias para a realização de um aplicativo web.

Haverá também uma pesquisa experimental, usando da técnica Agile voltado ao método Scrum, cujo processo será a forma que conduziremos o software em seus estágios de desenvolvimento.

Os dados a serem obtidos para a análise de requisitos serão obtidos através de pesquisa qualitativa, sendo que uma entrevista será realizada para a obtenção desses dados.

1.7 FERRAMENTAS PARA DESENVOLVIMENTO DO SISTEMA

JPA**:** Java Persistence API (ou simplesmente JPA) é uma API padrão da linguagem JAVA, que descreve uma interface comum para a persistência de dados. Permite ao usuário manipular os dados de um banco relacional, poupando-lhe tempo, por exemplo (FILLITO, 2015).

JSF**:** Java Server Faces (JSF) é uma especificação JAVA para a montagem de interfaces gráficas (templates) de usuário, inspirado em componentes para aplicações web (BUENO, 2013)

Hibernate**:** O Hibernate visa melhorar a relação do usuário a criação e manipulação de tabelas de dados em um banco de dados relacional, permitindo ao usuário maior iteração em menos espaço de tempo (SOUZA, 2012).

Prime Faces**:** Prime Faces é uma suíte de código aberto de componentes para Java Server Faces que otimizam a interface gráfica de seus templates (PILARI, 2016)

Tomcat: O servidor Apache Tomcat é um container Web de código fonte aberto baseado em Java que foi criado para executar aplicações Web que utilizam tecnologias Servlets e JSPs (VUKOTIC, GOODWILL, 2011).

2. SEGURANÇA DE DADOS

A expressão segurança de dados refere-se à proteção de um dado contra perdas e riscos, e atualmente, o bem mais valioso para uma empresa é a informação. Ter certeza que pessoas maliciosas não obtenham esses dados torna-se cada dia mais crítico, já que os dados são recursos estratégicos para muitas empresas. Por causa disso a segurança de dados tornou-se cada vez mais difundida.

Segurança da informação está relacionado ao fato de proteger um grupo de dados de uma empresa ou pessoa independente do seu valor, se é em dinheiro ou se é de valor sentimental.

A tecnologia de segurança em Java possui um grande grupo de ferramentas, APIs, protocolos e mecanismos de segurança. Incluem criptografia, comunicação, autenticação, infraestrutura da chave pública e controle de acesso.

2.1 INTRODUÇÃO AO JAAS

A tecnologia Java contém uma grande quantidade de APIs, implementações de algoritmos, mecanismos de utilização e protocolos e ferramentas.

O JAAS (Java Authentication and Authorization Service) é um adjacente de APIs que buscam desacoplar as aplicações de controle de acesso a recursos da mesma, ou seja, retirar do desenvolvedor a obrigação de ter que controlar o acesso a recurso por aspectos de maneira programática. A autenticação JAAS é extensível e manejada de forma coesa. Isso permite que aplicações JAVA continuem autônomas de tecnologias de autenticação básica (ANDRADE, 2010).

JAAS pode ser usado para dois desígnios:

1. De forma segura validar um usuário e definir qual usuário e qual a sua "*role"* está atualmente em execução no código JAVA.

2. Para a autorização dos usuários, admitindo que eles tenham direitos de controle de acesso necessárias para realizar as ações desejadas.

2.2 CONFIGURANDO O JAAS NO BANCO DE DADOS

Para demonstrar JAAS trabalhando, será implementada a segurança no sistema de Atendimento. Será preciso algumas tabelas no banco de dados para guardar o usuário, senha e as permissões(“*roles”*) de cada usuário. O modelo de dados será bastante simples.

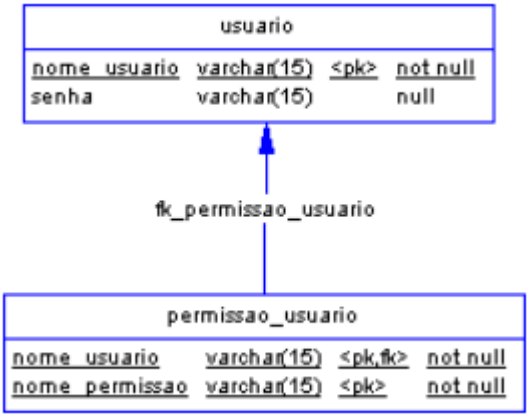

Figura 1: Diagrama E.R. JAAS

A tabela “usuário” guarda o nome e a senha do usuário e a tabela “permissao_usuario” relaciona o usuário já cadastrado a um nome de permissão(“*role*”).

2.3 CONFIGURANDO O JAAS NO ECLIPSE

A autenticação usando o JAAS é dinâmica, pois ela é feita através de *Login Modules*. *Login Module* é uma interface que é implementada por classes que examinam o usuário e senha no banco de dados ou no tomcat-user.xml e concretizam as regras de autenticação. Os containers web implementam alguns módulos de login básicos para autenticação a partir de arquivos de propriedades ou XML contendo os nomes e permissões dos usuários, banco de dados, etc. (ANDRADE, 2010).

O Tomcat implementa esses módulos de login em forma de domínios (*Realms*). Domínios refere-se a um conjunto de usuários e senhas que identificam usuários válidos para uma aplicação web, mais uma lista de permissões (roles) associadas para cada usuário.

Para a aplicação do sistema de Atendimento, será utilizado o *realm JDBCRealm*, que utiliza autenticação através de bancos de dados relacionais, acessada via JDBC. Para configurar o *JDBCRealm*, será necessário editar o arquivo “server.xml”. Se estiver iniciado o Tomcat isoladamente, este arquivo se encontra na pasta “conf.” do container, mas se estiver iniciado o Tomcat integrado ao Eclipse, deve-se encontrar o arquivo na pasta “Servers” do *workspace*.

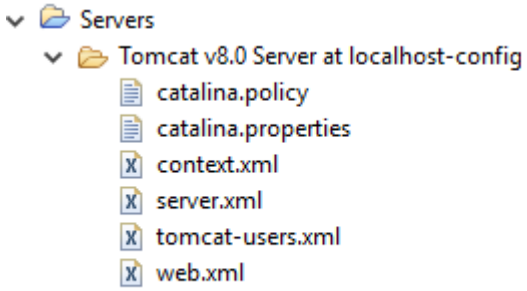

Figura 2: Tomcat server.xml

Deverá localizar o elemento Host, onde *appBase* é igual a “*webapps*” e deverá ser inserido a tag Realm com todos os atributos, conforme fragmento de código abaixo. Deve-se substituir atributo connectionURL, connectionName e connectionPassword com a string de conexão, o usuário e a senha correta do seu banco de dados, respectivamente.

```
<Realm className="org.apache.catalina.realm.JDBCRealm"
    connectionName="root" connectionPassword="root"
    connectionURL="jdbc:mysql://localhost:3306/atendimento" driverName="com.mysql.jdbc.Driver"
    roleNameCol="nome_permissao" userCredCol="senha" userNameCol="nome_usuario"
    userRoleTable="permissao_usuario" userTable="usuario" />
```

Figura 3: fragmento de código JDBCRealm server.xml

O módulo de login é executado em nível de servidor, e não da aplicação. Deverá ser incluído o arquivo do driver JDBC do MySQL na pasta “lib.” do Tomcat.

2.4 IMPLEMENTANDO JAAS A APLICAÇÃO

Para deixar a aplicação de atendimento segura, primeiramente precisa-se editar o arquivo “web.xml” para adicionar algumas configurações. A primeira delas é o método de autenticação que será utilizado, o básico.

```
<!-- Configurações de login -->
<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
```

Figura 4: método básico de autenticação com JAAS

Novamente no arquivo “web.xml”, precisaremos adicionar as possíveis *roles* (permissões) que o sistema pode utilizar. No sistema de atendimento, estaremos trabalhando apenas com as permissões “cadastro” e “consulta”.

```
<!-- Regras (permissões) da aplicação -->
<security-role>
    <role-name>cadastro</role-name>
</security-role>
<security-role>
    <role-name>consulta</role-name>
</security-role>
```

Figura 5: *Roles* da aplicação de atendimento

Precisaremos definir quais permissões são necessárias para acessar os recursos do sistema.

Exemplo de permissão de cadastro de atendente:

```
<!-- Mapeamento das permissões para os recursos web -->
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Cadastro de Atendente</web-resource-name>
        <url-pattern>/Atendimento/CadastroAtendente.xhtml</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>cadastro</role-name>
    </auth-constraint>
</security-constraint>
```

Figura 6: Mapeamento de permissões de role cadastro

Exemplo de permissão de consulta de atendente:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Consulta de Atendente</web-resource-name>
        <url-pattern>/Atendimento/ConsultaAtendente.xhtml</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>consulta</role-name>
    </auth-constraint>
</security-constraint>
```

Figura 7: Mapeamento de permissões de role consulta

É necessária uma permissão de cadastro e consulta para acessar o Index:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Index</web-resource-name>
        <url-pattern>/Atendimento/Index.xhtml</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>consulta</role-name>
        <role-name>cadastro</role-name>
    </auth-constraint>
</security-constraint>
```

Figura 8: Mapeamento de permissões do index

Exemplo de login para consulta e cadastro:

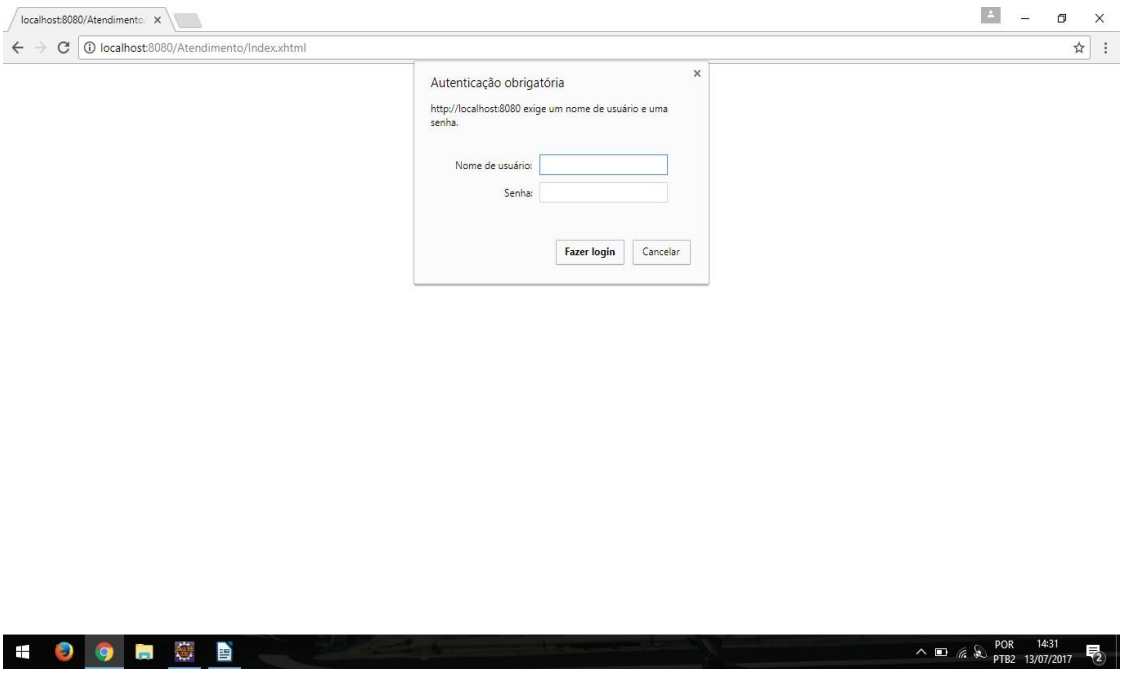

Figura 9: Exemplo de login utilizando JAAS

3. INJEÇÃO DE DEPENDÊNCIA

Injeção de dependências (dependency injection ou DI), é um padrão de desenvolvimento de software usado para conservar o baixo acoplamento entre classes do sistema. As dependências de um objeto não são mais instanciadas programaticamente, mas sim injetadas de alguma forma (ANDRADE, 2015).

CDI (contexts and Dependency Injection) é a especificação Java EE que trabalha com injeção de dependências. Pode-se usar CDI para instanciar e injetar objetos de uma aplicação.

3.1 CONFIGURANDO CDI

Para configurar CDI em um projeto, primeiro será preciso de uma implementação, pois CDI é uma especificação. Será utilizado o Weld. Como se está utilizando Maven, basta apenas ser adicionado as dependências no arquivo pom.xml do projeto.

```
<!-- Weld (implementação do CDI) -->
<dependency>
    <groupId>org.jboss.weld.servlet</groupId>
    <artifactId>weld-servlet</artifactId>
    <version>2.2.9.Final</version>
    <scope>compile</scope>
</dependency>
<!-- Weld depende do Jandex -->
<dependency>
    <groupId>org.jboss</groupId>
    <artifactId>jandex</artifactId>
    <version>1.2.3.Final</version>
    <scope>compile</scope>
</dependency>
```

Figura 10: Pom.xml com Weld

Será preciso criar um arquivo chamado context.xml no diretório src/main/webapp/META-INF da aplicação.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
<Manager pathname=""/>
<Resource name="BeanManager" auth="Container"
type="javax.enterprise.inject.spi.BeanManager"
factory="org.jboss.weld.resources.ManagerObjectFactory"/>
</Context>
```

Figura 11: configuração context.xml

No arquivo web.xml, será adicionado o código abaixo:

```
  <listener>
<listener-class>
org.jboss.weld.environment.servlet.Listener
</listener-class>
</listener>
<resource-env-ref>
<resource-env-ref-name>BeanManager</resource-env-ref-name>
<resource-env-ref-type>
javax.enterprise.inject.spi.BeanManager
</resource-env-ref-type>
</resource-env-ref>
```

Figura 12: configuração web.xml

No diretório src/main/resources/META-INF, será criado um arquivo vazio chamado beans.xml. A existência desse arquivo habilita CDI no projeto.

Em uma aplicação que utiliza CDI, praticamente todas as classes são consideradas *beans* CDI, também nomeadas como CDI *Managed beans*. *Beans* CDI conseguem ser injetados em outros beans. Apenas pelo fato de uma classe pública ter um construtor sem argumentos, ou ter um construtor com argumentos injetados, faz com que ela seja uma Bean CDI (ANDRADE, 2015).

Managed Beans CDI não podem ser injetados em *Managed Bean* JSF, por isso é necessário que os *Beans* sejam utilizados como um *Bean* CDI.

A anotação *@Named* possibilita o acesso ao *bean* CDI por *Expression Language*, através do nome da classe ou o nome colocado em *“value”* na anotação *@Named*.

3.2 ESCOPOS DE BEANS CDI

Escopo	Duração
@RequestScoped	Interação com usuário em uma única requisição HTTP.
@SessionScoped	Interação com usuário entre muitas requisições HTTP, ou seja, a sessão do usuário.
@ApplicationScoped	Estado compartilhado com todos os usuários durante toda a execução da aplicação.
@Dependent	É o escopo padrão, se nenhum for especificado. Mantém o mesmo ciclo de vida do bean que o injetou.
@ConversationScoped	Interação com usuário entre muitas requisições HTTP, com o início e término controlado pelo programador.

Figura 13: Escopos beans cdi

Fonte: Java ee7 com jsf, primefaces e cdi 2ª edição (ANDRADE, 2015).

Todas essas anotações devem ser importadas do pacote javax.enterprise.context.

3.3 IMPLEMENTANDO CDI A APLICAÇÃO

A partir da classe CadastroEscolaBean será exemplificado a implementação CDI; aonde se tem um *Managed bean* JSF, implementando um Service chamado de CadastroEscola e um repositório chamado de Escolas.

```
@ManagedBean
@SessionScoped
public class CadastroEscolaBean implements Serializable{
```

Figura 14: CadastroEscolaBean com anotações JSF

Com a implementação das devidas anotações a classe deverá ficar assim:

```
@Named
@javax.enterprise.context.SessionScoped
public class CadastroEscolaBean implements Serializable{
```

Figura 15: CadastroEscolaBean com anotações CDI

Será injetado o construtor do Service CadastroEscola e será injetado um construtor para o repositório Pendencias.

```
@Inject
private CadastroEscola cadastro;
@Inject
private Pendencias pendencias;
```

Figura 16: Injeção dos construtores

Os métodos prepararCadastro() e Salvar() sofrerão alterações devidas para poder utilizar as injeções e o padrão CDI. Aonde tinha-se:

```
public void prepararCadastro(){
    EntityManager manager = JpaUtil.geEntityManager();
    try{
        Pendencias pendencias = new Pendencias(manager);
        this.todasPendencias = pendencias.todos();
    }finally{
        manager.close();
    }
}
```

Figura 17: Método prepararCadastro()

```
public String salvar(ActionEvent ae){
    EntityManager manager = JpaUtil.geEntityManager();
    EntityTransaction trx = manager.getTransaction();
    FacesContext context = FacesContext.getCurrentInstance();
    try{
        trx.begin();
        CadastroEscola cadastro = new CadastroEscola(new Escolas(manager));
        cadastro.salvar(this.escola);

        this.escola = new Escola();
        context.addMessage(null, new FacesMessage("Historico de Atendimento salvo com sucesso!"));
        trx.commit();
    } catch (Exception e) {

        FacesMessage mensagem = new FacesMessage(e.getMessage());
        mensagem.setSeverity(FacesMessage.SEVERITY_ERROR);
        context.addMessage(null, mensagem);

    }
    return "ConsultaEscola.xhtml";
```

Figura 18: Método Salvar()

E com as devidas alterações os métodos atualizados ficaram assim:

```
public void prepararCadastro(){
    this.todasPendencias = this.pendencias.todos();
    }
```

Figura 19: Método prepararCadastro() atualizado

```
public String salvar(ActionEvent ae){
    FacesContext context = FacesContext.getCurrentInstance();
    try{
        this.cadastro.salvar(escola);
        this.escola = new Escola();
        context.addMessage(null, new FacesMessage("Escola salva com sucesso!"));

    } catch (Exception e) {

        FacesMessage mensagem = new FacesMessage(e.getMessage());
        mensagem.setSeverity(FacesMessage.SEVERITY_ERROR);
        context.addMessage(null, mensagem);

    }
    return "ConsultaEscola.xhtml";
}
```

Figura 20: Método salvar() atualizado

Enquanto que o método JSF necessitava do instanciamento do EntityManager, o método CDI não há necessidade do instanciamento do mesmo, devido as injeções aplicadas automáticamente. Outro ponto é que não há mais a necessidade de instanciar o objeto dentro do escopo do método, facilitando assim a escrita do código, legibilidade e manutenção do código.

.

4. PADRÕES DE PROJETO

Os padrões de projetos promovem reutilizar projetos e arquiteturas bem-sucedidas. Os *Design Patterns* (Padrões de projetos) auxiliam a melhor escolher opções bem concebidas de projeto que tornam um programa ou sistema reutilizável e a evitar escolhas que comprometam a reutilização. Os *Design Patterns* podem otimizar a documentação e o suporte ao fornecer uma melhor especificação explicita de interações de classes e objetos e seu objetivo subjacente, ou seja, ajuda o projetista alcançar com mais eficiência e rapidez um projeto adequado (VLISSIDES, 2000).

Design Patterns são descrições de objetos e classes que se comunicam que precisam ser personalizadas para resolver um problema geral de um projeto num contexto particular.

4.1 PADRÃO DE PROJETO SINGLETON

A intenção do padrão de projeto Singleton é criar uma classe que tenha apenas uma instancia e fornecer um ponto global de acesso para a mesma (VLISSIDES, 2000).

Utiliza-se o padrão Singleton quando for necessário haver exclusivamente uma instância de uma classe, e essa instância tiver que dar acesso aos clientes através de um ponto bem notório; a única instancia que tiver de ser extensível através de uma subclasse, possibilitando aos clientes usar uma instância estendida sem alterar o seu código-fonte.

4.2 IMPLEMENTANDO O PADRÃO SINGLETON

As aplicações que utilizam JPA carecem apenas de uma única instância do EntityManagerFactory, esta única instância pode ser aproveitada por qualquer código que queira obter um EntityManager. Um EntityManager é responsável por manusear entidades no contexto de persistência. Através dos métodos dessa interface, é possível persistir, pesquisar e excluir objetos do banco de dados (ANDRADE, 2015).

A inicialização de EntityManagerFactory pode demorar determinados instantes, por isso a instância dessa interface deve ser compartilhada na aplicação. Será preciso um lugar para alocar a instância compartilhada de EntityManagerFactory, onde qualquer código tenha acesso fácil e rápido. Será criada a classe JpaUtil para conter a instância em uma variável estática.

```
public class JpaUtil {

    private static EntityManagerFactory factory;

    static{
        factory = Persistence.createEntityManagerFactory("AtendimentoPU");

    }
    public static EntityManager geEntityManager(){
        return factory.createEntityManager();
    }
}
```

Figura 21: Classe JpaUtil em sua essência

Foi criado um bloco estático para inicializar a fábrica de Entity Manager. Isso ocorrerá apenas uma vez, no carregamento da classe. Sempre que for necessário utilizar de uma EntityManager, podemos chamar: EntityManager manager = JpaUtil.getEntityManager();

Um exemplo da instância sendo chamada em um método em um *Managed Bean:*

```
public String salvar(){
    EntityManager manager = JpaUtil.geEntityManager();
    EntityTransaction trx = manager.getTransaction();
    FacesContext context = FacesContext.getCurrentInstance();
```

Figura 22: Método com a instância do JpaUtil

O projeto atendimento utiliza-se de CDI, então teremos que transformar nossa instância de EntityManager em um método produtor *(“producer method),* pois o utilitário não é um *Bean* CDI, devido a isso, não é possível injetá-lo automaticamente.

```java
@ApplicationScoped
public class EntityManagerProducer {

    private EntityManagerFactory factory;

    public EntityManagerProducer(){
        this.factory = Persistence.createEntityManagerFactory("AtendimentoPU");
    }

    @Produces
    @RequestScoped
    public EntityManager createEntityManager(){
            return factory.createEntityManager();
    }
    public void closeEntityManager(@Disposes EntityManager manager){
        manager.close();
    }
}
```

Figura 23: *Producer method* EntityManagerFactory

Nesta classe é necessário a criação de um método definido como fábrica de EntityManager, que recebe como parâmetro o arquivo de persistencia.xml. Tem-se a criação de um método que recebe as anotações da produção e do escopo do ciclo de vida deste método, aonde se instancia a criação da fábrica, e outro método que tem como função fazer o fechamento da fábrica de EntityManager.

5. ANÁLISE DO SISTEMA

Este capítulo descreve as informações obtidas através da análise de requisitos.

5.1 MAPA MENTAL

Figura 24: Mapa Mental sistema de Atendimento

5.2 DIAGRAMA DE CASOS DE USO GERAL

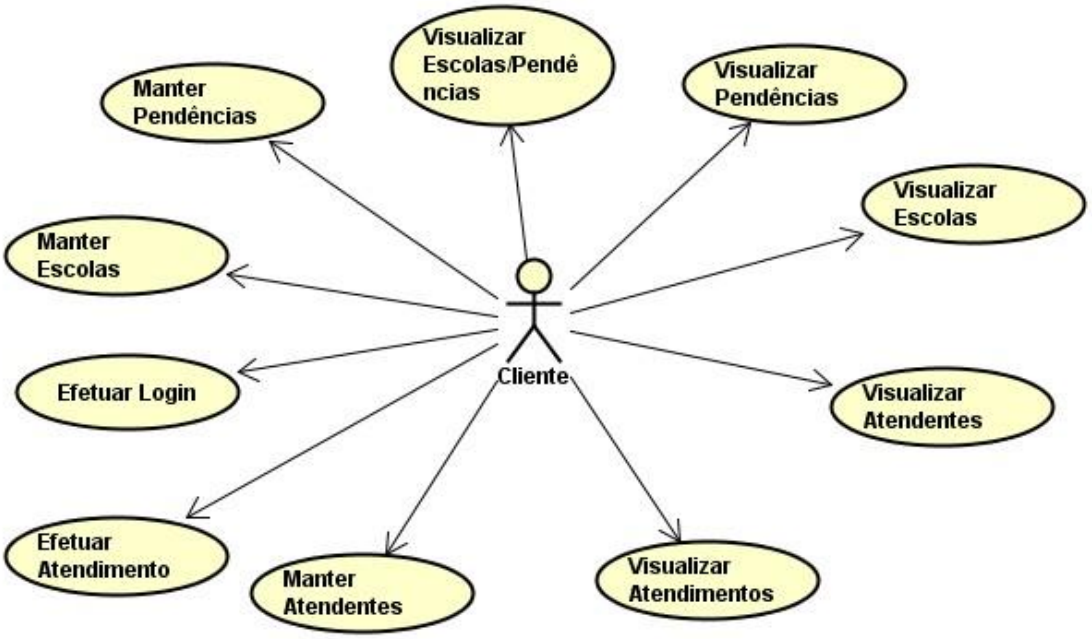

Figura 25: Diagrama de casos de uso geral

5.3 ESPECIFICAÇÕES DE CASOS DE USO

5.3.1 EFETUAR LOGIN

Figura 26: Diagrama de casos de uso: efetuar login

Nome do caso de uso	Efetuar Login.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente acesse o sistema devida suas permissões.
Pré-Requisitos	O usuário necessita estar cadastrado.
Evento Inicial	O cliente deve abrir o sistema.
Fluxo Principal	O sistema verifica os dados e efetua o acesso.
Fluxos Alternativos	O cliente cancela o Login.
Fluxos de Exceção:	O sistema não consegue identificar os dados digitados e não permite o acesso.
Pós-Condição	O cliente consegue permissão para acessar o sistema.

Tabela 1: Documentação do caso de uso: efetuar Login

5.3.2 MANTER ESCOLAS

Figura 27: Diagrama de casos de uso: manter escolas

Nome do caso de uso	Manter Escolas.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente cadastre escolas.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina cadastrar escolas.
Fluxo Principal	O cliente cadastra uma escola.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente digita dados inválidos e assim não consegue cadastrar.
Pós-Condição	O cliente consegue cadastrar uma escola.

Tabela 2: Documentação do caso de uso: manter escolas

5.3.3 MANTER ATENDENTES

Figura 28: Diagrama de casos de uso: manter atendentes

Nome do caso de uso	Manter Atendentes.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente cadastre atendentes.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina cadastrar atendentes.
Fluxo Principal	O cliente cadastra um atendente.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente digita dados inválidos e assim não consegue cadastrar.

Pós-Condição	O cliente consegue cadastrar um atendente.

Tabela 3: Documentação do caso de uso: manter atendentes

5.3.4 MANTER PENDÊNCIAS

Figura 29: Diagramas de casos de uso: manter pendencias

Nome do caso de uso	Manter Pendências.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente cadastre pendências.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina cadastrar pendências.
Fluxo Principal	O cliente cadastra uma pendência.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente digita dados inválidos e assim não consegue cadastrar.
Pós-Condição	O cliente consegue cadastrar uma pendência.

Tabela 4: Documentação de caso de uso: manter pendências

5.3.5 EFETUAR ATENDIMENTO

Figura 30: Diagrama de casos de uso: efetuar atendimento

Nome do caso de uso	Efetuar Atendimento.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente efetue um atendimento.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina cadastrar atendimento.
Fluxo Principal	O cliente cadastra um atendimento.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente digita dados inválidos e assim não consegue cadastrar.
Pós-Condição	O cliente consegue cadastrar um atendimento.

Tabela 5: Documentação do caso de uso: efetuar atendimento

5.3.6 VISUALIZAR ESCOLAS

Figura 31: Diagrama de caso de uso: visualizer escolas

Nome do caso de uso	Visualizar Escolas.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente visualize as escolas cadastradas.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina consultar escolas.
Fluxo Principal	O cliente consulta escolas
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente não tem acesso a página
Pós-Condição	O cliente consegue consultar escolas.

Tabela 6: Documentação do caso de uso: visualizar escolas

5.3.7 VISUALIZAR ATENDENTES

Figura 32: Diagrama de caso de uso: visualizar atendente

Nome do caso de uso	Visualizar Atendentes.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente visualize atendentes cadastradas.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina consultar atendentes.
Fluxo Principal	O cliente consulta atendentes
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente não tem acesso a página
Pós-Condição	O cliente consegue consultar atendentes

Tabela 7: Documentação do caso de uso: visualizar atendentes

5.3.8 VISUALIZAR PENDENCIAS

Figura 33: Diagrama de caso de uso: visualizar pendências

Nome do caso de uso	Visualizar Pendências.
Ator principal	Cliente.
Finalidade:	Permitir que o cliente visualize pendências cadastradas.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina consultar pendências
Fluxo Principal	O cliente consulta pendências.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente não tem acesso a página
Pós-Condição	O cliente consegue consultar pendências.

Tabela 8: Documentação do caso de uso: visualizar pendências

5.3.9 VISUALIZAR ATENDIMENTO

Figura 34: Diagrama do caso de uso: visualizar atendimento

Nome do caso de uso	Visualizar Atendimento.

Ator principal	Cliente.
Finalidade:	Permitir que o cliente visualize atendimentos cadastradas.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina consultar atendimento
Fluxo Principal	O cliente consulta atendimentos.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente não tem acesso a página
Pós-Condição	O cliente consegue consultar atendimentos.

Tabela 9: Documentação do caso de uso: visualizar atendimento

5.3.10 VISUALIZAR ESCOLA/PENDÊNCIA

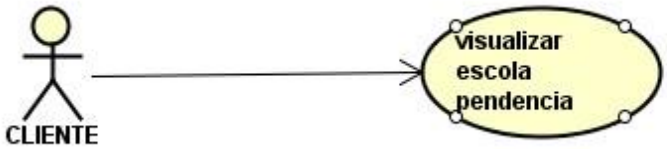

Figura 35: Diagrama do caso de uso: visualizar escola/pendência

Nome do caso de uso	Visualizar Escola/Pendência
Ator principal	Cliente.
Finalidade:	Permitir que o cliente visualize escolas com pendências cadastradas.
Pré-Requisitos	O usuário necessita estar logado no sistema.
Evento Inicial	O cliente deve abrir a pagina consultar Escola/Pendência
Fluxo Principal	O cliente consulta escolas com pendências.
Fluxos Alternativos	O cliente sai da página.
Fluxos de Exceção:	O cliente não tem acesso a página
Pós-Condição	O cliente consegue consultar escolas com pendências.

Tabela 10: Documentação caso de uso: Visualizar Escola/Pendência

5.4 DIAGRAMA E-R SISTEMA ATENDIMENTO

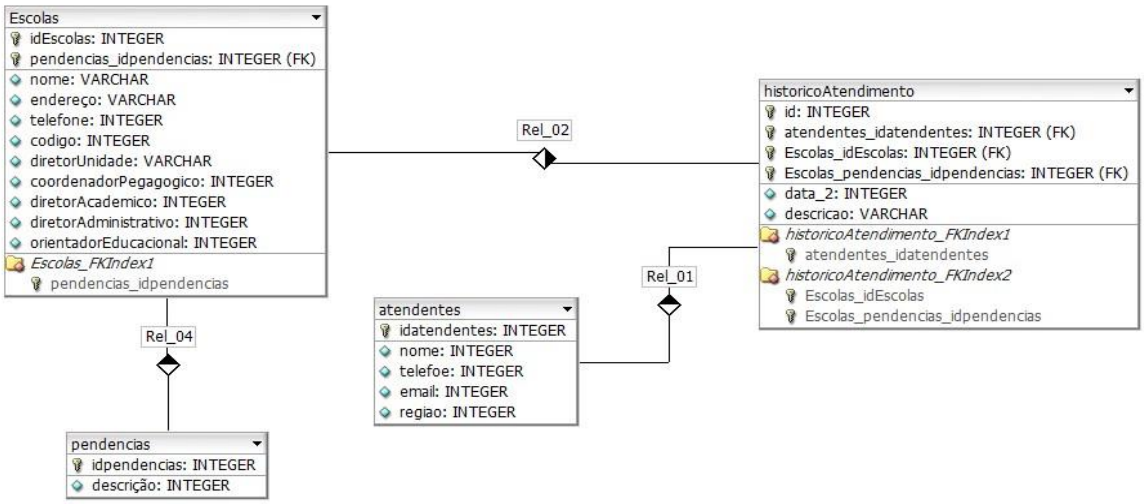

Figura 36: Diagrama E-R sistema de atendimento

5.4.1 DIAGRAMA E-R JAAS

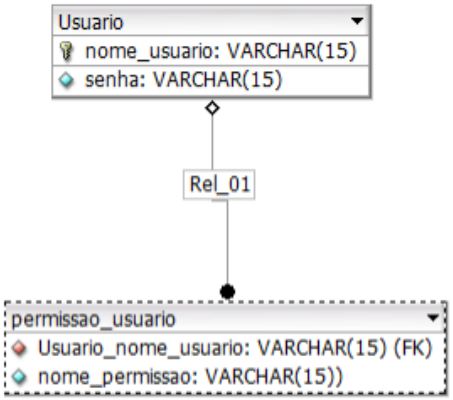

Figura 37: Diagrama E-R JAAS

6. TELAS DO SISTEMA

Neste capítulo será mostrado algumas telas do sistema.

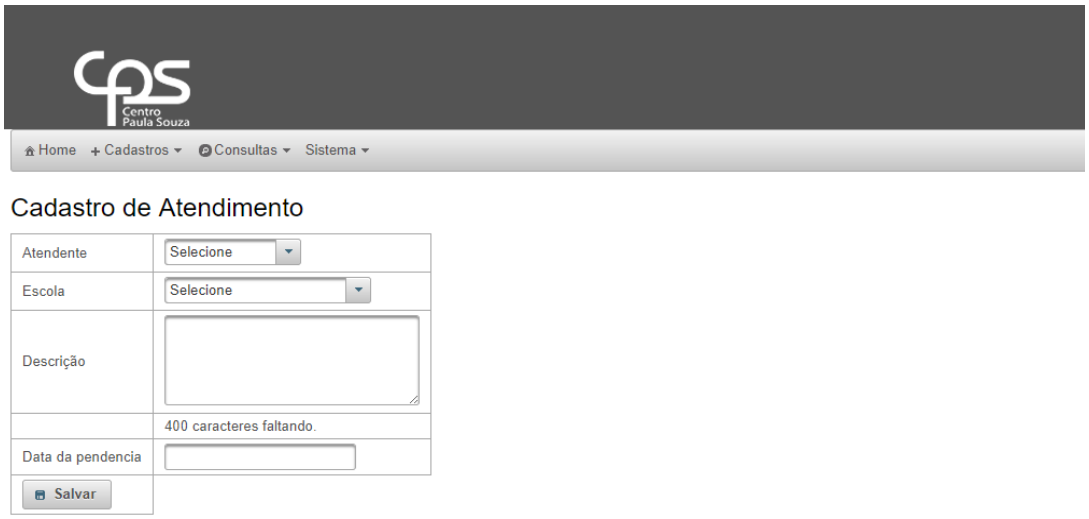

Figura 38: Tela de cadastro de atendimento

Figura 39: Tela cadastro de atendente

7. CONCLUSÃO

O trabalho apresentado foi designado para desenvolver um aplicativo a escola Centro Paula Souza de Assis, utilizando das tecnologias JAAS, CDI e o padrão de projeto Singleton. O sistema foi inicialmente desenvolvido em JSF com JPA e foi visível a mudança com a implementação do JAAS e CDI.

A tecnologia CDI trouxe resultados gratificantes conforme o esperado, o código tornou-se mais legível e de fácil manutenção, podendo obter as vantagens do baixo acoplamento de classes disponibilizado pela tecnologia CDI. A conclusão sobre está tecnologia é que a mesma tem um grande potencial de mercado, sendo de fácil adesão e de fácil compreensão até mesmo por quem não tem experiência com Java.

A tecnologia JAAS mostrou-se de tremendo potencial tratando-se de que a tecnologia é extremamente simples de ser implementada e mesmo assim traz diversos benefícios ao software. Com JAAS, independente da forma de segurança, seja ela básica ou mais trabalhada, cumpre com a função de proteger dados a partir de um usuário e senha cadastrado no banco de dados. A conclusão sobre está tecnologia é de que ela tem um grande potencial e pode ser aproveitada em qualquer software que necessite de um sistema de login para manter a segurança de seus dados, sendo de fácil adesão e compreendimento até por quem não tem experiência em Java.

O padrão de projeto Singleton, como qualquer outro padrão, cumpre com a solução imposta diante do possível problema ou lacuna do software. Com o padrão Singleton o software ganhou uma considerável otimização de tempo de execução e memória gasta. Se conclui que, o padrão Singleton pode e deve ser implementado sempre que for necessário, já que é simples sua implementação dependendo da estratégia imposta pelo analista do sistema, sendo de grande importância na hora de instanciar algo que não seja leve a nível de processamento.

REFERENCIAS

ANDRADE, Thiago Faria**. DWJSF – Desenvolvimento Web com JavaServer Faces**. ALGAWORKS. 2ª Edição. 2010

ANDRADE, Thiago Faria. **JAVA EE7 com JSF, Primefaces e CDI.** ALGAWORKS. 2ª Edição. 2015

BUENO, Kassia Jaqueline. O que é JSF, 2013. Disponível em:<:http://fabrica.ms.senac.br/2013/06/o-que-e-jsf-java-server-faces/. Acesso em: 02 de nov. 2016.

DEVMEDIA. 2015. Disponível em:< http://www.devmedia.com.br/introducao-a-seguranca-da-informacao-em-java/28247#> Acesso em: 05/03/2017

FILITTO, Danilo. JPA – O que é? Pra que serve?, 2015. Disponível em:< http://www.dfilitto.com.br/java/jpa-o-que-e-para-que-serve-como-implementar-um-sistema/>. Acesso em 02 de nov. 2016.

MAIA, Marco Aurélio. O que é Segurança da Informação, 2013. Disponível em:< http://segurancadainformacao.modulo.com.br/seguranca-da-informacao />. Acesso em: 02 de nov. 2016.

MEDEIROS, Higor. Injeção de Dependência no Java EE, 2015. Disponível em:< http://www.devmedia.com.br/injecao-de-dependencia-no-java-ee/31460 />. Acesso em: 02 de nov. 2016.

PILARI, Marcos, MATERA UTILIZA O PRIMEFACES, 2011. Disponível em:<http://www.matera.com/br/2011/09/30/matera-utiliza-o-primefaces/ />. Acesso em: 02 de nov. 2016.

SOUZA, Nailson. O que é hibernate, 2012. Disponível em:<http://blog.naison.com.br/java/o- que-e-hibernate />. Acesso em: 02 de nov 2016.

TECNOLOGICA. 2013. Disponível em:< https://www.teclogica.com.br/seguranca-em-java/> Acesso

VUKOTIC, Aleska; GOODWILL, James. Introduzindo o servidor de aplicação Apache Tomcat, 2011. Disponível em:< http://www.devmedia.com.br/introduzindo-o-servidor-de-aplicacao-apache-tomcat/27939/ >. Acesso em 02 de nov. 2016

VLISSIDES, John. **Padrões de projeto.** SOLUÇOES REUTILIZAVEIS DE SOFTWARE ORIENTADO A OBJ. Bookman Companhia Ed. 2000.

WEBSENSE, Inc. Estudo global diz que 30% dos profissionais de segurança consideram que existem lacunas nos atuais sistemas de segurança, 2014. Disponível em:<http://convergecom.com.br/embratel-pense-inovacao/acontece/estudo-global-diz-que-30-dos-profissionais-de-seguranca-consideram-que-existem-lacunas-nos-atuais-sistemas-de-seguranca/>. Acesso em: 02 de nov. 2016.