FACULDADE METODDISTA GRANBERY – FMG
CURSO DE SISTEMA DE INFORMAÇÃO

# MANUTENÇÃO

FREDERICO CASSEMIRO
MATHEUS PERES DE ARAÚJO

JUIZ DE FORA – 2018

INTRODUÇÃO

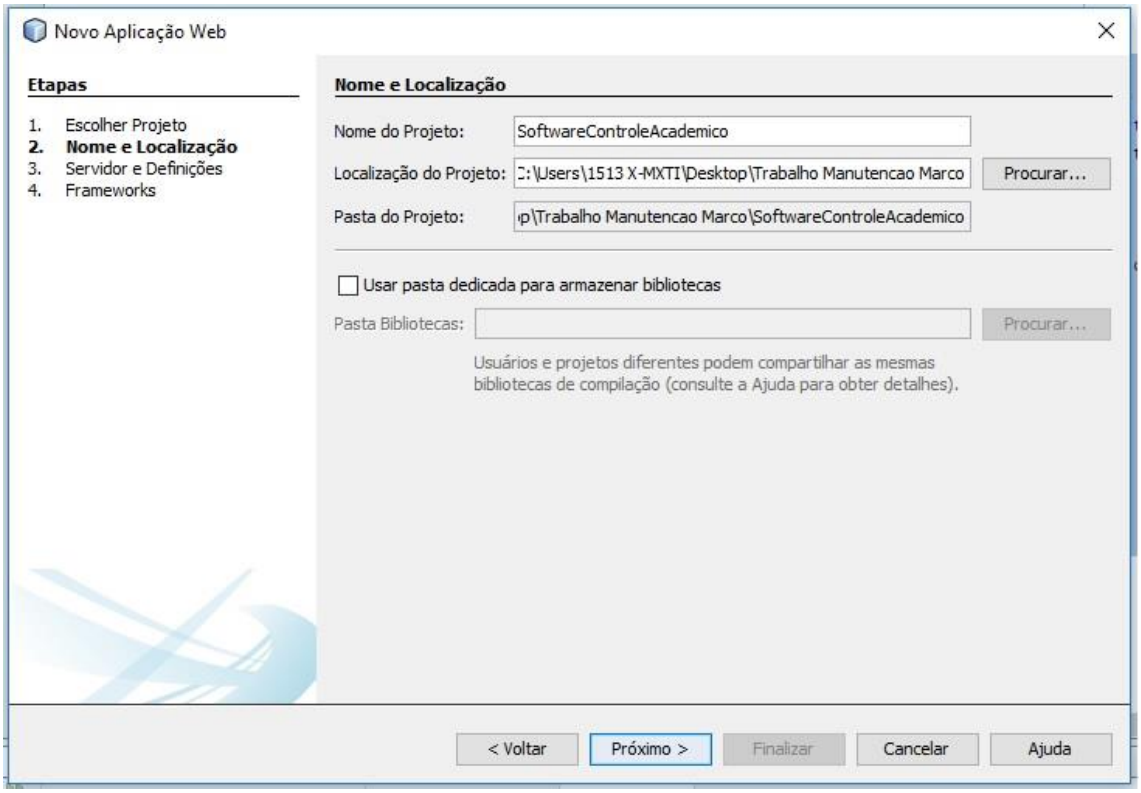
O seguinte documento irá apresentar os passos realizados durante a manutenção do software de controle acadêmico, também conhecido como SCA.

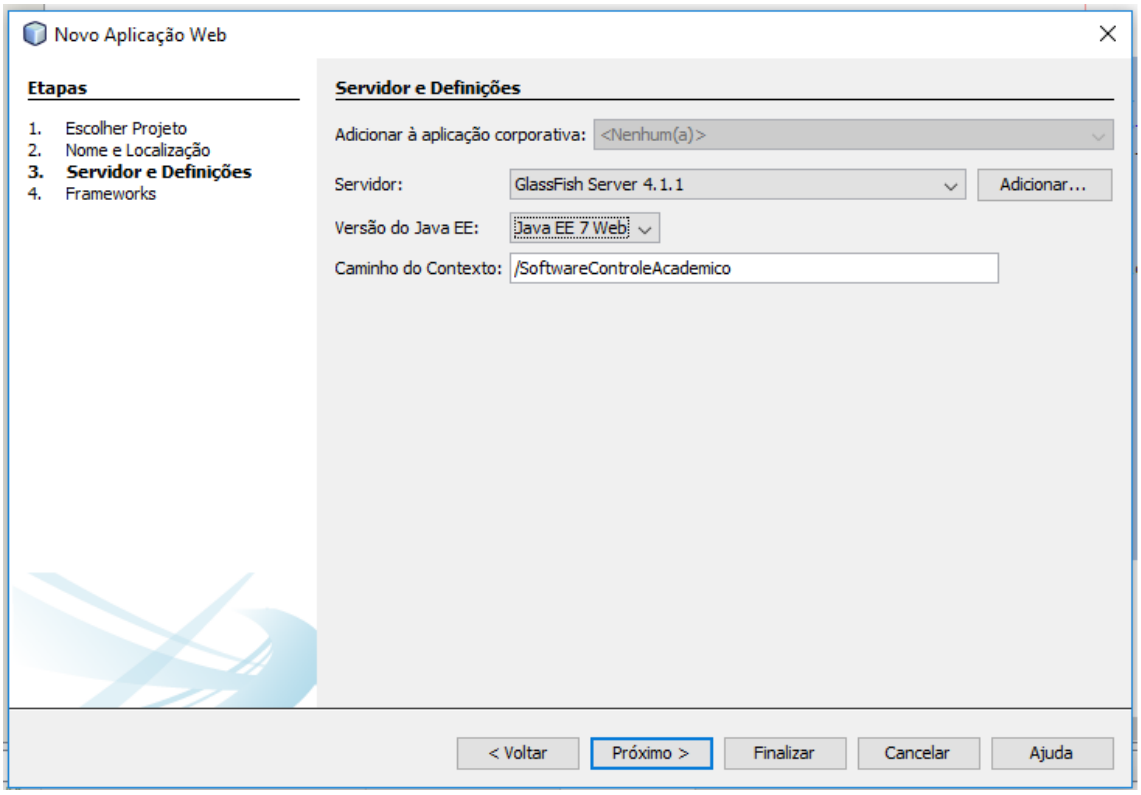
PASSO 01 – BANCO

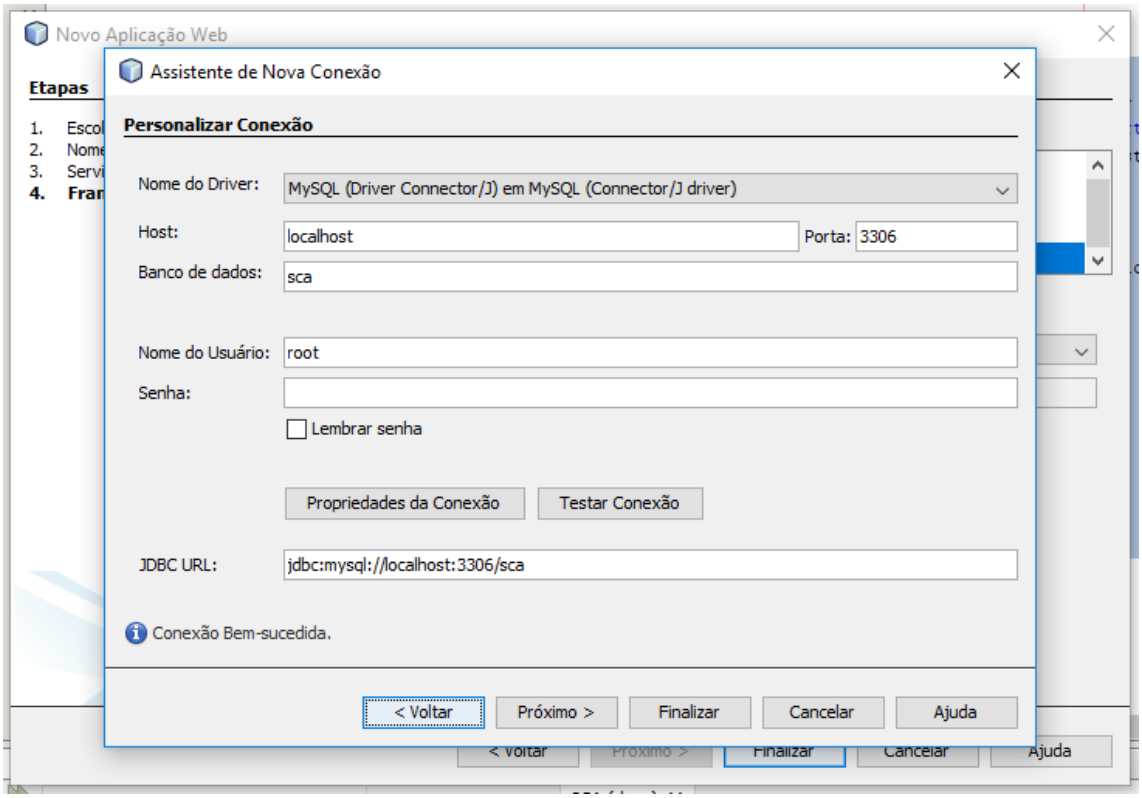
O seguinte projeto utiliza um banco de dados para persistir as informações adicionadas no programa. Antes de iniciar o projeto foi realizado a instalação do XAMP, iniciados os serviços do Apache e do MySql. Criado também um banco de dados vazio com o nome de `sca`.

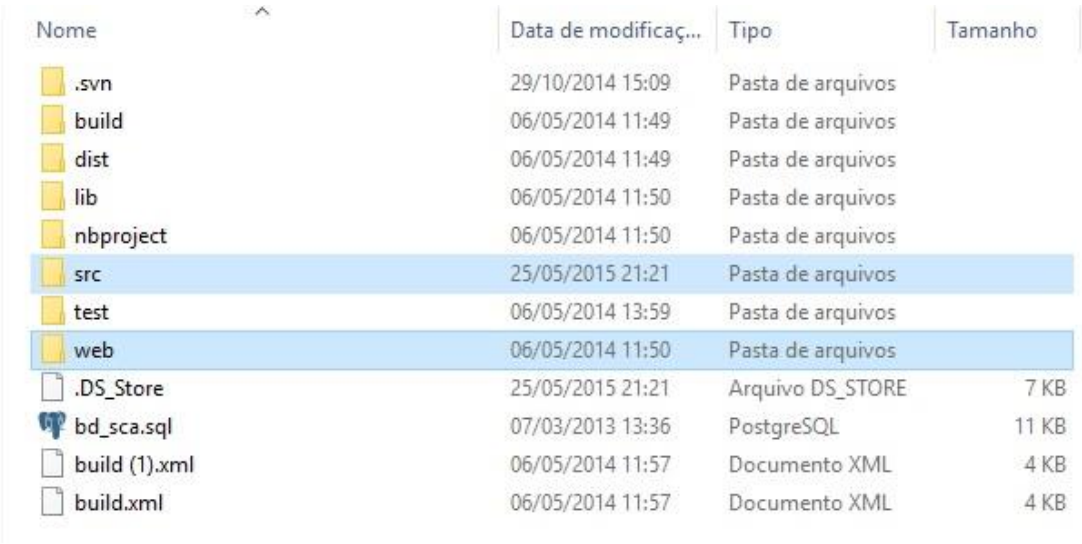
PASSO 02 – INICIALIZAÇÃO

A inicialização do projeto não foi realizada pela ferramenta de importação do NetBeans. Em outras tentativas de inicialização do projeto foi observado erro na execução do programa, portanto, a inicialização se deu através da criação de um novo projeto e posterior copia dos arquivos encontrados nos diretórios SRC e WEB do projeto original.




Novo Aplicação Web
Etapas
1. Escolher Projeto
2. Nome e Localização
3. Servidor e Definições
4. Frameworks
Servidor e Definições
Adicionar à aplicação corporativa: <Nenhum(a)>
Servidor: GlassFish Server 4.1.1
Adicionar...
Versão do Java EE: Java EE 7 Web
Caminho do Contexto: /SoftwareControleAcademico
< Voltar
Próximo >
Finalizar
Cancelar
Ajuda

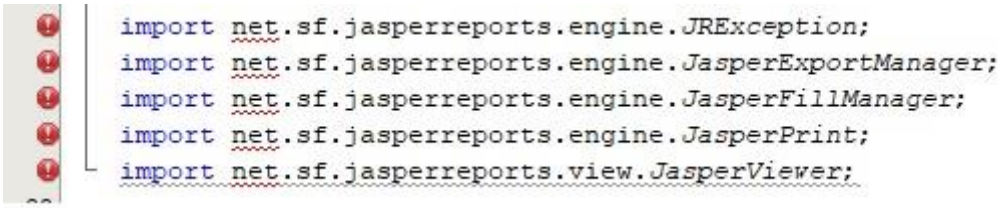

Novo Aplicação Web
Assistente de Nova Conexão
Personalizar Conexão
Nome do Driver: MySQL (Driver Connector/J) em MySQL (Connector/J driver)
Host: localhost
Porta: 3306
Banco de dados: sca
Nome do Usuário: root
Senha:
Lembrar senha
Propriedades da Conexão
Testar Conexão
JDBC URL: jdbc:mysql://localhost:3306/sca
Conexão Bem-sucedida.
< Voltar
Próximo >
Finalizar
Cancelar
Ajuda



| Nome | Data de modificaç... | Tipo | Tamanho |
|---|---|---|---|
| .svn | 29/10/2014 15:09 | Pasta de arquivos | |
| build | 06/05/2014 11:49 | Pasta de arquivos | |
| dist | 06/05/2014 11:49 | Pasta de arquivos | |
| lib | 06/05/2014 11:50 | Pasta de arquivos | |
| nbproject | 06/05/2014 11:50 | Pasta de arquivos | |
| src | 25/05/2015 21:21 | Pasta de arquivos | |
| test | 06/05/2014 13:59 | Pasta de arquivos | |
| web | 06/05/2014 11:50 | Pasta de arquivos | |
| .DS_Store | 25/05/2015 21:21 | Arquivo DS_STORE | 7 KB |
| bd_sca.sql | 07/03/2013 13:36 | PostgreSQL | 11 KB |
| build (1).xml | 06/05/2014 11:57 | Documento XML | 4 KB |
| build.xml | 06/05/2014 11:57 | Documento XML | 4 KB |

PASSO 03 – BIBLIOTECA

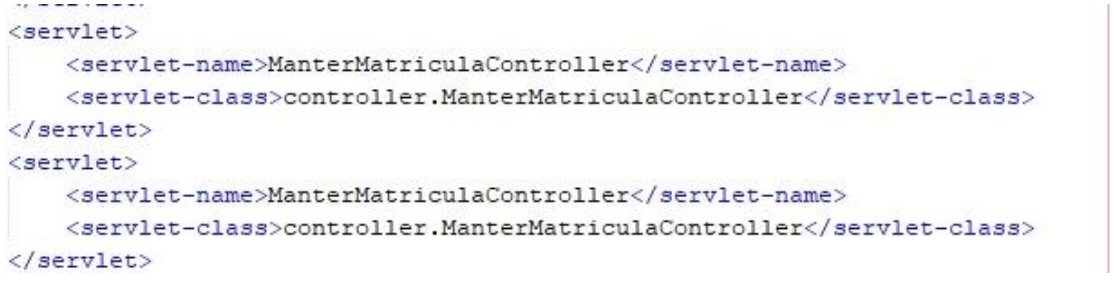
Depois da etapa dois o NetBeans já informa que existem importações incorretas dentro do programa, está sendo resultado da falta de uma biblioteca. Realizado a copia da biblioteca jasperreports-5.0.4 que se encontrava no diretório lib do projeto original, realizando posteriormente a adição dessa biblioteca no projeto.



```
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.view.JasperViewer;
```

PASSO 04 – DUPLICIDADE DE SERVLET

Após executar o programa tem-se como resultado erro por duplicidade de valor (ManterMatriculaController) para servlet dentro do arquivo web.xml. Removido a duplicidade para correção.



```
<servlet>
    <servlet-name>ManterMatriculaController</servlet-name>
    <servlet-class>controller.ManterMatriculaController</servlet-class>
</servlet>
<servlet>
    <servlet-name>ManterMatriculaController</servlet-name>
    <servlet-class>controller.ManterMatriculaController</servlet-class>
</servlet>
```

PASSO 05 – PÁGINA INICIAL

Removido o arquivo index.html do projeto, arquivo adicionado durante a criação do projeto, e alterado o atributo welcome-file no arquivo web.xml para index.jsp.

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

PASSO 06 – HIBERNATEUTIL

Após executar o programa e acessar algum menu que realiza busca de informações no banco de dados é apresentado mensagem de erro relacionado a inicialização do HibernateUtil. Solucionado o problema alterando a forma que esta cria a conexão.

```
java.lang.NoClassDefFoundError: Could not initialize class util.HibernateUtil
```

Antes:

```
package util;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/**
 * Hibernate Utility class with a convenient method to get Session Factory
 * object.
 *
 * @author José Augusto
 */
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Depois:

```java
package util;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {
    private static SessionFactory sessionFactory;

    public static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {
            // loads configuration and mappings
            Configuration configuration = new Configuration().configure();
            ServiceRegistry serviceRegistry
                = new StandardServiceRegistryBuilder()
                    .applySettings(configuration.getProperties()).build();

            // builds a session factory from the service registry
            sessionFactory = configuration.buildSessionFactory(serviceRegistry);
        }

        return sessionFactory;
    }
}
```

PASSO 07 – QueryTranslatorFactory

Executando o programa é apresentado um novo problema referente a não inicialização do atributo QueryTranslatorFactory dentro do arquivo hibernate.cfg.xml. Corrigido atribuindo valor correto para o elemento.

Antes:

```xml
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/sca?zeroDateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password"></property>
    <property name="hibernate.current_session_context_class">thread</property>
    <property name="hibernate.query.factory_class">org.hibernate.hql.classic.ClassicQueryTranslatorFactory</property>
    <property name="hibernate.show_sql">true</property>
    <mapping class="model.Professor"/>
    <mapping class="model.Horario"/>
    <mapping class="model.Aluno"/>
    <mapping class="model.AvaliacaoId"/>
    <mapping class="model.Disciplina"/>
    <mapping class="model.Curso"/>
    <mapping class="model.Turma"/>
    <mapping class="model.Avaliacao"/>
  </session-factory>
</hibernate-configuration>
```

Depois:

```
<session-factory>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/sca?zeroDateTimeBehavior=convertToNull</property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password"></property>
  <property name="hibernate.current_session_context_class">thread</property>
  <property name="hibernate.query.factory_class">org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory</property>
  <property name="hibernate.show_sql">true</property>
  <mapping class="model.Professor"/>
  <mapping class="model.Horario"/>
  <mapping class="model.Aluno"/>
  <mapping class="model.AvaliacaoId"/>
  <mapping class="model.Disciplina"/>
  <mapping class="model.Curso"/>
  <mapping class="model.Turma"/>
  <mapping class="model.Avaliacao"/>
</session-factory>
```

PASSO 08 – CRIAÇÃO DAS TABELAS

Ao executar o programa e abrir novamente um menu que realiza consultas no banco tem-se como erro problemas ao encontrar a tabela para buscar as informações. Resolvido o problema adicionando um novo atributo nas configurações.

```
org.hibernate.exception.SQLGrammarException: could not extract ResultSet
```

Atributo adicionado:

```
<property name="hibernate.hbm2ddl.auto">update</property>
```

PASSO 09 – NESTED TRANSACTIONS

A principio o programa executa normalmente, mas após abrir e fechar diversos menus ele acaba por retornar erro relacionado as conexões realizadas. O problema foi resolvido ao alterar o método que estava sendo utilizado para iniciar a conexão com o banco de dados.

```
org.hibernate.TransactionException: nested transactions not supported
```

Antes:

```
Session session = HibernateUtil.getSessionFactory().getCurrentSession();
```

Depois:

```
Session session = HibernateUtil.getSessionFactory().openSession();

session.close();
```

PASSO 10 – LAZZY

Após a alteração algumas chamadas obtiveram problemas, sendo este resolvido com uma nova configuração.

```
org.hibernate.LazyInitializationException: could not initialize proxy - no Session
```

Atributo adicionado:

```
<property name="hibernate.enable_lazy_load_no_trans">true</property>
```

PASSO 11 – BD.CLASS

Alteração do atributo senha na conexão realizada pela classe BD.java

PASSO 12 – BIBLIOTECAS

Para gerar relatórios foram adicionadas todas as outras bibliotecas do projeto original.

PASSO 13 – MANUTENÇÃO INCREMENTAL

Adicionado ao programa um novo pedido. Anteriormente este realiza cálculo de avaliações referente a duas notas informadas e se necessário o valor de uma avaliação final. Com a modificação agora existem 3 campos para a consolidação de notas.

Foi necessário alterar o modelo, adicionando um novo atributo.

Não foi necessário alterar o dao, pois o hibernate já faz as alterações com base no modelo de classe.

Alterado o controller ManterAlunoNotaFrequenciaController apenas adicionando o atributo na hora de enviar as informações para a view.

Alterado 3 classes de visualização sendo elas consultarNotaFrequencia, manterAlunoNotaFrequencia e manterNotaFrequencia adicionando as colunas para apresentação do novo campo e os cálculos que são realizados dentro da própria página.

Alteração no modelo Avaliacao:

```
public class Avaliacao implements java.io.Serializable {

    private AvaliacaoId id;
    private Aluno aluno;
    private Turma turma;
    private Float nota1;
    private Float nota2;
    private Float nota3;
    private Integer numFaltas;
    private Float notaProvaFinal;
```

Alteração Controller ManterAlunoNotaFrequenciaController:

```
public void confirmarOperacao(HttpServletRequest request, HttpServletResponse response) throws
     int codTurma = Integer.parseInt(request.getParameter("txtCodTurma"));
    int matricula = Integer.parseInt(request.getParameter("txtMatricula"));
    float nota1 = 0;
    float nota2 = 0;
    float nota3 = 0;
    int numFaltas = 0;
    float notaProvaFinal = 0;
    if(!request.getParameter("txtNota1").equals("")){
        nota1 = Float.parseFloat(request.getParameter("txtNota1"));
    }

    if(!request.getParameter("txtNota2").equals("")){
        nota2 = Float.parseFloat(request.getParameter("txtNota2"));
    }

    if(!request.getParameter("txtNota3").equals("")){
        nota3 = Float.parseFloat(request.getParameter("txtNota3"));
    }

    if(!request.getParameter("txtNumFaltas").equals("")){
        numFaltas = Integer.parseInt(request.getParameter("txtNumFaltas"));
    }

    if(!request.getParameter("txtNotaProvaFinal").equals("")){
        numFaltas = Integer.parseInt(request.getParameter("txtNumFaltas"));
    }

    if(!request.getParameter("txtNotaProvaFinal").equals("")){
        notaProvaFinal = Float.parseFloat(request.getParameter("txtNotaProvaFinal"));
    }

    try {
        avaliacao.setNota1(nota1);
        avaliacao.setNota2(nota2);
        avaliacao.setNota3(nota3);
        avaliacao.setNumFaltas(numFaltas);
        avaliacao.setNotaProvaFinal(notaProvaFinal);
        avaliacao.alterar();
        RequestDispatcher view = request.getRequestDispatcher("ManterNotaFrequenciaController"
        view.forward(request, response);
```

Alteração View consultarNotaFrequencia

```
<td>${avaliacao.turma.disciplina.codDisciplina}</td>
<td>${avaliacao.turma.disciplina.periodo}</td>
<td>${avaliacao.turma.disciplina.nome}</td>
<td>${avaliacao.numFaltas}</td>
<td>${avaliacao.nota1}</td>
<td>${avaliacao.nota2}</td>
<td>${avaliacao.nota3}</td>
<td>${(avaliacao.nota1 + avaliacao.nota2 + avaliacao.nota3)/3}</td>
<td>${avaliacao.notaProvaFinal}</td>
```

Alteração View manterAlunoNotaFrequencia

```
<tr>
    <td>
        Nota 01:
    </td>
    <td>
        <input type="text" name="txtNota1" value="${avaliacao.nota1}"/>
    </td>
    <td>
        Nota 02:
    </td>
    <td>
        <input type="text" name="txtNota2" value="${avaliacao.nota2}"/>
    </td><td>
        Nota 03:
    </td>
    <td>
        <input type="text" name="txtNota3" value="${avaliacao.nota3}"/>
    </td>
</tr>
```

Alteração View manterNotaFrequencia

```
<td>${avaliacao.aluno.matricula}</td>
<td>${avaliacao.aluno.nome}</td>
<td>${avaliacao.numFaltas}</td>
<td>${avaliacao.nota1}</td>
<td>${avaliacao.nota2}</td>
<td>${avaliacao.nota3}</td>
<td>${avaliacao.notaProvaFinal}</td>
<td>${(avaliacao.nota1 + avaliacao.nota2 + avaliacao.nota3)/3}</td>
```

PASSO 14 – MANUTENÇÃO CORRETIVA CONTROLE DE CREDENCIAL

Adicionado ao programa o conceito de sessão e politicas de permissão para que as informações no documento de requisitos possam fazer sentido. Foi criado uma nova classe, Credenciais, e através de um de seus atributos acesso a apresentação do menu do sistema é modificado. Lembrando que só pode acessar o sistema se tiver as credenciais de acesso.

Mapeamento do Filtro:

```
<filter>
    <filter-name>FiltroSeguranca</filter-name>
    <filter-class>filter.FiltroSeguranca</filter-class>
</filter>
<filter-mapping>
    <filter-name>FiltroSeguranca</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Filtro:

```
HttpServletRequest httpServletRequest = (HttpServletRequest) request;

String url = httpServletRequest .getRequestURI();

HttpSession sessao = httpServletRequest .getSession();

if (sessao.getAttribute("credencial")!=null || url.lastIndexOf("login.jsp")>-1 ||
url.lastIndexOf("AutenticadorCredencialController") >-1 ){
chain.doFilter(request, response);
}else{
((HttpServletResponse) response).sendRedirect("login.jsp");
}
```

Exemplo de implementação em um menu:

```
<tr>
<td <c:if test="${!sessionScope.credencial.acesso.equals('SECRETARIA')}"> hidden</c:if>>
    <a href='PesquisarCursoController'>Manter Cursos</a>
</td>
</tr>
```

PASSO 15 – MANUTENÇÃO CORRETIVA APLICANDO OPÇÕES DE ANO

Alterado a apresentação de ano para ser automática e não com valores estáticos.

Antes:

```
<select name="optAno">
    <option value="0" selected>Todos</option>
    <option value="2013" selected>2013</option>
    <option value="2012" selected>2012</option>
    <option value="2011" selected>2011</option>
    <option value="2010" selected>2010</option>
</select>
```

Depois:

```
<td>
    <jsp:useBean id="data" class="java.util.Date"/>
    <select name="ano" >
            <option value="0" selected>Todos</option>
        <c:forEach var="cont" begin="2006" end="${data.year + 1900}">
            <option value="<c:out value="${cont}"/>"><c:out value="${cont}"/></option>
        </c:forEach>
    </select>
</td>
```