

Tutorial Desenvolvimento de aplicativos WEB com JAVA EE 2

Charles Henrique Porto Ferreira

13 de Outubro de 2015

Conteúdo

1	Introdução	4
2	Problema do $n + 1$ selects	5
3	Frameworks	6
3.1	Hibernate	6
3.2	Enterprise JavaBeans EJB	6
3.2.1	Entity Beans	6
3.2.2	Session Beans	6
3.2.3	Message Driven Beans	7
3.3	Java Server Face - JSF	7
4	Criando o projeto	8
5	Estrutura das classes	11
5.1	Modelo	11
5.2	Controller	12
5.3	Facade	13
5.4	Util	14
6	Páginas Web	15
6.1	Criando o índice - A primeira página	15
6.2	Criando Todas as Páginas	16
7	Colocando a Aplicação para Funcionar	18
8	Apêndice Códigos Fonte	23
8.1	Docente	23
8.2	Disciplina	25
8.3	Turma	27
8.4	HibernateUtil	29
8.5	AbstractFacade	30
8.6	DocenteFacade	32
8.7	DisciplinaFacade	32
8.8	TurmaFacade	33
8.9	DocenteController	33
8.10	DisciplinaController	38
8.11	TurmaController	42
8.12	JsfUtil	46
8.13	DocenteDataModel	48
8.14	DisciplinaDataModel	49

8.15	TurmaDataModel	50
8.16	ViewDocente	51
8.16.1	Create	51
8.16.2	Edit	52
8.16.3	List	53
8.17	ViewDisciplina	55
8.17.1	Create	55
8.17.2	Edit	56
8.17.3	List	57
8.18	ViewTurma	59
8.18.1	Create	59
8.18.2	Edit	60
8.18.3	List	61

1 Introdução

Na primeira parte do tutorial foi exemplificada uma pequena aplicação **CRUD**(Create Remove Update Destroy) usando a IDE *Netbeans* com tecnologia JAVA EE. Grande parte do código foi gerada pelo *Netbeans*, de forma a deixar transparente vários detalhes de programação que envolviam aquela pequena aplicação.

Este segundo tutorial terá como foco desenvolver a mesma aplicação do tutorial anterior, porém minimizando o uso de ferramentas de geração de código. Serão usados somente recursos de ajuda mínima disponibilizados pela IDE, tais como organização dos pacotes e arquivos de metadados para configurações de baixo nível. Um outro aspecto que será abordado neste segundo tutorial é o tratamento do problema $n + 1$ *selects* e a adoção do *framework* **Hibernate** para a solução do problema.

2 Problema do $n + 1$ selects

No exemplo da aplicação CRUD existe um problema de performance que pode passar despercebido devido à simplicidade daquela aplicação, mas que pode se tornar um grande agravante em uma aplicação maior. Ao clicar no *link* para listar as turmas, é aberta uma página com uma tabela onde são mostradas todas as turmas com seus respectivos docentes e disciplinas associados. Para mostrar esses dados, o banco de dados deve ser acessado para retirar tais informações. O problema aparece exatamente neste momento, pois para conseguir os dados de uma turma é necessário encontrar o docente e a disciplina associados a essa turma. Sendo assim, da forma como esta implementado, o algoritmo faz um *select* para buscar todas as turmas e para cada turma faz um novo *select* para buscar seu docente. Se tivermos n docentes totalizaremos $n + 1$ *selects* executados somente para buscar os docentes.

O processo de automatizar o acesso ao banco de dados pelo JPA traz inúmeras vantagens ao programador, deixando transparente vários aspectos de baixo nível na programação. Entretanto algumas otimizações feitas pelo JPA podem ajudar por um lado mas dificultar por outro. Como é o caso do problema do $n + 1$ selects. A solução adotada para resolver esse problema foi usar o framework do Hibernate que fornece otimizações tão boas quanto o JPA porém sem o problema do $n + 1$ selects.

3 Frameworks

Nessa seção, será dada uma breve explicação das ferramentas em destaque deste tutorial.

3.1 Hibernate

Hibernate é um serviço de persistência e *query* Objeto-Relacional de alta performance. A mais flexível e poderosa solução no mercado de Objeto-Relacional, Hibernate se responsabiliza pelo mapeamento das classes para as tabelas do banco de dados de tipo de dados Java para tipo SQL. Isto fornece *query* de dados e facilidades de recuperação de dados que reduzem o tempo de desenvolvimento significativamente. O objetivo principal do Hibernate é aliviar o desenvolvedor em 95% das tarefas de desenvolvimento relacionadas com a persistência, eliminando a tarefa manual do processamento de dados com SQL e JDBC. Entretanto, ao contrário de outras soluções de persistência, Hibernate não esconde o poder do SQL do desenvolvedor e garante que o seu investimento em tecnologia relacional e conhecimento é válido como sempre.

3.2 Enterprise JavaBeans EJB

Enterprise JavaBeans (EJB) é um componente da plataforma JEE que roda em um *container* de um servidor de aplicação. Seu principal objetivo consiste em fornecer um desenvolvimento rápido e simplificado de aplicações Java, com base em componentes distribuídos, transacionais, seguros e portáteis. Atualmente, na versão 3.1, o EJB tem seu futuro definido conjuntamente entre grandes empresas como IBM, Oracle e HP, como também por uma vasta comunidade de programadores numa rede mundial de colaboração sob o portal do JCP.

A grande mudança entre a versão 2.1 e a versão 3.0 se refere à introdução de anotações Java, que facilitam o desenvolvimento, diminuindo a quantidade de código e o uso de arquivos de configuração XML. A plataforma J2EE providencia algumas facilidades dedicadas à camada de lógica de negócio e para o acesso a banco de dados. Através do EJB o programador utiliza a infraestrutura do servidor de aplicação voltada para o desenvolvimento de aplicações de missão crítica (de alta importância para a empresa) e de aplicações empresariais em geral.

O componente EJB possui 3 (três) tipos fundamentais: *Entity beans*, *Session Beans* e *Message Driven Beans*.

3.2.1 Entity Beans

Representa um objeto que vai persistir em uma base de dados ou outra unidade de armazenamento.

3.2.2 Session Beans

Executa uma tarefa para o cliente. Pode manter o estado durante uma sessão com o cliente ou não, subtipos *Stateful* e *Stateless*, respectivamente.

3.2.3 Message Driven Beans

Processa mensagens de modo assíncrono entre os ejb's e cuja API de mensagens é Java Message Service (JMS). Interfaces de Acesso

Para acessar um EJB, é necessário definir as suas interfaces, que podem ser locais ou remotas. Uma interface local define o acesso ao bean somente no computador onde está sendo executado o servidor de aplicação, enquanto uma interface remota permite que o bean seja acessado também por elementos externos

3.3 Java Server Face - JSF

O JSF é suportado por servidores compatíveis com o Java Enterprise Edition 5(Java EE 5), como o GlassFish V2 URL2. Ele é um *framework* de aplicativo *Web* que simplifica o *design* da interface com o usuário de um aplicativo e separa ainda mais a apresentação de um aplicativo Web da sua lógica de negócio. Um *framework* fornece bibliotecas e as vezes ferramentas de software para ajudá-lo a organizar e construir seus aplicativos. O JSF fornece um conjunto de componentes interface com o usuário, ou **componentes JSF**, que simplificam o design de páginas Web. Esses componentes são semelhantes aos componentes Swing utilizados para construir aplicativos GUI. O JSF fornece duas bibliotecas de tags personalizados JSP para adicionar esses componentes a uma página JSP.

O 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 etc. Você cria a aparência e o funcionamento de uma página separadamente em arquivos de código-fonte java relacionados. Embora os componentes JSF padrão sejam suficientes para a maioria dos aplicativos Web básicos, você também pode escrever bibliotecas de componentes personalizados. Bibliotecas de componentes adicionais são disponibilizadas por vários projetos de código-fonte aberto e de fornecedores independentes. Por exemplo, o Oracle fornece quase 100 componentes na sua biblioteca ADF Faces.

4 Criando o projeto

A partir deste ponto será considerado que os programas necessários já foram instalados e devidamente configurados, conforme ensinado no primeiro tutorial.

Crie inicialmente um banco de dados chamado **turmashibernate**. Para isso abra o terminal(Linux),ou commandLine(Windows), do Mysql e digite:

```
create database turmashibernate;
```

```
mysql> create database turmashibernate;
```

Figura 1: Criando um banco de dados

Crie um projeto WEB novo no Netbeans clicando em Arquivo → Novo Projeto → Java Web → Aplicação Web

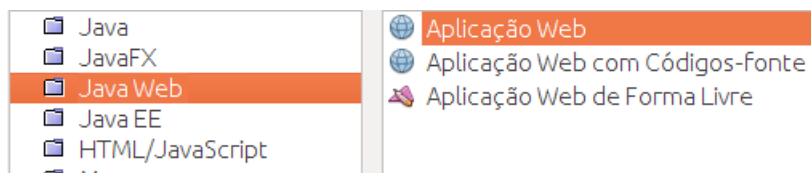


Figura 2: Criando um projeto novo

Dê um nome para o projeto, selecione o Servidor de Aplicações Jboss e a última versão do Java instalada. Durante o desenvolvimento deste tutorial foi usado o Java EE 6.

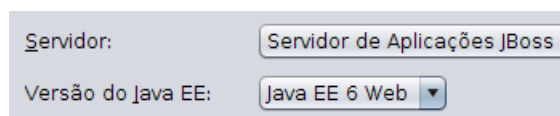


Figura 3: Selecionando o servidor e a versão do Java

Na sessão seguinte selecione o *framework* **JavaServerFace** e o componente **PrimeFaces**.

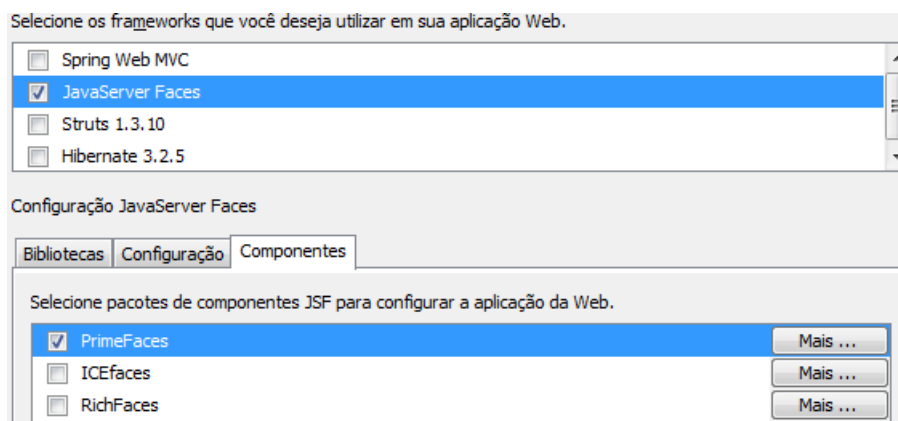


Figura 4: Selecionando o Framework

Em seguida selecione o **Hibernate**, mas não finalize ainda, é necessário configurar a conexão com o banco de dados. Essa etapa poderia ser feita posteriormente, mas é preferível já realizá-la nesse momento.

Configurar Hibernate

1. Conexão ao banco de dados

Em Pacotes de Código-fonte selecione: Novo → Assistente de Configuração do Hibernate, deixe o nome do arquivo como hibernate.cfg e clique em next, selecione: Nova Conexão de Banco de Dados

2. Driver

Mysql (Conector/J driver)

3. Assistente para nova Conexão

Informe o nome do banco de dados, no caso turmashibernate, o nome do usuário e senha para esse banco. É possível verificar se as informações foram inseridas corretamente e se a conexão vai funcionar, clicando em "Testar Conexão".

Personalizar Conexão

Nome do Driver: MySQL (Connector/J driver)

Host: localhost Porta: 3306

Banco de dados: turmashibernate

Nome do Usuário: root

Senha: ****

☐ Lembrar senha

Testar Conexão

JDBC URL: 306/turmashibernate?zeroDateTimeBehavior=convertToNull

Figura 5: Configuração da Conexão com o banco de dados

Após terminada a conexão com o banco de dados vá até a pasta Biblioteca e procure as bibliotecas PrimesFaces 5.0 e Hibernate 4.3.x e as exclua. Após isso adicione as seguintes bibliotecas:

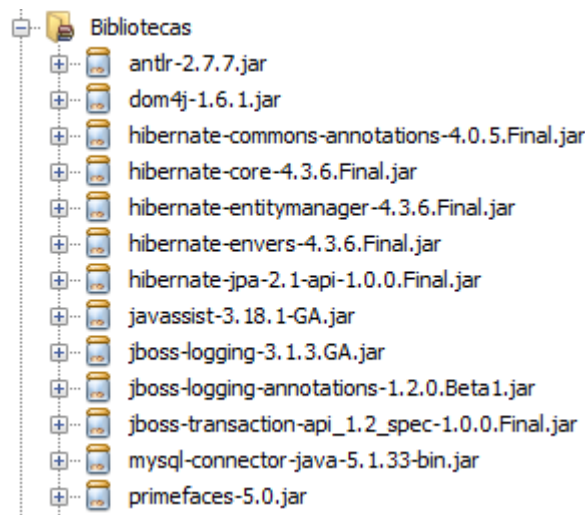


Figura 6: Bibliotecas

Essas bibliotecas podem ser baixadas do repositório desse tutorial no GitHub através dessa [página](#).

Obs.: Foram usadas as versões das bibliotecas mais atuais até o momento do desenvolvimento do código do projeto. Porém, novas versões podem ser lançadas e utilizadas posteriormente.

Agora já podemos finalizar as configurações iniciais.

Na pasta *pacote default* o Netbeans criou um arquivo chamado `hibernate.xfg.xml` o qual contém a configuração do banco de dados. Esse arquivo contém a configuração do banco de dados feita na etapa inicial da criação do projeto.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
   Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/
   hibernate-configuration-3.0.dtd">
3 <hibernate-configuration>
4   <session-factory>
5     <property name="hibernate.dialect">org.hibernate.dialect.
       MySQLDialect</property>
6     <property name="hibernate.connection.driver_class">com.mysql.jdbc.
       Driver</property>
7     <property name="hibernate.connection.url">jdbc:mysql://localhost
       :3306/turmas29?zeroDateTimeBehavior=convertToNull</property>
8     <property name="hibernate.connection.username">root</property>
9     <property name="hibernate.connection.password">root</property>
10   </session-factory>
11 </hibernate-configuration>
```

5 Estrutura das classes

O projeto seguirá a mesma ideia do projeto desenvolvido no primeiro tutorial. Teremos um Modelo MVC e faremos uso do padrão de projeto Facade.

5.1 Modelo

Vamos criar inicialmente um pacote chamado modelo, no qual conterá nossas classes que cuidarão das regras de negócio do sistema.

Crie três classes chamadas Docente 8.1, Disciplina 8.2 e uma última chamada Turma 8.3. Abaixo podemos ver o diagrama de classes que representa essa implementação.

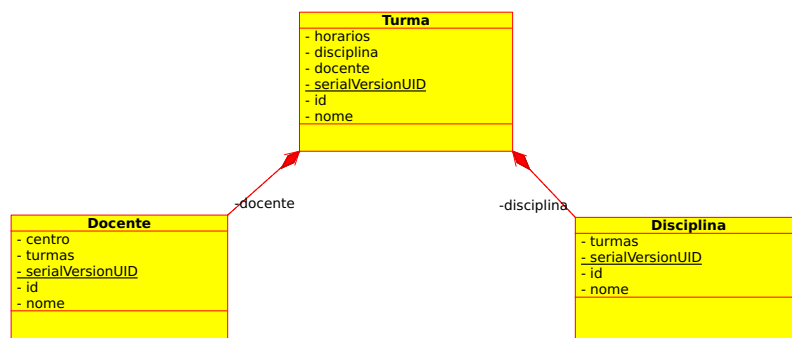


Figura 7: Diagrama de classes

A primeira coisa a ser observada neste exemplo de programa são as *anotações do hibernate* que permitem fazer o mapeamento da classe para uma tabela do banco de dados. As anotações padrões do EJB estão contidas em um pacote chamado javax.persistence.

@Entity Annotation:

A primeira anotação é a *@Entity*. Ela associa o nome da Classe com uma tabela no banco de dados. Logo em seguida usamos a anotação *@Entity* para as classes Docente, Disciplina e Turma, a qual demarca estas classes como *beans* de entidade, logo não deve haver um construtor visível sem argumentos ou no mínimo com escopo *protected*.

@Id e @GeneratedValue Annotations:

Cada entidade terá uma chave primária, que receberá a anotação *@Id*. A chave primária pode ser um campo único ou uma combinação de múltiplos campos dependendo da estrutura da tabela.

Por padrão, a anotação *@Id* irá automaticamente determinar a estratégia de geração de chave primária mais apropriada para ser usada, mas é possível sobreescrevê-la aplicando a anotação *@GeneratedValue*, que recebe dois parâmetros: *strategy* e *generator*.

As anotações *@Id* e *@GeneratedValue* fazem o mapeamento de identificação do banco de dados. O fornecedor de persistência do JPA detecta que a anotação *@Id* está em um campo e

assume que ele deve acessar as propriedades de um objeto diretamente pelos campos em tempos de execução.

@OneToMany Annotation:

A anotação *@OneToMany* faz o mapeamento das turmas no docente.

É importante observar que as classes devem implementar a interface *Serializable*. Agora precisamos identificar o mapeamento das classes no arquivo *Hibernate.cfg.xml*. Abra-o e acrescente as seguintes linhas abaixo da ultima "property":

```
1      <mapping class="modelo.Docente"/>
2      <mapping class="modelo.Disciplina"/>
3      <mapping class="modelo.Turma"/>
```

Para completar a configuração deste arquivo é possível acrescentar alguns parâmetros, não obrigatórios, porém extremamente úteis, como por exemplo o monitoramento das instruções SQL executadas pelo Hibernate. Assim podemos ver como é resolvido o problema do $n + 1$ selects. Para adicionar essa funcionalidade acrescente estes parâmetros no arquivo:

```
1      <property name="hibernate.show_sql">true</property>
2      <property name="hibernate.format_sql">true</property>
3      <property name="hibernate.use_sql_comments">true</property>
4      <property name="hibernate.hbm2ddl.auto">update</property>
```

Outra opção é ir na aba *Design* do arquivo *Hibernate.cfg.xml* e em *Propriedades Opcionais* → *Propriedades de Configuração* → *Adicionar* e adicionar as três primeiras propriedades mostradas anteriormente.

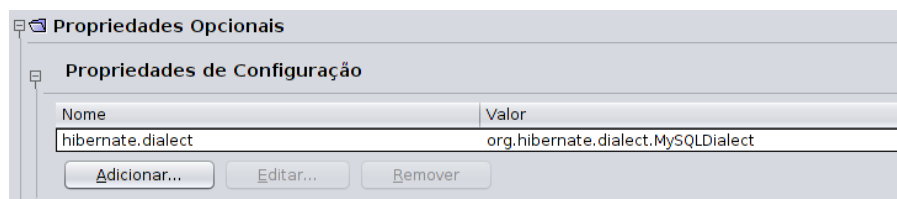


Figura 8: Propriedades de Configuração do Hibernate

Para adicionar a última propriedade ("hibernate.hbm2ddl.auto"), vá em *Propriedades Opcionais* → *Propriedades Diversas* → *Adicionar*, como mostrado na Figura 9.

5.2 Controller

No pacote *controller*, irão residir as classes que farão o controle das páginas de acesso ao usuário. Primeiramente crie uma classe chamada *JsfUtil.java* 8.12. Essa classe cuidará de detalhes específicos das páginas como manipulação de mensagens de erros e a recuperação de itens na página. Essa classe servirá como base para todas as outras que manipularão as páginas JSF.

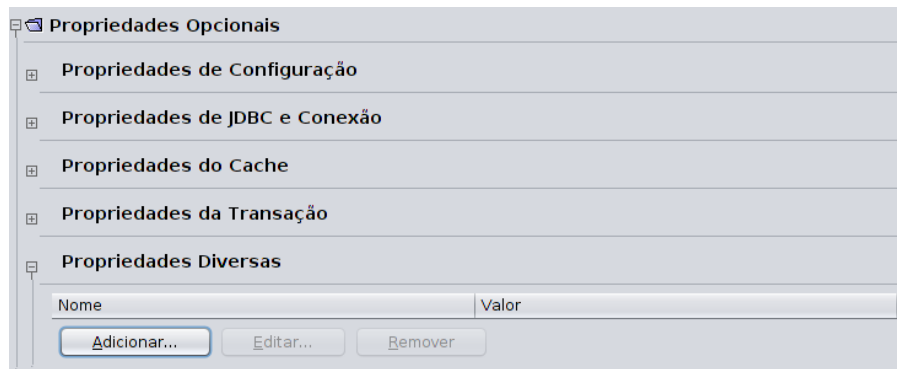


Figura 9: Propriedades diversas do Hibernate

Vamos começar criando a classe `DocenteController` 8.9. Algumas observações devem ser feitas sobre essas classes. Perceba que algumas anotações apareceram agora, como, por exemplo, as duas anotações:

```

1 @Named(value = "docenteController")
2 @SessionScoped
3 public class DocenteController implements Serializable {
4
5     public DocenteController() {
6         docente = new Docente();
7     }
8
9     private Docente docente;
10
11     @EJB
12     private DocenteFacade docenteFacade;
13     private static DocenteDataModel docenteDataModel;

```

A primeira mapeia um objeto da Classe `DocenteController`, que poderá ser usado nas páginas *xhtml*. A segunda especifica que um bean é escopo de sessão. A terceira marcação, **@EJB**, especifica o objeto `docenteController` que será injetado no código, e poderá ser usada nas páginas JSF sem precisar declarar o objeto.

Crie agora as classes `DisciplinaController` 8.10 e `TurmaController` 8.11. Essas três classes não possuem nenhum algoritmo especial, somente métodos para fazer a interação com as páginas e com a camada *facade*, que fará acesso ao banco de dados.

5.3 Facade

A camada *facade* fará toda a parte de comunicação com o banco de dados. Como as três classes que iremos trabalhar irão interagir com o banco, iremos criar uma classe abstrata, *AbstractFacade*, 8.5 que irá generalizar as funcionalidades de inserção, remoção e atualização do banco de dados. Essa classe contém as funcionalidades básicas de um sistema CRUD em

uma banco de dados.

Crie agora a classe `DocenteFacade` 8.6, que herdará da classe *AbstractFacade*. Nesta classe é usada a marcação `@Stateless` que indica que o estado da sessão do *session bean* não devera ser mantido. Nela também é implementado o método *getSectionFactory*, método abstrato declarado pela classe *AbstractFacade*. Este método faz uso da função *HibernateUtil.getSessionFactory()* que retorna um objeto *sectionFactory*. Em seguida crie as classes `DisciplinaFacade` 8.7 e `TurmaFacade` 8.8, seguindo os mesmos passos utilizados para a criação da classe `DocenteFacade`.

Implemente a classe `HibernateUtil` 8.4, para poder prosseguir no tutorial. Em seguida crie a Classe `DocenteFacade` 8.6 `DisciplinaFacade` 8.7 e `TurmaFacade` 8.8.

5.4 Util

Também será necessário criar um pacote chamado *util*. Ele será criado do mesmo modo que os outros pacotes. Esse pacote possui os `DataModels` das entidades. O `DataModel` armazena uma coleção de objetos da entidade que irão popular o componente `DataTable` do `Primefaces` das páginas *web*.

Dentro do pacote iremos criar uma classe chamada `DocenteDataModel` 8.13. Essa classe irá interagir com a classe `DocenteController`.

Após isso também é necessário criar as classes `DisciplinaDataModel` 8.14 e `TurmaDataModel` 8.15.

6 Páginas Web

Agora podemos começar a modelar as páginas *web*. Embora tenhamos feito muito código antes de chegar nesta parte, todo este trabalho começará a fazer sentido agora. Fazer páginas *web* não requer muito trabalho, mas pode causar uma certa insegurança inicial para quem nunca trabalhou com este tipo de tecnologia. Iremos fazer todo o desenho via código, sem usar ferramentas "drag and drop", assim ficará claro o que cada linha código significa.

Obs.:É fortemente recomendado que qualquer código deste tutorial seja digitado pelo programador e não simplesmente copiado e colado. Ao fazer a digitação as funcionalidades de autocomplete da IDE irão poder te guiar se está saindo tudo certo ou se tem algo errado.

6.1 Criando o index - A primeira página

Como já deve ter sido notado, o **index** é criado automaticamente pelo Netbeans. Vamos fazer algumas alterações nele e deixá-lo da seguinte forma:

```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
   www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:h="http://java.sun.com/jsf/html"
5     xmlns:f="http://java.sun.com/jsf/core"
6     xmlns:p="http://primefaces.org/ui">
7     <h:head>
8         <title>Facelet Title</title>
9     </h:head>
10    <h:body>
11        <f:view>
12            <h:form style="background-color: transparent">
13                <p:panel id="conpnl" header="Index" style="width:280px
                   ;height:300px; border-color: transparent; position:
                   relative; left: 530px; text-align: center;
                   background-color: transparent">
14                    <br/>
15                    <p:panelMenu style="width:250px; height: 300px;
                   text-align: justify; position: relative; ">
16                        <p:submenu label="Docentes">
17                            <p:menuitem value="Listar todos os docentes
                               " action="/view/docente/List" icon="ui-
                               icon-document" />
```

```

18         <p:menuitem value="Cadastrar um docente"
19             action="#{docenteController.
20                 prepareCreate()}" icon="ui-icon-disk"/>
21     </p:submenu>
22     <p:submenu label="Disciplinas">
23         <p:menuitem value="Listar todas as
24             disciplinas" action="/view/disciplina/
25             List" icon="ui-icon-document" />
26         <p:menuitem value="Cadastrar uma disciplina
27             " action="#{disciplinaController.
28                 prepareCreate()}" icon="ui-icon-disk"/>
29     </p:submenu>
30     <p:submenu label="Turmas">
31         <p:menuitem value="Listar todas as turmas"
32             action="/view/turma/List" icon="ui-icon-
33             document" />
34         <p:menuitem value="Cadastrar uma turma"
35             action="#{turmaController.prepareCreate
36                 ()}" icon="ui-icon-disk"/>
37     </p:submenu>
38 </p:panelMenu>
39 </p:panel>
40 </h:form>
41 </f:view>
42 </h:body>
43 </html>

```

Esta é uma página bem simples que gera três links que direcionarão para outras três páginas. Vamos criar uma estrutura de pastas para colocar as demais páginas que farão parte dessa aplicação.

Crie uma pasta chamada **view**. Dentro dela coloque outras 3 pastas, **docente**, **disciplina** e **turma**.

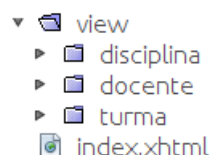


Figura 10: Estrutura das pastas

6.2 Criando Todas as Páginas

Nesse ponto, os links ainda não responderão, pois as páginas para as quais eles direcionam ainda não estão criadas. Vamos criá-las agora. Na pasta **docente** crie os seguintes arquivos:

Create.xhtml 8.16.1, Edit.xhtml 8.16.2, List.xhtml 8.16.3.

O código destas páginas possui as particularidades da linguagem *html* e está fora do escopo deste segundo tutorial explicá-las detalhadamente. Entretanto, é preciso chamar a atenção da forma que os métodos da classe `DocenteController` são chamados:

```
1 #{docenteController.docente.nome}
```

No exemplo acima é acessado o objeto `docenteController` que acessa o método `getDocente` e depois `getNome`.

Faça a mesma coisa nas pastas **disciplina** e **turma**, criando os arquivos Create 8.17.1, Edit 8.17.2, List 8.17.3 para Disciplina e novamente Create 8.18.1, Edit 8.18.2, List 8.18.3 para Turma.

7 Colocando a Aplicação para Funcionar

Assim como no tutorial anterior este programa tem funcionalidades bem simples que podem ser testadas criando ou excluindo Docentes, Disciplinas ou Turmas. Ao rodar a aplicação, deve aparecer uma página de índice.

index	
	Mostrar todos os Docentes
	Mostrar todas as Disciplinas
	Mostrar todas as Turmas

Figura 11: Pagina inicial

Para testar o código vamos criar um docente, uma disciplina e uma turma. Para criar o docente vá em Mostrar todos os docentes → Criar um Docente. Insira o nome do docente e clique em Cadastrar Docente.

Criar um novo registro

Nome : *

[Cadastrar Docente](#)

[Ver todos os docentes](#)

[Voltar ao índice](#)

Figura 12: Cadastro de um docente

Se o cadastro foi realizado com sucesso, aparecerá a seguinte mensagem:

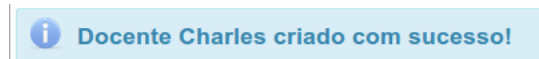


Figura 13: Cadastramento realizado com sucesso

Para visualizar o docente recém criado, clique em Ver todos os docentes.

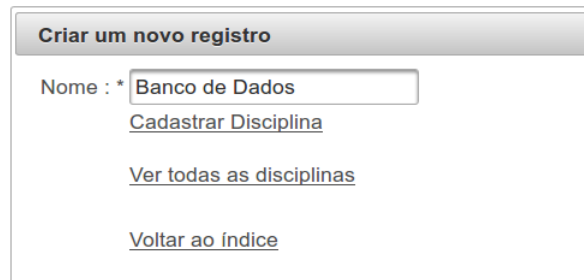
Lista de Docentes			
(1 of 1)		<input type="button" value="1"/>	
id	Nome		
1	Charles	Deletar	Editar
(1 of 1)		<input type="button" value="1"/>	

[Criar um Docente](#)

[Voltar ao índice](#)

Figura 14: Visualização de todos os docentes

Para criar uma disciplina vá em Mostrar todas as disciplinas → Criar uma Disciplina. Insira o nome da disciplina e clique em Cadastrar Disciplina.



The form is titled "Criar um novo registro". It contains a text input field for "Nome" with the value "Banco de Dados". Below the input field are three links: "Cadastrar Disciplina", "Ver todas as disciplinas", and "Voltar ao índice".

Figura 15: Cadastro de uma disciplina

Se o cadastro foi realizado com sucesso, aparecerá a uma mensagem semelhante à exibida no cadastro do Docente. Para visualizar a disciplina criada, clique em Ver todas as disciplinas.



The table is titled "Lista de Disciplinas". It shows a single record with id 1 and name "Banco de Dados". The table has columns for id, Nome, Deletar, and Editar. The record is highlighted in blue.

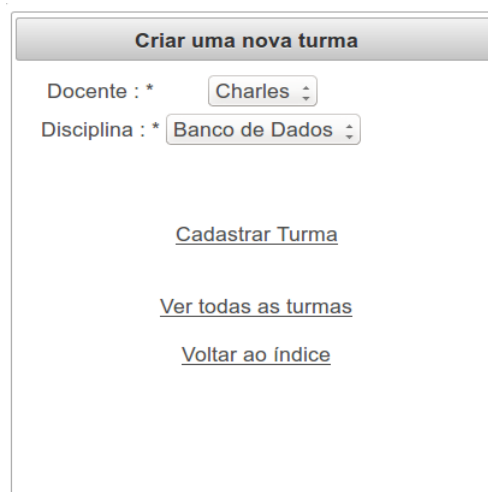
Lista de Disciplinas			
(1 of 1)			
id	Nome		
1	Banco de Dados	Deletar	Editar

[Criar uma Disciplina](#)

[Voltar ao índice](#)

Figura 16: Visualização de todas as disciplinas

Finalmente, para criar uma turma vá em Mostrar todas as turmas → Criar uma Turma. Escolha um docente e uma disciplina para essa turma nas *combo boxes* e clique em Cadastrar Turma.



The form is titled "Criar uma nova turma". It contains two dropdown menus: "Docente" with the value "Charles" and "Disciplina" with the value "Banco de Dados". Below the dropdowns are three links: "Cadastrar Turma", "Ver todas as turmas", and "Voltar ao índice".

Figura 17: Cadastro de uma turma

Se o cadastro foi realizado com sucesso, aparecerá a uma mensagem semelhante à exibida nos cadastros do Docente e da Disciplina. Para visualizar a turma criada, clique em Ver todas as turmas.

Lista de Turmas				
(1 of 1) << 1 >>				
id	Docente	Disciplina		
1	Charles	Banco de Dados	Deletar	Editar
(1 of 1) << 1 >>				

[Criar uma turma](#)

[Voltar ao índice](#)

Figura 18: Visualização de todas as turmas

Nesse ponto, podemos chamar a atenção para o problema do $n + 1$ *selects* que foi abordado no começo deste tutorial. É fácil verificar como ele foi resolvido pelo *Hibernate*, através do **carregamento ansioso** das associações. Abra a página para Listar todas as turmas e veja saída do Hibernate, mostrando a instrução SQL gerada:

```

1  /* criteria query */ select
2      this_.ID as ID1_2_2_,
3      this_.disciplina_ID as discipli2_2_2_,
4      this_.docente_ID as docente_3_2_2_,
5      disciplina2_.ID as ID1_0_0_,
6      disciplina2_.nome as nome2_0_0_,
7      docente3_.ID as ID1_1_1_,
8      docente3_.nome as nome2_1_1_
9  from
10     Turma this_
11  left outer join
12     Disciplina disciplina2_
13         on this_.disciplina_ID=disciplina2_.ID
14  left outer join
15     Docente docente3_
16         on this_.docente_ID=docente3_.ID

```

O carregamento ansioso é habilitado através do parâmetro `fetch = FetchType.EAGER` da notação de associação `@OneToMany` presente nas classes `Docente` 8.1 e `Disciplina` 8.2, para fazer a associação das mesmas com a classe `Turma`. Dessa forma, ao invés de fazer vários *selects* individuais para recuperar o docente e a disciplina de cada turma, o *Hibernate* faz um ***left outer join*** para buscar todos os dados com um única instrução.

Para ilustrar as funções `Edit` e `Delete`, vamos criar mais um docente.

Lista de Docentes			
(1 of 1) << 1 >> >>>			
id	Nome		
1	Charles	Deletar	Editar
2	Julia	Deletar	Editar
(1 of 1) << 1 >> >>>			

[Criar um Docente](#)

[Voltar ao índice](#)

Figura 19: Visualização de todos os docentes

Ao clicar em Editar, é aberta uma nova página, onde é possível editar as informações do docente escolhido. Vamos trocar o nome para Juliana e clicar em Editar Docente.

Editar um docente

Nome : *

[Editar Docente](#)

[Ver todos os docentes](#)

[Voltar ao índice](#)

Figura 20: Editar Docente

Após a confirmação de que o docente foi editado com sucesso, podemos clicar em Ver todos os docentes para verificar se ele realmente foi modificado.

Lista de Docentes			
(1 of 1) << 1 >> >>>			
id	Nome		
1	Charles	Deletar	Editar
2	Juliana	Deletar	Editar
(1 of 1) << 1 >> >>>			

[Criar um Docente](#)

[Voltar ao índice](#)

Figura 21: Visualização de todos os docentes

Para apagar o docente, basta clicar em deletar.



Figura 22: Visualização de todos os docentes após o delete

8 Apêndice Códigos Fonte

Esse Apêndice tem o código de todas as classes java e páginas web desenvolvidos no tutorial.

8.1 Docente

```
1 package model;
2
3 import java.io.Serializable;
4 import java.util.List;
5 import javax.persistence.CascadeType;
6 import javax.persistence.Entity;
7 import javax.persistence.FetchType;
8 import javax.persistence.GeneratedValue;
9 import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.OneToMany;
12
13 @Entity
14 public class Docente implements Serializable {
15
16     private static final long serialVersionUID = 1L;
17
18     public Docente() {
19
20     }
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Long ID;
25
26     public Long getID() {
27         return ID;
28     }
29
30     public void setID(Long ID) {
31         this.ID = ID;
32     }
33
34     private String nome;
35
36     public String getNome() {
```

```
37         return nome;
38     }
39
40     public void setNome(String nome) {
41         this.nome = nome;
42     }
43
44     @OneToMany(cascade = CascadeType.ALL, mappedBy = "docente", fetch =
45         FetchType.EAGER)
46     private List<Turma> turmas;
47
48     public List<Turma> getTurmas() {
49         return turmas;
50     }
51
52     public void setTurmas(List<Turma> turmas) {
53         this.turmas = turmas;
54     }
55
56     @Override
57     public int hashCode() {
58         int hash = 0;
59         hash += (ID != null ? ID.hashCode() : 0);
60         return hash;
61     }
62
63     @Override
64     public boolean equals(Object object) {
65
66         if (!(object instanceof Docente)) {
67             return false;
68         }
69
70         Docente other = (Docente) object;
71         if ((this.ID == null && other.ID != null) || (this.ID != null
72             && !(this.ID.equals(other.ID)))) {
73             return false;
74         }
75
76         return true;
77     }
```



```
78
79     @Override
80     public String toString() {
81         return this.nome;
82     }
83
84 }
```

8.2 Disciplina

```
1  package model;
2
3  import java.io.Serializable;
4  import java.util.List;
5  import javax.persistence.CascadeType;
6  import javax.persistence.Entity;
7  import javax.persistence.FetchType;
8  import javax.persistence.GeneratedValue;
9  import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.OneToMany;
12
13 @Entity
14 public class Disciplina implements Serializable {
15
16     private static final long serialVersionUID = 1L;
17
18     public Disciplina() {
19
20     }
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Long ID;
25
26     public Long getID() {
27         return ID;
28     }
29
30     public void setID(Long ID) {
31         this.ID = ID;
32     }
33
34     private String nome;
```

```
35
36     public String getNome() {
37         return nome;
38     }
39
40     public void setNome(String nome) {
41         this.nome = nome;
42     }
43
44     @OneToMany(cascade = CascadeType.ALL, mappedBy = "disciplina",
45         fetch = FetchType.EAGER)
46     private List<Turma> turmas;
47
48     public List<Turma> getTurmas() {
49         return turmas;
50     }
51
52     public void setTurmas(List<Turma> turmas) {
53         this.turmas = turmas;
54     }
55
56     @Override
57     public int hashCode() {
58         int hash = 0;
59         hash += (ID != null ? ID.hashCode() : 0);
60         return hash;
61     }
62
63     @Override
64     public boolean equals(Object object) {
65
66         if (!(object instanceof Disciplina)) {
67             return false;
68         }
69
70         Disciplina other = (Disciplina) object;
71         if ((this.ID == null && other.ID != null) || (this.ID != null
72             && !(this.ID.equals(other.ID)))) {
73             return false;
74         }
75         return true;
```

```
76
77     }
78
79     @Override
80     public String toString() {
81         return this.nome;
82     }
83
84 }
```

8.3 Turma

```
1  package model;
2
3  import java.io.Serializable;
4  import javax.persistence.Entity;
5  import javax.persistence.GeneratedValue;
6  import javax.persistence.GenerationType;
7  import javax.persistence.Id;
8  import javax.persistence.ManyToOne;
9
10 @Entity
11 public class Turma implements Serializable {
12
13     public Turma() {
14
15     }
16
17     private static final long serialVersionUID = 1L;
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Long ID;
22
23     public Long getID() {
24         return ID;
25     }
26
27     public void setID(Long ID) {
28         this.ID = ID;
29     }
30
31     @ManyToOne
32     private Disciplina disciplina;
```

```
33
34     public Disciplina getDisciplina() {
35         return disciplina;
36     }
37
38     public void setDisciplina(Disciplina disciplina) {
39         this.disciplina = disciplina;
40     }
41
42     @ManyToOne
43     private Docente docente;
44
45     public Docente getDocente() {
46         return docente;
47     }
48
49     public void setDocente(Docente docente) {
50         this.docente = docente;
51     }
52
53     @Override
54     public int hashCode() {
55         int hash = 0;
56         hash += (ID != null ? ID.hashCode() : 0);
57         return hash;
58     }
59
60
61     @Override
62     public boolean equals(Object object) {
63
64         if (!(object instanceof Turma)) {
65             return false;
66         }
67
68         Turma other = (Turma) object;
69         if ((this.ID == null && other.ID != null) || (this.ID != null
70             && !(this.ID.equals(other.ID)))) {
71             return false;
72         }
73
74         return true;
```

```
75     }
76
77     @Override
78     public String toString() {
79         return ("model.Turma[ id = " + this.ID + " ]");
80     }
81
82 }
```

8.4 HibernateUtil

```
1
2 package controller;
3
4 import org.hibernate.Session;
5 import org.hibernate.SessionFactory;
6 import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
7 import org.hibernate.cfg.Configuration;
8
9
10 public class HibernateUtil {
11
12     private static final SessionFactory sessionFactory;
13
14     static {
15         try {
16
17
18             Configuration configuration = new Configuration();
19             configuration.configure("hibernate.cfg.xml");
20             StandardServiceRegistryBuilder ssrb = new
21                 StandardServiceRegistryBuilder().applySettings(
22                     configuration.getProperties());
23             sessionFactory = configuration.buildSessionFactory(ssrb.
24                 build());
25             Session session = sessionFactory.openSession();
26
27         } catch (Throwable ex) {
28             // Log the exception.
29             System.err.println("Initial SessionFactory creation failed.
30                 " + ex);
31             throw new ExceptionInInitializerError(ex);
32         }
33     }
34 }
```

```
30
31     public static SessionFactory getSessionFactory() {
32         return sessionFactory;
33     }
34 }
```

8.5 AbstractFacade

```
1
2 package facade;
3
4 import java.util.List;
5 import org.hibernate.Criteria;
6 import org.hibernate.Session;
7 import org.hibernate.SessionFactory;
8 import org.hibernate.Transaction;
9 import org.hibernate.criterion.Projections;
10
11
12 public abstract class AbstractFacade<T> {
13
14     private Class<T> entityClass;
15
16     public AbstractFacade(Class<T> entityClass) {
17         this.entityClass = entityClass;
18     }
19
20     protected abstract SessionFactory getSessionFactory();
21
22     public void save(T entity) {
23         Session session = getSessionFactory().openSession();
24         Transaction transaction = session.beginTransaction();
25         session.save(entity);
26         transaction.commit();
27         session.close();
28     }
29
30     public void edit(T entity) {
31         Session session = getSessionFactory().openSession();
32         Transaction transaction = session.beginTransaction();
33         session.update(entity);
34         transaction.commit();
35         session.close();
36     }
}
```

```
37
38     public void remove(T entity) {
39         Session session = getSessionFactory().openSession();
40         Transaction transaction = session.beginTransaction();
41         session.delete(entity);
42         transaction.commit();
43         session.close();
44     }
45
46     public T find(Long id) {
47         Session session = getSessionFactory().openSession();
48         T entity = (T) session.get(entityClass, id);
49         session.close();
50         return entity;
51     }
52
53     public List<T> findAll() {
54         Session session = getSessionFactory().openSession();
55         Criteria crit = session.createCriteria(entityClass);
56         crit.setMaxResults(50);
57         List results = crit.list();
58         session.close();
59         return results;
60     }
61
62     public List<T> findRange(int[] range) {
63         Session session = getSessionFactory().openSession();
64         Criteria crit = session.createCriteria(entityClass);
65         crit.setMaxResults(range[1] - range[0]);
66         crit.setFirstResult(range[0]);
67         List results = crit.list();
68         session.close();
69         return results;
70     }
71
72     public int count() {
73         Session session = getSessionFactory().openSession();
74         Criteria crit = session.createCriteria(entityClass);
75         int count = ((Number)crit.setProjection(Projections.rowCount())
76             .uniqueResult()).intValue();
77         session.close();
78         return count;
79     }
```

79 }

8.6 DocenteFacade

```
1 package facade;
2
3 import controller.HibernateUtil;
4 import javax.ejb.Stateless;
5 import modelo.Docente;
6 import org.hibernate.SessionFactory;
7
8 @Stateless
9 public class DocenteFacade extends AbstractFacade<Docente> {
10
11     @Override
12     protected SessionFactory getSessionFactory() {
13         return HibernateUtil.getSessionFactory();
14     }
15
16     public DocenteFacade() {
17         super(Docente.class);
18     }
19 }
```

8.7 DisciplinaFacade

```
1 package facade;
2
3 import controller.HibernateUtil;
4 import javax.ejb.Stateless;
5 import modelo.Disciplina;
6 import org.hibernate.SessionFactory;
7
8 @Stateless
9 public class DisciplinaFacade extends AbstractFacade<Disciplina> {
10
11     public DisciplinaFacade() {
12         super(Disciplina.class);
13     }
14
15     @Override
16     protected SessionFactory getSessionFactory() {
17         return HibernateUtil.getSessionFactory();
18     }
19 }
```


8.8 TurmaFacade

```
1 package facade;
2
3 import controller.HibernateUtil;
4 import javax.ejb.Stateless;
5 import modelo.Turma;
6 import org.hibernate.SessionFactory;
7
8 @Stateless
9 public class TurmaFacade extends AbstractFacade<Turma> {
10
11     public TurmaFacade() {
12         super(Turma.class);
13     }
14
15     @Override
16     protected SessionFactory getSessionFactory() {
17         return HibernateUtil.getSessionFactory();
18     }
19 }
```

8.9 DocenteController

```
1 package controller;
2
3 import facade.DocenteFacade;
4 import java.io.Serializable;
5 import java.util.List;
6 import javax.ejb.EJB;
7 import javax.enterprise.context.SessionScoped;
8 import javax.faces.component.UIComponent;
9 import javax.faces.context.FacesContext;
10 import javax.faces.convert.Converter;
11 import javax.faces.convert.FacesConverter;
12 import javax.faces.model.SelectItem;
13 import javax.inject.Named;
14 import model.Docente;
15 import model.Turma;
16 import util.DocenteDataModel;
17
18 @Named(value = "docenteController")
19 @SessionScoped
20 public class DocenteController implements Serializable {
```

```
21
22     public DocenteController() {
23         docente = new Docente();
24     }
25
26     //Guarda o docente atual
27     private Docente docente;
28
29     @EJB
30     private DocenteFacade docenteFacade;
31     private static DocenteDataModel docenteDataModel;
32
33     public void setDocente(Docente docente) {
34         this.docente = docente;
35     }
36
37     private Docente getDocente(Long key) {
38         return this.find(key);
39     }
40
41
42     public Docente getDocente() {
43         if (docente == null) {
44             docente = new Docente();
45         }
46         return docente;
47     }
48
49     public List<Turma> getTurmas() {
50         return docente.getTurmas();
51     }
52
53     public SelectItem[] getItemsAvaivableSelectOne() {
54         return JsUtil.getSelectItems(docenteFacade.findAll(), true);
55     }
56
57     //Data Model*****
58
59     public DocenteDataModel getDocenteDataModel() {
60         if (DocenteController.docenteDataModel == null) {
61
62             List<Docente> docentes = this.findAll();
```

```
63         DocenteController.docenteDataModel = new DocenteDataModel(  
64             docentes);  
65     }  
66     return DocenteController.docenteDataModel;  
67 }  
68  
69 public static void setDocenteDataModel(DocenteDataModel  
70     docenteDataModel) {  
71     DocenteController.docenteDataModel = docenteDataModel;  
72 }  
73  
74 public void recriarModelo() {  
75     DocenteController.docenteDataModel = null;  
76 }  
77  
78 // "Prepares" das paginas*****  
79  
80 public String prepareCreate() {  
81     docente = new Docente();  
82     return "Create";  
83 }  
84  
85 public String prepareEdit() {  
86     docente = (Docente) docenteDataModel.getRowData();  
87     return "Edit";  
88 }  
89  
90 public String prepareList(){  
91     recriarModelo();  
92     return "List";  
93 }  
94  
95 // CRUD*****  
96 public String create() {  
97  
98     try {  
99         docenteFacade.save(docente);  
100         JsUtil.addSuccessMessage("Docente " + docente.getNome() +  
101             " criado com sucesso!");  
102         return prepareCreate();  
103     } catch (Exception e) {  
104         JsUtil.addErrorMessage(e, "Ocorreu um erro de persistencia  
105             para salvar o docente " + docente.getNome());
```

```
102         return null;
103     }
104 }
105
106 public Docente find(Long id) {
107
108     return docenteFacade.find(id);
109 }
110
111 private List<Docente> findAll() {
112     return docenteFacade.findAll();
113 }
114 }
115
116 public void update() {
117     try {
118         docenteFacade.edit(docente);
119         JsفUtil.addSuccessMessage("Docente Editado com sucesso!");
120         docente = null;
121     } catch (Exception e) {
122         JsفUtil.addErrorMessage(e, "Ocorreu um erro de persistencia
123             , nao foi possivel editar o docente: " + e.getMessage());
124     }
125 }
126
127 public void delete() {
128     docente = (Docente) docenteDataModel.getRowData();
129     try {
130         docenteFacade.remove(docente);
131         docente = null;
132         JsفUtil.addSuccessMessage("Docente Deletado");
133     } catch (Exception e) {
134         JsفUtil.addErrorMessage(e, "Ocorreu um erro de persistencia
135             : " + e.getMessage() + " para deletar o docente " +
136             docente.getNome());
137     }
138     recriarModelo();
139 }
140
141 // *****
```

```
141
142 @FacesConverter(forClass = Docente.class)
143 public static class DocenteControllerConverter implements Converter
144 {
145     @Override
146     public Object getAsObject(FacesContext facesContext,
147         UIComponent component, String value) {
148         if (value == null || value.length() == 0) {
149             return null;
150         }
151         DocenteController controller = (DocenteController)
152             facesContext.getApplication().getELResolver().
153                 getValue(facesContext.getELContext(), null, "
154                     docenteController");
155         return controller.getDocente(getKey(value));
156     }
157
158     java.lang.Long getKey(String value) {
159         java.lang.Long key;
160         key = Long.valueOf(value);
161         return key;
162     }
163
164     String getStringKey(java.lang.Long value) {
165         StringBuilder sb = new StringBuilder();
166         sb.append(value);
167         return sb.toString();
168     }
169
170     @Override
171     public String getAsString(FacesContext facesContext,
172         UIComponent component, Object object) {
173         if (object == null) {
174             return null;
175         }
176         if (object instanceof Docente) {
177             Docente d = (Docente) object;
178             return getStringKey(d.getID());
179         } else {
180             throw new IllegalArgumentException("object " + object +
181                 " is of type " + object.getClass().getName() + ";
182                 expected type: " + Docente.class.getName());
183         }
184     }
185 }
```

```
177         }
178     }
179 }
180
181 }
```

8.10 DisciplinaController

```
1 package controller;
2
3 import facade.DisciplinaFacade;
4 import java.io.Serializable;
5 import java.util.List;
6 import javax.ejb.EJB;
7 import javax.enterprise.context.SessionScoped;
8 import javax.faces.component.UIComponent;
9 import javax.faces.context.FacesContext;
10 import javax.faces.convert.Converter;
11 import javax.faces.convert.FacesConverter;
12 import javax.faces.model.SelectItem;
13 import javax.inject.Named;
14 import model.Disciplina;
15 import util.DisciplinaDataModel;
16
17 @Named(value = "disciplinaController")
18 @SessionScoped
19 public class DisciplinaController implements Serializable {
20
21     private Disciplina disciplina;
22
23     @EJB
24     DisciplinaFacade disciplinaFacade;
25     private static DisciplinaDataModel disciplinaDataModel;
26
27     public DisciplinaController() {
28         this.disciplina = new Disciplina();
29     }
30
31     public Disciplina getDisciplina() {
32         if (disciplina == null) {
33             disciplina = new Disciplina();
34         }
35         return disciplina;
36     }
37 }
```

```
37
38     private Disciplina getDisciplina(Long key) {
39         return this.find(key);
40     }
41
42     public SelectItem[] getItemsAvaivableSelectOne() {
43         return JsUtil.getSelectItems(disciplinaFacade.findAll(), true)
44         ;
45     }
46
47     public void setDisciplina(Disciplina disciplina) {
48         this.disciplina = disciplina;
49     }
50
51     //Data Model*****
52
53     public DisciplinaDataModel getDisciplinaDataModel() {
54         if (disciplinaDataModel == null) {
55             List<Disciplina> disciplinas = this.findAll();
56             disciplinaDataModel = new DisciplinaDataModel(disciplinas);
57         }
58         return disciplinaDataModel;
59     }
60
61     public static void setDisciplinaDataModel(DisciplinaDataModel
62     disciplinaDataModel) {
63         DisciplinaController.disciplinaDataModel = disciplinaDataModel;
64     }
65
66     public void recriarModelo() {
67         DisciplinaController.disciplinaDataModel = null;
68     }
69
70     // "Prepares" das paginas*****
71
72     public String prepareCreate() {
73         disciplina = new Disciplina();
74         return "Create";
75     }
76
77     public String prepareEdit() {
78         disciplina = (Disciplina) disciplinaDataModel.getRowData();
79         return "Edit";
80     }
```

```
78     }
79
80     public String prepareList(){
81         recriarModelo();
82         return "List";
83     }
84
85     //CRUD*****
86
87     public String create() {
88
89         try {
90             disciplinaFacade.save(disciplina);
91             JsفUtil.addSuccessMessage("Disciplina " + disciplina.
92                 getNome() + " criada com sucesso!");
93             return prepareCreate();
94         } catch (Exception e) {
95             JsفUtil.addErrorMessage(e, "Ocorreu um erro de persistencia
96                 para salvar a disciplina " + disciplina.getNome());
97             return null;
98         }
99     }
100
101     public Disciplina find(Long id) {
102
103         return disciplinaFacade.find(id);
104     }
105
106
107     public List<Disciplina> findAll() {
108         return disciplinaFacade.findAll();
109     }
110
111     public void update() {
112         try {
113             disciplinaFacade.edit(disciplina);
114             JsفUtil.addSuccessMessage("Disciplina editada com sucesso!"
115                 );
116             disciplina = null;
117         } catch (Exception e) {
```



```
118         JsفUtil.addErrorMessage(e, "Ocorreu um erro: " + e.
119             getMessage() + " para editar a disciplina " + disciplina
120             .getNome());
121     }
122
123     public void delete() {
124         disciplina = (Disciplina) disciplinaDataModel.getRowData();
125         try {
126             disciplinaFacade.remove(disciplina);
127             disciplina = null;
128             JsفUtil.addSuccessMessage("Disciplina Deletada");
129         } catch (Exception e) {
130             JsفUtil.addErrorMessage(e, "Ocorreu um erro de persistencia
131                 para deletar a disciplina " + disciplina.getNome());
132         }
133         recriarModelo();
134     }
135
136     // *****
137     @FacesConverter(forClass = Disciplina.class)
138     public static class DisciplinaControllerConverter implements
139         Converter {
140
141         @Override
142         public Object getAsObject(FacesContext facesContext,
143             UIComponent component, String value) {
144             if (value == null || value.length() == 0) {
145                 return null;
146             }
147             DisciplinaController controller = (DisciplinaController)
148                 facesContext.getApplication().getELResolver().
149                 getValue(facesContext.getELContext(), null, "
150                     disciplinaController");
151             return controller.getDisciplina(getKey(value));
152         }
153
154         java.lang.Long getKey(String value) {
155             java.lang.Long key;
156             key = Long.valueOf(value);
157             return key;
158         }
159     }
```

```
154     }
155
156     String getStringKey(java.lang.Long value) {
157         StringBuilder sb = new StringBuilder();
158         sb.append(value);
159         return sb.toString();
160     }
161
162     @Override
163     public String getAsString(FacesContext facesContext,
164         UIComponent component, Object object) {
165         if (object == null) {
166             return null;
167         }
168         if (object instanceof Disciplina) {
169             Disciplina d = (Disciplina) object;
170             return getStringKey(d.getID());
171         } else {
172             throw new IllegalArgumentException("object " + object +
173                 " is of type " + object.getClass().getName() + ";
174                 expected type: " + Disciplina.class.getName());
175         }
176     }
177 }
```

8.11 TurmaController

```
1 package controller;
2
3 import facade.TurmaFacade;
4 import java.io.Serializable;
5 import java.util.List;
6 import javax.ejb.EJB;
7 import javax.enterprise.context.SessionScoped;
8 import javax.faces.component.UIComponent;
9 import javax.faces.context.FacesContext;
10 import javax.faces.convert.Converter;
11 import javax.faces.convert.FacesConverter;
12 import javax.inject.Named;
13 import model.Turma;
14 import util.TurmaDataModel;
15
```

```
16 @Named(value = "turmaController")
17 @SessionScoped
18 public class TurmaController implements Serializable {
19
20     private Turma turma;
21
22     @EJB
23     private TurmaFacade turmaFacade;
24     private TurmaDataModel turmaDataModel;
25
26     public TurmaController() {
27         turma = new Turma();
28
29     }
30
31     public Turma getTurma() {
32         if (turma == null) {
33             turma = new Turma();
34         }
35         return turma;
36     }
37
38     private Turma getTurma(Long key) {
39         return this.find(key);
40
41     }
42
43     public void setTurma(Turma turma) {
44         this.turma = turma;
45     }
46
47     // "Prepares" das paginas*****
48     public String prepareCreate() {
49         turma = new Turma();
50         return "Create";
51
52     }
53
54     public String prepareEdit() {
55         turma = (Turma) turmaDataModel.getRowData();
56         return "Edit";
57     }
58 }
```

```
59     public String prepareList() {
60         recriarModelo();
61         return "List";
62     }
63
64     //DataModel*****
65
66     public TurmaDataModel getTurmaDataModel() {
67         if (turmaDataModel == null) {
68             List<Turma> turmas = this.findAll();
69             turmaDataModel = new TurmaDataModel(turmas);
70         }
71         return turmaDataModel;
72     }
73
74     public void setTurmaDataModel(TurmaDataModel turmaDataModel) {
75         this.turmaDataModel = turmaDataModel;
76     }
77
78     public void recriarModelo() {
79         this.turmaDataModel = null;
80     }
81
82     //CRUD*****
83
84     public String create() {
85
86         try {
87             turmaFacade.save(turma);
88             JsفUtil.addSuccessMessage("Turma " + turma.getID() + "
89                 criada com sucesso!");
90             return prepareCreate();
91         } catch (Exception e) {
92             JsفUtil.addErrorMessage(e, "Ocorreu um erro de persistencia
93                 para salvar a turma");
94             return null;
95         }
96
97     }
98
99     public Turma find(Long id) {
100
101         return turmaFacade.find(id);
102     }
```

```
100     }
101
102     private List<Turma> findAll() {
103         return turmaFacade.findAll();
104
105     }
106
107     public void update() {
108         try {
109             turmaFacade.edit(turma);
110             JsفUtil.addSuccessMessage("Turma editada com sucesso!");
111             turma = null;
112
113         } catch (Exception e) {
114             JsفUtil.addErrorMessage(e, "Nao foi possivel editar,
115                                     ocorreu o seguinte erro: " + e.getMessage());
116         }
117     }
118
119     public void delete() {
120         turma = (Turma) turmaDataModel.getRowData();
121         try {
122             turmaFacade.remove(turma);
123             turma = null;
124             JsفUtil.addSuccessMessage("Turma Deletada");
125         } catch (Exception e) {
126             JsفUtil.addErrorMessage(e, "Ocorreu um erro de persistencia
127                                     para deletar a turma");
128         }
129
130         recriarModelo();
131     }
132
133     // *****
134
135     @FacesConverter(forClass = Turma.class)
136     public static class TurmaControllerConverter implements Converter {
137
138         @Override
139         public Object getAsObject(FacesContext facesContext,
140                                 UIComponent component, String value) {
141             if (value == null || value.length() == 0) {
```

```

140         return null;
141     }
142     TurmaController controller = (TurmaController) facesContext
143         .getApplication().getELResolver().
144         getValue(facesContext.getELContext(), null, "
145             turmaController");
146     return controller.getTurma(getKey(value));
147 }
148
149 java.lang.Long getKey(String value) {
150     java.lang.Long key;
151     key = Long.valueOf(value);
152     return key;
153 }
154
155 String getStringKey(java.lang.Long value) {
156     StringBuilder sb = new StringBuilder();
157     sb.append(value);
158     return sb.toString();
159 }
160
161 @Override
162 public String getAsString(FacesContext facesContext,
163     UIComponent component, Object object) {
164     if (object == null) {
165         return null;
166     }
167     if (object instanceof Turma) {
168         Turma t = (Turma) object;
169         return getStringKey(t.getID());
170     } else {
171         throw new IllegalArgumentException("object " + object +
172             " is of type " + object.getClass().getName() + ";
173             expected type: " + Turma.class.getName());
174     }
175 }
176 }
177 }
178 }

```

8.12 JsUtil

```

1 package controller;
2

```

```
3 import static controller.JsfUtil.addErrorMessage;
4 import java.util.List;
5 import javax.faces.application.FacesMessage;
6 import javax.faces.context.FacesContext;
7 import javax.faces.model.SelectItem;
8
9 public class JsfUtil {
10
11     public static SelectItem[] getSelectItems(List<?> entities, boolean
        selectOne) {
12         int size = selectOne ? entities.size() + 1 : entities.size();
13         SelectItem[] items = new SelectItem[size];
14         int i = 0;
15         if (selectOne) {
16             items[0] = new SelectItem("", "---");
17             i++;
18         }
19         for (Object x : entities) {
20             items[i++] = new SelectItem(x, x.toString());
21         }
22         return items;
23     }
24
25     public static void addErrorMessage(Exception ex, String defaultMsg)
        {
26         String msg = ex.getLocalizedMessage();
27         if (msg != null && msg.length() > 0) {
28             addErrorMessage(msg);
29         } else {
30             addErrorMessage(defaultMsg);
31         }
32     }
33
34     public static void addErrorMessages(List<String> messages) {
35         for (String message : messages) {
36             addErrorMessage(message);
37         }
38     }
39
40     public static void addErrorMessage(String msg) {
41         FacesMessage facesMsg = new FacesMessage(FacesMessage.
            SEVERITY_ERROR, msg, msg);
42         FacesContext.getCurrentInstance().addMessage(null, facesMsg);
```

```
43     }
44
45     public static void addSuccessMessage(String msg) {
46         FacesMessage facesMsg = new FacesMessage(FacesMessage.
47             SEVERITY_INFO, msg, msg);
48         FacesContext.getCurrentInstance().addMessage("successInfo",
49             facesMsg);
50     }
51 }
```

8.13 DocenteDataModel

```
1  package util;
2
3  import java.util.List;
4  import javax.faces.model.ListDataModel;
5  import model.Docente;
6  import org.primefaces.model.SelectableDataModel;
7
8  public class DocenteDataModel extends ListDataModel implements
9      SelectableDataModel<Docente> {
10
11     public DocenteDataModel() {
12     }
13
14     public DocenteDataModel(List<Docente> data) {
15         super(data);
16     }
17
18     @Override
19     public Docente getRowData(String rowKey) {
20         //In a real app, a more efficient way like a query by rowKey
21         //should be implemented to deal with huge data
22
23         List<Docente> docentes = (List<Docente>) getWrappedData();
24
25         for (Docente docente : docentes) {
26             if (docente.getID().equals(rowKey)) {
27                 return docente;
28             }
29         }
30
31         return null;
32     }
33 }
```



```
31
32     @Override
33     public Object getRowKey(Docente docente) {
34         return docente.getID();
35     }
36
37 }
```

8.14 DisciplinaDataModel

```
1  package util;
2
3  import java.util.List;
4  import javax.faces.model.ListDataModel;
5  import model.Disciplina;
6  import org.primefaces.model.SelectableDataModel;
7
8  public class DisciplinaDataModel extends ListDataModel implements
9      SelectableDataModel<Disciplina> {
10
11     public DisciplinaDataModel() {
12
13     }
14
15     public DisciplinaDataModel(List<Disciplina> data) {
16         super(data);
17     }
18
19     @Override
20     public Disciplina getRowData(String rowKey) {
21         //In a real app, a more efficient way like a query by rowKey
22         //should be implemented to deal with huge data
23
24         List<Disciplina> disciplinas = (List<Disciplina>)
25             getWrappedData();
26
27         for (Disciplina disciplina : disciplinas) {
28             if (disciplina.getID().equals(rowKey)) {
29                 return disciplina;
30             }
31         }
32
33         return null;
34     }
35 }
```

```
32     @Override
33     public Object getRowKey(Disciplina disciplina) {
34         return disciplina.getID();
35     }
36
37 }
```

8.15 TurmaDataModel

```
1  package util;
2
3  import java.util.List;
4  import javax.faces.model.ListDataModel;
5  import model.Turma;
6  import org.primefaces.model.SelectableDataModel;
7
8  public class TurmaDataModel extends ListDataModel implements
9      SelectableDataModel<Turma> {
10
11     public TurmaDataModel() {
12     }
13
14     public TurmaDataModel(List<Turma> data) {
15         super(data);
16     }
17
18     @Override
19     public Turma getRowData(String rowKey) {
20         //In a real app, a more efficient way like a query by rowKey
21         //should be implemented to deal with huge data
22
23         List<Turma> turmas = (List<Turma>) getWrappedData();
24
25         for (Turma turma : turmas) {
26             if (turma.getID().equals(rowKey)) {
27                 return turma;
28             }
29         }
30
31         return null;
32     }
33
34     @Override
35     public Object getRowKey(Turma turma) {
```

```
34         return turma.getID();
35     }
36
37 }
```

8.16 ViewDocente

8.16.1 Create

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html>
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        xmlns:h="http://java.sun.com/jsf/html"
5        xmlns:p="http://primefaces.org/ui">
6
7      <h:head>
8          <title>Criacao de Docentes</title>
9          <meta name="viewport" content="width=device-width"/>
10     </h:head>
11     <h:body>
12         <p:layout fullPage="true">
13             <p:layoutUnit position="center">
14                 <p:messages showDetail="false" autoUpdate="true"
15                     closable="true"/>
16                 <h:form>
17                     <p:panel header="Criar um novo registro" style="
18                         width: 480px">
19                         <h:panelGrid columns="2" id="grid">
20                             <h:outputLabel value="Nome : *"/>
21                             <p:inputText value="#{docenteController.
22                                 docente.nome}"/>
23                             <br/><br/>
24                             <p:commandLink value="Cadastrar Docente"
25                                 action="#{docenteController.create()}"
26                                 update="grid"/>
27                             <br/><br/><br/>
28                             <p:commandLink value="Ver todos os
29                                 docentes" action="#{docenteController.
30                                     prepareList()}" />
31                             <br/>
32                             <p:commandLink value="Voltar ao indice"
33                                 action="/index"/>
34                         </h:panelGrid>
35                     </p:panel>
```

```

28         </h:form>
29     </p:layoutUnit>
30 </p:layout>
31 </h:body>
32 </html>

```

8.16.2 Edit

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html>
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        xmlns:h="http://java.sun.com/jsf/html"
5        xmlns:p="http://primefaces.org/ui">
6      <h:head>
7          <title>Editar Docente</title>
8          <meta name="viewport" content="width=device-width"/>
9      </h:head>
10     <h:body>
11         <p:layout fullPage="true">
12
13             <p:layoutUnit position="center">
14                 <p:messages showDetail="false" autoUpdate="true"
15                     closable="true" />
16                 <h:form>
17                     <p:panel header="Editar um docente" style="width
18                         :450px;height:300px">
19                         <h:panelGrid columns="2" id="grid">
20                             <h:outputLabel value="Nome : *"></h:
21                                 outputLabel>
22                             <p:inputText value="#{docenteController.
23                                 docente.nome}"/>
24                             <br/><br/><br/>
25                             <p:commandLink value="Editar Docente"
26                                 action="#{docenteController.update()}"
27                                 update="grid"/>
28                             <br/><br/><br/><br/>
29                             <p:commandLink value="Ver todos os docentes
30                                 " action="#{docenteController.
31                                     prepareList()}"/>
32                             <br/>
33                             <p:commandLink value="Voltar ao indice"
34                                 action="/index"/>
35                         </h:panelGrid>
36                     </p:panel>

```

```

28         </h:form>
29     </p:layoutUnit>
30 </p:layout>
31 </h:body>
32 </html>

```

8.16.3 List

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html>
3  <html xmlns="http://www.w3.org/1999/xhtml"
4      xmlns:h="http://java.sun.com/jsf/html"
5      xmlns:f="http://java.sun.com/jsf/core"
6      xmlns:p="http://primefaces.org/ui">
7
8      <h:head>
9          <title>Docentes</title>
10         <meta name="viewport" content="width=device-width"/>
11     </h:head>
12
13     <h:body>
14         <p:layout fullPage="true">
15             <p:layoutUnit position="center">
16                 <p:messages showDetail="false" autoUpdate="true"
17                     closable="true" />
18                 <h:form id="form" >
19                     <p:panel style="width:610px;height:400px; text-
20                         align: center; border-color: transparent;
21                         background-color: transparent">
22                         <p:dataTable id="db" var="docente" value="#{
23                             docenteController.docenteDataModel}"
24                             selection="#{docenteController.
25                                 docente}"
26                             style="text-align: center"
27                             editable="true"
28                             selectionMode="single"
29                             paginator="true" rows="5"
30                             paginatorTemplate="{
31                                 CurrentPageReport} {
32                                 FirstPageLink} {
33                                 PreviousPageLink} {PageLinks} {
34                                 NextPageLink}
35                                 {LastPageLink} {
36                                     RowsPerPageDropdown}"

```

```
27         lazy="true">
28
29         <f:facet name="header">
30             Lista de Docentes
31         </f:facet>
32
33         <p:column headerText="id">
34             <f:facet name="header">
35                 <h:outputText value="id"/>
36             </f:facet>
37             <h:outputText value="#{docente.ID}"/>
38         </p:column>
39
40         <p:column headerText="nome">
41             <f:facet name="header">
42                 <h:outputText value="Nome"/>
43             </f:facet>
44             <h:outputText value="#{docente.nome}"/>
45         </p:column>
46
47         <p:column headerText="">
48             <f:facet name="header">
49                 <h:outputText value=""/>
50             </f:facet>
51
52             <p:commandLink action="#{
53                 docenteController.delete()}" value="
54                 Deletar" update="db"/>
55         </p:column>
56
57         <p:column headerText="">
58             <f:facet name="header">
59                 <h:outputText value=""/>
60             </f:facet>
61             <p:commandLink action="#{
62                 docenteController.prepareEdit()}"
63                 value="Editar" update="db"/>
64         </p:column>
65     </p:dataTable>
66     <br/><br/><br/>
67     <p:commandLink action="#{docenteController.
68         prepareCreate()}" value="Criar um Docente"/>
69     <br/><br/>
```

```

65         <p:commandLink action="/index" value="Voltar ao
           indice"/>
66     </p:panel>
67 </h:form>
68 </p:layoutUnit>
69 </p:layout>
70 </h:body>
71 </html>

```

8.17 ViewDisciplina

8.17.1 Create

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html>
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://java.sun.com/jsf/html"
5       xmlns:p="http://primefaces.org/ui">
6
7     <h:head>
8         <title>Criacao de Disciplinas</title>
9         <meta name="viewport" content="width=device-width"/>
10    </h:head>
11    <h:body>
12        <p:layout fullPage="true">
13            <p:layoutUnit position="center">
14                <p:messages showDetail="false" autoUpdate="true"
15                    closable="true"/>
16                <h:form>
17                    <p:panel header="Criar um novo registro" style="
18                        width: 480px">
19                        <h:panelGrid columns="2" id="grid">
20                            <h:outputLabel value="Nome : *"/>
21                            <p:inputText value="#{disciplinaController.
22                                disciplina.nome}"/>
23                            <br/>
24                            <p:commandLink value="Cadastrar Disciplina"
25                                action="#{disciplinaController.create()}
26                                    update="grid"/>
27                            <br/><br/><br/>
28                            <p:commandLink value="Ver todas as
29                                disciplinas" action="#{
30                                    disciplinaController.prepareList()}"/>
31                            <br/><br/>

```

```

25         <p:commandLink value="Voltar ao indice"
26             action="/index"/>
27     </h:panelGrid>
28     </p:panel>
29     </h:form>
30 </p:layoutUnit>
31 </p:layout>
32 </h:body>
33 </html>

```

8.17.2 Edit

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html>
3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:h="http://java.sun.com/jsf/html"
5     xmlns:p="http://primefaces.org/ui">
6     <h:head>
7         <title>Editar Disciplina</title>
8         <meta name="viewport" content="width=device-width"/>
9     </h:head>
10    <h:body>
11        <p:layout fullPage="true">
12
13            <p:layoutUnit position="center">
14                <p:messages showDetail="false" autoUpdate="true"
15                    closable="true" />
16                <h:form>
17                    <p:panel header="Editar uma disciplina" style="
18                        width:450px;height:300px">
19                        <h:panelGrid columns="2" id="grid">
20                            <h:outputLabel value="Nome : *"></h:
21                                outputLabel>
22                            <p:inputText value="#{disciplinaController.
23                                disciplina.nome}"/>
24                            <br/><br/><br/>
25                            <p:commandLink value="Editar Disciplina"
26                                action="#{disciplinaController.update()}"
27                                update="grid"/>
28                            <br/><br/><br/><br/><br/>
29                            <p:commandLink value="Ver todas as
30                                disciplinas" action="#{docenteController
31                                    .prepareList()}"/>
32                            <br/>

```



```

25         <p:commandLink value="Voltar ao indice"
26             action="/index"/>
27     </h:panelGrid>
28     </p:panel>
29     </h:form>
30     </p:layoutUnit>
31     </p:layout>
32 </h:body>
</html>

```

8.17.3 List

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html>
3  <html xmlns="http://www.w3.org/1999/xhtml"
4      xmlns:h="http://java.sun.com/jsf/html"
5      xmlns:f="http://java.sun.com/jsf/core"
6      xmlns:p="http://primefaces.org/ui">
7
8      <h:head>
9          <title>Disciplinas</title>
10         <meta name="viewport" content="width=device-width"/>
11     </h:head>
12
13     <h:body>
14         <p:layout fullPage="true">
15             <p:layoutUnit position="center">
16                 <p:messages showDetail="false" autoUpdate="true"
17                     closable="true" />
18                 <h:form id="form" >
19                     <p:panel style="width:610px;height:400px; text-
20                         align: center; border-color: transparent;
21                         background-color: transparent">
22                         <p:dataTable id="db" var="disciplina" value="
23                             #{disciplinaController.disciplinaDataModel}"
24                             selection="#{disciplinaController.
25                                 disciplina}"
26                             style="text-align: center"
27                             editable="true"
28                             selectionMode="single"
29                             paginator="true" rows="5"
30                             paginatorTemplate="{
31                                 CurrentPageReport} {
32                                 FirstPageLink} {

```

```

PreviousPageLink} {PageLinks} {
NextPageLink}
26 {LastPageLink} {
RowsPerPageDropdown}"
27 lazy="true">
28 <f:facet name="header">
29     Lista de Disciplinas
30 </f:facet>
31 <p:column headerText="id">
32     <f:facet name="header">
33         <h:outputText value="id"/>
34     </f:facet>
35     <h:outputText value="#{disciplina.ID}"/>
36 </p:column>
37 <p:column headerText="nome">
38     <f:facet name="header">
39         <h:outputText value="Nome"/>
40     </f:facet>
41     <h:outputText value="#{disciplina.nome}"/>
42 </p:column>
43 <p:column headerText="">
44     <f:facet name="header">
45         <h:outputText value=""/>
46     </f:facet>
47     <p:commandLink action="#{disciplinaController.delete()}"
48         value="Deletar" update="db"/>
49 </p:column>
50 <p:column headerText="">
51     <f:facet name="header">
52         <h:outputText value=""/>
53     </f:facet>
54     <p:commandLink action="#{disciplinaController.prepareEdit()}"
55         value="Editar" update="db"/>
56 </p:column>
57 </p:dataTable>
<br/><br/> <br/>
<p:commandLink action="#{disciplinaController.prepareCreate()}" value="Criar uma
Disciplina"/>

```

```

58         <br/><br/>
59         <p:commandLink action="/index" value="Voltar ao
           indice"/>
60     </p:panel>
61 </h:form>
62 </p:layoutUnit>
63 </p:layout>
64 </h:body>
65 </html>

```

8.18 ViewTurma

8.18.1 Create

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html>
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://java.sun.com/jsf/html"
5       xmlns:p="http://primefaces.org/ui"
6       xmlns:f="http://java.sun.com/jsf/core">
7
8     <h:head>
9         <title>Criacao de Turmas</title>
10        <meta name="viewport" content="width=device-width"/>
11    </h:head>
12
13    <h:body>
14        <p:layout fullPage="true">
15            <p:layoutUnit position="center">
16                <p:messages showDetail="false" autoUpdate="true"
17                    closable="true"/>
18                <h:form>
19                    <p:panel header="Criar uma nova turma" style="width
20                        :400px;height:400px; text-align: center">
21                        <h:panelGrid columns="2" id="grid">
22                            <h:outputLabel value="Docente : *"></h:
23                                outputLabel>
24                            <h:selectOneMenu id="docente" value="#{
25                                turmaController.turma.docente}" title="
26                                    docente">
27                                <f:selectItems value="#{
28                                    docenteController.
29                                        itemsAvaivableSelectOne}"/>
29                            </h:selectOneMenu>

```

```

24         <h:outputLabel value="Disciplina : *"></h:
           outputLabel>
25         <h:selectOneMenu id="disciplina" value="#{
           turmaController.turma.disciplina}" title
           ="disciplina">
26             <f:selectItems value="#{
                 disciplinaController.
                 itemsAvaivableSelectOne}"/>
27         </h:selectOneMenu>
28     </h:panelGrid>
29     <br/><br/><br/>
30     <p:commandLink value="Cadastrar Turma" action="
           #{turmaController.create()}" />
31     <br/><br/><br/>
32     <p:commandLink value="Ver todas as turmas"
           action="#{turmaController.prepareList()}" />
33     <br/><br/>
34     <p:commandLink value="Voltar ao indice" action=
           "/index" />
35     </p:panel>
36     </h:form>
37     </p:layoutUnit>
38 </p:layout>
39 </h:body>
40 </html>

```

8.18.2 Edit

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html>
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://java.sun.com/jsf/html"
5       xmlns:p="http://primefaces.org/ui"
6       xmlns:f="http://java.sun.com/jsf/core">
7
8     <h:head>
9         <title>Editar Turmas</title>
10        <meta name="viewport" content="width=device-width"/>
11    </h:head>
12
13    <h:body>
14        <p:layout fullPage="true">
15            <p:layoutUnit position="center">

```

```

16         <p:messages showDetail="false" autoUpdate="true"
17             closable="true"/>
18     <h:form>
19         <p:panel header="Editar uma turma" style="width:370
20             px;height:200px;text-align: center">
21             <h:panelGrid columns="2" id="grid">
22                 <h:outputLabel value="Docente : *"></h:
23                     outputLabel>
24                 <h:selectOneMenu id="docente" value="#{
25                     turmaController.turma.docente}" title="
26                     docente">
27                     <f:selectItems value="#{
28                         docenteController.
29                         itemsAvaivableSelectOne}"/>
30                 </h:selectOneMenu>
31                 <h:outputLabel value="Disciplina : *"></h:
32                     outputLabel>
33                 <h:selectOneMenu id="disciplina" value="#{
34                     turmaController.turma.disciplina}" title
35                     ="disciplina">
36                     <f:selectItems value="#{
37                         disciplinaController.
38                         itemsAvaivableSelectOne}"/>
39                 </h:selectOneMenu>
40             </h:panelGrid>
41             <br/><br/>
42             <p:commandLink value="Editar Turma" action="#{
43                 turmaController.update()}"/>
44             <br/><br/><br/><br/>
45             <p:commandLink value="Ver todas as turmas"
46                 action="#{turmaController.prepareList()}"/>
47             <br/><br/>
48             <p:commandLink value="Voltar ao indice" action=
49                 "/index"/>
50         </p:panel></h:form>
51     </p:layoutUnit>
52 </p:layout>
53 </h:body>
54 </html>

```

8.18.3 List

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html>

```

```

3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:h="http://java.sun.com/jsf/html"
5     xmlns:f="http://java.sun.com/jsf/core"
6     xmlns:p="http://primefaces.org/ui">
7
8 <h:head>
9     <title>Turmas</title>
10    <meta name="viewport" content="width=device-width"/>
11 </h:head>
12
13 <h:body>
14    <h:panelGroup id="messagePanel" layout="block">
15        <h:messages errorStyle="color: red" infoStyle="color: green"
16            layout="table"/>
17    </h:panelGroup>
18    <h:form id="form" >
19        <p:panel style="width:700px;height:400px; text-align:
20            center; border-color: transparent">
21            <p:dataTable id="db" var="turma" value="#{
22                turmaController.turmaDataModel}"
23                selection="#{turmaController.turma}"
24                style="text-align: center"
25                editable="true"
26                selectionMode="single"
27                paginator="true" rows="5"
28                paginatorTemplate="{CurrentPageReport} {
29                    FirstPageLink} {PreviousPageLink}
30                    {PageLinks} {NextPageLink} {LastPageLink}
31                    {RowsPerPageDropdown}">
32                <f:facet name="header" >
33                    Lista de Turmas
34                </f:facet>
35                <p:column headerText="id">
36                    <f:facet name="header">
37                        <h:outputText value="id"/>
38                    </f:facet>
39                    <h:outputText value="#{turma.ID}"/>
40                </p:column>
41                <p:column headerText="docente">
42                    <f:facet name="header">
43                        <h:outputText value="Docente"/>
44                    </f:facet>
45                    <h:outputText value="#{turma.docente.nome}"/>

```

```
41         </p:column>
42         <p:column headerText="disciplina">
43             <f:facet name="header">
44                 <h:outputText value="Disciplina"/>
45             </f:facet>
46             <h:outputText value="#{turma.disciplina.nome}"/>
47         </p:column>
48         <p:column headerText="">
49             <f:facet name="header">
50                 <h:outputText value=""/>
51             </f:facet>
52             <p:commandLink value="Deletar" action="#{turmaController.delete()}">
53         </p:column>
54         <p:column headerText="">
55             <f:facet name="header">
56                 <h:outputText value=""/>
57             </f:facet>
58             <p:commandLink value="Editar" action="#{turmaController.prepareEdit()}">
59         </p:column>
60     </p:dataTable>
61     <br/><br/><br/><br/>
62     <p:commandLink value="Criar uma turma" action="#{turmaController.prepareCreate()}">
63 <br/><br/>
64     <p:commandLink value="Voltar ao indice" action="/index"
65     />
66 </p:panel>
67 </h:form>
68 </h:body>
69 </html>
```