

**Tema**

MARCO DE PÁGINA EN NETBEANS

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

Fecha

27/07/2023

MARCO DE PÁGINA EN NETBEANS.

[1. INTRODUCCIÓN 4](#_Toc129356678)

[2. OBJETIVO 4](#_Toc129356679)

[2.1 OBJETIVO GENERAL 4](#_Toc129356680)

[2.2 OBJETIVOS ESPECÍFICOS 4](#_Toc129356681)

[3. MARCO TEÓRICO 4](#_Toc129356682)

[3.1 INTEGRACION A BASES DE DATOS DESDE JAVA 4](#_Toc129356683)

[3.2 MYSQL CONNECTOR Y JDBC 5](#_Toc129356684)

[3.3 DATA ACCESS OBJECT. 7](#_Toc129356685)

[3.4 INTERFACES JAVA. 7](#_Toc129356686)

[3.5 MYSQL. 8](#_Toc129356687)

[3.6 MYSQL Y EL GESTOR DE XAMPP. 9](#_Toc129356688)

[3.7 MARCOS DE PÁGINA EN APLICACIONES WEB. 10](#_Toc129356689)

[3.8 MARCOS DE PAGINA CON PRIMEFACES. 11](#_Toc129356690)

[3.9 IMPLEMENTACIÓN DE MARCOS DE PÁGINA CON PRIMEFACES 12](#_Toc129356691)

[3.10 MÉTODOS. 14](#_Toc129356692)

[3.11 TÉRMINOS 15](#_Toc129356693)

[4. DESARROLLO 17](#_Toc129356694)

[4.1 CREACIÓN DE VISTAS DEL PROYECTO 17](#_Toc129356695)

[4.1.1 MODIFICACIÓN DE CRUD PARA PERSONAL. 17](#_Toc129356696)

[4.2 CREACIÓN DE MODELOS DEL PROYECTO 35](#_Toc129356697)

[4.2.1 CREACIÓN DE MÉTODOS Y CONSULTAS DEL MODELO 35](#_Toc129356698)

[4.3 CREACIÓN DE CONTROLADORES DEL PROYECTO 58](#_Toc129356699)

[4.3.1 CODIFICACIÓN DEL ARCHIVO PEEMPLECONTROLLER. 58](#_Toc129356700)

[5. EJECUCIÓN DEL PROYECTO 85](#_Toc129356701)

[6. CONCLUSIONES 87](#_Toc129356702)

[7. RECOMENDACIONES 88](#_Toc129356703)

[8. BIBLIOGRAFÍA 88](#_Toc129356711)

**ÍNDICE DE IMÁGENES**

[Figura 1. Diagrama básico de sockets. 7](#_Toc129356656)

[Figura 2. Código sencillo sobre el funcionamiento de timers. 9](#_Toc129356657)

[Figura 3. Código sencillo sobre el funcionamiento de timers. 10](#_Toc129356658)

[Figura 4. Código sencillo para implementar <p:layout> y <p:layoutUnit>. 12](#_Toc129356659)

[Figura 5. Código sencillo para implementar marcos de página con <p:tabView>. 14](#_Toc129356660)

[Figura 6. Ultimo avance de CRUD realizado en personal. 18](#_Toc129356661)

[Figura 7. Ubicación del CRUD de Personal. 19](#_Toc129356662)

[*Figura 8. Nueva codificación para Create.xhtml.* 19](#_Toc129356663)

[*Figura 9. Nueva codificación para Edit.xhtml* 20](#_Toc129356664)

[*Figura 10. Nueva codificación para View.xhtml* 20](#_Toc129356665)

[Figura 11. Selección del archivo para modificar. 36](#_Toc129356666)

[Figura 12. Codificación del archivo PeempEmple.java. 36](#_Toc129356667)

[Figura 13. Selección del archivo para modificar. 51](#_Toc129356668)

[Figura 14. Codificación del archivo PeempEmpleFacade.java. 51](#_Toc129356669)

[Figura 15. Selección del archivo Java Class. 59](#_Toc129356670)

[Figura 16. Codificación del archivo PeempEmpleController,java 59](#_Toc129356671)

[Figura 17. Botón para ejecutar el proyecto. 86](#_Toc129356672)

[Figura 18. Ejecución del proyecto. 86](#_Toc129356673)

[Figura 19. Crud de personal. 87](#_Toc129356674)

[Figura 20. Marco de pantalla de información de usuario. 87](#_Toc129356675)

[Figura 21. Marco de página de edición de usuario. 88](#_Toc129356676)

[Figura 22. Marco de página de creación de usuario. 88](#_Toc129356677)

**ÍNDICE DE TABLAS**

[Tabla 1. Métodos generalmente utilizados en conexión de java y bases de datos. 15](#_Toc129356648)

[Tabla 2. Términos 16](#_Toc129356649)

[Tabla 3. Nueva codificación del archivo peempEmple/Create.xhtml 21](file:///D:\7mo%20Semestre\Distribuidas\Tercer%20Parcial\Proyecto%20Tercer%20Parcial%20Monster\05.%20DOCUMENTACION\Marco%20Página%20NetBeans_GRUPO6.docx#_Toc129356650)

[Tabla 4. Nueva codificación del archivo peempEmple/ View.xhtml 27](file:///D:\7mo%20Semestre\Distribuidas\Tercer%20Parcial\Proyecto%20Tercer%20Parcial%20Monster\05.%20DOCUMENTACION\Marco%20Página%20NetBeans_GRUPO6.docx#_Toc129356651)

[Tabla 6. Nueva codificación del archivo peempEmple/ Edit.xhtml 30](file:///D:\7mo%20Semestre\Distribuidas\Tercer%20Parcial\Proyecto%20Tercer%20Parcial%20Monster\05.%20DOCUMENTACION\Marco%20Página%20NetBeans_GRUPO6.docx#_Toc129356652)

[Tabla 7. Codificación del archivo PeempEmple.java 37](#_Toc129356653)

[Tabla 8. Codificación del archivo PeempEmpleFacade.java 52](#_Toc129356654)

[Tabla 9. Codificación del archivo controlador PeempEmpleController.java 60](file:///D:\7mo%20Semestre\Distribuidas\Tercer%20Parcial\Proyecto%20Tercer%20Parcial%20Monster\05.%20DOCUMENTACION\Marco%20Página%20NetBeans_GRUPO6.docx#_Toc129356655)

# INTRODUCCIÓN

En este trabajo de investigación y desarrollo, se presenta la creación de una aplicación web desarrollada en Java EE para el sistema de inicio de sesión de la empresa Screams Inc. La aplicación cuenta con una conexión segura a una base de datos MySQL para el almacenamiento y recuperación de información de los usuarios. El trabajo se estructura en tres partes: en primer lugar, se presenta un marco teórico que aborda los componentes y herramientas utilizadas en el desarrollo de la aplicación. En segundo lugar, se proporciona una guía detallada sobre los pasos seguidos en el proceso de desarrollo de la aplicación. Por último, se presentan las conclusiones obtenidas a partir del trabajo realizado.

El objetivo principal de esta aplicación es proporcionar un marco de página para mostrar la información del personal de la aplicación para que sea fácil y sencillo para los administradores de la empresa encontrar la información y verla de forma más organizada. A través de este trabajo se pretende mejorar la eficiencia y organización en la gestión del personal en Screams Inc.

# OBJETIVO

### OBJETIVO GENERAL

Documentar los detalles técnicos de la implementación del marco de página incluyendo detalles sobre el código utilizado para integrar el marco de página en los formularios de la aplicación web, así como cualquier otra información técnica relevante.

### OBJETIVOS ESPECÍFICOS

* Proporcionar una guía de referencia para futuras actualizaciones y mantenimiento facilitando la tarea de actualizar o modificar la implementación en el futuro. Esto es especialmente importante si hay múltiples desarrolladores trabajando en la aplicación.
* Ayudar a otros desarrolladores a entender la implementación del marco de página siendo una guía útil para comprender cómo se ha implementado el marco de página.
* Facilitar la identificación y resolución de problemas: si hay algún problema o error relacionado con la implementación del marco de página, el documento de implementación puede ser útil para identificar el problema y proporcionar información relevante para su resolución.
* Servir como documentación para los usuarios finales: si los usuarios finales de la aplicación necesitan comprender cómo funciona el marco de página, el documento de implementación puede servir como una guía útil para ellos y proporcionar una guía detallada sobre el proceso de desarrollo de la aplicación.

# MARCO TEÓRICO

## 3.1 INTEGRACION A BASES DE DATOS DESDE JAVA

Cuando se habla de la conexión de una aplicación con una base de datos no [1] existe una única forma de realizar este proceso, ya que existen varias formas de conectar una aplicación web desarrollada en Java con una base de datos, algunos ejemplos son:

* JDBC (Java Database Connectivity): Es una interfaz de programación de aplicaciones (API) que permite a los desarrolladores de Java interactuar con cualquier base de datos soportada mediante un controlador JDBC. Con JDBC, una aplicación puede enviar comandos SQL a una base de datos y procesar los resultados.
* JPA (Java Persistence API): Es una interfaz de programación de aplicaciones que proporciona una manera de acceder a una base de datos mediante objetos Java. JPA utiliza un modelo de objeto-relacional (ORM) para mapear las tablas de una base de datos con las clases Java correspondientes.
* Hibernate: Es una implementación popular de JPA. Hibernate proporciona una interfaz para interactuar con una base de datos mediante objetos Java y mapea automáticamente los objetos a las tablas de la base de datos.
* Spring Data: Es un marco de trabajo que proporciona una interfaz para interactuar con una base de datos mediante objetos Java. Spring Data se basa en JPA y Hibernate, y proporciona una interfaz de programación de aplicaciones consistente para diferentes tipos de bases de datos.

En general, todas estas herramientas proveen una capa de abstracción sobre JDBC y permiten a los desarrolladores interactuar con la base de datos mediante objetos Java, lo que facilita la escritura de código y reduce los errores. Es importante que se escoja la herramienta adecuada para el proyecto y se tenga en cuenta las necesidades y requerimientos del proyecto.

## MYSQL CONNECTOR Y JDBC

MySQL Connector/J es un [2] driver JDBC (Java Database Connectivity) desarrollado por MySQL AB (ahora propiedad de Oracle) para conectar programas escritos en Java con una base de datos MySQL. JDBC es una API de Java que proporciona una interfaz estándar para conectarse a una base de datos desde un programa Java.

MySQL Connector/J es uno de los drivers JDBC disponibles para MySQL, y se utiliza para establecer una conexión a la base de datos, enviar consultas y recibir resultados. Es compatible con todas las versiones de MySQL desde la versión 3.23 y también es compatible con las últimas versiones de Java.

Para utilizar MySQL Connector/J en un programa Java, primero se debe incluir la biblioteca en el proyecto. Luego se puede utilizar la clase DriverManager para establecer una conexión a la base de datos. Una vez que se ha establecido la conexión, se puede utilizar la interfaz Connection para enviar consultas y recibir resultados.

Aquí hay un ejemplo de cómo utilizar MySQL Connector/J para conectarse a una base de datos:



Figura 1. Diagrama básico de sockets.

En este ejemplo, se importa la biblioteca de JDBC de Java y se crea una clase llamada MySQLExample. En el método main, se registra el driver JDBC de MySQL Connector/J con Class.forName(JDBC\_DRIVER). Luego se establece una conexión a la base de datos utilizando la clase DriverManager y se almacena en una variable conn. El DriverManager necesita los parámetros de conexión como el url, user y password para establecer la conexión.

Una vez que se ha establecido la conexión, se cierra la conexión y la declaración con conn.close() y stmt.close() respectivamente.

En caso de que ocurra algún error en la conexión o en el proceso, se maneja con los bloques catch que imprimen el error en la consola.

Es importante notar que en este ejemplo no se realiza ninguna operación en la base de datos, solo se establece la conexión y se cierra.

## DATA ACCESS OBJECT.

DAO (Data Access Object) es un [3] patrón de diseño en Java que se utiliza para separar la lógica de acceso a datos de la lógica de negocio de una aplicación. El objetivo de este patrón es proporcionar una interfaz para acceder a los datos almacenados en una base de datos de manera independiente de la implementación de la base de datos.

Un DAO es una clase Java que se encarga de realizar operaciones CRUD (create, read, update, delete) en una base de datos. Esta clase proporciona métodos para insertar, actualizar, eliminar y recuperar datos de la base de datos. Un DAO suele tener una conexión a la base de datos y utiliza JDBC para enviar y recibir datos.

El DAO ayuda a aislar la lógica de negocio de la aplicación de los detalles de acceso a datos y permite que la lógica de negocio se pueda reutilizar fácilmente con diferentes bases de datos. Por ejemplo, si se decide cambiar de una base de datos MySQL a una base de datos Oracle, solo se necesitaría cambiar la implementación del DAO para adaptarse a la nueva base de datos, en lugar de tener que cambiar toda la aplicación.

En resumen, el DAO es una clase que se encarga de abstraer los detalles de acceso a datos y se utiliza para separar la lógica de negocio de la aplicación de la lógica de acceso a datos, permitiendo una mayor flexibilidad y escalabilidad en el desarrollo de aplicaciones.

## INTERFACES JAVA.

Las interfaces en Java se [4] utilizan para definir un contrato de métodos que una clase debe cumplir. Por ejemplo, si varias clases deben tener un método "start" que se ejecuta de manera diferente en cada clase, se puede crear una interfaz "Startable" con un método "start" y hacer que cada clase implemente esa interfaz. De esta manera, se garantiza que todas las clases tengan un método "start" y se pueden utilizar de manera polimórfica.

Una interfaz se define utilizando la palabra clave "interface" seguida del nombre de la interfaz y los métodos que esta incluye. Los métodos en una interfaz solo tienen la declaración y no tienen una implementación.

Para utilizar una interfaz en una clase, se utiliza la palabra clave "implements" seguida del nombre de la interfaz. Luego, la clase debe proporcionar una implementación para cada uno de los métodos de la interfaz.

Un ejemplo de su uso en el mundo real es en la programación de aplicaciones móviles. Por ejemplo, se podría crear una interfaz "Connectable" que incluya métodos para conectar y desconectar un dispositivo. Luego, se podrían crear clases específicas para cada tipo de dispositivo (como Bluetooth, WiFi, etc.) que implementen esa interfaz.

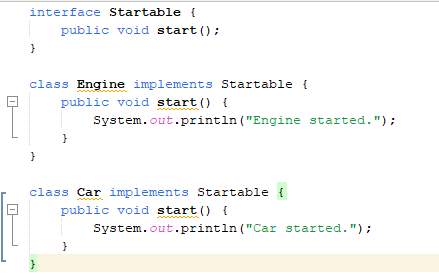


Figura 2. Código sencillo sobre el funcionamiento de timers.

En este ejemplo, se define una interfaz "Startable" con un método "start". Luego, se crean las clases "Engine" y "Car" que implementan esa interfaz y proporcionan una implementación específica para el método "start".

## MYSQL.

MySQL es [5] un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto. Fue desarrollado por la empresa sueca MySQL AB, y actualmente es propiedad de Oracle. MySQL es uno de los sistemas de gestión de bases de datos más populares y ampliamente utilizados en el mundo, especialmente en aplicaciones web y sistemas empresariales.

Este gestor almacena datos en tablas, que a su vez están compuestas por columnas y filas. Cada tabla tiene una estructura definida mediante un esquema, que especifica los nombres de las columnas, el tipo de datos que almacenan, y otras características. MySQL también permite la creación de relaciones entre tablas mediante claves foráneas, lo que permite la creación de bases de datos complejas y flexibles.

Cabe señalar que MySQL utiliza un lenguaje de consulta estándar llamado SQL (Structured Query Language) para interactuar con las bases de datos. Con SQL, se pueden realizar operaciones como insertar, actualizar, eliminar y consultar datos en las tablas.

MySQL también ofrece una serie de herramientas y características avanzadas para garantizar la escalabilidad, disponibilidad y rendimiento. Entre ellas se encuentran:

* Replicación: permite la replicación de datos entre varios servidores, lo que permite una mejor escalabilidad y disponibilidad.
* Particionamiento: permite dividir una tabla en varias particiones para mejorar el rendimiento en consultas de gran tamaño.
* Transacciones: soporta transacciones para garantizar la integridad de los datos.
* Seguridad: ofrece varias opciones de seguridad para proteger los datos, como autenticación de usuarios y encriptación de datos.

MySQL es compatible con una amplia variedad de sistemas operativos y lenguajes de programación, incluyendo Windows, Linux, MacOS, y lenguajes como Java, PHP, Python y C#. Además, existe una gran cantidad de herramientas y librerías disponibles para facilitar el desarrollo y la administración de bases de datos MySQL.

## MYSQL Y EL GESTOR DE XAMPP.

XAMPP es [6] una distribución de software gratuito y de código abierto que incluye un conjunto de aplicaciones de servidor web, como Apache, PHP y Perl, así como una base de datos MySQL. Se utiliza principalmente para desarrollar y probar aplicaciones web en un entorno local antes de desplegarlas en un servidor en producción.

El gestor de base de datos de MySQL en XAMPP es una versión de MySQL que está preconfigurada para trabajar con las aplicaciones de servidor web incluidas en XAMPP. Es una versión completa de MySQL con todas las características y funciones incluidas en la versión estable oficial de MySQL.

Al instalar XAMPP, se instala automáticamente el gestor de base de datos de MySQL, junto con otras aplicaciones de servidor web, como Apache y PHP. Una vez instalado, se puede acceder al gestor de base de datos de MySQL mediante una interfaz gráfica de usuario llamada "phpMyAdmin" que se incluye con XAMPP. Esta interfaz permite crear, modificar y gestionar bases de datos de manera sencilla.

El gestor de base de datos de MySQL en XAMPP también se puede acceder mediante línea de comandos utilizando el programa "mysql" que se incluye con XAMPP. Esto permite realizar tareas de administración de bases de datos de manera avanzada y automatizada.

En resumen, el gestor de base de datos de MySQL en XAMPP es una versión completa y preconfigurada de MySQL que se incluye en la distribución de software XAMPP, que permite desarrollar y probar aplicaciones web de manera sencilla en un entorno local.

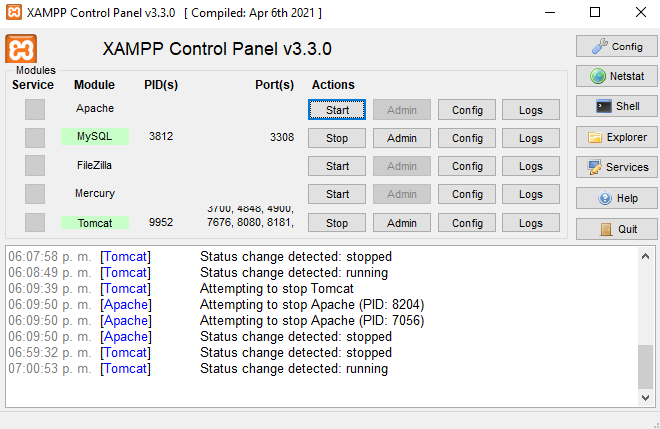


Figura 3. Código sencillo sobre el funcionamiento de timers.

## MARCOS DE PÁGINA EN APLICACIONES WEB.

Los marcos de página [7] (también conocidos como frames o iframes) en una aplicación web son una técnica que permite dividir una página web en varias secciones independientes, cada una de las cuales muestra contenido web diferente. Cada sección o marco está definida por un elemento HTML llamado "frame" o "iframe", que se utiliza para indicar al navegador web que muestre el contenido de otra página web dentro de la página actual.

Los marcos de página se crean mediante el uso de la etiqueta <frame> o <iframe>, que se incluyen en el código HTML de la página web. La etiqueta <frame> se utiliza para dividir una página web en varias secciones, mientras que la etiqueta <iframe> se utiliza para mostrar una página web dentro de otra página web. Ambas etiquetas aceptan varios atributos, como "src" para especificar la URL de la página web que se mostrará, "name" para identificar el marco, "height" y "width" para especificar las dimensiones del marco, entre otros. Las principales características de los marcos de página (frames o iframes) en una aplicación web son las siguientes:

* División de la pantalla: Los marcos de página permiten dividir la pantalla en varias secciones independientes, cada una de las cuales muestra contenido web diferente.
* Incorporación de contenido web: Los marcos de página se utilizan para incorporar contenido de una página web dentro de otra página web, lo que puede ser útil para mostrar contenido de manera constante en una parte del sitio web.
* Flexibilidad: Los marcos de página permiten que cada sección tenga su propio conjunto de características, como la capacidad de desplazamiento y de cambio de tamaño, lo que puede mejorar la experiencia del usuario.
* Control del contenido: Los marcos de página permiten a los desarrolladores tener mayor control sobre el contenido que se muestra en la página web, lo que puede ser útil en situaciones en las que se desea mantener una cierta consistencia en el diseño y el contenido.
* Dificultades de navegación: Los marcos de página pueden dificultar la navegación del usuario, ya que pueden hacer que sea difícil volver a la página anterior o identificar la ubicación exacta del contenido en la página.
* Problemas de SEO: Los marcos de página pueden dificultar el SEO (optimización para motores de búsqueda), ya que los motores de búsqueda pueden tener dificultades para indexar y clasificar el contenido en una página que utiliza marcos de página.

El uso de marcos de página puede tener algunos beneficios, como la capacidad de mostrar contenido de otras páginas web sin necesidad de cambiar de página, lo que puede mejorar la usabilidad y la experiencia del usuario. Sin embargo, también hay algunos inconvenientes que deben tenerse en cuenta. Uno de los mayores problemas es que los marcos de página pueden dificultar la navegación y la experiencia del usuario, ya que pueden ser confusos y hacer que sea difícil volver a la página anterior. Además, los motores de búsqueda pueden tener dificultades para indexar y clasificar el contenido en una página que utiliza marcos de página [7].

En la actualidad, los marcos de página han sido reemplazados en gran medida por otras técnicas de diseño web, como el uso de elementos HTML5 y CSS3 para crear diseños de página web más modernos y efectivos. En general, se recomienda evitar el uso excesivo de marcos de página en una aplicación web y considerar cuidadosamente los beneficios y desventajas antes de decidir utilizarlos.

## MARCOS DE PAGINA CON PRIMEFACES.

Los marcos de página [8] (frames o iframes) en una aplicación web con PrimeFaces en Java EE se pueden utilizar mediante el uso de los componentes <p:layout> y <p:layoutUnit> de PrimeFaces.

La etiqueta <p:layout> representa el contenedor principal para los marcos de página, y se utiliza para definir el diseño general del marco. Puedes especificar el número de filas y columnas que tendrá el diseño, así como la altura y anchura de cada celda. Cada celda de <p:layout> se puede llenar con un <p:layoutUnit> [9].

La etiqueta <p:layoutUnit> representa una celda de contenido dentro de <p:layout>. Se utiliza para definir qué contenido se mostrará dentro de cada celda, así como su posición y tamaño en el diseño. Puedes utilizar cualquier componente de PrimeFaces o HTML estándar dentro de un <p:layoutUnit> para mostrar el contenido deseado.

Además de <p:layout> y <p:layoutUnit>, PrimeFaces también ofrece otros componentes útiles para la implementación de marcos de página en aplicaciones web, como <p:tabView> y <p:accordionPanel>. Estos componentes te permiten crear marcos de página con pestañas o paneles que se expanden y se contraen al hacer clic en ellos. La etiqueta <p:tabView> de PrimeFaces en Java EE se puede utilizar para crear pestañas en una página web y mostrar contenido web independiente en cada pestaña, lo que puede ser considerado como una alternativa a los marcos de página.

Un ejemplo de cómo utilizar estas etiquetas y crear un marco de página es el siguiente [10].

Texto

Descripción generada automáticamente

Figura 4. Código sencillo para implementar <p:layout> y <p:layoutUnit>.

En este ejemplo, se utiliza <p:layout> como contenedor principal para los marcos y el contenido principal de la página. Se definen tres <p:layoutUnit> para los marcos superior, izquierdo y central.

El atributo "fullPage" de <p:layout> se establece en "true" para que la página web ocupe toda la ventana del navegador.

Los atributos "position" y "size" de <p:layoutUnit> definen la posición y el tamaño de cada marco, respectivamente.

Los atributos "resizable", "closable" y "collapsible" de <p:layoutUnit> permiten que el usuario pueda ajustar el tamaño de los marcos, cerrarlos o colapsarlos, según sea necesario.

La etiqueta <p:tabView> también permite crear una interfaz de usuario con pestañas y mostrar diferentes páginas web en cada pestaña. Para utilizar esta etiqueta, se debe agregar la dependencia de PrimeFaces en el archivo pom.xml del proyecto y se debe agregar el espacio de nombres (namespace) de PrimeFaces en la página web que utilizará la etiqueta [8].

Una vez que se ha agregado la dependencia y el namespace, se puede utilizar la etiqueta <p:tabView> en la página web de PrimeFaces. El componente acepta varios atributos, como "id" para identificar la etiqueta, "style" y "styleClass" para aplicar estilos CSS personalizados, "dynamic" para habilitar o deshabilitar la carga dinámica de pestañas, entre otros.

Cada pestaña se define mediante la etiqueta <p:tab> y se puede especificar la URL de la página web que se mostrará en esa pestaña utilizando el atributo "url". Además, se pueden agregar otros componentes de PrimeFaces o HTML dentro de cada pestaña para personalizar la presentación del contenido.

En comparación con los marcos de página, las pestañas creadas con <p:tabView> tienen algunas ventajas. Por ejemplo, permiten una navegación más fácil e intuitiva para el usuario, ya que no se separa el contenido en diferentes áreas de la pantalla. Además, las pestañas son más compatibles con los navegadores web modernos y pueden ser más amigables con los motores de búsqueda [11].

Algunos de los beneficios que trae utilizar <p:tabView> en una aplicación web como marco de página son los siguientes:

* Permite una navegación más fácil e intuitiva para el usuario
* Puede ser más compatible con los navegadores web modernos
* Es más amigable con los motores de búsqueda
* Permite mostrar contenido web independiente en cada pestaña
* Permite personalizar la presentación del contenido dentro de cada pestaña
* Es más adecuado para una interfaz de usuario moderna y atractiva
* Evita los problemas potenciales de los marcos de página, como la dificultad para volver a la página anterior o identificar la ubicación exacta del contenido en la página

Se puede decir entonces que, la etiqueta <p:tabView> de PrimeFaces en Java EE es una alternativa a los marcos de página que permite crear una interfaz de usuario con pestañas para mostrar contenido web independiente en cada pestaña. Es recomendable utilizar esta técnica en lugar de marcos de página para lograr una presentación más moderna y efectiva en tu aplicación web.

## IMPLEMENTACIÓN DE MARCOS DE PÁGINA CON PRIMEFACES

Un ejemplo de implementación de marcos de página con la etiqueta <p:tabView> puede ser el siguiente para poder ilustrar el funcionamiento del mismo en un entorno real [11].

Escala de tiempo

Descripción generada automáticamente

Figura 5. Código sencillo para implementar marcos de página con <p:tabView>.

En este ejemplo el código define una estructura de pestañas que contiene tres pestañas con diferentes contenidos en cada una de ellas. Cada pestaña es un objeto p:tab, y su título se define en el atributo title.

El contenido de cada pestaña se define dentro de un objeto p:panel, que puede contener otros componentes, como h1 para encabezados, p:commandButton para botones con acciones asociadas y p:dataTable para tablas con datos.

El código también hace referencia a un objeto de Java llamado myBean, el cual tiene métodos como doSomething y doSomethingElse que se llaman cuando se hacen clic en los botones de comando. myBean también contiene una lista llamada listaDeElementos y una variable llamada valorDeEntrada, que se usan en el código para mostrar datos y recibir entrada del usuario, respectivamente.

<p:tabView>: Este es el elemento principal que se utiliza para crear el componente de pestañas. Todos los elementos <p:tab> deben estar contenidos dentro de este elemento. Además, el componente <p:tabView> ofrece una variedad de atributos que se pueden utilizar para personalizar su aspecto y comportamiento.

<p:tab>: Este elemento se utiliza para crear una sola pestaña en el componente de pestañas. Cada elemento <p:tab> debe tener un atributo title que define el título de la pestaña. El contenido de la pestaña se especifica utilizando otros elementos, como el <p:panel> que se muestra a continuación.

<p:panel>: Este elemento se utiliza para crear un panel que contiene el contenido de una pestaña. En este caso, se utilizan dos elementos HTML (<h1> y <p:dataTable>) y un componente de PrimeFaces (<p:commandButton>) para crear el contenido de la primera pestaña.

<h1>: Este es un elemento HTML estándar que se utiliza para crear un encabezado de nivel 1. En este caso, se utiliza para mostrar el título de la página dentro del panel de la primera pestaña.

<p:commandButton>: Este es un componente de PrimeFaces que se utiliza para crear un botón que el usuario puede hacer clic para realizar una acción. En este caso, el botón está etiquetado con el texto "Haz clic aquí" y está asociado con el método doSomething del bean myBean.

<p:dataTable>: Este es un componente de PrimeFaces que se utiliza para crear una tabla de datos. En este caso, se utiliza para mostrar una lista de elementos (#{myBean.listaDeElementos}) con dos columnas (<p:column>): una para el nombre y otra para la descripción de cada elemento.

<p:column>: Este elemento se utiliza dentro de un componente <p:dataTable> para definir una columna de la tabla. En este caso, se utilizan dos elementos <p:column> para definir las dos columnas de la tabla de la primera pestaña.

<p:inputText>: Este es un componente de PrimeFaces que se utiliza para crear un campo de entrada de texto. En este caso, se utiliza para permitir que el usuario ingrese un valor (#{myBean.valorDeEntrada}) que se utilizará en el método doSomethingElse del bean myBean.

## MÉTODOS.

Antes de realizar la practica presente es necesario conocer algunos métodos que son necesarios para entender los códigos y algunas definiciones que se utilizaran durante la práctica.

Tabla 1. Métodos generalmente utilizados en conexión de java y bases de datos.

|  |  |
| --- | --- |
| MÉTODO | DESCRIPCIÓN |
| DriverManager.getConnection() | Este método se utiliza para establecer una conexión con la base de datos MySQL. Se puede utilizar con una cadena de conexión JDBC para especificar los detalles de la conexión, como el nombre de usuario y la contraseña. |
| Statement | Esta interfaz se utiliza para enviar sentencias SQL estáticas a la base de datos. Se puede utilizar para realizar operaciones de SELECT, INSERT, UPDATE y DELETE. |
| PreparedStatement | Esta interfaz se utiliza para enviar sentencias SQL parametrizadas a la base de datos. Es útil para evitar inyecciones SQL al usar parámetros en lugar de insertar directamente valores en la sentencia SQL. |
| ResultSet | Esta interfaz se utiliza para almacenar el resultado de una consulta SELECT. Se puede utilizar para recorrer los resultados de la consulta y recuperar los valores de cada columna. |
| com.mysql.jdbc.Driver | Esta clase se utiliza para cargar el controlador JDBC de MySQL. Se debe importar y registrar en el sistema antes de utilizar el DriverManager para establecer una conexión con la base de datos. |
| java.sql | Esta paquete contiene las clases y interfaces necesarias para trabajar con bases de datos en Java, incluyendo las mencionadas anteriormente. |
| SessionFactory.openSession() | Este método se utiliza para abrir una sesión con la base de datos. Es necesario para llevar a cabo cualquier operación CRUD con la base de datos. |
| Session.save() | Este método se utiliza para guardar un objeto en la base de datos. Es equivalente a una operación INSERT en SQL. |
| Session.update() | Este método se utiliza para actualizar un objeto en la base de datos. Es equivalente a una operación UPDATE en SQL. |
| Session.delete() | Este método se utiliza para eliminar un objeto de la base de datos. Es equivalente a una operación DELETE en SQL. |
| Session.get() | Este método se utiliza para recuperar un objeto de la base de datos a partir de su clave primaria. Es equivalente a una operación SELECT en SQL. |
| org.hibernate.SessionFactory | Esta clase es una fábrica para las sesiones de Hibernate. Es necesaria para abrir sesiones y realizar operaciones con la base de datos. |
| org.hibernate.Session | Esta clase representa una sesión con la base de datos. Es necesaria para llevar a cabo cualquier operación CRUD con la base de datos. |
| ActiveIndex | Índice de la pestaña activa para cambiar la pestaña seleccionada de forma programática. |
| controlCerrar | Si la pestaña cerrar se controla en el evento onClose o no. |
| estilo | Estilo en línea del componente. |
| clase de estilo | Clase de estilo del componente. |
| desplazable | Cuando está habilitado, muestra botones a cada lado de los encabezados de las pestañas para desplazar la lista de pestañas. |

## TÉRMINOS

Además, es necesario conocer el significado de una cantidad de términos un tanto complicados relacionados con la informática en cuanto a la conexión de una base de datos con java ee.

Tabla 2. Términos

|  |  |
| --- | --- |
| ORM (Object-Relational Mapping) | Una técnica que permite a un programador trabajar con objetos en lugar de con registros de una base de datos, haciendo que sea más fácil manipular y trabajar con datos en una aplicación. |
| JPA (Java Persistence API) | Una interfaz estándar para la implementación de ORM en aplicaciones Java EE. |
| Entity | Clases que representan las tablas de una base de datos en una aplicación JPA. |
| Persistence Unit | Un conjunto de entidades y configuraciones de una aplicación JPA. |
| EntityManager | Una interfaz para interactuar con entidades en una aplicación JPA. |
| JPQL (Java Persistence Query Language) | Un lenguaje de consulta similar a SQL para interactuar con entidades en una aplicación JPA. |
| Hibernate | Una popular implementación de ORM para aplicaciones Java. |
| MySQL Connector | Un driver que permite conectarse a una base de datos MySQL desde una aplicación Java. |
| DataSource | Un pool de conexiones a una base de datos que puede ser utilizado por una aplicación. |
| CRUD (Create, Read, Update, Delete) | Las operaciones básicas para interactuar con una base de datos, crear, leer, actualizar y eliminar registros. |
| Transacciones | Un conjunto de operaciones que se realizan en una base de datos como una unidad, ya sea todas se completen o ninguna. |
| DAO (Data Access Object) | Una clase o interfaz que proporciona métodos para interactuar con una base de datos. |
| JDBC (Java Database Connectivity) | Una API estándar para interactuar con bases de datos en aplicaciones Java. |
| Prepared Statement | Una sentencia SQL que ha sido precompilada para mejorar el rendimiento en aplicaciones JDBC. |
| ResultSet | Un conjunto de resultados obtenidos de una consulta a una base de datos en una aplicación JDBC. |
| Login | Proceso de verificación de credenciales de un usuario para acceder a un sistema o aplicación. |
| Session | Es un objeto que representa una conexión a la base de datos. Se utiliza para realizar operaciones CRUD en las entidades y manejar transacciones. |
| Persist | Es una operación que se utiliza para insertar una nueva entidad en la base de datos. |
| Merge | Es una operación que se utiliza para actualizar una entidad existente en la base de datos. |
| Remove | Es una operación que se utiliza para eliminar una entidad de la base de datos. |
| Find | Es una operación que se utiliza para recuperar una entidad de la base de datos a partir de su identificador. |
| Query | Es un objeto que se utiliza para ejecutar una consulta en la base de datos y recuperar una lista de resultados. |
| Criteria API | Es una interfaz de programación de consultas que se utiliza para construir consultas de forma dinámica. |
| JPQL | Es un lenguaje de consulta similar a SQL que se utiliza para realizar consultas en las entidades. |
| NamedQuery | Es una consulta que se define en la entidad mediante anotaciones o xml y se puede ejecutar mediante su nombre. |
| NativeQuery | Es una consulta que se ejecuta directamente en el motor de base de datos utilizando SQL nativo. |

# 4. DESARROLLO

Antes de continuar se debe tener en cuenta que el siguiente documento parte del trabajo realizado en Login\_Netbeans\_Grupo#, es decir, se omitirá la creación y codificación de algunos archivos y se asumirá su previa existencia .

En la presente practica se continuará trabajando sobre el programa realizado en el anterior proyecto, es decir, desde el Login hasta el listar usuarios en donde se presentaba el CRUD del personal tal y como se puede ver en la siguiente figura.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 6. Ultimo avance de CRUD realizado en personal.

Es por ello que, a partir de ahora en el documento, se comenzará a modificar una serie de archivos que fueron creados con anterioridad, así como una serie de nuevos a ser creados.

# 4.1 CREACIÓN DE VISTAS DEL PROYECTO

4.1.1 MODIFICACIÓN DE CRUD PARA PERSONAL.

Como se indicó anteriormente, se está trabajando sobre la base del trabajo previo sobre el Login y CRUDS por lo que será necesario hacer una actualización a los 3 archivos correspondientes a la vista de Personal, específicamente el archivo Create.xhtml, Edit.xhtml y View.xhtml, pero esta vez añadiendo las etiqueras <p:tabView> para poder integrar al proyecto marcos de página.

El primer paso será localizar los archivos, los cuales se encuentran en la carpeta peempEmple dentro de Web Pages.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 7. Ubicación del CRUD de Personal.

Una vez localizados y abiertos los archivos se procede a crear una nueva documentación añadiendo los marcos de página. Se dividirá y organizará la información en 3 pestañas, Información Personal, Información de Contacto e Información Laboral. Este cambio e implementación de marcos de página se incluirá en los 3 archivos de la vista referidos anteriormente.

Dentro del mismo lo que se debe hacer es crear un <h:panelGrid donde irán los TabView ubicados, paso siguiente se coloca la etiqueta de <p:tabView>. Dentro de esta ya se podrá usar <p:tab title=" "> segmentando lo que se coloca en cada una**.** Siendo entonces los archivos editados quedando de la siguiente manera.

Texto

Descripción generada automáticamente

*Figura 8. Nueva codificación para Create.xhtml.*

Texto

Descripción generada automáticamente

*Figura 9. Nueva codificación para Edit.xhtml*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Figura 10. Nueva codificación para View.xhtml*

Tabla . Nueva codificación del archivo peempEmple/Create.xhtml

<ui:composition

xmlns="http://www.w3.org/1999/xhtml"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:p="http://primefaces.org/ui"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:form enctype="multipart/form-data">

<f:event listener="#{peempEmpleController.limpiar}" type="preRenderComponent"/>

<p:dialog header="Crear Usuario" widgetVar="dlg2" modal="true" width="55%" height="75vh" style="font-size: 30px" >

<p:growl id="msgCedula" showDetail="true" life="7000" />

<p:panelGrid columns="2" style="font-size: 14px" >

<h:panelGrid columns="2" cellpadding="5px" >

<h:outputLabel for="peempCedula" value="Cédula:\*"/>

<h:inputText id="peempCedula" value="#{peempEmpleController.selected.peempCedula}">

<f:validator validatorId="cedulaValidator"/>

<p:ajax update="msgCedula" event="blur" />

</h:inputText>

<h:outputLabel value="Nombre:\*" for="peempNombre" />

<h:inputText id="peempNombre" value="#{peempEmpleController.selected.peempNombre}" title="#{bundle.CreatePeempEmpleTitle\_peempNombre}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempNombre}"/>

<h:outputLabel value="Apellido Paterno:\*" for="peempApellpaterno" />

<h:inputText id="peempApellpaterno" value="#{peempEmpleController.selected.peempApellpaterno}" title="#{bundle.CreatePeempEmpleTitle\_peempApellpaterno}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempApellpaterno}"/>

<h:outputLabel value="Apellido Materno:\*" for="peempApellmaterno" />

<h:inputText id="peempApellmaterno" value="#{peempEmpleController.selected.peempApellmaterno}" title="#{bundle.CreatePeempEmpleTitle\_peempApellmaterno}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_ppeempApellmaterno}"/>

</h:panelGrid>

<p:panelGrid columns="2" rendered="#{peempEmpleController.selected != null}" id="Image" styleClass="ui-noborder" >

<p:graphicImage value="../resources/img/user.png" style="width:100px;" rendered="#{peempEmpleController.foto == null}" cache="false"/>

<p:graphicImage id="imagePreview" value="#{peempEmpleController.imagepreview}" cache="false" style="width: 100px;" rendered="#{peempEmpleController.foto != null}"/>

<p:fileUpload sizeLimit="1048500" listener="#{peempEmpleController.preview}" mode="advanced" skinSimple="true" auto="true" update="Image"/>

<p:inputText id="aux" value="#{peempEmpleController.aux}" title="aux" type="hidden"/>

</p:panelGrid>

</p:panelGrid>

<h:panelGrid columns="1" style="width: 100%;font-size: 14px">

<p:tabView>

<p:tab title="Información Personal">

<h:panelGrid columns="2" border="0" cellpadding="5px" style="font-size: 14px">

<h:outputLabel value="Sexo:\*" for="pesexCodigo" />

<h:selectOneMenu id="pesexCodigo" value="#{peempEmpleController.selected.pesexCodigo}" title="#{bundle.CreatePeempEmpleTitle\_pesexCodigo}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_pesexCodigo}">

<f:selectItems value="#{pesexSexoController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Género:\*" for="codgenero" />

<h:selectOneMenu id="codgenero" value="#{peempEmpleController.selected.codgenero}" title="#{bundle.CreatePeempEmpleTitle\_codgenero}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_codgenero}">

<f:selectItems value="#{pegenGenerController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Fecha de Nacimiento:\*" for="peempFecnacimiento" />

<p:calendar id="peempFecnacimiento" value="#{peempEmpleController.selected.peempFecnacimiento}" title="#{bundle.CreatePeempEmpleTitle\_peempFecnacimiento}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempFecnacimiento}"

showOn="button" navigator="true">

<f:convertDateTime pattern="dd/MM/yy" />

</p:calendar>

<h:outputLabel value="Estado Civil:\*" for="peescCodigo" />

<h:selectOneMenu id="peescCodigo" value="#{peempEmpleController.selected.peescCodigo}" title="#{bundle.CreatePeempEmpleTitle\_peescCodigo}" >

<f:selectItems value="#{peestEstcivController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Nacionalidad:\*" for="codnacion" />

<h:selectOneMenu id="codnacion" value="#{peempEmpleController.selected.codnacion}" title="#{bundle.CreatePeempEmpleTitle\_codnacion}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_codnacion}">

<f:selectItems value="#{penacNacionController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

</h:panelGrid>

</p:tab>

<p:tab title="Información de Contacto">

<p:growl id="msgEmail" showDetail="true" life="7000" />

<h:panelGrid columns="2" cellpadding="5px">

<h:outputLabel value="País:\*" for="pepaPai" />

<h:selectOneMenu value="#{peempEmpleController.codPais}" style="display: block;width:250px;margin-top: 10px;">

<f:selectItem itemLabel="Seleccione un País" itemValue="0" noSelectionOption="true"></f:selectItem>

<f:selectItems value="#{peempEmpleController.listPaises}" var="pais" itemLabel="#{pais.pepaiDescri}" itemValue="#{pais.pepaiCodigo}" />

<p:ajax event="change" listener="#{peempEmpleController.obtenerProvincias()}" update="cboProvin cboCanton cboParroq"/>

</p:panelGrid>

<h:panelGrid columns="1" style="width: 100%;font-size: 14px">

<p:tabView>

<p:tab title="Información Personal">

<h:panelGrid columns="2" border="0" cellpadding="5px" style="font-size: 14px">

<h:outputLabel value="Sexo:\*" for="pesexCodigo" />

<h:selectOneMenu id="pesexCodigo" value="#{peempEmpleController.selected.pesexCodigo}" title="#{bundle.CreatePeempEmpleTitle\_pesexCodigo}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_pesexCodigo}">

<f:selectItems value="#{pesexSexoController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Género:\*" for="codgenero" />

<h:selectOneMenu id="codgenero" value="#{peempEmpleController.selected.codgenero}" title="#{bundle.CreatePeempEmpleTitle\_codgenero}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_codgenero}">

<f:selectItems value="#{pegenGenerController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Fecha de Nacimiento:\*" for="peempFecnacimiento" />

<p:calendar id="peempFecnacimiento" value="#{peempEmpleController.selected.peempFecnacimiento}" title="#{bundle.CreatePeempEmpleTitle\_peempFecnacimiento}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempFecnacimiento}"

showOn="button" navigator="true">

<f:convertDateTime pattern="dd/MM/yy" />

</p:calendar>

</h:selectOneMenu>

<h:outputLabel value="Provincia:\*" for="peproProv" />

<h:selectOneMenu id="cboProvin" value="#{peempEmpleController.codProvin}" style="display: block;width:250px;margin-top: 10px;">

<f:selectItem itemLabel="Seleccione una Provincia" itemValue="0" noSelectionOption="true"></f:selectItem>

<f:selectItems value="#{peempEmpleController.listProvin}" var="prov" itemLabel="#{prov.peproNombre}" itemValue="#{prov.peproProvinPK.peproCodigo}"/>

<p:ajax event="change" listener="#{peempEmpleController.obtenerCantones()}" update="cboCanton cboParroq"/>

</h:selectOneMenu>

<h:outputLabel value="Cantón:\*" for="pecanCan" />

<h:selectOneMenu id="cboCanton" value="#{peempEmpleController.codCanton}" style="display: block;width:250px;margin-top: 10px;">

<f:selectItem itemLabel="Seleccione un Cantón" itemValue="0" noSelectionOption="true"></f:selectItem>

<f:selectItems value="#{peempEmpleController.listCanton}" var="cant" itemLabel="#{cant.pecanDescri}" itemValue="#{cant.pecanCantonPK.pecanCodigo}"/>

<p:ajax event="change" listener="#{peempEmpleController.obtenerParroquias()}" update="cboParroq"/>

</h:selectOneMenu>

Tabla . Nueva codificación del archivo peempEmple/ View.xhtml

<ui:composition

xmlns="http://www.w3.org/1999/xhtml"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:p="http://primefaces.org/ui"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:form enctype="multipart/form-data" id="formViewUsuario">

<p:dialog header="Ver Usuario" widgetVar="digViewAcceso" modal="true" height="100%" style="font-size: 14px" id="digViewAcceso">

<p:panelGrid columns="2" style="width: 100%">

<h:panelGrid columns="2" cellpadding="7px">

<h:outputLabel value="Nombre:" for="peempNombre" />

<h:outputLabel id="peempNombre" value="#{peempEmpleController.selected.peempNombre}" title="#{bundle.CreatePeempEmpleTitle\_peempNombre}" />

<h:outputLabel value="Apellido Paterno:" for="peempApellpaterno" />

<h:outputLabel id="peempApellpaterno" value="#{peempEmpleController.selected.peempApellpaterno}" title="#{bundle.CreatePeempEmpleTitle\_peempApellpaterno}" />

<h:outputLabel value="Apellido Materno:" for="peempApellmaterno" />

<h:outputLabel id="peempApellmaterno" value="#{peempEmpleController.selected.peempApellmaterno}" title="#{bundle.CreatePeempEmpleTitle\_peempApellmaterno}" />

<h:outputLabel value="Cédula:" for="peempCedula" />

<h:outputLabel id="peempCedula" value="#{peempEmpleController.selected.peempCedula}" title="#{bundle.CreatePeempEmpleTitle\_peempCedula}" />

</h:panelGrid>

<p:panelGrid columns="1" rendered="#{peempEmpleController.selected != null}" id="Image" style="text-align: center" >

<p:graphicImage name="/img/#{peempEmpleController.selected.peempFoto}" width="100px" height="100px" rendered="#{peempEmpleController.selected.peempFoto ne 'NULL' }" />

</p:panelGrid>

</p:panelGrid>

<h:panelGrid columns="1" style="width: 100%">

<p:tabView>

<p:tab title="Información Personal">

<h:panelGrid columns="2" border="0" cellpadding="5px">

<h:outputLabel value="Sexo:" for="pesexCodigo" />

<h:outputLabel id="pesexCodigo" value="#{peempEmpleController.selected.pesexCodigo.pesexNomb}" title="#{bundle.CreatePeempEmpleTitle\_pesexCodigo}" />

<h:outputLabel value="Género:" for="codgenero" />

<h:outputLabel id="codgenero" value="#{peempEmpleController.selected.codgenero.descripcion}" title="#{bundle.CreatePeempEmpleTitle\_codgenero}" />

<h:outputLabel value="Fecha de Nacimiento:" for="peempFecnac" />

<h:outputLabel id="peempFecnac" value="#{peempEmpleController.selected.peempFecnacimiento}" title="#{bundle.CreatePeempEmpleTitle\_peempFecnacimiento}">

<f:convertDateTime pattern="MM/dd/yyyy" />

</h:outputLabel>

<h:outputLabel value="Estado Civil:" for="peescCodigo" />

<h:outputLabel id="peescCodigo" value="#{peempEmpleController.selected.peescCodigo.peescNomb}" title="#{bundle.CreatePeempEmpleTitle\_peescCodigo}" />

<h:outputLabel value="Nacionalidad:" for="codignac" />

<h:outputLabel id="codignac" value="#{peempEmpleController.selected.codnacion.descrinac}" title="#{bundle.CreatePeempEmpleTitle\_codnacion}" />

</h:panelGrid>

</p:tab>

<p:tab title="Información Laboral">

<h:panelGrid columns="2" cellpadding="5px">

<h:outputLabel value="Salario:" for="peempSalario" />

<h:outputLabel id="peempSalario" value="$#{peempEmpleController.selected.peempSalario}" title="#{bundle.CreatePeempEmpleTitle\_peempSalario}" />

<h:outputLabel value="Departamento:" for="coddepart" />

<h:outputLabel id="coddepart" value="#{peempEmpleController.selected.coddepart.nombredepto}" title="#{bundle.CreatePeempEmpleTitle\_coddepart}" />

</h:panelGrid>

</p:tab>

</p:tabView>

</h:panelGrid>

</p:dialog>

</h:form>

</ui:composition>

Tabla . Nueva codificación del archivo peempEmple/ Edit.xhtml

<ui:composition

xmlns="http://www.w3.org/1999/xhtml"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:p="http://primefaces.org/ui"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:form enctype="multipart/form-data" id="formEditarUsuario">

<p:dialog header="Editar Usuario" widgetVar="digEditarAcceso" modal="true" style="font-size: 14px" id="digEditarAcceso" width="55%" height="75vh">

<p:panelGrid columns="2">

<h:panelGrid columns="2" cellpadding="5px">

<h:outputLabel value="Código:\*" for="peempCurp" />

<h:inputText id="peempCurp" value="#{peempEmpleController.selected.peempCurp}" title="#{bundle.CreatePeempEmpleTitle\_peempCurp}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempCurp}" readonly="true"/>

<h:outputLabel value="Cédula:\*" for="peempCedula" />

<h:inputText id="peempCedula" value="#{peempEmpleController.selected.peempCedula}" title="#{bundle.CreatePeempEmpleTitle\_peempCedula}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempCedula}" readonly="true"/>

<h:outputLabel value="Nombre:\*" for="peempNombre" />

<h:inputText id="peempNombre" value="#{peempEmpleController.selected.peempNombre}" title="#{bundle.CreatePeempEmpleTitle\_peempNombre}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempNombre}"/>

<h:outputLabel value="Apellido Paterno:\*" for="peempApellpaterno" />

<h:inputText id="peempApellpaterno" value="#{peempEmpleController.selected.peempApellpaterno}" title="#{bundle.CreatePeempEmpleTitle\_peempApellpaterno}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempApellpaterno}"/>

<h:outputLabel value="Apellido Materno\*" for="peempApellmaterno" />

<h:inputText id="peempApellmaterno" value="#{peempEmpleController.selected.peempApellmaterno}" title="#{bundle.CreatePeempEmpleTitle\_peempApellmaterno}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempApellmaterno}"/>

</h:panelGrid>

<p:panelGrid columns="2" rendered="#{peempEmpleController.selected != null}" id="Image" styleClass="ui-noborder" >

<p:graphicImage name="/img/#{peempEmpleController.selected.peempFoto}" width="100px" height="100px" rendered="#{peempEmpleController.foto == null}" cache="false"/>

<p:graphicImage id="imagePreview" value="#{peempEmpleController.imagepreview}" cache="false" style="width: 100px;" rendered="#{peempEmpleController.foto != null}"/>

<p:fileUpload sizeLimit="1048500" listener="#{peempEmpleController.preview}" mode="advanced" skinSimple="true" auto="true" update="Image"/>

<p:inputText id="aux" value="#{peempEmpleController.aux}" title="aux" type="hidden"/>

</p:panelGrid>

</p:panelGrid>

<h:panelGrid columns="1" style="width: 100%">

<p:tabView>

<p:tab title="Información Personal">

<h:panelGrid columns="2" border="0" cellpadding="5px">

<h:outputLabel value="Sexo:\*" for="pesexCodigo" />

<h:selectOneMenu id="pesexCodigo" value="#{peempEmpleController.selected.pesexCodigo}" title="#{bundle.CreatePeempEmpleTitle\_pesexCodigo}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_pesexCodigo}">

<f:selectItems value="#{pesexSexoController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Género:\*" for="codgenero" />

<h:selectOneMenu id="codgenero" value="#{peempEmpleController.selected.codgenero}" title="#{bundle.CreatePeempEmpleTitle\_codgenero}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_codgenero}">

<f:selectItems value="#{pegenGenerController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Fecha de Nacimiento:\*" for="peempFecnacimiento" />

<p:calendar id="peempFecnacimiento" value="#{peempEmpleController.selected.peempFecnacimiento}" title="#{bundle.CreatePeempEmpleTitle\_peempFecnacimiento}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempFecnacimiento}"

showOn="button" navigator="true">

<f:convertDateTime pattern="dd/MM/yy" />

</p:calendar>

<h:outputLabel value="Estado Civil:\*" for="peescCodigo" />

<h:selectOneMenu id="peescCodigo" value="#{peempEmpleController.selected.peescCodigo}" title="#{bundle.CreatePeempEmpleTitle\_peescCodigo}" >

<f:selectItems value="#{peestEstcivController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

<h:outputLabel value="Nacionalidad:\*" for="codnacion" />

<h:selectOneMenu id="codnacion" value="#{peempEmpleController.selected.codnacion}" title="#{bundle.CreatePeempEmpleTitle\_codnacion}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_codnacion}">

<f:selectItems value="#{penacNacionController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

</h:panelGrid>

</p:tab>

<p:tab title="Información de Contacto">

<p:growl id="msgEmail" showDetail="true" life="7000" />

<h:panelGrid columns="2" cellpadding="5px">

<h:outputLabel value="Dirección/Calle Principal:\*" for="pedirCalleprincipal" />

<h:inputText id="pedirCalleprincipal" value="#{peempEmpleController.selected.pedirCodigodireccion.pedirCalleprincipal}" title="#{bundle.CreatePeempEmpleTitle\_pedirCalleprincipal}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_pedirCalleprincipal}"/>

<h:outputLabel value="Dirección/Calle Secundaria:\*" for="pedirCallesecundaria" />

<h:inputText id="pedirCallesecundaria" value="#{peempEmpleController.selected.pedirCodigodireccion.pedirCallesecundaria}" title="#{bundle.CreatePeempEmpleTitle\_pedirCallesecundaria}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_pedirCallesecundaria}"/>

<h:outputLabel value="Teléfono:\*" for="peempTelefono" />

<h:inputText id="peempTelefono" value="#{peempEmpleController.selected.peempTelefono}" title="#{bundle.CreatePeempEmpleTitle\_peempTelefono}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempTelefono}"/>

<h:outputLabel value="Email:\*" for="peempEmail" />

<h:inputText id="peempEmail" value="#{peempEmpleController.selected.peempEmail}" title="#{bundle.CreatePeempEmpleTitle\_peempEmail}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempEmail}">

<f:validator validatorId="EmailValidator"/>

<p:ajax update="msgEmail" event="blur" />

</h:inputText>

<h:outputLabel value="País/Provincia/Cantón/Parroquia:\*" for="peparParroq" />

<h:selectOneMenu id="peparParroq" value="#{peempEmpleController.selected.pedirCodigodireccion.peparParroq}" title="#{bundle.CreatePeempEmpleTitle\_peparParroq}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peparParroq}">

<f:selectItems value="#{peparParroqController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

</h:panelGrid>

</p:tab>

<p:tab title="Información Laboral">

<h:panelGrid columns="2" cellpadding="5px">

<h:outputLabel value="Salario:\*" for="peempSalario" />

<h:inputText id="peempSalario" value="#{peempEmpleController.selected.peempSalario}" title="#{bundle.CreatePeempEmpleTitle\_peempSalario}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_peempSalario}"/>

<h:outputLabel value="Departamento:\*" for="coddepart" />

<h:selectOneMenu id="coddepart" value="#{peempEmpleController.selected.coddepart}" title="#{bundle.CreatePeempEmpleTitle\_coddepart}" required="true" requiredMessage="#{bundle.CreatePeempEmpleRequiredMessage\_coddepart}">

<f:selectItems value="#{tedepDepartController.itemsAvailableSelectOne}"/>

</h:selectOneMenu>

</h:panelGrid>

</p:tab>

</p:tabView>

<p:commandButton action="#{peempEmpleController.update}" value="Editar" ajax="false"/>

</h:panelGrid>

</p:dialog>

</h:form>

</ui:composition>

# 4.2 CREACIÓN DE MODELOS DEL PROYECTO

### 4.2.1 CREACIÓN DE MÉTODOS Y CONSULTAS DEL MODELO

Para poder desplegar la información del personal se necesita hacer uso del modelo que trae la información de la base de datos, sin embargo, el que se creó anteriormente en el documento de Login es muy básico y no permite obtener toda la información deseada de forma personalizada, es por ello por lo que se necesitan nuevos métodos, consultas, querys y estructuras para poder utilizarlas en el controlador.

Para codificar los nuevos métodos se deben modificar dos archivos. El primero se encuentra en la carpeta ec.edu.monster.modelo y el archivo a modificar se llama PeempEmple.java.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Selección del archivo para modificar.

Se abre el archivo y se puede colocar los nuevos métodos en el mismo.

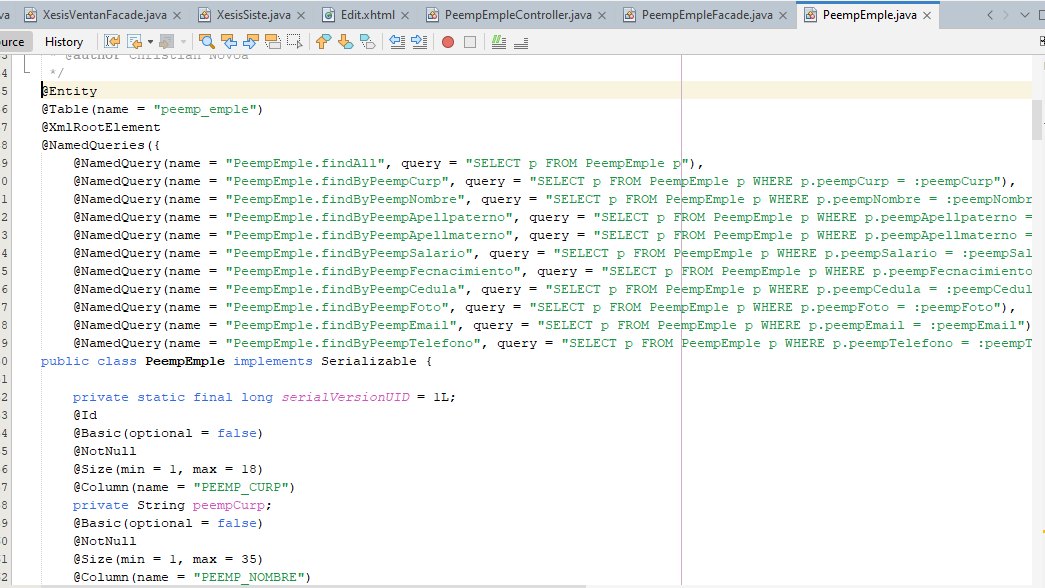


Figura . Codificación del archivo PeempEmple.java.

Tabla . Codificación del archivo PeempEmple.java

package ec.edu.monster.modelo;

import java.io.Serializable;

import java.math.BigDecimal;

import java.util.Collection;

import java.util.Date;

import java.util.Random;

import javax.persistence.Basic;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlTransient;

@Entity

@Table(name = "peemp\_emple")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "PeempEmple.findAll", query = "SELECT p FROM PeempEmple p"),

@NamedQuery(name = "PeempEmple.findByPeempCurp", query = "SELECT p FROM PeempEmple p WHERE p.peempCurp = :peempCurp"),

@NamedQuery(name = "PeempEmple.findByPeempNombre", query = "SELECT p FROM PeempEmple p WHERE p.peempNombre = :peempNombre"),

@NamedQuery(name = "PeempEmple.findByPeempApellpaterno", query = "SELECT p FROM PeempEmple p WHERE p.peempApellpaterno = :peempApellpaterno"),

@NamedQuery(name = "PeempEmple.findByPeempApellmaterno", query = "SELECT p FROM PeempEmple p WHERE p.peempApellmaterno = :peempApellmaterno"),

@NamedQuery(name = "PeempEmple.findByPeempSalario", query = "SELECT p FROM PeempEmple p WHERE p.peempSalario = :peempSalario"),

@NamedQuery(name = "PeempEmple.findByPeempFecnacimiento", query = "SELECT p FROM PeempEmple p WHERE p.peempFecnacimiento = :peempFecnacimiento"),

@NamedQuery(name = "PeempEmple.findByPeempCedula", query = "SELECT p FROM PeempEmple p WHERE p.peempCedula = :peempCedula"),

@NamedQuery(name = "PeempEmple.findByPeempFoto", query = "SELECT p FROM PeempEmple p WHERE p.peempFoto = :peempFoto"),

@NamedQuery(name = "PeempEmple.findByPeempEmail", query = "SELECT p FROM PeempEmple p WHERE p.peempEmail = :peempEmail"),

@NamedQuery(name = "PeempEmple.findByPeempTelefono", query = "SELECT p FROM PeempEmple p WHERE p.peempTelefono = :peempTelefono")})

public class PeempEmple implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 18)

@Column(name = "PEEMP\_CURP")

private String peempCurp;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 35)

@Column(name = "PEEMP\_NOMBRE")

private String peempNombre;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 35)

@Column(name = "PEEMP\_APELLPATERNO")

private String peempApellpaterno;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 35)

@Column(name = "PEEMP\_APELLMATERNO")

private String peempApellmaterno;

// @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation

@Basic(optional = false)

@NotNull

@Column(name = "PEEMP\_SALARIO")

private BigDecimal peempSalario;

@Basic(optional = false)

@NotNull

@Column(name = "PEEMP\_FECNACIMIENTO")

@Temporal(TemporalType.DATE)

private Date peempFecnacimiento;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 10)

@Column(name = "PEEMP\_CEDULA")

private String peempCedula;

@Size(max = 100)

@Column(name = "PEEMP\_FOTO")

private String peempFoto;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 100)

@Column(name = "PEEMP\_EMAIL")

private String peempEmail;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 15)

@Column(name = "PEEMP\_TELEFONO")

private String peempTelefono;

@OneToMany(cascade = CascadeType.ALL, mappedBy = "peempCurp")

private Collection<XeusuUsuar> xeusuUsuarCollection;

@OneToMany(cascade = CascadeType.ALL, mappedBy = "peempCurp")

private Collection<PtdirDirigi> ptdirDirigiCollection;

@JoinColumn(name = "PEDIR\_CODIGODIRECCION", referencedColumnName = "PEDIR\_CODIGODIRECCION")

@ManyToOne(optional = false)

private PedirDirecc pedirCodigodireccion;

@OneToMany(mappedBy = "peePeempCurp")

private Collection<PeempEmple> peempEmpleCollection;

@JoinColumn(name = "PEE\_PEEMP\_CURP", referencedColumnName = "PEEMP\_CURP")

@ManyToOne

private PeempEmple peePeempCurp;

@JoinColumn(name = "PEESC\_CODIGO", referencedColumnName = "PEESC\_CODIGO")

@ManyToOne(optional = false)

private PeestEstciv peescCodigo;

@JoinColumn(name = "CODGENERO", referencedColumnName = "CODGENERO")

@ManyToOne(optional = false)

private PegenGener codgenero;

@JoinColumn(name = "CODNACION", referencedColumnName = "CODNACION")

@ManyToOne(optional = false)

private PenacNacion codnacion;

@JoinColumn(name = "PESEX\_CODIGO", referencedColumnName = "PESEX\_CODIGO")

@ManyToOne(optional = false)

private PesexSexo pesexCodigo;

@JoinColumn(name = "CODDEPART", referencedColumnName = "CODDEPART")

@ManyToOne(optional = false)

private TedepDepart coddepart;

@OneToMany(cascade = CascadeType.ALL, mappedBy = "peempCurp")

private Collection<TeparPartic> teparParticCollection;

@OneToOne(cascade = CascadeType.ALL, mappedBy = "peempEmple")

private PefamFamili pefamFamili;

public PeempEmple() {}

public PeempEmple(String peempCurp) {

this.peempCurp = peempCurp;

}

public PeempEmple(String peempCurp, String peempNombre, String peempApellpaterno, String peempApellmaterno, BigDecimal peempSalario, Date peempFecnacimiento, String peempCedula, String peempEmail, String peempTelefono) {

this.peempCurp = peempCurp;

this.peempNombre = peempNombre;

this.peempApellpaterno = peempApellpaterno;

this.peempApellmaterno = peempApellmaterno;

this.peempSalario = peempSalario;

this.peempFecnacimiento = peempFecnacimiento;

this.peempCedula = peempCedula;

this.peempEmail = peempEmail;

this.peempTelefono = peempTelefono;

}

public String getPeempCurp() {

return peempCurp;

}

public void setPeempCurp(String peempCurp) {

this.peempCurp = peempCurp;

}

public String getPeempNombre() {

return peempNombre;

}

public void setPeempNombre(String peempNombre) {

this.peempNombre = peempNombre;

}

public String getPeempApellpaterno() {

return peempApellpaterno;

}

public void setPeempApellpaterno(String peempApellpaterno) {

this.peempApellpaterno = peempApellpaterno;

}

public String getPeempApellmaterno() {

return peempApellmaterno;

}

public void setPeempApellmaterno(String peempApellmaterno) {

this.peempApellmaterno = peempApellmaterno;

}

public BigDecimal getPeempSalario() {

return peempSalario;

}

public void setPeempSalario(BigDecimal peempSalario) {

this.peempSalario = peempSalario;

}

public Date getPeempFecnacimiento() {

return peempFecnacimiento;

}

public void setPeempFecnacimiento(Date peempFecnacimiento) {

this.peempFecnacimiento = peempFecnacimiento;

}

public String getPeempCedula() {

return peempCedula;

}

public void setPeempCedula(String peempCedula) {

this.peempCedula = peempCedula;

}

public String getPeempFoto() {

return peempFoto;

}

public void setPeempFoto(String peempFoto) {

this.peempFoto = peempFoto;

}

public String getPeempEmail() {

return peempEmail;

}

public void setPeempEmail(String peempEmail) {

this.peempEmail = peempEmail;

}

public String getPeempTelefono() {

return peempTelefono;

}

public void setPeempTelefono(String peempTelefono) {

this.peempTelefono = peempTelefono;

}

@XmlTransient

public Collection<XeusuUsuar> getXeusuUsuarCollection() {

return xeusuUsuarCollection;

}

public void setXeusuUsuarCollection(Collection<XeusuUsuar> xeusuUsuarCollection) {

this.xeusuUsuarCollection = xeusuUsuarCollection;

}

@XmlTransient

public Collection<PtdirDirigi> getPtdirDirigiCollection() {

return ptdirDirigiCollection;

}

public void setPtdirDirigiCollection(Collection<PtdirDirigi> ptdirDirigiCollection) {

this.ptdirDirigiCollection = ptdirDirigiCollection;

}

public PedirDirecc getPedirCodigodireccion() {

return pedirCodigodireccion;

}

public void setPedirCodigodireccion(PedirDirecc pedirCodigodireccion) {

this.pedirCodigodireccion = pedirCodigodireccion;

}

@XmlTransient

public Collection<PeempEmple> getPeempEmpleCollection() {

return peempEmpleCollection;

}

public void setPeempEmpleCollection(Collection<PeempEmple> peempEmpleCollection) {

this.peempEmpleCollection = peempEmpleCollection;

}

public PeempEmple getPeePeempCurp() {

return peePeempCurp;

}

public void setPeePeempCurp(PeempEmple peePeempCurp) {

this.peePeempCurp = peePeempCurp;

}

public PeestEstciv getPeescCodigo() {

return peescCodigo;

}

public void setPeescCodigo(PeestEstciv peescCodigo) {

this.peescCodigo = peescCodigo;

}

public PegenGener getCodgenero() {

return codgenero;

}

public void setCodgenero(PegenGener codgenero) {

this.codgenero = codgenero;

}

public PenacNacion getCodnacion() {

return codnacion;

}

public void setCodnacion(PenacNacion codnacion) {

this.codnacion = codnacion;

}

public PesexSexo getPesexCodigo() {

return pesexCodigo;

}

public void setPesexCodigo(PesexSexo pesexCodigo) {

this.pesexCodigo = pesexCodigo;

}

public TedepDepart getCoddepart() {

return coddepart;

}

public void setCoddepart(TedepDepart coddepart) {

this.coddepart = coddepart;

}

@XmlTransient

public Collection<TeparPartic> getTeparParticCollection() {

return teparParticCollection;

}

public void setTeparParticCollection(Collection<TeparPartic> teparParticCollection) {

this.teparParticCollection = teparParticCollection;

}

public PefamFamili getPefamFamili() {

return pefamFamili;

}

public void setPefamFamili(PefamFamili pefamFamili) {

this.pefamFamili = pefamFamili;

}

@Override

public int hashCode() {

int hash = 0;

hash += (peempCurp != null ? peempCurp.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof PeempEmple)) {

return false;

}

PeempEmple other = (PeempEmple) object;

if ((this.peempCurp == null && other.peempCurp != null) || (this.peempCurp != null && !this.peempCurp.equals(other.peempCurp))) {

return false;

}

return true;

}

@Override

public String toString() {

return peempCurp;

}

public String generarContraseña() {

Random rdn = new Random();

String caracteres = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";

int longitud = caracteres.length();

char letra;

int longitudContrasenia = 6;

String contraseniaAleatoria = "";

for (int i = 0; i < longitudContrasenia; i++) {

letra = caracteres.charAt(rdn.nextInt(longitud));

contraseniaAleatoria=contraseniaAleatoria+letra;

}

return contraseniaAleatoria;

}

}

El segundo archivo se encuentra en la misma carpeta ec.edu.monster.modelo y su nombre a buscar es PeempEmpleFacade.java.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Selección del archivo para modificar.

Se abre el archivo y se procede colocar los nuevos métodos en el mismo.

Texto

Descripción generada automáticamente

Figura . Codificación del archivo PeempEmpleFacade.java.

Tabla . Codificación del archivo PeempEmpleFacade.java

package ec.edu.monster.modelo;

import java.util.Date;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.Query;

@Stateless

public class PeempEmpleFacade extends AbstractFacade<PeempEmple> {

@PersistenceContext(unitName = "LOGIN\_JAVA\_GRUPO6PU")

private EntityManager em;

@Override

protected EntityManager getEntityManager() {

return em;

}

public PeempEmpleFacade() {

super(PeempEmple.class);

}

public List<PeempEmple> obtenerEmpleados(){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p";

Query query = em.createQuery(consulta);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public String crearCodigo(){

String consulta = "";

int maximoCodigo = 0;

try {

consulta = "SELECT MAX(p.peempCurp) FROM PeempEmple p";

Query query = em.createQuery(consulta);

Object result = query.getSingleResult();

if (result != null) {

maximoCodigo = Integer.parseInt(result.toString())+1;

}

} catch (Exception e) {

throw e;

}

return maximoCodigo+"";

}

public PedirDirecc consultarDireccion(String cod){

PedirDirecc pedir = null;

String consulta="";

try{

consulta ="SELECT p FROM PedirDirecc p WHERE p.pedirCodigodireccion = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, cod);

query.setMaxResults(1);

pedir = (PedirDirecc) query.getSingleResult();

}catch(Exception e){

throw e;

}

return pedir;

}

public List<PeempEmple> obtenerEmpleadosGeneros(String idGenero){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.pesexCodigo.pesexCodigo = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, idGenero);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosPorNombre(String nombre){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.pesexCodigo.pesexCodigo = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, nombre);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosDepartamentos(String tedepCodigo){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.coddepart.coddepart = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, tedepCodigo);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosFechas(Date fecha){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.peempFecnacimiento >= ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, fecha);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosPais(String codPais){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.pedirCodigodireccion.peparParroq.peparParroqPK.pepaiCodigo = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, codPais);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosProvin(String codProvin){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.pedirCodigodireccion.peparParroq.peparParroqPK.peproCodigo = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, codProvin);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosCanton(String codCanton){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.pedirCodigodireccion.peparParroq.peparParroqPK.pecanCodigo = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, codCanton);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

public List<PeempEmple> obtenerEmpleadosParroq(String codParr){

List<PeempEmple> lista= null;

String consulta="";

try{

consulta ="SELECT p FROM PeempEmple p WHERE p.pedirCodigodireccion.peparParroq.peparParroqPK.peparCodigo = ?1";

Query query = em.createQuery(consulta);

query.setParameter(1, codParr);

lista = query.getResultList();

}catch(Exception e){

throw e;

}

return lista;

}

}

# 4.3 CREACIÓN DE CONTROLADORES DEL PROYECTO

Un controlador como se indicó con anterioridad es aquel que se encarga de actuar a manera de comunicador entre la vista y el modelo, recibiendo ordenes de la vista, procesándolos y modificando respectivamente el modelo, en este caso nuestro controlador recibiría ordenes desde el archivo index para verificar si las credenciales son las correctas y este solicitaría tal información a los archivos del modelo.

En este caso con la generación de los archivos del modelo y la vista ya se generaron los archivos del controlador, por lo que sólo hace falta añadir algunos archivos adicionales.

### 4.3.1 CODIFICACIÓN DEL ARCHIVO PEEMPLECONTROLLER.

De clic izquierdo sobre la carpeta ec.edu.monster.controlador y escoja el archivo bajo el nombre PeempEmpleController.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Selección del archivo Java Class.

Se procede a colocar la siguiente codificación para utilizar los métodos del modelo y mandarlos a las vistas para organizar la información con marcos de página.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura . Codificación del archivo PeempEmpleController,java

Tabla 3. Codificación del archivo index.jsp

Tabla . Codificación del archivo controlador PeempEmpleController.java

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.PeempEmple;

import ec.edu.monster.controlador.util.JsfUtil;

import ec.edu.monster.controlador.util.JsfUtil.PersistAction;

import ec.edu.monster.controlador.util.PaginationHelper;

import ec.edu.monster.modelo.Correo;

import ec.edu.monster.modelo.PecanCanton;

import ec.edu.monster.modelo.PedirDirecc;

import ec.edu.monster.modelo.PeempEmpleFacade;

import ec.edu.monster.modelo.PepaiPais;

import ec.edu.monster.modelo.PeparParroq;

import ec.edu.monster.modelo.PeparParroqPK;

import ec.edu.monster.modelo.PeproProvin;

import ec.edu.monster.modelo.XeestEstad;

import ec.edu.monster.modelo.XeperPerfi;

import ec.edu.monster.modelo.XeusuUsuar;

import ec.edu.monster.modelo.XeuxpUsupe;

import java.io.ByteArrayInputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.Serializable;

import java.sql.Date;

import java.util.Formatter;

import java.util.List;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.ejb.EJB;

import javax.ejb.EJBException;

import javax.inject.Named;

import javax.enterprise.context.SessionScoped;

import javax.faces.application.FacesMessage;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;

import javax.faces.convert.Converter;

import javax.faces.convert.FacesConverter;

import javax.faces.event.ComponentSystemEvent;

import javax.faces.model.DataModel;

import javax.faces.model.ListDataModel;

import org.apache.commons.codec.digest.DigestUtils;

import org.primefaces.event.FileUploadEvent;

import org.primefaces.model.DefaultStreamedContent;

import org.primefaces.model.StreamedContent;

import org.primefaces.model.file.UploadedFile;

@Named("peempEmpleController")

@SessionScoped

public class PeempEmpleController implements Serializable {

private DataModel items = null;

private XeusuUsuar usuario;

private UploadedFile foto;

private String aux;

private XeperPerfi rol;

private XeuxpUsupe perfilUser;

private PedirDirecc direccion;

private StreamedContent imagepreview;

private String inLimiteAvaliacao;

private boolean mostrar = true;

private String codPais;

private List<PepaiPais> listPaises;

private String codProvin;

private List<PeproProvin> listProvin;

private String codCanton;

private List<PecanCanton> listCanton;

private String codParroquia;

private List<PeparParroq> listParroquias;

private String opcFiltro;

private String idGenero;

private List<PeempEmple> empleadoList;

private String codDepartamento;

private java.util.Date fechaFiltro;

private String nombreFiltro;

private String pedirCalleprincipal;

private String pedirCallesecundaria;

private String valor;

private int flag = 1;

private XeestEstad xeestCodigo;

private PaginationHelper pagination;

private int selectedItemIndex;

@EJB

private ec.edu.monster.modelo.PeempEmpleFacade ejbFacade;

private PeempEmple selected;

private PeempEmple current;

@EJB

private ec.edu.monster.modelo.PepaiPaisFacade ejbPais;

@EJB

private ec.edu.monster.modelo.PeproProvinFacade ejbProvincia;

@EJB

private ec.edu.monster.modelo.PecanCantonFacade ejbCanton;

@EJB

private ec.edu.monster.modelo.PeparParroqFacade ejbparroquia;

@EJB

private ec.edu.monster.modelo.XeusuUsuarFacade ejbUsuario;

@EJB

private ec.edu.monster.modelo.PedirDireccFacade ejbDireccion;

public PeempEmpleController() {

}

public void limiteAvaliacao(){

if(inLimiteAvaliacao.equals("valor1")){

mostrar = false;

} else {

mostrar = true;

}

}

public boolean isMostrar() {

return mostrar;

}

public void setMostrar(boolean mostrar) {

this.mostrar = mostrar;

}

public String getOpcFiltro() {

return opcFiltro;

}

public void setOpcFiltro(String opcFiltro) {

this.opcFiltro = opcFiltro;

}

public String getIdGenero() {

return idGenero;

}

public void setIdGenero(String idGenero) {

this.idGenero = idGenero;

}

public String getCodDepartamento() {

return codDepartamento;

}

public void setCodDepartamento(String codDepartamento) {

this.codDepartamento = codDepartamento;

}

public void obtenerEmpleadosGeneros(){

empleadoList= this.ejbFacade.obtenerEmpleadosGeneros(idGenero);

}

public void obtenerEmpleadosDepartamentos(){

empleadoList= this.ejbFacade.obtenerEmpleadosDepartamentos(codDepartamento);

}

public void obtenerEmpleadosPais(){

empleadoList= this.ejbFacade.obtenerEmpleadosPais(codPais);

}

public void obtenerEmpleadosProvin(){

empleadoList= this.ejbFacade.obtenerEmpleadosProvin(codProvin);

}

public void obtenerEmpleadosCanton(){

empleadoList= this.ejbFacade.obtenerEmpleadosCanton(codCanton);

}

public void obtenerEmpleadosParroq(){

empleadoList= this.ejbFacade.obtenerEmpleadosParroq(codParroquia);

}

public void obtenerEmpleadosFechas(){

empleadoList= this.ejbFacade.obtenerEmpleadosFechas(fechaFiltro);

}

public void obtenerEmpleadosNombre(){

empleadoList= this.ejbFacade.obtenerEmpleadosPorNombre(nombreFiltro);

}

public java.util.Date getFechaFiltro() {

return fechaFiltro;

}

public void setFechaFiltro(java.util.Date fechaFiltro) {

this.fechaFiltro = fechaFiltro;

}

public List<PeempEmple> getEmpleadoList() {

if(empleadoList==null)

empleadoList=ejbFacade.obtenerEmpleados();

return empleadoList;

}

public PeempEmple getSelected() {

if (current == null) {

current = new PeempEmple();

selectedItemIndex = -1;

}

return current;

}

public void setSelected(PeempEmple selected) {

this.selected = selected;

}

public String getAux() {

return aux;

}

public void setAux(String aux) {

this.aux = aux;

}

public List<PepaiPais> getListPaises() {

if(listPaises==null)

listPaises=ejbPais.findAll();

return listPaises;

}

public void setListPaises(List<PepaiPais> listPaises) {

this.listPaises = listPaises;

}

public String getCodPais() {

return codPais;

}

public void setCodPais(String codPais) {

this.codPais = codPais;

}

public String getCodProvin() {

return codProvin;

}

public void setCodProvin(String codProvin) {

this.codProvin = codProvin;

}

public List<PeproProvin> getListProvin() {

return listProvin;

}

public void setListProvin(List<PeproProvin> listProvincias) {

this.listProvin = listProvincias;

}

public void obtenerProvincias(){

if(listProvin!=null){

codCanton="";

codParroquia="";

listCanton= null;

listParroquias = null;

}

listProvin=ejbProvincia.obtenerProvinciasPais(codPais);

}

public void obtenerCantones(){

if(listCanton!=null){

listParroquias= null;

codParroquia="";

}

listCanton=ejbCanton.obtenerCantonesProvincia(codProvin);

}

public String getCodCanton() {

return codCanton;

}

public void setCodCanton(String codCanton) {

this.codCanton = codCanton;

}

public List<PecanCanton> getListCanton() {

return listCanton;

}

public void setListCanton(List<PecanCanton> listCanton) {

this.listCanton = listCanton;

}

public void obtenerParroquias(){

listParroquias=ejbparroquia.obtenerParrquiasCantones(codCanton);

}

public String getCodParroquia() {

return codParroquia;

}

public void setCodParroquia(String codParroquia) {

this.codParroquia = codParroquia;

}

public List<PeparParroq> getListParroquias() {

return listParroquias;

}

public void setListParroquias(List<PeparParroq> listParroquias) {

this.listParroquias = listParroquias;

}

protected void setEmbeddableKeys() {

}

protected void initializeEmbeddableKey() {

}

private PeempEmpleFacade getFacade() {

return ejbFacade;

}

public UploadedFile getFoto() {

return foto;

}

public void setFoto(UploadedFile foto) {

this.foto = foto;

}

public StreamedContent getImagepreview() {

return imagepreview;

}

public void setImagepreview(StreamedContent imagepreview) {

this.imagepreview = imagepreview;

}

public void preview(FileUploadEvent event) {

foto = event.getFile();

System.out.println("Uploaded");

System.out.println(event.getFile().getFileName());

imagepreview = DefaultStreamedContent.builder()

.contentType("image/\*")

.stream(() -> {

try {

return new ByteArrayInputStream(event.getFile().getContent());

} catch (Exception e) {

e.printStackTrace();

return null;

}

})

.build();

//return Base64.getEncoder().encodeToString(image.getContent());

//return Base64.getEncoder().encodeToString(image.getContent());

}

public String prepareCreate() {

foto=null;

current = new PeempEmple();

selectedItemIndex = -1;

recreatePagination();

recreateModel();

addSuccessMessage("Completado","Usuario Creado correctamente");

return "List";

}

public String prepareCreateInvitado() {

foto=null;

current = new PeempEmple();

selectedItemIndex = -1;

recreatePagination();

recreateModel();

addSuccessMessage("Completado","Usuario Creado correctamente, revise su correo para recibir su contraseña");

return "/index.xhtml";

}

private void recreatePagination() {

pagination = null;

}

private void recreateModel() {

items = null;

}

public void addSuccessMessage(String msg2,String msg) {

FacesMessage facesMsg = new FacesMessage(FacesMessage.SEVERITY\_INFO, msg2, msg);

FacesContext.getCurrentInstance().addMessage("successInfo", facesMsg);

}

public String create() {

try {

current.setPeempCurp(ejbFacade.crearCodigo());

subirArchivo();

current.setPeempFoto(aux);

current.setPedirCodigodireccion(ejbFacade.consultarDireccion("5"));

//current.setCoddepart(codDepartamento);

if(current.getPeempCedula() == null){

current.setPeempCedula("NULL");

}

PeparParroqPK p = new PeparParroqPK(codPais, codProvin, codCanton, codParroquia);

PeparParroq peparParroq = ejbparroquia.find(p);

current.getPedirCodigodireccion().setPeparParroq(peparParroq);

//java.util.Date fecha = new java.util.Date();

getFacade().create(current);

String codigoUser = registrarUsuario(current);

// perfilUser= new XeuxpUsupe(codigoUser, rol.getXeperCodigo(), "1");

// ejbPerfilUser.create(perfilUser);

//JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("PeempEmpleCreated"));

return prepareCreate();

} catch (Exception e) {

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

return null;

}

}

public String createInvitado() {

try {

current.setPeempCurp(ejbFacade.crearCodigo());

subirArchivo();

current.setPeempFoto(aux);

current.setPedirCodigodireccion(ejbFacade.consultarDireccion("5"));

//current.setCoddepart(codDepartamento);

if(current.getPeempCedula() == null){

current.setPeempCedula("NULL");

}

PeparParroqPK p = new PeparParroqPK(codPais, codProvin, codCanton, codParroquia);

PeparParroq peparParroq = ejbparroquia.find(p);

current.getPedirCodigodireccion().setPeparParroq(peparParroq);

//java.util.Date fecha = new java.util.Date();

getFacade().create(current);

String codigoUser = registrarUsuario(current);

// perfilUser= new XeuxpUsupe(codigoUser, rol.getXeperCodigo(), "1");

// ejbPerfilUser.create(perfilUser);

//JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("PeempEmpleCreated"));

return prepareCreateInvitado();

} catch (Exception e) {

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

return null;

}

}

public void subirArchivo() {

try {

aux = "resources/imgUsers";

String dir = JsfUtil.getPath();

String newDir = dir.replace("\\", "/");

File archivo = new File(newDir + aux);

if (!archivo.exists()) {

archivo.mkdirs();

System.out.println("Direccion inexistente");

} else {

System.out.println("Direccion válida");

}

System.out.println(getFoto().getInputStream());

copiarArchivo(getFoto().getFileName(), getFoto().getInputStream(), newDir);

//return true;

} catch (Exception e) {

//return false;

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString(" Error en a imagen"));

}

}

public void copiarArchivo(String nombreArchivo, InputStream i, String dir) {

try {

aux = aux + "/" + nombreArchivo;

System.out.println("Ruta Real: " + dir + aux);

OutputStream out = new FileOutputStream(new File(dir + aux));

int read = 0;

byte[] bytes = new byte[1024];

while ((read = i.read(bytes)) != -1) {

out.write(bytes, 0, read);

}

aux = aux.substring(19);

i.close();

out.flush();

out.close();

} catch (Exception e) {

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("Error"));

}

}

public String registrarUsuario(PeempEmple empleado) {

long miliseconds = System.currentTimeMillis();

Date fechaAct = new Date(miliseconds);

Correo c = new Correo();

String passsword = empleado.generarContraseña();

String encript = DigestUtils.md5Hex(passsword);

System.out.println("password:" + passsword);

c.enviarCorreo("Generación de Contraseña", "Su contraseña es: " + passsword + "\n\nInicie sesión de con esta contraseña para poder cambiarla.", empleado.getPeempEmail());

int codigo = this.ejbUsuario.crearCodigo();

Formatter fmt = new Formatter();

String codigoUser = codigo + "";

usuario = new XeusuUsuar(codigoUser, encript, fechaAct, fechaAct, "0");

usuario.setPeempEmple(empleado);

xeestCodigo = new XeestEstad("1", passsword);

usuario.setXeestCodigo(xeestCodigo);

ejbUsuario.create(usuario);

return codigoUser;

}

public void verificarSesion() {

try {

FacesContext context = FacesContext.getCurrentInstance();

XeusuUsuar u = (XeusuUsuar) FacesContext.getCurrentInstance().getExternalContext().getSessionMap().get("usuario");

if (u == null) {

context.getExternalContext().redirect("index.xhtml");

} else {

current = u.getPeempEmple();

}

} catch (Exception e) {

}

}

public void limpiar(ComponentSystemEvent event) {

current = null;

}

public String update() {

try {

if(getFoto() == null){

current.setPeempFoto(current.getPeempFoto());

}

else {

System.out.println("entra foto");

subirArchivo();

current.setPeempFoto(aux);

}

getFacade().edit(current);

foto=null;

recreatePagination();

recreateModel();

addSuccessMessage("Completado","Usuario Editado correctamente");

return "List";

} catch (Exception e) {

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

return null;

}

}

public String updateEmp() {

try {

if(getFoto() == null){

current.setPeempFoto(current.getPeempFoto());

}

else {

System.out.println("entra foto");

subirArchivo();

current.setPeempFoto(aux);

}

getFacade().edit(current);

foto=null;

recreatePagination();

recreateModel();

addSuccessMessage("Completado","Usuario Editado correctamente");

return "";

} catch (Exception e) {

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

return null;

}

}

public String destroy() {

current = (PeempEmple) getItems().getRowData();

selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();

performDestroy();

recreatePagination();

recreateModel();

return "List";

}

public String destroyAndView() {

performDestroy();

recreateModel();

updateCurrentItem();

if (selectedItemIndex >= 0) {

return "View";

} else {

// all items were removed - go back to list

recreateModel();

return "List";

}

}

private void performDestroy() {

try {

//usuario.setPeempEmple(current);

//ejbUsuario.remove(usuario);

getFacade().remove(current);

JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("PeempEmpleDeleted"));

} catch (Exception e) {

JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

}

}

private void updateCurrentItem() {

int count = getFacade().count();

if (selectedItemIndex >= count) {

// selected index cannot be bigger than number of items:

selectedItemIndex = count - 1;

// go to previous page if last page disappeared:

if (pagination.getPageFirstItem() >= count) {

pagination.previousPage();

}

}

if (selectedItemIndex >= 0) {

current = getFacade().findRange(new int[]{selectedItemIndex, selectedItemIndex + 1}).get(0);

}

}

public void inicializarEmpleado() {

current=null;

}

public String prepareView() {

current = (PeempEmple) getItems().getRowData();

selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();

return "View";

}

public String prepareEdit() {

flag = 2;

current = (PeempEmple) getItems().getRowData();

selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();

return "Edit";

}

public DataModel getItems() {

if (items == null) {

items = getPagination().createPageDataModel();

}

return items;

}

public PaginationHelper getPagination() {

if (pagination == null) {

pagination = new PaginationHelper(10) {

@Override

public int getItemsCount() {

return getFacade().count();

}

@Override

public DataModel createPageDataModel() {

return new ListDataModel(getFacade().findRange(new int[]{getPageFirstItem(), getPageFirstItem() + getPageSize()}));

}

};

}

return pagination;

}

public String getInLimiteAvaliacao() {

return inLimiteAvaliacao;

}

public void setInLimiteAvaliacao(String inLimiteAvaliacao) {

this.inLimiteAvaliacao = inLimiteAvaliacao;

}

private void persist(PersistAction persistAction, String successMessage) {

if (selected != null) {

setEmbeddableKeys();

try {

if (persistAction != PersistAction.DELETE) {

getFacade().edit(selected);

} else {

getFacade().remove(selected);

}

JsfUtil.addSuccessMessage(successMessage);

} catch (EJBException ex) {

String msg = "";

Throwable cause = ex.getCause();

if (cause != null) {

msg = cause.getLocalizedMessage();

}

if (msg.length() > 0) {

JsfUtil.addErrorMessage(msg);

} else {

JsfUtil.addErrorMessage(ex, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

}

} catch (Exception ex) {

Logger.getLogger(this.getClass().getName()).log(Level.SEVERE, null, ex);

JsfUtil.addErrorMessage(ex, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));

}

}

}

public PeempEmple getPeempEmple(java.lang.String id) {

return getFacade().find(id);

}

public List<PeempEmple> getItemsAvailableSelectMany() {

return getFacade().findAll();

}

public List<PeempEmple> getItemsAvailableSelectOne() {

return getFacade().findAll();

}

@FacesConverter(forClass = PeempEmple.class)

public static class PeempEmpleControllerConverter implements Converter {

@Override

public Object getAsObject(FacesContext facesContext, UIComponent component, String value) {

if (value == null || value.length() == 0) {

return null;

}

PeempEmpleController controller = (PeempEmpleController) facesContext.getApplication().getELResolver().

getValue(facesContext.getELContext(), null, "peempEmpleController");

return controller.getPeempEmple(getKey(value));

}

java.lang.String getKey(String value) {

java.lang.String key;

key = value;

return key;

}

String getStringKey(java.lang.String value) {

StringBuilder sb = new StringBuilder();

sb.append(value);

return sb.toString();

}

@Override

public String getAsString(FacesContext facesContext, UIComponent component, Object object) {

if (object == null) {

return null;

}

if (object instanceof PeempEmple) {

PeempEmple o = (PeempEmple) object;

return getStringKey(o.getPeempCurp());

} else {

Logger.getLogger(this.getClass().getName()).log(Level.SEVERE, "object {0} is of type {1}; expected type: {2}", new Object[]{object, object.getClass().getName(), PeempEmple.class.getName()});

return null;

}

}

}

}

# 5. EJECUCIÓN DEL PROYECTO

Verifique en su compilador de Netbeans en la barra de herramientas el icono en forma triangular de color verde, será necesario presionarlo compilar el proyecto.

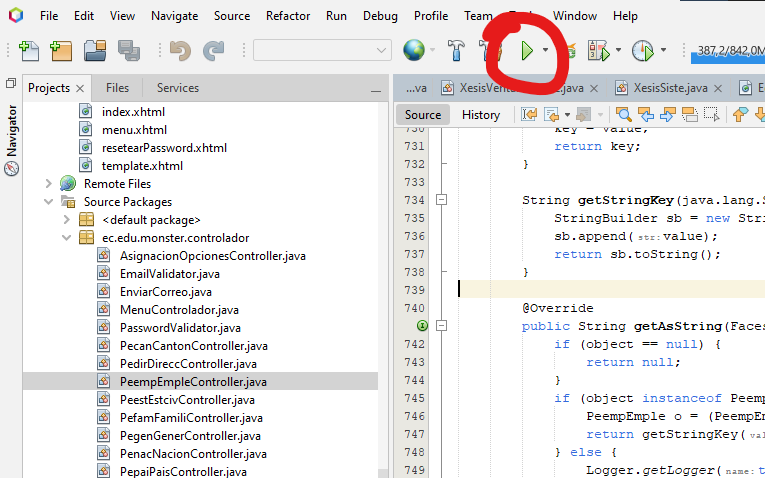


Figura 17. Botón para ejecutar el proyecto.

Al compilar el proyecto deberá obtener el siguiente resultado en pantalla.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Figura 18. Ejecución del proyecto.

Al ingresar con las credenciales y al dirigirse al apartado de CRUDS > Personal > Empleados se visualizará la siguiente pantalla.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 19. Crud de personal.

Al presionar en el botón verde se podrá visualizar el primer marco de página ubicado en la sección de Ver Usuario.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 20. Marco de pantalla de información de usuario.

Al presionar en el ícono azul de editar se visualizará el formulario con otro marco de página.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 21. Marco de página de edición de usuario.

Finalmente al presionar el botón “Crear” se desplegará el último marco de página creado para Personal / Usuarios.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 22. Marco de página de creación de usuario.

# 6. CONCLUSIONES

* Los marcos de página son una herramienta útil para organizar y presentar información en una aplicación web. En particular, el uso del elemento <p:tabview> de PrimeFaces permite presentar información en pestañas, lo que puede mejorar la navegación y la usabilidad.
* Es importante tener en cuenta que el uso excesivo de marcos de página puede hacer que la aplicación web sea menos accesible y más difícil de mantener. Por lo tanto, se debe utilizar esta herramienta de manera selectiva y solo cuando sea necesario para mejorar la experiencia del usuario.
* Al utilizar marcos de página en una aplicación web, es importante asegurarse de que la estructura de la página siga siendo clara y coherente. Los elementos de navegación deben ser fáciles de encontrar y utilizar, y la información debe estar organizada de manera lógica y coherente.

# 7. RECOMENDACIONES

* Utilizar el elemento <p:tabview> de PrimeFaces para organizar y presentar información en pestañas solo cuando sea necesario y en áreas específicas de la aplicación web. Considerar otras opciones de diseño, como una barra de navegación horizontal o vertical, para presentar información en otras áreas de la aplicación.
* Asegúrarse de que la información presentada en cada pestaña de <p:tabview> sea clara y esté organizada de manera coherente. Utilizar títulos y etiquetas descriptivas para ayudar a los usuarios a comprender el contenido de cada pestaña y evitar la confusión.
* Probar la aplicación web con diferentes tamaños de pantalla y dispositivos para asegurarse de que el diseño y la estructura de la página se mantengan claros y coherentes. Considerar utilizar técnicas de diseño responsivo para garantizar que la aplicación web sea accesible y fácil de usar en una amplia gama de dispositivos y tamaños de pantalla.



# BIBLIOGRAFÍA

|  |  |
| --- | --- |
| [1] | «Integración con servicios web, Java y bases de datos,» IBM, 03 04 2021. [En línea]. Available: https://www.ibm.com/docs/es/bpm/8.5.7?topic=deprecated-integrating-web-services-java-databases. [Último acceso: 23 01 2023]. |
| [2] | «MySQL Connectors,» MySQL, [En línea]. Available: https://www.mysql.com/products/connector/. [Último acceso: 03 01 2023]. |
| [3] | «Tratamiento de los datos entre las capas,» Junta de Andalucía, [En línea]. Available: https://tinyurl.com/2u8abays. [Último acceso: 23 01 2023]. |
| [4] | «CAPÍTULO 18 INTERFACES,» arkaitzgarro, [En línea]. Available: https://www.arkaitzgarro.com/java/capitulo-18.html. [Último acceso: 23 01 2023]. |
| [5] | «Qué es MySQL: Características y ventajas,» openwebinars, [En línea]. Available: https://openwebinars.net/blog/que-es-mysql/. [Último acceso: 23 01 2023]. |
| [6] | «Cómo crear una Base de Datos con phpMyAdmin en MySQL,» disenowebakus, [En línea]. Available: https://tinyurl.com/mvys95pb. [Último acceso: 23 01 2023]. |
| [7] | AMVO, «Marcos de aplicaciones web | AMVO,» AMVO, 27 09 2020. [En línea]. Available: https://www.amvo.org.mx/glosario/marcos-de-aplicaciones-web/. [Último acceso: 05 03 2023]. |
| [8] | PrimeNG , «Container component to group content,» PrimeNG , [En línea]. Available: https://primeng.org/tabview. [Último acceso: 05 03 2023]. |
| [9] | JavaPoint, «PrimeFaces Layout,» JavaPoint, [En línea]. Available: https://www.javatpoint.com/primefaces-layout. [Último acceso: 05 03 2023]. |
| [10] | PrimeTek, «Layout - Element,» PrimeTek , [En línea]. Available: https://www.primefaces.org/showcase-v8/ui/panel/layout/element.xhtml. [Último acceso: 05 03 2023]. |
| [11] | PrimeTek, «TabView,» PrimeTek, [En línea]. Available: http://www.primefaces.org:8080/showcase/ui/panel/tabView.xhtml?jfwid=0cc74. [Último acceso: 05 03 2023]. |