FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN

ESCUELA DE INGENIERIA DE SISTEMAS Y TELEMÁTICA

Manual para la creación de una aplicación web con el uso de la librería primefaces.

Trabajo de graduación previo a la obtención del título de Ingeniero de Sistemas y Telemática

Autor:

Braulio Machuca Rubio.

Director:

Ing. Gabriel León Paredes, Mst.

Cuenca, Ecuador

2015

Manual para la creación de una aplicación web con el uso de la librería primefaces.

DEDICATORIA:

Este trabajo de graduación dedico a mis padres, ya que ellos siempre han estado presentes para apoyarme moral y psicológicamente, a mis hermanas, pues ellas han sido el principal cimiento para la construcción de mi vida personal y deseos de superación.

También se la dedico a mis sobrinas, quienes han sido mi mayor motivación para jamás rendirme y llegar a ser así un gran ejemplo para ellas.

Braulio Machuca.

AGRADECIMINETOS:

Agradezco principalmente a Dios por haberme acompañado y guiado a lo largo de mi carrera y por haberme demostrado que con el todo es posible.

A la Universidad del Azuay por darme la oportunidad de estudiar y realizarme como un profesional, también a todos mis profesores y compañeros, ya que de ellos he recibido tanto conocimiento intelectual y humano que me han ayudado a crecer como persona y profesional.

A mi director, Ing. Gabriel León, quien gracias a sus conocimientos, experiencia, paciencia y motivación he logrado terminar mis estudios con éxito.

INDICE DE CONTENIDOS

Índice de Ilustraciones

Índice de Tablas.

INDICE DE ANEXOS

RESUMEN

En el presente trabajo de graduación se desarrollará un manual para la creación de una aplicación web con el uso de la librería Primefaces, proporcionando así una referencia útil para todos los interesados, tales sean estos profesionales, estudiantes, docentes, programadores o personas interesadas que tengan algún conocimiento básico en desarrollo web.

Esta monografía abarca temas como: instalación de los componentes necesarios, creación del proyecto, desarrollo de interfaces graficas de usuario, conexión de interfaz gráfica con el código fuente correspondiente. Cabe mencionar que creará una aplicación web base, demostrando así la importancia de dicho manual

ABSTRACT

The purpose of this graduation paper is to develop a manual for the creation of a web application with the use of Java PrimeFaces library; and therefore provide a useful reference for professionals, students, teachers, programmers or people interested in this application and who have some basic knowledge in web development.

This monograph covers topics such as installation of the necessary components, project creation, development of graphical user interfaces, and graphics user interface connection with the corresponding source code. It should be mentioned that a web based application will be created, demonstrating the importance of this manual.

UNIVERSIDAD DEL
AZUAY
DPTO. IDIOMAS

Translated by,
Lic. Lourdes Crespo

Introducción

En la actualidad existen diferentes plataformas al momento de inclinarnos por el desarrollo web, algunas de estas tienen costo instalación, otras tienen un costo para el uso de diferentes componentes, etc. En nuestro medio no existe la información adecuada para el uso de estas aplicaciones de código abierto gratuita en el campo de desarrollo web, debido a la falta de explotación de información en estos sistemas, crearé un manual de principio a fin y paso a paso para la creación de una aplicación web de código abierto utilizando la librería Primefaces de java, basándome en el desarrollo de un sistema de facturación web.

Cuando se trabaja con código abierto se puede añadir componentes o librerías externas, pero esto no es del todo eficiente, ya que al no ser compatibles puede ocurrir problemas y el sistema puede tornarse lento y pesado.

Al momento de desarrollar un aplicativo web, varios son los factores que nos limitan a tomar una decisión para optar la mejor opción, uno de estos factores son los costos al desarrollar nuestra aplicación, por otro lado también es la falta de componentes necesario para la realización de dicho sistema, y por ultimo pero no menos importante es la escases de documentación para la elaboración de un sistema paso a paso.

Es por esto que pretendo crear un manual para la creación de una aplicación web con la librería primefaces de java, ya que por ser código abierto, por sus números componentes y fácil uso, es un buen candidato para inclinarse el momento de decidirnos por el desarrollo de un sistema web.

CAPÍTULO 1

Teoría y principios básicos.

1. Java

Java es un lenguaje de programación de código abierto, orientado a objetos y una plataforma informática sacada al mercado por primera vez en los años 90 por Sun Microsystems, la cual posteriormente fue adquirida por la compañía Oracle. (Java, 2014). Java toma la estructura de sintaxis de C y C++ incorporando un modelo de objetos más simples y eliminando las herramientas de bajo nivel.

La Programación Orientada a Objetos consiente en desarrollar aplicaciones de forma más simple dividiendo el problema en objetos, así nos podemos centrar en cada objeto, de esa manera hace que nuestro código quede más simple y sea más fácil de entender, principalmente en aplicaciones muy extensas. (Coronel, 2010).

1.1 Características

Simple: Java minimiza los errores más comunes de programación a un 50% con respecto a otros lenguajes de programación.

Distribuido: Este lenguaje suministra librerías y herramientas para que los programas sean distribuidos, es decir, para que se ejecuten en varias máquinas e interactuando entre ellas.

Multiplataforma: Esto quiere decir que las aplicaciones Java pueden ser ejecutadas donde sea, siendo así que un programa puede ser desarrollado en una plataforma y puede ser desarrollado en cualquier otra plataforma.

Robusto: Java está buscando constantemente problemas, tanto en tiempo de ejecución como en tiempo de compilación, estas ayudan a detectar errores en ciclos de desarrollo, también es el que se encarga de la liberación de memoria.

Seguro: Las aplicaciones Java son seguras ya que no acceden a partes de memoria o de sistema, que están propensas a infectarse con ciertos virus.

Multihilo: Java permite varias tareas paralelamente en una aplicación, gracias a los hilos, que son procesos o piezas independientes dentro de un gran proceso; al ser éstos hilos

construidos en el mismo lenguaje, son más fáciles de usar y más robustos respecto a otros lenguajes. Con esto, se tiene mejor rendimiento interactivo y mejor comportamiento en tiempo real. (Coronel, 2010)

2. Persistencia en java.

Al momento de desarrollar una aplicación web, uno de los primeros requerimientos más importantes que debemos resolver es la integración con una base de datos para poder realizar de la manera más óptima la gestión de la información.

La persistencia de los objetos es la capacidad para guardar y recuperar la información desde una base de datos, sin importancia cual sea esta, ya que se puede conectar con la mayoría de los gestores de bases de datos. La persistencia en Base de Datos relacionales se suele implementar mediante el desarrollo de funcionalidad específica utilizando la tecnología JDBC o mediante frameworks que automatizan el proceso a partir de mapeos.

3. JavaServer Faces (JSF)

JavaServer Faces es un estándar de trabajo desarrollado por java orientada a componentes de interfaz de usuario para aplicaciones web de Java, es decir, es un framework para desarrollo web basado en el lenguaje de programación java.

Este estándar simplifica la construcción de interfaces de las aplicaciones web, la tecnología JSF, fue diseñado para ser flexible, esta aprovecha los estándar y conceptos web existentes, así una de las grandes ventajas de JavaServer Faces, es que nos permite tener un intérprete personalizado y una biblioteca propia de etiquetas JSP, una página jsf posee una extensión *.xhtml, es decir una unión de HTML y XML.

Una de las principales ventajas de JSF es su facilidad de uso, ya que tiene una arquitectura claramente separada entre la lógica de aplicación y presentación. Este diseño permite que cada integrante del desarrollo de la aplicación trabaje individualmente en su módulo, ya que esta tecnología proporciona una herramienta cómoda para la unión de cada uno de sus partes. (Fernando Pech-May, 2013)

4. Primefaces

PrimeFaces es una librería JavaScript de código abierto de componentes visuales, para su uso debemos únicamente adjuntar esta librería al proyecto y utilizar los componentes necesarios que nos ofrece esta librería, ya que por su facilidad de uso, hará que el trabajo del programador sea fácil y sencillo.

Primefaces es una librería sumamente ligera, a pesar de su gran número de componentes que nos permite utilizar, también posee una gran estabilidad ya que no depende de la funcionalidad de ningunas otras librerías, es decir, es totalmente independiente.

La principal característica de primefaces, es que los diseños de sus componentes son realmente estables, y la complejidad para usarlos es insignificante, dando así un trabajo de calidad y sin mayor dificultad para el programador. (Mert Caliskan, 2013)

5. Arquitectura Modelo – Vista – Controlador (MVC).

La arquitectura MVC surge con el objetivo de reducir e independizar el trabajo de programadores y diseñadores en la implementación de sistemas múltiples, esta arquitectura divide las partes que conforman una aplicación en el Modelo, las vistas y los controladores, permitiendo así la implementación por separado de cada una de sus partes, garantizando así el mantenimiento del software de una manera más sencilla y rápida e independiente, de esta manera se puede lograr un trabajo más eficaz y mejor organizado en el momento de la implementación.

5.1 Definición de sus partes.

Modelo: Representa la información con la que el sistema maneja, por lo tanto se encarga de todos los accesos a dicha información, ya sea consultas o actualizaciones. El Modelo manda a la vista la información que se le solicita para que sea mostrada en la interfaz al usuario. Las peticiones de acceso o manipulación de información recibe el modelo por medio del controlador.

Vista: Maneja la representación visual de los datos, mostrando la información en un formato adecuado para interactuar con el usuario, interactúa preferentemente con el

controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.

Controlador: Responde a las acciones del usuario que invoca peticiones al modelo cuando se hace realice una solicitud sobre la información, ya sea actualizaciones, listados, etc. También puede interactuar con la vista asociada si se solicita un cambio en la forma en que se presenta de modelo, por tanto se podría decir que el controlador hace de intermediario entre la vista y el modelo. (Yenisleidy Fernández Romero, 2012)

6. Base de datos MySQL

MySQL es un gestor de base de datos muy conocido y ampliamente usado en el mundo de la informática ya que por su simplicidad y alto rendimiento, hace que los usuarios de bases de datos se inclinen por este gran gestor.

MySQL es una opción tentadora tanto para el desarrollo de aplicaciones comerciales, o desarrollo con fines educativos, ya que posee un tiempo de respuesta reducido y por su facilidad de uso, además su libre distribución en Internet le proporciona beneficios adicionales, además de contar con un alto grado de estabilidad, un rápido desarrollo,

MySQL está disponible para múltiples plataformas, en nuestro caso utilizaremos MySQL para Windows 7. Sin embargo, las diferencias con cualquier otra plataforma son prácticamente nulas, ya que la herramienta utilizada en este caso es MySQL WorkBeanch versión 5.2, que permite interactuar con un servidor MySQL local , De este modo es posible realizar este ejemplo sobre un servidor instalado localmente o, a través de

Internet, sobre un servidor remoto. (Luis Alberto Casillas Santillán, 2012)

CAPÍTULO 2

Análisis

2.1 Diagramas.

2.1.1Modelo Entidad Relación.

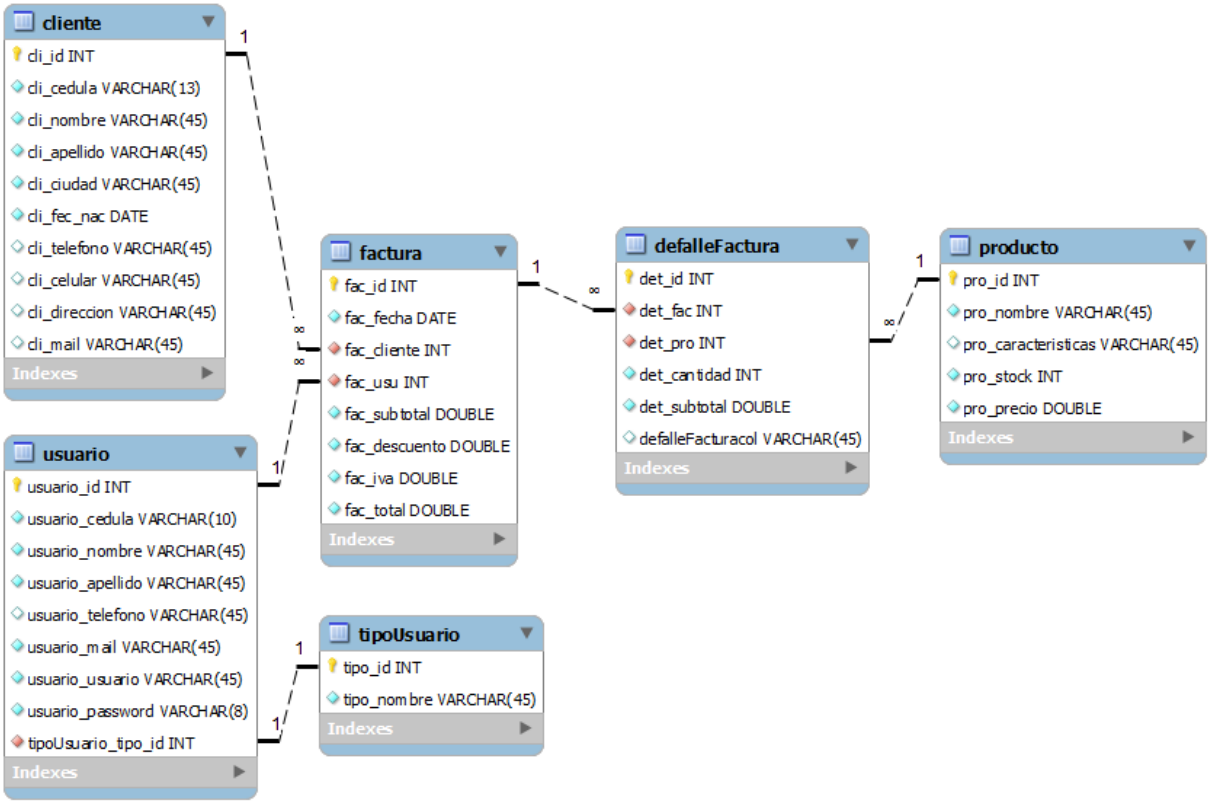


Ilustración 1 Modelo entidad relación

En el modelo entidad relación podremos observar cada una de las tablas y atributos que poseerá nuestra base datos, así también veremos la relaciones que tendrán entre las entidades que utilizaremos en nuestro sistema web.

2.1.2 Casos de uso.

2.1.2.1 Caso de uso Gestión Cliente.

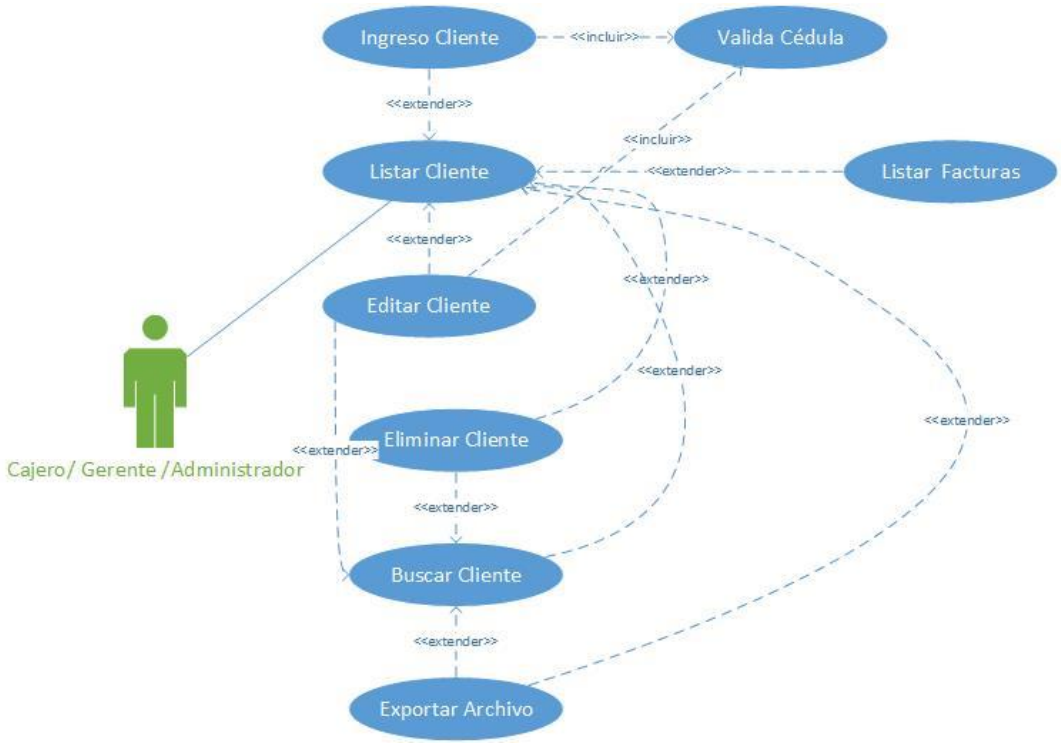


Ilustración 2 Caso de uso Gestión Cliente

Listar Cliente

Caso Uso	**Listar Cliente**
Actores	Cajero, gerente, administrador
Propósito	Permitir ver todos los clientes registrados en el sistema.
Resumen	Los usuarios, ya sean estos cajeros, administradores o gerentes podrán ver los datos de todos los cliente.
Tipo	Primario
PreCondiciones	Los usuarios registrados en el sistema, ya sean cajeros, administradores o gerentes, deben ingresar al módulo clientes de esta

	manera se les visualizara todos los clientes almacenados.
PostCondiciones	

Tabla 1 Caso de Uso Listar Cliente

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.

Tabla 2 Acción de los Actores Listar Cliente

Ingreso Cliente

Caso Uso	**Ingreso Cliente**
Actores	Cajero, gerente, administrador.
Propósito	Permitir registrar en la base datos todos los datos del cliente.
Resumen	Todos Los usuarios podrán ingresar los datos del cliente.
Tipo	Extensión.
PreCondiciones	Los usuarios registrados en el sistema, ya sean cajeros, administradores o gerentes, deben ingresar al módulo Clientes, después dar click en el botón crear.
PostCondiciones	Verificación datos del cliente.

Tabla 3 Caso de Uso Ingreso Cliente

Acción de los Actores	**Respuesta del Sistema**
Este caso de uso comienza cuando el usuario ha ingresado al módulo cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.
El usuario selecciona el botón crear o puede hacer click derecho sobre cualquier fila de la lista y presionar crear.	El sistema mostrará el formulario de ingreso del cliente.

El usuario almacena los datos haciendo click en el botón “Guardar”	El sistema mostrará el listado actualizado de los clientes ingresados anteriormente.

Tabla 4 Acción de los Actores Ingreso Cliente

Listar Factura

Caso Uso	Listar Facturas
Actores	Cajero, gerente, Administrador
Propósito	Permitir ver todas las facturas de cierto cliente registrado en el sistema.
Resumen	Los usuarios, ya sean estos cajeros, administradores o gerentes podrán ver las facturas de los clientes.
Tipo	Extensión.
PreCondiciones	Los usuarios registrados en el sistema, ya sean cajeros, administradores o gerentes, deben ingresar al módulo Clientes, una vez listados todos los clientes, podrán elegir un cliente y listar sus facturas.
PostCondiciones	

Tabla 5 Caso de uso Listar Factura

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.
Se desplegara un menú al hacer click derecho sobre un cliente, y seleccionaremos la opción que dice “Ver facturas”	El sistema responderá mostrando una lista con todas las facturas de ese cliente.

Tabla 6 Accion de los Actores Listar Factura

Modificar Datos Cliente

Caso Uso	**Modificar Datos Cliente**
Actores	Cajero, Gerente, Administrador

Propósito	Modificar algún dato del cliente que desee el actor.
Resumen	Los actores podrán cambiar cualquier valor del cliente que deseen, con excepción del id.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar logueados y el ingreso del cliente debería realizarse anteriormente.
PostCondiciones	Valida los datos del cliente.

Tabla 7 Caso de uso Modificar Datos Cliente

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.
El usuario selecciona el cliente que desea modificar, haciendo click derecho sobre el registro y luego click en modificar.	El sistema mostrará el formulario del cliente para que modifique los datos del mismo.
El usuario hace las modificaciones pertinentes en el formulario de cliente.	
El usuario hace click en la opción “Guardar”	El sistema nos muestra un cuadro de dialogo que dice “Desea aplicar los cambios.”
El usuario hace click en el botón “Si”.	El sistema guarda los cambios efectuados por el usuario. El sistema muestra una notificación que dice: “Cliente se ha actualizado correctamente”. Muestra una lista actualizada de todos los clientes.

Tabla 8 Acción de los Actores Modificar Datos Cliente

Eliminar Cliente

Caso Uso	**Eliminar Cliente**
Actores	Cajero, Gerente, Administrador
Propósito	Eliminar datos de un cliente
Resumen	Los actores podrán eliminar a cualquier cliente de la base de datos.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar logueados y el ingreso del cliente a eliminar debería realizarse anteriormente.
PostCondiciones	

Tabla 9 Caso de uso Eliminar Cliente

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.
El usuario selecciona el cliente que desea eliminar, haciendo click derecho sobre el registro y luego click en borrar	El sistema nos muestra un cuadro de dialogo que dice "Seguro que desea continuar?"
El usuario hace click en el botón "Si".	El sistema elimina el cliente. El sistema muestra una notificación de: "Cliente se ha borrado correctamente". Muestra una lista actualizada de todos los clientes.

Tabla 10 Acción de los actores Eliminar Cliente

Buscar Cliente

Caso Uso	**Buscar Cliente**
Actores	Cajero, Administrador, Gerente
Propósito	Buscará al cliente para consultar alguna información que desee el actor.

Resumen	Los actores podrán visualizar cualquier información del cliente que deseen.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar logueados y se deberá tener registrados algunos clientes anteriormente.
PostCondiciones	

Tabla 11 Caso de uso Buscar Cliente

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.
El usuario podrá buscar al cliente por su cedula, nombres, apellidos y ciudad.	El sistema filtrara los datos del cliente.

Tabla 12 Accion de los actores Buscar Cliente

Exportar archivo

Caso Uso	**Exportar Archivo**
Actores	Cajero, Administrador, Gerente
Propósito	Exportar la lista de los clientes.
Resumen	Los actores podrán exportar la lista de los clientes ya sea en los formatos Excel, PDF, CSV y XML.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar registrados, y al menos se debería haber realizado un ingreso del cliente anteriormente.
PostCondiciones	

Tabla 13 Caso de uso Exportar Archivo

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Cliente.	El sistema muestra una lista de todos los clientes registrados anteriormente.

El usuario podrá exportar la lista de los clientes registrados.	El sistema descargara un archivo dependiendo del formato seleccionado por el actor.

Tabla 14Accion de los actores Exportar Archivo

2.1.2.2 Caso de uso Gestión Producto.

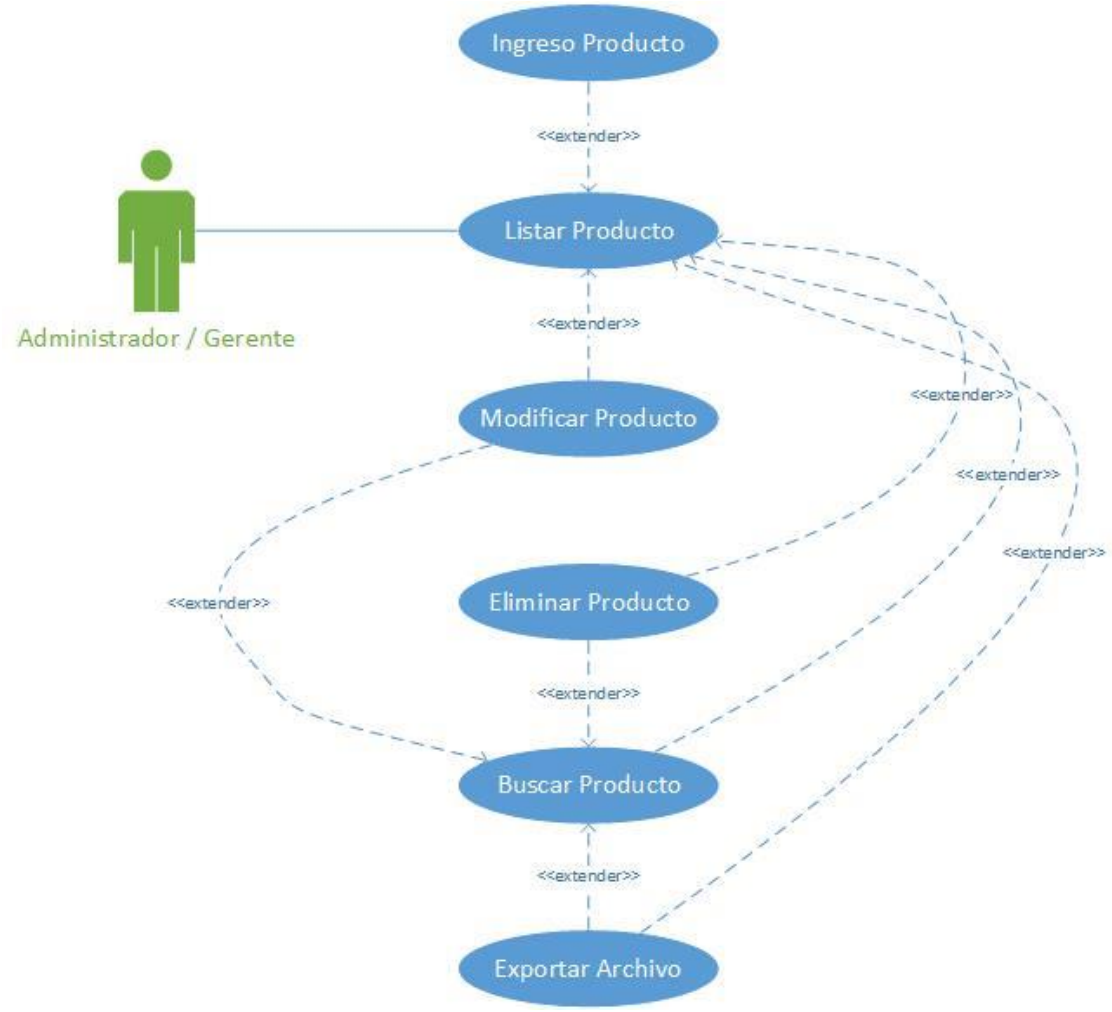


Ilustración 3 Caso de uso Gestión Producto

Listar Producto

Caso Uso	**Listar Producto**
Actores	Gerente, Administrador
Propósito	Permitir ver todos los productos registrados en el sistema.

Resumen	Los usuarios, ya sean estos, administradores o gerentes podrán ver los datos de todos los producto
Tipo	Primario.
PreCondiciones	Los usuarios registrados en el sistema, ya sean administradores o gerentes, deben ingresar al módulo productos y ahí se les visualizara todos los productos almacenados.
PostCondiciones	

Tabla 15 Caso de uso Listar Producto

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo producto.	El sistema muestra una lista de todos los productos registrados anteriormente.

Tabla 16 Acción de los actores Listar Producto

Ingreso Producto

Caso Uso	**Ingreso Producto**
Actores	Gerente, Administrador
Propósito	Permitir registrar en la base datos todos los datos del producto
Resumen	Los usuarios ya sean estos administradores o gerentes, podrán ingresar los datos del producto.
Tipo	Extensión.
PreCondiciones	Los usuarios registrados en el sistema, ya sean administradores o gerentes, deben ingresar al módulo productos, después dar click en el botón crear.
PostCondiciones	

Tabla 17 Caso de uso Ingreso Producto

Acción de los Actores	Respuesta del Sistema

Este caso de uso comienza cuando el usuario ha ingresado al módulo producto.	El sistema muestra una lista de todos los productos registrados anteriormente.
El usuario selecciona el botón crear o puede hacer click derecho sobre cualquier fila de la lista y presionar crear.	El sistema mostrará el formulario de ingreso del producto.
El usuario almacena los datos haciendo click en el botón "Guardar"	El sistema mostrará el listado actualizado de los productos ingresados anteriormente.

Tabla 18 Acción de los Actores Ingreso Producto

Modificar Datos Producto

Caso Uso	**Modificar Datos Producto**
Actores	Gerente, Administrador
Propósito	Modificar algún dato del producto que desee el actor.
Resumen	Los actores podrán cambiar cualquier valor del cliente que deseen, con excepción del id.
Tipo	Extensión
PreCondiciones	Los actores deberán estar logueados y el ingreso del producto debería realizarse anteriormente.
PostCondiciones	

Tabla 19 Caso de uso Modificar Datos Producto

Acción de los Actores	**Respuesta del Sistema**
Este caso de uso comienza cuando el usuario ha ingresado al módulo Producto.	El sistema muestra una lista de todos los productos registrados anteriormente.
El usuario selecciona el producto que desea modificar, haciendo click derecho sobre el registro y luego click en modificar.	El sistema mostrará el formulario del producto para que modifique los datos del mismo.
El usuario hace las modificaciones pertinentes el formulario de producto.	

El usuario hace click en la opción "Guardar"	El sistema nos muestra un cuadro de dialogo que dice "Desea aplicar los cambios?."
El usuario hace click en el botón "Si".	El sistema guarda los cambios efectuados por el usuario. El sistema muestra una notificación de: "Producto se ha actualizado correctamente". Muestra una lista actualizada de todos los Productos.

Tabla 20 Acción de los Actores Modifica Datos Producto

Eliminar Producto

Caso Uso	**Eliminar Producto**
Actores	Gerente, Administrador
Propósito	Eliminar registro de un producto.
Resumen	Los actores podrán eliminar a cualquier producto de la base de datos.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar logueados y el ingreso del producto debería realizarse anteriormente.
PostCondiciones	

Tabla 21 Caso de uso Eliminar Producto

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Producto.	El sistema muestra una lista de todos los productos registrados anteriormente.
El usuario selecciona el producto que desea eliminar, haciendo click derecho sobre el registro y luego click en borrar	El sistema nos muestra un cuadro de dialogo que dice "Seguro que desea continuar?"

El usuario hace click en el botón “Si”.	El sistema elimina el Producto. El sistema muestra una notificación de: “Producto se ha borrado correctamente”. Muestra una lista actualizada de todos los Productos.

Tabla 22 Acción de los actores Eliminar Producto

Buscar Producto

Caso Uso	**Buscar Producto**
Actores	Administrador, Gerente
Propósito	Buscará al producto para consultar alguna información que desee el actor.
Resumen	Los actores podrán visualizar cualquier información del producto que deseen.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar registrados, y al menos se debería haber realizado un ingreso del producto anteriormente.
PostCondiciones	

Tabla 23 Caso de uso Buscar Producto

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Producto.	El sistema muestra una lista de todos los productos registrados anteriormente.
El usuario podrá buscar al producto por su id, nombres, características, stock.	El sistema filtrara los datos del producto.

Tabla 24 Acción de los actores Buscar Producto

Exportar archivo

Caso Uso	**Exportar Archivo**
Actores	Administrador, Gerente
Propósito	Exportar la lista de los Productos.

Resumen	Los actores podrán exportar la lista de los productos ya sea en los formatos Excel, PDF, CSV y XML.
Tipo	Extensión.
Referencias Cruzadas	R3.1
PreCondiciones	Los actores deberán estar registrados, y al menos se debería haber realizado un ingreso del producto anteriormente.
PostCondiciones	

Tabla 25 Caso de uso Exportar Archivo

Acción de los Actores	**Respuesta del Sistema**
Este caso de uso comienza cuando el usuario ha ingresado al módulo Producto.	El sistema muestra una lista de todos los productos registrados anteriormente.
El usuario podrá exportar la lista de los clientes registrados.	El sistema descargara un archivo dependiendo del formato seleccionado por el actor.

Tabla 26 Acción de los actores Exportar Archivo

2.1.2.3 Caso de uso Gestión Usuario.

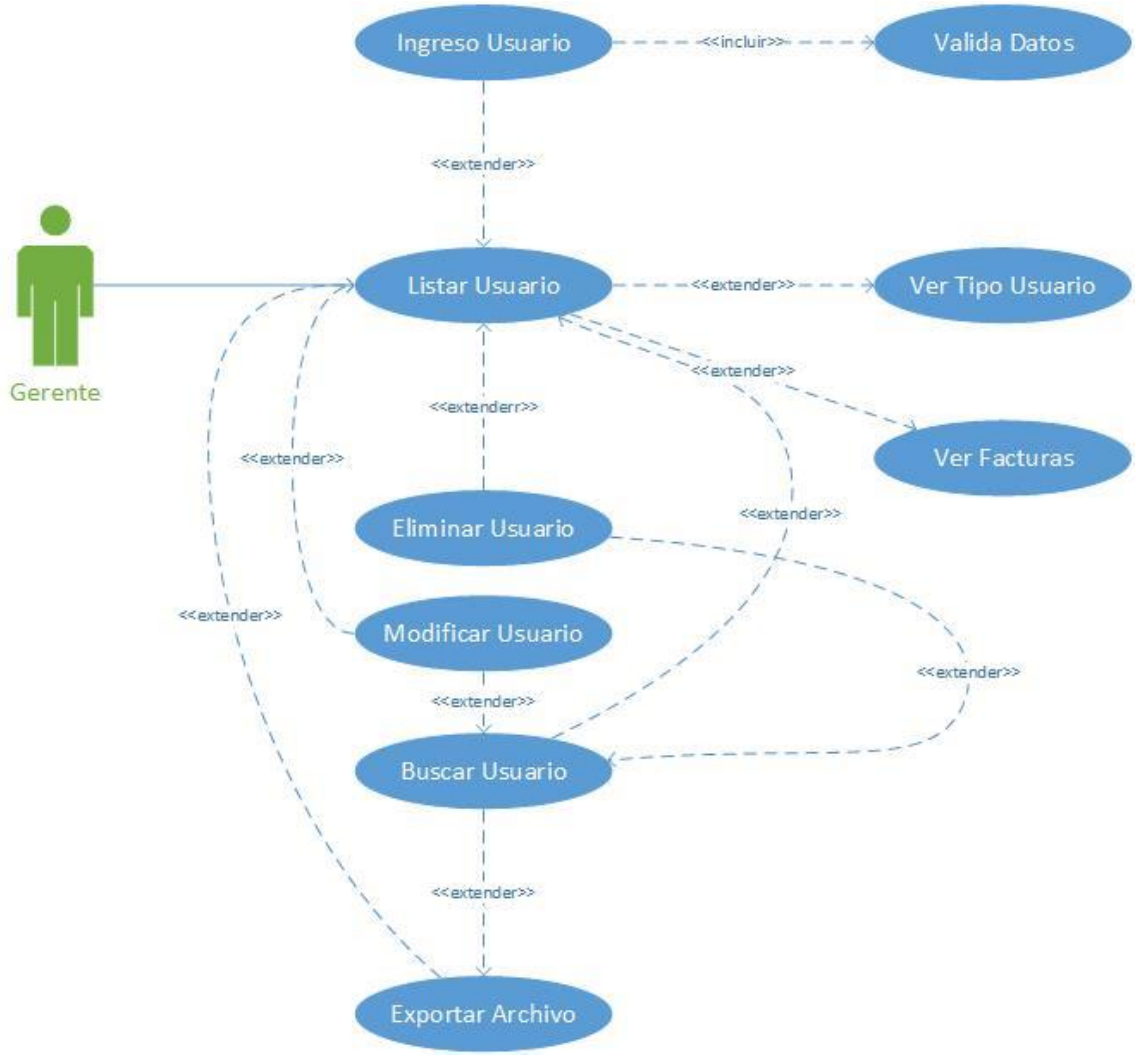


Ilustración 4 Caso de uso Gestión Usuario

Listar Usuario

Caso Uso	**Listar Usuario**
	Gerente.
Propósito	Permitir ver todos los usuarios registrados en el sistema.
Resumen	Los usuarios gerentes podrán ver los datos de todos los usuarios
Tipo	Primario

PreCondiciones	Los usuarios registrados gerentes, deben ingresar al módulo usuarios y ahí se les visualizara todos los usuarios almacenados.
PostCondiciones	

Tabla 27 Caso de uso Listar Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario gerente ha ingresado al módulo Usuario.	El sistema muestra una lista de todos los usuarios registrados anteriormente.

Tabla 28 Acción de los actores Listar Usuario

Ingreso Usuario

Caso Uso	**Ingreso Usuario**
Actores	Gerente
Propósito	Permitir registrar en la base datos todos los datos del usuario.
Resumen	El gerente podrá ingresar los datos de los usuarios.
Tipo	Extensión.
PreCondiciones	Los usuarios gerentes registrados en el sistema, Ingresaran al módulo usuarios, y después dar click en el botón crear.
PostCondiciones	Verificación datos del Usuario.

Tabla 29 Caso de uso Ingreso Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario gerente ha ingresado al módulo Usuario.	El sistema muestra una lista de todos los usuarios ingresados anteriormente.

El usuario selecciona el botón crear o puede hacer click derecho sobre cualquier fila de la lista y presionar crear.	El sistema mostrará el formulario de ingreso del Usuario.
El usuario almacena los datos haciendo click en el botón “Guardar”	El sistema nos muestra un cuadro de dialogo que dice “Desea aplicar los cambios.”
El usuario hace click en “Si”	El sistema guarda los datos del nuevo usuario. El sistema muestra una notificación de: “Usuario se ha creado correctamente”. Muestra una lista actualizada de todos los Usuario.

Tabla 30 Acción de los Actores Ingreso Usuario

Modificar Datos Usuario

Caso Uso	**Modificar Datos Usuario**
Actores	Gerente
Propósito	Modificar algún dato del usuario que desee el actor.
Resumen	Los actores podrán cambiar cualquier valor del usuario que deseen, con excepción del id.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar logueados y el ingreso del usuario debería realizarse anteriormente.
PostCondiciones	

Tabla 31 Caso de uso Modificar Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el actor ha ingresado al módulo Usuario.	El sistema muestra una lista de todos los usuarios registrados anteriormente.
El usuario selecciona el usuario que desea modificar, haciendo click derecho sobre el registro y luego click en modificar.	El sistema mostrará el formulario del usuario para que modifique los datos del mismo.
El actor hace las modificaciones pertinentes el formulario de usuario.	
El usuario hace click en la opción "Guardar"	El sistema nos muestra un cuadro de dialogo que dice "Desea aplicar los cambios."
El usuario hace click en el botón "Si".	El sistema guarda los cambios efectuados por el usuario. El sistema muestra una notificación de: "Usuario se ha actualizado correctamente". Muestra una lista actualizada de todos los usuarios.

Tabla 32 Accion de los actores Modificar Usuario

Eliminar Usuario

Caso Uso	Eliminar Usuario
Actores	Gerente
Propósito	Eliminar registro de un Usuario.
Resumen	Los actores podrán eliminar a cualquier usuario de la base de datos.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar logueados y el ingreso del usuario debería realizarse anteriormente.

PostCondiciones	

Tabla 33 Caso de uso Eliminar Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Usuario.	El sistema muestra una lista de todos los Usuarios registrados anteriormente.
El actor selecciona el usuario que desea eliminar, haciendo click derecho sobre el registro y luego click en borrar.	El sistema nos muestra un cuadro de dialogo que dice "Seguro que desea continuar?"
El usuario hace click en el botón "Si".	El sistema elimina el Usuario. El sistema muestra una notificación de: "Usuario se ha borrado correctamente". Muestra una lista actualizada de todos los Usuarios.

Tabla 34 Acción de los actores Eliminar Usuario

Buscar Usuario

Caso Uso	**Buscar Usuario**
Actores	Gerente
Propósito	Buscará al usuario para consultar alguna información que desee el actor.
Resumen	Los actores podrán visualizar cualquier información del usuario que deseen.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar registrados, y al menos se debería haber realizado un ingreso del usuario anteriormente.
PostCondiciones	

Tabla 35 Caso de uso Buscar Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el actor ha ingresado al módulo Usuario.	El sistema muestra una lista de todos los usuarios registrados anteriormente.
El Actor podrá buscar al cliente por su cedula, nombres, apellidos, teléfono, o mail.	El sistema filtrara los datos del usuario.

Tabla 36 Acción de los actores Buscar Usuario

Exportar archivo

Caso Uso	**Exportar Archivo**
Actores	Gerente
Propósito	Exportar la lista de los Productos.
Resumen	Los actores podrán exportar la lista de los productos ya sea en los formatos Excel, PDF, CSV y XML.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar registrados.
PostCondiciones	

Tabla 37 Caso de uso Exportar archivo

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Usuario.	El sistema muestra una lista de todos los usuarios registrados anteriormente.
El usuario podrá exportar la lista de los usuarios registrados.	El sistema descargara un archivo dependiendo del formato seleccionado por el actor.

Tabla 38 Acción de los actores Exportar Archivo

Ver Factura

Caso Uso	Ver Facturas
Actores	Gerente.

Propósito	Permitir ver todas las facturas emitidas por un usuario.
Resumen	Los actores, podrán ver las factura que emitió un usuario en particular
Tipo	Extensión
PreCondiciones	Los usuarios registrados en el sistema, como gerentes, deben ingresar al módulo usuario, una vez listados todos los usuarios registrados, podrán elegir uno y listar sus facturas.
PostCondiciones	

Tabla 39 Caso de uso ver Factura

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo usuario.	El sistema muestra una lista de todos los usuarios registrados anteriormente.
Se desplegara un menú al hacer click derecho sobre un usuario, y seleccionaremos la opción que dice "Ver facturas"	El sistema responderá mostrando una lista con todas las facturas de que emitió ese usuario.

Tabla 40 Acción de los Actores Ver Factura

Ver Tipo Usuario

Caso Uso	**Ver Tipo Usuario**
Actores	Gerente.
Propósito	Permitir ver qué tipo de usuario es, si es cajero, administrador, o gerente.
Resumen	Los actores, podrán que tipo de usuario es una persona registrada.
Tipo	Extensión
PreCondiciones	Los usuarios registrados en el sistema, como gerentes, deben ingresar al módulo usuario, una vez listados todos los usuarios

	registrados, podrán elegir uno y ver a qué tipo pertenece.
PostCondiciones	

Tabla 41 Caso de uso Ver Tipo Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo usuario.	El sistema muestra una lista de todos los usuarios registrados anteriormente.
Se desplegara un menú al hacer click derecho sobre un usuario, y seleccionaremos la opción que dice "Ver tipo Usuario"	El sistema responderá mostrando un dialogo que indique tipo de usuario que pertenece dicho registro.

Tabla 42 Acción de los actores Ver Tipo Usuario

2.1.2.4 Caso de uso Factura

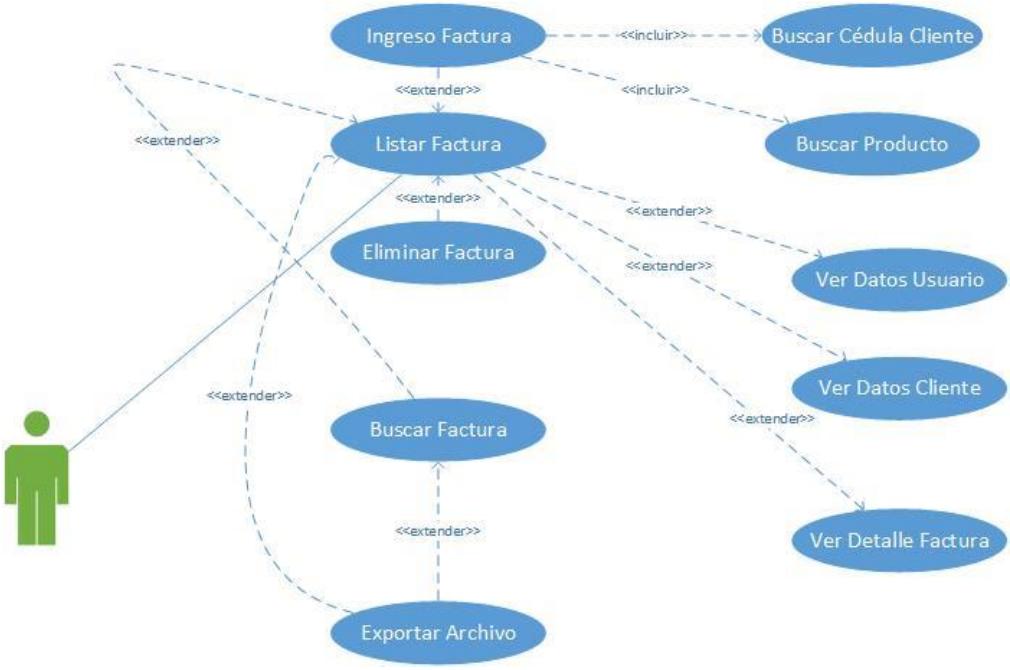


Ilustración 5 Caso de uso Factura

Listar Factura

Caso Uso	Listar Factura
Actores	Cajeros, Administradores, Gerentes
Propósito	Permitir ver todas las facturas emitidas.

Resumen	Los actores podrán ver los datos de todas las facturas.
Tipo	Primario
PreCondiciones	Los usuarios registrados, deben ingresar al módulo factura y ahí se les visualizara todas las facturas emitidas.
PostCondiciones	

Tabla 43 Caso de uso Listar Factura

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando los actores han ingresado al módulo Usuario.	El sistema muestra una lista las facturas ingresadas anteriormente.

Tabla 44 Acción de los actores Listar Factura

Ingreso Factura

Caso Uso	**Ingreso Factura**
Actores	Cajero, administrador, Gerente.
Propósito	Permitir registrar en la base datos una factura por la compra de un producto.
Resumen	Los actores podrán ingresar los datos de una factura realizada.
Tipo	Extensión.
PreCondiciones	Los usuarios registrados en el sistema, podrán ingresar al módulo factura, una vez se tenga ingresado por lo menos un cliente y un producto podrá generar una factura.
PostCondiciones	Buscar cédula del cliente, y productos.

Tabla 45 Caso de uso Ingreso Factura

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando los actores han ingresado al módulo Factura.	El sistema muestra una lista de todas las facturas emitidas hasta el momento.

El usuario selecciona el botón crear o puede hacer click derecho sobre cualquier fila de la lista y presionar crear.	El sistema mostrará el formulario de ingreso de la factura.
El usuario hace click en "Si"	El sistema guarda los datos de la nueva factura emitida El sistema muestra una notificación de: "Factura se ha generado Correctamente". Muestra una lista actualizada de todas las facturas.

Tabla 46 Acción de los Actores Ingreso Factura

Buscar Factura

Caso Uso	**Buscar Factura**
Actores	Cajero, Administrador , Gerente
Propósito	Buscará la factura para consultar alguna información que desee saber el actor.
Resumen	Los actores podrán visualizar cualquier información de la factura.
Tipo	Extensión.
PreCondiciones	Los actores deberán estar registrados, y al menos se debería haber realizado una factura anteriormente.
PostCondiciones	

Tabla 47 Caso de uso Buscar Factura

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el actor ha ingresado al módulo factura.	El sistema muestra una lista de todas las facturas almacenadas anteriormente
El Actor podrá buscar la factura por usuario que emitió la factura, por cliente o por id de la factura.	El sistema filtrara los datos de la factura.

Tabla 48 Acción de los Actores Buscar Factura

Ver Usuario

Caso Uso	**Ver Usuario**
Actores	Cajero, administrador, gerente.
Propósito	Permitir ver los datos del usuario que realizo una factura.
Resumen	Los actores, podrán ver toda la información del usuario que realizo una factura.
Tipo	Extensión
PreCondiciones	Los usuarios registrados en el sistema, deben ingresar al módulo factura, una vez listados todas las facturas registradas, podrán elegir uno y ver a que usuario realizo la factura seleccionada.
PostCondiciones	

Tabla 49 Caso de uso Ver Usuario

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Factura.	El sistema muestra una lista de todas las facturas registrados anteriormente.
Se desplegara un menú al hacer click derecho sobre una factura, seleccionaremos la opción que dice "Ver Usuario"	El sistema responderá mostrando un dialogo, donde podremos ver todos los datos del usuario que emitió esta factura.

Tabla 50 Acción de los Actores Ver Usuario

Ver Cliente

Caso Uso	Ver Cliente
Actores	Cajero, administrador, gerente.
Propósito	Permitir ver los datos del cliente para el cual se emitió la factura.

Resumen	Los actores, podrán ver toda la información del cliente para el cual se realizó una factura.
Tipo	Extensión
PreCondiciones	Los usuarios registrados en el sistema, deben ingresar al módulo factura, una vez listados todas las facturas registradas, podrán elegir uno y ver para que cliente se realizó la factura seleccionada.
PostCondiciones	

Tabla 51 Caso de usuario Ver Cliente

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Factura.	El sistema muestra una lista de todas las facturas registrados anteriormente.
Se desplegara un menú al hacer click derecho sobre una factura, seleccionaremos la opción que dice "Ver Cliente"	El sistema responderá mostrando un dialogo, donde podremos ver todos los datos del cliente para el cual se emitió esta factura.

Tabla 52 Acción de los Actores Ver Cliente

Ver Detalle Factura.

Caso Uso	Ver Detalle Factura.
Actores	Cajero, administrador, gerente.
Propósito	Permitir ver el detalle de una factura
Resumen	Los actores, podrán ver el detalle de una factura.
Tipo	Extensión
PreCondiciones	Los usuarios registrados en el sistema, deben ingresar al módulo factura, una vez listados todas las facturas registradas, podrán elegir uno y ver su detalle.
PostCondiciones	

Tabla 53 Caso de Uso Ver Detalle Factura

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando el usuario ha ingresado al módulo Factura.	El sistema muestra una lista de todas las facturas registrados anteriormente.
Se desplegara un menú al hacer click derecho sobre una factura, seleccionaremos la opción que dice "Ver Detalle"	El sistema responderá mostrando un dialogo, donde podremos ver todos los datos del cliente para el cual se emitió esta factura.

Tabla 54 Acción de los actores Ver Detalle Factura

2.1.2.5 Caso de uso Reportes.

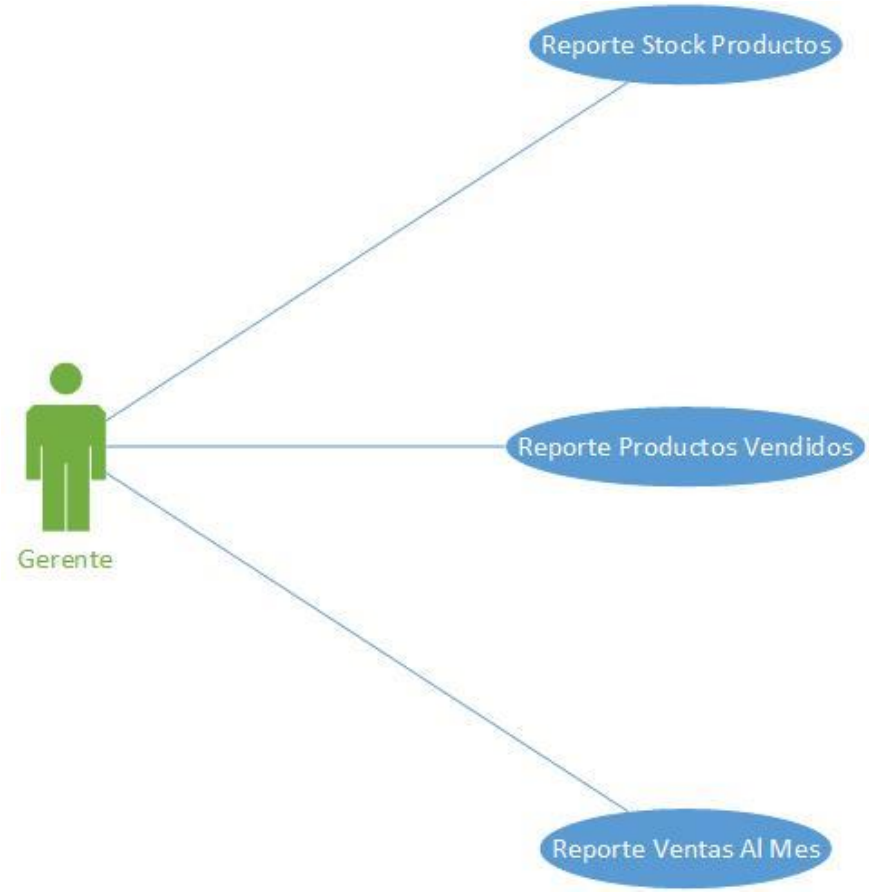


Ilustración 6 Caso de uso Reportes

Reporte Stock de Productos

Caso Uso	**Reporte Stock de productos**
Actores	Gerentes
Propósito	Permitir ver de manera gráfica el stock de productos.
Resumen	Los gerentes podrán ver un gráfico estadístico en barras de los productos en stock.
Tipo	Primario
PreCondiciones	Los usuarios registrados, deben ingresar al módulo Reportes y ahí seleccionar en Reportes de Stock.
PostCondiciones	

Tabla 55 Caso de Uso Reporte Stock de Productos

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando los actores han ingresado al módulo Reportes.	El sistema muestra un submenú con todos los reportes posibles para visualizar.
Seleccionamos el reporte "Reportes Stock"	El sistema responderá mostrando un cuadro estadístico en barras del stock de productos.
El cliente puede seleccionar cuantos productos mostar.	

Tabla 56 Acción de los Actores Stock de Productos

Reporte vendedores.

Caso Uso	**Reporte Vendedores**
Actores	Gerentes
Propósito	Permitir ver de manera gráfica un porcentaje de las ventas por usuario.
Resumen	Los gerentes podrán ver un gráfico estadístico tipo pie de las ventas por ano
Tipo	Primario

PreCondiciones	Los usuarios registrados, deben ingresar al módulo Reportes y ahí seleccionar en reporte productos vendidos.
PostCondiciones	

Tabla 57 Caso de uso Reporte Vendedores

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando los actores han ingresado al módulo Reportes.	El sistema muestra un submenú con todos los reportes posibles para visualizar.
Seleccionamos el reporte "Reportes Stock"	El sistema responderá mostrando un cuadro estadístico tipo pie de las ventas de los usuarios.
Podremos seleccionar el ano que queremos revisar.	
Hacemos click en ver.	El sistema carga nuevamente el grafico con el año deseado.

Tabla 58 Accion de los Actores Reporte Vendedores

Reporte ventas mes.

Caso Uso	**Reporte ventas mes**
Actores	Gerentes
Propósito	Permitir ver de manera gráfica las ventas realizadas cada mes.
Resumen	Los gerentes podrán ver un gráfico estadístico en línea de las ventas por mes.
Tipo	Primario
PreCondiciones	Los usuarios registrados, deben ingresar al módulo Reportes y ahí seleccionar en reporte ventas mes.
PostCondiciones	

Tabla 59 Caso de uso Reporte Venta Mes

Acción de los Actores	Respuesta del Sistema
Este caso de uso comienza cuando los actores han ingresado al módulo Reportes.	El sistema muestra un submenú con todos los reportes posibles para visualizar.
Seleccionamos el reporte "Reportes Stock"	El sistema responderá mostrando un cuadro estadístico en línea de las ventas al mes.

Tabla 60 Acción de los Actores Reporte venta Mes

2.1.2.5 Caso de uso Sesiones.

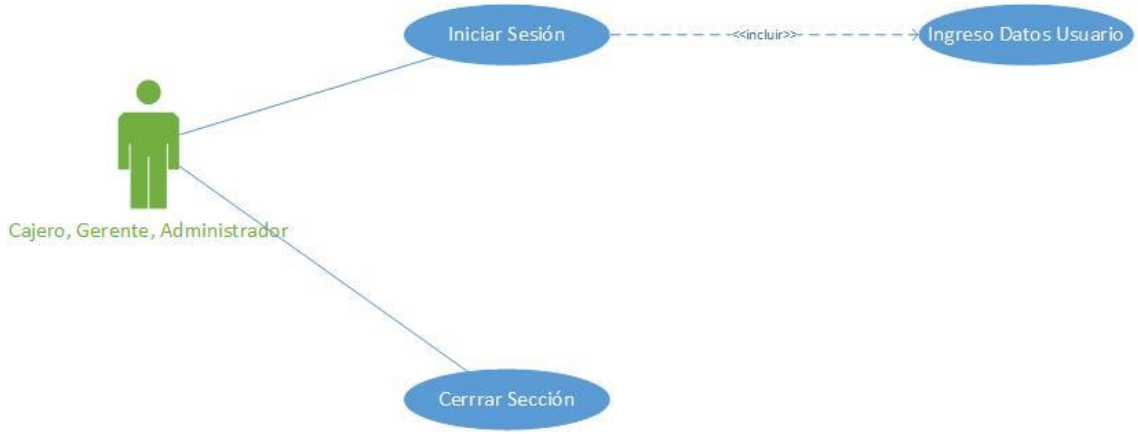


Ilustración 7 Caso de uso Sesiones

Inicio Sesión

Caso Uso	**Iniciar Sesión**
Actores	Cajeros, gerentes, administradores.
Propósito	Permitir ingresar al sistema de facturación.
Resumen	Permite el ingreso de los usuarios al sistema.
Tipo	Primario
PreCondiciones	

PostCondiciones	Realizar la facturación, ver listas, edición de clientes, etc.

Tabla 61 Caso de uso Inicio Sesión

Acción de los actores	Respuesta del Sistema
En este caso comienza cuando el usuario solicita ingresar al Sistema.	El sistema muestra el formulario de validación de datos.
Usuario proporciona datos vitales, nombre de usuario y contraseña.	El sistema valida los datos y de ser correctos permite que el usuario vea los módulos según sus privilegios.

Tabla 62 Acción de los actores Inicio Sesión.

Cerrar Sesión

Caso Uso	**Cerrar Sesión**
Actores	Cajero, Administrador, Gerente
Propósito	Permitir cerrar de los diferentes usuarios las sesiones.
Resumen	Permite cerrar sesión de los usuarios al sistema.
Tipo	Primario
PreCondiciones	El actor debe estar logueado antes de cerrar su sesión.
PostCondiciones	

Tabla 63 Caso de Uso Cerrar Sesión

Acción de los actores	Respuesta del Sistema
En este caso comienza cuando el usuario solicita cerrar la sesión del Sistema.	El sistema muestra el botón Login Out o Cerrar Sesión en el menú principal.
Usuario selecciona cerrar Sesión.	El sistema cierra la sesión, evitando el ingreso al sistema.

Tabla 64 Acción de los actores Cerrar Sesión

CAPITULO 3

Instalación.

3.1 Descarga e instalación de NetBeans 8.0.1 con JDK 7.

1. Primero realizaremos la descarga de la página oficial de netbeans, en el siguiente enlace.

   http://www.oracle.com/technetwork/java/jdk-7-netbeans-download-432126.html

2. Aceptamos el acuerdo de licencia dando click en "Accept License Agreement"

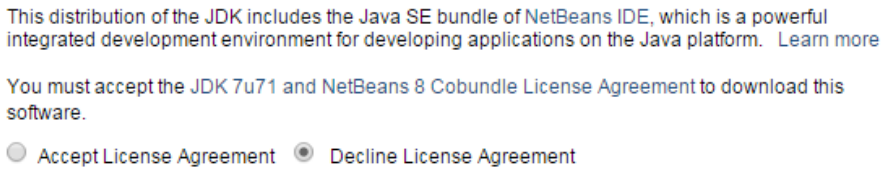


Ilustración 8 Acuerdo de licencia Netbeans

3. Obtendremos una pantalla asi:

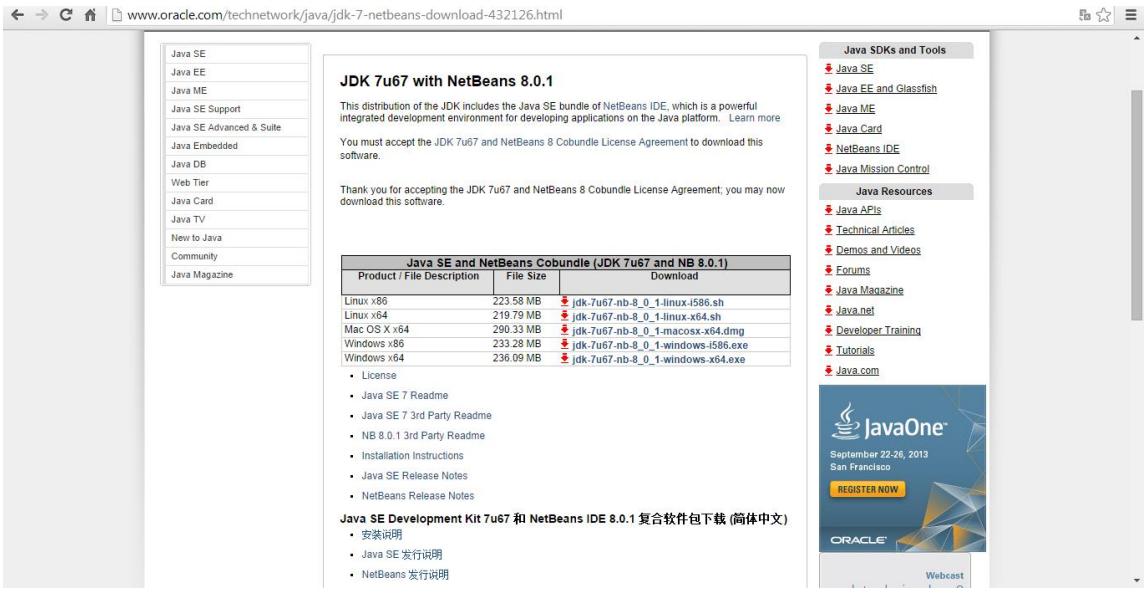
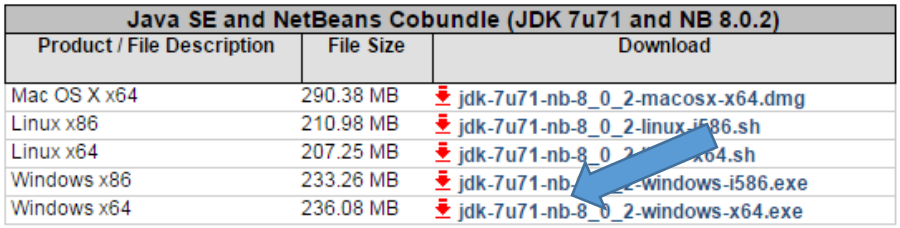


Ilustración 9 Plataformas de Descarga Netbeans

4. Ahora la plataforma sobre la cual deseamos instalar nuestro netbeans con 8.0.1 con jdk 7, para este caso seleccionaremos Windows X64.



Java SE and NetBeans Cobundle (JDK 7u71 and NB 8.0.2)		
Product / File Description	File Size	Download
Mac OS X x64	290.38 MB	jdk-7u71-nb-8_0_2-macosx-x64.dmg
Linux x86	210.98 MB	jdk-7u71-nb-8_0_2-linux-i586.sh
Linux x64	207.25 MB	jdk-7u71-nb-8_0_2-linux-x64.sh
Windows x86	233.26 MB	jdk-7u71-nb-8_0_2-windows-i586.exe
Windows x64	236.08 MB	jdk-7u71-nb-8_0_2-windows-x64.exe

Ilustración 10 Selección de la plataformaNetbeans

5. Después de haber descargado el archivo hacemos doble click sobre este y obtendremos una ventana así:

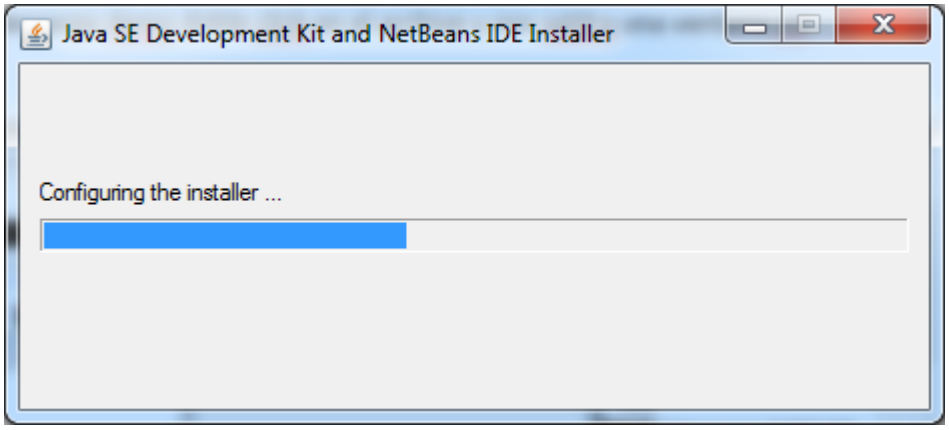


Ilustración 11Inicio de Instalación Netbeans

6. Una vez cargada la configuración del instalador damos click en “Next” en esta ventana.

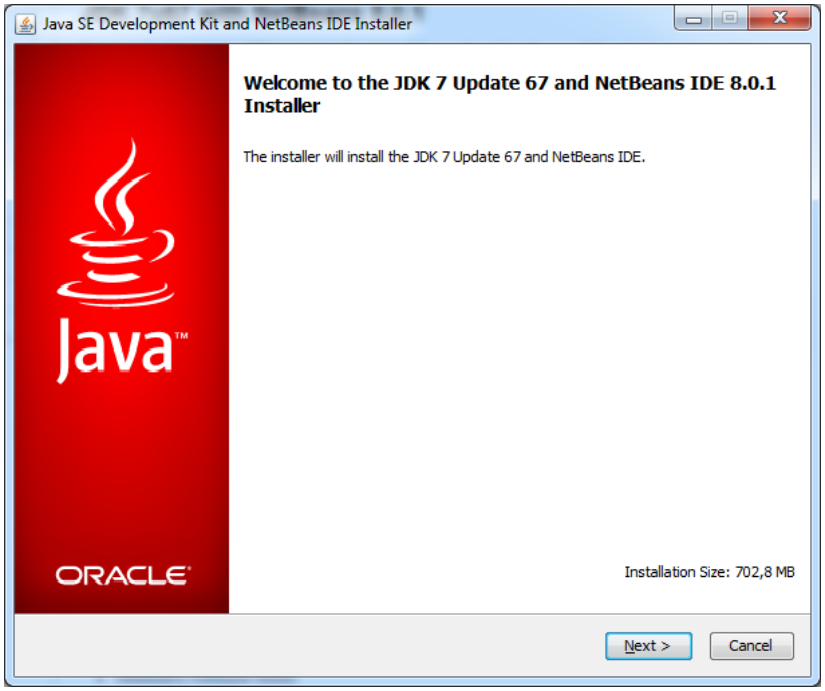


Ilustración 12Primera Pantalla de Instalación Netbeans

7. Después aceptamos los términos de la licencia, seleccionando la opción "I accept the terms in the licence agreement. install JUnit" y posteriormente hacemos click en "Next" como indica el siguiente gráfico:

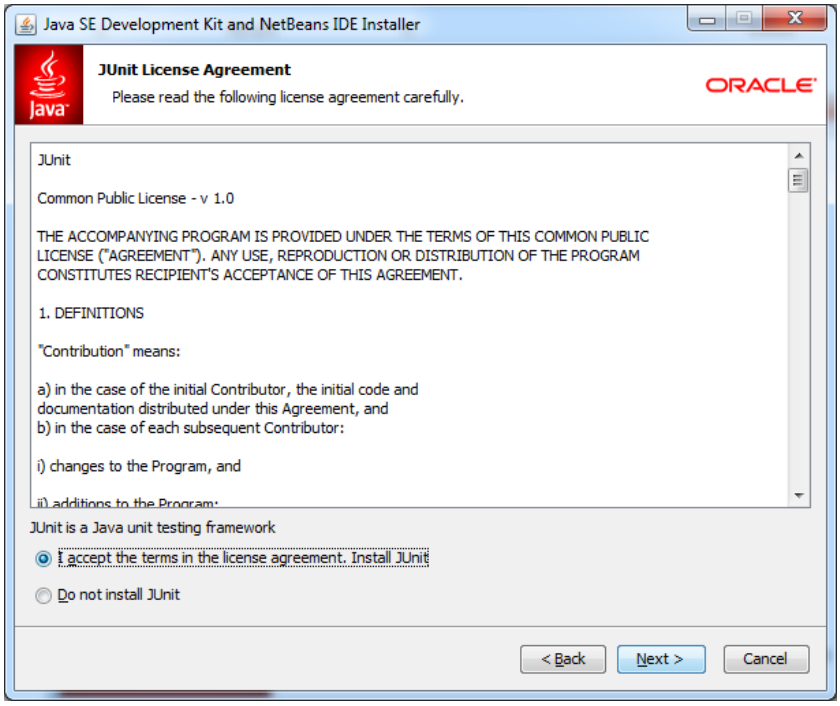


Ilustración 13 Licencia para instalación de Netbeans

8. Escogemos la dirección que para instalar el jdk, y seleccionamos “Next”.

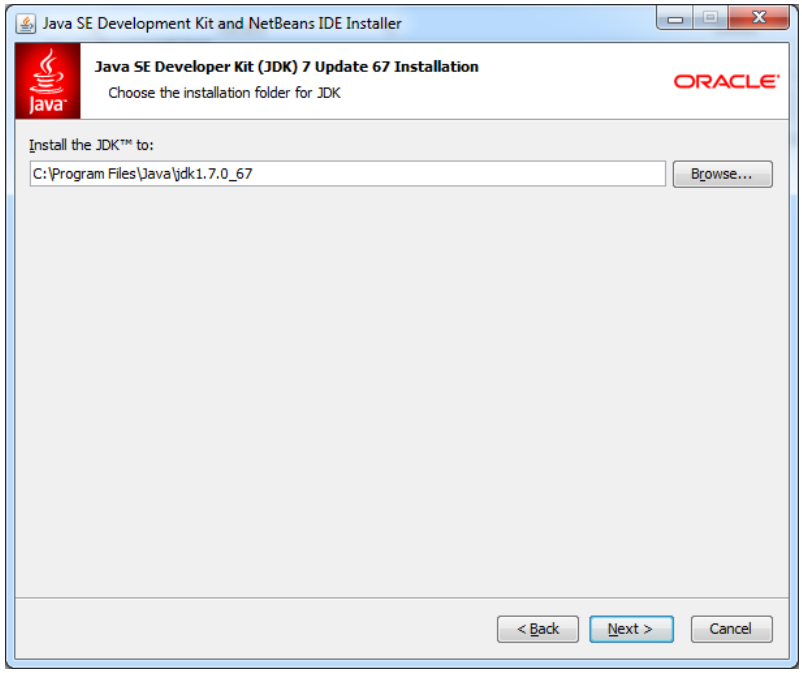


Ilustración 14 Path jdk

9. Ahora hacemos lo mismo para el Netbeans y por defecto ya nos sale seccionado en donde está ubicado el IDE del JDK para El NetBeans.

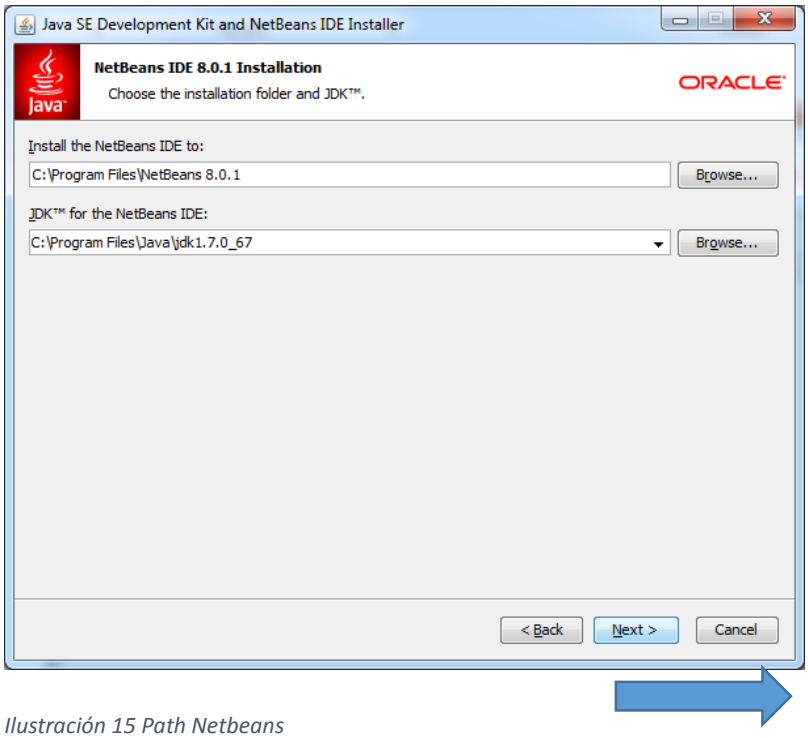


Ilustración 15 Path Netbeans

10. Finalmente podremos hacer click en “Install”, y por ultimo le damos en “Finish”.

3.2 Agregar Servidor GlassFish

1. Nos dirigimos a la web oficial de GlassFish.

https://glassfish.java.net/download.html

2.- Descargamos el archivo para la instalación, haciendo click en el paso 1 de la web, donde dice “Download”.

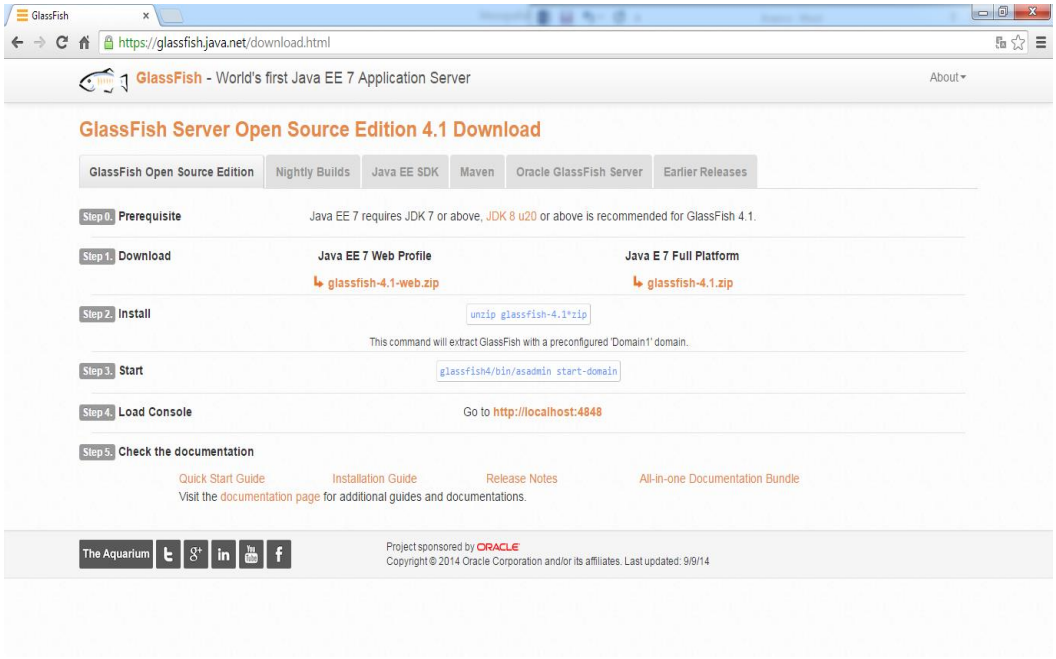


Ilustración 16 Pagina de descarga de GlassFish

3. Una vez descargado el archivo, nos dirigimos al NetBeans, a la pestaña “Tools”, y después damos click en la opción “Server”.

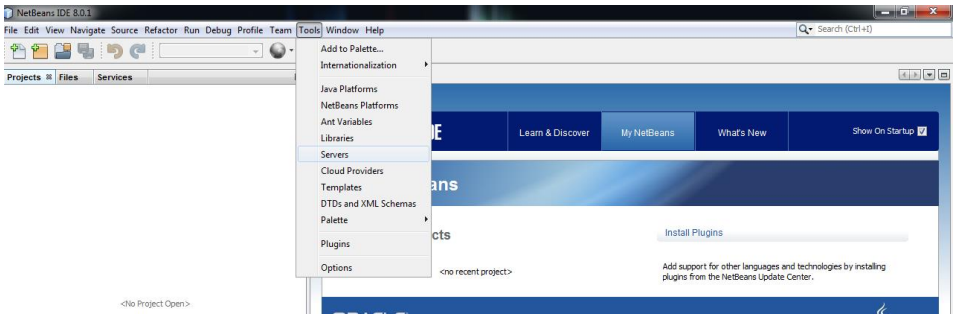


Ilustración 17 Menú Netbeans

4. Ahora se visualizara una nueva ventana, en la cual aremos click sobre el botón que dice "add Server…"

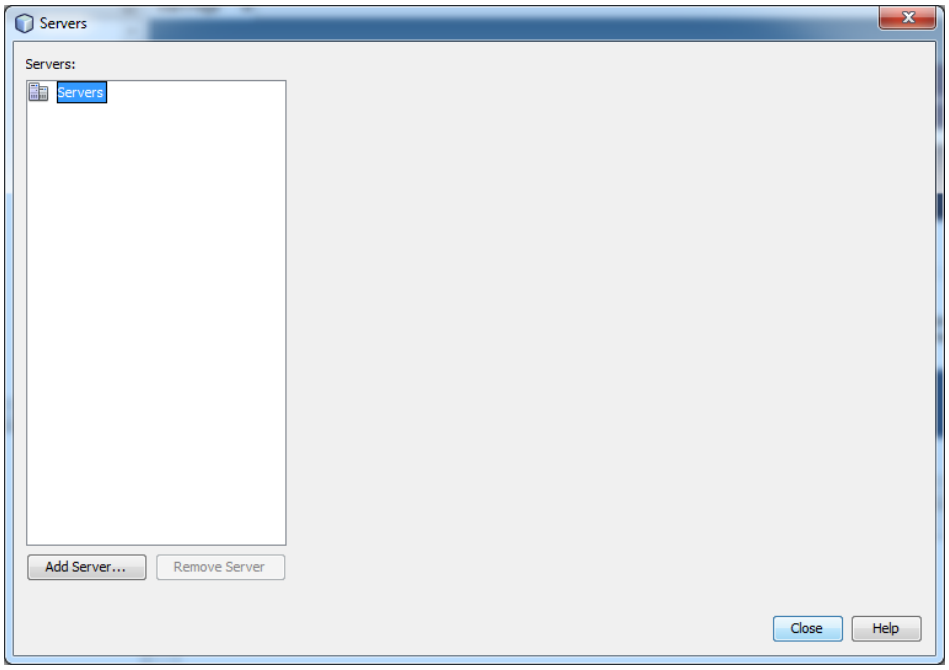


Ilustración 18 Pantalla agregar servicio

5. Posteriormente seleccionamos el servidor que vamos agregar, en este caso será "GlassFish Server " y luego click en el botón "Next".

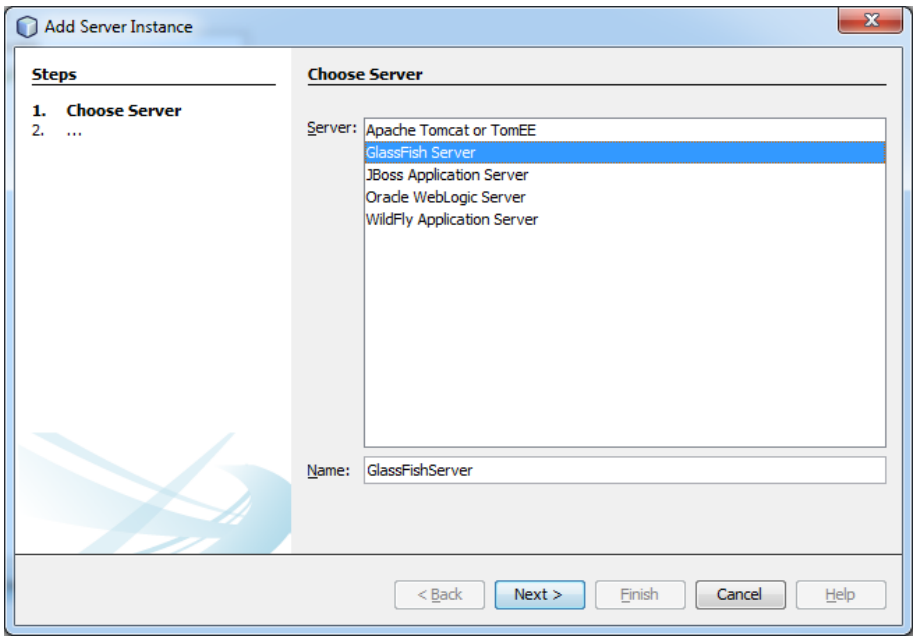


Ilustración 19 Agregar GlassFish

6. En esta ventana seleccionamos “Browse”, y escogemos el archivo que nos descargamos en el primer paso en la página de GlassFish”. Y posteriormente click en “Next”

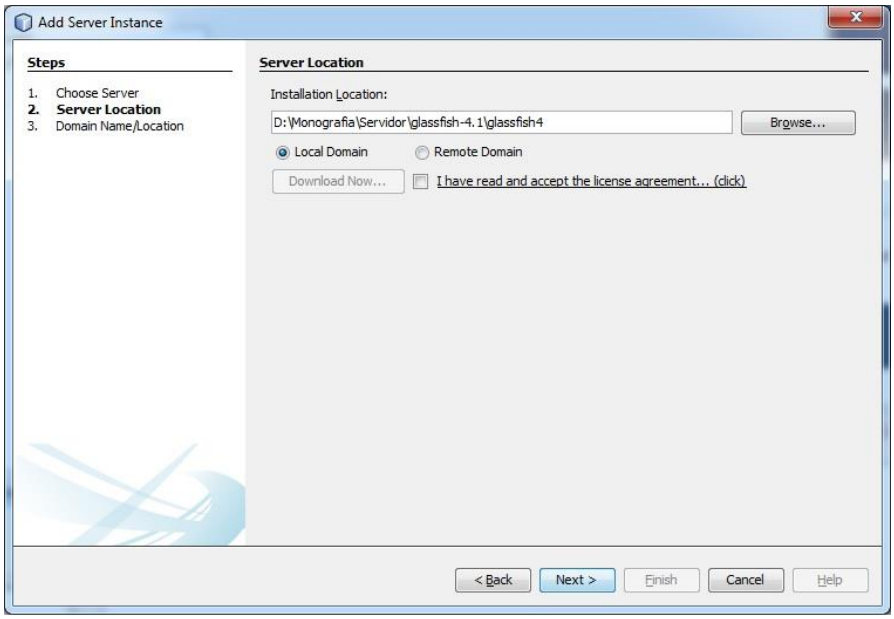


Ilustración 20 Path de descarga de GlassFish

7. En la siguiente ventana, vamos a poner un nombre a target, un usuario y una contraseña, y por ultimo click en “finish”.

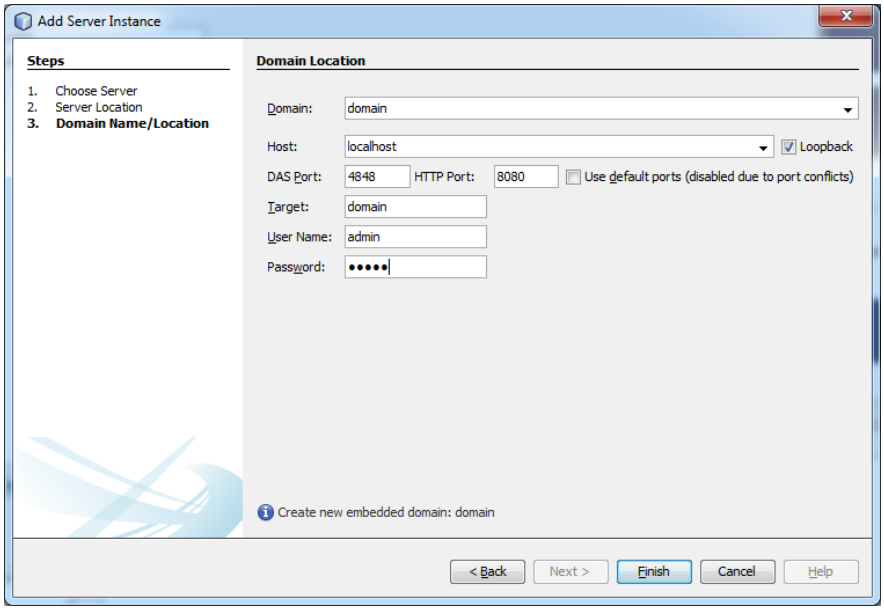


Ilustración 21 Dominios del Servidor

3.3 Pluging Crud Primefaces para Netbeans.

1. Nos dirigimos al siguiente enlace, que será en donde vamos a descargar el pluging.

http://sourceforge.net/projects/nbpfcrudgen/

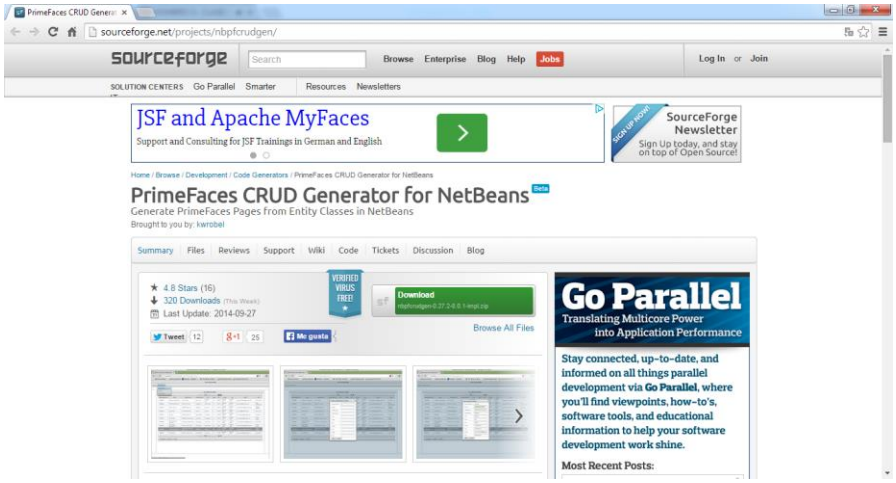


Ilustración 22 Pagina de descarga del Pluging Crud Primefaces

2. Abrimos NetBeans y nos dirigimos a "tools", en la opción "Plugins".

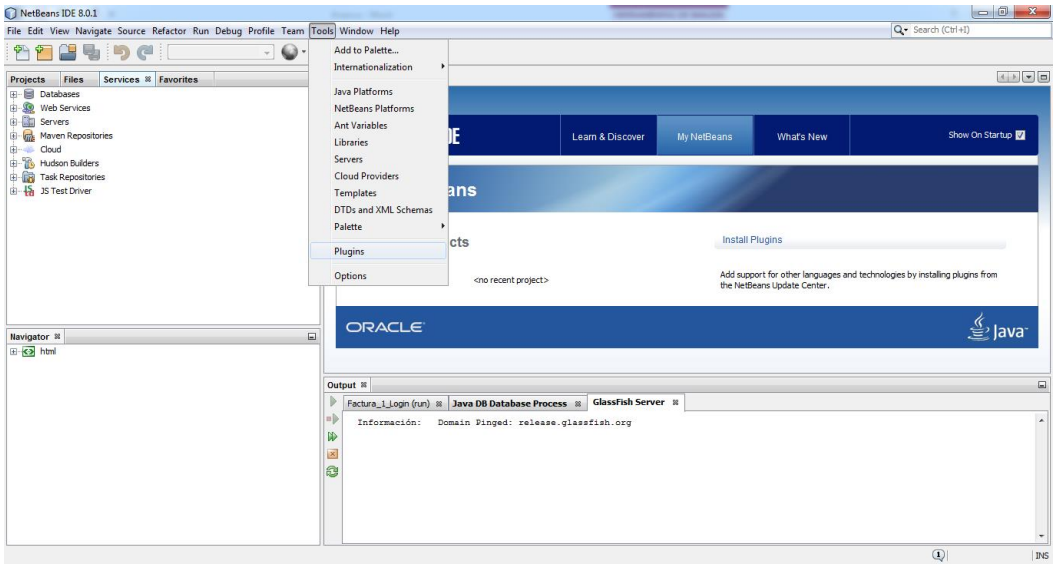


Ilustración 23 Menú Netbeans

3. Después nos dirigimos a la pestaña que dice "Downloaded", y hacemos click en el botón "Add Plugins".

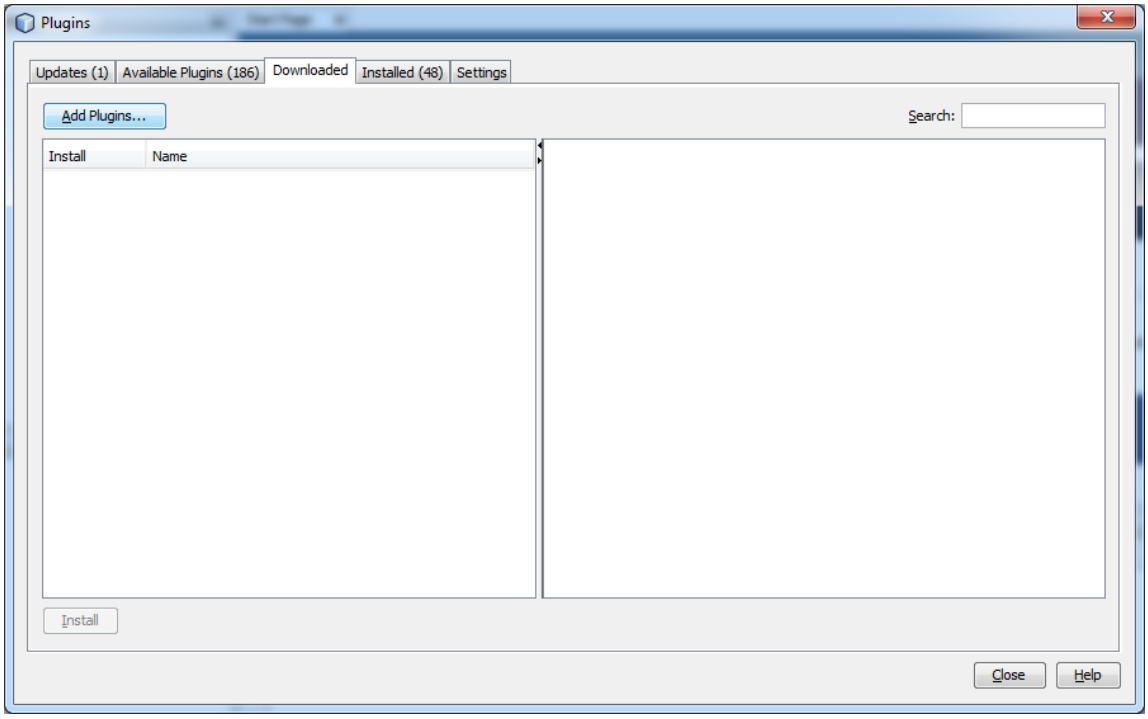


Ilustración 24 Agregar Plugins

4. Seleccionamos el plugin que nos acabamos de descargar y damos click en Abrir.

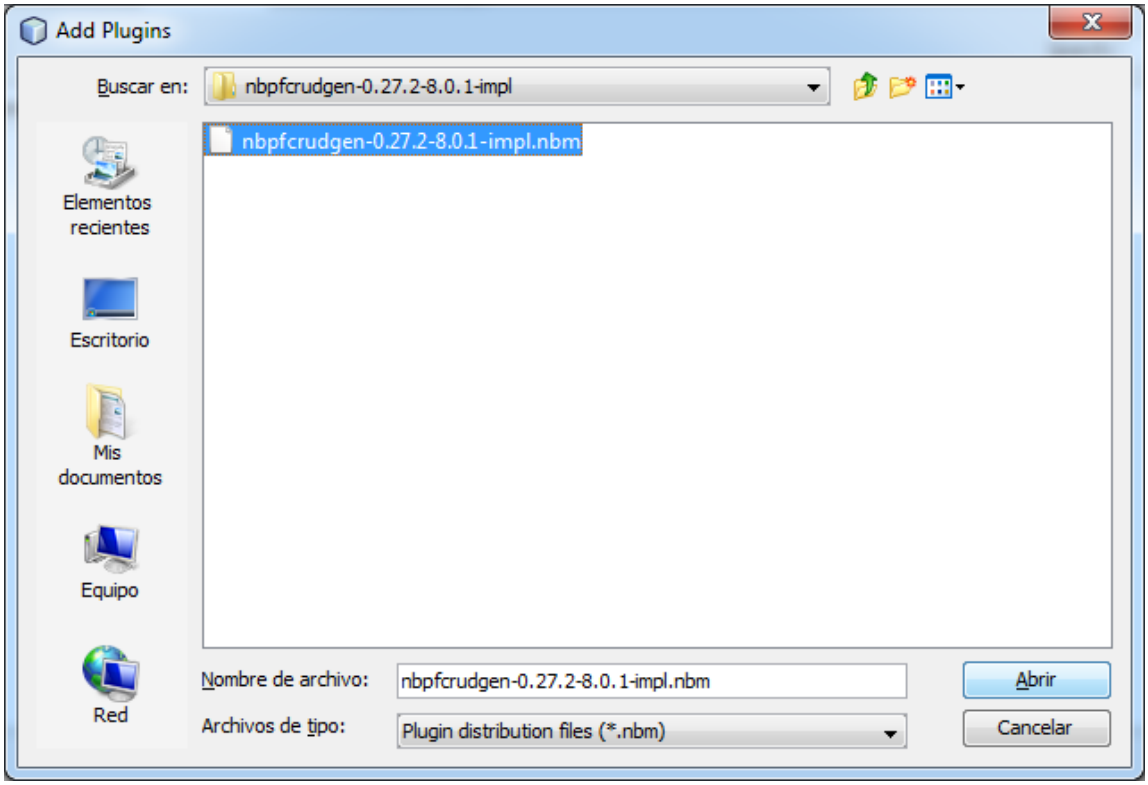


Ilustración 25 Buscar Plugin

5. En la nueva ventana que obtenemos tras seleccionar el plugin, damos click en el botón "Install".

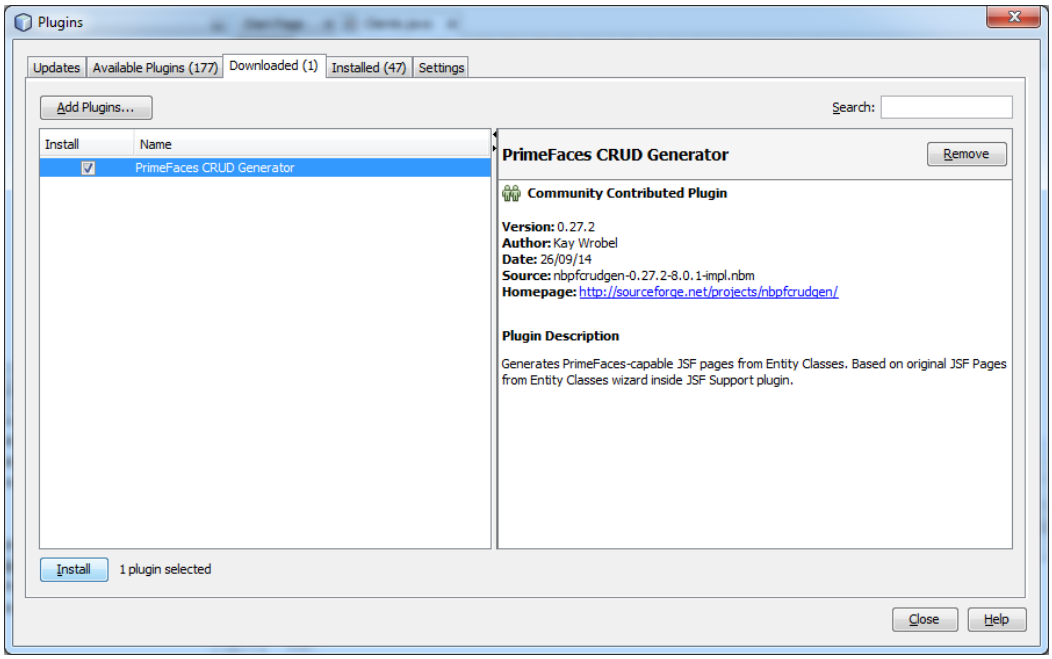


Ilustración 26 Intalar Plugin

6. Hacemos click en "Next", y posteriormente aceptamos los términos y condiciones, a continuación damos click en "Install".

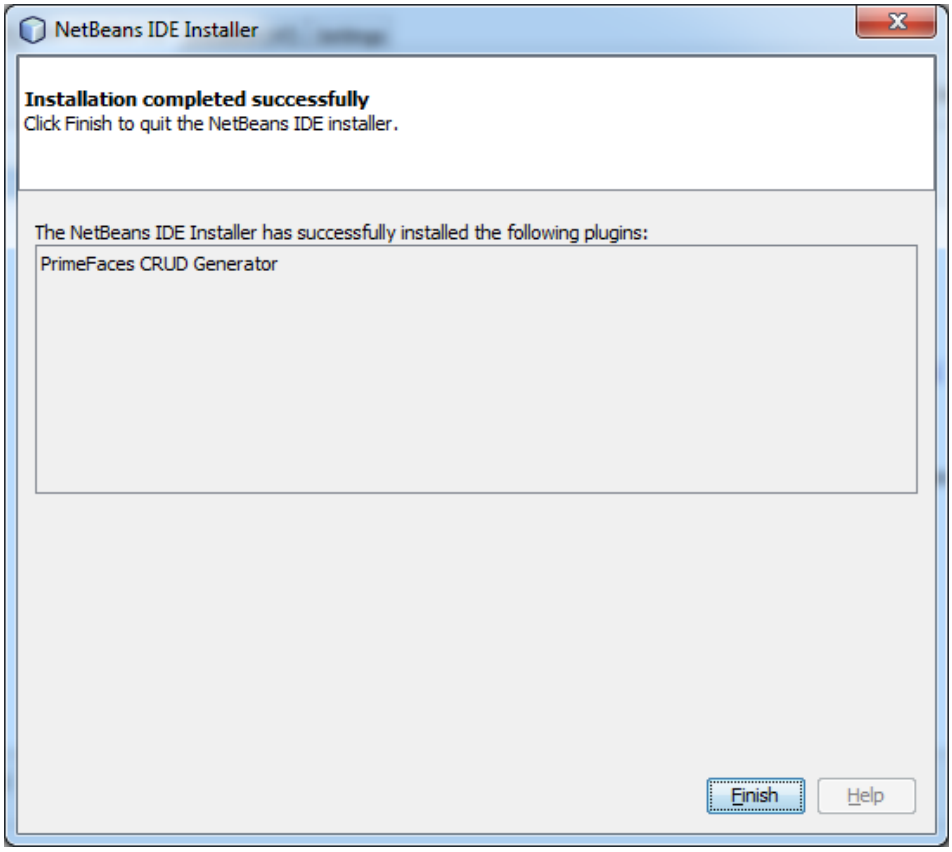


Ilustración 27 Instalacion Completa

Terminamos haciendo click en el botón “Finish”.

3.4 Descarga e instalación de MySql.

1. Como primer paso nos vamos a dirigir al siguiente enlace para realizar la descarga.

http://dev.mysql.com/downloads/workbench/

2. Una vez en la página, nos dirigiremos en la parte inferior, y seleccionaremos la platamorma en la cual se va a trabajar, en nuestro caso será “Microsft Windows”

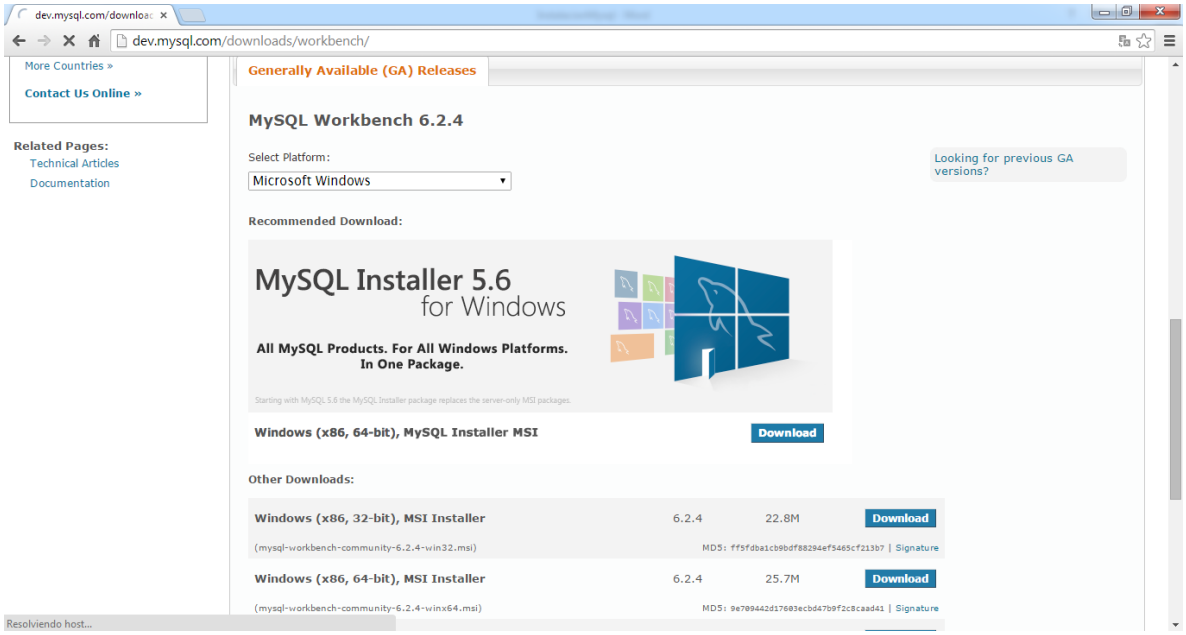


Ilustración 28 Pagina Descarga MySql

3. Después de haber seleccionado la plataforma seleccionaremos, en cuantos bits vamos a instalar, en este caso, lo aremos en una máquina de 64bits

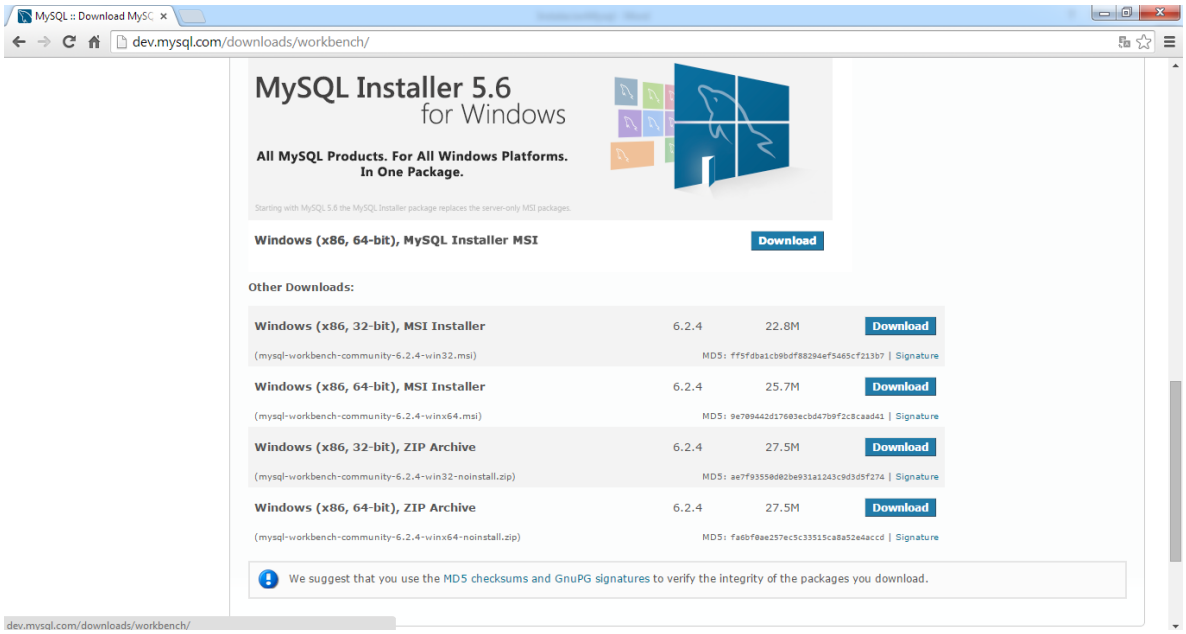


Ilustración 29Plataforma de descarga MySql

4. Después de finalizar la descarga, abriremos el archivo y obtendremos uno ventana así.

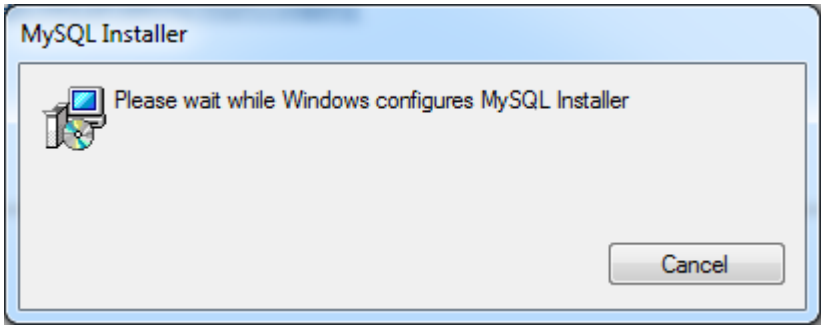


Ilustración 30 Carga de Instalador

5. Posteriormente, después de haber cargado el instalador, en la siguiente ventana, daremos click en la primera opción que dice "Install MYSQL Products"

Ilustración 31Venta de Bienvenida MySql

6. Aceptamos los términos y condiciones y damos click en "Next"

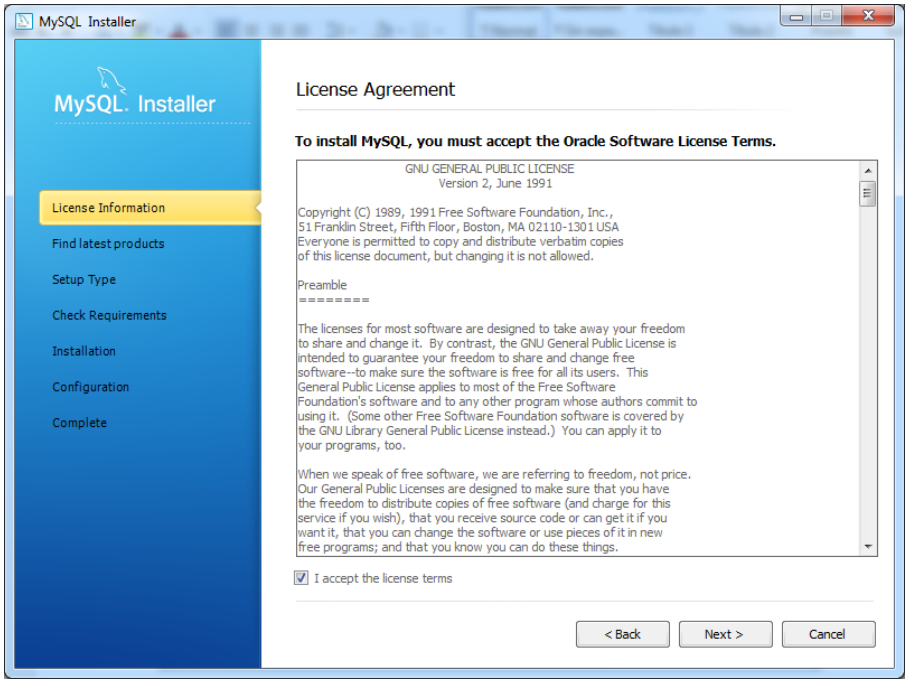


Ilustración 32Licencia Mysql

7. Hacemos nuevamente click en “Next”

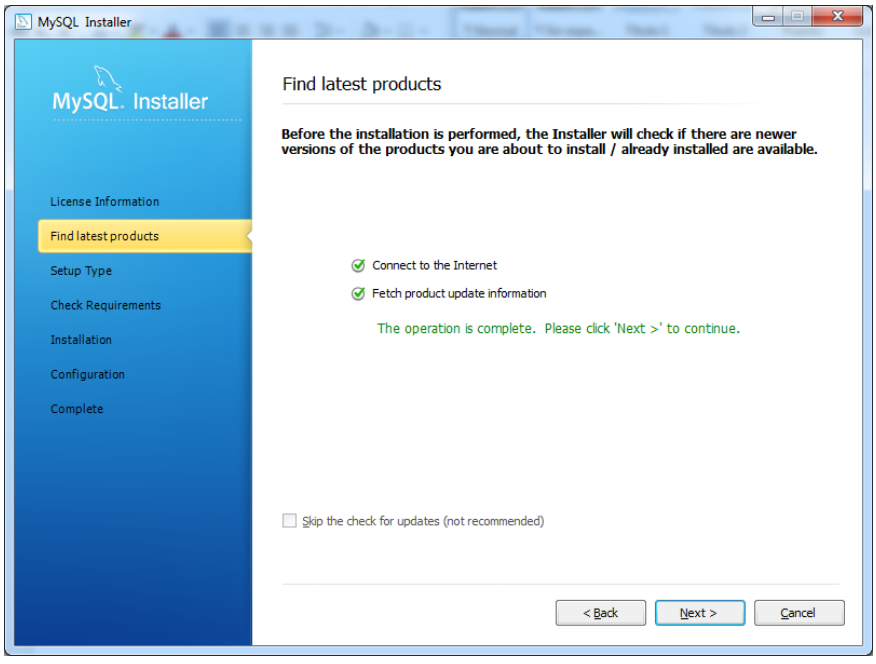


Ilustración 33 Verificación de conexión a Internet

8. Escogemos el tipo de instalación, que será “Developer Default ”, y click en “Next”

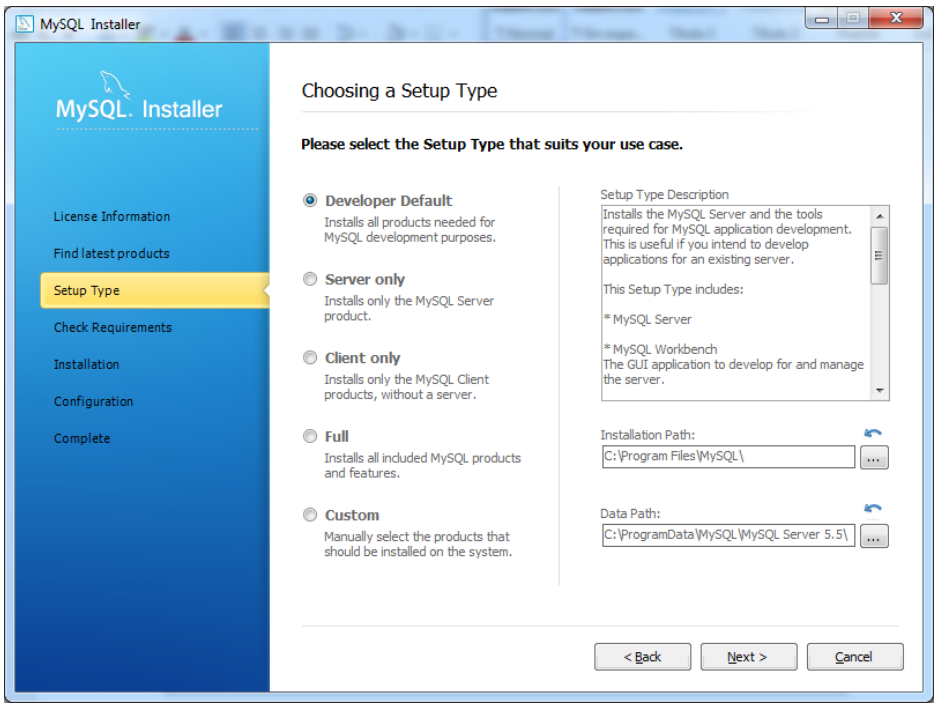


Ilustración 34 Tipo de Instalación

9. Nos indicara el progreso de la instalación de los componentes una vez terminado, click en “Next”.

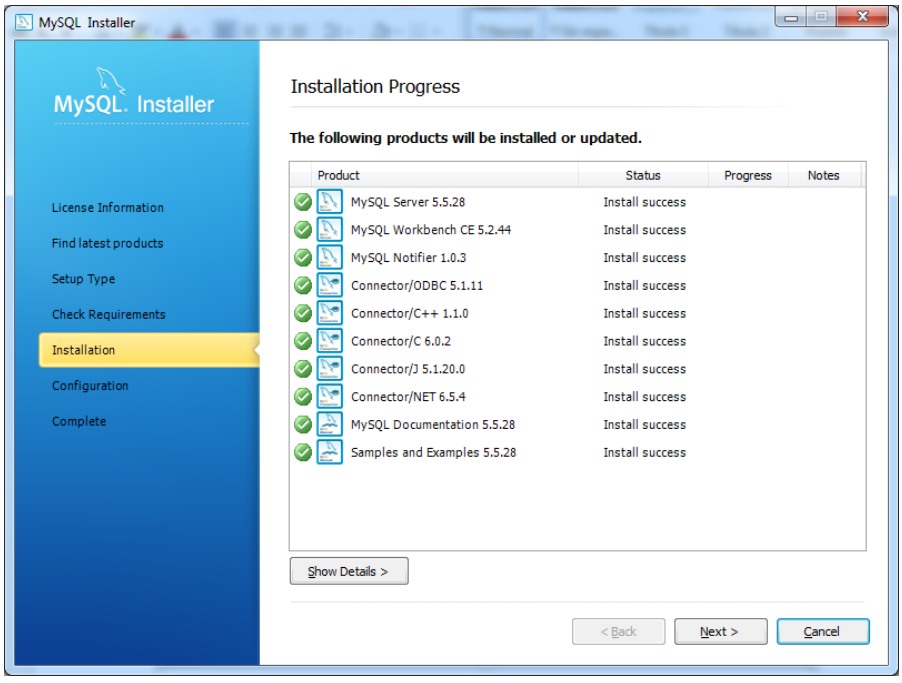


Ilustración 35 Instalación de componentes.

10. Se muestra el progreso de configuración de los componentes instalados.

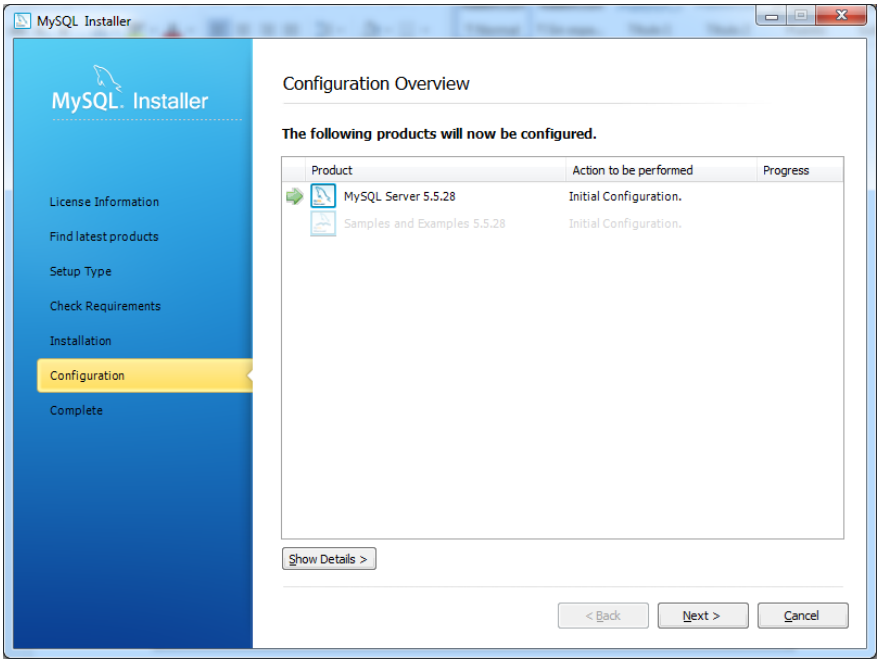


Ilustración 36 Configuración de los componentes

11. Después de haber configurado todo los componentes instalados, procederemos a configurar el número de puerto, en nuestro caso lo dejaremos como se encuentra.

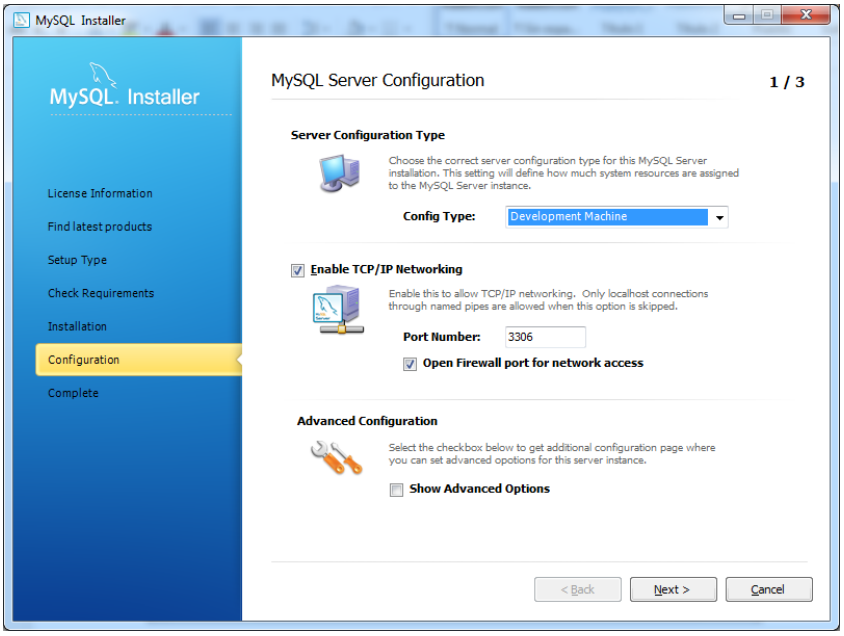


Ilustración 37 Puertos de MySql

12. Por ultimo haremos click en “Finish”.

CAPITULO 4

Creación del proyecto.

4.1 Creación del Proyecto.

1. En el menú nos dirigimos a “file”, y luego a “New proyect”.

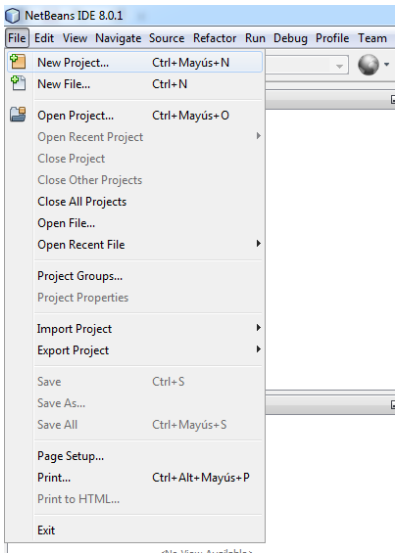


Ilustración 38 Menú NetBeans

2. En la siguiente ventana, en categorías escogemos “Java web” y proyecto escogemos, “web Aplications”, y a continuación “Next”.

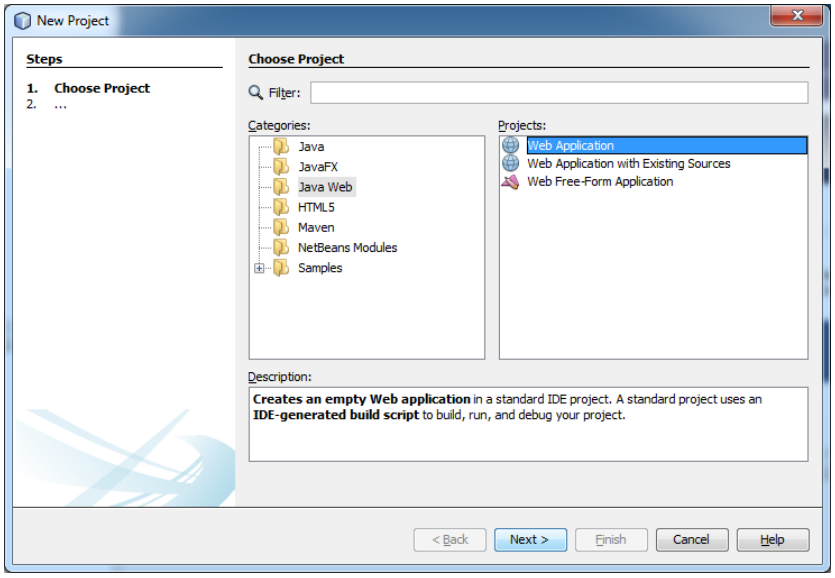


Ilustración 39 Selección Tipo Proyecto

3. En “Project Name” escribimos el nombre de nuestro proyecto, en “Proyect Location”, vamos a colocar la dirección en la cual queremos que se guarde nuestro proyecto, y posteriormente hacemos click en “Next”.

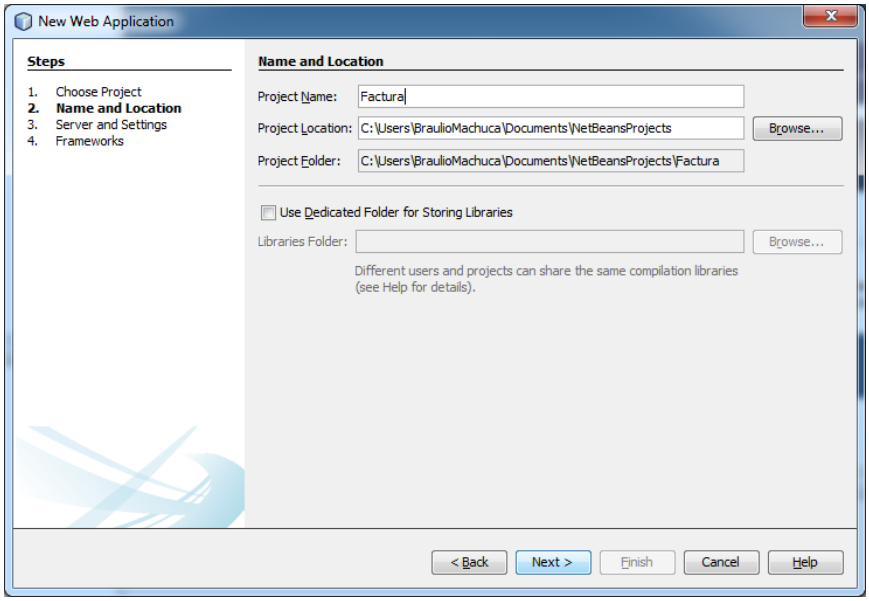


Ilustración 40 Nombre y dirección del Proyecto

4. En la siguiente ventana, Escogemos el servidor, en este caso escogemos el servidor GlassFish Server, en la versión escogemos “Java EE 7 Web” ya que es la última versión, en “Context Path”, dejamos el valor por defecto, y luego click en “Next”.

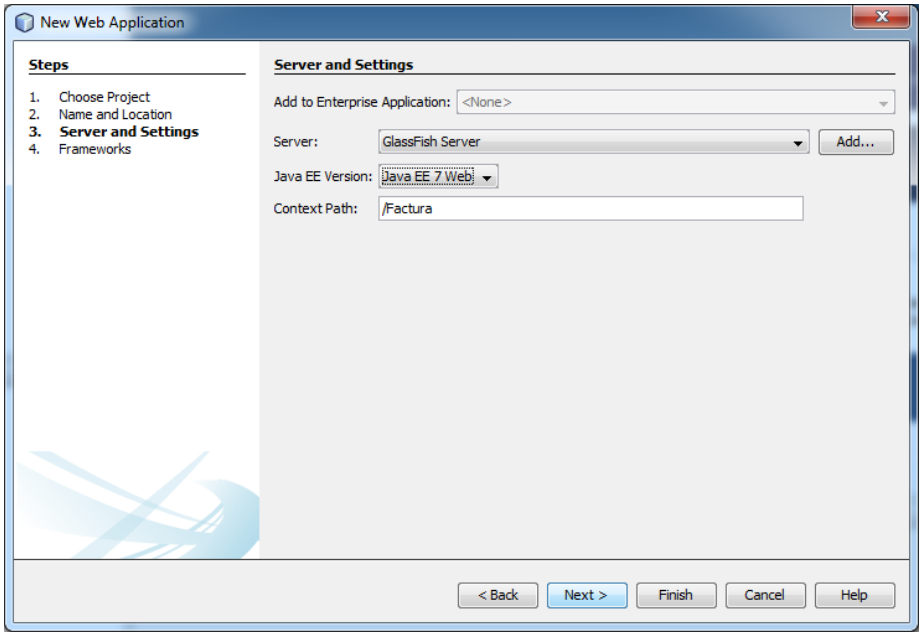


Ilustración 41 Seleccion del Servidor.

5. Ahora tendremos que escoger el Framework con el que vamos a trabajar, en este caso será con "JavaServer Faces", y en la pestaña de Componentes, vamos a escoger "Primefaces", dar click en more, para descargar la última versión de nuestro componente Primefaces.

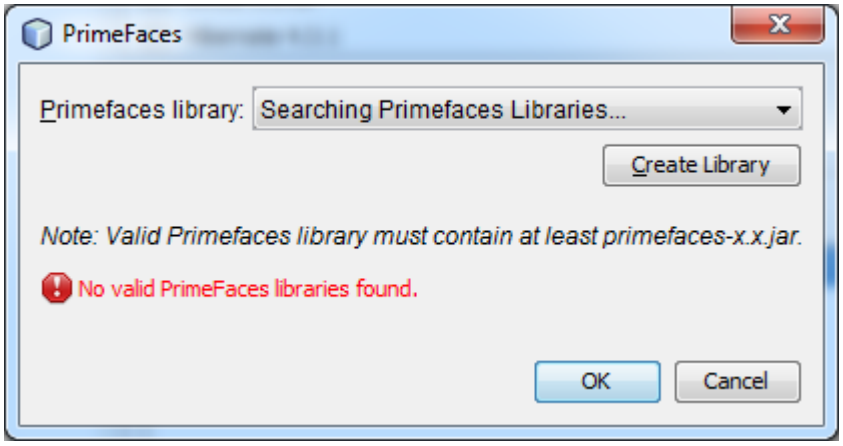


Ilustración 42 Buqueda Libreria Primefaces

6. Esperamos a que se valide la librería y procederemos a dar click en "OK".

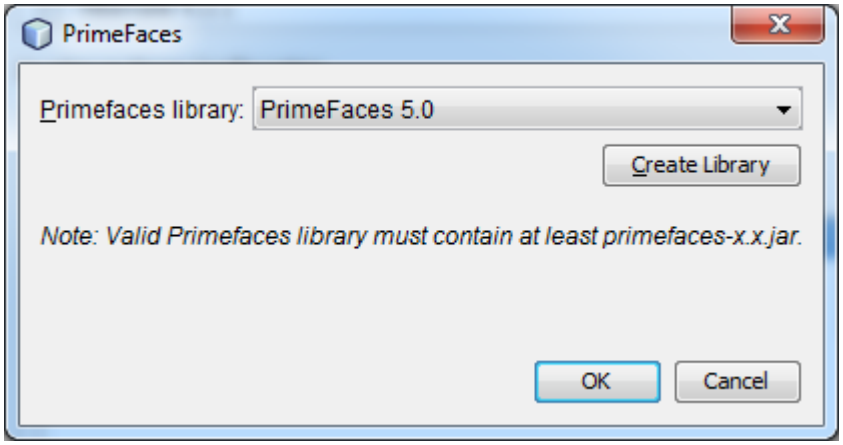


Ilustración 43 Validación Librería Primefaces

7. Finalmente verificamos q esté seleccionado el framework JavaServer faces, y en componentes Primefaces.

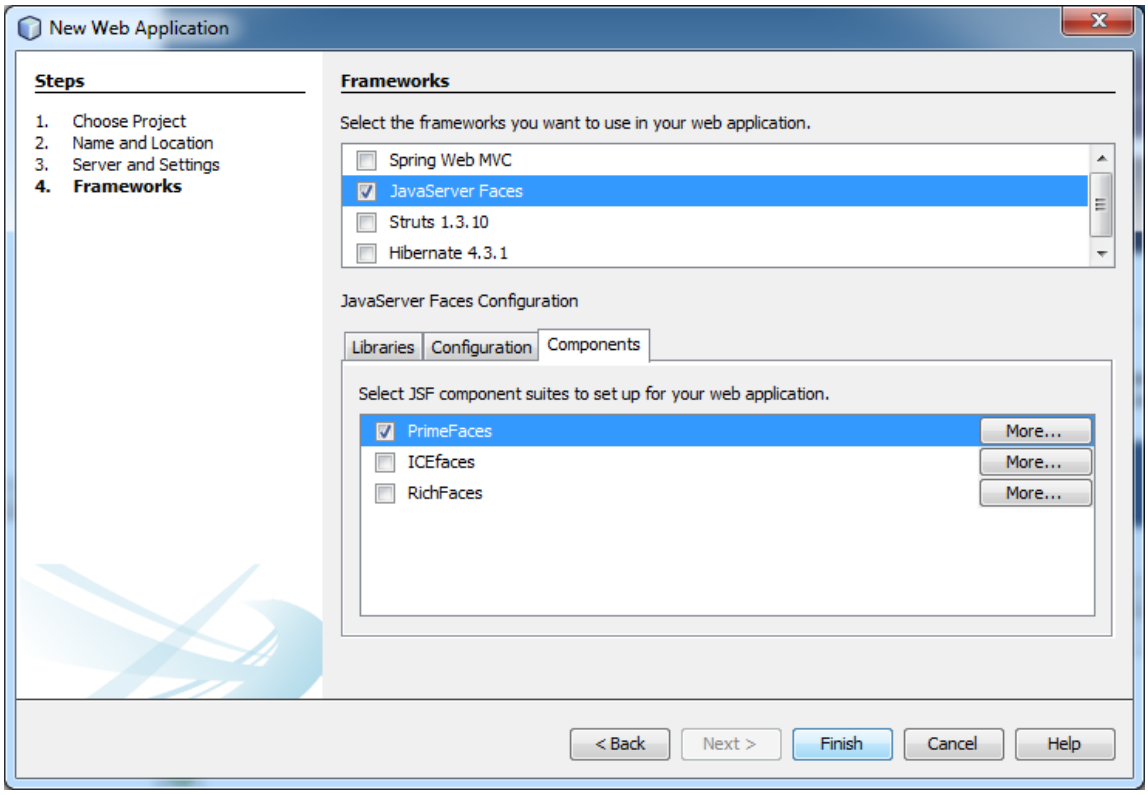


Ilustración 44 Componentes de la Aplicación

8. Por ultimo damos click en “Finish” para Crear Nuestro proyecto.

4.2 Conexión con la base de datos.

Después de haber creado nuestra base de datos, en este caso se ha creado con MySql, tenemos que realizar la conexión con nuestra aplicación.

1. Para esto necesitamos el conector de la base de datos para java, que lo podemos encontrar en la siguiente dirección.
   http://dev.mysql.com/downloads/connector/j/3.0.html

2. Una vez descargada nos dirigimos al NetBeans, a la pestaña “Service”, en la opción “Databases”

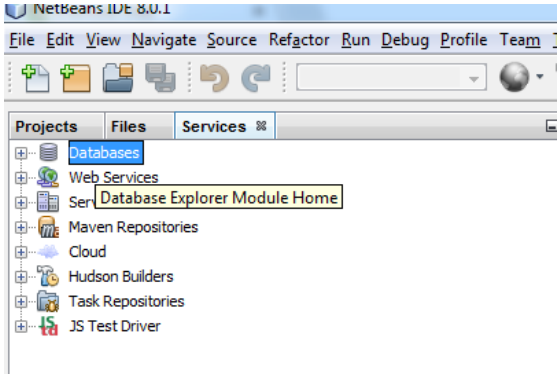


Ilustración 45 Opción Base de datos

3. Nos dirigimos a las opciones de “Databases” haciendo click derecho y escogemos la opción “New Conecction”.

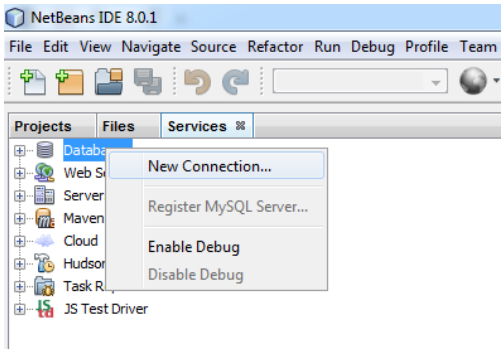


Ilustración 46 Nueva Conexión

4. En driver buscamos, el que dice “MySql (Conecctor /J driver)”

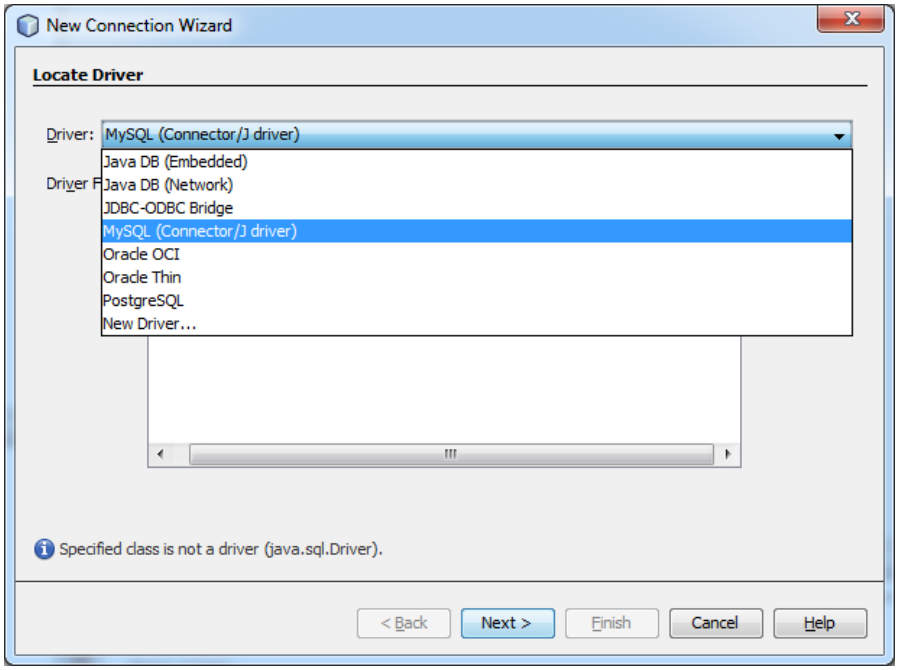


Ilustración 47 Driver Conexión Base de datos

5.-Ahora hacemos click en "Add" y buscamos el conector que nos descargamos posterior mente, después damos click en "Next".

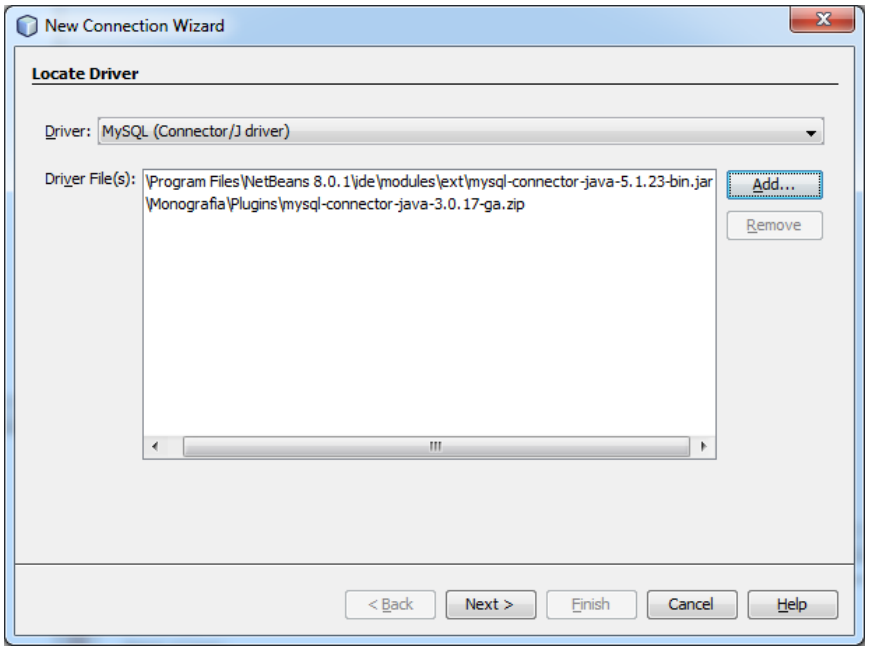


Ilustración 48 Path del driver de conexión

6.- Ahora escribimos las características para la conexión tal y como se configuro la base de datos en Mysql.

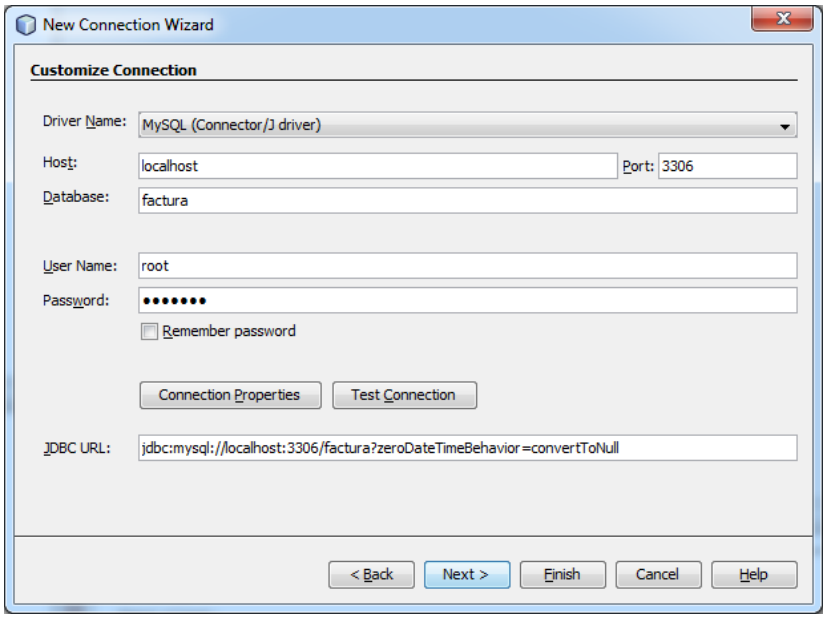


Ilustración 49 Características de la conexión

6.- Procedemos a hacer un test de la conexión para verificar que este bien realizada dicha conexión, si todo está bien nos saldrá un mensaje en la parte izquierda inferior de la ventana diciendo que se ha realizado la conexión, por ultimo damos click en "Finish"

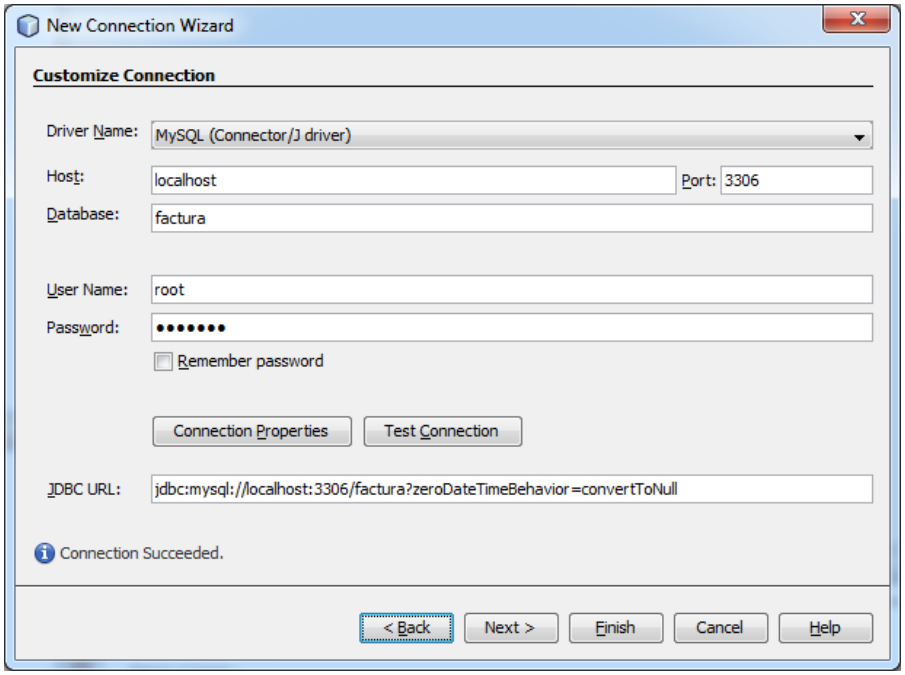


Ilustración 50Prueba de la Conexión

CAPITULO 5

Modelo

5.1 Creación de entidades.

Para generar nuestras entidades automáticamente desde nuestra base de datos realizamos los siguientes pasos.

1.- Sobre nuestro proyecto hacemos click derecho y en la opción “New” ponemos en “Entity Classes from DataBase”

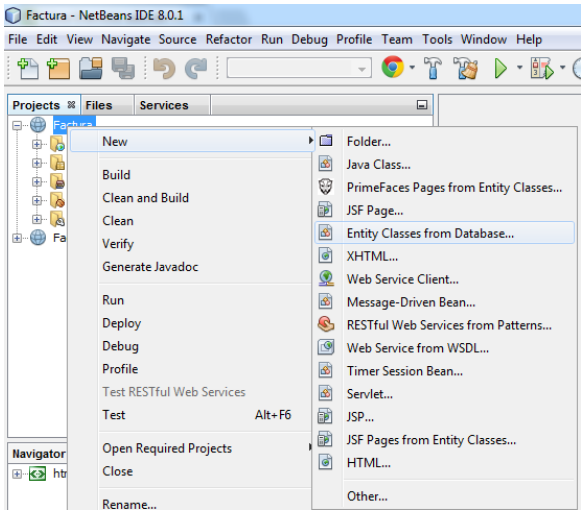


Ilustración 51 Opción Para Crear las entidades

2.- En “Data Sourse” seleccionamos nuestra base de datos, en este caso es “Factura” y movemos todas las tablas que vamos a utilizar al lado derecho, después click en “Next”.

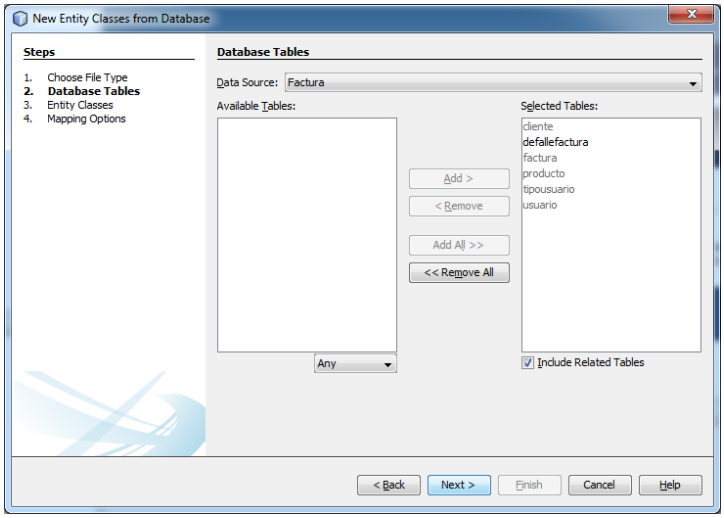


Ilustración 52 Seleccionar Base de datos y entidades.

3.- En la siguiente ventana nos saldrá como se van a llamar nuestras clases para las entidades, si deseamos podemos cambiar, caso contrario se quedara como esta, también tenemos que poner en nombre del paquete que contendrá dichas clases en "Package" en este caso la llamaremos edu.ec.uda.Entidades, y click en "Next".

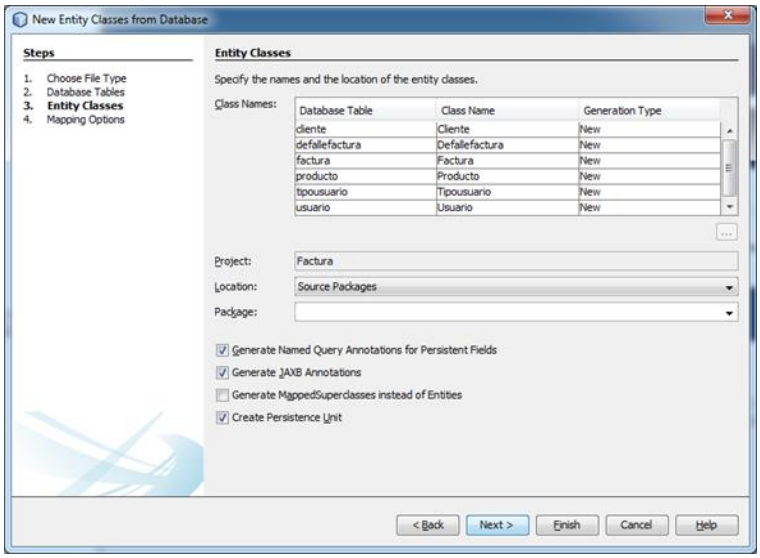


Ilustración 53 Configuración Clases y Paquete

4.- Por ultimo configuramos las opciones mapeo como se muestra a continuación, dejando los valores que se encuentran por default para poder generar consultas, y le damos click en "Finish".

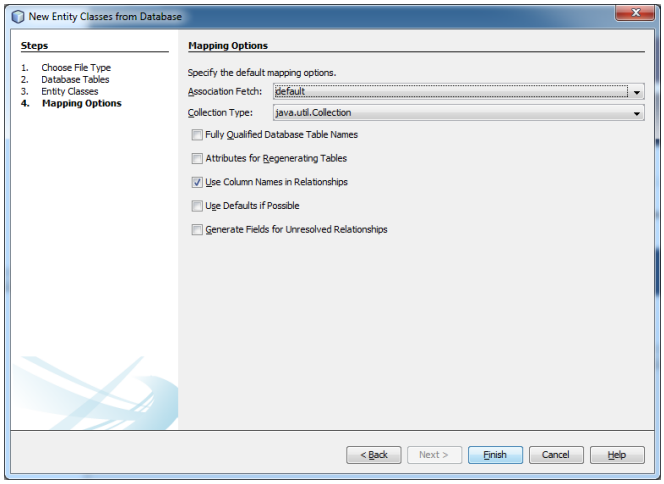


Ilustración 54Opciones de Mapeo

5. Al utilizar este asistente para crear clases de entidad de una base de datos, el IDE examina las relaciones entre las tablas de base de datos. En la ventana de proyectos, si se expande el nodo del paquete ec.edu.uda.entidades, se puede ver que el IDE genera una clase de entidad para cada tabla.

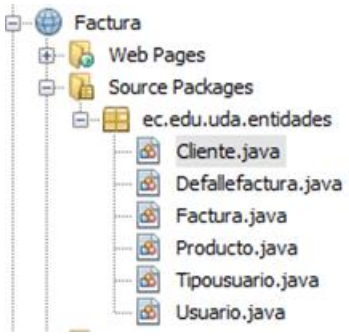


Ilustración 55 Clases Generadas

6.- Si nos fijamos en el código generado para las entidades se puede ver que el asistente añade anotaciones @GeneratedValue a los campos auto generados de ID y las anotaciones @Basic (optional = "false") a algunos de los campos en las entidades.

En base a las anotaciones @Basic (optional = "false"), las páginas que el asistente "Pages from Entity Classes" genera, incluyen un código con controles para prevenir las violaciones para columnas con valor no nulo.

Ahora veremos un ejemplo de la entidad Cliente:

Como podemos ver en la primera parte importa los paquetes que contiene la clase, luego las librerías que va a utilizar para la persistencia de la base de datos, validaciones de los datos, fechas, etc.

También podemos ver que genera unas consultas básicas a la base de datos que se utilizara a continuación en los controladores para las paginas xhtml, también podremos crear más consultas dependiendo de las necesidades que tengamos.

Como se puede observar se genera una clase con propiedades, setters y getters comunes. Para crear la entidad usaremos las anotaciones @Entity, @Table, @Id, @Column y @Temporal de la siguiente manera.

```
package ec.edu.uda.entidades;

import ...20 lines

/**
 *
 * @author BraulioMachuca
 */
@Entity
@Table(name = "cliente")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Cliente.findAll", query = "SELECT c FROM Cliente c"),
    @NamedQuery(name = "Cliente.findByCliId", query = "SELECT c FROM Cliente c WHERE c.cliId = :cliId"),
    @NamedQuery(name = "Cliente.findByCliCedula", query = "SELECT c FROM Cliente c WHERE c.cliCedula = :cliCedula"),
    @NamedQuery(name = "Cliente.findByCliNombre", query = "SELECT c FROM Cliente c WHERE c.cliNombre = :cliNombre"),
    @NamedQuery(name = "Cliente.findByCliApellido", query = "SELECT c FROM Cliente c WHERE c.cliApellido = :cliApellido"),
    @NamedQuery(name = "Cliente.findByCliCiudad", query = "SELECT c FROM Cliente c WHERE c.cliCiudad = :cliCiudad"),
    @NamedQuery(name = "Cliente.findByCliFecNac", query = "SELECT c FROM Cliente c WHERE c.cliFecNac = :cliFecNac"),
    @NamedQuery(name = "Cliente.findByCliTelefono", query = "SELECT c FROM Cliente c WHERE c.cliTelefono = :cliTelefono"),
    @NamedQuery(name = "Cliente.findByCliCelular", query = "SELECT c FROM Cliente c WHERE c.cliCelular = :cliCelular"),
    @NamedQuery(name = "Cliente.findByCliDireccion", query = "SELECT c FROM Cliente c WHERE c.cliDireccion = :cliDireccion"),
    @NamedQuery(name = "Cliente.findByCliMail", query = "SELECT c FROM Cliente c WHERE c.cliMail = :cliMail")})
```

Ilustración 56 Codigo de consultas Generado

```
public class Cliente implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "cli_id")
    private Integer cliId;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 13)
    @Column(name = "cli_cedula")
    private String cliCedula;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "cli_nombre")
    private String cliNombre;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "cli_apellido")
    private String cliApellido;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "cli_ciudad")
    private String cliCiudad;
    @Basic(optional = false)
    @NotNull
    @Column(name = "cli_fec_nac")
    @Temporal(TemporalType.DATE)
    private Date cliFecNac;
    @Size(max = 45)
    @Column(name = "cli_telefono")
    private String cliTelefono;
    @Column(name = "cli_celular")
    private String cliCelular;
    @Size(max = 45)
    @Column(name = "cli_direccion")
    private String cliDireccion;
    @Size(max = 45)
    @Column(name = "cli_mail")
    private String cliMail;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "facCliente")
    private Collection<Factura> facturaCollection;
```

Ilustración 57 Código Clase Cliente

Y por último estarán todos los métodos getters y setters.

```
public Integer getCliId() {
    return cliId;
}

public void setCliId(Integer cliId) {
    this.cliId = cliId;
}

public String getCliCedula() {
    return cliCedula;
}

public void setCliCedula(String cliCedula) {
    this.cliCedula = cliCedula;
}

public String getCliNombre() {
    return cliNombre;
}

public void setCliNombre(String cliNombre) {
    this.cliNombre = cliNombre;
}

public String getCliApellido() {
    return cliApellido;
}

public void setCliApellido(String cliApellido) {
    this.cliApellido = cliApellido;
}

public String getCliCiudad() {
    return cliCiudad;
}
```

Ilustración 58Getters y Setters

En los anexos se colocará todo el código fuente de la aplicación.

CAPITULO 6

Controladores y vistas

6.1 Generando Paginas xhtml desde las entidades.

Ahora que ya hemos creado las entidades, realizaremos la interfaz web para mostrar, crear, modificar y eliminar los datos, es decir un mantenimiento completo.

El código generado por el asistente o wizzard se basa en las anotaciones de persistencia contenidas en las clases de entidad.

1.- Como primer paso para crear nuestros controladores y vistas con primefaces, hacemos click derecho sobre nuestro proyecto, y buscamos la opción “PrimeFaces pages From Entity Classes” dentro la opción “New”.

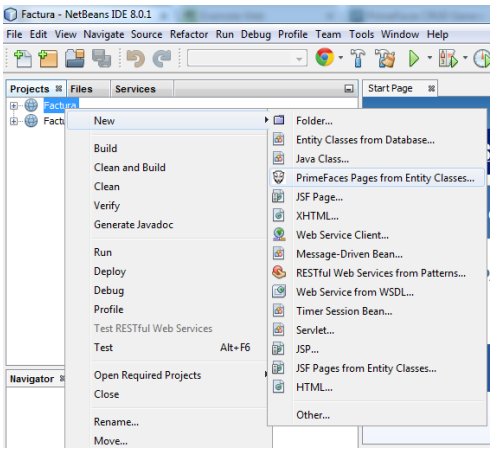


Ilustración 59Opcione para crear las paginas Primefaces

2.- En la siguiente ventana que obtenemos, vamos a agregar todas las entidades de las cuales queremos generar nuestras vistas y controladores, y a continuación ponemos “Next”.

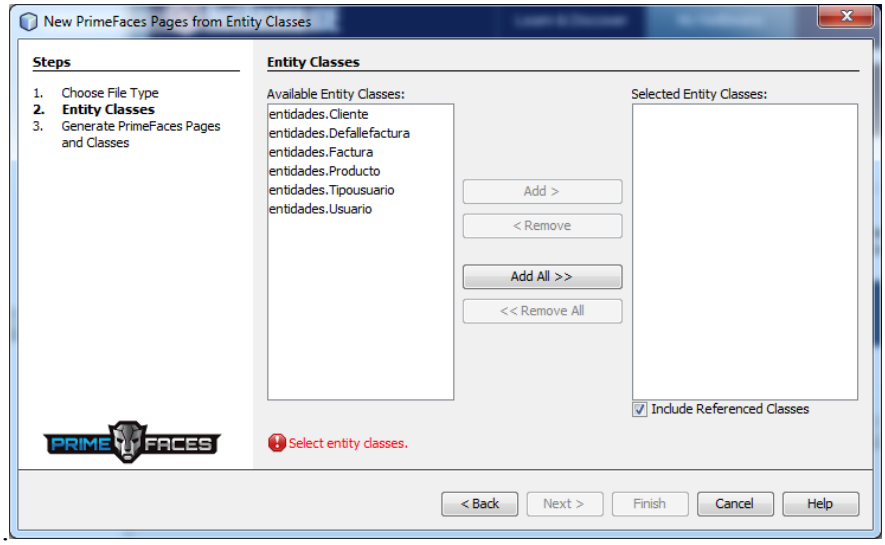


Ilustración 60Entidades para las Páginas

3.- En nuestra siguiente ventana vamos a configurar nuestras páginas y clases.

En las opciones "Session Bean Package", "Backing Bean Package" y "Converter Package" vamos a poner el nombre de la carpeta que va a contener los archivos de controladores, beans y convertidores respectivamente.

En "Primafaces Pages Folder" pondremos el nombre de la carpeta que contendrá las páginas de primefaces.

En la última sección vamos configurar:

"Default row display" cuantas filas de datos se van a mostar en la pantalla, en este caso vamos a dejar 10.

"Default page Selector", vamos a poner cuantas paginas por defecto nos va a mostrar en las tablas.

"Max Table Columns", nos dice cuántas columnas por tabla vamos a mostrar, y si tienen varias ponemos el número máximo que puede mostrar.

"Field name artifacts for foreign fields", aquí vamos a poner los nombres para campos foráneos.

Después en las casillas de verificación señalamos lo que deseamos que tenga nuestra aplicación.

Crear, editar, borrar, ver, ordenar, buscar, notificaciones de mensajes y su tiempo en milisegundos que se va a mostrar, mensajes de información y diálogos de confirmación, menú agregado a la tabla, y navegación por las relaciones.

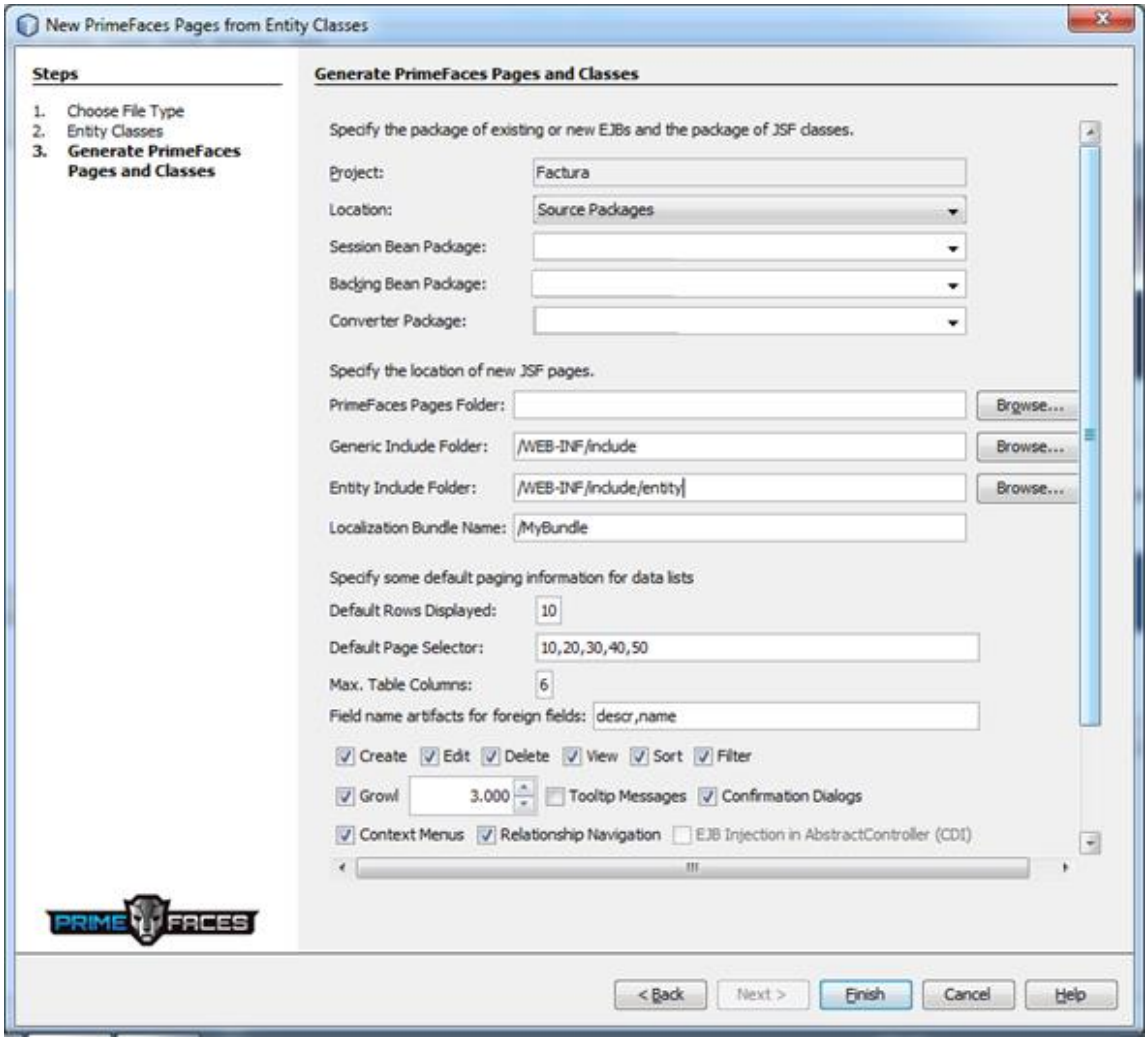


Ilustración 61Configuraciones Para Las Páginas

4. Por último “Finish”.

6.2 Vista.

Para revisar que se han creado con éxito nuestros paginas xhtml nos dirigimos a nuestro proyecto en Netbeans, Factura, y dentro de esa carpeta a la subcarpeta “Web Pages”, y ahí encontraremos todos nuestras páginas xhtml generadas, Cada carpeta contiene los archivos de Create.xhtml, Edit.xhtml, List.xhtml yView.xhtml. También modificó el archivo index.xhtml.

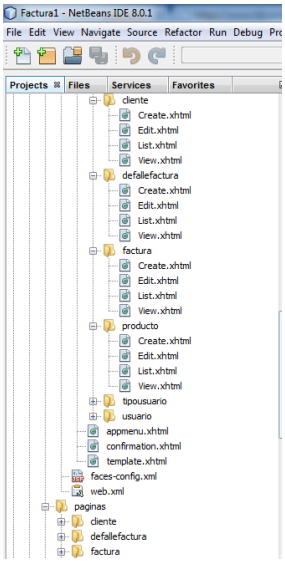


Ilustración 62 Páginas Creadas

La vista será la parte visual que vera el usuario del sistema web, es decir los archivos con extensiones .xhtml.

Para efectos de explicación nos basaremos en el archivo create.xhtml que se encuentra dentro de la carpeta cliente.

Examinaremos la etiqueta inputtext, Encontraremos un código asi:

```
<p:inputText id="cliCedula" value="#{clienteController.selected.cliCedula}"
             title="#{myBundle.CreateClienteTitle_cliCedula}" required="true"
             requiredMessage="#{myBundle.CreateClienteRequiredMessage_cliCedula}" size="13" maxlength="13"/>
```

Ilustración 63 Código InputText

Por ejemplo examinando el atributo "value", determina el texto que se obtendrá este componente, aquí podremos ver cómo hacemos la conexión entre la vista, y el controlador, un atributo del objeto de la clase a la que representara nuestro componente.

6.3 Controladores.

Para revisar que se han creado con éxito nuestros controladores nos dirigimos a nuestro proyecto en Netbeans, y dentro de esa carpeta a la subcarpeta "Source Package", y ahí encontraremos todos nuestros controladores, además un controlador extra, el AbstractController, que todos los controladores acceden a él para realizar la mayoría de las operaciones

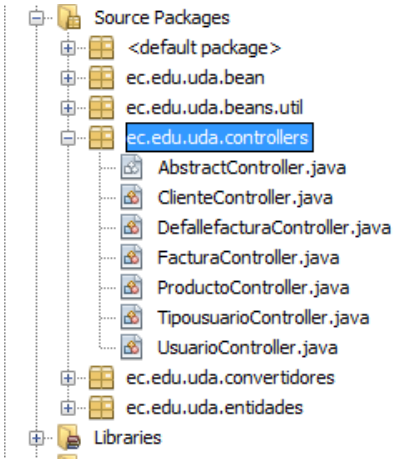


Ilustración 64Controladores Generados

Cada controlador es específico para los cuatro archivos .xhtml, correspondientes e incluye el código que invoca métodos en el bean adecuado.

Por ejemplo el controlador ClienteController, se utiliza para la vista de los archivos, Create.xhtml, Edit.xhtml, List.xhtml y View.xhtml del Cliente.

```
@ManagedBean(name = "clienteController")
@ViewScoped
public class ClienteController extends AbstractController<Cliente> {

    @EJB
    private ClienteFacade ejbFacade;
    @PostConstruct
    @Override
    public void init() {
        super.setFacade(ejbFacade);
        FacesContext context = FacesContext.getCurrentInstance();
    }

    public ClienteController() {
        super(Cliente.class);
    }
    public void resetParents() {
    }
    public String navigateFacturaCollection() {
        if (this.getSelected() != null) {
            FacesContext.getCurrentInstance().getExternalContext().getRequestMap().put("Factura_items", this.getSelected().getFacturaCollection());
        }
        return "/paginas/factura/index";
    }

}
```

Ilustración 65Codigo Controlador

A su vez el controlador también utiliza el bean clienteFacade, que es un patrón de diseño que sirve como puerta de acceso para el uso de las diferentes operaciones.

Este bean además hace una extensión del bean AbstractFacade, que posee los métodos generales entre todos los facade.

```
@Stateless
public class ClienteFacade extends AbstractFacade<Cliente> {
    @PersistenceContext(unitName = "Factura1PU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public ClienteFacade() {
        super(Cliente.class);
    }

}
```

El clienteFacade hace una llamada a la entidad a la que se referenciara en este caso es a la entidad Cliente, que es la que se hace conexión directa con la base de datos.

CAPITULO 7

Primefaces.

7.1 Componentes Primefaces.

7.1.1 InputText

El componente inputText es un elemento de entrada de texto.

Declaración**:**

```
<p:inputText value="#{controlador.ejemplo}" id="ejemplo"/>
```

Ilustración 66 Declaracion InputText

Resultado:

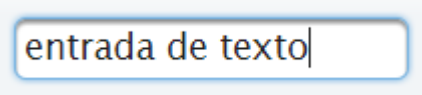


Ilustración 67 InputText

Tabla de Atributos.

Nombre	Default	Tipo	Descripción
Id	null	String	Identificador único del componente.
Rendered	true	Boolean	Indica si el componente esta visible o no.
Value	null	String	Texto que va a mostrar el componente.

Required	null	Boolean	Marca si el componente es requerido o no.
requiredMessage	Null	String	Mensaje que se visualiza después de que la validación ha fallado.
Acceskey	Null	String	Transfiere el foco cuando la llave de acceso es presionada.
Maxlength	null	Integer	Número máximo de caracteres que se permiten en este campo.
Readonly	False	Boolean	Indica si este componente es solo de lectura para el usuario
Title	null	String	Muestra un texto cuando el puntero se posiciona sobre el componente.
Size	null	Integer	Delimita el ancho del componente.

type	text	String	Tipo del elemento de entrada.
Style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.
tabindex	null	Integer	Posición del elemento en el orden de tabulación.
Disabled	false	Boolean	Desactiva el componente.
Onblur	null	String	Ejecución de un método cuando el componente pierde el foco
Onchange	null	String	Ejecución de un método cuando el componente pierde el foco y es cambiado desde que obtuvo el enfoque.
Onclick	null	String	Ejecución de un método cuando el componente de entrada se da click.

Ondblclick	Null	String	Ejecución de un método cuando el componente de entrada se da doble click.
Onfocus	null	String	Ejecución de un método cuando el componente de estrada posee el enfoque.
Onkeypress	null	String	Ejecución de un método cuando se presiona una tecla en el elemento de entrada.
Onkeyup	null	String	Ejecución de un método cuando suelta una tecla sobre el elemento de entrada.
Onkeydown	null	String	Ejecución de un método cuando esta pulsando una tecla sobre el elemento de entrada.
Onmousedown	null	String	Ejecución de un método cuando está pulsando el

			botón del puntero sobre el elemento de entrada.
Onmousemove	null	String	Ejecución de un método cuando se mueve el puntero dentro del elemento de entrada.
Onmouseout	null	String	Ejecución de un método cuando el puntero es alejado del elemento de entrada.
Onmouseover	null	String	Ejecución de un método cuando el puntero se mueve sobre el elemento de entrada.
Onmouseup	null	String	Ejecución de un método cuando el botón del puntero se suelta sobre el elemento de entrada.
Onselect	null	String	Ejecución de un método cuando

			el texto dentro del elemento de entrada esta seleccionado por el usuario.

Tabla 65 Atributos InputText

7.1.2 OutputLabel

El componente outputTextLabel una etiqueta de salida de texto.

7.1.2.1 Declaración:

```
<p:outputLabel  value="Ejemplo:" />
```

Ilustración 68 Declaración OutPut

7.1.2.2 Resultado:

Ejemplo:

Ilustración 69 OutputLabel

7.1.2.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	Null	String	Identificador único del componente.
Rendered	True	Boolean	Indica si el componente esta visible o no.
Value	null	String	Texto que va a mostrar la etiqueta.

For	null	String	Es el id de otro componente al que vamos a atar la etiqueta.
acceskey	Null	String	Transfiere el foco cuando la llave de acceso es presionada.
Onmouseout	null	String	Ejecución de un método cuando el puntero es alejado del componente
Onmouseover	null	String	Ejecución de un método cuando el puntero se mueve sobre componente.
Onmouseup	null	String	Ejecución de un método cuando el botón del puntero se suelta sobre el componente.
Onfocus	null	String	Ejecución de un método cuando el componente posee el enfoque.
Onkeypress	null	String	Ejecución de un método cuando se presiona una

			tecla componente.
onkeyup	null	String	Ejecución de un método cuando suelta una tecla sobre el componente.
Onkeydown	null	String	Ejecución de un método cuando está pulsando una tecla sobre componente.
Onclick	null	String	Ejecución de un método al dar click en el componente.
Ondblclick	Null	String	Ejecución de un método al hacer doble click en el componente
Onblur	null	String	Ejecución de un método cuando el componente pierde el foco
tabindex	null	Integer	Posición del elemento en el orden de tabulación.
Style	null	String	Estilo que tendrá el componente.

styleClass	null	String	Clase con el estilo del componente.
title	null	String	Muestra un texto cuando el puntero se posiciona sobre el componente.

Tabla 66 Atributos outputLabel

7.1.3 CommandButton

7.1.3.1 Declaración.

```
<p:commandButton id="prueba" value="Prueba"/>
```

Ilustración 70 Declaracion CommandButton

7.1.3.2 Resultado.

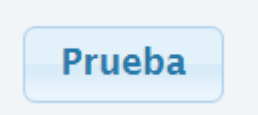


Ilustración 71CommandButton

7.1.3.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	Null	String	Identificador único del componente.
Rendered	True	Boolean	Indica si el componente esta visible o no.

Value	null	String	Texto que va a mostrar el componente.
action	null	Método / String	Método o resultado de cadena que se procesara al hacer click en el botón.
actionListener	Null	Método	Método que se procesa al dar click en el botón.
ajax	True	boolean	Establece el modo de procesar.
acceskey	Null	String	Transfiere el foco cuando la llave de acceso es presionada.
readonly	False	Boolean	Indica si este componente es solo de lectura para el usuario
title	Null	String	Muestra un texto cuando el puntero se posiciona sobre el componente.

Style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.
tabindex	Null	Integer	Posición del elemento en el orden de tabulación.
Disabled	False	Boolean	Desactiva el componente.
Onblur	Null	String	Ejecución de un método cuando el componente pierde el foco
onchange	Null	String	Ejecución de un método cuando el componente pierde el foco y es cambiado desde que obtuvo el enfoque.
onclick	Null	String	Ejecución de un método cuando el componente de entrada se da click.
ondblclick	Null	String	Ejecución de un método cuando el componente

			de entrada se da doble click.
Onfocus	null	String	Ejecución de un método cuando el componente de estrada posee el enfoque.
onkeypress	null	String	Ejecución de un método cuando se presiona una tecla en el elemento de entrada.
Onkeyup	Null	String	Ejecución de un método cuando suelta una tecla sobre el elemento de entrada.
onkeydown	Null	String	Ejecución de un método cuando está pulsando una tecla sobre el elemento de entrada.
onmousedown	null	String	Ejecución de un método cuando está pulsando el botón del puntero sobre el

			elemento de entrada.
onmousemove	Null	String	Ejecución de un método cuando se mueve el puntero dentro del elemento de entrada.
onmouseout	null	String	Ejecución de un método cuando el puntero es alejado del elemento de entrada.
onmouseover	null	String	Ejecución de un método cuando el puntero se mueve sobre el elemento de entrada.
onmouseup	Null	String	Ejecución de un método cuando el botón del puntero se suelta sobre el elemento de entrada.
onselect	Null	String	Ejecución de un método cuando el texto dentro del elemento de

			entrada esta seleccionado por el usuario.
process	null	String	Componentes para procesar parcialmente en lugar de hacer todo.
update	null	String	Componentes que se actualizarán con Ajax.
icon	null	String	Imagen que se mostrara como icono del botón.
iconPos	Left	String	Posición del icono.

Tabla 67Atributos CommandButon

7.1.4 Panel

Panel es un componente de agrupación de elementos.

7.1.4.1 Declaración

```
<p:panel header="Ejemplo" closable="true"    >

</p:panel>
```

Ilustración 72 Declaración Panel

Tabla 68 Declaración Panel

7.1.4.2 Resultado

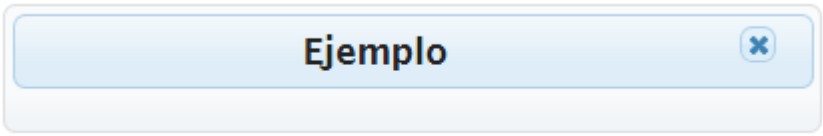


Ilustración 73Panel

7.1.4.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	Null	String	Identificador único del componente.
rendered	True	Boolean	Indica si el componente esta visible o no.
header	null	String	Texto de la cabecera.
Footer	null	String	Texto en el pie de página.
Toggleable	false	Boolean	Hace que el panel se minimice.
Style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.
Closable	false	boolean	Hace que el panel se pueda cerrar.
visible	true	Boolean	Hace que el panel se haga

			visible o invisible.
closeTitle	null	String	Título que aparece cuando se posiciona el puntero sobre el botón cerrar.
toggleTitle	null	String	Título que aparece cuando se posiciona el puntero sobre el botón minimizar.
toggleOrientation	vertical	String	Define la orientación de los botones del menú, estos pueden ser vertical u horizontal.

Tabla 69 Atributos Panel

7.1.5 Menubar

Menubar es un componente menú de navegación vertical.

7.1.5.1 Declaración

```
<p:menubar>
    <p:menuitem value="#{myBundle.Home}" outcome="/index" icon="ui-icon-home"/>
    <p:submenu label="#{myBundle.Maintenance}">
        <p:menuitem value="#{myBundle.ClienteHeading}..." outcome="/cliente/index" />
        <p:menuitem value="#{myBundle.FacturaHeading}..." outcome="/factura/index" />
        <p:menuitem value="#{myBundle.ProductoHeading}..." outcome="/producto/index" />
        <p:menuitem value="#{myBundle.UsuarioHeading}..." outcome="/usuario/index" />
    </p:submenu>
    <p:submenu label="#{myBundle.Reportes}">
        <p:menuitem value="#{myBundle.ReporteStockHeading}..." outcome="/reportes/index" />
        <p:menuitem value="#{myBundle.ReporteProductoHeading}..." outcome="/reportes/index1" />
        <p:menuitem value="#{myBundle.ReporteVentasMesHeading}..." outcome="/reportes/index2" />
    </p:submenu>
    <p:menuitem value="#{myBundle.CerrarSesion}"  actionListener="#{authBean.logout}" icon="ui-icon-power" />
</p:menubar>
```

Ilustración 74 Declaración MenuBar

7.1.5.2 Resultado

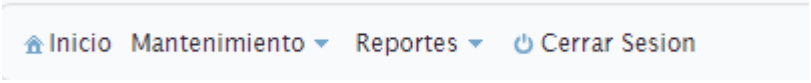


Ilustración 75 MenuBar

7.1.5.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	null	String	Identificador único del componente.
rendered	True	Boolean	Indica si el componente esta visible o no.
autodisplay	true	Boolean	Define si el primer nivel de submenús se mostrara cuando se apunte con mouse o es necesario hacer click.
Style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.

Tabla 70 Atributos MenuBar

7.1.6 MenuItem

MenuItem es usado por varios componentes de menú.

7.1.6.1 Declaración

```
<p:menuitem value="#{myBundle.Home}" outcome="/index" icon="ui-icon-home"/>
```

Ilustración 76 Declaración MenuItem

7.1.6.2 Resultado

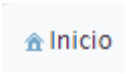


Ilustración 77 MenuItem

7.1.6.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	null	String	Identificador único del componente.
Rendered	True	Boolean	Indica si el componente esta visible o no.
action	null	Método / String	Método o resultado de cadena que se procesara al hacer click en el botón.
actionListener	Null	Método	Método que se procesa al dar click en el botón.
Ajax	True	boolean	Establece el modo de procesar.
title	null	String	Muestra un texto cuando el puntero se

			posiciona sobre el componente.
Process	null	String	Componentes para procesar parcialmente en lugar de hacer todo.
Update	null	String	Componentes que se actualizarán con Ajax.
icon	null	String	Imagen que se mostrara como icono del botón.
Outcome	null	String	Para navegar a otra página del sistema.

Tabla 71 Atributos MenuItem

7.1.7 SubMenu

Submenú es un componente que se anida en los componentes del menú y representa un de submenú.

7.1.7.1 Declaración.

```
<p:submenu label="#{myBundle.Maintenance}">
    <p:menuitem value="#{myBundle.ClienteHeading}..." outcome="/cliente/index" />
    <p:menuitem value="#{myBundle.FacturaHeading}..." outcome="/factura/index" />
    <p:menuitem value="#{myBundle.ProductoHeading}..." outcome="/producto/index" />
    <p:menuitem value="#{myBundle.UsuarioHeading}..." outcome="/usuario/index" />
</p:submenu>
```

Ilustración 78 Declaración SubMenu

7.1.7.2Resultado

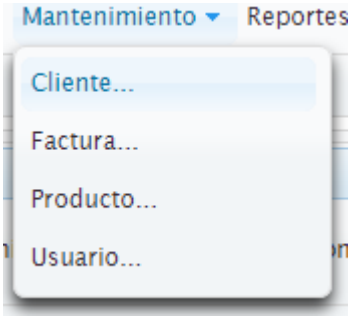


Ilustración 79 Submenu

7.1.7.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	Null	String	Identificador único del componente.
rendered	True	Boolean	Indica si el componente esta visible o no.
label	null	String	Texto de la cabecera del submenu
icon	null	String	Imagen que se mostrara como icono del botón.
outcome	null	String	Para navegar a otra página del sistema.
Style	null	String	Estilo que tendrá el elemento de entrada.

styleClass	null	String	Clase con el estilo del componente.

Tabla 72Atributos Submenu

7.1.8 DataTable

Muestras los datos en una tabla de forma tabulada.

7.1.8.1 Declaración.

```
<p:dataTable id="datalist"
             value="#{clienteController.items}"
             var="item"
             rowKey="#{item.cliId}"
             paginator="true"
             rows="10"
             rowsPerPageTemplate="10,20,30,40,50"
             selectionMode="single"
             selection="#{clienteController.selected}" >
```

Ilustración 80 Declaración DataTable

7.1.8.2 Resultado

Id	Cedula	Nombre	Apellido	Ciudad	Fecha de Nacimiento
2	0102155579	Ofelia Valeria	Machuca Rubio	Cuenca	05/25/1975
4	0105443691	Karla Estefania	Carreno Aquilar	Cuenca	08/06/1995
5	01049662721	PABLO ISRAEL	VEGA SERPA	TAMBO	05/08/1984

Ilustración 81 DataTable

7.1.8.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	null	String	Identificador único del componente.

Rendered	true	Boolean	Indica si el componente esta visible o no.
Value	null	Object	Datos a mostrar.
Var	null	String	Nombre de la variable que se utilizara para referirse a cada dato.
rowKey	null	String	Es el identificador único de las filas.
Paginator	false	Boolean	Permite paginación en la tabla.
Rows	null	Integer	Numero de filas que se mostrar por página en la tabla.
rowsPerPageTemplate	null	String	Filas por páginas que se van a mostrar.
selectionMode	null	String	Permite la selección de filas, los valores permitidos son “single” y “multiple”
selection	null	Object	Referencia al dato seleccionado.

Scrollable	false	Boolean	Hace los datos de la tabla desplegable con cabecera fija.
scrollHeight	null	Integer	Alto de la ventana de scroll.
scrollWidth	null	Integer	Ancho de la ventana de scroll.
rowsPerPageLabel	null	String	Etiqueta para las filas desplegables por página.
Style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.
sortBy	null	Object	Se utiliza para order por un campo por defecto la tabla.
sortOrder	ascending	String	Ordena en forma ascendente o descendente.
editingRow	false	Boolena	Define si se pueden mostrar editores de filas o no.

Tabla 73Atributos DataTable

7.1.9 Column

Define las columnas de los componentes como dataTable.

7.1.9.1 Declaración.

```
<p:column sortBy="#{item.cliId}" filterBy="#{item.cliId}"
          headerText="#{myBundle.ListClienteTitle_cliId}" >
    <h:outputText value="#{item.cliId}"/>
</p:column>
```

Ilustración 82Declaracion Column

7.1.9.2 Resultado

Ilustración 83Column

7.1.9.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	Null	String	Identificador único del componente.

Rendered	True	Boolean	Indica si el componente esta visible o no.
sortBy	Null	Expresión de valor	Valor de la expresión que se utiliza para ordenar.
filterBy	Null	Expresión de valor	Valor de la expresión que utiliza para filtrar la columna.
filterMaxLength	Null	Integer	Número máximo de caracteres para el filtro.
headerText	Null	String	Etiqueta de la cabecera de la columna.
footerText	Null	String	Etiqueta del pie de página de la columna.
width	Null	String	Ancho de la columna en pixeles o porcentaje.
disabledSelection	false	Boolean	Desactiva la selección de filas.
style	null	String	Estilo que tendrá el elemento de entrada.

styleClass	null	String	Clase con el estilo del componente.

Tabla 74Atributos Column

7.1.10 Chart

Chart es un componente grafico genérico que sirve para que sirve para crear varios tipos de gráficos estadísticos.

7.1.10.1 Declaración.

```
<p:chart type="bar" id="reporte" model="#{chartView.barModel}" />
```

Ilustración 84 Declaración Chart

7.1.10.2 Resultado

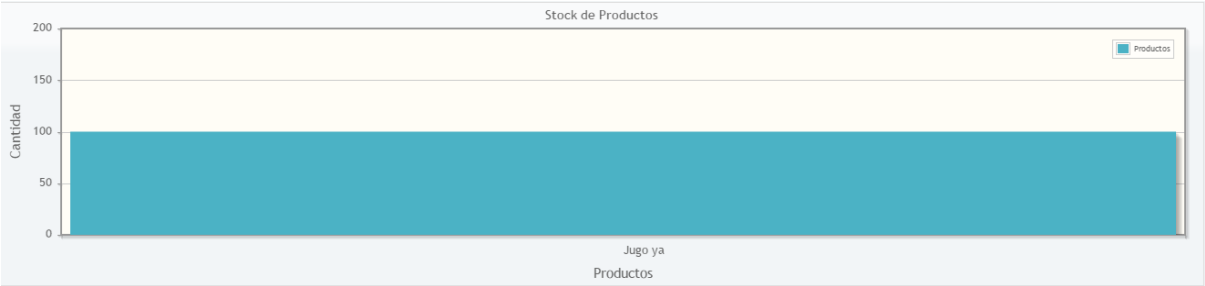


Ilustración 85Chart

7.1.10.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	Null	String	Identificador único del componente.
Rendered	True	Boolean	Indica si el componente esta visible o no.

type	Null	String	Tipo de grafico, puede ser, pie, bar, line, etc
model	Null	ChartModel	Objeto de datos y configuración del gráfico.
style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.

Tabla 75Atributos Chart

7.1.11Slider

Slider se utiliza para proporcionar una entrada numérica en forma deslizante.

7.1.11.1 Declaración.

```
<p:inputText id="numPro" value="#{chartView.numProductos}"  />
<p:slider for="numPro" animate="true"   minValue="1" maxValue="#{chartView.maxProductos}">
 </p:slider>
```

Ilustración 86Declaracion Slider

7.1.11.2 Resultado.

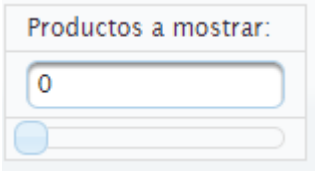


Ilustración 87Slider

7.1.11.3 Tabla de atributos.

Nombre	Default	Tipo	Descripción
Id	null	String	Identificador único del componente.

Rendered	true	Boolean	Indica si el componente esta visible o no.
for	null	String	Id del texto de entrada que el control deslizante utilizara.
minValue	0	Integer	Valor mínimo del componente
maxValue	100	Integer	Valor máximo del componente.
Type	horizontal	String	Establece la orientación del componente. Ya sea vertical u hrizontal.
Step	1	Integer	Incremento del componente al moverse.
Disabled	true	Boolean	Desactiva o activa el componente.
Range	false	Boolean	Cuando se activa, se proporcionan dos límites para la selección de un rango.

Style	null	String	Estilo que tendrá el elemento de entrada.
styleClass	null	String	Clase con el estilo del componente.

Tabla 76Atributos Slider

CONCLUSIONES.

Dentro de cualquier sistema es necesario realizar el análisis de los factores que determinan el cumplimento eficaz de los objetivos, es así que he llegado al final del proyecto, estableciendo algunos factores que han influencia en el proceso de desarrollo del sistema.

El modelo utilizado en este sistema ha facilitado el desarrollo de esta aplicación web, ya que al utilizar la arquitectura modelo, vista controlador, se hace totalmente independiente el desarrollo del sistema.

Gracias al uso de JSF y de los numerosos componentes Primefaces han permitido un manejo adecuado en lo que se refiere a validaciones, mantenimiento de datos, ahorro de código fuente, manejo de menús, gráficos, pero sobre todo la facilidad de uso e implementación de los componentes.

En cuanto a costos del desarrollo se ha minimizado al máximo ya que se utiliza componentes y librerías libre de descarga legal.

Con estos todos estos factores enunciados anteriormente podemos decir que un desarrollo de un sistema web bien podría ser un buen candidato JSF con la librería primefaces para el desarrollo del mismo.

Es manual brindara una gran ayuda y soporte para los usuarios que se inclinen en el desarrollo web con con java y todos sus componentes mencionas en esta monografía.

REFENCIAS BIBLIOGRÁFICAS

Coronel, G. (2010). *Lenguaje de programacion Java.* Lima,Perú: Etitorial Macro.

Fernando Pech-May, ,. M.-R.-D.-J. (2013). *Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces.* Tabasco, México.

Java. (1 de 4 de 2014). *Java*. Obtenido de Java: https://www.java.com/es/download/faq/whatis_java.xml

Luis Alberto Casillas Santillán, M. G. (1 de 12 de 2012). *Base de Datos en MySQL.* Catalu: Universitat Oberta de Catalunya. Obtenido de Universitat Oberta de Catalunya.

Mert Caliskan, O. V. (1 de 1 de 2013). *PrimeFaces Cookbook.* Packt Publishing Ltd. Obtenido de PrimeFaces.

Yenisleidy Fernández Romero, Y. D. (2012). Patrón Modelo-Vista-Controlador. *Telemática* , 47-50.

Anexos

Anexo #1

Manual de usuario de la Aplicación.
Manual de usuario.

Esta será la primera pantalla que se mostrara al usuario al ingresar al sistema, el usuario deberá estar registrado anterior mente en el sistema, aquí pondrá los datos del nombre de usuario y la respectiva contraseña.

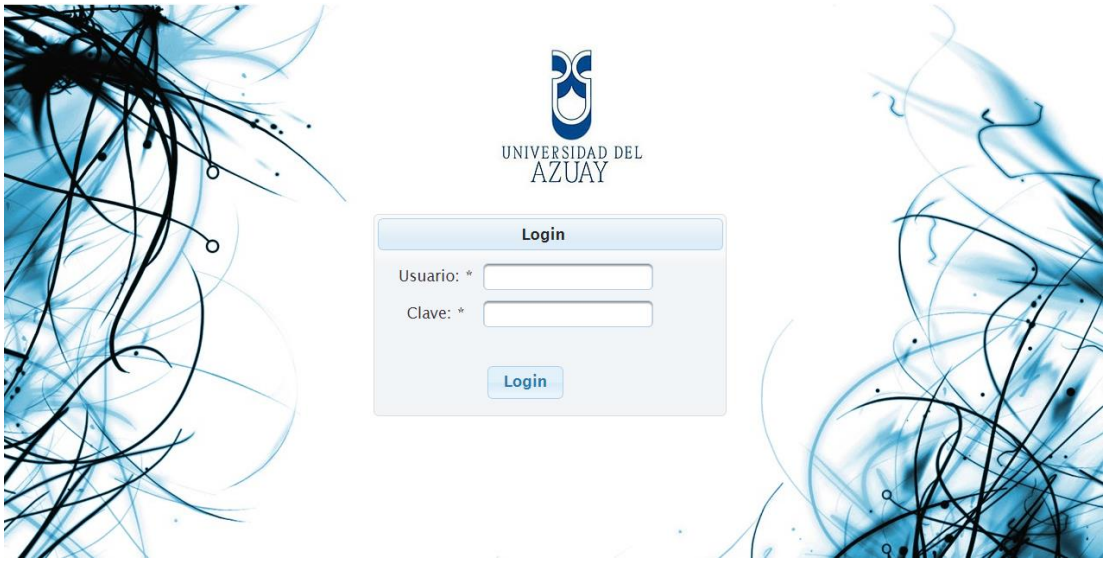


Ilustración 88 Logueo del Sistema

Si el usuario logra ingresar con éxito el sistema mostrara una pantalla con el menú en la parte superior izquierda de la pantalla, y en la parte derecha mostrara el nombre del usuario que esta logueado en el sistema.

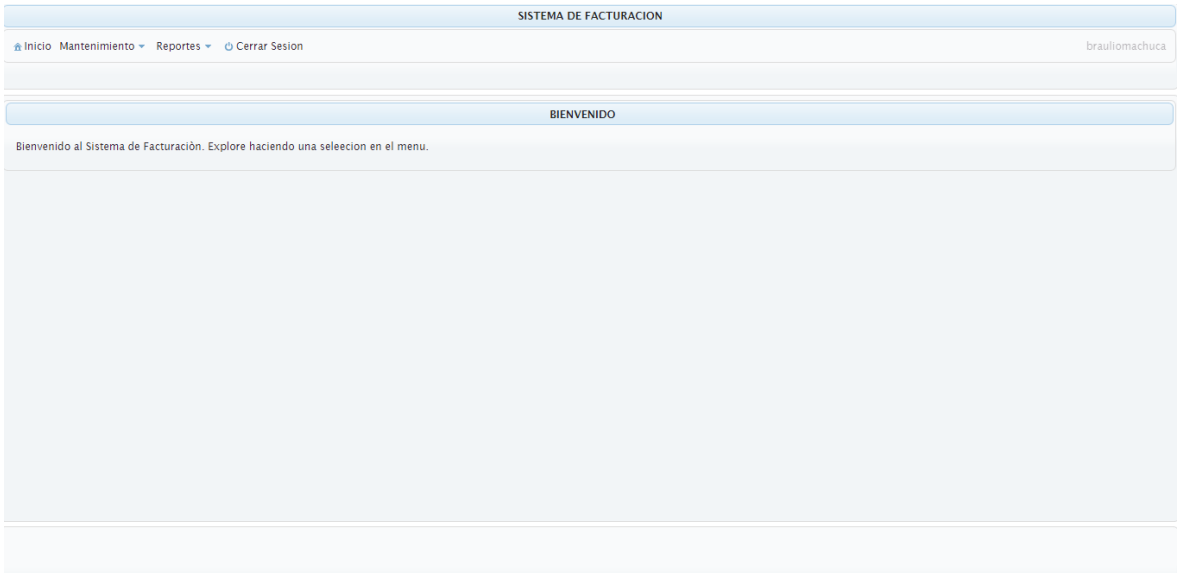


Ilustración 89 Pantalla Principal del Sistema

Caso contrario si el usuario no tiene problemas para ingresar se le mostrara un mensaje explicando el problema ocurrido.

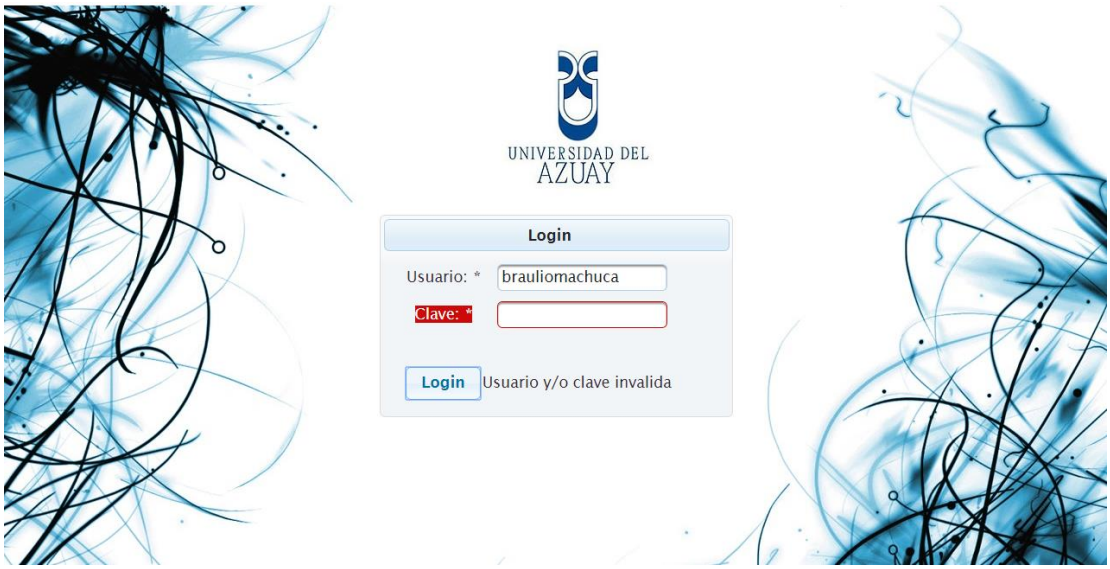


Ilustración 90 Error En el Loguin

En nuestra opción en menú esa el mantenimiento, en donde podremos acceder a cada uno de los módulos del sistema, que son cliente factura, producto o usuario.

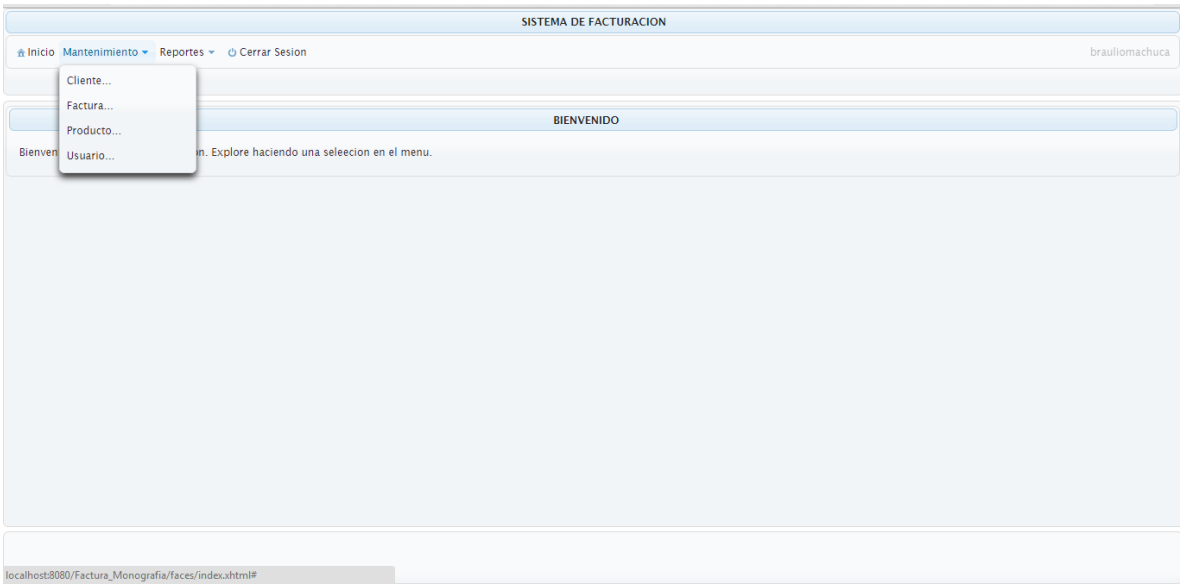


Ilustración 91 Menú del Sistema

Mantenimiento clientes.

Al hacer click en el menú cliente tendremos una pantalla con una lista de todos los clientes ingresados anteriormente.

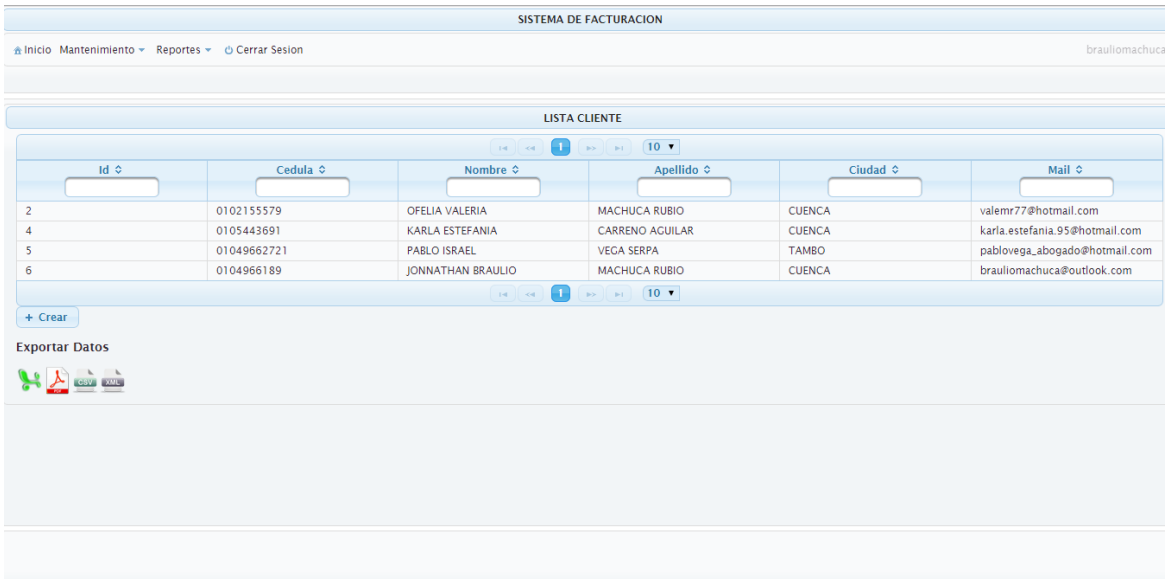


Ilustración 92 Lista de clientes

Aquí podemos filtrar a los clientes según los campos expuestos en la tabla.

También podremos exportar todos los datos que se encuentran en nuestra tabla a formatos Excel, pdf, csv, y xml.

Ver datos de un cliente.

Para ver todos los datos de un cliente, daremos doble click sobre el cliente que deseamos o click derecho y luego en la opción ver.

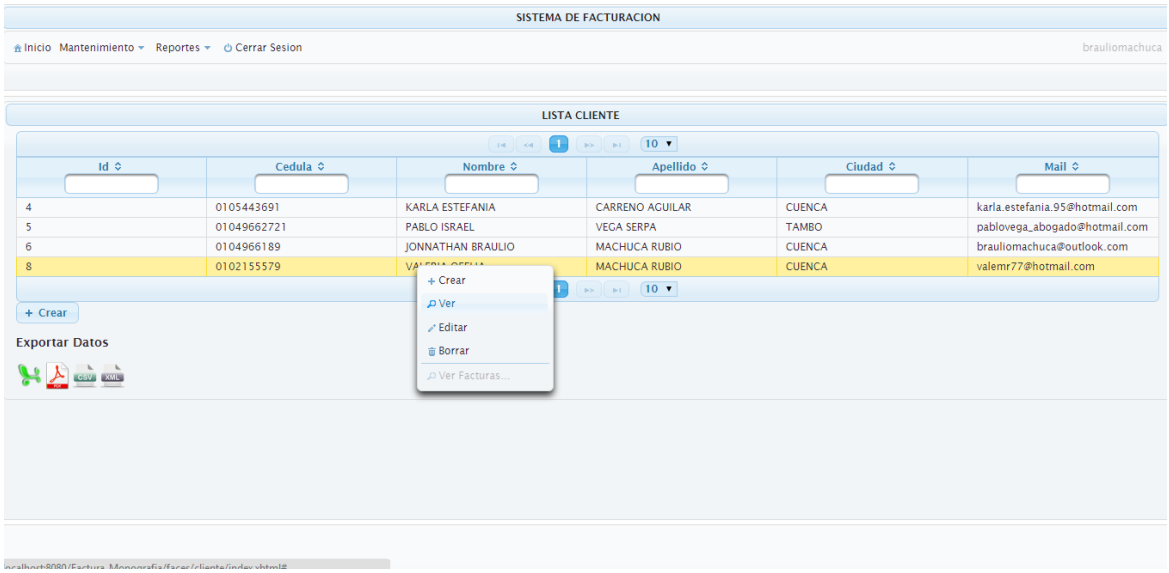


Ilustración 93 Opciones de menú en el cliente Ver

Crear cliente.

Para crear un nuevo cliente podremos hacer click directamente en el botón crear o hacer click derecho sobre un registro y podremos y seleccionar la opción crear.

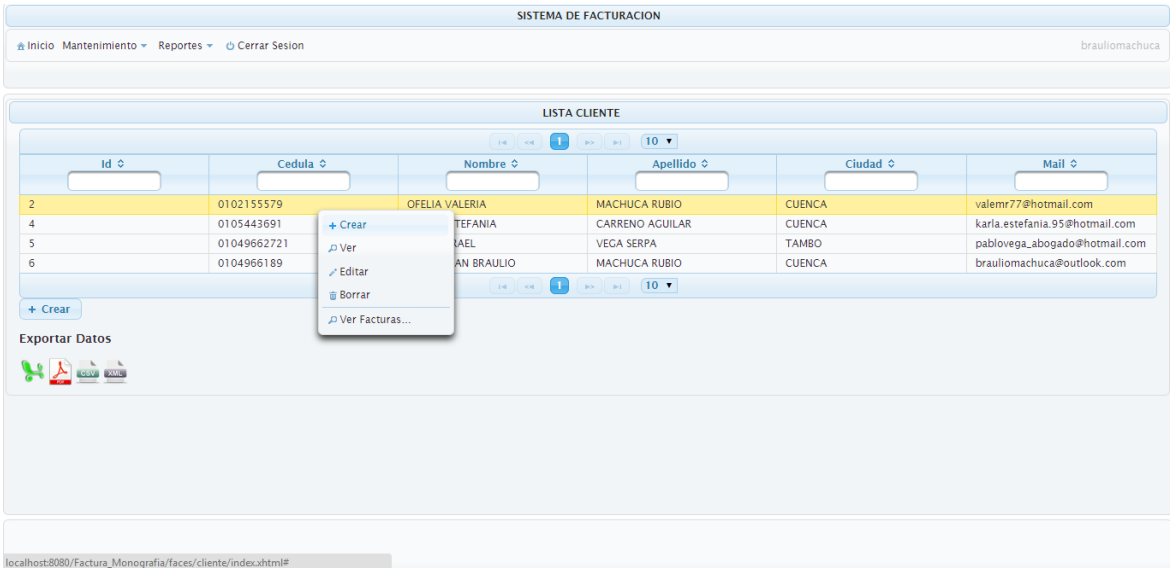


Ilustración 94 Opciones de menú en el cliente Crear

Despues de hacer click en la opción crear obtendremos un formulario para el registro del cliente asi.

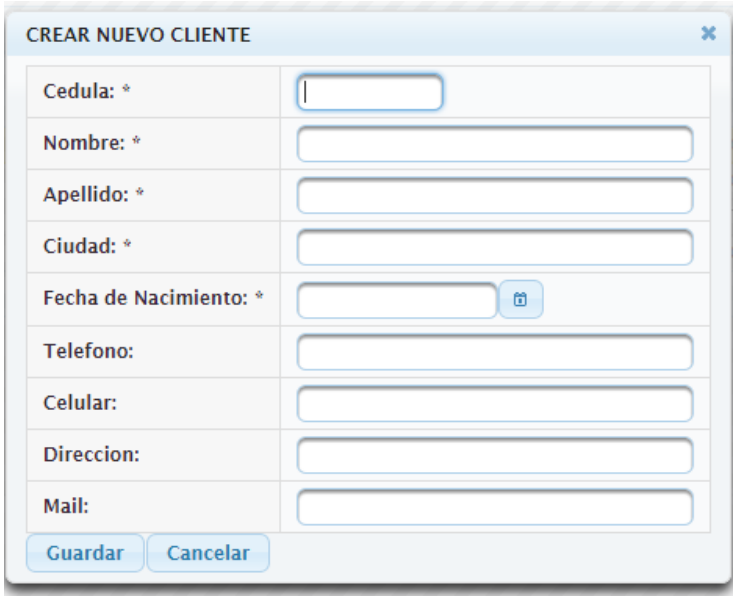


Ilustración 95Formulario de ingreso Cliente

Luego de llenar el formulario debemos hacer click en el botón guardar, nos saldrá un cuadro de confirmación.

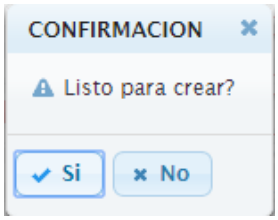


Ilustración 96Mensaje de confirmación

Hacemos click en si, si estamos listos.

Si hemos completado con éxito los datos, direccionara a la lista de clientes actualizados con una notificación indicando que

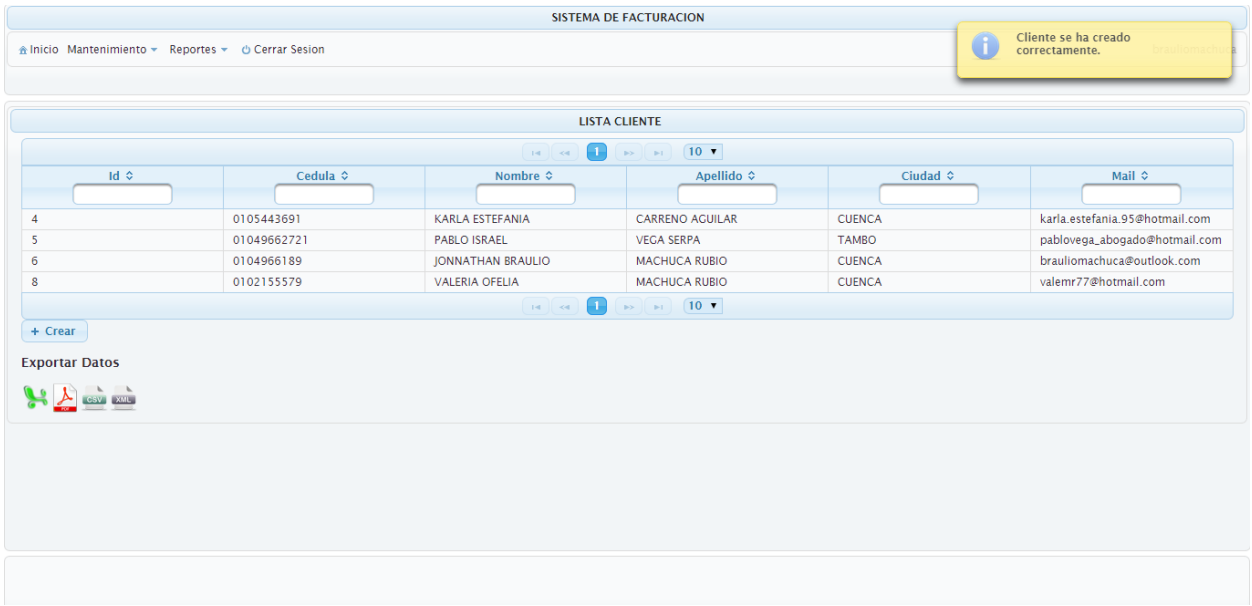


Ilustración 97 Notificación de Guardado correctamente

Caso contrario el sistema nos indicara que esta mal si no hemos llenado los campos correctamente.

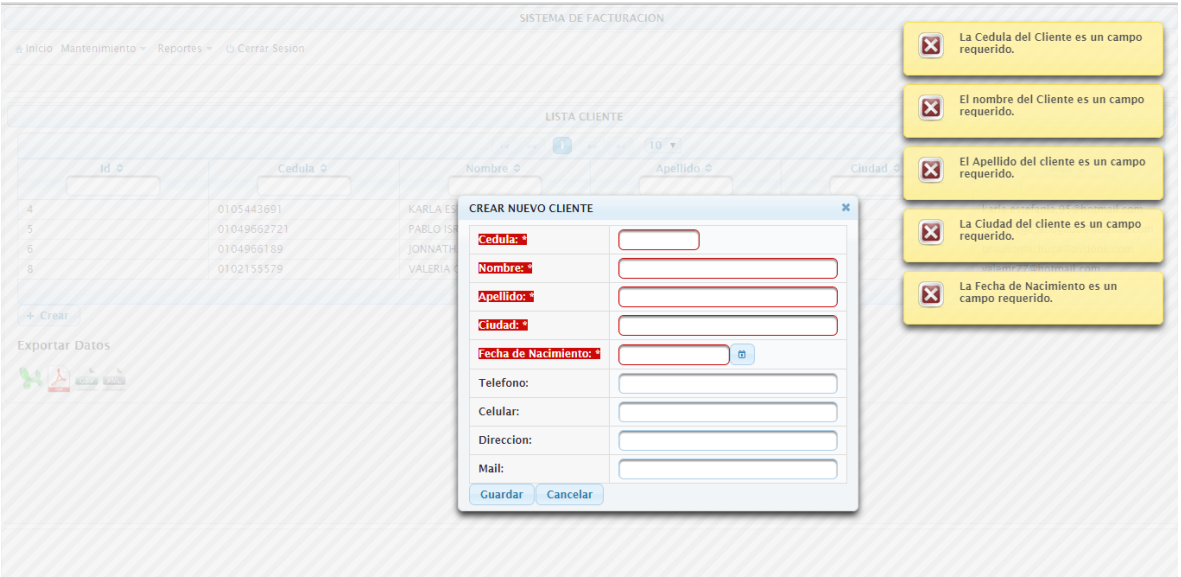


Ilustración 98Error en el guardado del cliente

Los campos con asterisco son obligados a llenar.

Editar cliente.

Para modificar el cliente, una listado los clientes hacemos click derecho sobre el campo que deseamos editar y luego click en editar.

Ilustración 99Formulario de edición de cliente

Podremos editar todos los campos con excepción del id, luego guardaremos los datos.

Al igual que al crear un nuevo cliente, también mostrara el cuadro de dialogo para confirmar,

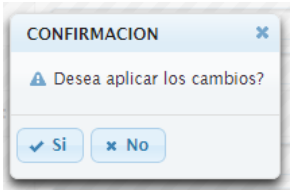


Ilustración 100 Confirmación de la edición

y si está correcto nos mostrara una notificación que se ha modificado correctamente

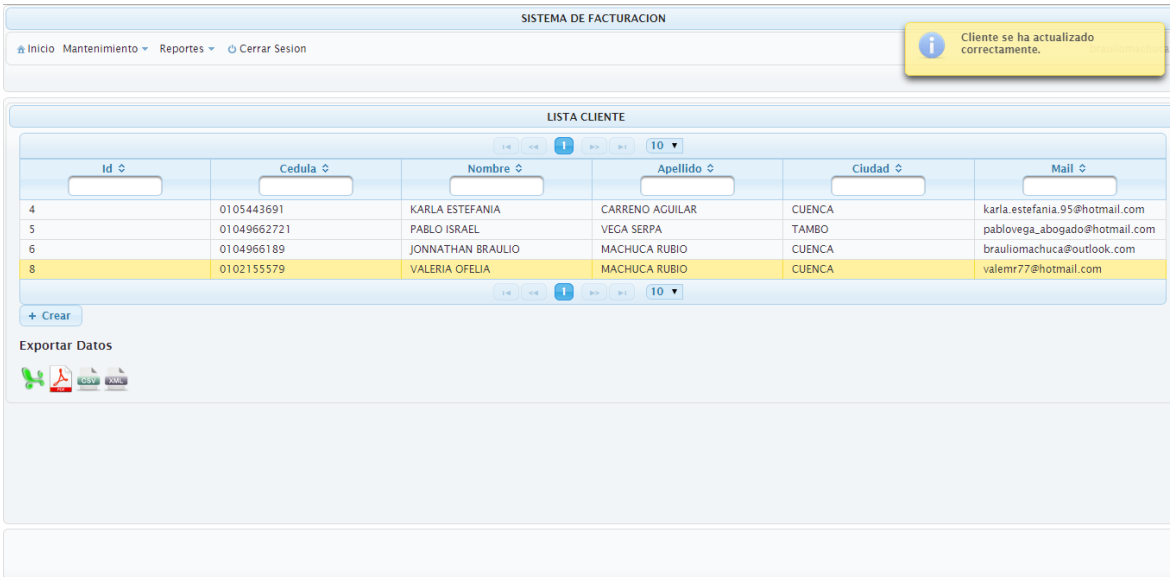


Ilustración 101 Notificación de Edición Correcta

caso contrario nos indicara que está mal.

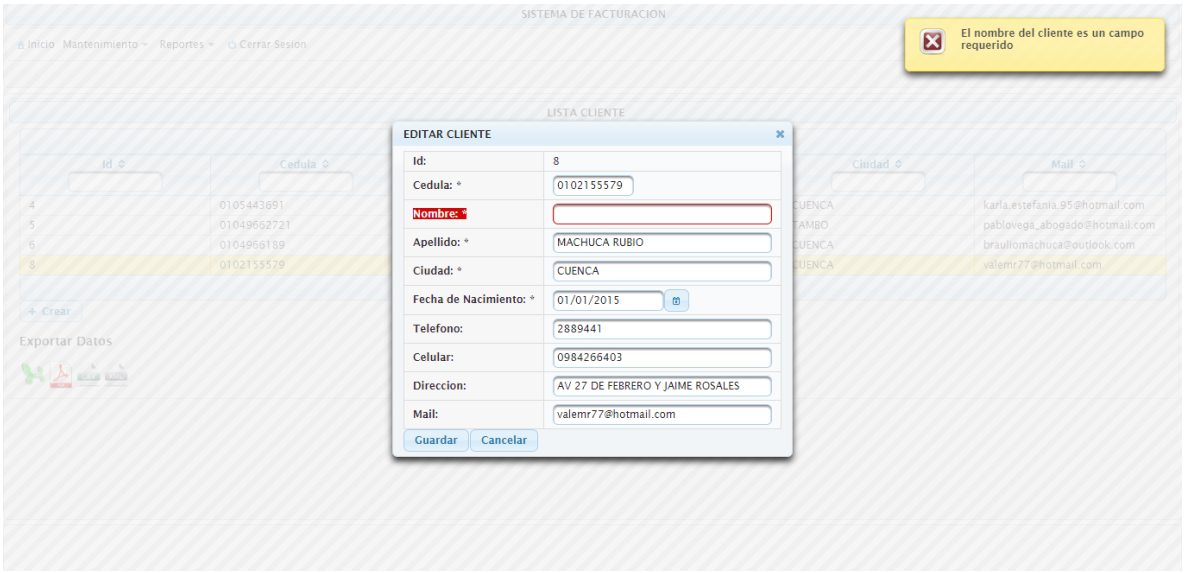


Ilustración 102 Notificación de error en la edición

Borrar cliente.

Después de listar los clientes, hacemos click derecho sobre el cliente que vamos a elimiar y hacemos click en la opción borrar. ‘

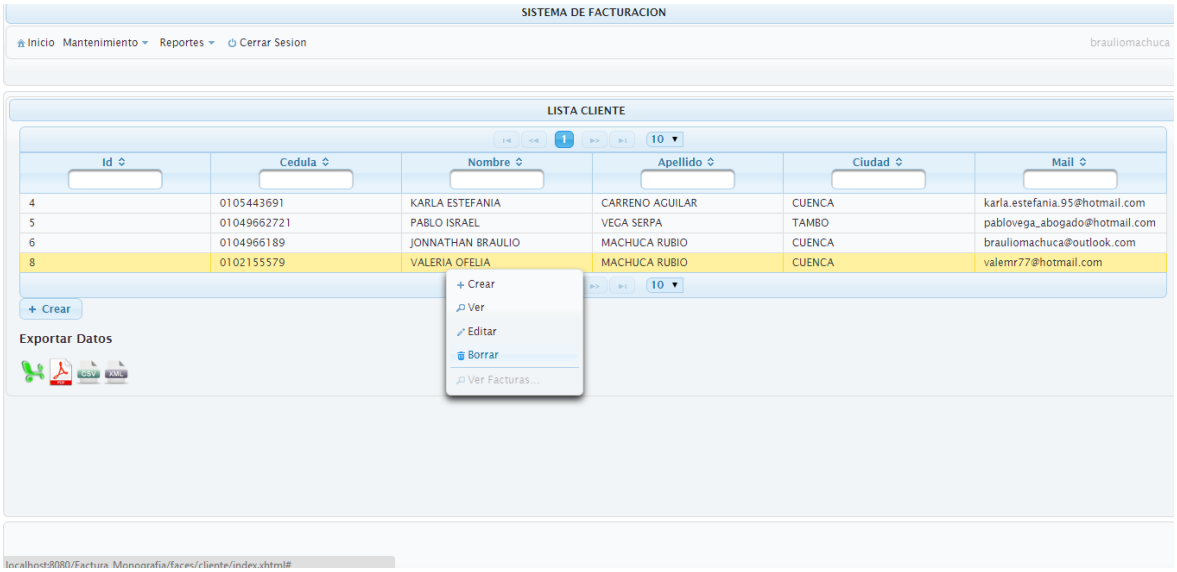


Ilustración 103Menu de opciones Cliente Borrar

Obtendremos un cuadro de confirmación para borrar el cliente

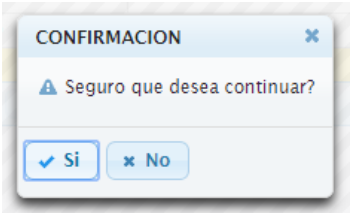


Ilustración 104 Confirmación de eliminación.

Mantenimiento de usuario.

Listar usuarios.

Para listar usuarios hacemos click en el menú principal y luego en la opción usuarios.

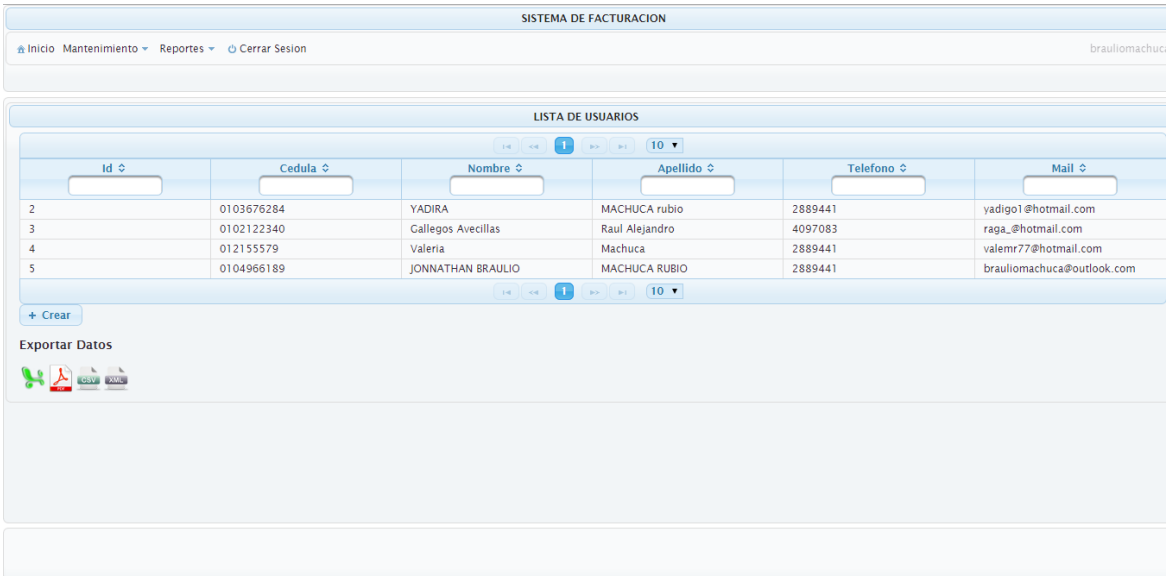


Ilustración 105Listado de usuarios

Aquí podremos filtrar los campos para buscar a un usuario.

Crear usuario

Para crear a un usuario, una vez listados podremos hacer click directamente en el botón crear o hacer click derecho sobre cualquier campo de la lista y escoger la opción crear.

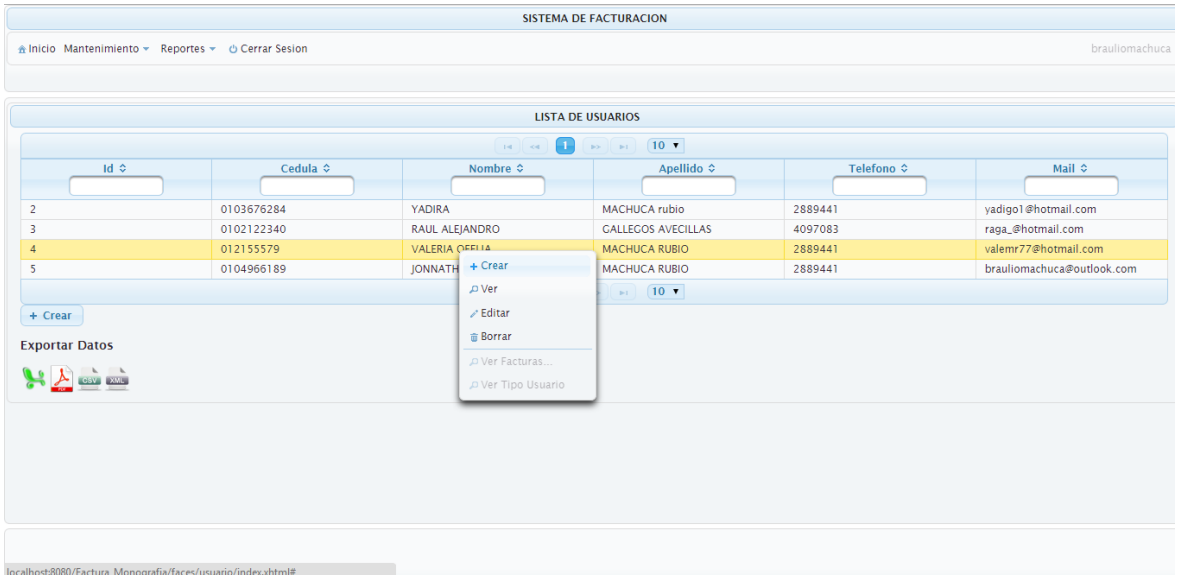


Ilustración 106Opciones de menu Usuario Crear

El sistema mostrara un formulario para el ingreso de los usuarios.

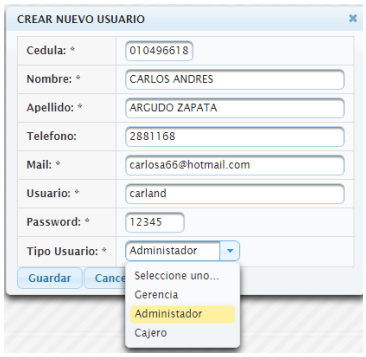


Ilustración 107Formulario ingreso Usuario

Aquí seleccionaremos el tipo de usuario, tomando en cuenta que el administrador puede acceder a todos los módulos, el cajero solo podrá acceder a los módulos de cliente y factura, y el gerente no podrá acceder a los reportes estadísticos.

Luego de llenar el formulario debemos hacer click en el botón guardar, nos saldrá un cuadro de confirmación.

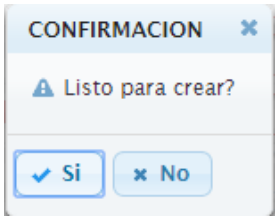
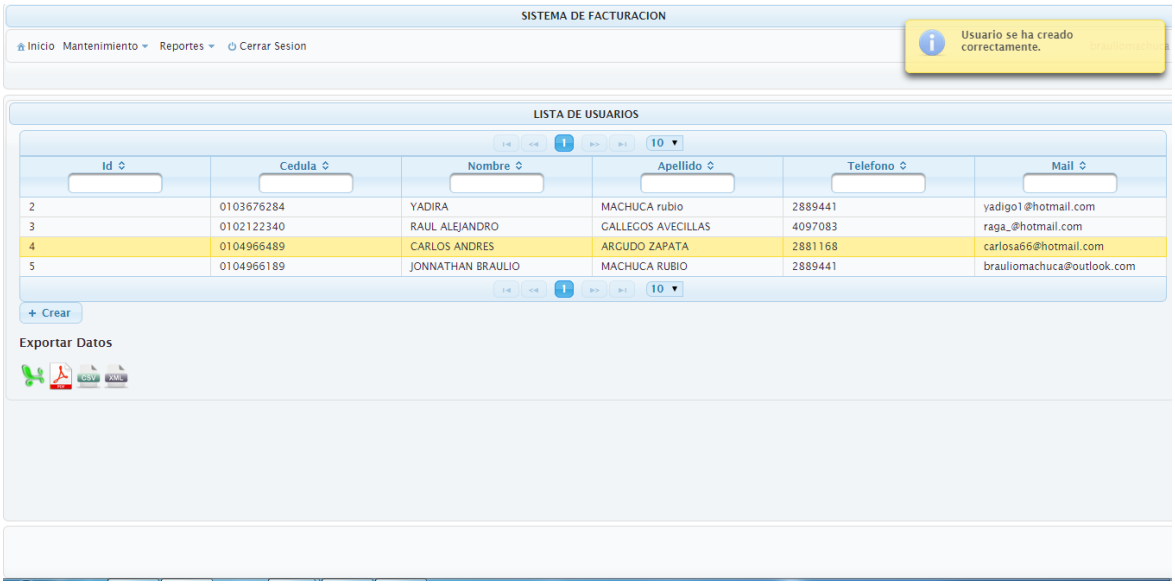


Ilustración 108 Confirmación de ingreso Usuario

Hacemos click en si, si estamos listos.

Si hemos completado con éxito los datos, direccionara a la lista de usuarios actualizados con una notificación indicando que nos indica que se creo correctamente.



Editar usuario.

Para modificar el usuario, una vez listado los usuarios hacemos click derecho sobre el usuario que deseamos editar y luego click en editar.

Ilustración 109 Formulario de edicion de Usuario

Podremos editar todos los campos con excepción del id, luego guardaremos los datos.

Al igual que al crear un nuevo usuario, también mostrara el cuadro de dialogo para confirmar,

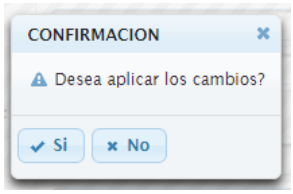
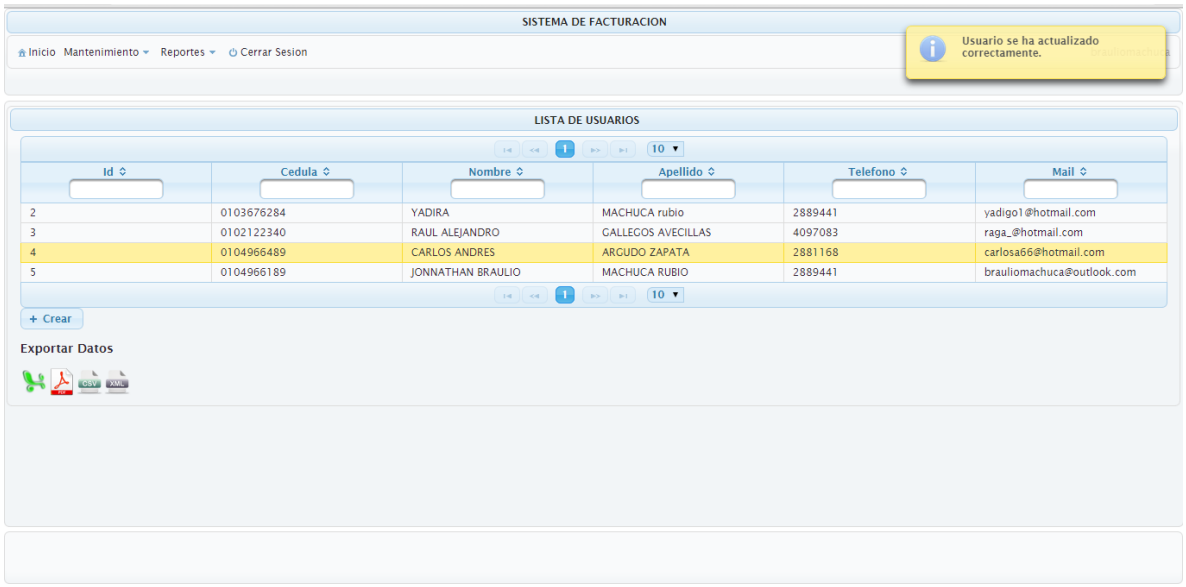


Ilustración 110 Confirmación de Edición de Usuario

y si está correcto nos mostrara una notificación que se ha modificado correctamente



caso contrario nos indicara que está mal.

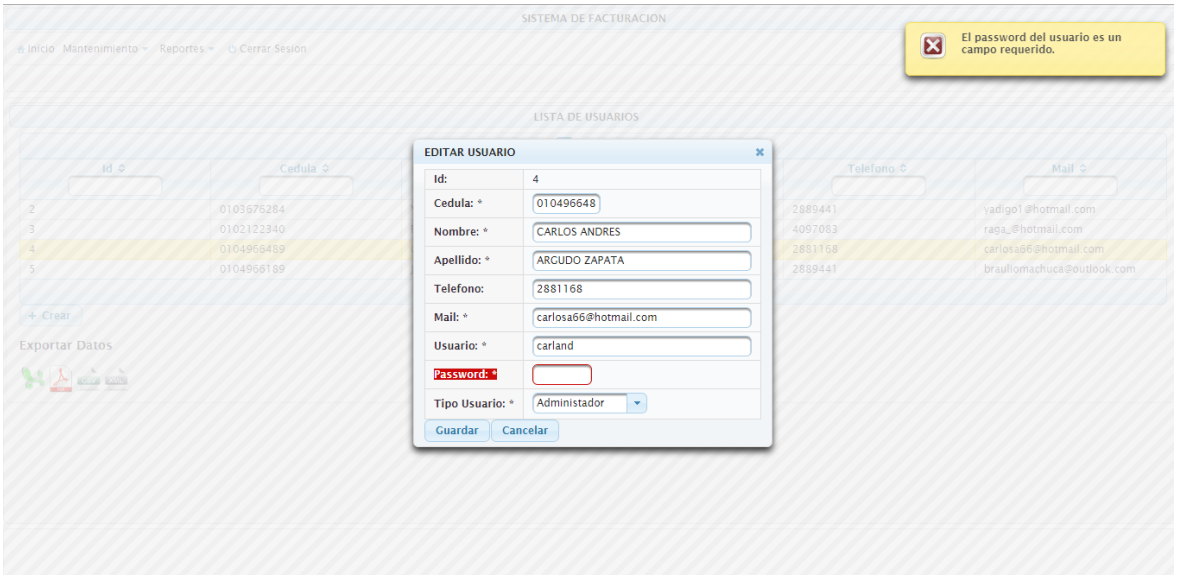


Ilustración 111Notificacion de error al editar usuario

Borrar usuario.

Después de listar los usuarios, hacemos click derecho sobre el cliente que vamos a eliminar y hacemos click en la opción borrar. ‘

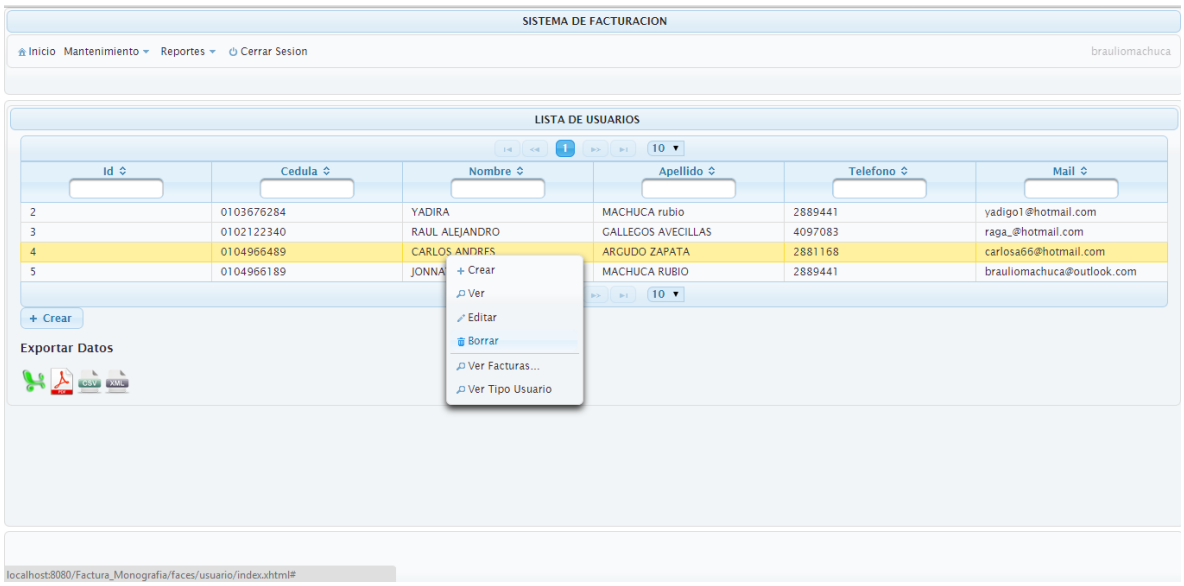


Ilustración 112Opciones de menu Usuario Borrar

Obtendremos un cuadro de confirmación para borrar el usuario.

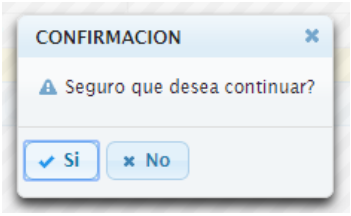


Ilustración 113Confirmación de borrar Usuario

Mantenimiento de producto.

Listar productos.

Para listar usuarios hacemos click en el menú principal y luego en la opción producto.

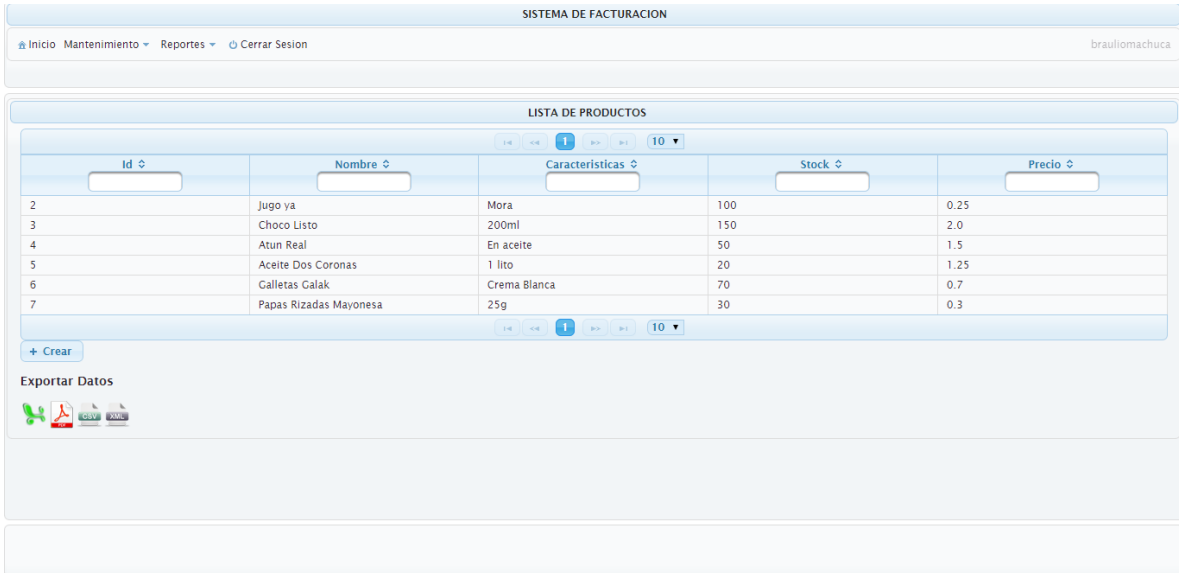


Ilustración 114Listado De Productos

Aquí podremos filtrar los campos para buscar a un producto y exportar a los diferentes exportados, ya sea Excel, pdf , csv o xml.

Crear Producto.

Para crear a un usuario, una vez listados podremos hacer click directamente en el botón crear o hacer click derecho sobre cualquier campo de la lista y escoger la opción crear.

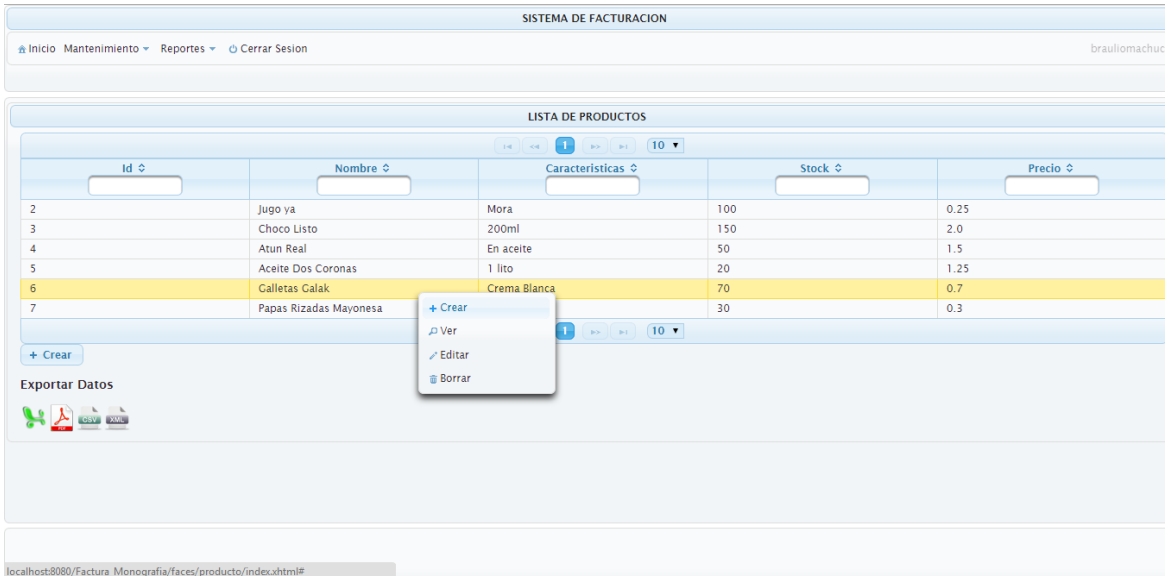


Ilustración 115Opciones de menú Producto Crear

El sistema mostrara un formulario para el ingreso de los productos.

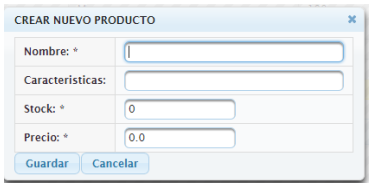


Ilustración 116 Formulario Ingreso Producto

Luego de llenar el formulario debemos hacer click en el botón guardar, nos saldrá un cuadro de confirmación.

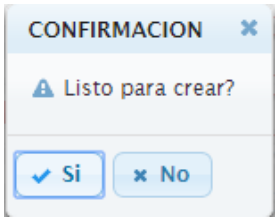


Ilustración 117 Confirmación de ingreso de producto

Hacemos click en sí, si estamos listos.

Si hemos completado con éxito los datos, direccionara a la lista de usuarios actualizados con una notificación indicando que nos indica que se creó correctamente.

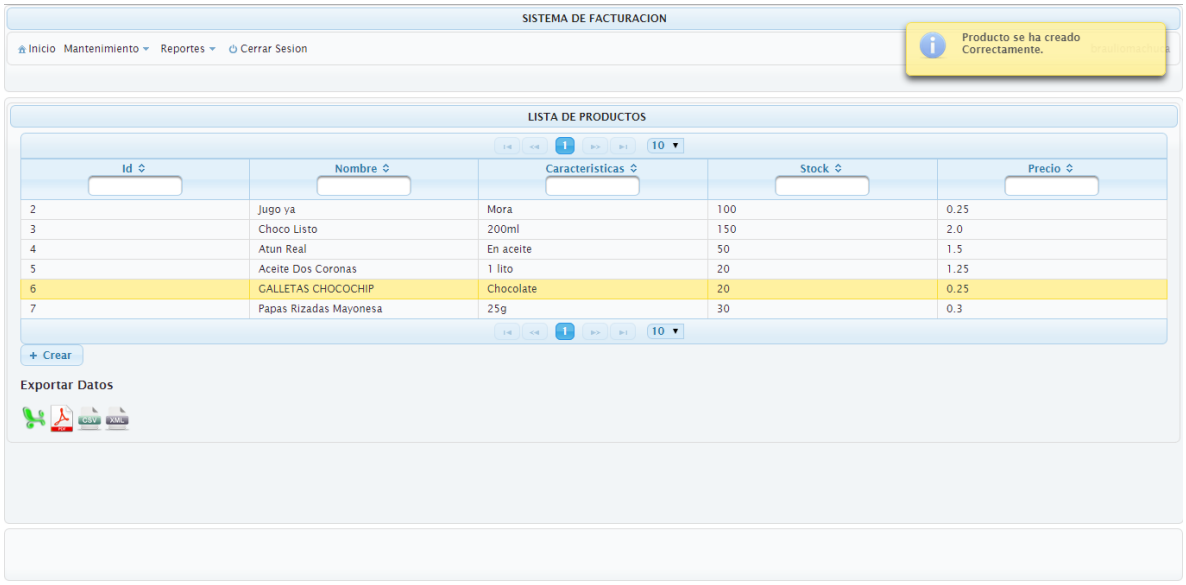


Ilustración 118Notificacion de Producto Ingresado Correctamente

Editar producto

Para modificar el producto una vez listado los productos hacemos click derecho sobre el producto que deseamos editar y luego click en editar.

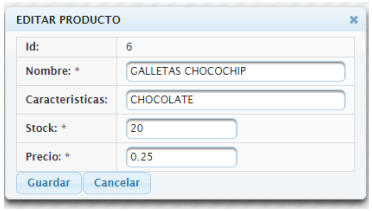


Ilustración 119Formulario De edición del Producto

Podremos editar todos los campos con excepción del id, luego guardaremos los datos.

Al igual que al crear un nuevo producto, también mostrara el cuadro de dialogo para confirmar,

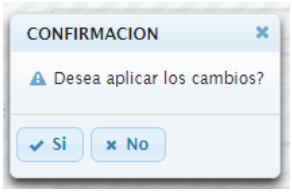


Ilustración 120Confirmación de edición de producto

y si está correcto nos mostrara una notificación que se ha modificado correctamente

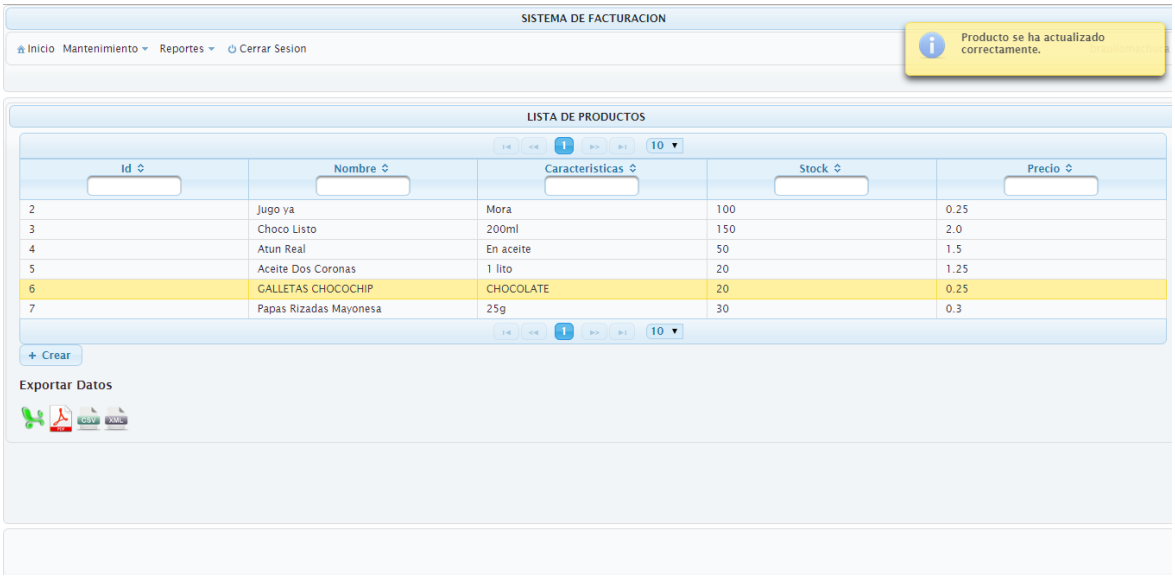


Ilustración 121 Notificación de ingreso Correcto

caso contrario nos indicara que está mal.

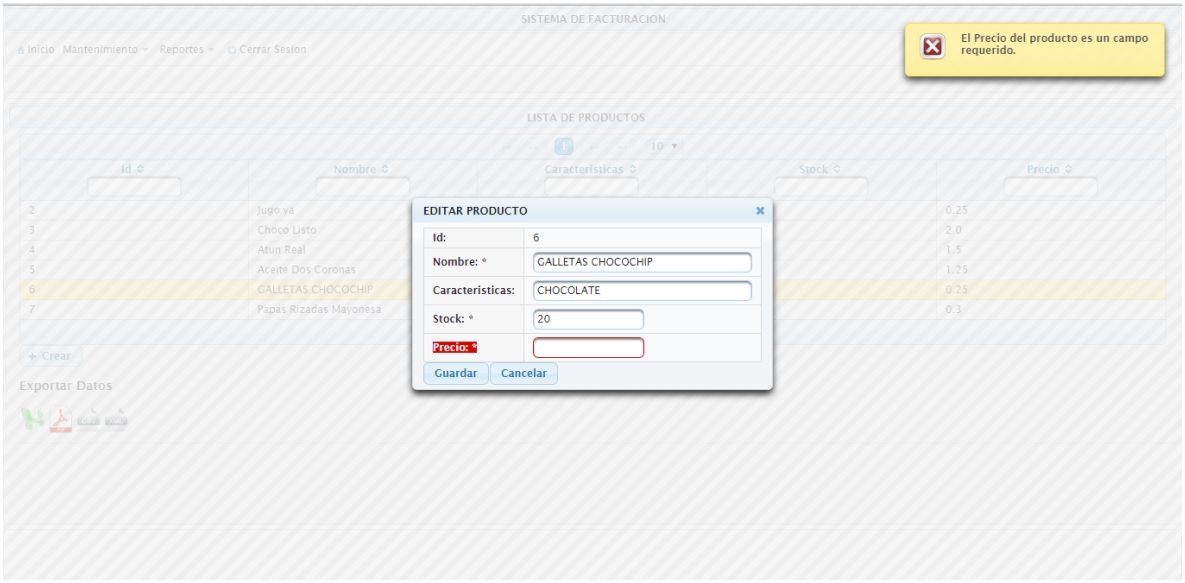


Ilustración 122Notificacion de Producto error en la Edición de Producto

Borrar producto.

Después de listar los productos, hacemos click derecho sobre el producto que vamos a eliminar y hacemos click en la opción borrar.

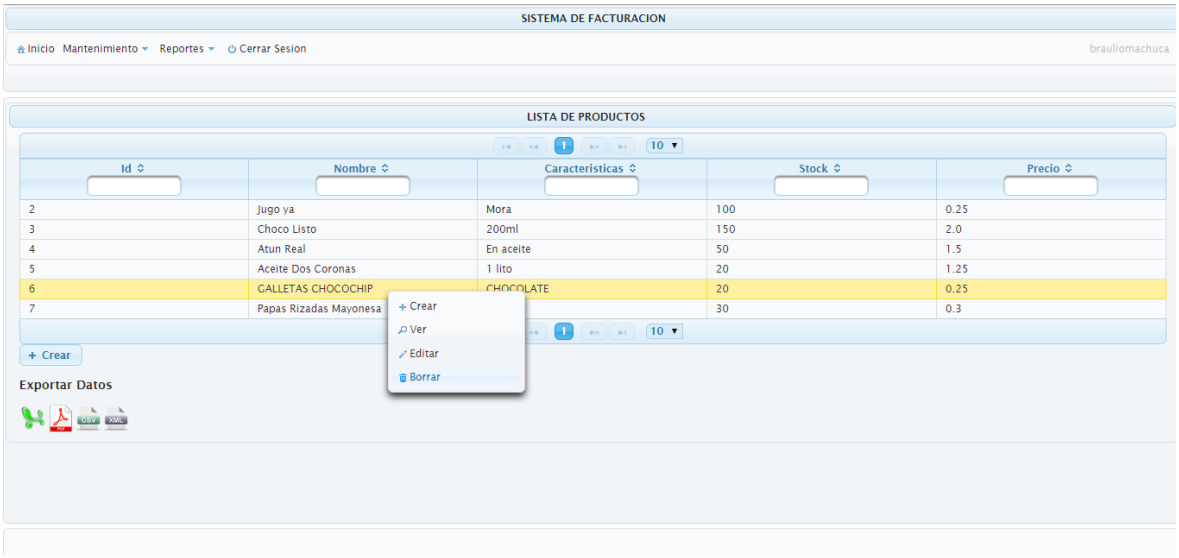


Ilustración 123Opciones de menú Producto Borrar

Obtendremos un cuadro de confirmación para borrar el usuario.

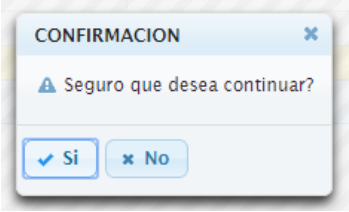


Ilustración 124Confirmación de Borrado de Producto

Mantenimiento Factura

Listar Facturas.

Para listar las facturas vamos al menú principal en facturas.

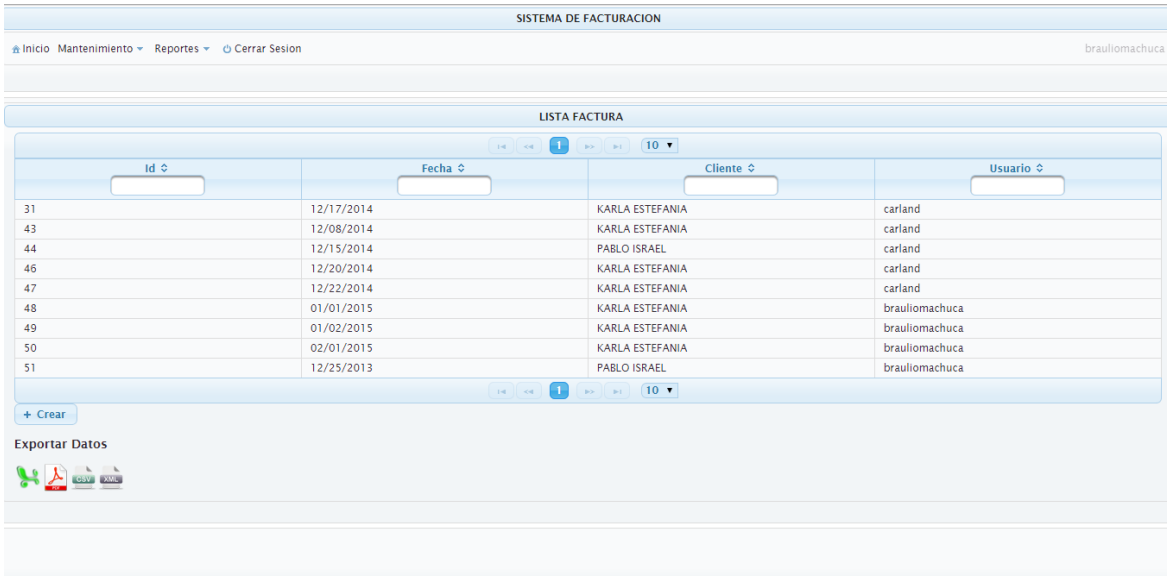


Ilustración 125Listado de factura

Aquí podremos buscar la facturas, y exportar los datos de la tabla.

Facturar.

Para crear una factura podremos hacer click directamente en el botón crear o hacemos click derecho sobre algún campo de la tabla y luego en la opción crear.

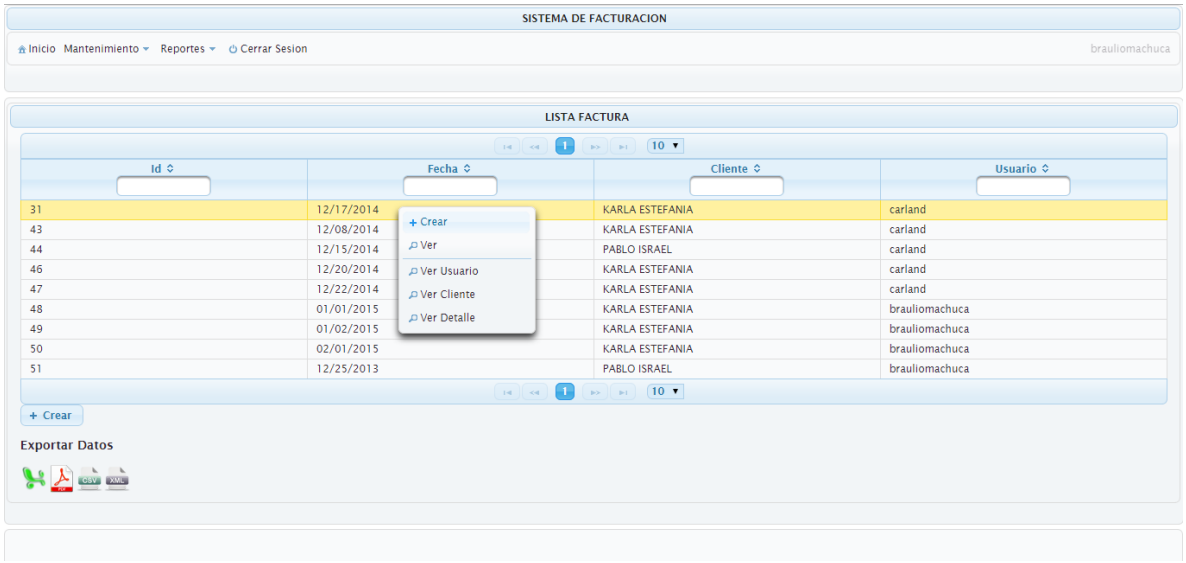


Ilustración 126 Opciones de menú Factura Crear

Obtendremos una pantalla en el formulario asi.

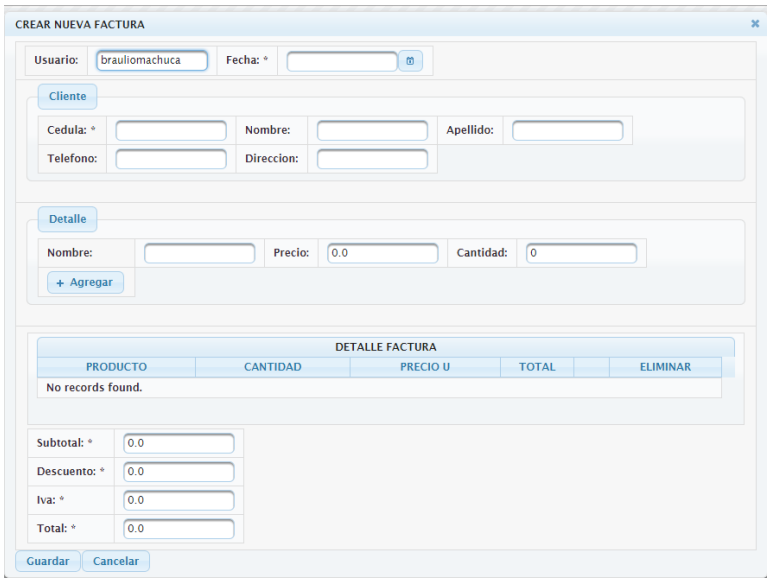


Ilustración 127Formulario ingreso de factura

En la parte superior escogeremos la fecha que se emitió la factura.

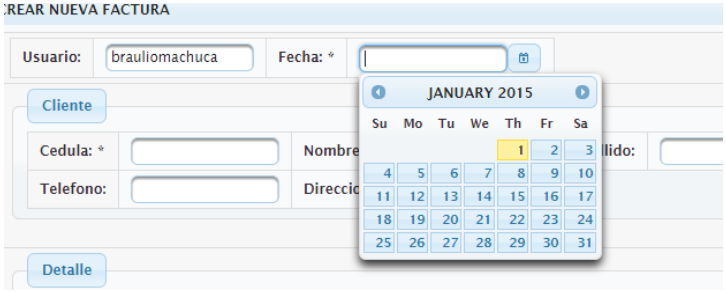


Ilustración 128Fecha de Factura

Luego escribiremos en el panel del cliente la cedula, y a medida de que vamos escribiendo nos saldrá una lista de los clientes.

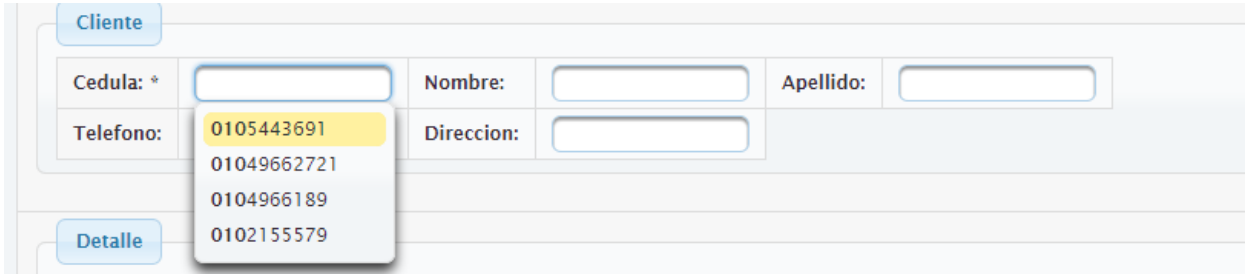


Ilustración 129Cliente En factura

Después de escoger el cliente, se nos llenara automáticamente el resto de datos del cliente.

Ilustración 130Datos del cliente en factura

En panel del detalle vamos a escribir el nombre del producto y seleccionar de la misma manera que lo hicimos con la cedula del cliente, y nos saldrá automáticamente el precio del producto, teniendo que escribir la cantidad de productos q se comprara. Y luego agregamos en la lista.

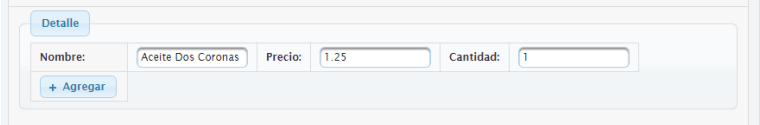


Ilustración 131Ingreso detalle

Así obtendremos la lista de todos los productos que hemos agregado.

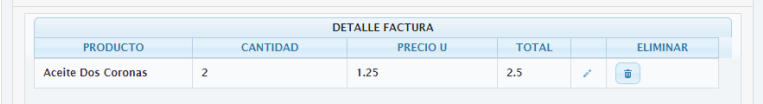


Ilustración 132Detalle de la factura

Aquí también podremos editar o eliminar los productos de la lista.

Finalmente podremos el descuento en el caso de q exista.

Ilustración 133Descuento en la factura

Y click en guardar.

Obtendremos un cuadro de confirmación.

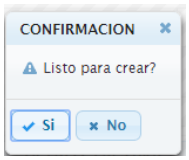


Ilustración 134 Confirmación para guardar Factura

Por ultimo si todo sale bien nos saldrá una notificación diciendo que se ha guardado con éxito y una lista actualizada con todas las facturas.

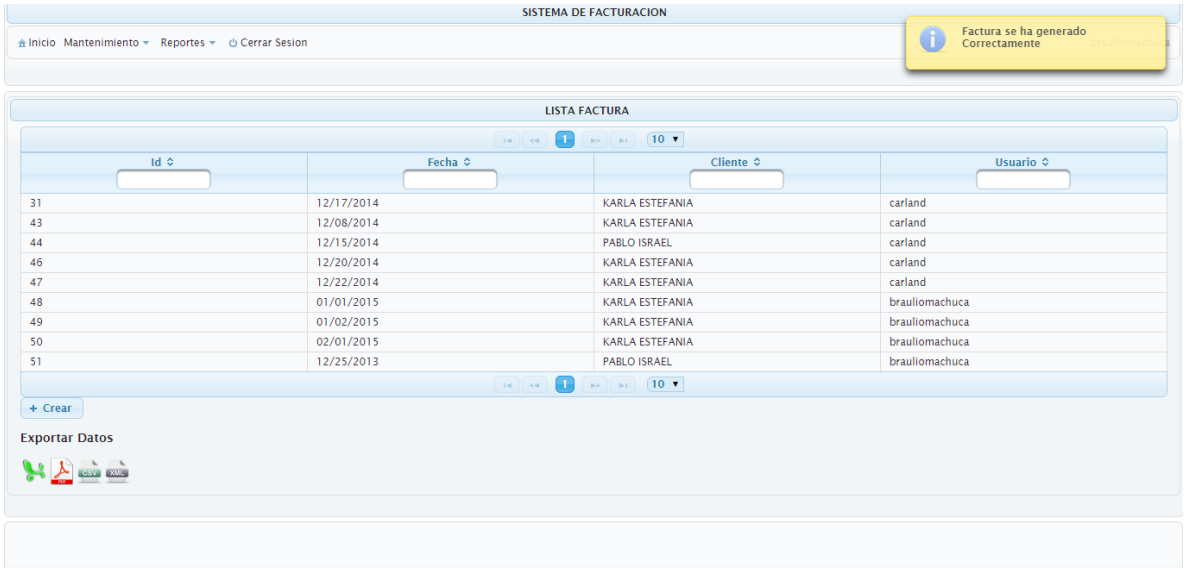


Ilustración 135Notificacion Ingreso Correctamente la factura.

Ver detalle

Para ver el detalle seleccionamos cualquier campo de la lista y hacemos click derecho para seleccionar la opción ver detalle.

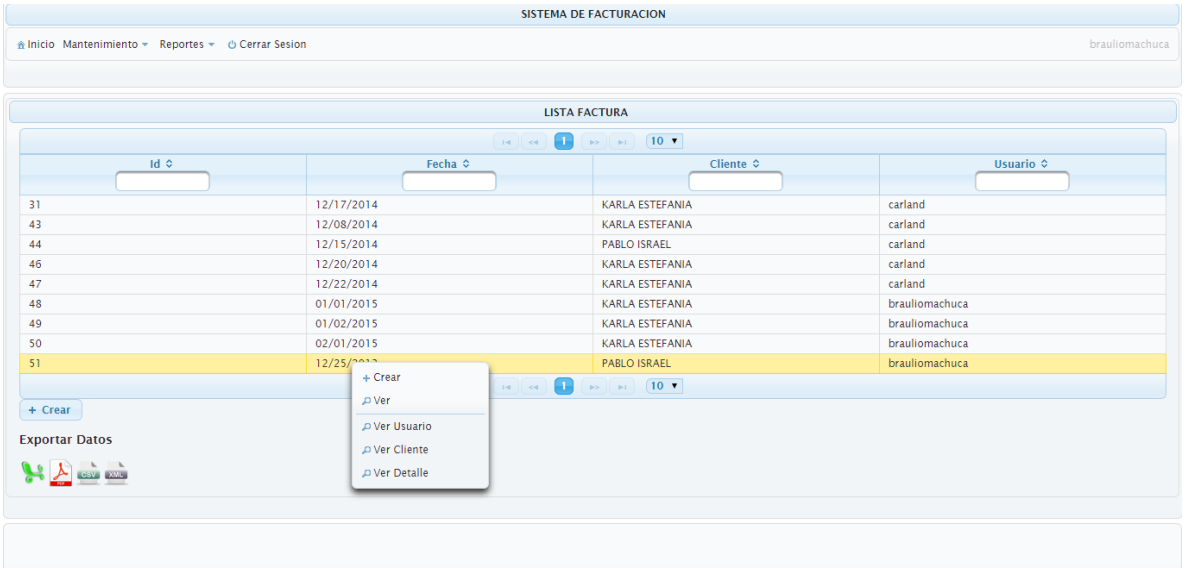


Ilustración 136 Opción de menú Ver detalle Factura

De la misma manera podremos ver el todos los datos del usuario que genero dicha factura y los datos del cliente de la misma.

Reportes

Para ver los reportes nos iremos al menú principal en la opción reportes.

Reportes de stock.

Aquí podremos ver de manera gráfica el stock de los productos que tenemos en stock.

Solo tenemos que selección el número de productos que vamos a visualizar y click en ver.

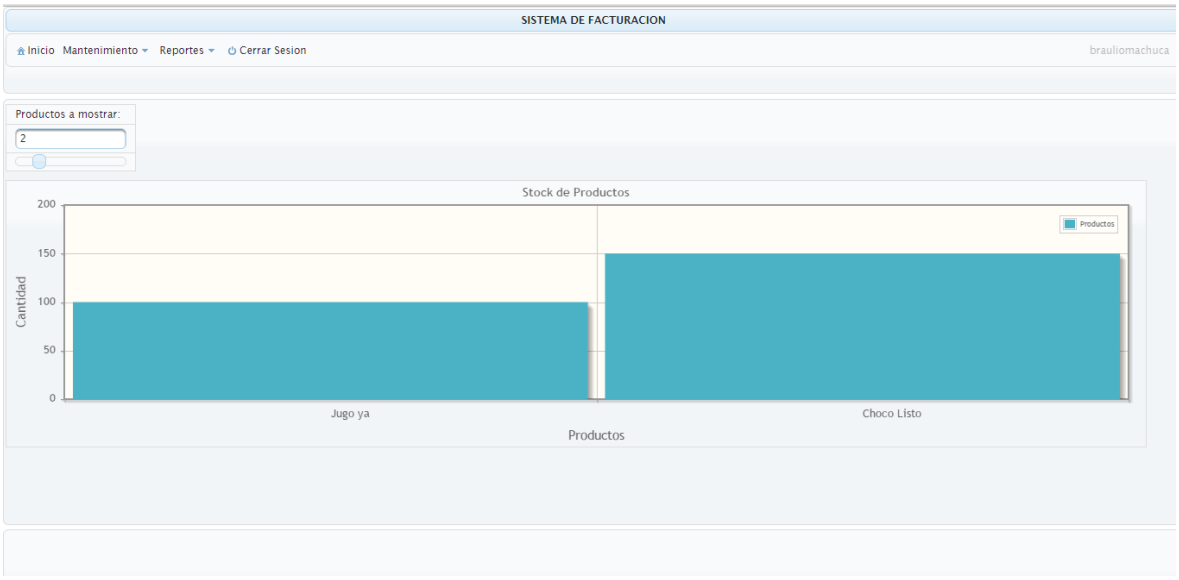


Ilustración 137Reporte stock de productos

Reporte de Vendedores

Aquí observaremos un gráfico con el porcentaje de las ventas del usuario por año.

Solo seleccionaremos el año que queremos y hacemos click en el botón ver.

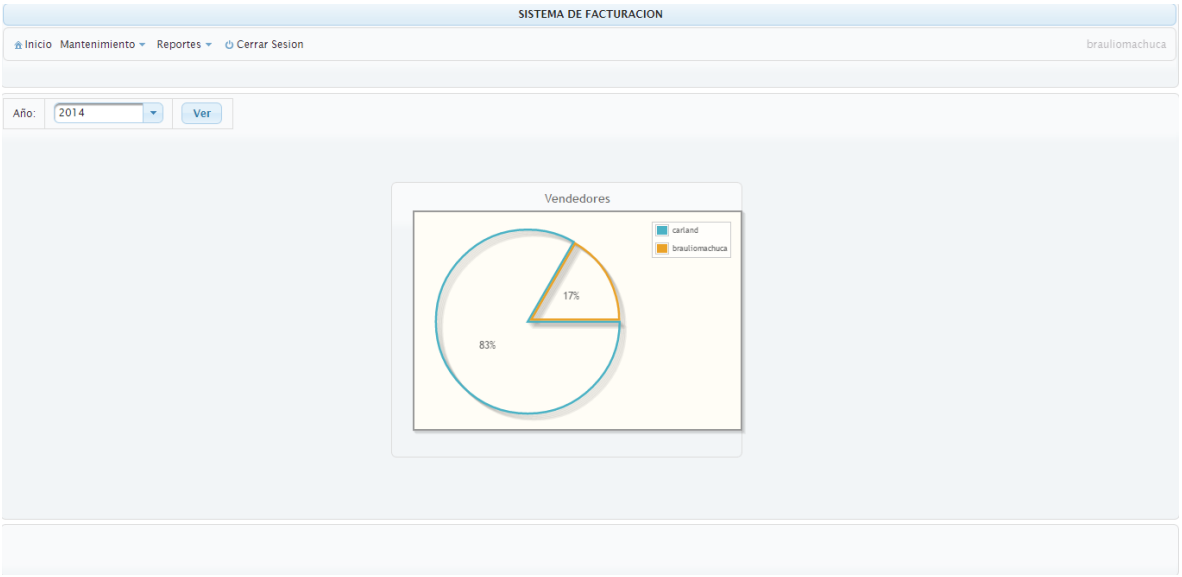


Ilustración 138Reporte Vendedores

Reporte de ventas

En este reporte nos muestra el número de ventas que tenemos por mes solo tenemos que seleccionar el año y hacer click en el botón ver.

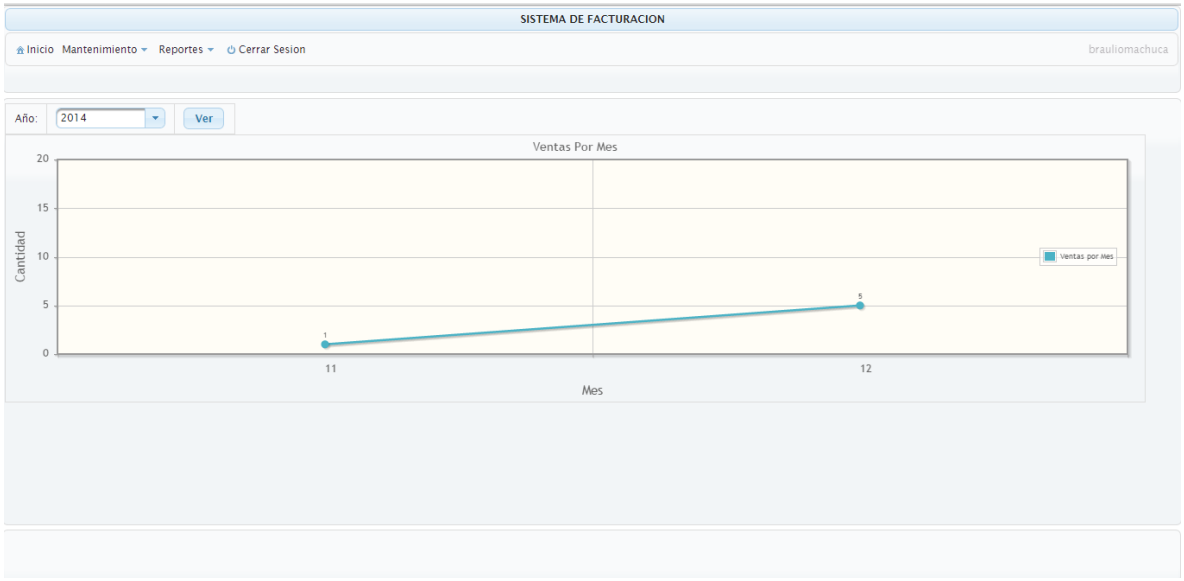


Ilustración 139Reportes por mes

Anexo # 2

Código Fuente de la Aplicación.

Entidad Cliente.

```
package ec.edu.uda.entidades;

import java.io.Serializable;
import java.util.Collection;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
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;

/**
 *
 * @author BraulioMachuca
 */
@Entity
@Table(name = "cliente")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Cliente.findAll", query = "SELECT c FROM
Cliente c"),
    @NamedQuery(name = "Cliente.findByCliId", query = "SELECT c FROM
Cliente c WHERE c.cliId = :cliId"),
    @NamedQuery(name = "Cliente.findByCliCedula", query = "SELECT c
FROM Cliente c WHERE c.cliCedula = :cliCedula"),
```

```
    @NamedQuery(name = "Cliente.findByCliNombre", query = "SELECT c
FROM Cliente c WHERE c.cliNombre = :cliNombre"),
    @NamedQuery(name = "Cliente.findByCliApellido", query = "SELECT c
FROM Cliente c WHERE c.cliApellido = :cliApellido"),
    @NamedQuery(name = "Cliente.findByCliCiudad", query = "SELECT c
FROM Cliente c WHERE c.cliCiudad = :cliCiudad"),
    @NamedQuery(name = "Cliente.findByCliFecNac", query = "SELECT c
FROM Cliente c WHERE c.cliFecNac = :cliFecNac"),
    @NamedQuery(name = "Cliente.findByCliTelefono", query = "SELECT c
FROM Cliente c WHERE c.cliTelefono = :cliTelefono"),
    @NamedQuery(name = "Cliente.findByCliCelular", query = "SELECT c
FROM Cliente c WHERE c.cliCelular = :cliCelular"),
    @NamedQuery(name = "Cliente.findByCliDireccion", query = "SELECT c
FROM Cliente c WHERE c.cliDireccion = :cliDireccion"),
    @NamedQuery(name = "Cliente.findByCliMail", query = "SELECT c
FROM Cliente c WHERE c.cliMail = :cliMail")})
public class Cliente implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "cli_id")
    private Integer cliId;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 13)
    @Column(name = "cli_cedula")
    private String cliCedula;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "cli_nombre")
    private String cliNombre;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
    @Column(name = "cli_apellido")
    private String cliApellido;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 45)
```

```
    @Column(name = "cli_ciudad")
    private String cliCiudad;
    @Basic(optional = false)
    @NotNull
    @Column(name = "cli_fec_nac")
    @Temporal(TemporalType.DATE)
    private Date cliFecNac;
    @Size(max = 45)
    @Column(name = "cli_telefono")
    private String cliTelefono;
    @Size(max = 45)
    @Column(name = "cli_celular")
    private String cliCelular;
    @Size(max = 45)
    @Column(name = "cli_direccion")
    private String cliDireccion;
    @Size(max = 45)
    @Column(name = "cli_mail")
    private String cliMail;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "facCliente")
    private Collection<Factura> facturaCollection;

    public Cliente() {
    }

    public Cliente(Integer cliId) {
        this.cliId = cliId;
    }

    public Cliente(Integer cliId, String cliCedula, String cliNombre, String
cliApellido, String cliCiudad, Date cliFecNac) {
        this.cliId = cliId;
        this.cliCedula = cliCedula;
        this.cliNombre = cliNombre;
        this.cliApellido = cliApellido;
        this.cliCiudad = cliCiudad;
        this.cliFecNac = cliFecNac;
    }

    public Integer getCliId() {
        return cliId;
```

```
}

public void setCliId(Integer cliId) {
   this.cliId = cliId;
}

public String getCliCedula() {
   return cliCedula;
}

public void setCliCedula(String cliCedula) {
   this.cliCedula = cliCedula;
}

public String getCliNombre() {
   return cliNombre;
}

public void setCliNombre(String cliNombre) {
   this.cliNombre = cliNombre;
}

public String getCliApellido() {
   return cliApellido;
}

public void setCliApellido(String cliApellido) {
   this.cliApellido = cliApellido;
}

public String getCliCiudad() {
   return cliCiudad;
}

public void setCliCiudad(String cliCiudad) {
   this.cliCiudad = cliCiudad;
}

public Date getCliFecNac() {
   return cliFecNac;
}
```

```
public void setCliFecNac(Date cliFecNac) {
    this.cliFecNac = cliFecNac;
}

public String getCliTelefono() {
    return cliTelefono;
}

public void setCliTelefono(String cliTelefono) {
    this.cliTelefono = cliTelefono;
}

public String getCliCelular() {
    return cliCelular;
}

public void setCliCelular(String cliCelular) {
    this.cliCelular = cliCelular;
}

public String getCliDireccion() {
    return cliDireccion;
}

public void setCliDireccion(String cliDireccion) {
    this.cliDireccion = cliDireccion;
}

public String getCliMail() {
    return cliMail;
}

public void setCliMail(String cliMail) {
    this.cliMail = cliMail;
}

@XmlTransient
public Collection<Factura> getFacturaCollection() {
    return facturaCollection;
}
```

```
    public void setFacturaCollection(Collection<Factura> facturaCollection) {
        this.facturaCollection = facturaCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (cliId != null ? cliId.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 Cliente)) {
            return false;
        }
        Cliente other = (Cliente) object;
        if ((this.cliId == null && other.cliId != null) || (this.cliId != null &&
!this.cliId.equals(other.cliId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "ec.edu.uda.entidades.Cliente[ cliId=" + cliId + " ]";
    }

}
```

Controlador Cliente.

```
package ec.edu.uda.controllers;

import ec.edu.uda.entidades.Cliente;
import ec.edu.uda.bean.ClienteFacade;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import javax.annotation.PostConstruct;

@ManagedBean(name = "clienteController")
@ViewScoped
public class ClienteController extends AbstractController<Cliente> {

    @EJB
    private ClienteFacade ejbFacade;
    @PostConstruct
    @Override
    public void init() {
        super.setFacade(ejbFacade);
        FacesContext context = FacesContext.getCurrentInstance();
    }

    public ClienteController() {
        super(Cliente.class);
    }
    public void resetParents() {
    }
    public String navigateFacturaCollection() {
        if (this.getSelected() != null) {

FacesContext.getCurrentInstance().getExternalContext().getRequestMap().put
("Factura_items", this.getSelected().getFacturaCollection());
        }
        return "/paginas/factura/index";
    }

}
```

Controlador Abstracto

```
package ec.edu.uda.controllers;

import ec.edu.uda.bean.AbstractFacade;
import ec.edu.uda.beans.util.JsfUtil;
import java.io.Serializable;
import java.util.Collection;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.faces.event.ActionEvent;

import java.util.ResourceBundle;
import javax.ejb.EJBException;
import javax.annotation.PostConstruct;
import javax.faces.context.FacesContext;
import javax.validation.ConstraintViolation;
import javax.validation.ConstraintViolationException;
public abstract class AbstractController<T> implements Serializable {

    private AbstractFacade<T> ejbFacade;
    private Class<T> itemClass;
    private T selected;
    private Collection<T> items;

    private enum PersistAction {

        CREATE,
        DELETE,
        UPDATE
    }

    public AbstractController() {
    }

    public AbstractController(Class<T> itemClass) {
        this.itemClass = itemClass;
    }


    public abstract void init();
```

```
    protected AbstractFacade<T> getFacade() {
        return ejbFacade;
    }
    protected void setFacade(AbstractFacade<T> ejbFacade) {
        this.ejbFacade = ejbFacade;
    }


    public T getSelected() {
        return selected;
    }

    public void setSelected(T selected) {
        this.selected = selected;
    }
    protected void setEmbeddableKeys() {
        // Nothing to do if entity does not have any embeddable key.
    }

    ;

    protected void initializeEmbeddableKey() {
        // Nothing to do if entity does not have any embeddable key.
    }

      public Collection<T> getItems() {
        if (items == null) {
            items = this.ejbFacade.findAll();
        }
        return items;
    }
    public void setItems(Collection<T> items) {
        this.items = items;
    }

    public void save(ActionEvent event) {
        String msg =
ResourceBundle.getBundle("/MyBundle").getString(itemClass.getSimpleNam
e() + "Updated");
        persist(PersistAction.UPDATE, msg);
    }
```

```
     */
    public void saveNew(ActionEvent event) {
        String msg =
ResourceBundle.getBundle("/MyBundle").getString(itemClass.getSimpleNam
e() + "Created");
        persist(PersistAction.CREATE, msg);
        if (!isValidationFailed()) {
            items = null; // Invalidate list of items to trigger re-query.
        }
    }

    public void delete(ActionEvent event) {
        String msg =
ResourceBundle.getBundle("/MyBundle").getString(itemClass.getSimpleNam
e() + "Deleted");
        persist(PersistAction.DELETE, msg);
        if (!isValidationFailed()) {
            selected = null; // Remove selection
            items = null; // Invalidate list of items to trigger re-query.
        }
    }

    private void persist(PersistAction persistAction, String successMessage) {
        if (selected != null) {
            this.setEmbeddableKeys();
            try {
                if (persistAction != PersistAction.DELETE) {
                    this.ejbFacade.edit(selected);
                } else {
                    this.ejbFacade.remove(selected);
                }
                JsfUtil.addSuccessMessage(successMessage);
            } catch (EJBException ex) {
                Throwable cause = JsfUtil.getRootCause(ex.getCause());
                if (cause != null) {
                    if (cause instanceof ConstraintViolationException) {
                        ConstraintViolationException excp =
(ConstraintViolationException) cause;
                        for (ConstraintViolation s : excp.getConstraintViolations()) {
                            JsfUtil.addErrorMessage(s.getMessage());
```

```
                    }
                } else {
                    String 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("/MyBundle").getString("PersistenceErrorOccured
"));
        }
    }
}

/**
public T prepareCreate(ActionEvent event) {
    T newItem;
    try {
        newItem = itemClass.newInstance();
        this.selected = newItem;
        initializeEmbeddableKey();
        return newItem;
    } catch (InstantiationException | IllegalAccessException ex) {
        Logger.getLogger(this.getClass().getName()).log(Level.SEVERE, null,
ex);
    }
    return null;
}

public boolean isValidationFailed() {
    return JsfUtil.isValidationFailed();
}
```

```
    public String getComponentMessages(String clientComponent, String
defaultMessage) {
        return JsfUtil.getComponentMessages(clientComponent,
defaultMessage);
    }

    @PostConstruct
    public void initParams() {
        Object paramItems =
FacesContext.getCurrentInstance().getExternalContext().getRequestMap().get
(itemClass.getSimpleName() + "_items");
        if (paramItems != null) {
            this.items = (Collection<T>) paramItems;
        }
    }

}
```

Convertidor Cliente

```
package ec.edu.uda.convertidores;

import ec.edu.uda.entidades.Cliente;
import ec.edu.uda.bean.ClienteFacade;
import ec.edu.uda.beans.util.JsfUtil;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;

@FacesConverter(value = "clienteConverter")
public class ClienteConverter implements Converter {

    @EJB
    private ClienteFacade ejbFacade;

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

```
        if (value == null || value.length() == 0 ||
JsfUtil.isDummySelectItem(component, value)) {
            return null;
        }
        return this.ejbFacade.find(getKey(value));
    }

    java.lang.Integer getKey(String value) {
        java.lang.Integer key;
        key = Integer.valueOf(value);
        return key;
    }

    String getStringKey(java.lang.Integer value) {
        StringBuffer sb = new StringBuffer();
        sb.append(value);
        return sb.toString();
    }

    @Override
    public String getAsString(FacesContext facesContext, UIComponent
component, Object object) {
        if (object == null
                || (object instanceof String && ((String) object).length() == 0)) {
            return null;
        }
        if (object instanceof Cliente) {
            Cliente o = (Cliente) object;
            return getStringKey(o.getCliId());
        } else {
            Logger.getLogger(this.getClass().getName()).log(Level.SEVERE,
"object {0} is of type {1}; expected type: {2}", new Object[]{object,
object.getClass().getName(), Cliente.class.getName()});
            return null;
        }
    }

}
```

Bean del cliente

```
package ec.edu.uda.bean;

import ec.edu.uda.entidades.Cliente;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author BraulioMachuca
 */
@Stateless
public class ClienteFacade extends AbstractFacade<Cliente> {
    @PersistenceContext(unitName = "Factura1PU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public ClienteFacade() {
        super(Cliente.class);
    }

}
```

Bean Abstracto

```
package ec.edu.uda.bean;

import java.util.List;
import javax.persistence.EntityManager;

/**
 *
 * @author BraulioMachuca
 */
public abstract class AbstractFacade<T> {
    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
```

```
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    }

    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    public List<T> findAll() {
        javax.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    public List<T> findRange(int[] range) {
        javax.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    public int count() {
        javax.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
```

```
        javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }

}
```

Página de inicio del cliente

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns:ui="http://java.sun.com/jsf/facelets"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:f="http://java.sun.com/jsf/core"
                xmlns:p="http://primefaces.org/ui"
                template="/WEB-INF/include/template.xhtml">

    <ui:define name="title">
        <h:outputText value="#{myBundle.ClienteTitle}"/>
    </ui:define>

    <ui:define name="body">

        <ui:include src="/WEB-INF/include/entity/cliente/List.xhtml"/>
        <ui:include src="/WEB-INF/include/entity/cliente/View.xhtml"/>
        <ui:include src="/WEB-INF/include/entity/cliente/Edit.xhtml"/>
        <ui:include src="/WEB-INF/include/entity/cliente/Create.xhtml"/>

    </ui:define>

</ui:composition>
```

Página de Creación del Cliente.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
```

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

    <ui:composition>

        <p:dialog id="ClienteCreateDlg" widgetVar="ClienteCreateDialog"
modal="true" resizable="false" appendTo="@(body)"
header="#{myBundle.CreateClienteTitle}" closeOnEscape="true">

            <h:form id="ClienteCreateForm">

                <h:panelGroup id="display"
rendered="#{clienteController.selected != null}">
                    <p:panelGrid columns="2" columnClasses="column">

                        <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliCedula}" for="cliCedula" />
                        <p:inputText id="cliCedula"
value="#{clienteController.selected.cliCedula}"
title="#{myBundle.CreateClienteTitle_cliCedula}" required="true"
requiredMessage="#{myBundle.CreateClienteRequiredMessage_cliCedula}"
size="13" maxlength="13"/>

                        <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliNombre}" for="cliNombre" />
                        <p:inputText id="cliNombre"
value="#{clienteController.selected.cliNombre}"
title="#{myBundle.CreateClienteTitle_cliNombre}" required="true"
requiredMessage="#{myBundle.CreateClienteRequiredMessage_cliNombre}"
size="45" maxlength="45"/>

                        <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliApellido}" for="cliApellido" />
                        <p:inputText id="cliApellido"
value="#{clienteController.selected.cliApellido}"
title="#{myBundle.CreateClienteTitle_cliApellido}" required="true"
requiredMessage="#{myBundle.CreateClienteRequiredMessage_cliApellido}
" size="45" maxlength="45"/>

                        <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliCiudad}" for="cliCiudad" />
```

```
                <p:inputText id="cliCiudad"
value="#{clienteController.selected.cliCiudad}"
title="#{myBundle.CreateClienteTitle_cliCiudad}" required="true"
requiredMessage="#{myBundle.CreateClienteRequiredMessage_cliCiudad}"
size="45" maxlength="45"/>

                <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliFecNac}" for="cliFecNac" />
                <p:calendar id="cliFecNac" pattern="MM/dd/yyyy"
value="#{clienteController.selected.cliFecNac}"
title="#{myBundle.CreateClienteTitle_cliFecNac}" required="true"
requiredMessage="#{myBundle.CreateClienteRequiredMessage_cliFecNac}"
showOn="button"/>

                <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliTelefono}" for="cliTelefono" />
                <p:inputText id="cliTelefono"
value="#{clienteController.selected.cliTelefono}"
title="#{myBundle.CreateClienteTitle_cliTelefono}" size="45"
maxlength="45"/>

                <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliCelular}" for="cliCelular" />
                <p:inputText id="cliCelular"
value="#{clienteController.selected.cliCelular}"
title="#{myBundle.CreateClienteTitle_cliCelular}" size="45"
maxlength="45"/>

                <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliDireccion}" for="cliDireccion" />
                <p:inputText id="cliDireccion"
value="#{clienteController.selected.cliDireccion}"
title="#{myBundle.CreateClienteTitle_cliDireccion}" size="45"
maxlength="45"/>

                <p:outputLabel
value="#{myBundle.CreateClienteLabel_cliMail}" for="cliMail" />
                <p:inputText id="cliMail"
value="#{clienteController.selected.cliMail}"
title="#{myBundle.CreateClienteTitle_cliMail}" size="45" maxlength="45"/>
            </p:panelGrid>
```

```
            <p:commandButton
actionListener="#{clienteController.saveNew}" value="#{myBundle.Save}"
update="display,:ClienteListForm:datalist,:growl"
oncomplete="handleSubmit(xhr,status,args,PF('ClienteCreateDialog'));">
                <p:confirm header="#{myBundle.ConfirmationHeader}"
message="#{myBundle.ConfirmCreateMessage}" icon="ui-icon-alert"/>
            </p:commandButton>

            <p:commandButton value="#{myBundle.Cancel}"
onclick="PF('ClienteCreateDialog').hide()"/>
          </h:panelGroup>

        </h:form>

      </p:dialog>

   </ui:composition>

</html>
```

Página de edición del cliente.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">

   <ui:composition>
      <p:dialog id="ClienteEditDlg" widgetVar="ClienteEditDialog"
modal="true" resizable="false"  appendTo="@(body)"
header="#{myBundle.EditClienteTitle}" closeOnEscape="true">

        <h:form id="ClienteEditForm">

          <h:panelGroup id="display">
            <p:panelGrid  columns="2" columnClasses="column"
rendered="#{clienteController.selected != null}">
```

```
                <h:outputLabel value="#{myBundle.EditClienteLabel_cliId}"
for="cliId" />
                <h:outputText id="cliId"
value="#{clienteController.selected.cliId}" />

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliCedula}" for="cliCedula" />
                <p:inputText id="cliCedula"
value="#{clienteController.selected.cliCedula}"
title="#{myBundle.EditClienteTitle_cliCedula}" required="true"
requiredMessage="#{myBundle.EditClienteRequiredMessage_cliCedula}"
size="13" maxlength="13"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliNombre}" for="cliNombre" />
                <p:inputText id="cliNombre"
value="#{clienteController.selected.cliNombre}"
title="#{myBundle.EditClienteTitle_cliNombre}" required="true"
requiredMessage="#{myBundle.EditClienteRequiredMessage_cliNombre}"
size="45" maxlength="45"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliApellido}" for="cliApellido" />
                <p:inputText id="cliApellido"
value="#{clienteController.selected.cliApellido}"
title="#{myBundle.EditClienteTitle_cliApellido}" required="true"
requiredMessage="#{myBundle.EditClienteRequiredMessage_cliApellido}"
size="45" maxlength="45"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliCiudad}" for="cliCiudad" />
                <p:inputText id="cliCiudad"
value="#{clienteController.selected.cliCiudad}"
title="#{myBundle.EditClienteTitle_cliCiudad}" required="true"
requiredMessage="#{myBundle.EditClienteRequiredMessage_cliCiudad}"
size="45" maxlength="45"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliFecNac}" for="cliFecNac" />
```

```
                <p:calendar id="cliFecNac" pattern="MM/dd/yyyy"
value="#{clienteController.selected.cliFecNac}"
title="#{myBundle.EditClienteTitle_cliFecNac}" required="true"
requiredMessage="#{myBundle.EditClienteRequiredMessage_cliFecNac}"
showOn="button"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliTelefono}" for="cliTelefono" />
                <p:inputText id="cliTelefono"
value="#{clienteController.selected.cliTelefono}"
title="#{myBundle.EditClienteTitle_cliTelefono}" size="45"
maxlength="45"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliCelular}" for="cliCelular" />
                <p:inputText id="cliCelular"
value="#{clienteController.selected.cliCelular}"
title="#{myBundle.EditClienteTitle_cliCelular}" size="45" maxlength="45"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliDireccion}" for="cliDireccion" />
                <p:inputText id="cliDireccion"
value="#{clienteController.selected.cliDireccion}"
title="#{myBundle.EditClienteTitle_cliDireccion}" size="45"
maxlength="45"/>

                <p:outputLabel
value="#{myBundle.EditClienteLabel_cliMail}" for="cliMail" />
                <p:inputText id="cliMail"
value="#{clienteController.selected.cliMail}"
title="#{myBundle.EditClienteTitle_cliMail}" size="45" maxlength="45"/>
            </p:panelGrid>
            <p:commandButton actionListener="#{clienteController.save}"
value="#{myBundle.Save}"
update="display,:ClienteListForm:datalist,:growl"
oncomplete="handleSubmit(xhr,status,args,PF('ClienteEditDialog'));">
                <p:confirm header="#{myBundle.ConfirmationHeader}"
message="#{myBundle.ConfirmEditMessage}" icon="ui-icon-alert"/>
            </p:commandButton>
            <p:commandButton value="#{myBundle.Cancel}"
onclick="PF('ClienteEditDialog').hide()"/>
```

```
            </h:panelGroup>

        </h:form>

    </p:dialog>

</ui:composition>

</html>
```

Página de Listado de Clientes

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:f="http://java.sun.com/jsf/core"
                xmlns:p="http://primefaces.org/ui">

    <h:form id="ClienteListForm">

        <p:panel header="#{myBundle.ListClienteTitle}">
            <p:contextMenu id="ClienteDataTableContextMenu" for="datalist">
                <p:menuitem value="#{myBundle.Create}"
onclick="document.getElementById('ClienteListForm:createButton').click();"
icon="ui-icon-plus"/>
                <p:menuitem value="#{myBundle.View}"
onclick="document.getElementById('ClienteListForm:viewButton').click();"
icon="ui-icon-search"/>
                <p:menuitem value="#{myBundle.Edit}"
onclick="document.getElementById('ClienteListForm:editButton').click();"
icon="ui-icon-pencil"/>
                <p:menuitem value="#{myBundle.Delete}"
onclick="document.getElementById('ClienteListForm:deleteButton').click();"
icon="ui-icon-trash"/>
                <p:separator/>
                <p:menuitem
value="#{myBundle.ClienteMenuItem_facturaCollection}" icon="ui-icon-
search"  action="#{clienteController.navigateFacturaCollection}"
```

```
disabled="#{empty clienteController.selected.facturaCollection}"
ajax="false"/>
        </p:contextMenu>
        <p:dataTable id="datalist"
                     value="#{clienteController.items}"
                     var="item"
                     rowKey="#{item.cliId}"
                     paginator="true"
                     rows="10"
                     rowsPerPageTemplate="10,20,30,40,50"
                     selectionMode="single"
                     selection="#{clienteController.selected}">

            <p:ajax event="rowSelect"   update=":ClienteListForm:createButton
:ClienteListForm:viewButton :ClienteListForm:editButton
:ClienteListForm:deleteButton
:ClienteListForm:ClienteDataTableContextMenu"
listener="#{clienteController.resetParents}"/>
            <p:ajax event="rowUnselect"
update=":ClienteListForm:createButton :ClienteListForm:viewButton
:ClienteListForm:editButton :ClienteListForm:deleteButton
:ClienteListForm:ClienteDataTableContextMenu"
listener="#{clienteController.resetParents}"/>
            <p:ajax event="rowDblselect"
onsuccess="document.getElementById('ClienteListForm:viewButton').click();
"/>

            <p:column sortBy="#{item.cliId}" filterBy="#{item.cliId}">
                <f:facet name="header">
                    <h:outputText value="#{myBundle.ListClienteTitle_cliId}"/>
                </f:facet>
                <h:outputText value="#{item.cliId}"/>
            </p:column>


            <p:column sortBy="#{item.cliCedula}"
filterBy="#{item.cliCedula}">
                <f:facet name="header">
                    <h:outputText
value="#{myBundle.ListClienteTitle_cliCedula}"/>
                </f:facet>
```

```
                <h:outputText value="#{item.cliCedula}"/>
            </p:column>


            <p:column sortBy="#{item.cliNombre}"
filterBy="#{item.cliNombre}">
                <f:facet name="header">
                    <h:outputText
value="#{myBundle.ListClienteTitle_cliNombre}"/>
                </f:facet>
                <h:outputText value="#{item.cliNombre}"/>
            </p:column>


            <p:column sortBy="#{item.cliApellido}"
filterBy="#{item.cliApellido}">
                <f:facet name="header">
                    <h:outputText
value="#{myBundle.ListClienteTitle_cliApellido}"/>
                </f:facet>
                <h:outputText value="#{item.cliApellido}"/>
            </p:column>


            <p:column sortBy="#{item.cliCiudad}"
filterBy="#{item.cliCiudad}">
                <f:facet name="header">
                    <h:outputText
value="#{myBundle.ListClienteTitle_cliCiudad}"/>
                </f:facet>
                <h:outputText value="#{item.cliCiudad}"/>
            </p:column>


            <p:column sortBy="#{item.cliFecNac}"
filterBy="#{item.cliFecNac}">
                <f:facet name="header">
                    <h:outputText
value="#{myBundle.ListClienteTitle_cliFecNac}"/>
                </f:facet>
                <h:outputText value="#{item.cliFecNac}">
```

```
                <f:convertDateTime pattern="MM/dd/yyyy" />
            </h:outputText>
        </p:column>


    </p:dataTable>

    <p:commandButton id="createButton"                    icon="ui-icon-
plus"   value="#{myBundle.Create}"
actionListener="#{clienteController.prepareCreate}"
update=":ClienteCreateForm"
oncomplete="PF('ClienteCreateDialog').show()"/>
    <p:commandButton id="viewButton"   style="visibility: hidden;"
icon="ui-icon-search" value="#{myBundle.View}"
update=":ClienteViewForm" oncomplete="PF('ClienteViewDialog').show()"
disabled="#{empty clienteController.selected}"/>
    <p:commandButton id="editButton"   style="visibility: hidden;"
icon="ui-icon-pencil" value="#{myBundle.Edit}" update=":ClienteEditForm"
oncomplete="PF('ClienteEditDialog').show()" disabled="#{empty
clienteController.selected}"/>
    <p:commandButton id="deleteButton" style="visibility: hidden;"
icon="ui-icon-trash"  value="#{myBundle.Delete}"
actionListener="#{clienteController.delete}" update=":growl,datalist"
disabled="#{empty clienteController.selected}">
        <p:confirm header="#{myBundle.ConfirmationHeader}"
message="#{myBundle.ConfirmDeleteMessage}" icon="ui-icon-alert"/>
    </p:commandButton>

  </p:panel>
  <ui:include src="/WEB-INF/include/confirmation.xhtml"/>
 </h:form>

</ui:composition>
```

Página Para ver el cliente

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">

    <ui:composition>

        <p:dialog id="ClienteViewDlg" widgetVar="ClienteViewDialog"
modal="true" resizable="false"  appendTo="@(body)"
header="#{myBundle.ViewClienteTitle}" closeOnEscape="true">

            <h:form id="ClienteViewForm">

                <h:panelGroup id="display">
                    <p:panelGrid columns="2" columnClasses="column"
rendered="#{clienteController.selected != null}">

                        <h:outputText
value="#{myBundle.ViewClienteLabel_cliId}"/>
                        <h:outputText value="#{clienteController.selected.cliId}"
title="#{myBundle.ViewClienteTitle_cliId}"/>

                        <h:outputText
value="#{myBundle.ViewClienteLabel_cliCedula}"/>
                        <h:outputText
value="#{clienteController.selected.cliCedula}"
title="#{myBundle.ViewClienteTitle_cliCedula}"/>

                        <h:outputText
value="#{myBundle.ViewClienteLabel_cliNombre}"/>
                        <h:outputText
value="#{clienteController.selected.cliNombre}"
title="#{myBundle.ViewClienteTitle_cliNombre}"/>

                        <h:outputText
value="#{myBundle.ViewClienteLabel_cliApellido}"/>
```

```
                <h:outputText
value="#{clienteController.selected.cliApellido}"
title="#{myBundle.ViewClienteTitle_cliApellido}"/>

                <h:outputText
value="#{myBundle.ViewClienteLabel_cliCiudad}"/>
                <h:outputText
value="#{clienteController.selected.cliCiudad}"
title="#{myBundle.ViewClienteTitle_cliCiudad}"/>

                <h:outputText
value="#{myBundle.ViewClienteLabel_cliFecNac}"/>
                <h:outputText
value="#{clienteController.selected.cliFecNac}"
title="#{myBundle.ViewClienteTitle_cliFecNac}">
                    <f:convertDateTime pattern="MM/dd/yyyy" />
                </h:outputText>

                <h:outputText
value="#{myBundle.ViewClienteLabel_cliTelefono}"/>
                <h:outputText
value="#{clienteController.selected.cliTelefono}"
title="#{myBundle.ViewClienteTitle_cliTelefono}"/>

                <h:outputText
value="#{myBundle.ViewClienteLabel_cliCelular}"/>
                <h:outputText
value="#{clienteController.selected.cliCelular}"
title="#{myBundle.ViewClienteTitle_cliCelular}"/>

                <h:outputText
value="#{myBundle.ViewClienteLabel_cliDireccion}"/>
                <h:outputText
value="#{clienteController.selected.cliDireccion}"
title="#{myBundle.ViewClienteTitle_cliDireccion}"/>

                <h:outputText
value="#{myBundle.ViewClienteLabel_cliMail}"/>
                <h:outputText value="#{clienteController.selected.cliMail}"
title="#{myBundle.ViewClienteTitle_cliMail}"/>
            </p:panelGrid>
```

```
                <p:commandButton value="#{myBundle.Close}"
onclick="PF('ClienteViewDialog').hide()"/>
            </h:panelGroup>

        </h:form>

    </p:dialog>

</ui:composition>

</html>
```

DOCTOR ROMEL MACHADO CLAVIJO,

SECRETARIO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION

DE LA UNIVERSIDAD DEL AZUAY,

C E R T I F I C A:

Que, el H. Consejo de Facultad de Ciencias de la Administración en sesión efectuada el 2 de octubre de 2014, conoció la petición Del señor **JONNATHAN BRAULIO MACHUCA RUBIO (46810)** que denuncia su trabajo de titulación denominado **"MANUAL PARA LA CREACION DE UNA APLICACION WEB CON EL USO DE LA LIBRERÍA PRIMEFACES"** presentado como requisito previo a la obtención del grado de Ingeniero de Sistemas y Telemática. El Consejo acoge el informe de la Junta Académica y designa como Director al Ing. Gabriel León y como miembro del Tribunal Examinador al Ing. Leopoldo Vázquez (MONOGRAFIA).

Cuenca, octubre 3 de 2014

UNIVERSIDAD DEL AZUAY
FACULTAD DE ADMINISTRACION
SECRETARIA

De conformidad a las disposiciones reglamentarias los denunciantes deberán presentar su trabajo de monografía en un plazo máximo de **TRES MESES** contados a partir de la fecha de aprobación, esto es hasta el 3 de enero de 2015

Facultad de Ciencias de la Administración
Escuela de Ingeniería de Sistemas y Telemática

Oficio Nro. 071-2014-DIST-UDA

Cuenca, 03 de Julio de 2014

Señor Ingeniero
Xavier Ortega Vázquez
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMNISTRACIÓN
Presente.-

De nuestras consideraciones:

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 01 de Julio del 2014, recibió el proyecto de monografía titulado "Manual para la creación de aplicaciones web con el uso de la librería pimefaces", presentada por la estudiante Braulio Machuca, estudiante de la Escuela de Ingeniería de Sistemas y Telemática y revisado por el Ing. Gabriel León (Docente del curso de graduación), previo a la obtención del título de Ingeniero de Sistemas y Telemática.

La Junta solicita por su digno intermedio notificar al tribunal designado y determinar lugar, fecha y hora de sustentación.

Por lo expuesto, y de conformidad con el Reglamento de Graduación de la Facultad, recomienda como director y responsable de aplicar cualquier modificación al diseño del trabajo de graduación posterior a al Ing. Gabriel León y como miembro del Tribunal a la Ing. Leopoldo Vazquez.

Atentamente,

Ing. Marcos Orellana Cordero
Director Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

CONVOCATORIA

Por disposición de la Junta Académica de Ingeniería de Sistemas y Telemática **CONVOCO** a los Miembros del Tribunal Examinador, a la sustentación del Protocolo del Trabajo de Titulación denominado: "**MANUAL PARA LA CREACION DE APLICACIONES WEB CON EL USO DE LA LIBRERÍA PRIMEFACES**" presentado por el señor **JONNATHAN BRAULIO MACHUCA RUBIO (46810),** previa a la obtención del grado de Ingeniero de Sistemas, para el día **LUNES 14 DE JULIO DE 2014, a las 18h30**

Cuenca, 9 de julio de 2014

Dr. Romel Machado Clavijo
Secretario de la Facultad

Ing. Gabriel León

Ing. Leopoldo Vázquez

Comunicado
alumno

Sustentación del Diseño de Monografía (Doctor Romel Machado Clavijo)

Fecha: 08-07-2014

ESCUELA DE INGENIERIA DE SISTEMAS

Diseños de Monografía
Escuela de Sistemas
Estudiante: Jonnathan Braulio Machuca Rubio con código 46810.
Tema: "MANUAL PARA LA CREACION DE APLICACIÓNES WEB CON EL USO DE LA LIBRERÍA PRIMEFACES"
Para: La obtención del título de Ingeniero en Sistemas
Director: Ing. Gabriel León.
Tribunal: Ing. Leopolodo Vázquez

DIA:

FECHA:

HORA:

1.

ACTA
SUSTENTACIÓN DE PROTOCOLO/DENUNCIA DEL TRABAJO DE TITULACIÓN

1.1 Nombre del estudiante: JONNATHAN BRAULIO MACHUCA RUBIO

1.1.1. Código 46810

1.1.2. **Director sugerido**: Ing. Gabriel León

1.1.3 Codirector (opcional): _____

1.2 Tribunal: Ing. Leopoldo Vázquez

***1.3* Título propuesto:** MANUAL PARA LA CREACION DE APLICACIONES WEB CON EL USO DE LA LIBRERÍA PRIMEFACES

1.4 Resolución:

1.4.1 Aceptado sin modificaciones _____

1.4.2 Aceptado con las siguientes modificaciones:

Cambio de titulo a: "Manual para la creación de una aplicación web con el uso de la libreria primefaces"

1.1.1 Responsable de dar seguimiento a las modificaciones (designado por la Junta Académica de entre los Miembros del Tribunal): Ing. Gabriel León

1.1.2 No aceptado

- Justificación:

_____
_____
_____

Tribunal

.....	.....
Ing. Gabriel León	Ing. Leopoldo Vázquez
.....	.....
Sr. Jonnathan B. Machuca R.	Secretario de Facultad
.....	.....

Fecha de sustentación: 14 de julio 2014

1

1.

RÚBRICA PARA LA EVALUACIÓN DEL PROTOCOLO DE TRABAJO DE TITULACIÓN

1.1Nombre del estudiante: JONNATHAN BRAULIO MACHUCA RUBIO
1.1.1. Código 46810
1.1.2. **1.2 Director sugerido**: Ing. Gabriel León
1.1.3. 1.3 **Codirector (opcional)**:
1.4. **Título propuesto:** MANUAL PARA LA CREACION DE APLICACIONES WEB CON EL USO DE LA LIBRERÍA PRIMEFACES
1.1 **1.5 Revisores (tribunal)**: Ing. Leopoldo Vázquez
1.2 **1.6 Recomendaciones generales de la revisión:**

	Cumple totalmente	Cumple parcialmente	No cumple	Observaciones (*)
Línea de investigación				
1. ¿El contenido se enmarca en la línea de investigación seleccionada?	X			
Título Propuesto				
2. ¿Es informativo?	X			
3. ¿Es conciso?	X			
Estado del arte				
4. ¿Identifica claramente el contexto histórico, científico, global y regional del tema del trabajo?	X			
5. ¿Describe la teoría en la que se enmarca el trabajo	X			
6. ¿Describe los trabajos relacionados más relevantes?	X			
7. ¿Utiliza citas bibliográficas?	X			
Problemática y/o pregunta de investigación				
8. ¿Presenta una descripción precisa y clara?	X			
9. ¿Tiene relevancia profesional y social?	X			
Hipótesis (opcional)				
10. ¿Se expresa de forma clara?	X			
11. ¿Es factible de verificación?	X			
Objetivo general				
12. ¿Concuerda con el problema formulado?	X			
13. ¿Se encuentra redactado en tiempo verbal infinitivo?	X			
Objetivos específicos				
14. ¿Concuerdan con el objetivo general?	X			
15. ¿Son comprobables cualitativa o cuantitativamente?	X			
Metodología				
16. ¿Se encuentran disponibles	X			

1

1.

los datos y materiales mencionados?				
17. ¿Las actividades se presentan siguiendo una secuencia lógica?	X			
18. ¿Las actividades permitirán la consecución de los objetivos específicos planteados?	X			
19. ¿Los datos, materiales y actividades mencionadas son adecuados para resolver el problema formulado?	X			
Resultados esperados				
20. ¿Son relevantes para resolver o contribuir con el problema formulado?	X			
21. ¿Concuerdan con los objetivos específicos?	X			
22. ¿Se detalla la forma de presentación de los resultados?	X			
23. ¿Los resultados esperados son consecuencia, en todos los casos, de las actividades mencionadas?	X			
Supuestos y riesgos				
24. ¿Se mencionan los supuestos y riesgos más relevantes?	X			
25. ¿Es conveniente llevar a cabo el trabajo dado los supuestos y riesgos mencionados?	X			
Presupuesto				
26. ¿El presupuesto es razonable?	X			
27. ¿Se consideran los rubros más relevantes?	X			
Cronograma				
28. ¿Los plazos para las actividades son realistas?	X			
Referencias				
29. ¿Se siguen las recomendaciones de normas internacionales para citar?	X			
Expresión escrita				
30. ¿La redacción es clara y fácilmente comprensible?	X			
31. ¿El texto se encuentra libre de faltas ortográficas?	X			

(*) Breve justificación, explicación o recomendación.

- Opcional cuando cumple totalmente,
- Obligatorio cuando cumple parcialmente y NO cumple.

2

1.

.....

.....

.....

.....

Ing. Gabriel León

Ing. Leopoldo Vázquez

Cuenca, 25 de julio del 2014

Ing. Xavier Ortega

Decano de la Facultad de Ciencias de la Administración

Ciudad

De mi consideración:

Suscribo e informo a Usted que he procedido a revisar el trabajo de diseño de investigación de tercer nivel titulado: **"Manual para la creación de una aplicación web con el uso de la librería primefaces."** presentado por el estudiante Jonnathan Braulio Machuca Rubio egresado de la Escuela de Sistemas y Telemática, como requisito previo a la obtención del Título de Ingeniero en Sistemas y Telemática, cumpliendo con los requisitos académicos, por lo que considero oportuno su aceptación. Finalmente informo a usted Sr. Decano que acepto la dirección del presente trabajo.

Sin más por el momento me despido de usted.

Atentamente

Ing. Gabriel León

Cuenca, 16 de julio de 2014

Yo, Gabriel Alejandro León Paredes con número de cédula 010365218-6 CERTIFICO que el estudiante Jonnathan Braulio Machuca Rubio con numero cédula 010496618-9 estudiante de la carrera de Ingeniería de Sistemas realizo los cambios en el diseño de monografía que se solicitaron hacer por parte del tribunal en el día de la sustentación.

El cambio realizado por parte del estudiante es en el titulo propuesto, el cual debe ir: *"Manual para la creación de una aplicación web con el uso de la librería Primefaces"*.

Ing. Gabriel León Paredes

010365218-6

UNIVERSIDAD DEL AZUAY

DOCTORA JENNY RIOS COELLO, SECRETARIA DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION DE LA UNIVERSIDAD DEL AZUAY

CERTIFICA:

Que, el Señor **Jonnathan Braulio Machuca Rubio,** registrado con código **46810** perteneciente a la Escuela de Ingeniería de Sistemas, luego de cumplir con todas las asignaturas de su Pensum de estudios, egresó de la Facultad el día 20 de Julio de 2013.

Cuenca, de Julio 01 de 2014

UNIVERSIDAD DEL AZUAY FACULTAD DE ADMINISTRACION SECRETARIA

Derecho No. 57220
vcf.-

Edición autorizada de 20.000 ejemplares
Del 618.501 al 638.500 Nº 0638084

UNIVERSIDAD DEL AZUAY

1. DATOS GENERALES

1.1 Nombre del estudiante:

Machuca Rubio Jonnathan Braulio.

1.2 Código: 46810

1.1.2 Contacto: 072889443, 0983846280

brauliomachuca@outlook.com.

1.2 Director sugerido: León Gabriel, Ing.

1.2.1 Contacto: 0998414482

gabreielleonp@hotmail.com

1.3 Tribunal designado:

1.4 Aprobación:

1.5 Línea de investigación de la carrera:

1203 Informática de Computadores
1203.17 Informática

1.6 Área de estudio: Desarrollo web con librerías Primefaces para java.

1.7 Título propuesto: Manual para la creación de una aplicación web con el uso de la librería primefaces.

1.8 Subtítulo: Manual para la creación de un sistema de facturación web.

2. CONTENIDO

2.1 Motivación de la investigación:

Existen diversas plataformas para el desarrollo web, en mucho de los casos tiene costo el uso de diferentes componentes para desarrollo de estas, en nuestro medio no existe suficiente información para el uso de código abierto en el campo de la programación web, debido a la falta de explotación de estos sistemas, pretendo crear un manual para una aplicación web de código abierto utilizando la librería primefaces de java, basándome en el desarrollo de un sistema de facturación.

2.2 Problemática:

Edición autorizada de 20.000 ejemplares
Del 638.501 al 658.500 Nº 0641664

Al momento de desarrollar una página web, varios son los factores que nos limitan a tomar una decisión para elegir la mejor opción, uno de estos factores son los costos al desarrollar nuestra aplicación, por otro lado también es la falta de componentes necesarios para la realización de dicho sistema, y por último pero no menos importante es la escases de documentación para la realización de un sistema paso o paso.

Cuando se trabaja con código abierto se disminuyen los costos para el desarrollador, inclusive se puede contar con una serie de componentes o librerías que se logran reutilizar para el desarrollo del sistema. Sin embargo la reutilización de estas librerías puede generar problemas con los cuales se necesite documentación o algún tipo de soporte para resolverlos en menor tiempo.

2.3 Pregunta de investigación:

¿Tenemos en nuestro medio un manual para creación de una aplicación web paso a paso con las librerías primefaces para JSF de Java?

2.4 Resumen:

El propósito de esta monografía es proporcionar una referencia útil para los interesados, tales como desarrolladores, estudiantes, docentes o personas que tengan algún conocimiento básico de desarrollo web.

Esta monografía abarca temas como: Descarga e instalación del software necesario, creación de proyecto, configuraciones de las respectivas conexiones, desarrollo de las interfaces gráficas y código fuente de la aplicación.

Finalmente el manual contará con las recomendaciones necesarias para facilitar así el trabajo de los desarrolladores con la librería Primefaces.

2.5 Estado del Arte y marco teórico:

2.5.1 Java.

Java es un lenguaje de programación de código abierto, orientado a objetos y una plataforma informática sacada al mercado por primera vez en los años 90 por Sun Microsystems, la cual posteriormente fue adquirida por la compañía Oracle. (Java, 2014)

2.5.2 JavaServer Faces (JSF)

JavaServer Faces es estándar de trabajo desarrollado por java orientada a componentes de interfaz de usuario para aplicaciones web de Java, es decir, es un framework para desarrollo web basado en el lenguaje de programación java.

Este estándar simplifica la construcción de interfaces de las aplicaciones web, la tecnología JSF, fue diseñado para ser flexible, esta aprovecha los estándar y conceptos web existentes, así una de las grandes ventajas de JavaServer Faces, es que nos permite tener un intérprete personalizado y una biblioteca propia de etiquetas JSP, una página jsf posee una extensión *.xhtml, es decir una unión de HTML y XML.

Edición autorizada de 20.000 ejemplares
Del 638.501 al 658.500 Nº 0641665

Una de las principales ventajas de JSF es su facilidad de uso, ya que tiene una arquitectura claramente separada entre la lógica de aplicación y presentación. Este diseño permite que cada integrante del desarrollo de la aplicación trabaje individualmente en su módulo, ya que esta tecnología proporciona una herramienta cómoda para la unión de cada uno de sus partes. (Fernando Pech-May, 2013)

2.5.3 Primefaces

PrimeFaces es una librería JavaScript de código abierto de componentes visuales, para su uso debemos únicamente adjuntar esta librería al proyecto y utilizar los componentes necesarios que nos ofrece esta librería, ya que por su facilidad de uso, hará que el trabajo del programador sea fácil y sencillo.

Primefaces es una librería sumamente ligera, a pesar de su gran número de componentes que nos permite utilizar, también posee una gran estabilidad ya que no depende de la funcionalidad de ningunas otras librerías, es decir, es totalmente independiente.

La principal característica de primefaces, es que los diseños de sus componentes son realmente estables, y la complejidad para usarlos es insignificante, dando así un trabajo de calidad y sin mayor dificultad para el programador. (Mert Caliskan, 2013)

2.6 Objetivo general:

Realizar un manual técnico para la creación de aplicaciones web con la librería Primefaces.

2.7 Objetivos específicos:

Diseñar los diagramas de clases y casos de uso para el sistema de facturación.

Diseñar e implementar una base de datos.

Desarrollar la aplicación web con arquitectura Modelo – vista – Controlador (MVC).

Utilizar la librería Primefaces para el manejo de las vistas de la aplicación web.

Crear el manual técnico de la aplicación web.

2.8 Metodología:

Para realizar esta monografía se emplea la investigación documental para plasmar los conocimientos técnicos que conlleva una aplicación java web con Primefaces.

Además se recurrirá a la investigación aplicada ya que con el desarrollo de la aplicación facturación se demostrará las utilidades que presenta la librería Primefaces.

2.9 Alcances y resultados esperados:

Al finalizar la monografía se obtendrá un manual que contenga una guía detallada para los interesados, empleando un lenguaje adecuado que facilite la comprensión a los lectores. Además logrará realizar una

aplicación demostrativa de facturación que evidenciara de manera práctica lo que está plasmado en el manual.

2.10 Supuestos y riesgos:

Uno de los principales riesgos que se pueden dar son las modificaciones en el transcurso del desarrollo de la aplicación, lo que ocasionaría que el tiempo previsto se prolongue. Para minimizar este riesgo se debe analizar detalladamente los requisitos oportunos para la aplicación de facturación.

Incompatibilidad del software desarrollado con algunos navegadores web.

2.11 Presupuesto:

Rubro denominación	Costo (Usd)	Justificación (¿para qué?)
Mensualidad de internet	$25.00	Para realizar las investigaciones necesarias, ya que esta es la fuente principal para nuestro trabajo.
Papel, esferos, grampas, etc.	$20	Estos son materias de oficina q se necesitaran durante el desarrollo del trabajo de graduación.

2.12 Financiamiento:

El financiamiento será propiamente auspiciado por el estudiante, ya que es la persona que será beneficiada con la elaboración de esta monografía.

2.13 Esquema tentativo:

2.13.1 Introducción.

2.13.2 Objetivos.

2.13.3 Resumen

2.13.4 Capitulo 1: Teoría y principios básicos.

2.13.4.1 Java

2.13.4.2 JSF.

2.13.4.3 Primefaces.

2.13.4.4 Arquitectura MVC

2.13.5 Capitulo 2: Análisis.

2.13.5.1 Diagramas UML

2.13.6 Capitulo 2: Instalación.

UNIVERSIDAD DEL AZUAY

2.13.6.1 Prerrequisitos.

2.13.6.2 Descargas.

2.13.6.3 Instalación.

2.13.7 Capitulo 3: Creación del proyecto.

2.13.7.1 Configuraciones.

2.13.7.2 Conexión con base de datos.

2.13.8 Capitulo 4: Modelo.

2.13.9 Creación de entidades

2.13.9 Capitulo 5: Controladores.

2.13.9.1 Creación de los beans.

2.13.10 Capitulo 6: Vistas.

2.13.10.1 Creación de las paginas xhtml con primefaces.

2.13.11 Mantenimiento de datos.

2.13.11.1 Insertar Datos.

2.13.11.2 Modificar Datos.

2.13.11.3 Eliminar Datos.

2.13.6.4 Tablas.

2.13.12 Menú de acceso.

2.13.13 Reportes estadísticos.

2.13.4 Exportación de datos.

2.14 Cronograma:

Objetivo especifico	Actividad	Resultados Esperados	Tiempo Semanas
Definir la especificación de requisitos de nuestra aplicación para la facturación.	Revisar e investigar diferentes sistemas de facturación ya realizados.	Definir correctamente nuestras especificaciones con el fin de realizar el manual de la aplicación.	2 semana
	Realizar Diagramas UML.		
Estructurar la base de datos a utilizar	Realizar el modelo entidad relación.	Definir correctamente nuestra base datos para la aplicación.	1 Semana.
Definir el diseño	Realizar un modboard	Definir un diseño	1 semana

	para nuestra aplicación	sencillo y eficiente para la explicación de nuestro manual	
Desarrollo de la aplicación.	Creación de las entidades. Creación de los controladores. Creación de las vistas. Documentar paso a paso cada las actividades.	Obtener nuestro manual basado en la aplicación desarrollada.	4 semanas.
Corregir errores.	Realizar pruebas y correcciones.	Obtener un software de calidad con un manual correctamente definido.	2 semanas.

2.15 Referencias:

Fernando Pech-May, ,. M.-R.-D.-J. (2013). *Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces.* Tabasco, México.

Java. (1 de 04 de 2014). *JAVA*. Obtenido de JAVA: https://www.java.com/es/download/faq/whatis_java.xml

javaserverfaces. (10 de 11 de 2012). *javaserverfaces*. Obtenido de javaserverfaces: http://www.javaserverfaces.org/

López, D. Á. (1 de 11 de 2011). *Universidad Oberta de Catalunya*. Obtenido de Universidad Oberta de Catalunya: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/11585/1/indianadani_TFC_0112.pdf

Mert Caliskan, O. V. (1 de 1 de 2013). *PrimeFaces Cookbook*. Packt Publishing Ltd. Obtenido de PrimeFaces.

ORACLE. (15 de 1 de 2014). *ORACLE*. Obtenido de ORACLE: http://www.oracle.com/

PrimeTek. (1 de 1 de 2014). *Primefaces*. Obtenido de Primefaces: http://www.primefaces.org/

UNIVERSIDAD DEL AZUAY

2.16 Firma de responsabilidad

Braulio Machuca Rubio

2.19 Firma de responsabilidad

Ing. Gabriel León.

2.20 Fecha de entrega: 29 Julio/2014

Edición autorizada de 20.000 ejemplares
Del 638.501 al 658.500 Nº 0641669

UNIVERSIDAD DEL AZUAY

Cuenca, 30 de junio del 2014

Ing. Xavier Ortega

Decano de la Facultad de Ciencias de la Administración

Ciudad

De mis consideraciones:

YO, JONNATHAN BRAULIO MACHUCA RUBIO con código 46810, estudiante de la Escuela de Sistemas y Telemática, solicito a usted de la manera más respetuosa y por su intermedio al Honorable Consejo de Facultad, se sirvan revisar mi diseño de tesis titulado: **"Manual para la creación de aplicaciones web con el uso de la librería primefaces"**, previo a la obtención del título de Ingeniero en Sistemas y Telemática.

Me permito sugerir el nombre del Ing. Gabriel León como director, el cual fue docente de la catedra de "Programación Web" en el curso de graduación, quién me ha asesorado en la elaboración del presente esquema y cuento con su aceptación.

Por la favorable acogida que se sirva a la presente, suscribo a usted.

Atentamente

Jonnathan Braulio Machuca Rubio
0104966189

Edición autorizada de 20.000 ejemplares
Del 618.501 al 638.500 Nº 0638078