

**Tema**

LOGIN EN JAVA

**AUTORES:**

BURGOS RIASCOS ADRIÁN ALEJANDRO

RIASCOS MORNEO ERICK GERMÁN

VERA CADENA ELKIN DAVID

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

08/02/2023

CONTENIDO

[MARCO TEÓRICO 5](#_Toc126764602)

[MYSQL 5](#_Toc126764603)

[CARACTERÍSTICAS DE MYSQL 5](#_Toc126764604)

[VENTAJAS DE USAR MYSQL 6](#_Toc126764605)

[HISTORIA Y VERSIONES DE MYSQL 6](#_Toc126764606)

[DDL Y DML 7](#_Toc126764607)

[PERSISTENCIA DE LA INFORMACIÓN 8](#_Toc126764608)

[MVC (DATA ACCESS PRESENTACIÓN DOMINIO) 8](#_Toc126764609)

[DESARROLLO 8](#_Toc126764610)

[DESCRIPCIÓN TALLER 8](#_Toc126764611)

[CREACIÓN Y CONFIGURACIÓN DEL PROYECTO 8](#_Toc126764612)

[CREACIÓN DE LA BASE DE DATOS 11](#_Toc126764613)

[CONFIGURACIÓN DE LA BASE DE DATOS CON NETBEANS 17](#_Toc126764614)

[MODELO 19](#_Toc126764615)

[EMPLEADO 19](#_Toc126764616)

[ESTADO 25](#_Toc126764617)

[USUARIO 30](#_Toc126764618)

[CONTROLADOR 37](#_Toc126764619)

[CONEXIÓN 37](#_Toc126764620)

[EMPLEADOCONTROLADOR 40](#_Toc126764621)

[USUARIOCONTROLADOR 45](#_Toc126764622)

[CONTROLADOR PRINCIPAL 55](#_Toc126764623)

[VISTA 60](#_Toc126764624)

[PÁGINA DE LOGIN 60](#_Toc126764625)

[PÁGINA DE REGISTRO 61](#_Toc126764626)

[PÁGINA DE INICIO 64](#_Toc126764627)

[PÁGINA DE CAMBIO DE CONTRASEÑA 65](#_Toc126764628)

[EJECUCIÓN 67](#_Toc126764629)

[Pantalla de inicio 67](#_Toc126764630)

[Pantalla de registro 68](#_Toc126764631)

[Pantalla de cambio de clave 69](#_Toc126764632)

[Pantalla de inicio 70](#_Toc126764633)

[CONCLUSIONES 70](#_Toc126764634)

[RECOMENDACIONES 70](#_Toc126764635)

[REFERENCIAS 70](#_Toc126764636)

**ÍNDICE DE FIGURAS**

[Figura 1. Historia MySQL 7](#_Toc126764525)

[Figura 2 Creación de nuevo proyecto 9](#_Toc126764526)

[Figura 3 Selección del tipo de proyecto 9](#_Toc126764527)

[Figura 4 Asignación del nombre del proyecto 10](#_Toc126764528)

[Figura 5 Configuración del proyecto 10](#_Toc126764529)

[Figura 6 Diagrama ER 11](#_Toc126764530)

[Figura 7 Script generado por el case 12](#_Toc126764531)

[Figura 8 Página principal de AppServ 13](#_Toc126764532)

[Figura 9 Inicio de sesión phpMyAdmin 14](#_Toc126764533)

[Figura 10 Pantalla de inicio phpMyAdmin 15](#_Toc126764534)

[Figura 11 Creación de nueva base de datos 15](#_Toc126764535)

[Figura 12 Entrada de texto para código SQL 16](#_Toc126764536)

[Figura 13 Tablas generadas por script 16](#_Toc126764537)

[Figura 14 Creación de nueva conexión 17](#_Toc126764538)

[Figura 15 Selección del controlador de la base de datos 17](#_Toc126764539)

[Figura 16 Configuracióon del controlador 18](#_Toc126764540)

[Figura 17 Contorlador agregado 18](#_Toc126764541)

[Figura 18 Creación de nuevo archivo 19](#_Toc126764542)

[Figura 19 Selección de JSF Managed Bean 20](#_Toc126764543)

[Figura 20 Asignación del nombre del Bean 21](#_Toc126764544)

[Figura 21 Nombre y atributos del Bean creado 22](#_Toc126764545)

[Figura 22 Funciones del Bean 23](#_Toc126764546)

[Figura 23 Creación de nuevo archivo 26](#_Toc126764547)

[Figura 24 Selección de JSF Managed Bean 27](#_Toc126764548)

[Figura 25 Asignación del nombre del Bean 27](#_Toc126764549)

[Figura 26 Nombre y atributos del Bean creado 28](#_Toc126764550)

[Figura 27 Funciones del Bean 28](#_Toc126764551)

[Figura 28 Creación de nuevo archivo 30](#_Toc126764552)

[Figura 29 Selección de JSF Managed Bean 31](#_Toc126764553)

[Figura 30 Asignación del nombre del Bean 32](#_Toc126764554)

[Figura 31 Nombre y atributos del Bean creado 33](#_Toc126764555)

[Figura 32 Funciones del Bean 34](#_Toc126764556)

[Figura 33 Crecación de nuevo archivo 37](#_Toc126764557)

[Figura 34 Creación de archivo Java Class 38](#_Toc126764558)

[Figura 35 Atributos y constructor del controlador coenxión 38](#_Toc126764559)

[Figura 36 Método que permite la conexión con la base de datos 39](#_Toc126764560)

[Figura 37 Método que retorna la instancia 39](#_Toc126764561)

[Figura 38 Creación de nuevo archivo 41](#_Toc126764562)

[Figura 39 Creación de archivo Java Class 42](#_Toc126764563)

[Figura 40 Atributos y constructor del controlador 42](#_Toc126764564)

[Figura 41 Método para insertar nuevo empleado 43](#_Toc126764565)

[Figura 42 Método para la obtención de un empleado 43](#_Toc126764566)

[Figura 43 Creación de nuevo archivo 45](#_Toc126764567)

[Figura 44 Creación de archivo Java Class 46](#_Toc126764568)

[Figura 45 Atributos y constructor del controlador 46](#_Toc126764569)

[Figura 46 Método que genera clave temporal 47](#_Toc126764570)

[Figura 47 Método para enviar la clave al correo 47](#_Toc126764571)

[Figura 48 Método que encripta la contraseña 48](#_Toc126764572)

[Figura 49 Método para ingresar un nuevo usuario en la base de datos 48](#_Toc126764573)

[Figura 50 Méotodo para validar el correo 49](#_Toc126764574)

[Figura 51 Método encargado de efectura el login 49](#_Toc126764575)

[Figura 52 Método que permite la actualización de la contraseña 50](#_Toc126764576)

[Figura 53 Creación de nuevo archivo 55](#_Toc126764577)

[Figura 54 Creación de nuevo archivo Java Class 56](#_Toc126764578)

[Figura 55 Atributos y constructor del controlador 56](#_Toc126764579)

[Figura 56 Método de registro 57](#_Toc126764580)

[Figura 57 Méotodo de login 57](#_Toc126764581)

[Figura 58 Método para actualizar la clave 58](#_Toc126764582)

[Figura 59 Código del index.xhtml 60](#_Toc126764583)

[Figura 60 Código de registro.xhtml 62](#_Toc126764584)

[Figura 61 Código de InicioUsuario.xhtml 64](#_Toc126764585)

[Figura 62 Código de CambiarClave.xhtml 65](#_Toc126764586)

[Figura 63 Pantalla de inicio 67](#_Toc126764587)

[Figura 64 Pantalla de registro 68](#_Toc126764588)

[Figura 65 Pantalla de cambio de contraseña 69](#_Toc126764589)

[Figura 66 Pantalla de inicio 70](#_Toc126764590)

**ÍNDICE DE TABLAS**

[Tabla 1 Código del bean Empleado 25](#_Toc126764591)

[Tabla 2 Código del bean Estado 29](#_Toc126764592)

[Tabla 3 Código del bean Usuario 37](#_Toc126764593)

[Tabla 4 Código del controlador conexión 40](#_Toc126764594)

[Tabla 5 Código del controlador del empleado 44](#_Toc126764595)

[Tabla 6 Código del controlador usuario 54](#_Toc126764596)

[Tabla 7 Código del controlador principal 59](#_Toc126764597)

[Tabla 8 Código index.xhtml 61](#_Toc126764598)

[Tabla 9 Código de Registro.xhtml 64](#_Toc126764599)

[Tabla 10 Código de InicioUsuario.xhtml 65](#_Toc126764600)

[Tabla 11 Código CambiarClave.xhtml 66](#_Toc126764601)

# MARCO TEÓRICO

## MYSQL

En programación es prácticamente inevitable trabajar con algún tipo de sistema de gestión de bases de datos. Cualquier programa que tarde o temprano necesitará almacenar datos en algún lugar, como mínimo para poder almacenar la lista de usuarios autorizados, sus permisos y propiedades.

MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basada en código abierto. Desarrollado originalmente por MySQL AB, fue adquirida por Sun MicroSystems en 2008 y esta su vez comprada por Oracle Corporation en 2010, la cual ya era dueña de un motor propio InnoDB para MySQL.

MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle.

Las versiones Enterprise, diseñadas para aquellas empresas que quieran incorporarlo en productos privativos, incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial.

## CARACTERÍSTICAS DE MYSQL

MySQL presenta algunas ventajas que lo hacen muy interesante para los desarrolladores. La más evidente es que trabaja con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente.

Al ser basada en código abierto es fácilmente accesible y la inmensa mayoría de programadores que trabajan en desarrollo web han pasado usar MySQL en alguno de sus proyectos porque al estar ampliamente extendido cuenta además con una ingente comunidad que ofrece soporte a otros usuarios. Pero estas no son las únicas características como se vera a continuación:

Arquitectura Cliente y Servidor: MySQL basa su funcionamiento en un modelo cliente y servidor. Es decir, clientes y servidores se comunican entre sí de manera diferenciada para un mejor rendimiento. Cada cliente puede hacer consultas a través del sistema de registro para obtener datos, modificarlos, guardar estos cambios o establecer nuevas tablas de registros, por ejemplo.

Compatibilidad con SQL: SQL es un lenguaje generalizado dentro de la industria. Al ser un estándar MySQL ofrece plena compatibilidad por lo que si has trabajado en otro motor de bases de datos no tendrás problemas en migrar a MySQL.

Vistas: Desde la versión 5.0 de MySQL se ofrece compatibilidad para poder configurar vistas personalizadas del mismo modo que se podra hacerlo en otras bases de datos SQL. En bases de datos de gran tamaño las vistas se hacen un recurso imprescindible.

Procedimientos almacenados. MySQL posee la característica de no procesar las tablas directamente sino que a través de procedimientos almacenados es posible incrementar la eficacia de la implementación.

Desencadenantes. MySQL permite además poder automatizar ciertas tareas dentro de la base de datos. En el momento que se produce un evento otro es lanzado para actualizar registros o optimizar su funcionalidad.

Transacciones. Una transacción representa la actuación de diversas operaciones en la base de datos como un dispositivo. El sistema de base de registros avala que todos los procedimientos se establezcan correctamente o ninguna de ellas. En caso por ejemplo de una falla de energía, cuando el monitor falla u ocurre algún otro inconveniente, el sistema opta por preservar la integridad de la base de datos resguardando la información.

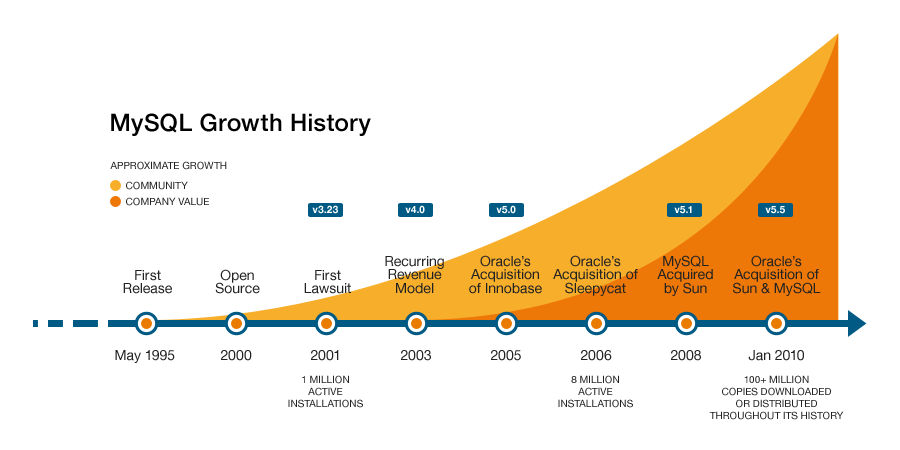
## VENTAJAS DE USAR MYSQL

Descritas las principales características de MySQL es fácil ver sus ventajas. MySQL es una opción razonable para ser usado en ámbito empresarial. Al estar basado en código abierto permite a pequeñas empresas y desarrolladores disponer de una solución fiable y estandarizada para sus aplicaciones. Por ejemplo, si se cuenta con un listado de clientes, una tienda online con un catálogo de productos o incluso una gran selección de contenidos multimedia disponible, MySQL ayuda a gestionarlo todo debida y ordenadamente.

## HISTORIA Y VERSIONES DE MYSQL

La historia de MySQL comienza cuando Michael Widenius, un programador de software, buscaba una solución sencilla que permitiera dar salida a sus proyectos sin recurrir a soluciones de terceros. Al comienzo MySQL carecía de elementos considerados esenciales como la integridad referencial y transacciones pero a pesar de ello atrajo rápidamente a desarrolladores web.

Esto dió lugar a la fundación de la empresa MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius) para dar continuidad al proyecto. Posteriormente la empresa fue adquirida por Sun MicroSystems en 2008 quien vió en MySQL una solución para competir con su competidor Oracle quien, curiosamente, acabó por hacerse con la compañía en 2010.



**Figura 1. Historia MySQL**

Desde 1995, momento en el que la compañía de software de MySQL AB creó la versión original hasta hoy MySQL ha ido evolucionando, incorporando nuevas funciones. En 1999, se lanzó la versión 3.23.0, la primera versión en importancia a la que luego siguieron otras como las versiones 5.0.0, liberada en 2005, y la 5.5.x que fue lanzada en diciembre de 2009.

## DDL Y DML

Las sentencias SQL se dividen en dos grandes categorías; la primera denominada Lenguaje de Definición de Datos (Data definition Language DDL) y Lenguaje de Manipulación de Datos (Data Manipulation Language DML)

Lenguaje de definición de datos (DDL)

• CREATE - para crear objetos en la base de datos.

• ALTER - modifica la estructura de la base de datos.

• DROP - borra objetos de la base de datos.

• TRUNCATE - elimina todos los registros de la tabla, incluyendo todos los espacios asignados a los registros.

Lenguaje de manipulación de datos (DML)

• SELECT - para obtener datos de una base de datos.

• INSERT - para insertar datos a una tabla.

• UPDATE - para modificar datos existentes dentro de una tabla.

• DELETE - elimina todos los registros de la tabla; no borra los espacios asignados a los registros. [3]

## PERSISTENCIA DE LA INFORMACIÓN

Para el almacenamiento de información persistente ante diferentes ejecuciones, no cabe la posibilidad de almacenarla en memoria del programa ya sea estática o dinámica, porque tras la terminación de ejecución del programa, dicha información desaparece o aún peor, queda inaccesible consumiendo memoria (en el caso de las fugas de memoria). Por ello, para el almacenamiento de información persistente, se deben utilizar otros métodos, tales como los siguientes:

• Almacenamiento en una Base de datos

Son utilidades que permiten el almacenamiento persistente a través de una serie tablas. Permiten definir tablas, sus primary keys, las relaciones entre tablas, etc. y están optimizadas para realizar operaciones de datos a gran velocidad.

• Almacenamiento en ficheros

Se puede utilizar uno o varios ficheros para almacenar información de manera persistente entre diferentes ejecuciones de programa. Los ficheros en sí, sobrevivirán a diferentes ejecuciones del programa, por lo que la información será persistente, y se podrá leer y escribir en diferentes ejecuciones. Si se utiliza esta opción, se debe seguir un criterio a la hora de almacenar los datos en un fichero, de forma que se puedan recuperar posteriormente siguiendo el mismo criterio.

## MVC (DATA ACCESS PRESENTACIÓN DOMINIO)

Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

• DataAccess: Contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

• Presentación: También conocido como interfaz de usuario, compone la información que se envía al cliente y los mecanismos interacción con éste.

• Dominio: Actúa como intermediario entre el DataAccess y la Presentación, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

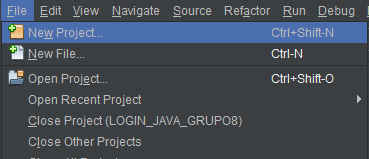
# DESARROLLO

## DESCRIPCIÓN TALLER

Implementar un login y un registro en java, que encripte la contraseña del usuario y envíe una clave temporal al correo electrónico.

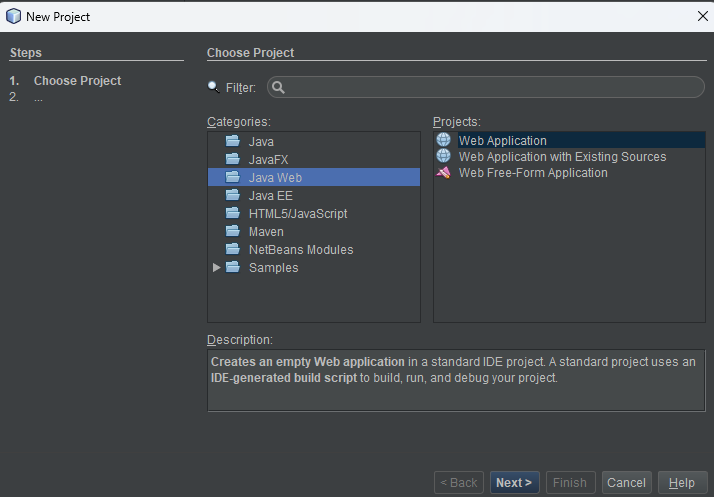
## CREACIÓN Y CONFIGURACIÓN DEL PROYECTO

Para la creación del proyecto, se requiere de una aplicación web de java, para ello, se debe hacer click en el apartado de Archivo o file, y se selecciona la opción nuevo proyecto.



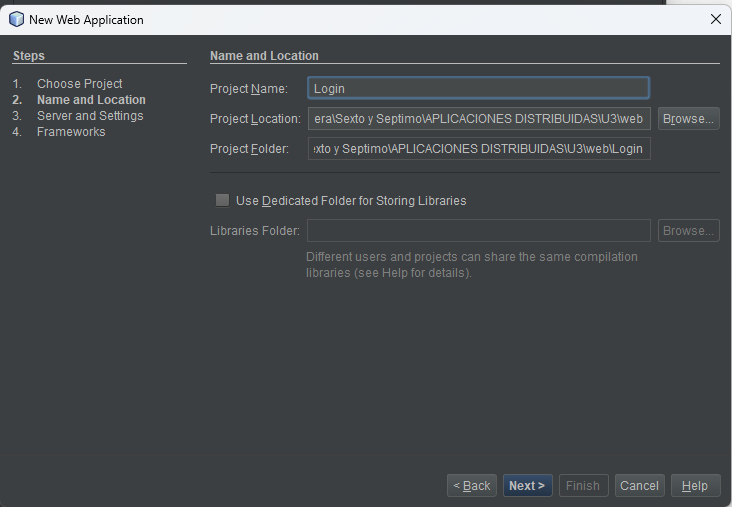
**Figura 2 Creación de nuevo proyecto**

Se debe seleccionar una aplicación de tipo Web application.



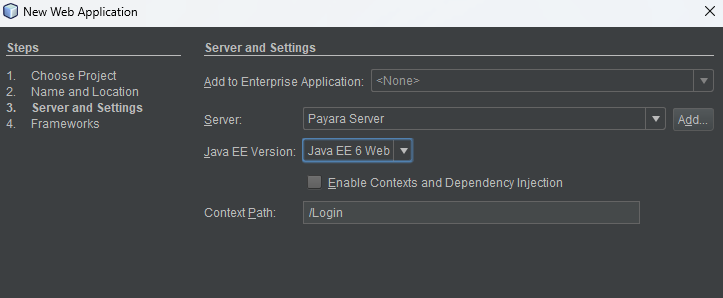
**Figura 3 Selección del tipo de proyecto**

Seguido a eso, se asigna un nombre al proyecto y se hace click en next para poder configurar los frameworks que serán usados.



**Figura 4 Asignación del nombre del proyecto**

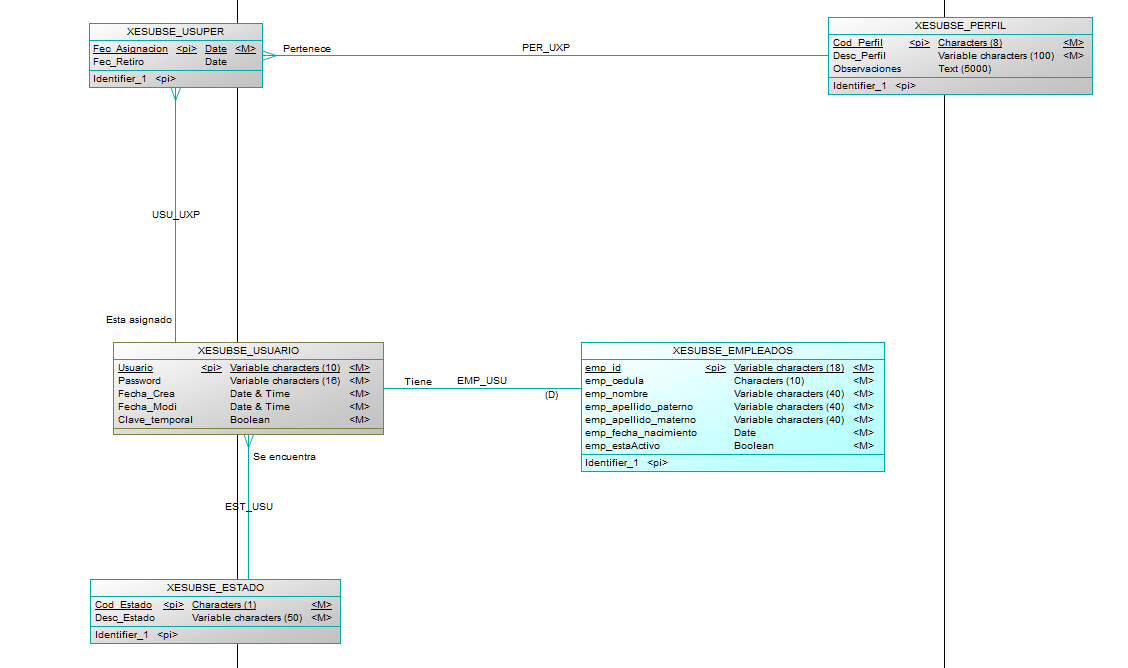
Con la configuración que se muestra en la siguiente figura, se hace click en finalizar para que se cree el proyecto.



**Figura 5 Configuración del proyecto**

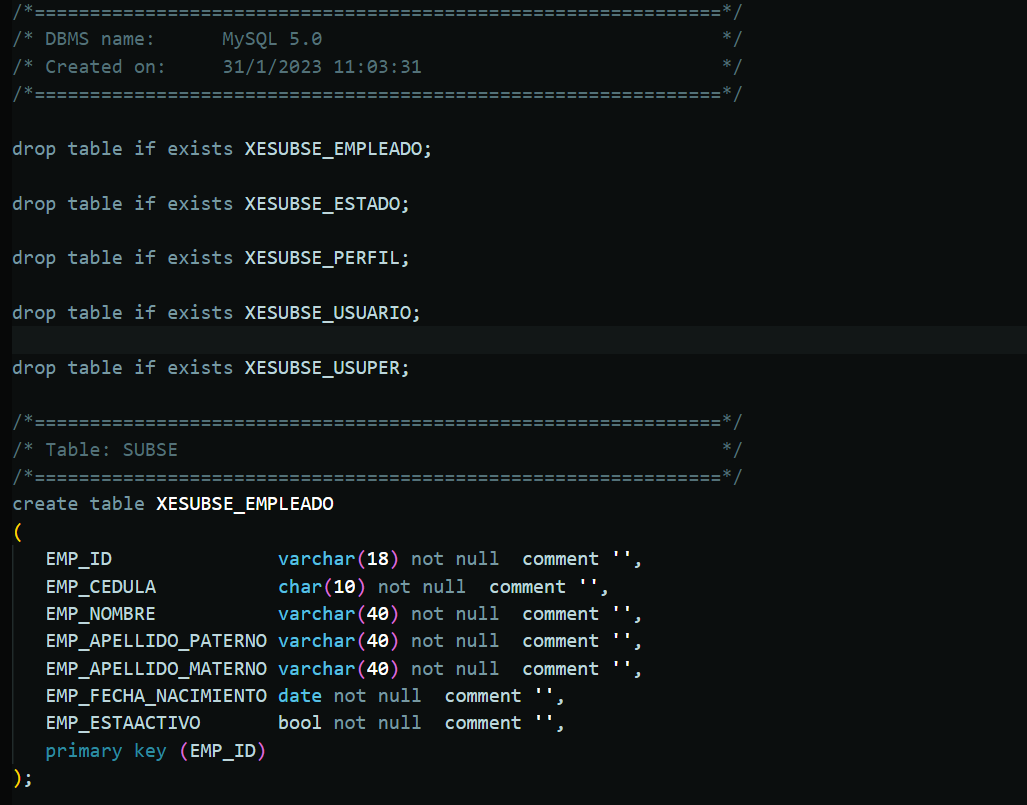
### CREACIÓN DE LA BASE DE DATOS

A continuación, se muestra el diagrama ER de la base de datos que será usada en el proyecto, aplicando la capa de seguridad.



**Figura 6 Diagrama ER**

Para poder crear la base de datos dentro del servidor apache AppServ, se debe tener creado el modelo lógico y exportar el script creado por el case.



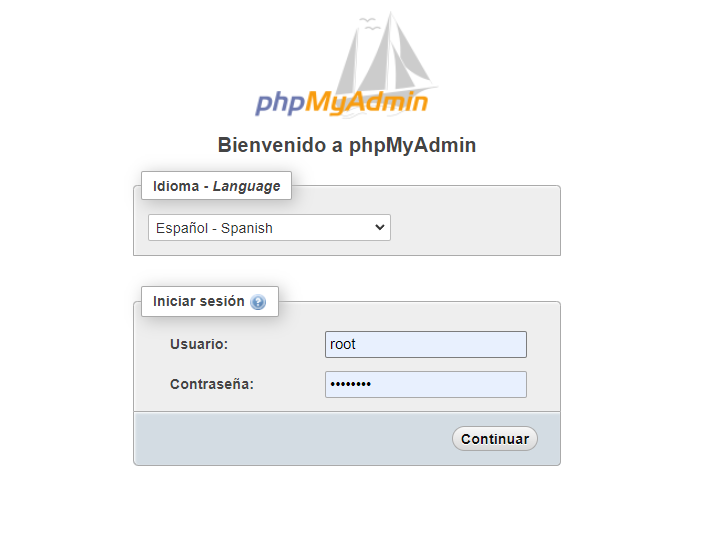
**Figura 7 Script generado por el case**

Una vez realizado esto, se debe dirigir a la ubicación del AppServ, que en este caso es localhost:80 y se tendrá la siguiente pantalla.



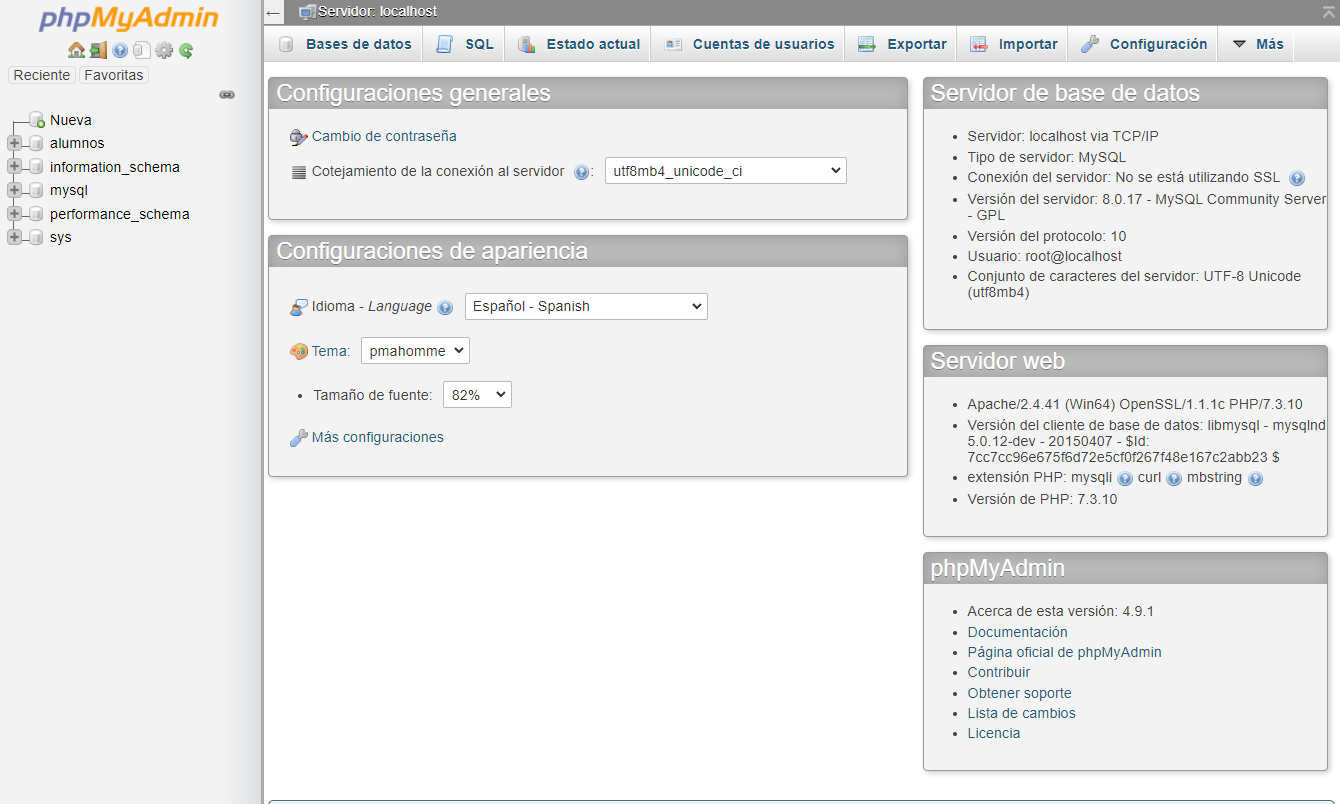
**Figura 8 Página principal de AppServ**

Se selecciona la opción phpMyAdmin Database Manager Version 4.9.1 para poder acceder a las bases de datos contenidas en el servidor.



**Figura 9 Inicio de sesión phpMyAdmin**

Una vez que se ingresan las credenciales correctas, se tendrá la siguiente pantalla que consiste en el panel de trabajo del servidor, a la izquierda se tienen las bases de datos creadas.



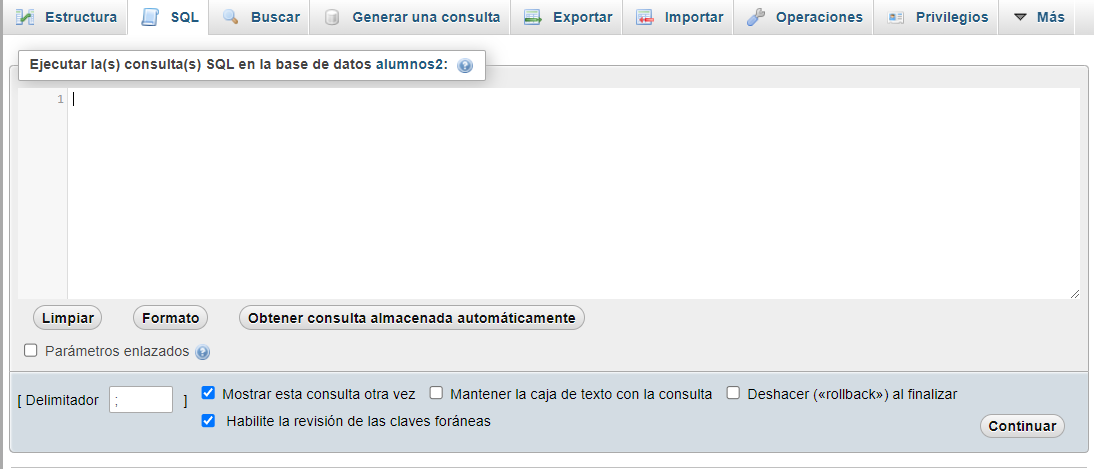
**Figura 10 Pantalla de inicio phpMyAdmin**

Para crear la base de datos a usar en el proyecto, se hace click en nueva, y de deberá ingresar el nombre de la base de datos y en el tipo de formato se debe ingresar utf8\_spanish\_ci para que permita el ingreso de caracteres en español.



**Figura 11 Creación de nueva base de datos**

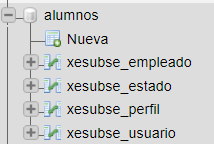
Ahora, es turno de crear el esquema de la base de datos en base al script generado por el case previamente mencionado, para ella se deberá hacer click en la opción superior SQL, lo que mostrará una entrada de texto, específicamente para consultas SQL.



**Figura 12 Entrada de texto para código SQL**

Y ahora solo se debe copiar y pegar el código SQL y al hacer click en continuar se mostrará un mensaje indicando que la creación se realizó con éxito.

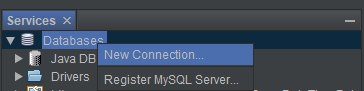
Finalmente, quedará la estructura de la base de datos de la siguiente manera.



**Figura 13 Tablas generadas por script**

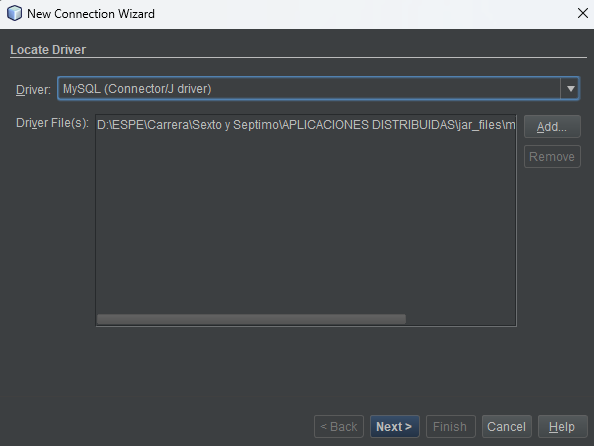
### CONFIGURACIÓN DE LA BASE DE DATOS CON NETBEANS

Para poder crear la conexión de netbeans con mysql, se debe hacer click derecho en la opción DataBase y se selecciona en Nueva Conexión



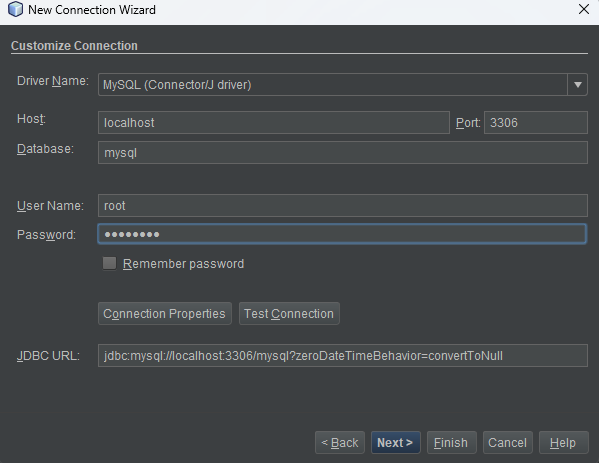
**Figura 14 Creación de nueva conexión**

Se deberá seleccionar la opción de MySQL y en driver file se tendrá que seleccionar el driver mysql-connector-j-8.0.3.1.jar



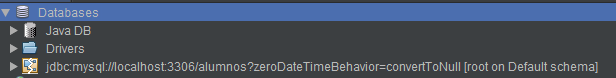
**Figura 15 Selección del controlador de la base de datos**

Ahora se debe terminar la configuración de la conexión, para ello, se debe ingresar las credenciales del AppServ y se da click en finalizar.



**Figura 16 Configuracióon del controlador**

Como se puede apreciar, ya se cuenta con la conexión creada previamente.

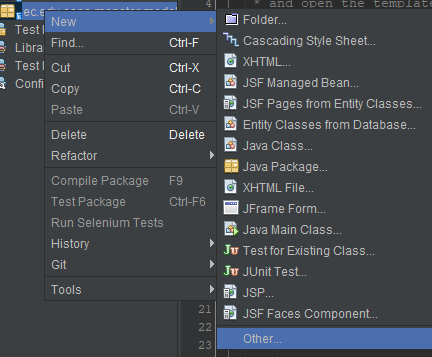


**Figura 17 Contorlador agregado**

## MODELO

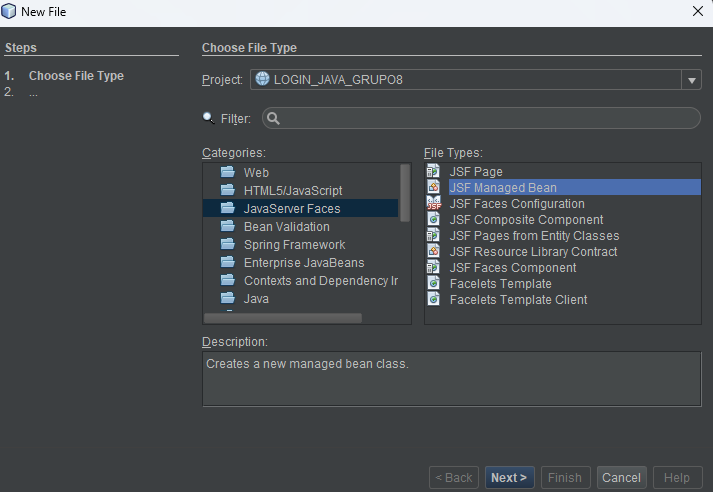
### EMPLEADO

Para crear el modelo empleado, se debe crear un ManagedBean, por que lo se debe hacer click derecho sobre el paquete de modelos y en la opción otros se desplegará un conjunto de categorías y de tipos de archivos a crear.



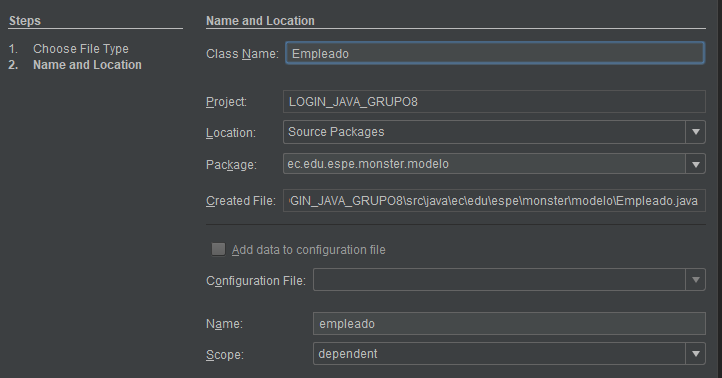
**Figura 18 Creación de nuevo archivo**

En la categoría JavaServerFaces se selecciona la opción JSF ManagedBean.



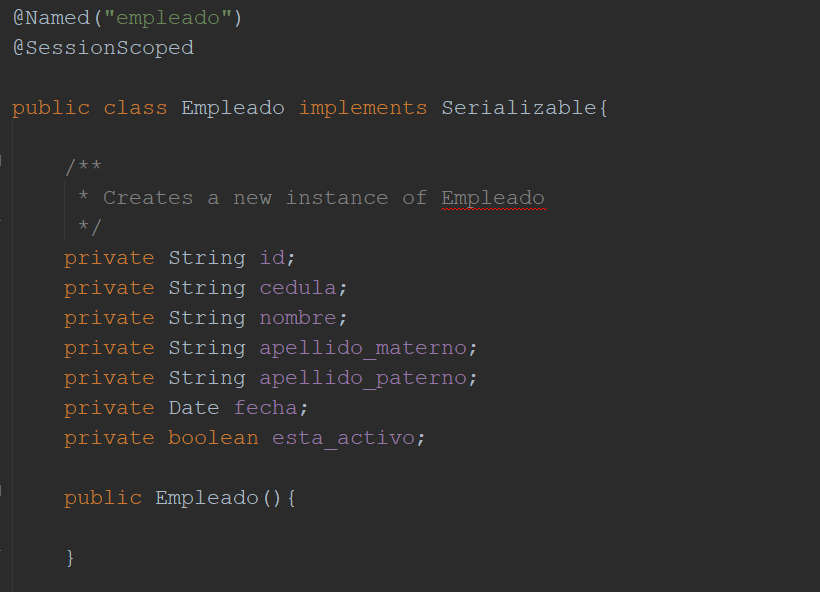
**Figura 19 Selección de JSF Managed Bean**

Y se le asigna el nombre del modelo, en este caso, Empleado, y se le da click en finish para terminar la creación del Bean.



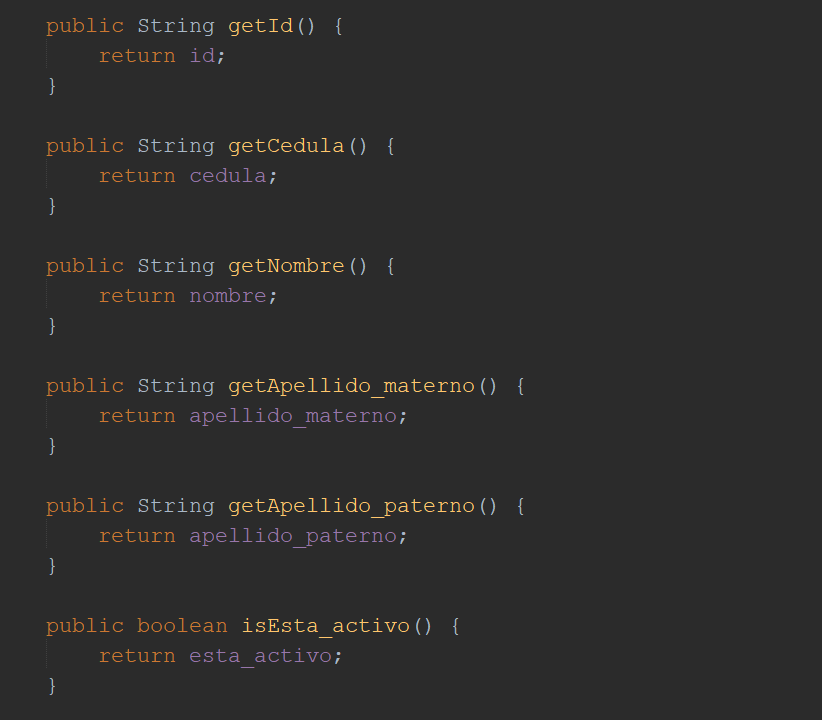
**Figura 20 Asignación del nombre del Bean**

Es importante tener nombre al bean con el nombre ideal para poder ser reconocido durante la ejecución del aplicativo, para este caso, el nombre será empleado, también se agrega como atributos, los nombres correspondientes a las columnas de la tabla a la cual el bean está siendo referenciado y como punto final, un bean debe tener un constructor vacío.



**Figura 21 Nombre y atributos del Bean creado**

Al tratarse de un bean, los métodos que existen en él, deberán ser getters y setter para el manejo de los atributos.



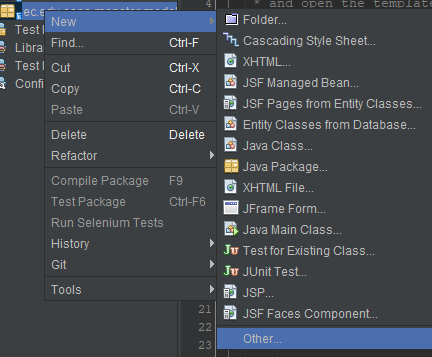
**Figura 22 Funciones del Bean**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package ec.edu.espe.monster.modelo;  import java.io.Serializable;  import java.sql.Date;  import javax.enterprise.context.\*;  import javax.inject.Named;  /\*\*  \*  \* @author aburg  \*/  @Named("empleado")  @SessionScoped  public class Empleado implements Serializable{  /\*\*  \* Creates a new instance of Empleado  \*/  private String id;  private String cedula;  private String nombre;  private String apellido\_materno;  private String apellido\_paterno;  private Date fecha;  private boolean esta\_activo;  public Empleado(){    }    /\*public Empleado(String id, String cedula, String nombre, String apellido\_materno, String apellido\_paterno, Date fecha, boolean esta\_activo) {  this.id = id;  this.cedula = cedula;  this.nombre = nombre;  this.apellido\_materno = apellido\_materno;  this.apellido\_paterno = apellido\_paterno;  this.fecha = fecha;  this.esta\_activo = esta\_activo;  }\*/  public String getId() {  return id;  }  public String getCedula() {  return cedula;  }  public String getNombre() {  return nombre;  }  public String getApellido\_materno() {  return apellido\_materno;  }  public String getApellido\_paterno() {  return apellido\_paterno;  }  public boolean isEsta\_activo() {  return esta\_activo;  }  public void setId(String id) {  this.id = id;  }  public void setCedula(String cedula) {  this.cedula = cedula;  }  public void setNombre(String nombre) {  this.nombre = nombre;  }  public void setApellido\_materno(String apellido\_materno) {  this.apellido\_materno = apellido\_materno;  }  public void setApellido\_paterno(String apellido\_paterno) {  this.apellido\_paterno = apellido\_paterno;  }  public void setEsta\_activo(boolean esta\_activo) {  this.esta\_activo = esta\_activo;  }  public Date getFecha() {  return fecha;  }  public void setFecha(Date fecha) {  this.fecha = fecha;  }  } |

**Tabla 1 Código del bean Empleado**

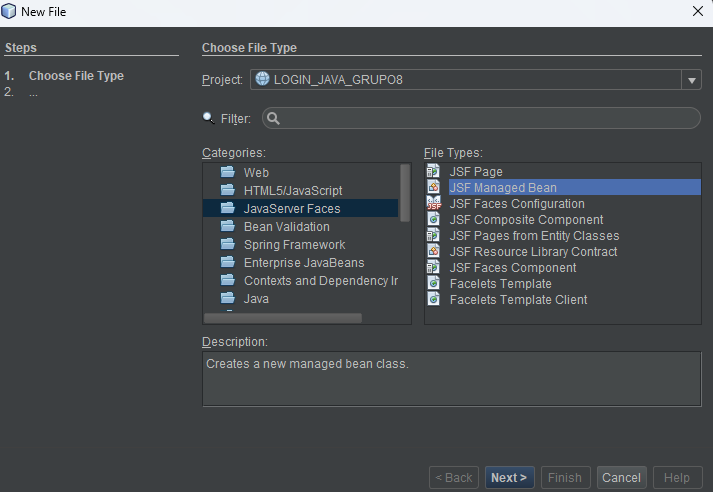
### ESTADO

Para crear el modelo estado, se debe crear un ManagedBean, porque lo se debe hacer click derecho sobre el paquete de modelos y en la opción otros se desplegará un conjunto de categorías y de tipos de archivos a crear.



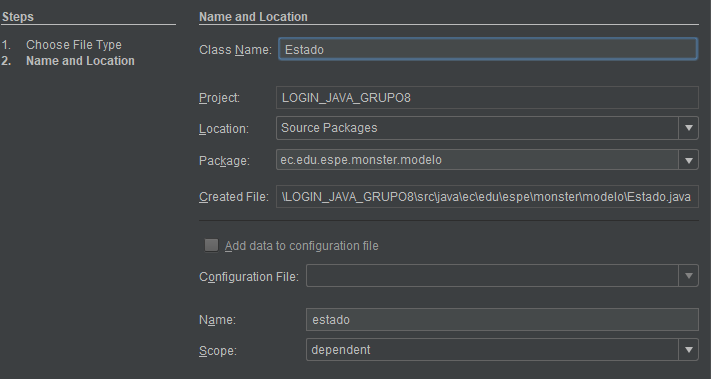
**Figura 23 Creación de nuevo archivo**

En la categoría JavaServerFaces se selecciona la opción JSF ManagedBean.



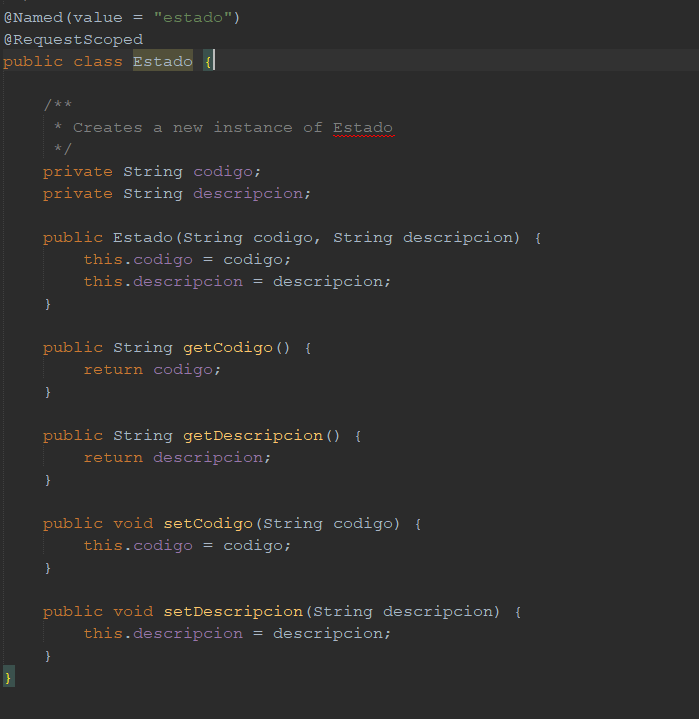
**Figura 24 Selección de JSF Managed Bean**

Y se le asigna el nombre del modelo, en este caso, Estado, y se le da click en finish para terminar la creación del Bean.



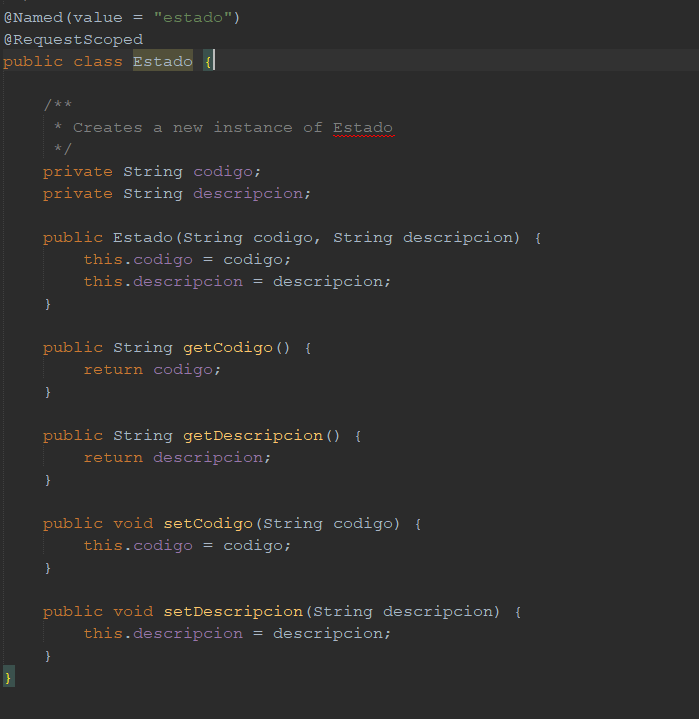
**Figura 25 Asignación del nombre del Bean**

Es importante tener nombre al bean con el nombre ideal para poder ser reconocido durante la ejecución del aplicativo, para este caso, el nombre será estado, también se agrega como atributos, los nombres correspondientes a las columnas de la tabla a la cual el bean está siendo referenciado y como punto final, un bean debe tener un constructor vacío.



**Figura 26 Nombre y atributos del Bean creado**

Al tratarse de un bean, los métodos que existen en él, deberán ser getters y setter para el manejo de los atributos.

****

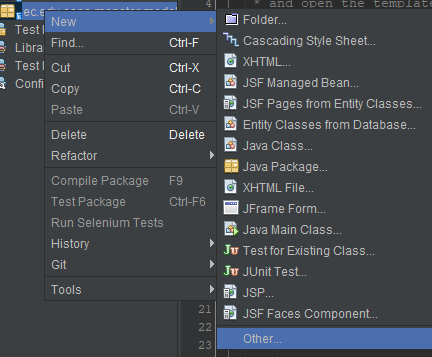
**Figura 27 Funciones del Bean**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package ec.edu.espe.monster.modelo;  import javax.enterprise.context.RequestScoped;  import javax.inject.Named;  /\*\*  \*  \* @author aburg  \*/  @Named(value = "estado")  @RequestScoped  public class Estado {  /\*\*  \* Creates a new instance of Estado  \*/  private String codigo;  private String descripcion;  public Estado(String codigo, String descripcion) {  this.codigo = codigo;  this.descripcion = descripcion;  }  public String getCodigo() {  return codigo;  }  public String getDescripcion() {  return descripcion;  }  public void setCodigo(String codigo) {  this.codigo = codigo;  }  public void setDescripcion(String descripcion) {  this.descripcion = descripcion;  }  } |

**Tabla 2 Código del bean Estado**

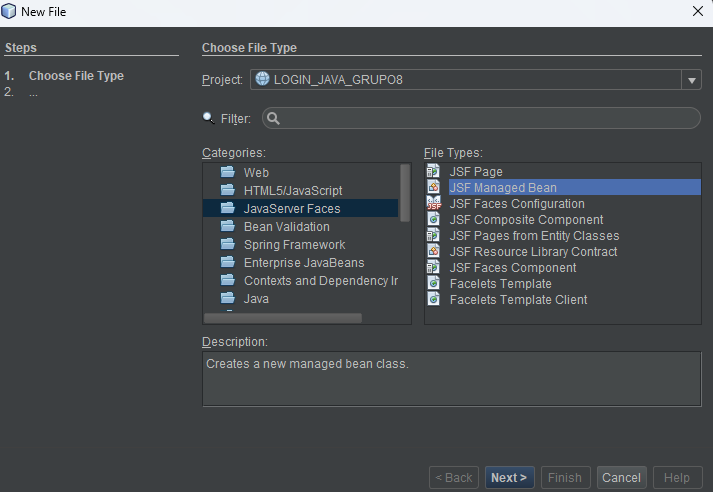
### USUARIO

Para crear el modelo estado, se debe crear un ManagedBean, porque lo se debe hacer click derecho sobre el paquete de modelos y en la opción otros se desplegará un conjunto de categorías y de tipos de archivos a crear.



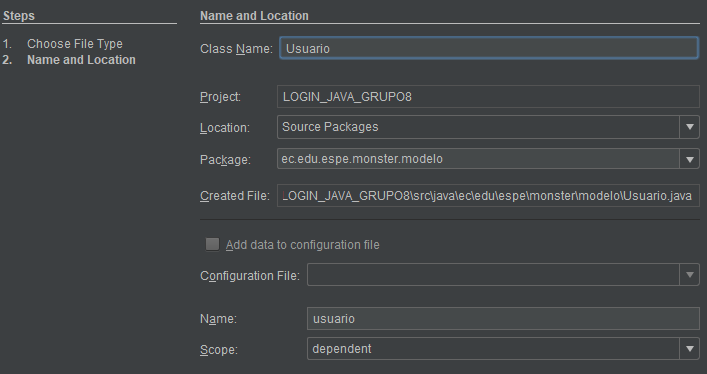
**Figura 28 Creación de nuevo archivo**

En la categoría JavaServerFaces se selecciona la opción JSF ManagedBean.



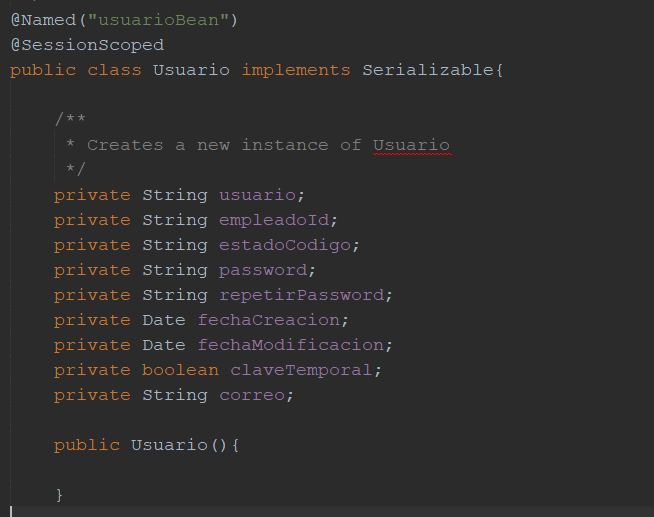
**Figura 29 Selección de JSF Managed Bean**

Y se le asigna el nombre del modelo, en este caso, Usuario, y se le da click en finish para terminar la creación del Bean.



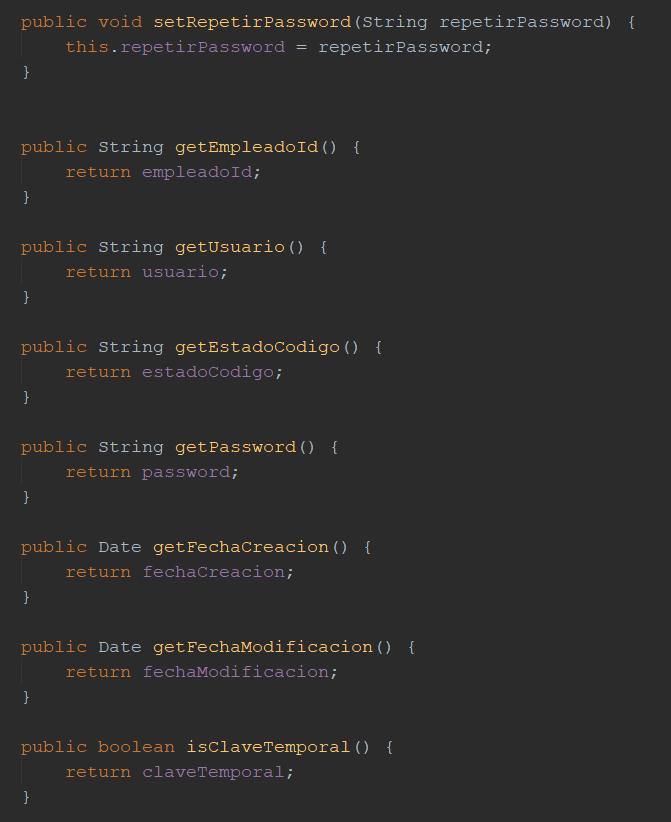
**Figura 30 Asignación del nombre del Bean**

Es importante tener nombre al bean con el nombre ideal para poder ser reconocido durante la ejecución del aplicativo, para este caso, el nombre será usuario, también se agrega como atributos, los nombres correspondientes a las columnas de la tabla a la cual el bean está siendo referenciado y como punto final, un bean debe tener un constructor vacío.



**Figura 31 Nombre y atributos del Bean creado**

Al tratarse de un bean, los métodos que existen en él, deberán ser getters y setter para el manejo de los atributos.

****

**Figura 32 Funciones del Bean**

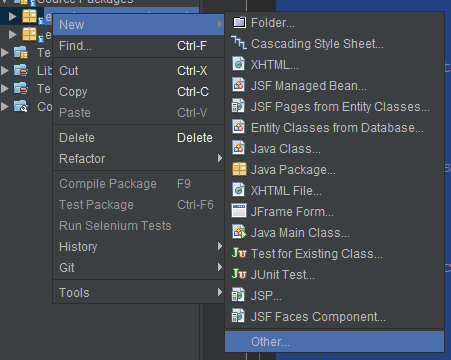
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package ec.edu.espe.monster.modelo;  import java.io.Serializable;  import java.sql.Date;  import javax.enterprise.context.SessionScoped;  import javax.inject.Named;  /\*\*  \*  \* @author aburg  \*/  @Named("usuarioBean")  @SessionScoped  public class Usuario implements Serializable{  /\*\*  \* Creates a new instance of Usuario  \*/  private String usuario;  private String empleadoId;  private String estadoCodigo;  private String password;  private String repetirPassword;  private Date fechaCreacion;  private Date fechaModificacion;  private boolean claveTemporal;  private String correo;  public Usuario(){    }    public String getRepetirPassword() {  return repetirPassword;  }  public void setRepetirPassword(String repetirPassword) {  this.repetirPassword = repetirPassword;  }    public String getEmpleadoId() {  return empleadoId;  }  public String getUsuario() {  return usuario;  }  public String getEstadoCodigo() {  return estadoCodigo;  }  public String getPassword() {  return password;  }  public Date getFechaCreacion() {  return fechaCreacion;  }  public Date getFechaModificacion() {  return fechaModificacion;  }  public boolean isClaveTemporal() {  return claveTemporal;  }  public void setEmpleadoId(String empleadoId) {  this.empleadoId = empleadoId;  }  public void setUsuario(String usuario) {  this.usuario = usuario;  }  public void setEstadoCodigo(String estadoCodigo) {  this.estadoCodigo = estadoCodigo;  }  public void setPassword(String password) {  this.password = password;  }  public void setFechaCreacion(Date fechaCreacion) {  this.fechaCreacion = fechaCreacion;  }  public void setFechaModificacion(Date fechaModificacion) {  this.fechaModificacion = fechaModificacion;  }  public void setClaveTemporal(boolean claveTemporal) {  this.claveTemporal = claveTemporal;  }  public String getCorreo() {  return correo;  }  public void setCorreo(String correo) {  this.correo = correo;  }  } |

**Tabla 3 Código del bean Usuario**

## CONTROLADOR

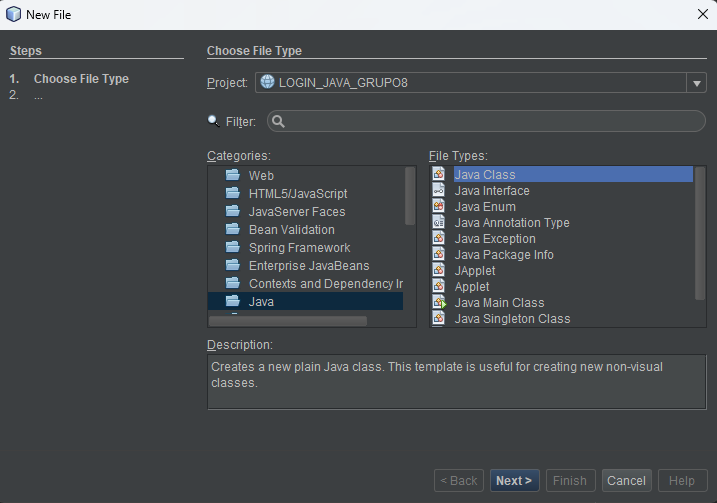
### CONEXIÓN

Para la creación de este controlador se deberá hacer click derecho sobre el paquete de los controladores y en la opción nuevo se debe seleccionar la opción Otro.



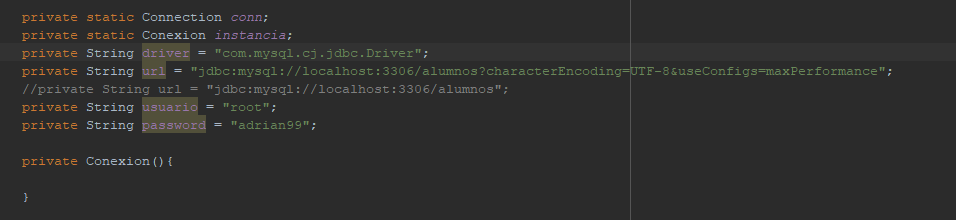
**Figura 33 Crecación de nuevo archivo**

Ahora se deberá seleccionar un archivo Java Class que se encuentra en la categoría de Java, para luego asignarle el nombre correspondiente.



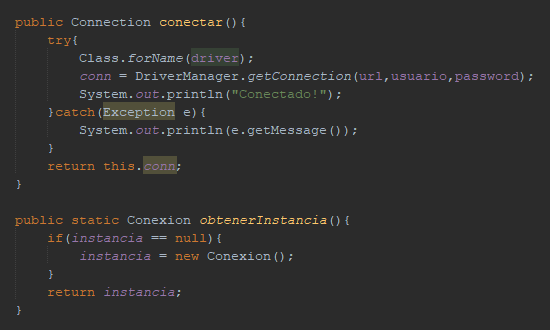
**Figura 34 Creación de archivo Java Class**

Para este controlador, se va a aplicar el patrón Singleton, por lo que se tendrán los atributos correspondientes a la conexión con el controlador de MySQL, la instancia del mismo controlador, el driver que en este caso es el de MySQL, la url de la base de datos que fue generado por la conexión creada en el apartado anterior, y finalmente las credenciales de la base de datos, este patrón indica que el constructor es privado y sin parámetros, por lo que quedaría de la siguiente manera.



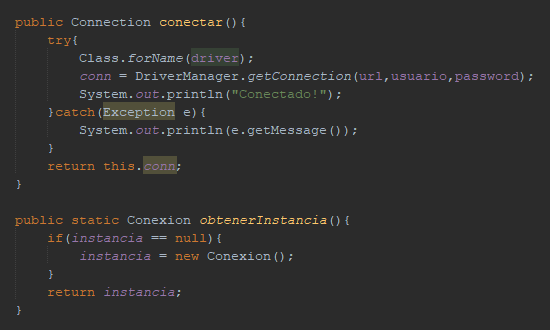
**Figura 35 Atributos y constructor del controlador coenxión**

A continuación, se muestra el método que retorna la conexión mediante el uso del driver de MySQL.



**Figura 36 Método que permite la conexión con la base de datos**

Así mismo, al implementar el patrón Singleton, se debe obtener la instancia, para ello, se crea un método que retorna la instancia de la Conexión.



**Figura 37 Método que retorna la instancia**

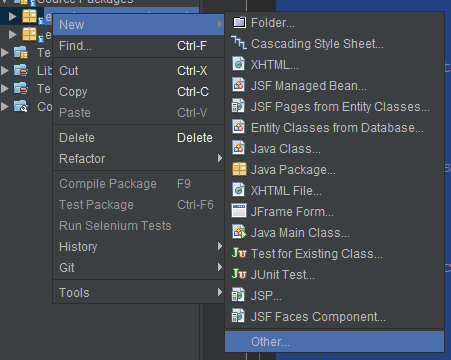
Teniendo el siguiente código:

|  |
| --- |
| package ec.edu.espe.monster.controlador;  import java.sql.\*;  import javax.swing.JOptionPane;  /\*\*  \*  \* @author aburg  \*/  public class Conexion {    private static Connection conn;  private static Conexion instancia;  private String driver = "com.mysql.cj.jdbc.Driver";  private String url = "jdbc:mysql://localhost:3306/alumnos?characterEncoding=UTF-8&useConfigs=maxPerformance";  //private String url = "jdbc:mysql://localhost:3306/alumnos";  private String usuario = "root";  private String password = "adrian99";    private Conexion(){    }    public Connection conectar(){  try{  Class.forName(driver);  conn = DriverManager.getConnection(url,usuario,password);  System.out.println("Conectado!");  }catch(Exception e){  System.out.println(e.getMessage());  }  return this.conn;  }    public static Conexion obtenerInstancia(){  if(instancia == null){  instancia = new Conexion();  }  return instancia;  }    public void cerrarConexion() throws SQLException{  try{  this.conn.close();  }catch(Exception e){  System.out.println(e.getMessage());  this.conn.close();  }finally{  this.conn.close();  }  }  } |

**Tabla 4 Código del controlador conexión**

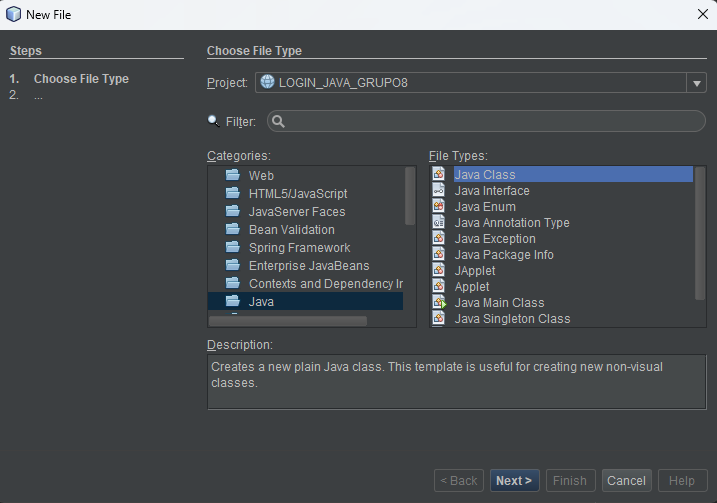
### EMPLEADOCONTROLADOR

Para la creación de este controlador se deberá hacer click derecho sobre el paquete de los controladores y en la opción nuevo se debe seleccionar la opción Otro.



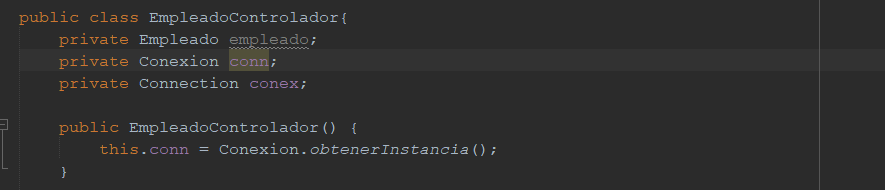
**Figura 38 Creación de nuevo archivo**

Ahora se deberá seleccionar un archivo Java Class que se encuentra en la categoría de Java, para luego asignarle el nombre correspondiente.



**Figura 39 Creación de archivo Java Class**

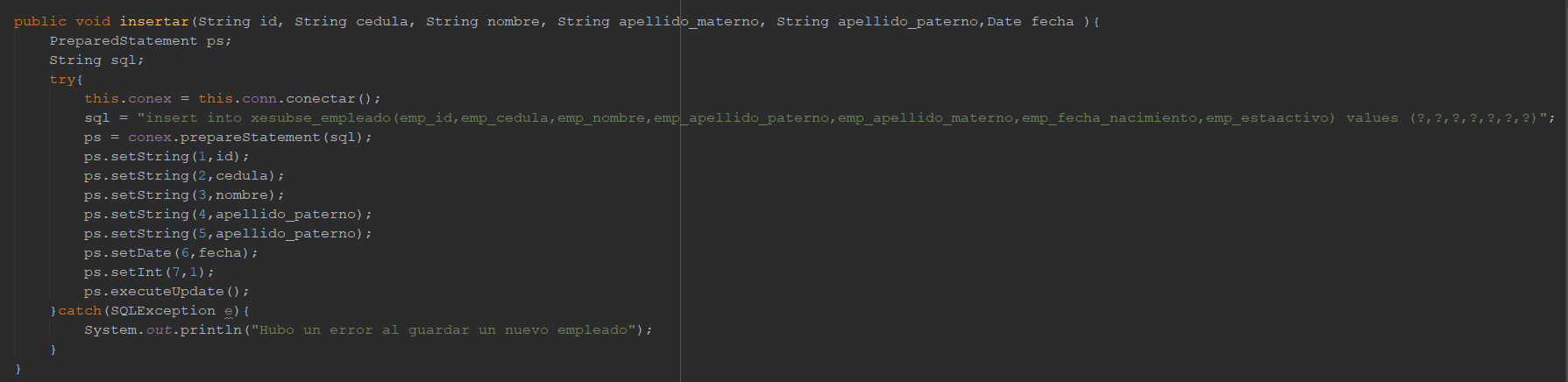
En el controlador del empleado, se tiene como atributo, un objeto de tipo empleado, otros dos objetos que permitirán la creación de la instancia de la conexión a la base de datos.



**Figura 40 Atributos y constructor del controlador**

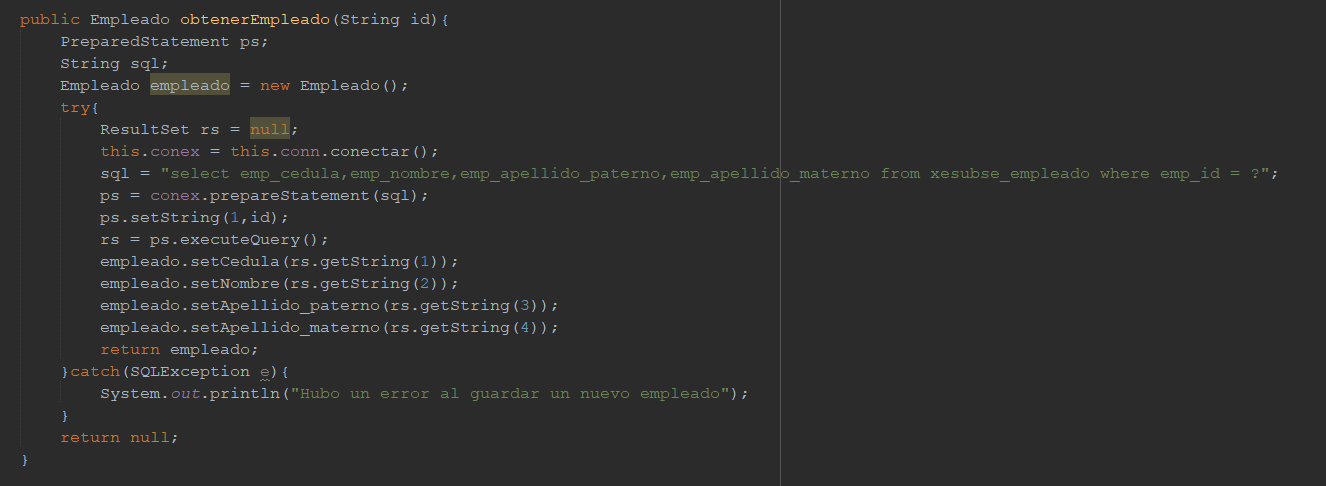
El siguiente método consiste en la adición de un nuevo empleado a la base de datos, donde se tiene la cadena de texto de la consulta a realizar y mediante el objeto de tipo PreparedStatement, permite dar a conocer o que la consulta entienda las parámetros con los que se va a trabajar.

Y así, mediante el comando ps.executeUpdate() se ejecuta la consulta.



**Figura 41 Método para insertar nuevo empleado**

El método obtenerEmpleado, permite la obtención de un empleado mediante el id del mismo, este método fue desarrollado con el objetivo de mostrar los datos del empleado que haya iniciado sesión.



**Figura 42 Método para la obtención de un empleado**

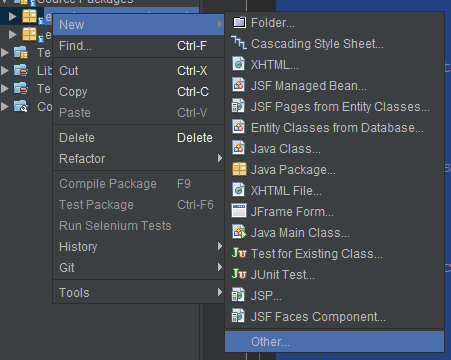
Teniendo finalmente el siguiente código

|  |
| --- |
| package ec.edu.espe.monster.controlador;  import ec.edu.espe.monster.modelo.\*;  import java.sql.\*;  /\*\*  \*  \* @author aburg  \*/  public class EmpleadoControlador{  private Empleado empleado;  private Conexion conn;  private Connection conex;    public EmpleadoControlador() {  this.conn = Conexion.obtenerInstancia();  }    public void insertar(String id, String cedula, String nombre, String apellido\_materno, String apellido\_paterno,Date fecha ){  PreparedStatement ps;  String sql;  try{  this.conex = this.conn.conectar();  sql = "insert into xesubse\_empleado(emp\_id,emp\_cedula,emp\_nombre,emp\_apellido\_paterno,emp\_apellido\_materno,emp\_fecha\_nacimiento,emp\_estaactivo) values (?,?,?,?,?,?,?)";  ps = conex.prepareStatement(sql);  ps.setString(1,id);  ps.setString(2,cedula);  ps.setString(3,nombre);  ps.setString(4,apellido\_paterno);  ps.setString(5,apellido\_paterno);  ps.setDate(6,fecha);  ps.setInt(7,1);  ps.executeUpdate();  }catch(SQLException e){  System.out.println("Hubo un error al guardar un nuevo empleado");  }  }    public Empleado obtenerEmpleado(String id){  PreparedStatement ps;  String sql;  Empleado empleado = new Empleado();  try{  ResultSet rs = null;  this.conex = this.conn.conectar();  sql = "select emp\_cedula,emp\_nombre,emp\_apellido\_paterno,emp\_apellido\_materno from xesubse\_empleado where emp\_id = ?";  ps = conex.prepareStatement(sql);  ps.setString(1,id);  rs = ps.executeQuery();  empleado.setCedula(rs.getString(1));  empleado.setNombre(rs.getString(2));  empleado.setApellido\_paterno(rs.getString(3));  empleado.setApellido\_materno(rs.getString(4));  return empleado;  }catch(SQLException e){  System.out.println("Hubo un error al guardar un nuevo empleado");  }  return null;  }  } |

**Tabla 5 Código del controlador del empleado**

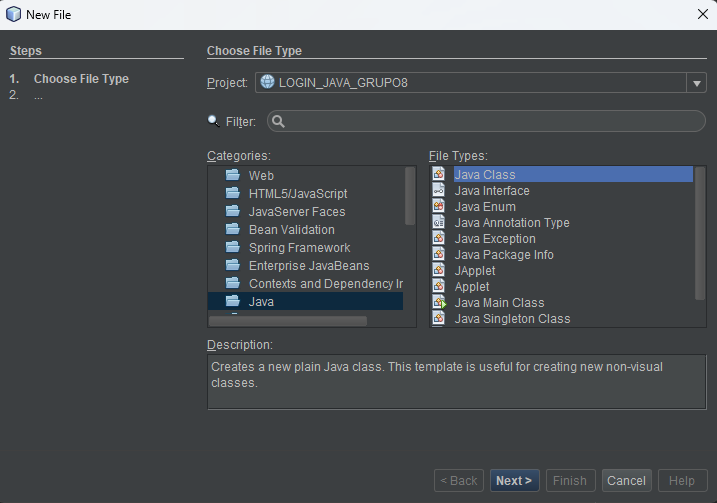
### USUARIOCONTROLADOR

Para la creación de este controlador se deberá hacer click derecho sobre el paquete de los controladores y en la opción nuevo se debe seleccionar la opción Otro.



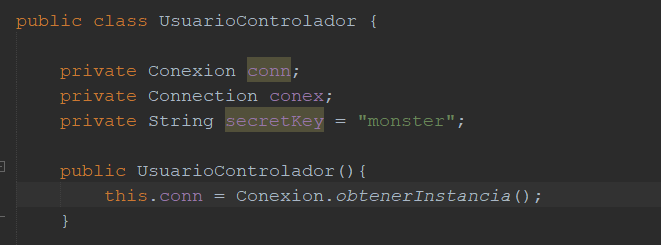
**Figura 43 Creación de nuevo archivo**

Ahora se deberá seleccionar un archivo Java Class que se encuentra en la categoría de Java, para luego asignarle el nombre correspondiente.



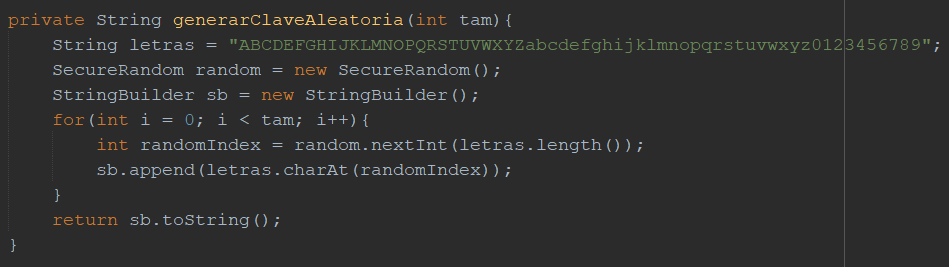
**Figura 44 Creación de archivo Java Class**

El controlador del usuario es el más complejo, pues este controlador se encarga de la generación de una clave temporal, de la encriptación de contraseñas y de enviar la clave temporal al correo, por lo que tiene como atributos la conexión a la base de datos y la secretKey, que permitirá la encriptación mediante el algoritmo MD5.



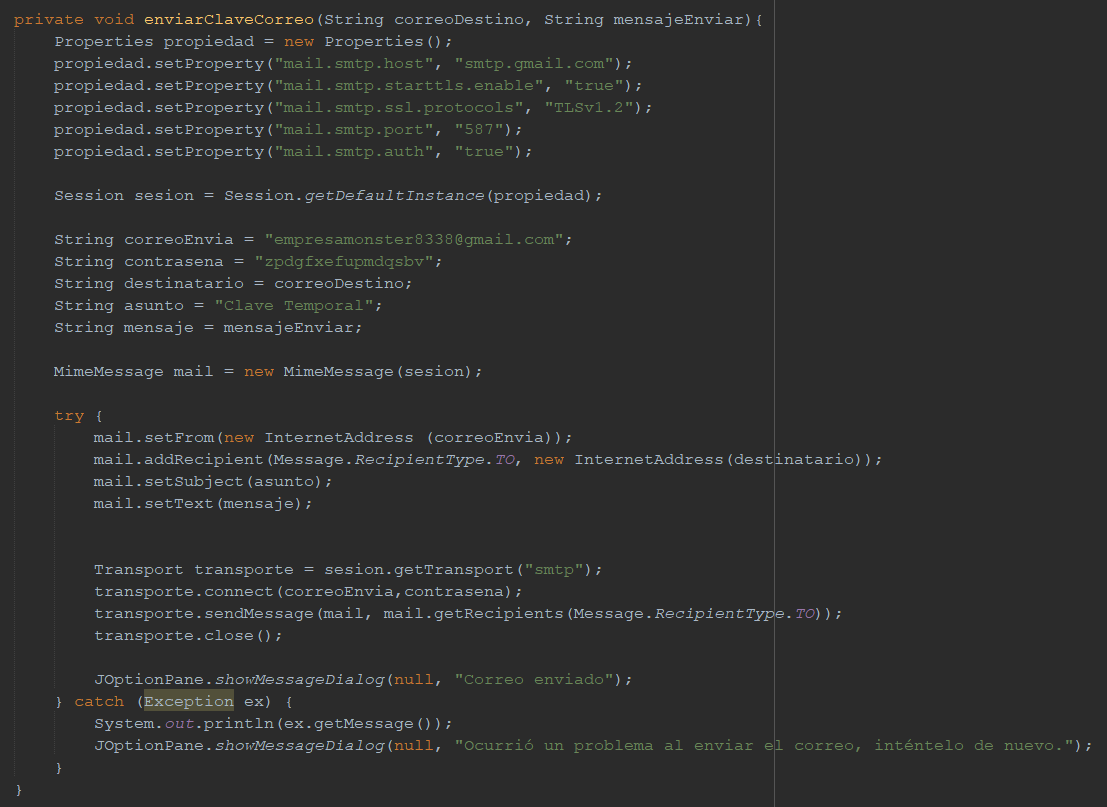
**Figura 45 Atributos y constructor del controlador**

El siguiente método, permite la generación de la clave temporal que será enviada al correo del usuario que se registre, por que lo genera la clave en base al tamaño deseado y al conjunto de posibles datos, entre estos son letras y números.



**Figura 46 Método que genera clave temporal**

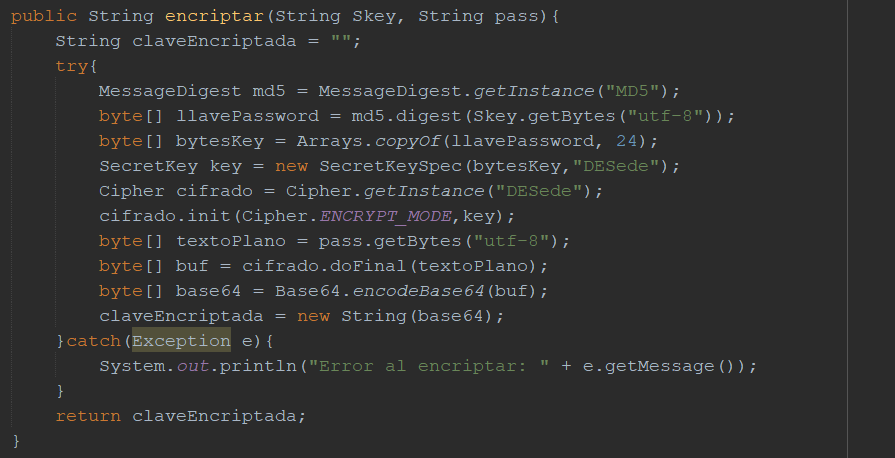
El método enviarClaveCorreo, se encarga de realizar la conexión con la cuenta de Gmail desde donde se hará el envío, también se encarga de establecer y recibir el correo de destino y el mensaje que se enviará.



**Figura 47 Método para enviar la clave al correo**

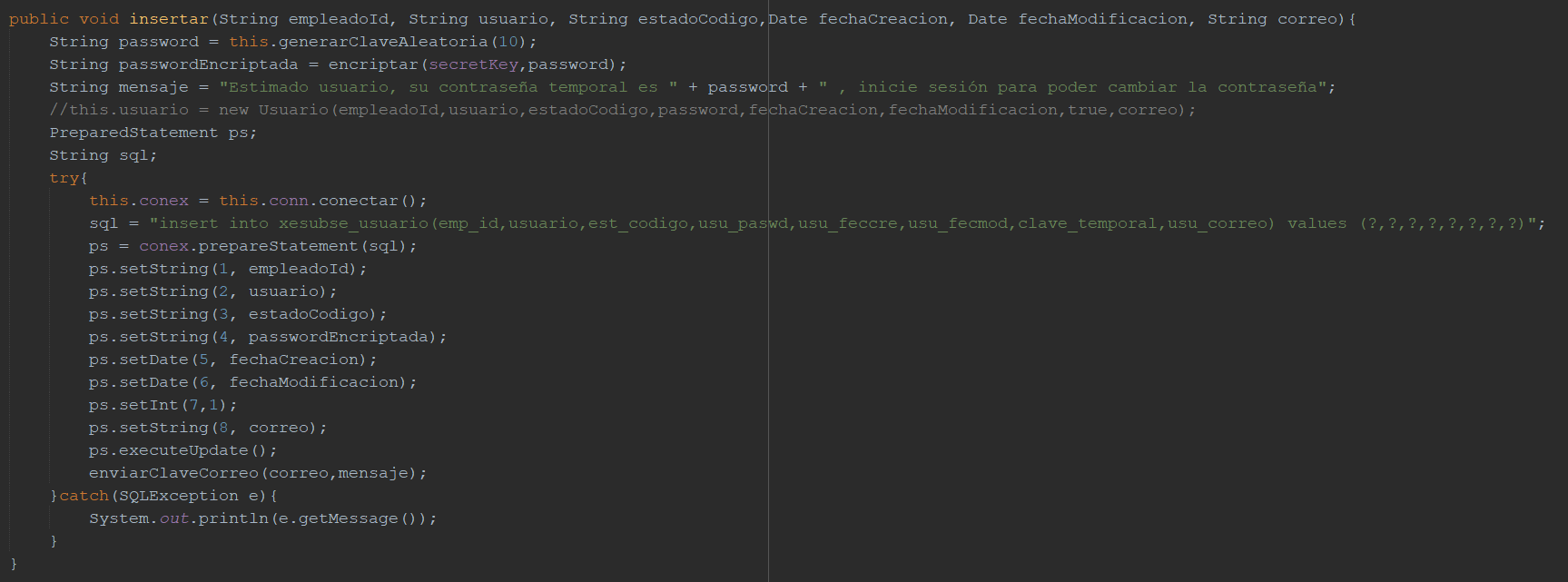
Es importante configurar la cuenta de Gmail desde donde se enviará, pues Google ya no permite el acceso a aplicaciones menos seguras, pero permite la generación de claves para aplicaciones.

Este método, se encarga de encriptar la contraseña que recibe como parámetro y la encriptación la realiza en base a la secretKey de la cual se habló previamente. Este método realiza una serie de conversiones, para que al final retorne una contraseña segura y encriptada.



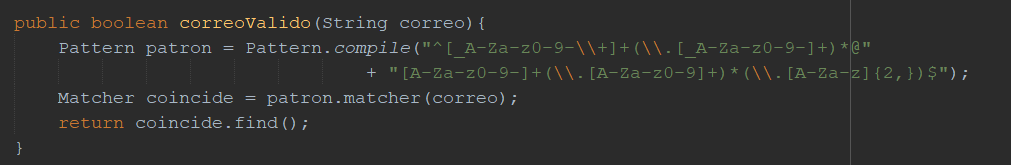
**Figura 48 Método que encripta la contraseña**

El método de insertar, consiste en el registro de un nuevo usuario, por lo que, de la misma manera como se realizó en el empleadocontrolador, se prepara la sentencia SQL en una cadena de texto y se setea los valores que serán ingresados mediante el objeto de tipo PreparedStatement, en esta sentencia se enviará la contraseña ya encriptada.



**Figura 49 Método para ingresar un nuevo usuario en la base de datos**

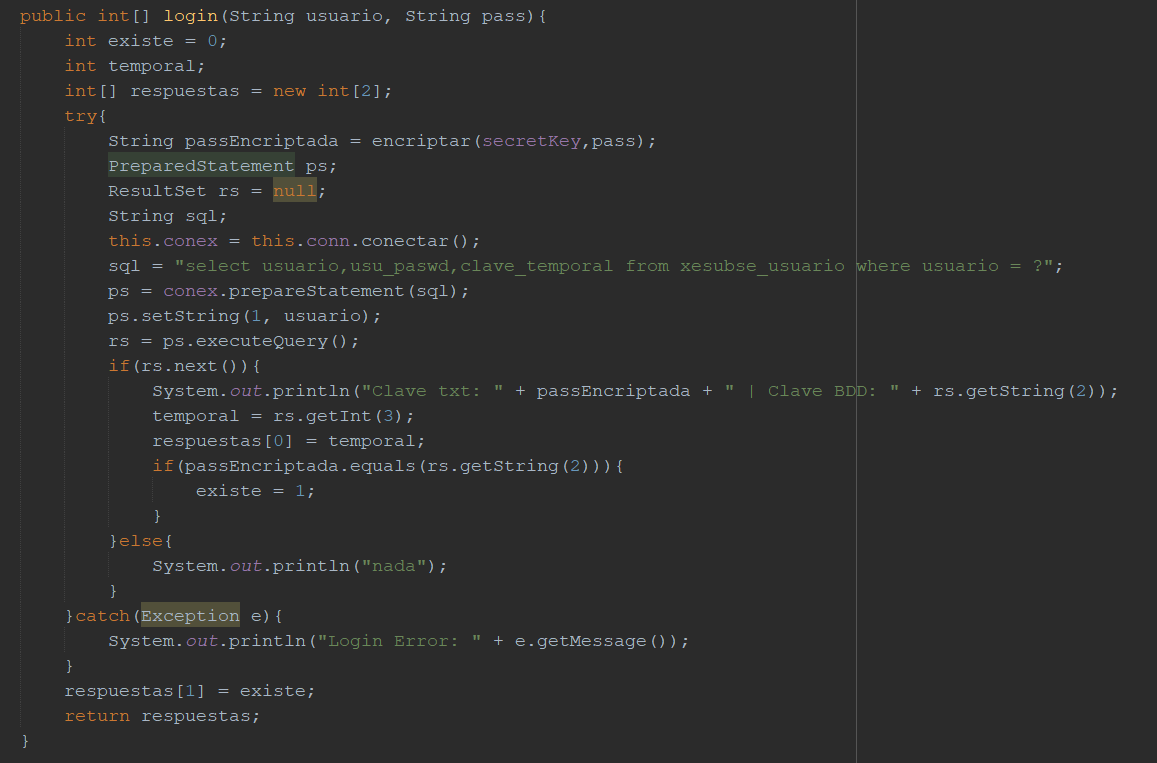
El siguiente método se encarga de validar el correo recibido como parámetro, este método será usado en el controlador principal.



**Figura 50 Méotodo para validar el correo**

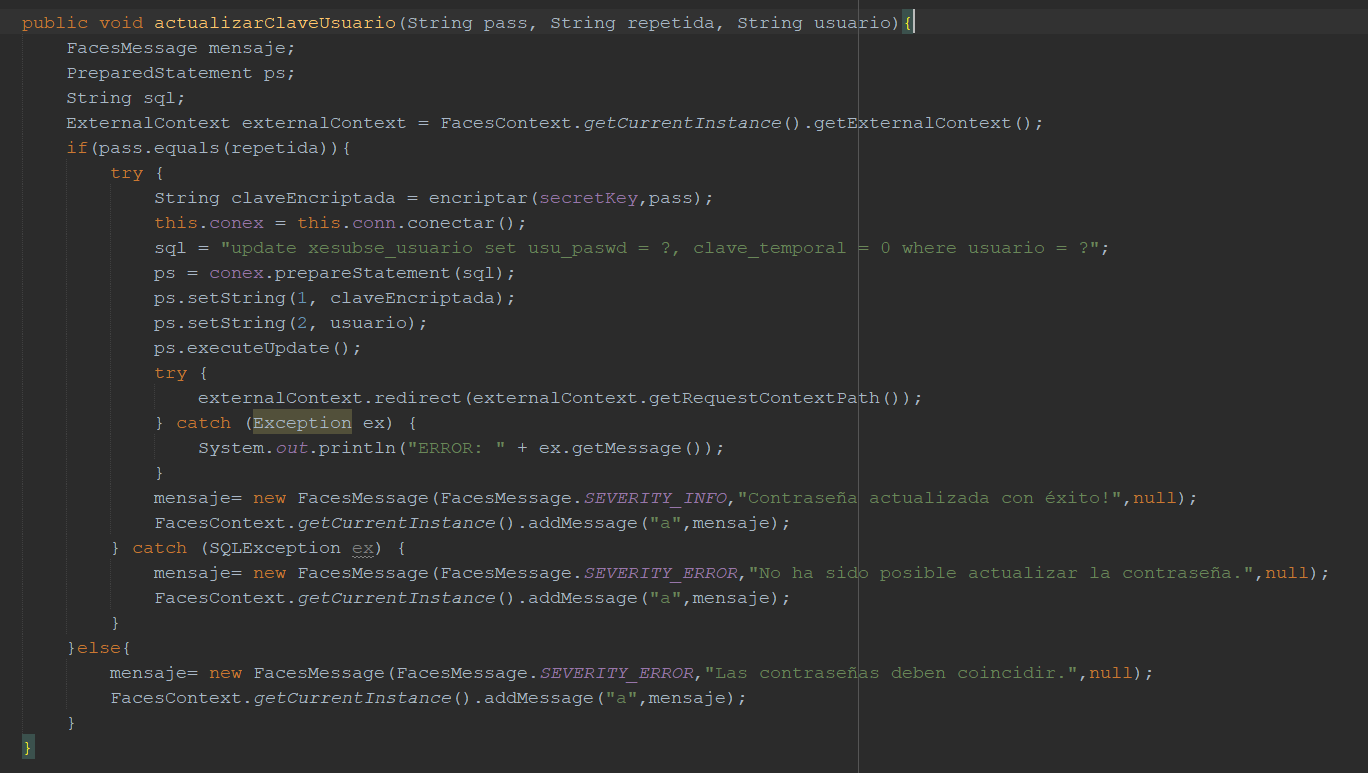
El método dedicado al login, se encarga de retornar un arreglo de enteros, donde se estará guardando 2 valores, estos pueden ser 0 o 1, corresponden al estado del login, es decir, si el usuario existe, y de ser así, averiguar si cuenta con una clave temporal para poder permitir el cambio de contraseña.

Los datos obtenidos serán almacenados en un objeto de tipo ResultSet, que permitirá recorrerlo y comparar las credenciales y determinar si son correctas o no.



**Figura 51 Método encargado de efectura el login**

Para actualizar la contraseña de un usuario que está iniciando sesión por primera vez, se realiza una actualización de la contraseña y del estado de clave temporal para que en la base de datos conste con una clave oficial, en este método se valida también la repetición de contraseña.



**Figura 52 Método que permite la actualización de la contraseña**

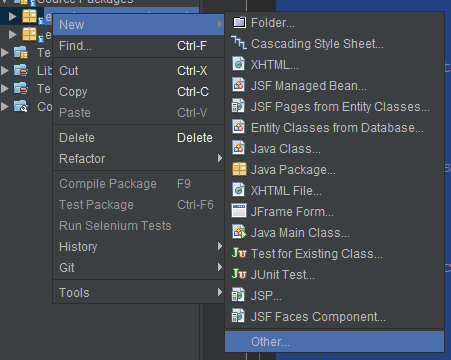
Una vez explicado el código del controlador, se tiene el siguiente código.

|  |
| --- |
| package ec.edu.espe.monster.controlador;  /\*\*  \*  \* @author aburg  \*/  import java.security.MessageDigest;  import java.security.SecureRandom;  import java.sql.\*;  import java.util.Arrays;  import java.util.Properties;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import javax.crypto.Cipher;  import javax.crypto.SecretKey;  import javax.crypto.spec.SecretKeySpec;  import javax.faces.application.FacesMessage;  import javax.faces.context.ExternalContext;  import javax.faces.context.FacesContext;  import javax.mail.Message;  import javax.mail.Session;  import javax.mail.Transport;  import javax.mail.internet.InternetAddress;  import javax.mail.internet.MimeMessage;  import javax.swing.JOptionPane;  import org.apache.commons.codec.binary.Base64;  public class UsuarioControlador {    private Conexion conn;  private Connection conex;  private String secretKey = "monster";    public UsuarioControlador(){  this.conn = Conexion.obtenerInstancia();  }    private String generarClaveAleatoria(int tam){  String letras = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  SecureRandom random = new SecureRandom();  StringBuilder sb = new StringBuilder();  for(int i = 0; i < tam; i++){  int randomIndex = random.nextInt(letras.length());  sb.append(letras.charAt(randomIndex));  }  return sb.toString();  }    private void enviarClaveCorreo(String correoDestino, String mensajeEnviar){  Properties propiedad = new Properties();  propiedad.setProperty("mail.smtp.host", "smtp.gmail.com");  propiedad.setProperty("mail.smtp.starttls.enable", "true");  propiedad.setProperty("mail.smtp.ssl.protocols", "TLSv1.2");  propiedad.setProperty("mail.smtp.port", "587");  propiedad.setProperty("mail.smtp.auth", "true");    Session sesion = Session.getDefaultInstance(propiedad);    String correoEnvia = "empresamonster8338@gmail.com";  String contrasena = "zpdgfxefupmdqsbv";  String destinatario = correoDestino;  String asunto = "Clave Temporal";  String mensaje = mensajeEnviar;    MimeMessage mail = new MimeMessage(sesion);    try {  mail.setFrom(new InternetAddress (correoEnvia));  mail.addRecipient(Message.RecipientType.TO, new InternetAddress(destinatario));  mail.setSubject(asunto);  mail.setText(mensaje);      Transport transporte = sesion.getTransport("smtp");  transporte.connect(correoEnvia,contrasena);  transporte.sendMessage(mail, mail.getRecipients(Message.RecipientType.TO));  transporte.close();    JOptionPane.showMessageDialog(null, "Correo enviado");  } catch (Exception ex) {  System.out.println(ex.getMessage());  JOptionPane.showMessageDialog(null, "Ocurrió un problema al enviar el correo, inténtelo de nuevo.");  }  }    public String encriptar(String Skey, String pass){  String claveEncriptada = "";  try{  MessageDigest md5 = MessageDigest.getInstance("MD5");  byte[] llavePassword = md5.digest(Skey.getBytes("utf-8"));  byte[] bytesKey = Arrays.copyOf(llavePassword, 24);  SecretKey key = new SecretKeySpec(bytesKey,"DESede");  Cipher cifrado = Cipher.getInstance("DESede");  cifrado.init(Cipher.ENCRYPT\_MODE,key);  byte[] textoPlano = pass.getBytes("utf-8");  byte[] buf = cifrado.doFinal(textoPlano);  byte[] base64 = Base64.encodeBase64(buf);  claveEncriptada = new String(base64);  }catch(Exception e){  System.out.println("Error al encriptar: " + e.getMessage());  }  return claveEncriptada;  }    public void insertar(String empleadoId, String usuario, String estadoCodigo,Date fechaCreacion, Date fechaModificacion, String correo){  String password = this.generarClaveAleatoria(10);  String passwordEncriptada = encriptar(secretKey,password);  String mensaje = "Estimado usuario, su contraseña temporal es " + password + " , inicie sesión para poder cambiar la contraseña";  //this.usuario = new Usuario(empleadoId,usuario,estadoCodigo,password,fechaCreacion,fechaModificacion,true,correo);  PreparedStatement ps;  String sql;  try{  this.conex = this.conn.conectar();  sql = "insert into xesubse\_usuario(emp\_id,usuario,est\_codigo,usu\_paswd,usu\_feccre,usu\_fecmod,clave\_temporal,usu\_correo) values (?,?,?,?,?,?,?,?)";  ps = conex.prepareStatement(sql);  ps.setString(1, empleadoId);  ps.setString(2, usuario);  ps.setString(3, estadoCodigo);  ps.setString(4, passwordEncriptada);  ps.setDate(5, fechaCreacion);  ps.setDate(6, fechaModificacion);  ps.setInt(7,1);  ps.setString(8, correo);  ps.executeUpdate();  enviarClaveCorreo(correo,mensaje);  }catch(SQLException e){  System.out.println(e.getMessage());  }  }    public boolean correoValido(String correo){  Pattern patron = Pattern.compile("^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@"  + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$");  Matcher coincide = patron.matcher(correo);  return coincide.find();  }    public int[] login(String usuario, String pass){  int existe = 0;  int temporal;  int[] respuestas = new int[2];  try{  String passEncriptada = encriptar(secretKey,pass);  PreparedStatement ps;  ResultSet rs = null;  String sql;  this.conex = this.conn.conectar();  sql = "select usuario,usu\_paswd,clave\_temporal from xesubse\_usuario where usuario = ?";  ps = conex.prepareStatement(sql);  ps.setString(1, usuario);  rs = ps.executeQuery();  if(rs.next()){  System.out.println("Clave txt: " + passEncriptada + " | Clave BDD: " + rs.getString(2));  temporal = rs.getInt(3);  respuestas[0] = temporal;  if(passEncriptada.equals(rs.getString(2))){  existe = 1;  }  }else{  System.out.println("nada");  }  }catch(Exception e){  System.out.println("Login Error: " + e.getMessage());  }  respuestas[1] = existe;  return respuestas;  }    public void actualizarClaveUsuario(String pass, String repetida, String usuario){  FacesMessage mensaje;  PreparedStatement ps;  String sql;  ExternalContext externalContext = FacesContext.getCurrentInstance().getExternalContext();  if(pass.equals(repetida)){  try {  String claveEncriptada = encriptar(secretKey,pass);  this.conex = this.conn.conectar();  sql = "update xesubse\_usuario set usu\_paswd = ?, clave\_temporal = 0 where usuario = ?";  ps = conex.prepareStatement(sql);  ps.setString(1, claveEncriptada);  ps.setString(2, usuario);  ps.executeUpdate();  try {  externalContext.redirect(externalContext.getRequestContextPath());  } catch (Exception ex) {  System.out.println("ERROR: " + ex.getMessage());  }  mensaje= new FacesMessage(FacesMessage.SEVERITY\_INFO,"Contraseña actualizada con éxito!",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  } catch (SQLException ex) {  mensaje= new FacesMessage(FacesMessage.SEVERITY\_ERROR,"No ha sido posible actualizar la contraseña.",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }  }else{  mensaje= new FacesMessage(FacesMessage.SEVERITY\_ERROR,"Las contraseñas deben coincidir.",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }  }  } |

**Tabla 6 Código del controlador usuario**

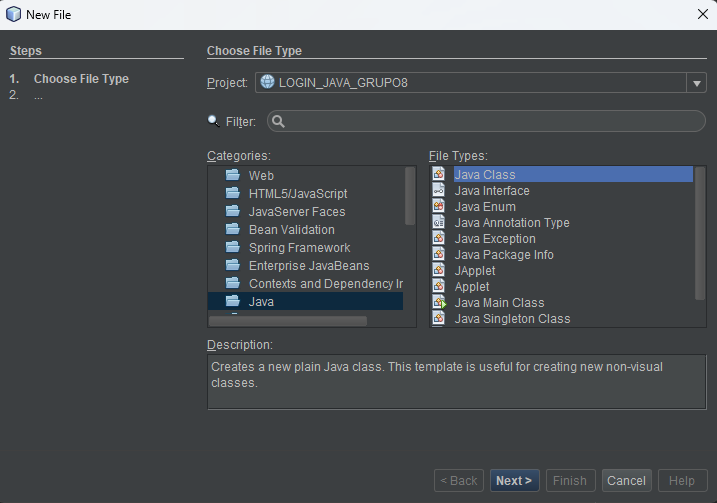
### CONTROLADOR PRINCIPAL

Para la creación de este controlador se deberá hacer click derecho sobre el paquete de los controladores y en la opción nuevo se debe seleccionar la opción Otro.



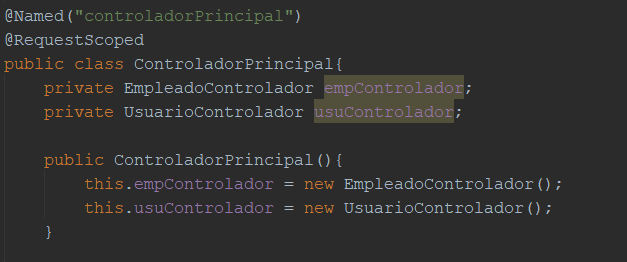
**Figura 53 Creación de nuevo archivo**

Ahora se deberá seleccionar un archivo Java Class que se encuentra en la categoría de Java, para luego asignarle el nombre correspondiente.



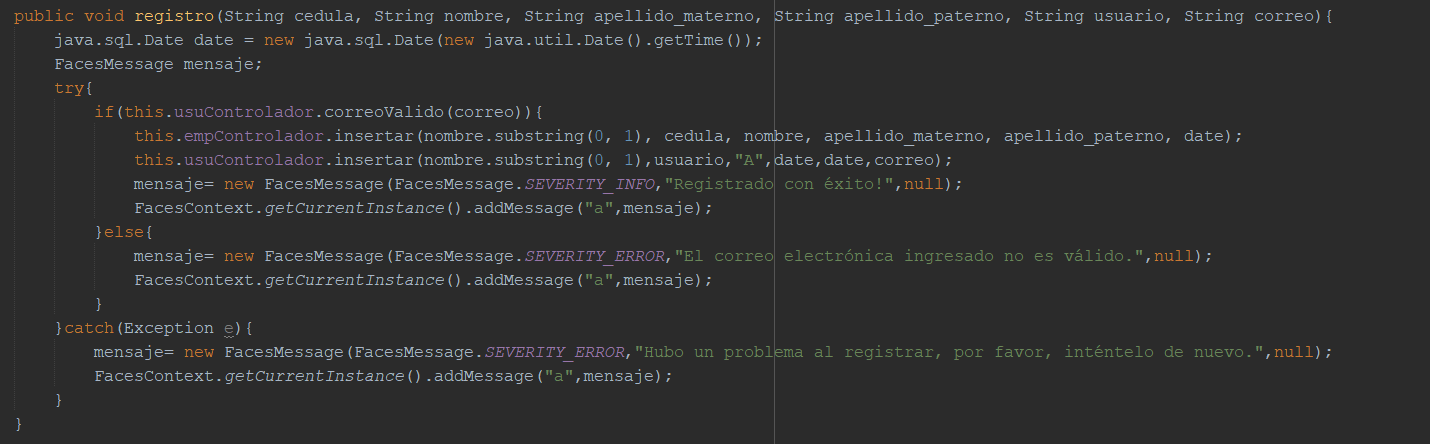
**Figura 54 Creación de nuevo archivo Java Class**

Los atributos del controlador, serán los controladores del empleado y del usuario, ya que al hacer un nuevo registro o iniciar sesión, se realiza en las dos tablas, ya que en la tabla de usuarios se tiene la llave primaria de la tabla empleado como foránea.



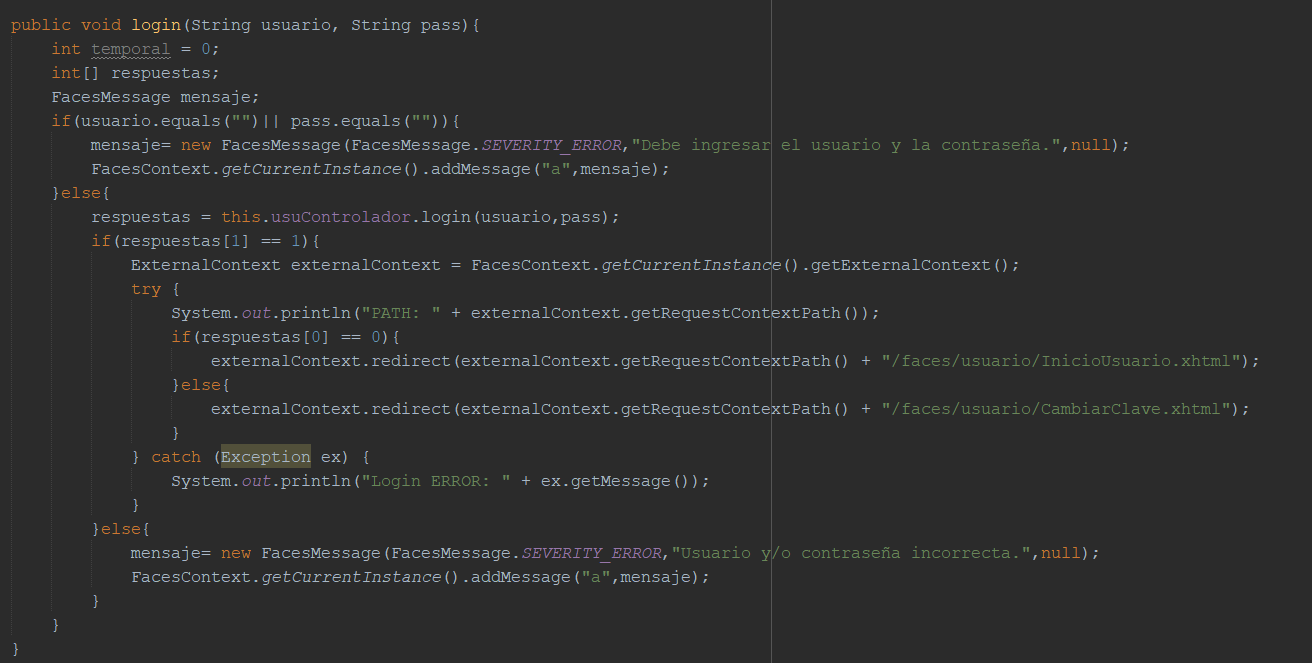
**Figura 55 Atributos y constructor del controlador**

El método registro permite realizar la adicción de un empleado y usuario en la base de datos pero antes de eso, valida el correo electrónico ingresado, y de manera simultánea realiza la adición mediante ambos controladores previamente mencionados.



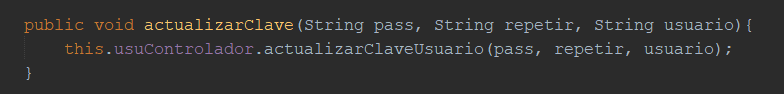
**Figura 56 Método de registro**

El método login, se encarga de verificar si las credenciales son correctas, pero para ello primero realiza algunas validaciones, como por ejemplo, que los campos del usuario y de la contraseña no deben estar vacíos, también antes de realizar el login, determina el estado del valor de la clave temporal, es decir, si el usuario cuenta con una clave temporal, tendrá que actualizar la contraseña para poder seguir, si no cuenta con clave temporal entonces le redirigirá a la pantalla principal.



**Figura 57 Méotodo de login**

Este método se encarga de actualizar la contraseña en la tabla de usuario.



**Figura 58 Método para actualizar la clave**

Teniendo el siguiente código.

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package ec.edu.espe.monster.controlador;  import javax.enterprise.context.RequestScoped;  import javax.faces.application.FacesMessage;  import javax.faces.context.ExternalContext;  import javax.faces.context.FacesContext;  import javax.inject.Named;  /\*\*  \*  \* @author aburg  \*/  @Named("controladorPrincipal")  @RequestScoped  public class ControladorPrincipal{  private EmpleadoControlador empControlador;  private UsuarioControlador usuControlador;    public ControladorPrincipal(){  this.empControlador = new EmpleadoControlador();  this.usuControlador = new UsuarioControlador();  }    public void registro(String cedula, String nombre, String apellido\_materno, String apellido\_paterno, String usuario, String correo){  java.sql.Date date = new java.sql.Date(new java.util.Date().getTime());  FacesMessage mensaje;  try{  if(this.usuControlador.correoValido(correo)){  this.empControlador.insertar(nombre.substring(0, 1), cedula, nombre, apellido\_materno, apellido\_paterno, date);  this.usuControlador.insertar(nombre.substring(0, 1),usuario,"A",date,date,correo);  mensaje= new FacesMessage(FacesMessage.SEVERITY\_INFO,"Registrado con éxito!",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }else{  mensaje= new FacesMessage(FacesMessage.SEVERITY\_ERROR,"El correo electrónica ingresado no es válido.",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }  }catch(Exception e){  mensaje= new FacesMessage(FacesMessage.SEVERITY\_ERROR,"Hubo un problema al registrar, por favor, inténtelo de nuevo.",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }  }    public void login(String usuario, String pass){  int temporal = 0;  int[] respuestas;  FacesMessage mensaje;  if(usuario.equals("")|| pass.equals("")){  mensaje= new FacesMessage(FacesMessage.SEVERITY\_ERROR,"Debe ingresar el usuario y la contraseña.",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }else{  respuestas = this.usuControlador.login(usuario,pass);  if(respuestas[1] == 1){  ExternalContext externalContext = FacesContext.getCurrentInstance().getExternalContext();  try {  System.out.println("PATH: " + externalContext.getRequestContextPath());  if(respuestas[0] == 0){  externalContext.redirect(externalContext.getRequestContextPath() + "/faces/usuario/InicioUsuario.xhtml");  }else{  externalContext.redirect(externalContext.getRequestContextPath() + "/faces/usuario/CambiarClave.xhtml");  }  } catch (Exception ex) {  System.out.println("Login ERROR: " + ex.getMessage());  }  }else{  mensaje= new FacesMessage(FacesMessage.SEVERITY\_ERROR,"Usuario y/o contraseña incorrecta.",null);  FacesContext.getCurrentInstance().addMessage("a",mensaje);  }  }  }    public void actualizarClave(String pass, String repetir, String usuario){  this.usuControlador.actualizarClaveUsuario(pass, repetir, usuario);  }  } |

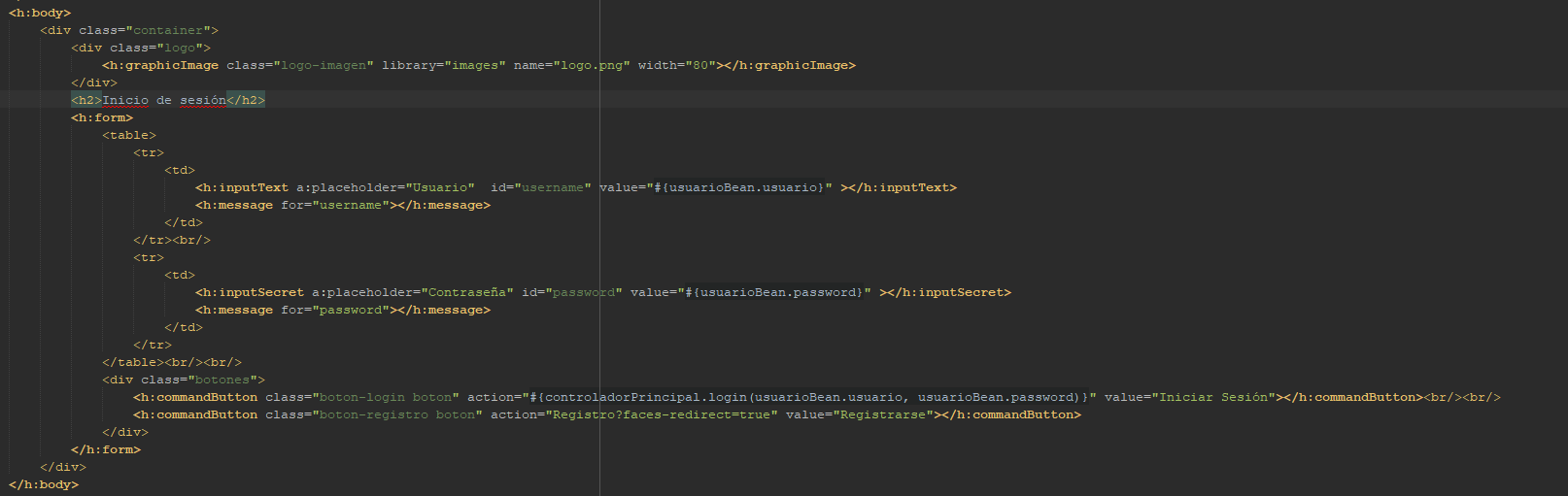
**Tabla 7 Código del controlador principal**

## VISTA

### PÁGINA DE LOGIN

Para esta pantalla del login, se implementa en los valores de las entradas de texto el valor del bean que en este caso sería para el usuario.

Y en los botones se asigna las funciones contenidas en el controlador, en este caso, el controlador del usuario, mientras que, para el botón de registro, se redirige a la página de registro.



**Figura 59 Código del index.xhtml**

Teniendo el siguiente código

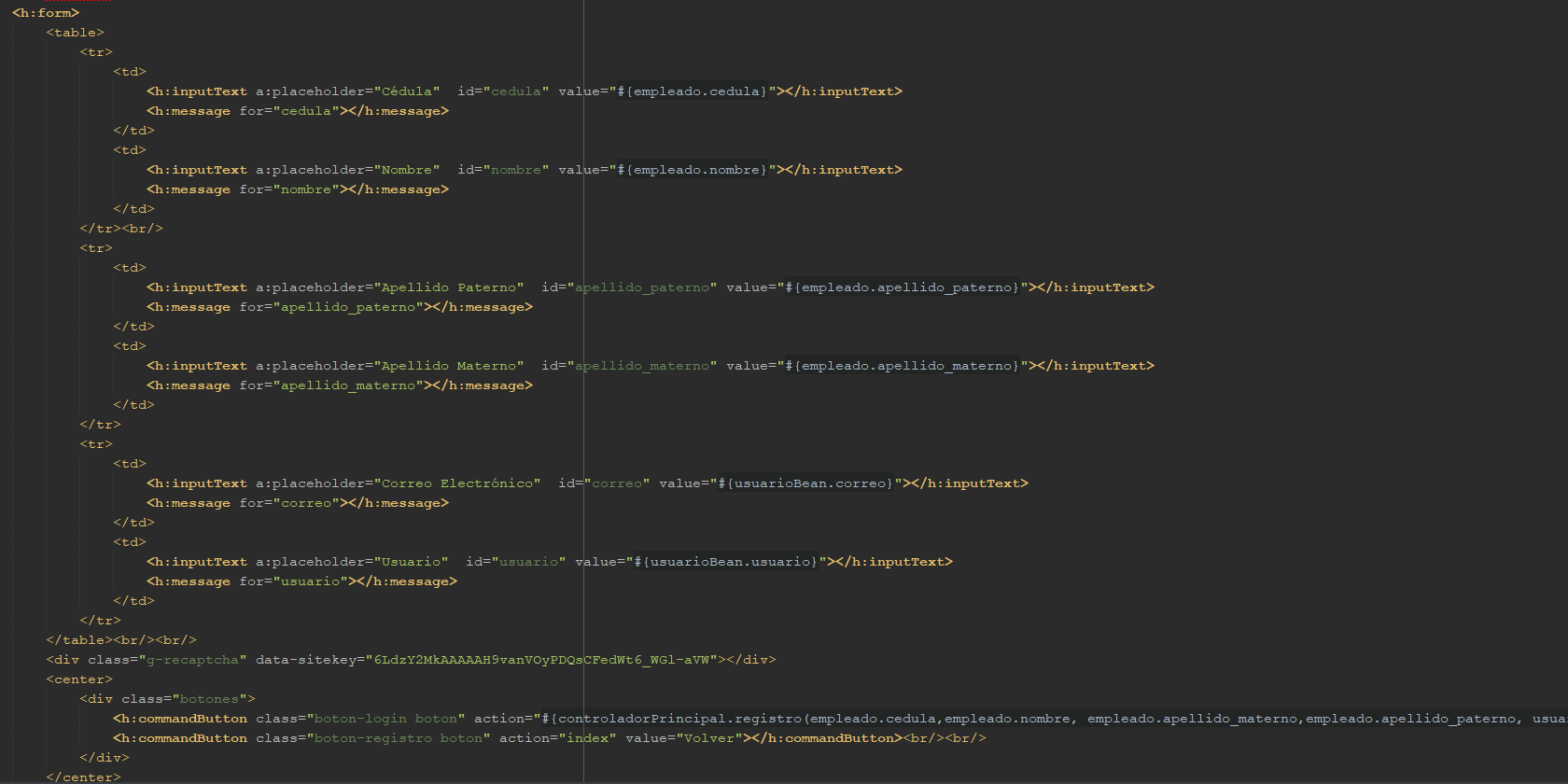
|  |
| --- |
| <?xml version='1.0' encoding='UTF-8' ?>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:h="http://xmlns.jcp.org/jsf/html"  xmlns:a="http://xmlns.jcp.org/jsf/passthrough">    <h:head>  <title> Página de inicio de sesión</title>  <h:outputStylesheet library="css" name="style.css"/>  </h:head>  <h:body>  <div class="container">  <div class="logo">  <h:graphicImage class="logo-imagen" library="images" name="logo.png" width="80"></h:graphicImage>  </div>  <h2>Inicio de sesión</h2>  <h:form>  <table>  <tr>  <td>  <h:inputText a:placeholder="Usuario" id="username" value="#{usuarioBean.usuario}" ></h:inputText>  <h:message for="username"></h:message>  </td>  </tr><br/>  <tr>  <td>  <h:inputSecret a:placeholder="Contraseña" id="password" value="#{usuarioBean.password}" ></h:inputSecret>  <h:message for="password"></h:message>  </td>  </tr>  </table><br/><br/>  <div class="botones">  <h:commandButton class="boton-login boton" action="#{controladorPrincipal.login(usuarioBean.usuario, usuarioBean.password)}" value="Iniciar Sesión"></h:commandButton><br/><br/>  <h:commandButton class="boton-registro boton" action="Registro?faces-redirect=true" value="Registrarse"></h:commandButton>  </div>  </h:form>  </div>  </h:body>  </html> |

**Tabla 8 Código index.xhtml**

### PÁGINA DE REGISTRO

De la misma manera como en el login, se tiene el formulario con los campos para registrar tanto un empleado y un usuario, a excepción de la contraseña ya que el controlador se encargar de generarlar.

También se establece los valores para cada input, con los valores correspondientes tanto para el empleado como para el usuario, así mismo también se tienen establecidos los métodos de los controladores en el botón de registro.



**Figura 60 Código de registro.xhtml**

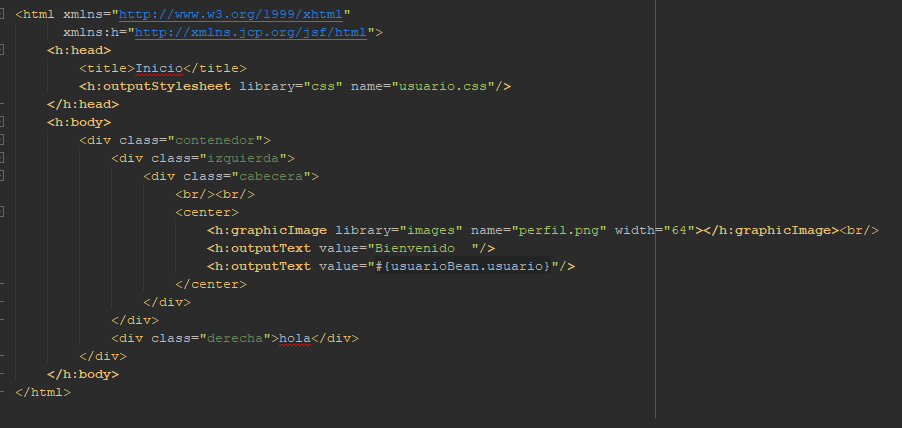
Teniendo el siguiente código

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8' ?>  <!--  To change this license header, choose License Headers in Project Properties.  To change this template file, choose Tools | Templates  and open the template in the editor.  -->  <!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:h="http://xmlns.jcp.org/jsf/html"  xmlns:a="http://xmlns.jcp.org/jsf/passthrough">  <h:head>  <title>Registro</title>  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>  <h:outputStylesheet library="css" name="style.css"/>  <script src="https://www.google.com/recaptcha/api.js" async="async" defer="defer"></script>  </h:head>  <body>  <div class="container">  <div class="logo">  <h:graphicImage class="logo-imagen" library="images" name="logo.png" width="80"></h:graphicImage>  </div>  <h2>Registro</h2>  <h:form>  <table>  <tr>  <td>  <h:inputText a:placeholder="Cédula" id="cedula" value="#{empleado.cedula}"></h:inputText>  <h:message for="cedula"></h:message>  </td>  <td>  <h:inputText a:placeholder="Nombre" id="nombre" value="#{empleado.nombre}"></h:inputText>  <h:message for="nombre"></h:message>  </td>  </tr><br/>  <tr>  <td>  <h:inputText a:placeholder="Apellido Paterno" id="apellido\_paterno" value="#{empleado.apellido\_paterno}"></h:inputText>  <h:message for="apellido\_paterno"></h:message>  </td>  <td>  <h:inputText a:placeholder="Apellido Materno" id="apellido\_materno" value="#{empleado.apellido\_materno}"></h:inputText>  <h:message for="apellido\_materno"></h:message>  </td>  </tr>  <tr>  <td>  <h:inputText a:placeholder="Correo Electrónico" id="correo" value="#{usuarioBean.correo}"></h:inputText>  <h:message for="correo"></h:message>  </td>  <td>  <h:inputText a:placeholder="Usuario" id="usuario" value="#{usuarioBean.usuario}"></h:inputText>  <h:message for="usuario"></h:message>  </td>  </tr>  </table><br/><br/>  <div class="g-recaptcha" data-sitekey="6LdzY2MkAAAAAH9vanVOyPDQsCFedWt6\_WGl-aVW"></div>  <center>  <div class="botones">  <h:commandButton class="boton-login boton" action="#{controladorPrincipal.registro(empleado.cedula,empleado.nombre, empleado.apellido\_materno,empleado.apellido\_paterno, usuarioBean.usuario,usuarioBean.correo)}" value="Registrarse"></h:commandButton><br/><br/>  <h:commandButton class="boton-registro boton" action="index" value="Volver"></h:commandButton><br/><br/>  </div>  </center>  </h:form>  </div>  </body>  </html> |

**Tabla 9 Código de Registro.xhtml**

### PÁGINA DE INICIO

El siguiente código muestra la pantalla de inicio donde se indica el nombre de usuario del usuario que ha iniciado sesión.



**Figura 61 Código de InicioUsuario.xhtml**

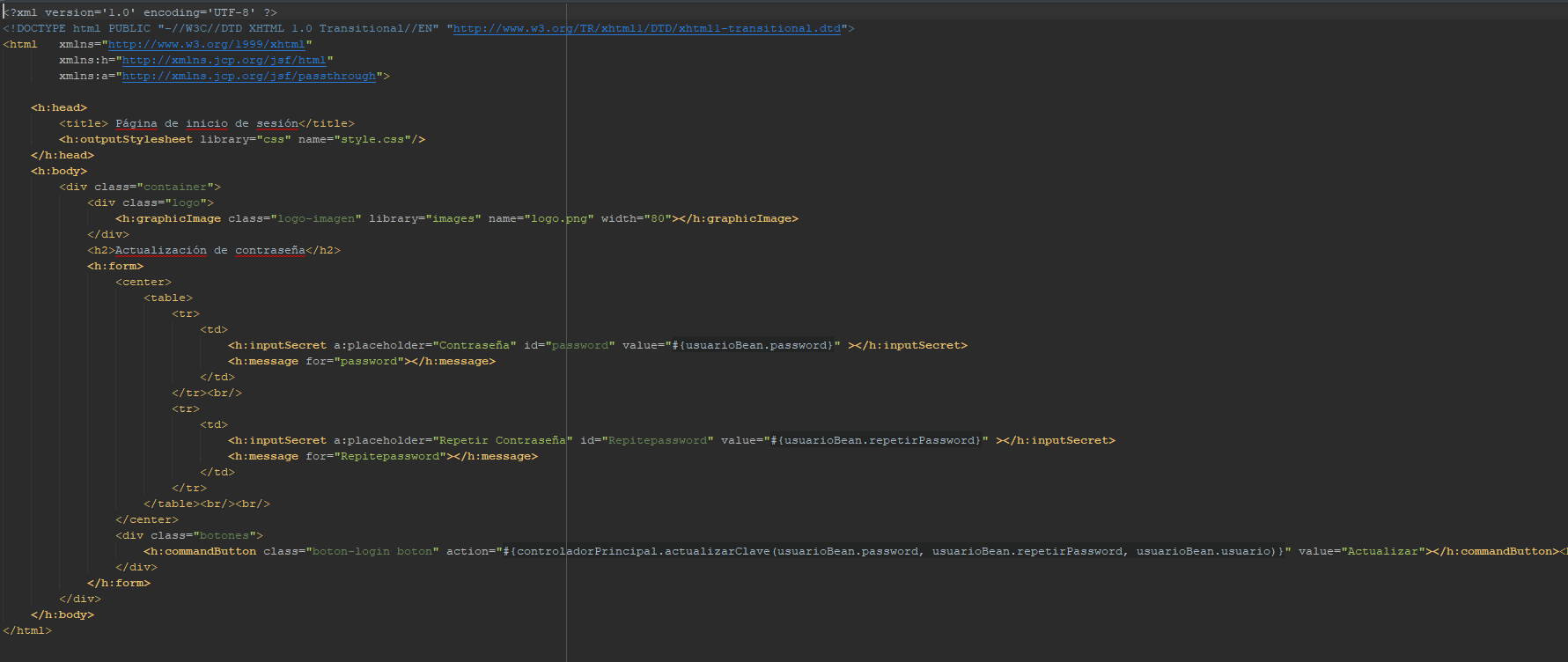
Teniendo el siguiente código.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!--  To change this license header, choose License Headers in Project Properties.  To change this template file, choose Tools | Templates  and open the template in the editor.  -->  <!DOCTYPE html>  <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:h="http://xmlns.jcp.org/jsf/html">  <h:head>  <title>Inicio</title>  <h:outputStylesheet library="css" name="usuario.css"/>  </h:head>  <h:body>  <div class="contenedor">  <div class="izquierda">  <div class="cabecera">  <br/><br/>  <center>  <h:graphicImage library="images" name="perfil.png" width="64"></h:graphicImage><br/>  <h:outputText value="Bienvenido "/>  <h:outputText value="#{usuarioBean.usuario}"/>  </center>  </div>  </div>  <div class="derecha">hola</div>  </div>  </h:body>  </html> |

**Tabla 10 Código de InicioUsuario.xhtml**

### PÁGINA DE CAMBIO DE CONTRASEÑA

En esta pantalla, se muestra un formulario donde el usuario podrá ingresar su nueva contraseña y esta automáticamente se encriptará y se actualizará en la base de datos.



**Figura 62 Código de CambiarClave.xhtml**

Teniendo el siguiente código

|  |
| --- |
| <?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:h="http://xmlns.jcp.org/jsf/html"  xmlns:a="http://xmlns.jcp.org/jsf/passthrough">    <h:head>  <title> Página de inicio de sesión</title>  <h:outputStylesheet library="css" name="style.css"/>  </h:head>  <h:body>  <div class="container">  <div class="logo">  <h:graphicImage class="logo-imagen" library="images" name="logo.png" width="80"></h:graphicImage>  </div>  <h2>Actualización de contraseña</h2>  <h:form>  <center>  <table>  <tr>  <td>  <h:inputSecret a:placeholder="Contraseña" id="password" value="#{usuarioBean.password}" ></h:inputSecret>  <h:message for="password"></h:message>  </td>  </tr><br/>  <tr>  <td>  <h:inputSecret a:placeholder="Repetir Contraseña" id="Repitepassword" value="#{usuarioBean.repetirPassword}" ></h:inputSecret>  <h:message for="Repitepassword"></h:message>  </td>  </tr>  </table><br/><br/>  </center>  <div class="botones">  <h:commandButton class="boton-login boton" action="#{controladorPrincipal.actualizarClave(usuarioBean.password, usuarioBean.repetirPassword, usuarioBean.usuario)}" value="Actualizar"></h:commandButton><br/><br/>  </div>  </h:form>  </div>  </h:body>  </html> |

**Tabla 11 Código CambiarClave.xhtml**

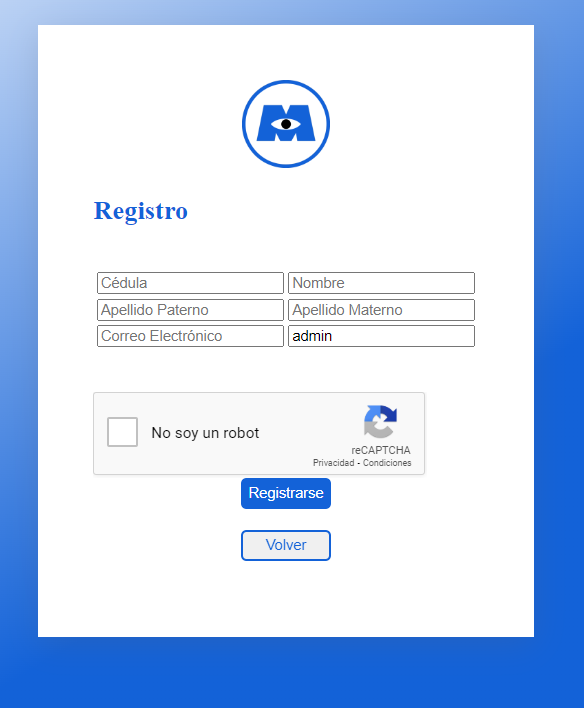
## EJECUCIÓN

### Pantalla de inicio



**Figura 63 Pantalla de inicio**

### Pantalla de registro



**Figura 64 Pantalla de registro**

### Pantalla de cambio de clave



**Figura 65 Pantalla de cambio de contraseña**

### Pantalla de inicio



**Figura 66 Pantalla de inicio**

# CONCLUSIONES

* Se pudo realizar el aplicativo aplicando las tecnologías y frameworks aprendidos en clase, y con la ayuda de librerías externas.
* Es importante manejar beans para formularios.

# RECOMENDACIONES

* Se recomienda instalar y configurar el gestor de BD SQL Server. El grupo de trabajo tuvo complicaciones con la instalación de este gestor, y por tanto trabajó con MySQL.
* Se recomendaría realizar más practicas autónomas con relación al tema y no únicamente las que se impartan en la clase.
* Se recomienda hacer una revisión bibliográfica de los materiales facilitados en clase y de fuentes bibliográficas externas para el mejoramiento del manejo de conceptos.

# REFERENCIAS

|  |  |
| --- | --- |
| [1] | D. Ortinau, «Entorno de ejecución para aplicaciones Xamarin.iOS,» Microsoft, 21 12 2022. [En línea]. Available: https://learn.microsoft.com/es-es/xamarin/ios/deploy-test/environment. [Último acceso: 1 2 2023]. |
| [2] | B. Wagner, «Paseo por el lenguaje C#,» Microsoft, 22 09 2022. [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/. [Último acceso: 1 2 2023]. |
| [3] | P. Carter, «¿Qué es F#?,» Microsoft, 2 1 2023. [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/fsharp/what-is-fsharp. [Último acceso: 1 2 2023]. |
| [4] | EcuRed, «Visual Basic,» EcuRed, 14 9 2019. [En línea]. Available: https://www.ecured.cu/Visual\_Basic. [Último acceso: 1 2 2023]. |
| [5] | G. Hogenson, «MSBuild,» Microsoft, 4 1 2023. [En línea]. Available: https://learn.microsoft.com/es-es/visualstudio/msbuild/msbuild?view=vs-2022. [Último acceso: 1 2 2023]. |