Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

Reporte de Proyecto Terminal II

Título:
Sistema de Gestión de Nómina para PyMEs

Alumno:
ITURBE GARCÍA HORACIO

No. de matrícula:
204203068

Asesorado por:

M. en C. **Rafaela Blanca Silva López**

Contenido

Objetivos Generales *4*

Antecedentes: *4*

Justificación. *7*

Descripción técnica *8*

Especificaciones Técnicas. *9*

DIAGRAMAS UML *11*

Administración de Recursos Humanos 11

Administración de Nómina 12

Administración de Nómina 12

Reportes 13

DIAGRAMAS *14*

DE NAVEGACIÓN *14*

DIAGRAMAS DE SECUENCIA *16*

REGISTRAR PERSONA 16

MODIFICAR PERSONA 17

REGISTRAR EMPLEADO 18

MODIFICAR EMPLEADO 19

REGISTRAR NÓMINA 20

MODIFICAR NÓMINA 21

REGISTRAR TABULADOR 22

MODIFICAR TABULADOR 23

PROCESAR NÓMINA 24

REGISTRAR INCIDENCIA 25

MODIFICAR INCIDENCIA 26

REPORTES NÓMINA 27

REPORTES DE EMPLEADO 28
Diccionario de datos *30*
datos_persona 30
domicilio_persona 30
referencias_persona 30
idiomas_persona 31
academicos_persona 31
laboral_persona 31
academicos_niveles 31
idioma 32
Empleado 32
Accesos 32
registro_incidencia 33
incidencias 33
nomina 33
nomina_conceptos 34
tabulador 34
tabulador_niveles 34
nomina_procesada 34
nominapro y reporteempe 35
Equema físico *35*
CÓDIGO FUENTE DE LA APLICACIÓN *50*

Objetivos Generales

Diseñar e implementar un sistema informático que permita gestionar el proceso de nómina y la administración de los recursos humanos involucrados en una Pequeña y mediana empresa.

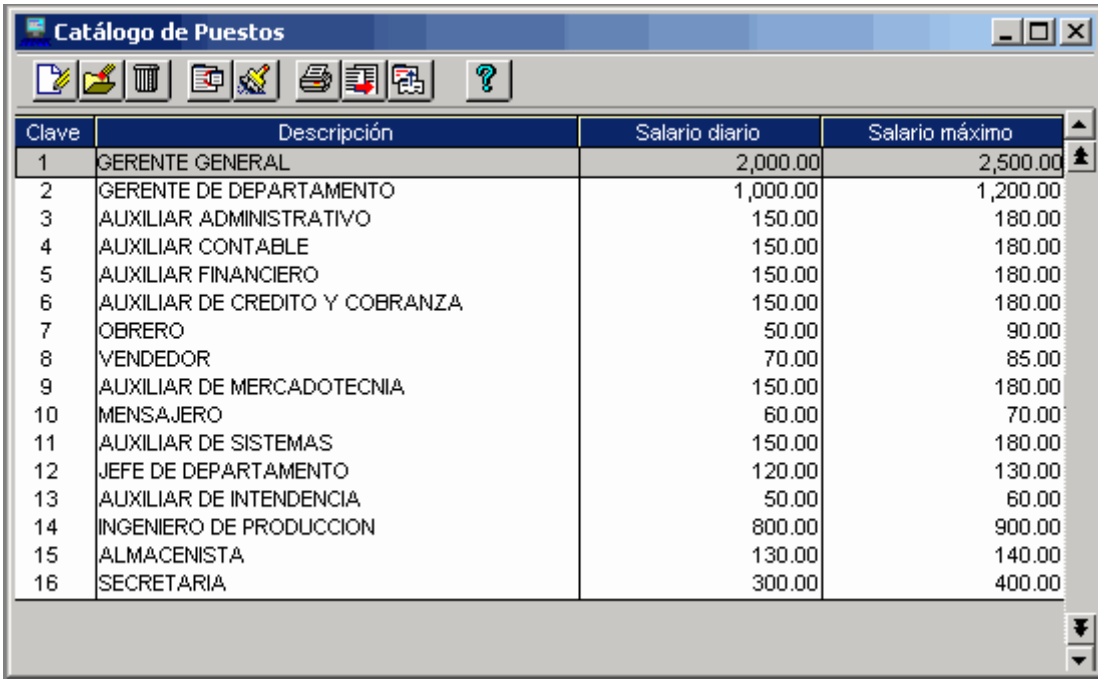
Objetivos Particulares:

- Diseñar un sistema que permita realizar procesos de nómina según se configure (semanal, quincenal, mensual).
- Diseñar módulos que permitan la configuración de los elementos que tendrá el proceso de la nómina.
- Implementar el diseño utilizando tecnologías abiertas y/o estándares.
- Documentar las diferentes etapas del desarrollo.

Antecedentes:

Producto comercial

Aspel-Noi



Clave	Descripción	Salario diario	Salario máximo
1	GERENTE GENERAL	2,000.00	2,500.00
2	GERENTE DE DEPARTAMENTO	1,000.00	1,200.00
3	AUXILIAR ADMINISTRATIVO	150.00	180.00
4	AUXILIAR CONTABLE	150.00	180.00
5	AUXILIAR FINANCIERO	150.00	180.00
6	AUXILIAR DE CREDITO Y COBRANZA	150.00	180.00
7	OBRERO	50.00	90.00
8	VENDEDOR	70.00	85.00
9	AUXILIAR DE MERCADOTECNIA	150.00	180.00
10	MENSAJERO	60.00	70.00
11	AUXILIAR DE SISTEMAS	150.00	180.00
12	JEFE DE DEPARTAMENTO	120.00	130.00
13	AUXILIAR DE INTENDENCIA	50.00	60.00
14	INGENIERO DE PRODUCCION	800.00	900.00
15	ALMACENISTA	130.00	140.00
16	SECRETARIA	300.00	400.00

Figura 1. Puestos

Aspel-NOI permite el manejo y control de los diferentes aspectos de la nómina y de la información de los empleados relacionándolos a los diferentes departamentos a los que pertenecen.

Algunas de las funcionalidades del sistema:

- Catálogo de trabajadores.
- Cálculo del salario diario integrado y consulta de su desglose.
- Control de incidencias.
- Cálculo del impuesto local.
- Manejo de reingresos.

Características:

- Se instala en un equipo solamente, se trabaja de manera local y la información se queda almacenada en dicho equipo.
- La información NO es portable.

Productos de Software Libre

iQdesk

Software diseñado para ofrecer algunas funciones básicas de gestión, tanto para profesionales como para pequeñas empresas; su sencilla ventana de trabajo y funciones esenciales lo convierten en una herramienta ideal para iniciar la organización. Este sistema está compuesto por 5 módulos:

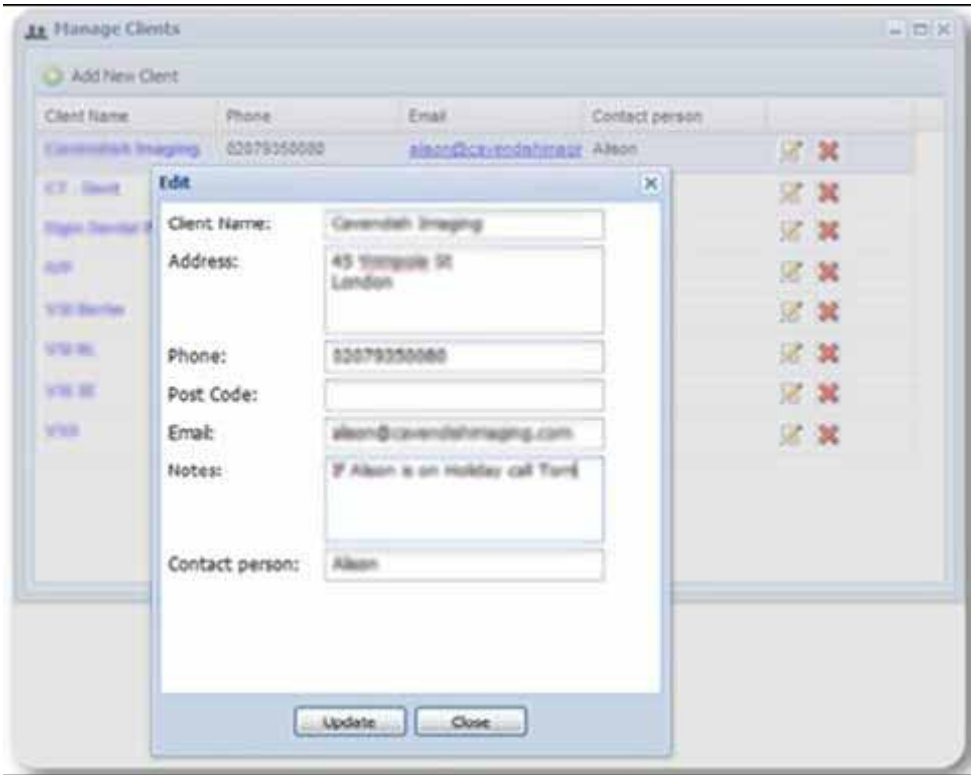


Figura 2. iQdesk

- Tareas.

- Administrar Clientes.
- Gestión de Empleados y Facturación.
- Costos.
- Administración

NolaPro

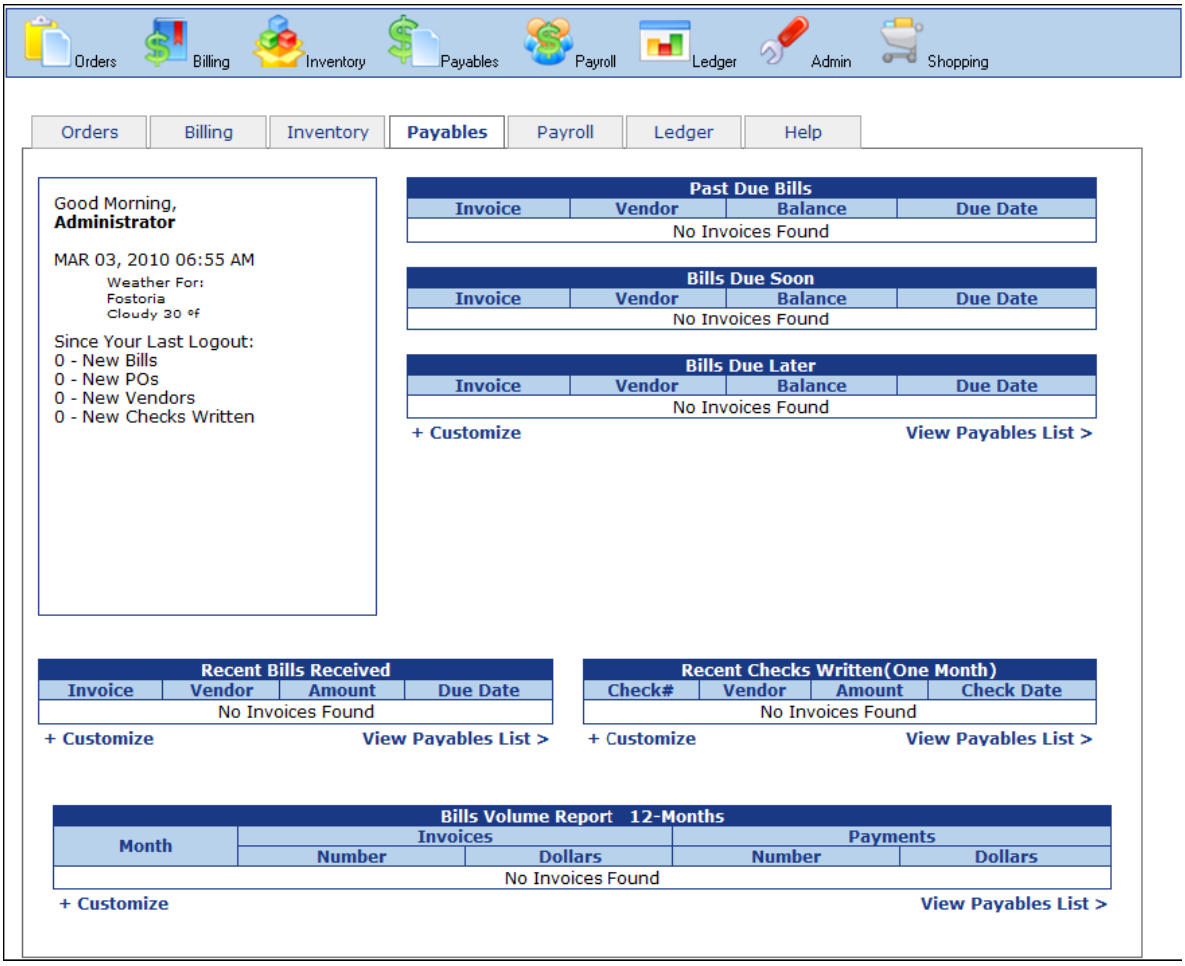


Figura 3. NolaPro

Aplicación que tiene como módulos de contabilidad, inventario, gestión de empleados y gestión de nómina.

Noi. El usuario tiene que adquirir licencias que van desde $7,000 a $13,000 pesos, dependiendo de la cantidad de usuarios. No se puede tener acceso a la información propia de la empresa, hay archivos binarios que sólo el programa puede acceder y los cuales tienen información que se registra al programa (empleados, departamentos). Dependiendo de la instalación se tendrían que adquirir licencias para Oracle o SQL server, lo cual representa un costo mayor. Llega a tener conflictos con algunos antivirus.
El proyecto que se propone trabajará sobre ambiente web, no tiene complicaciones con antivirus, es libre, ya que el acceso a los datos puede realizarse sin complicaciones. Se mantendrá el uso de estándares abiertos para que sea portable

i**Qdesk.** De los cinco módulos que tiene, uno de ellos se enfoca a la Gestión de Empleados, la propuesta de este proyecto es, además de tener una Gestión de empleados, llevar a cabo el procesamiento de nómina.

NolaPro. Tiene Gestión de Nómina y Gestión de Empleados, la limitante es que la configuración de impuestos no puede cambiarse y están basados en leyes de otro país.
Este proyecto otorga la facilidad de hacer configuraciones de nómina empezando por la periodicidad (semanal, quincenal, mensual), de los conceptos de percepciones y deducciones que el procesamiento pueda tener y permite el agregar incidencias (faltas, incapacidades, permisos).

Justificación.

Este proyecto está enfocado a satisfacer tres de las necesidades de una PyME, estas son:

- Administrar los recursos humanos
- Administrar la nómina
- Generar reportes de recursos humanos y de la nómina

El proyecto de Sistema de Gestión de Nómina ofrece a la PyME la facilidad de tener la organización en estos aspectos, a pesar de que existe Software Libre con algunas de estas caacterísticas, no satisfacen de manera completa las tres necesidades mencionadas anteriormente y en caso del producto comercial, es un egreso que este proyecto busca que la PyME no realice.
Este proyecto es un problema que resuelve un Ingeniero en Computación, ya que lo que se requiere es desarrollar un Sistema de Información que pueda integrar la información que la PyME posee, almacenarla en una base de datos y poder interactuar con ella a través de una interfaz web.

Además el Ingeniero en Computación es capaz de:
 Analizar, diseñar, adaptar, implementar y mantener proyectos de computación que involucren software y hardware

 Analizar y diseñar sistemas de información
 Dominar los principios teóricos y prácticos de las redes de computadoras y la interoperabilidad de aplicaciones.

Las cuales son habilidades requeridas para llevar a cabo este proyecto.
Una PyME está en constante crecimiento y con el tiempo puede volverse una empresa bien establecida, una posible continuación de este proyecto es agregar módulos que amplíen la funcionalidad del sistema, que pueden ser:

- Registrar centros de costos para configuraciones de nómina
- Implementar módulos que se relacionen con la contabilidad de la PyME
- Publicaciones de Bolsa de trabajo

Descripción técnica

El sistema de Gestión de Nómina para PyMEs, es un sistema que mantiene la información de los empleados, iniciando con la captura de su Curriculum Vitae. Después de esto, al empleado se le puede asignar una nómina ya configurada para percibir su sueldo y se puede generar dada la periodicidad configurada. Además permitirá obtener reportes para un conjunto de empleados sobre las nóminas procesadas en periodos seleccionados.
Se plantea desarrollar el sistema utilizando plataforma java, con el fin de que pueda ser instalado en servidores con alta disponibilidad, multiplataforma, con sistema operativo y capacidad de procesamiento diferentes.

El sistema contará con los siguientes bloques:

a) Interfaz de usuario.
b) Administración de nómina.
c) Proceso de nómina
d) Administración de recursos humanos
e) Generador de reportes de nómina y de recursos humanos.

Interfaz de usuario: Esta interfaz permitirá a un cliente de Internet basado en HTML, tener acceso a toda la funcionalidad del sistema.

Administración de nómina. En este módulo se opera todo lo relacionado con la configuración de nómina, algunas de las configuraciones son:

- Configuración de tabuladores salariales.
- Configuración de impuestos.
- Configuración de conceptos y deducciones.

Proceso de nómina: En este módulo se registran las incidencias, permisos, faltas o vacaciones para que la nómina sea procesada y se generen los pagos correspondientes.

Administración de recursos humanos: Dentro de este bloque se estructura y almacena la información del conocimiento tácito de los recursos humanos que conforman el grupo de trabajo.

Generador de reportes de nómina y de recursos humanos: En éste módulo se realizan reportes en pdf, de la nómina generada por empleado y reportes de los empleados que se encuentran activos.

El sistema considera 4 roles:

Administrador: se encarga de la administración del sistema, tiene todos los privilegios y acceso total al sistema.

Operador: Encargado de configurar la nómina y de agregar empleados en caso de necesitarlo o También puede generar reportes y consultar los recibos de nómina.

RH: Encargado de administrar la información de los recursos humanos: dar de alta, bloquear, modificar la información y generar reportes.

Otro: Puede entrar para registrar o actualizar su Curriculum Vitae y consultar el recibo de nómina.

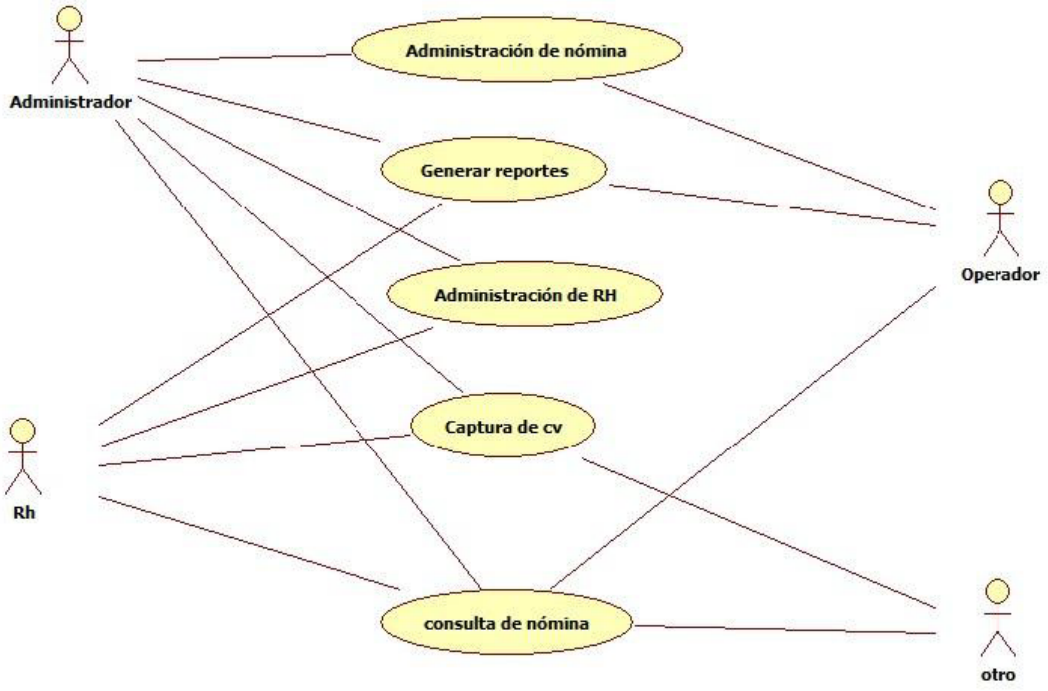


Figura 4

El diagrama de casos de uso mostrado en la Figura 4, describe los roles que se manejarán en el sistema asociado a algunas de las tareas principales.

El sistema se desarrollará utilizando, la plataforma abierta Eclipse, sobre sistema operativo Windows, ya que permite realizar el desarrollo tanto sobre la parte que se desarrollará en plataforma Java como en la parte de la interfaz de usuario. También permite integrar servidores de aplicaciones abiertas, así como bases de datos relacionales en general.

Especificaciones Técnicas.

Se plantea utilizar el estándar JEE como base, con el fin de que el sistema pueda ser montado a futuro sobre diferentes servidores de portal, como pueden ser Tomcat, Portal Server, lo cual permite la extensibilidad del sistema.

La figura 5 es un diagrama a bloques de los principales componente que integran el sistema, tanto del lado cliente, como del lado servidor en su ambiente de ejecución. En cada caso se muestran las tecnologías que se van a utilizar para su implementación. En el caso de lado servidor, se usará Hibernate para la conexión a la base de datos y Java.io para generar los archivos de registro (Logs) de la aplicación.

Bloques de lado cliente Bloques de lado servidor

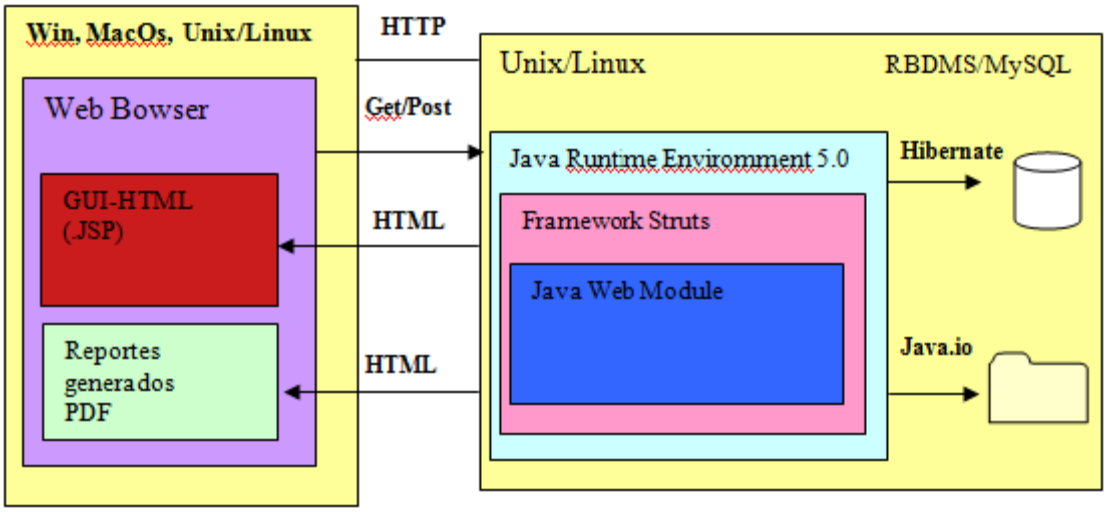


Figura 5. Diagrama de bloques del sistema

El protocolo de comunicación que se utilizará entre clientes y servidor será HTTP, debido a que en una aplicación que requiere ser ejecutada desde la red, para dar soporte a diferentes clientes que pueden encontrarse en diferentes lugares. Tanto los clientes como el servidor, podrán ser ejecutados sobre sistemas operativos distintos, con el fin de tener amplias posibilidades de implementación.

El sistema se dará por terminado una vez que hayan realizado los módulos:

- Registrar empleado.
- Registrar una nómina y sus parámetros de configuración (Administración de nómina).
- Generar reportes de empleados y de nóminas
- Administrar a un empleado registrado previamente
- Procesar una nómina.

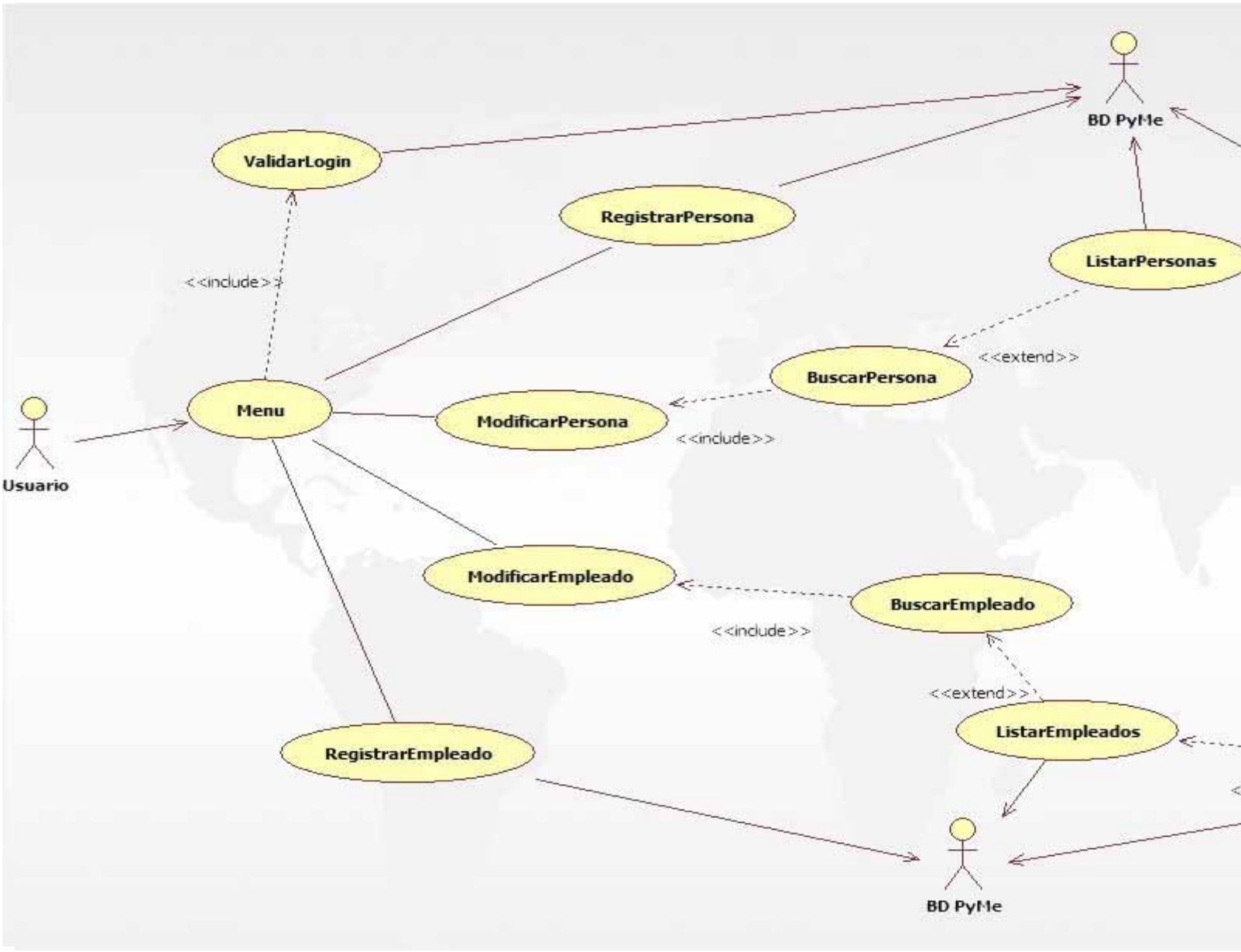
Se realizarán las pruebas con cuatro usuarios, cada uno de ellos con el rol que el sistema considerará.

- Usuario administrador.
- Usuario operador
- Usuario de recursos humanos
- Usuario público

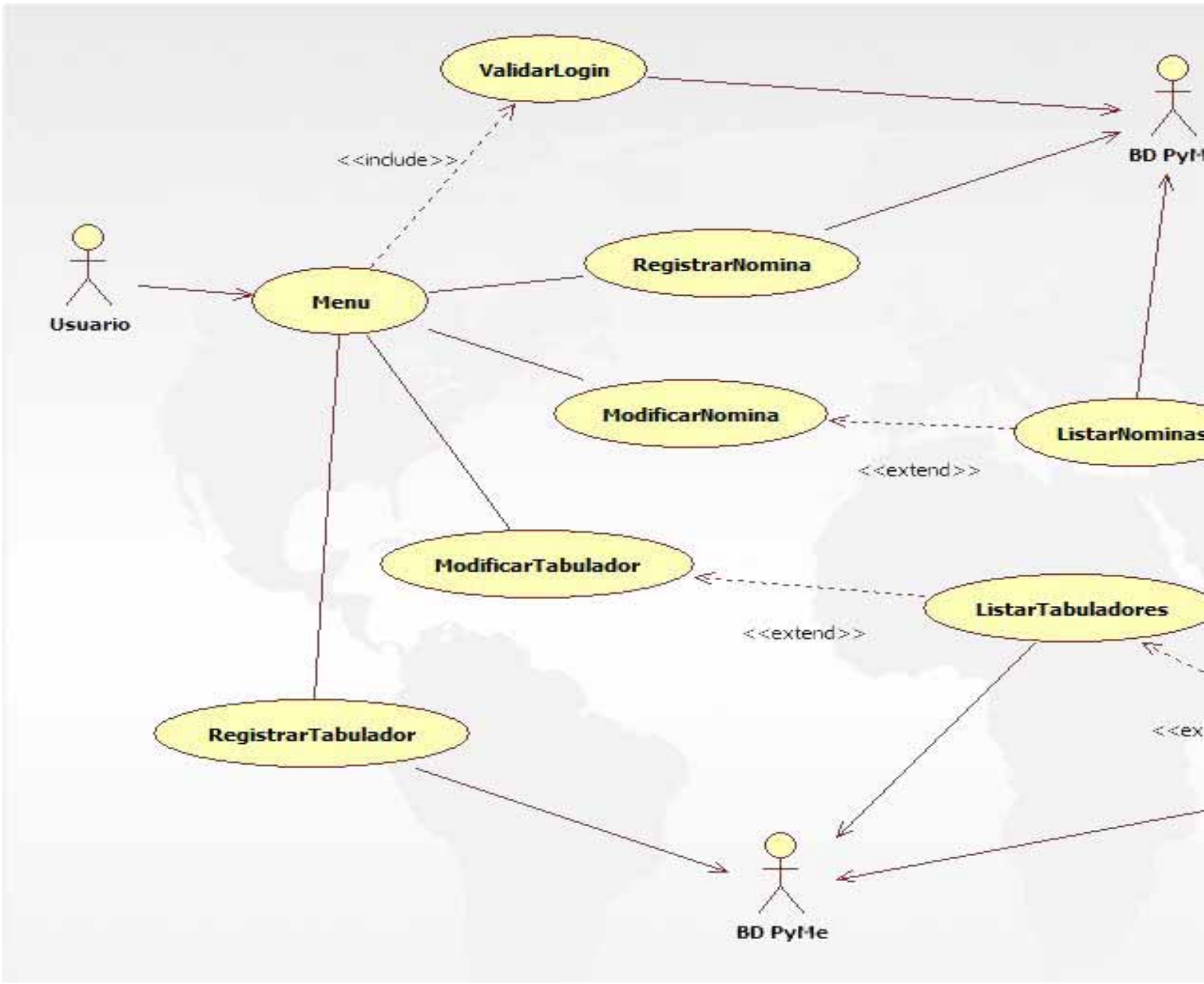
DIAGRAMAS UML

CASOS DE USO

Administración de Recursos Humanos

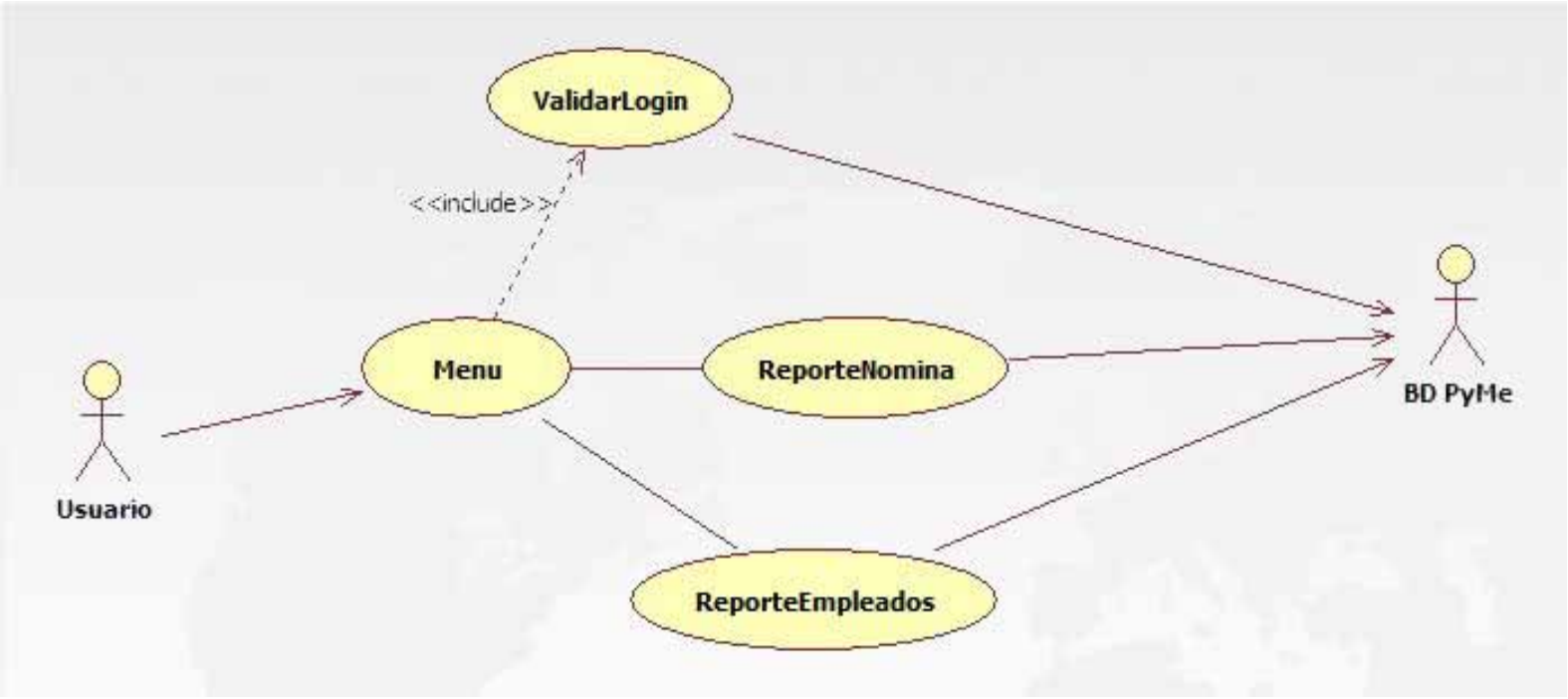


Administración de Nómina



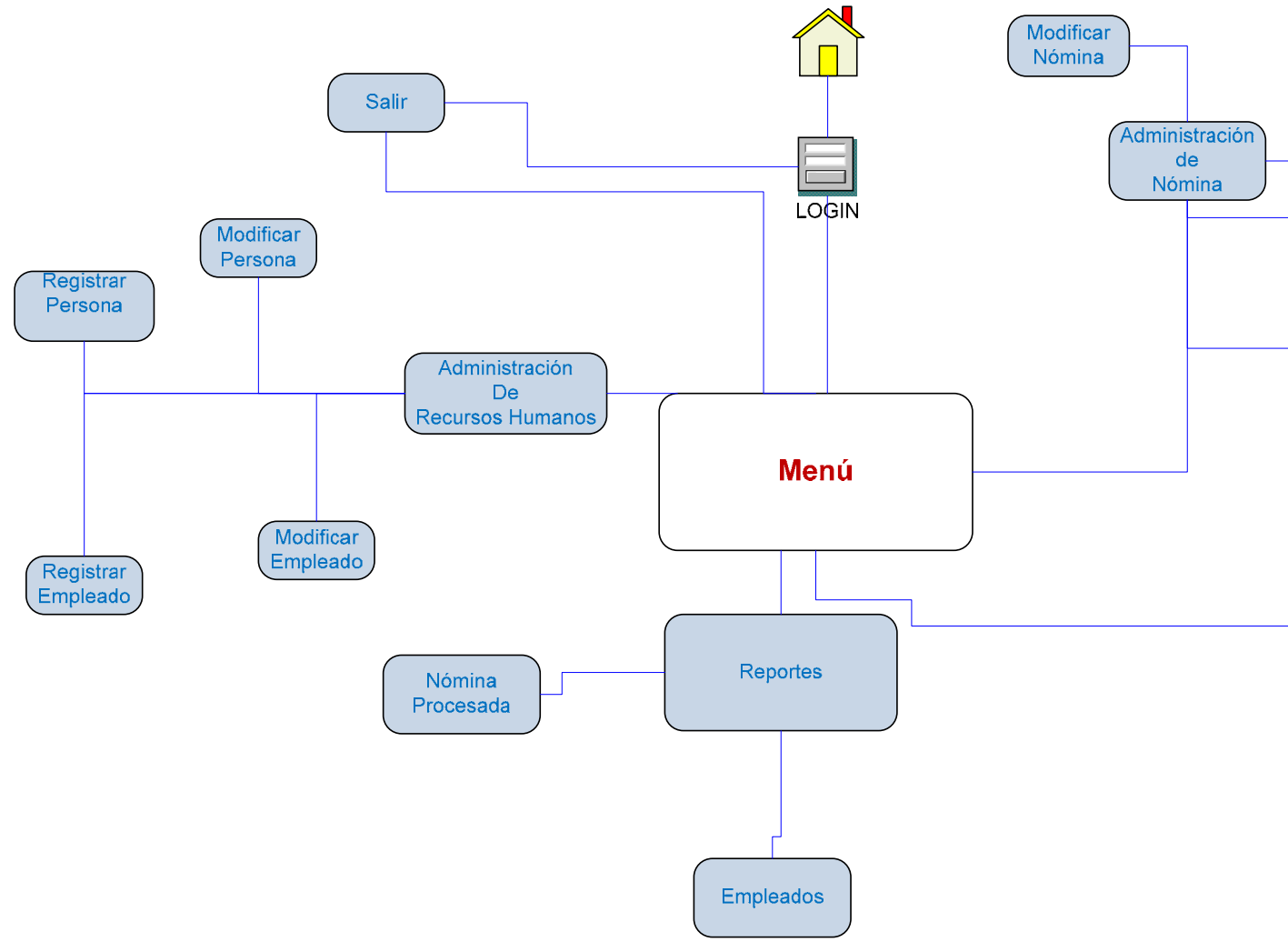
Administración de Nómina

Reportes

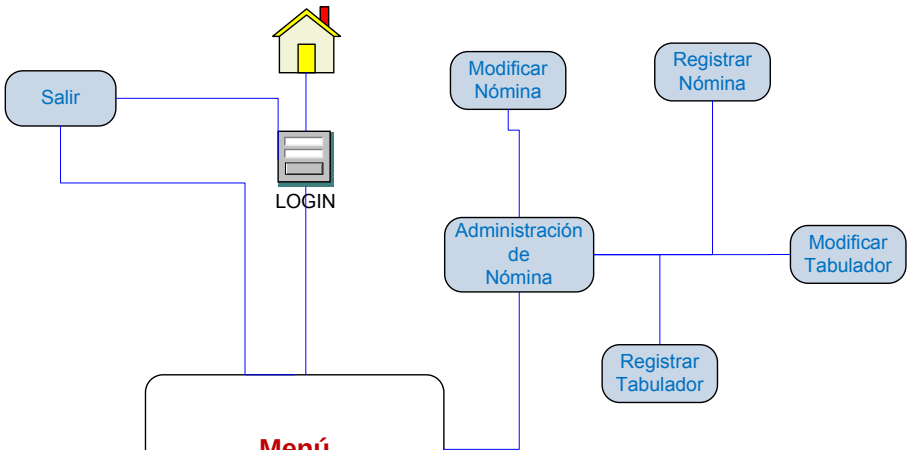

ValidarLogin
<<include>>
Menu
ReporteNomina
BD PyMe
Usuario
ReporteEmpleados

DIAGRAMAS
DE NAVEGACIÓN

PARA USUARIO CON ROL ***ADMINISTRADOR***

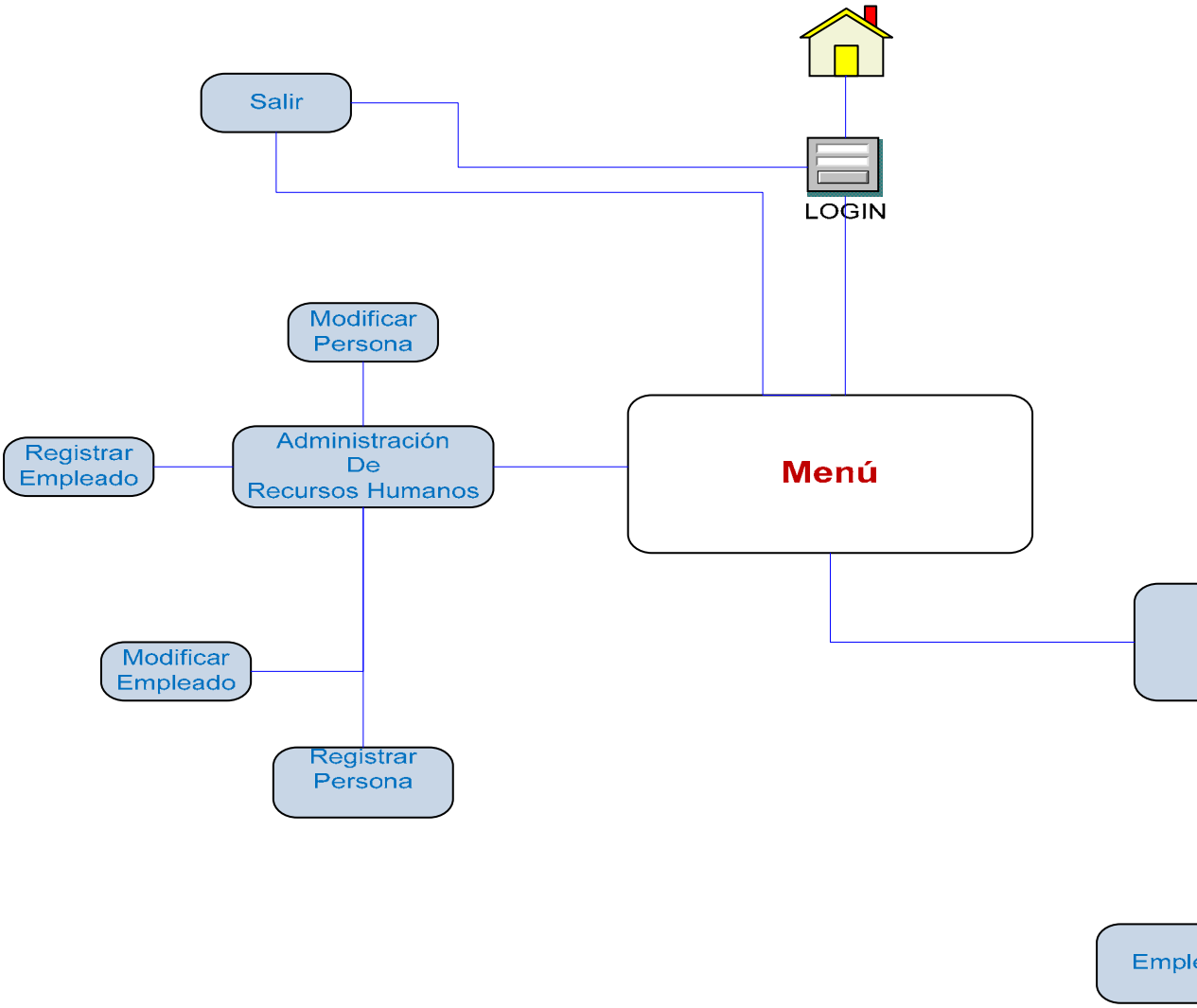


PARA USUARIO CON ROL ***OPERADOR***



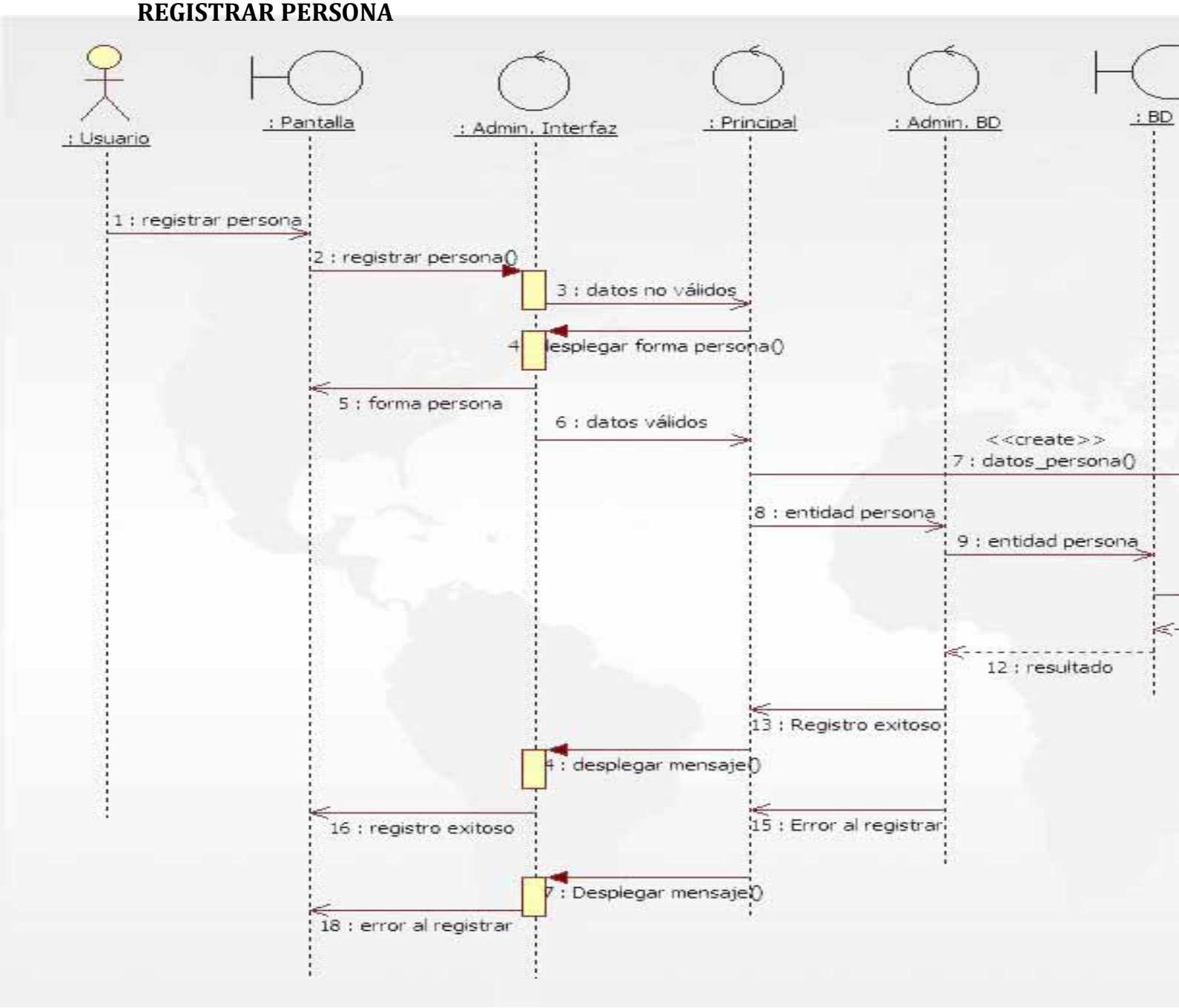
PAR

PARA USUARIO CON ROL *RH*

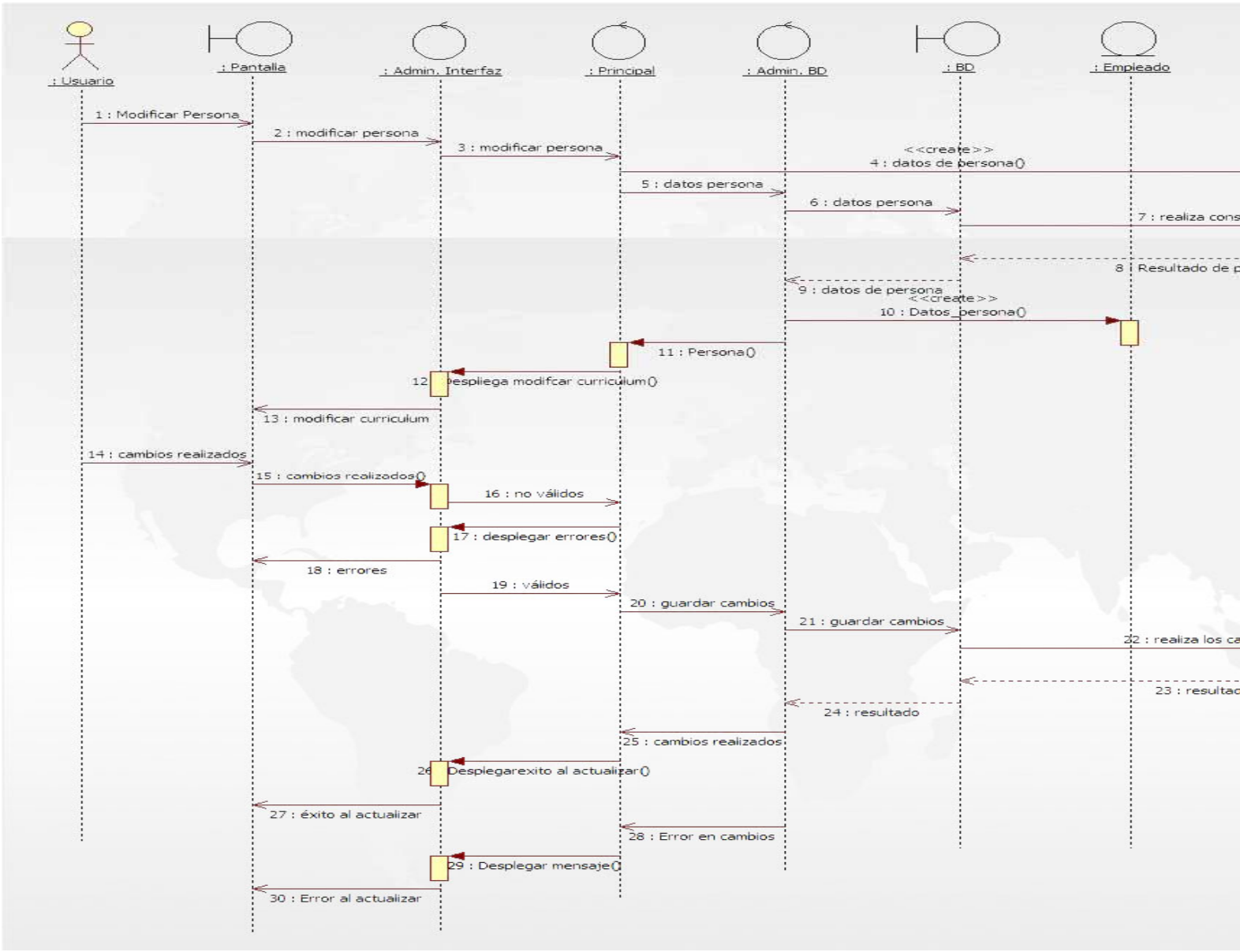


DIAGRAMAS DE SECUENCIA

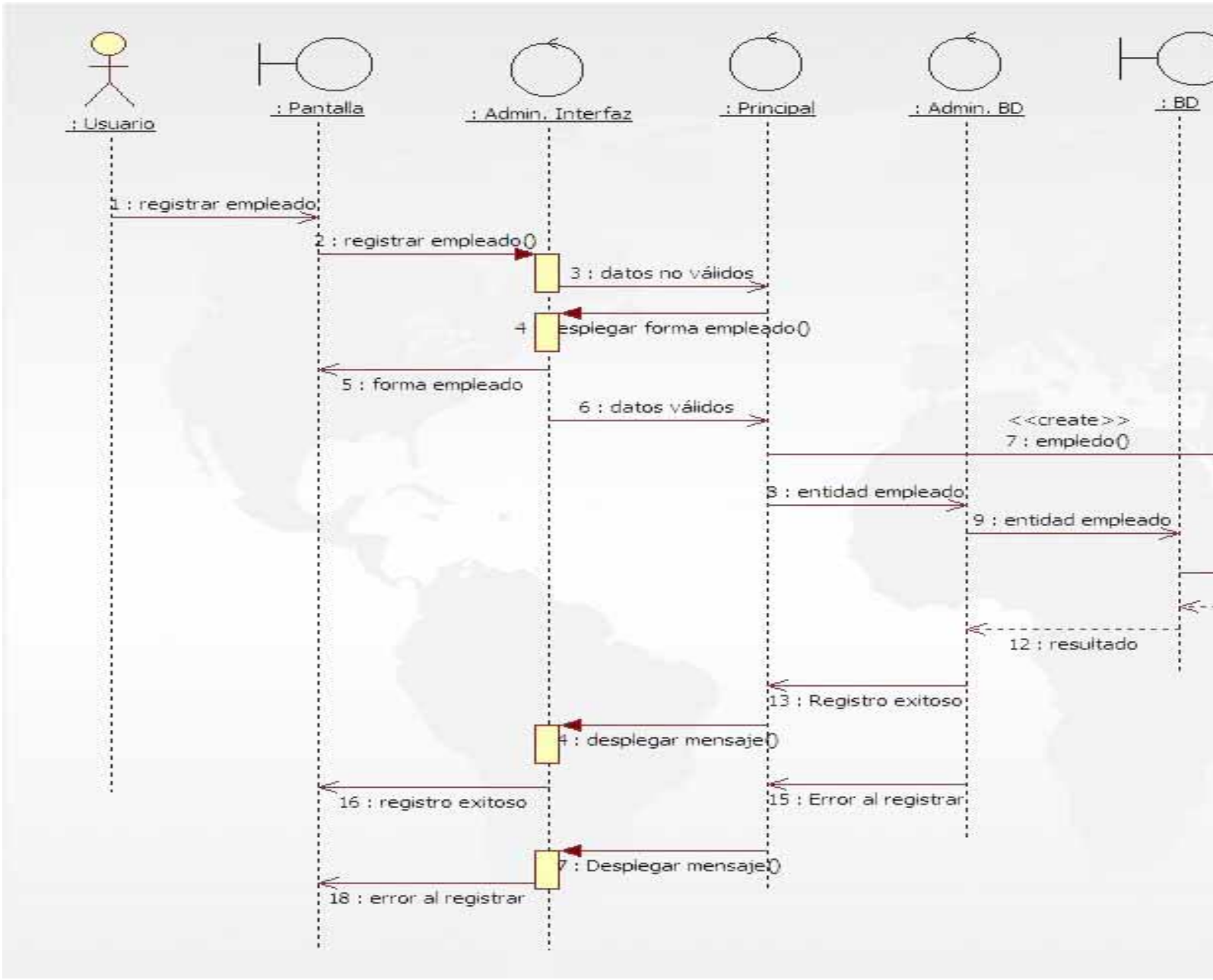
REGISTRAR PERSONA



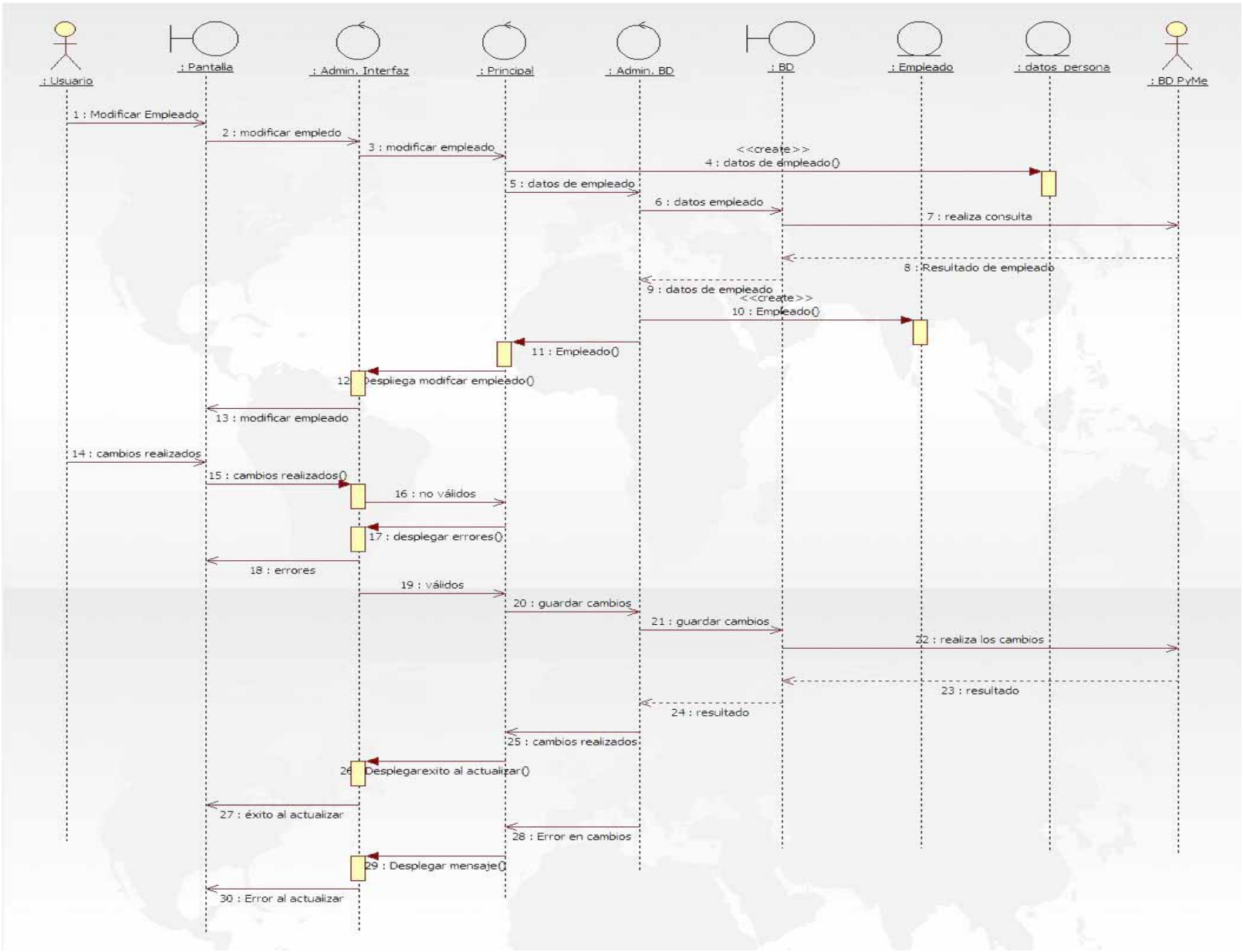
MODIFICAR PERSONA



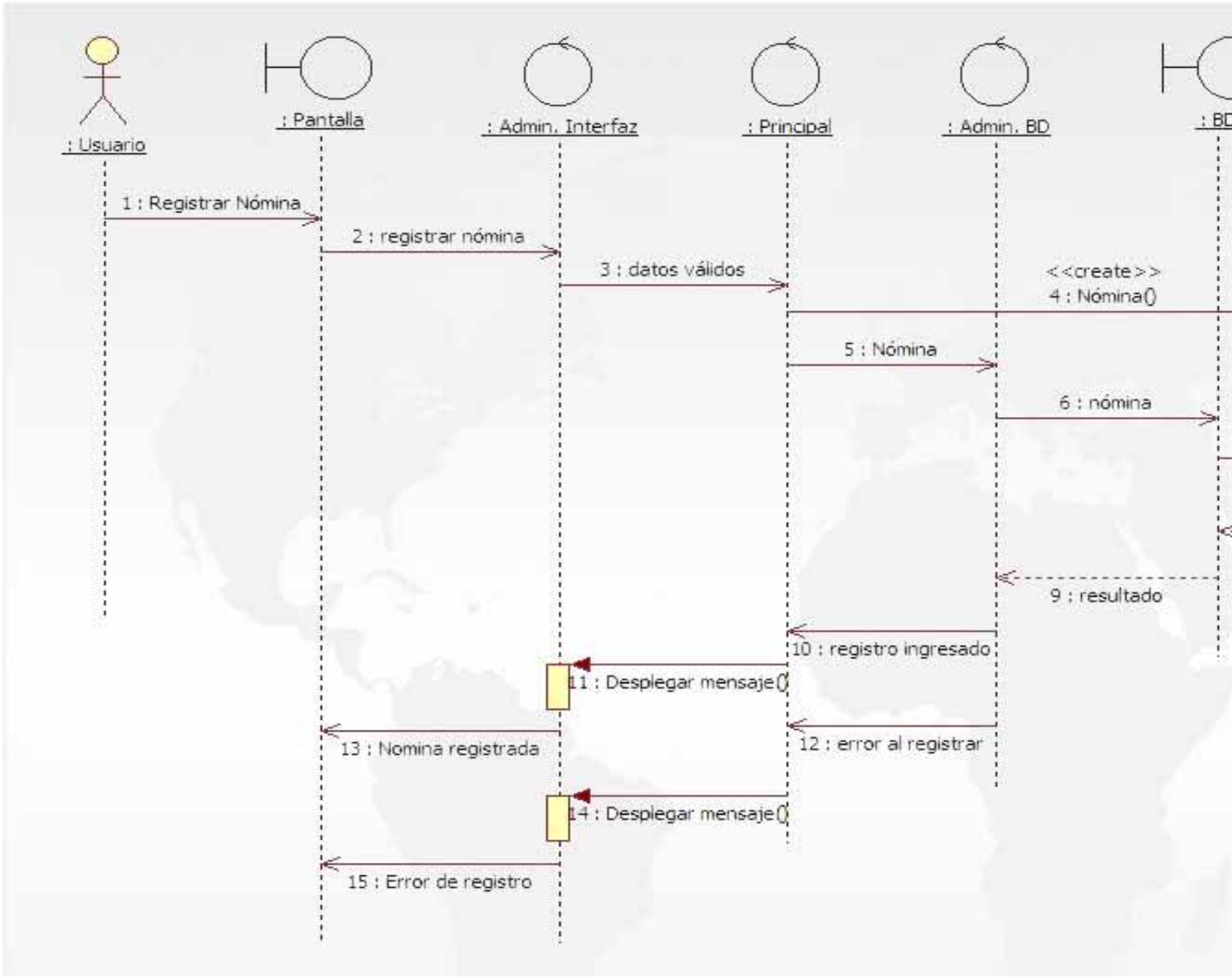
REGISTRAR EMPLEADO



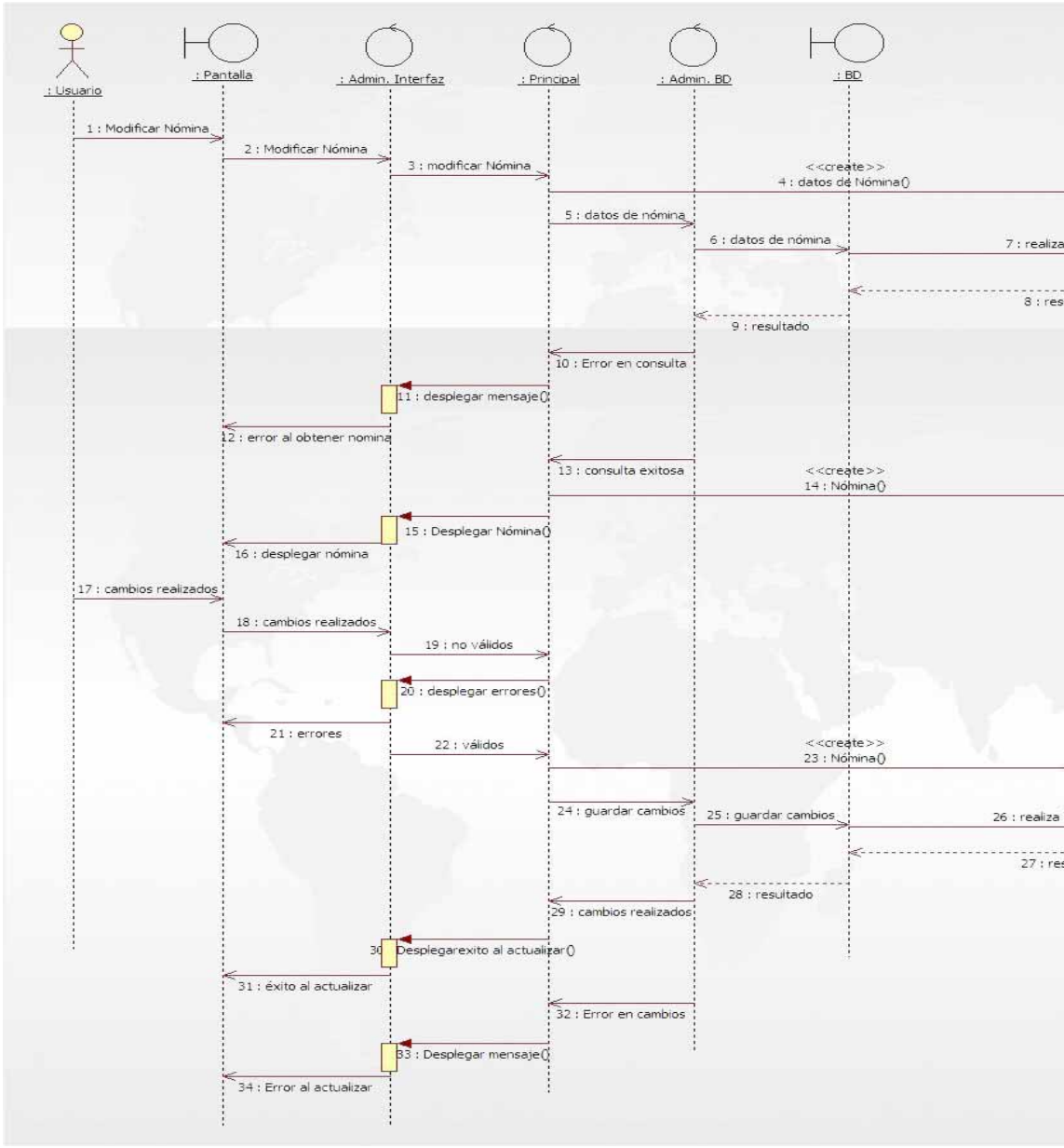
MODIFICAR EMPLEADO



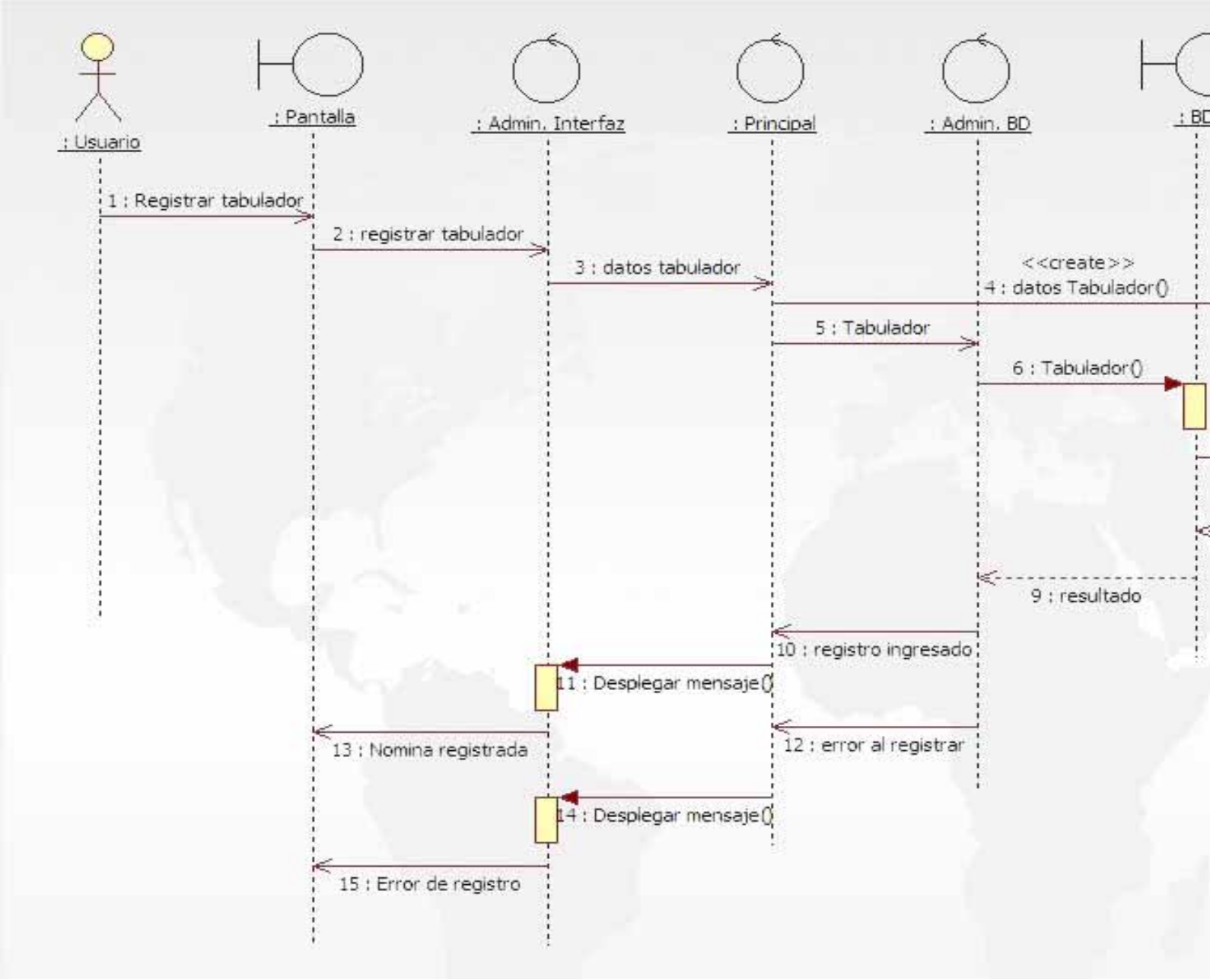
REGISTRAR NÓMINA



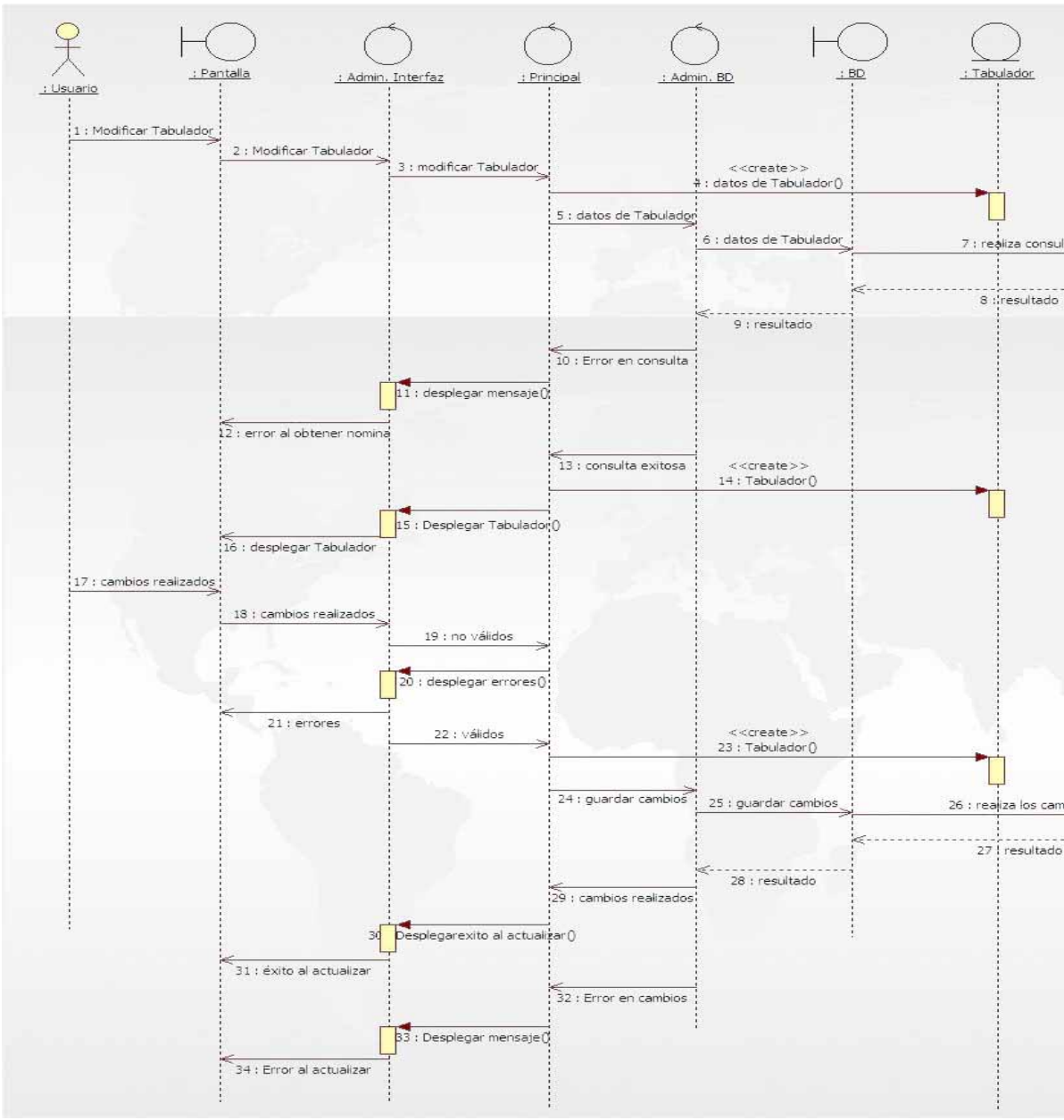
MODIFICAR NÓMINA



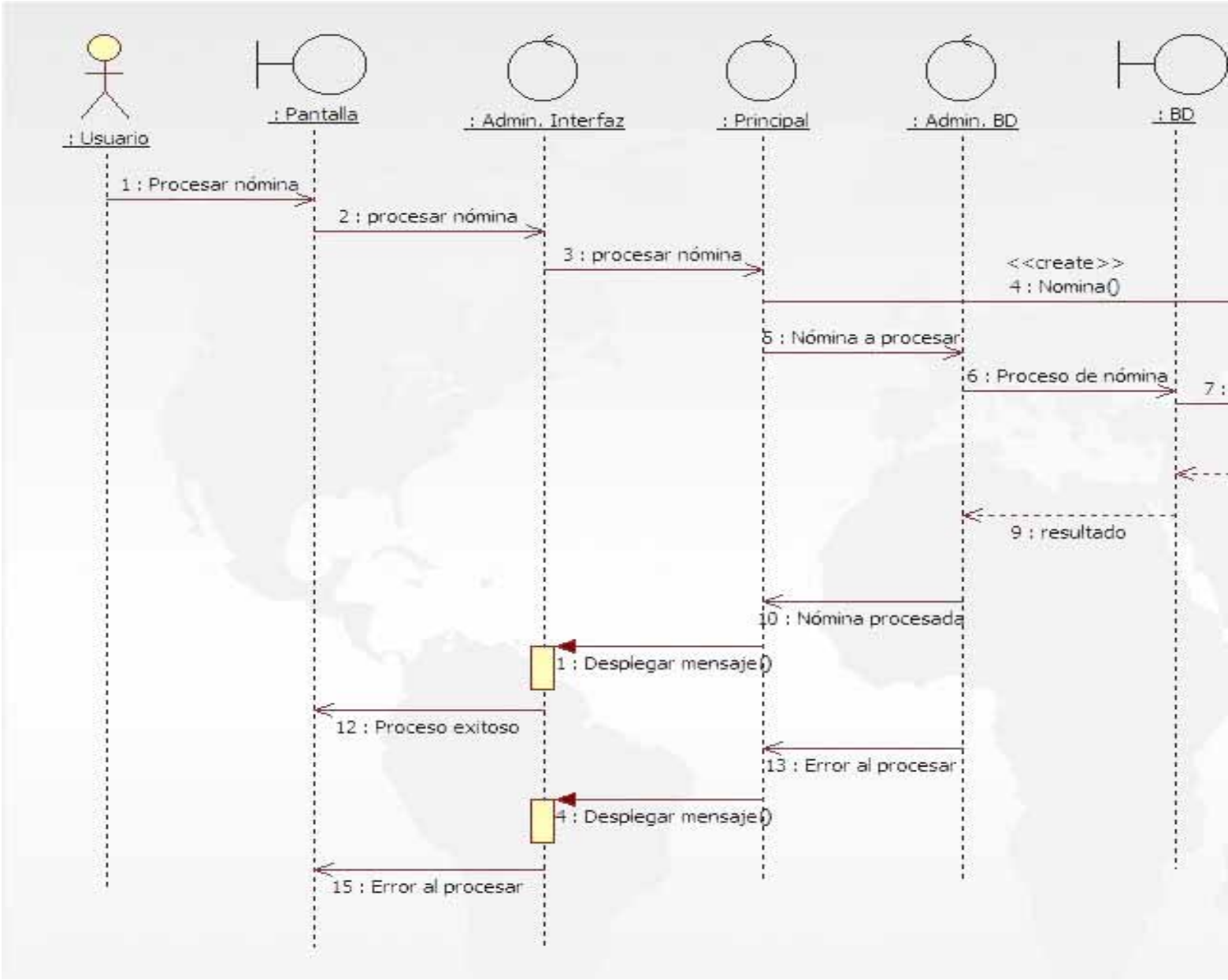
REGISTRAR TABULADOR



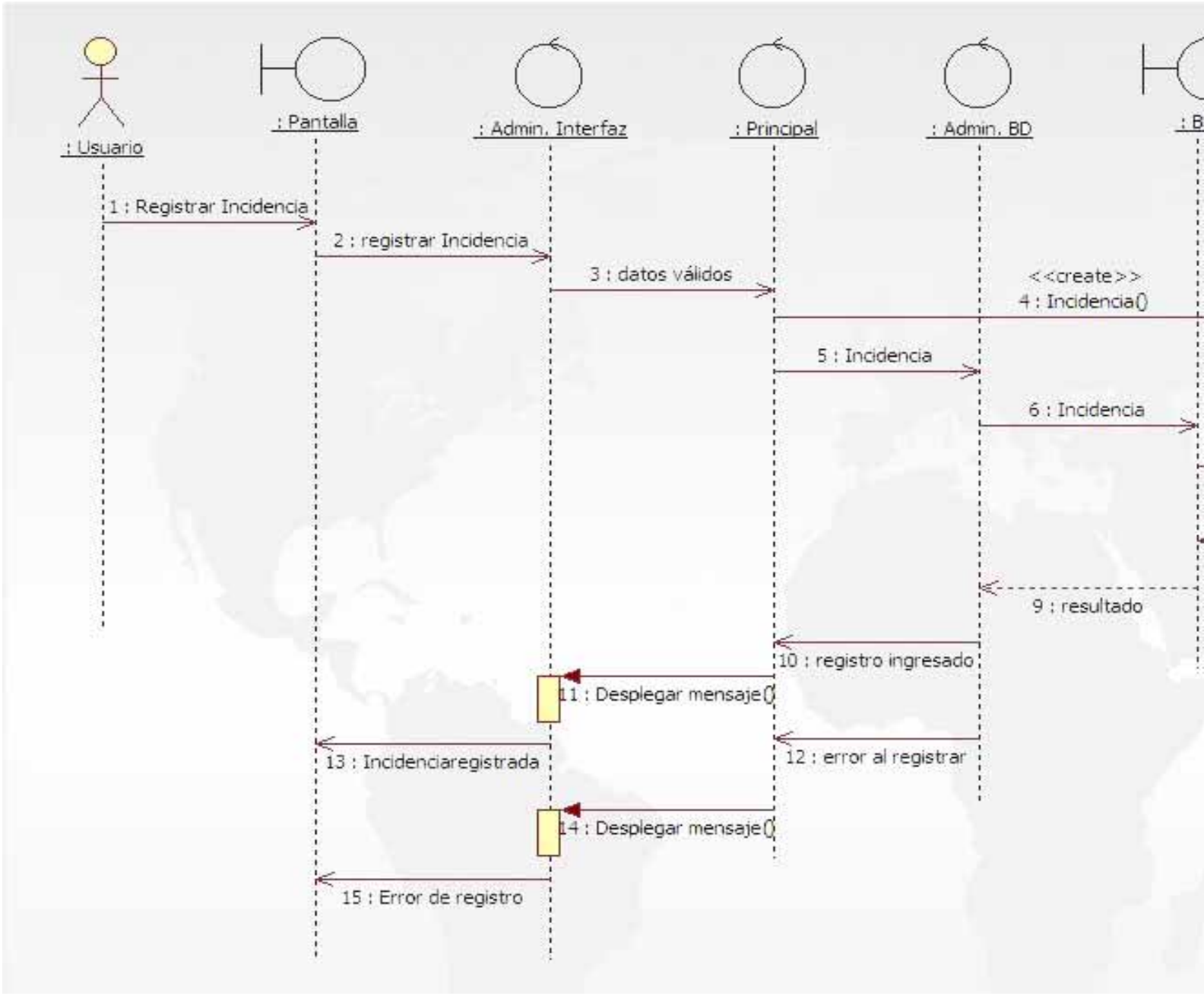
MODIFICAR TABULADOR



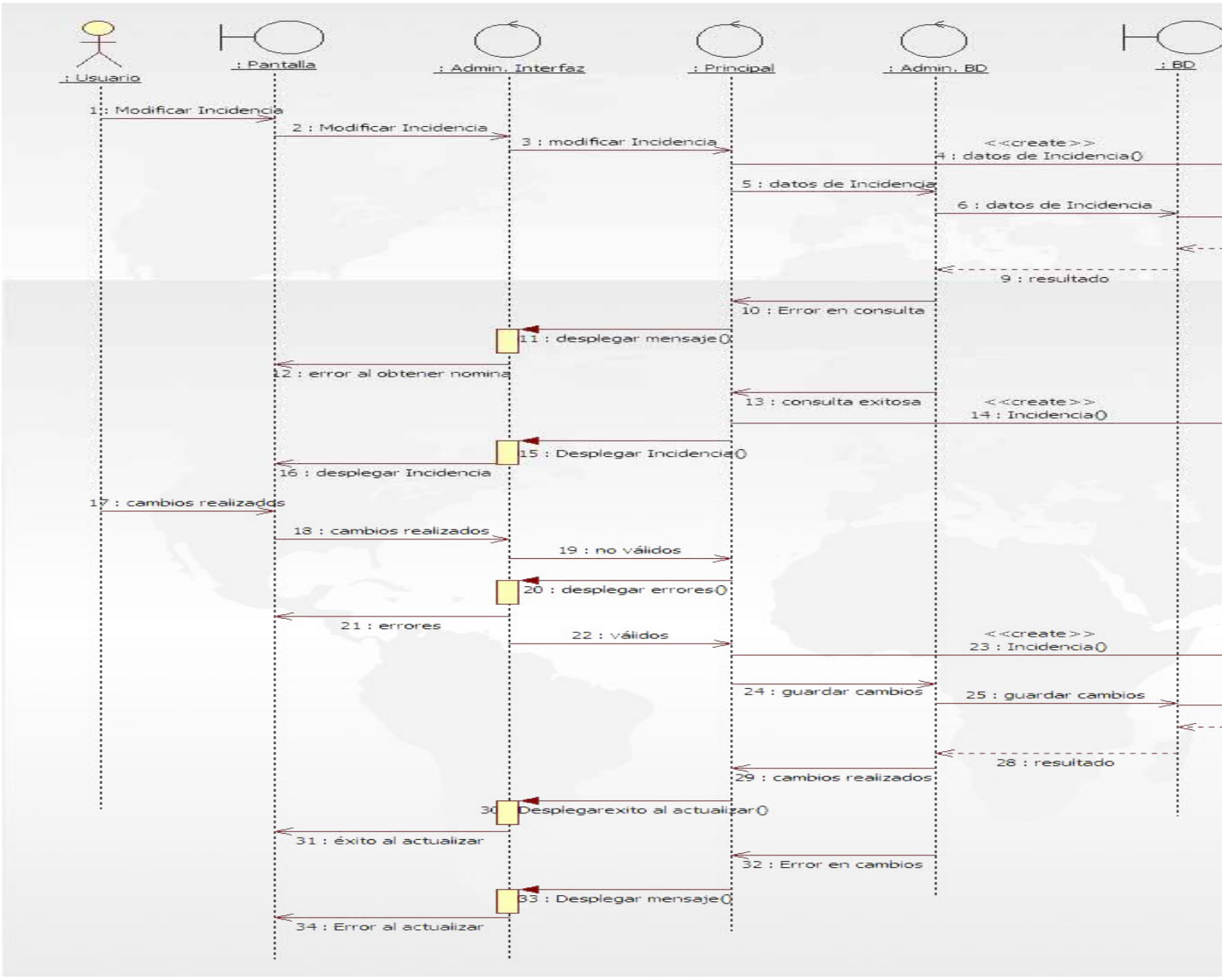
PROCESAR NÓMINA



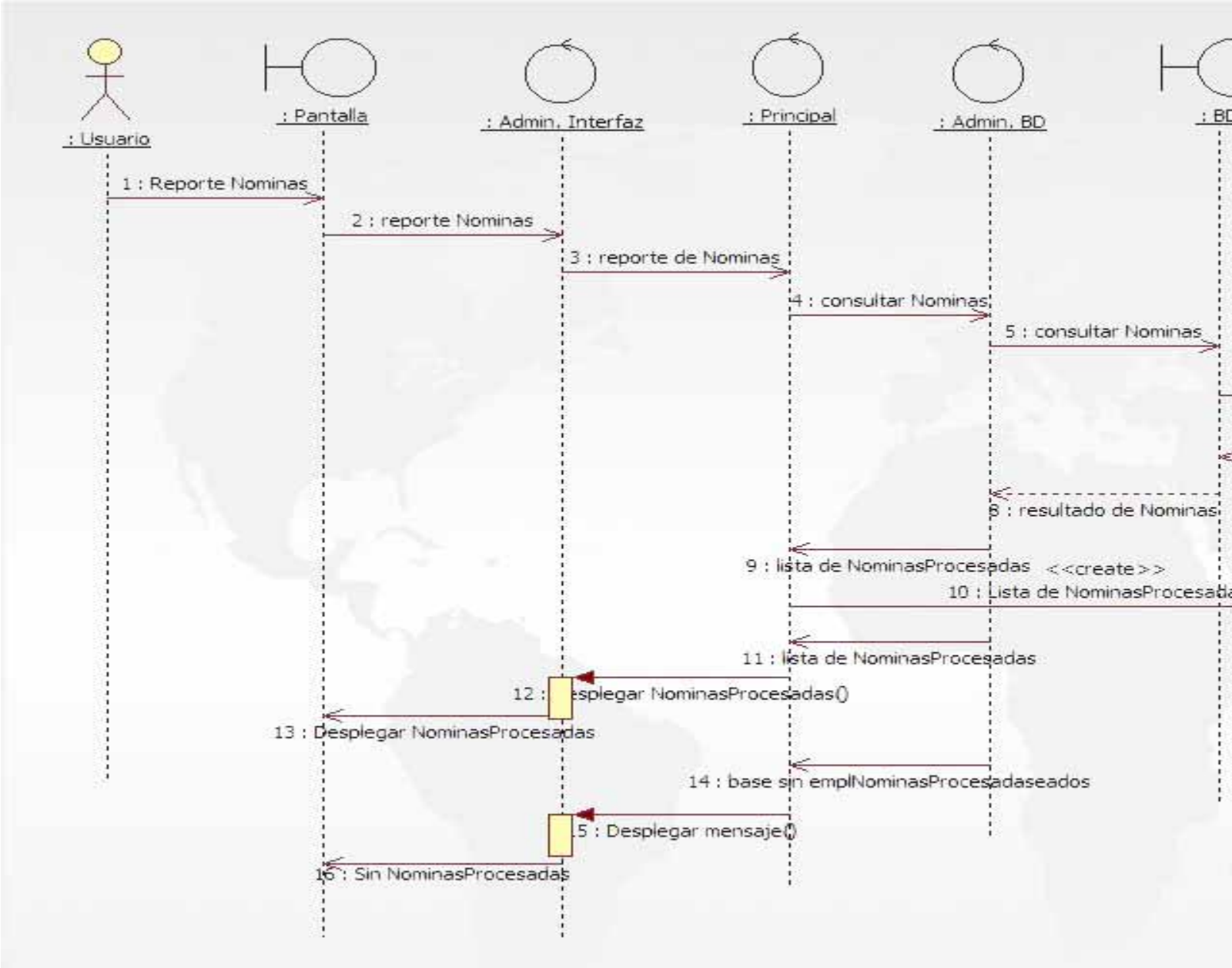
REGISTRAR INCIDENCIA



MODIFICAR INCIDENCIA



REPORTES NÓMINA



REPORTES DE EMPLEADO

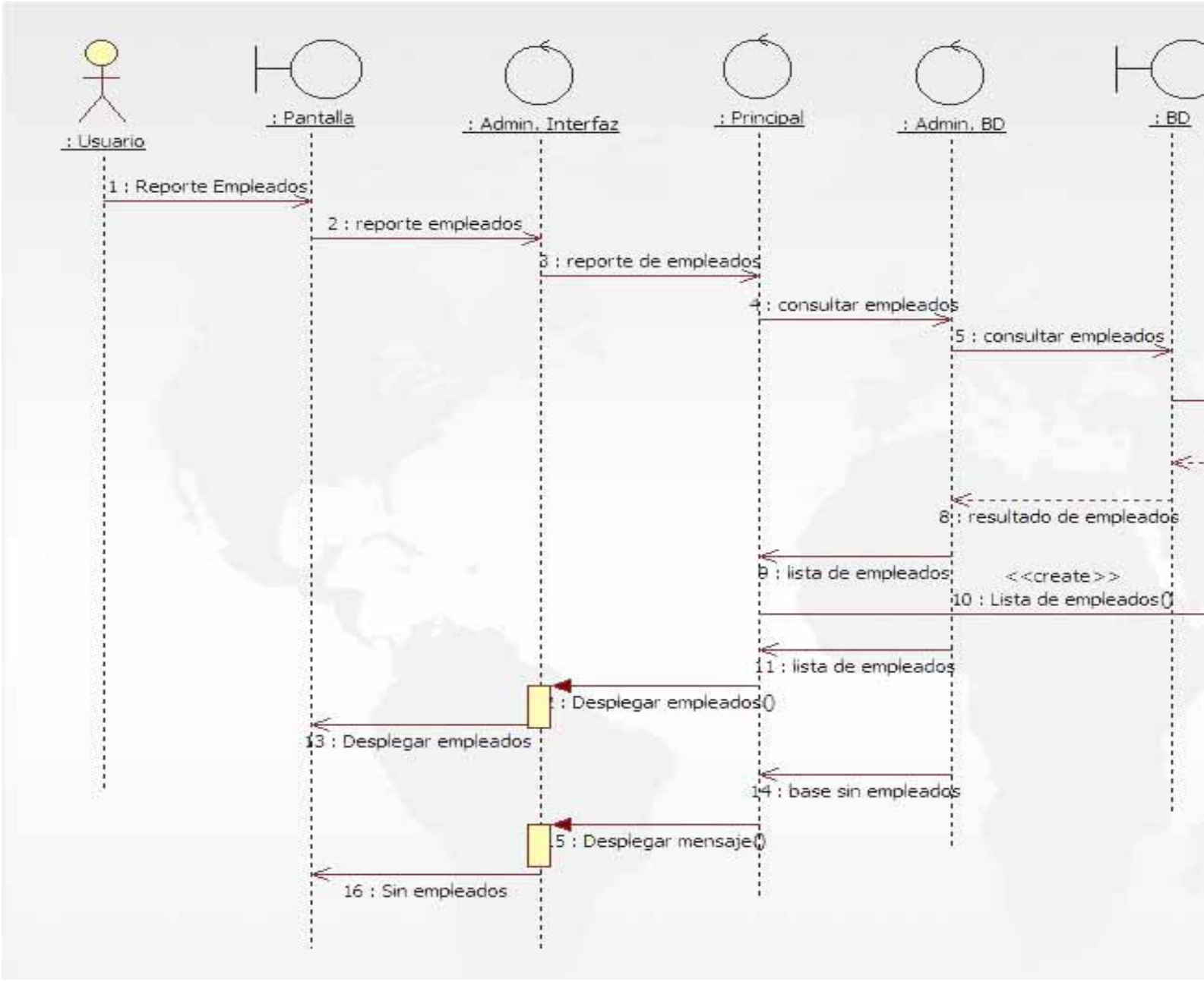
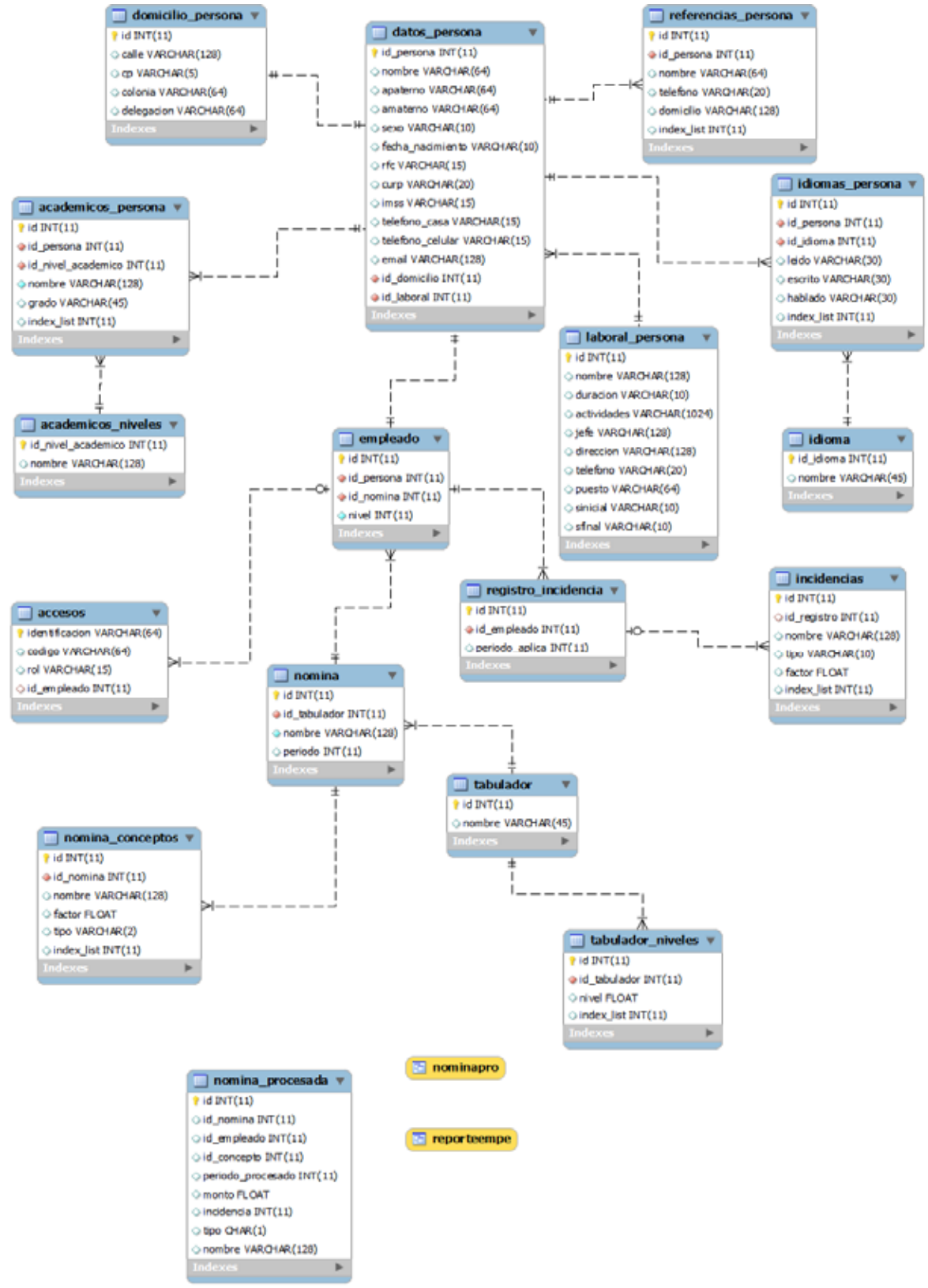


DIAGRAMA ENTIDAD
RELACION



Diccionario de datos

La información de las tablas es mostrada a continuación.

datos_persona

Contiene la información relacionada con el curriculum de la persona.

Relaciones:

- Uno a uno con domicilio_persona
- Uno a muchos con referencias_persona
- Uno a muchos con académicos_persona
- Uno a muchos con idiomas_persona
- Uno a uno con laboral_persona

domicilio_persona

Contiene la información sobre el domicilio de la persona.

Relaciones:

- Uno a uno con datos_persona

referencias_persona

Contiene la información de las referencias personales de la persona.

Relaciones:

- Muchos a uno con datos_persona

idiomas_persona

Contiene información de los idiomas que habla la persona.

Relaciones:

- Muchos a uno con datos_persona

academicos_persona

Contiene la información de los estudios relacionados con la persona.

Relaciones

- Muchos a Uno con datos_persona

laboral_persona

Contiene la información de la experiencia laboral de la persona.

Relaciones

- Uno a Uno con datos_persona

academicos_niveles

Es un catálogo de los niveles académicos existentes así como cursos y diplomados.

Relaciones

- Uno a Muchos con academicos_persona

idioma

Catálogo de idiomas

Relaciones

- Uno a Muchos con idiomas_persona

Empleado

Contiene información de las personas que han sido registradas como empleados

Relaciones

- Muchos a uno con nomina
- Uno a uno con datos_persona
- Uno a uno con accesos
- Uno a Muchos con registro_incidencia

Accesos

Contiene información de los empleados que tienen acceso al sistema junto con el rol.

Relaciones

- Uno a uno con empleado

registro_incidencia

Contiene la información del registro de incidencia que se le ha registrado al empleado cuando se va a procesar una nómina.

Relaciones

- Muchos a uno con empleado.
- Uno a Muchos con incidencias.

incidencias

Contiene la información detallada del registro de incidencia registrado al empleado, en esta tabla se muestran conceptos que se registraron como incidencias.

Relaciones

- Mucho a uno con registro_incidencia

nomina

Contiene información de las nóminas registradas en el sistema

Relaciones

- Uno a muchos con empleado
- Uno a muchos con nomina_conceptos

- Muchos a uno con tabulador

nomina_conceptos

Contiene la información de los conceptos relacionados a una nómina

Relaciones

- Muchos a uno con nomina

tabulador

Contiene la información de los tabuladores salariales.

Relaciones

- Uno a muchos con nomina
- Uno a Muchos con tabulador_niveles

tabulador_niveles

Contiene los niveles salariales del tabulador.

Relaciones

- Muchos a uno con tabulador.

nomina_procesada

Tabla para guardar información al procesar una nómina, esta tabla la usa el stored procedure procesar_nomina

Relaciones

- Por transitividad, debido a como está programado el stored procedure, se mantienen las relaciones de las tablas de las que depende nomina(id_nomina), empleado (id_empleado), nomina_conceptos(id_concepto)

nominapro y reporteempe

Son vistas y son utilizadas para generar los reportes.

Equema físico

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

DROP SCHEMA IF EXISTS `ptnominapymes` ;
CREATE SCHEMA IF NOT EXISTS `ptnominapymes` DEFAULT CHARACTER SET utf8
COLLATE utf8_spanish_ci ;

-- -----------------------------------------------------
-- Table `ptnominapymes`.`academicos_niveles`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`academicos_niveles` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`academicos_niveles` (
  `id_nivel_academico` INT(11) NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(128) NULL DEFAULT NULL ,
  PRIMARY KEY (`id_nivel_academico`) )
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`domicilio_persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`domicilio_persona` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`domicilio_persona` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `calle` VARCHAR(128) NULL DEFAULT NULL ,
  `cp` VARCHAR(5) NULL DEFAULT NULL ,
  `colonia` VARCHAR(64) NULL DEFAULT NULL ,
  `delegacion` VARCHAR(64) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 93
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `ptnominapymes`.`laboral_persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`laboral_persona` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`laboral_persona` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(128) NULL DEFAULT NULL ,
  `duracion` VARCHAR(64) NULL DEFAULT NULL ,
  `actividades` VARCHAR(1024) NULL DEFAULT NULL ,
  `jefe` VARCHAR(128) NULL DEFAULT NULL ,
  `direccion` VARCHAR(128) NULL DEFAULT NULL ,
  `telefono` VARCHAR(20) NULL DEFAULT NULL ,
  `puesto` VARCHAR(64) NULL DEFAULT NULL ,
  `sinicial` VARCHAR(10) NULL DEFAULT NULL ,
  `sfinal` VARCHAR(10) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 64
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`datos_persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`datos_persona` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`datos_persona` (
  `id_persona` INT(11) NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(64) NULL DEFAULT NULL ,
  `apaterno` VARCHAR(64) NULL DEFAULT NULL ,
  `amaterno` VARCHAR(64) NULL DEFAULT NULL ,
  `sexo` VARCHAR(10) NULL DEFAULT NULL ,
  `fecha_nacimiento` VARCHAR(10) NULL DEFAULT NULL ,
  `rfc` VARCHAR(15) NULL DEFAULT NULL ,
  `curp` VARCHAR(20) NULL DEFAULT NULL ,
  `imss` VARCHAR(15) NULL DEFAULT NULL ,
  `telefono_casa` VARCHAR(15) NULL DEFAULT NULL ,
  `telefono_celular` VARCHAR(15) NULL DEFAULT NULL ,
  `email` VARCHAR(128) NULL DEFAULT NULL ,
  `id_domicilio` INT(11) NOT NULL ,
  `id_laboral` INT(11) NOT NULL ,
  PRIMARY KEY (`id_persona`) ,
  INDEX `domicilio_persona` (`id_domicilio` ASC) ,
  INDEX `laboral_persona` (`id_laboral` ASC) ,
  CONSTRAINT `domicilio_persona`
    FOREIGN KEY (`id_domicilio` )
    REFERENCES `ptnominapymes`.`domicilio_persona` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `laboral_persona`
    FOREIGN KEY (`id_laboral` )
    REFERENCES `ptnominapymes`.`laboral_persona` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
```

```
AUTO_INCREMENT = 112
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`academicos_persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`academicos_persona` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`academicos_persona` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_persona` INT(11) NOT NULL ,
  `id_nivel_academico` INT(11) NOT NULL ,
  `nombre` VARCHAR(128) NOT NULL ,
  `grado` VARCHAR(45) NULL DEFAULT NULL ,
  `index_list` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_academicos_persona_datos_persona1` (`id_persona` ASC) ,
  INDEX `fk_academicos_persona_academicos_niveles1` (`id_nivel_academico`
ASC) ,
  CONSTRAINT `fk_academicos_persona_academicos_niveles1`
    FOREIGN KEY (`id_nivel_academico` )
    REFERENCES `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`
)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_academicos_persona_datos_persona1`
    FOREIGN KEY (`id_persona` )
    REFERENCES `ptnominapymes`.`datos_persona` (`id_persona` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 105
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`tabulador`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`tabulador` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`tabulador` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(45) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 10
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`nomina`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`nomina` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`nomina` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
```

```
  `id_tabulador` INT(11) NOT NULL ,
  `nombre` VARCHAR(128) NOT NULL ,
  `periodo` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_tabulador` (`id_tabulador` ASC) ,
  CONSTRAINT `fk_tabulador`
    FOREIGN KEY (`id_tabulador` )
    REFERENCES `ptnominapymes`.`tabulador` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 10
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`empleado`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`empleado` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`empleado` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_persona` INT(11) NOT NULL DEFAULT '0' ,
  `id_nomina` INT(11) NOT NULL ,
  `nivel` INT(11) NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `datos_persona` (`id_persona` ASC) ,
  INDEX `fk_nomina` (`id_nomina` ASC) ,
  CONSTRAINT `datos_persona`
    FOREIGN KEY (`id_persona` )
    REFERENCES `ptnominapymes`.`datos_persona` (`id_persona` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_nomina`
    FOREIGN KEY (`id_nomina` )
    REFERENCES `ptnominapymes`.`nomina` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 9
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`accesos`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`accesos` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`accesos` (
  `identificacion` VARCHAR(64) NOT NULL ,
  `codigo` VARCHAR(64) NULL DEFAULT NULL ,
  `rol` VARCHAR(15) NULL DEFAULT NULL ,
  `id_empleado` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`identificacion`) ,
  INDEX `fk_empleado` (`id_empleado` ASC) ,
  CONSTRAINT `fk_empleado`
    FOREIGN KEY (`id_empleado` )
```

```
    REFERENCES `ptnominapymes`.`empleado` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`idioma`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`idioma` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`idioma` (
  `id_idioma` INT(11) NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(45) NULL DEFAULT NULL ,
  PRIMARY KEY (`id_idioma`) )
ENGINE = InnoDB
AUTO_INCREMENT = 7
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`idiomas_persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`idiomas_persona` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`idiomas_persona` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_persona` INT(11) NOT NULL ,
  `id_idioma` INT(11) NOT NULL DEFAULT '0' ,
  `leido` VARCHAR(30) NULL DEFAULT NULL ,
  `escrito` VARCHAR(30) NULL DEFAULT NULL ,
  `hablado` VARCHAR(30) NULL DEFAULT NULL ,
  `index_list` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_idiomas_persona_datos_persona1` (`id_persona` ASC) ,
  INDEX `fk_idioma_id` (`id_idioma` ASC) ,
  CONSTRAINT `fk_idioma_id`
    FOREIGN KEY (`id_idioma` )
    REFERENCES `ptnominapymes`.`idioma` (`id_idioma` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_idiomas_persona_datos_persona1`
    FOREIGN KEY (`id_persona` )
    REFERENCES `ptnominapymes`.`datos_persona` (`id_persona` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 128
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`registro_incidencia`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`registro_incidencia` ;
```

```
CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`registro_incidencia` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_empleado` INT(11) NOT NULL ,
  `periodo_aplica` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_empleado_id` (`id_empleado` ASC) ,
  CONSTRAINT `fk_empleado_id`
    FOREIGN KEY (`id_empleado` )
    REFERENCES `ptnominapymes`.`empleado` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 5
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`incidencias`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`incidencias` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`incidencias` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_registro` INT(11) NULL DEFAULT NULL ,
  `nombre` VARCHAR(128) NULL DEFAULT NULL ,
  `tipo` VARCHAR(10) NULL DEFAULT NULL ,
  `factor` FLOAT NULL DEFAULT NULL ,
  `index_list` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_registro_incidencia` (`id_registro` ASC) ,
  CONSTRAINT `fk_registro_incidencia`
    FOREIGN KEY (`id_registro` )
    REFERENCES `ptnominapymes`.`registro_incidencia` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 16
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`nomina_conceptos`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`nomina_conceptos` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`nomina_conceptos` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_nomina` INT(11) NOT NULL ,
  `nombre` VARCHAR(128) NULL DEFAULT NULL ,
  `factor` FLOAT NULL DEFAULT NULL ,
  `tipo` VARCHAR(2) NULL DEFAULT NULL ,
  `index_list` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_idnomina` (`id_nomina` ASC) ,
  CONSTRAINT `fk_idnomina`
    FOREIGN KEY (`id_nomina` )
    REFERENCES `ptnominapymes`.`nomina` (`id` )
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 35
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`nomina_procesada`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`nomina_procesada` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`nomina_procesada` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_nomina` INT(11) NULL DEFAULT NULL ,
  `id_empleado` INT(11) NULL DEFAULT NULL ,
  `id_concepto` INT(11) NULL DEFAULT NULL ,
  `periodo_procesado` INT(11) NULL DEFAULT NULL ,
  `monto` FLOAT NULL DEFAULT NULL ,
  `incidencia` INT(11) NULL DEFAULT '0' ,
  `tipo` CHAR(1) NULL DEFAULT NULL ,
  `nombre` VARCHAR(128) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 47445
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`referencias_persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`referencias_persona` ;

CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`referencias_persona` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_persona` INT(11) NOT NULL ,
  `nombre` VARCHAR(64) NULL DEFAULT NULL ,
  `telefono` VARCHAR(20) NULL DEFAULT NULL ,
  `domicilio` VARCHAR(128) NULL DEFAULT NULL ,
  `index_list` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_referencias_persona_datos_persona1` (`id_persona` ASC) ,
  INDEX `index_list_field` (`index_list` ASC) ,
  CONSTRAINT `fk_referencias_persona_datos_persona1`
    FOREIGN KEY (`id_persona` )
    REFERENCES `ptnominapymes`.`datos_persona` (`id_persona` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 47
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `ptnominapymes`.`tabulador_niveles`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ptnominapymes`.`tabulador_niveles` ;
```

```
CREATE  TABLE IF NOT EXISTS `ptnominapymes`.`tabulador_niveles` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `id_tabulador` INT(11) NOT NULL ,
  `nivel` FLOAT NULL DEFAULT NULL ,
  `index_list` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_tabulador_id` (`id_tabulador` ASC) ,
  CONSTRAINT `fk_tabulador_id`
    FOREIGN KEY (`id_tabulador` )
    REFERENCES `ptnominapymes`.`tabulador` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 30
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Placeholder table for view `ptnominapymes`.`nominapro`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ptnominapymes`.`nominapro` (`id_nomina` INT,
`periodo_procesado` INT, `id_empleado` INT, `nombreempleado` INT,
`apaterno` INT, `amaterno` INT, `rfc` INT, `curp` INT, `imss` INT,
`id_concepto` INT, `concepto` INT, `monto` INT, `tipo` INT);

-- -----------------------------------------------------
-- Placeholder table for view `ptnominapymes`.`reporteempe`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ptnominapymes`.`reporteempe` (`id_nomina`
INT, `id_empleado` INT, `id_persona` INT, `NombreEmpleado` INT,
`apaterno` INT, `amaterno` INT, `sexo` INT, `fecha_nacimiento` INT, `rfc`
INT, `curp` INT, `imss` INT, `telefono_casa` INT, `telefono_celular` INT,
`email` INT, `calle` INT, `cp` INT, `colonia` INT, `delegacion` INT,
`nombreidioma` INT, `index_list` INT, `id_idioma` INT, `leido` INT,
`escrito` INT, `hablado` INT, `id_nivel_academico` INT, `nombreescuela`
INT, `grado` INT, `nombreempresa` INT, `duracion` INT, `puesto` INT,
`actividades` INT);

-- -----------------------------------------------------
-- procedure procesar_nomina
-- -----------------------------------------------------
DELIMITER $$
USE `ptnominapymes`$$
DROP procedure IF EXISTS `ptnominapymes`.`procesar_nomina`$$


USE `ptnominapymes`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `procesar_nomina`(nomina INT,
per INT)
BEGIN

DECLARE empleado_id INT; -- para abrir el cursor de las incidencias
DECLARE empleado_nivel INT;

DECLARE id_concepto INT;
DECLARE tipo_concepto CHAR;
```

```
DECLARE factor_concepto FLOAT;
DECLARE monto FLOAT;
DECLARE nombre_concepto varchar(128);
DECLARE periodo_nomina INT;

DECLARE fin_de_rows INT DEFAULT 0;

DECLARE empleados CURSOR  FOR -- empleados asignados a la nÃ³mina
    SELECT a.id,b.nivel
    FROM empleado a,tabulador_niveles b, nomina c
    WHERE c.id_tabulador=b.id_tabulador and a.nivel=b.index_list and
a.id_nomina=nomina group by id,nivel;

DECLARE nomina_conceptos CURSOR FOR  -- conceptos asignados a la nÃ³mina
    SELECT id,round(factor,2)factor,tipo,nombre
    FROM nomina_conceptos
    WHERE id_nomina=nomina;

DECLARE incidencias CURSOR FOR
    SELECT b.id,round(factor,2)factor,tipo,nombre
    FROM registro_incidencia a, incidencias b
    WHERE a.id=b.id_registro AND a.id_empleado=empleado_id AND
a.periodo_aplica=per;-- el id dle empleado lo saco del cursor abierto

DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin_de_rows=1;     -- Handler
para iterar en los cursores


SELECT periodo INTO periodo_nomina FROM nomina WHERE id=nomina; --
periodo 7 - 15 - 30

    OPEN empleados; -- Outer cursor

        empe: REPEAT
            FETCH empleados INTO empleado_id,empleado_nivel;

            IF fin_de_rows=1 THEN
                LEAVE empe;
            END IF;




                OPEN nomina_conceptos; -- Primer Procesamiento, conceptos
de la nÃ³mina inner cursor

                        conceptos: REPEAT



                            FETCH nomina_conceptos INTO
id_concepto,factor_concepto,tipo_concepto,nombre_concepto;



                                IF fin_de_rows=1 THEN
                                     LEAVE conceptos;
```

```
                                END IF;
                        -- se procesa el concepto y se inserta el
registro
                        SET monto =
periodo_nomina*(factor_concepto*empleado_nivel);
                        INSERT INTO
nomina_procesada(id_nomina,id_empleado,id_concepto,periodo_procesado,mont
o,tipo,nombre)
                        VALUES
(nomina,empleado_id,id_concepto,per,monto,tipo_concepto,nombre_concepto);


                        UNTIL fin_de_rows
                        END REPEAT conceptos;
                SET fin_de_rows=0;
                CLOSE nomina_conceptos;




          OPEN incidencias;

                incide: REPEAT


                    FETCH incidencias  INTO
id_concepto,factor_concepto,tipo_concepto,nombre_concepto; -- La
incidencia es el total del factor con relacion al nivel del tabulador del
empleado
                    SET monto = (factor_concepto*empleado_nivel);

                      IF fin_de_rows=1 THEN
                        LEAVE incide;
                    END IF;


                    INSERT INTO
nomina_procesada(id_nomina,id_empleado,id_concepto,periodo_procesado,mont
o,tipo,incidencia,nombre)
                    VALUES
(nomina,empleado_id,id_concepto,per,monto,tipo_concepto,1,nombre_concepto
);

                UNTIL fin_de_rows
                END REPEAT incide;
                SET fin_de_rows=0;
             CLOSE incidencias;

            --

             --



         UNTIL fin_de_rows
```

```
        END repeat empe;
    CLOSE empleados;


END


$$
DELIMITER ;

-- ----------------------------------------------------
-- View `ptnominapymes`.`nominapro`
-- ----------------------------------------------------
DROP VIEW IF EXISTS `ptnominapymes`.`nominapro` ;
DROP TABLE IF EXISTS `ptnominapymes`.`nominapro`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `nominapro` AS select `b`.`id_nomina` AS
`id_nomina`,`c`.`periodo_procesado` AS `periodo_procesado`,`b`.`id` AS
`id_empleado`,`a`.`nombre` AS `nombreempleado`,`a`.`apaterno` AS
`apaterno`,`a`.`amaterno` AS `amaterno`,`a`.`rfc` AS `rfc`,`a`.`curp` AS
`curp`,`a`.`imss` AS `imss`,`c`.`id_concepto` AS
`id_concepto`,`c`.`nombre` AS `concepto`,`c`.`monto` AS
`monto`,`c`.`tipo` AS `tipo` from ((`datos_persona` `a` join `empleado`
`b`) join `nomina_procesada` `c`) where ((`a`.`id_persona` =
`b`.`id_persona`) and (`b`.`id_nomina` = `c`.`id_nomina`) and (`b`.`id` =
`c`.`id_empleado`)) order by `c`.`tipo`;

-- ----------------------------------------------------
-- View `ptnominapymes`.`reporteempe`
-- ----------------------------------------------------
DROP VIEW IF EXISTS `ptnominapymes`.`reporteempe` ;
DROP TABLE IF EXISTS `ptnominapymes`.`reporteempe`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `reporteempe` AS select `a`.`id_nomina` AS
`id_nomina`,`a`.`id` AS `id_empleado`,`a`.`id_persona` AS
`id_persona`,`b`.`nombre` AS `NombreEmpleado`,`b`.`apaterno` AS
`apaterno`,`b`.`amaterno` AS `amaterno`,`b`.`sexo` AS
`sexo`,`b`.`fecha_nacimiento` AS `fecha_nacimiento`,`b`.`rfc` AS
`rfc`,`b`.`curp` AS `curp`,`b`.`imss` AS `imss`,`b`.`telefono_casa` AS
`telefono_casa`,`b`.`telefono_celular` AS `telefono_celular`,`b`.`email`
AS `email`,`d`.`calle` AS `calle`,`d`.`cp` AS `cp`,`d`.`colonia` AS
`colonia`,`d`.`delegacion` AS `delegacion`,`h`.`nombre` AS
`nombreidioma`,`f`.`index_list` AS `index_list`,`f`.`id_idioma` AS
`id_idioma`,`f`.`leido` AS `leido`,`f`.`escrito` AS
`escrito`,`f`.`hablado` AS `hablado`,`g`.`id_nivel_academico` AS
`id_nivel_academico`,`g`.`nombre` AS `nombreescuela`,`g`.`grado` AS
`grado`,`l`.`nombre` AS `nombreempresa`,`l`.`duracion` AS
`duracion`,`l`.`puesto` AS `puesto`,`l`.`actividades` AS `actividades`
from ((((((`empleado` `a` join `datos_persona` `b`) join
`domicilio_persona` `d`) join `idiomas_persona` `f`) join
`academicos_persona` `g`) join `idioma` `h`) join `laboral_persona` `l`)
where ((`a`.`id_persona` = `b`.`id_persona`) and (`b`.`id_domicilio` =
`d`.`id`) and (`b`.`id_persona` = `f`.`id_persona`) and (`f`.`id_idioma`
= `h`.`id_idioma`) and (`b`.`id_persona` = `g`.`id_persona`) and
(`g`.`index_list` = 0) and (`b`.`id_laboral` = `l`.`id`) and
(`f`.`id_idioma` != -1));
```

```
;
CREATE USER `ptnominapyme` IDENTIFIED BY 'ptnominapymepass';

grant ALL on procedure `ptnominapymes`.`procesar_nomina` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`nominapro` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`reporteempe` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`academicos_niveles` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`academicos_persona` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`accesos` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`datos_persona` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`domicilio_persona` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`empleado` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`idioma` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`idiomas_persona` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`incidencias` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`laboral_persona` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`nomina` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`nomina_conceptos` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`nomina_procesada` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`referencias_persona` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`registro_incidencia` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`tabulador` to ptnominapyme;
grant ALL on TABLE `ptnominapymes`.`tabulador_niveles` to ptnominapyme;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

-- ----------------------------------------------------
-- Data for table `ptnominapymes`.`academicos_niveles`
-- ----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (1, 'Primaria');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (2, 'Secundaria');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (3, 'Vocacional');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (4, 'Preparatoria');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (5, 'Nivel Medio Superior');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (6, 'Licenciatura');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (7, 'Ingenieria');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (8, 'Posgrado');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (9, 'Maestria');
INSERT INTO `ptnominapymes`.`academicos_niveles` (`id_nivel_academico`,
`nombre`) VALUES (10, 'Doctorado');

COMMIT;

-- ----------------------------------------------------
-- Data for table `ptnominapymes`.`domicilio_persona`
```

```
-- ----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`domicilio_persona` (`id`, `calle`, `cp`,
`colonia`, `delegacion`) VALUES (1, 'Calle ', '11111', 'Colonia',
'DelegaciÃ³n');
INSERT INTO `ptnominapymes`.`domicilio_persona` (`id`, `calle`, `cp`,
`colonia`, `delegacion`) VALUES (2, 'Calle ', '11111', 'Colonia',
'DelegaciÃ³n');

COMMIT;

-- ----------------------------------------------------
-- Data for table `ptnominapymes`.`laboral_persona`
-- ----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`laboral_persona` (`id`, `nombre`,
`duracion`, `actividades`, `jefe`, `direccion`, `telefono`, `puesto`,
`sinicial`, `sfinal`) VALUES (1, 'Empresa', 'DuraciÃ³n', 'Actividades',
'Jefe', 'DirecciÃ³n', '5555555555', 'Puesto', '$00000,00', '$00000,00');
INSERT INTO `ptnominapymes`.`laboral_persona` (`id`, `nombre`,
`duracion`, `actividades`, `jefe`, `direccion`, `telefono`, `puesto`,
`sinicial`, `sfinal`) VALUES (2, 'Empresa', 'DuraciÃ³n', 'Actividades',
'Jefe', 'DirecciÃ³n', '5555555555', 'Puesto', '$00000,00', '$00000,00');

COMMIT;

-- ----------------------------------------------------
-- Data for table `ptnominapymes`.`datos_persona`
-- ----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`datos_persona` (`id_persona`, `nombre`,
`apaterno`, `amaterno`, `sexo`, `fecha_nacimiento`, `rfc`, `curp`,
`imss`, `telefono_casa`, `telefono_celular`, `email`, `id_domicilio`,
`id_laboral`) VALUES (1, 'Horacio', 'Iturbe', 'GarcÃa', 'Masculino',
'29/08/1984', 'RFCRFCRFC', 'CURPCURPCURPCURPCU', '30000000000',
'5555555555', '5555555555', 'algo@algo.com', 1, 1);
INSERT INTO `ptnominapymes`.`datos_persona` (`id_persona`, `nombre`,
`apaterno`, `amaterno`, `sexo`, `fecha_nacimiento`, `rfc`, `curp`,
`imss`, `telefono_casa`, `telefono_celular`, `email`, `id_domicilio`,
`id_laboral`) VALUES (2, 'Brenda', 'Iturbe', 'GarcÃa', 'Femenino',
'19/09/1986', 'RFCRFCRFC', 'CURPCURPCURPCURPCU', '30000000001',
'5555555555', '5555555555', 'algo@algo.com', 2, 2);

COMMIT;

-- ----------------------------------------------------
-- Data for table `ptnominapymes`.`academicos_persona`
-- ----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`academicos_persona` (`id`, `id_persona`,
`id_nivel_academico`, `nombre`, `grado`, `index_list`) VALUES (1, 1, 7,
'Uam Azcapotzalco', 'Ingeniero en ComputaciÃ³n', 0);
INSERT INTO `ptnominapymes`.`academicos_persona` (`id`, `id_persona`,
`id_nivel_academico`, `nombre`, `grado`, `index_list`) VALUES (2, 2, 6,
'Uam Azcapotzalco', 'Carrera oooooo', 0);

COMMIT;
```

```
-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`tabulador`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`tabulador` (`id`, `nombre`) VALUES (1,
'Tabulador 1');

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`nomina`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`nomina` (`id`, `id_tabulador`, `nombre`,
`periodo`) VALUES (1, 1, 'Nomina 1', 15);
INSERT INTO `ptnominapymes`.`nomina` (`id`, `id_tabulador`, `nombre`,
`periodo`) VALUES (2, 1, 'Nomina 2', 15);

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`empleado`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`empleado` (`id`, `id_persona`, `id_nomina`,
`nivel`) VALUES (1, 1, 1, 0);
INSERT INTO `ptnominapymes`.`empleado` (`id`, `id_persona`, `id_nomina`,
`nivel`) VALUES (2, 2, 2, 1);

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`accesos`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`accesos` (`identificacion`, `codigo`, `rol`,
`id_empleado`) VALUES ('admin', 'admin', 'administrador', 1);
INSERT INTO `ptnominapymes`.`accesos` (`identificacion`, `codigo`, `rol`,
`id_empleado`) VALUES ('operador', 'operador', 'operador', 1);
INSERT INTO `ptnominapymes`.`accesos` (`identificacion`, `codigo`, `rol`,
`id_empleado`) VALUES ('rh', 'rh', 'recursoshumanos', 1);
INSERT INTO `ptnominapymes`.`accesos` (`identificacion`, `codigo`, `rol`,
`id_empleado`) VALUES ('otro', 'otro', 'simple', 1);

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`idioma`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (1,
'InglÃ©s');
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (2,
'AlemÃ¡n');
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (3,
'Frances');
```

```
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (4,
'Ruso');
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (5,
'Italiano');
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (6,
'Portugues');
INSERT INTO `ptnominapymes`.`idioma` (`id_idioma`, `nombre`) VALUES (-1,
'Vacio');

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`idiomas_persona`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`idiomas_persona` (`id`, `id_persona`,
`id_idioma`, `leido`, `escrito`, `hablado`, `index_list`) VALUES (1, 1,
1, '10', '10', '10', 0);
INSERT INTO `ptnominapymes`.`idiomas_persona` (`id`, `id_persona`,
`id_idioma`, `leido`, `escrito`, `hablado`, `index_list`) VALUES (2, 1, -
1, NULL, NULL, NULL, 1);
INSERT INTO `ptnominapymes`.`idiomas_persona` (`id`, `id_persona`,
`id_idioma`, `leido`, `escrito`, `hablado`, `index_list`) VALUES (3, 1, -
1, NULL, NULL, NULL, 2);
INSERT INTO `ptnominapymes`.`idiomas_persona` (`id`, `id_persona`,
`id_idioma`, `leido`, `escrito`, `hablado`, `index_list`) VALUES (4, 2,
2, '10', '10', '10', 0);
INSERT INTO `ptnominapymes`.`idiomas_persona` (`id`, `id_persona`,
`id_idioma`, `leido`, `escrito`, `hablado`, `index_list`) VALUES (5, 2, -
1, NULL, NULL, NULL, 1);
INSERT INTO `ptnominapymes`.`idiomas_persona` (`id`, `id_persona`,
`id_idioma`, `leido`, `escrito`, `hablado`, `index_list`) VALUES (6, 2, -
1, NULL, NULL, NULL, 2);

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`nomina_conceptos`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (1, 1, 'Sueldo Base',
15, '+', 0);
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (2, 1, 'ISR', 0.3, '-',
1);
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (3, 1, 'Vales de
despensa', 3, '+', 2);
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (4, 1, 'Imss', 4.5, '-',
3);
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (5, 2, 'Sueldo Base',
15, '+', 0);
```

```
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (6, 2, 'ISR', 0.3, '-',
1);
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (7, 2, 'Vales de
Despensa', 4, '+', 2);
INSERT INTO `ptnominapymes`.`nomina_conceptos` (`id`, `id_nomina`,
`nombre`, `factor`, `tipo`, `index_list`) VALUES (8, 2, 'Imss', 1.2, '-',
3);

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`referencias_persona`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`referencias_persona` (`id`, `id_persona`,
`nombre`, `telefono`, `domicilio`, `index_list`) VALUES (1, 1, 'Nombre',
'1111111111', 'Domicilio', 0);
INSERT INTO `ptnominapymes`.`referencias_persona` (`id`, `id_persona`,
`nombre`, `telefono`, `domicilio`, `index_list`) VALUES (2, 1, 'Nombre
2', '2222222222', 'Domicilio 2', 1);
INSERT INTO `ptnominapymes`.`referencias_persona` (`id`, `id_persona`,
`nombre`, `telefono`, `domicilio`, `index_list`) VALUES (3, 2, 'Nombre',
'1111111111', 'Domicilio', 0);
INSERT INTO `ptnominapymes`.`referencias_persona` (`id`, `id_persona`,
`nombre`, `telefono`, `domicilio`, `index_list`) VALUES (4, 2, 'Nombre
2', '2222222222', 'Domicilio 2', 1);

COMMIT;

-- -----------------------------------------------------
-- Data for table `ptnominapymes`.`tabulador_niveles`
-- -----------------------------------------------------
SET AUTOCOMMIT=0;
INSERT INTO `ptnominapymes`.`tabulador_niveles` (`id`, `id_tabulador`,
`nivel`, `index_list`) VALUES (1, 1, 350, 0);
INSERT INTO `ptnominapymes`.`tabulador_niveles` (`id`, `id_tabulador`,
`nivel`, `index_list`) VALUES (2, 1, 450, 1);
INSERT INTO `ptnominapymes`.`tabulador_niveles` (`id`, `id_tabulador`,
`nivel`, `index_list`) VALUES (3, 1, 550, 2);

COMMIT;
```

CÓDIGO FUENTE DE LA APLICACIÓN

```
package proyecto.terminal.horacio.customResult;



import java.io.PrintWriter;
```

```
import org.apache.struts2.ServletActionContext;


import com.opensymphony.xwork2.ActionInvocation;

import com.opensymphony.xwork2.Result;

import com.opensymphony.xwork2.util.ValueStack;

import com.thoughtworks.xstream.XStream;

import com.thoughtworks.xstream.io.json.JettisonMappedXmlDriver;

/**

 * Define un resultado personalizado para ser utilizado como resultType en Struts, esta

 * clase formatea una lista de objetos en representación JSON para que pueda ser manipulada en javascript

 *

 *

 * @author Horacio Iturbe

 *

 */

public class EventosJSON  implements Result{


	public static final String DEFAULT_PARAM = "classAlias";


	String classAlias;


	public String getClassAlias() {

		return classAlias;

	}
```

```
	public void setClassAlias(String classAlias) {
		this.classAlias = classAlias;
	}

	/**
	 * Realiza la conversion del objeto a representación JSON
	 */
	public void execute(ActionInvocation invocation) throws Exception {

		ServletActionContext.getResponse().setContentType("text/plain;charset=iso-8859-
1");
		PrintWriter responseStream = ServletActionContext.getResponse().getWriter();

		ValueStack valueStack = invocation.getStack();
		Object jsonModel = valueStack.findValue("jsonModel");

		XStream xstream = new XStream(new JettisonMappedXmlDriver());



		if (classAlias == null) {
			classAlias = "object";
		}

		xstream.alias(classAlias, jsonModel.getClass());
		responseStream.println(xstream.toXML(jsonModel));
```

```
        }



}
package proyecto.terminal.horacio.datos;

import java.io.Serializable;


import javax.persistence.*;


import static javax.persistence.GenerationType.IDENTITY;
/**
 * Tabla asociada <b>academicos_persona</b>
 * guarda una relación ManyToOne con la tabla de datos_personales
 *
 * @author Horacio Iturbe
 * @see DatosPersonalesDTO
 * @see NivelesAcademicosDTO
 *
 */
@Entity
@Table (name="ACADEMICOS_PERSONA")
public class AcademicosPersonaDTO implements Serializable{



        /**
         * Identificador del registro ingresado a la base de datos
```

```
 */
private int id;

/**
 * identificador obtenido del catálogo NivelesAcademicosDTO
 */
private int nivel_academico;
/**
 * Nombre de la institución donde la persona curso el niveñ
 */
private String nombre;
/**
 * Grado obtenido
 */
private String grado;

/**
 * Identificador de la persona
 */
private int id_persona;

/**
 * Debido a que este DTO representa el Many side de la relación,
 * el index_list se incremenate de 0,..,n-1
 */
private int index_list;
```

```
@Id
@Column(name="ID") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}


@Column (name="ID_NIVEL_ACADEMICO")
public int getNivel_academico() {
	return nivel_academico;
}

public void setNivel_academico(int nivelAcademico) {
	nivel_academico = nivelAcademico;
}

@Column(name="NOMBRE")
public String getNombre() {
	return nombre;
}
```

```
public void setNombre(String nombre) {
	this.nombre = nombre;
}
@Column(name="GRADO")
public String getGrado() {
	return grado;
}
public void setGrado(String grado) {
	this.grado = grado;
}

@Column(name="INDEX_LIST",insertable=false,updatable=false)
public int getIndex_list() {
	return index_list;
}

public void setIndex_list(int indexList) {
	index_list = indexList;
}

@Column(name="ID_PERSONA",insertable=false,updatable=false)
public int getId_persona() {
	return id_persona;
}

public void setId_persona(int idPersona) {
```

```
                id_persona = idPersona;
        }




}

package proyecto.terminal.horacio.datos;
import java.io.Serializable;

import javax.persistence.*;

import static javax.persistence.GenerationType.IDENTITY;
/**
 * Tabla asociada <b>academicos_persona</b>
 * guarda una relación ManyToOne con la tabla de datos_personales
 *
 * @author Horacio Iturbe
 * @see DatosPersonalesDTO
 * @see NivelesAcademicosDTO
 *
 */
@Entity
@Table (name="ACADEMICOS_PERSONA")
public class AcademicosPersonaDTO implements Serializable{
```

```
/**
 * Identificador del registro ingresado a la base de datos
 */
private int id;

/**
 * identificador obtenido del catálogo NivelesAcademicosDTO
 */
private int nivel_academico;
/**
 * Nombre de la institución donde la persona curso el niveñ
 */
private String nombre;
/**
 * Grado obtenido
 */
private String grado;

/**
 * Identificador de la persona
 */
private int id_persona;

/**
 * Debido a que este DTO representa el Many side de la relación,
```

```
 * el index_list se incremenate de 0,..,n-1
 */
private int index_list;



@Id
@Column(name="ID") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}



@Column (name="ID_NIVEL_ACADEMICO")
public int getNivel_academico() {
	return nivel_academico;
}

public void setNivel_academico(int nivelAcademico) {
	nivel_academico = nivelAcademico;
}

@Column(name="NOMBRE")
```

```
public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
@Column(name="GRADO")
public String getGrado() {
	return grado;
}
public void setGrado(String grado) {
	this.grado = grado;
}

@Column(name="INDEX_LIST",insertable=false,updatable=false)
public int getIndex_list() {
	return index_list;
}

public void setIndex_list(int indexList) {
	index_list = indexList;
}

@Column(name="ID_PERSONA",insertable=false,updatable=false)
public int getId_persona() {
	return id_persona;
```

```
	}

	public void setId_persona(int idPersona) {
		id_persona = idPersona;
	}



}
```

```
package proyecto.terminal.horacio.datos;

/**
 * Dto sin tabla asociada en la base de datos
 * este DTO se usa para realizar una busqueda de personas o empleados
 * @author Horacio
 *
 */
public class CamposSuchenDTO {

	/**
	 * Criterio para filtrar
	 */
	private String campo;
	/**
	 * Valor del criterio para filtrar
	 */
```

```
	private String valor;

	public String getCampo() {
		return campo;
	}
	public void setCampo(String campo) {
		this.campo = campo;
	}
	public String getValor() {
		return valor;
	}
	public void setValor(String valor) {
		this.valor = valor;
	}

}
```

```
package proyecto.terminal.horacio.datos;

import java.io.Serializable;
import java.util.List;

import javax.persistence.*;
import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.IndexColumn;
import static javax.persistence.GenerationType.IDENTITY;
/**
```

```
 * Tabla asociada <b> datos_persona</b>
 * @author Horacio Iturbe
 * @see ReferenciasPersonaDTO
 * @see AcademicosPersonaDTO
 * @see PersonaIdiomasDTO
 * @see LaboralPersonaDTO
 * @see DomicilioPersonaDTO
 */
@Entity
@Table (name="DATOS_PERSONA")

public class DatosPersonalesDTO implements Serializable{

        /**
         * Identificador de la persona
         */
        private int id;

        /**
         * Nombre
         */
        private String nombre;
        /**
         * Apellido paterno
         */
        private String apaterno;
```

```
/**
 * Apellido materno
 */
private String amaterno;
/**
 * Sexo
 */
private String sexo;
/**
 * Fecha de nacimiento
 */
private String fecha_nacimiento;
/**
 * Rfc
 */
private String rfc;
/**
 * Curp
 */
private String curp;
/**
 * Número de seguridad social NSS
 */
private String imss;
/**
 * Teléfono de casa
```

```
 */
private String telefono_casa;
/**
 * Teléfono celular
 */
private String telefono_celular;
/**
 * Correo electrónico
 */
private String email;



/**
 * Domicilio de la persona, guarda una relación OneToOne
 */
private DomicilioPersonaDTO dom;
/**
 * Referencias Personales, guarda una relación OneToMany
 */
private List<ReferenciasPersonaDTO> ref; //OneToMany
/**
 * Estudios realizados, guarda una relación OneToMany
 */
private List<AcademicosPersonaDTO> academicos;//OneToMany


/**
```

```
 * Idiomas, guarda una relación OneToMany
 */
private List<PersonaIdiomasDTO> idioma;//OneToMany
/**
 * Experiencia Laboral, guarda una relación OneToOne
 */
private LaboralPersonaDTO labora; //OneToOne


@Id
@Column (name="ID_PERSONA") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
@Column (name="NOMBRE")
public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
public String getApaterno() {
	return apaterno;
}
```

```
public void setApaterno(String apaterno) {
        this.apaterno = apaterno;
}
public String getAmaterno() {
        return amaterno;
}
public void setAmaterno(String amaterno) {
        this.amaterno = amaterno;
}
public String getSexo() {
        return sexo;
}
public void setSexo(String sexo) {
        this.sexo = sexo;
}
public String getFecha_nacimiento() {
        return fecha_nacimiento;
}
public void setFecha_nacimiento(String fechaNacimiento) {
        fecha_nacimiento = fechaNacimiento;
}
public String getRfc() {
        return rfc;
}
public void setRfc(String rfc) {
        this.rfc = rfc;
```

```
}
public String getCurp() {
	return curp;
}
public void setCurp(String curp) {
	this.curp = curp;
}
@Column(name="IMSS",updatable=false)
public String getImss() {
	return imss;
}
public void setImss(String imss) {
	this.imss = imss;
}
public String getTelefono_casa() {
	return telefono_casa;
}
public void setTelefono_casa(String telefonoCasa) {
	telefono_casa = telefonoCasa;
}
public String getTelefono_celular() {
	return telefono_celular;
}
public void setTelefono_celular(String telefonoCelular) {
	telefono_celular = telefonoCelular;
}
```

```
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}



	@OneToOne(targetEntity=DomicilioPersonaDTO.class,cascade=CascadeType.ALL)
	@JoinColumn(name="ID_DOMICILIO",nullable=false)
	public DomicilioPersonaDTO getDom() {
		return dom;
	}
	public void setDom(DomicilioPersonaDTO dom) {
		this.dom = dom;
	}



	@OneToMany(targetEntity=ReferenciasPersonaDTO.class,fetch=FetchType.EAGER,cascade=CascadeType.ALL)
	@JoinColumn(name="ID_PERSONA",nullable=false)
	@IndexColumn(name="index_list")
	public List<ReferenciasPersonaDTO> getRef() {
		return ref;
	}
	public void setRef(List<ReferenciasPersonaDTO> ref) {
```

```
		this.ref = ref;
	}

	@OneToMany(targetEntity=AcademicosPersonaDTO.class,fetch=FetchType.EAGER,cascade =CascadeType.ALL)
	@Cascade(value = org.hibernate.annotations.CascadeType.DELETE_ORPHAN)
	@JoinColumn(name="ID_PERSONA",nullable=false)
	@IndexColumn(name="index_list")
	public List<AcademicosPersonaDTO> getAcademicos() {
		return academicos;
	}
	public void setAcademicos(List<AcademicosPersonaDTO> academicos) {
		this.academicos = academicos;
	}


	@OneToMany(targetEntity=PersonaIdiomasDTO.class,fetch=FetchType.EAGER,cascade=CascadeType.ALL)
	@JoinColumn(name="ID_PERSONA",nullable=false)
	@IndexColumn(name="index_list")
	public List<PersonaIdiomasDTO> getIdioma() {
		return idioma;
	}
	public void setIdioma(List<PersonaIdiomasDTO> idioma) {
		this.idioma = idioma;
	}
```

```
	@OneToOne(targetEntity=LaboralPersonaDTO.class,cascade=CascadeType.ALL)
	@JoinColumn(name="ID_LABORAL",nullable=false)
	public LaboralPersonaDTO getLabora() {
		return labora;
	}
	public void setLabora(LaboralPersonaDTO labora) {
		this.labora = labora;
	}
	/**
	 * El total de estudios registrados por la persona
	 * @return int
	 */
	@Transient
	public int getTamAca()
	{
		return academicos.size();
	}

}
package proyecto.terminal.horacio.datos;

import java.io.Serializable;

import javax.persistence.*;
```

```
import static javax.persistence.GenerationType.IDENTITY;

/**

 * Tabla asociada <b>domicilio_persona</b>, guarda los registros relacionados con la dirección de una persona,

 * guarda una relación OneToOne con los datos personales

 * @author Horacio Iturbe

 * @see DatosPersonalesDTO

 *

 */

@Entity

@Table (name="DOMICILIO_PERSONA")

public class DomicilioPersonaDTO implements Serializable{


	/**

	 * Identificador del domicilio registrado

	 */

	private int id;

	/**

	 * Calle

	 */

	private String calle;

	/**

	 *Código postal

	 */

	private String cp;

	/**
```

```
 * Colonia
 */
private String colonia;
/**
 * Delegación
 */
private String delegacion;

@Id
@Column(name = "ID")  @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}

public String getCalle() {
	return calle;
}
public void setCalle(String calle) {
	this.calle = calle;
}
public String getCp() {
	return cp;
```

```
}
public void setCp(String cp) {
	this.cp = cp;
}
public String getColonia() {
	return colonia;
}
public void setColonia(String colonia) {
	this.colonia = colonia;
}
public String getDelegacion() {
	return delegacion;
}
public void setDelegacion(String delegacion) {
	this.delegacion = delegacion;
}
```

```
} package proyecto.terminal.horacio.datos;


import java.io.Serializable;

import javax.persistence.*;

import static javax.persistence.GenerationType.IDENTITY;

/**

 * Tabla asociada <b>empleado</b>, guarda la información referente al empleado

 * @author Horacio Iturbe

 * @see DatosPersonalesDTO

 * @see NominaDTO

 *

 */

@Entity

@Table(name="empleado")

public class EmpleadoDTO implements Serializable{


	/**

	 * Identificador del empleados

	 */

	private int id;

	/**

	 * Identificador de la persona en la tabla datos_personales

	 */

	private int id_persona;

	/**

	 * Identificador de la nómina en la tabla nomina
```

```
 */
private int id_nomina;
/**
 * Nivel que tendrá el empleado, el salario diario que tendrá
 *
 */
private int nivel;
/**
 * Referencia OneToOne a DatosPersonalesDTO
 */
private DatosPersonalesDTO persona;
/**
 * Referencia OneToOne a NominaDTO
 */
private NominaDTO nomina;

@Id
@Column (name="id") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
@Column(name="nivel")
public int getNivel() {
```

```
		return nivel;
}
public void setNivel(int nivel) {
		this.nivel = nivel;
}

@OneToOne(targetEntity=DatosPersonalesDTO.class,cascade=CascadeType.ALL)
@JoinColumn(name="id_persona",insertable=false,updatable=false)
public DatosPersonalesDTO getPersona() {
		return persona;
}
public void setPersona(DatosPersonalesDTO persona) {
		this.persona = persona;
}

@OneToOne(targetEntity=NominaDTO.class,cascade=CascadeType.ALL)
@JoinColumn(name="id_nomina",insertable=false,updatable=false)
public NominaDTO getNomina() {
		return nomina;
}
public void setNomina(NominaDTO nomina) {
		this.nomina = nomina;
}
@Column(name="id_persona",updatable=false)
public int getId_persona() {
		return id_persona;
```

```
	}
	public void setId_persona(int idPersona) {
		id_persona = idPersona;
	}
	@Column(name="id_nomina")
	public int getId_nomina() {
		return id_nomina;
	}
	public void setId_nomina(int idNomina) {
		id_nomina = idNomina;
	}



} package proyecto.terminal.horacio.datos;

import java.io.Serializable;
import javax.persistence.*;

/**
 * Tabla asociada <b>idioma</b>, catálogo de los idiomas disponibles en el sistema
 * @author Horacio
 *
 */
@Entity
@Table (name="IDIOMA")
public class IdiomasDTO implements Serializable{
```

```
/**
 * Identificador del idioma
 */
private int id;
/**
 * Nombre del idioma
 */
private String nombre;


@Id
@Column(name="ID_IDIOMA")
public int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
@Column(name="NOMBRE")
public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
```

```
} package proyecto.terminal.horacio.datos;


import java.io.Serializable;

import javax.persistence.*;

import static javax.persistence.GenerationType.IDENTITY;


/**

 * Tabla asociada <b>incidencias</b>, guarda una relación ManyToOne con
RegistroIncidenciaDTO,

 * @author Horacio

 * @see RegistroIncidenciaDTO

 *

 */

@Entity

@Table(name="incidencias")


public class IncidenciaDTO implements Serializable{


	/**

	 * Identificador de la incidencia

	 */

	private int id;

	/**

	 *  del concepto que da lugar a la incidencia

	 */

	private String nombre;
```

```
	/**
	 *  si es percepción o deducción
	 */
	private String tipo;
	/**
	 * Factor, el número de unidades que tendra que descontarse tomando en cuante el salario diario
	 *
	 */

	private float factor;
	/**
	 * Por ser el Many side de la relación, este campo se usa para control con Hibernate
	 */
	private int index_list;



	@Id
	@Column (name="id") @GeneratedValue(strategy=IDENTITY)
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
```

```
@Column(name="nombre")
public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}

@Column(name="tipo")
public String getTipo() {
	return tipo;
}
public void setTipo(String tipo) {
	this.tipo = tipo;
}

@Column(name="factor")
public float getFactor() {
	return factor;
}
public void setFactor(float factor) {
	this.factor = factor;
}

@Column(name="index_list",insertable=false,updatable=false)
public int getIndex_list() {
```

```
		return index_list;
	}
	public void setIndex_list(int indexList) {
		index_list = indexList;
	}
```

```
} package proyecto.terminal.horacio.datos;

import java.io.Serializable;

import javax.persistence.*;
import static javax.persistence.GenerationType.IDENTITY;

/**
 * Tabla asociada <b>laboral_persona</b>, guarda una relación OneToOne con DatosPersonalesDTO,
 * guarda la experiencia laboral de una persona
 * @author Horacio Iturbe
 * @see DatosPersonalesDTO
 *
 */
@Entity
@Table (name="LABORAL_PERSONA")
```

```
public class LaboralPersonaDTO implements Serializable{

	/**
	 * Identificador de la experiencia laboral
	 */
	private int id;
	/**
	 * Nombre de la empresa
	 */
	private String nombre;
	/**
	 * El tiempo de estancia
	 */
	private String duracion;
	/**
	 * Puesto desempeñado
	 */
	private String actividades;
	/**
	 * Nombre del jefe inmediato
	 */
	private String jefe;
	/**
	 * Domicilio de la empresa
	 */
```

```
private String direccion;
/**
 * Número telefónico
 */
private String telefono;
/**
 * Puesto desarrollado
 */
private String puesto;
/**
 * Salario inicial
 */
private String sinicial;
/**
 * Salario Final
 */
private String sfinal;

@Id
@Column(name="ID") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
```

```
}

public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
public String getDuracion() {
	return duracion;
}
public void setDuracion(String duracion) {
	this.duracion = duracion;
}
public String getActividades() {
	return actividades;
}
public void setActividades(String actividades) {
	this.actividades = actividades;
}
public String getJefe() {
	return jefe;
}
public void setJefe(String jefe) {
	this.jefe = jefe;
}
```

```
public String getDireccion() {
	return direccion;
}
public void setDireccion(String direccion) {
	this.direccion = direccion;
}
public String getTelefono() {
	return telefono;
}
public void setTelefono(String telefono) {
	this.telefono = telefono;
}
public String getPuesto() {
	return puesto;
}
public void setPuesto(String puesto) {
	this.puesto = puesto;
}
public String getSinicial() {
	return sinicial;
}
public void setSinicial(String sinicial) {
	this.sinicial = sinicial;
}
public String getSfinal() {
	return sfinal;
```

```
	}
	public void setSfinal(String sfinal) {
		this.sfinal = sfinal;
	}
```

} package proyecto.terminal.horacio.datos;

```
import java.io.Serializable;

import javax.persistence.*;
import static javax.persistence.GenerationType.IDENTITY;
/**
 * Tabla asociada <b>academicos_niveles</b>, catálogo de los niveles académicos disponibles en el sistema
 * @author Horacio Iturbe
 *
 */
@Entity
@Table (name="ACADEMICOS_NIVELES")
public class NivelesAcademicosDTO implements Serializable{

	/**
	 * Identificador del nivel
	 */
	private int id;
```

```
        /**
         * Nombre del nivel,
         */
        private String nombre;
        @Id
        @Column (name="ID_NIVEL_ACADEMICO") @GeneratedValue(strategy=IDENTITY)
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        @Column(name="NOMBRE")
        public String getNombre() {
                return nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
```

```
} package proyecto.terminal.horacio.datos;

import java.io.Serializable;
import javax.persistence.*;
```

```java
import static javax.persistence.GenerationType.IDENTITY;

/**

 * Tabla asociada <b> nomina_conceptos</b>, guarda una relación ManyToOne con NominaDTO

 * @author Horacio Iturbe

 * @see NominaDTO

 *

 */

@Entity

@Table(name="nomina_conceptos")

public class NominaConceptosDTO implements Serializable {


	/**

	 * Identificador del concepto

	 */

	private int id;

	/**

	 * nombre del concepto

	 */

	private String nombre;

	/**

	 * Factor relacionado con el salario diario

	 */

	private float factor;

	/**

	 * Percepción o deducción

	 */

	private String tipo;
```

```
	/**
	 * Por ser el Many side de la relación este campo se usa para
control con hibernate
	 */
	private int index_list;


	@Id
	@Column (name="id") @GeneratedValue(strategy=IDENTITY)
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	@Column(name="nombre")
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}


	@Column (name="factor")
	public float getFactor() {
		return factor;
	}
	public void setFactor(float factor) {
		this.factor = factor;
	}
```

```
	@Column(name="tipo")
	public String getTipo() {
		return tipo;
	}
	public void setTipo(String tipo) {
		this.tipo = tipo;
	}
	@Column(name="index_list",insertable=false,updatable=false)
	public int getIndex_list() {
		return index_list;
	}
	public void setIndex_list(int indexList) {
		index_list = indexList;
	}


}
```

```
package proyecto.terminal.horacio.datos;


import java.io.Serializable;
import java.util.List;
import static javax.persistence.GenerationType.IDENTITY;
import javax.persistence.*;
import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.IndexColumn;
/**
 * Tabla asociada <b>nomina</b>, guarda una relación OneToMany con NominaConceptosDTO y una OneToOne con TabuladorDTO,
```

```
 * mantiene la información de la nómina registrada en el sistema
 * @author Horacio Iturbe
 * @see NominaConceptosDTO
 * @see TabuladorDTO
 */
@Entity
@Table(name="nomina")
public class NominaDTO implements Serializable{

	/**
	 * Identificador de la nómina
	 */
	private int id;
	/**
	 * Identificador del tabulador asociado a la nómina
	 */
	private int id_tabulador;
	/**
	 * Nombre asignado a la nómina
	 */
	private String nombre;
	/**
	 * Que periodicidad tiene, quincenal, semanal, mensual
	 */
	private int periodo;
	/**
```

```
 * Relación OneToMany
 */
private List<NominaConceptosDTO> concepto;
/**
 * Relación OneToOne
 */
private TabuladorDTO tabulador;


@Id
@Column(name="id") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
@Column(name="id_tabulador")
public int getId_tabulador() {
	return id_tabulador;
}
public void setId_tabulador(int idTabulador) {
	id_tabulador = idTabulador;
}


@Column(name="nombre")
public String getNombre() {
```

```
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	@OneToMany(targetEntity=NominaConceptosDTO.class,fetch=FetchType.EAGER,cascade=
CascadeType.ALL)
	@Cascade(value=org.hibernate.annotations.CascadeType.DELETE_ORPHAN)
	@JoinColumn(name="id_nomina",nullable=false)
	@IndexColumn(name="index_list")
	public List<NominaConceptosDTO> getConcepto() {
		return concepto;
	}
	public void setConcepto(List<NominaConceptosDTO> concepto) {
		this.concepto = concepto;
	}
	@Column(name="periodo")
	public int getPeriodo() {
		return periodo;
	}
	public void setPeriodo(int periodo) {
		this.periodo = periodo;
	}

	@OneToOne(targetEntity=TabuladorDTO.class,cascade=CascadeType.ALL)
```

```
	@JoinColumn(name="id_tabulador",insertable=false,updatable=false)
	public TabuladorDTO getTabulador() {
		return tabulador;
	}
	public void setTabulador(TabuladorDTO tabulador) {
		this.tabulador = tabulador;
	}
	@Transient
	public int getConceptosSize()
	{
		return concepto.size();
	}
```

```
} package proyecto.terminal.horacio.datos;

import java.io.Serializable;
import javax.persistence.*;
import static javax.persistence.GenerationType.IDENTITY;

/**
 * Vista asociada <b>nominapro</b>, usada solo para generar los reportes de la nómina procesada
 * @author Horacio Iturbe
 *
```

```
 */
@Entity
@Table(name="nominapro")
public class NominaProcesadaDTO implements Serializable{



	private int id_nomina;
	private int periodo_procesado;
	private int id_empleado;
	private String nombreempleado;
	private String apaterno;
	private String amaterno;
	private String rfc;
	private String curp;
	private String imss;
	private int id_concepto;
	private String concepto;
	private float monto;
	private String tipo;
	private NominaDTO nomina;



	@Column(name="concepto")
	public String getConcepto() {
		return concepto;
	}
```

```
public void setConcepto(String concepto) {
	this.concepto = concepto;
}

@Column (name="id_nomina")
public int getId_nomina() {
	return id_nomina;
}



public void setId_nomina(int idNomina) {
	id_nomina = idNomina;
}

public int getPeriodo_procesado() {
	return periodo_procesado;
}

@Column(name="periodo_procesado")
public void setPeriodo_procesado(int periodoProcesado) {
	periodo_procesado = periodoProcesado;
}

@Column(name="id_empleado")
public int getId_empleado() {
	return id_empleado;
```

```
}
public void setId_empleado(int idEmpleado) {
	id_empleado = idEmpleado;
}
@Column(name="nombreempleado")
public String getNombreempleado() {
	return nombreempleado;
}
public void setNombreempleado(String nombreempleado) {
	this.nombreempleado = nombreempleado;
}

@Column(name="apaterno")
public String getApaterno() {
	return apaterno;
}
public void setApaterno(String apaterno) {
	this.apaterno = apaterno;
}

@Column(name="amaterno")
public String getAmaterno() {
	return amaterno;
}
public void setAmaterno(String amaterno) {
	this.amaterno = amaterno;
```

```
}

@Column(name="rfc")
public String getRfc() {
	return rfc;
}
public void setRfc(String rfc) {
	this.rfc = rfc;
}

@Column(name="curp")
public String getCurp() {
	return curp;
}
public void setCurp(String curp) {
	this.curp = curp;
}

@Column(name="imss")
public String getImss() {
	return imss;
}
public void setImss(String imss) {
	this.imss = imss;
}
```

```
@Column(name="monto")
public float getMonto() {
	return monto;
}
public void setMonto(float monto) {
	this.monto = monto;
}

@Column(name="tipo")
public String getTipo() {
	return tipo;
}
public void setTipo(String tipo) {
	this.tipo = tipo;
}
@Id
@Column(name="id_concepto")
public int getId_concepto() {
	return id_concepto;
}
public void setId_concepto(int idConcepto) {
	id_concepto = idConcepto;
}

@OneToOne(targetEntity=NominaDTO.class)
@JoinColumn(name="id_nomina",insertable=false,updatable=false)
```

```
        public NominaDTO getNomina() {
                return nomina;
        }
        public void setNomina(NominaDTO nomina) {
                this.nomina = nomina;
        }
```

```
} package proyecto.terminal.horacio.datos;
/**
 * Clase sin relación alguna con base de datos,
 * se usa para cargar manualmente periodos, quincenal,semanal,mensual
 * @author Horacio Iturbe
 *
 */
public class Periodos {

      private int id;
      private String nombre;

      public int getId() {
            return id;
```

```
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}


}
```

```
package proyecto.terminal.horacio.datos;


import java.io.Serializable;
import javax.persistence.*;
import static javax.persistence.GenerationType.IDENTITY;
/**
 * Tabla asociada <b>idiomas_persona</b>, guarda una relación ManyToOne con DatosPersonalesDTO,
 * tiene la información de algun idioma registrado por la persona
 * @author Horacio Iturbe
 * @see DatosPersonalesDTO
 *
 *
 */
```

```
@Entity
@Table (name="IDIOMAS_PERSONA")

public class PersonaIdiomasDTO implements Serializable{

	/**
	 * Identificador del idioma registrado y asociado a la persona
	 */
	private int id;
	/**
	 * Identificador del idioma en el catálogo de idiomas
	 */
	private int id_idioma;
	/**
	 * Porcentaje de lectura
	 */
	private String leido;
	/**
	 * Porcentaje de escritura
	 */
	private String escrito;
	/**
	 * Porcentaje idioma hablado
	 */
	private String hablado;
	/**
```

```
 * Por ser el Many side de la relación se tiene este campo para control con hibernate
 */
private int index_list;
@Id
@JoinColumn(name="ID") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}

public int getId_idioma() {
	return id_idioma;
}
public void setId_idioma(int idIdioma) {
	id_idioma = idIdioma;
}
public String getLeido() {
	return leido;
}
public void setLeido(String leido) {
	this.leido = leido;
}
public String getEscrito() {
```

```
		return escrito;
	}
	public void setEscrito(String escrito) {
		this.escrito = escrito;
	}
	public String getHablado() {
		return hablado;
	}
	public void setHablado(String hablado) {
		this.hablado = hablado;
	}

	@Column(name="INDEX_LIST",insertable=false,updatable=false)
	public int getIndex_list() {
		return index_list;
	}

	public void setIndex_list(int indexList) {
		index_list = indexList;
	}



} package proyecto.terminal.horacio.datos;

import java.io.Serializable;
```

```
import javax.persistence.*;


import static javax.persistence.GenerationType.IDENTITY;

/**

 * Tabla asociada <b>referencias_persona</b>,  guarda una relación ManyToOne con DatosPersonalesDTO,

 * tiene información de las referencias personales registradas por el usuario

 *

 * @author Horacio Iturbe

 * @see DatosPersonalesDTO

 *

 */

@Entity

@Table (name="REFERENCIAS_PERSONA")


public class ReferenciasPersonaDTO implements Serializable{


	/**

	 * Identificador de la referencia personal registrada

	 */

	private int id;


	private String nombre;

	private String telefono;

	private String domicilio;
```

```
/**
 * Por ser el Many side de la relación, se tiene este campo para control con hibernate
 */
private int index_list;


@Id
@Column(name="ID") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}


public void setId(int id) {
	this.id = id;
}
@Column(name="nombre")
public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
@Column(name="telefono")
public String getTelefono() {
	return telefono;
}
public void setTelefono(String telefono) {
```

```
		this.telefono = telefono;
	}
	@Column(name="domicilio")
	public String getDomicilio() {
		return domicilio;
	}
	public void setDomicilio(String domicilio) {
		this.domicilio = domicilio;
	}

	@Column(name="INDEX_LIST",insertable=false,updatable=false)
	public int getIndex_list() {
		return index_list;
	}

	public void setIndex_list(int indexList) {
		index_list = indexList;
	}



} package proyecto.terminal.horacio.datos;

import java.io.Serializable;
import java.util.List;
```

```
import javax.persistence.*;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.IndexColumn;

import static javax.persistence.GenerationType.IDENTITY;
/**
 * Tabla asociada <b>registro_incidencia</b>, guarda una relación OneToMany con IncidenciaDTO.
 * Tiene las incidencias registradas por empleado
 * @author Horacio Iturbe
 * @see IncidenciaDTO
 *
 */
@Entity
@Table(name="registro_incidencia")

public class RegistroIncidenciaDTO implements Serializable{

	/**
	 * Identificador de la incidencia registrada
	 */
	private int id;
	/**
	 * Identificador del empleado al cual se le asigno la incidencia
	 */
	private int id_empleado;
```

```
/**
 * Periodo en el que aplica
 */
private int periodo_aplica;
/**
 * Relación OneToMany
 */
private List<IncidenciaDTO> incidencias;
private EmpleadoDTO emp;


@Id
@Column(name="id") @GeneratedValue(strategy=IDENTITY)
public int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
@Column(name="id_empleado")
public int getId_empleado() {
	return id_empleado;
}
public void setId_empleado(int idEmpleado) {
	id_empleado = idEmpleado;
}
```

```
	@OneToMany(targetEntity=IncidenciaDTO.class,fetch=FetchType.EAGER,cascade=Cascade
Type.ALL)
	@Cascade(value=org.hibernate.annotations.CascadeType.DELETE_ORPHAN)
	@JoinColumn(name="id_registro",nullable=false)
	@IndexColumn(name="index_list")
	public List<IncidenciaDTO> getIncidencias() {
		return incidencias;
	}
	public void setIncidencias(List<IncidenciaDTO> incidencias) {
		this.incidencias = incidencias;
	}

	@Transient
	public int getIncidenciasSize()
	{
		return incidencias.size();
	}
	public int getPeriodo_aplica() {
		return periodo_aplica;
	}
	public void setPeriodo_aplica(int periodoAplica) {
		periodo_aplica = periodoAplica;
	}
	@OneToOne(targetEntity=EmpleadoDTO.class)
	@JoinColumn(name="id_empleado",insertable=false,updatable=false)
```

```
	public EmpleadoDTO getEmp() {
		return emp;
	}
	public void setEmp(EmpleadoDTO emp) {
		this.emp = emp;
	}
```

```
} package proyecto.terminal.horacio.datos;
import javax.persistence.*;
import java.io.Serializable;

/**
 * Vista asociada <b>reporteempe</b>, este DTO solo es usado para generar los reportes
 */
@Entity
@Table(name="reporteempe")
public class ReporteEmpleadoDTO implements Serializable{

	private int id_nomina;
	private int id_empleado;
	private int id_persona;
	private String nombreempleado;
```

```
private String apaterno;

private String amaterno;

private String sexo;

private String fecha_nacimiento;

private String rfc;

private String curp;

private String imss;

private String telefono_casa;

private String telefono_celular;

private String email;

private String calle;

private String cp;

private String colonia;

private String delegacion;

private int index_list; //id

private String nombreidioma;

private int id_idioma;

private String leido;

private String escrito;

private String hablado;

private int id_nivel_academico;

private String nombreescuela;

private String grado;

private String nombreempresa;

private String duracion;

private String puesto;
```

```
private String actividades;


public int getId_nomina() {
	return id_nomina;
}
public void setId_nomina(int idNomina) {
	id_nomina = idNomina;
}
public int getId_empleado() {
	return id_empleado;
}
public void setId_empleado(int idEmpleado) {
	id_empleado = idEmpleado;
}
public int getId_persona() {
	return id_persona;
}
public void setId_persona(int idPersona) {
	id_persona = idPersona;
}


public String getApaterno() {
	return apaterno;
}
public void setApaterno(String apaterno) {
	this.apaterno = apaterno;
```

```
}
public String getAmaterno() {
	return amaterno;
}
public void setAmaterno(String amaterno) {
	this.amaterno = amaterno;
}
public String getSexo() {
	return sexo;
}
public void setSexo(String sexo) {
	this.sexo = sexo;
}
public String getFecha_nacimiento() {
	return fecha_nacimiento;
}
public void setFecha_nacimiento(String fechaNacimiento) {
	fecha_nacimiento = fechaNacimiento;
}
public String getRfc() {
	return rfc;
}
public void setRfc(String rfc) {
	this.rfc = rfc;
}
public String getCurp() {
```

```
        return curp;
}
public void setCurp(String curp) {
        this.curp = curp;
}
public String getImss() {
        return imss;
}
public void setImss(String imss) {
        this.imss = imss;
}
public String getTelefono_casa() {
        return telefono_casa;
}
public void setTelefono_casa(String telefonoCasa) {
        telefono_casa = telefonoCasa;
}
public String getTelefono_celular() {
        return telefono_celular;
}
public void setTelefono_celular(String telefonoCelular) {
        telefono_celular = telefonoCelular;
}
public String getEmail() {
        return email;
}
```

```
public void setEmail(String email) {
        this.email = email;
}
public String getCalle() {
        return calle;
}
public void setCalle(String calle) {
        this.calle = calle;
}
public String getCp() {
        return cp;
}
public void setCp(String cp) {
        this.cp = cp;
}
public String getColonia() {
        return colonia;
}
public void setColonia(String colonia) {
        this.colonia = colonia;
}
public String getDelegacion() {
        return delegacion;
}
public void setDelegacion(String delegacion) {
        this.delegacion = delegacion;
```

```
}


@Id
public int getIndex_list() {
	return index_list;
}
public void setIndex_list(int indexList) {
	index_list = indexList;
}
public String getNombreidioma() {
	return nombreidioma;
}
public void setNombreidioma(String nombreidioma) {
	this.nombreidioma = nombreidioma;
}
public int getId_idioma() {
	return id_idioma;
}
public void setId_idioma(int idIdioma) {
	id_idioma = idIdioma;
}
public String getLeido() {
	return leido;
}
public void setLeido(String leido) {
	this.leido = leido;
```

```
}
public String getEscrito() {
	return escrito;
}
public void setEscrito(String escrito) {
	this.escrito = escrito;
}
public String getHablado() {
	return hablado;
}
public void setHablado(String hablado) {
	this.hablado = hablado;
}
public int getId_nivel_academico() {
	return id_nivel_academico;
}
public void setId_nivel_academico(int idNivelAcademico) {
	id_nivel_academico = idNivelAcademico;
}
public String getNombreescuela() {
	return nombreescuela;
}
public void setNombreescuela(String nombreescuela) {
	this.nombreescuela = nombreescuela;
}
public String getGrado() {
```

```
        return grado;
}
public void setGrado(String grado) {
        this.grado = grado;
}
public String getNombreempresa() {
        return nombreempresa;
}
public void setNombreempresa(String nombreempresa) {
        this.nombreempresa = nombreempresa;
}
public String getDuracion() {
        return duracion;
}
public void setDuracion(String duracion) {
        this.duracion = duracion;
}
public String getPuesto() {
        return puesto;
}
public void setPuesto(String puesto) {
        this.puesto = puesto;
}
public String getActividades() {
        return actividades;
}
```

```
	public void setActividades(String actividades) {
		this.actividades = actividades;
	}
	public String getNombreempleado() {
		return nombreempleado;
	}
	public void setNombreempleado(String nombreempleado) {
		this.nombreempleado = nombreempleado;
	}
```

```
} package proyecto.terminal.horacio.datos;

import java.io.Serializable;
import java.util.List;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.IndexColumn;
import javax.persistence.*;

import static javax.persistence.GenerationType.IDENTITY;
```

```
/**
 * Tabla asociada <b>tabulador</b>, tiene una relación OneToMany con TabuladorNivelesDTO.
 * Tiene la información de los tabuladores que se registran en el sistema y son usados para registrar nominas
 * @author Horacio Iturbe
 * @see TabuladorNivelesDTO
 *
 */
@Entity
@Table(name="tabulador")
public class TabuladorDTO implements Serializable{

	/**
	 * Identificador del tabulador registrado
	 */
	private int id;

	/**
	 * Nombre asociado
	 */
	private String nombre;
	/**
	 * Relación OneToMany con TabuladorNivelesDTO
	 */
	private List<TabuladorNivelesDTO> niveles;
```

```
	@Id
	@Column(name="id") @GeneratedValue(strategy=IDENTITY)
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}

	@Column(name="nombre")
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	@OneToMany(targetEntity=TabuladorNivelesDTO.class,cascade=CascadeType.ALL,fetch=F
etchType.EAGER)
	@JoinColumn(name="id_tabulador",nullable=false)
	@Cascade(value=org.hibernate.annotations.CascadeType.DELETE_ORPHAN)
	@IndexColumn(name="index_list")
	public List<TabuladorNivelesDTO> getNiveles() {
		return niveles;
	}
	public void setNiveles(List<TabuladorNivelesDTO> niveles) {
```

```
		this.niveles = niveles;
	}
	@Transient
	public int getTamNiv()
	{
		return niveles.size();
	}



} package proyecto.terminal.horacio.datos;


import java.io.Serializable;
import java.util.List;


import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.IndexColumn;
import javax.persistence.*;


import static javax.persistence.GenerationType.IDENTITY;
/**
 * Tabla asociada <b>tabulador</b>, tiene una relación OneToMany con TabuladorNivelesDTO.
 * Tiene la información de los tabuladores que se registran en el sistema y son usados para registrar nominas
 * @author Horacio Iturbe
 * @see TabuladorNivelesDTO
 *
```

```
 */
@Entity
@Table(name="tabulador")
public class TabuladorDTO implements Serializable{

	/**
	 * Identificador del tabulador registrado
	 */
	private int id;

	/**
	 * Nombre asociado
	 */
	private String nombre;
	/**
	 * Relación OneToMany con TabuladorNivelesDTO
	 */
	private List<TabuladorNivelesDTO> niveles;

	@Id
	@Column(name="id") @GeneratedValue(strategy=IDENTITY)
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
```

```
	}


	@Column(name="nombre")
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}


	@OneToMany(targetEntity=TabuladorNivelesDTO.class,cascade=CascadeType.ALL,fetch=F
etchType.EAGER)
	@JoinColumn(name="id_tabulador",nullable=false)
	@Cascade(value=org.hibernate.annotations.CascadeType.DELETE_ORPHAN)
	@IndexColumn(name="index_list")
	public List<TabuladorNivelesDTO> getNiveles() {
		return niveles;
	}
	public void setNiveles(List<TabuladorNivelesDTO> niveles) {
		this.niveles = niveles;
	}
	@Transient
	public int getTamNiv()
	{
		return niveles.size();
	}
```

```
} package proyecto.terminal.horacio.interceptor;


import java.util.Map;


import proyecto.terminal.horacio.datos.AccesoDTO;


import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;
/**
 * Interceptor para validar que haya un login registrado, de lo contrario, redirecciona a la
 * página de Login
 * @author Horacio Iturbe
 *
 */
public class ValidadorLogin implements Interceptor {


	@Override
	public void destroy() {
		// TODO Auto-generated method stub


	}


	@Override
```

```
	public void init() {
		// TODO Auto-generated method stub

	}
	/**
	 * verifica que en la session haya un objeto del tipo AccesoDTO
	 * @return String
	 */

	@Override
	public String intercept(ActionInvocation action) throws Exception {

		Map session=action.getInvocationContext().getSession();

		AccesoDTO acceso=(AccesoDTO)session.get("usuario");
			if(acceso==null)
				return Action.LOGIN;
			else
				return action.invoke();//Sigue con la accion
	}



} package proyecto.terminal.horacio.ruler;

import java.util.List;
```

```
import proyecto.terminal.horacio.datos.NivelesAcademicosDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.PersonalesRuler;
import com.opensymphony.xwork2.ActionSupport;
/**
 * <b>Esta acción es usada para atender una petición Ajax
 * desde el módulo Registrar Persona en la pestaña de
 * Estudios</b>
 *
 * @author Horacio Iturbe
 *
 */
public class AjaxEstudios extends ActionSupport{

	private PersonalesRuler _personasRecursos;
	private Object jsonModel;

	/**
	 * Regresa una colección de NivelesAcademicosDTO en su representación JSON
	 * para ser procesada desde el JavaScript que hace la petición
	 * @return String
	 */

	public String execute()
	{
		List<NivelesAcademicosDTO> estudios;
		estudios=_personasRecursos.listarNivelesAcademicos();
```

```
		setJsonModel(estudios);
		return SUCCESS;
}

/**
 * Spring inyecta este objeto con la implementación de la clase
 * PersonalesRulerImpl
 *
 * @param personasRecursos
 */

public void setPersonasRecursos(PersonalesRuler personasRecursos) {
		_personasRecursos = personasRecursos;
}

/**
 * Accesor para que Struts tome la colección del ValueStack y la
 * regrese en formato JSON
 * @return Object
 */
public Object getJsonModel() {
		return jsonModel;
}
/**
 * Accesor para que Struts tome la colección del ValueStack y la
```

```
	 * transforme en formato JSON
	 * @param jsonModel
	 */

	public void setJsonModel(Object jsonModel) {
		this.jsonModel = jsonModel;
	}

} package proyecto.terminal.horacio.ruler;

import java.util.List;
import proyecto.terminal.horacio.datos.NivelesAcademicosDTO;
import proyecto.terminal.horacio.datos.NominaProcesadaDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.EmpleadosRuler;
import proyecto.terminal.horacio.rulerBase.interfaz.PersonalesRuler;
import com.opensymphony.xwork2.ActionSupport;
/**
 * <b>Esta acción es usada para atender una petición Ajax
 * desde el módulo  Nómina Procesada en el menú de Reportes
 * </b>
 *
 * @author Horacio Iturbe
 *
 */
public class AjaxPeriodos extends ActionSupport{
```

```
private EmpleadosRuler _empleadoRecurso;


private Object jsonModel;
private int idnomina;



/**
 * Regresa una colección de NominaProcesadaDTO en su representación JSON
 * para ser procesada desde el JavaScript que hace la petición
 * @return String
 */

public String execute()
{
	List<String> nominas;
	nominas=_empleadoRecurso.listarPeriodosProcesados(idnomina);
	setJsonModel(nominas);
	return SUCCESS;
}



/**
 * Accesor para que Struts tome la colección del ValueStack y la
 * regrese en formato JSON
 * @return Object
 */
```

```
public Object getJsonModel() {
	return jsonModel;
}
/**
 * Accesor para que Struts tome la colección del ValueStack y la
 * transforme en formato JSON
 * @param jsonModel
 */

public void setJsonModel(Object jsonModel) {
	this.jsonModel = jsonModel;
}



/**
 * Spring inyecta este objeto con la implementación
 * EmpleadoREcursoImpl
 * @param empleadoRecurso
 */
public void setEmpleadoRecurso(EmpleadosRuler empleadoRecurso) {
	_empleadoRecurso = empleadoRecurso;
}



public int getIdnomina() {
	return idnomina;
```

```
	}



	public void setIdnomina(int idnomina) {
		this.idnomina = idnomina;
	}

} package proyecto.terminal.horacio.ruler;


import java.util.ArrayList;
import java.util.List;
import proyecto.terminal.horacio.datos.*;
import proyecto.terminal.horacio.rulerBase.interfaz.*;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

/**
 * <b> Acción que se encarga de la Alta y Modificación de los empleados</b>
 *
 * @author Horacio
 *
 */
public class EmpleadoAction extends ActionSupport implements ModelDriven<EmpleadoDTO> {
```

```
	private EmpleadosRuler _empleadoRecurso;

	private NominaTabRuler _nominaTabulador;

	private PersonalesRuler _personasRecursos;


	private EmpleadoDTO empleado=new EmpleadoDTO();

	private DatosPersonalesDTO personal= new DatosPersonalesDTO();

	private List<DatosPersonalesDTO> personasNoEmpleados= new
ArrayList<DatosPersonalesDTO>();

	private List<NominaDTO> nominas;


	private List<EmpleadoDTO> empleados= new ArrayList<EmpleadoDTO>();

	private int ida;//Id de la persona que se dará de alta

	private int ide;//Id del Empleado que se modificará

	private String mensaje;


	/**
	 * Carga la forma para registrar a un empleado, obtiene una colección que es, las nominas
	 * registradas en el sistema y un DTO de la persona que será registrada.
	 *@return String
	 */


	public String forma()
	{
		nominas=_nominaTabulador.listarNominas();
		personal=_personasRecursos.personaPorId(ida);
```

```
		return "forma_registro";
	}

	/**
	 * Realiza el registro a la base de datos y regresa un mensaje de éxito para ser mostrado al usuario
	 * @return String
	 */
	public String registrar()
	{
		_empleadoRecurso.salvar(empleado);
		mensaje=" registrar empleado";
		return "exito";
	}
	/**
	 * Carga la forma para la modificación del empleado, obtiene una colección que es, las
	 * nominas registradas en el systema y un DTO con la información del empleado que será modificado.
	 * @return String
	 */

	public String forma_modificar()
	{
		empleado=_empleadoRecurso.empleadoPorId(getIde());
		nominas=_nominaTabulador.listarNominas();
```

```
            return "forma_modificar";


        }


        /**
         * Realiza la modificación del empleado y regresa un mensaje de éxito para ser mostrado al
usuario
         * @return String
         */
        public String modificar()
        {
            _empleadoRecurso.modificar(empleado);
            mensaje=" modificar empleado";
            return "exito";
        }


        /**
         * Obtiene una colección de DTO's EmpleadoDTO de los empleados registrados en el
sistema
         * @return String
         */
        public String listado()
        {
            empleados=_empleadoRecurso.listarEmpleados();
            return "listadoEmpleados";
        }
```

```
	/**
	 * Obtiene una colección de DTO's DatosPersonalesDTO de las personas que estan registradas
	 * pero que no han sido registradas como empleados.
	 * @return String
	 */
	public String listadoNoEmpleados()
	{
		personasNoEmpleados=_empleadoRecurso.listarPersonasNoEmpleado();
		return "listadoNoEmpleados";
	}
	/**
	 * Se obtiene el DTO EmpleadoDTO de la forma enviada por el usuario
	 *
	 */
	public EmpleadoDTO getModel() {
		return empleado;
	}



	/**
	 * Spring inyecta este objeto con la implementacion de la clase
	 *  EmpleadosRulerImpl
	 * @param empleadoRecurso
	 */
```

```
public void setEmpleadoRecurso(EmpleadosRuler empleadoRecurso) {
	_empleadoRecurso = empleadoRecurso;
}

/**
 * Accesor para el DTO EmpleadoDTO
 * @return EmpleadoDTO
 */
public EmpleadoDTO getEmpleado() {
	return empleado;
}

/**
 * Accesor para el DTO EmpleadoDTO
 * @param empleado
 */
public void setEmpleado(EmpleadoDTO empleado) {
	this.empleado = empleado;
}

/**
 * Spring inyecta este objeto con la implementación
 * NominaTabRulerImpl
 * @param nominaTabulador
 */
```

```
public void setNominaTabulador(NominaTabRuler nominaTabulador) {
	_nominaTabulador = nominaTabulador;
}

/**
 * Accesor para la colección de las nominas que es cargada
 * desde la base de datos
 * @return List NominaDTO
 */
public List<NominaDTO> getNominas() {
	return nominas;
}
/**
 * Accesor para la colección de las nominas
 * @param nominas
 */

public void setNominas(List<NominaDTO> nominas) {
	this.nominas = nominas;
}
/**
 * Spring inyecta este objeto con la implementación de
 * PersonalesRulerImpl
 *
 * @param personasRecursos
 */
```

```
public void setPersonasRecursos(PersonalesRuler personasRecursos) {
	_personasRecursos = personasRecursos;
}

/**
 * Accesor que contiene el id de la persona(DatosPersonalesDTO)
 * que se dará de alta como empleado
 * @return int
 */

public int getIda() {
	return ida;
}


/**
 * Accesor para el id de la persona
 * @param ida
 */
public void setIda(int ida) {
	this.ida = ida;
}

/**
 * Accesor para el DTO DatosPersonalesDTO
```

```
 * @return DatosPersonalesDTO
 */

public DatosPersonalesDTO getPersonal() {
	return personal;
}

/**
 * Accesor para el DTO DatosPersonalesDTO
 * @param personal
 */
public void setPersonal(DatosPersonalesDTO personal) {
	this.personal = personal;
}
/**
 * Accesor del id del empleado que será modificado
 * @return int
 */

public int getIde() {
	return ide;
}

/**
 * Accesor del id del empleado
 * @param ide
```

```
 */
public void setIde(int ide) {
	this.ide = ide;
}

/**
 * Accesor para la colección de personas que no han sido
 * registradas como empleados
 * @return List DatosPersonalesDTO
 */
public List<DatosPersonalesDTO> getPersonasNoEmpleados() {
	return personasNoEmpleados;
}
/**
 * Accesor para la colección de  personas que no han sido registradas como empleados
 * @param personasNoEmpleados
 */
public void setPersonasNoEmpleados(List<DatosPersonalesDTO> personasNoEmpleados) {
	this.personasNoEmpleados = personasNoEmpleados;
}

/**
 * Accesor para la colección de empleados registrados en el sistema
 * @return List EmpleadoDTO
 */
public List<EmpleadoDTO> getEmpleados() {
```

```
		return empleados;
}
/**
 * Accesor para la colección de empleados registrados en el sistema
 * @param empleados
 */

public void setEmpleados(List<EmpleadoDTO> empleados) {
		this.empleados = empleados;
}

/**
 * Mensaje utilizado para la confirmación de la operación
 * realizada por el usuario, los mensajes pueden ser:
 * <ul>
 * <li> registrar empleado
 * <li> modificar empleado
 * </ul>
 * @return String
 */
public String getMensaje() {
		return mensaje;
}

/**
 * Accesor para el mensaje enviado al usuario
```

```
	 * @param mensaje
	 */
	public void setMensaje(String mensaje) {
		this.mensaje = mensaje;
	}



} package proyecto.terminal.horacio.ruler;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.MalformedURLException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.core.io.ClassPathResource;
```

```
import net.sf.jasperreports.engine.*;

import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

import net.sf.jasperreports.engine.design.JasperDesign;

import net.sf.jasperreports.engine.xml.JRXmlLoader;


import proyecto.terminal.horacio.datos.EmpleadoDTO;

import proyecto.terminal.horacio.datos.NominaDTO;

import proyecto.terminal.horacio.datos.NominaProcesadaDTO;

import proyecto.terminal.horacio.datos.RegistroIncidenciaDTO;

import proyecto.terminal.horacio.datos.ReporteEmpleadoDTO;

import proyecto.terminal.horacio.rulerBase.interfaz.EmpleadosRuler;

import proyecto.terminal.horacio.rulerBase.interfaz.NominaTabRuler;


import com.opensymphony.xwork2.ActionSupport;

import com.opensymphony.xwork2.ModelDriven;


/**

 * <b> Acción que se encarga de realizar las operaciones pertinentes para las incidencias del empleado</b>

 * @author Horacio Iturbe

 *

 */

public class IncidenciasAction extends ActionSupport implements ModelDriven<RegistroIncidenciaDTO>{
```

```
	private int id_emp;//para modificar las incidencas
	private int id_nom;//para registrar las incidencias con esta cargamos los empleados por nomina
	private int id_incidencia;

	private RegistroIncidenciaDTO reg= new RegistroIncidenciaDTO();
	private List<RegistroIncidenciaDTO> inciden= new ArrayList<RegistroIncidenciaDTO>();

	private EmpleadosRuler _empleadoRecurso;
	private NominaTabRuler _nominaTabulador;

	private List<EmpleadoDTO> empleados;
	private List<NominaDTO> nominas;
	private EmpleadoDTO empleado;

	private String mensaje;

	/**
	 * Carga la forma con una colección de DTO's EmpleadoDTO a
	 * los cuales se les puede registrar una incidencia
	 *
	 * @return String
	 */
	public String forma()
	{
		empleados=_empleadoRecurso.listarEmpleadosPorNomina(id_nom);
```

```
        return "forma_registrar";
}
/**
 * Carga la forma con una colección de DTO's NominaDTO para
 * que el usuario registre incidencias a un empleado asociado a la nómina seleccionada
 * @return String
 */
public String carga()
{
        nominas=_nominaTabulador.listarNominas();
        return "carganominas";
}

/**
 * Registra la incidencia en la base de datos
 * y regresa un mensaje al usuario
 * @return String
 */
public String registrar()
{
        _empleadoRecurso.salvar(reg);
        mensaje=" registrar incidencia";
        return "exito";
}

/**
```

```
 * Carga la forma con las incidencias asociadas al impleado
 * @return String
 */
public String forma_modificar()
{
	reg=_empleadoRecurso.incidenciaPorId(id_incidencia);
	return "forma_modificar";
}

/**
 * Realiza la modificación de las incidencias
 * @return String
 */
public String modificar()
{
	_empleadoRecurso.modificarIncidencias(reg);
	mensaje=" modificar incidencia";
	return "exito";
}

/**
 * Lista las incidencias existentes en la base de datos
 * @return String
 */
public String listado()
{
```

```
		inciden=_empleadoRecurso.listarIncidencias();


		return "listado";
}


/**
 * Accesor para el DTO RegistroIncidenciaDTO
 * @return RegistroIncidenciaDTO
 */
public RegistroIncidenciaDTO getModel()
{
		return reg;
}


/**
 * Accesor Id del empleado al cual se le tiene asociado una lista de incidencas
 * @return int
 */


public int getId_emp() {
		return id_emp;
}


/**
 * Accesor Id del empleado al cual se le tiene asociado una lista de incidencas
 * @param idEmp
```

```
	 */


	public void setId_emp(int idEmp) {
		id_emp = idEmp;
	}


	/**
	 * Accesor del id de la nómina que se selecciono para posteriormente cargar los empleados
	 * @return int
	 */
	public int getId_nom() {
		return id_nom;
	}


	/**
	 * Accesor del id de la nómina que se selecciono para posteriormente cargar los empleados
	 * @param idNom
	 */
	public void setId_nom(int idNom) {
		id_nom = idNom;
	}


	/**
	 * Accesor del DTO RegistroIncidenciaDTO
```

```
 * @return RegistroIncidenciaDTO
 */
public RegistroIncidenciaDTO getReg() {
	return reg;
}

/**
 * Accesor del DTO RegistroIncidenciaDTO
 * @param reg
 */
public void setReg(RegistroIncidenciaDTO reg) {
	this.reg = reg;
}

/**
 * Spring inyecta este objeto con la implementación
 * EmpleadosRulerImpl
 * @param empleadoRecurso
 */
public void setEmpleadoRecurso(EmpleadosRuler empleadoRecurso) {
	_empleadoRecurso = empleadoRecurso;
}

/**
 * Accesor de la colección de DTO's EmpleadoDTO que es cargada desde la base de datos
```

```
 * @return List<EmpleadoDTO>
 */
public List<EmpleadoDTO> getEmpleados() {
	return empleados;
}

/**
 * Accesor de la colección de DTO's EmpleadoDTO que es cargada desde la base de datos
 * @param empleados
 */
public void setEmpleados(List<EmpleadoDTO> empleados) {
	this.empleados = empleados;
}

/**
 * Accesor del DTO EmpleadoDTO
 * @return EmpleadoDTO
 */
public EmpleadoDTO getEmpleado() {
	return empleado;
}

/**
 * Accesor del DTO EmpleadoDTO
 * @param empleado
 */
```

```
public void setEmpleado(EmpleadoDTO empleado) {
	this.empleado = empleado;
}

/**
 * Accesor para el id de la incidencia
 * @return int
 */
public int getId_incidencia() {
	return id_incidencia;
}
/**
 * Accesor para el id de la incidencia
 * @param idIncidencia
 */

public void setId_incidencia(int idIncidencia) {
	id_incidencia = idIncidencia;
}



/**
 * Accesor para la colección de las incidencias registradas en el sistema
 * @return List RegistroIncidenciaDTO
 */
```

```
public List<RegistroIncidenciaDTO> getInciden() {
	return inciden;
}

/**
 * Accesor para la colección de las incidencias registradas en el sistema
 * @param inciden
 */

public void setInciden(List<RegistroIncidenciaDTO> inciden) {
	this.inciden = inciden;
}

/**
 * Spring inyecta este objeto con la implementación de
 * NominaTabRulerImpl
 * @param nominaTabulador
 */

public void setNominaTabulador(NominaTabRuler nominaTabulador) {
	_nominaTabulador = nominaTabulador;
}

/**
 * Accesor para la colección de DTO's NominaDTO registradas en el sistema
 * @return List<NominaDTO>
```

```
 */
public List<NominaDTO> getNominas() {
	return nominas;
}

/**
 * Accesor para la colección de DTO's NominaDTO registradas en el sistema
 * @param nominas
 */
public void setNominas(List<NominaDTO> nominas) {
	this.nominas = nominas;
}

/**
 * Accesor para el mensaje que  le es enviado al usuario:
 * <ul>
 * <li> registrar incidencia
 * <li> modificar incidencia
 * </ul>
 * @return String
 */
public String getMensaje() {
	return mensaje;
}

/**
```

```
	 * Accesor para el mensaje que le es enviado al usuario
	 * @param mensaje
	 */
	public void setMensaje(String mensaje) {
		this.mensaje = mensaje;
	}
```

```
} package proyecto.terminal.horacio.ruler;

import java.util.Map;
import org.apache.struts2.interceptor.SessionAware;
import proyecto.terminal.horacio.datos.AccesoDTO;
import com.opensymphony.xwork2.ActionSupport;

/**
 *<b> Esta acción es la encargada de agregar
 * a la sesion el bean AccesoDTO que es utilizado para determinar el acceso
 * que tiene el usuario que ha ingresado al sistema, los roles pueden ser:
 * <ul>
 *	<li> Administrador
```

```
*          <li> Operador
*          <li> Rh
*          <li> Otro
* </ul>
* </b>
*
*
*


 * @author Horacio Iturbe
 *
 */
public class InicioAction extends ActionSupport implements SessionAware {




          private Map session;
          private int id;     //Este Id solo lo utilizo para cuando el usuario tenga accesos simples


          /**
           * Regresa el rol asociado al usuario
           *
           * @return String
           */
          public String execute()
          {
```

```
		AccesoDTO acceso= (AccesoDTO)session.get("usuario");
		id=acceso.getEmpleado().getPersona().getId();
		return acceso.getRol();
}



/**
 * Obtiene la información con la sesión, método implementado
 * de la interfaz SessionAware de Struts
 * @param session
 *
 */

public void setSession(Map session) {
		this.session = session;
}

/**
 * Regresa el id del empleado para cuando
 * su tipo de acceso es Otro
 * @return id
 */

public int getId() {
		return id;
}
```

```
	public void setId(int id) {
		this.id = id;
	}
```

```
} package proyecto.terminal.horacio.ruler;

import java.util.Map;

import org.apache.struts2.interceptor.SessionAware;

import proyecto.terminal.horacio.datos.AccesoDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.EmpleadosRuler;
```

```
import com.opensymphony.xwork2.ActionSupport;

/**
 * <b>Acción que valida al usuario que se encuentre registrado en la base de datos</b>
 * @author Horacio Iturbe
 *
 */
public class LocalizaUsuario extends ActionSupport implements SessionAware {


	private AccesoDTO acceso=new AccesoDTO();


	private String usuario;
	private String pass;
	private EmpleadosRuler _empleadoRecurso;
	private Map<String,AccesoDTO> session;


	/**
	 * Si el usuario existe en la base, se carga el DTO AccesoDTO y se direcciona a la acción
	 * InicioAction
	 * @return String
	 */
	public String execute()
	{


		return "success";
	}
	/**
```

```
 * valida la existencia del usuario en la base de datos y en caso de existir
 * coloca en sesion el DTO AccesoDTO
 */
public void validate()
{

	acceso=_empleadoRecurso.isUsuario(usuario,pass);

	if(acceso==null)
		addFieldError("usuario","Usuario sin acceso al sistema");

	session.put("usuario",acceso);

}
/**
 * Accesor para el usuario capturado en la forma de login
 * @return String
 */
public String getUsuario() {
	return usuario;
}

/**
 * Accesor para el usuario capturado en la forma de login
 * @param usuario
 */
```

```
public void setUsuario(String usuario) {
	this.usuario = usuario;
}

/**
 * Accesor para el password capturado en la forma de login
 * @return String
 */
public String getPass() {
	return pass;
}

/**
 * Accesor para el password capturado en la forma de login
 * @param pass
 */

public void setPass(String pass) {
	this.pass = pass;
}

/**
 * Spring inyecta este objeto con la implementación de
 * EmpleadosRulerImpl
 * @param empleadoRecurso
```

```
 */
public void setEmpleadoRecurso(EmpleadosRuler empleadoRecurso) {
	_empleadoRecurso = empleadoRecurso;
}



/**
 * Accesor para el DTO AccesoDTO
 * @return AccesoDTO
 */

public AccesoDTO getAcceso() {
	return acceso;
}
/**
 * Accesor para el DTO AccesoDTO
 * @param acceso
 */
public void setAcceso(AccesoDTO acceso) {
	this.acceso = acceso;
}

/**
 * Accesor para la sesion del usuario
 * @param session
 */
```

```
	public void setSession(Map session) {
		this.session = session;
	}
```

```
} package proyecto.terminal.horacio.ruler;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import org.apache.struts2.interceptor.SessionAware;

import proyecto.terminal.horacio.datos.*;
import proyecto.terminal.horacio.rulerBase.interfaz.NominaTabRuler;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
/**
 *
 * <b>Esta acción se encarga de gestionar los registros de una nómina al sistema</b>
 * @author Horacio Iturbe
 *
 */
```

```
public class NominaAction extends ActionSupport implements
ModelDriven<NominaDTO>,SessionAware {



	private NominaTabRuler _nominaTabulador;

	private NominaDTO nomina=new NominaDTO();

	private List<TabuladorDTO> tabuladores;

	private List<Periodos> periodos;

	private List<NominaDTO> nominas= new ArrayList<NominaDTO>();

	private int id_nom;


	private Map session;

	private String mensaje;



	/**

	 * Carga la forma para registrar la nómina con una colección TabuladorDTO de tabuladores
disponibles

	 * y una colección Periodos de periodos disponibles

	 * @return String

	 */


	public String forma()

	{

		setTabuladores(_nominaTabulador.listarTabuladores());
```

```
		setPeriodos(cargaPeriodos());



		return "forma_registro";
	}

	/**
	 * registra la nómina a la base de datos
	 * @return String
	 */
	public String registrar()
	{
		_nominaTabulador.salvar(nomina);
		mensaje=" registrar n&oacute;mina";
		return "exito";
	}

	/**
	 * Carga la forma para modificar la nómina con sus conceptos asociados y los tabuladores disponibles
	 * @return String
	 */

	public String formaModificar()
	{
		setNomina(_nominaTabulador.nominaPorId(getId_nom()));
```

```
            setTabuladores(_nominaTabulador.listarTabuladores());
            setPeriodos(cargaPeriodos());
            return "forma_modificar";
}

/**
 * realiza la modificación a la base de datos
 * @return String
 */
public String modificar()
{
            _nominaTabulador.modificar(nomina);
            mensaje=" modificar n&oacute;mina";
            return"exito";
}

/***
 * Lista las nominas registradas en el sistema
 * @return String
 */

public String listado() //Buscar nom
{
            nominas=_nominaTabulador.listarNominas();
            return "listado";
}
```

```
/**
 * Spring inyecta este objeto con la implementación
 * NominaTabRulerImpl
 * @param nominaTabulador
 */

public void setNominaTabulador(NominaTabRuler nominaTabulador) {
	_nominaTabulador = nominaTabulador;
}

/**
 * Accesor para el id de la nómina
 * @return int
 */

public int getId_nom() {
	return id_nom;
}

/**
 * Accesor para el id de la nómina
 * @param id
 */
```

```
public void setId_nom(int id) {
		this.id_nom = id;
}

/**
 * Accesor de la forma de registro de nómina
 * @return NominaDTO
 */
public NominaDTO getModel() {

		return nomina;
}
/**
 * Accesor del DTO NominaDTO
 * @return NominaDTO
 */

public NominaDTO getNomina() {
		return nomina;
}

/**
 * Accesor del DTO NominaDTO
 * @param nomina
 */
public void setNomina(NominaDTO nomina) {
```

```
		this.nomina = nomina;
}

/**
 * Accesor de la colección TabuladorDTO de los tabuladores registrados en el sistema
 * @return List TabuladorDTO
 */
public List<TabuladorDTO> getTabuladores() {
		return tabuladores;
}

/**
 * Accesor de la colección TabuladorDTO de los tabuladores registrados en el sistema
 * @param tabuladores
 */
public void setTabuladores(List<TabuladorDTO> tabuladores) {
		this.tabuladores = tabuladores;
}


/**
 * Accesor de la colección<Periodos>
 * @return List Periodos
 */
public List<Periodos> getPeriodos() {
		return periodos;
```

```
}

/**
 * Acceosr de la colección Periodos
 * @param periodos
 */
public void setPeriodos(List<Periodos> periodos) {
	this.periodos = periodos;
}

/**
 * carga la colección de periodos
 * @return List Periodos
 */
public List<Periodos> cargaPeriodos()
{
	List<Periodos> a=new ArrayList<Periodos>();
	Periodos p=new Periodos();
	p.setId(7);
	p.setNombre("Semanal");
	a.add(p);

	p=new Periodos();
	p.setId(15);
	p.setNombre("Quincenal");
	a.add(p);
```

```
		p=new Periodos();
		p.setId(30);
		p.setNombre("Mensual");
		a.add(p);

		return a;

}



/**
 * Accesor de la colección NominaDTO
 * @return List NominaDTO
 */

public List<NominaDTO> getNominas() {
		return nominas;
}

/**
 * Accesor de la colección NominaDTO
 * @param nominas
 */

public void setNominas(List<NominaDTO> nominas) {
```

```
		this.nominas = nominas;
}

/**
 * Accesor para la sesión del usuario
 * @return Map session
 */
public Map getSession() {
		return session;
}

/**
 * Accesor para la sesión del usuario
 * @param session
 */
public void setSession(Map session) {
		this.session = session;
}

/**
 * Accesor para el mensaje enviado al usuario
 * @return String
 */
public String getMensaje() {
		return mensaje;
}
```

```
	/**
	 * Accesor para el mensaje enviado al usuario
	 * @param mensaje
	 */

	public void setMensaje(String mensaje) {
		this.mensaje = mensaje;
	}
```

```
} package proyecto.terminal.horacio.ruler;

import java.util.ArrayList;
import java.util.List;
import proyecto.terminal.horacio.datos.*;
import proyecto.terminal.horacio.rulerBase.interfaz.*;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
/**
 * Acción encargada de registrar y modificar datos personales
 * @author Horacio Iturbe
 *
```

```
 */

public class PersonaAction extends ActionSupport implements
ModelDriven<DatosPersonalesDTO> {



        private PersonalesRuler _personasRecursos;


        private List<NivelesAcademicosDTO> _niveles= new ArrayList<NivelesAcademicosDTO>();

        private List<IdiomasDTO> _idiomas= new ArrayList<IdiomasDTO>();

        private List<DatosPersonalesDTO> personas;

        private DatosPersonalesDTO binder = new DatosPersonalesDTO();

        private int idp;

        private String mensaje;


        /**
         * Carga la forma de registro de una persona
         * @return String
         */
        public String nuevo()
        {


                _niveles=_personasRecursos.listarNivelesAcademicos();

                _idiomas=_personasRecursos.listarIdiomas();

                return "forma_registro";
```

```
}

/**
 * Registra el dto DatosPersonalesDTO  y envia mensaje de éxito
 * @return String
 *
 */

public String registrar()
{
	_personasRecursos.salvar(binder);
	System.out.println("---------- "+binder.getAmaterno());
	mensaje=" registrar persona";

	return "exito";
}



/**
 * carga la forma para modificar el dto DatosPersonalesDTO
 * @return String
 */
public String forma_modificar()
{
	_niveles=_personasRecursos.listarNivelesAcademicos();
	_idiomas=_personasRecursos.listarIdiomas();
```

```
        setBinder(_personasRecursos.personaPorId(getIdp()));
        return "forma_modificar";
}

/**
 * Realiza la modificacion del dto DatosPersonalesDTO
 * @return String
 */

public String modificar()
{
        _personasRecursos.modificarPersona(binder);
        System.out.println("---------------- "+binder.getAmaterno());
        mensaje=" modificar persona";
        return "exito";
}

/**
 * lista las personas registradas en el sistema
 * @return String
 */

public String listado()
{
        personas=_personasRecursos.listarPersonas();
        return "listado_personas";
```

```
}



/**
 * Spring inyecta este objeto con la implementación
 * PersonalesRulerImpl
 *
 * @param personasRecursos
 */

public void setPersonasRecursos(PersonalesRuler personasRecursos) {
	_personasRecursos = personasRecursos;

}

/**
 * Niveles académicos disponibles en el sistema
 * @return List NivelesAcademicosDTO
 */
public List<NivelesAcademicosDTO> getNiveles() {
	return _niveles;
}

/**
 * Niveles académicos disponibles en el sistema
 * @param niveles
```

```
 */
public void setNiveles(List<NivelesAcademicosDTO> niveles) {
	_niveles = niveles;
}

/**
 * Idiomas disponibles en el sistema
 * @return List IdiomasDTO
 */
public List<IdiomasDTO> getIdiomas() {
	return _idiomas;
}

/**
 * Idiomas disponibles en el sistema
 * @param idiomas
 */

public void setIdiomas(List<IdiomasDTO> idiomas) {
	_idiomas = idiomas;
}

/**
 * Datos obtenidos de la forma para registrar persona
 * @return DatosPersonalesDTO
 */
```

```
public DatosPersonalesDTO getBinder() {
		return binder;
}

/**
 * Datos obtenidos de la forma para registrar persona
 * @param binder
 */
public void setBinder(DatosPersonalesDTO binder) {
		this.binder = binder;
}


/**
 * Carga el DTO con los datos obtenidos de la forma
 * @return DatosPersonalesDTO
 */
public DatosPersonalesDTO getModel() {
		return binder;
}


/**
 * Personas registradas en el sistema
 * @return List DatosPersonalesDTO
 */
```

```
public List<DatosPersonalesDTO> getPersonas() {
		return personas;
}

/**
 * Personas registradas en el sistema
 * @param personas
 */

public void setPersonas(List<DatosPersonalesDTO> personas) {
		this.personas = personas;
}

/**
 * Id de la persona a modificar
 * @return int
 */
public int getIdp() {
		return idp;
}

/**
 * Id de la persona a modificar
 * @param idp
 */
```

```
	public void setIdp(int idp) {
		this.idp = idp;
	}

	/**
	 * Mensaje de éxito que se le envia al usuario
	 * @return String
	 */

	public String getMensaje() {
		return mensaje;
	}

	/**
	 * Mensaje de éxito que se le envia al usuario
	 * @param mensaje
	 */
	public void setMensaje(String mensaje) {
		this.mensaje = mensaje;
	}
```

} package proyecto.terminal.horacio.ruler;

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Map;

import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import net.sf.jasperreports.engine.design.JasperDesign;
import net.sf.jasperreports.engine.xml.JRXmlLoader;

import org.apache.struts2.interceptor.SessionAware;
import org.springframework.core.io.ClassPathResource;

import proyecto.terminal.horacio.datos.AccesoDTO;
import proyecto.terminal.horacio.datos.NominaDTO;
import proyecto.terminal.horacio.datos.NominaProcesadaDTO;
import proyecto.terminal.horacio.datos.ReporteEmpleadoDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.EmpleadosRuler;
```

```
import proyecto.terminal.horacio.rulerBase.interfaz.NominaTabRuler;

import com.opensymphony.xwork2.ActionSupport;

/**
 * Acción que se encarga de realizar los reportes e imprimirlos en PDF
 * @author Horacio Iturbe
 *
 */
public class ReportesAction extends ActionSupport implements SessionAware{

	private EmpleadosRuler _empleadoRecurso;

	private int idnomina;
	private int periodo;
	private int empleadoid;


	private String link;

	private InputStream salidaPdf;
	private List<NominaProcesadaDTO> nom;
	private List<NominaDTO>nominas;
	private Map session;
	private String mensaje;
	public int algo;
```

```
private NominaTabRuler _nominaTabulador;
private List<String> periodos;

/**
 * Carga la forma para reporte de nominas procesadas
 * @return String
 */
public String formaNomina()
{
	nominas=_nominaTabulador.listarNominas();
	return "forma_nomina";
}

/**
 * Carga la forma para reporte de empleados por nómina
 * @return String
 */
public String formaEmpleado()
{
	nominas=_nominaTabulador.listarNominas();
	return "forma_empleado";
}
/**
 * carga la forma para que un empleado consulte su recibo de nómina
 * @return String
 */
```

```
public String formaSimple()
{

idnomina=((AccesoDTO)getSession().get("usuario")).getEmpleado().getId_nomina();
        periodos=_empleadoRecurso.listarPeriodosProcesados(idnomina);
        return "forma_empleadoSimple";
}

/**
 * Carga la forma para procesar la nómina
 * @return String
 */
public String formaProcesar()
{
        link="Procesar N&oacute;mina";
        nominas=_nominaTabulador.listarNominas();
        return "forma_procesar";
}

/**
 * Genera el reporte de la nomina procesada
 * @return String
 * @throws IOException En caso de que haya problemas de generar el archivo de salida PDf que muestra el reporte
 * @throws JRException En caso de haber algún conflicto con la colección recibida por JasperReports para generar el reporte
```

```
	 */
	public String nomina() throws IOException, JRException
	{
		List<NominaProcesadaDTO> nomi=_empleadoRecurso.listarNominaProcesada(idnomina,periodo); // tengo que crear otra que diga la nomina,el period y el id del empleado

		if(nomi.size()==0){
			mensaje="¡No hubo datos para generar el reporte verifique sus parametros!";
			return "reportevacio";
		}

		else{
			genera_nomina(nomi);
			return "nominaprocesada";
		}
	}

	/**
	 * Genera el reporte de la nomina por empleado
	 * @return String
	 * @throws IOException En caso de que haya problemas de generar el archivo de salida PDf que muestra el reporte
	 * @throws JRException En caso de haber algún conflicto con la colección recibida por JasperReports para generar el reporte
	 */
```

```
	public String nominaPorEmpleado() throws IOException, JRException

	{


	idnomina=((AccesoDTO)getSession().get("usuario")).getEmpleado().getId_nomina();

		empleadoid=((AccesoDTO)getSession().get("usuario")).getEmpleado().getId();


		List<NominaProcesadaDTO>
nomi=_empleadoRecurso.listarNominaProcesadaPorEmpleado(idnomina, periodo, empleadoid);


		if(nomi.size()==0){

			mensaje="¡No hubo datos para generar el reporte verifique sus
parametros!";

			return "reportevacio";

		}


		else{

			genera_nomina(nomi);

			return "nominaprocesada";

		}

	}

	/**

	 * Genera el reporte de los empleados registrados en el sistema

	 * @return String

	 * @throws IOException En caso de que haya problemas de generar el archivo de salida
PDf que muestra el reporte

	 * @throws JRException En caso de haber algún conflicto con la colección recibida por
JasperReports para generar el reporte
```

```
	 */

	public String empleado() throws IOException, JRException
	{
		List<ReporteEmpleadoDTO>
empleados=_empleadoRecurso.listarReporteEmpleado(idnomina);
		if(empleados.size()==0){
			mensaje="¡Nómina sin empleados asociados, verifique su configuración!";
			return "reportevacio";
		}

		else{
			genera_empleado(empleados);
			return "empleados";
		}

	}

	/**
	 * Procesa la nómina
	 * @return String
	 */
	public String procesar()
	{
		_empleadoRecurso.procesarNomina(idnomina, periodo);
```

```
                mensaje=" procesar n&oacute;mina";

                return "exito";

        }



        /**

         * Genera el archivo pdf y lo muestra en el browser, del reporte de nominas procesada

         * @param nomi

         * @throws IOException En caso de que haya problemas de generar el archivo de salida PDf que muestra el reporte

         * @throws JRException En caso de haber algún conflicto con la colección recibida por JasperReports para generar el reporte

         */


        public void genera_nomina(List<NominaProcesadaDTO> nomi)throws IOException, JRException

        {

                ByteArrayOutputStream baos = new ByteArrayOutputStream();

                ByteArrayInputStream bis; //Excelente solucion


                InputStream reportStream= new ClassPathResource("nomina.xml").getInputStream();


                JasperDesign     jasperDesign = JRXmlLoader.load(reportStream);

                JasperReport     jasperReport = JasperCompileManager.compileReport(jasperDesign);
```

```
                JRBeanCollectionDataSource ds = new JRBeanCollectionDataSource(nomi);// Aqui pasarle la coleccion



                JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, null, ds);



                JasperExportManager.exportReportToPdfStream(jasperPrint, baos);


                bis=new ByteArrayInputStream(baos.toByteArray());

                salidaPdf=bis;



        }


        /**

         * Genera el archivo pdf y lo muestra en el browser, del reporte de empleados

         * @param empleados

         * @throws IOException En caso de que haya problemas de generar el archivo de salida PDf que muestra el reporte

         * @throws JRException En caso de haber algún conflicto con la colección recibida por JasperReports para generar el reporte

         */


        public void genera_empleado(List<ReporteEmpleadoDTO> empleados) throws IOException, JRException

        {
```

```
                ByteArrayOutputStream baos = new ByteArrayOutputStream();

                ByteArrayInputStream bis; //Excelente solucion


                InputStream reportStream= new
ClassPathResource("empleados.xml").getInputStream();

                JasperDesign     jasperDesign = JRXmlLoader.load(reportStream);

                JasperReport     jasperReport =
JasperCompileManager.compileReport(jasperDesign);


                JRBeanCollectionDataSource ds = new JRBeanCollectionDataSource(empleados);//
Aqui pasarle la coleccion

                JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, null, ds);

                JasperExportManager.exportReportToPdfStream(jasperPrint, baos);


                bis=new ByteArrayInputStream(baos.toByteArray());

                salidaPdf=bis;



        }


        /**
         * Spring inyecta este objeto con la implementación de
         * EmpleadosRulerImpo
         * @param empleadoRecurso
         */
```

```
public void setEmpleadoRecurso(EmpleadosRuler empleadoRecurso) {
	_empleadoRecurso = empleadoRecurso;
}



/**
 * id de la nómina para procesar y/o generar reportes
 * @return int
 */
public int getIdnomina() {
	return idnomina;
}

/**
 * id de la nómina para procesar y/o generar reportes
 * @param idnomina
 */
public void setIdnomina(int idnomina) {
	this.idnomina = idnomina;
}

/**
 * periodo para procesar la nómina y/o generar reportes
 * @return int
 */
```

```
public int getPeriodo() {
		return periodo;
}
/**
 * periodo para procesar la nómina y/o generar reportes
 * @param periodo
 */

public void setPeriodo(int periodo) {
		this.periodo = periodo;
}

/**
 * id del empleado para generar reportes personalizados
 * @return int
 */
public int getEmpleadoid() {
		return empleadoid;
}

/**
 * id del empleado para generar reportes personalizados
 * @param empleadoid
 */

public void setEmpleadoid(int empleadoid) {
```

```
		this.empleadoid = empleadoid;
}



/**
 * obtiene el encabezado de la forma, este puede ser:
 * <ul>
 * <li> Procesar nómina
 * <li> Reporte de nómina
 * </ul>
 * @return String
 */
public String getLink() {
		return link;
}



public void setLink(String link) {
		this.link = link;
}

/**
 * Obtiene la lista de la nómina procesada para llenar el reporte
 * @return List NominaProcesadaDTO
 */
```

```
public List<NominaProcesadaDTO> getNom() {
		return nom;
}

/**
 * Lista de la nómina procesada para llenar el reporte
 * @param nom
 */
public void setNom(List<NominaProcesadaDTO> nom) {
		this.nom = nom;
}

/**
 * Stream del pdf que se muestra en el browser
 * @return InputStream
 */
public InputStream getSalidaPdf() {
		return salidaPdf;
}
/**
 * Stream del pdf que se muestra en el browser
 * @param salidaPdf
 */

public void setSalidaPdf(InputStream salidaPdf) {
```

```
        this.salidaPdf = salidaPdf;
}

/**
 * Sesion del usuario en el sistema
 * @return Map
 */
public Map getSession() {
        return session;
}

/**
 * Sesion del usuario en el sistema
 * @param session
 */

public void setSession(Map session) {
        this.session = session;
}

/**
 * mensaje de exito enviado al usuario
 * @return String
 */

public String getMensaje() {
```

```
		return mensaje;
}

/**
 *  mensaje de exito enviado al usuario
 * @param mensaje
 */
public void setMensaje(String mensaje) {
		this.mensaje = mensaje;
}

public void setNominaTabulador(NominaTabRuler nominaTabulador) {
		_nominaTabulador = nominaTabulador;
}

public List<NominaDTO> getNominas() {
		return nominas;
}

public void setNominas(List<NominaDTO> nominas) {
		this.nominas = nominas;
}

public List<String> getPeriodos() {
		return periodos;
```

```
	}

	public void setPeriodos(List<String> periodos) {
		this.periodos = periodos;
	}

} package proyecto.terminal.horacio.ruler;

import java.util.Map;

import org.apache.struts2.interceptor.SessionAware;

import proyecto.terminal.horacio.datos.AccesoDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.EmpleadosRuler;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
/**
 * Acción que invalida la sesión del usuario en el sistema
 * @author Horacio Iturbe
 *
 */
public class SalirAction extends ActionSupport implements SessionAware {

	private Map session;
```

```
/**
 * invalida la sesion y regresa a la página principal
 * @return String
 */
public String execute()
{
	getSession().clear();
	return "success";
}



/**
 * Sesión del usuario en el sistema
 * @param session
 */
public void setSession(Map session) {
	this.session = session;
}

/**
 * Sesión del usuario en el sistema
 * @return Map
 */

public Map getSession()
```

```
	{
		return session;
	}
```

```
} package proyecto.terminal.horacio.ruler;

import java.util.ArrayList;
import java.util.List;

import proyecto.terminal.horacio.datos.DatosPersonalesDTO;
import proyecto.terminal.horacio.datos.CamposSuchenDTO;
import proyecto.terminal.horacio.datos.EmpleadoDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.EmpleadosRuler;
import proyecto.terminal.horacio.rulerBase.interfaz.PersonalesRuler;

import com.opensymphony.xwork2.ActionSupport;
```

```
/**
 * Acción que muestra una forma de busqueda para
 * Personas y Empleados, realiza la busqueda según el criterio utilizado
 * y lo muestra al usuario
 * @author Horacio Iturbe
 *
 */
public class SucherAction extends ActionSupport{

	private String campo;
	private String valor; //Id Imss, Nombre,Apaterno
	private PersonalesRuler _personasRecursos;
	private EmpleadosRuler _empleadoRecurso;

	private List<DatosPersonalesDTO> personas;
	private List<CamposSuchenDTO> campos=new ArrayList<CamposSuchenDTO>();

	private List<EmpleadoDTO>empleados = new ArrayList<EmpleadoDTO>();

	private String tipo;
	private String total;

	/**
	 * realiza la consulta según los parametros ingresados en la forma
	 * @return String
```

```
 */
public String consulta()
{

        if(campo.equals("todo"))
                return total;



        if(tipo.equals("persona"))
                personas=_personasRecursos.listarPersonaPorCriterios(campo, valor);//

        if(tipo.equals("empleado"))
                empleados=_empleadoRecurso.listarEmpleadosPorCriterios(campo, valor);







        return tipo;
}

/**
 * carga la forma de busqueda
 * @return String
 */
public String forma()
{
```

```
if(tipo.equals("empleado"))
        total="listar_empleados";

CamposSuchenDTO fields=new CamposSuchenDTO();
fields.setValor("id");
fields.setCampo("ID");
campos.add(fields);

if(tipo.equals("persona"))
{
        total="listar_personas";
        fields=new CamposSuchenDTO();
        fields.setValor("imss");
        fields.setCampo("IMSS");
        campos.add(fields);
}

fields=new CamposSuchenDTO();
fields.setValor("nombre");
fields.setCampo("Nombre");
campos.add(fields);

fields=new CamposSuchenDTO();
```

```
		fields.setValor("apaterno");
		fields.setCampo("Apellido Paterno");
		campos.add(fields);



		return "forma_suchen";
}



/**
 * Spring inyecta este objeto con la implementación de
 * PersonalesRulerImpl
 * @param personasRecursos
 */
public void setPersonasRecursos(PersonalesRuler personasRecursos) {
		_personasRecursos = personasRecursos;

}

/**
 * Personas filtradas según los campos de la forma
 * @return List DatosPersonalesDTO
 */

public List<DatosPersonalesDTO> getPersonas() {
		return personas;
```

```
}

/**
 * Personas filtradas según los campos de la forma
 * @param personas
 */


public void setPersonas(List<DatosPersonalesDTO> personas) {
	this.personas = personas;
}

/**
 * tiene le nombre del campo que será utilizado como criterio
 * para hacer la busqueda en la base de datos
 * @return String
 */
public String getCampo() {
	return campo;
}

/**
 *	Tiene le nombre del campo que será utilizado como criterio
 * para hacer la busqueda en la base de datos
 * @param campo
 */
```

```
public void setCampo(String campo) {
	this.campo = campo;
}



/**
 * Tiene el valor del campo, para ser filtrado
 * @return String
 */
public String getValor() {
	return valor;
}

/**
 * Tiene el valor del campo, para ser filtrado
 * @param valor
 */

public void setValor(String valor) {
	this.valor = valor;
}

/**
 * Colección para cargar el combo de campos
 * @return List CamposSuchenDTO
```

```
 */
public List<CamposSuchenDTO> getCampos() {
	return campos;
}



/**
 * Colección para cargar el combo de campos
 * @param campos
 */
public void setCampos(List<CamposSuchenDTO> campos) {
	this.campos = campos;
}



public String getTipo() {
	return tipo;
}

public void setTipo(String tipo) {
	this.tipo = tipo;
}

/**
 * Parametro usado para determinar si se listan empleados o personas
 * @return String
```

```
 */
public String getTotal() {
	return total;
}

public void setTotal(String total) {
	this.total = total;
}

/**
 * Empleados registrados en el sistema según los criterios seleccionados
 * @return List EmpleadoDTO
 */
public List<EmpleadoDTO> getEmpleados() {
	return empleados;
}

/**
 * Empleados registrados en el sistema según los criterios seleccionados
 * @param empleados
 */
public void setEmpleados(List<EmpleadoDTO> empleados) {
	this.empleados = empleados;
}
/**
 * Spring inyecta este objeto con la implementación de
```

```
	 * EmpleadosRulerImpl
	 * @param empleadoRecurso
	 */

	public void setEmpleadoRecurso(EmpleadosRuler empleadoRecurso) {
		_empleadoRecurso = empleadoRecurso;
	}
} package proyecto.terminal.horacio.ruler;

import java.util.ArrayList;
import java.util.List;

import proyecto.terminal.horacio.datos.TabuladorDTO;
import proyecto.terminal.horacio.rulerBase.interfaz.NominaTabRuler;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
/**
 * Acción para registrar y modificar tabuladores salariales
 * @author Horacio Iturbe
 *
 */
public class TabuladorAction extends ActionSupport implements ModelDriven<TabuladorDTO> {

	private TabuladorDTO tab=new TabuladorDTO();
	private List<TabuladorDTO> tabuladores= new ArrayList<TabuladorDTO>();
```

```
private NominaTabRuler _nominaTabulador;
private int id_tab;
private String mensaje;



/**
 * Carga la forma para registrar Tabulador
 * @return String
 */
public String forma()
{
	return "forma_registro";
}


/**
 * Registra el tabulador
 * @return String
 */
public String registrar()
{
	_nominaTabulador.salvar(tab);
	mensaje=" registrar tabulador";
	return "exito";
}


/**
```

```
 * Carga la forma para modificar el tabulador
 * @return String
 */
public String formaModificar()
{
	tab=_nominaTabulador.tabuladorPorId(id_tab);
	return "forma_modificar";
}

/**
 * Modifica el tabulador en la base de datos
 * @return String
 */
public String modificar()
{

	_nominaTabulador.modificar(tab);
	mensaje=" modificar tabulador";
	return"exito";
}

/**
 * Lista los tabuladores registrados en sistema
 * @return String
 */
public String listado()
```

```
{
        tabuladores=_nominaTabulador.listarTabuladores();
        return "listado";
}

/**
 * Dto para la forma para el registro o modificacion del Tabulador
 * @return TabuladorDTO
 */
public TabuladorDTO getTab() {
        return tab;
}

/**
 * Dto para la forma para el registro o modificacion del Tabulador
 * @param tab
 */

public void setTab(TabuladorDTO tab) {
        this.tab = tab;
}



/**
 * Spring inyecta este objeto con la implementación de
 * NominaTabRulerImpl
```

```
 * @param nominaTabulador
 */
public void setNominaTabulador(NominaTabRuler nominaTabulador) {
	_nominaTabulador = nominaTabulador;
}

/**
 * Carga los campos de la forma en TabuladorDTO
 * @return TabuladorDTO
 */
public TabuladorDTO getModel() {
	return tab;
}

/**
 * id del tabulador a modificar
 * @return int
 */

public int getId_tab() {
	return id_tab;
}
/**
 * id del tabulador a modificar
 * @param id
 */
```

```
public void setId_tab(int id) {
	this.id_tab = id;
}



/**
 * Lista de tabuladores registrados en el sistema
 * @return List TabuladorDTO
 */
public List<TabuladorDTO> getTabuladores() {
	return tabuladores;
}

/**
 * Lista de tabuladores registrados en el sistema
 * @param tabuladores
 */
public void setTabuladores(List<TabuladorDTO> tabuladores) {
	this.tabuladores = tabuladores;
}

/**
 * Mensaje de éxito enviado al usuario
 * @return String
 */
```

```
	public String getMensaje() {
		return mensaje;
	}

	/**
	 * Mensaje de éxito enviado al usuario
	 * @param mensaje
	 */
	public void setMensaje(String mensaje) {
		this.mensaje = mensaje;
	}

} package proyecto.terminal.horacio.rulerBase.impl;

import java.util.ArrayList;
import java.util.List;

import org.hibernate.*;

import proyecto.terminal.horacio.datos.*;
import proyecto.terminal.horacio.rulerBase.interfaz.*;

/**
 * Se encarga de  lo relacionado con el registro y modificacion del DTO EmpleadoDTO e IncidenciaDTO
```

```
 * y del listado
 * @author Horacio Iturbe
 * @see EmpleadoDTO
 * @see DatosPersonalesDTO
 * @see NominaProcesadaDTO
 */

public class EmpleadosRulerImpl implements EmpleadosRuler{

	/**
	 * Objeto para interacción con la base de datos
	 */
	private SessionFactory _sessionFactory;

	 /**
	  * Lista las personas que no han sido registradadas como empleados
	  */
	public List<DatosPersonalesDTO> listarPersonasNoEmpleado()
	{
		Session session=getSession();
		Query query=session.createQuery("SELECT t FROM DatosPersonalesDTO t where
t.id NOT IN (SELECT e.id_persona FROM EmpleadoDTO e)");
		return query.list();
	}

	/**
```

```
	 * Lista los empleados registrados en el sistema
	 */
	public List<EmpleadoDTO> listarEmpleados()
	{
		Session session=getSession();
		Query query=session.createQuery("SELECT e FROM EmpleadoDTO e");
		return query.list();
	}

	/**
	 * Lista los empleados segun el criterio
	 * @param campo Campo para filtrar, puede ser Id, nombre
	 * @param valor Valor asignado al campo
	 */
	public List<EmpleadoDTO >listarEmpleadosPorCriterios(String campo,String valor)
	{

		Session session=getSession();
		int tmp;
		Query query;

			if(campo.equals("id"))
			{
				tmp=Integer.parseInt(valor);
				query=session.createQuery("SELECT t FROM EmpleadoDTO t
where t."+campo+"=?");
```

```
                query.setInteger(0, tmp);
            }
                else
                    query=session.createQuery("SELECT t FROM EmpleadoDTO
t where t.persona."+campo+" LIKE '%"+valor+"%'");

        return query.list();

    }
    /**
     * Lista la nomina procesada
     * @param nomina identificador de la nómina que se proceso
     * @param periodo Periodo que fue procesado
     */
    public List<NominaProcesadaDTO> listarNominaProcesada(int nomina,int periodo)
    {
        Session session=getSession();
        Query query=session.createQuery("SELECT e FROM NominaProcesadaDTO e
where e.id_nomina=? and e.periodo_procesado=? order by e.id_empleado,e.tipo asc");
        query.setInteger(0, nomina);
        query.setInteger(1, periodo);
        return query.list();
    }

    public List<String> listarPeriodosProcesados(int nomina)
    {
```

```
		Session session=getSession();

		Query query=session.createQuery("SELECT e.periodo_procesado FROM
NominaProcesadaDTO e where e.id_nomina=? group by e.periodo_procesado");

		query.setInteger(0, nomina);

		return query.list();

	}


	/**

	 * Lista los empleados por nómina

	 * @param nomina Identificador de la nómina para filtrar los empleados

	 */

	public List<EmpleadoDTO> listarEmpleadosPorNomina(int nomina)

	{

		Session session = getSession();

		Query query=session.createQuery("SELECT t FROM EmpleadoDTO t WHERE
t.id_nomina=?");

		query.setInteger(0, nomina);


		return query.list();

	}


	/**

	 * Lista las incidencias registradas en la base de datos

	 */

	public List<RegistroIncidenciaDTO> listarIncidencias()
```

```
	{
		Session session = getSession();
		Query query=session.createQuery("SELECT t FROM RegistroIncidenciaDTO t");
		System.out.println("generando Incidencias");
		return query.list();
	}

	/**
	 * Lista los empleados por nómina e identificador,
	 * @param nomina Identificador de la nómina
	 * @param empleadoid Identificador del empleado
	 */
	public List<EmpleadoDTO> listarEmpleadosPorNominaPorId(int nomina,int empleadoid) {
		Session session = getSession();
		Query query=session.createQuery("SELECT t FROM EmpleadoDTO t WHERE
t.id_nomina=? and t.id=?");
		query.setInteger(0, nomina);
		query.setInteger(1, empleadoid);
		return query.list();
	}

	/**
	 * Lista los empleados para ser mostrados en el reporte de empleados
	 * @param nomina Identificador de la nómina para consultar el reporte
	 */
```

```
	public List<ReporteEmpleadoDTO> listarReporteEmpleado(int nomina)

	{

		Session session = getSession();

		Query query=session.createQuery("SELECT t FROM ReporteEmpleadoDTO t
WHERE t.id_nomina=?");

		query.setInteger(0, nomina);

		return query.list();



	}

	/**

	 * Usado para generar el reporte de Empleado por id

	 * @param nomina Identificador de la nómina asociada al empleado

	 * @param empleadoid Identificador del empleado el cual se quiere hacer el reporte

	 */

	public List<ReporteEmpleadoDTO> listarReporteEmpleadoPorId(int nomina,int
empleadoid)

	{

		Session session = getSession();

		Query query=session.createQuery("SELECT t FROM ReporteEmpleadoDTO t
WHERE t.id_nomina=? and t.id_empleado=?");

		query.setInteger(0, nomina);

		query.setInteger(1, empleadoid);

		return query.list();

	}



	/**

	 * Lista la nomina procesada para generar el reporte
```

```
	 * @param nomina Identificador de la nomina
	 * @param periodo Periodo que fue procesado
	 * @param empleadoid Identificador del empleado
	 */

	public List<NominaProcesadaDTO> listarNominaProcesadaPorEmpleado(int nomina, int periodo, int empleadoid) {
		Session session=getSession();
		Query query=session.createQuery("SELECT e FROM NominaProcesadaDTO e where e.id_nomina=? and e.periodo_procesado=? and e.id_empleado=?");
		query.setInteger(0, nomina);
		query.setInteger(1, periodo);
		query.setInteger(2, empleadoid);

		return query.list();
	}

	/**
	 * Ejecuta el Stored Procedure en la base de datos para procesar la nómina
	 * @param nomina Identificadore de la nómina a procesar
	 * @param periodo Periodo a procesar
	 */
	public void procesarNomina(int nomina,int periodo)
	{
		Session session=getSession();
		Query query=session.getNamedQuery("procesar_nomina");
```

```
                query.setInteger("idnomina", nomina);
                query.setInteger("periodo", periodo);
                boolean a;
                //Cuando se ejecuta el stored procedura, no registro nada de regreso y me regresa un nulo
                try{
                        query.list();
                        a=false;
                }
                catch(NullPointerException e)
                {
                        a=true;
                }
        }

        /**
         * Valida en base de datos que el usuario exista en la base. Regresa el DTO en caso de existir
         *
         */
        public AccesoDTO isUsuario(String usr,String pss)
        {
                Session session=getSession();
                Query query=session.createQuery("SELECT t FROM AccesoDTO t where t.identificacion=? and t.codigo=?");
                query.setString(0, usr);
                query.setString(1, pss);
```

```
		List<AccesoDTO> li=query.list();
		return li.isEmpty()? null: li.get(0);
}
/**
 * Filtra incidencia por id
 * @param id Identificador de la incidencia
 */
public RegistroIncidenciaDTO incidenciaPorId(int id)
{
		Session session=getSession();
		return (RegistroIncidenciaDTO)session.get(RegistroIncidenciaDTO.class, id);
}

/**
 * Empleado por id
 * @param id Identificador del empleado a consultar
 */
public EmpleadoDTO empleadoPorId(int id)
{
		Session session=getSession();
		return (EmpleadoDTO)session.get(EmpleadoDTO.class, id);
}

 /**
  * Registra en base de datos el objeto enviado
```

```
 * @param obj Puede ser EmpleadoDTO,RegistroIncidenciaDTO
 */
public void salvar(Object obj)
{
        Session session=getSession();
        session.save(obj);
}



/**
 * Modifica en base de datos el objeto enviado
 * @param obj Puede ser EmpleadoDTO,RegistroIncidenciaDTO
 */
public void modificar(Object obj)
{
        Session session=getSession();
        session.update(obj);
}


/**
 * Método especifico para modificar incidencias
 */
public void modificarIncidencias(Object obj)
{
        Session session=getSession();
        session.merge(obj);
```

```
}

/**
 *  Objeto utilizado por Hibernate para interacción con la base de datos
 * @param sessionFactory
 */
public void setSessionFactory( SessionFactory sessionFactory )
 {
   _sessionFactory = sessionFactory;
 }

 /**
  *
  * Obtiene la session actual con la base de datos
  */
 private Session getSession()
 {
   return _sessionFactory.getCurrentSession();
 }
```

```
} package proyecto.terminal.horacio.rulerBase.impl;


import java.util.ArrayList;

import java.util.List;


import org.hibernate.*;

import proyecto.terminal.horacio.datos.*;

import proyecto.terminal.horacio.rulerBase.interfaz.*;

/**

 * Se encarga de todo lo relacionado con los DTO's NominaDTO y TabuladorDTO

 * @author Horacio Iturbe

 * @see TabuladorDTO

 * @see NominaDTO

 *

 */
```

```
public class NominaTabRulerImpl implements NominaTabRuler{

	/**
	 * Objeto que interacciona con la base de datos
	 */
	private SessionFactory _sessionFactory;

	/**
	 * Carga los tabuladores registrados en el sistema
	 */
	 public List<TabuladorDTO> listarTabuladores()
	 {
		Session session=getSession();
		Query query=session.createQuery("SELECT t FROM TabuladorDTO t");
		return query.list();
	}

	 /**
	  * Carga las nominas registradas en el sistema
	  */
	 public List<NominaDTO> listarNominas()
	 {
		Session session=getSession();
		Query query=session.createQuery("SELECT t FROM NominaDTO t");
		return query.list();
	 }
```

```
/**
 * Registra a base de datos
 */
public void salvar(Object obj)
{
	Session session=getSession();
	session.save(obj);
}

/**
 * Regresa TabuladorDTO registrado en la base de datos en caso de existir
 * de  lo contrario regresa null
 * @param id Identificador del Tabulador a ser cargado
 */

public TabuladorDTO tabuladorPorId(int id)
{
	Session session= getSession();
	return (TabuladorDTO)session.get(TabuladorDTO.class, id);
}

/**
 * Regresa NominaDTO registrado en la base de datos en caso de existir
 * de  lo contrario regresa null
 * @param id Identificador de la nómina a ser cargada
```

```
 */
 public NominaDTO nominaPorId(int id)
 {
            Session session=getSession();
            return (NominaDTO)session.get(NominaDTO.class, id);
 }
 /**
  * Modifica el DTO en la base de datos
  */
public void modificar(Object obj)
{
            Session session=getSession();
            session.merge(obj);
}

/**
 *  Objeto utilizado por Hibernate para interacción con la base de datos
 * @param sessionFactory
 */

 public void setSessionFactory( SessionFactory sessionFactory )
  {
    _sessionFactory = sessionFactory;
  }
  /**
   * Regresa la sesion activa en la base de datos
```

```
     *
     */
    private Session getSession()
    {
      return _sessionFactory.getCurrentSession();
    }
```

```
} package proyecto.terminal.horacio.rulerBase.impl;


import java.util.ArrayList;
import java.util.List;
import org.hibernate.*;
import proyecto.terminal.horacio.datos.*;
import proyecto.terminal.horacio.rulerBase.interfaz.*;
/**
 * Se encarga de todo lo relacionado con la operación del DTO DatosPersonalesDTO en la base de datos
 * @author Horacio Iturbe
 * @see DatosPersonalesDTO
 *
 */
```

```
public class PersonalesRulerImpl implements PersonalesRuler{

	/**
	 * Objeto que interacciona con la base de datos
	 */
	private SessionFactory _sessionFactory;



	/**
	 * Carga el catálogo de idiomas desde la base de datos
	 */
	public List<IdiomasDTO> listarIdiomas() {
		Session session = getSession();
		Query query = session.createQuery( "SELECT t FROM IdiomasDTO t where t.id!=-1"
);
		return query.list();
	}


	/**
	 * Carga el catálogo del niveles academicos de la base de datos
	 */
	public List<NivelesAcademicosDTO> listarNivelesAcademicos() {
```

```
		Session session = getSession();
		Query query = session.createQuery( "SELECT t FROM NivelesAcademicosDTO t" );
		return query.list();
	}

	/**
	 * Carga la lista con las personas registradas en el sistema
	 */
	public List<DatosPersonalesDTO> listarPersonas() {

		Session session=getSession();
		Query query=session.createQuery("SELECT t FROM DatosPersonalesDTO t");
		return query.list();
	}

	/**
	 * Carga la lista con las personas registrada en el sistema, mientras cumplan el criterio
	 * de busqueda
	 */
	public List<DatosPersonalesDTO> listarPersonaPorCriterios(String campo, String
valor) {

		Session session=getSession();
		int tmp;
		Query query;
```

```
                        if(campo.equals("id"))
                        {
                                tmp=Integer.parseInt(valor);
                                query=session.createQuery("SELECT t FROM DatosPersonalesDTO
t where t."+campo+"=?");
                                query.setInteger(0, tmp);
                        }
                                else
                                        query=session.createQuery("SELECT t FROM
DatosPersonalesDTO t where t."+campo+" LIKE '%"+valor+"%'");

                return query.list();
        }

        /**
         * Obtiene los datos personales de la persona
         * @param id Identificador de la persona a obtener información
         */
        public DatosPersonalesDTO personaPorId(int id)
        {
                Session session=getSession();
                DatosPersonalesDTO
persona=(DatosPersonalesDTO)session.get(DatosPersonalesDTO.class, id);

                return persona;

        }
```

```
/**
 * Realiza la modificación en la base de datos de la persona
 */
public void modificarPersona(DatosPersonalesDTO obj)
{
	Session session = getSession();
	session.merge(obj);



}

/**
 * Realiza el registro de la persona en la base de datos
 */
public void salvar(Object obj)
{
	Session session=getSession();
	session.save(obj);


}



/**
 * Objeto utilizado por Hibernate para interacción con la base de datos
 * @param sessionFactory
 */
```

```
    public void setSessionFactory( SessionFactory sessionFactory )
    {
      _sessionFactory = sessionFactory;
    }

    /**
     * Regresa la sesión actual en la base de datos
     *
     */
     private Session getSession()
     {
       return _sessionFactory.getCurrentSession();
     }
```

```
} package proyecto.terminal.horacio.rulerBase.interfaz;

import java.util.List;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;
```

```
import proyecto.terminal.horacio.datos.*;

/**

 * Define los métodos necesarios para la interaccion de la base de datos sobre lo relacionado con EmpleadoDTO

 * @author Horacio

 *

 */

@Transactional(rollbackFor=Exception.class,propagation=Propagation.REQUIRED)

public interface EmpleadosRuler {



	public List<DatosPersonalesDTO> listarPersonasNoEmpleado();

	public List<EmpleadoDTO> listarEmpleados();


	public List<EmpleadoDTO> listarEmpleadosPorNomina(int nomina);

	public List<EmpleadoDTO> listarEmpleadosPorNominaPorId(int nomina,int empleadoid);


	public List<EmpleadoDTO> listarEmpleadosPorCriterios(String campo,String valor);


	public RegistroIncidenciaDTO incidenciaPorId(int id);

	public List<RegistroIncidenciaDTO> listarIncidencias();



	public EmpleadoDTO empleadoPorId(int id);

	public void salvar(Object obj);

	public void modificar(Object obj);
```

```
	public void modificarIncidencias(Object obj);



	public List<NominaProcesadaDTO> listarNominaProcesada(int nomina,int periodo);

	public List<String> listarPeriodosProcesados(int nomina);


	public List<NominaProcesadaDTO> listarNominaProcesadaPorEmpleado(int nomina,int
periodo,int empleadoid);


	public List<ReporteEmpleadoDTO> listarReporteEmpleado(int nomina);

	public List<ReporteEmpleadoDTO> listarReporteEmpleadoPorId(int nomina,int
empleadoid);


	public void procesarNomina(int nomina,int periodo);


	public AccesoDTO isUsuario(String usr,String pss);




} package proyecto.terminal.horacio.rulerBase.interfaz;


import java.util.List;


import org.springframework.transaction.annotation.Propagation;

import org.springframework.transaction.annotation.Transactional;
```

```
import proyecto.terminal.horacio.datos.*;

/**

 * Define los métodos necesarios para la interacción de la base de datos sobre lo relacionado con NominadDTO y TabuladorDTO

 * @author Horacio Iturbe

 *

 */

@Transactional(rollbackFor=Exception.class,propagation=Propagation.REQUIRED)

public interface NominaTabRuler {

	public List<TabuladorDTO> listarTabuladores();

	public List<NominaDTO> listarNominas();

	public TabuladorDTO tabuladorPorId(int id);

	public NominaDTO nominaPorId(int id);

	public void salvar(Object obj);

	public void modificar(Object obj);

} package proyecto.terminal.horacio.rulerBase.interfaz;
```

```
import java.util.List;


import org.springframework.transaction.annotation.Propagation;

import org.springframework.transaction.annotation.Transactional;

import proyecto.terminal.horacio.datos.*;

/**

 * Define los métodos necesarios para la interacción de la base de datos sobre lo relacionado con DatosPersonalesDTO

 * @author Horacio Iturbe

 *

 */

@Transactional(rollbackFor=Exception.class,propagation=Propagation.REQUIRED)

public interface PersonalesRuler {


	public List<DatosPersonalesDTO> listarPersonas();

	public List<IdiomasDTO> listarIdiomas();

	public List<NivelesAcademicosDTO> listarNivelesAcademicos();

	public void salvar(Object obj);

	public DatosPersonalesDTO personaPorId(int id);

	public void modificarPersona(DatosPersonalesDTO obj);

	public List<DatosPersonalesDTO> listarPersonaPorCriterios(String campo, String valor);




}
```