USTA Empresarial The Alirios

ESPECIFICACIÓN FUNCIONAL

Finanzas

Versión 1.0

Enero 22 de 2016

|  |  |  |  |
| --- | --- | --- | --- |
| **REGISTRO DE CAMBIOS AL DOCUMENTO** | | | |
|  | | | |
| **Versión** | **Fecha** | **Descripción del cambio** | **Autor** |
| 1 | 6 mayo 2019 | Creación del documento | Juan Gomez |
| 1 | 6 mayo 2019 | Creación del DDL | The Alirios |
| 2 | 21 mayo 2019 | Creaciòn del Ejb de la tabla CRM | The Alirios |
| 2 | 21 mayo 2019 | Correcciòn de errores de la tabla Seguridad | Nicolle Saavedra  Mafe |
|  |  |  |  |

**TABLA DE CONTENIDO**

INTRODUCCIÓN

1. DESCRIPCIÓN DE SERVICIOS

1.1. Plan contable

1.1.1. Parámetros de entrada (especificación)

1.1.2. Retorno

1.2 Concepto

1.2.1 Parámetros de entrada (especificación)

1.3 Libro Mayor

1.3.1 Parámetros de entrada (especificación)

1.4 Libro Diario

1.4.1 Parámetros de entrada (especificación)

1.5 Periodo

1.5.1 Parámetros de entrada (especificación)

**INTRODUCCIÓN**

Se expone el servicio que presenta el módulo para el usuario que consiste en la creación, eliminación, listar y actualizar en relación con cada uno de los registros que presente como identificación del usuario, tipo de usuario que se compone por un tabla aparte con diferentes especificaciones, nombre del usuario, apellido del usuario y el respectivo documento del usuario

Se crea el lenguaje de definición de datos, más conocido como DDL del grupo finanzas.

1. **DESCRIPCIÓN DEL SERVICIO**
   1. Plan contable
      1. **Parámetros de entrada**

* **Código:** Es un valor establecido por Plan único de cuentas más conocido por PUC; este número depende del tipo de la cuenta que se clasifica en Clase, Grupo, Cuenta y Subcuenta.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo** | **Clase** | **Tamaño de caracteres** | **Ejemplo** |
| 1 | Clase | 1 | 2 (Pasivo) |
| 2 | Grupo | 2 | 11 (Disponible) |
| 3 | Cuenta | 4 | 2205 (Proveedores) |
| 4 | Subcuenta | 6 | 130505 (Nacionales) |

* **Nombre:** Es un valor que está compuesto únicamente por caracteres de longitud máxima de 100 caracteres siendo un campo obligatorio. El cual representa los nombre de la cuenta.
* **Descripción:** Es un valor que está compuesto únicamente por caracteres de longitud máxima de 300 caracteres siendo un campo obligatorio. El cual representa una explicación o definición de la cuenta.
* **Tipo:** Valor caracter con un tamaño máximo de 1 dígito que indica el tipo de PUC y los tipos son:

|  |  |
| --- | --- |
| **Tipo** | **Clase** |
| 1 | Clase |
| 2 | Grupo |
| 3 | Cuenta |
| 4 | Subcuenta |

* **Vigencia:** Valor carácter con un tamaño máximo de 10 dígitos que indica la fecha de inicio de vigencia.
* **Código padre:** Es el número que representa una subcuenta derivada de una otra cuenta denominada padre y a su vez pertenece al grupo de padre.

**EJEMPLO:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Código** | **Nombre** | **Descripción** | **Código Padre** | **Tipo** | **Vigencia** |
| **0** | **ROOT** | **\*\*\*\*\*** | **Null** | **0** | **2020-12-31** |
| **1** | **Activos** | **\*\*\*\*\*** | **0** | **1** | **2020-12-31** |
| **11** | **Disponible** | **\*\*\*\*\*** | **1** | **2** | **2020-12-31** |
| **1105** | **Caja** | **\*\*\*\*\*** | **11** | **3** | **2020-12-31** |
| **110505** | **Caja General** | **\*\*\*\*\*** | **1105** | **4** | **2020-12-31** |
| **110510** | **Cajas Menores** | **\*\*\*\*\*** | **1105** | **4** | **2020-12-31** |

* 1. desidoSe creó la tabla plan Concepto la cual cuenta con los atributos mencionados y descritos en la parte de abajo.
     1. **Parámetros de entrada**
* **CODIGO:** Primary key de la tabla concepto de tipo serial. Valores incrementales
* **NOMBRE:** Atributo de tipo character varying con una longitud de 20 en el que se especifican los nombres de los conceptos .Este atributo no puede ser nulo
* **DESCRPCION:** Atributo de tipo character varying con una longitud de 100 donde se alMAcena la descripciòn de cada concepto. Este atributo puede ser nulo.
* **TIPO:** Atributo de tipo boolean no nulo donde se define el tipo de cargo y sus valores son: (TRUE) Debe (FALSE) Haber.
* **ESTADO:** Atributo de tipo boolean no nulo donde se guarda el estado de actividad (TRUE) o inactividad (FALSE) del concepto.
* **PLAN\_CONTABLE\_CODIGO:** Llave foránea heredada de la tabla plan contable. Esta llave cuenta con la mismas características de la llave primaria (codigo) de la tabla plan contable.

**Ejemplos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CODIGO** | **NOMBRE** | **DESCRPCION** | **TIPO** | **ESTADO** | **PLAN\_CONTABLE\_CODIGO** |
| 1 | Prueba 1 | Pago de Bebidas hidratantes | TRUE | TRUE | 2205 |
| 2 | Prueba 1 | Pago de Bebidas hidratantes | FALSE | TRUE | 1105 |
| 3 | Prueba 2 | Pago de factura No. 034 | TRUE | TRUE | 1110 |
| 4 | Prueba 2 | Pago de factura No. 034 | FALSE | TRUE | 1305 |

* **DML de la tabla plan contable**

CREATE TABLE "FINANZAS".CONCEPTO

(

CODIGO SERIAL NOT NULL PRIMARY KEY,

NOMBRE CHARACTER VARYING(20) NOT NULL,

DESCRPCION CHARACTER VARYING(100),

TIPO BOOLEAN NOT NULL,

ESTADO BOOLEAN NOT NULL,

PLAN\_CONTABLE\_CODIGO INTEGER NOT NULL

);

ALTER TABLE "FINANZAS".CONCEPTO

ADD CONSTRAINT "FK\_PLANCONTABLE\_CONCEPTO\_PL" FOREIGN KEY (PLAN\_CONTABLE\_CODIGO)

REFERENCES "FINANZAS".PLAN\_CONTABLE(CODIGO);

COMMENT ON TABLE "FINANZAS".CONCEPTO

IS 'Tabla donde se almacenan los datos de cada cargo o los CONCEPTOS ';

COMMENT ON COLUMN "FINANZAS".CONCEPTO.CODIGO

IS 'Llave primaria de la tabla CONCEPTO, CODIGO del cargo';

COMMENT ON COLUMN "FINANZAS".CONCEPTO.DESCRPCION

IS 'Funciones del cargo ';

COMMENT ON COLUMN "FINANZAS".CONCEPTO.TIPO

IS 'Define el TIPO del cargo y Los valores son: (1)Debe (0)Haber';

COMMENT ON COLUMN "FINANZAS".CONCEPTO.ESTADO

IS 'Este atributo define el ESTADO de actividad o inactividad del cargo donde: (1) Activo (0) Inactivo';

COMMENT ON COLUMN "FINANZAS".CONCEPTO.PLAN\_CONTABLE\_CODIGO

IS 'CODIGO de la tabla plan contable';

COMMENT ON COLUMN "FINANZAS".CONCEPTO.NOMBRE

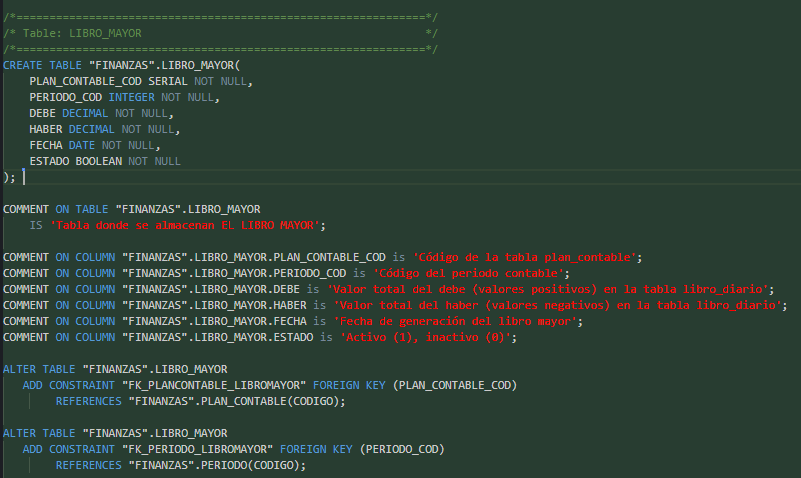
IS 'NOMBRE del cargo';

**1.3** **Tabla Libro mayor**

**1.3.1 Parámetros de entrada**

* **plan\_contable\_cod:** Código heredado de la tabla Código de la tabla plan\_contable. Llave foránea y primaria.
* **periodo\_cod:** Código heredado de la tabla de la tabla periodo. Llave foránea y primaria.
* **debe:** Campo entero con valores positivos. Valor total del campo **debe** en la tabla libro\_diario.
* **haber:** Campo entero con valores positivos y negativos. Valor total del campo **haber** en la tabla libro\_diario
* **fecha:** Campo de tipo *Date*. Fecha de generación del libro mayor.
* **estado:** Campo de tipo *boolean.* Activo (1) si el un registro está vigente, inactivo (0) si es eliminado.

Código DDL de la tabla:

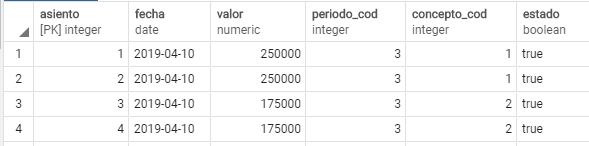


**1.4** Tabla Libro diario

**1.4.1 Parámetros de entrada**

* **Asiento:** Código secuencial del movimiento contable (la anotación de un hecho económico, en el libro diario ), llave primaria.
* **Fecha:** Campo de tipo *Date*, Fecha del movimiento contable.
* **Valor:** Valor de transacciones realizadas por una empresa (asiento contable)
* **Periodo\_cod:** Código heredado de la tabla periodo, llave foráne**a.**
* **Concepto\_cod:** Código heredado de la tabla concepto, llave foráne**a.**
* **Estado:** Atributo de tipo boolean no nulo donde se guarda el estado de actividad (TRUE) o inactividad (FALSE) del concepto.

EJEMPLO:



* **DDL de la tabla libro diario.**

CREATE TABLE "FINANZAS".LIBRO\_DIARIO(

ASIENTO SERIAL PRIMARY KEY NOT NULL,

FECHA DATE NOT NULL,

VALOR DECIMAL NOT NULL,

PERIODO\_COD INTEGER NOT NULL,

CONCEPTO\_COD INTEGER NOT NULL,

ESTADO BOOLEAN NOT NULL

);

COMMENT ON TABLE "FINANZAS".LIBRO\_DIARIO

IS 'Tabla donde se almacenan EL LIBRO DIARIO ';

COMMENT ON COLUMN "FINANZAS".LIBRO\_DIARIO.ASIENTO IS 'Código secuencial del movimiento contable';

COMMENT ON COLUMN "FINANZAS".LIBRO\_DIARIO.FECHA IS 'Fecha del movimiento contable';

COMMENT ON COLUMN "FINANZAS".LIBRO\_DIARIO.VALOR IS 'Valor del asiento contable';

COMMENT ON COLUMN "FINANZAS".LIBRO\_DIARIO.PERIODO\_COD IS 'Código del periodo contable';

COMMENT ON COLUMN "FINANZAS".LIBRO\_DIARIO.CONCEPTO\_COD IS 'Código del concepto contable';

COMMENT ON COLUMN "FINANZAS".LIBRO\_DIARIO.ESTADO IS 'Activo (1), inactivo(0)';

ALTER TABLE "FINANZAS".LIBRO\_DIARIO

ADD CONSTRAINT "Fk\_LIBRODIARIO\_PERIODO" FOREIGN KEY (PERIODO\_COD)

REFERENCES "FINANZAS".PERIODO(CODIGO);

ALTER TABLE "FINANZAS".LIBRO\_DIARIO

ADD CONSTRAINT "Fk\_LIBRODIARIO\_CONCEPTO" FOREIGN KEY (CONCEPTO\_COD)

REFERENCES "FINANZAS".CONCEPTO(CODIGO);

**1.5 Tabla Periodo**

**1.5.1 Parámetros de entrada**

Se crea la tabla periodo que cuenta con los siguientes atributos:

* **CODIGO:** llave primaria de la tabla. Valores incrementales
* **FECHA INICIO:** Atributo de tipo date donde se especifica la fecha de inicio del periodo.
* **FECHA FIN:** Atributo de tipo date donde se especifica la fecha fin del periodo.
* **ESTADO:** Atributo de tipo boolean no nulo donde se define el tipo de estado y sus valores son: TRUE activo, FALSE inactivo.

EJEMPLO:

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **FECHA INICIO** | **FECHA FIN** | **ESTADO** |
| 1 | 13/04/2018 | 12/05/2021 | ACTIVO |
| 2 | 14/01/2018 | 12/05/2020 | INACTIVO |

**DDL DE LA TABLA PERIODO**

CREATE TABLE finanzas.PERIODO

(

CÓDIGO SERIAL PRIMARY KEY NOT NULL ,

FECHA\_INICIO DATE NOT NULL,

FECHA\_FIN DATE NOT NULL,

ESTADO BOOLEAN NOT NULL

);

COMMENT ON COLUMN finanzas.PERIODO.CODIGO IS 'codigo del periodo';

COMMENT ON COLUMN finanzas.PERIODO.FECHA\_INICIO IS 'fecha inicio';

COMMENT ON COLUMN finanzas.PERIODO.FECHA\_FIN IS 'fecha final';

COMMENT ON COLUMN finanzas.PERIODO.ESTADO IS 'Activo (1), inactivo (0)';

**TABLA DE CONTENIDO EJB CRM FACHADA**

INTRODUCCIÓN

2. EJB

2.1. Tabla periodo

2.2. Tabla campania

2.3. Tabla evento

2.4. Tabla medio

2.5. Tabla recurso

2.6. Tabla tipo medio

**INTRODUCCIÓN**

Se crearon las secciones de fachada, been y servicio de CRM. Donde se crean los métodos del CRUD por cada una de las tablas de este parte de el proyecto.

1. **TABLA DE CONTENIDO EJB CRM FACHADA**
   1. **Tabla Periodo**
   2. **Tabla Campaña**

Se crean los siguiente métodos para la tabla recurso en cada una de las secciones:

**Fachada:** Interfaz utilizada para la declaracion de metodos.

* Listar Campaña:



* Crear Campaña:



* Editar Campaña:



* Eliminar Campaña:



* Buscar Campaña por nombre:



**Bean:** En el bean se hace la funcionalidad de los métodos declarados en la fachada.

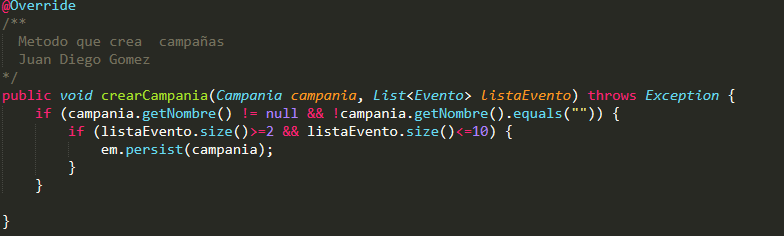
* Listar Campaña:

Método creado para listar todas las campañas publicitarias creadas este método contiene un Throws Exception lo que hace es que captura los posibles errores sobre la ejecución; además de eso utiliza una consulta hecha en el ORM pero se llama desde el modelo de CRM y retorna una lista tipo campania.



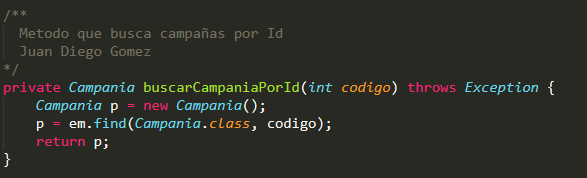
* Crear Campaña:

Método creado que nos permite crear una campaña, tiene como un parámetro un objeto de tipo Campania llamado campania y una lista de eventos, este método no retorna nada y tiene un Throws Exception que permite capturar errores posibles sobre la ejecución.



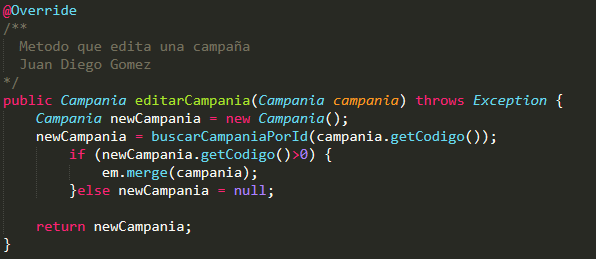
* Buscar Campaña por Id:

Es un método privado ya que solo lo usamos para retornar un objeto tipo campania que tenga el código que entra por parámetro.



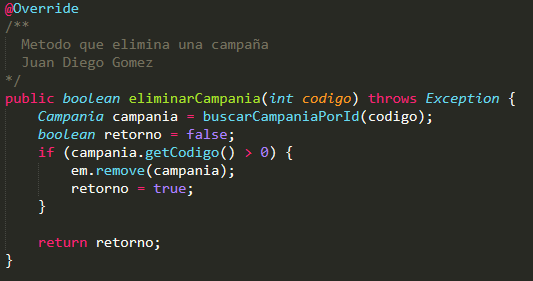
* Editar Campaña:

Este método permite editar una Campaña ya creada, para esto recibe como parámetro un objeto de tipo Campania, retornando un objeto tipo campania que se editado y actualizado. Tiene un Throws Exception que permite capturar errores posibles sobre la ejecución.



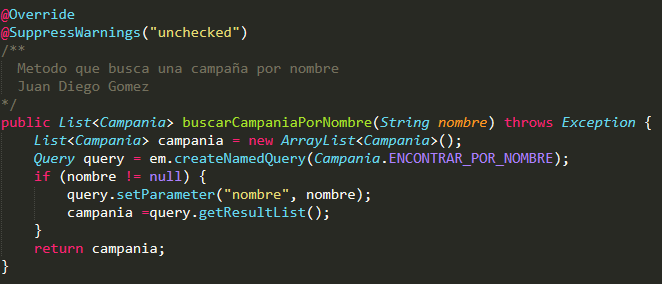
* Eliminar Campaña:

Este método elimina un objeto de tipo Campaña para esto tenemos un parámetro que es el código de la campaña que se quiere eliminar, el método es boolean ya que si lo borra retornara True y false si no es posible realizar la acción requerida y posee un Throws Exception que permite capturar errores posibles sobre la ejecución.



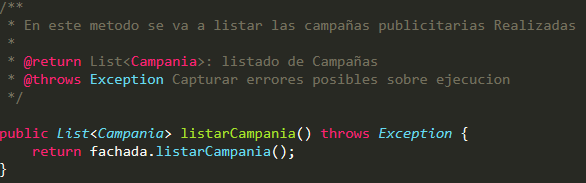
* Buscar Campaña por nombre:

Este método busca una Campaña por el nombre, tiene como parámetro un dato tipo String llamado nombre que será el nombre de la campaña y retorna una lista de campaña que contengan ese nombre y posee un Throws Exception que permite capturar errores posibles sobre la ejecución.

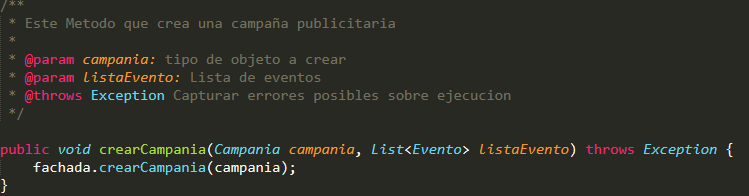


**Servicio:**

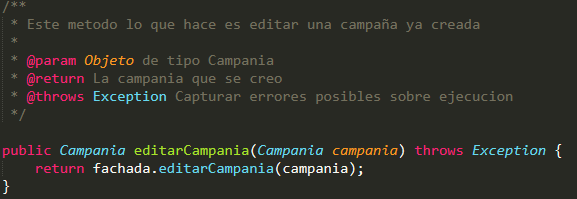
* Listar Campaña:



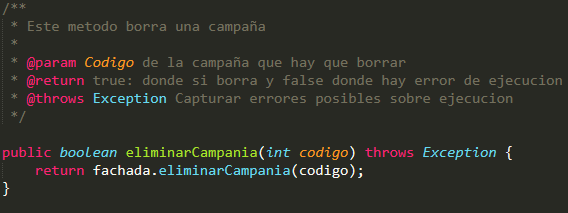
* Crear Campaña:



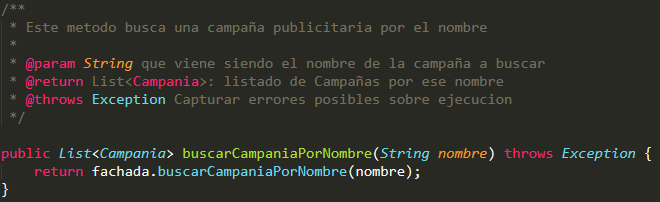
* Editar Campaña:



* Eliminar Campaña:



* Buscar Campaña por nombre:

****

**2.3 Tabla evento**

A Continuación se listan los métodos para cada una de las secciones

Fachada:

* public List<Evento> listarEvento() throws Exception;
* public void crearEvento(Evento evento) throws Exception;
* public Evento editarEvento(Evento evento) throws Exception;
* public boolean eliminarEvento(int codigo) throws Exception;
* public List<Evento> buscarEventoPorNombre(String nombre) throws Exception;

Servicio:

* public List<Evento> listarEvento() throws Exception {

return fachada.listarEvento();

}

* public void crearEvento(Evento evento) throws Exception {

fachada.crearEvento(evento);

}

* public Evento editarEvento(Evento evento) throws Exception {

return fachada.editarEvento(evento);

}

* public boolean eliminarEvento(int codigo) throws Exception {

return fachada.eliminarEvento(codigo);

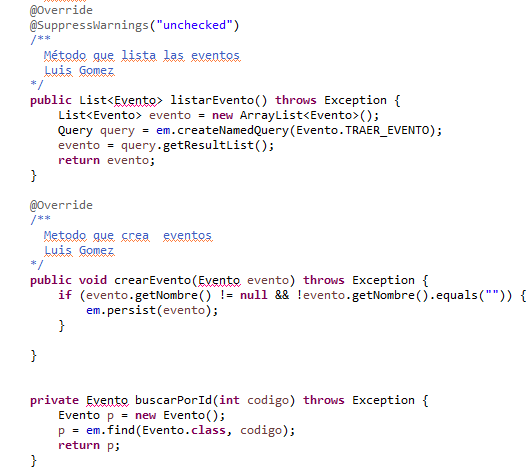
}

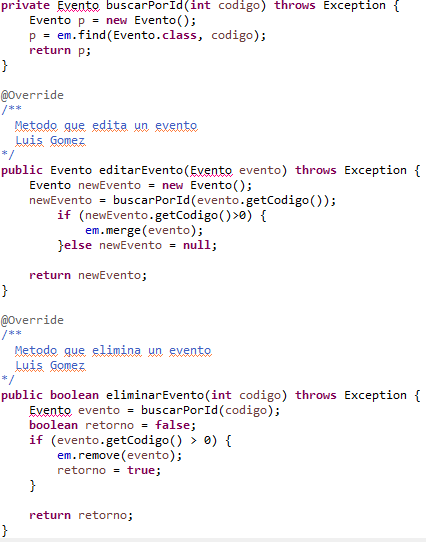
* public List<Evento> buscarEventoPorNombre(String nombre) throws Exception {

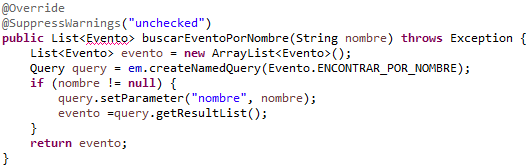
return fachada.buscarEventoPorNombre(nombre);

}

Bean:







Explicación de los métodos:

* listarEvento():

Este método lista los eventos publicitarios realizados. Retorna el listado de eventos y lanza una excepción para capturar errores posibles sobre ejecución.

* crearEvento(Evento evento):

Permite crear un nuevo evento publicitario. Recibe como parámetro el nuevo evento a crear y lanza una excepción para capturar errores posibles sobre ejecución.

* editarEvento(Evento evento):

Permite editar un evento ya creado. Recibe como parámetro el nuevo evento y retorna el evento que ha sido creado. Lanza una excepción para capturar errores posibles sobre ejecución.

* eliminarEvento(int codigo):

Permite eliminar un evento ya creado. Recibe como parámetro el código del evento a eliminar. Retorna **true** si el evento ha sido eliminado correctamente **false** en caso de error. Lanza una excepción para capturar errores posibles sobre ejecución.

* buscarEventoPorNombre(String nombre):

Permite buscar un evento. Recibe como parámetro el nombre del evento a buscar. Retorna una lista con los eventos que coinciden con la búsqueda. Lanza una excepción para capturar errores posibles sobre ejecución.

**2.4 Tabla medio**

Se crean los siguiente métodos para la tabla medio en cada una de las secciones:

Fachada

* public List<Medio> listarMedio() throws Exception
* public void crearMedio(Medio medio) throws Exception
* public Medio editarMedio(Medio medio) throws Exception
* public boolean eliminarMedio(int codigo) throws Exception
* public List<Medio> buscarMedioPorNombre(String nombre) throws Exception

Servicio

* public List<Medio> listarMedio() throws Exception {

return fachada.listarMedio();

}

* public void crearMedio(Medio medio) throws Exception {

fachada.crearMedio(medio);

}

* public Medio editarMedio(Medio medio) throws Exception {

return fachada.editarMedio(medio);

}

* public boolean eliminarMedio(int codigo) throws Exception {

return fachada.eliminarMedio(codigo);

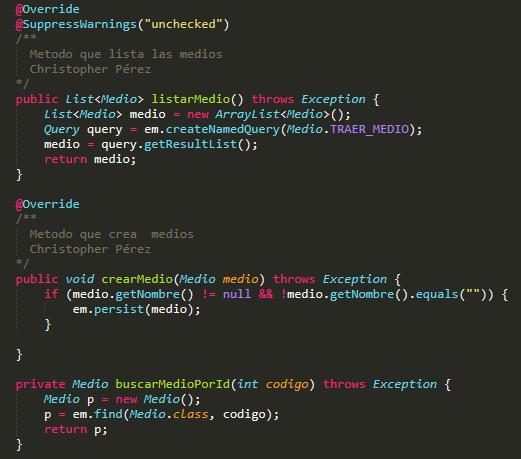
}

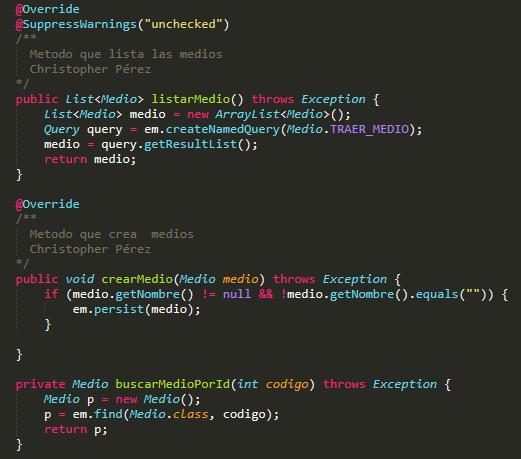
* public List<Medio> buscarMedioPorNombre(String nombre) throws Exception {

return fachada.buscarMedioPorNombre(nombre);

}

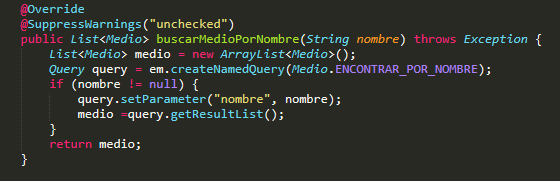
Bean











Explicaciòn de los métodos:

* listarMedio()

En este método permite listar los medios publicitarios utilizados, retorna listado de todos los medios sin pedir ningun parametro y tiene una Excepción que permite capturar errores posibles sobre ejecución

* crearMedio(Medio medio)

Este Método crea un objeto de tipo medio, pide un atributo tipo Medio llamado medio, no retorna nada pero tiene una Excepción que permite capturar errores posibles sobre la ejecución.

* editarMedio(Medio medio)

Este método permite editar un medio ya creado, para esto recibe como parámetro un objeto de tipo Medio, retornando el medio que se editado y actualizado. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

* eliminarMedio(int codigo)

Este método elimina un objeto de tipo Medio para esto pide como parámetro un dato int llamado codigo que representa o equivale al código del medio a eliminar. Si lo borra retornara True y false si no es posible realizar la acción requerida. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

* buscarMedioPorNombre(String nombre)

Este método busca un medio publicitario por el nombre, para esto pide un String que viene siendo el nombre del medio a buscar y retorna una List<Medio> que es el listado de medios por ese nombre.Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

**2.5 Tabla recurso**

Se crean los siguiente métodos para la tabla recurso en cada una de las secciones:

Fachada

* public List<Recurso> listarRecursos() throws Exception;
* public void crearRecurso(Recurso puc) throws Exception;
* public Recurso editarRecurso(Recurso puc) throws Exception;
* public boolean eliminarRecurso(int codigo) throws Exception;
* public Recurso listarRecursoNombre(String nombre) throws Exception;

Servicio

* public List<Recurso> listarRecursos() throws Exception {

return fachada.listarRecursos();

}

* public void crearRecurso(Recurso puc) throws Exception {

fachada.crearRecurso(puc);

}

* public Recurso editarRecurso(Recurso puc) throws Exception {

return fachada.editarRecurso(puc);

}

* public boolean eliminarRecurso(int codigo) throws Exception {

return fachada.eliminarRecurso(codigo);

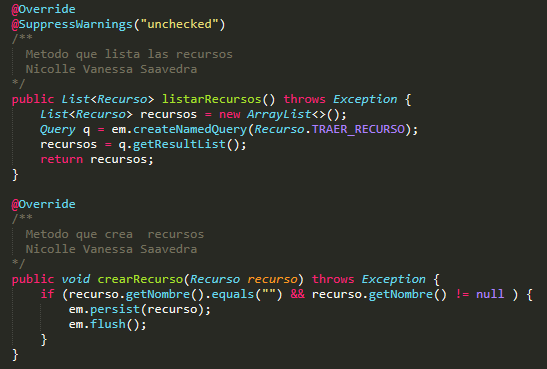
}

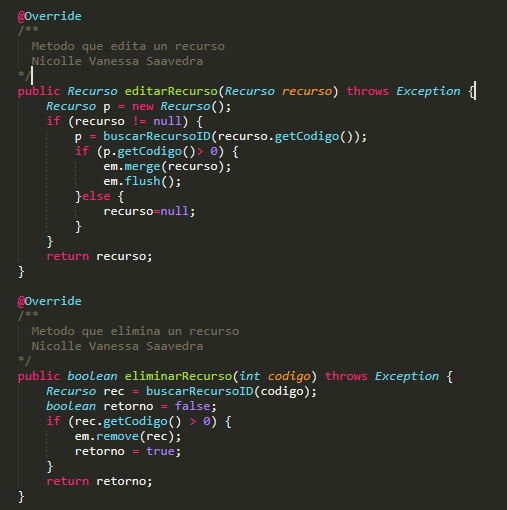
* public Recurso listarRecursoNombre(String nombre) throws Exception {

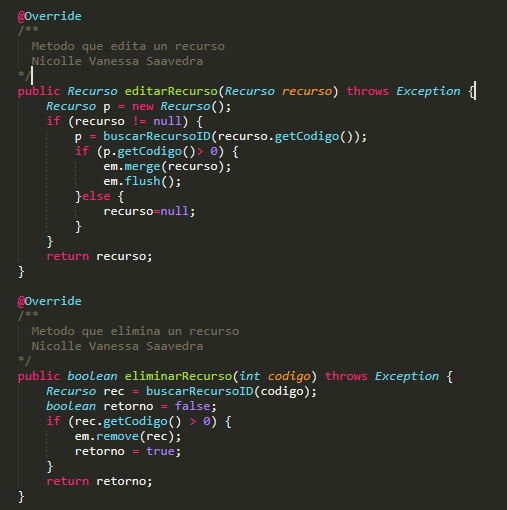
return fachada.listarRecursoNombre(nombre);

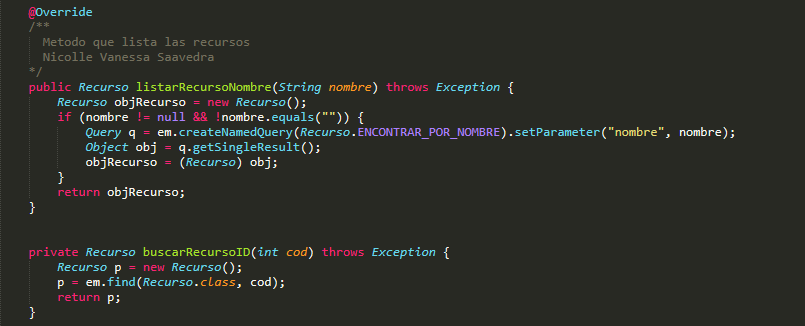
}

Bean









Explicaciòn de los métodos:

* listarRecursos()

Método creado para listar todos los recursos contenidos en la tabla. Contiene una Excepción Captura los posibles errores sobre la ejecución y retorna.

* crearRecurso(Recurso puc)

Método que permite crear un recurso, este método recibe como parámetro un objeto de tipo Recurso llamado puc, este método no retorna nada. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

* editarRecurso(Recurso puc)

Método que permite actualizar un recurso, recibe como parámetro un objeto de tipo recurso y retorna el mismo objeto pero con las modificaciònes realizadas. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

* eliminarRecurso(int codigo)

Método que permite eliminar un recurso, recibe como parámetro un entero que representa el código del recurso y retorna true: Si se borró y false si no se borró. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

* listarRecursoNombre(String nombre)

Método que busca una recurso por nombre, recibe como parámetro un dato de tipo string (nombre) que equivale a el nombre del recurso que va buscar y retorna objeto de tipo recurso. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.

**2.6 Tabla Tipo Medio**

Se crean los siguiente métodos para la tabla Tipo Medio en cada una de las secciones:

**Fachada:**

* public List<TipoMedio> listarTipoMedio() throws Exception;
* public void crearTipoMedio(TipoMedio tipo) throws Exception;
* public TipoMedio editarTipoMedio(TipoMedio tipo) throws Exception;
* public boolean eliminarTipoMedio(int codigo) throws Exception;
* public TipoMedio listarTipoMedioNombre(String nombre) throws Exception;

**Servicio**

* public List<TipoMedio> listarTipoMedio() throws Exception {

return fachada.listarTipoMedio();

}

* public void crearTipoMedio(TipoMedio tipo) throws Exception {

fachada.crearTipoMedio(tipo);

}

* public TipoMedio editarTipoMedio(TipoMedio tipo) throws Exception {

return fachada.editarTipoMedio(tipo);

}

* public boolean eliminarTipoMedio(int codigo) throws Exception {

return fachada.eliminarTipoMedio(codigo);

}

* public TipoMedio listarTipoMedioPorNombre(String nombre) throws Exception {

return fachada.listarTipoMedioPorNombre(nombre);

}

**Bean:** En el bean se hace la funcionalidad de los métodos declarados en la fachada.

Listar Tipo Medio:

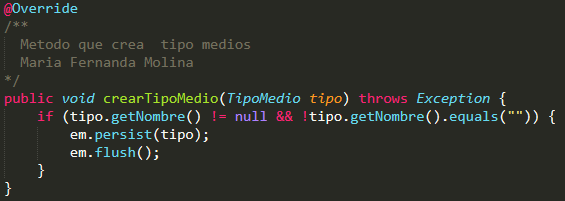
* listarTipoMedio()

Método creado para listar todos los Tipo medio contenidos en la tabla. Contiene una Excepción Captura los posibles errores sobre la ejecución y retorna.



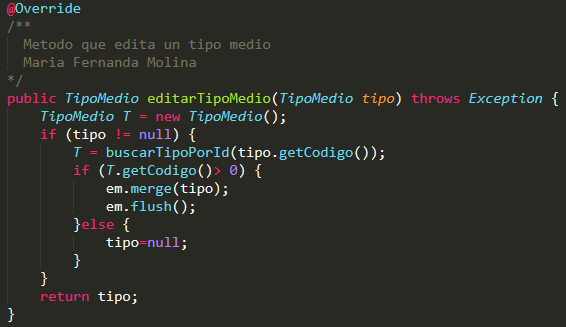
* crearTipoMedio(TipoMedio tipo)

Método que permite crear un Tipo Medio, este método recibe como parámetro un objeto de tipo TipoMedio llamado tipo, este método no retorna nada. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.



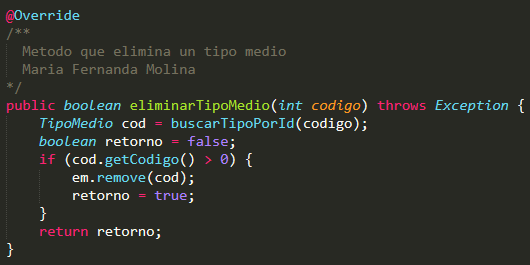
* editarTipoMedio(TipoMedio tipo)

Este método permite editar un Tipo Medio ya creado, para esto recibe como parámetro un objeto de tipo Tipo Medio, retornando el tipo que se editado y actualizado. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.



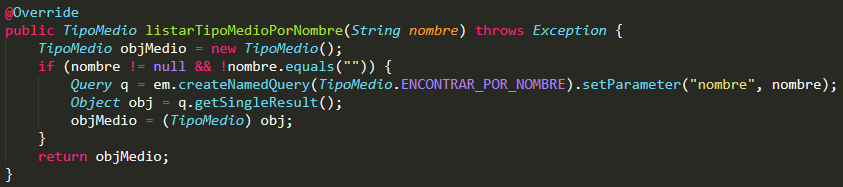
* eliminarTipoMedio(int codigo)

Método que permite eliminar un Tipo Medio, recibe como parámetro un entero que representa el código del Tipo Medio y retorna true: Si se borró y false si no se borró. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.



* listarTipoMedioPorNombre(String nombre)

Método que busca una Tipo Medio por nombre, recibe como parámetro un dato de tipo string (nombre) que equivale a el nombre del Tipo Medio que va buscar y retorna objeto de tipo Tipo Medio. Tiene una Excepción que permite capturar errores posibles sobre la ejecución.



**2.6 Tabla periodo**

**Fachada:**

* public List<Periodo> listarPeriodo() throws Exception;
* public void crearPeriodo(Periodo per) throws Exception;
* public Periodo editarPeriodo(Periodo per) throws Exception;
* public boolean eliminarRegistros(int codigo) throws Exception;

**Servicio**

public List<Periodo> listarPeriodo() throws Exception {

return fachada.listarPeriodo();

}

public void crearPeriodo(Periodo per) throws Exception {

fachada.crearPeriodo(per);

}

public Periodo editarPeriodo(Periodo per) throws Exception {

return fachada.editarPeriodo(per);

}

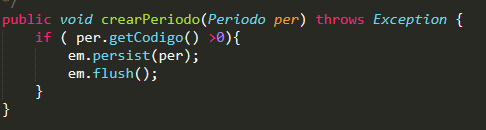
public boolean eliminarRegistros(int codigo) throws Exception {

return fachada.eliminarRegistros(codigo);

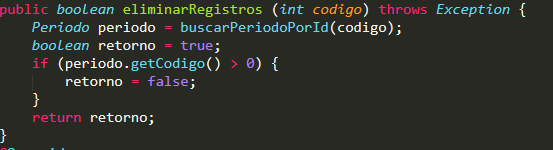
}

**Bean:** En el bean se hace la funcionalidad de los métodos declarados en la fachada.

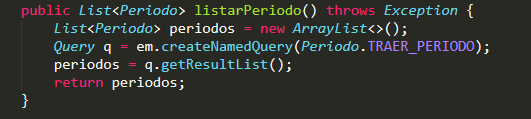
* crear período: método que permite crear un periodo este método recibe un parámetro de tipo periodo y no retorna nada.



* eliminar registros: este método permite eliminar los registros que tenga la tabla por el código revise un parámetro entero utiliza el método buscar por id y si encuentra ese codigo lo convierte en falso o inactivo



* listar periodo: este método lista todo lo que tenga la tabla hay un arraylist de tipo periodos hace la consulta de traer periodo y retornar el arraylist de tipo periodo



* buscar periodo por id: este método busca el periodo por id recibe un parámetro de tipo entero que corresponde al código que desea buscar, retorna el objeto de tipo periodo

