Trabajo Práctico N° 5

En la quinta entrega del TP continuaremos modificando nuestro sistema de gestión de usuarios aplicando los conocimientos adquiridos sobre patrones de diseño de manera que podamos facilitar el mantenimiento del mismo, tanto en la persistencia de datos como en el control de la información en sesión. Para ello partiremos de un sistema base el cual deberemos ir completando ya sea creando clases o editando los scripts ya desarrollados.

Objetivo: Comprender y profundizar el funcionamiento de la programación Web. También se pretende repasar y profundizar la programación de scripts PHP para la implementación de patrones de diseño.

Introducción: En nuestra actualidad vemos que el desarrollo de Internet ha sido inminente y con ello las aplicaciones Web, por lo tanto, se hace indispensable el uso de un lenguaje que permita desarrollar aplicaciones Web como PHP, entre otros. Teniendo en cuenta la situación anterior es que veremos la como desarrollar aplicaciones Web que se ejecutarán del lado del servidor utilizando uno de los mejores lenguajes del ambiente del Software Libre.

Puntos a realizar en la entrega:

1. Crear la clase que implemente el patrón **Registry**.
2. Crear las clases que implementen el patrón **Singleton**.
3. Crear las clases que implementen el patrón **ActiveRecord**.
4. Crear la clase que implemente el patrón **Factory**.
5. Editar los scripts **Paso1**, **Paso2**, **Paso3** y **Finalizar** para que utilicen los patrones de diseño.
6. Editar los scripts que se encuentran dentro de la carpeta **admin** para que utilicen los patrones de diseño creados.

Desarrollo

1. Crear la clase que implemente el patrón Registry.

Cree una clase con el nombre **Registry** dentro de la carpeta **/tp5/includes/php/Registry**, esta clase deberá implementar el patrón Registry y deberá hacer uso de espacio de nombres el cual será **Sgu\Registry**.

2. Crear las clases que implementen el patrón Singleton.

Cree dos clases dentro de la carpeta **/tp5/includes/php/Singleton**, estas clases deberán implementar el patrón Singleton y deberá hacer uso de espacio de nombres el cual será **Sgu\Singleton**.

La primera clase tendrá como nombre **Sesion**, y además de los métodos y atributos que requiere el patrón, deberá tener los métodos **getRegistry** y **destruir**. A continuación se detalla el funcionamiento de los métodos requeridos:

- **constructor**: deberá iniciar o restaurar la sesión de navegación y verificar si en sesión se encuentra establecida una clave con el nombre “registry”, de no encontrase o si su valor es nulo, deberá crearla con una instancia de un objeto del tipo **Registry**.
- **getRegistry**: deberá devolver el contenido de la clave en sesicón “registry”.
- **destruir**: deberá eliminar la información de la sesión.

La segunda clase deberá tener como nombre **BaseDeDatos**, y además de los métodos y atributos que requiere el patrón, deberá tener un atributo con el nombre “conexion” y un método llamado “getConexion” el cual deberá devolver la instancia de la conexión a la base de datos. También en el método constructor se deberá crear la conexión al servidor de base de datos y seleccionar la base de datos **sgu** (adjunta en el proyecto dentro de la carpeta includes), y resguardar el enlace de conexión dentro del atributo “conexion”.

3. Crear las clases que implementen el patrón ActiveRecord.

Se deberán crear dentro de la carpeta **/tp5/includes/php/ActiveRecord**, estas clases deberán implementar el patrón ActiveRecord y deberá hacer uso de espacio de nombres el cual será **Sgu\Activerecord**. A continuación se describen las clases necesarias.

ActiveRecord

Esta deberá ser una clase abstracta la cual servirá de base para el resto de las clases que implementen este patrón, deberá contener un atributo protegido con el nombre “conexion” y los metodos que se detallan a continuación:

- **constructor**: deberá establecer al atributo “conexion” el enlace de conexión a la base de datos el cual será obtenido mediante la clase BaseDeDatos.
- **Insert, update, delete, fetch**: estos métodos serán métodos abstractos los cuales requiere el patrón y forzará a las clases hijas a que lo desarrollen.

TipoDocumento

Deberá extender de la clase ActiveRecord y por lo tanto deberá desarrollar los métodos que requiere el patrón, además deberá contener los atributos necesarios para representar a la tabla tipodocumento de la base de dato y sus métodos getter y setter.

TipoUsuario

Deberá extender de la clase ActiveRecord y por lo tanto deberá desarrollar los métodos que requiere el patrón, además deberá contener los atributos necesarios para representar a la tabla tipousuario de la base de dato y sus métodos getter y setter.

Persona

Deberá extender de la clase ActiveRecord y por lo tanto deberá desarrollar los métodos que requiere el patrón, además deberá contener los atributos necesarios para representar a la tabla persona de la base de dato y sus métodos getter y setter. Además deberá contener un método llamado **buscarPorNumeroDocumento** el cual deberá buscar en la base de datos el registro correspondiente al número de documento solicitado.

Usuario

Deberá extender de la clase ActiveRecord y por lo tanto deberá desarrollar los métodos que requiere el patrón, además deberá contener los atributos necesarios para representar a la tabla usuario de la base de dato y sus métodos getter y setter. Además deberá contener el método llamado **buscarPorUsuarioContrasenia** el cual deberá recibir como parámetro un nombre de usuario y una contraseña y buscar en la base de datos el registro correspondiente a las credenciales solicitadas; también deberá contener un método llamado **buscarPorIdPersona** el cual buscar en la base de datos el registro correspondiente al id de la persona solicitada.

4. Crear la clase que implemente el patrón Factory.

Cree una clase abstracta con el nombre **ActiveRecordFactory** dentro de la carpeta **/tp5/includes/php/ActiveRecord**, esta clase deberá implementar el patrón Factory y deberá hacer uso de espacio de nombres el cual será **Sgu\ActiveRecord**. Deberá contener un método para cada una de las clases que implementan el patrón ActiveRecord el cual devolverá una instancia de dicho objeto.

5. Editar los scripts Paso1, Paso2, Paso3 y Finalizar para que utilicen los patrones de diseño.

Edite los scripts Paso1, Paso2, Paso3 y Finalizar, de modo que vayan utilizando las clases creadas en los puntos anteriores y de esta manera poder completar el registro de alta de nuevos usuarios dentro de la aplicación.

6. Editar los scripts que se encuentran dentro de la carpeta admin para que utilicen los patrones de diseño creados.

Dentro de la carpeta admin, edite todos los scripts para utilicen las clases con los patrones de diseño creados en los puntos anteriores y de este modo lograr su objetivo que es el de listar, editar y eliminar usuarios dentro de la aplicación. También se deberá hacer el control de credenciales mediante los patrones.

Entrega

La quinta entrega deberá contener el proyecto con todos los scripts modificados, y deberá ser cargado a la plataforma como un único archivo comprimido en formato *.rar o *.tar.gz.