Trabajo Práctico N° 5

En la quinta entrega del trabajo práctico 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 **ValueObject**.
4. Crear las clases que implementen el patrón **ActiveRecord**.
5. Crear la clase que implemente el patrón **Factory**.
6. Editar los scripts **Paso1**, **Paso2**, **Paso3** y **Finalizar** para que utilicen los patrones de diseño.
7. Editar los scripts que se encuentran dentro de la carpeta **administrador** 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**.

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**.

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** (que se encuentra en el proyecto adjunto, dentro de la carpeta **/tp5/includes/db**), y resguardar el enlace de conexión dentro del atributo **"conexion"**.

3. Crear las clases que implementen el patrón ValueObject.

Cree las clases **TipoDocumentoVO, TipoUsuarioVO, PersonaVO** y **UsuarioVO** las cuales según el patrón deberán contener los atributos públicos o sus respectivos **getter** y **setter**. Estas clases se utilizarán para representar la estructura de las tablas en la base de datos, por lo tanto sus atributos tendrán el mismo nombre que las tablas que representan. Dichas clases deberán ser creadas dentro de la carpeta **/tp5/includes/php/ValueObject**.

4. Crear las clases que implementen el patrón ActiveRecord.

Se deberán crear dentro de la carpeta **/tp5/includes/php/ActiveRecord**, y a diferencia de lo visto en la teoría, en vez de tener un atributo por cada columna de la tabla que representa sólo tendrá un atributo el cual será un objeto que implemente el patrón **ValueObject** junto con sus métodos **getter** y **setter**. A continuación se describen las clases necesarias.

TipoDocumento

Representa a la tabla **tipodocumento** y estará compuesta por la siguiente estructura:

- Atributo privado que contiene la instancia del objeto **TipoDocumentoVO**.
- Métodos **getter** y **setter** para el atributo.
- Métodos **fetch, insert, update** y **delete**. Cada uno de estos métodos hará uso de la conexión a la base de datos mediante la clase **BaseDeDatos** que implementa el patrón **Singleton**.
- Un método llamado **fetchAll** el cual deberá devolver un **array** de objetos del tipo **TipoDocumentoVO** con todos los registros de dicha tabla.

TipoUsuario

Representa a la tabla **tipousuario** y estará compuesta por la siguiente estructura:

- Atributo privado que contiene la instancia del objeto **TipoUsuarioVO**.
- Métodos **getter** y **setter** para el atributo.
- Métodos **fetch, insert, update** y **delete**. Cada uno de estos métodos hará uso de la conexión a la base de datos mediante la clase **BaseDeDatos** que implementa el patrón **Singleton**.
- Un método llamado **fetchAll** el cual deberá devolver un **array** de objetos del tipo **TipoUsuarioVO** con todos los registros de dicha tabla.

Persona

Representa a la tabla **persona** y estará compuesta por la siguiente estructura:

- Atributo privado que contiene la instancia del objeto **PersonaVO**.
- Métodos **getter** y **setter** para el atributo.
- Métodos **fetch, insert, update** y **delete**. Cada uno de estos métodos hará uso de la conexión a la base de datos mediante la clase **BaseDeDatos** que implementa el patrón **Singleton**.

Usuario

Representa a la tabla **usuario** y estará compuesta por la siguiente estructura:

- Atributo privado que contiene la instancia del objeto **UsuarioVO**.
- Métodos **getter** y **setter** para el atributo.
- Métodos **fetch, insert, update** y **delete**. Cada uno de estos métodos hará uso de la conexión a la base de datos mediante la clase **BaseDeDatos** que implementa el patrón **Singleton**.

Nota: *Además de los métodos mencionados se podrán agregar los que usted crea necesario, como ser métodos de validación para las formas de contacto y contraseña definidos en entregas anteriores.*

5. Crear la clase que implemente el patrón Factory.

Cree una clase abstracta con el nombre **ActiveRecordFactory** dentro de la carpeta **/tp5/includes/php/Factory**, esta clase deberá implementar el patrón Factory y deberá tener un método para cada una de las clases que implementan el patrón **ActiveRecord** el cual devolverá una instancia de dicho objeto.

6. 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.

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

Dentro de la carpeta administrador, 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.

Entrega

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

Para exportar la base de datos puede hacer uso de la herramienta phpMyAdmin la cual generará un archivo con extensión .sql el cual contiene la estructura de la base de datos junto con la información contenida.