UF1467 – Aplicaciones microinformaticas e internet para consulta y generacion de documentacion

ENUNCIADO

*Documentar usando la sintaxis de javadoc el proyecto suministrado para la resolución del ejercicio. Para ello se crearan los comentarios para los siguientes elementos en todos los archivos *.java del proyecto:*

- *El autor de cada archivo*
- *La versión de cada documento*
- *Parámetros y retornos de cada método*
- *Descripción de cada clase*

Debido a la extesión del proyecto suministrado, se podrá un ejemplo de cada una de las 5 capas de que se compone el proyecto: acceso a datos, controladores, entidades, listener y lógica de negocio.

ACCESO A DATOS

Dao.java

```java
package com.ipartek.formacion.ejemplofinal.accesodatos;

import java.util.Set;

/**
 * Interfaz donde se declaran los métodos con los datos
 * @author Arturo Montañez
 * @version 1.0
 * @param <T>
 */

public interface Dao<T> {
    default Set<T> obtenerTodos(){
        throw new AccesoDatosException("NO IMPLEMENTADO");
    }
    default T obtenerPorId(long id){
        throw new AccesoDatosException("NO IMPLEMENTADO");
    }
    default T insertar(T objeto){
        throw new AccesoDatosException("NO IMPLEMENTADO");
    }
    default T modificar (T objeto){
        throw new AccesoDatosException("NO IMPLEMENTADO");
    }
    default void borrar(Long id){
        throw new AccesoDatosException("NO IMPLEMENTADO");
    }


}
```

ProductoDaoMySql.java

```java
package com.ipartek.formacion.ejemplofinal.accesodatos;

import java.sql.Connection;

/**
 * Implementa los métodos del Dao para producto
 *
 * @author Arturo Montañez
 * @version 1.0
 */

class ProductoDaoMySql implements Dao<Producto> {
    private static final String SQL_SELECT = "SELECT p.id AS id, p.nombre AS nombre, p.des
            + "FROM productos p\r\n" + "JOIN departamentos d ON p.departamentos_id = d.id"
    private static final String SQL_SELECT_ID = SQL_SELECT + " WHERE p.id = ?";

    @Override
    /**
     * Método donde se obtienen todos los prodcutos
     *
     * @return Set<Producto>
     */
    public Set<Producto> obtenerTodos() {
        try (Connection con = Config.dataSource.getConnection();
                Statement st = con.createStatement();
                ResultSet rs = st.executeQuery(SQL_SELECT)) {
            Set<Producto> productos = new HashSet<>();

            Producto producto;

            while (rs.next()) {
                producto = mapearResultSetProducto(rs);

                productos.add(producto);
            }

            return productos;
        } catch (Exception e) {
            throw new AccesoDatosException("Error al obtener todos los productos", e);
        }
    }
```

```java
    @Override
    /**
     * Método donde se obtiene un producto por Id
     *
     * @param id
     * @return Producto
     */
    public Producto obtenerPorId(long id) {
        try (Connection con = Config.dataSource.getConnection();
                PreparedStatement pst = con.prepareStatement(SQL_SELECT_ID);
                ) {

            pst.setLong(1, id);

            ResultSet rs = pst.executeQuery();

            Producto producto = null;

            if (rs.next()) {
                producto = mapearResultSetProducto(rs);
            }

            return producto;
        } catch (Exception e) {
            throw new AccesoDatosException("Error al obtener el producto id " + id, e);
        }
    }

    private Producto mapearResultSetProducto(ResultSet rs) throws SQLException {
        Producto producto;
        Departamento departamento;
        departamento = new Departamento(rs.getLong("d_id"), rs.getString("d_nombre"),
                rs.getString("d_descripcion"), null);
        producto = new Producto(rs.getLong("id"), rs.getString("nombre"), rs.getString("descripcion"), rs.getString("url_imagen"),
                rs.getBigDecimal("precio"), rs.getInt("descuento"), rs.getString("unidad_medida"),
                rs.getBigDecimal("precio_unidad_medida"), rs.getInt("cantidad"), departamento,
                rs.getBoolean("activo"), null);
        return producto;
    }

}
```

CONTROLADORES

IndexServlet.java

```java
package com.ipartek.formacion.ejemplofinal.controladores;

import java.io.IOException;

/**
 * Controlador que muestra la página inicial de la aplicación.
 * En la vista se muestran los productos disponibles para la compra
 *
 * @author Arturo Montañez
 * @version 1.0
 */
@WebServlet("/IndexServlet")
public class IndexServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setAttribute("productos", Config.carritoNegocio.listadoProductos());
        request.getRequestDispatcher(Config.PATH_VISTAS + "index.jsp").forward(request,  response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

}
```

ENTIDADES

Factura.java

```java
package com.ipartek.formacion.ejemplofinal.entidades;

    /**
     * Método que halla el precio total de todos los productos adquiridos
     * @return precio total de la factura sin IVA
     */
    public BigDecimal getTotal() {
        BigDecimal total = BigDecimal.ZERO;

        for(DetalleFactura detalle: detallesFactura) {
            total = total.add(detalle.getTotal());
        }

        return total;
    }

    /**
     * Método que halla el valor del IVA del precio total
     * @return valor del IVA
     */
    public BigDecimal getIva() {
        return getTotal().multiply(IVA);
    }

    /**
     * Método que suma el valor del IVA al precio total
     * @return precio total con IVA incluido
     */
    public BigDecimal getTotalConIva() {
        return getTotal().add(getIva());
    }
}
```

LISTENER

SesionListener.java

```java
package com.ipartek.formacion.ejemplofinal.listener;

import javax.servlet.annotation.WebListener;


/**
 * Clase listener que usamos para asignar un carrito a un usuario al iniciar la sesión
 *
 * @author Arturo Montañez
 * @version 1.0
 */

@WebListener
public class SesionListener implements HttpSessionListener {

    /**
     * Método que al iniciar una sesión, asigna un carrito al usuario.
     *
     * @param se
     */

    @Override
    public void sessionCreated(HttpSessionEvent se)  {
        Carrito carrito = new Carrito();
        se.getSession().setAttribute("carrito", carrito);

    }

    /**
     * Método que cierra la sesión
     * @param se
     */

    @Override
    public void sessionDestroyed(HttpSessionEvent se)  {
        //No necesario
    }

}
```

LÓGICA DE NEGOCIO

UsuarioNegocio.java

```java
package com.ipartek.formacion.ejemplofinal.logicanegocio;

import com.ipartek.formacion.ejemplofinal.entidades.Usuario;

/**
 * Interfaz donde se declaran métodos de UsuarioNegocio
 *
 * @author Arturo Montañez
 * @version 1.0
 */
public interface UsuarioNegocio {

    boolean validarUsuario(Usuario usuario);
}
```

UsuarioNegocioImpl.java

```java
package com.ipartek.formacion.ejemplofinal.logicanegocio;

import com.ipartek.formacion.ejemplofinal.accesodatos.DaoFabrica;

/**
 * Clase donde se implementan los métodos declarados en UsuarioNegocio
 *
 * @author Arturo Montañez
 * @version 1.0
 */
public class UsuarioNegocioImpl implements UsuarioNegocio{

    private DaoUsuario dao = DaoFabrica.getDaoUsuario();

    /**
     * Método que usamos para validar al usuario, Devuelve true o false.
     *
     * @param usuario
     * @return boolean
     */
    @Override
    public boolean validarUsuario(Usuario usuario) {
        Usuario usuarioBdd = dao.obtenerPorEmail(usuario.getEmail());

        if(usuarioBdd != null && usuario.getPassword().equals(usuarioBdd.getPassword())) {
            usuario.setId(usuarioBdd.getId());
            usuario.setCliente(usuarioBdd.getCliente());

            return true;
        } else {
            return false;
        }
    }

}
```