CONECTORES

```
public class Conectores {

    private Connection conexion;

    public int ejercicio3(String bd,Double valMenor,Double valMayor){
        abrirConexion(bd, "localhost", "root", "");
        int nOcurrencias=0;
        try{
            Statement stmt = conexion.createStatement();
            String query = "SELECT nombre,sigloAcConstruccion,valoracion FROM maravillas WHERE
valoracion>"+valMenor+" AND valoracion<"+valMayor;
            ResultSet filas = stmt.executeQuery(query);
            while(filas.next()){
                System.out.println("Nombre: "+filas.getString("nombre")+", Siglo Ac Construcción:
"+filas.getInt("sigloAcConstruccion")+", Valoración: "+filas.getLong("valoracion"));
                nOcurrencias++;                                    getDouble
            }
            return nOcurrencias;
        }catch(SQLException e){
            System.out.println("Codigo Error SQL: "+e.getErrorCode());
            System.out.println("Estado SQL: "+e.getSQLState());
            System.out.println("Tipo de excepción: "+e.getNextException());
            return-1;
        }finally{
            cerrarConexion();      No cierras el Statement
        }

    }

    public int ejercicio4(String bd,int codmaravilla,String nombre,Double valoracion){
        abrirConexion(bd, "localhost", "root", "");
        int nOcurrencias=0;
        try{
            Statement stmt = conexion.createStatement();          El update esta mal
            String query = "UPDATE table maravillas INSERT INTO (nombre,valoracion)
VALUES("+nombre+","+valoracion+") WHERE codmaravilla="+codmaravilla;
            stmt.executeUpdate(query);
            No capturas el nº de filas devueltas -> podrían ser mayor que 1
            return 1;
        }catch(SQLException e){
            if(this.conexion==null){
                return 0;  Esto seria un valor distinto de <0
            }else{
                System.out.println("Codigo Error SQL: "+e.getErrorCode());
                System.out.println("Estado SQL: "+e.getSQLState());
                System.out.println("Tipo de excepción: "+e.getNextException());
                return -1;
            }

        }finally{
            cerrarConexion(); No cierras el Statement
        }

    }

    public void ejercicio5(String bd,String cadenaContenida,Double valoracion){
        abrirConexion(bd, "localhost", "root", "");
        try{
            String query = "SELECT nombre,nombreCiv,valoracion FROM maravillas WHERE
valoracion>? AND nombre LIKE ?";
                                  Así preparas la sentencia cada vez que la ejecutas
            PreparedStatement stmt = conexion.prepareStatement(query);
            stmt.setDouble(1, valoracion);   Pierdes las ventajas de la PreparedStatement
```

```
            stmt.setString(2, cadenaContenida);
            ResultSet filas = stmt.executeQuery();

            while(filas.next()){
                System.out.println("Nombre: "+filas.getString("nombre")+", Siglo Ac Construcción:
"+filas.getInt(3)+", Valoración: "+filas.getDouble("valoracion"));
            }
        }catch(SQLException e){
            System.out.println("Codigo Error SQL: "+e.getErrorCode());
            System.out.println("Estado SQL: "+e.getSQLState());
            System.out.println("Tipo de excepción: "+e.getNextException());
        }finally{
            cerrarConexion();
        }

    }

    public void ejercicio6(String bd,String nombreTabla){
        abrirConexion(bd, "localhost", "root", "");
        try{
            DatabaseMetaData dbmt = conexion.getMetaData();

            ResultSet table = dbmt.getColumns(bd, null, nombreTabla, null);
            while (table.next()) {
                System.out.println("Nombre de la columna: "+table.getString("COLUMN_NAME"));
                System.out.println("Tipo de dato: "+table.getString("TYPE_NAME"));
                System.out.println("Permite nulos: "+table.getString("NULLABLE"));
                System.out.println("Autoincrementado: "+table.getString("IS_AUTOINCREMENT"));
            }
        }catch(SQLException e){
            System.out.println("Codigo Error SQL: "+e.getErrorCode());
            System.out.println("Estado SQL: "+e.getSQLState());
            System.out.println("Tipo de excepción: "+e.getNextException());
        }finally{
            cerrarConexion();
        }
    }

    public static void main(String[] args) {
        Conectores conectores = new Conectores();
        //EJERCICIO 3 System.out.println("Número ocurrencias:
"+conectores.ejercicio3("admarzo23", 0, 10));
        //EJERCICIO 4 System.out.println("Resultado: "+conectores.ejercicio4("admarzo23",5,
"Athenas",9.2));
        //EJERCICIO 5 conectores.ejercicio5("admarzo23", "%stat%", 1.0);
        //EJERCICIO 6 conectores.ejercicio6("admarzo23", "maravillas");
    }
}
```

Como lo tienes tendriáis que cerras el PreparedStatement

SERVICIOS WEB

```
@Path("/api")
public class Examen{
    public static ArrayList<Maravilla> maravillas = new ArrayList<Maravilla>();
    private ConectoresRest cntRest = new ConectoresRest();

    /**
     * http://localhost:8080/examenrest/rest/api
     * Produce XML y JSON
     * @return maravilla en su forma JSON o XML
     */
    @GET
    @Produces({MediaType.APPLICATION_JSON,MediaType.APPLICATION_XML})
    public Response ejercicio7() {
```

Faltan los comentarios: método http y parámetros cabecera

```
        Maravilla maravillaNueva = new Maravilla(10, "Maccu-Piccu", 30, 12, "Yucatecas", "Perú",
"Barro", 3.2);
        maravillas.add(maravillaNueva);
        return Response.ok(maravillaNueva).build();
    }

    /**
     * http://localhost:8080/examenrest/rest/api/{id}
     * Produce XML y JSON
     * Parámetros de cabecera
     * @param id Es el pathParam
     * @return Número de filas borradas en forma de json o xml
     */
                                Faltan los comentarios: método http y parámetros cabecera
    @Path("/{id}")
    @DELETE
    @Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
    public Response ejercicio8(@PathParam("id") int id) {
        int borradas=0;
        for (int i = 0; i < maravillas.size(); i++) {
            if(maravillas.get(i).getCodmaravilla()==id) {
                maravillas.remove(i);
                i--;
                borradas++;
            }
        }

        return Response.ok(String.format("%d",borradas)).build();
    }

    /**
     * http://localhost:8080/examenrest/rest/api/tabla
     * Produce array de todas las maravillas a JSON
     * No recibe valores
     */

    @Path("/tabla")
    @GET                        Faltan los comentarios: método http y parámetros cabecera
    @Produces({MediaType.APPLICATION_JSON})
    public Response ejercicio9() {


        ArrayList<Maravilla> maravillasBase = new ArrayList<Maravilla>();
        cntRest.abrirConexion("admarzo23", "localhost", "root", "");
        try {
            String query = "SELECT * from maravillas";
            Statement stmt = cntRest.conexion.createStatement();
            ResultSet rslt = stmt.executeQuery(query);
            while(rslt.next()) {
                maravillasBase.add(new
Maravilla(rslt.getInt("codmaravilla"),rslt.getString("nombre"),rslt.getInt("sigloAcConstruccion"),rslt.get
Int("codCivilizacion"),rslt.getString("nombreCiv"),rslt.getString("zonaOrigen"),rslt.getString("material")
,rslt.getDouble("valoracion")));
            }
        }catch(SQLException e) {        Falta el mensaje explicativo en el error
            return Response.serverError().build();
        }finally {
            cntRest.cerrarConexion();           No cierras el statement
        }
        return Response.ok(maravillasBase).build();

    }
```

```
    /**
     *  http://localhost:8080/examenrest/rest/api/consulta?nombre=valor1&valoracion=valor2
     *  Parámetros de cabecera
     * @param nombre Cadena que debe contener el nombre
     * @param valoracion Valoracion a la que tiene que ser superior
     * @return Array en json con maravillas que cumplen requisito
     */
    @Path("/consulta")
    @GET
    @Produces({MediaType.APPLICATION_JSON})
    @HeaderParam("Cant")
    public Response ejercicio10(@DefaultValue("sin valor") @QueryParam("nombre") String
nombre,@DefaultValue("0") @QueryParam("valoracion") String valoracion) {
        ConectoresRest cntRest = new ConectoresRest();

        ArrayList<Maravilla> maravillasBase = new ArrayList<Maravilla>();
        cntRest.abrirConexion("admarzo23", "localhost", "root", "");
        try {
            String query = "SELECT * from maravillas WHERE nombre LIKE \"%"+nombre+"%\" AND
valoracion>"+valoracion;

            Statement stmt = cntRest.conexion.createStatement();
            ResultSet rslt = stmt.executeQuery(query);
            while(rslt.next()) {
                maravillasBase.add(new
Maravilla(rslt.getInt("codmaravilla"),rslt.getString("nombre"),rslt.getInt("sigloAcConstruccion"),rslt.get
Int("codCivilizacion"),rslt.getString("nombreCiv"),rslt.getString("zonaOrigen"),rslt.getString("material")
,rslt.getDouble("valoracion")));
            }
        }catch(SQLException e) {
            return Response.serverError().build();
        }

        return Response.ok(maravillasBase).build();

    }


}


@XmlRootElement
public class Maravilla{

    private int codmaravilla;
    private String nombre;
    private int sigloAcConstruccion;
    private int codCivilizacion;
    private String nombreCiv;
    private String zonaOrigen;
    private String material;
    private Double valoracion;

    public Maravilla() {

    }

    public Maravilla(int codmaravilla, String nombre, int sigloAcConstruccion, int codCivilizacion,
String nombreCiv,
            String zonaOrigen, String material, Double valoracion) {
        this.codmaravilla = codmaravilla;
        this.nombre = nombre;
```

Faltan los comentarios: método http y parámetros cabecera

Esto hace fallar el metodo

Falta el mensaje explicativo en el error

No cierras el statement

No cierras la base de datos

```
        this.sigloAcConstruccion = sigloAcConstruccion;
        this.codCivilizacion = codCivilizacion;
        this.nombreCiv = nombreCiv;
        this.zonaOrigen = zonaOrigen;
        this.material = material;
        this.valoracion = valoracion;
    }
    public int getCodmaravilla() {
        return codmaravilla;
    }
    public void setCodmaravilla(int codmaravilla) {
        this.codmaravilla = codmaravilla;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getSigloAcConstruccion() {
        return sigloAcConstruccion;
    }
    public void setSigloAcConstruccion(int sigloAcConstruccion) {
        this.sigloAcConstruccion = sigloAcConstruccion;
    }
    public int getCodCivilizacion() {
        return codCivilizacion;
    }
    public void setCodCivilizacion(int codCivilizacion) {
        this.codCivilizacion = codCivilizacion;
    }
    public String getNombreCiv() {
        return nombreCiv;
    }
    public void setNombreCiv(String nombreCiv) {
        this.nombreCiv = nombreCiv;
    }
    public String getZonaOrigen() {
        return zonaOrigen;
    }
    public void setZonaOrigen(String zonaOrigen) {
        this.zonaOrigen = zonaOrigen;
    }
    public String getMaterial() {
        return material;
    }
    public void setMaterial(String material) {
        this.material = material;
    }
    public Double getValoracion() {
        return valoracion;
    }
    public void setValoracion(Double valoracion) {
        this.valoracion = valoracion;
    }



}
```