

# PROYECTO CONCESIONARIO



RUBEN OSMA MADRIGAL

# INDICE

1. Descripción.
2. Fuente
3. Estructura de nodos
4. Estructura de datos
5. Archivo del proyecto
6. JDBC
7. Proyecto
8. Conclusiones

# DESCRIPCION

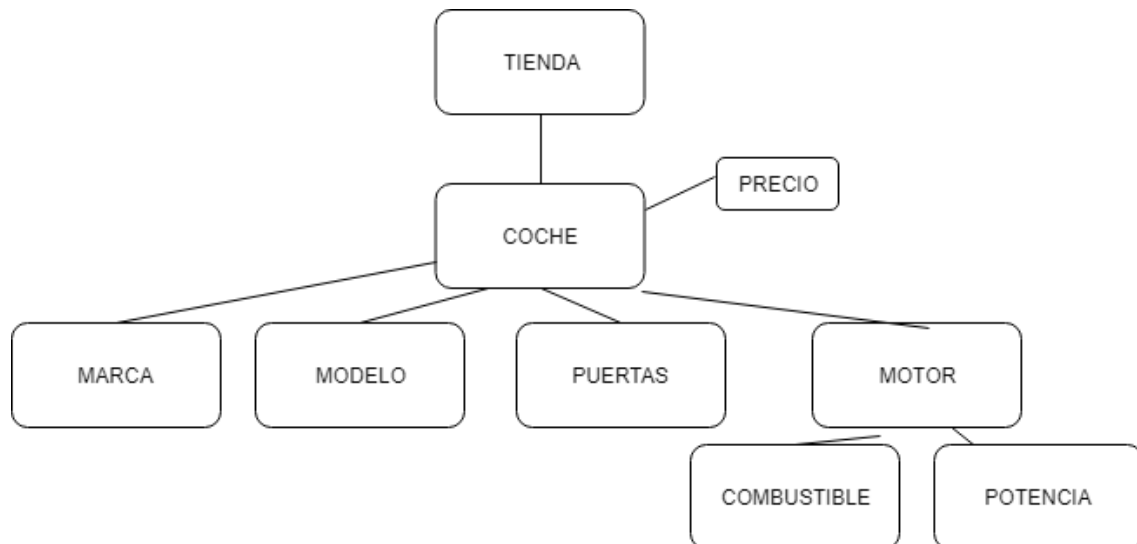
Se desea realizar una aplicación para le gestión de la información de un concesionario.

Proyecto actualizado el 25 de Octubre de 2018.

## FUENTE

[Enlace- GitHub](#)

## ESTRUCTURA DE NODOS



# ESTRUCTURA DE DATOS

Clase Tienda:

```
public class Tienda extends ArrayList<Coche> {  
  
    public ArrayList<Coche> getTienda() { return this; }  
  
    @Override  
    public String toString() {  
        String texto="";  
        for (int i = 0; i < getTienda().size(); i++) {  
            texto+=getTienda().get(i).toString()+"\n";  
        }  
        return texto;  
    }  
}
```

Clase Coche:

```
public class Coche {  
    private int precio;  
    private String marca;  
    private String modelo;  
    private String puertas;  
    private ArrayList<Motor> listaMotores = new ArrayList<>();  
}
```

Clase Motor:

```
public class Motor {  
    private String combustible;  
    private String potencia;  
}
```

# ARCHIVO DEL PROYECTO

## Clase ControlDOM:

```
public class ControlDOM {

    //De XML a DOM
    public Document deXMLaDOM() throws ParserConfigurationException {
        Document doc = null;
        doc = (Document) DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
        return doc;
    }

    public Document deXMLaDOM(File fXmlFile) throws ParserConfigurationException, IOException, SAXException {
        Document doc = null;
        doc = (Document) DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(fXmlFile);
        doc.getDocumentElement().normalize();
        return doc;
    }

    //De DOM a XML
    public static void deDOMaXML(Document doc, File file) throws TransformerConfigurationException, TransformerException {
        Transformer trans = TransformerFactory.newInstance().newTransformer();
        trans.setOutputProperty(OutputKeys.INDENT, value: "yes"); //indentar XML

        StreamResult result = new StreamResult(file);
        DOMSource source = new DOMSource(doc);
        trans.transform(source, result);
    }

    //Obtener ATRIBUTO
    public static String getAtributoEtiqueta(Element elemento, String ETIQUETA) {
        return elemento.getAttribute(ETIQUETA);
    }

    //Obtener ETIQUETAS
    public static Element getElementEtiqueta(String ETIQUETA, Element elemento) {
        return (Element) elemento.getElementsByTagName(ETIQUETA).item(0);
    }

    //Obtener VALOR de las etiquetas
    public static String getValorEtiqueta(String ETIQUETA, Element elemento) {
        Node nValue = elemento.getElementsByTagName(ETIQUETA).item(0);
        return nValue.getChildNodes().item(0).getNodeValue();
    }
}
```

## Clase ControlTienda:

```
public class ControlTienda extends ControlDOM {

    Tienda tienda = null;

    public ControlTienda() { this.tienda = new Tienda(); }
    public Tienda getTienda() { return tienda; }

    public void setTienda(Tienda tienda) { this.tienda = tienda; }

    public Document Recuperar(File file) throws IOException, SAXException, ParserConfigurationException {
        Document doc = null;
        doc = deXMLaDOM(file);
        return doc;
    }

    public Tienda Leer(Document doc) {
        Element elemTienda = doc.getDocumentElement();
        NodeList listaCoches = elemTienda.getChildNodes();

        for (int i = 0; i < listaCoches.getLength(); i++) {
            if (listaCoches.item(i).getNodeType() == Node.ELEMENT_NODE) {
                tienda.add(ControlCoche.LeerCoche((Element) listaCoches.item(i)));
            }
        }
        return tienda;
    }

    public void Escribir(Document doc) {
        Element elemTienda = doc.createElement( tagName: "tienda");

        for (Coche coche : tienda) {
            ControlCoche.EscribirCoche(doc, elemTienda, coche);
        }

        doc.appendChild(elemTienda);
    }

    public void Guardar(File f, Document doc) throws TransformerException {
        deDOMaXML(doc, f);
    }
}
```



## Clase ControlCoche:

```
public class ControlCoche extends ControlDOM{

    private final static String ET_TIENDA="tienda";
    private final static String ET_COCHE="coche";
    private final static String ET_MARCA="marca";
    private final static String ET_MODELO="modelo";
    private final static String ET_MOTORES="motores";
    private final static String ET_MOTOR="motor";
    private final static String ET_COMBUSTIBLE="combustible";
    private final static String ET_POTENCIA="potencia";
    private final static String ET_PUERTAS="puertas";
    private final static String AT_PRECIO="precio";

    public static Coche LeerCoche(Element elemCoche) {
        Coche coche = new Coche();

        coche.setMarca(getValorEtiqueta(ET_MARCA, elemCoche));
        coche.setModelo(getValorEtiqueta(ET_MODELO, elemCoche));
        coche.setPuertas(getValorEtiqueta(ET_PUERTAS, elemCoche));

        coche.setPrecio(Integer.parseInt(getAtributoEtiqueta(elemCoche, AT_PRECIO)));

        coche.setListaMotores(LeerMotor(getElementEtiqueta(ET_MOTORES, elemCoche)));

        return coche;
    }

    private static ArrayList<Motor> LeerMotor(Element elemMotores) {
        ArrayList<Motor> listaMotor = new ArrayList<>();
        NodeList nListaMotor = elemMotores.getChildNodes();

        for (int i = 0; i < nListaMotor.getLength(); i++) {
            if (nListaMotor.item(i).getNodeType() == Node.ELEMENT_NODE) {
                Motor motor = new Motor();

                motor.setCombustible(getValorEtiqueta(ET_COMBUSTIBLE, (Element)nListaMotor.item(i)));
                motor.setPotencia(getValorEtiqueta(ET_POTENCIA, (Element)nListaMotor.item(i)));

                listaMotor.add(motor);
            }
        }
        return listaMotor;
    }

    public static void EscribirCoche(Document doc, Element elemTienda, Coche coche) {
        Element elemCoche = doc.createElement(ET_COCHE);
        elemCoche.setAttribute(AT_PRECIO, Integer.toString(coche.getPrecio()));

        Element elemMarca = doc.createElement(ET_MARCA);
        elemMarca.setTextContent(coche.getMarca());
        elemCoche.appendChild(elemMarca);

        Element elemModelo = doc.createElement(ET_MODELO);
        elemModelo.setTextContent(coche.getModelo());
        elemCoche.appendChild(elemModelo);

        Element elemPuertas = doc.createElement(ET_PUERTAS);
        elemPuertas.setTextContent(coche.getPuertas());
        elemCoche.appendChild(elemPuertas);

        Element elemMotores = doc.createElement(ET_MOTORES);
        EscribirMotores(coche, elemMotores, doc);
        elemCoche.appendChild(elemMotores);

        elemTienda.appendChild(elemCoche);
    }

    private static void EscribirMotores(Coche coche, Element elemMotores, Document doc) {
        ArrayList<Motor> listaMotores = coche.getListaMotores();

        for (int i = 0; i < listaMotores.size(); i++) {
            Element elemMotor = doc.createElement(ET_MOTOR);

            Element elemCombustible = doc.createElement(ET_COMBUSTIBLE);
            elemCombustible.setTextContent(listaMotores.get(i).getCombustible());
            elemMotor.appendChild(elemCombustible);

            Element elemPotencia = doc.createElement(ET_POTENCIA);
            elemPotencia.setTextContent(listaMotores.get(i).getPotencia());
            elemMotor.appendChild(elemPotencia);

            elemMotores.appendChild(elemMotor);
        }
    }
}
```

# JDBC

## Tabla de Coches

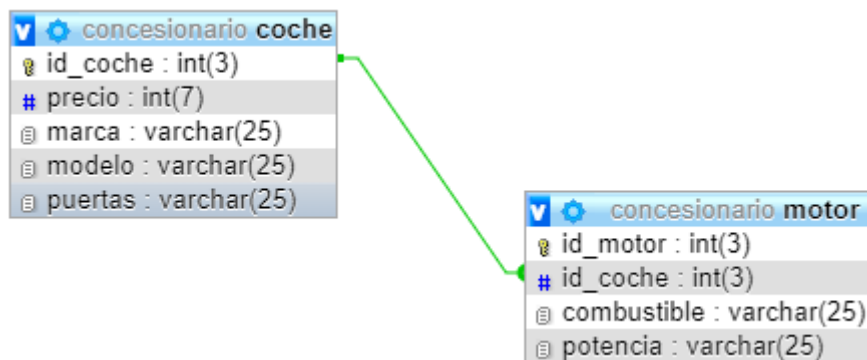
#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	<b>id_coche</b>	int(3)			No	Ninguna		AUTO_INCREMENT	Cambiar  Eliminar  Más
<input type="checkbox"/> 2	<b>precio</b>	int(7)			No	Ninguna			Cambiar  Eliminar  Más
<input type="checkbox"/> 3	<b>marca</b>	varchar(25)	latin1_swedish_ci		No	Ninguna			Cambiar  Eliminar  Más
<input type="checkbox"/> 4	<b>modelo</b>	varchar(25)	latin1_swedish_ci		No	Ninguna			Cambiar  Eliminar  Más
<input type="checkbox"/> 5	<b>puertas</b>	varchar(25)	latin1_swedish_ci		No	Ninguna			Cambiar  Eliminar  Más

## Tabla de Motores

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	<b>id_motor</b>	int(3)			No	Ninguna		AUTO_INCREMENT	Cambiar  Eliminar  Más
<input type="checkbox"/> 2	<b>id_coche</b>	int(3)			No	Ninguna			Cambiar  Eliminar  Más
<input type="checkbox"/> 3	<b>combustible</b>	varchar(25)	latin1_swedish_ci		No	Ninguna			Cambiar  Eliminar  Más
<input type="checkbox"/> 4	<b>potencia</b>	varchar(25)	latin1_swedish_ci		No	Ninguna			Cambiar  Eliminar  Más

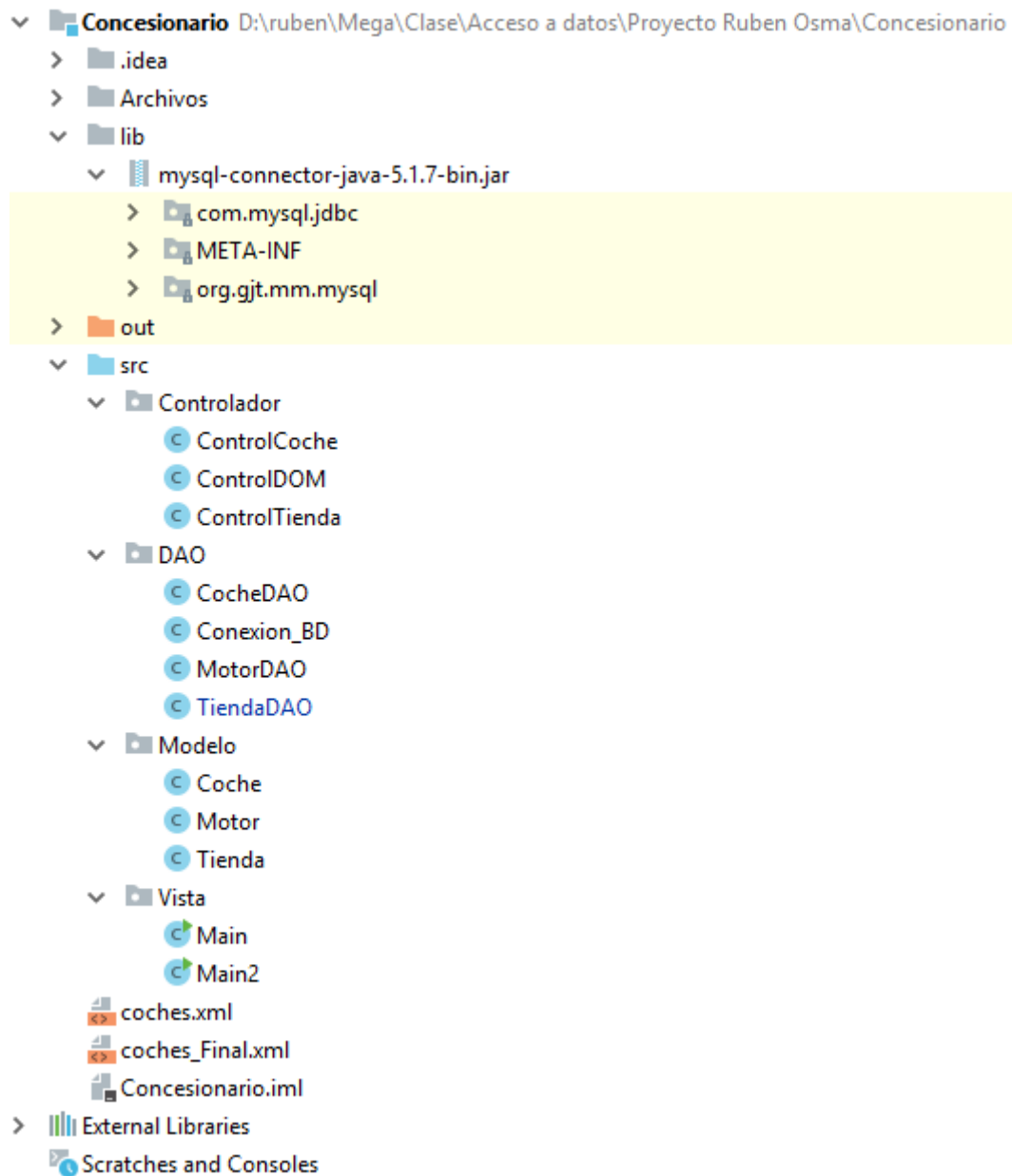
Script SQL de la base de dato en los archivos del proyecto

Diseño de la base de datos



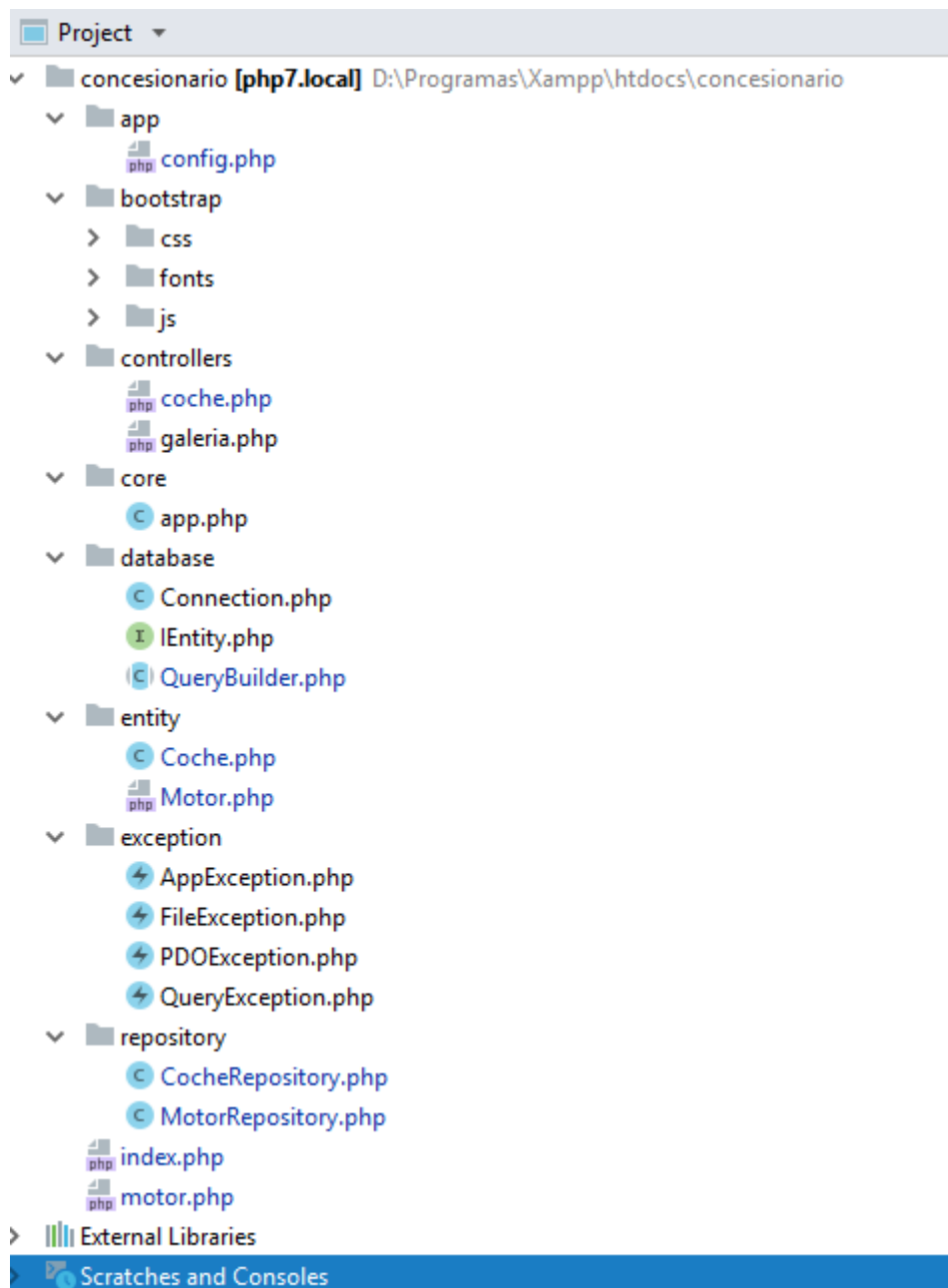
# PROYECTO

## Estructura del proyecto





# PHP



# VISTA PHP

## AÑADIR COCHES

Marca Coche

Modelo Coche

Puertas Coche

Precio Coche

Añadir

## TABLA CONCESIONARIO

#	Marca Coche	Modelo Coche	Puertas Coche	Precio Coche
2	Skoda	Vision RS	cinco	20000
3	Toyota	Corolla	tres	14000
4	Audi	A3	tres	34000
5	Bmw	325d Berlina	cinco	34000
6	Tesla	Model S	tres	80000
7	Toyota	Corolla	tres	14000
8	Audi	A3	tres	34000

# CONCLUSIONES

La parte más interesante del proyecto para mí fue la de acceso a la base de datos, en sí no fue difícil, pero surgieron problemas como el de los id en las clases y al añadirlos a las bases de datos para hacerlos auto numéricos, tanto en java(clases) como en la base de datos.