



ENCICLOPEDIA DE MONUMENTOS

Proyecto Servidores Catedrático: José Rafael Rojano Cáceres



12 DE DICIEMBRE DE 2022 EQUIPO 3

RIVERA VIVEROS CARLA GUADALUPE FRANYUTTI PULIDO ANGEL FABRIZIO BONILLA MARTINEZ
JOSE MANUEL PANES LANDA KEVIN OSVALDO

Contenido

Frond End	2
Bootstrap.....	2
CSS	4
Base de datos DB4free	5
Back End	7
DAO	8
Monumento	8
Conexión.....	9
Pom.xml.....	9

Front End

Bootstrap

Se utilizó la librería Bootstrap para hacer el proyecto, esta librería fue utilizada en la mayor parte del proyecto realizado, ya que este es un código abierto para el diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

Bootstrap, originalmente llamado Blueprint de Twitter, fue desarrollado por Mark Otto y Jacob Thornton de Twitter, como un marco de trabajo (*framework*) para fomentar la consistencia entre las herramientas internas. Antes de Bootstrap, se usaron varias bibliotecas para el desarrollo de interfaces de usuario, lo que generó inconsistencias y una gran carga de trabajo en su mantenimiento.

El primer desarrollo en condiciones reales ocurrió durante la primera «Semana de Hackeo (*Hackweek*) de Twitter». ⁵ Mark Otto mostró a algunos colegas cómo acelerar el desarrollo de sus proyectos con la ayuda de la herramienta de trabajo. Como resultado, decenas de temas se han introducido en el marco de trabajo.

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por Bootstrap a pesar de la falta de soporte de navegadores antiguos. Esto extiende la funcionalidad de la herramienta, pero no es requerida para su uso.

Desde la versión 2.0 también soporta diseños web adaptables o responsivos "Responsive". Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (computadoras, tabletas o teléfonos móviles).

Bootstrap es modular y consiste esencialmente en una serie de hojas de estilo LESS que implementan la variedad de componentes de la herramienta. Una hoja de estilo llamada `bootstrap.less` incluye los componentes de las hojas de estilo. Los desarrolladores pueden adaptar el mismo archivo de Bootstrap, seleccionando los componentes que deseen usar en su proyecto.

Los ajustes son posibles en una medida limitada a través de una hoja de estilo de configuración central. Los cambios más profundos son posibles mediante las declaraciones LESS.

El uso del lenguaje de hojas de estilo LESS permite el uso de variables, funciones y operadores, selectores anidados, así como clases mixin.

Desde la versión 2.0, la configuración de Bootstrap también tiene una opción especial de «Personalizar» en la documentación. Por otra parte, los desarrolladores eligen en un formulario los componentes y ajustes deseados, y de ser necesario, los valores de varias opciones a sus necesidades. El paquete consecuentemente generado ya incluye la hoja de estilo CSS compilada previamente.

Bootstrap viene con una disposición de cuadrilla estándar de 940 píxeles de ancho. Alternativamente, el desarrollador puede usar un diseño de ancho-variable. Para ambos

casos, la herramienta tiene cuatro variaciones para hacer uso de distintas resoluciones y tipos de dispositivos: teléfonos móviles, formato vertical y horizontal, tabletas y computadoras con baja y alta resolución (pantalla ancha). Esto ajusta el ancho de las columnas automáticamente.

Bootstrap proporciona un conjunto de hojas de estilo que proveen definiciones básicas de estilo para todos los elementos de HTML. Esto otorga una uniformidad al navegador y al sistema de anchura, da una apariencia moderna para el formateo de los elementos de texto, tablas y formularios.

Además de los elementos regulares de HTML, Bootstrap contiene otra interfaz de elementos comúnmente usados. Esta incluye botones con características avanzadas (p.ej. grupo de botones o botones con opción de menú desplegable, listas de navegación, etiquetas horizontales y verticales, ruta de navegación, paginación, etc.), etiquetas, capacidades avanzadas de miniaturas tipográficas, formatos para mensajes de alerta y barras de progreso.

Los componentes de JavaScript para Bootstrap están basados en la librería jQuery de JavaScript. Los *plug-ins* se encuentran en la herramienta de *plug-in* de jQuery. Proveen elementos adicionales de interfaz de usuario como diálogos, tooltips y carruseles. También extienden la funcionalidad de algunos elementos de interfaz existentes, incluyendo por ejemplo una función de auto-completar para campos de entrada (input). La versión 2.0 soporta los siguientes *plug-ins* de JavaScript: Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel y Typeahead.

Una implementación de Bootstrap usando el Dojo toolkit también está disponible. Es llamada *Dojo Bootstrap*⁶⁷ y es un puerto de los plug-ins de Twitter Bootstrap. Usa el código Dojo al 100% y tiene soporte para AMD (Asynchronous Module Definition).

```
TF-8">
="X-UA-Compatible" content="IE=edge">
port" content="width=device-width, initial-scale=1.0">
dia de Monumentos</title>
s://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKQhggwzXh7pPCaAqO46MgnOM8
ps://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBIe4zVF0s0G1MSb4hcxpyD9F7jL+jjXkk+Q
sheet" href=".../.../index.css">
album.css" rel="stylesheet">
```

```

  <p class="card-text">La tour Eiffel</p>
  <div class="d-flex justify-content-between align-items-center">
    <div class="btn-group">
      <button type="button" class="btn btn-sm btn-outline-secondary"><a href="La tour Eiffel.html">Ver</a></button>
    </div>
    <small class="text-muted">Símbolo de París y más ampliamente de Francia </small>
  </div>
</div>
</div>
</div>
<div class="col">
  <div class="card shadow-sm">
    
      <p class="card-text">Musée du Louvre</p>
      <div class="d-flex justify-content-between align-items-center">
        <div class="btn-group">
          <button type="button" class="btn btn-sm btn-outline-secondary"><a href="Musée du Louvre.html">Ver</a></button>
        </div>
        <small class="text-muted">El Louvre, antaño palacio de los reyes, ha acogido la historia de Francia durante ocho s
      </div>
    </div>
  </div>
</div>
<div class="col">
  <div class="card shadow-sm">
    
      <p class="card-text">" Musée d'Orsay"</p>
      <div class="d-flex justify-content-between align-items-center">
```

CSS

Este estilo es el encargado para el home y dar el formato para las diversas paginas que se van a utilizar, esta enlazado con la pagina principal para que se aplique a las demás, es una hoja con estilo de cascada, este establece un diseño visual para la pagina web, está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o *layouts*, los colores y las fuentes.

```
.bd-placeholder-img {
  font-size: 1.125rem;
  text-anchor: middle;
  -webkit-user-select: none;
  -moz-user-select: none;
  user-select: none;
}

@media (min-width: 768px) {
  .bd-placeholder-img-lg {
    font-size: 3.5rem;
  }
}

.b-example-divider {
  height: 3rem;
  background-color: rgba(0, 0, 0, .1);
  border: solid rgba(0, 0, 0, .15);
  border-width: 1px 0;
  box-shadow: inset 0 .5em 1.5em rgba(0, 0, 0, .1), inset 0 .125em .5em rgba(0, 0, 0, .15);
}

.b-example-vr {
  flex-shrink: 0;
  width: 1.5rem;
  height: 100vh;
}

.bi {
  vertical-align: -.125em;
  fill: currentColor;
}

.nav-scroller {
  position: relative;
  z-index: 2;
```

```

/* GLOBAL STYLES
-----sad */
/* Padding below the footer and lighter body text */

body {
  /* padding-top: 3rem; */
  padding-bottom: 3rem;
  color: #5a5a5a;
}

/* CUSTOMIZE THE CAROUSEL
----- */

/* Carousel base class */
.carousel {
  margin-bottom: 4rem;
}
/* Since positioning the image, we need to help out the caption */
.carousel-caption {
  bottom: 10rem;
  z-index: 10;
}

/* Declare heights because of positioning of img element */
.carousel-item {
  height: 45rem;
}

/* MARKETING CONTENT
----- */

/* Center align the text within the three columns below the carousel */
.marketing .col-lg-4 {
  margin-bottom: 1.5rem;
  text-align: center;
}

```

Base de datos DB4free

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo,¹² y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una *Community*, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones *Enterprise*, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones *Enterprise* incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial.

The screenshot shows the phpMyAdmin web interface. The top navigation bar includes links for 'Examinar', 'Estructura', 'SQL', 'Buscar', 'Insertar', 'Exportar', 'Importar', 'Operaciones', 'Seguimiento', and 'Disparadores'. The left sidebar shows a database structure with 'monumentos' selected. The main area displays a table with the following data:

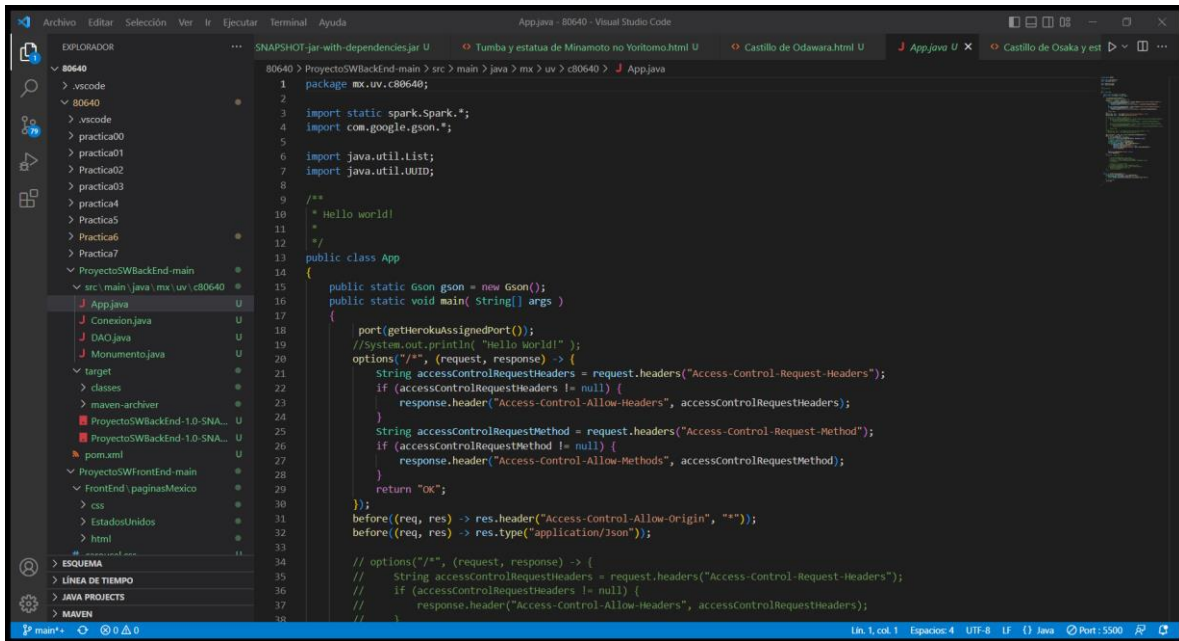
		id	pais	nombreMonumento	añoConstrucción	anoinaguración	altura	largo	materiales
<input type="checkbox"/>				Italia	Consejo Romano	12-	60	60	ladr...
<input type="checkbox"/>				3 Italia	Torre Inclinada de Pisa	1173	1372	60	15 Mermol Blanco
<input type="checkbox"/>				4 Italia	Basilica San Marco	828	1094	98	76 mármoles orientales, esculturas, bronce, dorados,...
<input type="checkbox"/>				5 Italia	La catedral De Monreale	1267	1172	24	102 Barro, gres extrusionado y ladrillo cara vista.
<input type="checkbox"/>				6 Italia	El Ponte Vecchio	1335	1345	30	67 Piedra
<input type="checkbox"/>				7 Italia	El Duomo	1386	1386	108	157 ladrillo y mármol de Candoglia
<input type="checkbox"/>				8 Italia	La Arena	30	30	31	140 piedra caliza blanca y rosa de Veipocella
<input type="checkbox"/>				9 Italia	La Fontana de Trevi	537	1453	25	19 fue construida en su totalidad en mármol de traver...
<input type="checkbox"/>				10 Italia	El Castel dell'Ovo	1128	XVI	35	200 Ladrillo, Piedra
<input type="checkbox"/>				11 Japón	Templo Sensō-ji	628	645	60	1 Piedra, Madera
<input type="checkbox"/>				12 Japón	Fushimi Inari Taisha	711	711	60	4 Bambú, madera
<input type="checkbox"/>				13 Japón	Tumba y estatua de Minamoto no Yoritomo	1199	1199	2	4 Piedra
<input type="checkbox"/>				14 Japón	Castillo de Odawara	1418	1706	40	4 Piedra, madera, metal, barro
<input type="checkbox"/>				15 Japón	Castillo de Osaka y estatua de Toyotomi Hideyoshi	XVI	1583	8	4 Madera, piedra, yeso
<input type="checkbox"/>				16 Japón	Museo Edo Tokyo	1992	1993	20	60 concreto, ladrillos
<input type="checkbox"/>				17 Japón	Nikkō Toshō-gu	1636	1617	5	40 madera, bambú, metal
<input type="checkbox"/>				18 Japón	Gasho-zukuri de Ogimachi	1	1	4	6 madera, cuerda, palmas
<input type="checkbox"/>				19 Japón	Fuerte Goryōkaku	1857	1866	90	10 Piedra, madera
<input type="checkbox"/>				20 Estados Unidos	National Mall	1791	1793	300	56 Piedra, mármol, metal

En este se tomo en cuenta el diseño visual de la página, las imágenes, las letras y los botones utilizados en su creación, también se implementó los Enlace de las imágenes que se utilizaron, datos del monumento además del botón que se utiliza para ver dichos datos, la implementación del vínculo para que mande a otra página y la visualización de las imágenes.

[illegible]

Back End

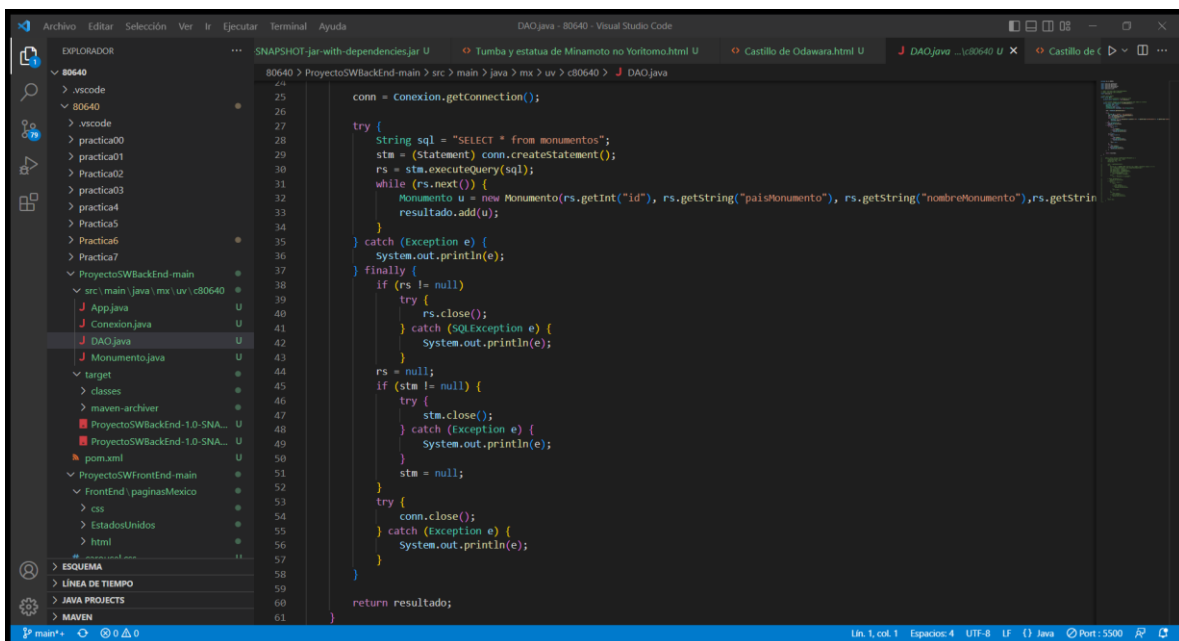
Java, este se encarga de la conexión por puertos, y manda a llamar el script de java DAO, en este se muestran los monumentos que se van a utilizar, se muestra una lista de ellos, con el request se muestran el acceso, se busca el course para que se comparen los id y en caso de que encuentre una coincidencia, muestra la coincidencia y si no la hay regresa a no encontrar el valor



```
1 package mx.uv.c80640;
2
3 import static spark.Spark.*;
4 import com.google.gson.*;
5
6 import java.util.List;
7 import java.util.UUID;
8
9 /**
10  * Hello world!
11  *
12  */
13 public class App
14 {
15     public static Gson gson = new Gson();
16     public static void main( String[] args )
17     {
18         port(getHerokuAssignedPort());
19         //System.out.println( "Hello world!" );
20         options("/*", (request, response) -> {
21             String accessControlRequestHeaders = request.headers("Access-Control-Request-Headers");
22             if (accessControlRequestHeaders != null) {
23                 response.header("Access-Control-Allow-Headers", accessControlRequestHeaders);
24             }
25             String accessControlRequestMethod = request.headers("Access-Control-Request-Method");
26             if (accessControlRequestMethod != null) {
27                 response.header("Access-Control-Allow-Methods", accessControlRequestMethod);
28             }
29             return "OK";
30         });
31         before((req, res) -> res.header("Access-Control-Allow-Origin", "*"));
32         before((req, res) -> res.type("application/json"));
33
34         // options("/*", (request, response) -> {
35         //     String accessControlRequestHeaders = request.headers("Access-Control-Request-Headers");
36         //     if (accessControlRequestHeaders != null) {
37         //         response.header("Access-Control-Allow-Headers", accessControlRequestHeaders);
38         //     }
39         // });
```

DAO

Hace la conexión con la base de datos, consulta la tabla de monumentos, hace una comparación y una consulta de los datos y muestra una lista de los datos solicitados.

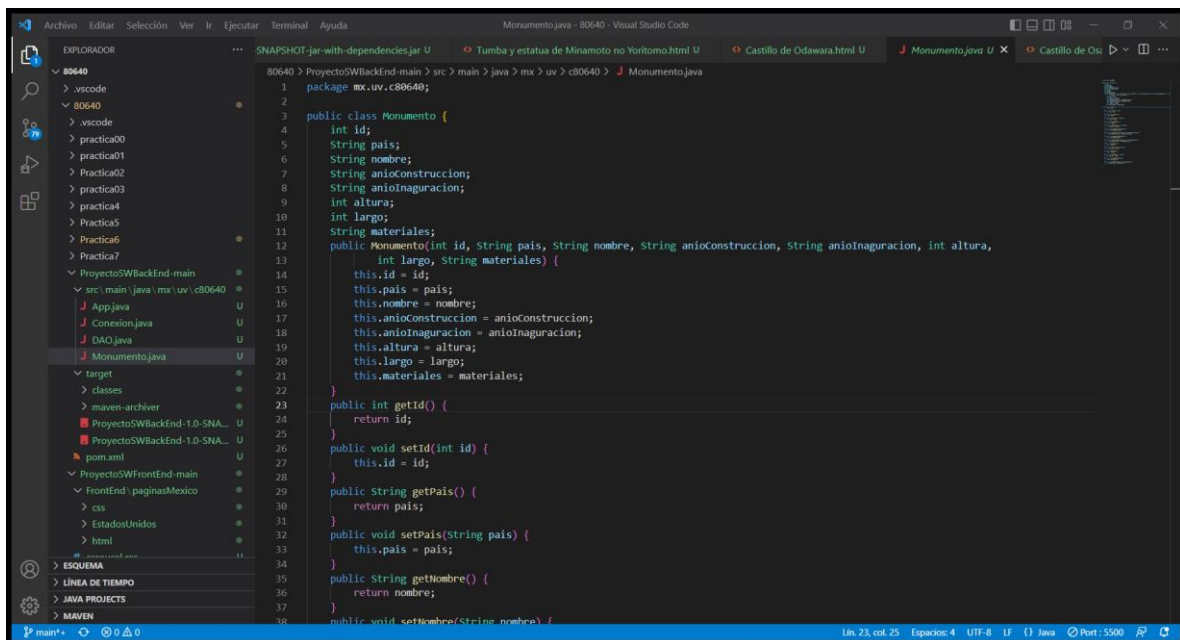


```
25 conn = Conexion.getConnection();
26
27 try {
28     String sql = "SELECT * from monumentos";
29     stm = (Statement) conn.createStatement();
30     rs = stm.executeQuery(sql);
31     while (rs.next()) {
32         Monumento u = new Monumento(rs.getInt("id"), rs.getString("paisMonumento"), rs.getString("nombreMonumento"), rs.getString("descripcionMonumento"));
33         resultado.add(u);
34     }
35 } catch (Exception e) {
36     System.out.println(e);
37 } finally {
38     if (rs != null) {
39         try {
40             rs.close();
41         } catch (SQLException e) {
42             System.out.println(e);
43         }
44     }
45     rs = null;
46     if (stm != null) {
47         try {
48             stm.close();
49         } catch (Exception e) {
50             System.out.println(e);
51         }
52     }
53     stm = null;
54     try {
55         conn.close();
56     } catch (Exception e) {
57         System.out.println(e);
58     }
59 }
60 return resultado;
61 }
```

Monumento

Es una clase monumento, en esta se aplican los getters a setters para que se llene la clase con cada uno de los atributos correspondientes, es un objeto base para el monumento que se presenta en la pagina donde su objetivo es mostrar las características que presenta cada

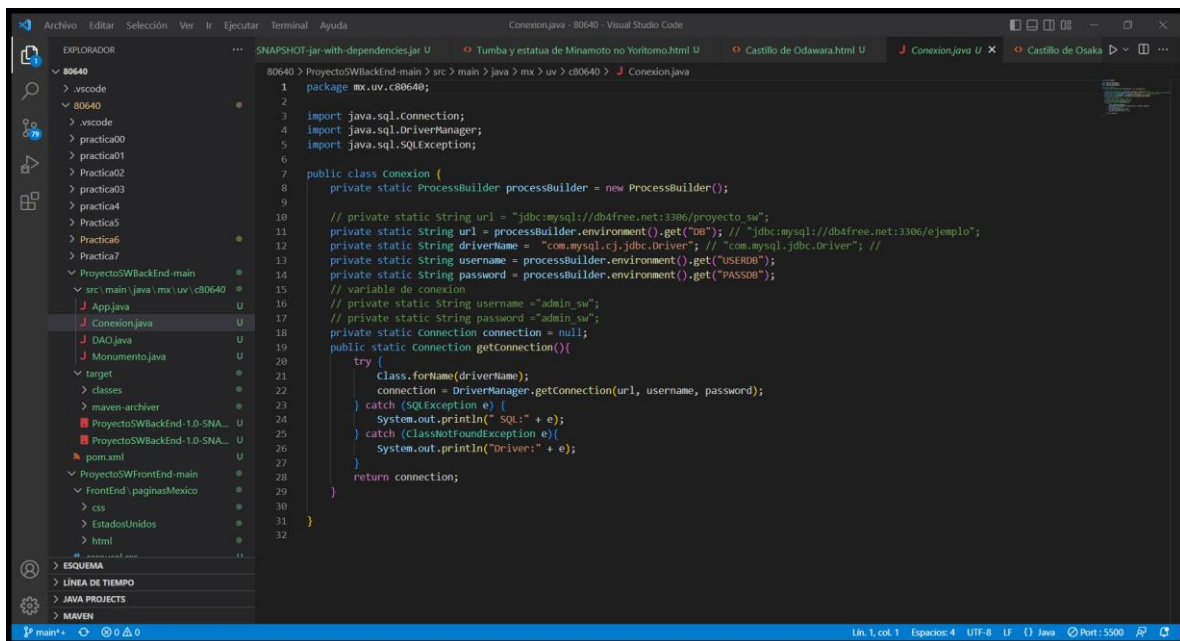
monumento.



```
1 package mx.uv.c80640;
2
3 public class Monumento {
4     int id;
5     String pais;
6     String nombre;
7     String anioConstruccion;
8     String anioInaguracion;
9     int altura;
10    int largo;
11    String materiales;
12    public Monumento(int id, String pais, String nombre, String anioConstruccion, String anioInaguracion, int altura,
13        int largo, String materiales) {
14        this.id = id;
15        this.pais = pais;
16        this.nombre = nombre;
17        this.anioConstruccion = anioConstruccion;
18        this.anioInaguracion = anioInaguracion;
19        this.altura = altura;
20        this.largo = largo;
21        this.materiales = materiales;
22    }
23    public int getId() {
24        return id;
25    }
26    public void setId(int id) {
27        this.id = id;
28    }
29    public String getPais() {
30        return pais;
31    }
32    public void setPais(String pais) {
33        this.pais = pais;
34    }
35    public String getNombre() {
36        return nombre;
37    }
38    public void setNombre(String nombre) {
```

Conexión

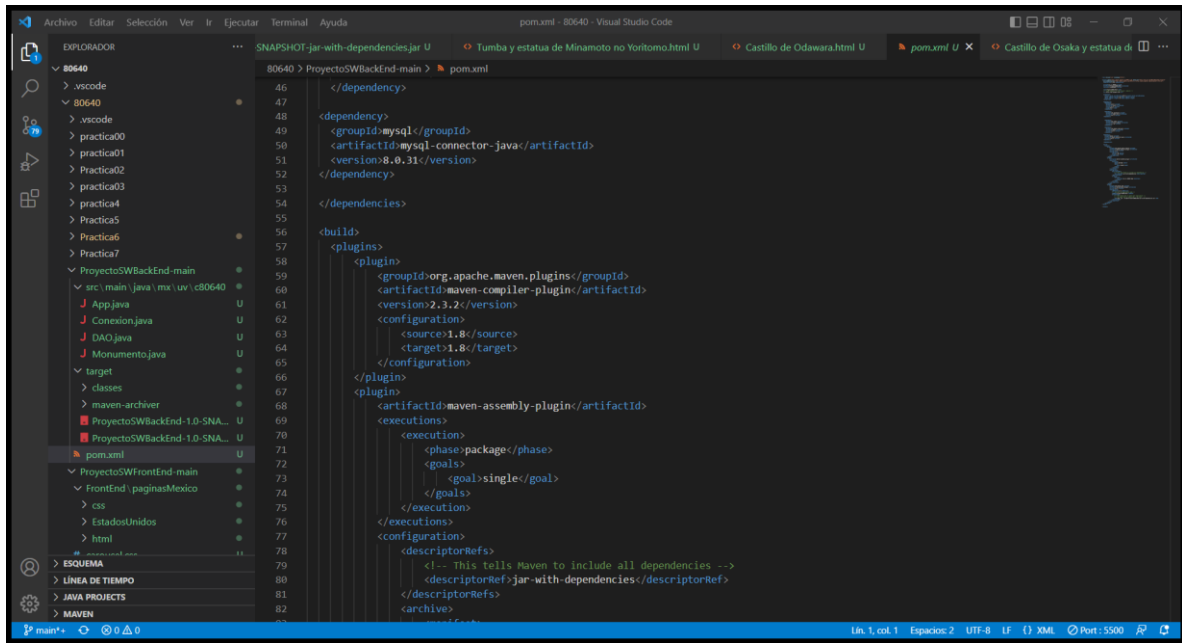
Esta es la parte para la conexión a la base de datos la cual esta en la nube, en la base se datos se pueden realizar las consultas, eliminar, editar o agregar una clase monumento respectivamente



```
1 package mx.uv.c80640;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class Conexion {
8     private static ProcessBuilder processBuilder = new ProcessBuilder();
9
10    // private static String url = "jdbc:mysql://db4free.net:3306/proyecto_sw";
11    private static String url = processBuilder.environment().get("DB"); // "jdbc:mysql://db4free.net:3306/ejemplo";
12    private static String driverName = "com.mysql.cj.jdbc.Driver"; // "com.mysql.jdbc.Driver"; //
13    private static String username = processBuilder.environment().get("USERDB");
14    private static String password = processBuilder.environment().get("PASSWORD");
15    // variable de conexion
16    // private static String username = "admin_sw";
17    // private static String password = "admin_sw";
18    private static Connection connection = null;
19    public static Connection getConnection(){
20        try {
21            Class.forName(driverName);
22            connection = DriverManager.getConnection(url, username, password);
23        } catch (SQLException e) {
24            System.out.println("SQL:" + e);
25        } catch (ClassNotFoundException e){
26            System.out.println("Driver:" + e);
27        }
28        return connection;
29    }
30
31 }
32
```

Pom.xml

Estas son las dependencias que se utilizaron, como la jason, spark , MYQL conector entre otros, estas son las utilizadas para las librerías del proyecto para hacer las conexiones y los objetos utilizados



Implementación

Se conecta con la base de datos, busca el Id, cuando recibe la respuesta hace una tabla con los datos que encontró, en caso de que no encontrar el estatus será nulo y no se podrá generar la tabla

