Universidad Autónoma de Nuevo León.
Facultad de Ciencias Físico Matemáticas.

Práctica #11 – Servlet.

Laboratorio de Programación Orientada a Objetos.

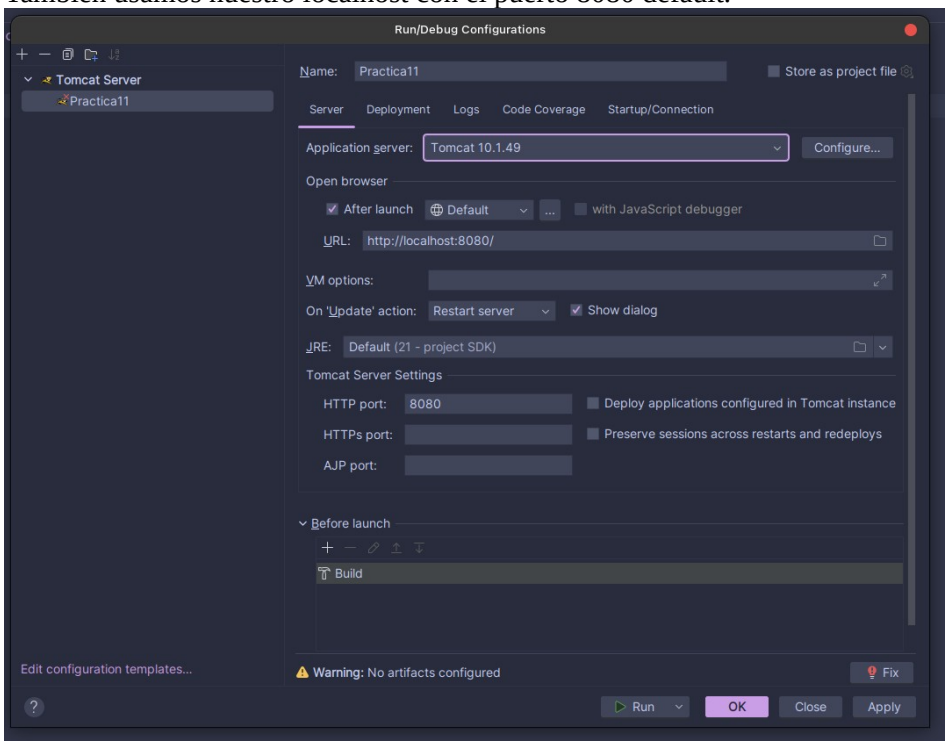
Alumno: Jeremy Uriel Rossell Segura.
Matrícula: 2173396.
Grupo: 036.
Horario: VIERNES 7:00-9:00.
Materia: Laboratorio de Programación Orientada a objetos.
Docente: Jorge Alberto Islas Pineda.

Inicialización.

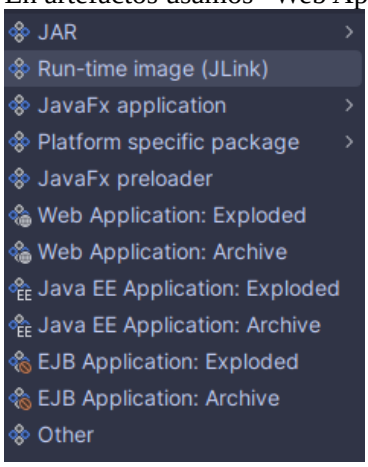
Esta práctica es un poco diferente al resto, ya que aquí no solo hacemos un programa Java local, sino que configuramos un web server, con servlet, usando Tomcat 10, y en mi caso usando IntelliJ como mi IDE, y Jakarta EE 10 como mi proyecto, que este mismo usa Maven. El profesor usó NetBeans con Ant, que funciona igual pero es más básico.

Primero creamos nuestro proyecto IntelliJ con Jakarta EE 10, y usamos Tomcat 10.1.49 como nuestro application server, y elegimos el local. Descargué el tar.gz de la web oficial de Tomcat, y lo guardé en mi directorio ~/jetbrains/Apache/apache-tomcat-10.1.49, como se aprecia, es directorio con permiso de usuario, esto es porque es mi computadora personal, si quisieramos usarlo en un servidor, ahí no usamos el tar.gz, sino el package manager de nuestra distribución Linux, como Debian es el rey de los servidores, usaríamos sudo apt install tomcat10, o si hay una versión nueva checar usando sudo apt update && apt search tomcat.

Usamos el Tomcat de forma local para no rompernos la cabeza con los puertos, ni configurar el servidor, ni permisos de root ni nada de eso. Simplemente buscamos la carpeta ya extraída en el directorio ya dado. También usamos nuestro localhost con el puerto 8080 default.



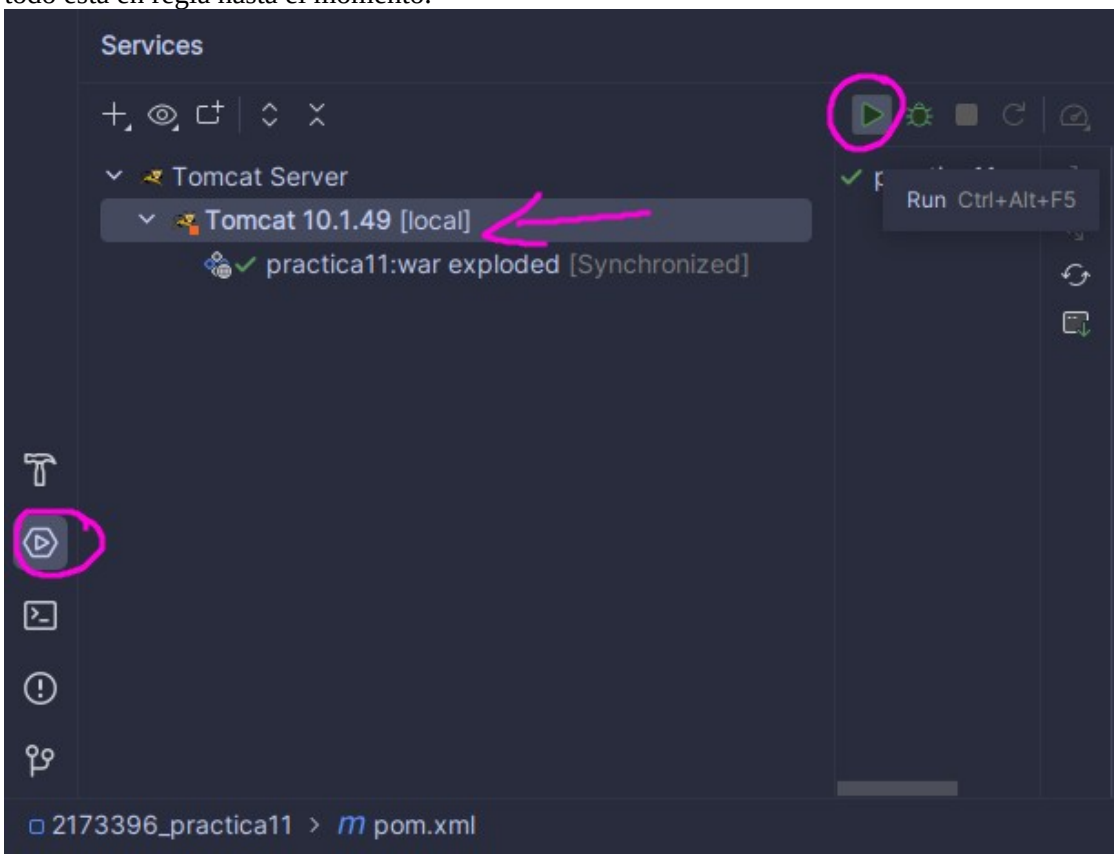
En artefactos usamos “Web Application: Exploded”



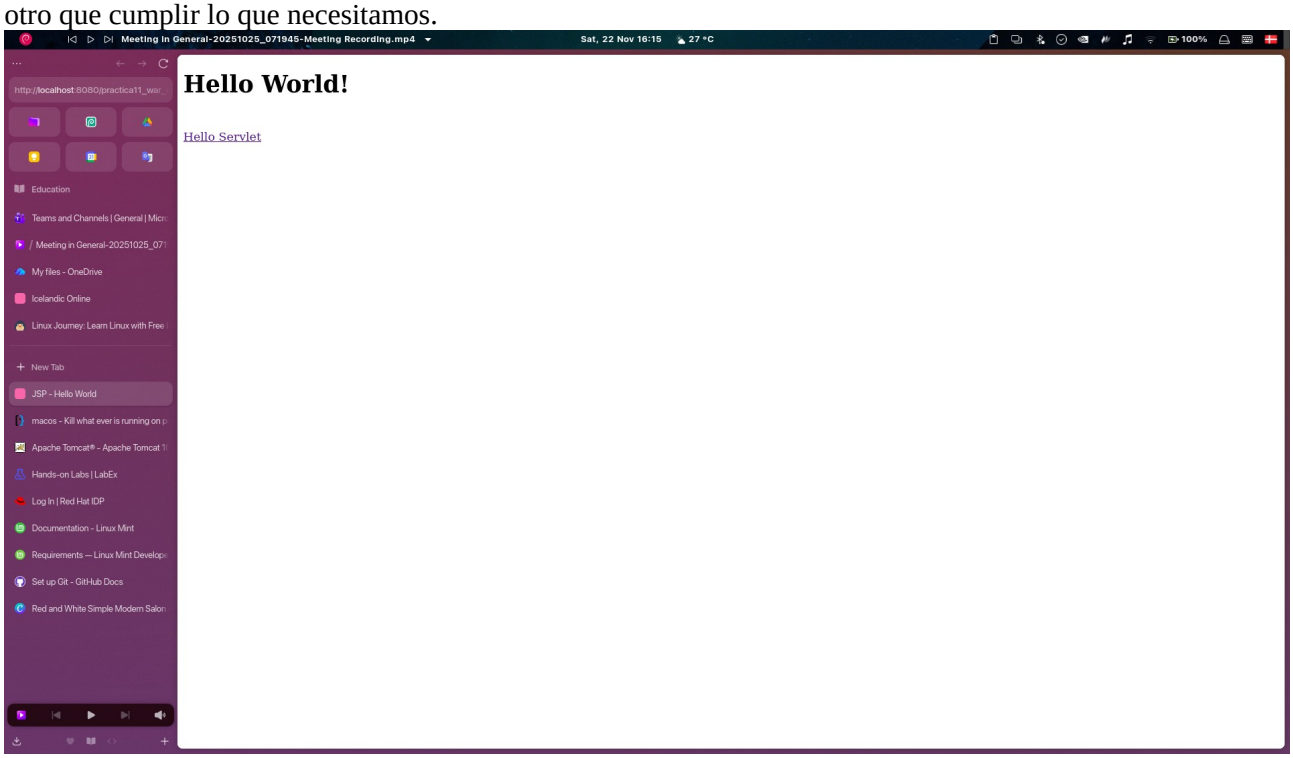
Ahora rellenamos los campos de groupId con un paquete, si estamos en producción puede ser el nombre del proyecto o empresa, como soy yo en un ámbito académico, usaré “dev.jeremyrossell” como el paquete. Y en artifactId ponemos el nombre de nuestro proyecto “practica11”. El packaging por default debe ser .war.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>dev.jeremyrossell</groupId>
  <artifactId>practica11</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>2173396_practica11</name>
  <packaging>war</packaging>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.target>21</maven.compiler.target>
    <maven.compiler.source>21</maven.compiler.source>
    <junit.version>5.13.2</junit.version>
  </properties>

  <dependencies>
<dependency>
      <groupId>jakarta.servlet</groupId>
      <artifactId>jakarta.servlet-api</artifactId>
      <version>6.1.0</version>
      <scope>provided</scope>
    </dependency>
<dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
    </dependency>
      <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
```

Una vez chequemos que nuestro pom.xml esté correcto, vamos a la sección de servicios, seleccionamos nuestro Tomcat server, y la instancia local, y le damos a Run para correr nuestro servidor web y probar que todo está en regla hasta el momento.

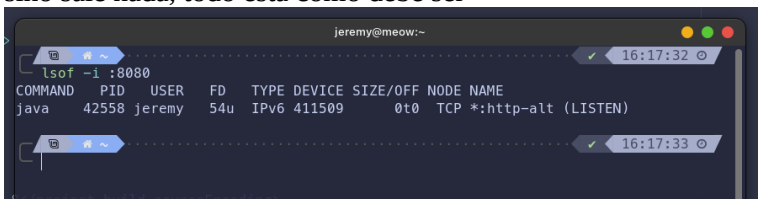


Y como podemos apreciar, si funcionó, este html es uno que se generó por defecto, pero nosotros crearemos otro que cumplir lo que necesitamos.

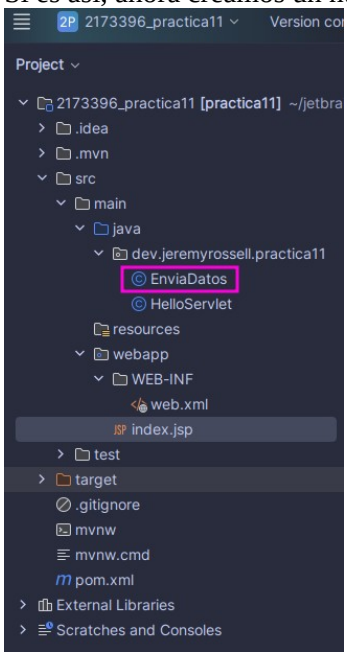


Código.

Ya que tenemos nuestro proyecto con todo “bien conectado”, entonces ya iniciamos con lo que queremos. Primero cerramos el proceso del Tomcat, y nos aseguramos que nuestro puerto 8080 del localhost no está siendo usado por nuestra instancia Tomcat en Java o alguna otra, si sale, debemos cerrar bien el servidor, sino sale nada, todo está como debe ser



Si es así, ahora creamos un nuevo archivo Servlet en nuestro package llamado EnviaDatos.



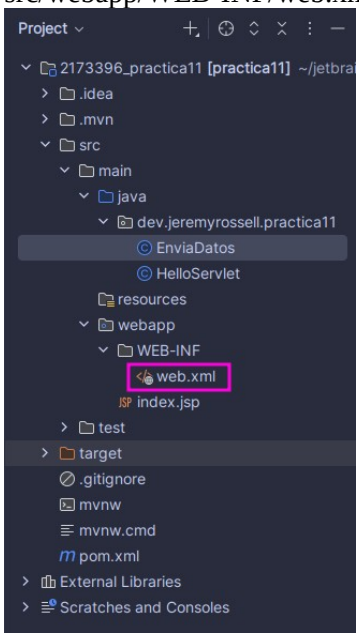
Dentro de ese archivo .java, creamos nuestro código que conectará con nuestro index, en mi caso es index.jsp por ser un proyecto Jakarta con Maven, pero en otros se verá como index.html.

pom.xml (practica11) | EnviaDatos.java | web.xml | index.jsp

```java
package dev.jeremyrossell.practica11;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class EnviaDatos extends HttpServlet {  1 usage
    @Override  no usages
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // configurar tipo de respuesta
        resp.setContentType("text/html;charset=UTF-8");

        // obtener los datos del formulario
        String nombre = req.getParameter( s: "nombre");
        String apellido = req.getParameter( s: "apellido");
        String email = req.getParameter( s: "email");
        String telefono = req.getParameter( s: "telefono");

        // imprimir respuesta en el navegador
        try (PrintWriter out = resp.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head><title>Datos Recibidos</title></head>");
            out.println("<body>");
            out.println("<h1>¡Datos recibidos con éxito!</h1>");
            out.println("<p><strong>Nombre:</strong> " + nombre + "</p>");
            out.println("<p><strong>Apellido:</strong> " + apellido + "</p>");
            out.println("<p><strong>Email:</strong> " + email + "</p>");
            out.println("<p><strong>Teléfono:</strong> " + telefono + "</p>");
            out.println("<a href='index.jsp'>Volver</a>");
            out.println("</body>");
            out.println("</html>");
```

jeremyrossell › practica11 › EnviaDatos

Ahora toca reflejar los cambios manualmente en el web.xml, que se encuentra en el directorio src/webapp/WEB-INF/web.xml



Y agregamos el nombre y clase de nuestro Servlet que acabamos de crear. También agregamos un mapping con el nombre, y un patrón URL, que sería el nombre de nuestro Servlet, con un “/” de prefijo.

pom.xml (practica11) | EnviaDatos.java | web.xml | index.jsp

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         version="6.0">
    <servlet>
        <servlet-name>ServletEnviaDatos</servlet-name>
        <servlet-class>dev.jeremyrossell.practica11.EnviaDatos</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ServletEnviaDatos</servlet-name>
        <url-pattern>/EnviaDatos</url-pattern>
    </servlet-mapping>
</web-app>
```

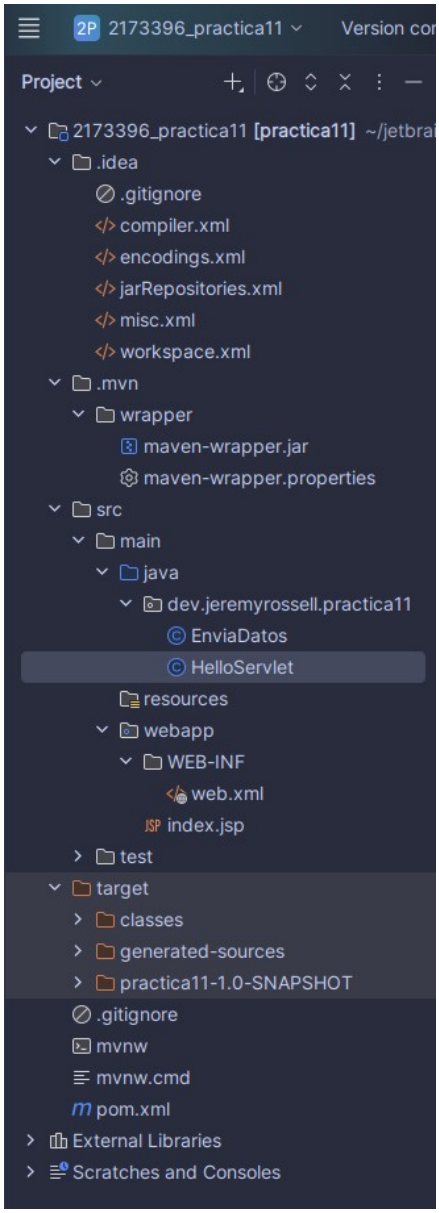
De ahí ahora si nos dirigimos a nuestro index.jsp ubicado en el directorio src/main/webapp/index.jsp, y usando HTML creamos el formulario pedido. Dentro de dicho formulario usamos el método GET y de acción ponemos nuestro EnviaDatos.

pom.xml (practica11) | EnviaDatos.java | web.xml | index.jsp

```jsp
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>Formulario de Registro</title>
</head>
<body>
<h1>Registro de Usuario</h1>

<form action="EnviaDatos" method="GET">

    <label for="nombre">Nombre:</label><br>
    <input type="text" id="nombre" name="nombre" required><br><br>

    <label for="apellido">Apellido:</label><br>
    <input type="text" id="apellido" name="apellido" required><br><br>

    <label for="email">E-mail:</label><br>
    <input type="email" id="email" name="email" required><br><br>

    <label for="telefono">Teléfono:</label><br>
    <input type="tel" id="telefono" name="telefono"><br><br>

    <input type="submit" value="Enviar Datos">

</form>
<body>
</html>
```

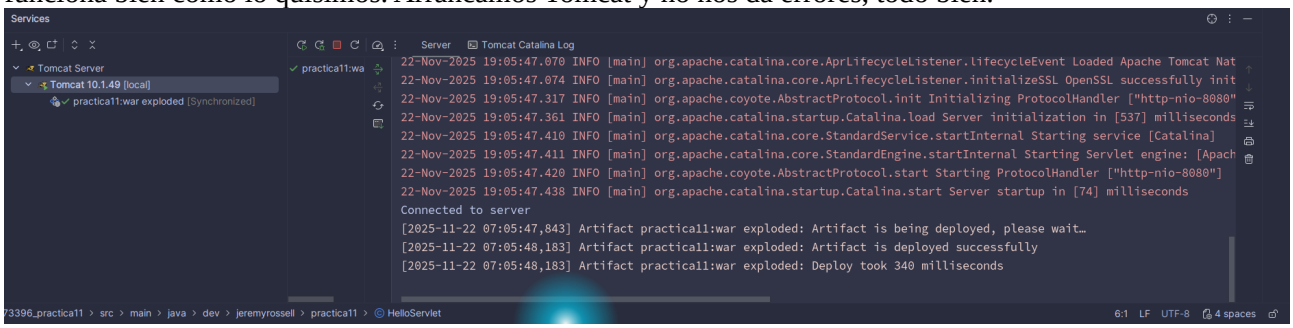
Archivos.

La estructura de nuestros archivos es la estándar, solo creamos uno nuevo en nuestro paquete, el cual es el servlet.

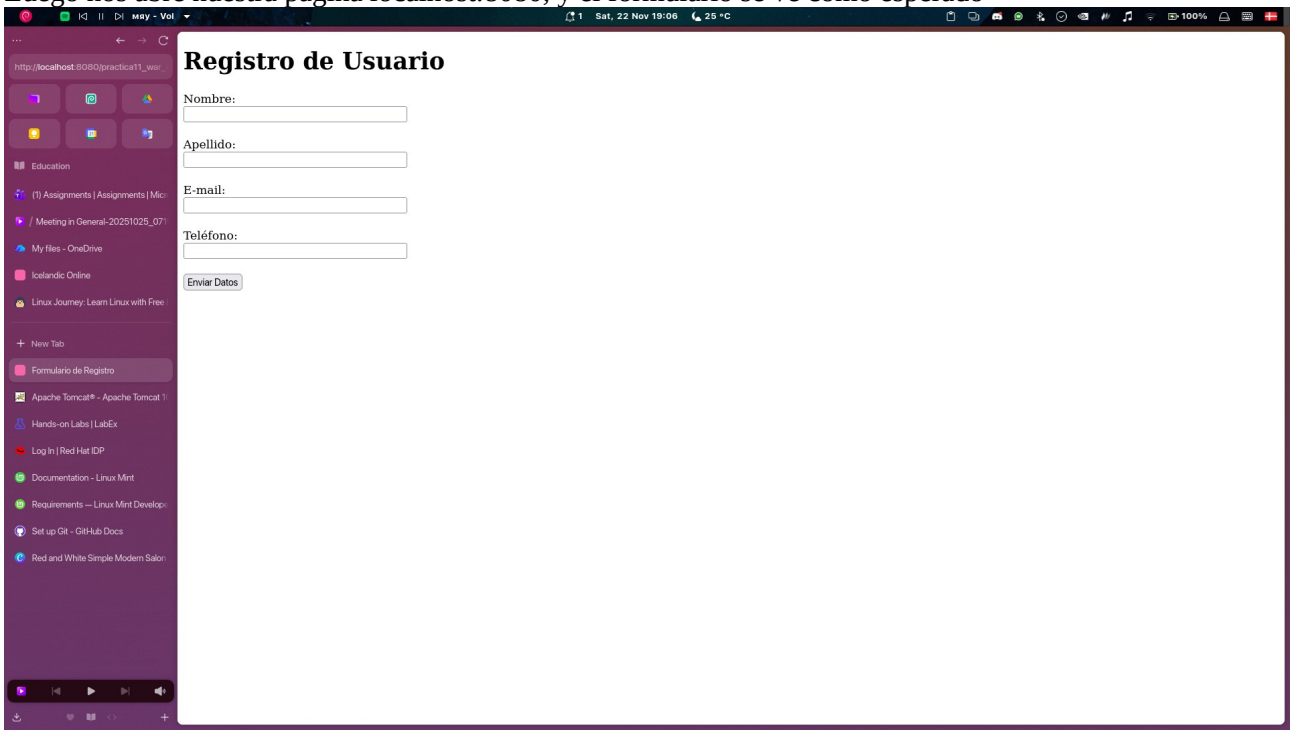


Compilación y prueba.

Ahora viene la prueba de fuego, con esto aseguramos que no solo compila y está depurado, sino que funciona bien como lo quisimos. Arrancamos Tomcat y no nos da errores, todo bien.



Luego nos abre nuestra página localhost:8080, y el formulario se ve como esperado



Rellenamos nuestros datos y le damos a Submit.

Y como era de esperarse, funcionó todo con éxito.

¡Datos recibidos con éxito!

Nombre: Jeremy

Apellido: Rossell

Email: jeremy.rossells@uanl.edu.mx

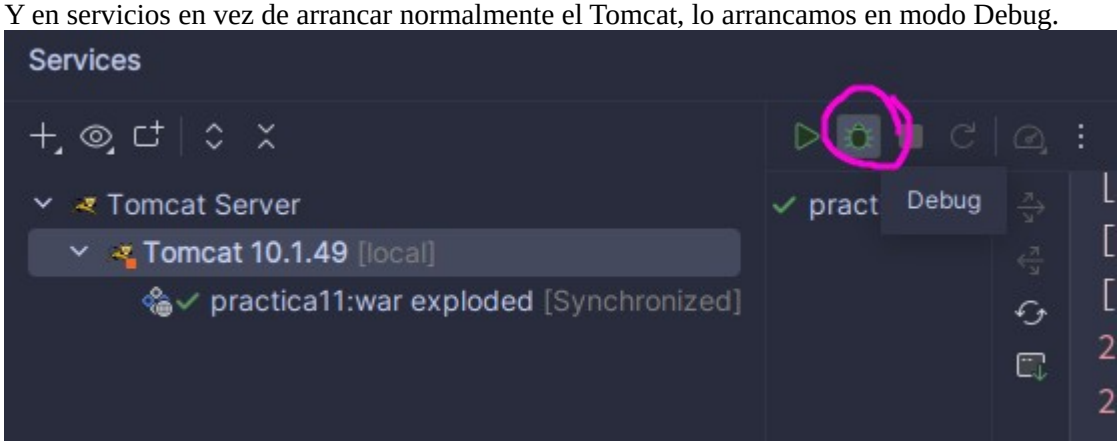
Teléfono: 8120203030

Volver

Depuración.

Si queremos depurar o debuggear, con IntelliJ solo ponemos breakpoints donde queremos revisar el código.

Y en servicios en vez de arrancar normalmente el Tomcat, lo arrancamos en modo Debug.



Y ya así podremos ver qué hace nuestro programa pasito por pasito.