

**CODIFICACIÓN DE MÓDULOS DEL SOFTWARE STAND ALONE, WEB Y MÓVIL DE  
ACUERDO AL PROYECTO A DESARROLLAR**

**GA7-220501096-AA3-EV01**

**PRESENTADO POR:**

**JULIO CESAR LOPEZ PAEZ**

**APRENDIZ**

**PRESENTADO A:**

**JOSE GREGORIO GAITAN TORRES**

**INSTRUCTOR**

**ANALISIS Y DESARROLLO DE SOFTWARE**

**FICHA (2977442)**

**SERVICIO NACIONAL DE APRENDIZAJE SENA**

**FEBRERO DE 2026**

## INDICE

<b>1. INTRUDUCTION .....</b>	<b>3</b>
<b>2. OBJETIVOS.....</b>	<b>4</b>
Objetivo general.....	4
Objetivos específicos.....	4
<b>3. ALCANCE.....</b>	<b>4</b>
<b>4. DESARROLLO DEL MODELO DEL SOFTWARE.....</b>	<b>4</b>
Uso de Servlets y JSP .....	5
Uso de métodos GET y POST.....	5
Frameworks utilizados .....	5
Estándares de codificación .....	5
<b>5. MATRIX EXPLICTIVA POR ARCHIVO .....</b>	<b>6</b>
<b>6. URL GITHUB.....</b>	<b>8</b>
<b>7. CONCLUSIONES: .....</b>	<b>9</b>

## **1. INTRUCTION**

El presente documento académico corresponde a la evidencia GA7-220501096-AA3-EV01 – Codificación de módulos del software Stand alone, web y móvil, desarrollada en el marco del programa de formación del SENA. En este documento se describe el proceso de codificación de un sistema de información web desarrollado en lenguaje Java, aplicando tecnologías estándar para aplicaciones web empresariales.

El proyecto fue construido bajo el patrón de arquitectura Modelo – Vista – Controlador (MVC), utilizando Servlets y JSP como stack web, Apache Maven como gestor de dependencias y herramienta de construcción, y Bootstrap como Framework Frontend para la interfaz gráfica. El sistema implementa control de sesiones, gestión de usuarios y un módulo funcional para el registro y consulta de información técnica, con persistencia de datos en una base de datos MySQL.

## 2. OBJETIVOS

### Objetivo general

Desarrollar y documentar la codificación de los módulos de un sistema de información web en Java, aplicando Frameworks y buenas prácticas de desarrollo de software, de acuerdo con los lineamientos establecidos por el contenido aprendido en el SENA.

### Objetivos específicos

- Implementar un sistema web utilizando Servlets y JSP bajo el patrón MVC.
- Aplicar el framework Apache Maven para la gestión de dependencias y la construcción del proyecto.
- Diseñar una interfaz web responsiva utilizando el framework Bootstrap.
- Implementar el uso de métodos HTTP GET y POST para el intercambio de información entre cliente y servidor.
- Gestionar la persistencia de datos mediante una base de datos MySQL.
- Controlar versiones del código fuente mediante el uso de Git.

## 3. ALCANCE

El proyecto abarca el desarrollo de un sistema de información web que permite la gestión de usuarios y la administración de información técnica por parte del rol Analista. Incluye la implementación de la capa de presentación, la lógica de negocio y el acceso a datos, así como el control de sesiones y la validación básica de usuarios.

No se contempla el despliegue en un servidor de producción ni la implementación de servicios móviles nativos, limitándose a una aplicación web ejecutada en un servidor de aplicaciones Java.

## 4. DESARROLLO DEL MODELO DEL SOFTWARE

El sistema fue desarrollado como una aplicación web Java siguiendo el patrón Modelo – Vista – Controlador (MVC), separando claramente las responsabilidades de cada componente:

- **Modelo:** Representado por las clases Java encargadas de la lógica de negocio y el acceso a datos (DAO), las cuales interactúan con la base de datos MySQL.
- **Vista:** Implementada mediante páginas JSP, encargadas de la presentación de la información al usuario final.
- **Controlador:** Implementado mediante Servlets, los cuales reciben las solicitudes HTTP, procesan la información y determinan la vista correspondiente.

## Uso de Servlets y JSP

Los Servlets actúan como controladores del sistema, gestionando las solicitudes del cliente, validando datos y ejecutando la lógica de negocio. Las JSP se utilizan para renderizar la interfaz gráfica y mostrar la información procesada por los Servlets.

## Uso de métodos GET y POST

- **GET:** Utilizado para solicitudes de consulta de información, navegación entre vistas y recuperación de datos sin alterar el estado del sistema.
- **POST:** Utilizado para el envío de formularios, registro de información, autenticación de usuarios y operaciones que modifican los datos almacenados.

Esta separación garantiza seguridad, claridad y cumplimiento de los estándares HTTP.

## Frameworks utilizados

### Apache Maven

Apache Maven fue utilizado como framework y gestor de dependencias del proyecto. Permite administrar las librerías necesarias, definir el ciclo de vida de construcción y facilitar la compilación y despliegue del sistema mediante el archivo pom.xml.

### Bootstrap

Bootstrap se empleó como framework frontend para el diseño de la interfaz gráfica del sistema. Gracias a este framework se logró una interfaz responsiva, organizada y compatible con diferentes dispositivos.

## Servlets y JSP

El uso combinado de Servlets y JSP constituye el stack web Java estándar para aplicaciones empresariales, permitiendo la implementación del patrón MVC y el desarrollo de aplicaciones web robustas.

## Estándares de codificación

Durante el desarrollo del proyecto se aplicaron los siguientes estándares:

- Nomenclatura de clases en formato PascalCase.
- Métodos y variables en formato camelCase.
- Separación de capas según el patrón MVC.
- Uso de comentarios para documentar métodos y procesos relevantes.
- Organización del proyecto siguiendo la estructura estándar de Maven.

## 5. MATRIX EXPLICITIVA POR ARCHIVO

MATRIZ COMPLETA DE ARCHIVOS – PROYECTO TELEP			
Nombre del archivo	Ruta completa	Función general	Funciones específicas
PasswordUtil.java	src/com/telep/util/PasswordUtil.java	Utilidad de seguridad para contraseñas.	- Genera hash SHA-256.
			- Convierte bytes a hexadecimal.
			- Compara contraseña con hash.
			- Centraliza cifrado.
ConexionBD.java	src/com/telep/util/ConexionBD.java	Gestión de conexión a base de datos.	- Crea conexión JDBC.
			- Maneja driver y credenciales.
			- Centraliza acceso a BD.
Usuario.java	src/com/telep/model/Usuario.java	Modelo de entidad Usuario.	- Define atributos (id, nombre, correo, rol, password).
			- Getters y setters.
			- Representa tabla usuarios.
Registro.java	src/com/telep/model/Registro.java	Modelo de entidad Registro técnico.	- Define datos cliente/equipo.
			- Getters y setters.
			- Representa tabla registros.
UsuarioDAO.java	src/com/telep/dao/UsuarioDAO.java	Acceso a datos de usuarios.	- Insertar usuario.
			- Actualizar usuario.
			- Eliminar usuario.
			- Listar todos.
			- Buscar por ID.
			- Buscar por usuario.
RegistroDAO.java	src/com/telep/dao/RegistroDAO.java	Acceso a datos de registros técnicos.	- Insertar registro.
			- Listar todos.
			- Listar por analista.
			- Consulta completa para admin.
LoginServlet.java	src/com/telep/controller/LoginServlet.java	Controlador de autenticación.	- Recibe POST login.
			- Valida usuario y contraseña.
			- Usa PasswordUtil.
			- Crea sesión.
			- Redirige a dashboard.
LogoutServlet.java	src/com/telep/controller/LogoutServlet.java	Controlador de cierre de sesión.	- Invalida sesión.
			- Redirige a login.
UsuarioServlet.java	src/com/telep/controller/UsuarioServlet.java	Controlador CRUD usuarios.	- Crear usuario.
			- Actualizar usuario.
			- Eliminar usuario.
			- Redirecciones con mensajes.

<b>RegistroServlet.java</b>	src/com/telep/controller/RegistroServlet.java	Controlador registro técnico.	<ul style="list-style-type: none"> <li>- Recibe datos formulario.</li> <li>- Guarda registro en BD.</li> <li>- Asocia registro al analista.</li> <li>- Maneja mensajes Toast.</li> </ul>
<b>ModoAnalistaServlet.java</b>	src/com/telep/controller/ModoAnalistaServlet.java	Controlador cambio de modo admin.	<ul style="list-style-type: none"> <li>- Activa modoAnalista.</li> <li>- Desactiva modoAnalista.</li> <li>- Modifica atributo de sesión.</li> </ul>
<b>login.jsp</b>	webapp/login.jsp	Vista pública de autenticación.	<ul style="list-style-type: none"> <li>- Formulario login.</li> <li>- Envía datos a LoginServlet.</li> <li>- Muestra error si falla autenticación.</li> </ul>
<b>dashboard.jsp</b>	webapp/dashboard.jsp	Panel principal dinámico.	<ul style="list-style-type: none"> <li>- Valida sesión.</li> <li>- Detecta rol.</li> <li>- Muestra menú según perfil.</li> <li>- Maneja modoAnalista.</li> </ul>
<b>editarUsuario.jsp</b>	webapp/admin/editarUsuario.jsp	Vista edición de usuarios.	<ul style="list-style-type: none"> <li>- Obtiene ID.</li> <li>- Carga datos desde DAO.</li> <li>- Envía actualización.</li> </ul>
<b>usuarios.jsp</b>	webapp/admin/usuarios.jsp	Vista gestión usuarios.	<ul style="list-style-type: none"> <li>- Lista usuarios.</li> <li>- Permite editar/eliminar.</li> <li>- Acceso solo admin.</li> </ul>
<b>registros.jsp</b>	webapp/admin/registros.jsp	Vista registros completos (admin).	<ul style="list-style-type: none"> <li>- Lista todos los registros.</li> <li>- Consulta general.</li> </ul>
<b>registro.jsp</b>	webapp/analista/registro.jsp	Formulario registro técnico.	<ul style="list-style-type: none"> <li>- Valida rol 2.</li> <li>- Captura datos cliente/equipo.</li> <li>- Envía a RegistroServlet.</li> </ul>
<b>misRegistros.jsp</b>	webapp/analista/misRegistros.jsp	Vista registros del analista.	<ul style="list-style-type: none"> <li>- Lista registros propios.</li> <li>- Consulta por usuario logueado.</li> </ul>
<b>listar_registros.jsp</b>	webapp/analista/listar_registros.jsp	Alternativa listado analista.	<ul style="list-style-type: none"> <li>- Muestra registros filtrados.</li> <li>- Consulta personalizada.</li> </ul>
<b>header.jsp</b>	webapp/includes/header.jsp	Encabezado reutilizable global.	<ul style="list-style-type: none"> <li>- Carga Bootstrap.</li> <li>- Navbar principal.</li> <li>- Enlace Dashboard y Logout.</li> </ul>
<b>footer.jsp</b>	webapp/includes/footer.jsp	Pie de página reutilizable.	<ul style="list-style-type: none"> <li>- Muestra copyright.</li> <li>- Maneja Toast.</li> <li>- Inicializa Bootstrap JS.</li> </ul>
<b>telep.css</b>	webapp/css/telep.css	Estilos globales corporativos.	<ul style="list-style-type: none"> <li>- Define colores.</li> <li>- Estiliza tarjetas, navbar y botones.</li> <li>- Consistencia visual.</li> </ul>

## **6. URL GITHUB**

De conformidad con lo solicitado en la evidencia, se procede a la herramienta de repositorio y versionamiento GIBHUB, a continuación, dejo la URL:

[https://github.com/jclopezpaez-beep/JULIOLOPEZ\\_AA3\\_EVA02.git](https://github.com/jclopezpaez-beep/JULIOLOPEZ_AA3_EVA02.git)



## **7. CONCLUSIONES:**

El desarrollo del proyecto permitió aplicar de manera práctica los conceptos de programación web en Java, haciendo uso de frameworks y buenas prácticas de desarrollo de software. La implementación del patrón MVC facilitó la organización del código y la mantenibilidad del sistema.

El uso de Apache Maven y Bootstrap cumplió con los requerimientos de la evidencia en cuanto a la aplicación de frameworks, mientras que el control de versiones con Git permitió llevar un seguimiento ordenado del desarrollo. En conclusión, el proyecto cumple con los objetivos planteados y demuestra la correcta codificación de módulos de software web.