# Introducción

Esta aplicación web ha sido desarrollada para gestionar contenido dinámico basado en diferentes roles de usuario, otorgando funcionalidades específicas para administradores, editores, suscriptores y compradores de anuncios. La plataforma facilita la creación y administración de revistas digitales, además de permitir la interacción entre los usuarios mediante comentarios, suscripciones y la compra de anuncios.

Para su desarrollo, se han empleado tecnologías modernas y robustas ampliamente utilizadas en el ámbito empresarial y web. Estas incluyen:

- **HTML**: Lenguaje de marcado responsable de estructurar y organizar el contenido de las páginas web.
- **Bootstrap**: Framework de CSS que permite diseñar interfaces visuales responsivas y atractivas, asegurando una visualización adecuada en todo tipo de dispositivos, desde móviles hasta pantallas de escritorio.
- **JSP (JavaServer Pages)**: Tecnología que permite la integración de código Java en páginas web, generando contenido dinámico en función de los datos del servidor.
- **Servlets**: Componentes Java que gestionan las solicitudes y respuestas HTTP, actuando como intermediarios entre el cliente y la lógica de negocio de la aplicación.
- **JDBC (Java Database Connectivity)**: API que facilita la conexión entre la aplicación Java y la base de datos MySQL, permitiendo ejecutar consultas y mantener la persistencia de los datos.
- **MySQL**: Sistema de gestión de bases de datos relacional que almacena y gestiona la información de los usuarios, revistas, comentarios, anuncios, y otros elementos clave de la aplicación.
- **Jakarta EE**: Plataforma empresarial robusta para el desarrollo de aplicaciones escalables en Java, que ofrece un conjunto completo de servicios y herramientas para construir aplicaciones web eficientes y seguras.
- **Java**: El lenguaje de programación utilizado para implementar la lógica de negocio y las funcionalidades de la aplicación, proporcionando seguridad, estabilidad y escalabilidad.
- **JSTL (JSP Standard Tag Library)**: Biblioteca estándar de etiquetas JSP que permite incluir lógica de programación en las páginas web de forma simplificada, reduciendo la necesidad de código Java en las vistas.

## Detalles de Herramientas y Tecnologías

### 1. Jakarta EE

- **Versión utilizada**: Jakarta EE 10

### 2. Apache Tomcat

- **Versión utilizada**: Tomcat 10

### 3. MySQL

- **Versión utilizada**: MySQL 8.0

### 4. JDBC (Java Database Connectivity)

- **Versión utilizada**: JDBC 4.3

### 5. NetBeans IDE

- **Versión utilizada**: NetBeans 12.6

### 6. HTML

- **Versión utilizada**: HTML5

### 7. Bootstrap

- **Versión utilizada**: Bootstrap 5

### 8. JSTL (JSP Standard Tag Library)

- **Versión utilizada**: JSTL 1.2

## Sección 1: Diseño de la Base de Datos

### Estructura de la base de datos

La base de datos de la aplicación web está organizada en varias tablas que representan las entidades clave de la plataforma. A continuación, se describen las principales tablas y su función en el sistema.

### Tablas principales

1. **suarios**
   Esta tabla contiene la información de los usuarios del sistema y su rol asignado. Los roles definen el acceso y las funcionalidades disponibles en la plataforma.

   - **Columnas principales:**
     - `nombre_usuario`: Identificador único para cada usuario.
     - `password_usuario`: Contraseña encriptada del usuario.
     - `rol_usuario`: Define el rol del usuario (COMPRADOR, SUSCRIPTOR, EDITOR, ADMINISTRADOR).
     - `nombre_pila`: Nombre del usuario.
     - `descripcion_usuario`: Descripción opcional del usuario.
     - `id_foto`: Referencia a la tabla `fotos_usuario` para almacenar la imagen del perfil.
2. **Revistas**
   Tabla que almacena la información sobre las revistas disponibles en la plataforma.

   - **Columnas principales:**
     - `id_revista`: Identificador único para cada revista.
     - `id_archivo`: Relación con la tabla `archivos_revista`, que almacena archivos multimedia asociados con la revista.
     - `id_categoria`: Relación con la tabla `categorias`, para clasificar las revistas.
     - `titulo_revista`, `nombre_autor`, `descripcion`: Información textual relacionada con la revista.
     - `estado_revista`: Define el estado de la revista (ACTIVA, INACTIVA, EN_ESPERA).
     - `costo_mantenimiento`: Costo asociado a la revista.
3. **Categorías**
   Tabla que define las diferentes categorías a las que pueden pertenecer las revistas.

   - **Columnas principales:**
     - `id_categoria`: Identificador único para la categoría.
     - `nombre_categoria`: Nombre de la categoría.
4. **Etiquetas**
   Tabla para almacenar las etiquetas que se pueden asignar a las revistas para una mejor clasificación.

   - **Columnas principales:**
     - `id_etiqueta`: Identificador único para la etiqueta.

    - `nombre_etiqueta`: Nombre de la etiqueta.
5. **Anuncios**
   Tabla utilizada para almacenar los anuncios comprados por los usuarios, que se pueden visualizar en la plataforma.
    - **Columnas principales:**
        - `id_anuncio`: Identificador único para el anuncio.
        - `nombre_usuario`: Relación con el usuario que ha comprado el anuncio.
        - `texto`, `imagen`, `video`: Contenido multimedia del anuncio.
6. **Suscripciones**
   Almacena la relación entre los usuarios y las revistas a las que están suscritos.
    - **Columnas principales:**
        - `id_suscripcion`: Identificador único para la suscripción.
        - `nombre_usuario`: Relación con la tabla `usuarios`.
        - `id_revista`: Relación con la tabla `revistas`.
        - `fecha_suscripcion`: Fecha en la que el usuario se suscribió a la revista.
7. **Preferencias de Usuario**
   Almacena las preferencias de los usuarios, que pueden incluir hobbies, gustos y temas de interés.
    - **Columnas principales:**
        - `id_preferencia`: Identificador único de la preferencia.
        - `nombre_usuario`: Relación con la tabla `usuarios`.
        - `tipo_preferencia`: Tipo de preferencia (HOBBIE, GUSTO, TEMA_PREFERENCIA).
        - `valor_preferencia`: Valor de la preferencia.
8. **Comentarios**
   Tabla que contiene los comentarios que los usuarios pueden dejar en las revistas.
    - **Columnas principales:**
        - `id_comentario`: Identificador único del comentario.
        - `comentario`: Contenido del comentario.
        - `nombre_usuario`: Relación con la tabla `usuarios`.
        - `id_revista`: Relación con la tabla `revistas`.
9. **Likes de Revistas**
   Almacena la cantidad de "likes" que cada revista ha recibido de los usuarios.
    - **Columnas principales:**
        - `id_revista`: Relación con la tabla `revistas`.
        - `numero_likes`: Número de "likes" recibidos.
        - `nombre_usuario`: Relación con el usuario que dio "like".
10. **Carteras Digitales**
    Tabla utilizada para gestionar los fondos disponibles de los usuarios que realizan compras en la plataforma.

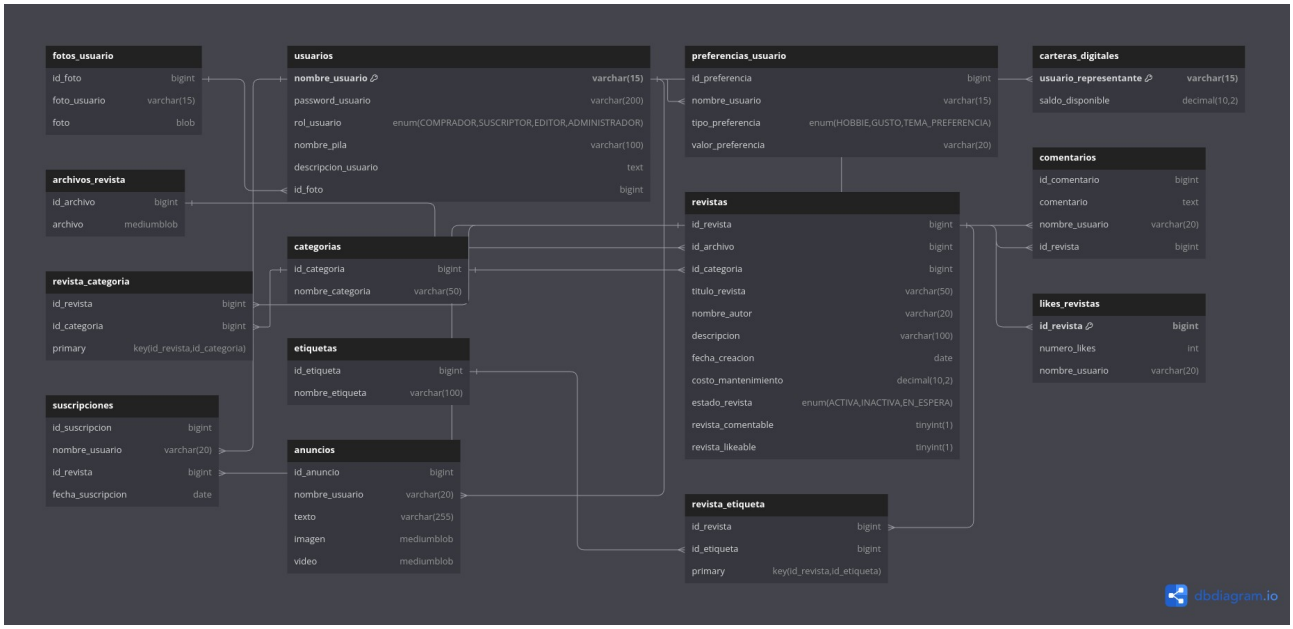
- **Columnas principales:**
    - `usuario_representante`: Relación con la tabla `usuarios`.

- `saldo_disponible`: Monto disponible en la cuenta del usuario.

### 7.3. Relaciones entre tablas

Las tablas del sistema están interrelacionadas para asegurar la integridad de los datos. Algunas de las relaciones clave incluyen:

- **Usuarios y Fotos**: Un usuario puede tener asociada una foto de perfil en la tabla `fotos_usuario`.
- **Revistas y Categorías**: Cada revista pertenece a una categoría específica, lo que se gestiona mediante la relación con la tabla `categorias`.
- **Revistas y Etiquetas**: Las revistas pueden tener varias etiquetas asignadas, almacenadas en la tabla `revista_etiqueta`.
- **Usuarios y Suscripciones**: Los usuarios pueden suscribirse a varias revistas, y esta relación se maneja a través de la tabla `suscripciones`.
- **Usuarios y Anuncios**: Los usuarios pueden crear y gestionar anuncios, que se almacenan en la tabla `anuncios`



## Sección 2: Diseño de la aplicacion

La aplicación ha sido desarrollada en **Jakarta EE**, utilizando tecnologías como **JSP, Servlets, JDBC, y AJAX**, entre otras. El diseño de la arquitectura Java está basado en la división por roles, donde los distintos tipos de usuarios (Administrador, Editor, Suscriptor, Comprador de anuncios) tienen acceso a diferentes funcionalidades del sistema.

### Descripción del Diagrama de Clases

El siguiente diagrama de clases refleja la estructura general del sistema, mostrando las clases clave que componen la lógica de la aplicación. Aquí se presentan algunas de las clases más importantes del sistema y su relación con otros componentes.

1. **Clases de Usuario y Roles:**
    - La clase **Usuario** representa a los usuarios de la plataforma, con atributos como `nombreUsuario`, `password`, y `rolUsuario` (comprador, editor, suscriptor, administrador).
    - La clase **RolUsuario** se utiliza para definir los diferentes permisos y accesos de cada tipo de usuario. Se implementan interfaces específicas para cada rol (por ejemplo, `Comprador`, `Editor`, `Suscriptor`, `Administrador`).
2. **Manejo de Contenido:**
    - La clase **Revista** es central en el sistema, y maneja la lógica de las publicaciones. Tiene atributos como `titulo`, `descripcion`, `estadoRevista`, y referencias a otras clases como **Categoría** y **Etiquetas**.
    - **Archivos** está asociado con las revistas para almacenar documentos o archivos multimedia (PDFs, imágenes, etc.).
3. **Clases para la Gestión de Anuncios:**
    - La clase **Anuncio** permite a los usuarios comprar espacios publicitarios, con atributos como `idAnuncio`, `texto`, `imagen`, y `video`.
4. **Clases para el Sistema de Comentarios:**
    - **ComentarioRevista** permite a los usuarios dejar comentarios en las revistas, mientras que **LikeRevista** gestiona los "me gusta" que recibe cada publicación.
    - Estas clases están relacionadas con las revistas y los usuarios para gestionar las interacciones.
5. **Cartera Digital:**
    - La clase **CarteraDigital** maneja el saldo disponible para los usuarios que desean comprar anuncios en la plataforma.
6. **Controladores y Servicios:**
    - **ServletControlador** y los diversos servicios de la aplicación manejan la lógica de negocio y la interacción entre la base de datos y la capa de presentación. Estos controladores utilizan **AJAX** para actualizar dinámicamente el contenido sin recargar toda la página.
    - La clase **RevistaController** gestiona las solicitudes relacionadas con las revistas (crear, editar, eliminar), mientras que **ComentarioController** se encarga de los comentarios y likes de las publicaciones.

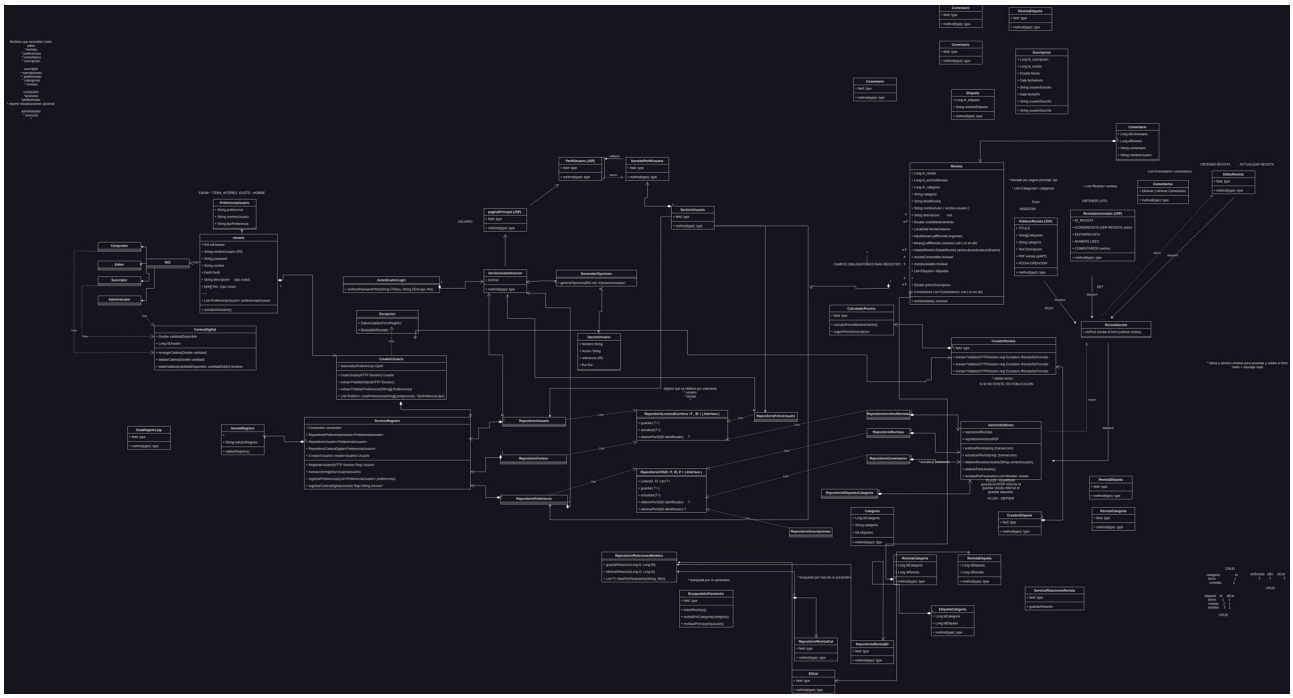
## SECCION 3: Estructura del Sistema

El diagrama de clases se puede dividir en varias capas:

- **Capa de presentación (JSP, HTML, JavaScript, AJAX):** La interfaz de usuario y las interacciones dinámicas se gestionan desde esta capa.
- **Capa de negocio (Clases Java):** Aquí es donde se implementa la lógica de negocio, incluyendo clases como **Revista**, **Usuario**, **Anuncio** y **Comentario**.
- **Capa de datos (JDBC, MySQL):** Se maneja el acceso a la base de datos y la persistencia de la información.

## Seccion 4: Diagrama de Clases

**A continuación se incluye el diagrama de clases que muestra la estructura y las relaciones entre las principales clases de la aplicación:**



## Seccion 5. Repositorios en la Aplicación

### 5,1 Repositorios CRUD

En aplicaciones Java, especialmente las que están construidas con **Jakarta EE**, el patrón **Repository** es utilizado para gestionar la persistencia de datos. Los repositorios actúan como intermediarios entre la aplicación y la base de datos, encapsulando toda la lógica de acceso y gestión de los datos.

En lugar de escribir consultas SQL directamente en la lógica de negocio, los repositorios permiten que las operaciones de **CRUD** (Crear, Leer, Actualizar y Eliminar) se realicen de una manera más abstracta y reutilizable. Los repositorios ayudan a mantener el código más limpio, modular y fácil de mantener.

### 5.2 Funciones Clave de los Repositorios

Un repositorio suele tener métodos específicos que permiten interactuar con la base de datos sin exponer los detalles de la implementación. Algunas de las funciones más comunes que implementan los repositorios incluyen:

- **Guardar**: Inserta un nuevo registro en la base de datos.
- **Actualizar**: Modifica un registro existente en la base de datos.
- **Eliminar**: Elimina un registro de la base de datos.
- **Buscar**: Recupera datos de la base de datos, ya sea por clave primaria o mediante consultas más complejas (filtrado por parámetros como el nombre del usuario, rol, etc.).

### 5.3 Cómo Interactúan los Repositorios con los Servicios

En una arquitectura típica de capas, los **servicios** llaman a los repositorios para realizar las operaciones de acceso a la base de datos. Los controladores o "controllers" de la aplicación interactúan con los servicios, que a su vez delegan las operaciones de persistencia en los repositorios.

Por ejemplo:

- Un **controlador** que gestiona la creación de usuarios puede llamar al `UsuarioServicio`, que a su vez llamará al `UsuarioRepositorioCrud` para guardar un nuevo usuario en la base de datos.
- En el caso de las **revistas**, el `RevistaServicio` invocará al `RevistaRepositorio` para realizar operaciones de búsqueda, filtrado y actualización de las publicaciones.