SITIO WEB PARA LA PUBLICACION DE REVISTAS DIGITALES

MANUAL TECNINCO

Presentación

El siguiente manual se ha desarrollado con la finalidad de dar a conocer la información necesaria para realizar mantenimiento, instalación y exploración del software para un Sito Web dedicado a la Publicación de Revistas Digitales en Formato PDF, el cual consta de diferentes funcionalidades para poder tener una experiencia de usuario adecuada. El manual ofrece la información necesaria de ¿cómo está realizado el software? para que la persona (Desarrollador en JAVA) que quiera editar el software lo haga de una manera apropiada, dando a conocer la estructura del desarrollo del aplicativo.

Resumen

El manual detalla los aspectos técnicos e informáticos del software para el Sito Web dedicado a la Publicación de Revistas Digitales en Formato PDF con la finalidad de explicar la estructura del aplicativo al personal que quiera administrarlo, editarlo o configurarlo. La siguiente guía se encuentra dividida en las herramientas que se usaron para la creación del software con una breve explicación paso a paso. El aplicativo maneja diferentes funcionalidades los cuales se explicará que funcionamiento realiza cada uno de ellos, dando sugerencias para el debido uso del sistema de información.

Introducción

El manual se realiza con el fin de detallar el software para el Sitio Web dedicado a la Publicación de Revistas Digitales en Formato PDF en términos técnicos para que la persona que vaya a administrar, editar o configurar el aplicativo lo haga de una manera apropiada. El documento se encuentra dividido en las siguientes secciones:

- ASPECTOS TEORICOS: Se darán a conocer conceptos, definiciones y explicaciones de los componentes del aplicativo desde un punto de vista teórico para mayor entendimiento por parte del lector sobre el funcionamiento del sistema de información y herramientas.
- DIAGRAMAS DE MODELAMIENTO: Se compone por diagramas e ilustraciones alusivos al funcionamiento del aplicativo.

ASPECTOS TECNICOS

El aplicativo del Sitio Web dedicado a la Publicación de Revistas Digitales en Formato PDF tiene la finalidad de ser un Blog de Revistas en donde hay usuarios que se encargaran de Publicar las Revistas así como usuarios que podrán suscribirse a ellas y poder leerlas. Se recomienda que el siguiente manual sea manipulado únicamente por la persona que quiera administrar, editar o configurar el software para el Sitio Web dedicado a la Publicación de Revistas Digitales en Formato PDF para velar por la seguridad de los datos que se almacenan.

1. Herramientas utilizadas para el desarrollo: En ésta sección se procede a explicar las herramientas informáticas empleadas para el desarrollo del aplicativo:
   1.1. Apache NetBeans: NetBeans es un entorno de desarrollo integrado (IDE) para Java. NetBeans permite desarrollar aplicaciones a partir de un conjunto de componentes de software modulares llamados módulos. NetBeans se ejecuta en Windows, macOS, Linux y Solaris. Además del desarrollo en Java, cuenta con extensiones para otros lenguajes como PHP, C, C++, HTML5 y JavaScript. Las aplicaciones basadas en NetBeans, incluido NetBeans IDE, pueden ser ampliadas por desarrolladores externos. NetBeans IDE es un entorno de desarrollo integrado de código abierto. NetBeans IDE admite el desarrollo de todos los tipos de aplicaciones Java (Java SE (incluido JavaFX), Java ME, web, EJB y aplicaciones móviles) listas para usar. Entre otras características se encuentran un sistema de proyectos basado en Ant, soporte Maven, refactorizaciones, control de versiones (compatible con CVS, Subversion, Git, Mercurial y Clearcase). La plataforma Apache NetBeans es un marco genérico para aplicaciones Swing. Proporciona la "plomería" que, antes, cada desarrollador tenía que escribir por sí mismo: guardar el estado, conectar acciones a elementos del menú, elementos de la barra de herramientas y atajos de teclado; gestión de ventanas, etc.
   1.2. MySQL Workbench: Workbench es una herramienta gráfica para trabajar con servidores y bases de datos MySQL. MySQL Workbench es totalmente compatible con MySQL Server 8.0. Las versiones obsoletas de MySQL Server pueden ser incompatibles con MySQL Workbench y deben actualizarse antes de intentar realizar una conexión. Las versiones posteriores a la 8.0, como la 8.4, pueden conectarse, pero es posible que algunas funciones no sean compatibles. La funcionalidad de MySQL Workbench cubre cinco temas principales:

1.2.1. Desarrollo de SQL: le permite crear y administrar conexiones a servidores de bases de datos. Además de permitirle configurar parámetros de conexión, MySQL Workbench brinda la capacidad de ejecutar consultas SQL en las conexiones de bases de datos mediante el Editor SQL integrado.

1.2.2. Modelado de Datos (diseño): le permite crear modelos de su esquema de base de datos de forma gráfica, realizar ingeniería inversa y directa entre un esquema y una base de datos activa, y editar todos los aspectos de su base de datos mediante el completo Editor de tablas. El Editor de tablas proporciona funciones fáciles de usar para editar tablas, columnas, índices, activadores, particiones, opciones, inserciones y privilegios, rutinas y vistas.

1.2.3. Administración de Servidor: le permite administrar instancias del servidor MySQL administrando usuarios, realizando copias de seguridad y recuperaciones, inspeccionando datos de auditoría, visualizando el estado de la base de datos y monitoreando el rendimiento del servidor MySQL.

1.2.4. Migración de Datos: permite migrar desde Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL y otras tablas, objetos y datos RDBMS a MySQL. La migración también permite migrar desde versiones anteriores de MySQL a las versiones más recientes.

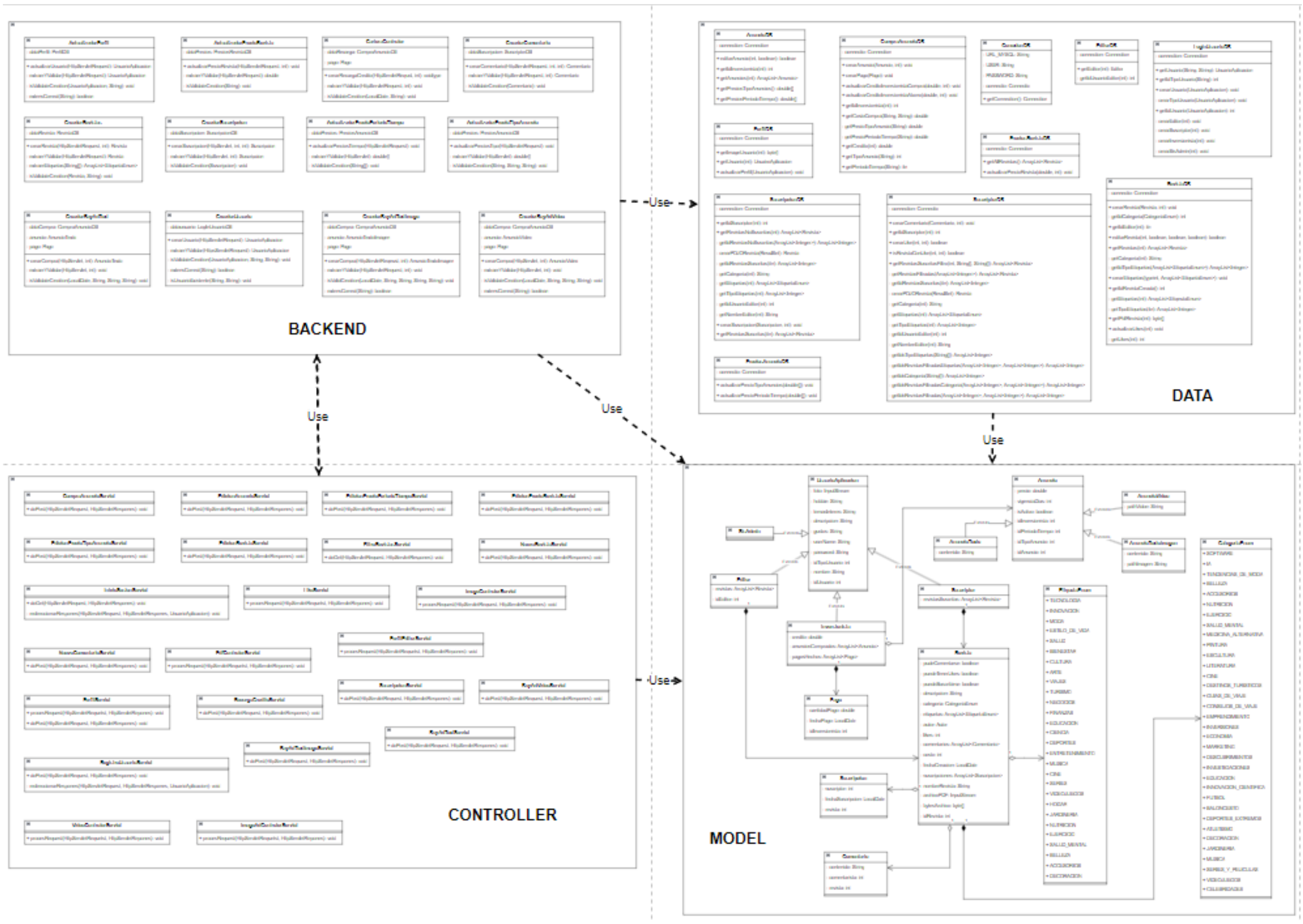
1.2.5. Soporte para MySQL Enterprise: Soporte para productos empresariales como MySQL Enterprise Backup, MySQL Firewall y MySQLAudit.

1.3. Apache Tomcat:

DIAGRAMAS DE MODELAMIENTO

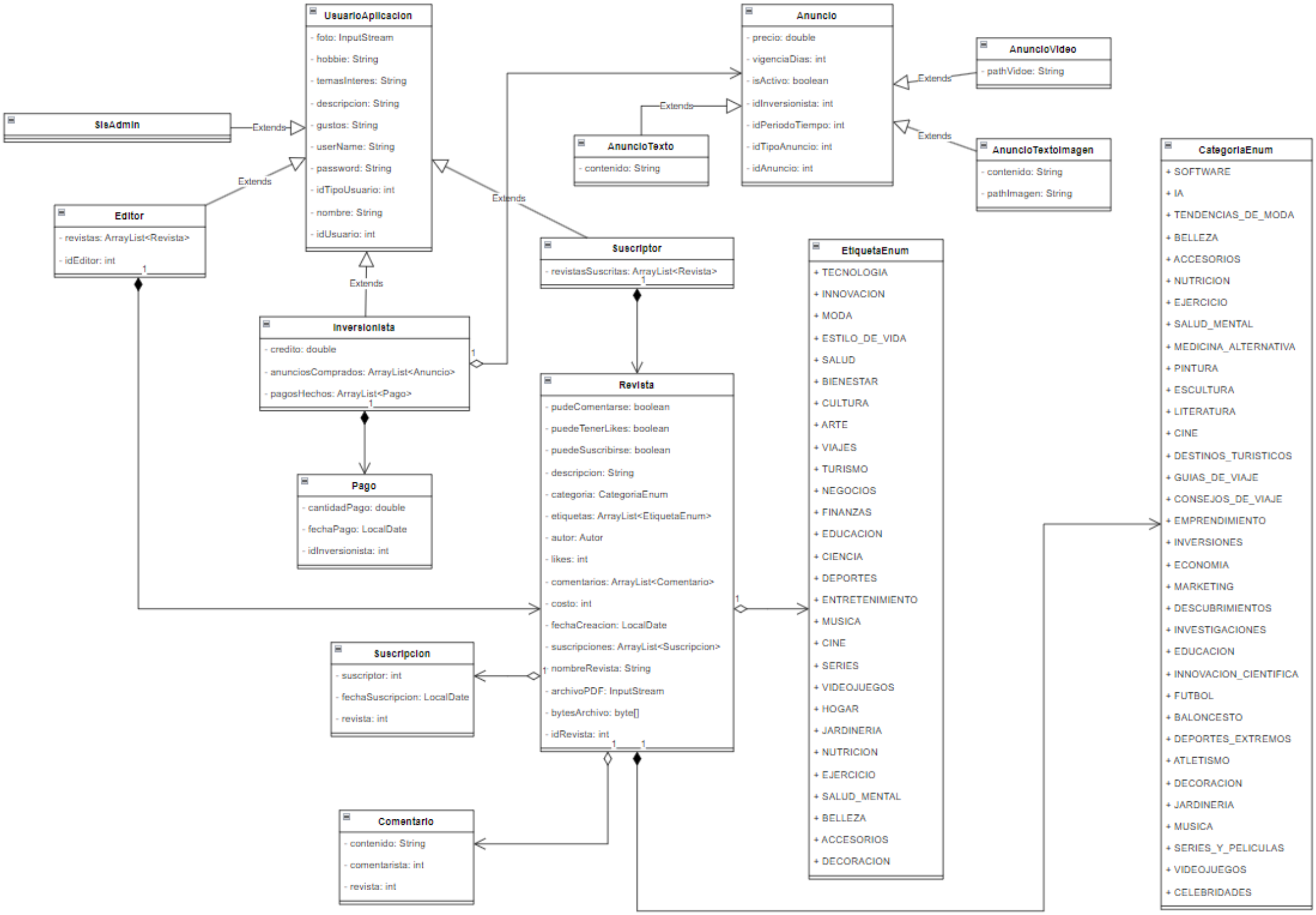
1. DIAGRAMA DE CLASES

El diagrama de clases está compuesto de los diferentes tipos de objetos, es decir, objetos de entidad, objetos de control y objetos de vista, métodos y atributos que se crearon y abstrajeron para el funcionamiento del software.



1.1. MODEL

Los Objetos que forman parte del MODEL son aquellos que en su mayoría son Entidades del tipo POJO para poder modelar los datos de la Base de Datos o para modelar los datos que son requeridos por el mismo sistema, además hay clases/entidades del tipo ENUMERADOS



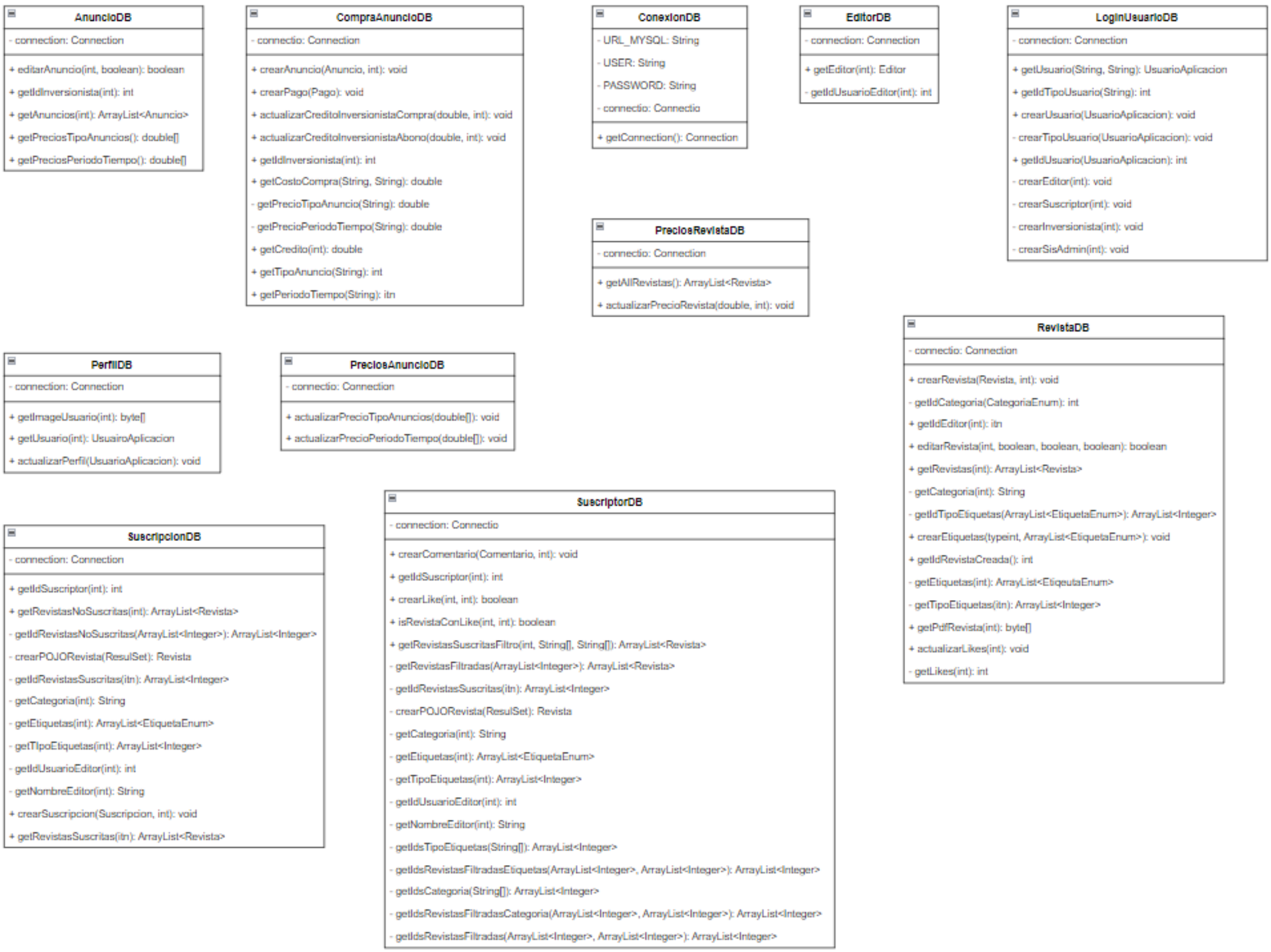
1.2. CONTROLLER

Los objetos que forman parte del CONTROLLER son todos los SERVLETS que se usaron para poder recibir las Peticiones del Cliente como también para poder Responder al Cliente con lo que corresponde

1.3. BACKEND

1.3.1. DATA OBJECTS

En la siguiente imagen se detalla cada una de las entidades/clases usadas para el funcionamiento lógico del aplicativo, en el cual cada una de ellas realiza las funciones/métodos posteriormente descritas.



- **AnuncioDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para un anuncio.
    - editarAnuncio: Método que hace un UPDATE a la Base de Datos para poder actualizar el estado de un determinado anuncio.
    - getIdInversionista: Método que hace un SELECT a la Base de Datos para poder obtener el ID de un determinado usuario que tiene el rol de Inversionista.
    - getAnuncios: Método que hace un SELECT a la Base de Datos para poder obtener todas los Anuncios de un Determinado Inversionista
    - getPreciosTipoAnuncios: Método que hace un SELECT a la Base de Datos para poder obtener el precio de los diferentes Tipos de Anuncios que se encuentran registrados.
    - getPrecioPeriodoTiempo: Método que hace un SELECT a la Base de Datos para poder obtener el precio de los diferentes Periodos de Tiempo que se encuentran registrados.
- **CompraAnuncioDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para la compra de un anuncio.
    - crearAnuncio: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo anuncio comprado.
    - crearPago: Método que hace un INSERT INTO a la Base de Datos para poder Registrar un nuevo Pago realizado por un Inversionista
    - actualizarCreditoInversionistaCompra: Método que se encarga de hacer un UPDATE a la Base de Datos para poder actualizar el crédito de un determinado Inversionista
    - actualizarCreditoInversionistaPago: Método que se encarga de hacer un UPDATE a la Base de Datos para poder actualizar el crédito de un determinado Inversionista
    - getIdInversionista: Método que hace un SELECT a la Base de Datos para poder obtener el ID de un determinado usuario que tiene el rol de Inversionista.
    - getCostoCompra: Método que se encarga de calcular el costo de compra para un determinado tipo de anuncio.
    - getPrecioTipoAnuncio: Método que hace un SELECT a la Base de Datos para poder obtener el precio de los diferentes Tipos de Anuncios que se encuentran registrados.
    - getPrecioPeriodoTiempo: Método que hace un SELECT a la Base de Datos para poder obtener el precio de los diferentes Periodos de Tiempo que se encuentran registrados.
    - getCredito: Método que hace un SELECT a la Base de Datos para poder obtener el crédito actual que tiene un cierto Inversionista
    - getTipoAnuncio: Método que hace un SELECT a la Base de Datos para poder obtener el ID de un determinado tipo de anuncio.
    - getPeriodoTiempo: Método que hace un SELECT a la Base de Datos para poder obtener el ID de un determinado periodo de tiempo.
- **ConexionDB:** Entidad que se encarga de conectarse con la Base de Datos MYSQL mediante la USRL, USER y PASSWORD.
- **EditorDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para un editor.
    - getEditor: Meto do que hace un SELECT a la Base de Datos para poder obtener un Editor registrado.
    - getIdUsuarioEditor: Método que hace un SELECT a la Base de Datos para poder obtener el ID de usuario de un determinado Editor.
- **LogInUsuarioDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen el inicio o registro de usuario.
    - getUsuario: Método que hace un SELECT a la Base de Datos para obtener un Usuario con determinado Username y Password
    - getIdTipoUsuario: Método que hace un SELECT a la Base de Datos para obtener el ID de tipo de Usuario para un determinado tipo de usuario
    - crearUsuario: Método que hace un INSERT INTO para poder registrar un nuevo Usuario

- crearTipoUsuario: Meto do que hace un INSERT INTO a la Base de Datos para poder registrar un Nuevo tipo de usuario.
- getIdUsuario: Meto do que hace un SELECT a la Base de Datos para poder obtener el ID de un determinado Usuario.
- crearEditor: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo Editor.
- crearSuscriptor: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo Suscriptor.
- crearInversionista Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo Inversionista. crearSisAdmin: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo Administrador del Sistema.

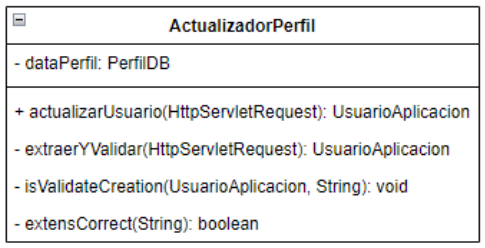
- **PerfilDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para un perfil de cierto usuario.
  - getImageUsuario: Método que se hace un SELECT a la Base de Datos para poder obtener los bytes del imagen que identifica a un cierto usuario
  - getUsuario: Método que hace un SELECT a la Base de Datos para poder obtener un Usuario en particular
  - actualizarPerfil: Método que hace un UPDATE a la Base de Datos para poder actualizar los datos de un usuario
- **PreciosRevistaDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para el conocimiento del precio de la revistas.
  - getAllRevistas: Método que hace un SELECT a la Base d Datos para obtener todas las Revistas registradas en el sistema
  - actualizarPrecioRevista: Método que hace un UPDATE a la Base de Datos para poder actualizar el precio de una determinada revista
- **SuscripcionDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para una suscripción a alguna revista.
  - getIdSuscriptor: Método que hace un SELECT a la Base de Datos para poder obtener el ID de suscriptor de cierto usuario
  - getRevistasNoSuscritas: Método que hace un SELECT a la Base de Datos para poder obtener las revistas a las que cierto usuario no esta suscrito
  - getIdRevistasNoSuscritas: Método que hace un SELECT a la Base de Datos para poder obtener los IDS de las revistas a las que cierto usuario no está suscrito
  - crearPOJORevista: Método que crear un objeto del tipo Revistas.
  - getIdRevistasSuscritas: Método que hace un SELECT a la Base de Datos para poder obtener las revistas a las que cierto usuario si está suscrito
  - getCategoria: Meto do que hace un SELECT a la Base de Datos para poder obtener el nombre de cierta categoría
  - getEtiquetas: Método que hace un SELECT a la Base de Datos para poder obtén las etiquetas de cierta revista
  - getTipoEtiquetas: Método que hace un SELECT a la Base de Datos para poder obtén las IDS de las etiquetas de cierta revista
  - getIdUsuarioEditor: Método que hace un SELECT a la Base de Datos para poder obtener el ID de usuario de cierto Editor
  - getNombreEditor: Método que hace un SELECT a la Base de Datos para poder obtener el nombre de cierto Editor
  - crearSuscripcion: Método que hace un INSERT INTO a la Base de Datos para poder registrar una nueva suscripción
  - getRevistasSuscritas: Método que hace un SELECT a la Base de Datos para poder obtener las revista a las que cierto suscriptor esta suscrito.
- **SuscriptorDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para un suscriptor.
  - crearComentario: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo Comentario
  - getIdSuscriptor: Método que hace un SELECT a la Base de Datos para poder obtener el ID de suscriptor de cierto usuario
  - crearLike: Método que hace un INSERT INTO a la Base de Datos para poder registrar un nuevo Like
  - isRevistaConLike: Método que hace un SELECT a la Base de Datos para poder saber si una determinada revista ya recibió el like de un determinado suscriptor
  - getRevistasSuscritas: Método que hace un SELECT a la Base de Datos para poder obtener la revistas de un determinado suscriptor

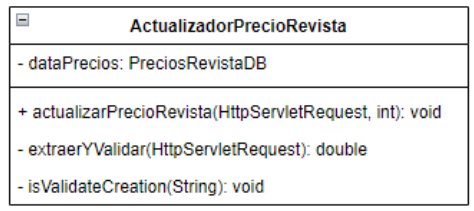
- getRevistasFiltradas: Método que hace un SELECT a la Base de Datos para poder obtener las revistas que cumplan con los filtro de etiquetas y categorías seleccionadas por el usuario suscriptor
- crearPOJORevista: Método que crear un objeto del tipo Revistas.
- getCategoria: Meto do que hace un SELECT a la Base de Datos para poder obtener el nombre de cierta categoría
- getEtiquetas: Método que hace un SELECT a la Base de Datos para poder obtén las etiquetas de cierta revista
- getTipoEtiquetas: Método que hace un SELECT a la Base de Datos para poder obtén las IDS de las etiquetas de cierta revista
- getIdUsuairioEditor: Método que hace un SELECT a la Base de Datos para poder obtener el ID de usuario de cierto Editor
- getNombreEditor: Método que hace un SELECT a la Base de Datos para poder obtener el nombre de cierto Editor
- getIdsRevistasFiltradasEtiquetas: Método que hace un SELECT a la Base de Datos para poder obtener el ID de las revistas que cumplieron con el filtro de la etiquetas seleccionadas por el suscriptor.
- getIdsCategorias: Método que hace un SELECT a la Base de Datos para obtener los IDS de las categorías seleccionadas para filtrar por el suscriptor
- getIdsRevistasFiltradasCategoria: Método que hace un SELECT a la Base de Datos para poder obtener el ID de las revistas que cumplieron con el filtro de las categorías seleccionadas por el suscriptor.
- getIdsRevistasFiltradas: Método que hace un SELECT a la Base de Datos para obtener los IDS de las revistas que cumplieron con el filtro de las etiquetas y de las categorías seleccionadas por el suscriptor

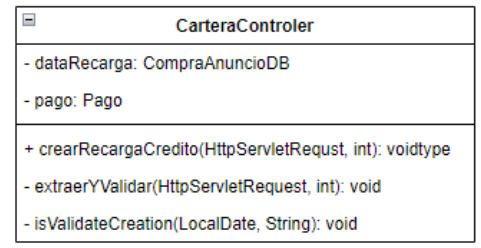
- **RevistaDB:** Entidad que se encarga de conectarse con la Base de Datos para poder hacer las consultas y actualizaciones necesarias en base a los requerimientos que se tienen para una revista.
  - crearRevista: Método que hace un INSERT INTO a la Base de Datos para poder registrar una nueva revista
  - getIdCategoria: Método que hace un SLECT a la Base de Datos para poder obtener el ID de una categoría en particular
  - getIdEditor: Método que hace un SLECT a la Base de Datos para poder obtener el ID de editor de un determinado usuario
  - editarRevistas: Método que hace un UPDATE en la Base de Datos para poder actualizar los estado de una determinada revista
  - getRevistas: Método que hace un SLECT a la Base de Datos para poder obtener las revistas de un determinad editor
  - getCategoria: Método que hace un SLECT a la Base de Datos para poder obtener el nombre de la categoría de una determinad categoría
  - getIdTipoEtiqueta: Método que hace un SLECT a la Base de Datos para poder obtener los IDS de determinadas etiquetas
  - crearEtiquetas: Metodo que hace un INSERT INTO a la Base de Datos para poder registrar una nueva etiqueta
  - getIdRevistaCreada: Método que hace un SLECT a la Base de Datos para poder obtener el ID de la última revista registrada en el sistema
  - getEtiquetas: Método que hace un SLECT a la Base de Datos para poder obtener las etiquetas que tiene una determinada revista
  - getTipoEtiquetas: Método que hace un SLECT a la Base de Datos para poder obtener los IDS de las etiquetas de una determinada revista
  - getPdfRevista: Método que hace un SLECT a la Base de Datos para poder obtener los bytes del PDF de una determinada revista
  - actualizarLikes: Metodo que hace un UPDATE a la Base de Datos para poder Actualizar los likes de una determinada revista
  - getLikes: Método que hace un SLECT a la Base de Datos para poder obtener los likes que ha tenido una determinada revista

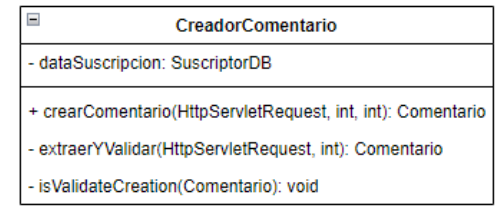
1.3.2. ACTION OBJECTS

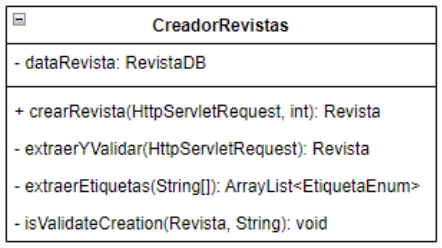
En la siguiente imagen se detalla cada una de las entidades (clases) usada para el funcionamiento lógico del aplicativo, en el cual cada una de ellas realiza acciones posteriormente descritas

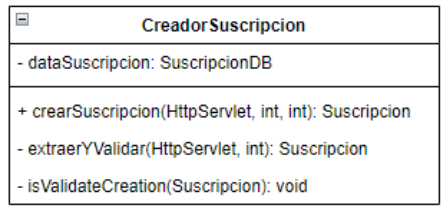


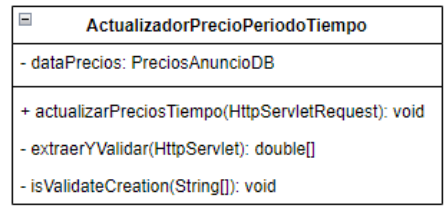


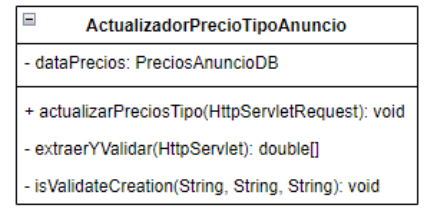


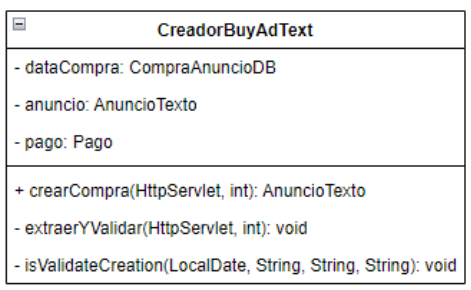


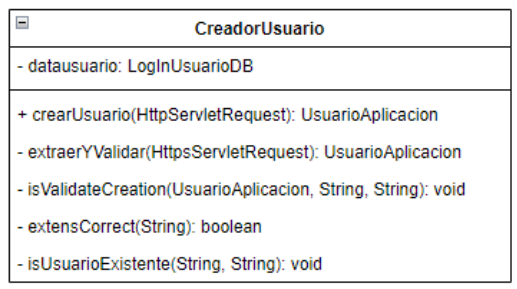


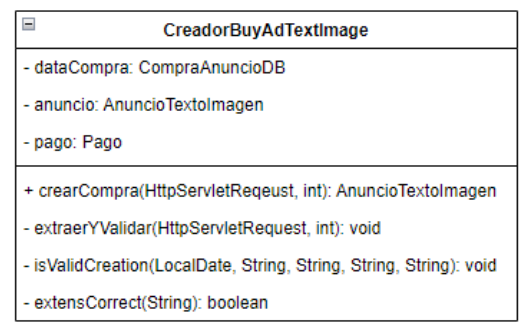


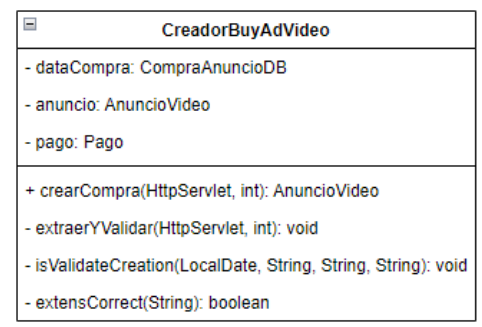












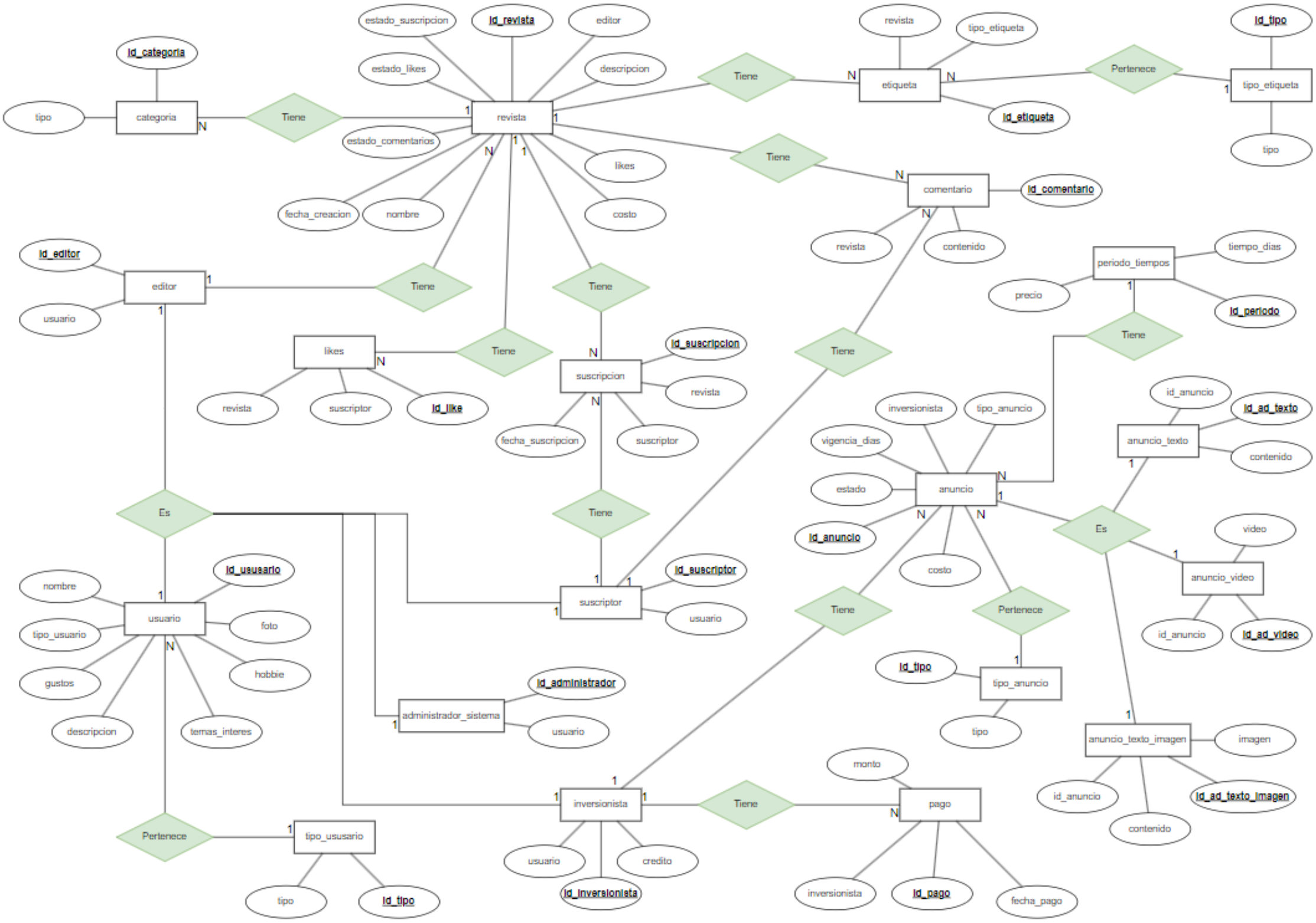
- **ActualizadorPerfil:**
  - actualizarUsuario: Método que se comunica con las clases de Data y actualizar los datos del usuario siempre que los datos ingresados y obtenidos sean correctos
  - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
  - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
  - extensCorrect: Método que se encarga de evaluar si la extensión del archivo recibido es la adecuada para que se pueda guardar en Base de Datos
- **ActualizadorPrecioRevista:**
  - actualizarPrecioRevista: Método que se comunica con las clases de Data y actualizar el precio de una revista siempre que los datos ingresados y obtenidos sean correctos
  - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
  - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **CarteraControler:**
  - crearRecargaCredito: Método que se comunica con las clases de Data y crea la recarga de crédito siempre que los datos ingresado y obtenidos sean correctos

    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
    - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **CreadorComentario:**
    - crearComentario: Método que se comunica con las clases de Data y crea un comentario siempre que los datos ingresado y obtenidos sean correctos
    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
    - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **CreadorRevista:**
    - crearRevista: Método que se comunica con las clases de Data y crea una revista siempre que los datos ingresado y obtenidos sean correctos
    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
    - extraerEtiquetas: Método que se encarga de saber si el usuario selecciono o no etiquetas y así poder seguir validando los siguientes datos de lo contrario se manda un error ya que se deben de seleccionar etiquetas de forma obligatoria para una revista
    - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **CreadorSuscripcion:**
    - crearSuscripcion: Método que se comunica con las clases de Data y crea una suscripción siempre que los datos ingresado y obtenidos sean correctos
    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
    - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **ActualizadorPrecioPeriodoTiempo:**
    - actualizarPrecioTiempo: Método que se comunica con las clases de Data y actualizar el precio de los periodos de tiempo siempre que los datos ingresados y obtenidos sean correctos
    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
    - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **Actualizador PrecioTipoAnuncio:**
    - actualizarPrecioTIpo: Método que se comunica con las clases de Data y actualizar el precio de los diferentes tipos de anuncios siempre que los datos ingresados y obtenidos sean correctos
    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
    - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
- **CreadorBuyAdText:**
    - crearCompra: Método que se comunica con las clases de Data y crea la compra siempre que los datos ingresado y obtenidos sean correctos
    - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos

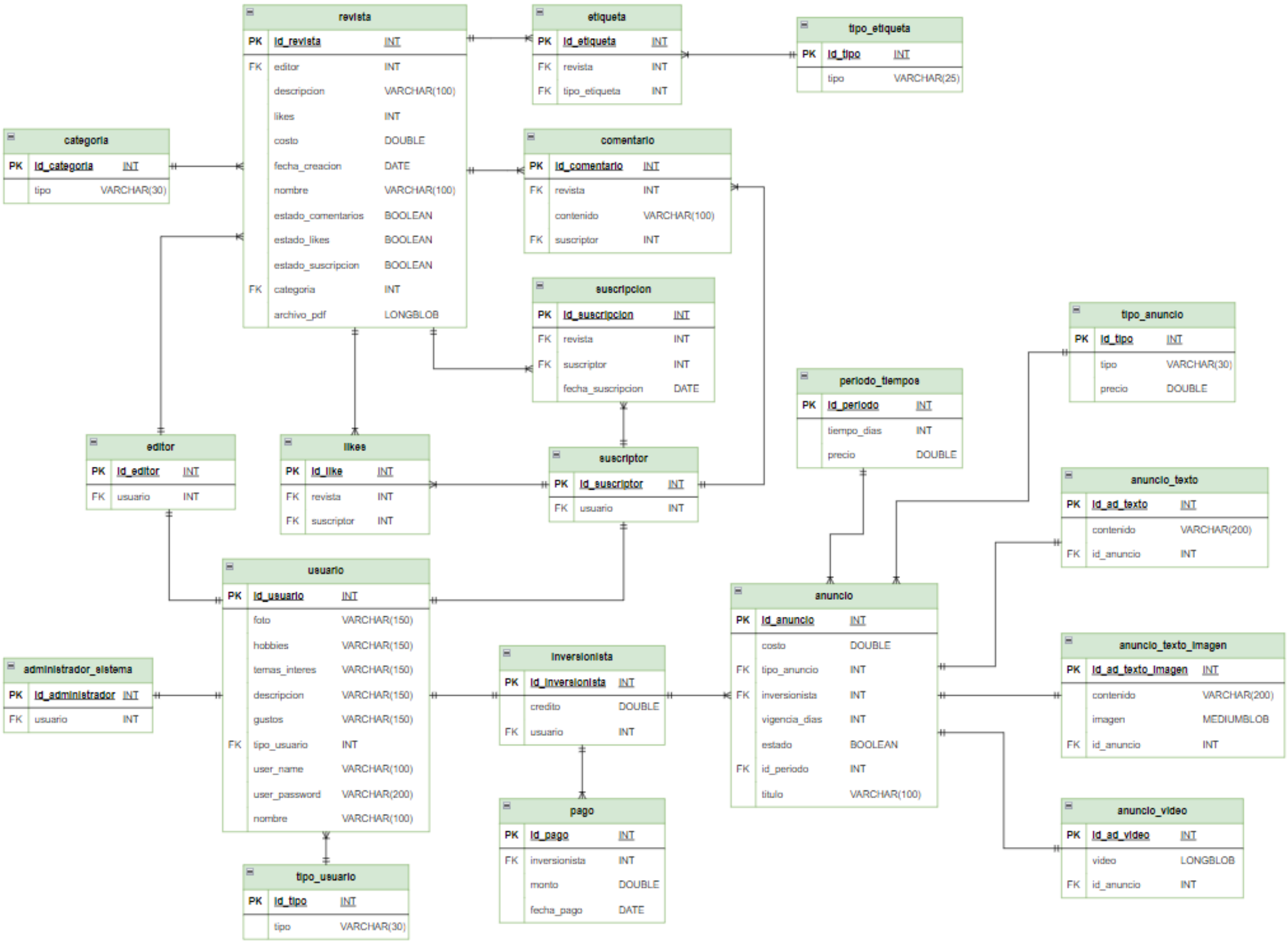
- isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente

- **CreadorUsuario:**
  - crearUsuario: Método que se comunica con las clases de Data y crea el usuario siempre que los datos ingresado y obtenidos sean correctos
  - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
  - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
  - extensCorrect: Método que se encarga de evaluar si la extensión del archivo recibido es la adecuada para que se pueda guardar en Base de Datos
  - isUsuarioExistente: Método que se encarga de saber si los datos de Username y Passwor ya pertenece a algún usuario registrado
- **CreadorBuyAdTextImage:**
  - crearCompra: Método que se comunica con las clases de Data y crea la compra siempre que los datos ingresado y obtenidos sean correctos
  - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
  - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
  - extensCorrect: Método que se encarga de evaluar si la extensión del archivo recibido es la adecuada para que se pueda guardar en Base de Datos
- **CreadorBuyAdVideo:**
  - crearCompra: Método que se comunica con las clases de Data y crea la compra siempre que los datos ingresado y obtenidos sean correctos
  - extraerYValidar: Método que extrae los datos que se han enviado por el usuario para su posterior evaluación para saber si son o no validos
  - isValidateCreation: Método que se encarga de validar que los datos recibidos son válidos y aceptados para que se puedan guardar o ejecutar posteriormente
  - extensCorrect: Método que se encarga de evaluar si la extensión del archivo recibido es la adecuada para que se pueda guardar en Base de Datos

2. DIAGRAMA ENTIDAD RELACION E/R



3. DIAGRAMA DE TABLAS



4. MAPEO FISICO DE LA BASE DE DATOS

```
CREATE SCHEMA blog_revistas_bd;

USE blog_revistas_bd;

CREATE TABLE categoria (
        id_categoria INT NOT NULL AUTO_INCREMENT,
        tipo VARCHAR(30) NOT NULL,
        CONSTRAINT PK_TIPO_CATEGORIA PRIMARY KEY(id_categoria)
);

INSERT INTO categoria (tipo) VALUES ('SOFTWARE'), ('IA'), ('TENDENCIAS_DE_MODA'), ('BELLEZA'),
('ACCESORIOS'), ('NUTRUCION'), ('EJERCICIO'), ('SALUD_MENTAL'), ('MEDICINA_ALTERNATIVA'),
('PINTURA'), ('ESCULTURA'), ('LITERATURA'), ('CINE'), ('DESTINOS_TURISTICOS'),
('GUIAS_DE_VIAJE'), ('CONSEJOS_DE_VIAJE'), ('EMPRENDIMIENTO'), ('INVERSIONES'),
('ECONOMIA'), ('MARKETING'), ('DESCUBRIMIENTOS'), ('INVESTIGACIONES'), ('EDUCACION'),
('INNOVACION_CIENTIFICA'), ('FUTBOL'), ('BALONCESTO'), ('DEPORTES_EXTREMOS'),
('ATLETISMO'), ('DECORACION'), ('JARDINERIA'), ('MUSICA'), ('SERIES_Y_PELICULAS'),
('VIDEOJUEGOS'), ('CELEBRIDADES');

CREATE TABLE tipo_etiqueta (
        id_tipo INT NOT NULL AUTO_INCREMENT,
        tipo VARCHAR(25) NOT NULL,
        CONSTRAINT PK_TIPO_ETIQUETA PRIMARY KEY(id_tipo)
);

INSERT INTO tipo_etiqueta (tipo) VALUES ('TECNOLOGIA'), ('INNOVACION'), ('MODA'),
```

```
('ESTILO_DE_VIDA'), ('SALUD'), ('BIENESTAR'), ('CULTURA'), ('ARTE'), ('VIAJES'), ('TURISMO'),
('NEGOCIOS'), ('FINANZAS'), ('EDUCACION'), ('CIENCIA'), ('DEPORTES'), ('ENTRETENIMIENTO'),
('MUSICA'), ('CINE'), ('SERIES'), ('VIDEOJUEGOS'), ('HOGAR'), ('JARDINERIA'), ('NUTRICION'),
('EJERCICIO'), ('SALUD_MENTAL'), ('BELLEZA'), ('ACCESORIOS'), ('DECORACION');

CREATE TABLE tipo_anuncio (
        id_tipo INT NOT NULL AUTO_INCREMENT,
        tipo VARCHAR(30) NOT NULL,
        precio DOUBLE NOT NULL,
        CONSTRAINT PK_TIPO_ANUNCIO PRIMARY KEY(id_tipo)
);

INSERT INTO tipo_anuncio (tipo, precio) VALUES ('ANUNCIO_TEXTO', 3),
('ANUNCIO_TEXTO_IMAGEN', 6), ('ANUNCIO_VIDEO', 9);

CREATE TABLE periodo_tiempos (
        id_periodo INT NOT NULL AUTO_INCREMENT,
        tiempo_dias INT NOT NULL,
        precio DOUBLE NOT NULL,
        CONSTRAINT PK_PERIODO_TIEMPO PRIMARY KEY(id_periodo)
);

INSERT INTO periodo_tiempos (tiempo_dias, precio) VALUES (1, 1), (3, 3), (7, 7), (14, 14);

CREATE TABLE tipo_usuario (
        id_tipo INT NOT NULL AUTO_INCREMENT,
        tipo VARCHAR(30) NOT NULL,
        CONSTRAINT PK_TIPO_USUARIO PRIMARY KEY(id_tipo)
);
```

```
INSERT INTO tipo_usuario (tipo) VALUES ('editor'), ('suscriptor'), ('inversionista'),
('administrador_sistema');
CREATE TABLE usuario (
        id_usuario INT NOT NULL AUTO_INCREMENT,
        foto MEDIUMBLOB NOT NULL,
        hobbie VARCHAR(150) NULL,
        temas_interes VARCHAR(150) NULL,
        descripcion VARCHAR(200) NULL,
        gustos VARCHAR(150) NULL,
        tipo_usuario INT NOT NULL,
        user_name VARCHAR(100) NOT NULL,
        user_password VARCHAR(300) NOT NULL,
        nombre VARCHAR(100) NOT NULL,
        CONSTRAINT PK_USUAIRO PRIMARY KEY(id_usuario),
        CONSTRAINT FK_USUARIO_IN_TIPO_USUARIO FOREIGN KEY(tipo_usuario) REFERENCES
        tipo_usuario(id_tipo)
);

CREATE TABLE administrador_sistema (
        id_administrador INT NOT NULL AUTO_INCREMENT,
        usuario INT NOT NULL,
        CONSTRAINT PK_ADMINISTRADOR PRIMARY KEY(id_administrador),
        CONSTRAINT FK_USUARIO_IN_USUAIRO FOREIGN KEY(usuario) REFERENCES usuario(id_usuario)
);

CREATE TABLE editor (
        id_editor INT NOT NULL AUTO_INCREMENT,
        usuario INT NOT NULL,
        CONSTRAINT PK_EDITOR PRIMARY KEY(id_editor),
```

```
    CONSTRAINT FK_USUARIO_IN_USUARIO_EDITOR FOREIGN KEY(usuario) REFERENCES
    usuario(id_usuario)
);

CREATE TABLE suscriptor (
    id_suscriptor INT NOT NULL AUTO_INCREMENT,
    usuario INT NOT NULL,
    CONSTRAINT PK_SUSCRIPTOR PRIMARY KEY(id_suscriptor),
    CONSTRAINT FK_USUARIO_IN_USUARIO_SUSCRIPTOR FOREIGN KEY(usuario) REFERENCES
    usuario(id_usuario)
);

CREATE TABLE inversionista (
    id_inversionista INT NOT NULL AUTO_INCREMENT,
    credito DOUBLE NOT NULL,
    usuario INT NOT NULL,
    CONSTRAINT PK_INVERSIONISTA PRIMARY KEY(id_inversionista),
    CONSTRAINT FK_USUARIO_IN_USUARIO_INVERSIONISTA FOREIGN KEY(usuario) REFERENCES
    usuario(id_usuario)
);

CREATE TABLE pago (
    id_pago INT NOT NULL AUTO_INCREMENT,
    inversionista INT NOT NULL,
    monto DOUBLE NOT NULL,
    fecha_pago DATE NOT NULL,
    CONSTRAINT PK_PAGO PRIMARY KEY(id_pago),
    CONSTRAINT FK_INVERSIONISTA_IN_INVERSIONISTA FOREIGN KEY(inversionista) REFERENCES
    inversionista(id_inversionista)
);
```

```
CREATE TABLE anuncio (
        id_anuncio INT NOT NULL AUTO_INCREMENT,
        costo DOUBLE NOT NULL,
        tipo_anuncio INT NOT NULL,
        inversionista INT NOT NULL,
        vigencia_dias INT NOT NULL,
        estado BOOLEAN NOT NULL,
        id_periodo INT NOT NULL,
        titulo VARCHAR(100) NOT NULL,
        CONSTRAINT PK_ANUNCIO PRIMARY KEY(id_anuncio),
        CONSTRAINT FK_TIPO_ANUNCIO_IN_TIPO_ANUNCIO FOREIGN KEY(tipo_anuncio) REFERENCES
        tipo_anuncio(id_tipo),
        CONSTRAINT FK_ANUNCIO_INVERSIONISTA_IN_INVERSIONISTA FOREIGN KEY(inversionista)
        REFERENCES inversionista(id_inversionista),
        CONSTRAINT FK_ID_PERIODO_IN_PERIODO_TIEMPOS FOREIGN KEY(id_periodo) REFERENCES
        periodo_tiempos(id_periodo)
);

CREATE TABLE revista (
        id_revista INT NOT NULL AUTO_INCREMENT,
        editor INT NOT NULL,
        descripcion VARCHAR(200) NOT NULL,
        likes INT NOT NULL,
        costo DOUBLE NOT NULL,
        fecha_creacion DATE NOT NULL,
        nombre VARCHAR(100) NOT NULL,
        estado_comentarios BOOLEAN NOT NULL,
        estado_likes BOOLEAN NOT NULL,
        estado_suscripcion BOOLEAN NOT NULL,
        categoria INT NOT NULL,
```

```
    archivo_pdf LONGBLOB NOT NULL,
    costo_global DOUBLE NOT NULL,
    CONSTRAINT PK_REVISTA PRIMARY KEY(id_revista),
    CONSTRAINT FK_EDITOR_IN_EDITOR FOREIGN KEY(editor) REFERENCES editor(id_editor),
    CONSTRAINT FK_CATEGORIA_INCATEGORIA FOREIGN KEY(categoria) REFERENCES
    categoria(id_categoria)
);

CREATE TABLE etiqueta (
    id_etiqueta INT NOT NULL AUTO_INCREMENT,
    revista INT NOT NULL,
    tipo_etiqueta INT NOT NULL,
    CONSTRAINT PK_ETIQUETA PRIMARY KEY(id_etiqueta),
    CONSTRAINT FK_REVISTA_IN_REVISTA FOREIGN KEY(revista) REFERENCES revista(id_revista),
    CONSTRAINT FK_TIPO_ETIQUETA_IN_TIPO_ETIQUETA FOREIGN KEY(tipo_etiqueta) REFERENCES
    tipo_etiqueta(id_tipo)
);

CREATE TABLE comentario (
    id_comentario INT NOT NULL AUTO_INCREMENT,
    revista INT NOT NULL,
    contenido VARCHAR(200) NOT NULL,
    suscriptor INT NOT NULL,
    CONSTRAINT PK_COMENTARIO PRIMARY KEY(id_comentario),
    CONSTRAINT FK_COMENTARIO_REVISTA_IN_REVISTA FOREIGN KEY(revista) REFERENCES
    revista(id_revista),
    CONSTRAINT FK_COMENTARIO_SUSCRIPTOR_IN_SUSCRIPTOR FOREIGN KEY(suscriptor)
    REFERENCES suscriptor(id_suscriptor)
);
```

```
CREATE TABLE suscripcion (
    id_suscripcion INT NOT NULL AUTO_INCREMENT,
    revista INT NOT NULL,
    suscriptor INT NOT NULL,
    fecha_suscripcion DATE NOT NULL,
    CONSTRAINT PK_SUSCRIPCION PRIMARY KEY(id_suscripcion),
    CONSTRAINT FK_SUSCRIPCION_REVISTA_IN_REVISTA FOREIGN KEY(revista) REFERENCES
    revista(id_revista),
    CONSTRAINT FK_SUSCRIPTOR_IN_SUSCRIPTOR FOREIGN KEY(suscriptor) REFERENCES
    suscriptor(id_suscriptor)
);

CREATE TABLE likes (
    id_like INT NOT NULL AUTO_INCREMENT,
    revista INT NOT NULL,
    suscriptor INT NOT NULL,
    CONSTRAINT PK_LIKES PRIMARY KEY(id_like),
    CONSTRAINT FK_LIKES_REVISTA_IN_REVISTA FOREIGN KEY(revista) REFERENCES
    revista(id_revista),
    CONSTRAINT FK_LIKES_SUSCRIPTOR_IN_SUSCRIPTOR FOREIGN KEY(suscriptor) REFERENCES
    suscriptor(id_suscriptor)
);

CREATE TABLE anuncio_texto (
    id_ad_texto INT NOT NULL AUTO_INCREMENT,
    contenido VARCHAR(200) NOT NULL,
    id_anuncio INT NOT NULL,
    CONSTRAINT PK_ANUNCIO_TEXTO PRIMARY KEY(id_ad_texto),
    CONSTRAINT FK_ID_ANUNCIO_IN_ANUNCIO FOREIGN KEY(id_anuncio) REFERENCES
    anuncio(id_anuncio)
);
```

```
CREATE TABLE anuncio_texto_imagen (
        id_ad_texto_imagen INT NOT NULL AUTO_INCREMENT,
        contenido VARCHAR(200) NOT NULL,
        imagen MEDIUMBLOB NOT NULL,
        id_anuncio INT NOT NULL,
        CONSTRAINT PK_ANUNCIO_TEXTO_IMAGEN PRIMARY KEY(id_ad_texto_imagen),
        CONSTRAINT FK_AD_TEXTO_IMAGEN_ID_ANUNCIO_IN_ANUNCIO FOREIGN KEY(id_anuncio)
        REFERENCES anuncio(id_anuncio)
);

CREATE TABLE anuncio_video (
        id_ad_video INT NOT NULL AUTO_INCREMENT,
        video LONGBLOB NOT NULL,
        id_anuncio INT NOT NULL,
        CONSTRAINT PK_ANUNCIO_TEXTO_IMAGEN PRIMARY KEY(id_ad_video),
        CONSTRAINT FK_AD_VIDEO_ID_ANUNCIO_IN_ANUNCIO FOREIGN KEY(id_anuncio) REFERENCES
        anuncio(id_anuncio)
);
```