Universidad San Carlos de Guatemala

Centro Universitario de Occidente

Teoría de Sistemas 1

Ing. Pedro Domingo

HOLAQH

(Manual Técnico)

Javier Isaías Coyoy Marín

202030556

Quetzaltenango 4 de noviembre de 2024

INDICE

MARCO TEORICO

La organización es muy importante en cualquier ámbito de nuestra vida y nos permite tener un control sobre las cosas que pasan a nuestro alrededor. Y en esta situación las herramientas digitales son de gran ayuda para facilitar estos procesos. Bajo esta premisa nosotros desarrollamos un sistema de gestión de eventos que se espera ayude a la población a estar informada sobre los eventos que se llevan a cabo en diferentes partes del país y de poder tener una organización de eventos que ayuden al usuario a organizar su tiempo y establecer horarios para poder asistir a las actividades.

El sistema que presentamos cuenta con roles los cuales desempeñan factores importantes para el funcionamiento total de nuestro sistema entre los cuales tenemos:

Administración: Este usuario se encarga del monitoreo de las publicaciones que estén dentro de la página procurando que todos los eventos cumplan con las restricciones de publicaciones establecidas por el sistema.

Anunciante: Esta la persona encarga de crear las publicaciones o eventos, su objetivo es la de mostrar información y esperar que su publicación logre llamar la atención de los clientes.

Cliente: Es la persona que tiene una cuenta y puede tomar la decisión de asistir o no según sea su interés, por otro lado, son ellos las personas que ayudan a la administración a poder llevar un control de las publicaciones tratando que ninguna publicación incumpla las normas y esto se logra a través de los reportes que realiza el cliente.

Visitante: Es la persona que puede visitar el sistema, pero no va poder interactuar con él a menos que se cree una cuenta.

Para la realización de esta práctica hicimos uso de las tecnologías de Xampp que es una herramienta que nos ayuda a crear un entorno local en nuestra computadora que nos permite probar aplicaciones web. Dentro los componentes que utilizamos están:

- Apache: Es el servidor web que gestiona las solicitudes de los navegadores y sirve páginas web.
- MySQL/MariaDB: Son sistemas de gestión de bases de datos que almacenan la información que las aplicaciones web pueden usar.
- PHP: Es un lenguaje de programación del lado del servidor que permite crear dinámicamente contenido web.

Y para la trabajar con los html, css y javascript utilizaremos Visual Studio Code como editor de código. Y con estos componentes tendríamos las herramientas necesarias para poder trabajar todos los puntos de nuestro proyecto y lograr así tener el mejor desempeño posible.

TECNOLOGÍAS UTILIZADAS

Para la práctica utilizamos las siguientes tecnologías Xampp (Versión 8.1.25) para Windows el proceso de instalación es solo ir dándole siguiente a cada una de las opciones que nos aparecen al ejecutar el programa la única ventana que nos interesa es en donde se nos pregunta acerca de los componentes que queremos seleccionar para la instalación es importante seleccionar Apache y MySQL el resto de opciones pueden no ser seleccionadas. Luego de esto y de terminar la instalación abrimos el programa y corremos Apache y MySQL.

Importante tomar en cuenta que al ejecutar estas acciones puede lanzar un error si por ejemplo el puerto que viene por defecto está ocupado, en ese caso se pueden tomar dos opciones se puede cambiar la dirección del puerto por medio del archivo my.init que se encuentra en el apartado de Config de MySQL también debemos agregar ese nuevo puerto al archivo phpMyAdmin que se encuentra en el apartado de Apache en la opción config, por ultimo podemos establecer este nuevo puerto en las configuraciones generales de nuestro servicio, la otra opción es detener lo que sea que esté utilizando ese puerto para que podamos correr MySQL con normalidad.

Para trabajar con PHP, HTML, CSS y Javascript utilizamos Visual Studio Code (Versión 1.92.2) para Windows, además para el manejo del lenguaje PHP se recomienda instalar la extensión (PHP Intelephense).

Para la base de datos utilizaremos MySQL para poder crear la base y las tablas podemos hacerlo desde phpAdmin esta opción la pueden encontrar al colocar en el navegador http://localhost. Es importante tomar en cuenta que tanto el apache o el motor de MySQL deben estar corriendo en sus respectivos puertos para poder utilizar el phpAdmin.

Por último, para la arquitectura del programa utilizamos el patrón MVC que nos brinda una forma eficiente y ordenada de trabajar haciendo que cada elemento tenga su lugar y su funcionamiento. También hicimos uso de servicios que nos ayudaran a obtener las solicitudes de los usuarios y por medio de un parámetro llamar al controlador correspondiente que llamara al modelo y realizara los procesos de INSERT, SELECT, UPDATE Y DELETE de los datos solicitados. Además de la configuración de la conexión a la base de datos que estará en una clase llamada conexión.

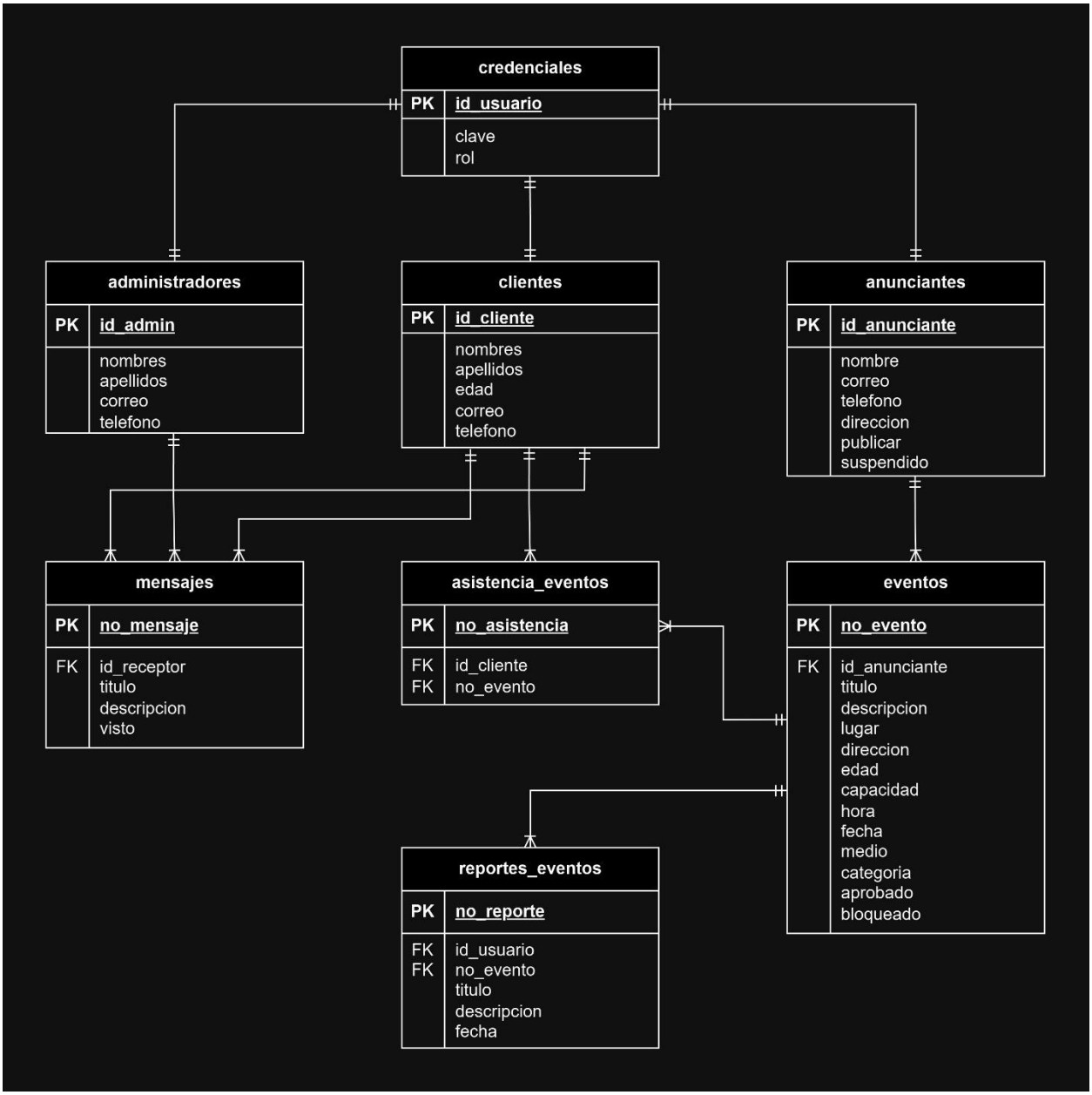
Y estas serían algunas recomendaciones para el uso de las tecnologías que se utilizaron para este proyecto.

DIAGRAMAS

A continuación, presentamos los diagramas de nuestro proyecto iniciando con una comparación entre la presentación de solución inicial y la final del proyecto para algunos diagramas.

DIAGRAMA BASE DE DATOS

(ANTERIOR)



(ACTUAL)

En la solución final se eliminó el campo de visto de la tabla de mensajes porque no se le miro la utilidad importante más haya de la presentación y se agregó un campo editar en la tabla eventos utilizada para habilitar la edición a eventos que no han sido aprobados por la administración y por último se agregó un campo de aprobados en la tabla de anunciantes para llevar un conteo de las publicaciones que tiene aprobadas y determinar cuando debía ser liberado la opción de publicación automática.

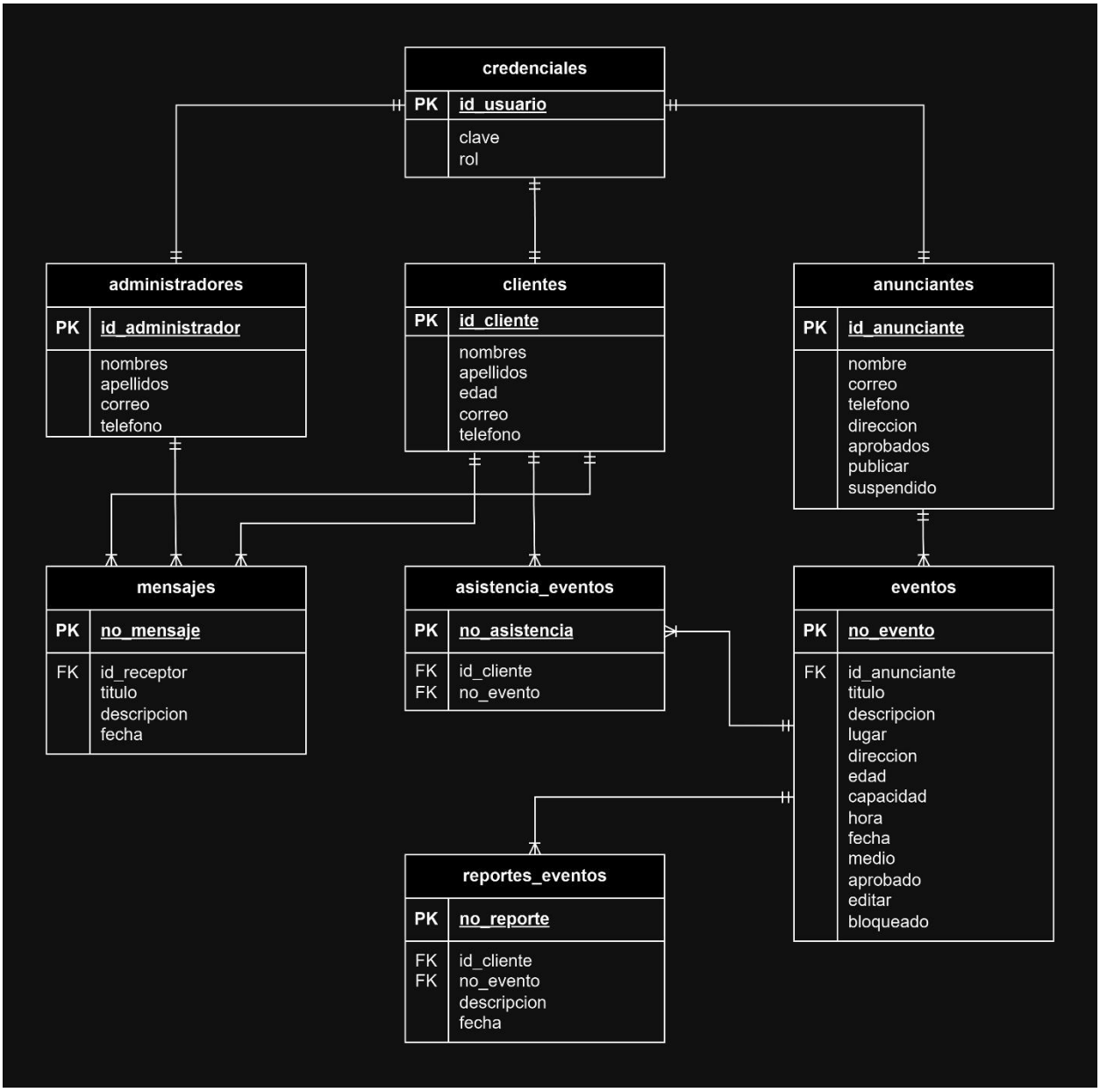
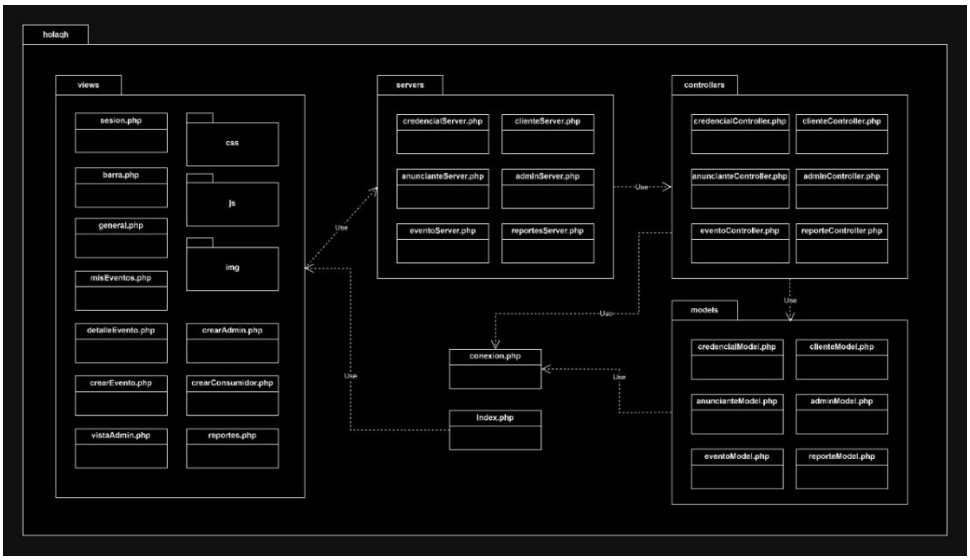


DIAGRAMA PAQUETES

(ANTERIOR)



(ACTUAL)

En la final se modificaron 2 vistas que pasaron de ser las de crearAdmin.php y vistaAdmin.php a la de mensajes.php y la de perfil.php, además se agregó otra siendo esta de la cerrar.php que sirve para cerrar sesión.

En los servers, controllers y models se agregaron todo lo relacionado con el administrador(Trabajador) y los mensajes.

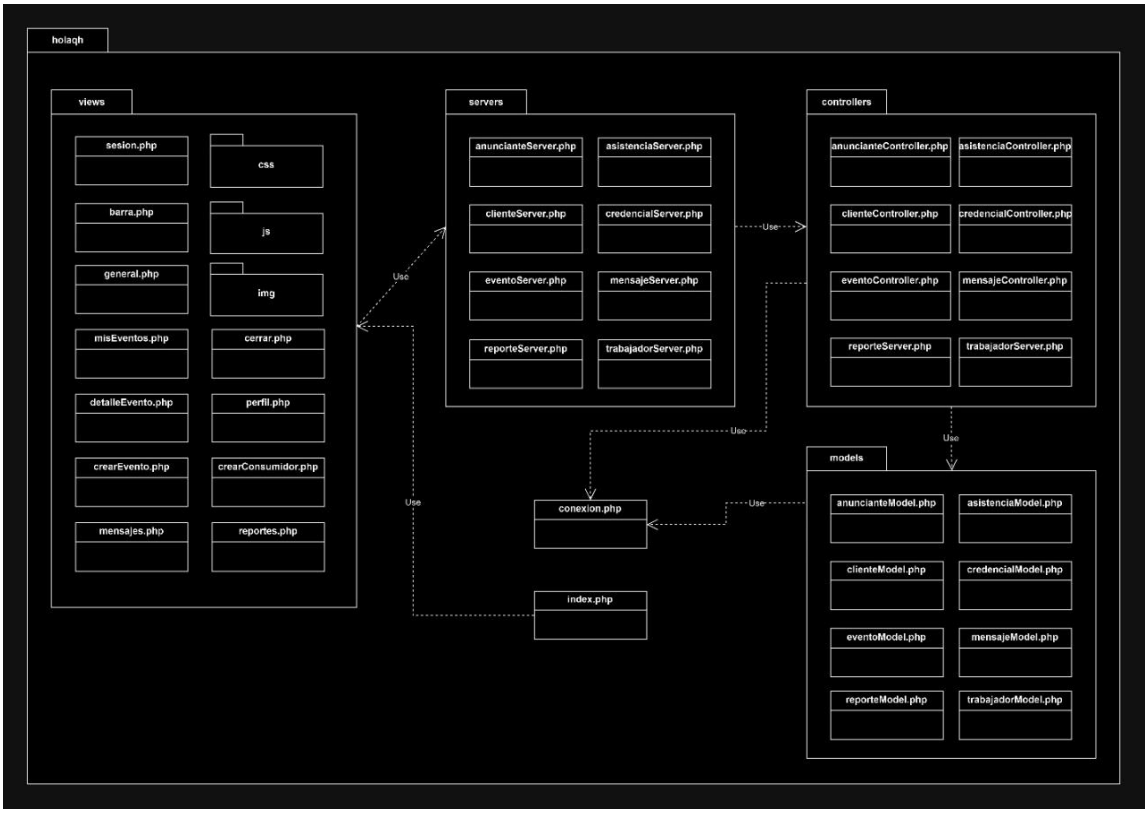
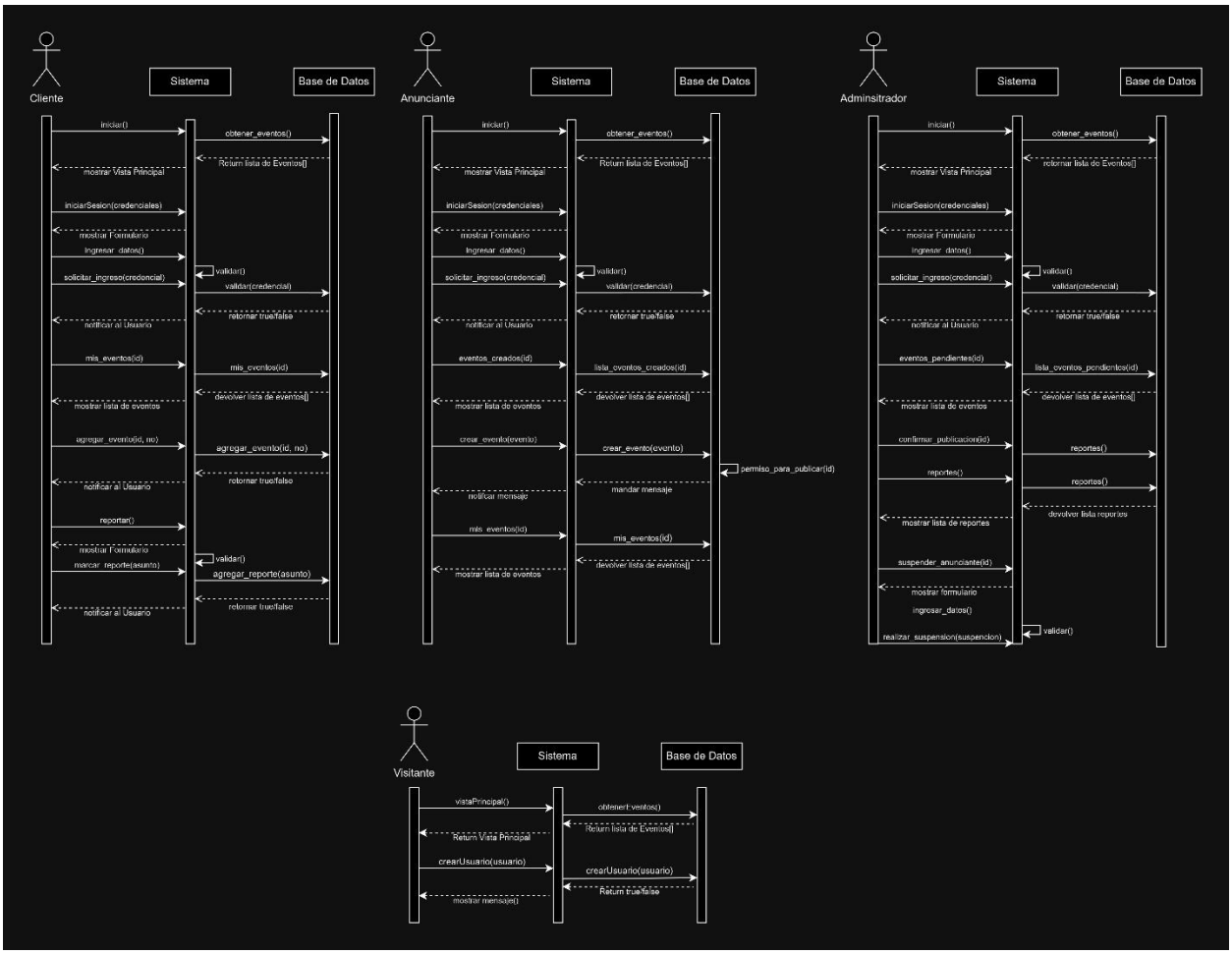


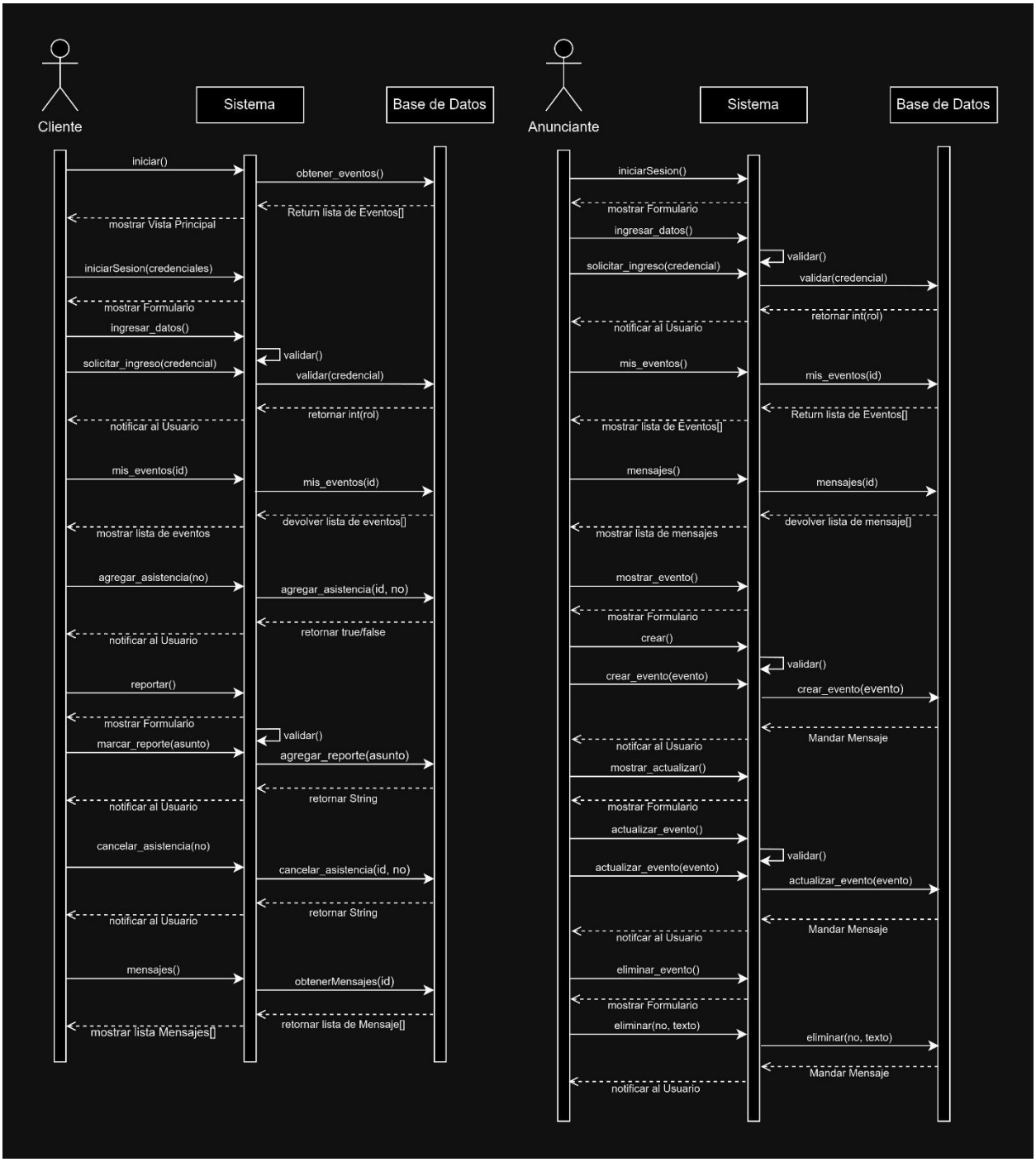
DIAGRAMA SECUENCIA

(ANTERIOR)



(ACTUAL)

Este diagrama tuvo los cambios más importantes ya que se modifico casi la mitad del planteamiento en cada uno de los casos ya que con forme se fue avanzando en el desarrollo del proyecto notamos que algunas ideas estaban bien mientras que otras eran deficientes por lo tanto se hicieron las correcciones necesarias



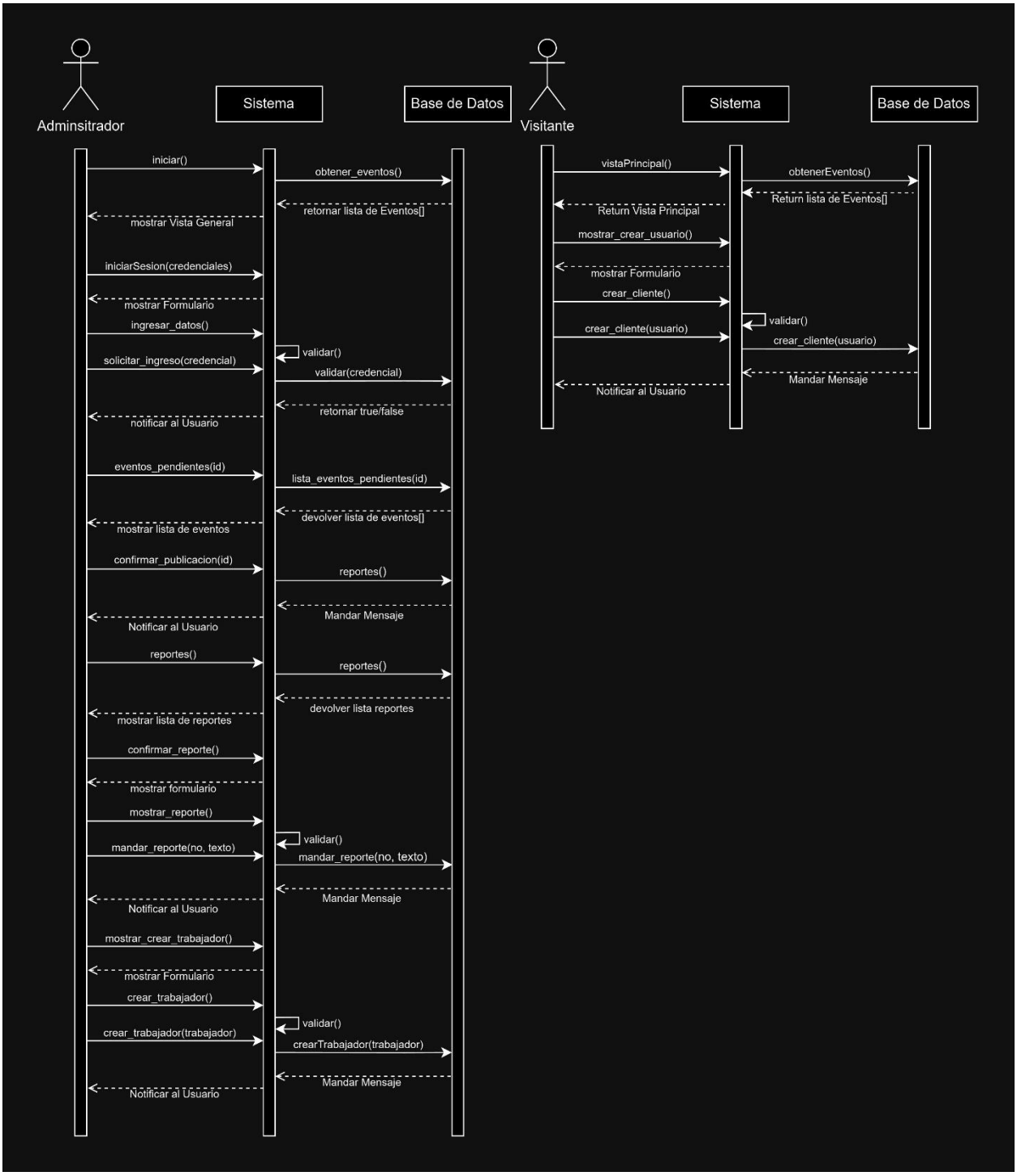

Adminsitrador
Sistema
Base de Datos
Visitante
Sistema
Base de Datos
iniciar()
obtener_eventos()
retornar lista de Eventos[]
mostrar Vista General
iniciarSesion(credenciales)
mostrar Formulario
ingresar_datos()
validar()
solicitar_ingreso(credencial)
validar(credencial)
retornar true/false
notificar al Usuario
eventos_pendientes(id)
lista_eventos_pendientes(id)
devolver lista de eventos[]
mostrar lista de eventos
confirmar_publicacion(id)
reportes()
Mandar Mensaje
Notificar al Usuario
reportes()
reportes()
devolver lista reportes
mostrar lista de reportes
confirmar_reporte()
mostrar formulario
mostrar_reporte()
validar()
mandar_reporte(no, texto)
mandar_reporte(no, texto)
Mandar Mensaje
Notificar al Usuario
mostrar_crear_trabajador()
mostrar Formulario
crear_trabajador()
validar()
crear_trabajador(trabajador)
crearTrabajador(trabajador)
Mandar Mensaje
Notificar al Usuario
vistaPrincipal()
obtenerEventos()
Return lista de Eventos[]
Return Vista Principal
mostrar_crear_usuario()
mostrar Formulario
crear_cliente()
validar()
crear_cliente(usuario)
crear_cliente(usuario)
Mandar Mensaje
Notificar al Usuario

DIAGRAMA DESPLIEGUE

Dado a que el uso en general de la apliación no cambio ya que se siguen usando solicitudes HTTP para realizar acciones en la aplicación, se sigue utilizando Xampp como servicio y la base de datos sigue siendo la misma la representación (ANTERIOR) y (ACTUAL) de este diagrama sigue conservando la misma estructura en comparación con a la entrega anterior.

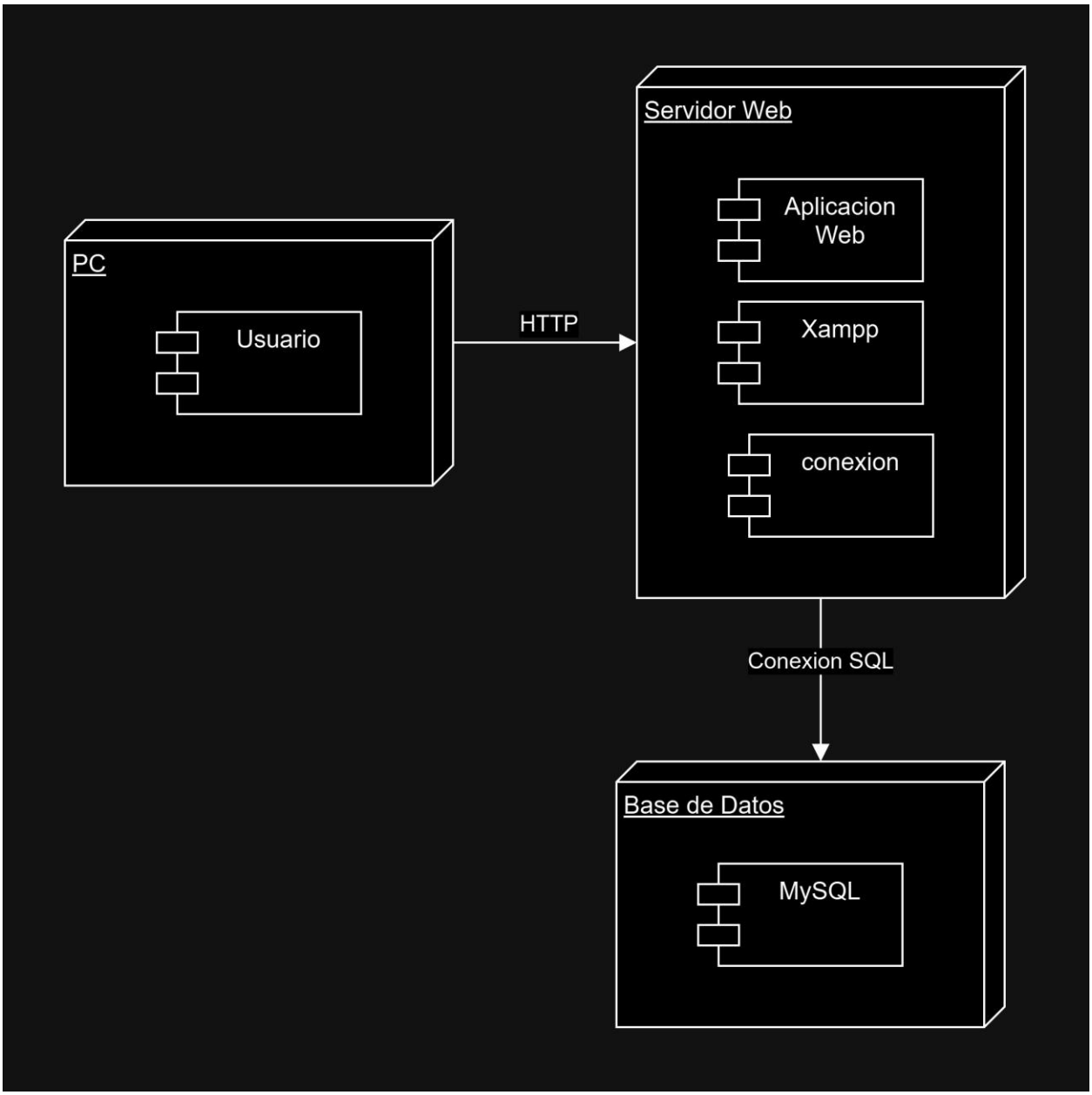


DIAGRAMA CASO DE USO

Este es un diagrama nuevo para nuestro proyecto, (NO PRESENTE EN LA ENTREGA ANTERIOR) el cual refleja el uso que cada usuario realiza en la aplicación y a que vistas y acciones tiene en teoría permiso según sea su rol dentro del sistema

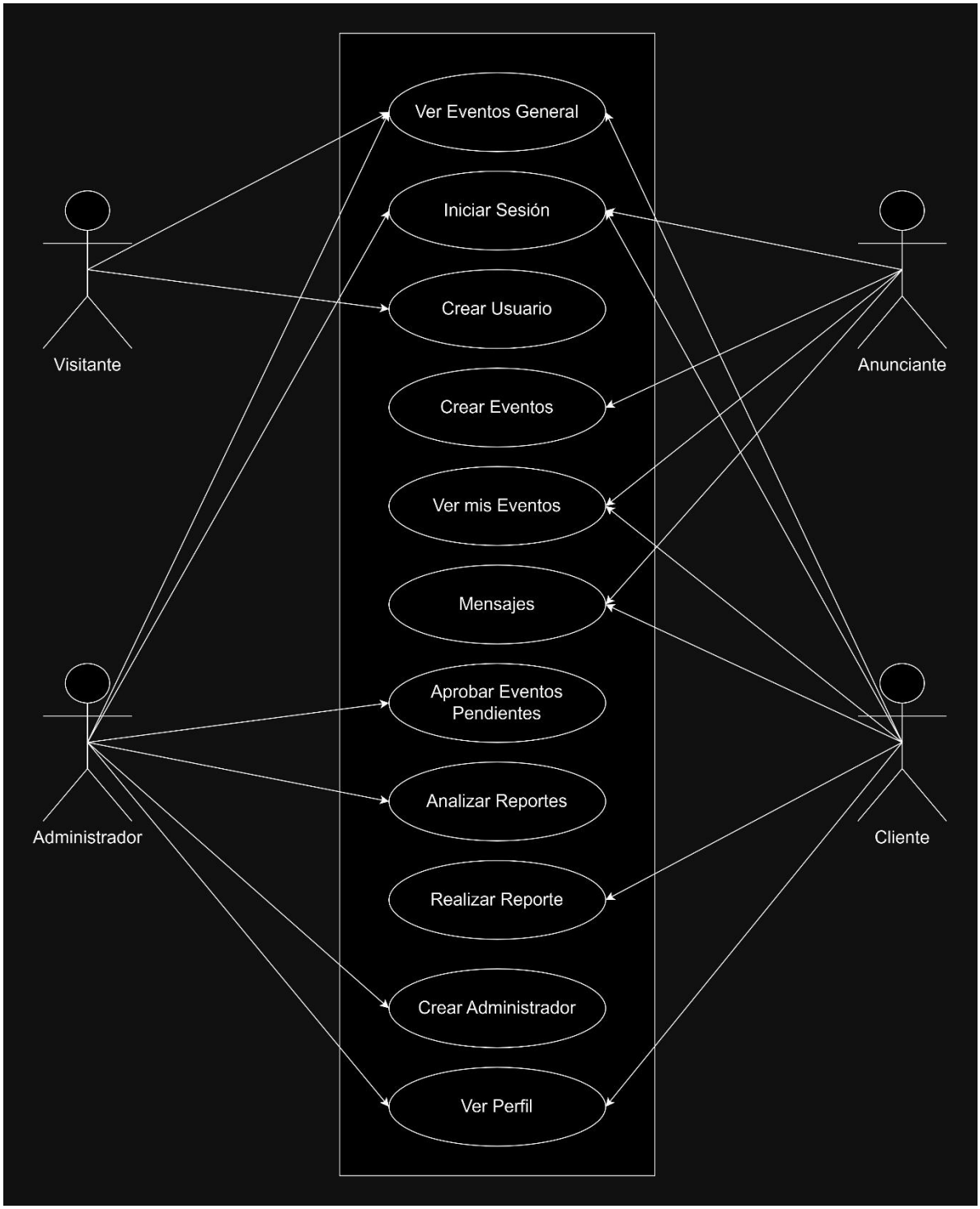
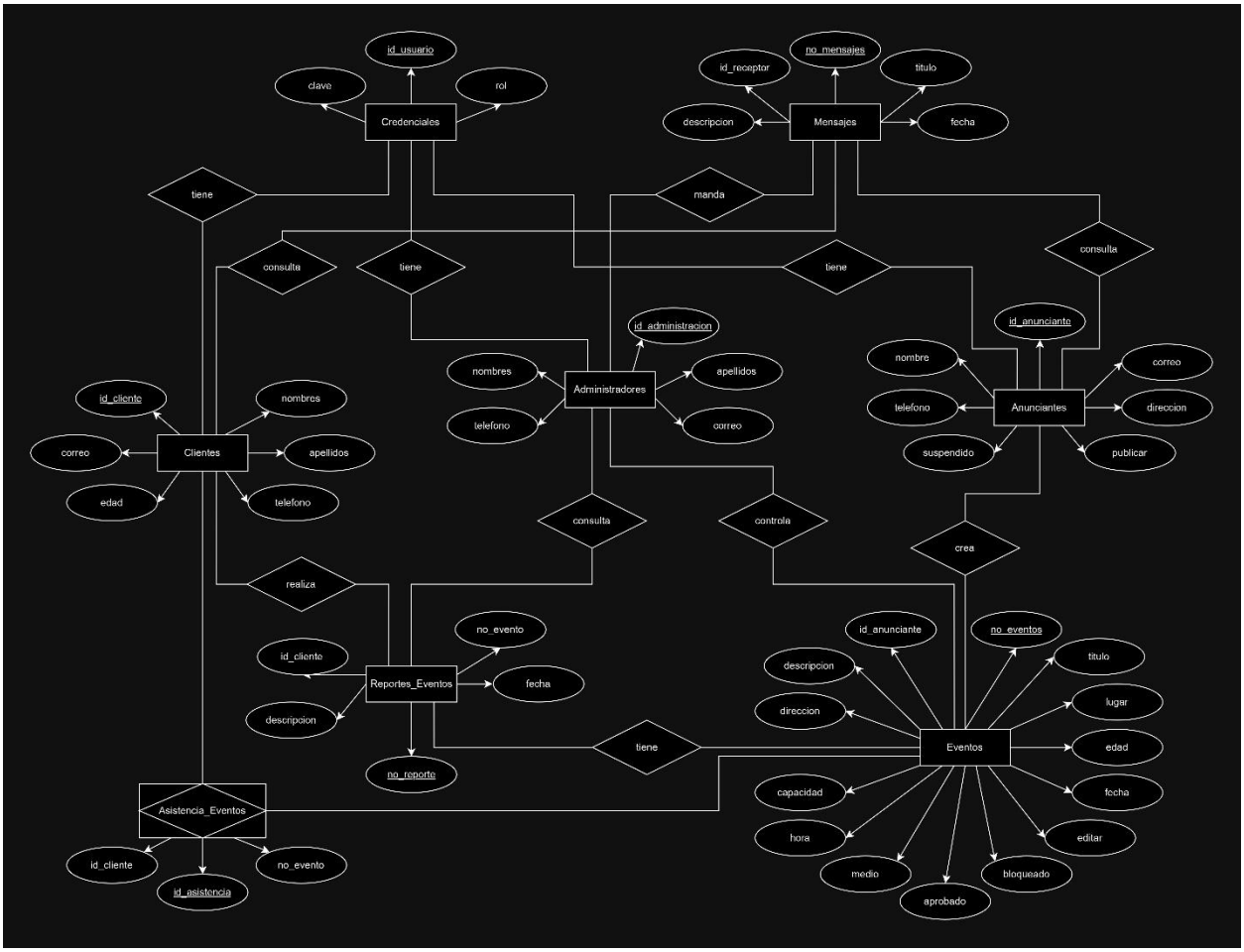


DIAGRAMA ENTIDAD/RELACIÓN

Este es un diagrama nuevo para nuestro proyecto, (NO PRESENTE EN LA ENTREGA ANTERIOR), nos presenta las relaciones entre las diferentes tablas que componen nuestra base de datos.



FUNCIONES DE LA BASE DE DATOS

Para el desarrollo de la base de datos se utilizaron un número de funciones que nos ayudaron a agilizar los procesos de nuestra aplicación:

```
/* Esta función nos sirve para eliminar el evento, pero antes de eso verifica que
la publicación no este reportada
   en caso lo este no permite la eliminación y en caso contrario borra el evento
y notifica a las personas que habian
   confirmado sus asistencia que el evento fue eliminado por x motivo. */
DELIMITER //
CREATE FUNCTION eliminar_evento(evento_no INT, mensaje_evento VARCHAR(400))
RETURNS VARCHAR(150)
BEGIN
    DECLARE conteo INT;

    SELECT COUNT(*) INTO conteo
    FROM reportes_eventos
    WHERE no_evento = evento_no;

    IF conteo > 0 THEN
        RETURN 'No se puede eliminar el Evento, porque tiene reportes';
    ELSE
        INSERT INTO mensajes(id_receptor, titulo, descripcion)
        SELECT id_cliente, 'Evento Eliminado', mensaje_evento
        FROM asistencia_eventos
        WHERE no_evento = evento_no;

        DELETE FROM asistencia_eventos WHERE no_evento = evento_no;
        DELETE FROM eventos WHERE no_evento = evento_no;

        RETURN 'Evento Eliminado';
    END IF;
END
// DELIMITER ;



/* Esta función es utilizada cuando el cliente quiere marcar sus asistencia a un
evento en caso ya lo haga hecho se
   le notificará al usuario.  */
DELIMITER //
CREATE FUNCTION verificar_asistencia_evento(cliente_id VARCHAR(75), evento_no
int)
```

```
RETURNS VARCHAR(50)
BEGIN
    DECLARE existe BOOLEAN;

    SELECT COUNT(*) INTO existe
    FROM asistencia_eventos
    WHERE id_cliente = cliente_id AND no_evento = evento_no;

    IF existe = 0 THEN
        INSERT INTO asistencia_eventos (id_cliente, no_evento) VALUES
(cliente_id, evento_no);
        RETURN 'Asistencia Confirmada';
    ELSE
        RETURN 'Ya confirmaste tú asistencia a este Evento';
    END IF;
END
// DELIMITER ;




/* Esta función sirve cuando el cliente quiere reportar una publicación, primero
verificamos que sea la primera vez
   que dicho cliente hace un reporte hacia dicha publicación en caso contrario no
se permite la acción, luego se
   realiza el reporte y se verifica el número de reportes de la página en caso
sea igual a 3 se tiene que
   desabilitar la opción de visualizar la publicación.  */
DELIMITER //
CREATE FUNCTION verificar_reporte_realizado(cliente_id VARCHAR(75), evento_no
INT, descripcion_reporte VARCHAR(255))
RETURNS VARCHAR(50)
BEGIN
    DECLARE existe INT;
    DECLARE numero_reportes INT;
    DECLARE anunciante_id VARCHAR(75);
    DECLARE nombre_evento VARCHAR(75);

    SELECT COUNT(*) INTO existe
    FROM reportes_eventos
    WHERE id_cliente = cliente_id AND no_evento = evento_no;

    SELECT id_anunciante,titulo INTO anunciante_id,nombre_evento
    FROM eventos
    WHERE no_evento = evento_no
```

```
    LIMIT 1;

    IF existe = 0 THEN

        INSERT INTO reportes_eventos (id_cliente, no_evento, descripcion)
        VALUES (cliente_id, evento_no, descripcion_reporte);

        SELECT COUNT(*) INTO numero_reportes
        FROM reportes_eventos
        WHERE no_evento = evento_no;

        IF numero_reportes >= 3 THEN
            INSERT INTO mensajes(id_receptor, titulo, descripcion)
            VALUES (anunciante_id, 'Publicación Bloqueada', CONCAT('La
Publicacion (', nombre_evento, ') fue retirada de la vista general al recibir 3
Reportes, espere la resolución de Administración sobre su caso.'));
            UPDATE eventos SET bloqueado = 1 WHERE no_evento = evento_no;
        END IF;

        RETURN 'Reporte Enviado';
    ELSE
        RETURN 'Ya hiciste un Reporte hacia esta Publicación';
    END IF;
END //
DELIMITER ;




/* Esta función es utilizada para aprobar el evento del Anunciante */
DELIMITER //
CREATE FUNCTION verificar_reportes_y_aprobacion_evento(evento_no int)
RETURNS INT
BEGIN
    DECLARE evento_aprobado INT;
    DECLARE existe INT;

    SELECT aprobado INTO evento_aprobado
    FROM eventos
    WHERE no_evento = evento_no
    LIMIT 1;

    SELECT COUNT(*) INTO existe
    FROM reportes_eventos
```

```
    WHERE no_evento = evento_no;

    IF existe = 0 THEN
        IF evento_aprobado = 1 THEN
            RETURN 1;
        ELSE
            RETURN 2;
        END IF;
    ELSE
        RETURN 3;
    END IF;
END
// DELIMITER ;




/* Este función se encarga de eliminar el evento que a sido seleccionado y de
hacer las notificaciones correspondientes
   tanto al administrador como al anunciante. */
DELIMITER //

CREATE FUNCTION aprobar_reporte(evento_no INT)
RETURNS VARCHAR(150)
BEGIN
    DECLARE anunciante_id VARCHAR(25);
    DECLARE titulo_evento VARCHAR(75);
    DECLARE publicar_anunciante TINYINT;
    DECLARE total_eventos INT;

    SELECT id_anunciante,titulo INTO anunciante_id,titulo_evento
    FROM eventos
    WHERE no_evento = evento_no
    LIMIT 1;

    SELECT COUNT(*) INTO total_eventos
    FROM eventos
    WHERE id_anunciante = anunciante_id;

    SELECT publicar INTO publicar_anunciante
    FROM anunciantes
    WHERE id_anunciante = anunciante_id
    LIMIT 1;
```

```
    INSERT INTO mensajes(id_receptor, titulo, descripcion)
    SELECT id_cliente, 'Evento Eliminado por Reporte', CONCAT('La Publicación del
evento (', titulo_evento, ') ha sido eliminado por infringir normas de la
Página.')
    FROM asistencia_eventos
    WHERE no_evento = evento_no;

    DELETE FROM reportes_eventos WHERE no_evento = evento_no;
    DELETE FROM asistencia_eventos WHERE no_evento = evento_no;
    DELETE FROM eventos WHERE no_evento = evento_no;

    IF publicar_anunciante = 1 THEN
        UPDATE anunciantes SET publicar = 0, aprobados = 0 WHERE id_anunciante =
anunciante_id;
        INSERT INTO mensajes(id_receptor, titulo, descripcion) VALUES
(anunciante_id, 'Retiro de Publicación Automatica', 'Debido a reportes recibidos
en sus publicaciones recientes se le retiro el permiso de publicación automatica,
debe acumular 2 publicaciones aprobadas para volver a obtener el beneficio.');
        RETURN 'Publicación Eliminada y mensaje de "Retiro de Publicación
Automatica" enviado al Anunciante';
    ELSE
        IF total_eventos <= 2 THEN
            UPDATE anunciantes SET suspendido = 1 WHERE id_anunciante =
anunciante_id;
            RETURN 'Publicación Eliminada y Anunciante Baneado';
        ELSE
            RETURN 'Publicación Eliminada, por favor verifique las publicaciones
que son Aprobadas';
        END IF;
    END IF;
END
// DELIMITER ;



/* Esta Función es utilizada para poder ignorar los reportes que tenga una
publicación y liberarla en caso este
   bloqueada o desmarcarla en caso tenga una advertencia. */
DELIMITER //
CREATE FUNCTION ignorar_reporte(evento_no INT)
RETURNS VARCHAR(150)
BEGIN
    DECLARE anunciante_id VARCHAR(25);
    DECLARE titulo_evento VARCHAR(75);
```

```
    SELECT id_anunciante,titulo INTO anunciante_id,titulo_evento
    FROM eventos
    WHERE no_evento = evento_no
    LIMIT 1;

    DELETE FROM reportes_eventos WHERE no_evento = evento_no;
    UPDATE eventos SET bloqueado = 0 WHERE no_evento = evento_no;
    INSERT INTO mensajes(id_receptor, titulo, descripcion) VALUES (anunciante_id,
'Retiro de Bloqueos',
    CONCAT('La publicación (', titulo_evento ,') oficialmente ha sido liberada de
los reportes que tenia.'));
    RETURN 'La Publicación ha sido Liberada de los Reportes';
END
// DELIMITER ;




/* Esta Función es utilida para validar la publicación de un anunciante y ponerlo
a la vista publica al realizarlo,
   se le notifica al usuario que su solicitud publicación ha sido exitosa, en
caso sea la tercera publicación aprobada
   se le notificara que ya puede publicar de manera automatica. */
DELIMITER //
CREATE FUNCTION autorizar_publicacion(evento_no INT)
RETURNS VARCHAR(150)
BEGIN
    DECLARE anunciante_id VARCHAR(25);
    DECLARE titulo_evento VARCHAR(75);
    DECLARE editado_evento VARCHAR(75);

    SELECT id_anunciante,titulo,editar INTO
anunciante_id,titulo_evento,editado_evento
    FROM eventos
    WHERE no_evento = evento_no
    LIMIT 1;

    IF editado_evento = 0 THEN
        UPDATE anunciantes SET aprobados = aprobados + 1 WHERE id_anunciante =
anunciante_id;
        UPDATE eventos SET aprobado = 1 WHERE no_evento = evento_no;
    ELSE
        UPDATE eventos SET aprobado = 1, editar = 0 WHERE no_evento = evento_no;
    END IF;
```

```
    INSERT INTO mensajes(id_receptor, titulo, descripcion)
    VALUES (anunciante_id, 'Publicación Autorizada',
    CONCAT('La publicación (', titulo_evento, ') ha sido autorizada por
administración.'));

    RETURN 'Publicación Autorizada, se notificará al Anunciante';
END
// DELIMITER ;




/* Esta Función es utilida para crear publicaciones. */
DELIMITER //
CREATE FUNCTION crear_publicacion(
    anunciante_id VARCHAR(75),
    titulo_nuevo VARCHAR(75),
    descripcion_nuevo VARCHAR(250),
    lugar_nuevo VARCHAR(75),
    direccion_nuevo VARCHAR(75),
    edad_nuevo INT,
    capacidad_nuevo INT,
    hora_nuevo TIME,
    fecha_nuevo DATE,
    medio_nuevo VARCHAR(300)
)
RETURNS VARCHAR(150)
BEGIN
    DECLARE publicar_anunciante TINYINT;
    DECLARE aprobados_anunciante INT;
    DECLARE total_eventos_no_aprobados INT;

    SELECT publicar, aprobados INTO publicar_anunciante, aprobados_anunciante
    FROM anunciantes
    WHERE id_anunciante = anunciante_id
    LIMIT 1;

    SELECT COUNT(*) INTO total_eventos_no_aprobados
    FROM eventos
    WHERE aprobado = 0 AND id_anunciante = anunciante_id;

    IF publicar_anunciante = 1 THEN
        INSERT INTO eventos (id_anunciante, titulo, descripcion, lugar,
direccion, edad, capacidad, hora, fecha, medio)
```

```
        VALUES (anunciante_id, titulo_nuevo, descripcion_nuevo, lugar_nuevo,
direccion_nuevo, edad_nuevo, capacidad_nuevo, hora_nuevo, fecha_nuevo,
medio_nuevo);
        RETURN 'Publicación creada Exitosamente';
    ELSE
        IF aprobados_anunciante >= 2 THEN
            UPDATE anunciantes SET publicar = 1 WHERE id_anunciante =
anunciante_id;
            INSERT INTO eventos (id_anunciante, titulo, descripcion, lugar,
direccion, edad, capacidad, hora, fecha, medio)
            VALUES (anunciante_id, titulo_nuevo, descripcion_nuevo, lugar_nuevo,
direccion_nuevo, edad_nuevo, capacidad_nuevo, hora_nuevo, fecha_nuevo,
medio_nuevo);
            RETURN 'Publicación creada Exitosamente, desde ahora obtienes el
beneficio de Publicación Automatica';
        ELSE
            /* IF aprobados_anunciante = 0 AND total_eventos_no_aprobados = 0
THEN */
            IF total_eventos_no_aprobados = 0 THEN
                INSERT INTO eventos (id_anunciante, titulo, descripcion, lugar,
direccion, edad, capacidad, hora, fecha, medio)
                VALUES (anunciante_id, titulo_nuevo, descripcion_nuevo,
lugar_nuevo, direccion_nuevo, edad_nuevo, capacidad_nuevo, hora_nuevo,
fecha_nuevo, medio_nuevo);
                RETURN 'Publicación creada Exitosamente, espere la aprobación de
Administración para que sea Visible para Todos';
            ELSE
                RETURN 'Debe esperar a que su Publicación anterior sea Aprobada
para poder Crear una Nueva.';
            END IF;
        END IF;
    END IF;

    RETURN 'Error al Crear la Publicación.';
END
// DELIMITER ;



/* Esta Función es utilida cuando no se aprueba una publiación y se le habilita
al usuario para que pueda realizar
   las restructuracciones necesarias para poder aprobar su publicación. */
DELIMITER //
CREATE FUNCTION denegar_publicacion(evento_no INT, descripcion_nuevo
VARCHAR(300))
```

```
RETURNS VARCHAR(200)
BEGIN
    DECLARE anunciante_id VARCHAR(25);
    DECLARE titulo_evento VARCHAR(75);

    SELECT id_anunciante, titulo INTO anunciante_id, titulo_evento
    FROM eventos
    WHERE no_evento = evento_no
    LIMIT 1;

    UPDATE eventos SET editar = 1 WHERE no_evento = evento_no;

    INSERT INTO mensajes(id_receptor, titulo, descripcion)
    VALUES (anunciante_id, 'Publicación Denegada', CONCAT('La publicación (',
titulo_evento ,') ha sido denegada por la Administración debido a que: (',
descripcion_nuevo ,') se le ha habilitado los derechos de edición para que haga
los cambios necesarios para que sea aprobada su publicación.'));

    RETURN 'Publicación Denegada, se notificará al Anunciante';
END
// DELIMITER ;
```