**Tablas**

Cabe señalar que todas las llaves primarias de todas la tablas tienen propiedad AUTO INCREMENT, para facilitar la inserción de nuevos campos.

**Usuario**

Es la tabla principal para el control de usuarios en la página. Guardará información necesaria para crear una cuenta, sea de Artista o de Fan. Es la primera información que se le pedirá a un usuario cuando quiera crear una cuenta.

* ID\_Usuario (int) (PK)

Cuando se registre un usuario en el sitio, se le asignará un identificador.

* Contrasena (varchar(50))

En el registro de la página se le pide al usuario una contraseña para ingresar.

* Correo (varchar(30))

Al registrarse, se pide un correo electrónico que se asociará con la cuenta.

* Nombre\_usuario (varchar(50))

El nombre de usuario que tendrá y lo identificará dentro de la página, se pide al registrarse.

* Imagen\_Perfil (varchar(50))

Guardará el path a la imagen que el usuario sube para utilizarla de “Imagen de perfil”.

* FN (date)

Fecha de Nacimiento, se pide con el fin de conocer si es mayor de edad o no.

* Pais (int) (FK)

El usuario seleccionará su país de residencia, este campo se relaciona con la tabla País para conocer el mismo.

* Estado (int) (FK)

El usuario también puede seleccionar su estado de residencia dependiendo del país que haya seleccionado, se relaciona con estado para obtener la lista.

* Bloqueado (int)

Indica si el usuario puede o no acceder al sitio, 0 es no bloqueado, 1 bloqueado. El moderador puede cambiar este campo y también se cambia por eventos.

* Tipo\_Usuario (int) (FK)

Se relaciona con la tabla “Tipo\_Usuario” para saber si será Artista o Fan.

* Permitir\_18 (int)

Identificará si el usuario desea ver contenido explícito dentro de la página o no, 0 no, 1 sí.

* Reset (varchar(50))

Campo auxiliar para cuando el usuario desee resetear su contraseña.

* Auto5 (int)

Sirve como bandera para saber cuando un usuario ya fue suspendido automáticamente por recibir 5 reportes (1 día suspensión).

* Auto10 (int)

Sirve como bandera para saber cuando un usuario ya fue suspendido automáticamente por recibir 10 reportes (7 días suspensión).

* Auto15 (int)

Sirve como bandera para saber cuando un usuario ya fue suspendido automáticamente por recibir 15 reportes (30 días suspensión).

* Auto20 (int)

Sirve como bandera para saber cuando un usuario ya fue suspendido automáticamente por recibir 20 reportes (100 años suspensión).

**Tipo\_Usuario**

Con esta tabla se identifica si el usuario, al momento de registrarse, seleccionó crear una cuenta de Artista o de Fan, para así pedir más información dependiendo del tipo de usuario. También los permisos que tendrán dentro de la página dependen de esto.

* Id\_TipoUsuario (int) (PK)

Identificador del tipo de usuario.

* Nombre\_TipoUsuario (varchar(10))

Valores estáticos: “Artista” y “Fan”.

**Artista**

La tabla de Artista guardará información específicamente de usuarios Artistas, es información que no se le pediría a un usuario Fan. Esta información se pide al momento que el usuario selecciona Artista durante el proceso de registro.

* Id\_Artista (int) (PK)

Identificador para un usuario artista.

* Informacion\_Contacto (varchar(200))

Se pedirá información de contacto al usuario artista, que puede ser correo electrónico, teléfono, vínculos a otras redes sociales, etc., y se mostrará en su perfil.

* Tecnica\_Interes (varchar(100))

El artista indicará cuál es su técnica de interés principal.

* Id\_Usuario (int) (FK)

Se relaciona con la tabla de Usuario para conocer su identificador.

* Id\_Diseno (int) (FK)

Se relaciona con Diseno para identificar el diseño de su perfil.

* Id\_Portafolio (int) (FK)

Se relaciona con Portafolio para saber la información de este.

* Id\_Perfil (int) (FK)

Se relaciona con Perfil para mostrar la información adicional en su perfil.

**Fan**

Cuando un usuario Fan se registra, se le pide información y se mostrará en su perfil de Fan, el cual es distinto a un perfil de Artista.

* Id\_Fan (PK)

Identificador usuario fan.

* Informacion\_Contacto (varachar(200))

Se pedirá información de contacto al usuario fan, que puede ser correo electrónico, teléfono, vínculos a otras redes sociales, etc., y se mostrará en su perfil.

* Perfil (varachar(200))

Guardará descripción del perfil del fan.

* Id\_Usuario (int) (FK)

Se relaciona con la tabla de Usuario para obtener la información previa.

**País**

La tabla país guarda un listado de países de los cuales el usuario puede seleccionar uno durante el proceso de registro para indicar en dónde vive.

* Id\_Pais (int) (PK)

Identificador de país.

* Nombre\_Pais (varchar(100))

Valores estáticos, lista de 20 países hispanohablantes: México, Costa Rica, El Salvador, Guatemala, Honduras, Nicaragua, Panamá, Cuba, Puerto Rico, República Dominicana, Argentina, Bolivia, Chile, Colombia, Ecuador, Paraguay, Perú, Uruguay, Venezuela, España.

**Estado**

La tabla Estado guarda un listado de estados de los cuales el usuario puede seleccionar uno durante el proceso de registro dependiendo del país que haya seleccionado previamente.

* Id\_Pais (PK)

Identificador de estado.

* Id\_Pais (FK)

Identifica a qué país corresponde el estado.

* Estado (varchar(50))

Valores estáticos, lista de todos los estados de México.

**Portafolio**

Esta tabla guardará la información del portafolio de un artista. Cada artista puede tener un portafolio. Será visible bajo la sección de “Portafolio” en el perfil.

* Id\_Portafolio (int) (PK)

Identificador del portafolio del artista.

* Descripcion (varchar(200))

El artista ingresará una descripción del portafolio..

**Colección**

Un portafolio tiene varias colecciones, así que esta tabla guardará la información de cada colección.

* Id\_Coleccion (int) (PK)

Identificador de la colección.

* Descripcion (varchar(200))

Guardará la descripción que el artista ingrese de la colección.

* Id\_Portafolio (int) (FK)

Se relaciona con Portafolio para saber a qué portafolio pertenece una colección.

I**magen\_Colección**

Esta tabla guardará todas las direcciones de las imágenes subidas por los artistas a sus respectivas colecciones.

* Id\_ImagenColeccion (int) (PK)

Identificador de la imagen.

* Imagen (varchar(200))

El artista subirá las imágenes para agregar a sus colecciones y se guardará la dirección de estas en el campo.

* Id\_Coleccion (int) (FK)

Se relaciona con Coleccion para identificar a qué colección le pertenece.

**Diseno**

Un artista podrá modificar el diseño de su perfil al crear una cuenta y después bajo la sección de Configuración, esta tabla guardará la información del diseño que tiene el perfil actualmente.

* Id\_Diseno (int) (PK)

Identificador de un diseño.

* Imagen\_Fondo (varchar(200))

El artista podrá subir una imagen para usarla de fondo de perfil.

* Color\_Bordes (char(7))

El artista podrá seleccionar el color de los bordes de los elementos del perfil.

* Color\_Titulos (char(7))

También podrá seleccionar el color de los títulos en el perfil.

* Color\_Botones (char(7))

Podrá modificar el color de los botones de su perfil.

* Color\_Fondo (char(7))

Podrá cambiar el color de fondo si no subió una imagen. Se guardará el código hexadecimal de los colores en estos campos

* Tipo\_Perfil (int) (FK)

Se relaciona con Tipo\_Perfil para identificar qué tipo de estructura tendrá el perfil.

**Tipo\_Perfil**

Esta tabla guardará el diseño de las tres estructuras de perfil que se tienen en la página.

* Id\_TipoPerfil (int) (PK)

Identificador de los tipos.

* Tipo\_Perfil (varchar(30))

Tres valores: “Una columna”, “Dos columnas”, “Tres columnas”, el artista podrá seleccionar uno de estos.

**Perfil**

La tabla perfil guardará información adicional en el perfil del Artista, esta información es opcional, podrá ingresarla o no.

* Id\_Perfil (int) (PK)

Identificador de perfil, se refiere a información adicional de la requerida.

* Metas (varchar(200))

El artista podrá dar información sobre sus metas personales.

* Exper (varchar(200))

También podrá proporcionar información referente a su experiencia en el campo artístico.

* Otro (varchar(200))

En este campo se podrá dar cualquier tipo de información, por ejemplo, gustos, preferencias, ideas, etc.

* Estudios (varchar(200))

Campo para dar información sobre estudios del artista.

**Publicación**

Esta tabla guardará la información de las publicaciones en la página cada que los artistas realicen una.

* Id\_Publicacion (int) (PK)

Identificador de una publicación.

* Fecha (date)

Indicará la fecha en la que se hizo una publicación.

* Contenido\_Explicito (int)

Una publicación podrá ser marcada como “Contenido explícito”, el campo guarda ese estado, para saber si se mostra a usuarios con el filtro o no.

* Contenido (varchar(200))

Guardará el texto de la publicación.

* Etiquetas (varchar(200))

Guardará las etiquetas que se le dan a la publicación.

* Privacidad (int)

El artista podrá indicar si la publicación es pública o privada.

* Imagen (varchar(200))

Guardará la dirección de la imagen adjunta a la publicación.

* Id\_Artista (int) (FK)

Se relaciona con Artista para saber qué artista hizo la publicación.

* Ocultar (int)

El campo cambia a 1 cuando se desea ocultar la publicación, si lo desea el usuario o por reportes que haya recibido.

**Me\_gusta**

Cuando un usuario realice una evaluación de “Me gusta” o “No me gusta”, se guardará la información en esta tabla.

* Id\_MeGusta (int) (PK)

Identificador de un Me gusta.

* Tipo\_MeGusta (int)

Indica si es “Me gusta” o “No me gusta”, si es el segundo, se pedirá crítica constructiva al evaluador.

* Id\_Publicacion (int) (FK)

Se relaciona con Publicación para saber en qué publicación se aplicó el Me Gusta.

* Id\_Usuario (int) (FK)

Se relaciona con Usuario para saber qué usuario hizo la evaluación.

**Comentario**

Los comentarios hechos en publicaciones se guardarán junto con toda su información en esta tabla, incluyendo quién realizó el comentario.

* Id\_Comentario (int) (PK)

Identificador de un comentario.

* Fecha (date)

Indicará la fecha en la que se realizó el comentario.

* Contenido (varchar(200))

Guardará el texto del comentario.

* Id\_Usuario (int) (FK)

Se relaciona con Usuario para saber qué usuario hizo el comentario.

* Id\_Publicacion (int) (FK)

Se relaciona con Publicación para saber en dónde se realizó el comentario.

* Ocultar (int)

El campo cambia a 1 cuando se desea ocultar el comentario, si lo desea el usuario o por reportes que haya recibido.

**Retos\_aceptados**

Los usuarios podrán ver los retos y aceptarlos, en caso de ser así, esta tabla se actualizará con la información, para que el artista pueda participar en el reto.

* Id\_Aceptado (int) (PK)

Identificador de un reto ya aceptado por el artista.

* Id\_Artista (int) (FK)

Se relaciona con Artista para saber cuál artista aceptó un reto.

* Id\_Reto (int) (FK)

Se relaciona con Reto para saber qué reto fue el que se aceptó.

**Retos**

La información de los retos semanales se actualizará en esta tabla, para así mostrar a los artistas qué retos están disponibles.

* Id\_Reto (int) (PK)

Identifica a los retos.

* Fecha (date)

Indicará la fecha en la que se publicó el reto.

* Descripcion (varchar(200))

Guardará la descripción del reto.

**Apoyo\_visual**

Los retos contendrán apoyo visual adjunto, estos apoyos se guardarán en esta tabla para ser mostrados con su respectivo reto.

* Id\_Apoyo (int) (PK)

Identifica a un apoyo visual.

* Imagen (varchar(200))

Guardará la dirección de la imagen usada como apoyo visual.

* Descripcion (varchar(200))

Guardará la descripción de un apoyo visual.

* Id\_Reto (int) (FK)

Se relaciona con reto para saber en cuál reto se mostrará el apoyo visual.

**Imagen\_Reto**

Cuando un usuario artista ya aceptó un reto para participar en él, subirá una imagen para mostrarla a los demás. Las direcciones de las imágenes se guardarán en esta tabla para mostrarlas en sus respectivos retos.

* Id\_ImagenReto (int) (PK)

Identificador de una imagen subida por el artista para participar en un reto.

* Imagen (varchar(200))

Guardará la dirección de la imagen subida por el artista.

* Descripcion (varchar(200))

El artista podrá dar una descripción al participar en el reto.

* Id\_Aceptado (int) (FK)

Se relaciona con la tabla Retos\_aceptados para saber en cuál reto estará participando.

**Foro\_Hilo**

Ambos tipos de usuarios pueden crear un nuevo hilo en el foro que se les permita, cuando pase esto, se guardará la información del hilo en esta tabla.

* Id\_ForoHilo (int) (PK)

Identifica un hilo en los foros.

* Fecha (date)

Guardará la fecha en la que se abrió el hilo.

* Contenido (varchar(200))

Guardará el texto dado por el usuario al crear el hilo.

* Titulo (varchar(100))

Guardará el título del hilo.

* Id\_ForoTipo (int) (FK)

Se relaciona con ForoTipo para saber en qué sección de los foros se subirá el hilo.

* Id\_Usuario (int) (FK)

Se relaciona con Usuario para saber qué usuario hizo el hilo.

* Ocultar (int)

El campo cambia a 1 cuando se desea ocultar el comentario, si lo desea el usuario o por reportes que haya recibido.

**Foro\_Respuesta**

Los dos tipos de usuarios también pueden responder a un hilo si tienen visibilidad de este, si agregan una respuesta, se guardará la información de esta en la tabla Foro\_Respuesta.

* Id\_ForoRespuesta (int) (PK)

Identifica una respuesta a un hilo.

* Fecha (date)

Guardará la fecha en la que se realizó la respuesta.

* Contenido (varchar(200))

Guardará el texto de la respuesta.

* Id\_ForoHilo (int) (FK)

Se relaciona con Foro\_Hilo para saber en qué hilo se realizó la respuesta.

* Id\_Usuario (int) (FK)

Se relaciona con Usuario para saber qué usuario respondió.

* Ocultar (int)

El campo cambia a 1 cuando se desea ocultar el comentario, si lo desea el usuario o por reportes que haya recibido.

**Foro\_Tipo**

Los tipos de foro son dos, los usuarios Artistas pueden acceder a ambos pero los usuarios Fans sólo pueden acceder al foro de Ideas, con esta tabla se denegará o permitirá el acceso a los usuarios.

* Id\_Tipo (int) (PK)

Identifica los tipos de foros.

* Tipo (varchar(50))

Dos valores: “Ideas” y “Objetos”.

**Foro\_Favs**

Un usuario de cualquier tipo puede guardar un hilo dentro de sus Favoritos, para así darle seguimiento al hilo. Los hilos que tenga marcados como favoritos se mostrarán con la información de esta tabla.

* Id\_Favs (int) (PK)

Identifica un hilo marcado como favorito.

* Id\_Usuario (int) (FK)

Se relaciona con usuario para conocer cuáles son los hilos que ha marcado como favoritos.

* Id\_ForoHilo (int) (FK)

Se relaciona con ForoHilo para saber cuál hilo se guardará en favoritos.

**Amigos**

Los usuarios artistas podrán agregarse uno al otro como Amigos, esta tabla se utiliza para llevar registro de estas relaciones de amistad y así dar funciones de Amigos entre los artistas, como mostrar publicaciones privadas.

* Id\_Amigos (int) (PK)

Identifica una relación de amistad entre dos artistas.

* Estado (int)

Indica el estado de amistad, si ya son amigos o no.

* Id\_Usuario1 (int) (FK)

Identifica a un artista de la relación de amistad.

* Id\_Usuario2 (int) (FK)

Identifica al otro artista de la relación de amistad.

**Seguidores**

Un usuario fan o artista podrán seguir a un artista para poder ver sus publicaciones en su inicio y llevar seguimiento de su trabajo. La tabla se utiliza para llevar registro de la relación de seguimiento.

* Id\_Seguidores (int) (PK)

Identifica una relación de seguimiento.

* Id\_Usuario1 (int) (FK)

Indica qué usuario es el seguidor.

* Id\_Usuario2 (int) (FK)

Identifica al artista siendo seguido por el usuario1.

**Conversación**

Esta tabla creará el vínculo entre dos usuarios para que puedan mandar mensajes uno al otro.

* Id\_Conversacion (int) (PK)

Identifica la conversación entre los usuarios.

* Id\_Usuario1 (int) (FK)

Identificará a un usuario de la conversación.

* Id\_Usuario2 (int) (FK)

Identificará al otro usuario de la conversación.

**Mensaje**

Los usuarios podrán mandar mensajes entre ellos, esta tabla llevará el registro de todos los mensajes enviados dentro de la página.

* Id\_Mensaje (int) (PK)

Identificador del mensaje.

* Fecha (datetime)

Indicará cuándo se mandó el mensaje.

* Texto (varchar(200))

Contendrá el texto del mensaje.

* Id\_Conversacion (int) (FK)

Se relaciona con la tabla Conversación para saber qué usuarios son los que participan.

* Id\_Usuario (int) (FK)

Se relaciona con Usuario para saber cuál usuario es el remitente.

**Reportes\_comentarios**

Cuando un usuario reporte un comentario, se mandará registro de este reporte a esta tabla de reportes.

* Id\_reporte (int) (PK)

Identificador del reporte.

* Id\_comentario (int) (FK)

Identifica el comentario que fue reportado.

* Id\_reportado (int) (FK)

Indica cuál fue el usuario que fue reportado.

* Id\_reportero (int) (FK)

Identifica al usuario que hizo el reporte.

* Razon (varchar(50))

Guarda la razón por la cuál fue reportado el comentario.

* Estatus (int)

Indica si el reporte ya fue revisado por un moderador o no.

* Fecha (datetime)

Indicará cuándo se mandó el reporte.

**Reportes\_usuarios**

Cuando un usuario reporte a otro usuario, se mandará registro de este reporte a esta tabla de reportes.

* Id\_reporte (int) (PK)

Identificador del reporte.

* Id\_reportado (int) (FK)

Indica cuál fue el usuario que fue reportado.

* Id\_reportero (int) (FK)

Identifica al usuario que hizo el reporte.

* Razon (varchar(50))

Guarda la razón por la cuál fue reportado el usuario.

* Estatus (int)

Indica si el reporte ya fue revisado por un moderador o no.

* Fecha (datetime)

Indicará cuándo se mandó el reporte.

**Reportes\_publicaciones**

Cuando un usuario reporte una publicación, se mandará registro de este reporte a esta tabla de reportes.

* Id\_reporte (int) (PK)

Identificador del reporte.

* Id\_publicacion (int) (FK)

Identifica la publicación que fue reportada.

* Id\_reportado (int) (FK)

Indica cuál fue el usuario que fue reportado.

* Id\_reportero (int) (FK)

Identifica al usuario que hizo el reporte.

* Razon (varchar(50))

Guarda la razón por la cuál fue reportada la publicación.

* Estatus (int)

Indica si el reporte ya fue revisado por un moderador o no.

* Fecha (datetime)

Indicará cuándo se mandó el reporte.

**Bloqueados**

Esta tabla guarda registro de los usuarios que han sido bloqueados, el tiempo que fueron bloqueados y si el bloqueo ya expiró o no.

* Id\_bloqueado (int) (PK)

Identificador del registro de bloqueo.

* Id\_usuario (int) (FK)

Indica cuál fue el usuario bloqueado del sistema.

* Inicio (datetime)

Marca la fecha cuando inició el bloqueo.

* Fin (datetime)

Marca la fecha cuando se terminará el bloqueo.

* Expirado (int)

Indica si el bloqueo todavía está vigente o ya expiró.

**Notificaciones**

Cuando se presente un nuevo evento de Me gusta, Comentario, Respuesta en foro, Nuevo seguidor, Nueva solicitud, Nuevo amigo, Reporte, se le notificará al usuario afectado del evento y se guardará en esta tabla el registro de la notificación.

* Id\_notificacion (int) (PK)

Identificador de la notificación.

* Tipo (int)

Identifica el tipo de notificación.

* Contenido (varchar(200))

Guarda el texto que mostrará la notificación, por ejemplo, “recibiste un nuevo reporte”.

* Id\_usuario (int) (FK)

Indica cuál fue el usuario afectado por el evento.

* Visto (int)

Guarda un estado si la notificación ya fue vista o no.

* Fecha (datetime)

Indicará cuándo se originó la notificación.

* Id\_evento (int)

Guarda el id de la publicación, perfil, hilo, etc. que fue afectado por el evento.

* Id\_usuario1 (int) (FK)

Indica cuál fue el usuario que realizó el evento.

**Triggers**

**Desbloquear**

CREATE TRIGGER `desbloquear` AFTER UPDATE ON `bloqueados`

FOR EACH ROW begin

update usuario set bloqueado = 0 where usuario.id\_usuario = old.id\_usuario and (old.expirado = 1);

end

El trigger Desbloquear checa la tabla Bloqueados después de que esta es updateada y busca el campo Expirado, comprueba si es 1, que significa que el bloqueo ya expiró, y entonces hace un update en la tabla de Usuario en el campo Bloqueado, actualizándolo con un 0 para indicar que el usuario fue desbloqueado. Comprueba el Id del usuario que expiró su bloqueo con el de la tabla de Usuarios.

**Events**

**Expirar\_bloqueado**

CREATE DEFINER=`root`@`localhost` EVENT `expirar\_bloqueado` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-07 13:50:54' ON COMPLETION NOT PRESERVE ENABLE DO UPDATE bloqueados SET expirado = 1 WHERE fin < NOW()

Este evento está checando constantemente la tabla de Bloqueados en el campo Fin, en caso de que un campo Fin sea menor a NOW(), o sea el tiempo en el que está checando el evento, va a updatear el campo Expirado y lo pondrá en 1, indicando que ya pasó ese bloqueo.

**Borrar\_noti**

CREATE DEFINER=`root`@`localhost` EVENT `borrar\_noti` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-07 13:50:54' ON COMPLETION NOT PRESERVE ENABLE DO DELETE FROM notificaciones WHERE fecha < DATE\_ADD(fecha, INTERVAL 7 DAY) AND visto=1

El evento borrar\_noti está haciendo una consulta constantemente a la tabla de Notificaciones en el campo de Visto y en el campo de Fecha, si Visto es 1, que significa que el usuario ya vio la notificación, y si ya pasaron 7 días desde el registro de Fecha, el evento borra el registro de la notificación.

**Autobloq\_5reportes**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq\_5reportes` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 12:56:39' ON COMPLETION NOT PRESERVE ENABLE DO begin

update usuario join (select id\_reportado, estatus from reportes\_comentarios where estatus = 1 group by id\_reportado having count(\*) = 5 ) reportes\_comentarios ON usuario.id\_usuario = reportes\_comentarios.id\_reportado AND auto5 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto5 = 1;

update usuario join (select id\_reportado, estatus from reportes\_publicaciones where estatus = 1 group by id\_reportado having count(\*) = 5 ) reportes\_publicaciones ON usuario.id\_usuario = reportes\_publicaciones.id\_reportado AND auto5 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto5 = 1;

update usuario join (select id\_reportado, estatus from reportes\_usuarios where estatus = 1 group by id\_reportado having count(\*) = 5 ) reportes\_usuarios ON usuario.id\_usuario = reportes\_usuarios.id\_reportado AND auto5 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto5 = 1;

end

Este evento se encarga de bloquear usuarios cuando estos hayan alcanzado 5 reportes por la misma razón. El evento hace consultas en 3 tablas, que son reportes\_comentarios, reportes\_publicaciones y reportes\_usuarios, y busca campos en los que el estatus del reporte sea 1, que indica aprobado, si encuentra 5 incidencias del mismo id\_reportado, se hace un update en la tabla usuario con un join en id\_usuario = id\_reportado y auto5 = 0, para comprobar que el usuario no ha sido bloqueado de esta manera antes, actualizando los valores de Bloqueado y Auto5 a 1, bloqueando al usuario del sistema y activando la bandera para hacer la inserción en la tabla de bloqueados.

**Autobloq5\_insert**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq5\_insert` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 13:47:54' ON COMPLETION PRESERVE ENABLE DO begin

insert into bloqueados values (null, (select id\_usuario from usuario where auto5=1), NOW(), DATE\_ADD(NOW(), INTERVAL 1 DAY), 0);

update usuario set auto5 = 2 where auto5 = 1;

end

El evento de autobloq5\_insert se encarga de estar checando constantemente la bandera de Auto5 en la tabla de Usuarios, como se comentaba en el evento anterior, esta se activa cuando un usuario queda bloqueado por 5 reportes por la misma razón. Si la bandera está en 1, el evento inserta en la tabla de Bloqueados el id del usuario bloqueado, el tiempo actual (NOW()) como inicio y un día después como fin del b loqueo. También actualiza la bandera en la tabla de usuario con valor 2 para que no vuelva a ocurrir el mismo evento varias veces.

**Autobloq\_10reportes**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq\_10reportes` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 12:56:41' ON COMPLETION NOT PRESERVE ENABLE DO begin

update usuario join (select id\_reportado, estatus from reportes\_comentarios where estatus = 1 group by id\_reportado having count(\*) = 10 ) reportes\_comentarios ON usuario.id\_usuario = reportes\_comentarios.id\_reportado AND auto10 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto10 = 1;

update usuario join (select id\_reportado, estatus from reportes\_publicaciones where estatus = 1 group by id\_reportado having count(\*) = 10 ) reportes\_publicaciones ON usuario.id\_usuario = reportes\_publicaciones.id\_reportado AND auto10 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto10 = 1;

update usuario join (select id\_reportado, estatus from reportes\_usuarios where estatus = 1 group by id\_reportado having count(\*) = 10 ) reportes\_usuarios ON usuario.id\_usuario = reportes\_usuarios.id\_reportado AND auto10 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto10 = 1;

end

Este evento y los consecutivos funcionan igual que Autobloq\_5reportes, con la diferencia que este checa por 10 incidencias del mismo usuario y activa la bandera de Auto10.

**Autobloq10\_insert**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq10\_insert` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 13:47:54' ON COMPLETION PRESERVE ENABLE DO begin

insert into bloqueados values (null, (select id\_usuario from usuario where auto10=1), NOW(), DATE\_ADD(NOW(), INTERVAL 7 DAY), 0);

update usuario set auto10 = 2 where auto10 = 1;

end

La inserción de 10 checa el campo de Auto10 y si está en 1, inserta el id del usuario en Bloqueados con Inicio ahora y Fin 7 días después. También actualiza la bandera a 2.

**Autobloq\_15reportes**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq\_15reportes` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 12:56:43' ON COMPLETION NOT PRESERVE ENABLE DO begin

update usuario join (select id\_reportado, estatus from reportes\_comentarios where estatus = 1 group by id\_reportado having count(\*) = 15 ) reportes\_comentarios ON usuario.id\_usuario = reportes\_comentarios.id\_reportado AND auto15 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto15 = 1;

update usuario join (select id\_reportado, estatus from reportes\_publicaciones where estatus = 1 group by id\_reportado having count(\*) = 15 ) reportes\_publicaciones ON usuario.id\_usuario = reportes\_publicaciones.id\_reportado AND auto15 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto15 = 1;

update usuario join (select id\_reportado, estatus from reportes\_usuarios where estatus = 1 group by id\_reportado having count(\*) = 15 ) reportes\_usuarios ON usuario.id\_usuario = reportes\_usuarios.id\_reportado AND auto15 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto15 = 1;

end

Como se comentaba, este evento funciona igual a los otros 2 de bloqueo, pero busca 15 incidencias y activa la bandera Auto15.

**Autobloq15\_insert**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq15\_insert` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 13:47:54' ON COMPLETION PRESERVE ENABLE DO begin

insert into bloqueados values (null, (select id\_usuario from usuario where auto15=1), NOW(), DATE\_ADD(NOW(), INTERVAL 30 DAY), 0);

update usuario set auto15 = 2 where auto15 = 1;

end

La inserción por 15 incidencias bloquea al usuario por un total de 30 días después del momento de bloqueo y actualiza la bandera Auto15 a 2.

**Autobloq\_20reportes**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq\_20reportes` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 12:56:45' ON COMPLETION NOT PRESERVE ENABLE DO begin

update usuario join (select id\_reportado, estatus from reportes\_comentarios where estatus = 1 group by id\_reportado having count(\*) = 20 ) reportes\_comentarios ON usuario.id\_usuario = reportes\_comentarios.id\_reportado AND auto20 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto20 = 1;

update usuario join (select id\_reportado, estatus from reportes\_publicaciones where estatus = 1 group by id\_reportado having count(\*) = 20 ) reportes\_publicaciones ON usuario.id\_usuario = reportes\_publicaciones.id\_reportado AND auto20 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto20 = 1;

update usuario join (select id\_reportado, estatus from reportes\_usuarios where estatus = 1 group by id\_reportado having count(\*) = 20 ) reportes\_usuarios ON usuario.id\_usuario = reportes\_usuarios.id\_reportado AND auto20 = 0 AND bloqueado = 0 SET usuario.bloqueado=1, auto20 = 1;

end

Por último de eventos de reportes, la modalidad de 20 busca por 20 reportes aceptados con el mismo id de usuario y activa la bandera Auto20.

**Autobloq20\_insert**

CREATE DEFINER=`root`@`localhost` EVENT `autobloq20\_insert` ON SCHEDULE EVERY 1 SECOND STARTS '2018-06-11 13:47:54' ON COMPLETION PRESERVE ENABLE DO begin

insert into bloqueados values (null, (select id\_usuario from usuario where auto20=1), NOW(), DATE\_ADD(NOW(), INTERVAL 100 YEAR), 0);

update usuario set auto20 = 2 where auto20 = 1;

end

El bloqueo por 20 reportes equivale a 100 años de bloqueo, simbolizando un bloqueo permanente del sistema al usuario.