PERÍODO 2020

MANUAL TÉCNICO

"THE BOX"- CAJA DE SEGURIDAD INTELIGENTE

GRUPO # 10

CHIQUITO MOLINA, GUZMAN AVELLAN, TROYA TORO, VALDEZ YEPEZ

MATERIA: PROGRAMACIÓN DE SISTEMAS TELEMÁTICOS

CARRERA DE INGENIERÍA EN MECATRÓNICA

FIEC - ESPOL

1. Resumen Ejecutivo

Hoy en día estamos viviendo una era de cambios exponenciales impulsados por la innovación y el rápido avance de las tecnologías que juegan un rol fundamental en la transformación de las organizaciones y de las personas, lo cual nos lleva a crear soluciones eficientes y de alta calidad.

Nuestra aplicación móvil "The Box" sintetiza la visión del grupo, desde la cual aspiramos a desarrollar un sistema de seguridad aplicado a cajas fuertes o cajas inteligentes basado en la tecnología electrónica y de programación orientándolo a generar respuestas al problema de la seguridad. Se trata de administrar el acceso a una caja fuerte a través de un sistema de escaneo de código QR, el cual será ejecutado desde un dispositivo móvil que cuente con una cámara fotográfica y acceso a internet garantizando su invulnerabilidad.

El desarrollo de esta caja inteligente conduce a explorar diferentes usos y adaptaciones de dispositivos tecnológicos que se encuentran disponibles, no generando costos adicionales.

2. Descripción del problema

La inseguridad es, desde épocas muy antiguas, un gran problema de las sociedades, porque conforme se genera nuevos sistemas de seguridad, otros buscan hacerlos caer. Eso conlleva a estar constantemente trabajando para generar nuevas soluciones.

El desarrollo de una caja fuerte inteligente surge como respuesta a la creciente demanda por mejores sistemas de seguridad de parte de muchos sectores como los gobiernos, bancos, empresas y el público en general, que necesitan soluciones óptimas, eficientes y que les permita controlarlas directamente, evitando así la intervención de terceros.

3. Objetivos Específicos

- Crear un prototipo para automatizar el acceso a cajas de seguridad a través de una aplicación móvil que contiene un escáner de código QR.
- Mejorar la administración y control de las cajas a través de la combinación de ciencias y tecnologías que disponemos.

4. ¿Cómo funciona la solución propuesta?

Estableciendo la idea propuesta en un contexto comercial se ha dispuesto que el proceso de adquisición de una caja por parte del usuario ocurra de la siguiente manera:

1. Los proveedores -el grupo de trabajo- elaboran la caja junto con los componentes de control de acceso, se genera un código QR aleatorio -compuesto de números y/o caracteres- que también es situado en la caja. El código se generará con la librería PHP QR Code, la cual permite generar un código ingresando el contenido que se desea representar.
2. La caja ya con el código QR se almacena hasta que un cliente desee comprar alguna.
3. Al vender la caja se asesora al cliente en el registro de la caja dentro de la aplicación.

Con el objetivo de almacenar y manipular información de manera eficiente se creará una base de datos llamada "CajaSeguridadInteligente" en donde existirán 5 tablas:

- La tabla Usuario almacenará la información de cada una de las cuentas creadas por cada usuario.
- La tabla Caja tendrá la información de cada caja que esté asociada con los servicios de la aplicación.

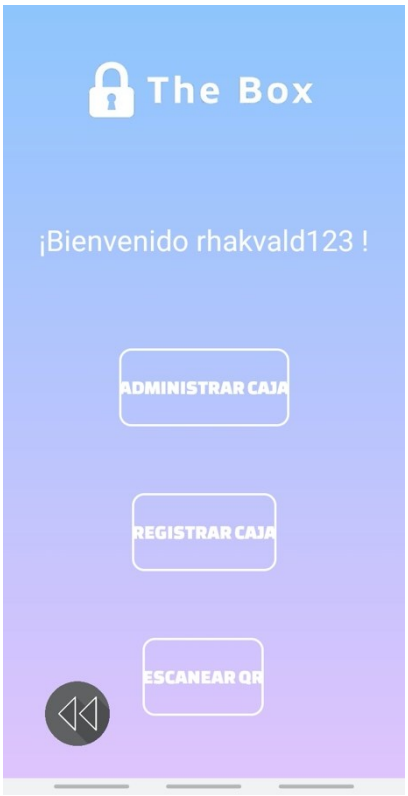
Según el proceso de adquisición explicado previamente, los campos de idUbicacion y contraseñaCaja deberán ser Null mientras las cajas aún no tengan dueño. El campo infoQR sí será inicializado desde la creación de la caja ya que este código es generado "desde fábrica".

- La tabla Permiso es la conexión entre las cajas y los usuarios, en ella se enlistarán quiénes tienen acceso a cuáles cajas.
- La tabla Ubicación proporciona datos adicionales sobre dónde se encuentra la caja.
- La tabla Acceso guarda todos los accesos de todas las cajas, en casa que algún usuario desee hacer seguimiento de quién logró tener acceso a la caja -no se consideran los casos de accesos denegados ya que se ha cumplido la función de resguardar el contenido y no representa algún peligro-.

En lo que respecta netamente a la aplicación móvil desarrollada en Android Studio, el usuario al ingresar a la misma se encontrará con una pantalla principal que cuente con un logo o un formato representativo de la aplicación. En dicha pantalla se apreciará lo siguiente:

En caso de que el usuario no se haya registrado previamente deberá escoger la opción “REGISTRARSE”, en donde ingresará datos básicos para identificarlo, tales como: sus nombres y apellidos, su correo electrónico, una contraseña relacionada con su cuenta en la aplicación y un nombre de usuario que debe ser único; se verificará en la base de datos que el nombre de usuario que se plantea ingresar no exista antes de registrar la información del nuevo usuario.

Una vez el usuario haya creado una cuenta e ingresado sus credenciales en la pantalla de inicio, se presentará la siguiente pantalla que contará con 3 opciones:



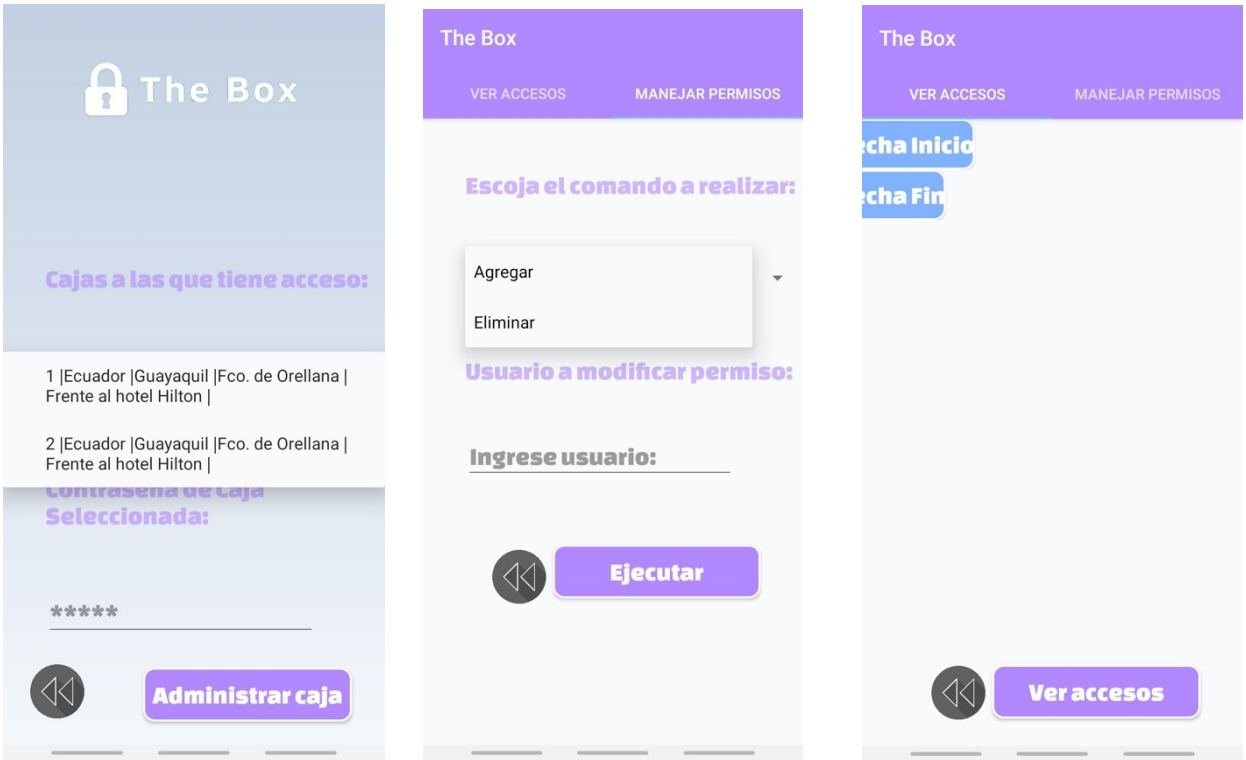
En la opción “REGISTRAR CAJA” se podrá registrar una nueva caja, abriéndose una nueva pantalla donde se registrarán los siguientes datos referentes a la caja: una contraseña para administrar la caja en momentos posteriores y el país, ciudad, calle y más detalles sobre dónde se encuentra la caja. En aquellos datos no existirán restricciones al momento de registrarlos. Por otro lado, al usuario que acabó de “crear” la caja se le concederá permiso para abrir y administrar la caja. Es también en este momento donde se escanea por primera vez el código QR presente en la caja, aquello permitirá identificar sobre cuál caja se busca ingresar información en la base de datos. Una vez que toda la información necesaria ha sido ingresada se procede a enviar la información e insertarla en la base de datos.

En la opción “ESCANEAR QR” se procesarán los permisos para acceder a la cámara del dispositivo con la cual se procede a realizar a lectura del código QR. La transformación de código QR a cadena de caracteres -contenido representado- se llevará a cabo con la librería ZXingScanner, dicho contenido del QR se comparará con el que está registrado en la base de datos para esa caja en específico y también se verifica si el usuario que intenta ingresar tiene permiso para abrir esa caja. Existirán dos tipos de mensajes

de alerta: uno indicará que el código QR leído no es compatible con ningún registro dentro de la base de datos y otra alerta que indicará al usuario que no tiene permiso para desbloquear la caja. En caso de que el código QR esté correcto se procede a enviar una señal hacia el módulo IoT que se encuentra en esa caja en específico con el fin de retirar el seguro y poder abrirla -descripción detallada más adelante-. Aquel acceso quedará registrado en la base de datos en la tabla Acceso junto con el id de la Caja.

En la opción “ADMINISTRAR CAJA” se permitirá otorgar permisos a otros usuarios para acceder a la caja. Sin embargo, esto tendrá ciertas restricciones: se deberá ingresar la contraseña de la caja -registrada en base de datos- y además no se permitirá el ingreso de permisos si el usuario que intenta dar permiso tampoco tiene permiso para ingresar a la caja -considerando que por alguna razón sabe la contraseña de la caja y quiera asignarse permiso indebidamente a sí mismo o a alguien más-. También se puede eliminar los permisos para acceder la caja. Tamien se tendrá la opcion de ver accesos en un rango de fechas. Para que así haya un control

La pantalla de esta opción será basada en la siguiente ilustración:



Al llegar a aquella pantalla ya se ha verificado si el usuario en cuestión tiene permisos para acceder a la caja, ahora se verificará si cuenta con la contraseña para otorgar permisos. En caso de que la contraseña ingresada sea incorrecta no se ingresa el nuevo permiso en la base de datos, se muestra una alerta y se regresa a la pantalla con opciones.

5. ¿Qué van a construir para resolver el problema?

A nivel de Hardware se va a implementar el circuito mostrado en la imagen número 3, se tendrá una caja la cual va a tener una electro-cerradura y esta se accionará con un solenoide, este solenoide a su vez será conectado y desconectado con un Relay que cerrará el circuito entre el positivo de la fuente de alimentación y el negativo sólo cuando se active el relay.

Para activar el Relay se conectará este al emisor de un transistor tip 41C, mientras que la base va a ir al controlador del sistema, y el colector irá al colector y una fuente de 5V.

El transistor va a estar conectado al controlador (ESP32), el cual es el centro de todo el sistema a este también estarán conectadas salidas a leds que indiquen el estado de la caja fuerte y entradas a botones capacitivos en la caja para que el usuario pueda decidir cuándo abrir la caja y cuando cerrarla (los permisos para abrir la caja serán otorgados a través de la aplicación durante un tiempo, pero el usuario la abre o cierra con estos botones en este tiempo).

6. Recursos de software que se utilizarán.

Para este proyecto los recursos de software a utilizar serian los siguientes.

- MySQL: Con este sistema de base de datos relacional se permitirá que se guarden las tablas las cuales tendrán información que necesitaremos para esta propuesta de solución. Este sistema permite una lectura rápida de información por lo que lo hace ideal para este tipo de aplicación.

- phpMyAdmin: Con esta herramienta de software nos permitirá administrar de una manera más fácil MySQL con una interfaz gráfica. Además, se tendrá aun la posibilidad de administrar MySQL de forma tradicional. Con esta herramienta se podrá crear las tablas necesarias como los son “Usuario”, “Caja”, “Permiso”, “Ubicación” y “Acceso”. Esta creación de tablas se vera facilitada por la herramienta de phpMyAdmin, haciendo que haya un manejo más ágil y eficiente.

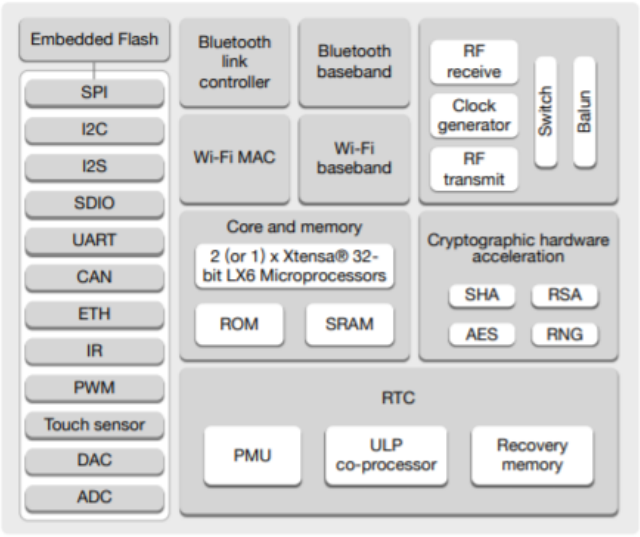
- Android Studio: Con esta herramienta podremos crear la aplicación móvil que requiere el proyecto. Con Android Studio mediante la programación de java se podrá crear las diferentes opciones que tiene el usuario al momento de ingresar a la aplicación como la ventana de registro de usuario, la lectura de código QR y la ventana que permite la administración de alguna caja. También se configurará para que la aplicación establezca una conexión con la base de datos.

7. Recursos de Hardware para utilizar:

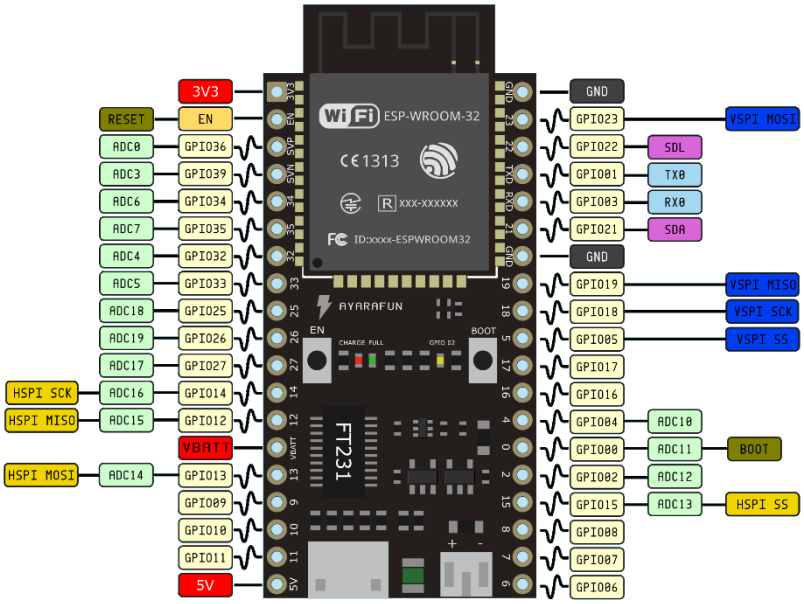
- Microcontrolador: El microcontrolador será el componente de hardware que se encargara de integrar el software con los demás componentes de hardware del resto de la caja, este será el que reciba señales del servidor y ejecute las acciones necesarias para abrir la caja fuerte, es decir mandar señales de control a los demás componentes del circuito.

El microcontrolador para usar en este proyecto será un ESP32, siendo más específicos un ESP32 DEVKITV1, este es un paquete en combo que viene con 2.4 GHz wifi, Bluetooth y RF, especialmente diseñado para aplicaciones IoT de bajo consumo energético. El ESP32 cuenta con distintos modos destinados al ahorra de energía en el cual el microcontrolador solo es despertado al recibir señales específicas [2].

Este Chip contiene un procesador de dos núcleos de 32 bits (lo cual facilita la programación de tareas que requieran concurrencia), que opera hasta a 600 MIPS, posee 448KB de ROM, 520 KB SRAM, 16 KB de SRAM en RTC. Además, soporta los siguientes protocolos de comunicación: SPI, UART, I^2C, I^2S, entre otros [2], también mencionados en la Ilustración 5.



El microcontrolador además posee 32 pines de entrada y salida de propósito general, 10 de estos soportan entradas táctiles capacitivas, además soporta PWM de 16 bits en 16 canales que pueden hacer uso de un oscilador de 8 MHz. Se pueden observar sus entradas en detalle en la Ilustración a continuacion



Como se puede observar el esp32 es un microcontrolador de altas capacidades que exceden los requerimientos de este proyecto, sin embargo, es una plataforma de desarrollo de alta disponibilidad en el mercado y especialmente desarrollada para el prototipado de proyectos por lo cual será usado en este proyecto, en caso de desear implementar el proyecto a gran escala se podría usar un microcontrolador más específico para los requerimientos.

- Relay: Se va a usar un relay para controlar el seguro de la caja fuerte ya que el controlador no puede suministrar la corriente necesaria para activar este.

- Transistor: Un transistor será usado en conjunto con el microcontrolador y el Relay para cambiar el estado del relay y así poder alimentar el inductor que posee internamente sin causar problemas de picos de voltaje en el controlador. El transistor para usar será un TIP41C ya que cualquier transistor de propósito general sirve para la implementación del proyecto y este tiene una alta disponibilidad y bajo precio.

- LEDs: Se usarán LEDs para indicar el estado de la caja, es decir si esta se encuentra cerrada, se usara un LED rojo conectado en serie con una resistencia de 220 Ohm al controlador, y en caso de que la caja se encuentre lista para ser abierta se activará un LED verde conectado de igual forma al controlador para indicar que está lista.

8. Explicación paso a paso de la implementación del proyecto

Lo primero en realizar del proyecto fue la elaboración de la base de datos y también las creaciones de las tablas que alojarían las diferentes tipos de información. Y una vez creada las tablas se procedió a relacionarlas entre ellas mediante las claves primarias y secundarias.

Luego de que ya tuvimos la base de datos creada se creo la estructura principal de la aplicación es decir los Textview fijos y botones de las diferentes actividades. Una vez hecho esto se procedió a darle función a los botones para que se pueda navegar entre las actividades. Luego se le dio estilo es decir se le pusieron colores y la fuente de las letras. Era necesario que se escogieras colores que armonizaran y se vieran bien juntos, ya que la apariencia de una aplicación es esencial para su presentación.

Luego de la estructura y de la apariencia se procedió a realizar la funcionalidad. En este paso las funcionalidades de la aplicación se hicieron casi de manera simultanea, pero la primera que se puede mencionar fue la importación y la implementación de la librerías indicadas para la lectura y generación de código QR, también se realizaron las pruebas pertinentes de lectura y generación aleatoria de códigos QR. Luego la siguiente parte esencial era la conexión de la base de datos que permitía en la pagina principal poder iniciar sesión y poder registrarse en caso de no ser tener una cuenta, también la conexión permitió que se verificaran los permisos de acceso a la de los usuarios que intentaban abrir una caja al momento de leer el código. Luego también se realizaron acciones que permitían registrar una caja y asignarle un código QR, también permitían administrar cajas que incluía la capacidad de que el administrador de la caja pueda dar y quitar permisos a otros usuarios y también de poder ver la cantidad e información de los accesos en un intervalo de tiempo.

Luego de realizar las pruebas con la conexión de base de datos entramos al tema de verificación de datos ingresados por el usuario, es decir verificar que los datos ingresados por el usuario sean coherentes con lo solicitado y en caso de no hacerlo mostrar las alertas correspondientes en este caso en forma de TOAST permitiendo la aplicación no se caiga. También se realizaron otras alertas pertinentes como por ejemplo mostrar si el acceso era negado o exitoso al momento que un usuario haya hecho una lectura de código o mostrar que la eliminación y la proporción de permisos haya sido exitosa o fallida.

Una vez realizada todas las verificaciones correspondientes y pruebas con la mayor de cantidad de escenarios posibles se procedió a la configuración y del modulo ESP32 la cual era la encargada de recibir la información del celular y dar la apertura en caso de que el usuario que haya leído el código QR tenga los permisos necesarios para abrirla.

Posterior a la configuración del modulo se dio paso a la construcción del prototipo de la caja de seguridad, del circuito eléctrico y los componentes mecánicos que permitían que la puerta pueda mantearse cerrada hasta que se reciba la señal de apertura y se proceda a liberar el seguro para que el usuario pueda acceder al contenido de la caja.

9. Diagramas

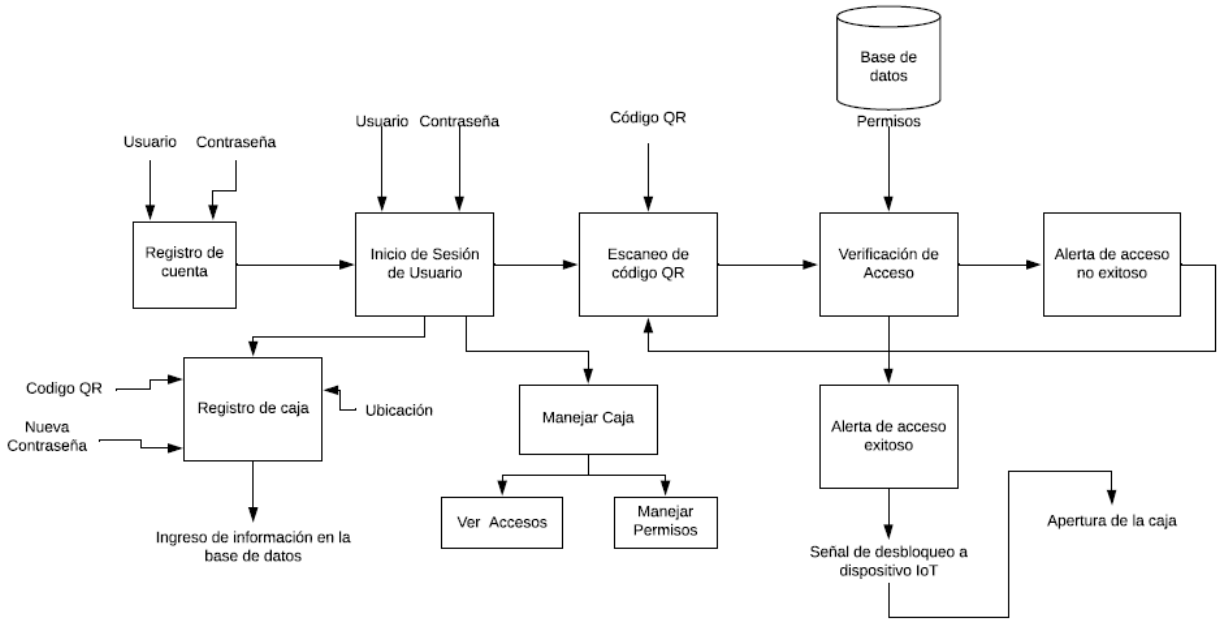


Ilustración 1. Diagrama de flujo para el proceso del proyecto.

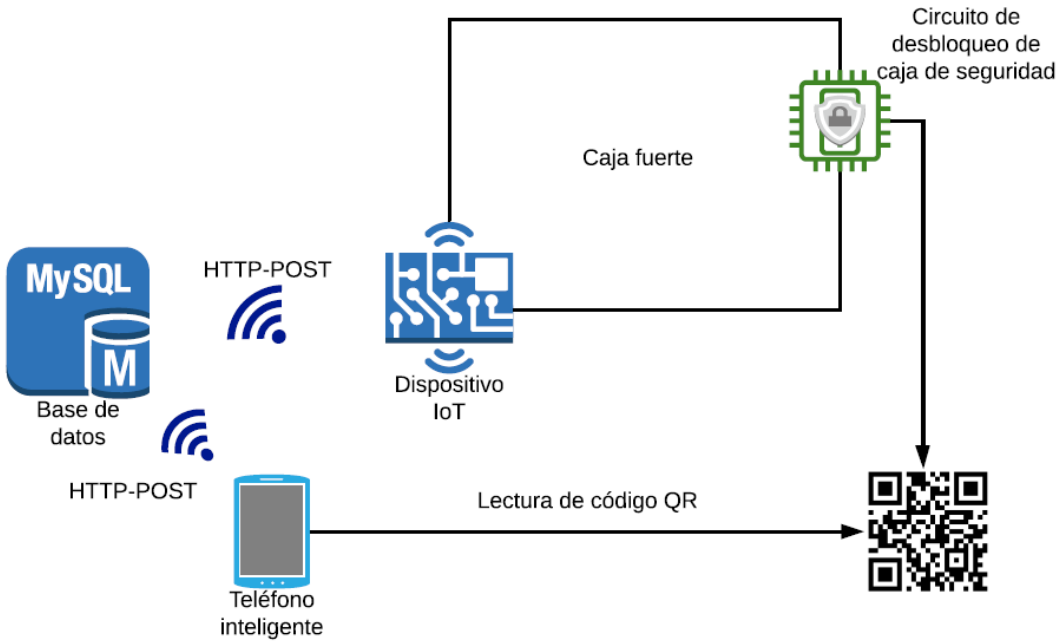


Ilustración 2. Diagrama de diseño del proyecto.

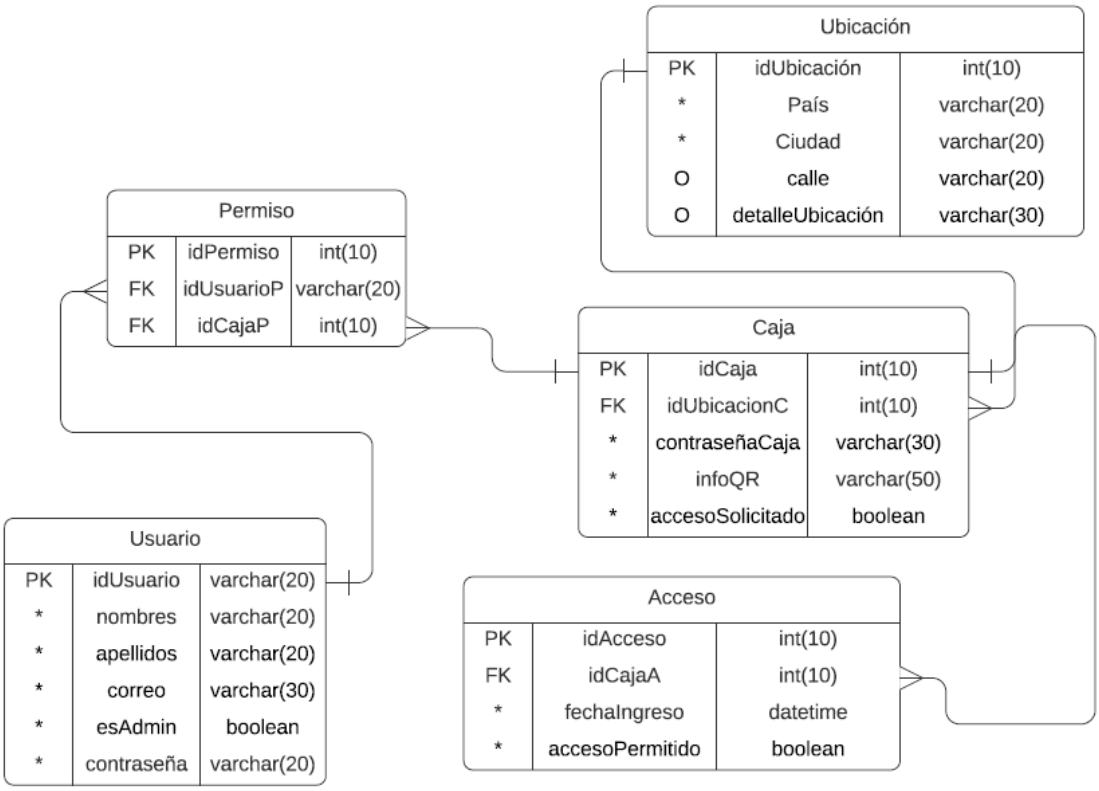


Ilustración 3. Modelo entidad-relación del proyecto.

Acotaciones adicionales sobre el modelo entidad-relación:

- Suponiendo una aplicación a gran escala del prototipo a presentar se ha creado la entidad Ubicación, que de cierta manera permite conocer a detalle dónde se encuentra cada una de las cajas que se han registrado.
- La entidad Acceso permite registrar cada uno de los accesos realizados para cada caja presente. Se registra el día y la hora en la que ocurrió dicho acceso.
- El atributo contraseñaCaja permitirá la administración de las configuraciones de la caja -permisos- dentro de la aplicación.
- infoQR almacena una cadena de caracteres, este deberá ser el contenido representado por el código QR de la caja a la que se desea ingresar.

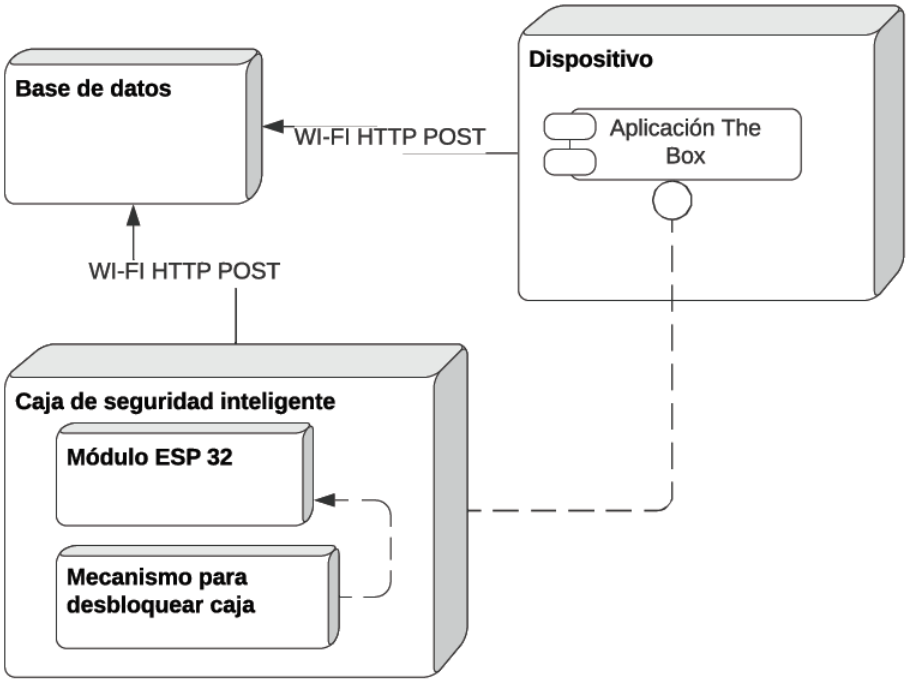


Ilustración 4. Diagrama de despliegue.

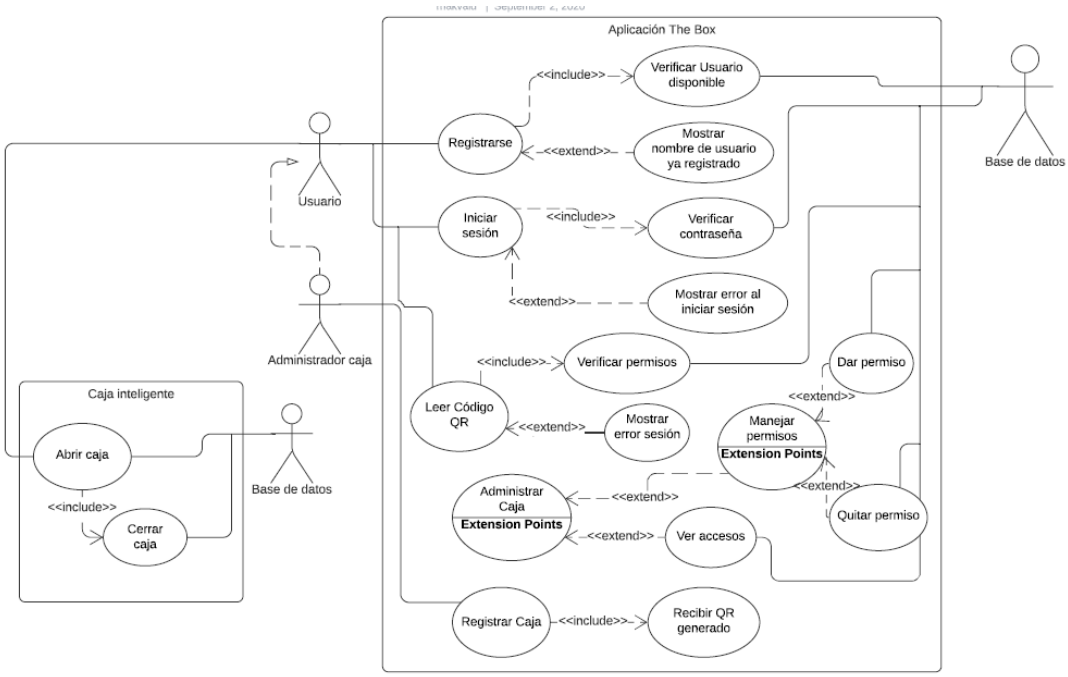


Ilustración 5. Diagrama de casos.

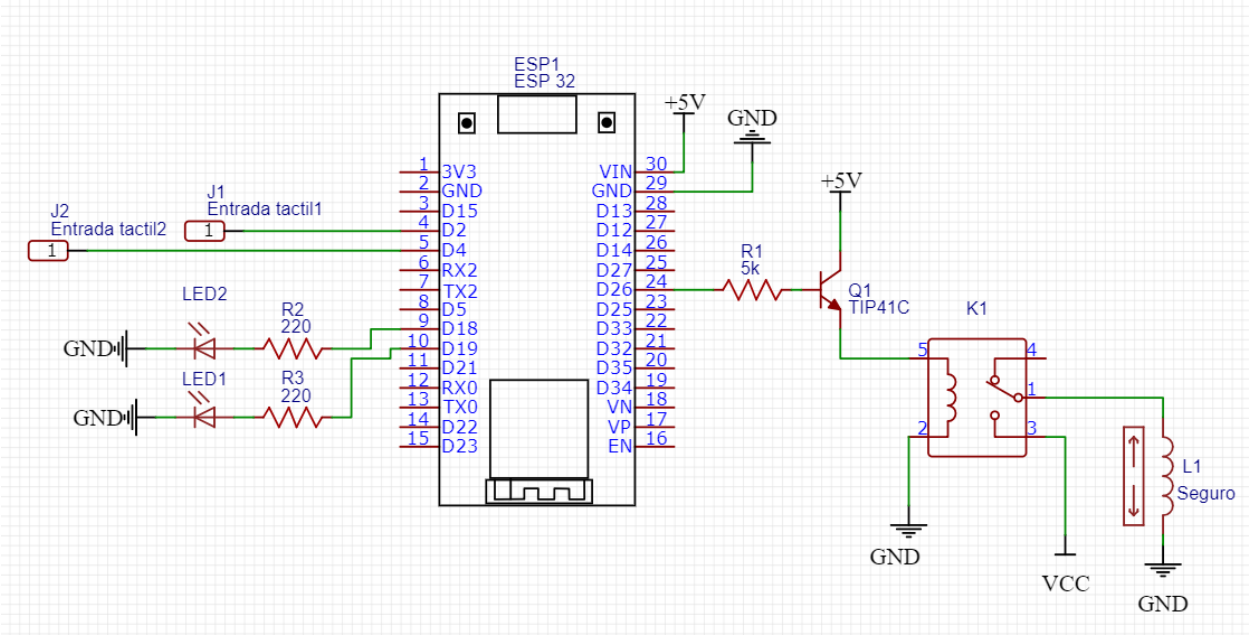


Ilustración 4. Diagrama del circuito.

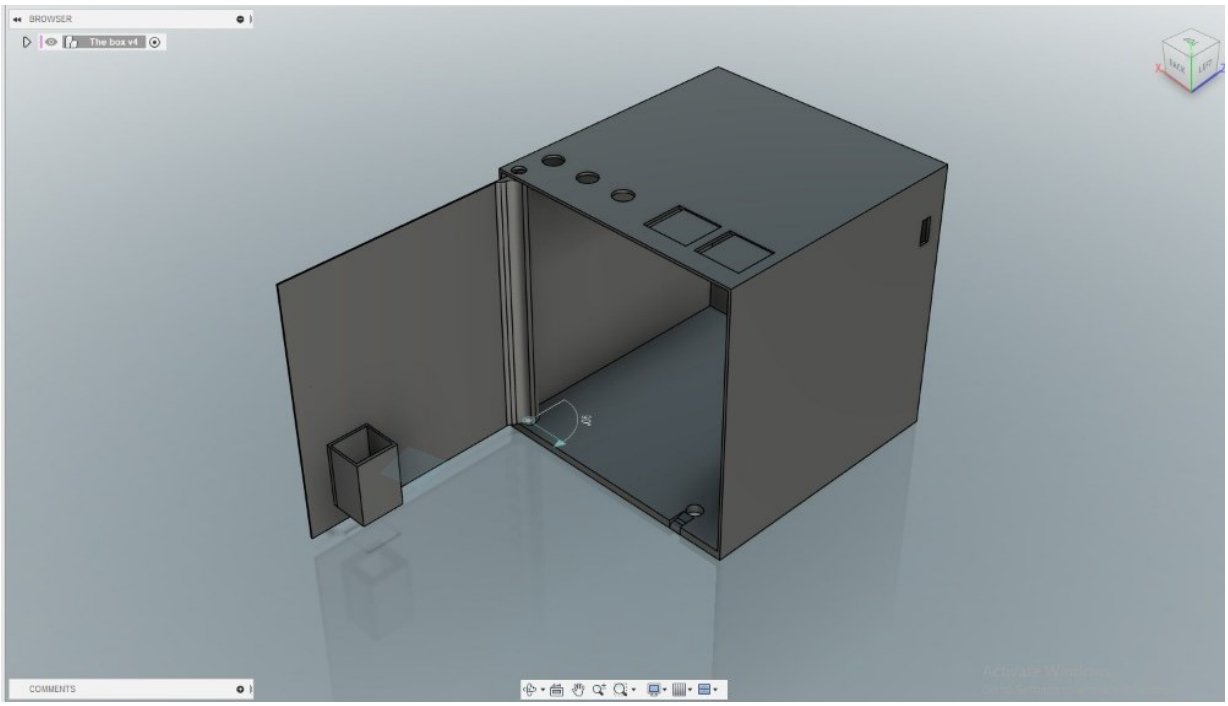


Ilustración 5. Modelo 3D de la caja elaborada.

10.Descripción de base de datos

Tabla Usuario:

- **IdUsuario:** Primary Key de la tabla, de tipo varchar con longitud máxima de 20 caracteres. Corresponde al nombre de usuario que identifica de manera única a cada una de las personas que hacen uso de la aplicación.
- **Nombres:** Campo obligatorio de tipo varchar con longitud máxima de 20 caracteres. Corresponde a los nombres del usuario registrado.
- **Apellidos:** Campo obligatorio de tipo varchar con longitud máxima de 20 caracteres. Corresponde a los apellidos del usuario registrado.
- **Correo:** Campo obligatorio de tipo varchar con longitud máxima de 30 caracteres. Corresponde al correo del usuario registrado.
- **EsAdmin:** Campo obligatorio de tipo boolean. Indica si el usuario tiene permiso para registrar cajas.
- **Contraseña:** Campo obligatorio de tipo varchar con longitud máxima de 20 caracteres. Corresponde a la contraseña que el usuario deberá ingresar para iniciar sesión bajo la identificación de un determinado nombre de usuario.

Tabla Caja:

- **IdCaja:** Primary Key de la tabla, de tipo int con máximo de 10 dígitos. Corresponde al identificador de la caja dentro de la base de datos.
- **IdUbicacionC:** Foreign Key de tipo int con máximo de 10 dígitos. Corresponde al identificador de la ubicación, hace referencia al identificador que se encuentra en la tabla Ubicación como PK.
- **ContraseñaCaja:** Campo obligatorio de tipo varchar con longitud máxima de 30 caracteres. Corresponde a la contraseña que los usuarios con permiso para administrar la caja deben ingresar para validar el acceso.
- **InfoQR:** Campo obligatorio de tipo varchar con longitud máxima de 50 caracteres. Corresponde a la cadena de caracteres que el código QR representa y que ha sido asignado a la presente caja.
- **AccesoSolicitado:** Campo obligatorio de tipo boolean. Indica si ha ocurrido verificación de credenciales de un usuario que ha leído el código QR de la caja y se está solicitando un acceso -ya validado- a la caja.

Tabla Acceso:

- **IdAcceso:** Primary Key de la tabla, de tipo int con máximo de 10 dígitos. Corresponde al identificador del acceso dentro de la base de datos.
- **IdCajaA:** Foreign Key asociada con la PK de la tabla Caja, de tipo int con máximo de 10 dígitos. Corresponde al identificador de una caja dentro de la base de datos.
- **FechaIngreso:** Campo obligatorio de tipo datetime, que corresponde a la fecha y hora en el cual ocurrió el acceso, ya sea que este haya sido exitoso o no.
- **AccesoPermitido:** Campo obligatorio de tipo boolean, indica si el acceso fue exitoso o no -si usuario que intentó acceder contaba con los permisos necesarios-.

Tabla Permiso:

- **IdPermiso:** Primary Key de la tabla, de tipo int con máximo de 10 dígitos. Corresponde al identificador del permiso dentro de la base de datos.
- **IdUsuarioP:** Foreign Key asociada con la PK de la tabla Usuario, de tipo varchar con longitud máxima de 20 caracteres. Corresponde al identificador de un usuario dentro de la base de datos. Este usuario tendrá permiso para acceder a la caja que se encuentra en el mismo registro (fila).
- **IdCajaP:** Foreign Key asociada con la PK de la tabla Caja, de tipo int con máximo de 10 dígitos. Corresponde al identificador de una caja dentro de la base de datos.

Tabla Ubicación:

- **IdUbicacion:** Primary Key de la tabla, de tipo int con máximo de 10 dígitos. Corresponde al identificador de la ubicación dentro de la base de datos.
- **País:** Campo obligatorio de tipo varchar con longitud máxima de 20 caracteres. Corresponde al país en donde se encuentra la caja referenciada en la tabla Caja por el IdUbicacion.
- **Ciudad:** Campo obligatorio de tipo varchar con longitud máxima de 20 caracteres. Corresponde a la ciudad en donde se encuentra la caja referenciada en la tabla Caja por el IdUbicacion.
- **Calle:** Campo opcional de tipo varchar con longitud máxima de 20 caracteres. Correspondea la calle en donde se encuentra la caja referenciada en la tabla Caja por el IdUbicacion.
- **DetalleUbicación:** Campo opcional de tipo varchar con longitud máxima de 50 caracteres. Corresponde a una descripción adicional -sector, manzana,villa- sobre dónde se encuentra la caja referenciada en la tabla Caja por el IdUbicacion.

Descripción de código PHP con comandos SQL

administrarCaja.php:

Permite verificar si una caja tiene una determinada contraseña con el fin de otorgar permiso al usuario para que pueda administrar a caja. No se verifica que el usuario tenga permisos de acceso a la caja pues aquello se comprueba en otro archivo.

```
<?php
$db_user="id14296398_grupo10"; #se establecen las credenciales para establecer conexión con la base de datos
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";


$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);#se establece la conexión
if (!$con) {
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
}
$ingreso = $_POST["SQL"]; #ingresa el string mediante POST
list($idCaja,$contraCaja) = explode(",",$ingreso); #se separa el string que ingresa mediante el carácter de la coma y se asigna cada parte a las variables puestas entre paréntesis en el list
$query = "SELECT id FROM Caja WHERE id = ".$idCaja." AND contraseñaCaja = '".$contraCaja."';#este query permite seleccionar la columna de ids que cumplan con que el id sea igual al de la caja que se pretende entrar y la contraseña de la caja sea igual a la ingresada, con ello si no existe ningún registro significa que la contraseña no es la correcta ya que el id de la caja sí existe pues en la aplicación solo se podían escoger cajas que ya existían y que el usuario ya contaba con permiso para abrir
$result = mysqli_query($con, $query);
$filas = mysqli_num_rows($result);
$infoAcceso = "denegado";
if($filas > 0){ #en caso que exista registro, es decir, que se haya ingresado todo correctamente, se manda un mensaje de concedido; caso contrario se niega el acceso
   $infoAcceso="concedido";
}
echo $infoAcceso;
#se Cierra la consulta y la conexión
```

```
mysqli_free_result($result);
mysqli_close($con);
?>
```

login.php:

Permite verificar que las credenciales de nombre de usuario y contraseña son correctas, con lo que se niega o se permite el inicio de sesión.

```
<?php #se establecen las credenciales de la base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";


$con = mysqli_connect($db_server,$db_user,$db_password,$db_name); #conexión a la base de datos
if (!$con) { #muestra mensajes de error en caso que falle la conexión
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
}
$ingreso = $_POST["SQL"]; #se recibe un string por post mediante el código SQL
list($usuarioIngresado,$contraIngresada) = explode(",",$ingreso); #se separa el string recibido y se asigna cada separación a las variables establecidas en el list
$query = "SELECT * FROM Usuario WHERE id = '".$usuarioIngresado."' AND contraseña = '".$contraIngresada."'"; #este query selecciona todas las columnas de la tabla usuario en donde el nombre de usuario sea igual al ingresado y la contraseña coincida también con la que se recibe, de esta manera se buscará en la tabla por el registro con el nombre de usuario y se validará si la contraseña es igual a la que está en el registro
#se genera el resultado de la consulta
$result = mysqli_query($con, $query);
$columnas = mysqli_num_fields($result); #columnas de la tabla usuario
$filas = mysqli_num_rows($result); #filas de la consulta
$infoAcceso = "denegado,,,,"; #información de retorno predeterminada, cambiará si se dio la validación
#en caso de que la consulta tenga alguna fila se verifica que hubo un correcto ingreso de credenciales y se debe modificar el mensaje a mandar
if($filas > 0){
   #empieza modificación de mensaje de retorno
```

```
    $infoAcceso="concedido,";
    while($row = mysqli_fetch_array($result)){
    #se añaden al mensaje de retorno todas las columnas menos la de la contraseña de usuario, última columna en la tabla Usuario
        for($i=0;$i<$columnas;$i++){
            if($i!=($columnas-1)){
            $infoAcceso .= $row[$i].",";
            }
        }
    }
}
#se manda mensaje, se cierra consulta y conexión
echo $infoAcceso;
mysqli_free_result($result);
mysqli_close($con);
?>
```

manejarAccesoCaja.php:

Provee la funcionalidad de la ventana donde el usuario puede quitar o dar permisos a otro usuario en particular. Se verifica que el usuario exista, comprueba que exista el registro antes de eliminarlo y previene el ingreso de registros duplicados para un mismo usuario respecto a una misma caja.

```
<?php  #se establecen credenciales de la base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
#conexión a la base de datos, se comprueba que se haya establecido conexión
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
if (!$con) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
```

```
    exit;
}
#se recibe un string con código SQL mediante POST
$ingreso = $_POST["SQL"];
list($usuario,$comando,$idCaja)=explode(",",$ingreso); #se divide string ingresada en las variables enunciadas dentro del list
$mensaje = ""; #mensaje en blanco, se llenará según la situación que ocurra con los registros
#se prepara la conexión y se establece el query a ejecutar, en donde en usuarioExistente y registroPermisoExistente se guarda un valor booleano ya que para cada uno se ejecuta un subquery encerrrado por el EXISTS, que devuelve true si hay registros debidos al subquery; usuarioExistente guarda true si existe un usuario con el nombre de usuario ingresado, mientras que registroPermisoExistente guarda true si ya existe un registro de un permiso para un usuario de la caja en análisis
if ($stmt = $con->prepare('SELECT
        EXISTS (SELECT * FROM Usuario WHERE id = ?) AS usuarioExistente,
        EXISTS (SELECT * FROM Permiso WHERE idUsuarioP = ? AND idCajaP = ?) AS registroPermisoExistente')) {
#los signos de interrogación se reemplazan por lo valores establecidos en bind_param, siendo los dos primeros strings (s) y el último un int(i)
    $stmt->bind_param('ssi', $usuario, $usuario, $idCaja);
    $stmt->execute(); #se ejecuta el query
    $stmt->bind_result($usuarioExistente, $registroPermisoExistente); #se asignan las variables en el query a las variables establecidas en el bind_result
    $stmt->fetch();
    $stmt->close(); #se Cierra la consulta ya que las variables ya guardaron el valor
    if (!$usuarioExistente) { #si usuario no existe se manda un mensaje a la aplicación
        $mensaje="Usuario no se encuentra registrado.";
    }
    else {
        if($comando=="Agregar"){ #ingresa si el comando elegido es agregar un permiso, caso contrario es eliminar -solo hay 2 opciones-
            if($registroPermisoExistente){ #si ya existe un registro de permiso para ese usuario en esa caja se manda un mensaje, caso contrario se procede a ingresar el registro
                $mensaje="Permiso a aquel usuario ya existe.";
            }
            else{
                $queryIngreso = "INSERT INTO `Permiso` (`id`,`idUsuarioP`,`idCajaP`) VALUES (NULL,'".$usuario."','".$idCaja."')"; #se ingresa el permiso, primer argumento es NULL ya que id está como auto incrementado
```

```
            mysqli_query($con,$queryIngreso); #se ejecuta el query
            $mensaje="Ingreso de permiso exitoso.";
        }
    }
    else{ #si entra acá se busca eliminar un permiso
        if($registroPermisoExistente){ #si existe un registro se lo elimina, caso contrario se manda un mensaje
            $queryIngreso = "DELETE FROM Permiso WHERE idUsuarioP = '".$usuario."' AND idCajaP = '".$idCaja."'"; #con este query se borra el permiso donde el usuario y la caja sean iguales a los ingresados
            mysqli_query($con,$queryIngreso); #se ejecuta query
            $mensaje="Eliminación exitosa de permiso.";
        }
        else{
            $mensaje="Fallo al eliminar, usuario no ha tenido permiso.";
        }
    }
  }
}
#se manda mensaje y se cierra la conexión
echo $mensaje;
mysqli_close($con);
?>
```

registro.php:

Recibe la información ingresada por el usuario cuando está registrando una nueva cuenta; posteriormente si no existe un usuario con el mismo nombre de usuario procede a ingresar la información como un nuevo registro en la tabla Usuario.

```
<?php  #se establecen credenciales de la base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
#se da la conexión con la base de datos y se verifica si ocurrió exitosamente
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
if (!$con) {
```

```
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
    exit;
}
$ingreso = $_POST["SQL"]; #se recibe un string con código SQL mediante POST
list($nombreUsuario,$nombres,$apellidos,$correo,$contraseña) = explode(",",$ingreso); #información ingresada se separa por comas y se asigna en el mismo orden a las variables dentro del list
$query = "SELECT id FROM Usuario WHERE id = '".$nombreUsuario."'"; #se selecciona la columna id de la tabla Usuario donde el usuario sea igual al ingresado, si el query devuelve registros significa que ya existía el nombre de usuario
$result = mysqli_query($con,$query);
$filas = mysqli_num_rows($result);
mysqli_free_result($result);
$mensaje = "denegado";
#solo en caso de que no existan registros del query establecido anteriormente se cambia el mensaje y se ejecuta otro query para ingresar un registro a la tabla Usuario
if($filas == 0){
    $mensaje = "ingresado";
    $queryIngreso = "INSERT INTO `Usuario` (`id`,`nombres`,`apellidos`,`correo`,`contraseña`) VALUES ('".$nombreUsuario."','".$nombres."','".$apellidos."','".$correo."','".$contraseña."')";
    mysqli_query($con,$queryIngreso);
}
#se manda mensaje -denegado en caso de que ya existía el nombre de usuario- y se cierra la conexión
echo $mensaje;
mysqli_close($con);    ?>
```

registroCaja1.php:

Permite registrar una caja, asociarla con una ubicación que existía previamente o generar una nueva, así como conceder permisos de acceso a quien haya creado la caja.

```
<?php #credenciales de base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
# verificación de conexión
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
```

```
if (!$con) {
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
}

$ingreso = $_POST["SQL"];
#se recibe por post un string separado por comas y con el explode se divide y se asigna cada parte a las variables en el list
list($pais,$ciudad,$calle,$detalleUbicacion,$contrasena,$infoQR,$userName) = explode(",",$ingreso);
$query = "SELECT id FROM Ubicacion WHERE pais = '".$pais."' AND ciudad = '".$ciudad."' AND calle = '".$calle."' AND detalleUbicacion = '".$detalleUbicacion."' "; #se busca si ya existía una ubicación igual
$result = mysqli_query($con,$query);
$filas = mysqli_num_rows($result);
mysqli_free_result($result);
$mensaje = "denegado";
# en caso de que la ubicación no exista se hace un insert a la tabla ubicación ingresado un nuevo registro
if($filas == 0){
   $mensaje = "ingresado";
   $queryIngresoUbicacion = "INSERT INTO `Ubicacion` (`id`, `pais`, `ciudad`, `calle`, `detalleUbicacion`) VALUES (NULL,'".$pais."','".$ciudad."','".$calle."','".$detalleUbicacion."')";
   mysqli_query($con,$queryIngresoUbicacion);
}
echo $mensaje;

# se obtiene el id de la ubicación independientemente si fue creada recientemente o antes
$queryConsulta= "SELECT * FROM Ubicacion WHERE pais = '".$pais."' AND ciudad = '".$ciudad."' AND calle = '".$calle."' AND detalleUbicacion = '".$detalleUbicacion."' ";
$result2 = mysqli_query($con,$queryConsulta);
#se ingresa un nuevo registro de caja con los valores ingresados por post y el id de la ubicación; id es null ya que está como auto incremental
$row = mysqli_fetch_array($result2);
$queryIngresoCaja = "INSERT INTO `Caja` (`id`, `idUbicacionC`, `contraseñaCaja`, `infoQR`, `accesoSolicitado`) VALUES (NULL,'".$row["id"]."','".$contrasena."','".$infoQR."','0')";
mysqli_query($con,$queryIngresoCaja);
```

```
mysqli_free_result($result2);
#se procede a obtener el id de la caja para luego en el otro query conceder permiso al usuario que registró la caja
$queryConsulta2= "SELECT * FROM Caja WHERE infoQR = '".$infoQR."' ";
$result3 = mysqli_query($con,$queryConsulta2);
$row2 = mysqli_fetch_array($result3);
$queryPermiso = "INSERT INTO `Permiso` (`id`, `idUsuarioP`, `idCajaP`) VALUES (NULL,'".$userName."','".$row2["id"]."')";
mysqli_query($con,$queryPermiso);
mysqli_free_result($result3);
mysqli_close($con);
?>
```

tablaAccesos.php:

Obtiene todos los registros de accesos relacionados con una caja en particular en un período de tiempo específico. Esta información es dirigida a la aplicación con el fin de mostrarla al usuario.

```
<?php #credenciales de la base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
#conexión a base de datos
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
if (!$con) {
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
}
$ingreso = $_POST["SQL"];
#se recibe información por post y se la separa por comas
list($idCaja,$fechaInicio,$fechaFin) = explode(",",$ingreso);
$query = "Select fechaIngreso,accesoPermitido from Acceso where (fechaIngreso between '".$fechaInicio."' and '".$fechaFin."') and (idCajaA = '".$idCaja."')"; #se seleccionan las columnas fechaIngreso y accesoPermitido de la tabla Acceso que cumplan con lo siguiente: que el id de la caja sea igual al ingresado y que la fecha de ingreso se encuentre dentro del rango especificado entre fechaInicio y
```

fechaFin, un mal ingreso de dicho rango, es decir, fechaFin es antes que fechaInicio, devolverá el mensaje "Vacío"

```
$result = mysqli_query($con, $query);
$filas = mysqli_num_rows($result);
$columnas = mysqli_num_fields($result);
$resultado = "";

if($filas > 0){ #ingresa si hay resultado en la consulta
   $i=0;
#recorre las columnas obteniendo el nombre de las mismas y situando un encabezado para la tabla
   while ($column = mysqli_fetch_field($result)) {
      $resultado .= $column->name . ",";
   }
   $resultado .= "\r\n";
#recorre las filas obtenidas en la consulta y va agregándolas al string acumulador, separando cada fila por un salto de línea y cada celda dentro de la fila por una coma
   while($row = mysqli_fetch_array($result)){
      for($i=0;$i<$columnas;$i++){
         $resultado .= $row[$i].",";
      }
      $resultado .= "\r\n";
   }
}
else{
   echo "Tabla vacía\r\n";
}

echo $resultado; #se manda la cadena con la información para la tabla, se cierra la consulta y conexión
mysqli_free_result($result);
mysqli_close($con);
?>
```

verificacionAdmin.php:

```
<?php  #credenciales de base de datos
```

```
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
#conexión a base de datos
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
if (!$con) {
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
}
$nombre = $_POST["SQL"]; #se recibe el nombre de usuario mediante post
$query = "SELECT esAdmin FROM Usuario WHERE id = '".$nombre."'"; #se consulta si el usuario ingresado tiene permisos de administrador
$result = mysqli_query($con,$query);
$permiso = "denegado";
while($row = mysqli_fetch_array($result)){
#si tiene un 1 en la columna esAdmin se le concede permiso
   if($row['esAdmin'] == 1){
      $permiso = "concedido";
   }
}
echo $permiso; #se manda el permiso, se cierra la consulta y conexión
mysqli_free_result($result);
mysqli_close($con);
?>
```

verificacionQR.php:

Permite verificar que el códigoQR corresponde a alguna caja dentro de la base de datos y si el usuario que intenta ingresar tiene permiso para abrir la caja.

```
<?php #credenciales de base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
```

```
#verificación de conexión a base de datos
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
if (!$con) {
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
}
#se recibe por post información mandada desde la aplicación y se separa según las comas que existan
$ingreso = $_POST["SQL"];
list($usuario,$infoQR) = explode(",",$ingreso);
$query = "SELECT idCajaP FROM Permiso WHERE idUsuarioP = '".$usuario."';#se seleccionan los ids de las cajas a las que el usuario ingresado tiene permiso de acceder
$result = mysqli_query($con, $query);
$filas = mysqli_num_rows($result);
$infoAcceso = "denegado";

if($filas > 0){

   while($row = mysqli_fetch_array($result)){
      $query1 = "SELECT infoQR FROM Caja WHERE id = '".$row[0]."';
      $result1 = mysqli_query($con, $query1);
      while($row1 = mysqli_fetch_array($result1)){
#dentro de la anterior iteración va obteniendo varios registros, pero para entrar al siguiente if el registro debe contar con el infoQR igual al que se ha leído desde la aplicación
         if($row1[0] == $infoQR ){
            $infoAcceso="acceso";
#se ingresa un registro de acceso permitido a la caja
            $queryIngreso = "INSERT INTO `Acceso` (`id`, `idCajaA`, `fechaIngreso`, `accesoPermitido`) VALUES (NULL,'".$row[0]."',CURRENT_TIMESTAMP, '1')";
            mysqli_query($con,$queryIngreso);
#se actualiza el campo accesoSolicitado a 1 -pasó los filtros con éxito- en la tabla Caja de la caja en cuestión
            $queryUpdate="UPDATE Caja SET accesoSolicitado = 1 WHERE id = '".$row[0]."';
            mysqli_query($con,$queryUpdate);
         }
      }
   }
```

```
}
#si no se cambió el estado a concedido se ingresa un registro en la tabla Accesos -antes obteniendo el id de la caja leída- el cual tendrá en acceso permitido un 0, pues no se permitió el acceso
if ($infoAcceso == "denegado"){
   $query2 = "SELECT id FROM Caja WHERE infoQR = '".$infoQR."'";
   $result2 = mysqli_query($con, $query2);
    while($row2 = mysqli_fetch_array($result2)){
#current timestamp permite poner la fecha y hora actual
        $queryIngreso1 = "INSERT INTO `Acceso` (`id`, `idCajaA`, `fechaIngreso`, `accesoPermitido`) VALUES (NULL,'".$row2[0]."',CURRENT_TIMESTAMP, '0')";
            mysqli_query($con,$queryIngreso1);
    }
}
#se manda la condición de acceso y se cierra la consulta y conexión
echo $infoAcceso;
mysqli_free_result($result);
mysqli_close($con);
?>
```

verPermisoCajasUsuario.php:

Permite obtener información sobre todas las cajas a las que tiene acceso el usuario con sesión iniciada. Devuelve información de cada caja a la aplicación para que se despliegue en un Spinner.

```
<?php #credenciales de base de datos
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";
#conexión a base de datos
$con = mysqli_connect($db_server,$db_user,$db_password,$db_name);
if (!$con) {
   echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
   echo "error de depuración: " . mysqli_connect_errno() . PHP_EOL;
   exit;
```

```
}
$nombreUsuario = $_POST["SQL"]; #se recibe el nombre de usuario con sesión iniciada en la aplicación
$query = "Select Caja.id,Ubicacion.pais,Ubicacion.ciudad,Ubicacion.calle,Ubicacion.detalleUbicacion from Permiso join Caja on Permiso.idCajaP = Caja.id join Ubicacion on Caja.idUbicacionC = Ubicacion.id where Permiso.idUsuarioP = '".$nombreUsuario."' "; # este query permite partir desde la tabla Permiso, en donde se pone la condición de solo considerar los registros donde se encuentra al usuario en cuestión, y se van haciendo joins mediante las FK y PK entre las tablas Caja y Ubicación, con el fin de obtener el id de la caja y datos sobre su ubicación para presentarlos a usuario y decida correctamente cuál caja desea administrar
$result = mysqli_query($con, $query);
$filas = mysqli_num_rows($result);
$columnas = mysqli_num_fields($result);
$resultado = "";
#si existen registros -más de una fila- se procede a acumular en la variable resultado la información de cada caja, situando un salto de línea para diferenciar los registros
if($filas > 0){
   $i=0;
   while($row = mysqli_fetch_array($result)){
      for($i=0;$i<$columnas;$i++){
         $resultado .= $row[$i].",";
      }
      $resultado .= "\r\n";
   }
}
#si no hay registros se manda el siguiente mensaje
else{
   echo "Vacío.\r\n";
}
#se manda el contenido organizado de la consulta a la aplicación, se cierra la consulta y la conexión
echo $resultado;
mysqli_free_result($result);
mysqli_close($con);

?>
```

Post-esp-data.php:

```
<?php


$api_key_value = "tPmAT5Ab3j7F9";//api key para validar que se tengan permisos para cambiar el estado

include_once('esp-database.php');

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $api_key = test_input($_POST["api_key"]);//se lee el api key dado por el esp32

    if($api_key == $api_key_value) {//se confirma que el apikey sea el correcto antes de realizar cambios
        $id=test_input($_POST["id"]);
        $state=test_input($_POST["state"]);//se seleccionan los valores dados por el modulo

        updateState($id,$state);}//se llama el metodo para cambiar el estado de la caja para esp-database.php
    else {
        echo "Wrong API Key provided.";
    }

}
else {
    echo "No data posted with HTTP POST.";
}

function test_input($data) {//formato de los datos ingresados
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

Esp-outputs-action.php:

```
<?php
    include_once('esp-database.php');

    $action = $id = $name = $gpio = $state = "";//para metros pasados por el módulo al servidor

    if ($_SERVER["REQUEST_METHOD"] == "GET") {//se obtienen los valores de estado
        $action = test_input($_GET["action"]);
        if ($action == "outputs_state") {
            $id = test_input($_GET["id"]);
            $result = getState($id);//se usa el metodo definido en esp-database.php

            echo $result;//se retorna el estado al modulo



        }
        else {
            echo "Invalid HTTP request.";
        }
    }
    function test_input($data) {//revision del formato de entrada de datos
        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
        return $data;
    }


?>
```

Esp-database.php:

```
<?php
$db_user="id14296398_grupo10";
$db_password="H<o?X>OvP4r&ME{g";
$db_name="id14296398_proyecto";
$db_server="localhost";//credenciales de la base de datos
   function updateState($id, $state) {//funcion para actualizar el estado de la caja en la base de datos
      global $db_server, $db_user, $db_password, $db_name;
      $conn = new mysqli($db_server, $db_user,$db_password,$db_name);
      if ($conn->connect_error) {
         die("Connection failed: " . $conn->connect_error);//mensaje dde una conexion sin exito
      }
      $sql = "UPDATE Caja SET accesoSolicitado='" . $state . "' WHERE id='". $id .  "'";//set del estado
de la caja con el id especificado
     if ($conn->query($sql) === TRUE) {
         return "Caja state updated successfully";
      }
      else {
         return "Error: " . $sql . "<br>" . $conn->error;
      }
      $conn->close();
   }
   function getState($id) {//metodo para la lectura del estado de la caja con estado id
      global $db_server, $db_user, $db_password, $db_name;
      $conn = new mysqli($db_server, $db_user,$db_password,$db_name);
      if ($conn->connect_error) {
         die("Connection failed: " . $conn->connect_error);
      }
      $sql = "SELECT accesoSolicitado FROM Caja WHERE id='" . $id . "'";//se lee el estado de la caja
con identificacion id
      if ($result = $conn->query($sql)) {
         if ($row = $result->fetch_assoc()) {
```

```
            return $row['accesoSolicitado'];//se retorna la columna de acceso solicitado de la caja
                }
        }
        else {//en caso de que no se pueda leer el estado se retorna un 0 como estado cerrado
            return 0;
        }
        $conn->close();
    }

?>
```

11. Descripción de código fuente

AndroidManifest.xml:

Contiene diversos permisos: uso de la cámara con el fin de realizar la lectura del código QR, acceso a Internet desde el dispositivo y para guardar la imagen del código QR cuando se registra la caja -al momento de la venta-. Además, se enumeran las diversas actividades de la aplicación -en donde una Splash Activity compuesta por el logo de la marca es la actividad inicial-. También se establecen configuraciones de la actividad donde se captura el código QR con la cámara -p.ej. orientación de la actividad-.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.thebox">
    //PERMISOS
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.hardware.camera" />
    <uses-permission android:name="android.hardware.camera.autofocus" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
            //ACTIVIDADES EN LA APLICACIÓN
        <activity
            android:name=".ManejoCaja"
            android:label="@string/title_activity_manejo_caja"
```

```
        android:theme="@style/AppTheme.NoActionBar"></activity>
    <activity android:name=".VerificacionCaja" />
    <activity android:name=".registroCaja" />
    <activity android:name=".OpcionesIniciales" />
    <activity android:name=".LectorQR" />
    <activity android:name=".Registro" />
    <activity
        android:name=".SplashInicial"
        android:screenOrientation="portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".Inicio" />
    <activity
        android:name=".CaptureAct"
        android:screenOrientation="fullSensor"
        android:stateNotNeeded="true"
        android:windowSoftInputMode="stateAlwaysHidden" />

    <meta-data
        android:name="preloaded_fonts"
        android:resource="@array/preloaded_fonts" />
  </application>

</manifest>
```

AccesosFragment.java:

Fragment que contiene el código que permite configurar ciertos aspectos del diseño (color de fondo de botones p.ej.) y provee la funcionalidad a la opción “Ver Accesos” en “Administrar Caja”. Sobrescribe el método onClick de todos los botones presentes, permitiendo regresar a la actividad anterior, establecer la fecha de inicio y fin de búsqueda de accesos -al dar clic se muestra un cuadro de diálogo para seleccionar una fecha-, y también ejecutar la búsqueda de accesos que ocurrieron para la caja que se está

administrando en el período de tiempo establecido y mostrando una tabla con la fecha y hora del acceso y si este fue permitido o no. Al seleccionar una fecha existe un TextView asociado a cada botón que muestra al usuario la fecha seleccionada. Luego, cuando se escoge ejecutar la búsqueda se verifica que el período de tiempo establecido sea válido, es decir, que la fecha de fin sea antes de la fecha de inicio -sí se permite establecer la misma fecha en ambos, en dicho caso se muestran los accesos que ocurren únicamente en ese día-. Finalmente, se conecta al servicio de hosting utilizado en donde se encuentra el archivo PHP que regresa a la aplicación los resultados de la consulta de los accesos.

```
package com.example.thebox;

import android.annotation.SuppressLint;
import android.app.DatePickerDialog;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;

import androidx.fragment.app.DialogFragment;
import androidx.fragment.app.Fragment;

import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
```

```
import java.util.Arrays;
import java.util.Date;
import java.util.concurrent.ExecutionException;


public class AccesosFragment extends Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
    private Button ButFechaInicio,ButFechaFin,ButEjecutar;
    private TextView etFin,etInicio;
    private TableLayout table;
    private String url_llenarTabla = "https://lab6-chiquito.000webhostapp.com/tablaAccesos.php";


    private String mParam1;
    private String mParam2;

    public AccesosFragment() {
        // Required empty public constructor
    }

    public static AccesosFragment newInstance(String param1, String param2) {
        AccesosFragment fragment = new AccesosFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
```

```
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }
    //FUNCIÓN EJECUTADA CUANDO EL FRAGMENT SE VA A MOSTRAR POR PANTALLA Y SUSTITUYE AL FRAGMENT GENERAL CONTENIDO EN LA ACTIVIDAD PRINCIPAL
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_accesos, container, false);//SE ASOCIA EL FRAGMENT CON SU LAYOUT
        table = view.findViewById(R.id.tablaConsulta);
        etFin = view.findViewById(R.id.textViewFin);
        etInicio = view.findViewById(R.id.textViewInicio);
        ImageButton regresar = view.findViewById(R.id.volverVerificarCaja1);
        //ONCLICK SOBRESCRITO PARA QUE TERMINE LA ACTIVIDAD PRESENTE DONDE ESTÁN LOS FRAGMENTS
        regresar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getActivity().finish();
            }
        });
        //CONFIGURO ASPECTO DEL BOTON, COLOR DE FONDO
        ButFechaInicio = view.findViewById(R.id.fechaInicio);
        GradientDrawable drawableInicio = (GradientDrawable) ButFechaInicio.getBackground();
        drawableInicio.setColor(Color.parseColor("#82B1FF"));


        ButFechaInicio.setOnClickListener(new View.OnClickListener() {
        //BOTON AL SER PRESIONADO GENERA EL FRAGMENT PARA SELECCIONAR UNA FECHA Y AL ESCOGERLA SE MUESTRA EN UN TEXT VIEW
            @Override
            public void onClick(View v) {
                DatePickerFragment newFragment = DatePickerFragment.newInstance(new DatePickerDialog.OnDateSetListener() {
                    @Override
```

```
            public void onDateSet(DatePicker datePicker, int year, int month, int day) {
                final String selectedDate = year+"-"+(month+1) + "-" + day;
                etInicio.setText(selectedDate);
            }
        });
        newFragment.show(getActivity().getSupportFragmentManager(), "datePicker");


    }
});
//ASPECTO DEL BOTON
ButFechaFin = view.findViewById(R.id.fechaFin);
GradientDrawable drawableFin = (GradientDrawable) ButFechaFin.getBackground();
drawableFin.setColor(Color.parseColor("#82B1FF"));
//MUESTRA FRAGMENT PARA SELECCIONAR UNA FECHA Y LA SITUA EN EL TEXT VIEW
ButFechaFin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        DatePickerFragment newFragment = DatePickerFragment.newInstance(new DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker datePicker, int year, int month, int day) {
                final String selectedDate = year+"-"+(month+1) + "-" + day;
                etFin.setText(selectedDate);
            }
        });
        newFragment.show(getActivity().getSupportFragmentManager(), "datePicker");


    }
});


ButEjecutar = view.findViewById(R.id.llenarTabla);
GradientDrawable drawableLlenar = (GradientDrawable) ButEjecutar.getBackground();
drawableLlenar.setColor(Color.parseColor("#B388FF"));
//EMPIEZA LA BUSQUEDA DE REGISTROS DE ACCESOS
```

```
        ButEjecutar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String[] resultado = null;
                String fechaInicio = etInicio.getText().toString();
                String fechaFin = etFin.getText().toString();
//SE VERIFICA QUE FECHA FINAL DEL RANGO SEA LA MISMA O DESPUES DE LA FECHA INICIAL, CASO CONTRARIO SE CORTA LA BUSQUEDA
                if(fechasIncorrectas(fechaInicio,fechaFin)){
                    Toast.makeText(getActivity(), "Ingrese fechas correctamente.", Toast.LENGTH_SHORT).show();
                    return;
                }
//SE VERIFICA QUE SE HAYAN ESTABLECIDO FECHAS, SINO SE PROCEDE A CORTAR LA BUSQUEDA
                if(fechaInicio.isEmpty()||fechaFin.isEmpty()){
                    Toast.makeText(getActivity(), "Ingrese fechas.", Toast.LENGTH_SHORT).show();
                    return;
                }
//EMPIEZA LA COMUNICACION CON EL SERVIDOR
                try {
//SE MANDAN LOS DATOS NECESARIOS AL SERVIDOR, PARA LA FECHA INICIO SE PONE DESDE QUE EMPIEZA ESE DIA MIENTRAS QUE PARA LA FECHA FINAL SE PONE LA HORA CUANDO TERMINA EL DÍA
                    String[] datos = new String[]{
                            "llenarTablaAccesos",
                            url_llenarTabla,
                            UsuarioActual.getIdCajaActualenRevision(),
                            fechaInicio +" 00:00:00",
                            fechaFin+" 23:59:59"
                    };
                    AsyncQuery async = new AsyncQuery();
                    resultado = async.execute(datos).get();
                    System.out.println(resultado[0]);
                    if(resultado[0].equals("Tabla vacía")){
                        Toast.makeText(getActivity(), "Sin resultados.", Toast.LENGTH_SHORT).show();
```

```
            }
            String resultadoTabla = resultado[0].trim();
            String[] myData= resultadoTabla.split("\\n");
            ArrayList<ArrayList<String>> infoQuery = new ArrayList<>();
//SE GENERA UN ARREGLO 2D PARA FACILITAR GENERAR LA TABLA
            for(int i=0;i<myData.length;i++){
               String[] filaStrings = myData[i].split(",");
               ArrayList<String> fila = new ArrayList<String>(Arrays.asList(filaStrings));
               infoQuery.add(fila);
            }
//LIMPIEZA DE TABLA
            table.removeAllViews();
//SE VA GENERANDO CADA FILA POR CADA ARREGLO PRESENTE EN INFOQUERY, CADA CELDA ES UN TEXTVIEW; SE VA AÑADIENDO DE FILA EN FILA A LA TABLA
            for(int x=0;x<infoQuery.size();x++){
               TableRow tbrow = new TableRow(getActivity());
               for(int y=0;y<infoQuery.get(x).size();y++){
                  TextView tv = new TextView(getActivity());
                  tv.setText(infoQuery.get(x).get(y));
                  if(x==0){
                     tv.setTextColor(Color.WHITE);
                     tv.setBackgroundColor(Color.rgb(157,186,213));
                  }else{
                     tv.setTextColor(Color.GRAY);
                     tv.setBackgroundColor(Color.rgb(250,243,221));
                  }
                  tv.setTextSize(12);
                  tv.setGravity(Gravity.CENTER);
                  tv.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT,
TableRow.LayoutParams.WRAP_CONTENT));
                  tv.setPadding(15,0,15,0);
                  tbrow.addView(tv);
               }
```

```
                table.addView(tbrow, new TableLayout.LayoutParams(new
TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT,
TableLayout.LayoutParams.WRAP_CONTENT)));
                }
            } catch (ExecutionException e) {
                e.printStackTrace();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    });
    return view;
  }
//RECIBE LOS STRINGS DE LOS TEXTVIEW CON LAS FECHAS, SE ESTABLECE UN FORMATO
PARA QUE SE PUEDAN COMPARARLOS COMO FECHA, DEVULVE BOOLEANO VERDADERO
SI FECHAFIN ES DESPUES O ES EL MISMO DIA QUE FECHA INICIO
  public boolean fechasIncorrectas(String fechaInicio,String fechaFin){
    @SuppressLint("SimpleDateFormat") SimpleDateFormat sdformat = new
SimpleDateFormat("MM-dd-yyyy");
    boolean FinMenor = true;
    try {
      Date d1 = sdformat.parse(fechaInicio);
      Date d2 = sdformat.parse(fechaFin);
      if(d2.compareTo(d1)>=0){
        FinMenor = false;
      }
    } catch (ParseException e) {
      e.printStackTrace();
    }
    return FinMenor;

  }
}
```

AsyncQuery.java:

Se procesan todas las conexiones desde la aplicación hacia el servidor para realizar consultas en la base de datos. Existen diversos tipos de consultas y diversos parámetros a enviar para cada una. Devuelven los resultados -en caso de que sea necesario- como cadena de caracteres luego de decodificarlo.

```
package com.example.thebox;

import android.os.AsyncTask;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

//METODOS INPUT/OUTPUT INFORMACION SE EXPLICAN AL ULTIMO

public class AsyncQuery extends AsyncTask<String[],Void,String[]> {
   @Override
   protected String[] doInBackground(String[]... datos) {
      String[] totalResultadoSQL = null; //la posición 0 de esta lista contiene la cadena de resultados
      String type = datos[0][0];//toda consulta tendrá un tipo y una url, por lo que se inicializan estas variables antes de comparar cuál tipo de solicitud es
      String login_url = datos[0][1];
      if(type.equals("login")){
      try {
//si es login se manda el usuario y contraseña ingresada, devuelve “denegado” si hay inconsistencias con lo que se encuentra en la base de datos
         String usuarioIngresado = datos[0][2];
         String contraIngresada = datos[0][3];
         String SQL = usuarioIngresado+","+contraIngresada;
```

```
            HttpURLConnection conexion = outputInformacion(SQL,login_url);
            totalResultadoSQL = inputInformacion(conexion);
        } catch (MalformedURLException e ) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
        else if(type.equals("registrar")){
        try {
//al registrar un usuario se manda a la base los siguientes datos sobre el mismo para que se ingresen en la tabla Usuario
            String usuario = datos[0][2];
            String nombres = datos[0][3];
            String apellidos = datos[0][4];
            String correo = datos[0][5];
            String contrasena = datos[0][6];
            String SQL = usuario+","+nombres + "," + apellidos + "," + correo +","+ contrasena;
            HttpURLConnection conexion = outputInformacion(SQL,login_url);
            totalResultadoSQL = inputInformacion(conexion);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        else if(type.equals("verificarAdmin")){
            try {
//manda el nombre de Usuario y devuelve denegado si este no es admin de la caja, es decir, no puede dar permisos o ver accesos
                String nombreUsuario = datos[0][2];
                HttpURLConnection conexion = outputInformacion(nombreUsuario,login_url);
                totalResultadoSQL = inputInformacion(conexion);
            } catch (MalformedURLException e ) {
                e.printStackTrace();
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else if(type.equals("verificarQR")){

        try {
//verifica que el contenido del QR leído coincida con el registrado en la base para cierta caja, manda el usuario para ver si tiene permiso de acceder a esa caja
            String usuario = datos[0][2];
            String infoQR = datos[0][3];

            String SQL = usuario+","+infoQR ;
            HttpURLConnection conexion = outputInformacion(SQL,login_url);
            totalResultadoSQL = inputInformacion(conexion);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else if(type.equals("manejarCaja")){
        try {
//manda el usuario y devuelve todas las cajas a las que este puede acceder, con el fin de mostrarlas en un spinner, que el usuario las seleccione y pueda administrala
            String usuario = datos[0][2];
            String SQL = usuario;
            HttpURLConnection conexion = outputInformacion(SQL,login_url);
            totalResultadoSQL = inputInformacion(conexion);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    else if(type.equals("registrarCaja")){
        try {
```

//se registra una nueva caja en la base de datos, se manda la información necesaria; se verifica que la ubicación no se encuentre repetida aunque el usuario puede cambiar el detalle de la ubicación y ya generaría un nuevo registro

```
            String paisCaja = datos[0][2];
            String ciudadCaja = datos[0][3];
            String calleCaja = datos[0][4];
            String detalleCaja = datos[0][5];
            String contrasenaCaja = datos[0][6];
            String infoQR = datos [0][7];
            String SQL = paisCaja+","+ciudadCaja + "," + calleCaja + "," + detalleCaja +","+ contrasenaCaja + "," + infoQR;
            HttpURLConnection conexion = outputInformacion(SQL,login_url);
            totalResultadoSQL = inputInformacion(conexion);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else if(type.equals("verificarCaja")){
        try {
```

//se verifica que se hayan ingresado las credenciales correctas para ver los accesos y dar permisos de una caja en particular; el usuario escoge la caja de las opciones mostradas en spinner; únicamente puede intentar acceder a cajas a las que tiene permiso pero si no sabe la contraseña de la caja no podrá

```
            String idCaja = datos[0][2];
            String contra = datos[0][3];
            String SQL = idCaja+","+contra;
            HttpURLConnection conexion = outputInformacion(SQL,login_url);
            totalResultadoSQL = inputInformacion(conexion);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        }
        else if(type.equals("manipularAccesos")){
            try {
//manda el comando que se quiere llevar a cabo: agregar o eliminar un permiso del  usuario respecto a una caja determinada
                String usuario = datos[0][2];
                String comando = datos[0][3];
                String idCaja = datos[0][4];
                String SQL = usuario+","+comando+","+idCaja;
                HttpURLConnection conexion = outputInformacion(SQL,login_url);
                totalResultadoSQL = inputInformacion(conexion);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        else if(type.equals("llenarTablaAccesos")){
            try {
//manda el idCaja con las fechas y devuelve los registros de accesos correspondientes a dicha caja y en el rango especificado
                String idCaja = datos[0][2];
                String fechaInicio = datos[0][3];
                String fechaFin = datos[0][4];
                String SQL = idCaja+","+fechaInicio+","+fechaFin;
                HttpURLConnection conexion = outputInformacion(SQL,login_url);
                totalResultadoSQL = inputInformacion(conexion);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return totalResultadoSQL;
    }
```

```
   public HttpURLConnection outputInformacion(String SQL,String login_url) throws IOException{
//se instancia una url y se establece conexion a internet, los datos se mandarin por el protocol POST
      URL url = new URL(login_url);
      HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
      httpURLConnection.setRequestMethod("POST");
      httpURLConnection.setDoOutput(true);
      httpURLConnection.setDoInput(true);
//se codifica la informacion a mandar como UTF-8 y se la manda a la url especificada
      OutputStream outputStream = httpURLConnection.getOutputStream();
      BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
      String tablaPost = URLEncoder.encode("SQL", "UTF-8") + "=" + URLEncoder.encode(SQL, "UTF-8");
      bufferedWriter.write(tablaPost);
      bufferedWriter.flush();
      bufferedWriter.close();
      outputStream.close();
//se devuelve la conexion porque siempre luego del output existe una función de input y ahí se cierra la conexión una vez ya se deja de utilizar la conexión con el servidor
return httpURLConnection;
   }
   public String[] inputInformacion(HttpURLConnection httpURLConnection) throws IOException{
//se receptan los valores devueltos por el PHP y las consultas a la base de datos, decodificándolas y acumulándolas en un cadena de caracteres
      InputStream iStr = httpURLConnection.getInputStream();
      BufferedReader bR = new BufferedReader(new InputStreamReader(iStr,"UTF-8"));
      String resultado="";
      String line="";

      while((line = bR.readLine()) != null){
         resultado += line + System.getProperty("line.separator") ;
      }
      bR.close();
      iStr.close();
//finalmente ya se cierra la conexión a internet
      httpURLConnection.disconnect();
```

```
        return  new String[]{resultado};
    }
}
```

CaptureAct.java:

Esta actividad se encuentra vacía, no obstante, es un requerimiento de la librería empleada para la lectura del código QR ya que extiende de la clase padre CaptureActivity y le concede funcionalidades para llevar a cabo una correcta lectura del código. En el Código de "LectorQR" se especificará dónde se utiliza y para qué.

```
package com.example.thebox;

import com.journeyapps.barcodescanner.CaptureActivity;

public class CaptureAct extends CaptureActivity {
}
```

DatePickerFragment.java:

Fragment que se superpone en la actividad al momento de elegir las fechas de inicio y fin para que el usuario visualice los accesos en ese rango de fechas. Muestra un calendario que le facilita al usuario seleccionar.

```
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class DatePickerFragment extends DialogFragment{
    private DatePickerDialog.OnDateSetListener listener;

    public static DatePickerFragment newInstance(DatePickerDialog.OnDateSetListener listener) {
//se recibe el listener como argumento ya que este será modificado al instanciar este fragment con el fin de obtener la fecha y enviarla a la base de datos, además de mostrarla en un TextView
        DatePickerFragment fragment = new DatePickerFragment();
        fragment.setListener(listener);
        return fragment;
    }
```

```
    public void setListener(DatePickerDialog.OnDateSetListener listener) {
        this.listener = listener;
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
//se instancia un calendario, el cual se mostrará para seleccionar la fecha; se obtienen los valores numéricos del año, mes y día
        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);

        return new DatePickerDialog(getActivity(), listener, year, month, day);
    }
}
```

Inicio.java:

Actividad donde el usuario puede iniciar sesión ingresando sus credenciales o crear un nuevo usuario.

```
package com.example.thebox;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.ExecutionException;

public class Inicio extends AppCompatActivity {
```

```
    private EditText etUser,etContra;
    private View botRegistrar,botIngresar;
    private String login = "https://lab6-chiquito.000webhostapp.com/login.php";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
//se configura el diseño de los botones, sus bordes circulares en las esquinas y se establece su color de background
//también se instancian los edit text donde se ingresan las credenciales del usuario
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etUser = findViewById(R.id.usuario);
        etContra = findViewById(R.id.contrasena);
        botRegistrar = findViewById(R.id.botonRegistrar);
        botIngresar = findViewById(R.id.botonIngresar);
        GradientDrawable drawableRegistrar = (GradientDrawable) botRegistrar.getBackground();
        drawableRegistrar.setColor(Color.parseColor("#82B1FF"));
        GradientDrawable drawableIngresar = (GradientDrawable) botIngresar.getBackground();
        drawableIngresar.setColor(Color.parseColor("#B388FF"));
    }
    public void ingresar(View v){
        String[] resultado;
        String nomUs = etUser.getText().toString();
        String contra = etContra.getText().toString();
//se verifica que no esté vacío ningún edit text, si lo está no se permite la verificación de las credenciales en la base de datos
        if(nomUs.isEmpty() || contra.isEmpty()){
            Toast.makeText(Inicio.this, "Ingrese información solicitada.", Toast.LENGTH_SHORT).show();
            return;
        }
        try {
            String[] datos = new String[]{
                    "login",
                    login,
                    nomUs,
                    contra
```

```
        };
```
//se establece la conexión y se devuelve denegado en caso que las credenciales no sean las apropiadas, se muestra mensaje para indicar al usuario del error; también se devuelve información del usuario -menos contraseña- para usos posteriores en la aplicación
```
        AsyncQuery async = new AsyncQuery();
        resultado = async.execute(datos).get();
        String[] myData= resultado[0].trim().split("\\n")[0].split(",");
        ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
        if (infoImp.get(0).equals("denegado")){
          Toast.makeText(Inicio.this, "Usuario no registrado o datos ingresados incorrectos.", Toast.LENGTH_SHORT).show();
          return;
        }
```
//se instancia un objeto UsuarioActual con el fin de contar con la información del usuario para otras funciones; de igual manera se establece a esa instancia como un atributo estático de la clase para facilitar su acceso
```
        UsuarioActual actUsuario = new UsuarioActual(infoImp.get(1), infoImp.get(2),infoImp.get(3),infoImp.get(4));
        UsuarioActual.setUser(actUsuario);
```
//si no ha ocurrido ningún error el usuario pasa a la actividad con las opciones de registrar caja, leer QR o administrar la caja
```
        Intent intent=new Intent(this,OpcionesIniciales.class);

        startActivity(intent);
      } catch (ExecutionException e) {
        e.printStackTrace();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }



  }
```
//Usuario pasa hacia la actividad Registro para registrar un nuevo usuario
```
  public void irRegistro(View v){
    Intent intent = new Intent(this,Registro.class);
    startActivity(intent);
```

```
    }
}
```

LectorQR.java:

Al escoger la opción “Escanear QR” se abre esta actividad que permite hacer uso de la cámara para leer el código QR.

```
package com.example.thebox;



import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

public class LectorQR extends AppCompatActivity implements View.OnClickListener {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lector_q_r);



    }
    @Override
```

```
    public void onClick(View view) {
        scanCode();
    }

    public void  scanCode(){
        IntentIntegrator intergrator = new IntentIntegrator(this);
//el lector necesita de argumento a la actividad CaptureAct para realizar la lectura y realizar las configuraciones
        intergrator.setCaptureActivity(CaptureAct.class);
//permite al dispositivo girar la pantalla y en cualquier caso hace la lectura
        intergrator.setOrientationLocked(false);
//permite lectura de cualquier código
        intergrator.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
//mensaje mientras escanea
        intergrator.setPrompt("Scanning Code");
//inicia el escaneo
        intergrator.initiateScan();
    }
//una vez que ya se obtiene un Código escaneado
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data){
        IntentResult result = IntentIntegrator.parseActivityResult(requestCode,resultCode,data);
        if (result != null){
            if (result.getContents() != null){
//se instancia cuadro de dialogo
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
//se establece el mensaje del cuadro de dialogo según sean los resultados de la lectura
                builder.setMessage(result.getContents());
                builder.setTitle("Resultado de Escaneado");
//se configura los botones del cuadro, para escanear de nuevo o terminar
                builder.setPositiveButton("Escanear de Nuevo", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int which) {
                        scanCode();
                    }
```

```
            }).setNegativeButton("Terminado", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int which) {
                    finish();
                }
            });
 //dialogo mostrado segun se haya dado el permiso o no
            AlertDialog dialog = builder.create();
            dialog.show();

        }
        else {
            Toast.makeText(this,"No hay resultado", Toast.LENGTH_LONG).show();
        }
      }else{
        super.onActivityResult(requestCode,resultCode,data);
      }
   }


}
```

ManejoCaja.java

Actividad que funciona como contenedora de los fragments “Accesos Fragment” y “Permisos Fragment”.

```
package com.example.thebox;

import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.tabs.TabLayout;

import androidx.viewpager.widget.ViewPager;
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import com.example.thebox.ui.main.SectionsPagerAdapter;

public class ManejoCaja extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_manejo_caja);
//instancia los adaptadores para los fragments que van a ser situados
        SectionsPagerAdapter sectionsPagerAdapter = new SectionsPagerAdapter(this, getSupportFragmentManager());
        ViewPager viewPager = findViewById(R.id.view_pager);
        viewPager.setAdapter(sectionsPagerAdapter);
//se configura el tabLayout para escoger en la barra superior el fragment que se quiere visualizar
        TabLayout tabs = findViewById(R.id.tabs);
        tabs.setupWithViewPager(viewPager);
    }
    public void regresar(View v){finish();}}
```

OpcionesIniciales.java:

Funcionalidad y configuración de la página principal luego que el usuario ingresa sus credenciales en la pantalla de inicio. Se muestran las opciones de registrar o administrar una caja, o leer un código QR para abrir una caja.

```
package com.example.thebox;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.ExecutionException;

public class OpcionesIniciales extends AppCompatActivity {
    private TextView mostrarSaludo;
    private View manejoCaja,registroCaja,escanear;
    private String verificacionAdmin = "https://lab6-chiquito.000webhostapp.com/verificacionAdmin.php";
    private String verificacionQR = "https://lab6-chiquito.000webhostapp.com/verificacionQR.php";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_opciones_iniciales);
//Configuración de diseño de botones y se conectan los elementos del layout con los declarados como variables en el código
        mostrarSaludo=findViewById(R.id.mostrarNombreUsuario);
        String saludo = "¡Bienvenido "+UsuarioActual.getUser().getNomUsuario()+" !";
        mostrarSaludo.setText(saludo);
        manejoCaja = findViewById(R.id.administrar);
        registroCaja = findViewById(R.id.registrarCaja);
        escanear = findViewById(R.id.escanearQR);
        GradientDrawable drawableManejo = (GradientDrawable) manejoCaja.getBackground();
        drawableManejo.setColor(Color.TRANSPARENT);
        GradientDrawable drawableRegistro = (GradientDrawable) registroCaja.getBackground();
        drawableRegistro.setColor(Color.TRANSPARENT);
        GradientDrawable drawableEscanear = (GradientDrawable) escanear.getBackground();
```

```
        drawableEscanear.setColor(Color.TRANSPARENT);
    }
    public void regresar(View v){
        finish();
    }
//Realice el registro de una caja, verificando que el usuario tenga los permisos de administrador para luego abrir la ventana de registrar Caja.
    public void registrarCaja(View v){
        String[] resultado;
        try {
            String[] datos = new String[]{
                    "verificarAdmin",
                    verificacionAdmin,
                    UsuarioActual.getUser().getNomUsuario()
            };
            AsyncQuery async = new AsyncQuery();
            resultado = async.execute(datos).get();
            System.out.println(resultado[0]);
            String[] myData= resultado[0].trim().split("\\n")[0].split(",");
            ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
            if (infoImp.get(0).equals("denegado")){
                Toast.makeText(OpcionesIniciales.this, "Usuario no tiene permisos de administrador.", Toast.LENGTH_SHORT).show();
                return;
            }
//si tiene permisos pasa a registrar la caja
            Intent intent=new Intent(this,registroCaja.class);
            startActivity(intent);
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
//lleva al usuario a la actividad para verificar credenciales de la caja y escoja una de las que tiene permiso
```

```
    public void accederCaja(View v){
        Intent intent=new Intent(this,VerificacionCaja.class);
        startActivity(intent);
    }
    public void leerQR(View v){
//lectura de Código QR
        scanCode();
    }
    public void  scanCode(){
        IntentIntegrator intergrator = new IntentIntegrator(this);
        intergrator.setCaptureActivity(CaptureAct.class);
        intergrator.setOrientationLocked(false);
        intergrator.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
        intergrator.setPrompt("Escaneando codigo");
        intergrator.initiateScan();
    }
//se sobreecriben métodos para lectura de QR
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data){
        IntentResult result = IntentIntegrator.parseActivityResult(requestCode,resultCode,data);
        if (result != null){
            if (result.getContents() != null){
                String[] resultado;
                try {
                    String[] datos = new String[]{
                            "verificarQR",
                            verificacionQR,
                            UsuarioActual.getUser().getNomUsuario(),
                            result.getContents()

                    };
                    AsyncQuery async = new AsyncQuery();
                    resultado = async.execute(datos).get();
                    System.out.println(resultado[0]);
                    String[] myData= resultado[0].trim().split("\\n")[0].split(",");
```

```
            ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
//si la lectura del Código es exitosa se muestra un mensaje acorde a ello y se muestra un cuadro de diálogo
            if (infoImp.get(0).equals("acceso")){
               AlertDialog.Builder builder = new AlertDialog.Builder(this);
               builder.setMessage("Acceso Exitoso");
               builder.setTitle("Resultado de Escaneado");
               builder.setPositiveButton("Escanear de Nuevo", new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialogInterface, int which) {
                     scanCode();
                  }
               }).setNegativeButton("Terminado", new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialogInterface, int which) {
                     dialogInterface.cancel();
                  }
               });
               AlertDialog dialog = builder.create();
               dialog.show();
            }else {
               AlertDialog.Builder builder1 = new AlertDialog.Builder(this);
               builder1.setMessage("Acceso Negado");
               builder1.setTitle("Resultado de Escaneado");
               builder1.setPositiveButton("Escanear de Nuevo", new DialogInterface.OnClickListener()
{
                  @Override
                  public void onClick(DialogInterface dialogInterface, int which) {
                     scanCode();
                  }
               }).setNegativeButton("Terminado", new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialogInterface, int which) {
                     dialogInterface.cancel();
                  }
               });
```

```
                AlertDialog dialog = builder1.create();
                dialog.show();
            }

        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }


    }

}else{
    super.onActivityResult(requestCode,resultCode,data);
}
}

}
```

PermisosFragment.java:

Fragment que cuenta con las opciones de dar o quitar permisos a un usuario respecto a la caja que se deseó ingresar las credenciales para administrarla.

```
package com.example.thebox;

import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.ExecutionException;



public class PermisosFragment extends Fragment {
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
    private String mParam1;
    private String mParam2;
    private EditText nombreUsuario;
    private Spinner spinnerAccion;
    private String url_manejoAccesos = "https://lab6-
chiquito.000webhostapp.com/manejarAccesoCaja.php" ;



    public PermisosFragment() {
        // Required empty public constructor
    }

    public static PermisosFragment newInstance(String param1, String param2) {
        PermisosFragment fragment = new PermisosFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }
//lo que se ejecuta cuando está por mostrarse el fragment
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
//se relaciona el fragment con su layout
        View view = inflater.inflate(R.layout.fragment_permisos, container, false);
        nombreUsuario=view.findViewById(R.id.usuarioOrden);
        spinnerAccion = view.findViewById(R.id.spinnerComando);
//opciones del Spinner, comandos para agregar o eliminar usuario de los permisos
        String[] opciones = new String[]{"Agregar","Eliminar"};
//se configura el spinner con las opciones que proveerá
        ArrayAdapter<String> aa = new
ArrayAdapter<String>(getActivity(),android.R.layout.simple_spinner_dropdown_item,opciones);
        aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinnerAccion.setAdapter(aa);
        ImageButton regresar = view.findViewById(R.id.volverVerificarCaja);
        regresar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getActivity().finish();
            }
        });
        Button botonComando = (Button) view.findViewById(R.id.ejecutarOrden);
        GradientDrawable drawableCaja = (GradientDrawable) botonComando.getBackground();
        drawableCaja.setColor(Color.parseColor("#B388FF"));
//al dar clic en el botón commando se ejecuta el query de ingreso o eliminación de la table Permisos,
existen verificaciones para ver si el usuario se encuentra registrado, si en verdad existía un permiso para
ese usuario respecto a la caja en cuestión
```

```
botonComando.setOnClickListener(new View.OnClickListener()
{
   @Override
   public void onClick(View v)
   {String[] resultado;
      String nomUs = nombreUsuario.getText().toString();
      String comando = spinnerAccion.getSelectedItem().toString();
      if(nomUs.isEmpty()){
         Toast.makeText(getActivity(), "Ingrese información completa.",
Toast.LENGTH_SHORT).show();
         return;
      }
      try {
         String[] datos = new String[]{
               "manipularAccesos",
               url_manejoAccesos,
               nomUs,
               comando,
               UsuarioActual.getIdCajaActualenRevision()
         };
         AsyncQuery async = new AsyncQuery();
         resultado = async.execute(datos).get();
         System.out.println(resultado[0]);
         String[] myData= resultado[0].trim().split("\\n")[0].split(",");
         ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
         if (!infoImp.get(0).isEmpty()){
            Toast.makeText(getActivity(), infoImp.get(0), Toast.LENGTH_SHORT).show();
         } else {
            Toast.makeText(getActivity(), "Ocurrió un error.", Toast.LENGTH_SHORT).show();
         }
      } catch (ExecutionException e) {
         e.printStackTrace();
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
```

```
            }
        });
        return view;
    }
}
```

Registro.java:

Se establecen los comandos para la actividad donde el usuario ingresa sus datos para registrase como un usuario en la base de datos.

```
package com.example.thebox;


import androidx.appcompat.app.AppCompatActivity;


import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.ExecutionException;


public class Registro extends AppCompatActivity {
    private EditText edNombre,edApellido,edCorreo,edUsuario,edContraseña;
    private Button botonRegistrarse;
```

```
    private String register = "https://lab6-chiquito.000webhostapp.com/registro.php";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro);
//se asocian los elementos del layout con las variables en este código
        edNombre = findViewById(R.id.nombre);
        edApellido = findViewById(R.id.apellido);
        edCorreo = findViewById(R.id.correo);
        edUsuario = findViewById(R.id.usuario);
        edContraseña = findViewById(R.id.contrasena);
        botonRegistrarse = findViewById(R.id.botonRegistrarse);
        GradientDrawable drawableIngresar = (GradientDrawable) botonRegistrarse.getBackground();
        drawableIngresar.setColor(Color.parseColor("#B388FF"));
    }
    public void registrarse(View v) {
        String[] resultado;
        String nombre = edNombre.getText().toString();
        String apellido = edApellido.getText().toString();
        String correo = edCorreo.getText().toString();
        String usuario = edUsuario.getText().toString();
        String contrasena = edContraseña.getText().toString();
//se verifica que usuario ingresa todos los datos requeridos
        if(nombre.isEmpty()                                                                    ||
apellido.isEmpty()||correo.isEmpty()||usuario.isEmpty()||contrasena.isEmpty()){
```

```
        Toast.makeText(Registro.this, "Ingrese información solicitada.", Toast.LENGTH_SHORT).show();
        return;
    }
    String[] datos = new String[]{
        "registrar",
        register,
        usuario,
        nombre,
        apellido,
        correo,
        contrasena
    };
    AsyncQuery async = new AsyncQuery();
//se lleva a cabo el registro del usuario; si existe un usuario con un mismo nombre de usuario se le indica al usuario que no se registraron sus datos
    try {
        resultado = async.execute(datos).get();
        String[] myData= resultado[0].trim().split("\\n")[0].split(",");
        ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
        if (infoImp.get(0).equals("denegado")){
            Toast.makeText(Registro.this, "Usuario ya se encuentra registrado.", Toast.LENGTH_SHORT).show();
        } else{
            Toast.makeText(Registro.this, "Usuario registrado exitosamente.", Toast.LENGTH_SHORT).show();
        }
```

```
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    public void regresar(View v){
        finish();
    }


}
```

registroCaja.java:

Se establecen los comandos para la actividad donde el usuario ingresa los datos necesarios para registrar una caja en la base de datos. Se genere el código QR con el nombre de usuario concatenado a un número aleatorio.

```
package com.example.thebox;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.Point;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.util.Log;
import android.view.Display;
import android.view.View;
import android.view.WindowManager;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import com.google.zxing.WriterException;

import java.io.File;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;
import java.util.concurrent.ExecutionException;

import androidmads.library.qrgenearator.QRGContents;
import androidmads.library.qrgenearator.QRGEncoder;
import androidmads.library.qrgenearator.QRGSaver;

public class registroCaja extends AppCompatActivity {
    private View botonRegistro;
    private EditText pais,ciudad,calle,detalle,contrasena;
    private String url = "https://lab6-chiquito.000webhostapp.com/registroCaja1.php";
    String TAG = "GenerateQRCode";
    String infoQR, userName;
    Bitmap bitmap;
    QRGEncoder qrgEncoder;
    Random r;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro_caja);
```

```
        botonRegistro = findViewById(R.id.mandarRegistroCaja);
        pais = findViewById(R.id.paisCaja);
        ciudad = findViewById(R.id.ciudadCaja);
        calle = findViewById(R.id.calleCaja);
        detalle=findViewById(R.id.detalleCaja);
        contrasena = findViewById(R.id.contrasenaCaja);
//se modifica el fondo del boton a determinado color
        GradientDrawable drawableIngresar = (GradientDrawable) botonRegistro.getBackground();
        drawableIngresar.setColor(Color.parseColor("#B388FF"));
//se establecen los permisos para guardar la imagen del Código QR a generar
        ActivityCompat.requestPermissions(registroCaja.this,                                    new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
        ActivityCompat.requestPermissions(registroCaja.this,                                    new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE}, 1);


    }
    public void registrar(View v){


        String[] resultado;
        String paisCaja = pais.getText().toString();
        String ciudadCaja = ciudad.getText().toString();
        String calleCaja = calle.getText().toString();
        String detalleCaja = detalle.getText().toString();
        String contrasenaCaja = contrasena.getText().toString();
//se verifica que todos los campos necesarios hayan sido ingresados
        if(paisCaja.isEmpty()                                                                   ||
ciudadCaja.isEmpty()||calleCaja.isEmpty()||detalleCaja.isEmpty()||contrasenaCaja.isEmpty()){
            Toast.makeText(registroCaja.this,          "Complete          todos          los          campos.",
Toast.LENGTH_SHORT).show();
            return;
        }
        r = new Random();
        userName = UsuarioActual.getUser().getNomUsuario();
//la informacion del qr sera el nombre de usuario concatenado a un número aleatorio del 0 al 10000
        infoQR = userName + String.valueOf(r.nextInt(10001));
```

```
        String[] datos = new String[]{
                "registrarCaja",
                url,
                paisCaja,
                ciudadCaja,
                calleCaja,
                detalleCaja,
                contrasenaCaja,
                infoQR,
                userName
        };
        AsyncQuery async = new AsyncQuery();
        try {
            resultado = async.execute(datos).get();
            String[] myData= resultado[0].trim().split("\\n")[0].split(",");
//se muestra al usuario mensajes según lo que haya sucedido al registrarse
            ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
            if (infoImp.get(0).equals(" denegado ")){
                Toast.makeText(registroCaja.this, "La ubicación ya existía y el registro de la caja fue existoso.",
Toast.LENGTH_LONG).show();
            } else{
                Toast.makeText(registroCaja.this, infoImp.get(0), Toast.LENGTH_SHORT).show();
            }
//se establecen las dimensiones del Código QR a generar
            WindowManager manager = (WindowManager) getSystemService(WINDOW_SERVICE);
            Display display = manager.getDefaultDisplay();
            Point point = new Point();
            display.getSize(point);
            int width = point.x;
            int height = point.y;
            int smallerDimension = width < height ? width : height;
            smallerDimension = smallerDimension * 3 / 4;
//se codifica el String infoQR y se lo convierte a una imagen -bitmap- que representa el Código QR
generado
            qrgEncoder = new QRGEncoder(
```

```
                infoQR, null,
                QRGContents.Type.TEXT,
                smallerDimension);
        try {
            bitmap = qrgEncoder.encodeAsBitmap();
        } catch (WriterException e) {
            Log.v(TAG, e.toString());
        }

//se guarda la imagen en el dispositivo
        FileOutputStream outputStream = null;
        File file = Environment.getExternalStorageDirectory();
        File dir = new File(file.getPath() + "/QRCode/");
        dir.mkdirs();

        String filename = String.format("%d.jpeg", System.currentTimeMillis());
        File outfile = new File(dir, filename);

        try {
            outputStream = new FileOutputStream(outfile);
            //Toast.makeText(getApplicationContext(),infoQR,Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
            e.printStackTrace();
        }
// se le da un formato a la imagen
        bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
        try {
            outputStream.flush();
            outputStream.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
      } catch (ExecutionException e) {
        e.printStackTrace();
      } catch (InterruptedException e) {
```

```
            e.printStackTrace();
        }

    }

    public void regresar(View v){
        finish();
    }

}
```

SplashInicial.java:

Primera actividad mostrada, compuesta únicamente del logo de la aplicación y se muestra durante un corto período de tiempo.

```
package com.example.thebox;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.Window;
import android.view.WindowManager;

public class SplashInicial extends AppCompatActivity {
    private final int SLEEP_TIME = 3000;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);    // Remueve la barra de título
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);    // Remueve la barra de notificación
        setContentView(R.layout.activity_splash_inicial);
//se genera un proceso parecido a un hilo en el que esta actividad estará presente durante un período de tiempo SLEEP TIME y luego se pasa a la actividad Inicio

        new Handler().postDelayed(new Runnable(){
```

```
        @Override
        public void run() {
            Intent intent = new Intent(SplashInicial.this, Inicio.class);
            startActivity(intent);
            finish();
        }
    },SLEEP_TIME);
  }
}
```

UsuarioActual**.java:**

Clase que se destina a almacenar información sobre el usuario que está actualmente haciendo uso de la aplicación, con tal de utilizar dichos datos en procedimientos donde se involucra a la base de datos.

```
package com.example.thebox;


public class UsuarioActual {
  private String nombres,apellidos,correo,nomUsuario;
  private static UsuarioActual user;//permite guardar la información del usuario que está en sesión
  private static String idCajaActualenRevision;//permite guardar el id de la caja que se está administrando, viendo accesos; se accede a él fácilmente


  public UsuarioActual(String nomUsuario,String nombres, String apellidos, String correo) {
    this.nombres = nombres;
    this.apellidos = apellidos;
    this.correo = correo;
    this.nomUsuario = nomUsuario;
  }
//getters y setters
  public static String getIdCajaActualenRevision() {
    return idCajaActualenRevision;
  }
  public static void setIdCajaActualenRevision(String idCajaActualenRevision) {
    UsuarioActual.idCajaActualenRevision = idCajaActualenRevision;
  }
  public static void setUser(UsuarioActual user) {
```

```
        UsuarioActual.user = user;
    }
    public static UsuarioActual getUser() {
        return user;
    }
    public String getNomUsuario() {
        return nomUsuario;
    }
}
```

VerificacionCaja.java:

Actividad donde se muestran únicamente las cajas a las que el usuario actual tiene acceso y en donde él deberá ingresar la contraseña de la caja que desea administrar con tal de acceder a aquellas opciones.

```
package com.example.thebox;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.ExecutionException;

public class VerificacionCaja extends AppCompatActivity {
    private View botonVerificar;
    private Spinner spinnerCajas;
```

```
    private EditText contraCaja;
    private String[] listaInfoCajas;
    private ArrayList<String> listaIDCajas;
    private String urlRellenarSpinner = "https://lab6-
chiquito.000webhostapp.com/verPermisoCajasUsuario.php";
    private String urlVerificarContra = "https://lab6-chiquito.000webhostapp.com/administrarCaja.php";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_verificacion_caja);
//configuración diseño de botones
        botonVerificar = findViewById(R.id.botonVerificarCaja);
        GradientDrawable drawableCaja = (GradientDrawable) botonVerificar.getBackground();
        drawableCaja.setColor(Color.parseColor("#B388FF"));
        contraCaja=findViewById(R.id.contraVerificacionCaja);
        spinnerCajas = findViewById(R.id.spinnerCajas);
//se rellena el spinner con descripción de las cajas a las que el usuario tiene acceso
        listaInfoCajas = rellenarSpinner();
//se configura el spinner con la lista de opciones
        ArrayAdapter<String> aa = new
ArrayAdapter<String>(this,android.R.layout.simple_expandable_list_item_1,listaInfoCajas);
        aa.setDropDownViewResource(android.R.layout.simple_expandable_list_item_1);
        spinnerCajas.setAdapter(aa);
    }
//se hace la consulta a la base de datos, viendo en la tabla permisos los registros que coincidan con el
usuario actual, con tal de devolver todas las cajas a las que tiene acceso
    private String[] rellenarSpinner(){
        String[] resultado;
        try {
            String[] datos = new String[]{
                    "manejarCaja",
                    urlRellenarSpinner,
                    UsuarioActual.getUser().getNomUsuario()
            };
            AsyncQuery async = new AsyncQuery();
```

```
            resultado = async.execute(datos).get();
            System.out.println(resultado[0]);
            String[] myData= resultado[0].trim().split("\\n");
            ArrayList<String> infoImp = new ArrayList<>();
            listaIDCajas = new ArrayList<>();
//información en spinner se muestra en el formato info|info|info|
            for(int i=0;i<myData.length;i++){
                String strInfoCaja = myData[i].replace("," |");
                String id = myData[i].split(",")[0];
                listaIDCajas.add(id);
                infoImp.add(strInfoCaja);
            }
            return infoImp.toArray(new String[0]);

        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return new String[]{};
    }
    public void administrarCaja(View v){
        String[] resultado;
        String idCajaEscogidaSpinner =listaIDCajas.get(spinnerCajas.getSelectedItemPosition());
        String contra = contraCaja.getText().toString();
        if(spinnerCajas.getSelectedItem().toString().isEmpty() || contra.isEmpty()){
            Toast.makeText(VerificacionCaja.this, "Escoja una caja o ingrese contraseña.",
Toast.LENGTH_SHORT).show();
            return;
        }
        try {
            String[] datos = new String[]{
                    "verificarCaja",
                    urlVerificarContra,
                    idCajaEscogidaSpinner,
```

```
                contra
            };
            AsyncQuery async = new AsyncQuery();
            resultado = async.execute(datos).get();
            System.out.println(resultado[0]);
            String[] myData = resultado[0].trim().split("\\n")[0].split(",");
            ArrayList<String> infoImp = new ArrayList<>(Arrays.asList(myData));
            if (infoImp.get(0).equals("denegado")) {
               Toast.makeText(VerificacionCaja.this, "Ingrese contraseña correcta.",
Toast.LENGTH_SHORT).show();


               return;
            }
            UsuarioActual.setIdCajaActualenRevision(idCajaEscogidaSpinner);
            Intent intent = new Intent(this, ManejoCaja.class);
            startActivity(intent);
        }catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    public void regresar(View v){
        finish();
    }
}
```

Archivos XML pueden ser accedidos desde Github, no es preciso comentarlos pues al apreciar el diseño y la aplicación en funcionamiento se muestran los diversos Layouts. También se cuenta con imágenes en formato PNG.

Código fuente del módulo ESP-32

```
#include <HTTPClient.h>
 #include <WiFi.h>
const char* ssid = "Claro_TORO0011021002";//credenciales de la red wifi
```

```
const char* password = "GISELA61731338747385";
String id="1";

const String serverName = "https://lab6-chiquito.000webhostapp.com/esp-outputs-
action.php?action=outputs_state&id="+id+"";//php para la lectura de los estados de la caja

const long interval = 5000;//cantidad de tiempo para revisar la base de datos
unsigned long previousMillis = 0;//variable de tiempo actual
String apiKeyValue="tPmAT5Ab3j7F9";//token para poder escribir valores de estado en la base de datos
String State;//lectura de estado de acceso permitido o no
HTTPClient http;
boolean openned=false;
int openButton=13;//pin de lectura del boton para abrir la caja
int closeButton=14;//pin de lectura del boton para cerrar la caja
int ledclosed=12;//led indicativo de que la caja se encuentra cerrada
int ledopenned=15;//led indicativo de que la caja se encuentra abierta
int ledwarning=4;//led indicativo de que el usuario puede abrir la caja
int relay=5;//pin usado para activar la cerradura

void setup() {
 pinMode(ledclosed,OUTPUT);
  pinMode(ledopenned,OUTPUT);
   pinMode(ledwarning,OUTPUT);
    pinMode(relay,OUTPUT);
 Serial.begin(115200);
 WiFi.begin(ssid, password);
 Serial.println("Connecting");//intento de conexion a la red wifi
 while(WiFi.status() != WL_CONNECTED) {
  delay(500);//tiempo de espera para imprimir
  Serial.print(".");
 }
 Serial.println("");
 Serial.print("Connected to WiFi network with IP Address: ");
```

```
  Serial.println(WiFi.localIP());
}

void loop() {
  unsigned long currentMillis = millis();
  if(currentMillis - previousMillis >= interval) {//revisa si ha pasado suficiente tiempo para la siguiente consulta

    if(WiFi.status()== WL_CONNECTED ){ //revisa si hay conexion a internet
      State = httpGETRequest(serverName);//realiza request del estado de la caja
      Serial.println(State);
      Serial.println(State.toInt());

     if (State.toInt()){//usa el valor para ver si esta con permiso de abrirse o no
      openBox();//abre la caja
     }
     else{
  digitalWrite(ledclosed,HIGH);
 digitalWrite(ledwarning,LOW);
 digitalWrite(ledopenned,LOW);
 digitalWrite(relay,LOW);
    //cierra la caja
     }
      previousMillis = currentMillis;
    }
    else {
      Serial.println("WiFi Disconnected");
    }
  }
}
void openBox(){
    unsigned long initialTime = millis();
  unsigned long maxTime  = 60000 ;//un minuto//define tiempo de espera para tener permiso de abrir la caj
```

```
 digitalWrite(ledclosed,LOW);
  digitalWrite(ledwarning,HIGH);


 while((millis()-initialTime<maxTime)&&(!openned) ) {
 int reading=touchRead(openButton);//se queda en el bucle esperando a que se aplaste el buton o se acabe el tiempo


  if(reading<10){
     digitalWrite(relay,HIGH);
     Serial.println("HIGH");//abre la caja
     openned=true;
  }
  }


   while(openned){
    int readingClosed=touchRead(closeButton);


    digitalWrite(ledopenned,HIGH);//mientras esta abierta se queda en el bucle para que se mantanga abierta hasta que el usuario indique lo contrario
    digitalWrite(ledwarning,LOW);
    if(readingClosed<10){//revisa si el usuario ha aplastado el boton de cerrar
      setCloseState();//pone el valor de cerrado en la base de datos
      openned=false;
digitalWrite(ledclosed,HIGH);
digitalWrite(ledopenned,LOW);
digitalWrite(ledwarning,LOW);
digitalWrite(relay,LOW);
  }
  }
  setCloseState();
     openned=false;


digitalWrite(ledclosed,HIGH);
 digitalWrite(ledwarning,LOW);
```

```
 digitalWrite(ledopenned,LOW);
 digitalWrite(relay,LOW);
}
String httpGETRequest(String serverName) {
  http.begin(serverName);// se conecta con el servidor para realiazr los requests
  // Send HTTP POST request
  int httpResponseCode = http.GET(); //realiza el request del estado de la caja
  String payload;
  if (httpResponseCode>0) {
    Serial.print("HTTP Response code Get :  ");
    Serial.println(httpResponseCode);
    payload = http.getString();
  }
  else {
    Serial.println("efesota");
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  // Free resources
  http.end();
  return payload;
}
void setCloseState()
{
String server="https://lab6-chiquito.000webhostapp.com/post-esp-data.php";
http.begin(server);
String request="api_key="+apiKeyValue+"&id="+id+"&state=0";//manda el estado de la caja como
cerrado
Serial.print("request: ");
Serial.println(request);

  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
int httpResponseCode=http.POST(request);
```

```
Serial.println(http.getString());
  if (httpResponseCode>0) {
    Serial.print("HTTP Response code Post: ");
    Serial.println(httpResponseCode);
   }
   else {
    Serial.print("Error code Post: ");
    Serial.println(httpResponseCode);
   }
   // Free resources
   http.end();  }
```

12. Análisis de Presupuesto

12.1. Presupuesto de Hardware

En los costos de los recursos de hardware encontramos los siguientes gastos:

HARDWARE			
Concepto	**Cantidad**	**Valor Unitario**	**Total**
ESP 32	1	$8	$8
Relay	4	$1	$4
Material de impresión 3D	1	$2	$2
Solenoide	1	$4	$4
Jumpers	15	$0.1	$1.5
Leds	3	$0.15	$0.45
Resistencias 220 Ω	3	$0.15	$0.45
		Total	$20.4

Elaborado: Grupo #10
Fuente: Grupo#10

12.2. Presupuesto de Software

A continuación, se describen los programas a utilizar que son de código abierto:

SOFTWARE			
Concepto	**Cantidad**	**Valor Unitario**	**Total**
Base de datos MySQL	1	$0	$0
Android Studio	1	$0	$0
phpMyAdmin	1	$0	$0
ESP32	1	$0	$0
		Total	$0

Elaborado: Grupo#10
Fuente: Grupo#10

13. Conclusiones

En conclusión, la creación de una caja fuerte inteligente surge como respuesta a la demanda de soluciones efectivas aplicables a la administración y control de la seguridad para sectores como los gobiernos, bancos, empresas y el público en general. Además, a través de este proyecto, se implementaron nuevos usos a herramientas preexistentes aprovechando su eficiencia en lo que respecta a la individualización de la seguridad (códigos únicos) y la disponibilidad de un Smartphone, que tiene gran parte de la población. El desarrollo de esta caja inteligente conduce a explorar diferentes usos y adaptaciones de dispositivos tecnológicos que se encuentran disponibles, demostrando que con la debida investigación y análisis se podrán expandir sus alcances. Finalmente, en el campo de la seguridad, la creación de nuevas soluciones debe ser constante porque debemos reconocer que cada vez que se crea un sistema de seguridad, por más confiable que parezca, siempre habrá personas que quieran vulnerarlo.

14. Referencias bibliográficas

[1] Vanpoucke,B., “Lock Changer: Open Door with QR-code”, Instructables. [Online]. Disponible en: https://www.instructables.com/id/LockChanger/.

[2] “ESP32 Series Datasheet.” Espressif Systems, (Shanghai, 2020). Disponible en: “https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf”