**Universidad Hispanoamericana**

**PROGRAMACION 5**

**Documentación Proyecto Programado #2**

**Aplicación de Salón Comunal**

**Android Studio**

**Estudiante:**

**Fabian Carmona Solís**

**Danny Castro Barboza**

**Pablo Sánchez Ocampo**

**Tutor: JAIRO ALBERTO ZUÑIGA GOMEZ**

**Diciembre, 2023.**



**Tabla de contenidos**

[**Introducción** 2](#_Toc153472074)

[**Desarrollo** 2](#_Toc153472075)

[**Diagrama de clases de la aplicación** 2](#_Toc153472076)

[**Diagrama de base de datos** 2](#_Toc153472077)

[**Descripción de las clases y pantallas implementadas** 2](#_Toc153472078)

[**Conclusiones** 2](#_Toc153472079)

[**Recomendaciones** 2](#_Toc153472080)

# **Introducción**

El proyecto que se verá a continuación está basado en una aplicación móvil que fue realizada utilizando Android Studio, la funcionalidad del proyecto principalmente se enfocara en mostrar una pantalla de mantenimiento de los usuarios que se encuentran registrados en el backend del proyecto, además por cada usuario que ingrese a la aplicación (únicamente ingresando desde la pantalla de inicio de sesión) este mismo será agregado a la base de datos SQLite de la aplicación en Android Studio, una vez ingresado este usuario también podrá ser actualizado y eliminado desde el mantenimiento dentro de la aplicación. En este caso todos los usuarios serán administradores en la aplicación por lo tanto todos los usuarios verán las mismas pantallas. A continuación se detallan de una mejor manera las funciones con las cuales contara esta aplicación:

Registro e Ingreso de usuarios: Al ingresar a la aplicación se mostrará en pantalla un formulario a rellenar con los datos de inicio de sesión, además también se encontrará un botón de registro que al presionarlo enviará al usuario al formulario de registro para que cree una cuenta nueva y pueda iniciar sesión en la aplicación.

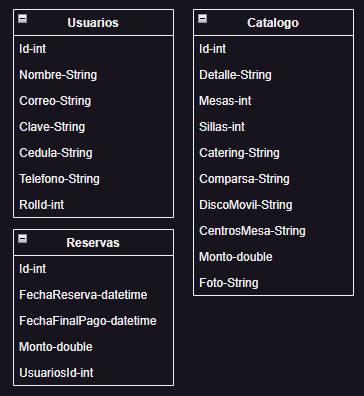
Mantenimiento de usuarios: En esta área de la aplicación se podrá agregar usuarios nuevos, de igual manera se podrá actualizar y eliminar usuarios, además también será posible mostrar todos los usuarios que existen agregados en el backend del proyecto o de lo contrario se podrá mostrar los datos de un usuario en específico por medio de su cedula.

Reservaciones: En este apartado se podrán crear, editar y borrar reservaciones indicando la fecha de reserva, la fecha final de pago, el monto a pagar y el usuario que realiza la reserva, además se podrá buscar una reserva por medio de su ID o por medio del nombre del usuario.

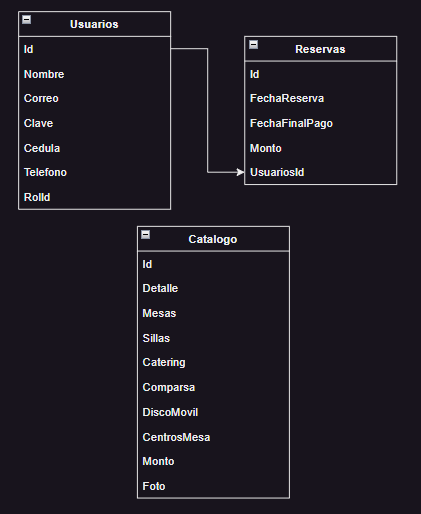
Catalogo: En esta pestaña se podrán agregar paquetes los cuales serán mostrados en la pantalla de Inicio y que incluyen un detalle, cantidad de mesas, silla, etc. De igual manera todo lo agregado al catálogo se podrá editar, eliminar y buscar por medio de su ID

# **Desarrollo**

## **Diagrama de clases de la aplicación**

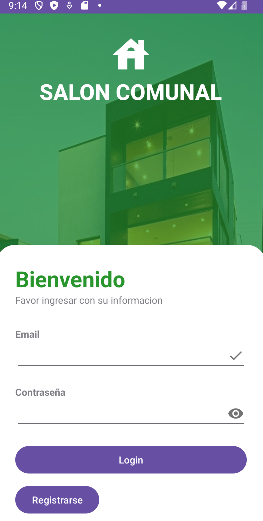


## **Diagrama de base de datos**

****

## **Descripción de las clases y pantallas implementadas**

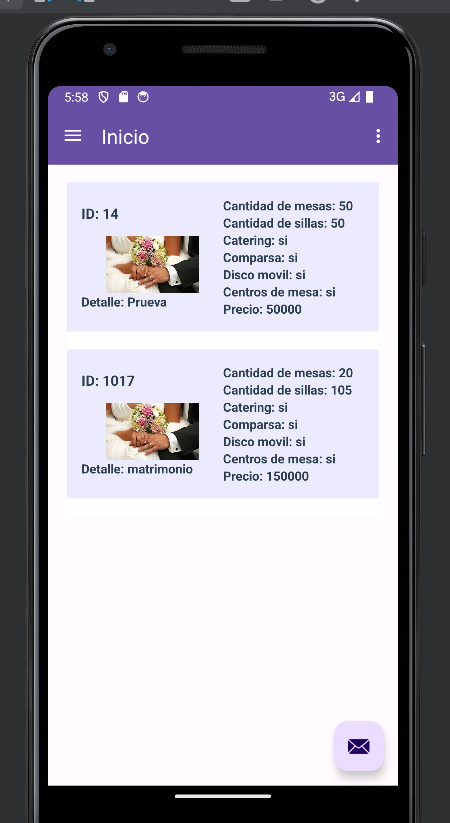
**Pantallas de Inicio de sesión y registro**

En la pantalla de inicio de sesión los usuarios ya registrados en la aplicación podrán ingresar la aplicación al rellenando el campo de Email y Contraseña.

En la pantalla de registro el usuario deberá rellenar el formulario de registro para así registrarse en el sistema y crear una cuenta con la cual podrán iniciar sesión en la pantalla de inicio de sesión.

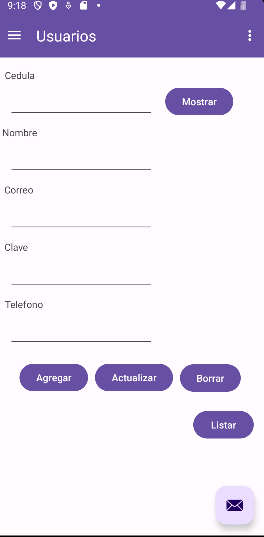
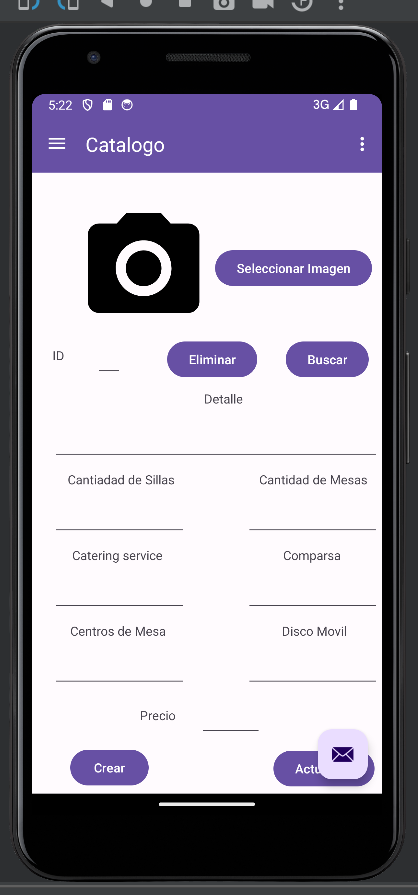
**Pantalla de inicio y reservaciones**

La pantalla de inicio es la primera pantalla que podrá ver el usuario al ingresar por defecto estará vacía, pero si algún paquete ha sido agregado con anterioridad al catálogo este se mostrara en la pantalla de inicio.

En la pantalla de reservaciones los usuarios podrán crear reservas, editarlas, actualizarlas y eliminarlas, además se podrá buscar una reserva por medio del ID de la reserva o por medio del nombre del usuario.

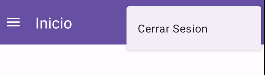
**Pantalla de usuarios y catalogo**

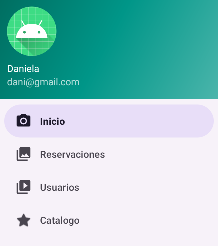
 

La pantalla de usuarios será un dónde se podrá agregar un nuevo usuario a aplicación, actualizar los datos de los usuarios y borrar los usuarios de la aplicación, también se podrá mostrar los datos de un usuario por medio de su cedula y también se podrá listar todos los usuarios registrados en el sistema.

La pantalla de catalogo permitirá al usuario crear paquetes los cuales serán mostrados en la pantalla de inicio solamente se necesita seleccionar una imagen, llenar campos de texto que se muestran en pantalla y darle al botón crear, también estos paquetes podrán ser actualizados y eliminados, además, se podrá buscar un paquete por medio de su ID.

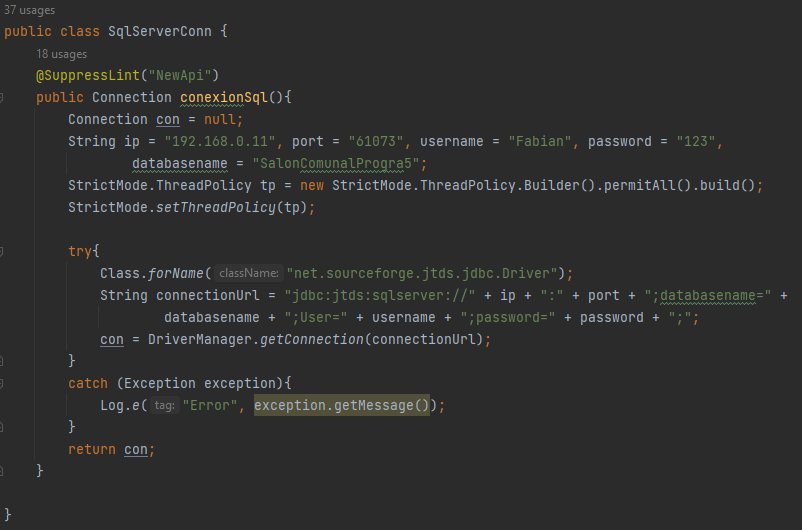
**Menú lateral y opción de cerrar sesión**



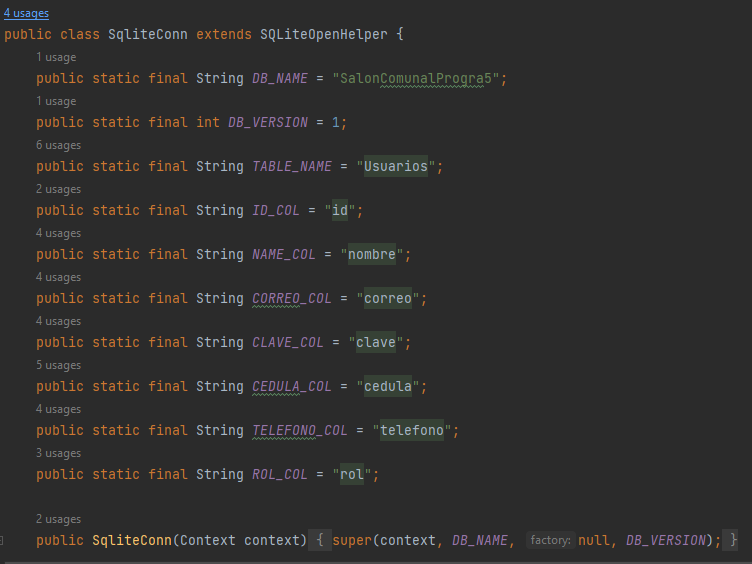
Al presionar el icono de la esquina superior izquierda se mostrará al usuario un menú lateral el cual incluye los datos del usuario que ingreso al sistema y la opción de trasladarse a las otras pantallas que incluye la aplicación. Si se presiona el icono de la esquina superior derecha se mostrará la opción de cerrar sesión la cual enviará al usuario nuevamente a la pantalla de inicio de sesión.

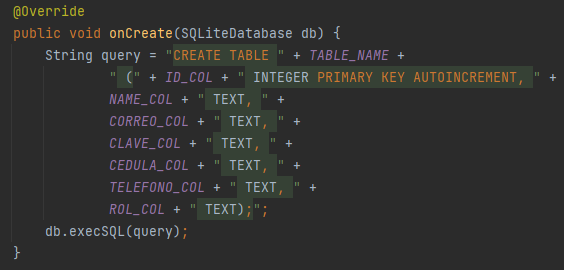
**Conexión con SQL Server**

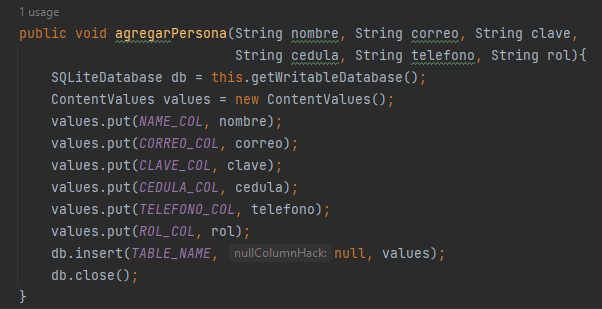


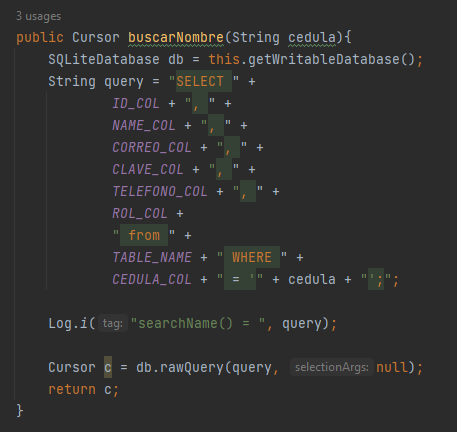
Esta clase realiza una conexión con SQL server mediante la ip, el puerto, el nombre de usuario y la contraseña en SQL server y la base de datos con la cual se hará la conexión.

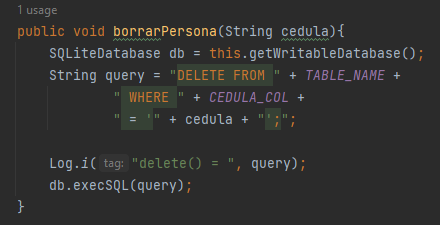
**Conexión con SQLite**

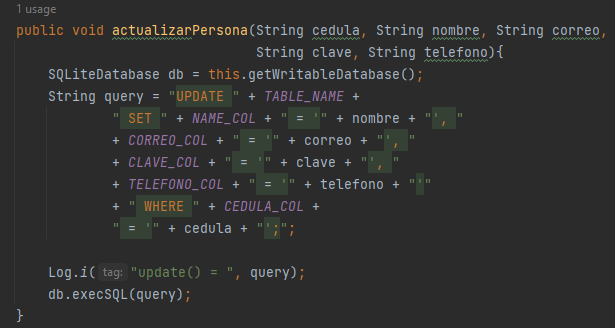


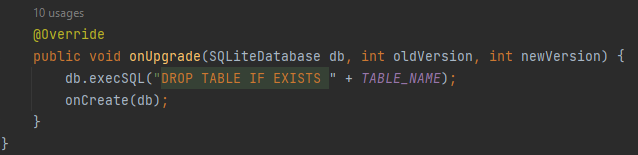








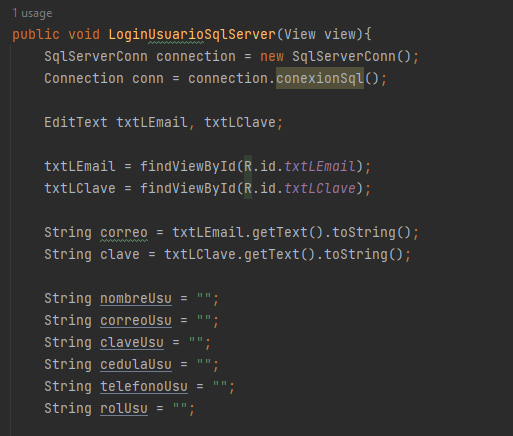


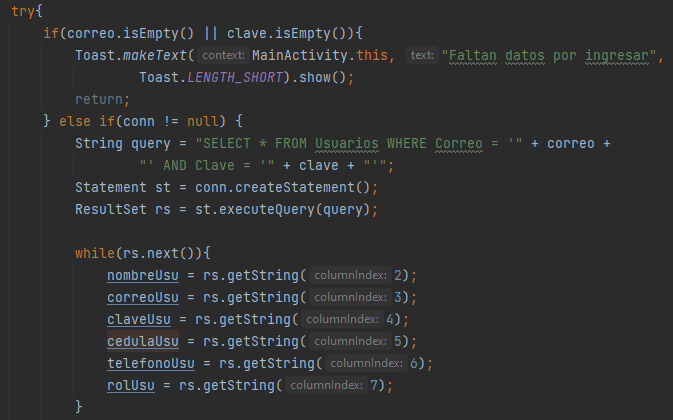


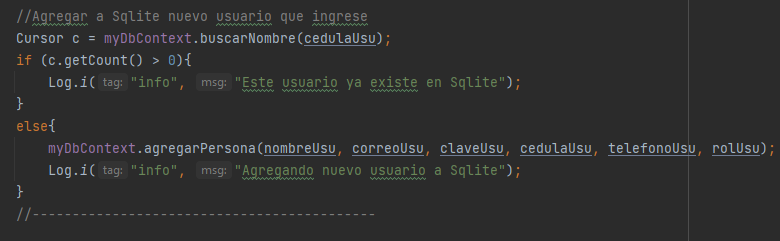
Esta clase realiza la conexión con la base de datos de SQLite, la función que tendrá esta clase es agregar a cada usuario registrado en el backend que ingrese a la aplicación para que quede guardado en la base de datos de SQLite, también actualizara los datos únicamente si el usuario que se encuentra agregado en SQLite sino únicamente se actualizara en SQL server indicante que el usuario aun no se encuentra en SQLite de igual manera pasara lo mismo con eliminar un usuario.

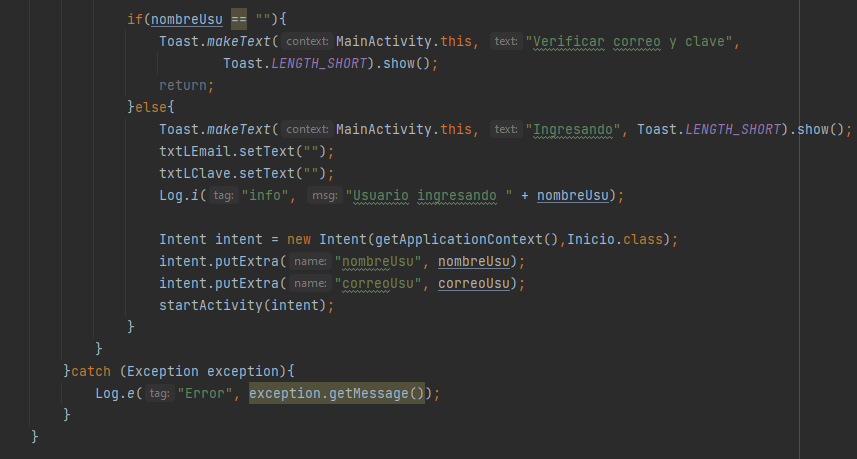
**Clase MainActivity**

**Inicio de sesión de usuarios**







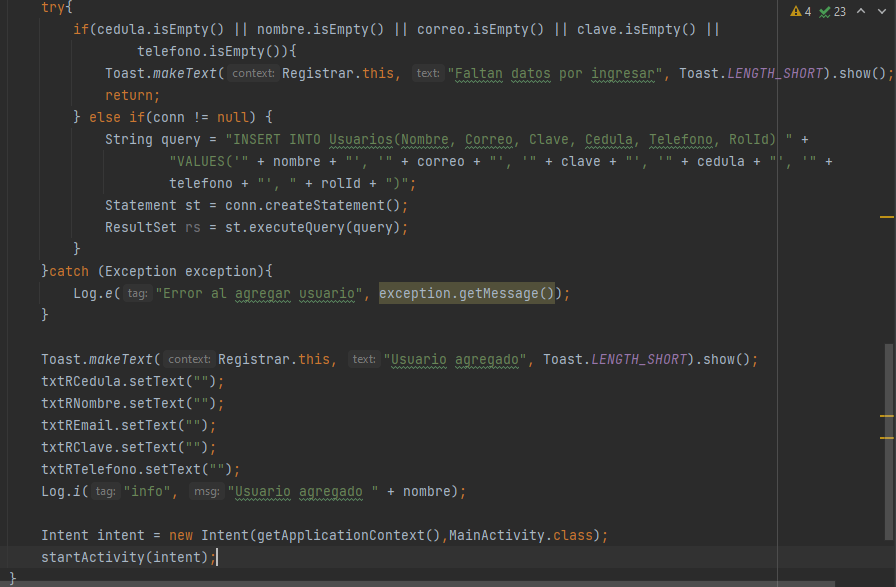


En esta clase estará la funcionalidad de la pantalla de inicio de sesión lo que permitirá a los usuarios ingresar en la aplicación.

**Clase Registrar**

**Registro de usuarios**





Esta clase tendrá la funcionalidad que habrá en la pantalla de registro de usuarios lo que les permitirá crear una cuenta a nuevos usuarios.

**Clase Inicio**

**Cargar datos del usuario**

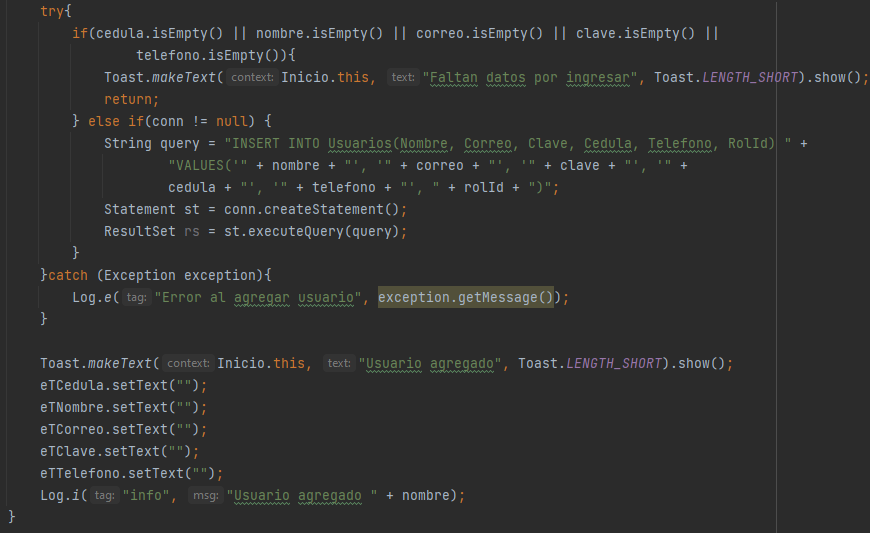


Esta función de cargar datos es la que cargara los datos del usuario que ingrese en la aplicación en el menú lateral.

**Mantenimiento de usuarios**

**Agregar usuario**

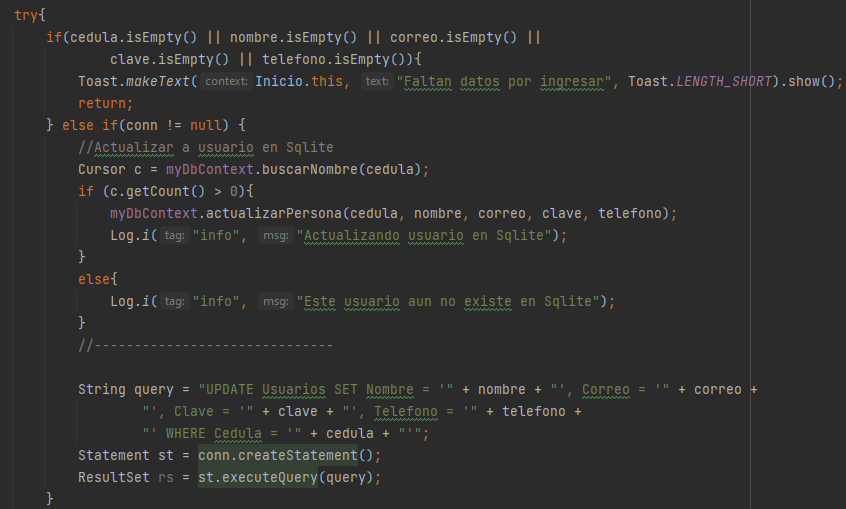
****

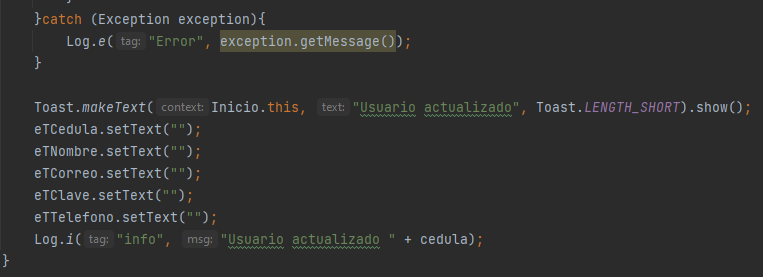
****

Esta función permite agregar nuevos usuarios a la aplicación.

**Actualizar usuario**

****

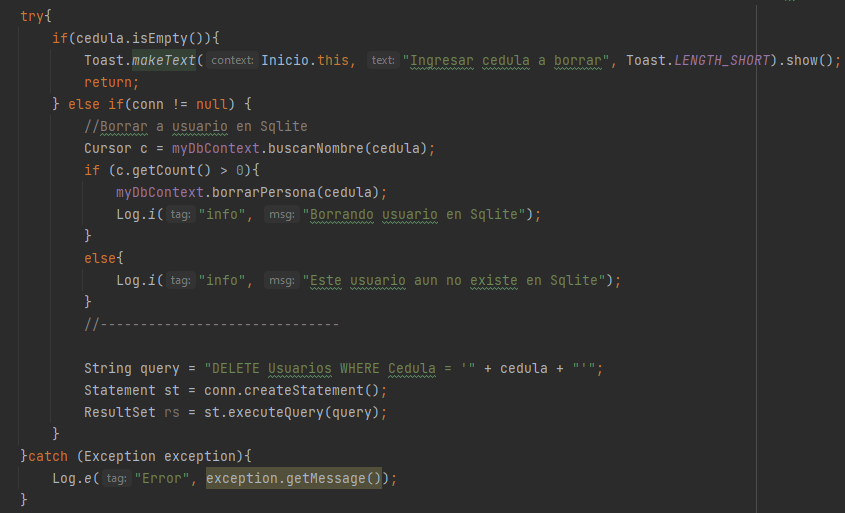
****

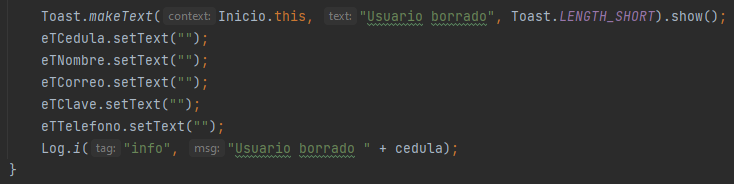
****

Esta función permite actualizar los datos de un usuario registrado además si el usuario se encuentra en SQLite también será actualizado en SQLite.

**Borrar usuario**

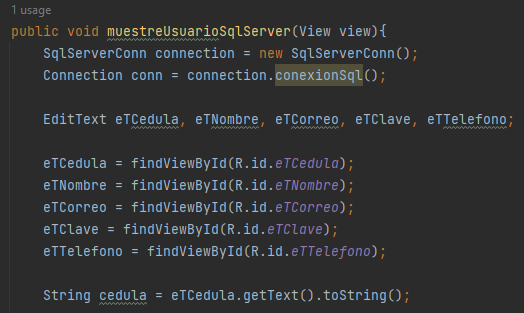
****

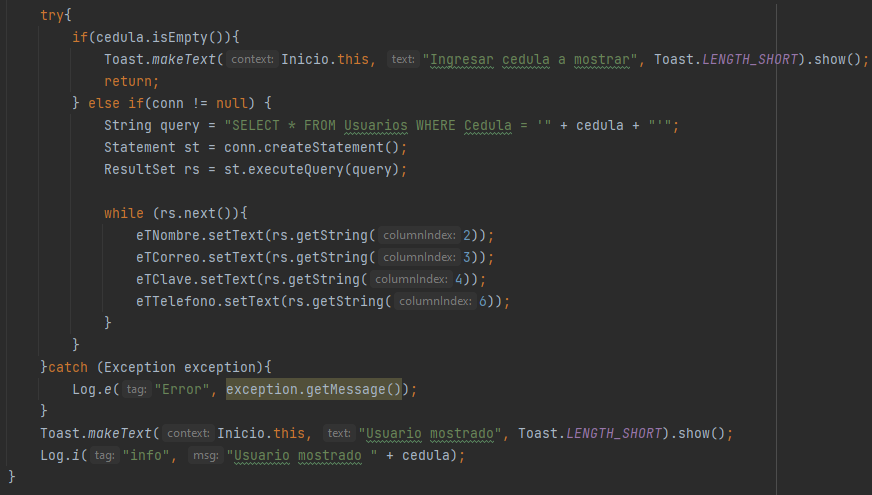
****

****

Esta función eliminará a los usuarios que se encuentren registrados en la aplicación, además, si el usuario en encuentra en SQLite este será eliminado de igual manera.

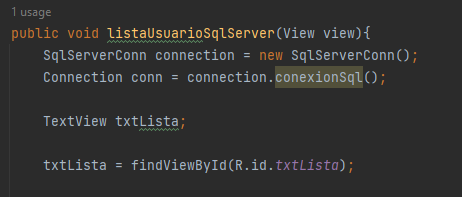
**Mostrar datos de un usuario**

****

****

Esta función mostrara los datos de un usuario mediante la cedula de este mismo.

**Mostrar todos los usuarios**

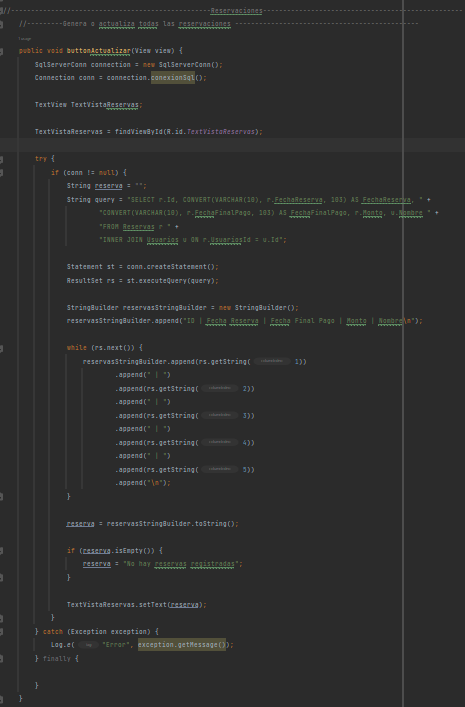
****

****

Esta función permitirá mostrar todos los usuarios que se encuentren registrados en la aplicación.

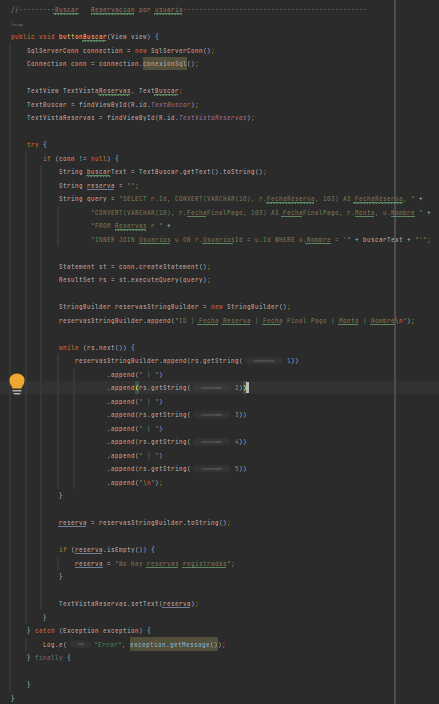
**Reservaciones**

En el siguiente método se encarga de mostrar la información de las Reservaciones desde la base de datos SQL Server. A continuación, se proporciona el código y el código detallado.



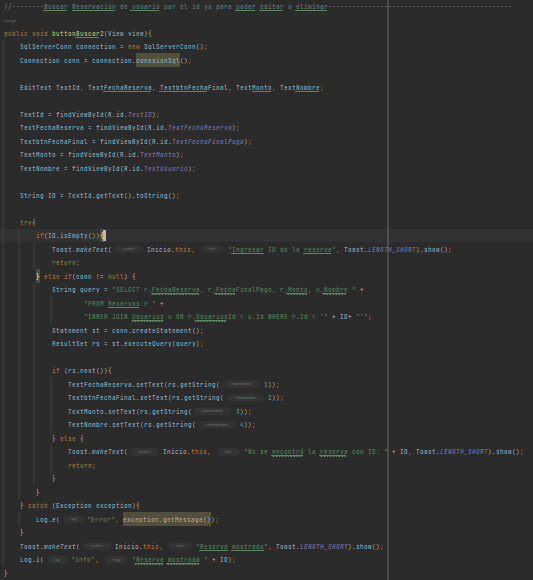
* **Establecer Conexión a la Base de Datos**: Se inicia una conexión a la base de datos SQL Server.
* **Obtener Referencia al TextView:** Se obtiene la referencia al TextView donde se mostrará la información de las reservas.
* **Consulta SQL:** Se define una consulta SQL para recuperar la información relevante de las reservas y los usuarios asociados.
* **Ejecutar la Consulta:** La consulta se ejecuta y se obtiene un conjunto de resultados.
* **Construir la Cadena de Reservas:** Se itera a través de los resultados y se construye una cadena de texto con la información de las reservas en formato de tabla.
* **Mostrar la Información en el TextView:** La cadena de reservas se establece como el texto del TextView para que sea visible en la interfaz de usuario.
* **Manejar Excepciones:** Cualquier excepción que ocurra durante el proceso se maneja y se registra un mensaje de error.

En el siguiente método se encarga de buscar la información de las Reservación a un usuario en específico buscado solo por el nombre y mostrando solo los datos de ese usuario en el TextView. A continuación, se proporciona el código y el código detallado.



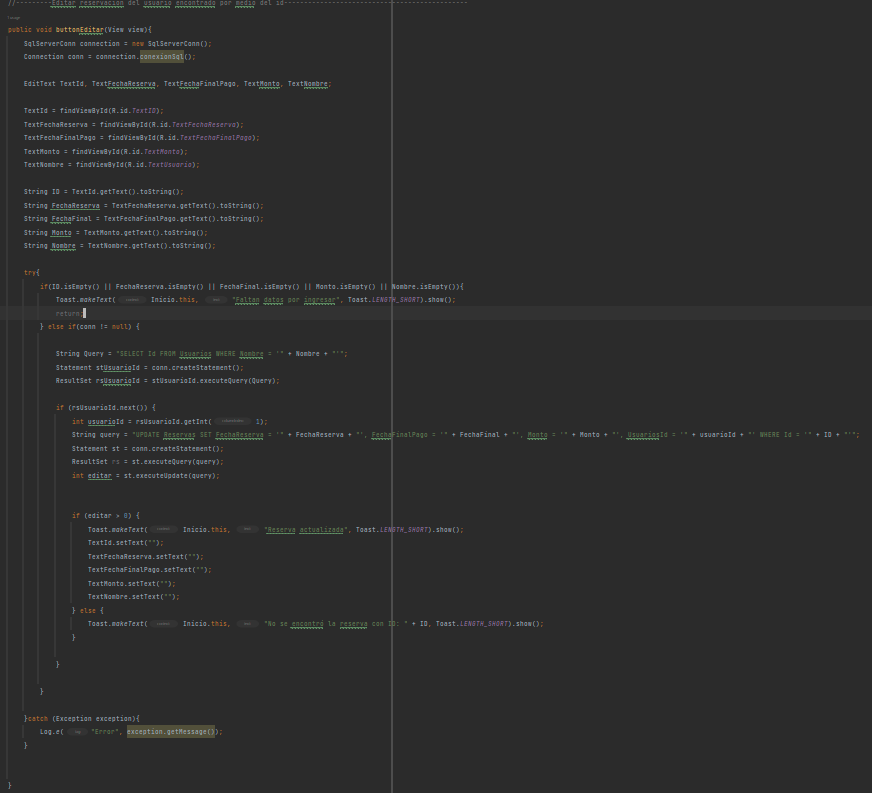
* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Serve
* **Obtener Referencias a los TextView:** Se obtienen las referencias a los TextView donde se ingresará el nombre a buscar y se mostrarán las reservas.
* **Obtener Nombre a Buscar:** Se recupera el nombre ingresado por el usuario desde el TextView.
* **Consulta SQL para Búsqueda:** Se define una consulta SQL para obtener la información de las reservas asociadas al usuario específico.
* **Ejecutar la Consulta:** La consulta se ejecuta y se obtiene un conjunto de resultados.
* **Construir la Cadena de Reservas**: Se itera a través de los resultados y se construye una cadena de texto con la información de las reservas en formato de tabla.
* **Mostrar la Información en el TextView:** La cadena de reservas se establece como el texto del TextView para que sea visible en la interfaz de usuario.
* **Manejar Excepciones:** Cualquier excepción registra un mensaje de error.

Este método se encarga de buscar y mostrar la información de una reserva específica mediante su ID en una interfaz de usuario. Aquí está el código y la documentación detallada del código:



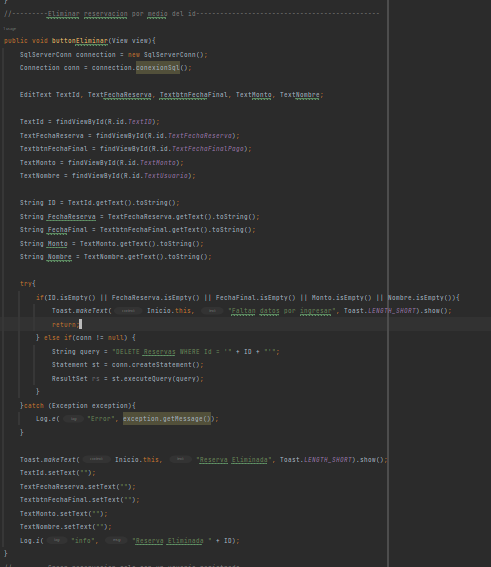
* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server.
* **Obtener Referencias a los EditText:** Se obtienen las referencias a los EditText donde se mostrará la información de la reserva por ID.
* **Obtener ID de la Reserva**: Se recupera el ID ingresado por el usuario desde el EditText.
* **Verificar ID:** Se verifica si el ID está vacío y se muestra un mensaje de error en ese caso.
* **Consulta SQL para Búsqueda por ID:** Se define una consulta SQL para obtener la información de la reserva por ID.
* **Ejecutar la Consulta:** La consulta se ejecuta y se obtiene un conjunto de resultados.
* **Verificar Resultados y Mostrar Información:** Se verifica si se encontró una reserva con el ID especificado y se muestra la información en los campos correspondientes.
* **Manejar Excepciones:** Cualquier excepción que ocurra durante el proceso se maneja y se registra un mensaje de error.
* **Mostrar Mensaje y Registrar en el Log:** Se muestra un mensaje indicando que la reserva ha sido mostrada y se registra esta información en el log.

Este método se encarga de editar la información de una reserva específica encontrada por su ID. Aquí está el código y la documentación detallada del código:



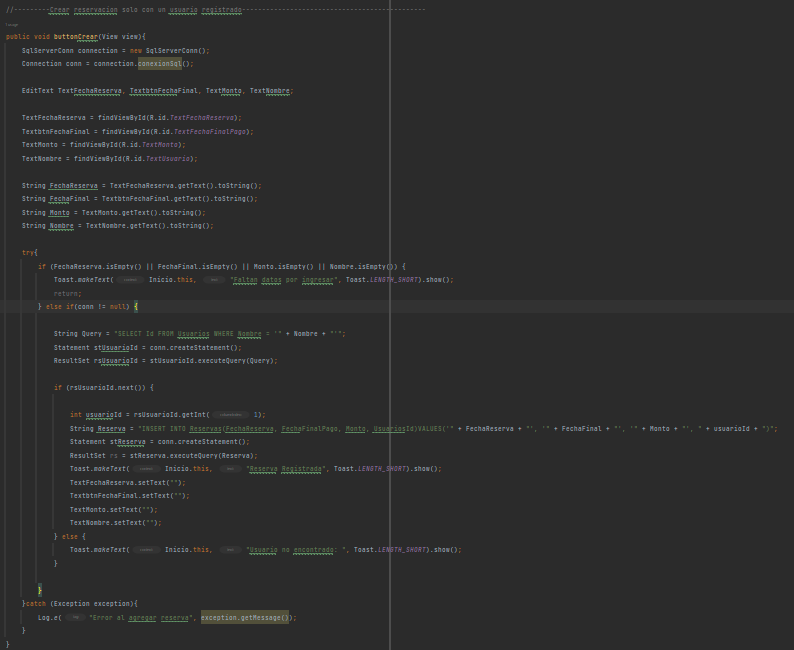
* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server
* **Obtener Referencias a los EditText:** Se obtienen las referencias a los EditText donde se ingresan los datos de la reserva a editar.
* **Obtener Valores de los EditText**: Se recuperan los valores ingresados por el usuario desde los EditText.
* **Verificar Datos Vacíos:** Se verifica si algún dato necesario está vacío y se muestra un mensaje de error en ese caso.
* **Consulta SQL para Obtener ID de Usuario:** Se define una consulta SQL para obtener el ID del usuario asociado al nombre ingresado.
* **Ejecutar Consulta y Obtener ID de Usuario:** La consulta se ejecuta y se obtiene el ID del usuario.
* **Consulta SQL para Actualización:** Se define una consulta SQL para actualizar la información de la reserva en la base de datos.
* **Ejecutar Actualización y Verificar Éxito:** La actualización se ejecuta y se obtiene el número de filas afectadas. Se verifica si la actualización fue exitosa.
* **Mostrar Mensajes y Limpiar EditText:** Se muestran mensajes indicando el resultado de la actualización y se limpian los EditText después de una actualización exitosa.

En el siguiente método se encarga de eliminar una reserva específica de la base de datos SQL Server, utilizando el ID de la reserva proporcionado. A continuación, se presenta el código y la documentación detallada del código:



* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server
* **Obtener Referencias a los EditText:** Se obtienen las referencias a los EditText donde se ingresan los datos de la reserva a eliminar.
* **Obtener Valores de los EditText:** Se recuperan los valores ingresados por el usuario desde los EditText.
* **Verificar Datos Vacíos:** Se verifica si algún dato necesario está vacío y se muestra un mensaje de error en ese caso.
* **Consulta SQL para Eliminación:** Se define una consulta SQL para eliminar la reserva de la base de datos.
* **Ejecutar Eliminación:** La eliminación se ejecuta.
* **Mostrar Mensajes y Limpiar EditText:** Se muestra un mensaje indicando que la reserva ha sido eliminada y se limpian los EditText después de la eliminación.
* **Manejar Excepciones:** Cualquier excepción que ocurra durante el proceso se maneja y se registra un mensaje de error.

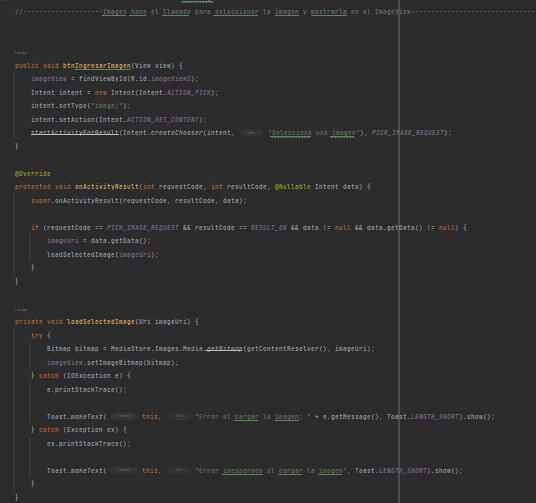
En el siguiente método se encarga de crear una nueva reserva en la base de datos SQL Server asociada a un usuario registrado. A continuación, se presenta el código y la documentación detallada del código:



* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server
* **Obtener Referencias a los EditText:** Se obtienen las referencias a los EditText donde se ingresan los datos de la nueva reserva.
* **Obtener Valores de los EditText:** Se recuperan los valores ingresados por el usuario desde los EditText.
* **Verificar Datos Vacíos:** Se verifica si algún dato necesario está vacío y se muestra un mensaje de error en ese caso.
* **Consulta SQL para Obtener ID del Usuario:** Se define una consulta SQL para obtener el ID del usuario registrado utilizando su nombre.
* **Ejecutar Consulta para ID del Usuario:** La consulta se ejecuta.
* **Verificar Existencia del Usuario:** Se verifica si el usuario existe.
* **Obtener ID del Usuario:** Si el usuario existe, se obtiene su ID.
* **Consulta SQL para Insertar Reserva:** Se define una consulta SQL para insertar una nueva reserva asociada al usuario.
* **Ejecutar Inserción de Reserva:** La inserción se ejecuta.
* **Mostrar Mensajes y Limpiar EditText:** Se muestra un mensaje indicando que la reserva ha sido registrada y se limpian los EditText después de la inserción.

**Selección y conversión de imágenes**

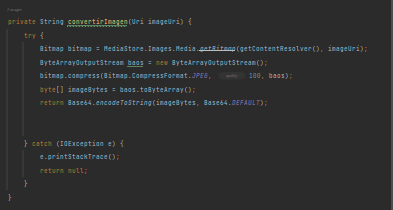
A continuación, se proporciona código y una documentación para la selección de imágenes en la aplicación:



* **btnIngresarImagen**: Este método inicia una actividad para permitir al usuario seleccionar una imagen de la galería cuando se presiona el botón de ingreso de imagen.
* **onActivityResult:** Este método se llama automáticamente cuando la actividad de selección de imágenes finaliza. Verifica si la selección de imágenes fue exitosa y, en ese caso, llama al método loadSelectedImage para cargar la imagen seleccionada en el ImageView.
* **loadSelectedImage:** Este método carga la imagen seleccionada desde la URI en el ImageView. Convierte la URI en un objeto Bitmap y lo establece en el ImageView. Además, maneja posibles errores y muestra mensajes de error en caso de fallos.

A continuación, se proporciona código y una documentación detallada para el código relacionado con la creación de catálogos y la conversión de imágenes:

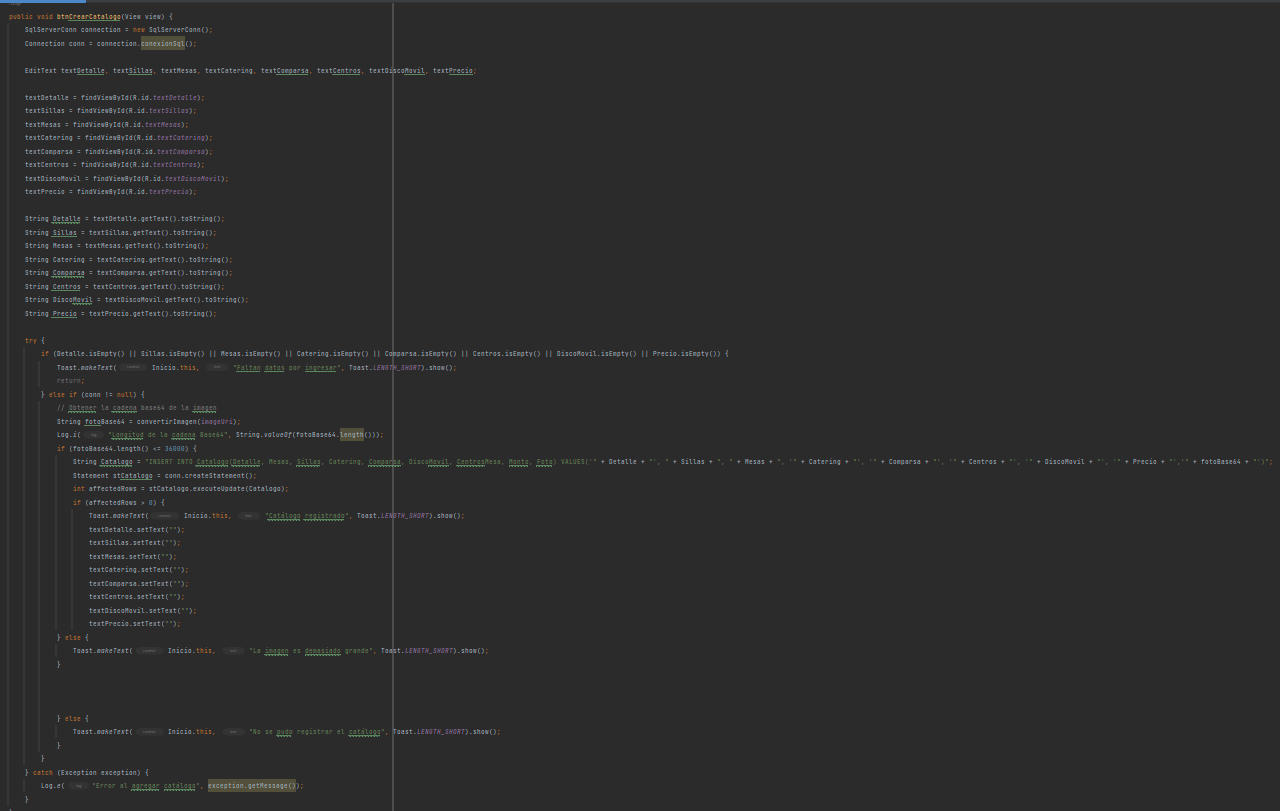
Este método toma una URI de imagen como entrada, carga la imagen desde la galería del dispositivo, la convierte en un formato de cadena base64 y la devuelve.



* **Obtener Bitmap a partir de la URI:** Se obtiene un objeto Bitmap a partir de la URI de la imagen seleccionada.
* **Convertir Bitmap a Bytes en Formato JPEG:** Se convierte el objeto Bitmap a bytes en formato JPEG.
* **Convertir Bytes a Cadena Base64:** Los bytes se convierten a una cadena base64 utilizando Base64.encodeToString.
* **Retornar la Cadena Base64:** La cadena base64 resultante se devuelve como resultado del método.
* **Manejar Excepciones:** Se manejan posibles excepciones, como errores al cargar la imagen, y se retorna null en caso de fallo.

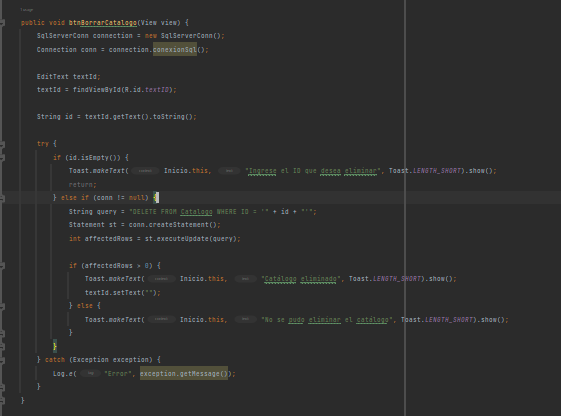
**Catalogo**

Este método se llama cuando se presiona el botón para crear un catálogo



* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server.
* **Obtener Referencias a los Elementos de Entrada de Texto:** Se obtienen referencias a los elementos de entrada de texto desde el diseño mediante su identificador.
* **Obtener Valores de los Elementos de Entrada de Texto:** Se recuperan los valores ingresados por el usuario.
* **Verificar Datos Vacíos:** Se verifica si algún dato necesario está vacío.
* **Obtener Cadena Base64 de la Imagen Seleccionada:** Se obtiene la cadena base64 de la imagen seleccionada mediante la llamada al método convertirImagen(imageUri).
* **Validar Longitud de la Cadena Base64:** Se valida que la longitud de la cadena base64 no supere un límite predefinido (36000 caracteres).
* **Si la longitud es aceptable, se procede con la inserción del catálogo**.
* **Construir y Ejecutar Consulta SQL para Inserción del Catálogo:** Se construye la consulta SQL para insertar un nuevo catálogo en la base de datos.
* **La consulta se ejecuta mediante stCatalogo.executeUpdate(Catalogo)**.
* **Verificar Inserción Exitosa:** Se verifica si la inserción del catálogo fue exitosa (afectó más de una fila):
* **Limpiar Elementos:** Se limpian los campos de entrada después del registro exitoso para facilitar la entrada de nuevos catálogos.

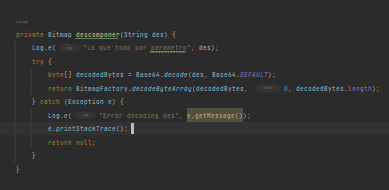
El siguiente método se encarga de manejar el evento de clic del botón para eliminar un catálogo por su ID. A continuación, se proporciona el código y la documentación:



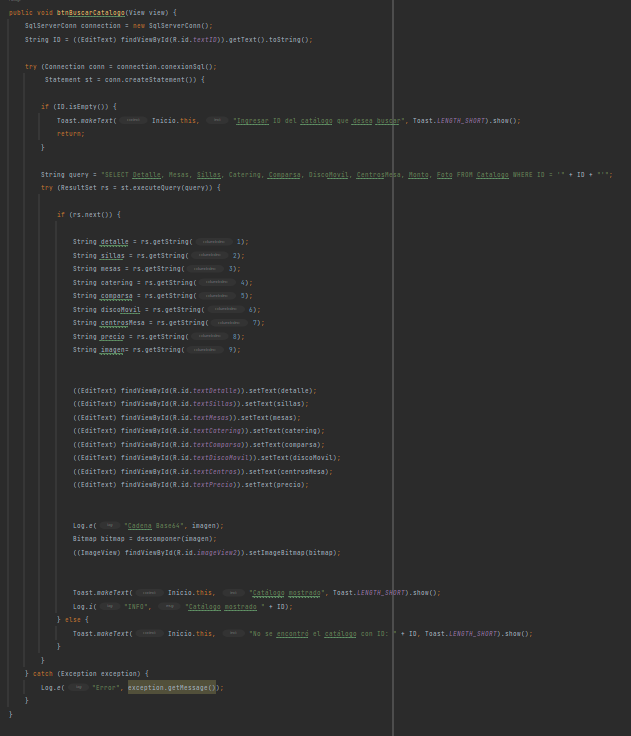
* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server.
* **Obtener Referencias a los Elementos de Entrada de Texto**: Se obtiene una referencia al elemento de entrada de texto mediante su identificador.
* **Obtener Valor del Elemento de Entrada de Texto:** Se recupera el valor ingresado por el usuario desde el elemento de entrada de texto.
* **Verificar Datos Vacíos:** Se verifica si el campo de ID está vacío.
* Construir y Ejecutar Consulta SQL para Eliminar el Catálogo: Se construye la consulta SQL para eliminar el catálogo de la base de datos utilizando el ID proporcionado.
* **Verificar Eliminación Exitosa:** Si es exitosa, se muestra un mensaje indicando que el catálogo ha sido eliminado.
* **Limpiar Elemento de Entrada de Texto Después de la Eliminación Exitosa.**
* **Manejar Excepciones:** Cualquier excepción que ocurra durante el proceso se maneja en el bloque catch.

Los siguientes métodos se encarga de buscar un catálogo por su ID y mostrar la información asociada en los campos de entrada de texto y la imagen correspondiente. Aquí está la explicación:

El método descomponer toma una cadena Base64 como entrada y devuelve un objeto Bitmap:



* **Decodificar la Cadena Base64:** La cadena Base64 se decodifica utilizando Base64.decode(des, Base64.DEFAULT) para obtener un array de bytes.
* **Crear un Objeto Bitmap:** Se crea un objeto Bitmap pasando el array de bytes, el índice de inicio y la longitud.
* **Manejar Excepciones:** Cualquier excepción que ocurra durante el proceso se maneja en el bloque catch.



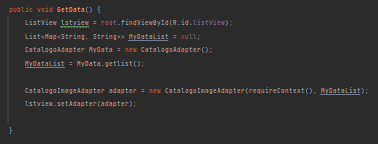
* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server.
* Obtener Valor del Elemento de Entrada de Texto para el ID: Se obtiene el valor del elemento de entrada de texto.
* **Verificar ID Vacío:** Se verifica si el campo de ID está vacío.
* **Construir y Ejecutar Consulta SQL para Buscar el Catálogo:** Se construye la consulta SQL para seleccionar los detalles del catálogo utilizando el ID proporcionado.
* **Se ejecuta la consulta.**
* **Procesar Resultado de la Consulta:** Si se encuentra un catálogo con el ID especificado, se extraen los detalles y se asignan a los campos de entrada de texto correspondientes.
* **La cadena Base64 de la imagen:** Se extrae del resultado y se utiliza para cargar la imagen en el ImageView mediante la función descomponer(imagen).

El siguiente método se encarga de actualizar la información de un catálogo existente en la base de datos, utilizando el ID proporcionado, a continuación se proporciona el código y la documentación.:



* **Establecer Conexión a la Base de Datos:** Se inicia una conexión a la base de datos SQL Server.
* Obtener Referencias a los Elementos de Entrada de Texto: Se obtienen referencias a los elementos de entrada de texto.
* **Obtener Valores de los Elementos de Entrada de Texto:** Se recuperan los valores ingresados por el usuario desde los elementos de entrada de texto.
* **Verificar Datos Vacíos:** Se verifica si algún dato necesario está vacío.
* **Construir Consulta SQL para Actualizar el Catálogo:** Se construye la consulta SQL para actualizar el catálogo utilizando los datos ingresados y el ID proporcionado.
* **Obtener la Cadena Base64 de la Imagen:** Se obtiene la cadena Base64 de la imagen seleccionada mediante la función convertirImagen(imageUri).
* **Verificar Longitud de la Cadena Base64:** Se verifica que la longitud de la cadena Base64 sea menor o igual a 36000 caracteres.
* **Ejecutar Actualización del Catálogo:** La actualización se ejecuta mediante st.executeUpdate(query).
* **Mostrar Mensajes y Limpiar Elementos de Entrada de Texto:** Se limpian los elementos de entrada de texto después de la actualización.

**El siguiente método está relacionada con la parte de una funcionalidad relacionada con la obtención y visualización de datos en un ListView. Aquí está la explicación paso a paso:**



**Obtener Referencia al ListView:**

* Se obtiene una referencia al ListView utilizando el identificador R.id.listView.
* Parece que el ListView está declarado en el layout asociado a la variable root.

**Inicializar una Lista de Mapas:**

* Se inicializa una lista de mapas llamada MyDataList que contendrá datos en forma de pares clave-valor.
* La lista se inicializa a null.
* Crear una Instancia de CatalogoAdapter:
* Se crea una instancia de la clase CatalogoAdapter llamada MyData.

**Obtener Datos para la Lista:**

* Se llama al método getlist() de MyData para obtener la lista de datos.

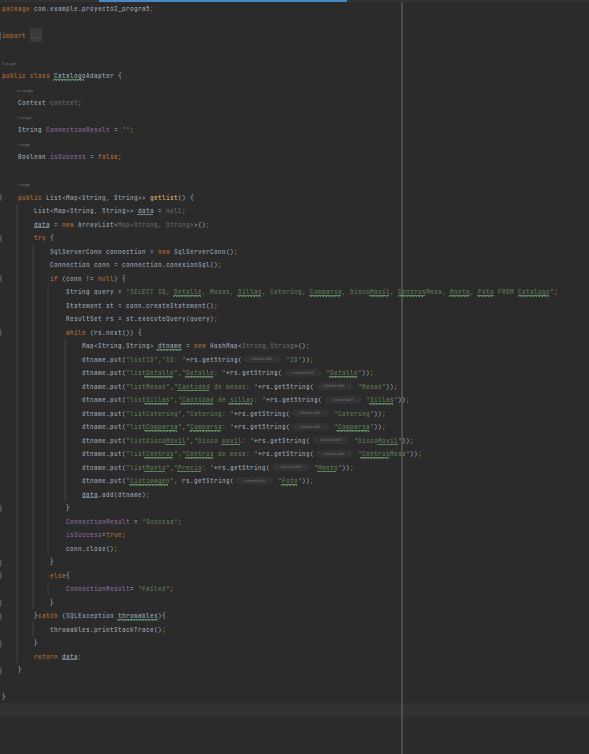
**Crear un Adaptador CatalogoImageAdapter:**

* Se crea una instancia de la clase CatalogoImageAdapter pasando el contexto (requireContext()) y la lista de datos (MyDataList).

**Establecer el Adaptador en el ListView:**

* Se establece el adaptador en el ListView utilizando lstview.setAdapter(adapter).

**La clase CatalogoAdapter es una clase de adaptador que se encarga de interactuar con la base de datos y proporcionar datos en un formato adecuado para la visualización con el ListView. Aquí está el código y la documentación:**



En el método getList es la siguiente documentación:

Este método devuelve una lista de mapas (List<Map<String, String>>).

Se inicializa la lista de datos (data) como un nuevo ArrayList.

Se intenta establecer una conexión a la base de datos SQL.

Si la conexión es exitosa, se ejecuta una consulta SQL para obtener datos del catálogo.

Se recorren los resultados de la consulta y se crean mapas para cada fila de la tabla.

Los mapas se agregan a la lista data.

Se establece ConnectionResult en "Success" y isSuccess en true si la conexión fue exitosa.

Se cierra la conexión a la base de datos.

Si la conexión falla, se establece ConnectionResult en "Failed".

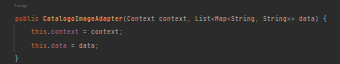
En caso de excepciones de SQL, se imprimen en la consola, pero no se manejan de otra manera.

Finalmente, se devuelve la lista de datos.

**La siguiente clase es un adaptador personalizado que extiende BaseAdapter y se utiliza para vincular datos a una interfaz de usuario, especialmente para un ListView. Aquí está el código y la documentacion:**

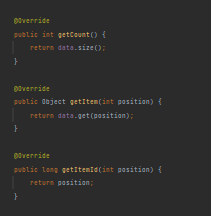
**Constructor:**

* Se proporciona un constructor que recibe el contexto y la lista de datos (data) como parámetros. Los asigna a los atributos correspondientes de la clase.



**Métodos de BaseAdapter:**

* getCount(): Devuelve la cantidad de elementos en la lista (data).
* getItem(int position): Devuelve el objeto en la posición especificada de la lista.
* getItemId(int position): Devuelve el identificador único de la posición especificada. En este caso, simplemente devuelve la posición.

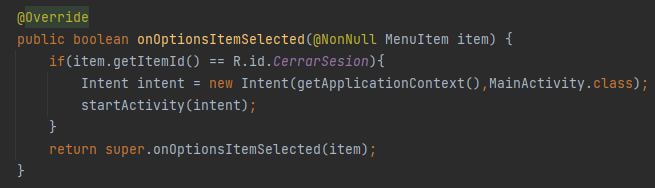


**Método getView():**

* Este método se encarga de crear y devolver la vista que se mostrará en cada elemento de la lista.
* Utiliza un patrón de vista reutilizable (ViewHolder) para mejorar el rendimiento al evitar la recreación de vistas innecesarias.
* Si la vista (convertView) es nula, infla el diseño de fila (R.layout.row\_list) y crea un nuevo objeto ViewHolder.
* Si la vista no es nula, recupera el ViewHolder de la etiqueta de la vista reutilizada.
* Obtiene los datos del catálogo para la posición actual desde la lista data.
* Decodifica la cadena Base64 de la imagen a un array de bytes y luego a un objeto Bitmap.
* Asigna los datos a los elementos de la vista (TextView y ImageView) utilizando el ViewHolder.
* Devuelve la vista actualizada.



**Cerrar sesión**

****

Esta será la opción que se encontrara en la esquina superior derecha una vez el usuario haya ingresado a la aplicación, al presionar esta opción los usuarios serán enviados a la pantalla de inicio de sesión.

# **Conclusiones**

Este proyecto fue desarrollado en Android Studio con el lenguaje de programación Java, el objetivo de este proyecto fue crear una aplicación de salón comunal que incluyera pantallas de inicio de sesión y registro, una pantalla de inicio, un mantenimiento de usuarios, un área de reservaciones y un catálogo. Todo esto conectado al backend del proyecto que fue trabajado en Asp.NET Core 7 en C#. De tal manera que lo usuarios creados en la aplicación se agregaran directamente al backend del proyecto de igual manera con las reservaciones creadas y con los paquetes creados en el catálogo.

# **Recomendaciones**

Aunque el proyecto funciono sin problemas el método que fue utilizado para su conexión con el backend no fue el más recomendable ya que fue realizado con un por lo tanto, lo mejor habría sido intentar implementar un mejor método que el que se utilizó para el desarrollo de este proyecto, sin embargo por falta de tiempo no se logro el cometido.