104. Proveedor de contenidos

Los proveedores de contenido en Android son básicamente una forma de compartir datos entre diferentes aplicaciones. Como un puente que permite a una app acceder a datos de otra app, siempre y cuando tenga los permisos necesarios. Por ejemplo, podrías acceder a las fotos del usuario que están gestionadas por la app de galería.

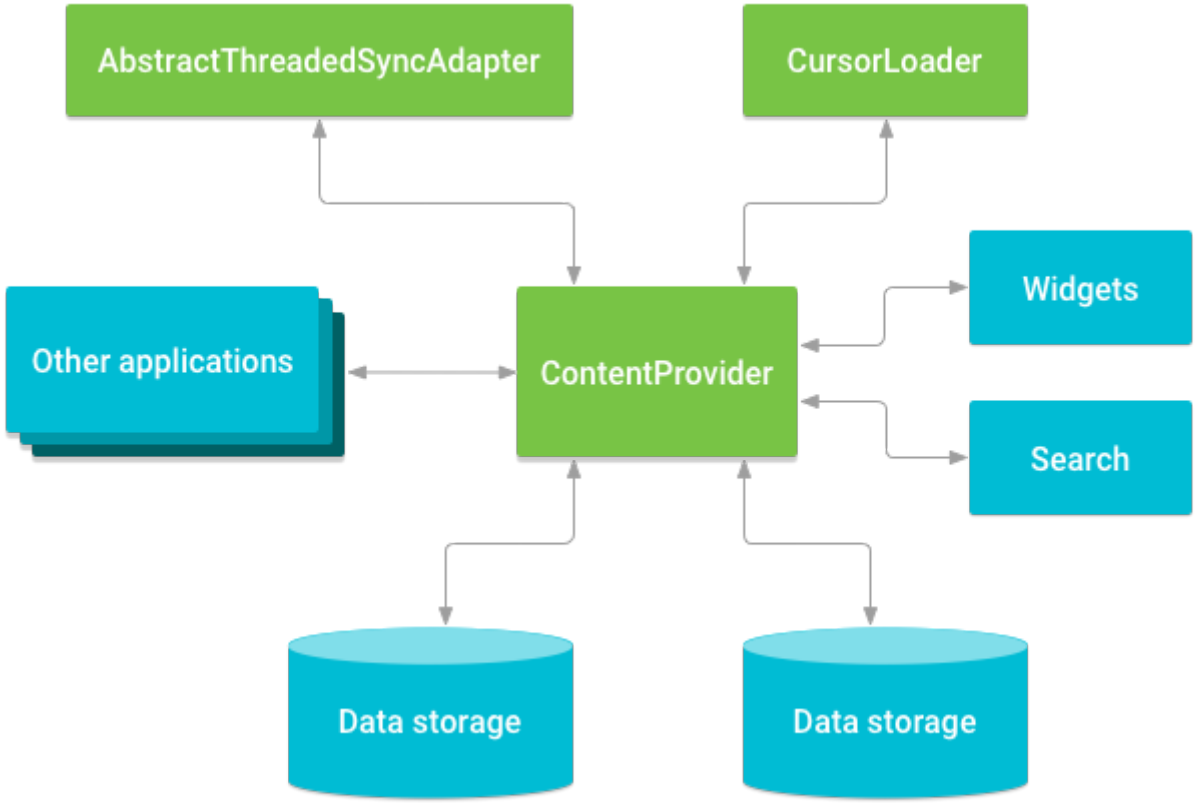
Ahora, cuando hablamos de Kotlin y Jetpack Compose no cambia la forma en que interactúas con los proveedores de contenido en Android.

104.1 Conceptos básicos de Proveedores de Contenido

Un proveedor de contenido presenta datos a aplicaciones externas en forma de una o más tablas que son similares a las tablas de una base de datos relacional. Una fila representa una instancia de algún tipo de datos que recopila el proveedor, y cada columna de la fila representa un ítem individual de los datos recopilados para una instancia.

El proveedor de contenido organiza el acceso a la capa de almacenamiento de los datos en tu aplicación para una serie de API y componentes diferentes e incluye lo siguiente:

- Compartir con otras aplicaciones el acceso a los datos de tu aplicación Enviar datos a un widget
- Mostrar sugerencias personalizadas de búsqueda para tu aplicación mediante el marco de trabajo de búsqueda usando SearchRecentSuggestionsProvider
- Sincronizar los datos de la aplicación con tu servidor mediante una * implementación de AbstractThreadedSyncAdapter
- Cargar datos en tu IU usando CursorLoader

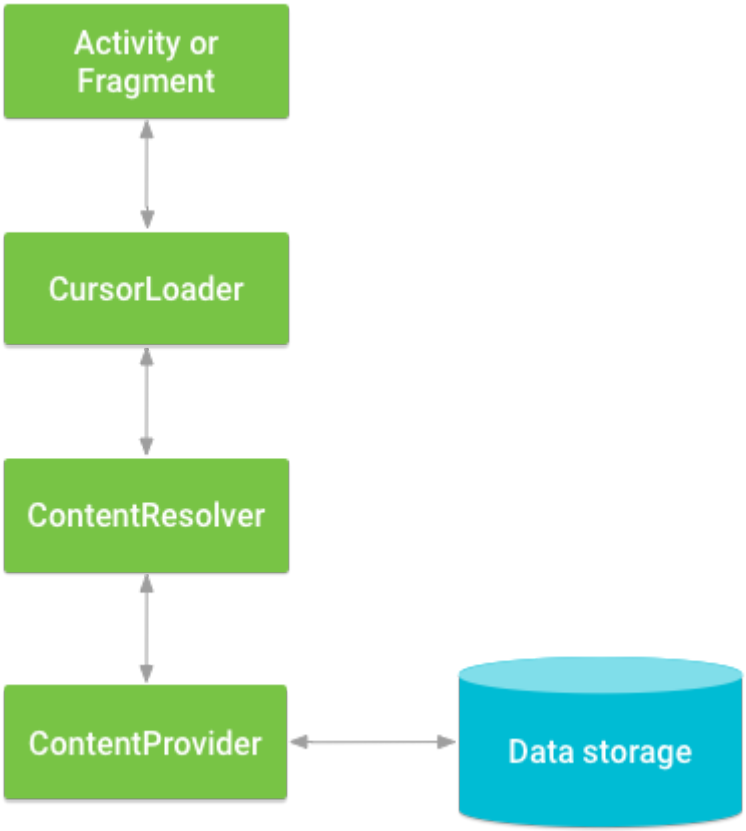


104.2 Acceder a un Proveedor de Contenidos

Los clientes se conectan al Proveedor de Contenidos usando el objeto ContentResolver en el Context de la aplicación. `ContentResolver` se comunica con una objeto de la clase que implemente el interfaz ContentProvider que se comunica con el objeto de datos, realiza la acción solicitada y muestra los resultados.

Los métodos que proporciona el `ContentResolver` son las funciones básicas "CRUD" (proveniente de los términos en inglés que equivalen a crear, recuperar, actualizar y borrar) del almacenamiento persistente.

Se suele acceder desde el IU al ContentProvider usando unCursorLoader que permite ejecutar una consulta asíncrona en segundo plano.
Un patrón común para acceder a un ContentProvider desde tu IU usa un para ejecutar una consulta asíncrona en segundo plano dejando disponible la IU



Ejemplo.

Uno de los proveedores integrados en la plataforma de Android es el diccionario del usuario, que guarda la ortografía de palabras no estándar que el usuario quiere conservar. En la tabla 1 se detalla cómo se verían los datos en la tabla de este proveedor:

palabra	ID de la app	frecuencia	configuración regional	_ID
mapreduce	user1	100	es-419	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5

Cada fila es el regitro de una palabra del diccionario del usuaro. La columna _ID cumple las funciones de clave primaria.

Si quieres obtener una lista de palabras y sus configuraciones regionales del proveedor de diccionario del usuario, debes llamar a `ContentResolver.query()`. El método query() llama al método `ContentProvider.query()` definido por el proveedor de diccionario del usuario. Las siguientes líneas de código muestran una llamada ContentResolver.query():

```
// Queries the user dictionary and returns results
cursor = contentResolver.query(
        UserDictionary.Words.CONTENT_URI,   // The content URI of the words table
        projection,                        // Las columnas para devolver en la consulta
        selectionClause,                   // Criterio de selección
        selectionArgs.toTypedArray(),      // Criterio de selección
        sortOrder                          // Orden de las filas devueltas
)
```

Argumento de query()	Palabra clave/parámetro SELECT	Notas	_ID
Uri	FROM table_name	Uri se asigna a la tabla del proveedor llamada table_name.	1
projection	col,col,col,...	projection es una matriz de columnas que se debe incluir para cada fila recuperada.	2
selection	WHERE col = value	selection especifica los criterios para seleccionar filas.	3
selectionArgs	(Sin equivalente exacto; los argumentos de selección reemplazan a los marcadores de posición ? en la cláusula de selección).	255	4
sortOrder	ORDER BY col,col,...	sortOrder especifica el orden en que aparece la fila en el Cursor que se muestra.	5

En los siguientes apartados se describe como obtener los datos del diccionario de usuario usando el Proveedor de Contenidos

104.2.1 URI Del Content Provider

Un URI de contenido es un URI que identifica datos de un proveedor. Los URI de contenido incluyen el nombre simbólico de todo el proveedor (su autoridad) y un nombre que apunta a una tabla (una ruta de acceso). Cuando llamas al método del cliente para acceder a una tabla del proveedor, el

URI de contenido de la tabla es uno de los argumentos.

En el ejemplo anterior la constante `UserDictionary.Words.CONTENT_URI` representa:

`content://user_dictionary/words`

104.2.2 Solicitar Permisos de acceso de lectura

Si se va a acceder a datos que requieren permisos (como contactos o fotos), hay que asegurarse de solicitar esos permisos en tu AndroidManifest.xml y en tiempo de ejecución.

El proveedor de diccionario del usuario define el permiso android.permission. `READ_USER_DICTIONARY` en su archivo de manifiesto, de modo que una aplicación que quiera leer desde el proveedor debe solicitar este permiso.

NOTA IMPORTANTE:

```
A partir de la versión 23 de la API de Android (Android 6.0 Marshmallow), hubo un cambio importante en cómo las aplicaciones pueden acceder al User Dictionary

Accesible Solo a través de IME y Corrector Ortográfico: IME significa "Input Method Editor" (Editor de Método de Entrada). Esto se refiere a las aplicaciones que proporcionan una forma de ingresar texto, como los teclados en pantalla. La nota indica que solo las aplicaciones que son IME (como los teclados) o los correctores ortográficos tienen permiso para acceder al User Dictionary de manera directa.

Implicaciones para los Desarrolladores: Si estás desarrollando una aplicación que no es un IME o un corrector ortográfico, no podrás acceder directamente al User Dictionary para leer o modificarlo a partir de la API 23.
```

```
<uses-permission android:name="android.permission.READ_USER_DICTIONARY"/>
```

104.2.3 Construir la consulta

Para poder usar el `ContentResolver` creamos todos los argumentos necesarios para la consulta.

Buscamos las referencias necesarias para usar el diccionario de usuario en Android Developer Referencia UserDictionary

Projection
Array con las columnas de la selección :

```
// A "projection" defines the columns that will be returned for each row
private val mProjection: Array<String> = arrayOf(
        UserDictionary.Words._ID,    // Contract class constant for the _ID column name
        UserDictionary.Words.WORD,   // Contract class constant for the word column name
        UserDictionary.Words.LOCALE  // Contract class constant for the locale column name
)

// Defines a string to contain the selection clause
private var selectionClause: String? = null

// Declares an array to contain selection arguments
private lateinit var selectionArgs: Array<String>
```

La consulta es equivalente a la siguiente sentencia SQL:

```
SELECT _ID, word, locale FROM words WHERE word = <userinput> ORDER BY word ASC;
```

Y utilizamos:

1. `ContentResolver.query()` que necesita:
2. Proyección (array de columnas) anterior
3. Expresión de busqueda formado por
4. *clausula de seleccion* , expresión lógica con nombres de columnas valores (la variable mSelectionClause) Si especificas el parámetro reemplazable `?` en lugar de un valor, el método de consulta recupera el valor de la matriz de argumentos de selección (la variable

mSelectionArgs).

5. *argumentos de selección*
6. Usamos el Cursor para leer la respuesta. Tendremos en cuenta que en algunos casos el contenido puede ser `null`

```
/*
 * This declares String array to contain the selection arguments.
 */
private lateinit var selectionArgs: Array<String>


// suponemos que searchWord contiene la palabra que buscamos en
searchString = "electroencefalografista"

// Remember to insert code here to check for invalid or malicious input.

// Si la palabra es nula se obtienen todas
// Observar que inicializamos selectionArgs y selectionClause
// UserDictionary.Words.WORD es una cte
selectionArgs = searchString?.takeIf { it.isNotEmpty() }?.let {
    selectionClause = "${UserDictionary.Words.WORD} = ?"
    arrayOf(it)
} ?: run {
    selectionClause = null
    emptyArray<String>()
}


// Realiza la llavalmada y se devuelve un Cursor
val mCursor = contexto.contentResolver.query(
        UserDictionary.Words.CONTENT_URI,  // The content URI of the words table
        mProjection,                        // The columns to return for each row
        selectionClause,                   // null o la palabra buscada
        selectionArgs,                     // Either empty, or the string the user entered
        sortOrder                          // The sort order for the returned rows
)
```

```
// Some providers return null if an error occurs, others throw an exception
// con el comentario anterior , deberíamos capturar las excepciones en la
// sentencia anterior
when (mCursor?.count) {
    null -> {
        /*
         * Insert code here to handle the error. Be sure not to use the cursor!
         * You may want to call android.util.Log.e() to log this error.
         *
         */
    }
    0 -> {
        /*
         * Insert code here to notify the user that the search was unsuccessful. This isn't
         * necessarily an error. You may want to offer the user the option to insert a new
         * row, or re-type the search term.
         */
    }
    else -> {
        // Insert code here to do something with the results
    }
}
```

104.2.4 Insertar , actualizar y borrar datos

104.2.4.1 Insertar

Para insertar datos en un proveedor, debes llamar al método ContentResolver.insert(). Este método inserta una nueva fila en el proveedor y devuelve un URI de contenido para esa fila. Este fragmento de código muestra cómo insertar una nueva palabra en el proveedor de diccionario del usuario:

```
// Defines a new Uri object that receives the result of the insertion
lateinit var newUri: Uri
```

```
...

// Defines an object to contain the new values to insert
val newValues = ContentValues().apply {
    /*
     * Sets the values of each column and inserts the word. The arguments to the "put"
     * method are "column name" and "value"
     */
    put(UserDictionary.Words.APP_ID, "example.user")
    put(UserDictionary.Words.LOCALE, "en_US")
    put(UserDictionary.Words.WORD, "insert")
    put(UserDictionary.Words.FREQUENCY, "100")

}

newUri = contentResolver.insert(
        UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
        newValues                          // the values to insert
)
```

El fragmento no agrega la columna _ID, ya que esta columna se mantiene automáticamente. El proveedor asigna un valor único de _ID a cada fila que se agrega. Por lo general, los proveedores usan este valor como la clave primaria de la tabla.

El URI de contenido que se muestra en newUri identifica la fila nueva con el siguiente formato:

```
content://user_dictionary/words/<id_value>
```

Para obtener el valor de _ID del Uri que se muestra, llama a ContentUris.parseId().

104.2.4.2 Actualizar datos

Para actualizar una fila, debes usar un objeto `ContentValues` con los valores actualizados, tal como lo haces con una inserción, y con criterios de selección como lo haces con una consulta. El método de cliente que usas es `ContentResolver.update()`. Solo debes agregar valores al objeto ContentValues para las columnas que estés actualizando. Si quieres borrar el contenido de una columna, establece el valor en null.

```
// Defines an object to contain the updated values
val updateValues = ContentValues().apply {
    /*
     * Sets the updated value and updates the selected words.
     */
    putNull(UserDictionary.Words.LOCALE)
}

// Defines selection criteria for the rows you want to update
val selectionClause: String = UserDictionary.Words.LOCALE + "LIKE ?"
val selectionArgs: Array<String> = arrayOf("en_%")

// Defines a variable to contain the number of updated rows
var rowsUpdated: Int = 0

...

rowsUpdated = contentResolver.update(
        UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
        updateValues,                      // the columns to update
        selectionClause,                   // the column to select on
        selectionArgs                      // the value to compare to
)
```

104.2.4.3 BORRAR DATOS

Borrar filas es similar a recuperar datos de una fila: debes especificar criterios de selección para las filas que quieras borrar y el método de cliente muestra el número de filas borradas. El siguiente fragmento borra las filas cuyo ID de aplicación coincide con "user". El método muestra el número de filas borradas.

```
// Defines selection criteria for the rows you want to delete
val selectionClause = "${UserDictionary.Words.LOCALE} LIKE ?"
val selectionArgs: Array<String> = arrayOf("user")
```

```
// Defines a variable to contain the number of rows deleted
var rowsDeleted: Int = 0

...

// Deletes the words that match the selection criteria
rowsDeleted = contentResolver.delete(
        UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
        selectionClause,                    // the column to select on
        selectionArgs                       // the value to compare to
)
```

Cuando llamas a `ContentResolver.delete()`, también debes depurar las entradas del usuario. Para obtener más información, consulta la sección Protección contra entradas malintencionadas.

104.2.5 Tipo de datos del proveedor

Los proveedores de contenido pueden ofrecer muchos tipos de datos diferentes. El proveedor de diccionario del usuario solo ofrece texto, pero en general los proveedores también pueden ofrecer los siguientes formatos:

- número entero
- entero largo (largo)
- punto flotante
- punto flotante largo (doble)
- Otro tipo de datos que los proveedores usan con frecuencia es el objeto binario grande (`BLOB`) implementado como una matriz de 64 KB. Si quieres ver los tipos de datos disponibles, puedes ver los métodos GET de la clase Cursor.

El tipo de datos para cada columna de un proveedor suele indicarse en su documentación. Los tipos de datos para el proveedor de diccionario del usuario se indican en la documentación de referencia para su clase de contratos UserDictionary.Words (las clases de contratos se describen en la sección Clases de contratos). También puedes determinar el tipo de datos llamando a Cursor.getType().

Los proveedores también mantienen información del tipo de datos `MIME` para cada URI de contenido que definen. Usa la información del tipo de MIME para averiguar si tu aplicación puede administrar datos que ofrece el proveedor o seleccionar un tipo de administración en función del tipo de MIME. Generalmente necesitas el tipo de MIME cuando trabajas con un proveedor que contiene estructuras de datos o archivos complejos. Por ejemplo, la tabla ContactsContract.Data del proveedor de contactos usa tipos de MIME para etiquetar el tipo de datos de contacto guardado en cada fila. Para obtener el tipo de MIME correspondiente a un URI de contenido, llama a ContentResolver.getType().

La sección Referencia a tipos de MIME describe la sintaxis de los tipos de MIME estándar y personalizado.

104.3 Como crear un Proveedor de contenidos

https://developer.android.com/guide/topics/providers/content-provider-creating?hl=es-419

104.4 Pasos para trabajar con Proveedores de contenido:

1. Definir un **Uri**: Este es un identificador único que apunta a los datos que quieres en el proveedor de contenido.
2. Solicitar **Permisos**: Si se va a acceder a datos que requieren permisos (como contactos o fotos), hay que asegurarse de solicitar esos permisos en tu AndroidManifest.xml y en tiempo de ejecución.
3. Usar **ContentResolver**: En Kotlin, utilizarás un ContentResolver para consultar, insertar, actualizar o eliminar datos en un proveedor de contenido. El ContentResolver gestiona tu solicitud y comunica con el proveedor de contenido apropiado.
4. Integrar con Compose: Aunque la interacción con el proveedor de contenido se hace en Kotlin, puedes fácilmente integrar los datos obtenidos en tu UI Compose. Por ejemplo, puedes tener un @Composable que muestra una lista de contactos obtenidos de un proveedor de contenido.
5. Manejo de Datos: Trabaja con los datos obtenidos (Cursor) y adáptalos según sea necesario para mostrarlos en tu UI Compose.

104.5 Ejemplo Proveedor de Libros:

Vamos a crear un módulo que lleva el control de los libros de una biblioteca. Queremos que otras aplicaciones puedan acceder a esta información, pero de manera controlada y segura. Aquí es donde entra en juego el proveedor de contenidos

1. Creamos el proveedor de contenidos en nuestra aplicación

```
class LibroProvider : ContentProvider() {
    override fun onCreate(): Boolean {
        // Inicializar el proveedor
        return true
    }

    // Implementar los métodos query, insert, delete, y update
    override fun query(
        uri: Uri,
        projection: Array<String>?,
        selection: String?,
        selectionArgs: Array<String>?,
        sortOrder: String?
    ): Cursor? {
        // Lógica para consultar los libros
        return null
    }

    // Los otros métodos se implementan de forma similar
}
```

1. Registramos el proveedor de contenidos en `AndroidManifest.xml`

```
<provider
    android:name=".LibroProvider"
    android:authorities="com.miapp.biblioteca"
    android:exported="true"/>
```

1. Acceder al Proveedor de Contenidos Desde tu código Kotlin (por ejemplo, en tu ViewModel o en alguna función de tu UI Compose), puedes acceder al proveedor de contenidos para obtener los datos.

```
val uri = Uri.parse("content://com.miapp.biblioteca/books")
val cursor = context.contentResolver.query(uri, null, null, null, null)
// Convertir el cursor a una lista de libros y actualizar la UI
```

104.6 Apendice

Enlaces

- Conceptos básicos Content Provider
- Proveedor de calendario
- Proveedor de contactos
- Revisar tutorial lista de contactos
- REVISAR
- https://github.com/mtuenaydin/ContactViewModel/tree/master
-

Versión 0.5 18-12-23 Versión 0.8 7-1-24

¿Fue útil esta página?

☺ ☹