Documentación

Proyecto agenda

Iván Gallego

Índice de contenidos

> Todas los accesos a base de datos se hacen desde `MainActivity`
>
> El código no está completo, omitido por brevedad

Base de datos

La clase de base de datos inicializa una base de datos si no existe.

```
public class BDContactos extends SQLiteOpenHelper {
    private String statement = "CREATE TABLE IF NOT EXISTS contactos(" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "nombre TEXT," +
            "apellido TEXT," +
            "telefono TEXT," +
            "correo TEXT," +
            "imagen BLOB," +
            "amigo INTEGER," +
            "familia INTEGER," +
            "trabajo INTEGER" +
            ")";

    public BDContactos(@Nullable Context context, @Nullable String name,
            @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(statement);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

Accesos a base de datos desde `MainActivity`

1. Al filtrar un dato, se hace una consulta con un `WHERE`, para crear un cursor nuevo y actualizar el recycler
2. Al elimnar un contacto, se hace un `DELETE`
3. Al editar un contacto se hace un `UPDATE`
4. Al cargar los datos al inicio se hace un `SELECT`
5. Al guardar un contacto se hace un `INSERT`

```
public class MainActivity extends AppCompatActivity {
    private BDContactos dbContactos;
    private DrawerLayout drawerLayout;
    private Layout layout;
    private UpdateViewModel updateViewModel;
    private Contacto contactoTemp;
    private Cursor cursor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        dbContactos = new BDContactos(this, "DBCONTACTOS", null, 1);
        cargarDatos();

        NavigationView navigationView = findViewById(R.id.navigation_view);
        navigationView.setNavigationItemSelectedListener(
                new NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                SQLiteDatabase readableDatabase = dbContactos.getReadableDatabase();
                switch (menuItem.getItemId()) {
                    case R.id.familia_menu_option:
                        cursor = readableDatabase.rawQuery
                        ("SELECT * FROM contactos WHERE familia = ?", new String[]{"1"});
                        break;
                    case R.id.amigos_menu_option:
                        cursor = readableDatabase.rawQuery
                        ("SELECT * FROM contactos WHERE amigo = ?", new String[]{"1"});
                        break;
                    case R.id.trabajo_menu_option:
                        cursor = readableDatabase.rawQuery
                        ("SELECT * FROM contactos WHERE trabajo = ?", new String[]{"1"});
                        break;
                    default:
                        cursor = readableDatabase.rawQuery
                        ("SELECT * FROM contactos", null);
                        break;
                }
                updateViewModel.setData(new UpdateContainer(layout, cursor));
                menuItem.setChecked(true);
                drawerLayout.closeDrawers();
                return true;
            }
        });
    }

    private void eliminarContacto(final Contacto contacto) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("¿Quieres eliminar el contacto de " + contacto.getNombre() + "?");
        builder.setPositiveButton("ELIMINAR", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                if (dbContactos != null) {
                    SQLiteDatabase contactosDatabase = dbContactos.getWritableDatabase();
                    contactosDatabase.delete("contactos", "id = ?",
                        new String[] {String.valueOf(contacto.getId())});
                    contactosDatabase.close();
                    actualizarDatos();
                }
            }
        });
        builder.setNegativeButton("CANCELAR", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
```

```
            }
        });
        builder.create().show();
    }

    private void actualizarDatos(){
        cargarDatos();
        updateViewModel.setData(new UpdateContainer(layout, cursor));
    }

    private void cargarDatos() {
        if (dbContactos != null) {
            SQLiteDatabase writableDatabase = dbContactos.getReadableDatabase();
            cursor = writableDatabase.rawQuery("SELECT * FROM contactos", null);
        }
    }

    private void editarContactoDatabase(Contacto editado, Contacto nuevo) {
        SQLiteDatabase writableDatabase = dbContactos.getWritableDatabase();
        SQLiteDatabase readableDatabase = dbContactos.getReadableDatabase();
        long count = DatabaseUtils.queryNumEntries(readableDatabase,
            "contactos", "id = ?", new String[]{String.valueOf(editado.getId())});
        if (count == 1){
            ContentValues values = new ContentValues();
            values.put("nombre", nuevo.getNombre());
            values.put("apellido", nuevo.getApellido());
            values.put("telefono", nuevo.getTelefono());
            values.put("correo", nuevo.getCorreo());
            Bitmap b = nuevo.getImagen();
            byte[] bytesImagen = null;
            if (b != null) {
                bytesImagen = convertirImagenBytes(nuevo.getImagen());
            }
            values.put("imagen", bytesImagen);
            values.put("amigo", nuevo.isAmigo());
            values.put("trabajo", nuevo.isTrabajo());
            values.put("familia", nuevo.isFamilia());
            writableDatabase.update("contactos", values, "id = ?",
                new String[]{String.valueOf(editado.getId())});
        }
        writableDatabase.close();
        readableDatabase.close();
    }

    private void guardarContacto(Contacto c) {
        SQLiteDatabase writableDatabase = dbContactos.getWritableDatabase();
        SQLiteDatabase readableDatabase = dbContactos.getReadableDatabase();
        ContentValues values = new ContentValues();
        values.put("nombre", c.getNombre());
        values.put("apellido", c.getApellido());
        values.put("telefono", c.getTelefono());
        values.put("correo", c.getCorreo());
        Bitmap b = c.getImagen();
        byte[] bytesImagen = null;
        if (b != null) {
            bytesImagen = convertirImagenBytes(c.getImagen());
        }
        values.put("imagen", bytesImagen);
        values.put("amigo", c.isAmigo());
        values.put("trabajo", c.isTrabajo());
        values.put("familia", c.isFamilia());
        writableDatabase.insert("contactos", null, values);
        writableDatabase.close();
        readableDatabase.close();
    }
}
```

Al hacer el filtro, en el `WHERE` le indice que, dependiendo del filtro, el campo a buscar de `trabajo`, `amigo` o `familia`; debe ser `1` porque así se manejan los tipos `boolean` en SQLite

Cada vez que se hace una consulta se contruye (si se ha de sustituir el fragment de `AccionContacto` por el de `VistaContactos`) o se notifica a `VistaContactos` a través del objeto `UpdateViewModel` (si no hay que sustituir ningún fragment, solo actualizar los datos).

CursorRecyclerAdapter

Para crear el adaptador del `RecyclerView` lo hago a través de dos clases, una abstracta y otra que extiende de ella.

```
public abstract class AbsAdaptadorCursorRecycler extends RecyclerView.Adapter {

    private Cursor cursor;

    public AbsAdaptadorCursorRecycler(Cursor cursor) {
        this.cursor = cursor;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
        if (cursor == null) throw new IllegalStateException("Error cursor vacío");
        if (!cursor.moveToPosition(position)) throw new IllegalStateException("Error no se puede encontrar la
posición " + position);
        onBindViewHolder(holder, cursor);
    }

    public abstract void onBindViewHolder(RecyclerView.ViewHolder holder, Cursor cursor);

    @Override
    public int getItemCount() {
        if (cursor != null) return cursor.getCount();
        else return 0;
    }

    @Override
    public long getItemId(int position) {
        if (hasStableIds() && cursor != null) {
            if (cursor.moveToPosition(position)) {
                return cursor.getLong(cursor.getColumnIndexOrThrow("id"));
            }
        }
        return RecyclerView.NO_ID;
    }

    public Cursor getCursor() {
        return cursor;
    }
}
```

`Adaptador` extiende de `AbsAdaptadorCursorRecycler` y es como el adaptador de versiones anteriores de la agenda pero ahora se le incluye un cursor con los datos a cargar. Ahora el método bind se hace con el `Cursor`

```
public class AdaptadorCursorRecycler extends AbsAdaptadorCursorRecycler
        implements View.OnClickListener, View.OnLongClickListener, View.OnTouchListener {
    private View.OnClickListener clickListener;
    private OnImageClickListener imageClickListener;
    private View.OnLongClickListener longClickListener;
    private View.OnTouchListener touchListener;
    private Layout layout;

    public AdaptadorCursorRecycler(Cursor c, Layout layout) {
```

```
        super(c);
        this.layout = layout;
    }

    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        final View v;
        if (layout == Layout.GRID) {
            v = LayoutInflater.from(parent.getContext()).inflate(R.layout.entrada_agenda_compact, parent,
false);
            v.setOnLongClickListener(this);
            v.setOnClickListener(this);
            v.setOnTouchListener(this);
            HolderCompact h = new HolderCompact(v);
            h.setImageClickListener(new OnImageClickListener() {
                @Override
                public void onImageClick(Contacto contacto, View view) {
                    imageClickListener.onImageClick(contacto, v);
                }
            });
            return h;
        } else {
            v = LayoutInflater.from(parent.getContext()).inflate(R.layout.entrada_agenda, parent, false);
            v.setOnLongClickListener(this);
            v.setOnClickListener(this);
            v.setOnTouchListener(this);
            Holder h = new Holder(v);
            h.setImageClickListener(new OnImageClickListener() {
                @Override
                public void onImageClick(Contacto contacto, View view) {
                    imageClickListener.onImageClick(contacto, v);
                }
            });
            return h;
        }
    }

    private Contacto getContactoFromCursor(Cursor cursor) {
        Contacto c = new Contacto();
        c.setId(cursor.getInt(0));
        c.setNombre(cursor.getString(1));
        c.setApellido(cursor.getString(2));
        c.setTelefono(cursor.getString(3));
        c.setCorreo(cursor.getString(4));
        c.setImagen(Util.convertirBytesBitmap(cursor.getBlob(5)));
        c.setAmigo(cursor.getInt(6) == 1);
        c.setFamilia(cursor.getInt(7) == 1);
        c.setTrabajo(cursor.getInt(8) == 1);
        return c;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, Cursor cursor) {
        if (layout == Layout.GRID) {
            ((HolderCompact) holder).bind(getContactoFromCursor(cursor));
        } else {
            ((Holder) holder).bind(getContactoFromCursor(cursor));
        }
    }

    public void setOnClickListener(View.OnClickListener listener) {
        if (listener != null) {
            this.clickListener = listener;
        }
    }
```

```
    @Override
    public void onClick(View v) {
        if (clickListener != null) {
            clickListener.onClick(v);
        }
    }

    public void setOnLongClickListener(View.OnLongClickListener listener) {
        if (listener != null) {
            this.longClickListener = listener;
        }
    }

    @Override
    public boolean onLongClick(View v) {
        if (longClickListener != null) {
            longClickListener.onLongClick(v);
        }
        return false;
    }

    public void setOnTouchListener(View.OnTouchListener listener) {
        if (listener != null) {
            this.touchListener = listener;
        }
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (touchListener != null) {
            touchListener.onTouch(v, event);
        }
        return false;
    }

    public void setImageClickListener(OnImageClickListener listener) {
        if (listener != null) {
            imageClickListener = listener;
        }
    }

    public Contacto getItem(int position) {
        getCursor().moveToPosition(position);
        return getContactoFromCursor(getCursor());
    }
}
```