Documentación

Proyecto agenda

Iván Gallego

Índice de contenidos

1. Comunicación de contactos
2. Comunicación de actualización

Comunicación de contactos

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    private ContactoViewModel contactoViewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        contactoViewModel = ViewModelProviders.of(this).get(ContactoViewModel.class);
        contactoViewModel.getData().observe(this, new Observer<ContactoContainer>() {
            @Override
            public void onChanged(ContactoContainer contacto) {
                switch (contacto.getAccion()) {
                    case CLICK:
                        itemClick(contacto.getContacto());
                        break;
                    case EDITAR:
                        editContact(contacto.getContacto());
                        break;
                    case ADD:
                        addContact(contacto);
                        break;
                    case FAB_CLICK:
                        fabClicked();
                        break;
                    case ELIMINAR:
                        eliminarContacto(contacto.getContacto());
                        break;
                    case IMAGE_CLICK:
                        contactoTemp = contacto.getContacto();
                        break;
                }
            }
        });
    }
}
```

Con este código, `MainActivity` reaccionará cuando se le notifique por medio del objeto `contactoViewModel` y realizará las acciones necesarias dependiendo de la acción.

Cuando se le notifica algo, se le pasa también un objeto, este objeto es de tipo `ContactoContainer`, que contiene un objeto de tipo `Contacto` y un enumerado de tipo `Accion`.

ContactoContainer

```
public class ContactoContainer implements Parcelable {
    private Contacto contacto;
    private Util.Accion accion;
```

```
    public ContactoContainer(Contacto contacto, Util.Accion accion) {
        this.contacto = contacto;
        this.accion = accion;
    }

    public ContactoContainer() {
    }

    public Contacto getContacto() {
        return contacto;
    }

    public void setContacto(Contacto contacto) {
        this.contacto = contacto;
    }

    public Util.Accion getAccion() {
        return accion;
    }

    public void setAccion(Util.Accion accion) {
        this.accion = accion;
    }


    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeParcelable(this.contacto, flags);
        dest.writeInt(this.accion == null ? -1 : this.accion.ordinal());
    }

    protected ContactoContainer(Parcel in) {
        this.contacto = in.readParcelable(Contacto.class.getClassLoader());
        int tmpAccion = in.readInt();
        this.accion = tmpAccion == -1 ? null : Util.Accion.values()[tmpAccion];
    }

    public static final Parcelable.Creator<ContactoContainer> CREATOR =
            new Parcelable.Creator<ContactoContainer>() {
        @Override
        public ContactoContainer createFromParcel(Parcel source) {
            return new ContactoContainer(source);
        }

        @Override
        public ContactoContainer[] newArray(int size) {
            return new ContactoContainer[size];
        }
    };
}
```

ContactoViewModel

```
public class ContactoViewModel extends ViewModel {
    private MutableLiveData<ContactoContainer> liveData;

    public LiveData<ContactoContainer> getData(){
        if (liveData != null) {
```

```
            setData(liveData.getValue());
        } else {
            liveData = new MutableLiveData<>();
        }
        return liveData;
    }

    public void setData(ContactoContainer contacto) {
        if (liveData == null) {
            liveData = new MutableLiveData<>();
        } else {
            liveData.setValue(contacto);
        }
    }
}
```

Uso y notificación desde otros fragments

Desde VistaContactos

1. Al inicalizarlo
2. Al hacer click sobre una imagen para cambiarla, eliminarla, etc...
3. Al borrar un contacto
4. Al borrar un imagen
5. Al hacer click sobre un contacto para editarlo

```
public class VistaContactos extends Fragment implements View.OnClickListener {
    ...
    private ContactoViewModel contactoViewModel;
    ...

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
            @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View rootView = inflater.inflate(R.layout.vista_contactos, container, false);
        ...
        contactoViewModel = ViewModelProviders.of(getActivity()).get(ContactoViewModel.class);
        adaptador.setImageClickListener(new OnImageClickListener() {
            @Override
            public void onImageClick(final Contacto contacto, View v) {
                mostrarPopupMenu(v);
                contactoViewModel.setData(new ContactoContainer(contacto, Util.Accion.IMAGE_CLICK));
            }
        });
        ...
    }

    private void mostrarPopupMenu(View v) {
        PopupMenu pop = new PopupMenu(getContext(), v);
        final Contacto contacto = adaptador.getItem(recyclerView.getChildAdapterPosition(v));
        indiceListaPulsado = recyclerView.getChildAdapterPosition(v);
        pop.getMenuInflater().inflate(R.menu.menu_foto, pop.getMenu());
        pop.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.camera:
                        tomarFoto();
                        break;
                    case R.id.galeria:
                        elegirImagen();
                        break;
```

```
                    case R.id.borrar:
                        contacto.setImagen(null);
                        contactoViewModel.setData(
                            new ContactoContainer(contacto, Util.Accion.EDITAR));
                        recyclerView.setAdapter(adaptador);
                        break;
                }
                return true;
            }
        });
        pop.show();
    }

    private void eliminarContacto(View v) {
        contactoViewModel.setData(new ContactoContainer(
            adaptador.getItem(recyclerView.getChildAdapterPosition(v)), Util.Accion.ELIMINAR));
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        Contacto c = adaptador.getItem(indiceListaPulsado);
        if (requestCode == COD_ELEGIR_IMAGEN && resultCode == RESULT_OK && data != null) {
            Uri rutaImagen = data.getData();
            c.setImagen(bitmapFromUri(rutaImagen, getContext()));
            contactoViewModel.setData(new ContactoContainer(c, Util.Accion.EDITAR));
        } else if (requestCode == COD_TOMAR_FOTO && resultCode == RESULT_OK && data != null) {
            c.setImagen((Bitmap) data.getExtras().get("data"));
            contactoViewModel.setData(new ContactoContainer(c, Util.Accion.EDITAR));
        } else if (resultCode == RESULT_CANCELED) {
            Toast.makeText(getContext(), "Se ha cancelado la operación", Toast.LENGTH_LONG).show();
        }
        recyclerView.setAdapter(adaptador);
    }

    @Override
    public void onClick(View v) {
        indiceListaPulsado = recyclerView.getChildAdapterPosition(v);
        Contacto contacto = adaptador.getItem(recyclerView.getChildAdapterPosition(v));
        if (swipeDetector.swipeDetected()) {
            switch (swipeDetector.getAction()) {
                case LR:
                    llamarContacto(contacto);
                    break;
                case RL:
                    enviarMensaje(contacto);
                    break;
            }
        } else {
            contactoViewModel.setData(new ContactoContainer(contacto, Util.Accion.CLICK));
        }
    }
}
```

Desde AccionContacto

1. Para hacer alguna acción sobre una imagen
2. Para editar o generar un nuevo contacto

```
public class AccionContacto extends Fragment implements View.OnClickListener {
    private ContactoViewModel viewModel;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
```

```
            @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View v = inflater.inflate(R.layout.accion_contacto, container, false);
        Bundle args = getArguments();

        viewModel = ViewModelProviders.of(getActivity()).get(ContactoViewModel.class);

        imageView = v.findViewById(R.id.profile_image_view);
        imageView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                PopupMenu pop = new PopupMenu(getContext(), v);
                pop.getMenuInflater().inflate(R.menu.menu_foto, pop.getMenu());
                pop.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        switch (item.getItemId()) {
                            case R.id.camera:
                                tomarFoto();
                                break;
                            case R.id.galeria:
                                elegirImagen();
                                break;
                            case R.id.borrar:
                                bitmap = null;
                                contacto.setImagen(bitmap);
                                imageView.setImageBitmap(bitmap);
                                break;
                        }
                        return true;
                    }
                });
                pop.show();
            }
        });
    }

    @Override
    public void onClick(View v) {
        if (nombreCorrecto(editNombre.getText().toString())) {
            if (accion == Util.Accion.EDITAR) {
                editarContacto();
                viewModel.setData(new ContactoContainer(contacto, Util.Accion.EDITAR));
            } else {
                viewModel.setData(new ContactoContainer(generarNuevoContacto(), Util.Accion.ADD));
            }
        }
    }
}
```

Comunicación de actualización

MainActivity

La comunicación de actualización la hago para cuando se produzca algún cambio en los datos de la base de datos, se le notifique a `VistaContactos`, para no tener que construirlo de nuevo , este caso es al contrario que el anterior, ya que el escuchador se coloca esta vez en el `VistaContactos` y se notifica desde `MainActivity`. La finalidad principal es que se actualice el recycler con los datos nuevos o layout nuevo

Lo utilizo en las siguientes situaciones:

1. Cuando hago una nueva consulta de filtrado de datos, le notifico que ha habido cambios y le paso el cursor con los datos actualizados
2. Al inicializarlo
3. Al editar un contacto
4. Al eliminar
5. Al cambiar el tipo de layout

```
public class MainActivity extends AppCompatActivity {
    private UpdateViewModel updateViewModel;

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

        updateViewModel = ViewModelProviders.of(this).get(UpdateViewModel.class);

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

    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();
    }

    public void editContact(Contacto contacto) {
        editarContactoDatabase(contactoTemp, contacto);
        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction transaction = fragmentManager.beginTransaction();
        actualizarDatos();
        VistaContactos fragment = new VistaContactos(layout, cursor);
        transaction.replace(R.id.fragment_container, fragment);
        transaction.addToBackStack(null);
        transaction.commit();
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();
        switch (id) {
            case android.R.id.home:
                drawerLayout.openDrawer(GravityCompat.START);
                break;
            case R.id.linear_option_menu:
                layout = Layout.LINEAR;
                updateViewModel.setData(new UpdateContainer(layout, cursor));
                break;
            case R.id.grid_option_menu:
                layout = Layout.GRID;
                updateViewModel.setData(new UpdateContainer(layout, cursor));
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

VistaContactos

En `VistaContactos` simplemente se escucha cuando se notifique un cambio y se realizan las acciones correspondientes a la acción que se comunica en el objeto `UpdateContainer`

```
public class VistaContactos extends Fragment implements View.OnClickListener {
    private UpdateViewModel updateViewModel;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
        @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View rootView = inflater.inflate(R.layout.vista_contactos, container, false);

        updateViewModel = ViewModelProviders.of(getActivity()).get(UpdateViewModel.class);

        updateViewModel.getData().observe(this, new Observer<UpdateContainer>() {
            @Override
            public void onChanged(UpdateContainer container) {
                layout = container.getLayout();
                cursor = container.getCursor();
                updateRecycler();
            }
        });
    }
}
```

UpdateContainer

Este objeto solo contiene un objeto de tipo enumerado `Layout` y un objeto de tipo `Cursor` con los datos para actualizar el recycler

```
public class UpdateContainer {
    private Util.Layout layout;
    private Cursor cursor;

    public UpdateContainer(Util.Layout layout, Cursor cursor) {
        this.layout = layout;
        this.cursor = cursor;
    }

    public UpdateContainer() {
    }

    public Util.Layout getLayout() {
        return layout;
    }

    public void setLayout(Util.Layout layout) {
        this.layout = layout;
    }

    public Cursor getCursor() {
        return cursor;
    }

    public void setCursor(Cursor cursor) {
        this.cursor = cursor;
    }
}
```

UpdateViewModel

```
public class UpdateViewModel extends ViewModel {
    private MutableLiveData<UpdateContainer> liveData;

    public LiveData<UpdateContainer> getData(){
        if (liveData != null) {
            setData(liveData.getValue());
        } else {
            liveData = new MutableLiveData<>();
        }
        return liveData;
    }

    public void setData(UpdateContainer updateContainer) {
        if (liveData == null) {
            liveData = new MutableLiveData<>();
        } else {
            liveData.setValue(updateContainer);
        }
    }
}
```