9. Argumentos en Fragments

En este capítulo se verá como podemos mostrar en la Actividad de detalle del Crimen la información de este, y si ésta es modificada, también se verá modificada en la lista de todos los crímenes.

La CrimeActivity será iniciada mediante un Fragment en vez de generar una nueva Activity.

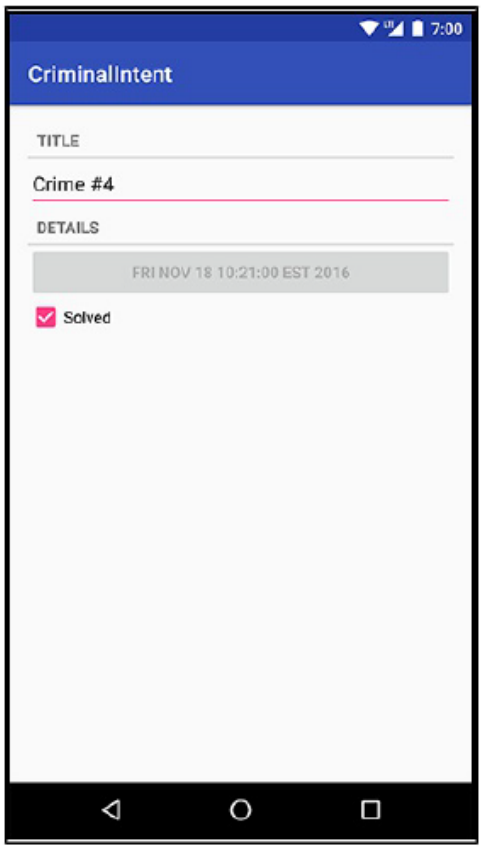
Veremos que, en el CrimeHolder de la lista de crímenes, ahora no hacemos un Toast, sino que generamos un Intent con la CrimeActivity.

Haremos ahora que la CrimeActivity herede de la clase SingleFragmentActivity y que la función newIntent genere la nueva CrimeActivity con la información del crimen especificado por el parámetro UUID crimeId.

Tras crear el Intent, deberemos pasarle la información mediante la función PutExtra.

Para poder mostrar esa información en la vista, deberemos modificar el método onCreateView de la clase CrimeFragment, y añadir que el texto de los campos sea el del contenido del objeto Crime que tiene asociado. De la misma forma lo haremos con el checkbox y con todos los campos que sean visibles.

La vista de la aplicación cuando cliquemos en un crimen, tiene el siguiente aspecto.



Cada instancia de Fragment puede tener un Bundle asociado a él, que contiene pares de clave-valor a modo igual que lo hacen los Extras del Intent. Cada par de clave-valor son llamados argumentos.

```
Bundle args = new Bundle();
args.putSerializable(ARG_MY_OBJECT, myObject);
args.putInt(ARG_MY_INT, myInt);
args.putCharSequence(ARG_MY_STRING, myString);
```

Para poder utilizar este método de paso de argumentos a un Fragment, modificaremos la case CrimeFragment, añadiendo un identificador estático con el que identificar el valor que añadiremos al Bundle.

```
private static final String ARG_CRIME_ID = "crime_id";

private Crime mCrime;
private EditText mTitleField;
private Button mDateButton;
private CheckBox mSolvedCheckbox;

public static CrimeFragment newInstance(UUID crimeId) {
    Bundle args = new Bundle();
    args.putSerializable(ARG_CRIME_ID, crimeId);

    CrimeFragment fragment = new CrimeFragment();
    fragment.setArguments(args);
    return fragment;
}
```

En la clase CrimeActivity, podremos hacer que el EXTRA_CRIME_ID sea privado, ya que ninguna otra clase va a intentar acceder a él.

```
public class CrimeActivity extends SingleFragmentActivity {

    public private static final String EXTRA_CRIME_ID =
            "com.bignerdranch.android.criminalintent.crime_id";
    ...
    @Override
    protected Fragment createFragment() {
        return new CrimeFragment();
        UUID crimeId = (UUID) getIntent()
                .getSerializableExtra(EXTRA_CRIME_ID);
        return CrimeFragment.newInstance(crimeId);
    }
}
```

Para que un Fragment pueda acceder a sus argumentos, será necesario llamar a la función getArguments().

Si probamos la aplicación ahora, veremos que los datos del crimen no han sido modificados en la lista cuando los editamos en la pantalla de detalle. Eso es debido a que no hemos notificado a la Recylcer View que sus datos han cambiado.

Lo conseguiremos implementando el método onResume() de la clase CrimeListFragment de la siguiente forma:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    ...
}

@Override
public void onResume() {
    super.onResume();
    updateUI();
}

private void updateUI() {
    CrimeLab crimeLab = CrimeLab.get(getActivity());
    List<Crime> crimes = crimeLab.getCrimes();

    if (mAdapter == null) {
        mAdapter = new CrimeAdapter(crimes);
        mCrimeRecyclerView.setAdapter(mAdapter);
    } else {
        mAdapter.notifyDataSetChanged();
    }
}
```

Ahora ya podemos ver como si editamos un campo de un crimen en su pantalla de detalle, éste campo se verá también actualizado en la vista general de la lista de crímenes.

Challenges

Para los alumnos más curiosos, a continuación, se presentan algunos tópicos sobre los que investigar u otros que pueden ser implementados.

- Investigar más acerca de la función onSaveInstanceState(Bundle) para su uso en esta app (guardar ID del crimen, en vez de paso de argumentos puro).
- Investiga sobre cómo notificar al Adapter de la lista de crímenes que un crimen ha sido modificado, y de esta forma que recargue sólo ese ítem, por lo que no deberá cargar toda la lista de crímenes de nuevo.
- Implementa la solución investigada en el punto anterior.
- Optimiza el código de la clase CrimeLab para que encuentre más rápidamente un crimen dado su UUID.