ITESO

Universidad Jesuita
de Guadalajara

Maritza Mendoza Sicard

702016

Tarea 4

Tarea 4. SQLite, Fragmentos, Recicler View y ActivityResult

Se crearon 4 beans, Store, Category, ItemProduct y City. Cada uno se le implementó su parcelable, se le agregó dos constructores y se generó los getters and setters. Se muestra a continuación un ejemplo de lo que se hizo. Se repetió lo mismo para las otras tres clases, solo que con sus diferentes atributos.

```
public class City implements Parcelable {


    private int idCity;
    private String name;

    public City(){
    }

    protected City(Parcel in) {
        idCity = in.readInt();
        name = in.readString();
    }

    public static final Creator<City> CREATOR = new Creator<City>() {
        @Override
        public City createFromParcel(Parcel in) {
            return new City(in);
        }

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

    public int getIdCity() {
        return idCity;
    }

    public City(int idCity, String name) {
        this.idCity = idCity;
        this.name = name;
    }

    public void setIdCity(int idCity) {
        this.idCity = idCity;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

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

```
    @Override
    public void writeToParcel(Parcel parcel, int i) {
        parcel.writeInt(idCity);
        parcel.writeString(name);
    }
}
```

Se creó la clase de DataBaseHandler que se presentó en las diapositivas. También se hizo el control de cada clase de los beans generados anteriormente, cada uno con sus métodos respectivos que nos ayuden. A continuación se muestra el ejemplo con ItemProductsControl:

```
public class ItemProductControl {

    public long addItemProduct(ItemProduct product, DataBaseHandler dh){
        long inserted = 0;
        SQLiteDatabase db = dh.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put(DataBaseHandler.KEY_PRODUCT_TITLE, product.getTitle());
        values.put(DataBaseHandler.KEY_PRODUCT_IMAGE, product.getImage());
        values.put(DataBaseHandler.KEY_PRODUCT_CATEGORY,
product.getCategory().getIdCategory());
        // Inserting Row
        inserted = db.insert(DataBaseHandler.TABLE_PRODUCT, null, values);
        // Closing database connection
        try {db.close();} catch (Exception e) {}
        db = null; values = null;
        return inserted;
    }
    public int updateProduct(ItemProduct product, DataBaseHandler dh){
        long inserted = 0;
        SQLiteDatabase db = dh.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put(DataBaseHandler.KEY_PRODUCT_TITLE, product.getTitle());
        values.put(DataBaseHandler.KEY_PRODUCT_IMAGE, product.getImage());
        values.put(DataBaseHandler.KEY_PRODUCT_CATEGORY,
product.getCategory().getIdCategory());
        int count = db.update(DataBaseHandler.TABLE_PRODUCT, values,
                DataBaseHandler.KEY_PRODUCT_ID + " = ?",
                new String[] { String.valueOf(product.getCode()) });
        try { db.close();} catch (Exception e) {}
        db = null;
        return count;
    }
    public void deleteProduct(int idProduct, DataBaseHandler dh){
        SQLiteDatabase db = dh.getWritableDatabase();
        db.delete(DataBaseHandler.TABLE_STORE, DataBaseHandler.KEY_PRODUCT_ID
                + " = ?", new String[] { String.valueOf(idProduct) });
        try {
            db.close();
        } catch (Exception e) {
        }
        db = null;
    }
    public ItemProduct getProductById(int idProduct, DataBaseHandler dh){
        ItemProduct product = new ItemProduct();
        String selectQuery = "SELECT P." + DataBaseHandler.KEY_PRODUCT_ID + ","
                + "P." + DataBaseHandler.KEY_PRODUCT_TITLE + ","
                + "P." + DataBaseHandler.KEY_PRODUCT_IMAGE + ","
                + "C." + DataBaseHandler.KEY_CATEGORY_ID + ","
```

```
                + "C." + DataBaseHandler.KEY_CATEGORY_NAME + " FROM "
                + DataBaseHandler.TABLE_PRODUCT + " P,"
                + DataBaseHandler.TABLE_CATEGORY + " C WHERE P."
                + DataBaseHandler.KEY_PRODUCT_ID + "= " + idProduct
                + " AND P." + DataBaseHandler.KEY_PRODUCT_CATEGORY
                + " = C." + DataBaseHandler.KEY_CATEGORY_ID;
        SQLiteDatabase db = dh.getReadableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        if (cursor.moveToFirst()) {
            product.setCode(cursor.getInt(0));
            product.setTitle(cursor.getString(1));
            product.setImage(cursor.getInt(2));
            Category category = new Category();
            category.setIdCategory(cursor.getInt(3));
            category.setName(cursor.getString(4));
            product.setCategory(category);
        }
        try {cursor.close();db.close();
        } catch (Exception e) {}
        db = null;
        cursor = null;
        return product;
    }
    public ArrayList<ItemProduct> getProductsWhere(String strWhere, String
strOrderBy, DataBaseHandler dh){
        ArrayList<ItemProduct> products = new ArrayList<ItemProduct>();
        String query;
        if(strWhere != null){
            query = "SELECT P." + DataBaseHandler.KEY_PRODUCT_ID + ","
                    + "P." + DataBaseHandler.KEY_PRODUCT_TITLE + ","
                    + "P." + DataBaseHandler.KEY_PRODUCT_IMAGE + ","
                    + "C." + DataBaseHandler.KEY_CATEGORY_ID + ","
                    + "C." + DataBaseHandler.KEY_CATEGORY_NAME + " FROM "
                    + DataBaseHandler.TABLE_PRODUCT + " P,"
                    + DataBaseHandler.TABLE_CATEGORY + " C WHERE "
                    + " P." + DataBaseHandler.KEY_PRODUCT_CATEGORY
                    + " = C." + DataBaseHandler.KEY_CATEGORY_ID + " AND "
                    + strWhere + " ORDER BY " + strOrderBy;
        }else{
            query = "SELECT P." + DataBaseHandler.KEY_PRODUCT_ID + ","
                    + "P." + DataBaseHandler.KEY_PRODUCT_TITLE + ","
                    + "P." + DataBaseHandler.KEY_PRODUCT_IMAGE + ","
                    + "C." + DataBaseHandler.KEY_CATEGORY_ID + ","
                    + "C." + DataBaseHandler.KEY_CATEGORY_NAME + " FROM "
                    + DataBaseHandler.TABLE_PRODUCT + " P,"
                    + DataBaseHandler.TABLE_CATEGORY + " C WHERE "
                    + " P." + DataBaseHandler.KEY_PRODUCT_CATEGORY
                    + " = C." + DataBaseHandler.KEY_CATEGORY_ID
                    + " ORDER BY " + strOrderBy;
        }
        SQLiteDatabase db = dh.getReadableDatabase();

        // El null funcion como en el where del delete, ahi iria el arreglo
        Cursor cursor = db.rawQuery(query,null);

        while (cursor.moveToNext()) {
            ItemProduct product = new ItemProduct();
            product.setCode(cursor.getInt(0));
            product.setTitle(cursor.getString(1));
            product.setImage(cursor.getInt(2));
            Category category = new Category();
            category.setIdCategory(cursor.getInt(3));
            category.setName(cursor.getString(4));
```

```
            product.setCategory(category);
            products.add(product);
        }

        try {
            db.close();
        } catch (Exception e) {

        }
        db = null;
        cursor = null;

        return products;
    }
}
```

A cada Fragmento se le cambió lo pasa al pedir los productos con el método de getProductsWhere. Se muestra el ejemplo con FragmentsTechnology:

```
public class FragmentTechnology extends Fragment {
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mlayoutManager;
    private ArrayList myDataSet;

    public FragmentTechnology() {
        // Required empty public constructor
    }



    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_technology, container,
false);
        RecyclerView recyclerView = (RecyclerView)
view.findViewById(R.id.fragment_technology_recycler_view);
        //improve performance
        mlayoutManager = new LinearLayoutManager(getActivity());
        recyclerView.setLayoutManager(mlayoutManager);

        ItemProductControl itemProductControl = new ItemProductControl();
        myDataSet = itemProductControl.getProductsWhere(
                "C.name = 'TECHNOLOGY'", DataBaseHandler.KEY_PRODUCT_ID + " ASC",
                DataBaseHandler.getInstance(getActivity()));

        mAdapter = new AdapterProduct(myDataSet, getActivity());
        recyclerView.setAdapter(mAdapter);
        return view;
    }

    public void notifyDataSetChanged(ItemProduct itemProduct){
        myDataSet.add(itemProduct);
        mAdapter.notifyDataSetChanged();
    }

}
```

Se creó un ActivityProducts que busca o crea un nuevo producto en la base de datos. Eso se apoyó en las presentaciones. También se creó una clase de constantes y se modificó parte del ActivityMain para que generara cada fragmento y para que abriera la actividad de ActivityProducts al presionar el botón del menú.

Se presenta el ActivityProducts:

```
public class ActivityProduct extends AppCompatActivity {

    protected Spinner stores;
    protected Spinner categories;
    protected Spinner images;
    protected EditText id;
    protected EditText title;
    protected EditText description;
    protected ArrayAdapter<Store> storesAdapter;
    protected ArrayAdapter<Category> categoriesAdapter;
    protected ArrayAdapter<String> imagesAdapter;
    protected DataBaseHandler dh; //DataBase Instance
    protected Store storeSelected; //Store selected in spinner
    protected Category categorySelected; //Category selected in spinner
    protected int imageSelected; //Image selected in spinner

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_product);
        stores = (Spinner)findViewById(R.id.activity_product_store);
        categories = (Spinner)findViewById(R.id.activity_product_category);
        images = (Spinner)findViewById(R.id.activity_product_image);
        id = (EditText)findViewById(R.id.activity_product_id);
        title = (EditText)findViewById(R.id.activity_product_title);
        description = (EditText)findViewById(R.id.activity_product_description);
        storeSelected = null;
        categorySelected = null;
        imageSelected = -1;

        //DataBase Objects
        dh = DataBaseHandler.getInstance(this);
        StoreControl storeControl = new StoreControl();
        CategoryControl categoryControl = new CategoryControl();
        //Fill info from Database
        ArrayList<Store> storesList = storeControl.getStoresWhere(null, null,
dh);
        ArrayList<Category> categoriesList =
categoryControl.getAllCategories(dh);
        //Create Adapter to show into Spinner, ListView or GridLayout
        storesAdapter =
                new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
storesList);
        stores.setAdapter(storesAdapter);
        categoriesAdapter =
                new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
categoriesList);
        categories.setAdapter(categoriesAdapter);
        ArrayList<String> myimages = new ArrayList<>();
        myimages.add("Mac"); myimages.add("Alienware");
        imagesAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, myimages);
```

```
        images.setAdapter(imagesAdapter);
        stores.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
{
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                storeSelected = storesAdapter.getItem(position);
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {}
        });
        images.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
{
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                imageSelected = position;
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {}
        });
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.activity_product_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_save) {
            if(isValidProduct()){
                ItemProduct itemProduct = new ItemProduct();
                itemProduct.setTitle(title.getText().toString().trim());

itemProduct.setDescription(description.getText().toString().trim());
                itemProduct.setStore(storeSelected);
                itemProduct.setCategory(categorySelected);
                itemProduct.setImage(imageSelected);
                ItemProductControl itemProductControl = new ItemProductControl();
                itemProductControl.addItemProduct(itemProduct, dh);
                Intent intent = new Intent();
                intent.putExtra("ITEM", itemProduct);
                setResult(Activity.RESULT_OK, intent);
                finish();
            }
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    private boolean isValidProduct() {
        if(!stores.isSelected()) return false;
        if(!categories.isSelected())return false;
        if(!images.isSelected())return false;
        if(id.getText().toString() == "") return false;
        if(title.getText().toString() == "")return false;
        if(description.getText().toString() == "")return false;
        return true;
    }

    public void setCategorySelected(int categoryId){
        for (int position = 0; position < categoriesAdapter.getCount();
position++)
```

```
        {
            if(((Category)categoriesAdapter.getItem(position)).getIdCategory() ==
categoryId)
            {
                categories.setSelection(position);
                return;
            }
        }
    }
    public void setStoreSelected(int storeId){
        for (int position = 0; position < storesAdapter.getCount(); position++)
        {
            if(((Store)storesAdapter.getItem(position)).getId() == storeId)
            {
                stores.setSelection(position);
                return;
            }
        }
    }
    public void setImageSelected(int imageId){
        images.setSelection(imageId);
    }

    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.activity_product_search:
                int idProduct = 0;
                try {
                    idProduct = Integer.parseInt(id.getText().toString().trim());
                }catch(NumberFormatException e){ return; }
                ItemProductControl itemProductControl = new ItemProductControl();
                ItemProduct itemProduct =
itemProductControl.getProductById(idProduct, dh);
                if(itemProduct != null) {
                    title.setText(itemProduct.getTitle());
                    description.setText(itemProduct.getDescription());
                    if(itemProduct.getCategory() != null) {

setCategorySelected(itemProduct.getCategory().getIdCategory());}
                    if(itemProduct.getStore()!= null) {
                        setStoreSelected(itemProduct.getStore().getId());}
                    setImageSelected(itemProduct.getImage());
                }
                break;
        }
    }
}
```

Para observar el Código completo se tiene en el siguiente repositorio en github:
https://github.com/MSicard/ProgramacionMovil/tree/master/S9/MyApplication