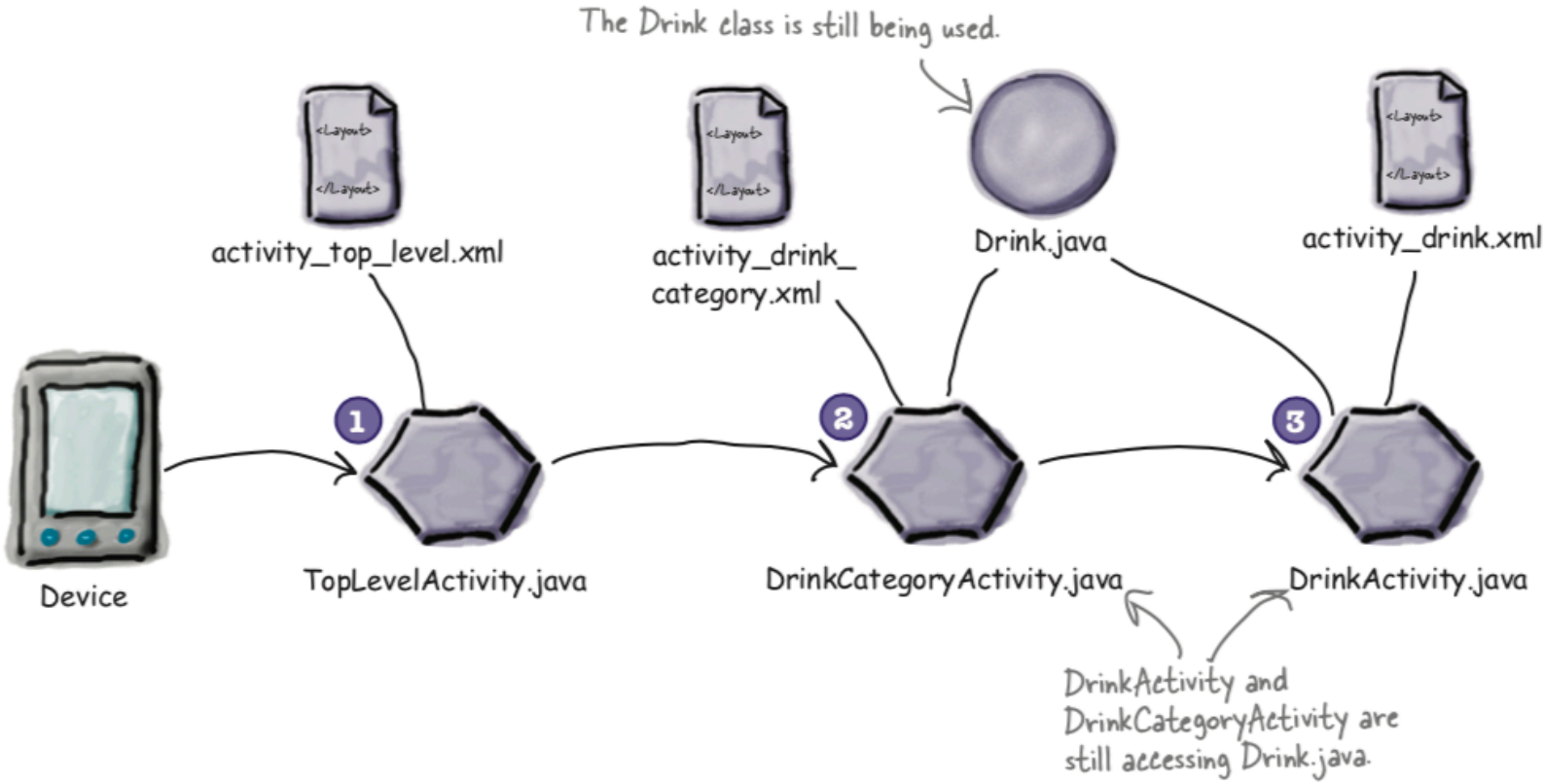
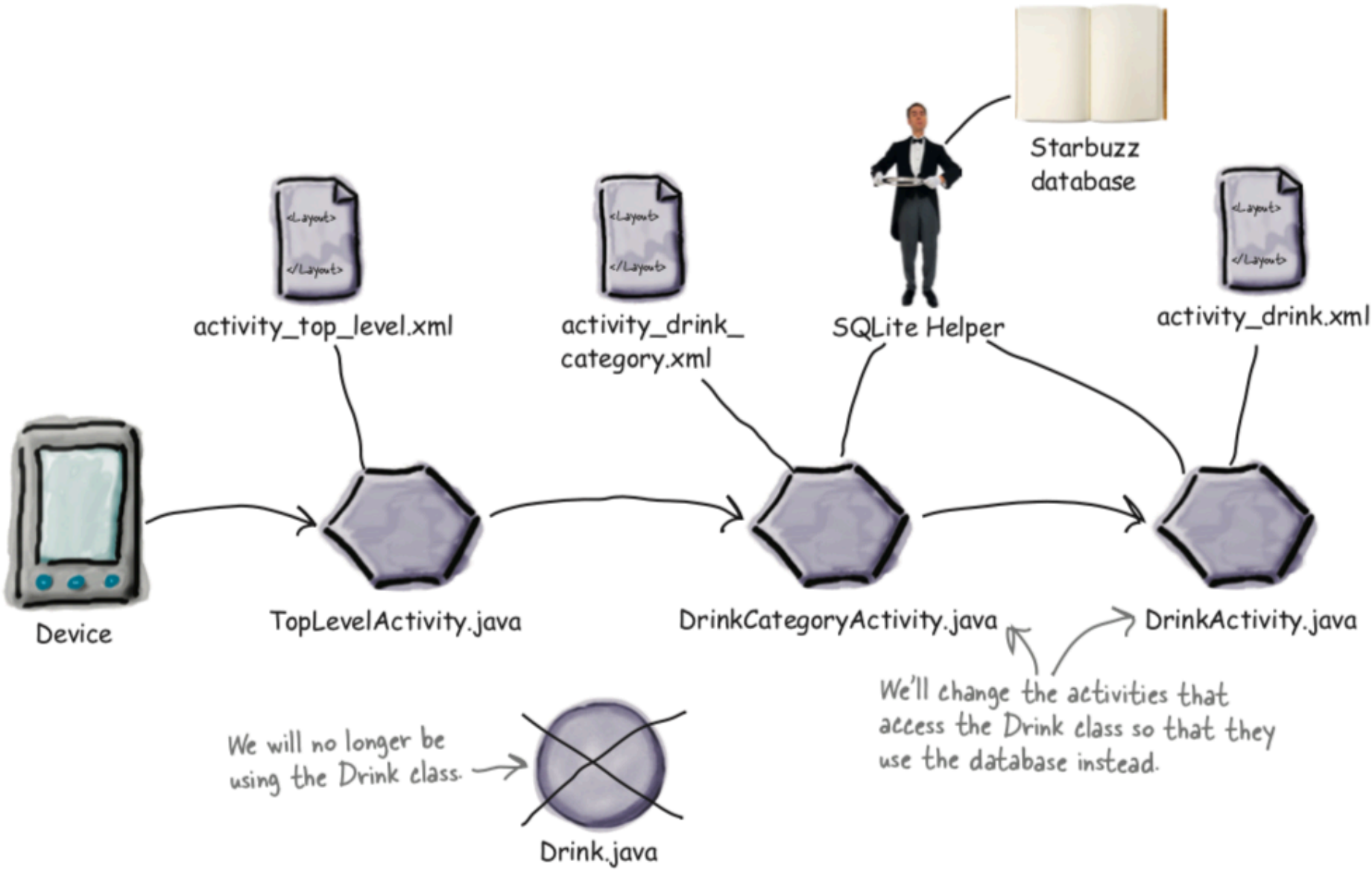
9
Cursores básicos

Hasta donde hemos llegado

1. **TopLevelActivity displays a list of options for Drinks, Food, and Stores.**
2. **When the user clicks on the Drinks option, it launches DrinkCategoryActivity.**
   This activity displays a list of drinks that it gets from the Java `Drink` class.
3. **When the user clicks on a drink, its details get displayed in DrinkActivity.**
   `DrinkActivity` gets details of the drink from the Java `Drink` class.



Cambiando la aplicación para utilizar la base de datos



Cambiando la aplicación para utilizar la base de datos

1. **Get a reference to the Starbuzz database.**
We'll do this using the Starbuzz SQLite helper we created. 5.

2. **Create a cursor to read drink data from the database.**
We need to read the data held in the Starbuzz database for the drink the user selects in `DrinkCategoryActivity`. The cursor will give us access to this data. (We'll explain cursors soon.)

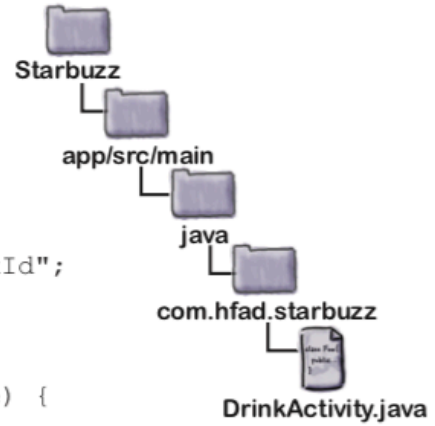
3. **Navigate to the drink record.**
Before we can use the data retrieved by the cursor, we need to explicitly navigate to it.

4. **Display details of the drink in DrinkActivity.**
Once we've navigated to the drink record in the cursor, we need to read the data and display it in `DrinkActivity`.

El código actual de DrinkActivity

```
package com.hfad.starbuzz;

...
public class DrinkActivity extends Activity {

    public static final String EXTRA_DRINKID = "drinkId";

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

        //Get the drink from the intent
        int drinkId = (Integer)getIntent().getExtras().get(EXTRA_DRINKID);
        Drink drink = Drink.drinks[drinkId];

        //Populate the drink name
        TextView name = (TextView)findViewById(R.id.name);
        name.setText(drink.getName());

        //Populate the drink description
        TextView description = (TextView)findViewById(R.id.description);
        description.setText(drink.getDescription());

        //Populate the drink image
        ImageView photo = (ImageView)findViewById(R.id.photo);
        photo.setImageResource(drink.getImageResourceId());
        photo.setContentDescription(drink.getName());
    }
}
```

We're not showing you the import statements.

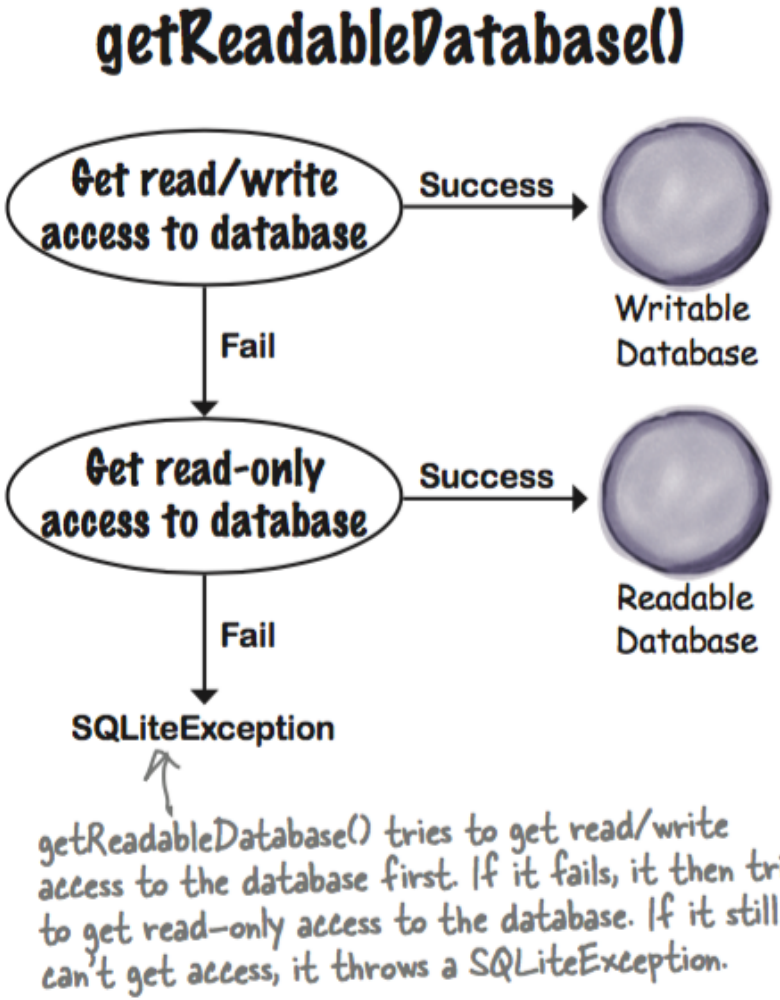


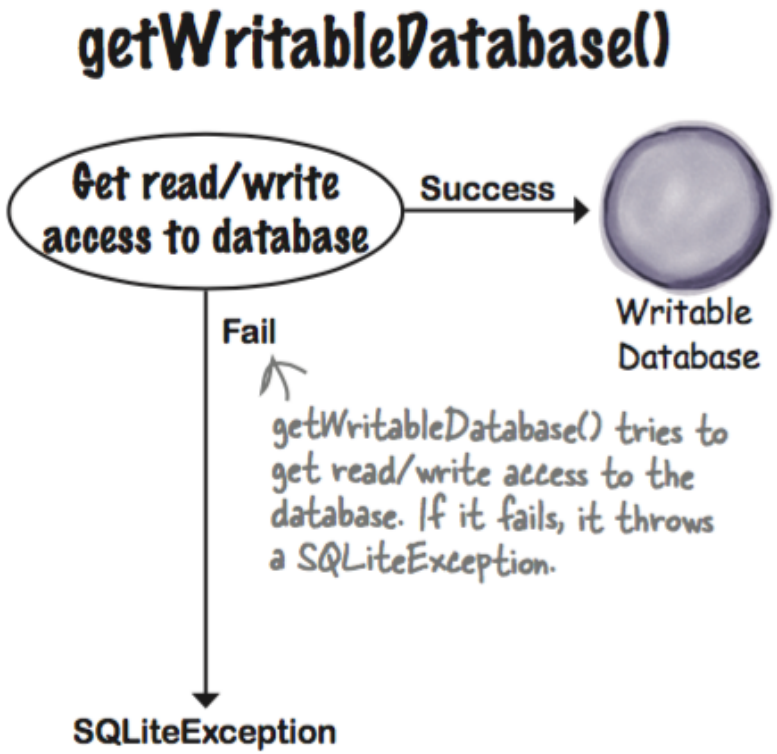
This is the drink the user selected.

Use the drink ID from the intent to get the drink details from the Drink class. We'll need to change this so the drink comes from the database.

We need to populate the views in the layout with values from the database, not from the Drink class.

Obteniendo una referencia a la base de datos



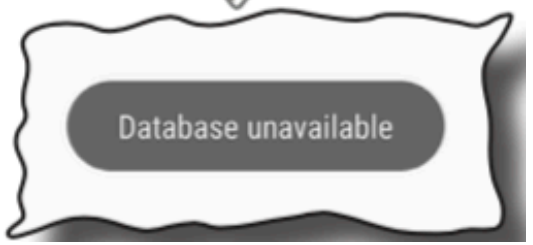


Obteniendo una referencia a la base de datos

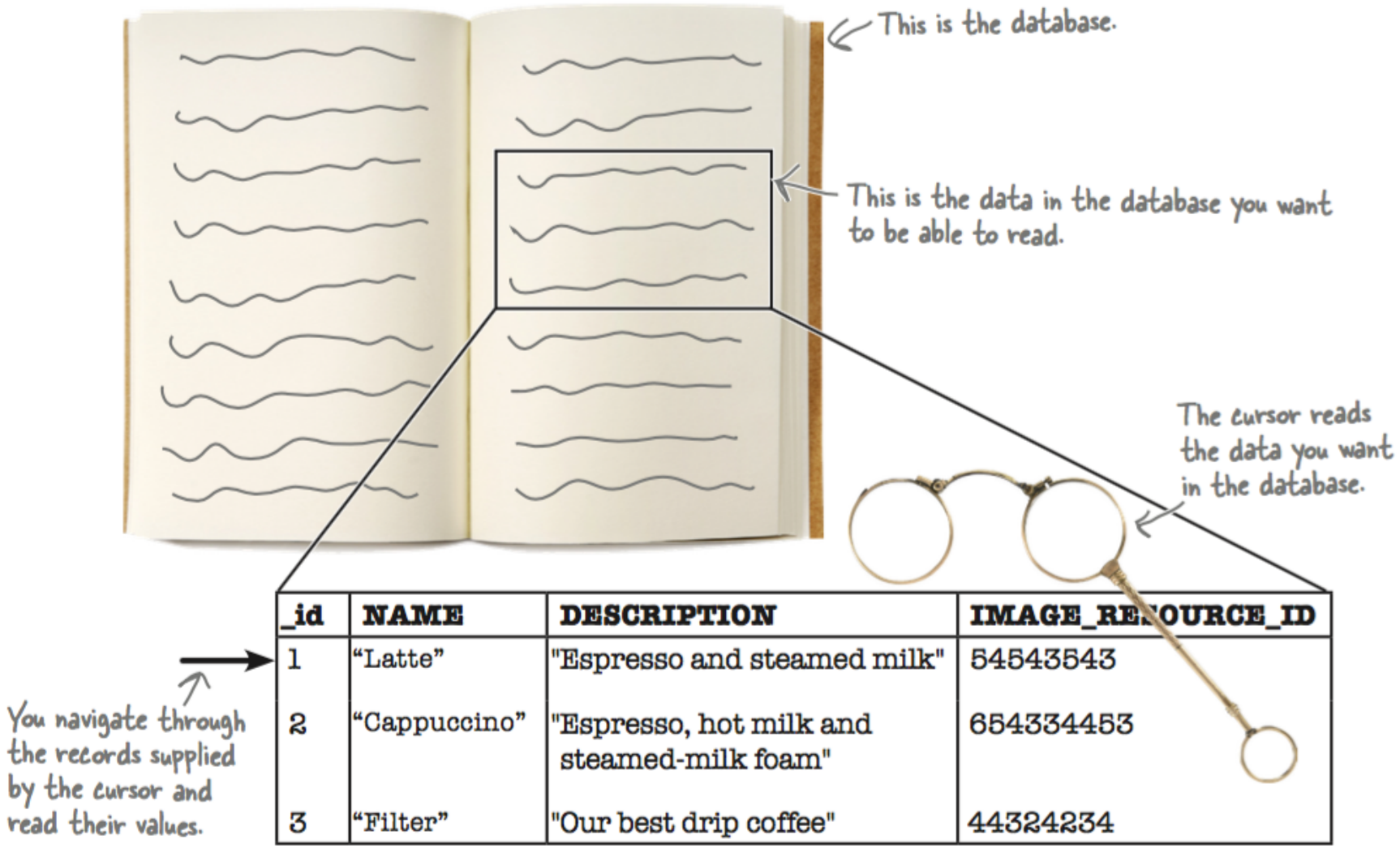
```
SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
try {
    SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
    //Code to read data from the database
} catch(SQLiteException e) {
    Toast toast = Toast.makeText(this,
                                 "Database unavailable",
                                 Toast.LENGTH_SHORT);
    toast.show();
}
```

This line displays the toast.

These lines create a Toast that will display the message "Database unavailable" for a few seconds.



Obteniendo datos de la base de datos con un cursor



_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk"	54543543
2	"Cappuccino"	"Espresso, hot milk and steamed-milk foam"	654334453
3	"Filter"	"Our best drip coffee"	44324234

Especificando tablas y columnas

Put each column you want back as a separate value in a String array.

```
Cursor cursor = db.query("DRINK",
                         new String[] {"NAME", "DESCRIPTION"},
                         null, null, null, null, null);
```

This query only uses the first two parameters, hence the null values.

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table.

NAME	DESCRIPTION
"Latte"	"Espresso and steamed milk"
"Cappuccino"	"Espresso, hot milk and steamed-milk foam"
"Filter"	"Our best drip coffee"

Restringiendo la consulta

```
Cursor cursor = db.query("DRINK",
        new String[] {"_id", "NAME", "FAVORITE"},
        null, null, null, null,
        "NAME ASC");
```

← Order by NAME in ascending order.

_id	NAME	FAVORITE
2	"Cappuccino"	0
3	"Filter"	1
1	"Latte"	0

```
Cursor cursor = db.query("DRINK",
        new String[] {"_id", "NAME", "FAVORITE"},
        null, null, null, null,
        "FAVORITE DESC, NAME");
```

← Order by FAVORITE in descending order, then NAME in ascending order.

_id	NAME	FAVORITE
3	"Filter"	1
2	"Cappuccino"	0
1	"Latte"	0

Restringiendo la consulta

```
Cursor cursor = db.query("DRINK",
                    new String[] {"_id", "NAME", "DESCRIPTION"},
                    "NAME = ?",
                    new String[] {"Latte"},
                    null, null, null);
```

These are the columns we want to return.

We want to return records where the value of the NAME column is "Latte".

_id	NAME	DESCRIPTION
1	"Latte"	"Espresso and steamed milk"

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table where the value of the NAME column is "Latte".

Restringiendo la consulta

```
Cursor cursor = db.query("DRINK",
                         new String[] {"NAME", "DESCRIPTION"},
                         "NAME = ? OR DESCRIPTION = ?",
                         new String[] {"Latte", "Our best drip coffee"},
                         null, null, null);
```

This means "where NAME is 'Latte' or DESCRIPTION is "'Our best drip coffee'".

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table where the value of the NAME column is "Latte" or the value of the DESCRIPTION column is "Our best drip coffee".

NAME	DESCRIPTION
"Latte"	"Espresso and steamed milk"
"Filter"	"Our best drip coffee"

Restringiendo la consulta

```
Cursor cursor = db.query("Drink",
                         new String[] {"NAME", "DESCRIPTION"},
                         "_id = ?",
                         new String[] {Integer.toString(1)},
                         null, null, null);
```

Convert the int 1 to a String value.

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table where the value of the _id column is 1.

_id	NAME	DESCRIPTION
1	"Latte"	"Espresso and steamed milk"

El código para obtener un cursor

DrinkActivity

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

    //Get the drink from the intent
    int drinkId = (Integer)getIntent().getExtras().get(EXTRA_DRINKID);
    Drink drink = Drink.drinks[drinkId];

    //Create a cursor
    SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
    try {
        SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
        Cursor cursor = db.query ("DRINK",
                new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID"},
                "_id = ?",
                new String[] {Integer.toString(drinkId)},
                null, null, null);
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }
    ...
}
}
```

We'll add the code to the onCreate() method.

We're no longer getting the drinks from Drink.java.

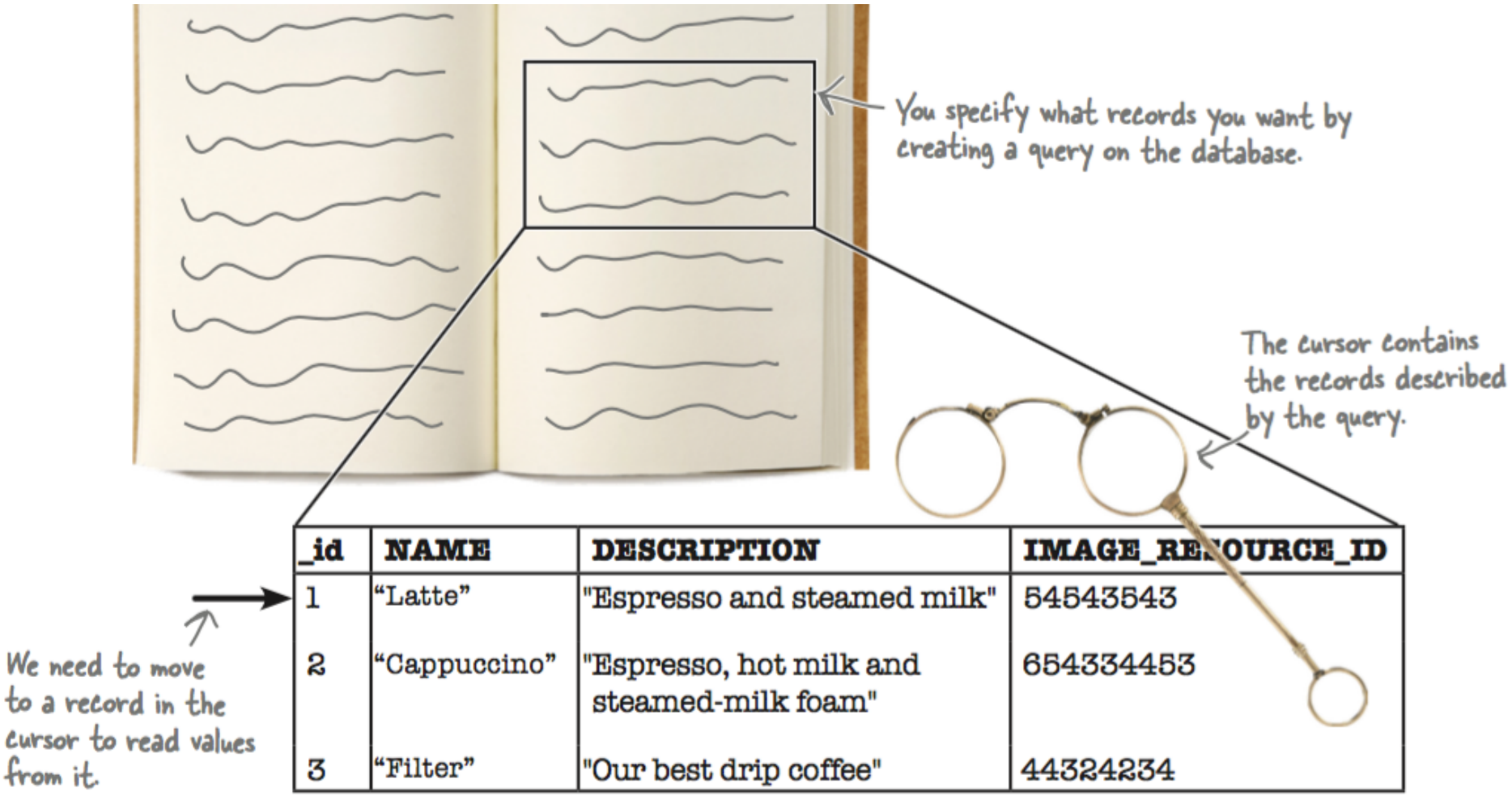
Get a reference to the database.

Create a cursor to get the name, description, and image resource ID of the drink the user selected.

Display a pop-up message if a SQLiteException is thrown.

There's more code we haven't changed yet, but you don't need to see it right now.

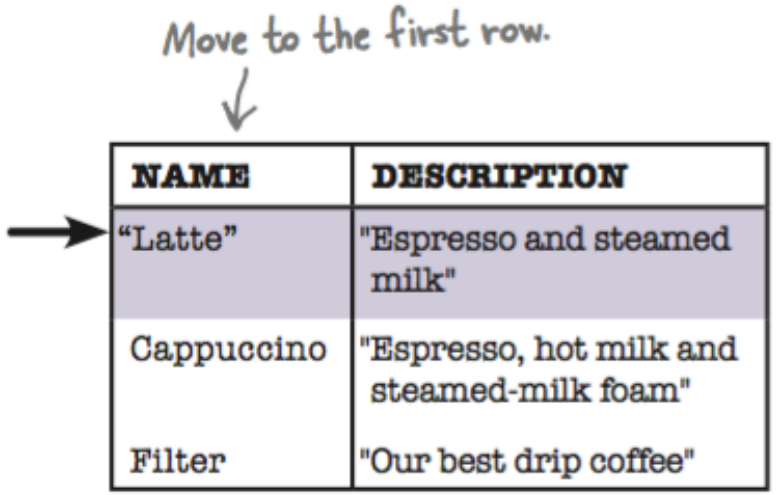
Navegando a través de un cursor



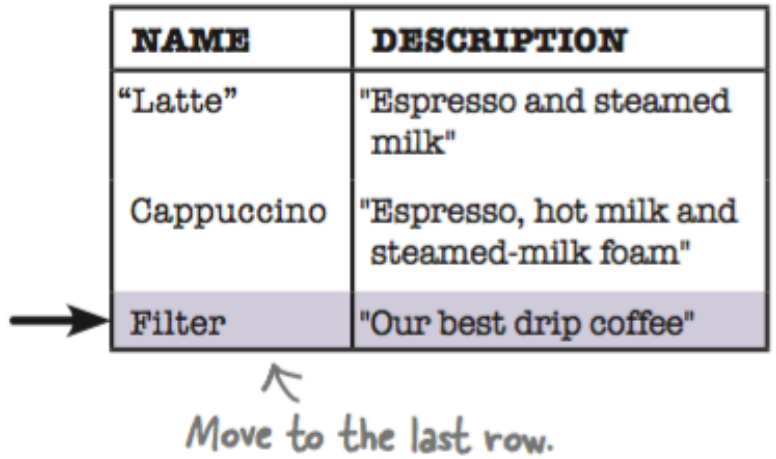
_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk"	54543543
2	"Cappuccino"	"Espresso, hot milk and steamed-milk foam"	654334453
3	"Filter"	"Our best drip coffee"	44324234

Navegando a través de un cursor

```
if (cursor.moveToFirst()) {
    //Do something
};
```

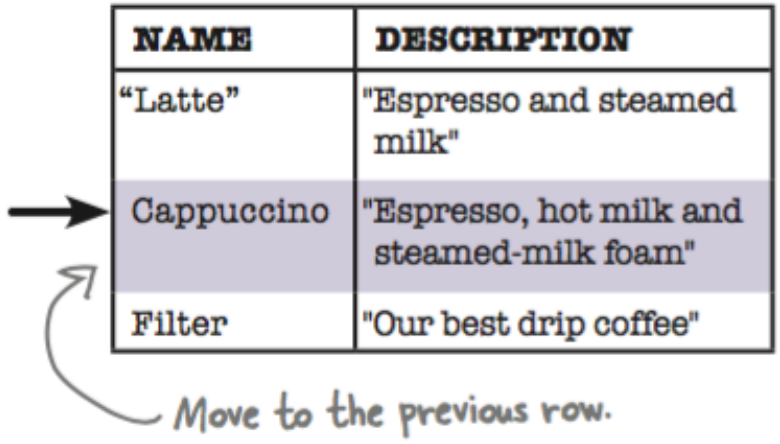



```
if (cursor.moveToLast()) {
    //Do something
};
```



Navegando a través de un cursor

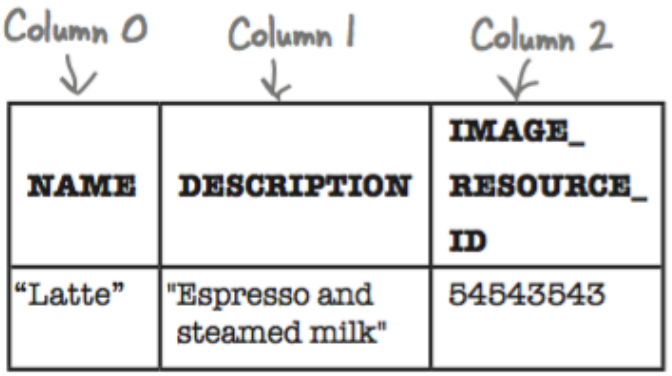
```
if (cursor.moveToPrevious()) {
    //Do something
};
```




```
if (cursor.moveToNext()) {
    //Do something
};
```

Obteniendo valores de cursor

```
Cursor cursor = db.query ("Drink",
                          new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID"},
                          "_id = ?",
                          new String[] {Integer.toString(1)},
                          null, null,null);
```




```
String name = cursor.getString(0);
```
← This is the first column in the cursor.

Cerrando el cursor

```
cursor.close();
db.close();
```

El código de DrinkActivity

```
package com.hfad.starbuzz;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;

public class DrinkActivity extends Activity {

    public static final String EXTRA_DRINKID = "drinkId";
```

We're using these extra classes in the code.

El código de DrinkActivity

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

    //Get the drink from the intent
    int drinkId = (Integer)getIntent().getExtras().get(EXTRA_DRINKID);
    Drink drink = Drink.drinks[drinkId];

    //Create a cursor
    SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
    try {
        SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
        Cursor cursor = db.query ("DRINK",
                new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID"},
                "_id = ?",
                new String[] {Integer.toString(drinkId)},
                null, null, null);
```

This is the ID of the drink the user chose.

We're no longer getting our data from the drinks array, so we need to delete this line.

Create a cursor that gets the NAME, DESCRIPTION, and IMAGE_RESOURCE_ID data from the DRINK table where _id matches drinkId.

El código de DrinkActivity

```
        //Move to the first record in the Cursor
        if (cursor.moveToFirst()) {

            //Get the drink details from the cursor
            String nameText = cursor.getString(0);
            String descriptionText = cursor.getString(1);
            int photoId = cursor.getInt(2);

            //Populate the drink name
            TextView name = (TextView)findViewById(R.id.name);
            name.setText(drink.getName());
            name.setText(nameText);

            //Populate the drink description
            TextView description = (TextView)findViewById(R.id.description);
            description.setText(drink.getDescription());
            description.setText(descriptionText);

            //Populate the drink image
            ImageView photo = (ImageView)findViewById(R.id.photo);
            photo.setImageResource(drink.getImageResourceId());
            photo.setContentDescription(drink.getName());
            photo.setImageResource(photoId);
            photo.setContentDescription(nameText);
        }
        cursor.close();
        db.close();
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this,
                                     "Database unavailable",
                                     Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

There's only one record in the cursor, but we still need to move to it.

The name of the drink is the first item in the cursor, the description is the second column, and the image resource ID is the third. That's because we told the cursor to use the NAME, DESCRIPTION, and IMAGE_RESOURCE_ID columns from the database in that order.

Set the drink name to the value from the database.

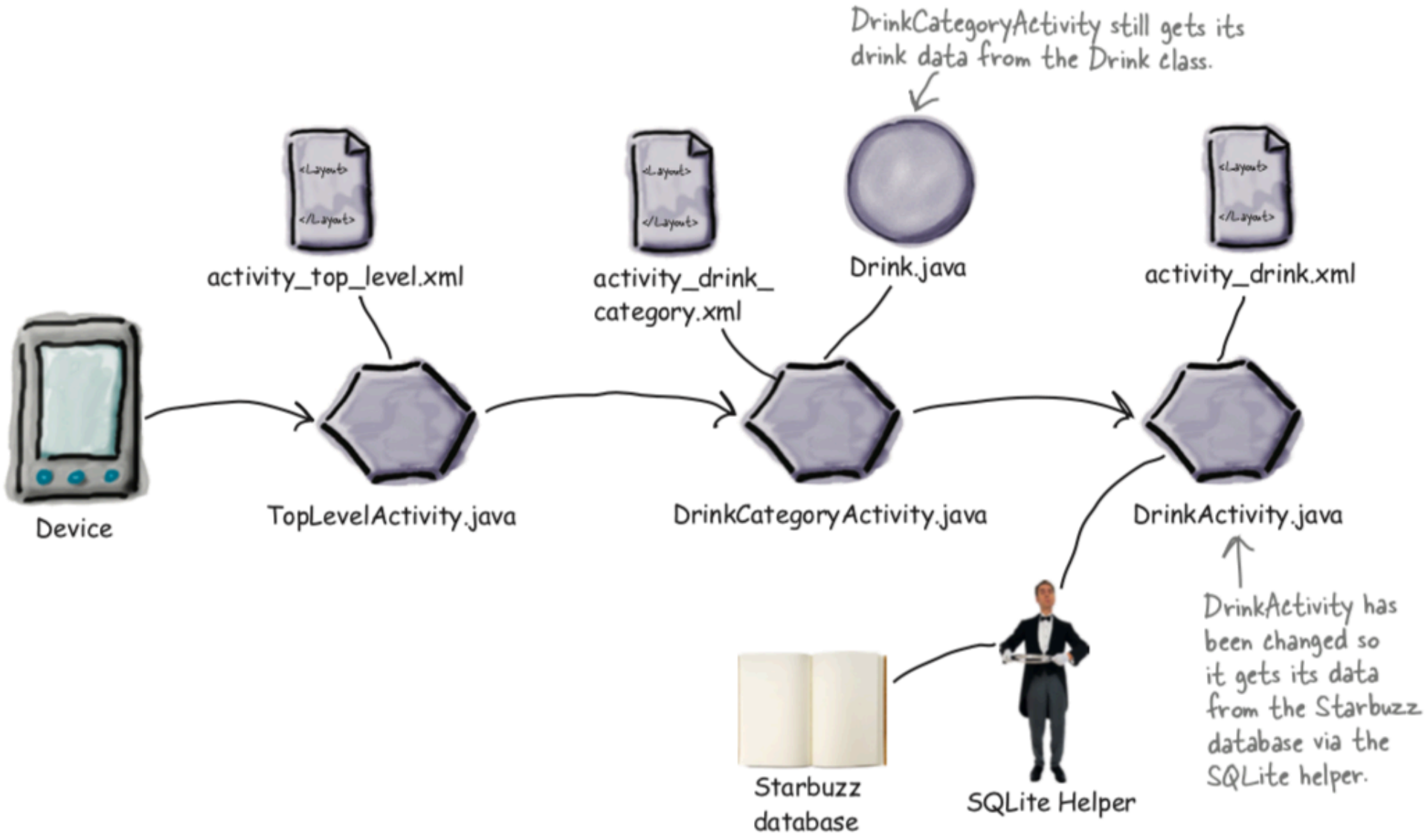
Use the drink description from the database.

Set the image resource ID and description to the values from the database.

Close the cursor and database.

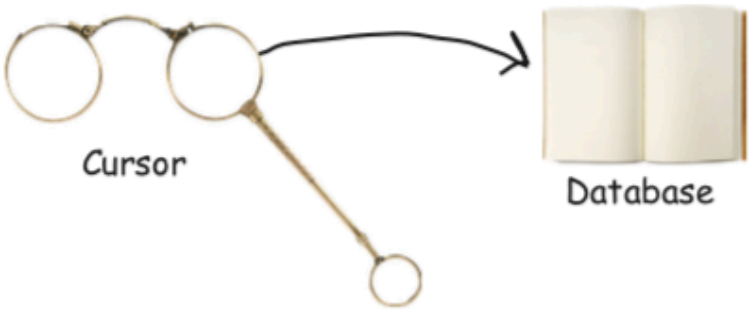
If a SQLiteException is thrown, this means there's a problem with the database. In this case, we'll use a toast to display a message to the user.

Donde vamos hasta ahora

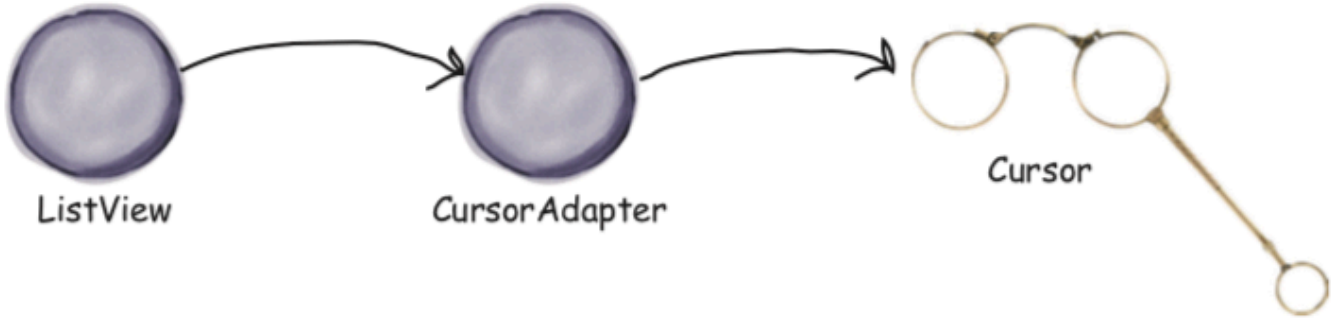


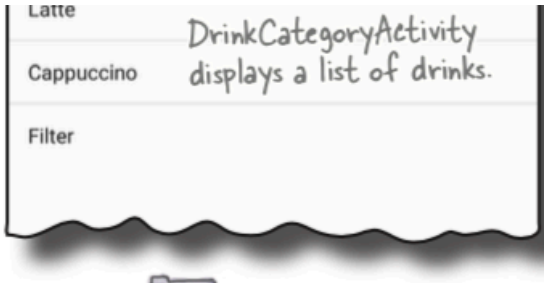
Que vamos a hacer

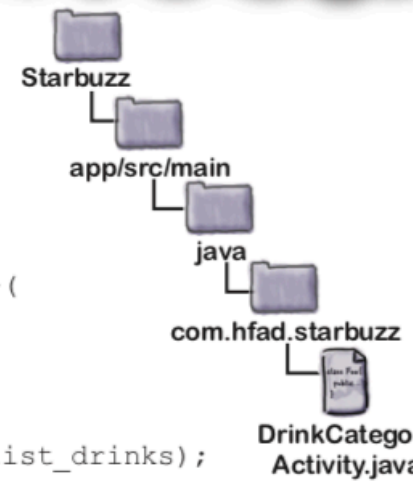
1 **Create a cursor to read drink data from the database.**
As before, we need to get a reference to the Starbuzz database. Then we'll create a cursor to retrieve the drink names from the DRINK table.



2 **Replace the list view's array adapter with a cursor adapter.**
The list view currently uses an array adapter to get its drink names. This is because the data's held in an array in the `Drink` class. Because we're now accessing the data using a cursor, we'll use a cursor adapter instead.





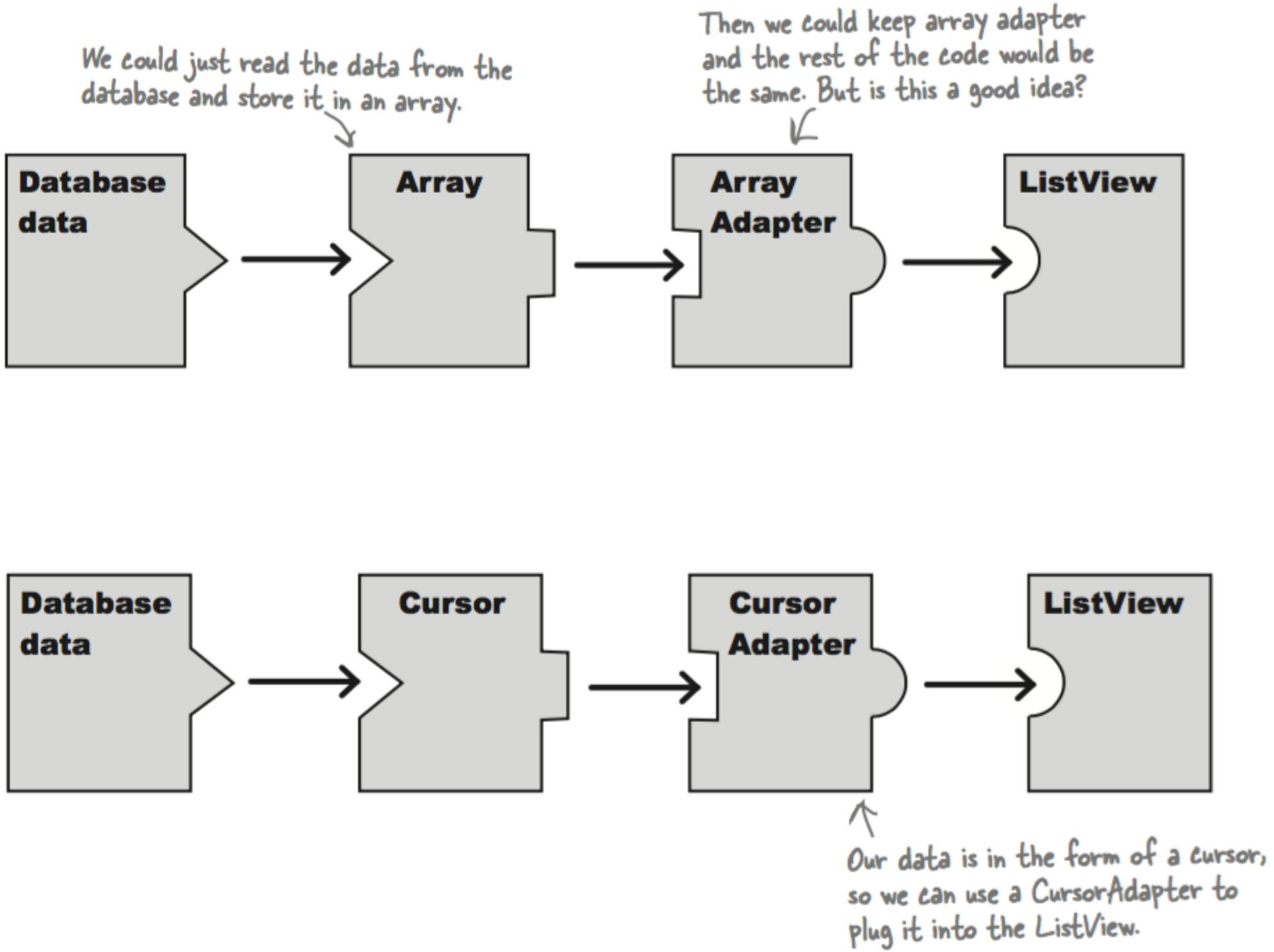



```
package com.hfad.starbuzz;

...

public class DrinkCategoryActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_drink_category);
        ArrayAdapter<Drink> listAdapter = new ArrayAdapter<>(
                this,
                android.R.layout.simple_list_item_1,
                Drink.drinks);
        ListView listDrinks = (ListView) findViewById(R.id.list_drinks);
        listDrinks.setAdapter(listAdapter);

        //Create a listener to listen for clicks in the list view
        AdapterView.OnItemClickListener itemClickListener =
                new AdapterView.OnItemClickListener(){
                    public void onItemClick(AdapterView<?> listDrinks,
                                            View itemView,
                                            int position,
                                            long id) {
                        //Pass the drink the user clicks on to DrinkActivity
                        Intent intent = new Intent(DrinkCategoryActivity.this,
                                DrinkActivity.class);
                        intent.putExtra(DrinkActivity.EXTRA_DRINKID, (int) id);
                        startActivity(intent);
                    }
                };

        //Assign the listener to the list view
        listDrinks.setOnItemClickListener(itemClickListener);
    }
```

At the moment, we're using an ArrayAdapter to bind an array to the ListView. We need to replace this code so that the data comes from a database instead.

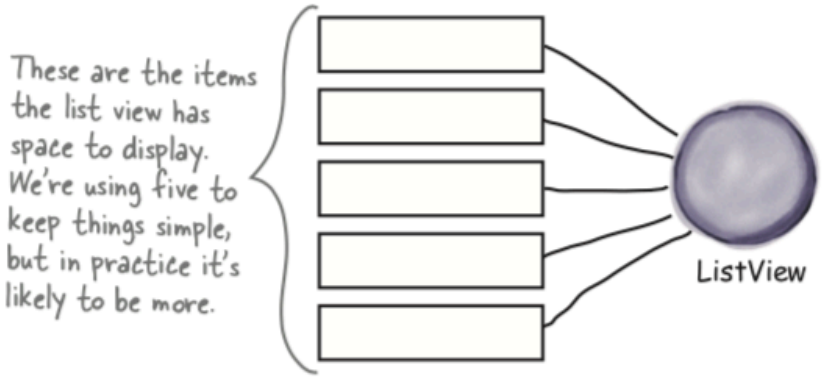
El código actual de DrinkCategoryActivity

Reemplazando el arreglo de datos en el ListView

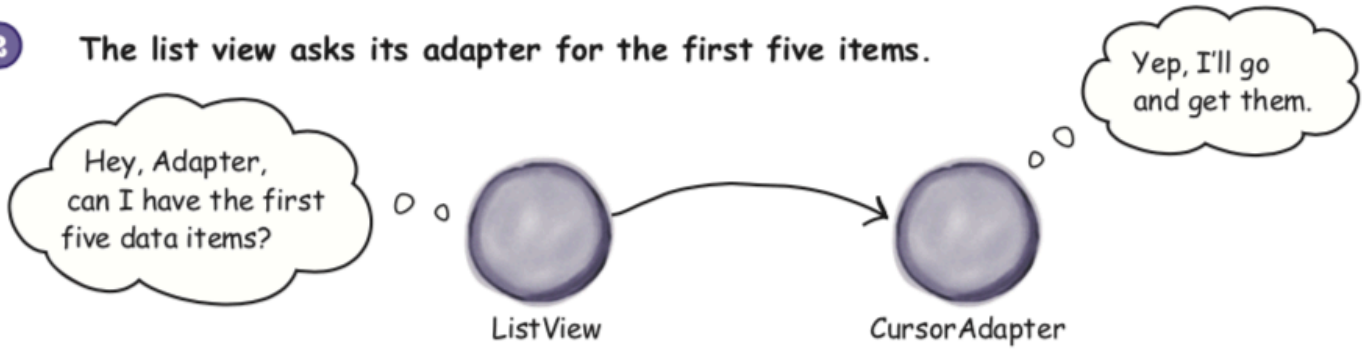


Como funciona un cursor adapter

1 **The list view gets displayed on the screen.**
When the list is first displayed, it will be sized to fit the screen. Let's say it has space to show five items.



2 **The list view asks its adapter for the first five items.**

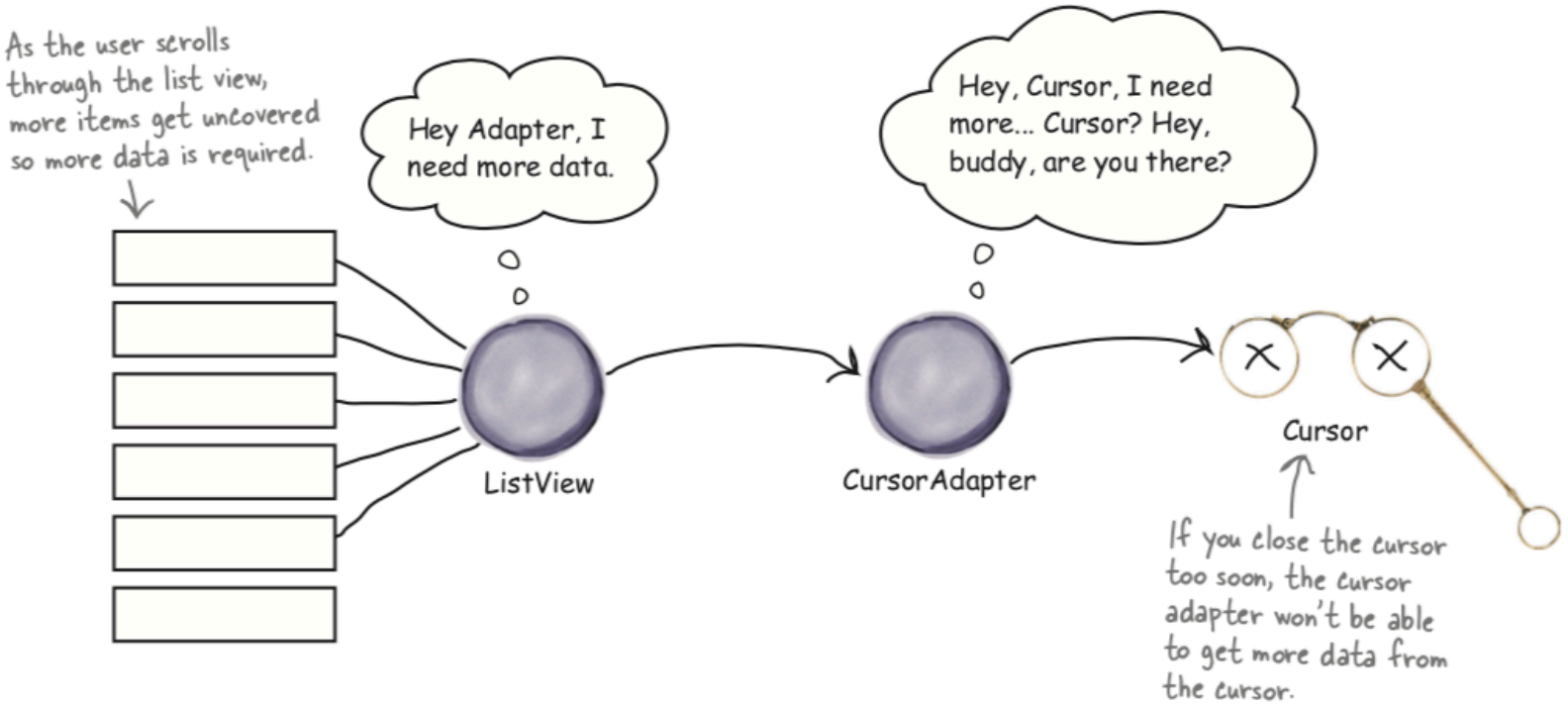


3 **The cursor adapter asks its cursor to read five rows from the database.**
No matter how many rows the database table contains, the cursor only needs to read the first five rows.

Como funciona un cursor adapter

4 **The user scrolls the list.**

As the user scrolls the list, the adapter asks the cursor to read more rows from the database. This works fine if the cursor's still open. But if the cursor's already been closed, the cursor adapter can't get any more data from the database.




```
public void onDestroy(){
    super.onDestroy();
    cursor.close();
    db.close();
}
```

Close the cursor and database when the activity is destroyed.

Creando el SimpleCursorAdapter

```
cursor = db.query("DRINK", new String[]{"_id", "NAME"},
                  null, null, null, null, null);
CursorAdapter listAdapter = new SimpleCursorAdapter(this,
                  android.R.layout.simple_list_item_1,
                  cursor,
                  new String[]{"NAME"},
                  new int[]{android.R.id.text1},
                  0);
listDrinks.setAdapter(listAdapter);
```

This is the cursor from the previous page.

This is the same layout we used with the array adapter. It displays a single value for each row in the list view.

This is the cursor.

Display the contents of the NAME column in the ListView text views.

Use setAdapter() to connect the adapter to the list view.

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(Context context,
                  int layout,
                  Cursor cursor,
                  String[] fromColumns,
                  int[] toViews,
                  int flags)
```

How to display the data. You can use the same layout you used with an array adapter.

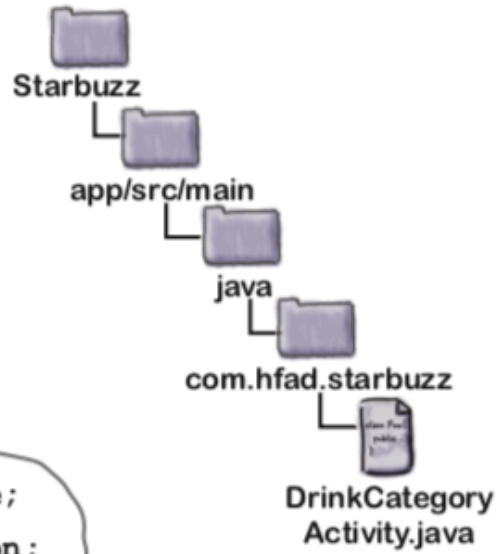
The cursor you create. The cursor should include the _id column, and the data you want to appear.

Which columns in the cursor to match to which views

Used to determine the behavior of the cursor.

El código revisado de DrinkCategoryActivity

```
package com.hfad.starbuzz;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;
import android.view.View;
import android.content.Intent;
import android.widget.AdapterView;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;

public class DrinkCategoryActivity extends Activity {

    private SQLiteDatabase db;
    private Cursor cursor;
```



We're using these extra classes, so you need to import them.

We're adding these as private variables so we can close the database and cursor in our onDestroy() method.

El código revisado de DrinkCategoryActivity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_drink_category);
    ArrayAdapter<Drink> listAdapter = new ArrayAdapter<>(
            this,
            android.R.layout.simple_list_item_1,
            Drink.drinks);
    ListView listDrinks = (ListView) findViewById(R.id.list_drinks);
    SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
    try {
        db = starbuzzDatabaseHelper.getReadableDatabase();
        cursor = db.query("DRINK",
                new String[]{"_id", "NAME"},
                null, null, null, null, null);
        SimpleCursorAdapter listAdapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_1,
                cursor,
                new String[]{"NAME"},
                new int[]{android.R.id.text1},
                0);
        listDrinks.setAdapter(listAdapter);
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }
```

We're no longer using an array adapter, so delete these lines of code.

Get a reference to the database.

Create the cursor.

Create the cursor adapter.

Map the contents of the NAME column to the text in the ListView.

Set the adapter to the ListView.

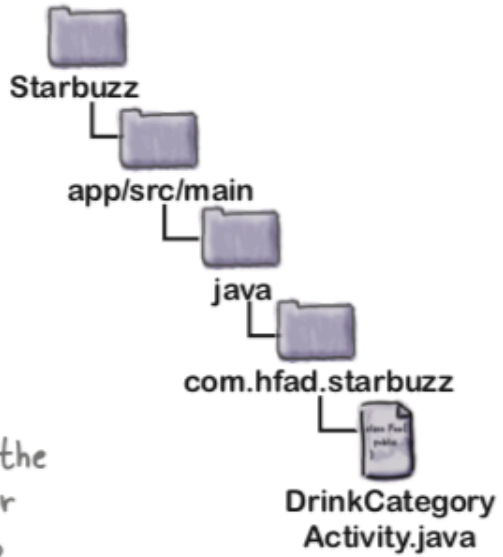
Display a message to the user if a SQLiteException gets thrown.

El código revisado de DrinkCategoryActivity

```
        //Create a listener to listen for clicks in the list view
        AdapterView.OnItemClickListener itemClickListener =
                new AdapterView.OnItemClickListener(){
                    public void onItemClick(AdapterView<?> listDrinks,
                                            View itemView,
                                            int position,
                                            long id) {
                        //Pass the drink the user clicks on to DrinkActivity
                        Intent intent = new Intent(DrinkCategoryActivity.this,
                                DrinkActivity.class);
                        intent.putExtra(DrinkActivity.EXTRA_DRINKID, (int) id);
                        startActivity(intent);
                    }
                };

        //Assign the listener to the list view
        listDrinks.setOnItemClickListener(itemClickListener);
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        cursor.close();
        db.close();
    }
}
```

We didn't need to change any of the listener code.

We're closing the database and cursor in the activity's onDestroy() method. The cursor will stay open until the cursor adapter no longer needs it.



Prueba


2:27
Starbuzz
Starbuzz Coffee
Drinks
Food
Stores
2:28
Starbuzz
Latte
Cappuccino
Filter
We clicked on the Latte option...
2:28
Starbuzz
...and here are details of the Latte.
Latte
A couple of espresso shots with steamed milk

BULLET POINTS

- A **cursor** lets you read from and write to the database.
- You create a cursor by calling the `SQLiteDatabase query()` method. Behind the scenes, this builds a SQL `SELECT` statement.
- The `getWritableDatabase()` method returns a `SQLiteDatabase` object that allows you to read from and write to the database.
- The `getReadableDatabase()` returns a `SQLiteDatabase` object. This gives you read-only access to the database. It *may* also allow you to write to the database, but this isn't guaranteed.
- Navigate through a cursor using the `moveTo*()` methods.
- Get values from a cursor using the `get*()` methods. Close cursors and database connections after you've finished with them.
- A **cursor adapter** is an adapter that works with cursors. Use `SimpleCursorAdapter` to populate a list view with the values returned by a cursor.