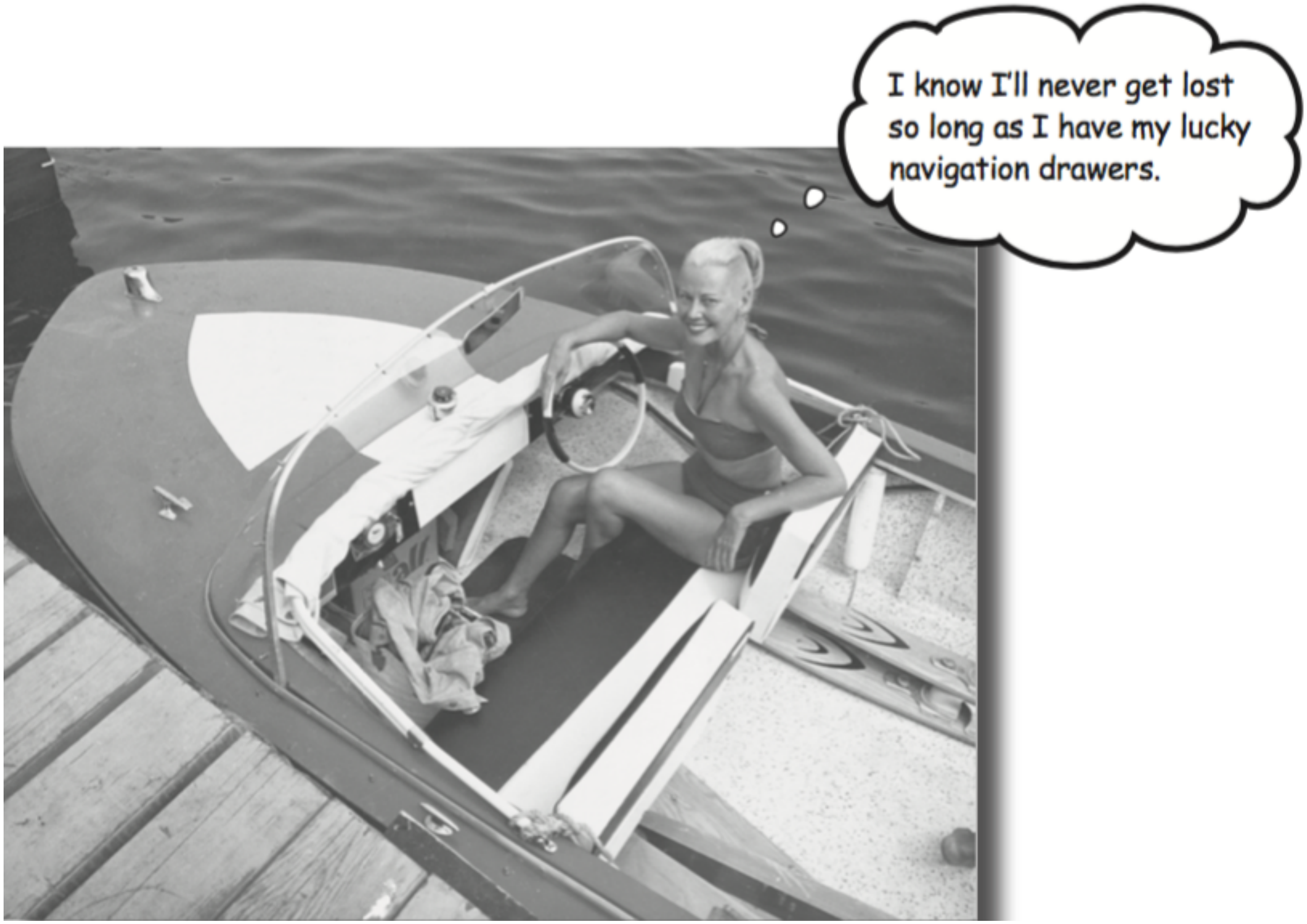
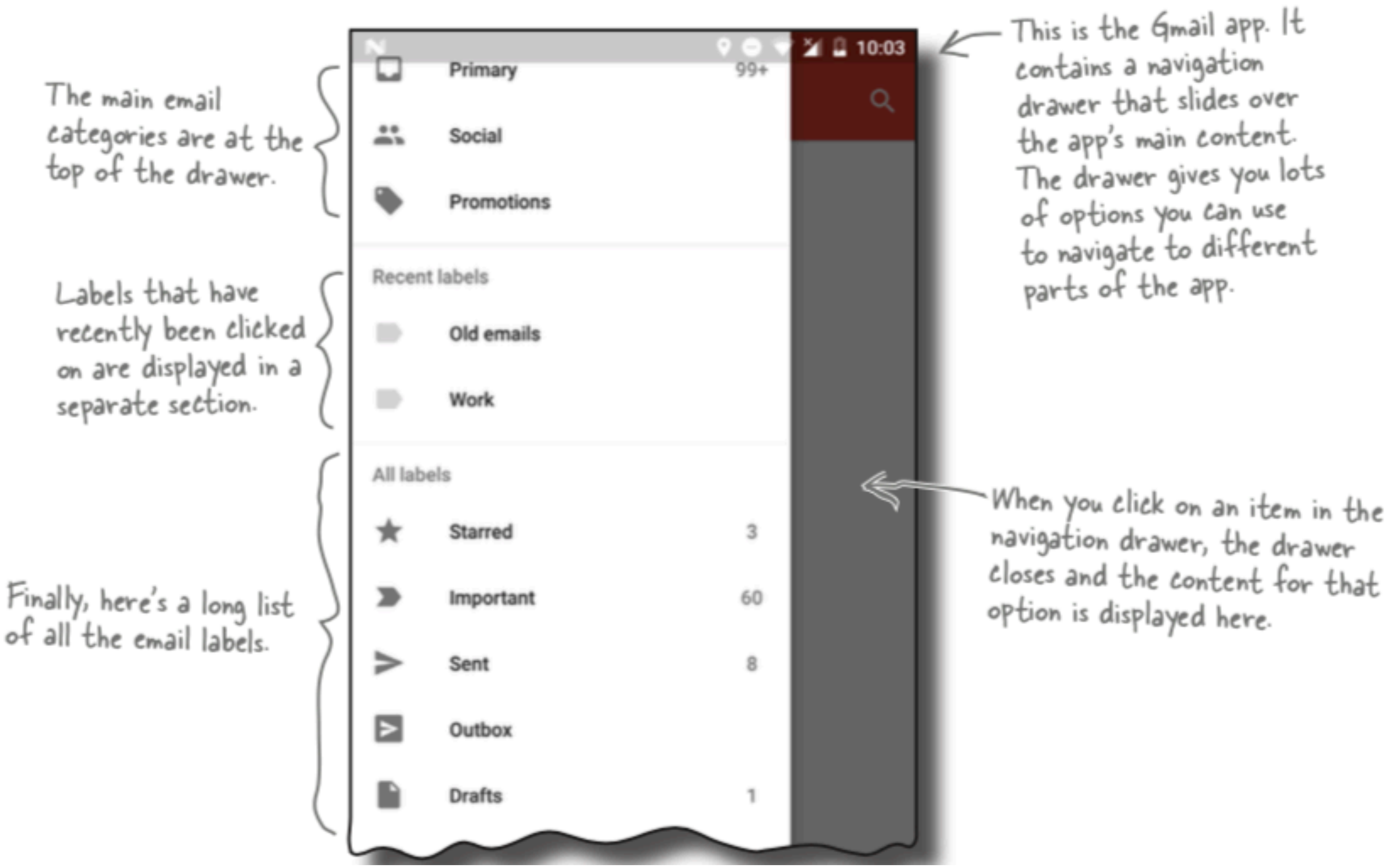
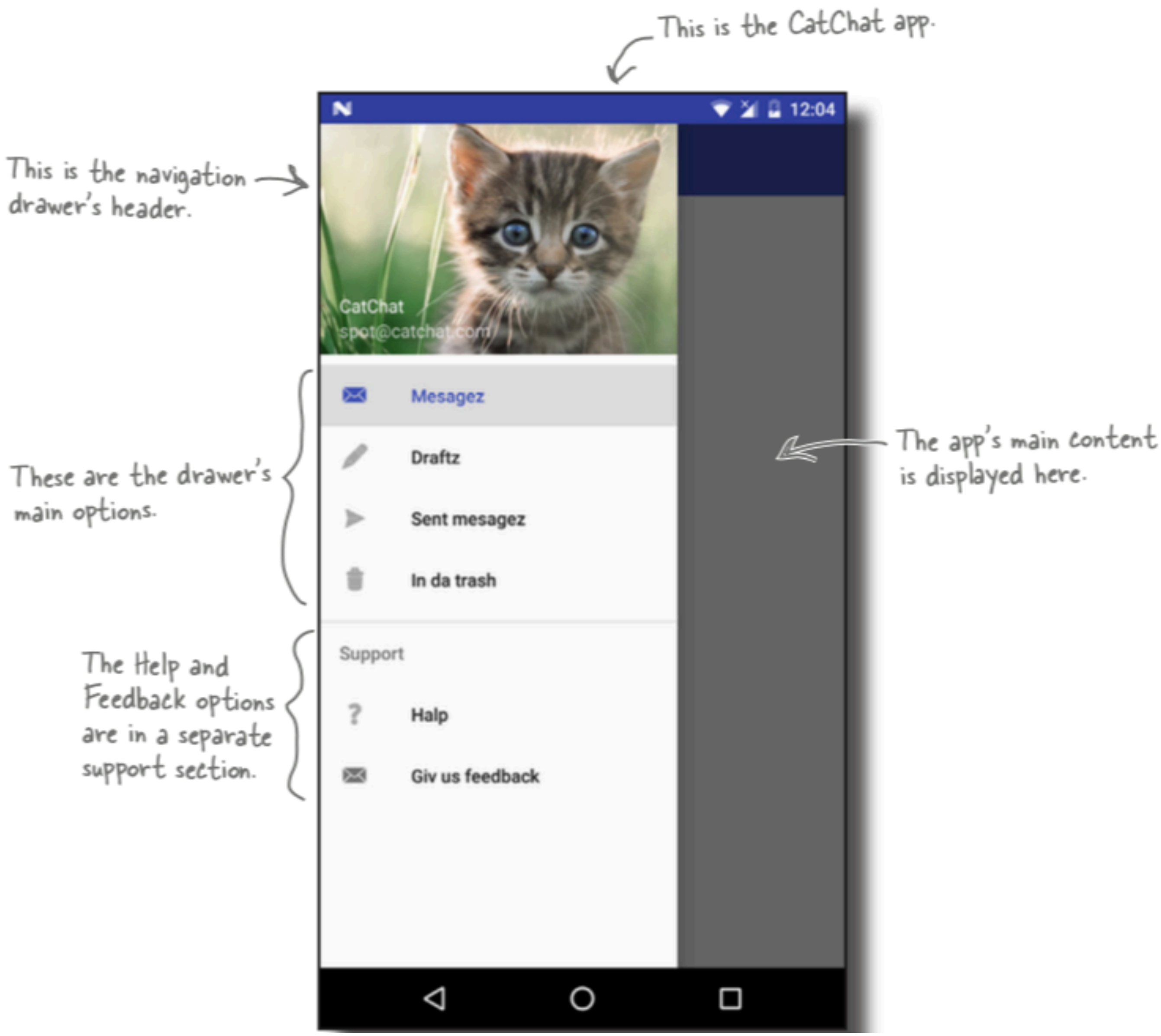
14
Opciones de navegación



Los *navigation drawers* permiten mostrar más opciones

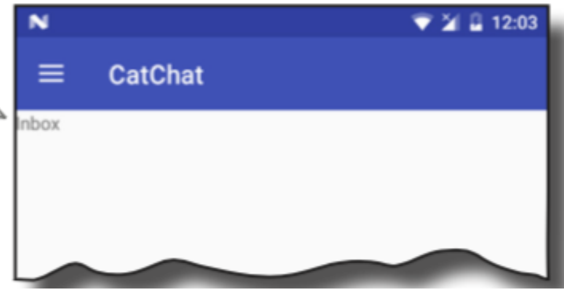


Crearemos una nueva app que utiliza un *navigation drawer*



Como funciona un navigation drawer

Here the drawer's closed, so it looks like a plain old activity.

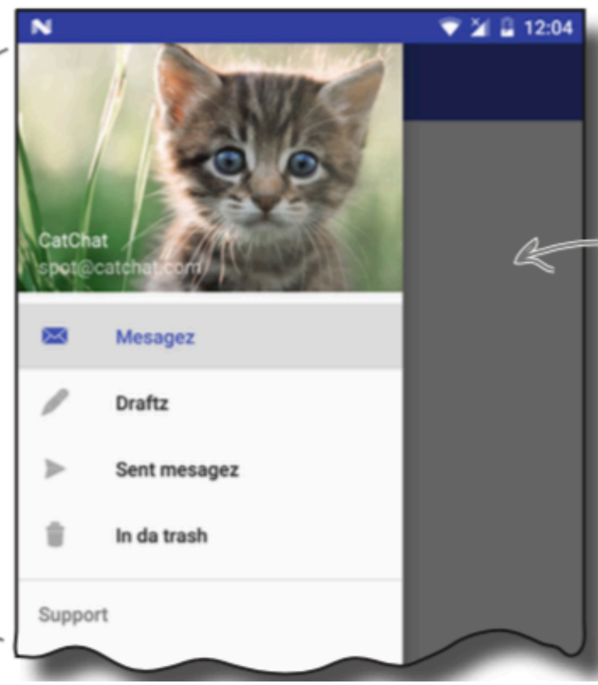


The main content of the activity usually is composed of a toolbar and a frame layout that's used to display fragments.

1. **A view for the main content.**
   This is usually a layout containing a toolbar and a frame layout, which you use to display fragments.

2. **A view for the drawer contents.**
   This is usually a navigation view, which controls most of the drawer's behavior.

The drawer's contents are defined by a navigation view.



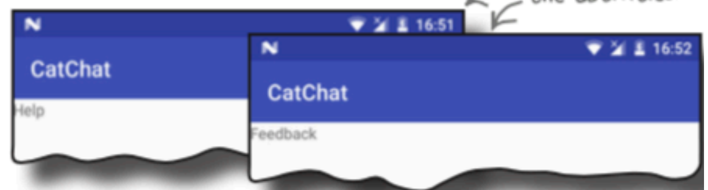
The drawer slides over the main content when it opens.

Qué vamos a hacer ?

1. **Create basic fragments and activities for the app's contents.**
When the user clicks on one of the options in the navigation drawer, we want to display the fragment or activity for that option. We'll create the fragments `InboxFragment`, `DraftsFragment`, `SentItemsFragment`, and `TrashFragment`, and activities `HelpActivity` and `FeedbackActivity`.

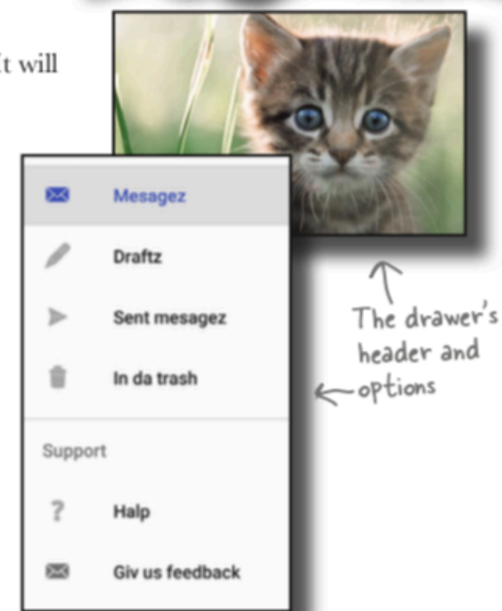
These are the fragments.

These are the activities.



2. **Create the drawer's header.**
We'll build a layout, *nav_header.xml*, for the drawer's header. It will contain an image and text.

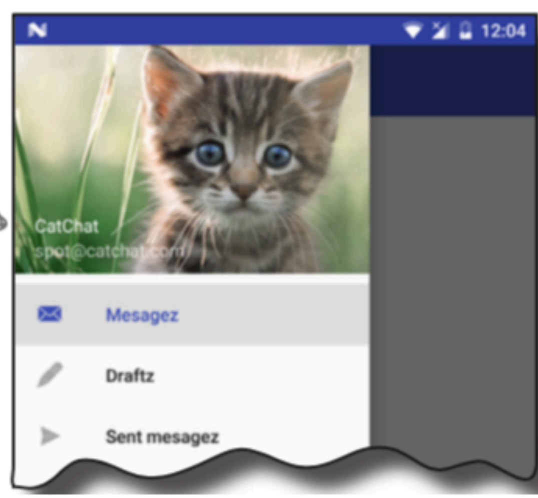
3. **Create the drawer's options.**
We'll build a menu, *menu_nav.xml*, for the options the drawer will display.



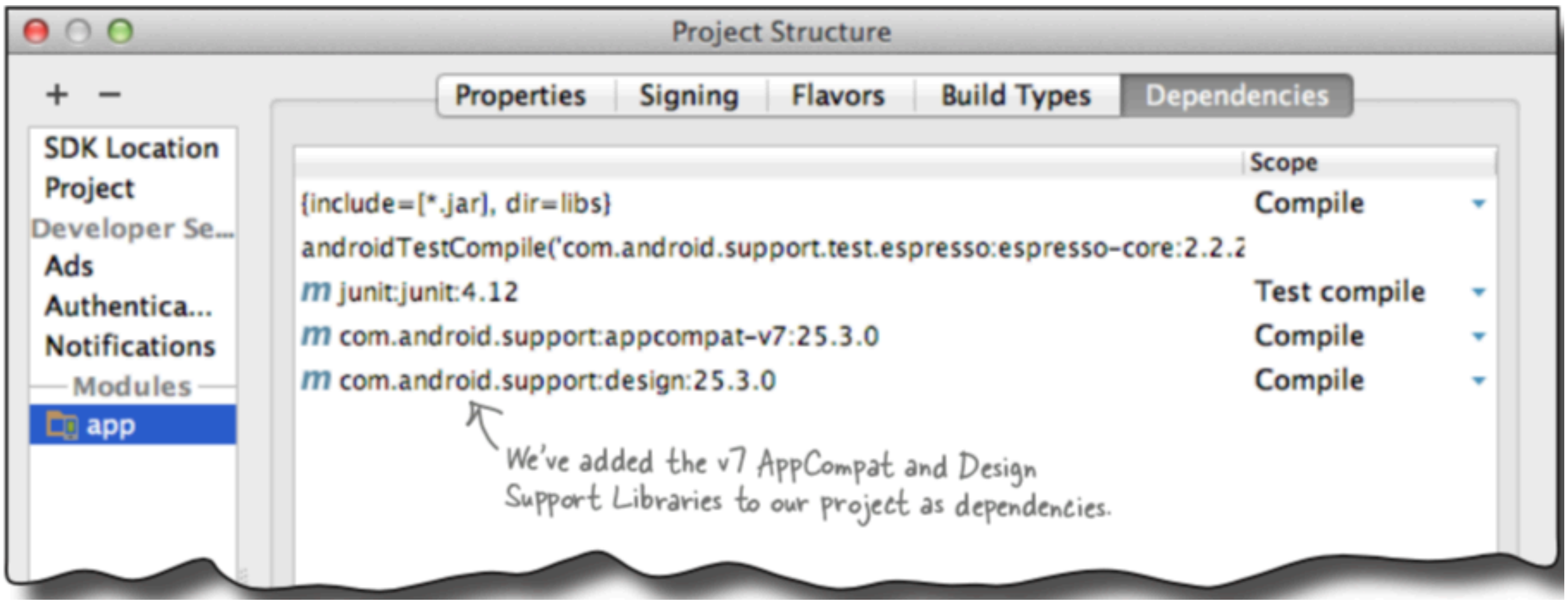
The drawer's header and options

4. **Create the navigation drawer.**
We'll add the navigation drawer to the app's main activity, and get it to display the header and options. We'll then write activity code to control the drawer's behavior.

We'll create this navigation drawer.



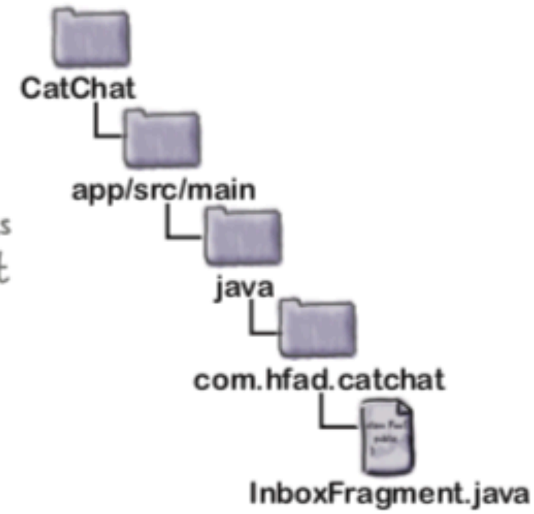
Creando el proyecto CatChat



Creando el fragmento InboxFragment

```java
package com.hfad.catchat;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class InboxFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_inbox, container, false);
    }
}
```

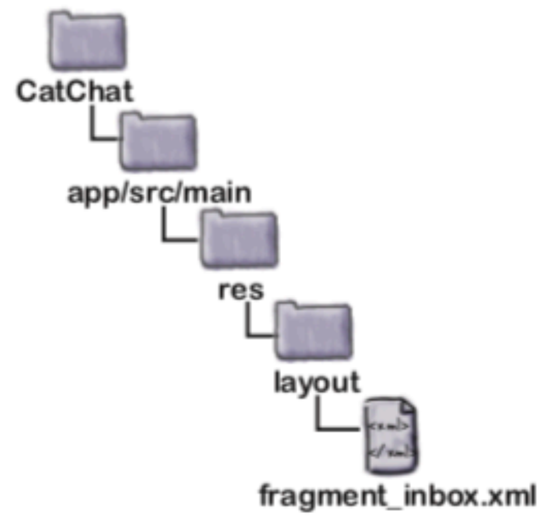
All the fragments use the Fragment class from the Support Library.



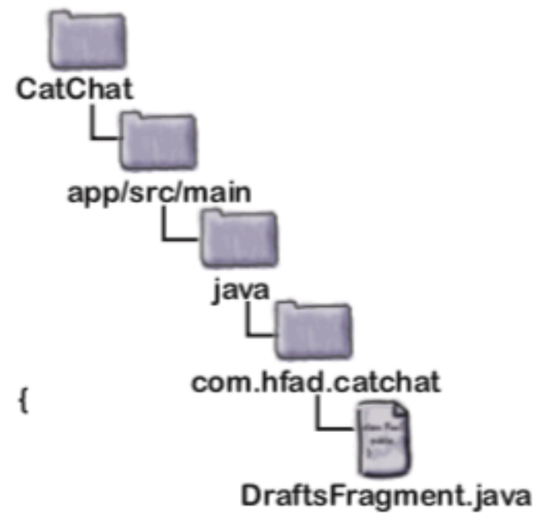
Creando el layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.hfad.catchat.InboxFragment">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Inbox" />
</LinearLayout>
```

InboxFragment's layout just contains a TextView. We're adding this text so we can easily tell when it's displayed.



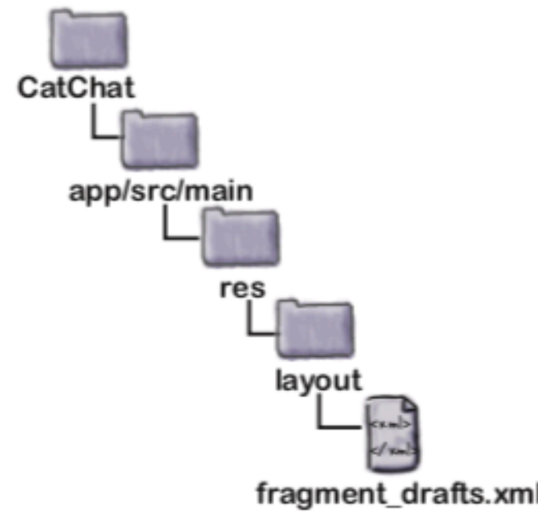
Creando el fragmento DraftsFragment




```
package com.hfad.catchat;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class DraftsFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_drafts, container, false);
    }
}
```

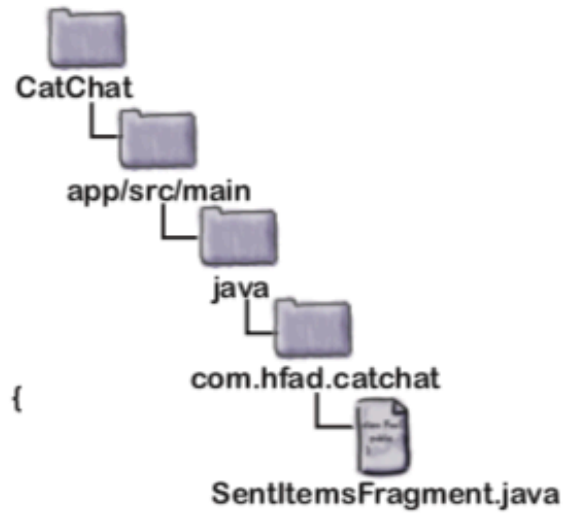
Creando el layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.hfad.catchat.DraftsFragment">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Drafts" />
</LinearLayout>
```



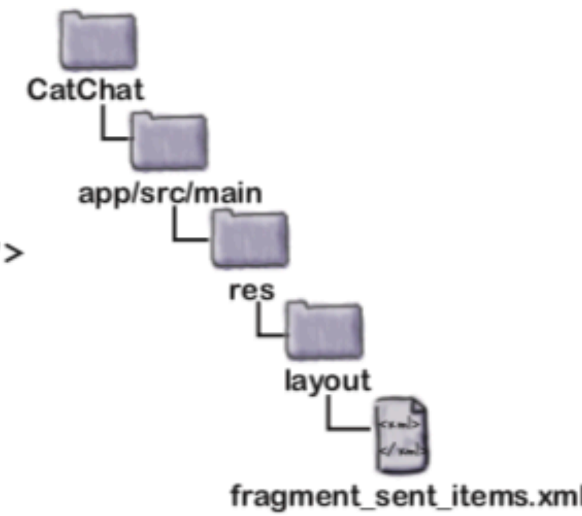
Creando el fragmento SentItemsFragment

```
package com.hfad.catchat;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class SentItemsFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_sent_items, container, false);
    }
}
```

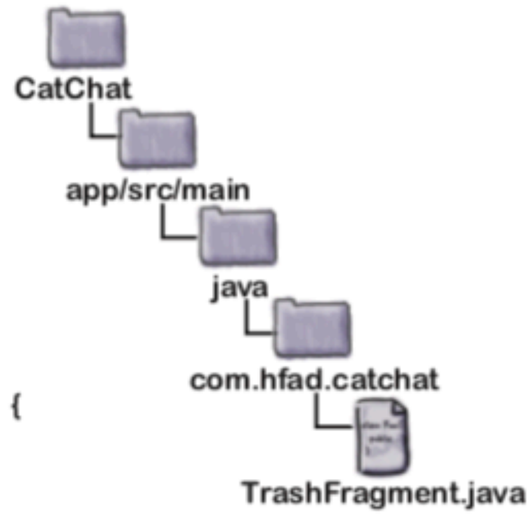


Creando el layout

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.hfad.catchat.SentItemsFragment">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Sent items" />
</LinearLayout>
```



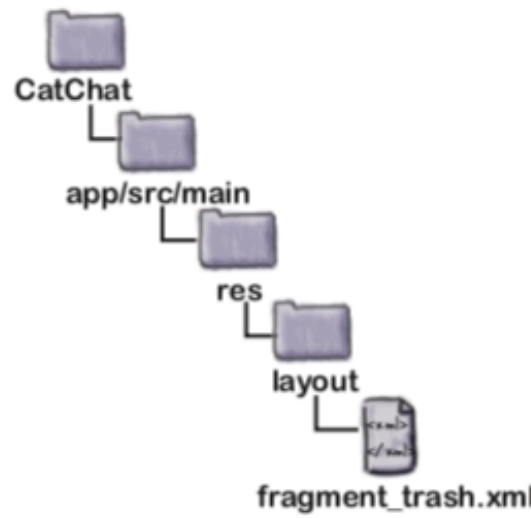
Creando el fragmento TrashFragment




```java
package com.hfad.catchat;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class TrashFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_trash, container, false);
    }
}
```

Creando el layout

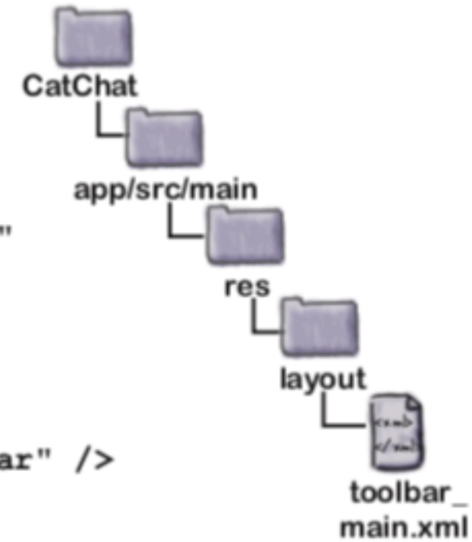
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.hfad.catchat.TrashFragment">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Trash" />
</LinearLayout>
```



Creando un layout para la barra

This is the same toolbar code we've used in previous chapters.

```
<android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />
```




```
<?xml version="1.0" encoding="utf-8"?>
<manifest ...>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            ...
        </activity>
    </application>
</manifest>
```

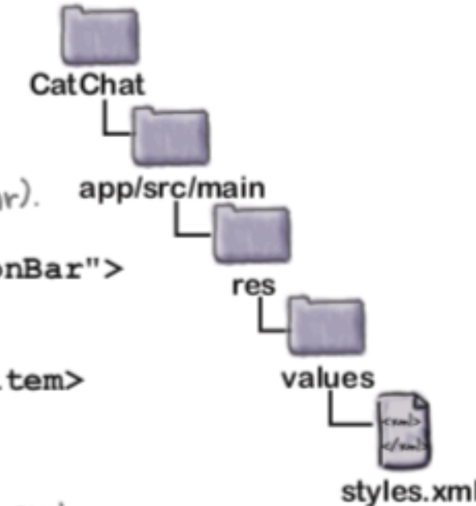
Android Studio may have already added this value for you.

Actualizando el tema de la aplicación

This theme removes the default app bar (we're replacing it with a toolbar).

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Android Studio may have added these colors for you.



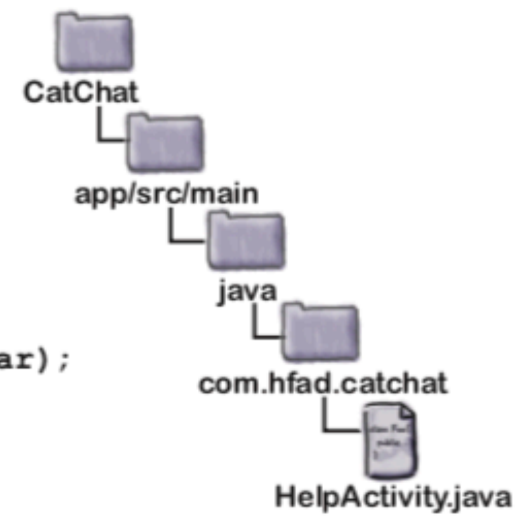

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Add these colors if Android Studio hasn't already done it for you.

Creando la actividad HelpActivity

```
package com.hfad.catchat;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;

public class HelpActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_help);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }
}
```

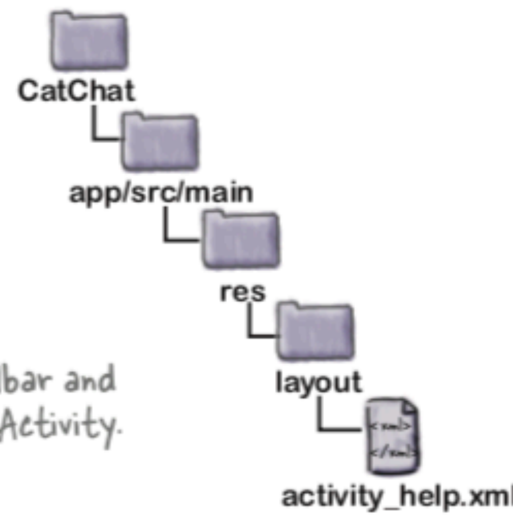
The activity needs to extend AppCompatActivity because we're using an AppCompat theme.



Y su layout


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.hfad.catchat.HelpActivity">

    <include
        layout="@layout/toolbar_main"
        android:id="@+id/toolbar" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Help" />
</LinearLayout>
```

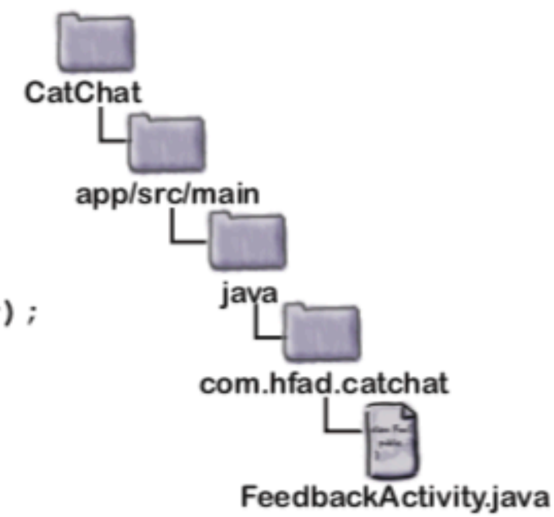
We're adding a toolbar and "Help" text to HelpActivity.



Creando la actividad FeedbackActivity

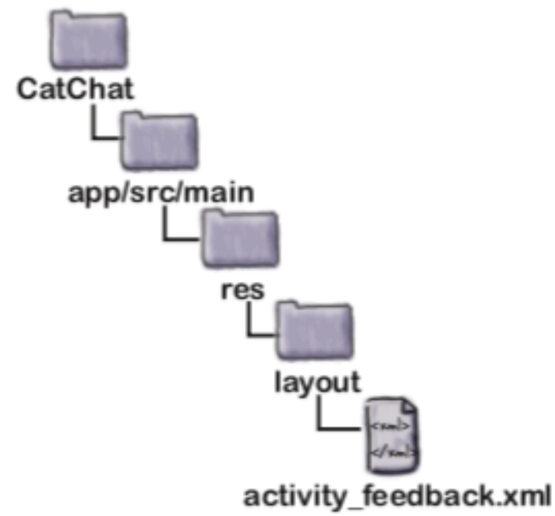
```java
package com.hfad.catchat;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;

public class FeedbackActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_feedback);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }
}
```

This activity needs to extend AppCompatActivity as well.



Y su layout


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.hfad.catchat.FeedbackActivity">

    <include
        layout="@layout/toolbar_main"
        android:id="@+id/toolbar" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Feedback" />
</LinearLayout>
```



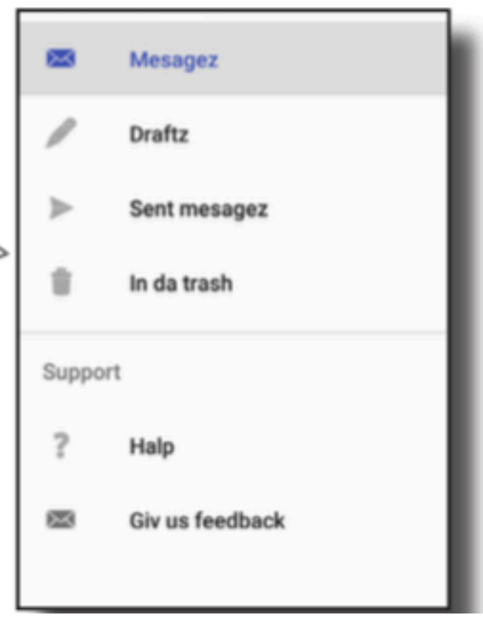
Construyendo el *navigation drawer*

A navigation drawer header.
This is a layout that appears at the top of the navigation drawer. It usually consists of an image with some text, for example a photo of the user and their email account.

This is the header we'll create. It consists of an image and two pieces of text.

A set of options.
You define a set of options to be displayed in the navigation drawer underneath the header. When the user clicks on one of these options, the screen for that option is displayed as a fragment within the navigation drawer's activity, or as a new activity.

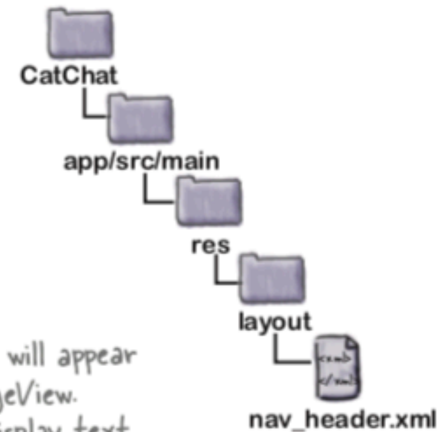
The navigation drawer will contain these options.



```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="180dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/kitten_small" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="bottom|start"
        android:layout_margin="16dp" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/app_name"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/user_name" />
    </LinearLayout>
</FrameLayout>
```

We're explicitly setting the height of the layout to 180dp so that it doesn't take up too much space in the drawer.

The image background is quite dark, so we're using this line to make the text light.



This LinearLayout will appear on top of the ImageView. We're using it to display text at the bottom of the image.

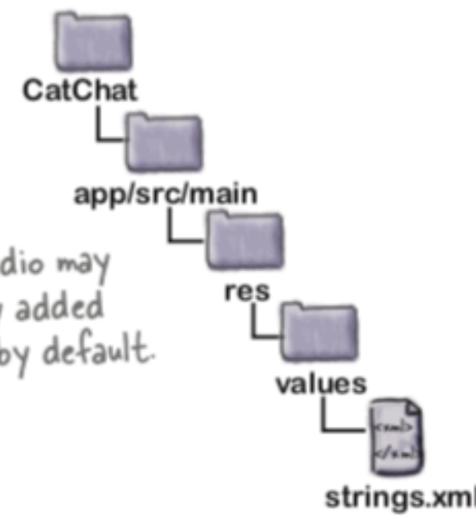
This is a built-in style that makes the text look slightly bolder. It comes from the AppCompat Support Library.

Construyendo el encabezado

Y actualizando las cadenas

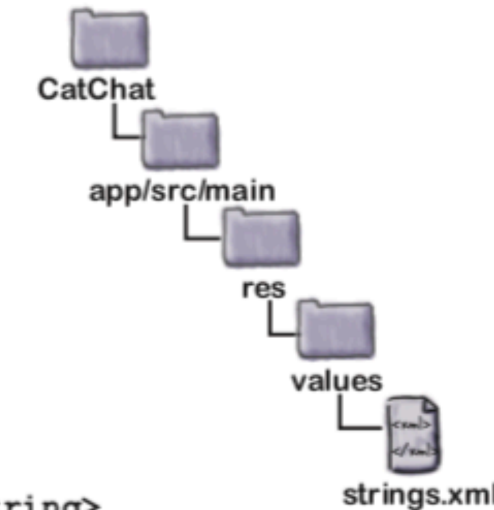
```
<resources>
    ...
    <string name="app_name">CatChat</string>
    <string name="user_name">spot@catchat.com</string>
</resources>
```

Android Studio may have already added this String by default.

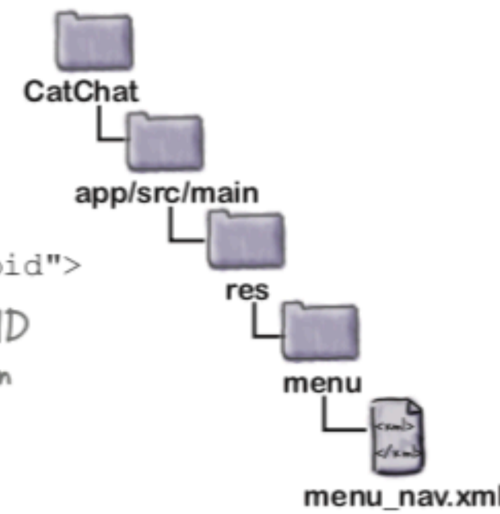


Las opciones del menu

```
<resources>
    ...
    <string name="nav_inbox">Mesagez</string>
    <string name="nav_drafts">Draftz</string>
    <string name="nav_sent">Sent mesagez</string>
    <string name="nav_trash">In da trash</string>
    <string name="nav_support">Support</string>
    <string name="nav_help">Halp</string>
    <string name="nav_feedback">Giv us feedback</string>
</resources>
```



Agregar elementos en el orden apropiado

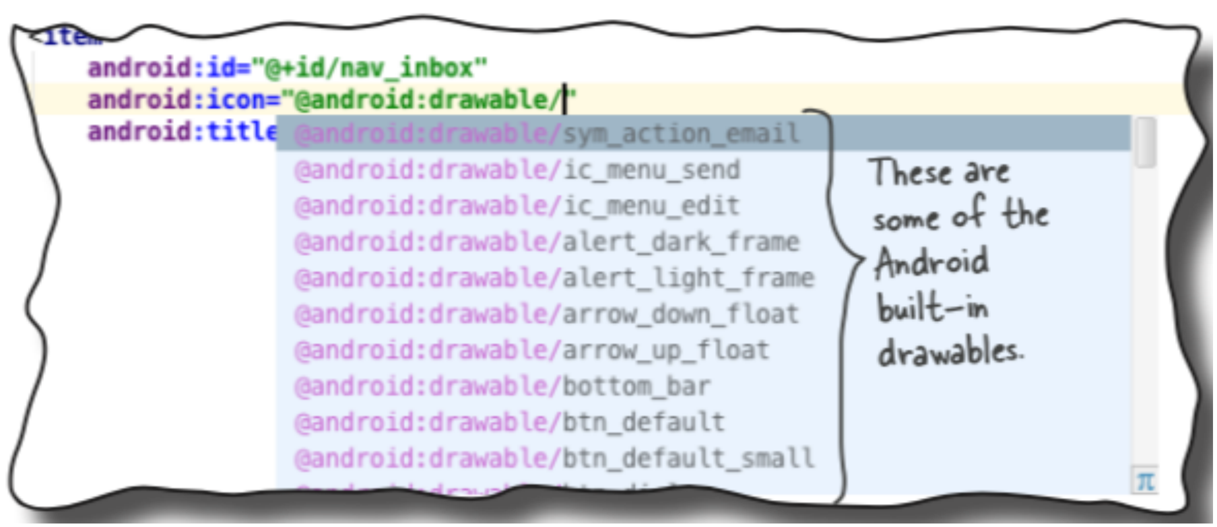



```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/nav_inbox"
        android:icon="@android:drawable/sym_action_email"
        android:title="@string/nav_inbox" />

    ...

</menu>
```

You need to give the item an ID so that your activity code can respond to it being clicked.

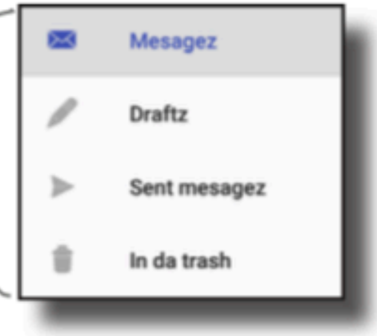
This is a built-in drawable you can use to display an email icon.

This is the text that appears in the navigation drawer.



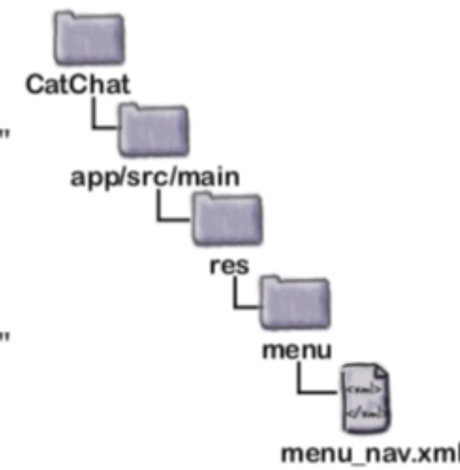
Agrupando elementos

The code on this page adds these four items




```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_inbox"
            android:icon="@android:drawable/sym_action_email"
            android:title="@string/nav_inbox"
            android:checked="true" />
        <item
            android:id="@+id/nav_drafts"
            android:icon="@android:drawable/ic_menu_edit"
            android:title="@string/nav_drafts" />
        <item
            android:id="@+id/nav_sent"
            android:icon="@android:drawable/ic_menu_send"
            android:title="@string/nav_sent" />
        <item
            android:id="@+id/nav_trash"
            android:icon="@android:drawable/ic_menu_delete"
            android:title="@string/nav_trash" />
    </group>

</menu>
```

Add this group and the four items it contains to your menu resource file so they'll appear in the navigation drawer.

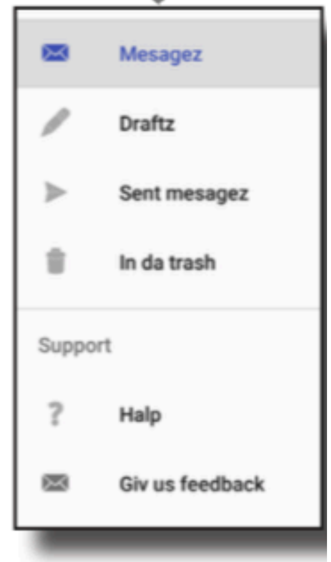


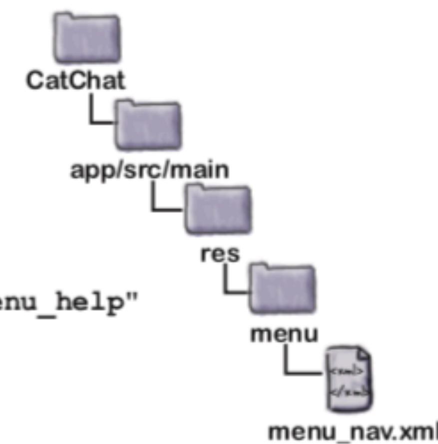
These are the main options.

This is the support section.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_inbox"
            android:icon="@android:drawable/sym_action_email"
            android:title="@string/nav_inbox" />
        <item
            android:id="@+id/nav_drafts"
            android:icon="@android:drawable/ic_menu_edit"
            android:title="@string/nav_drafts" />
        <item
            android:id="@+id/nav_sent"
            android:icon="@android:drawable/ic_menu_send"
            android:title="@string/nav_sent" />
        <item
            android:id="@+id/nav_trash"
            android:icon="@android:drawable/ic_menu_delete"
            android:title="@string/nav_trash" />
    </group>

    <item android:title="@string/nav_support">
        <menu>
            <item
                android:id="@+id/nav_help"
                android:icon="@android:drawable/ic_menu_help"
                android:title="@string/nav_help"/>
            <item
                android:id="@+id/nav_feedback"
                android:icon="@android:drawable/sym_action_email"
                android:title="@string/nav_feedback" />
        </menu>
    </item>
</menu>
```

The code on this page creates the full menu.





Agregando un submenú

Creando el *navigation drawer*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

The DrawerLayout defines the drawer.

You give it an ID so you can refer to it in your activity code.

The DrawerLayout's first view is a layout for the activity's main content. You see it when the drawer is closed.

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
        ...
    </LinearLayout>
```

The NavigationView defines the drawer's contents.

```
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/menu_nav" />
</android.support.v4.widget.DrawerLayout>
```

This attaches the drawer to the start edge of the activity (the left for left-to-right languages).

This is the layout for the drawer's header.

This is the menu resource file containing the drawer's options.

El código completo de activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <include
            layout="@layout/toolbar_main"
            android:id="@+id/toolbar" />

        <FrameLayout
            android:id="@+id/content_frame"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </LinearLayout>

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/menu_nav" />
</android.support.v4.widget.DrawerLayout>
```

The layout's root element is a DrawerLayout.

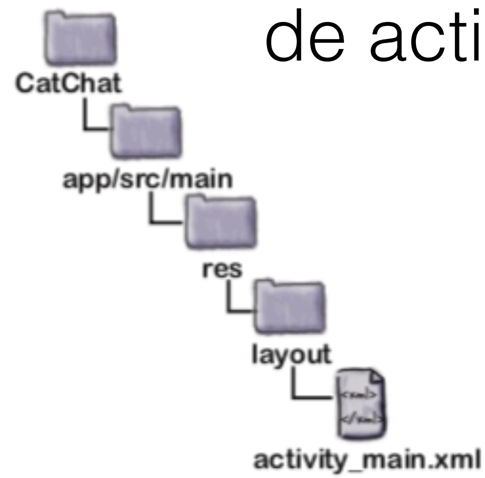
It has an ID so we can refer to it in our activity code later.

This is for the drawer's main content.

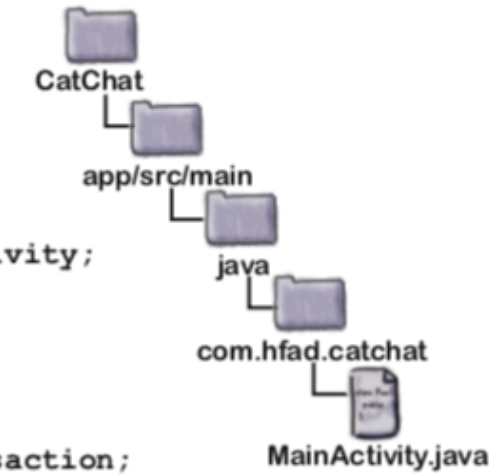
The activity's main content is composed of a Toolbar, and a FrameLayout in which we'll display fragments.

The NavigationView defines the drawer's appearance and much of its behavior. We're giving it an ID, as we'll need to refer to it in our activity code.

We're using the layout we created earlier as the drawer's header, and the menu resource file for the list of options.



Agregando InboxFragment a MainActivity



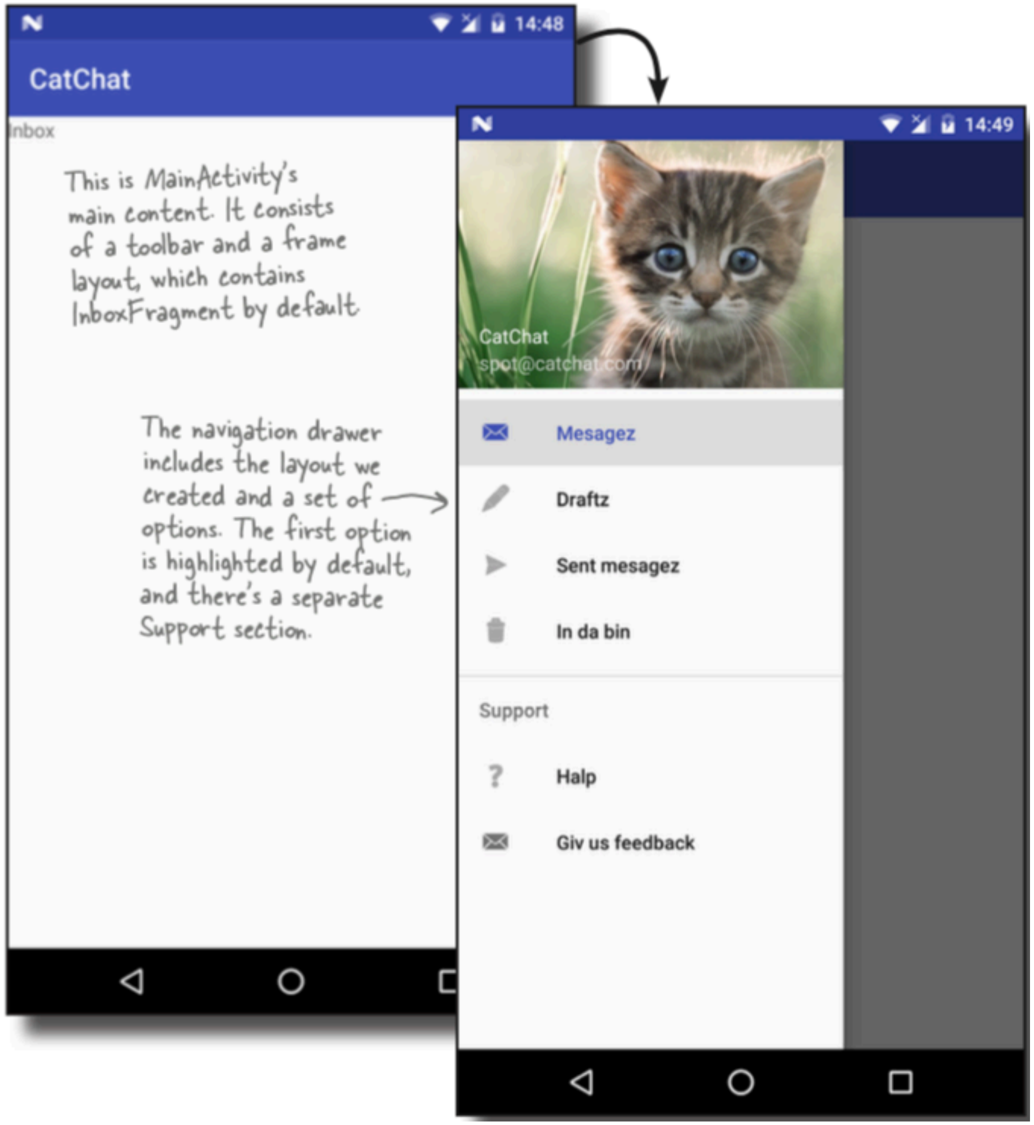

```
package com.hfad.catchat;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        Fragment fragment = new InboxFragment();
        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
        ft.add(R.id.content_frame, fragment);
        ft.commit();
    }
}
```

Make sure the activity extends the AppCompatActivity class, as we're using an AppCompat theme and support fragments.

Set the Toolbar as the activity's app bar.

Use a fragment transaction to display an instance of InboxFragment

Prueba



14:48
CatChat
Inbox
This is MainActivity's main content. It consists of a toolbar and a frame layout, which contains InboxFragment by default.
The navigation drawer includes the layout we created and a set of options. The first option is highlighted by default, and there's a separate Support section.
14:49
CatChat
spot@catchat.com
Mesagez
Draftz
Sent mesagez
In da bin
Support
Halp
Giv us feedback

Qué debe hacer la actividad ?

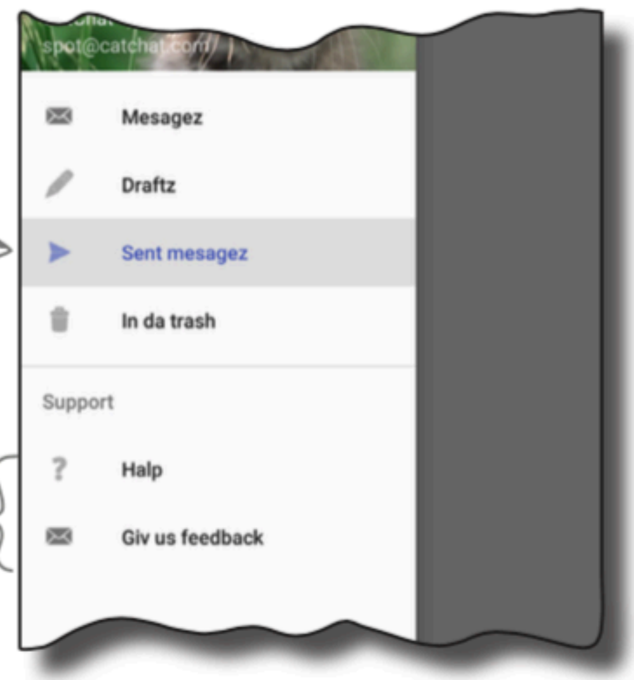
1. **Add a drawer toggle.**
This provides a visual sign to the user that the activity contains a navigation drawer. It adds a "burger" icon to the toolbar, and you can click on this icon to open the drawer.

This is the "burger" icon. Clicking on it opens the navigation drawer.

2. **Make the drawer respond to clicks.**
When the user clicks on one of the options in the navigation drawer, we'll display the appropriate fragment or activity and close the drawer.

When the user clicks on one of the main options, we'll display the fragment for that option and close the drawer. The option will be highlighted in the navigation drawer the next time we open it.

When the user clicks on one of these options, we'll start the appropriate activity.



3. **Close the drawer when the user presses the Back button.**
If the drawer's open, we'll close it when the user clicks on the Back button. If the drawer's already closed, we'll get the Back button to function as normal.

Agregando un elemento para abrir y cerrar

```java
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
...
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
                                                         drawer,
                                                         toolbar,
                                                         R.string.nav_open_drawer,
                                                         R.string.nav_close_drawer);
```

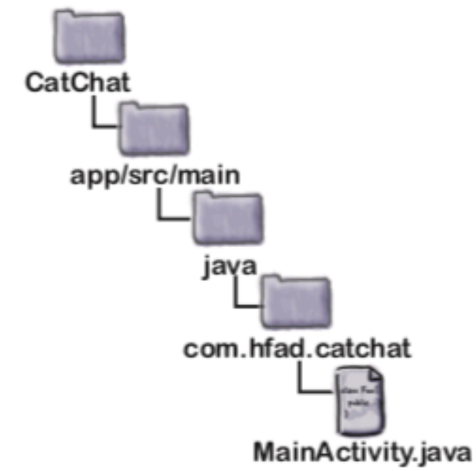
The current activity

The activity's DrawerLayout

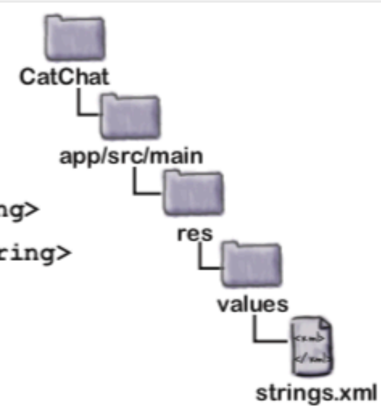
The activity's toolbar

These Strings are needed for accessibility.

This adds the burger icon to your toolbar.

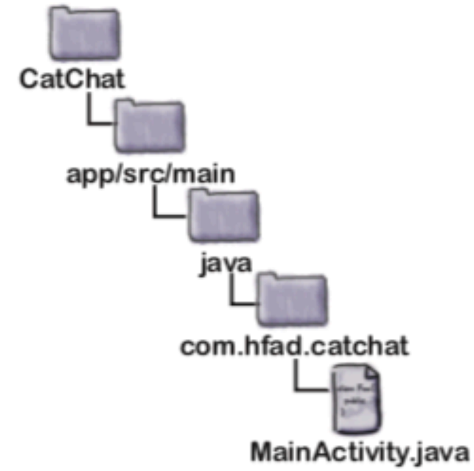



```xml
<string name="nav_open_drawer">Open navigation drawer</string>
<string name="nav_close_drawer">Close navigation drawer</string>
```




```java
drawer.addDrawerListener(toggle);

toggle.syncState();
```

Respondiendo a eventos del usuario




```java
...
import android.support.design.widget.NavigationView;

public class MainActivity extends AppCompatActivity
                              implements NavigationView.OnNavigationItemSelectedListener {
    ...
}
```

Implementing this interface means that your activity can respond to the user clicking options in the navigation drawer.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
}
```

This registers the activity as a listener on the navigation view so it will be notified if the user clicks on an item.

Implementando onNavigationItemSelected()

```
@Override
public boolean onNavigationItemSelected(MenuItem item) {

    int id = item.getItemId();
    Fragment fragment = null;
    Intent intent = null;

    switch(id){
        case R.id.nav_drafts:

            fragment = new DraftsFragment();

            break;
        case R.id.nav_sent:

            fragment = new SentItemsFragment();

            break;
        case R.id.nav_trash:

            fragment = new TrashFragment();

            break;
        case R.id.nav_help:

            intent = new Intent(this, HelpActivity.class);

            break;
```

Get the ID of the item that was selected.

Save an instance of the fragment we want to display in the fragment variable.

Construct an intent to start HelpActivity if the Help option's clicked.

```
        case R.id.nav_feedback:
            intent = new Intent( this , FeedbackActivity.class );
            break ;
        default:
            fragment = new InboxFragment() ;
    }

    if ( fragment != null) {
        FragmentTransaction ft = getSupportFragmentManager(). beginTransaction() ;
        ft.replace(R.id.content_frame, fragment );
        ft. commit() ;
    } else {
        startActivity( intent );
    }

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer. closeDrawer ( GravityCompat.START );
    return true;
}
```

If the feedback option is clicked, we need to start FeedbackActivity.

Display InboxFragment by default, as it's the first option in the drawer.

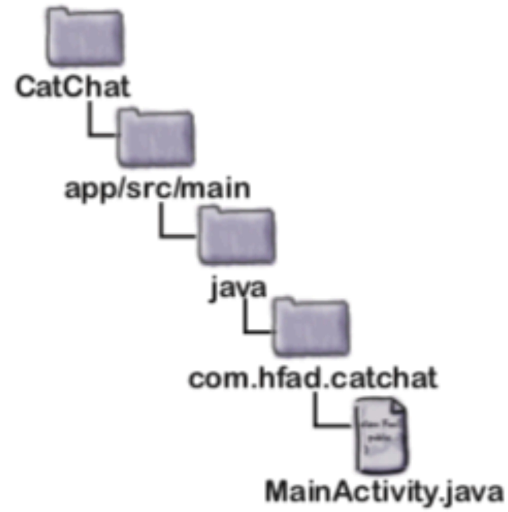
If we need to display a fragment, use a fragment transaction.

If we need to display an activity, use the intent we constructed to start it

Finally, close the drawer.

Implementando onNavigationItemSelected()

Cerrando el *navigation drawer* con el botón hacia atrás



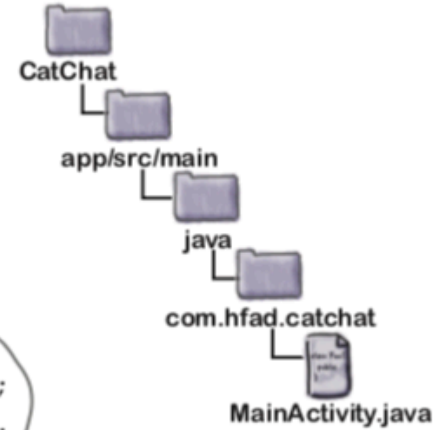
This gets called when the Back button gets pressed.

```
@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}
```

If the drawer is currently open, close it.

Otherwise, call up to the superclass onBackPressed() method.

El código completo de MainActivity




```
package com.hfad.catchat;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.design.widget.NavigationView;
import android.view.MenuItem;
import android.content.Intent;
import android.support.v4.view.GravityCompat;
```

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

```
public class MainActivity extends AppCompatActivity
                          implements NavigationView.OnNavigationItemSelectedListener {
```

Implementing this interface means the activity can listen for clicks.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
                                                                 drawer,
                                                                 toolbar,
                                                                 R.string.nav_open_drawer,
                                                                 R.string.nav_close_drawer);
```

Add a drawer toggle.

```
        drawer.addDrawerListener(toggle);
        toggle.syncState();
```

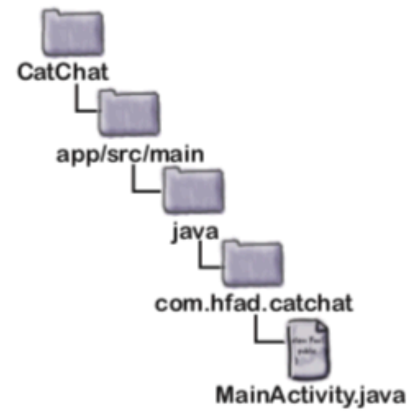
```java
    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
```

Register the activity with the navigation view as a listener.

```java
    Fragment fragment = new InboxFragment();
    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    ft.add(R.id.content_frame, fragment);
    ft.commit();
}
```

This method gets called when the user clicks on one of the items in the drawer.

```java
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();
    Fragment fragment = null;
    Intent intent = null;

    switch(id){
        case R.id.nav_drafts:
            fragment = new DraftsFragment();
            break;
        case R.id.nav_sent:
            fragment = new SentItemsFragment();
            break;
        case R.id.nav_trash:
            fragment = new TrashFragment();
            break;
        case R.id.nav_help:
            intent = new Intent(this, HelpActivity.class);
            break;
        case R.id.nav_feedback:
            intent = new Intent(this, FeedbackActivity.class);
            break;
        default:
            fragment = new InboxFragment();
    }
```



El código completo de MainActivity

```
        if (fragment != null) {
            FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            ft.replace(R.id.content_frame, fragment);
            ft.commit();
        } else {
            startActivity(intent);
        }
```

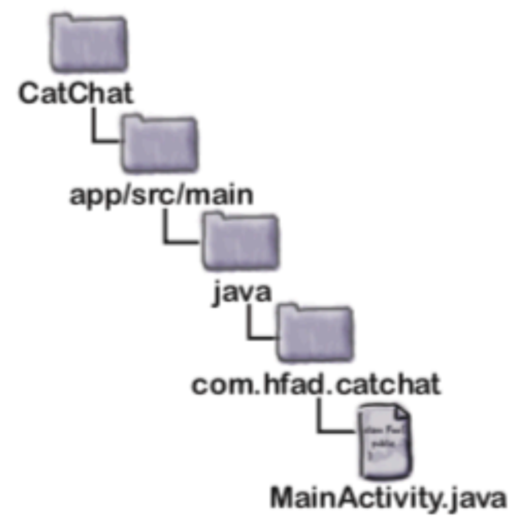
Display the appropriate fragment or activity, depending on which option in the drawer the user selects.

```
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
```

Close the drawer when the user selects one of the options.

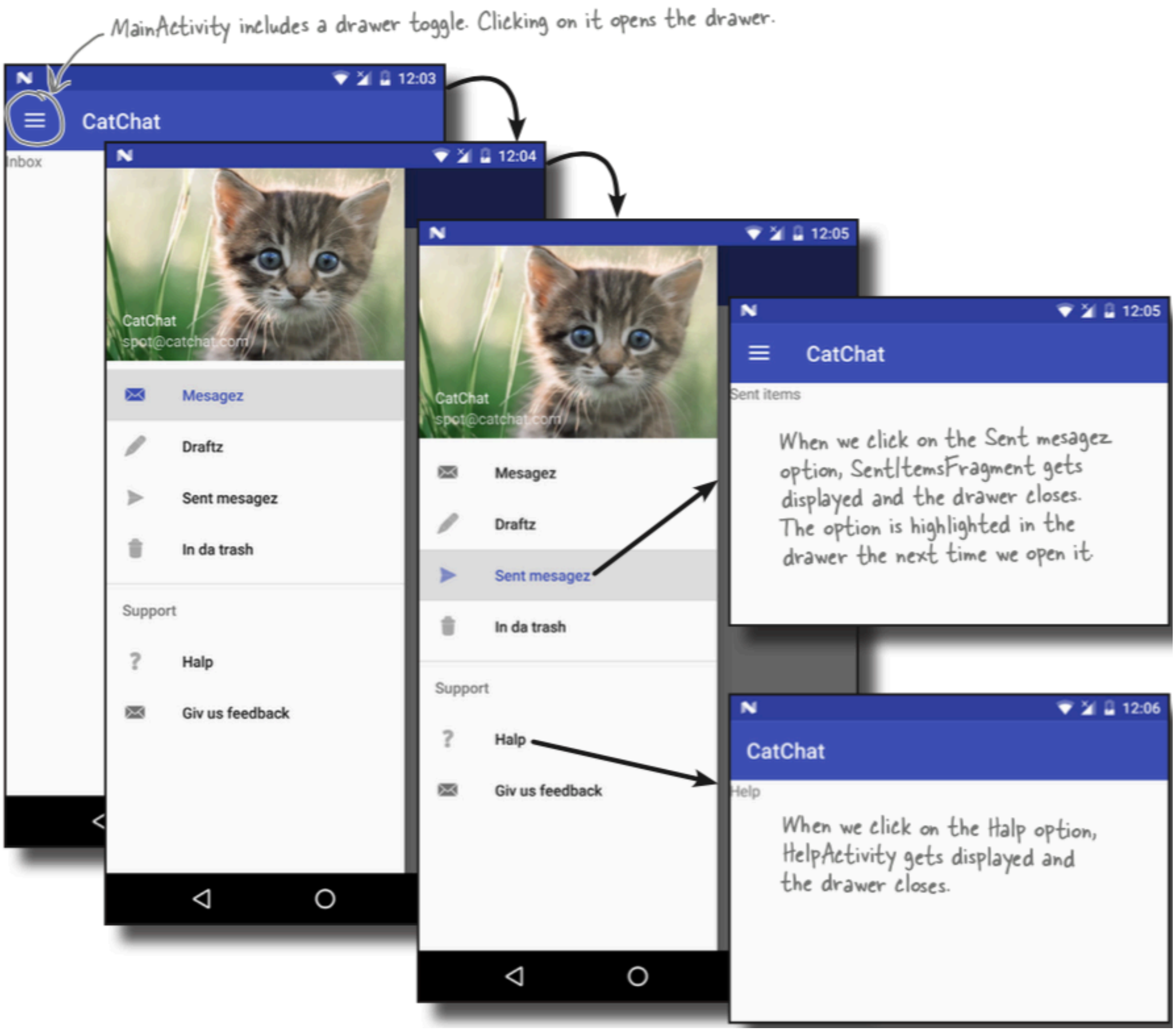
When the user presses the Back button, close the drawer if it's open.

```
    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }
}
```



El código completo de MainActivity

Prueba



MainActivity includes a drawer toggle. Clicking on it opens the drawer.
12:03
CatChat
Inbox
12:04
CatChat
spot@catchat.com
Mesagez
Draftz
Sent mesagez
In da trash
Support
Halp
Giv us feedback
12:05
CatChat
spot@catchat.com
Mesagez
Draftz
Sent mesagez
In da trash
Support
Halp
Giv us feedback
12:05
CatChat
Sent items
When we click on the Sent mesagez option, SentItemsFragment gets displayed and the drawer closes. The option is highlighted in the drawer the next time we open it.
12:06
CatChat
Help
When we click on the Halp option, HelpActivity gets displayed and the drawer closes.

BULLET POINTS

- Use a navigation drawer if you want to provide the user with a large number of shortcuts, or group them into sections.
- Create a navigation drawer by adding a **drawer layout** to your activity's layout. The drawer layout's first element needs to be a view that defines the activity's main content, usually a layout containing a `Toolbar` and `FrameLayout`. Its second element defines the contents of the drawer, usually a `NavigationView`.
- The **`NavigationView`** comes from the Design Support Library. It controls most of the drawer's behavior.
- You add a header to your drawer by creating a layout for it, and adding the header's resource ID to the navigation view's `headerLayout` attribute.
- You add items to the drawer by creating a menu resource, and adding the menu's resource ID to the navigation view's `menu` attribute.
- Add items to the menu resource in the order in which you want them to appear in the drawer.
- If you want to highlight which item in the drawer the user selects, add the menu items to a group and set the group's `checkableBehavior` attribute to `"single"`.
- Use an **`ActionBarDrawerToggle`** to display a "burger" icon in the activity's toolbar. This provides a visual sign that the activity has a navigation drawer. Clicking on it opens the drawer.
- Respond to the user clicking on items in the drawer by making your activity implement the **`NavigationView.OnNavigationItemSelectedListener`** interface. Register the activity with the navigation view as a listener, then implement the `onNavigationItemSelected()` method.
- Close the navigation drawer using the `DrawerLayout closeDrawer()` method.