17 Двухпанельные интерфейсы

В этой главе мы создадим для CriminalIntent планшетный интерфейс, в котором пользователь может одновременно видеть и взаимодействовать со списком преступлений и подробным описанием конкретного преступления. На рис. 17.1 изображен такой интерфейс, также часто называемый интерфейсом типа *«список-детализация»*.

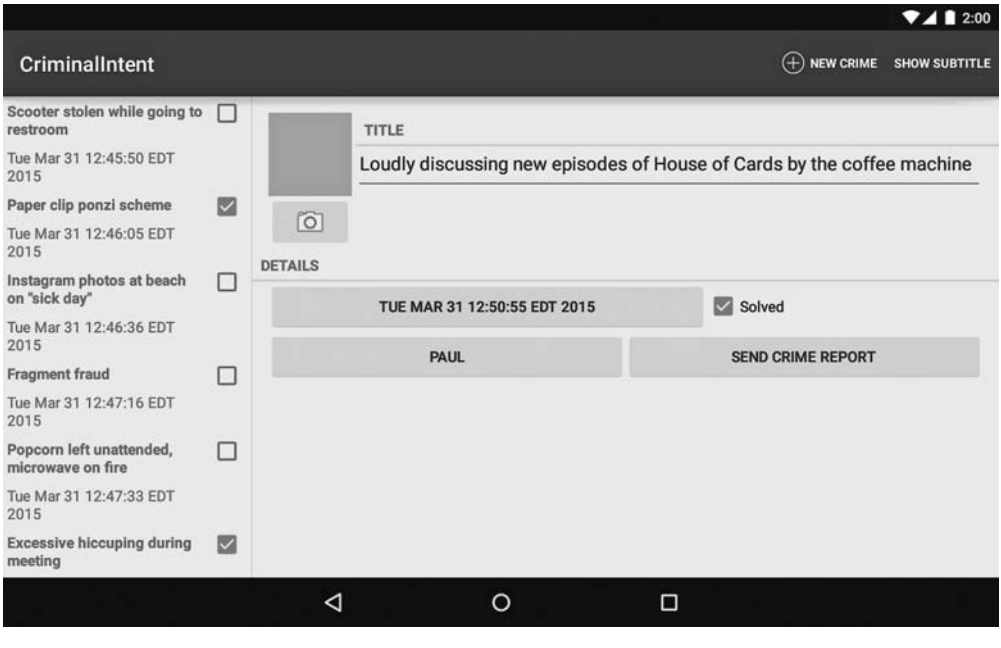

Рис. 17.1. Главное и детализированное представления одновременно находятся на экране

Для тестирования программ этой главы вам понадобится планшетное устройство или AVD. Чтобы создать виртуальный планшет AVD, выполните команду

Tools ▸ Android ▸ Android Virtual Device Manager. Щелкните на кнопке Create Virtual Device... и выберите слева категорию Tablet. Выберите аппаратный профиль, щелкните на кнопке Next и задайте для AVD целевой API не менее уровня 21 (рис. 17.2).

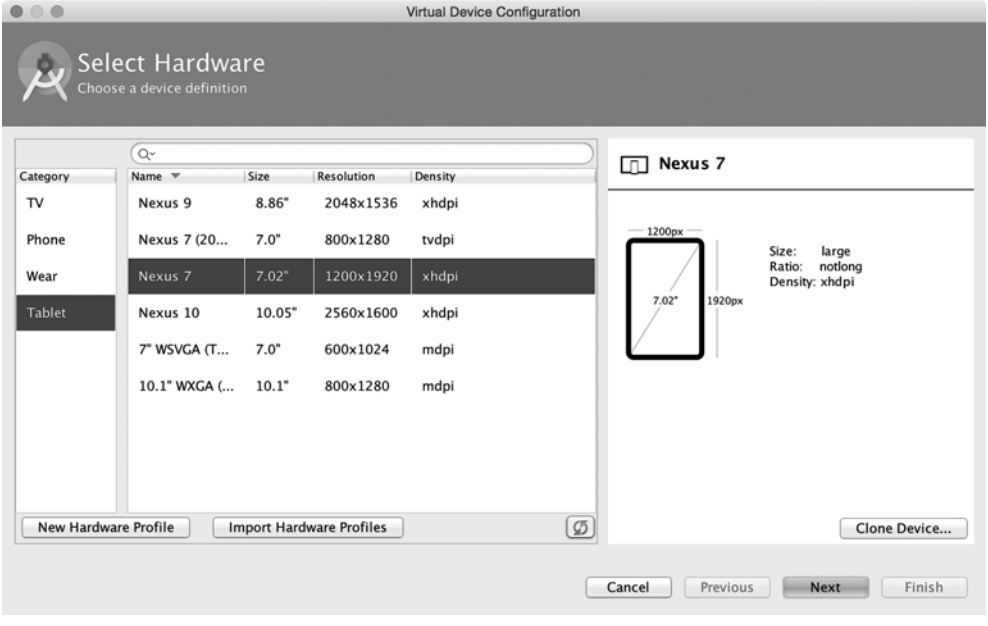

Рис. 17.2. Выбор устройства для планшетного AVD

Гибкость макета

На телефоне активность `CrimeListActivity` должна заполнять однопанельный макет, как она делает в настоящее время. На планшете она должна заполнять двухпанельный макет, способный одновременно отображать главное и детализированное представления.

В двухпанельном макете `CrimeListActivity` будет отображать как `CrimeListFragment`, так и `CrimeFragment`, как показано на рис. 17.3.

Для этого необходимо:

- изменить `SingleFragmentActivity`, чтобы выбор заполняемого макета не был жестко фиксирован в программе;
- создать новый макет, состоящий из двух контейнеров фрагментов;
- изменить `CrimeListActivity`, чтобы на телефонах заполнялся однопанельный макет, а на планшетах — двухпанельный.

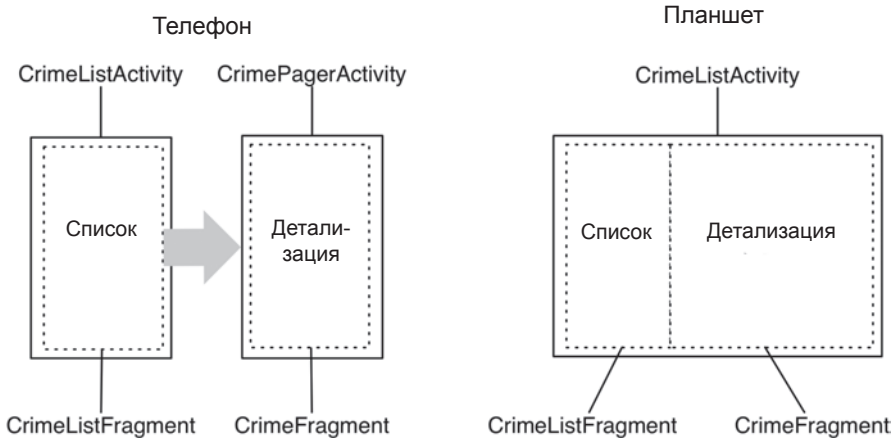

Рис. 17.3. Разновидности макетов

Модификация SingleFragmentActivity

`CrimeListActivity` является субклассом `SingleFragmentActivity`. В настоящее время класс `SingleFragmentActivity` настроен таким образом, чтобы он всегда заполнял activity_fragment.xml. Чтобы класс `SingleFragmentActivity` стал более гибким, мы сделаем так, чтобы субкласс мог предоставлять свой идентификатор ресурса макета.

В файле SingleFragmentActivity.java добавьте защищенный метод, который возвращает идентификатор макета, заполняемого активностью.

Листинг 17.1. Обеспечение гибкости SingleFragmentActivity (SingleFragmentActivity.java)

```
public abstract class SingleFragmentActivity extends AppCompatActivity {
    protected abstract Fragment createFragment();

    @LayoutRes
    protected int getLayoutResId() {
        return R.layout.activity_fragment;
}

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fragment);
        setContentView(getLayoutResId());
        FragmentManager fm = getSupportFragmentManager();
        Fragment fragment = fm.findFragmentById(R.id.fragment_container);
        if (fragment == null) {
            fragment = createFragment();
            fm.beginTransaction()
```

```
                .add(R.id.fragment_container, fragment)
                .commit();
        }
    }
}
```

Реализация класса `SingleFragmentActivity` по умолчанию будет работать так же, как и прежде, но теперь его субклассы могут переопределить `getLayoutResId()` для возвращения макета, отличного от activity_fragment.xml. Метод `getLayoutResId()` помечается аннотацией `@LayoutRes`, чтобы сообщить Android Studio, что любая реализация этого метода должна возвращать действительный идентификатор ресурса макета.

Создание макета с двумя контейнерами фрагментов

В окне инструментов Project щелкните правой кнопкой мыши на каталоге res/layout/ и создайте новый файл Android в формате XML. Убедитесь в том, что для файла выбран тип ресурса `Layout`, присвойте файлу имя activity_twopane.xml и назначьте его корневым элементом `LinearLayout`.

Используйте рис. 17.4 для построения разметки XML макета с двумя панелями.

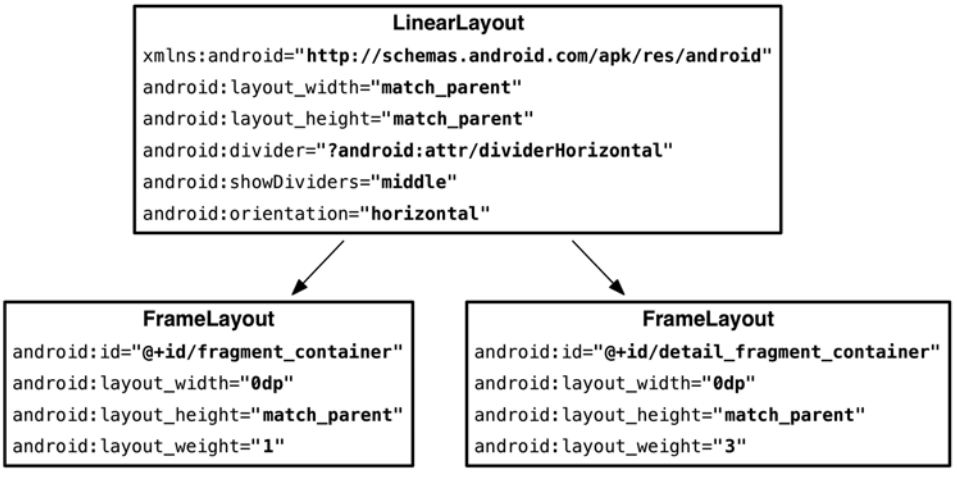

Рис. 17.4. Макет с двумя контейнерами фрагментов (layout/activity_twopane.xml)

Обратите внимание: у первого виджета `FrameLayout` задан идентификатор макета `fragmentContainer`, поэтому код `SingleFragmentActivity.onCreate(…)` может работать так же, как прежде. При создании активности фрагмент, возвращаемый `createFragment()`, появится на левой панели.

Протестируйте макет в `CrimeListActivity`; для этого переопределите метод `getLayoutResId()` так, чтобы он возвращал `R.layout.activity_twopane`.

Листинг 17.2. Переход к файлу двухпанельного макета (CrimeListActivity.java)

```
public class CrimeListActivity extends SingleFragmentActivity {

    @Override
    protected Fragment createFragment() {
        return new CrimeListFragment();
    }

    @Override
    protected int getLayoutResId() {
        return R.layout.activity_twopane;
    }
}
```

Запустите приложение CriminalIntent на планшетном устройстве и убедитесь в том, что на экране отображаются две панели (рис. 17.5). Большая панель детализации пуста, а нажатие на элементе списка не отображает подробную информацию о преступлении. Контейнер детализированного представления будет подключен позднее в этой главе.

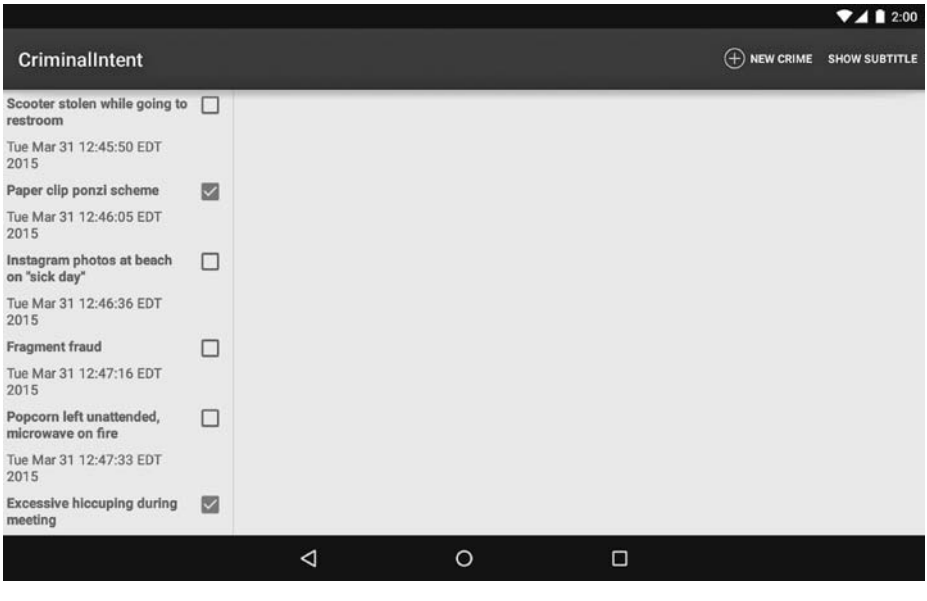

Рис. 17.5. Двухпанельный макет на планшете

В своей текущей версии `CrimeListActivity` также заполняет двухпанельный интерфейс при запуске на телефоне. В следующем разделе мы исправим этот недостаток при помощи ресурса-псевдонима.

Использование ресурса-псевдонима

Ресурс-псевдоним (alias resource) представляет собой ресурс, указывающий на другой ресурс. Ресурсы-псевдонимы находятся в каталоге res/values/ и по умолчанию определяются в файле refs.xml.

В этом разделе мы создадим ресурс-псевдоним, который на телефонах ссылается на макет activity_fragment.xml и на планшетах — на макет activity_twopane.xml.

На следующем шаге мы должны добиться того, чтобы в `CrimeListActivity` отображались разные файлы макетов в зависимости от того, на каком устройстве работает приложение. Это делается точно так же, как мы отображаем разные макеты для книжной и альбомной ориентации: при помощи квалификатора ресурса.

Решение на уровне файлов в res/layout работает, но у него есть свои недостатки. Каждый файл макета должен содержать полную копию отображаемого макета, а это может привести к заметной избыточности. Чтобы создать файл макета activity_masterdetail.xml, пришлось бы скопировать все содержимое activity_fragment.xml в res/layout/activity_masterdetail.xml, а все содержимое activity_twopane.xml — в res/layout-sw600dp/activity_masterdetail.xml. (Вскоре вы увидите, что означает `sw600dp`.)

Вместо этого мы воспользуемся ресурсом-псевдонимом. В этом разделе мы создадим ресурс-псевдоним, который ссылается на activity_fragment.xml на телефонах и макет activity_twopane.xml на планшетах.

На панели Package Explorer щелкните правой кнопкой мыши на каталоге res/layout/ и создайте новый ресурсный файл значений. Присвойте файлу имя refs.xml, а каталогу — имя `values`. Квалификаторов в именах быть не должно. Щелкните на кнопке Finish. Затем добавьте элемент, приведенный в листинге 17.3.

Листинг 17.3. Создание значения по умолчанию для ресурса-псевдонима (res/values/refs.xml)

```
<resources>
  <item name="activity_masterdetail" type="layout">@layout/
                                        activity_fragment</item>
</resources>
```

Значение ресурса представляет собой ссылку на однопанельный макет. Ресурс также обладает идентификатором: `R.layout.activity_masterdetail`. Обратите внимание: внутренний класс идентификатора определяется атрибутом `type` псевдонима. И хотя сам псевдоним находится в res/values/, его идентификатор хранится в `R.layout`.

Теперь этот идентификатор ресурса может использоваться вместо `R.layout.activity_fragment`. Внесите следующее изменение в `CrimeListActivity`.

Листинг 17.4. Повторная замена макета (CrimeListActivity.java)

```
@Override
protected int getLayoutResId() {
    return R.layout.activity_twopane;
    return R.layout.activity_masterdetail;
}
```