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*».

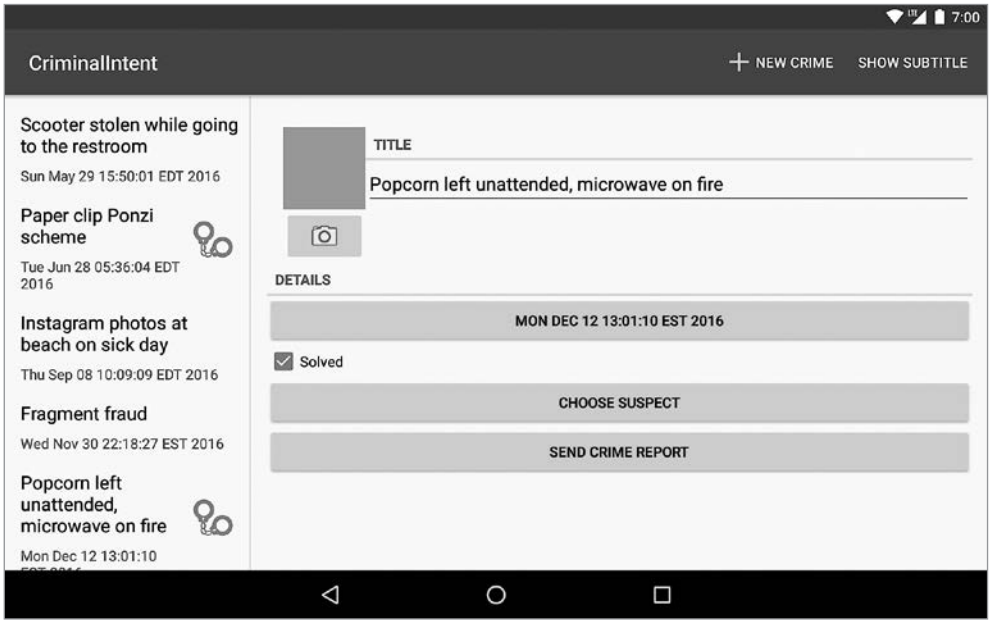

Рис. 17.1. Главное и детализированное представления одновременно находятся на экране

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 целевой API не менее уровня 21 (рис. 17.2).

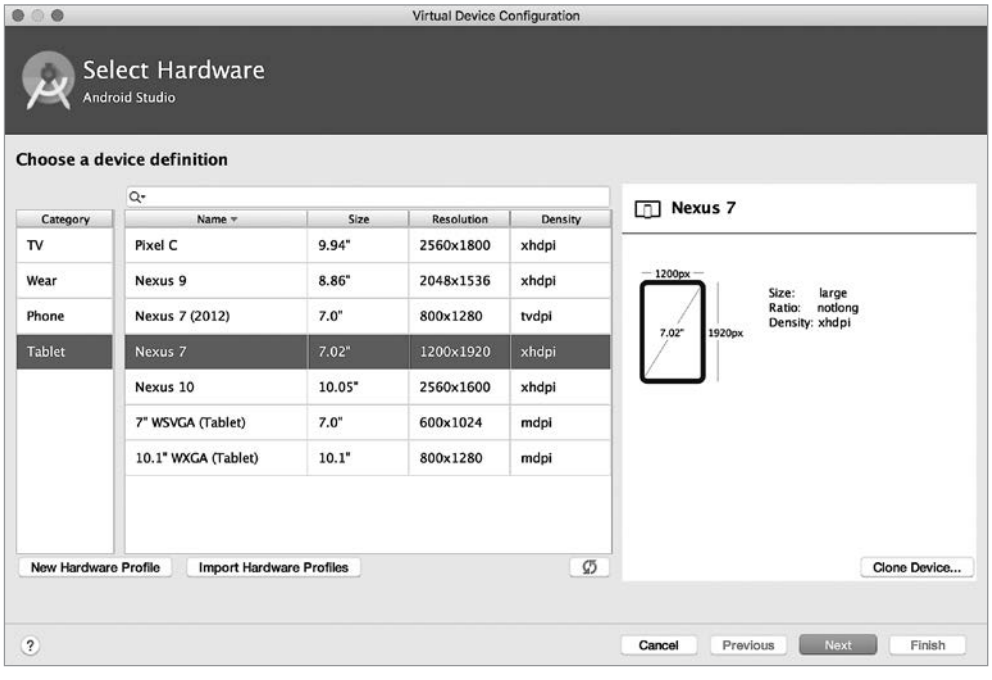

Рис. 17.2. Выбор устройства для планшетного AVD

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.

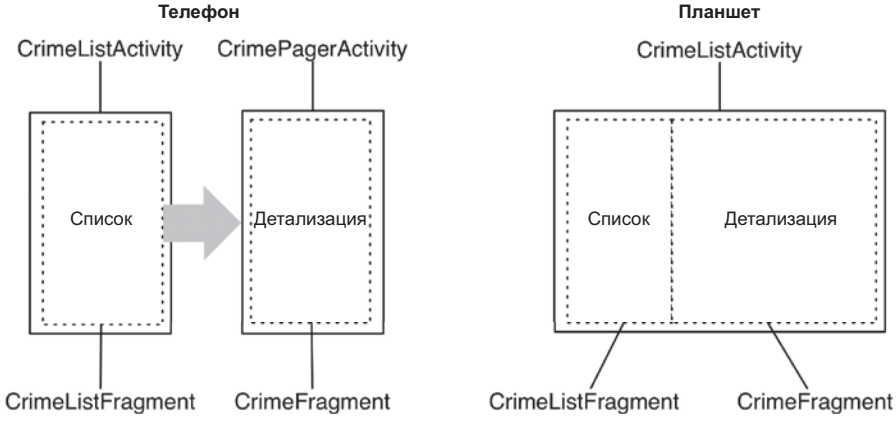

Рис. 17.3. Разновидности макетов

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();
        ...
    }
}
```

Реализация класса `SingleFragmentActivity` по умолчанию будет работать так же, как и прежде, но теперь его субклассы могут переопределить `getLayoutResId()` для воз-

вращения 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.

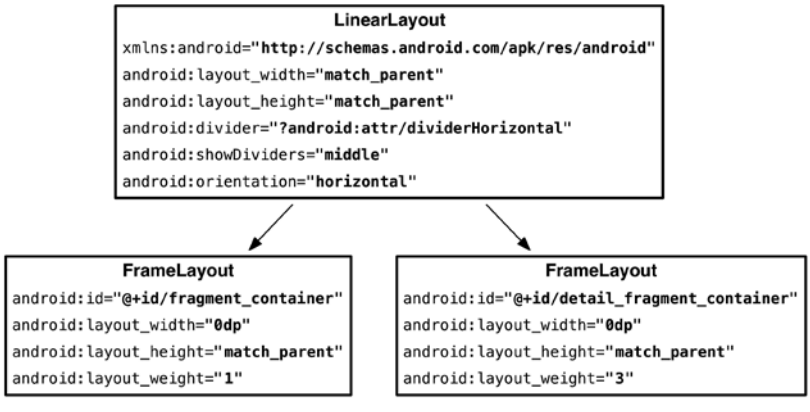

Рис. 17.4. Макет с двумя контейнерами фрагментов (layout/activity_twopane.xml)

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). (Чтобы увидеть панели, необходимо добавить преступление.) Большая панель детализации пуста, а нажатие на элементе списка не отображает подробную информацию о преступлении. Контейнер детализированного представления будет подключен далее в этой главе.

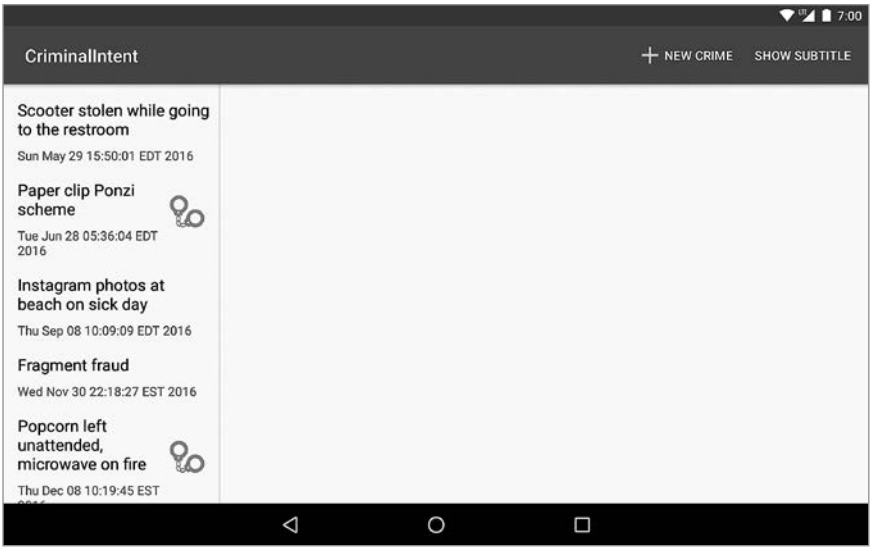

Рис. 17.5. Двухпанельный макет на планшете

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.

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.

В окне инструментов Project 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~~;masterdetail;
}
```

Запустите приложение CriminalIntent и убедитесь в том, что псевдоним работает правильно. Активность `CrimeListActivity` снова должна заполнять однопанельный макет.

Создание альтернативы для планшета

Так как псевдоним находится в res/values/, он используется по умолчанию. Следовательно, по умолчанию `CrimeListActivity` заполняет однопанельный макет.

Теперь мы создадим альтернативный ресурс, чтобы псевдоним `activity_masterdetail` на планшетных устройствах ссылался на activity_twopane.xml.

В окне инструментов Project щелкните правой кнопкой мыши на каталоге res/values и создайте новый файл ресурсов значений. Присвойте ему имя refs.xml и снова присвойте каталогу имя values. Но на этот раз выберите в категории Available qualifiers вариант Smallest Screen Width и щелкните на кнопке >>, чтобы переместить квалификатор вправо (рис. 17.6).

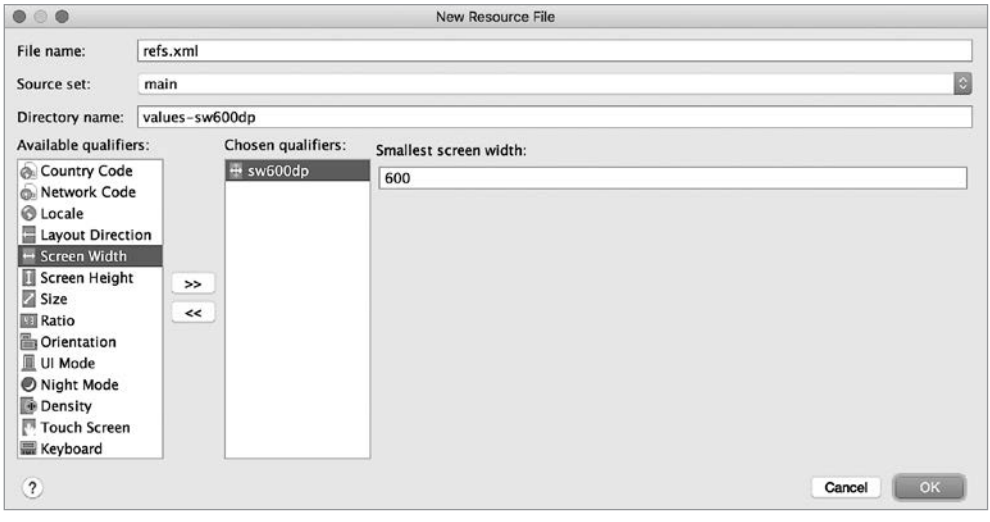

Рис. 17.6. Добавление квалификатора

Этот квалификатор работает несколько иначе: вам предлагается задать значение минимальной длины экрана. Введите 600 и щелкните на кнопке OK. После открытия нового файла ресурсов добавьте псевдоним `activity_masterdetail` и в этот файл, но свяжите его с другим файлом макета.

Листинг 17.5. Альтернативный псевдоним для устройств с экраном большего размера (res/values-sw600dp/refs.xml)

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

Разберемся, что же здесь происходит. Наша цель — сформировать логику, которая работает следующим образом:

- для устройств с экраном меньше заданного размера использовать activity_fragment.xml;
- для устройств с экраном больше заданного размера использовать activity_twopane.xml.

Android не предоставляет возможности использовать ресурс только в том случае, если экран устройства меньше определенного размера, но предлагает неплохую альтернативу. Конфигурационный квалификатор `-sw600dp` позволяет предоставить ресурсы только в том случае, если экран устройства больше определенного размера. Сокращение «sw» происходит от «Smallest Width» (наименьшая ширина), но относится к меньшему размеру экрана, а следовательно, не зависит от текущей ориентации устройства.

Используя квалификатор `-sw600dp`, вы указываете: «Этот ресурс должен использоваться на любых устройствах, у которых меньший размер составляет 600dp и выше». Это хороший критерий для определения экранов планшетных устройств.

А что делать с другой частью, где нужно использовать activity_fragment.xml на меньших устройствах? Меньшие устройства не соответствуют квалификатору `-sw600dp`, поэтому для них будет использован вариант по умолчанию: activity_fragment.xml.

Запустите приложение CriminalIntent на телефоне и на планшете. Убедитесь в том, что одно- и двухпанельные макеты отображаются там, где предполагалось.

Активность: управление фрагментами

Итак, макеты ведут себя так, как положено, и мы можем перейти к добавлению `CrimeFragment` в контейнер фрагмента детализации при использовании двухпанельного макета `CrimeListActivity`.

На первый взгляд кажется, что для этого достаточно написать альтернативную реализацию `CrimeHolder.onClick(View)` для планшетов. Вместо запуска нового экземпляра `CrimePagerActivity` метод `onClick(View)` получает экземпляр `FragmentManager`, принадлежащий `CrimeListActivity`, и закрепляет транзакцию, которая добавляет `CrimeFragment` в контейнер фрагмента детализации.

Код из `CrimeListFragment.CrimeHolder` выглядит примерно так:

```
public void onClick(View view) {
    // Включение нового экземпляра CrimeFragment в макете активности
    Fragment fragment = CrimeFragment.newInstance(mCrime.getId());
    FragmentManager fm = getActivity().getSupportFragmentManager();
    fm.beginTransaction()
        .add(R.id.detail_fragment_container, fragment)
        .commit();
}
```

Такое решение работает, но оно противоречит хорошему стилю программирования Android. Предполагается, что фрагменты представляют собой автономные компонуемые блоки. Если написанный вами фрагмент добавляет фрагменты в `FragmentManager` активности, значит, он делает допущения относительно того, как работает активность-хост, и перестает быть автономным компонуемым блоком.

Например, в приведенном выше коде `CrimeListFragment` добавляет `CrimeFragment` в `CrimeListActivity` и предполагает, что в макете `CrimeListActivity` присутствует контейнер `detail_fragment_container`. Такими вопросами должна заниматься активность-хост `CrimeListFragment`, а не `CrimeListFragment`.

Для сохранения независимости фрагментов мы делегируем выполнение работы активности-хосту, определяя интерфейсы обратного вызова в ваших фрагментах. Активности-хосты реализуют эти интерфейсы для выполнения операций по управлению фрагментами и обеспечения макетно-зависимого поведения.