Навигационные UI-компоненты

В двух предыдущих работах вы научились использовать Android-компонент `Navigation` для перехода между фрагментами путем щелчка на кнопке. Такой подход хорошо работал в приложении `Secret Message`, потому что переходы между целями были линейными.

Однако не все приложения используют такую структуру. Во многих приложениях существуют экраны, к которым пользователь должен переходить независимо от того, в какой точке приложения он находится. Например, в почтовом клиенте могут быть папки `Входящие`, `Отправленные` и `Справка`, которые должны открываться с минимальными усилиями.

Итак, в вашем приложении используется нелинейная схема навигации. Как же сделать так, чтобы все экраны были постоянно доступными?

В Android существуют навигационные UI-компоненты

Если у вас есть экраны, которые должны быть доступны из любой точки приложения, возможно, стоит воспользоваться одним из навигационных UI-компонентов Android.

Панель приложения Это панель, которая отображается у верхнего края экрана. Android обычно включает такую панель по умолчанию. На нее можно добавлять элементы, по щелчку на которых происходит переход к целям.

Нижняя панель навигации Отображается у нижнего края экрана. На ней размещаются элементы, которые могут использоваться для навигации.

Выдвижная панель навигации Панель, которая выдвигается от стороны экрана. Такие панели используются во многих приложениях, так как они обладают очень гибкими возможностями.

В этой работе вы узнаете, как реализовать все три типа навигационных UI-компонентов. Для этого мы построим прототип почтового клиента `CatChat`.

Как работает приложение CatChat

В приложении `CatChat` будет использоваться одна активность с именем `MainActivity` и три фрагмента: `InboxFragment`, `SentItemsFragment`, `HelpFragment`. Каждый фрагмент будет отображаться в `MainActivity` при переходе к нему.

Приложение будет включать панель приложения, на которой размещается элемент для открытия меню справки. Когда пользователь щелкает на нем, приложение переходит к фрагменту `HelpFragment` и отображает его в `MainActivity`.

В приложении также будет использоваться выдвижная панель для перехода между всеми тремя фрагментами. На выдвижной панели размещается элемент для каждого фрагмента, и при щелчке на каждом элементе будет отображаться соответствующий фрагмент

Рассмотрим последовательность действий для создания приложения `CatChat`

1 Замена стандартной панели приложения панелью инструментов. Панель инструментов внешне похожа на стандартную панель приложения, но она обладает большей гибкостью и включает новейшую функциональность панелей приложений. Заодно вы научитесь применять темы и стили.

2 Добавление элемента справки на панель инструментов. По щелчку на элементе меню `Help` будет происходить переход к `HelpFragment`.

3 Реализация нижней панели навигации. Мы добавим в приложение нижнюю панель, с которой можно перейти к каждому из фрагментов приложения.

4 Создание выдвижной панели. Наконец, нижняя панель навигации будет заменена выдвижной панелью. На выдвижной панели будет отображаться графический заголовок и элементы, представляющие все фрагменты приложения, разделенные на группы.

Создание нового проекта

Для приложения `CatChat` понадобится новый проект; создайте его в Android Studio по схеме, уже знакомой вам по предыдущим главам. Выберите вариант `Empty Activity`, введите имя «CatChat» и имя пакета «com.hfad.catchat», подтвердите папку для сохранения по умолчанию. Убедитесь в том, что выбран язык Kotlin с минимальным уровнем SDK API 29, чтобы приложение работало на большинстве устройств Android.

Создается стандартная панель приложения

Когда вы создаете новый проект с пустой активностью, Android Studio обычно добавляет панель приложения, на которой выводится имя приложения. Эта панель отображается в верхней части экрана при запуске приложения.

Панель приложения полезна по нескольким причинам:

- Она делает ключевые действия (например, публикацию контента и выполнение поиска) более заметными и при этом более предсказуемыми.
- Она помогает пользователю понять, в какой точке приложения он находится; для этого на панели выводится имя приложения или описание текущего экрана.
- Она может использоваться для перехода к разным целям.

Стандартная панель приложения добавляется в приложение путем применения темы. Тема обеспечивает целостность оформления приложения на разных экранах. Она управляет внешним видом приложения и наличием у него панели приложения. Android включает ряд тем, которые могут использоваться в ваших приложениях. По умолчанию Android Studio применяет тему, включающую панель приложения.

Использование темы Material Design

В приложении `CatChat` будет использоваться тема `Material Design`. Система `Material Design` была разработана компанией Google. Она помогает строить качественные приложения и веб-сайты с целостным оформлением. Идея заключалась в том, чтобы пользователь, переключающийся с приложения Google (такого, как Play Store) на приложение, созданное сторонним разработчиком, мгновенно чувствовал себя в знакомой обстановке и знал, что делать.

В основу Material Design изначально были заложены принципы дизайна печатных материалов, отражающие внешний вид и поведение реальных объектов (например, карточек для записей и листков бумаги.) Новейшей стадией ее эволюции стала система Material You, предоставляющая пользователю более динамичный опыт взаимодействия с персонализированной цветовой палитрой.

Файл приложения build.gradle должен содержать зависимость от библиотеки Material

Темы Material хранятся в отдельной библиотеке, которая должна быть включена в приложение. Для этого следует добавить зависимость для библиотеки com.google.android.material в файл build.gradle приложения. Так как мы собираемся использовать тему Material в приложении CatChat, необходимо проверить, что эта библиотека включена в приложение. Откройте файл *CatChat/app/build.gradle* и убедитесь в том, что он включает следующую строку:

```
dependencies {
 ...
 implementation 'com.google.android.material:material:1.4.0'
 ...
}
```

Возможно, среда Android Studio уже включила эту зависимость за вас. Если это не было сделано, вам придется добавить ее самостоятельно и щелкнуть на ссылке Sync Now, появляющейся в верхней части редактора кода, чтобы синхронизировать изменения с оставшейся частью проекта.

Мы хотим применить тему Material в приложении CatChat для управления его панелью приложения. Давайте посмотрим, как это делается.

Применение темы в AndroidManifest.xml

Тема применяется в файле AndroidManifest.xml приложения. Этот файл предоставляет информацию о конфигурации приложения. В нем содержатся атрибуты — включая тему, — непосредственно влияющие на панель приложения. Ниже приведен код AndroidManifest.xml, созданный за нас средой Android Studio в проекте CatChat:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.hfad.catchat">
<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/Theme.CatChat">
    <activity
        android:name=".MainActivity"
```

```
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

Атрибут `android:label` ссылается на текст, выводимый в панели приложения, — то самое имя, которое вы указываете при исходном создании приложения. В приведенном выше коде используется строковый ресурс с именем `app_name`, который был добавлен средой Android Studio в `strings.xml`.

Атрибут `android:theme` определяет тему. В приведенном выше коде он задается директивой:

```
android:theme="@style/Theme.CatChat"
```

Это значение сообщает, что тема определяется как стилевой ресурс (обозначаемый `@style`) с именем `Theme.CatChat`.

Определение стилей в файлах стилевых ресурсов

Стилевые ресурсы используются для описания любых тем и стилей, которые должны использоваться приложением, и хранятся в одном или в нескольких файлах стилевых ресурсов.

Когда мы создаем приложение `CatChat`, среда Android Studio создает два файла стилевых ресурсов за нас. Обоим файлам присваивается имя `themes.xml`, и они размещаются в папках *app/src/main/res/values* и *app/src/main/res/values-night*. Файл в папке *values* содержит файл стилевых ресурсов приложения, используемый по умолчанию, а файл в папке *values-night* используется ночью. `themes.xml` в папке *values* определяет код стиля следующего вида:

```
<resources xmlns:tools="http://schemas.android.com/tools">
<!-- Base application theme. -->
<style name="Theme.CatChat"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/purple_500</item>
    <item name="colorPrimaryVariant">@color/purple_700</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
    ...
</style>
</resources>
```

Элемент `<style>` сообщает Android, что определяется стилевой ресурс. Каждому стилю должно быть присвоено имя, по которому он идентифицируется. Имя определяется атрибутом name:

```
name="Theme.CatChat"
```

Атрибут `theme` из `AndroidManifest.xml` использует это имя для назначения темы приложения, например "@style/Theme.CatChat".

Стиль также включает атрибут `parent`, который указывает, на основе какой темы должен определяться стиль. В приведенном выше коде используется значение:

```
parent="Theme.MaterialComponents.DayNight.DarkActionBar"
```

соответствующее теме с темной панелью приложения, которая позволяет приложению переходить между дневной и ночной темой. В дневное время он используется стиль, определенный в файле стилевых ресурсов из папки *values*, а по ночам устройство переключается на стиль из папки *values-night*. Приведенный выше стиль также включает элементы `<item>`, которые переопределяют некоторые цвета темы. Рассмотрим их более подробно.

Стили могут переопределять цвета темы

Если вы захотите переопределить любые свойства родительской темы (например, цветовую схему), для этого следует добавить в стиль элементы `<item>`. Например, в приложении `CatChat` включены элементы для переопределения основного и дополнительного цвета. Основной цвет используется приложением для прорисовки таких объектов, как панель приложения. Дополнительный цвет используется некоторыми представлениями для создания контраста.

Можно использовать несколько файлов стилевых ресурсов для применения разных цветовых схем в разных ситуациях. Так, приложение `CatChat` включает один файл стилевых ресурсов в папке *values* и другой файл в папке *values-night*. Такая схема — в сочетании с темой `DayNight`, на основе которой определяется каждый стиль, — позволяет применять разные цветовые схемы в дневное и ночное время. Ниже приведен код, используемый приложением `CatChat` для переопределения основных цветов в файле `themes.xml` из папки *values*:

```
<item name="colorPrimary">@color/purple_500</item>
<item name="colorPrimaryVariant">@color/purple_700</item>
<item name="colorOnPrimary">@color/white</item>
```

Как видите, у каждого элемента имеется атрибут `name`, а обозначение `@color` ссылается на цвет из файла цветовых ресурсов.

Файлы цветовых ресурсов определяют набор цветов

Когда вы создаете новый проект, среда Android Studio обычно включает стандартный файл цветовых ресурсов с именем `colors.xml`. Он находится в папке *app/src/main/res/values*, а содержащиеся в нем цвета могут использоваться в вашем приложении. Типичный файл цветовых ресурсов выглядит примерно так:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <color name="purple_200">#FFBB86FC</color>
 <color name="purple_500">#FF6200EE</color>
 <color name="purple_700">#FF3700B3</color>
 ...
</resources>
```

Чтобы изменить цветовую схему приложения, включите нужные цвета в файл цветовых ресурсов и обращайтесь к ним из файлов стилевых ресурсов.

Замена панели приложения по умолчанию панелью инструментов

Ранее вы видели, как Android Studio добавляет в приложение панель приложения по умолчанию, назначая ему тему. Добавить панель приложения таким способом несложно, но в более гибком варианте она заменяется панелью инструментов. Простейшая панель инструментов не отличается от панели приложения по умолчанию, которую вы уже видели, но она выделяется намного большей гибкостью. Например, вы можете изменить ее высоту, а в следующей главе вы научитесь сворачивать и разворачивать ее при прокрутке экрана устройства. Также в нее включена новейшая функциональность панелей приложений, так что создание панели инструментов упрощает реализацию этой функциональности в ваших приложениях.

Чтобы заменить панель приложения по умолчанию, следует удалить исходную панель приложения, включить панель инструментов в ваш макет, а затем приказать активности использовать панель инструментов как его панель приложения. На ближайших страницах мы покажем, как это делается.

Удаление панели приложения по умолчанию с использованием темы

Чтобы удалить панель приложения по умолчанию, примените к приложению тему, не включающую панель приложения. Например, текущая версия приложения `CatChat` использует тему `Theme.MaterialComponents.DayNight.DarkActionBar`. Мы удалим эту панель приложения и заменим ее на `Theme.MaterialComponents.DayNight.NoActionBar`. Эти две темы почти не отличаются, не считая того, что во второй нет панели приложения. Давайте изменим тему: обновите код в файлах `themes.xml` из папок *values* и *values-night* и включите в них следующие изменения:

```
<resources>
 <style name="Theme.CatChat"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
 ...
 </style>
</resources>
```

Вот и все, что необходимо знать для удаления панели приложения по умолчанию. На следующем шаге вы научитесь добавлять панели инструментов.

Панель инструментов является разновидностью View

В отличие от панели приложения по умолчанию, панель инструментов является разновидностью представления, которая добавляется в макет. А раз это представление, это означает, что вы можете в полной мере управлять его размером и позицией.

Доступны разные типы представлений, и в приложении `CatChat` мы будем использовать панель инструментов `Material`. Эта разновидность панели инструментов хорошо работает в сочетании с темами `Material` — такими, как использованная в нашем приложении. Код добавления панели инструментов `Material` выглядит примерно так:

```
<com.google.android.material.appbar.MaterialToolbar
 android:id="@+id/toolbar"
 android:layout_width="match_parent"
 android:layout_height="?attr/actionBarSize"
 style="@style/Widget.MaterialComponents.Toolbar.Primary" />
```

Сначала вы определяете панель инструментов:

```
<com.google.android.material.appbar.MaterialToolbar
 ... />
```

где *com.google.android.material.appbar.MaterialToolbar* — полный путь к классу `MaterialToolbar`. Затем другие атрибуты представлений используются для назначения идентификатора и определения внешнего вида панели. Например, чтобы ширина панели инструментов соответствовала ширине родителя, а высота — высоте панели приложения по умолчанию из соответствующей темы, можно использовать следующий код:

```
android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize"
```

Префикс `?attr` в этом коде означает, что вы хотите использовать атрибут из текущей темы. В нашем конкретном случае запись `?attr/actionBarSize` обозначает высоту панели приложения по умолчанию для темы. Также к панели инструментов можно применить стилевое оформление, чтобы она использовала основные цвета приложения. Это делается так:

```
style="@style/Widget.MaterialComponents.Toolbar.Primary"
```

Итак, теперь вы знаете, как выглядит код панели инструментов. Добавим панель инструментов в макет `MainActivity`.

Добавление панели инструментов в activity_main.xml

Мы добавим панель инструментов в макет `MainActivity`, чтобы она отображалась у верхнего края экрана. Обновите код в `activity_main.xml` и приведите его к следующему виду

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">
 <com.google.android.material.appbar.MaterialToolbar
 android:id="@+id/toolbar"
 android:layout_width="match_parent"
 android:layout_height="?attr/actionBarSize"
 style="@style/Widget.MaterialComponents.Toolbar.Primary" />
</LinearLayout>
```

Где имя приложения?

Приведенный выше код добавляет панель инструментов в макет `MainActivity`, но панель еще не обладает никакой функциональностью панелей приложений. Например, если запустить приложение на этой стадии, вы увидите, что имя приложения не выводится на панели инструментов, как на стандартной панели приложения в предыдущей версии.

Чтобы панель инструментов вела себя как нормальная панель приложения, необходимо внести изменения в код активности, написанный на Kotlin.

Замена панели инструментов панелью приложения MainActivity

Чтобы панель инструментов вела себя как привычная панель приложения, необходимо сообщить об этом `MainActivity`. Для этого следует вызвать в коде активности метод `setSupportActionBar()` и передать ему ссылку на панель инструментов:

```
val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
setSupportActionBar(toolbar)
```

Этот код будет добавлен в метод `onCreate()` активности `MainActivity`, чтобы он выполнялся сразу же после создания активности. Обновите код `MainActivity.kt`:

```
package com.hfad.catchat
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.google.android.material.appbar.MaterialToolbar
```

```
class MainActivity : AppCompatActivity() {
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
 setSupportActionBar(toolbar)
 }
}
```

Вот и все, что необходимо сделать для замены панели приложения панелью инструментов. Давайте проведем тест-драйв приложения и убедимся в том, что имя выводится правильно.

Использование панели инструментов для навигации

В начале этой главы мы сказали, что наше приложение должно обеспечивать переходы к разным экранам приложения `CatChat` с использованием навигационных UI компонентов Android. После добавления панели инструментов в приложение создадим пару экранов и воспользуемся панелью инструментов для перемещения между ними.

Как работает навигация с панели инструментов

Начнем с создания двух новых фрагментов — `InboxFragment` и `HelpFragment`, которые будут отображаться в `MainActivity` при переходе к ним пользователя. Фрагмент `InboxFragment` будет отображаться при запуске приложения, а `HelpFragment` — при переходе к нему. Для перехода к `HelpFragment` мы добавим на панель инструментов элемент `Help`, чтобы панель выглядела так:

Для добавления элемента `Help` мы воспользуемся меню — как это делается, вы узнаете через несколько страниц. При щелчке на элементе `Help` приложение будет переходить к `HelpFragment`. Мы также обновим заголовок панели инструментов, чтобы в ней выводилось название текущего экрана, и добавим кнопку `Up` для упрощения возврата к `InboxFragment`.

Так работает навигация с панели инструментов. Начнем с ее реализации, для чего в приложение будут добавлены фрагменты `InboxFragment` и `HelpFragment`.

Создание InboxFragment

При запуске приложения в `MainActivity` должен отображаться фрагмент `InboxFragment`. Выделите пакет *com.hfad.catchat* в папке *app/src/main/java* и выберите команду `File→New→Fragment→Fragment (Blank)`. Введите имя фрагмента «InboxFragment» и имя макета «fragment_inbox»; убедитесь в том, что выбран язык Kotlin. Затем обновите код `InboxFragment.kt` и приведите его к следующему виду:

```
package com.hfad.catchat
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
class InboxFragment : Fragment() {
 override fun onCreateView(
```

```
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
 ): View? {
 return inflater.inflate(R.layout.fragment_inbox, container, false)
 }
}
```

А это код фрагмента `fragment_inbox.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 tools:context=".InboxFragment">
 <TextView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:text="Inbox" />
</FrameLayout>
```

Создание HelpFragment

Фрагмент `HelpFragment` отображается в `MainActivity`, когда пользователь щелкает на элементе `Help` панели инструментов. Выделите пакет *com.hfad.catchat* в папке *app/src/main/java* и выберите команду *File→New→Fragment→Fragment (Blank)*. Введите имя фрагмента «HelpFragment» с именем макета «fragment_help» и убедитесь в том, что выбран язык Kotlin. Код `HelpFragment.kt` должен выглядеть так:

```
package com.hfad.catchat
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
class HelpFragment : Fragment() {
 override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
 ): View? {
 return inflater.inflate(R.layout.fragment_help, container, false)
 }
}
```

Файл `fragment_help.xml` должен содержать следующую разметку:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
```

```
 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"
 tools:context=".HelpFragment">
 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Help" />
</FrameLayout>
```

Использование компонента Navigation для перехода к HelpFragment

Вы научились перемещаться между фрагментами при помощи Android-компонента `Navigation`. И хотя для перехода к новой цели по-прежнему будет использоваться панель инструментов, для всех задач навигации все равно можно пользоваться компонентом `Navigation`. Сначала необходимо добавить компонент `Navigation` в проект `CatChat` средствами `Gradle`.

Добавление номера версии в файл проекта build.gradle

Как и прежде, необходимо добавить в файл `build.gradle` проекта новую переменную с версией компонента `Navigation`, который должен быть добавлен в приложение. Откройте файл *CatChat/build.gradle* и добавьте следующую строку в раздел `buildscript`:

```
buildscript {
 ext.nav_version = '2.3.5'
}
```

Добавление зависимостей в файл проекта build.gradle

Затем необходимо добавить две зависимости компонента `Navigation` в файл `build.gradle` приложения. Откройте файл *CatChat/app/build.gradle* и добавьте две строки в раздел `dependencies`:

```
dependencies {
 implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
 implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
}
```

После внесения этих изменений щелкните на ссылке `Sync Now`, появляющейся в верхней части редактора кода, чтобы синхронизировать изменения с оставшейся частью проекта. После того как компонент `Navigation` будет добавлен в проект, мы воспользуемся им для создания графа навигации — после следующего упражнения.

Добавление фрагментов в граф навигации

Как вы уже знаете, граф навигации вашего приложения содержит подробную информацию о целях вашего приложения и возможных путях, по которым к ним можно перейти. В приложении `CatChat` фрагменты `InboxFragment` и `HelpFragment` являются возможными целями, поэтому мы создадим новый граф навигации и добавим в него два фрагмента. Чтобы создать граф навигации, выделите папку *CatChat/app/src/main/res* на панели проекта, а затем выберите команду *File→New→Android Resource File*. Введите имя файла «nav_graph», выберите тип ресурса «Navigation» и щелкните на кнопке OK. Затем откройте граф навигации (файл `nav_graph.xml`), переключитесь в режим `Code` и обновите файл, чтобы он соответствовал приведенному ниже коду:

```
<?xml version="1.0" encoding="utf-8"?>
<navigation
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/nav_graph"
 app:startDestination="@id/inboxFragment">
 <fragment
 android:id="@+id/inboxFragment"
 android:name="com.hfad.catchat.InboxFragment"
 android:label="Inbox"
 tools:layout="@layout/fragment_inbox" />
 <fragment
 android:id="@+id/helpFragment"
 android:name="com.hfad.catchat.HelpFragment"
 android:label="Help"
 tools:layout="@layout/fragment_help" />
</navigation>
```

Приведенный выше код добавляет фрагменты `InboxFragment` и `HelpFragment` в граф навигации и назначает каждому из них метку, удобную для пользователя. И это все изменения, которые необходимо внести в граф навигации. Перейдем к обновлению макета `MainActivity`, чтобы он мог отображать каждый фрагмент, к которому вы переходите.

Добавление хоста навигации в activity_main.xml

Чтобы в приложении отображались цели, к которым вы переходите, следует включить в макет активности хост навигации. Для этого используется элемент `FragmentContainerView`, определяющий тип применяемого хоста навигации и имя графа навигации. Код для решения этой задачи в приложении `CatChat` почти полностью совпадает с кодом, добавленным в приложение `Secret Message`. Обновите код `activity_main.xml` и включите в него изменения, выделенные жирным шрифтом:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
```

```
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">
 <com.google.android.material.appbar.MaterialToolbar
 android:id="@+id/toolbar"
 android:layout_width="match_parent"
 android:layout_height="?attr/actionBarSize"
 style="@style/Widget.MaterialComponents.Toolbar.Primary" />
 <androidx.fragment.app.FragmentContainerView
 android:id="@+id/nav_host_fragment"
 android:name="androidx.navigation.fragment.NavHostFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:defaultNavHost="true"
 app:navGraph="@navigation/nav_graph" />
</LinearLayout>
```

После добавления хоста навигации мы пополним панель инструментов элементом для перехода к `HelpFragment`.

Определение элементов панели инструментов в файле ресурсов меню

Чтобы указать Android, какие элементы должны размещаться на панели инструментов, следует определить меню. Каждое меню определяется в XML-файле ресурсов меню. Мы создадим новый файл ресурсов меню с именем `menu_toolbar.xml`, который будет использоваться для добавления элемента `Help` на панель инструментов. Выделите папку *app/src/main/res*, откройте меню `File`, выберите команду `New`, а затем — вариант создания нового файла ресурсов Android. Введите имя «menu_toolbar», укажите тип ресурса «Menu» и убедитесь в том, что выбрано имя каталога menu. При нажатии кнопки OK среда Android Studio создает файл и добавляет его в папку *app/src/main/res/menu*. Как и в случае с графами навигации и файлами макетов, вы можете редактировать файлы ресурсов меню путем изменения кода XML или с помощью встроенного визуального редактора.

Добавление элемента Help в меню

Мы хотим иметь возможность перейти к фрагменту `HelpFragment` с панели инструментов, поэтому в файл ресурсов меню будет добавлен элемент `Help`. Это будет сделано прямым редактированием кода XML. Переключитесь в режим Code для файла `menu_toolbar.xml` и обновите код:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto">
 <item
 android:id="@+id/helpFragment"
 android:icon="@android:drawable/ic_menu_help"
 android:title="Help"
 app:showAsAction="always" />
</menu>
```

Каждый файл ресурсов меню, включая приведенный выше, содержит корневой элемент `<menu>`. Он сообщает Android, что в файле определяется меню. Внутри элемента `<menu>` обычно находится группа элементов `<item>`, каждый из которых описывает отдельный элемент меню. В нашем конкретном примере определяется один элемент `Help` с заголовком «Help». Элемент `<item>` содержит ряд атрибутов, управляющих внешним видом элемента меню. Атрибут `android:id` назначает идентификатор элемента. Идентификатор используется компонентом `Navigation` для перехода к цели; он должен совпадать с идентификатором цели, к которой происходит переход на графе навигации. Через несколько страниц вы увидите, как работает эта схема. Атрибут `android:icon` указывает, какой значок должен отображаться для элемента (и должен ли). В Android есть много встроенных значков, и IDE выводит список доступных вариантов, когда вы начинаете вводить имя значка. Атрибут `android:title` определяет текст элемента. Атрибут `app:showAsAction` указывает, когда элемент должен появляться на панели инструментов. Значение «always» означает, что он всегда должен отображаться в основной области панели инструментов.

onCreateOptionsMenu() добавляет элементы меню на панель инструментов

После определения файла ресурсов меню необходимо добавить элементы на панель инструментов. Для этого в коде активности реализуется метод `onCreateOptionsMenu()`. Этот метод вызывается в тот момент, когда активность готова к добавлению элементов на панель инструментов. Он заполняет файл ресурсов меню и добавляет каждый описанный в нем элемент на панель инструментов. В приложении `CatChat` элемент, определенный в `menu_toolbar.xml`, должен быть добавлен на панель инструментов `MainActivity`. Код добавления элемента выделен ниже жирным шрифтом (мы обновим его потом):

```
package com.hfad.catchat
...
import android.view.Menu
class MainActivity : AppCompatActivity() {
 override fun onCreate(savedInstanceState: Bundle?) {
 ...
 }
 override fun onCreateOptionsMenu(menu: Menu): Boolean {
 menuInflater.inflate(R.menu.menu_toolbar, menu)
 return super.onCreateOptionsMenu(menu)
 }
}
```

Строка

```
menuInflater.inflate(R.menu.menu_toolbar, menu)
```

заполняет файл ресурсов меню. Во внутренней реализации она создает объект `Menu`, представляющий файл ресурсов меню, а все элементы, содержащиеся в файле ресурсов меню, преобразуются в объекты `MenuItem`, которые затем добавляются на панель инструментов. Вот и все, что необходимо сделать для

добавления меню на панель инструментов. Теперь нужно сделать так, чтобы активность `MainActivity` переходила к `HelpFragment`, когда пользователь щелкает на элементе `Help`.

Реакция на щелчки на командах в onOptionsItemSelected()

После того как элементы меню будут добавлены на панель инструментов методом `onCreateOptionsMenu()`, необходимо обеспечить их реакцию на щелчки. Для этого следует реализовать метод `onOptionsItemSelected()`. Этот метод выполняется каждый раз, когда на элементе панели инструментов делается щелчок:

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
 //Код, выполняемый при щелчке на элементе
}
```

Приложение `CatChat` должно переходить к `HelpFragment` каждый раз, когда пользователь щелкает на элементе меню Help. Компонент `Navigation` поможет нам с решением этой задачи.

Ниже показано, как выглядит код; эти изменения будут добавлены в `MainActivity.kt` через несколько страниц:

```
package com.hfad.catchat
...
import android.view.MenuItem
import androidx.navigation.findNavController
import androidx.navigation.ui.onNavDestinationSelected
class MainActivity : AppCompatActivity() {
 ...
 override fun onOptionsItemSelected(item: MenuItem): Boolean {
 val navController = findNavController(R.id.nav_host_fragment)
 return item.onNavDestinationSelected(navController)
 || super.onOptionsItemSelected(item)
 }
}
```

Каждый раз, когда пользователь щелкает на элементе меню `Help`, контроллер навигации получает его идентификатор и ищет совпадающий идентификатор в графе навигации. Затем цель с таким идентификатором передается хосту навигации, чтобы он был отображен на экране устройства.

Настройка панели инструментов

Мы рассмотрели все, что необходимо знать для добавления элемента меню `Help` на панель инструментов и настройки перехода от него к `HelpFragment`. Но прежде чем опробовать приложение на практике, нужно внести еще одно изменение. Когда мы определяли граф навигации `nav_graph.xml` ранее в этой главе, для каждой цели была добавлена метка:

```
<?xml version="1.0" encoding="utf-8"?>
<navigation ...>
 <fragment
 android:id="@+id/inboxFragment"
 android:name="com.hfad.catchat.InboxFragment"
 android:label="Inbox"
 tools:layout="@layout/fragment_inbox" />
 <fragment
 android:id="@+id/helpFragment"
 android:name="com.hfad.catchat.HelpFragment"
 android:label="Help"
 tools:layout="@layout/fragment_help" />
</navigation>
```

Для понимания, какой экран отображается в настоящий момент, мы настроим панель инструментов, чтобы при переходе к новой цели ее метка отображалась на панели инструментов. Также на панель инструментов будет добавлена кнопка `Up`, чтобы после перехода к `HelpFragment` пользователь мог легко вернуться к `InboxFragment`.

Может показаться, что кнопка `Up` делает то же, что и кнопка `Back` на устройстве, но это не совсем так. Кнопка `Back` позволяет пользователю пройти весь путь по стеку возврата, то есть историю посещенных им экранов. С другой стороны, кнопка `Up` работает на основании иерархии графа навигации. Она предоставляет средства для быстрого перехода вверх по иерархии. Панель инструментов можно настроить, чтобы текст на ней обновлялся при смене фрагмента, и добавить кнопку `Up` при помощи компонента `Navigation`. Давайте посмотрим, как это делается.

Настройка панели инструментов с использованием AppBarConfiguration

Панель инструментов требуется настроить так, чтобы текст на ней содержал метку текущей цели в графе навигации и чтобы на ней выводилась кнопка `Up`. Для этого можно построить объект `AppBarConfiguration`, основанный на графе навигации, и связать его с панелью инструментов. Класс `AppBarConfiguration` является частью компонента `Navigation`; с его помощью можно организовать взаимодействие панелей приложений и панелей инструментов с контроллером навигации. Ниже приведен код построения `AppBarConfiguration` и связывания объекта с панелью инструментов:

```
override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
 setSupportActionBar(toolbar)
 val navHostFragment = supportFragmentManager
 .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
 val navController = navHostFragment.navController
 val builder = AppBarConfiguration.Builder(navController.graph)
 val appBarConfiguration = builder.build()
 toolbar.setupWithNavController(navController, appBarConfiguration)
}
```

Приведенный выше код сначала получает ссылку на контроллер навигации из хоста навигации, для чего используется следующий код:

```
val navHostFragment = supportFragmentManager
 .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
val navController = navHostFragment.navController
```

Такой код приходится использовать везде, где вам нужно получить ссылку на контроллер навигации из метода `onCreate()` активности. Затем в коде строится объект конфигурации, связывающий панель инструментов с графом навигации, и этот объект применяется к панели инструментов. При выполнении кода информация из графа навигации используется для отображения метки текущей цели. Также на панель инструментов добавляется кнопка `Up` для всех целей, кроме стартовой цели графа навигации, которой в нашем случае является `InboxFragment`. Вот и все, что необходимо знать для реализации навигации с панели инструментов. Полный код `MainActivity` приведен на следующей странице.

Полный код MainActivity.kt

Ниже приведен полный код `MainActivity.kt`; обновите свой код:

```
package com.hfad.catchat
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import androidx.navigation.findNavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.onNavDestinationSelected
import androidx.navigation.ui.setupWithNavController
import com.google.android.material.appbar.MaterialToolbar
class MainActivity : AppCompatActivity() {
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
 setSupportActionBar(toolbar)
 val navHostFragment = supportFragmentManager
 .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
 val navController = navHostFragment.navController
 val builder = AppBarConfiguration.Builder(navController.graph)
 val appBarConfiguration = builder.build()
 toolbar.setupWithNavController(navController, appBarConfiguration)
 }
 override fun onCreateOptionsMenu(menu: Menu): Boolean {
 menuInflater.inflate(R.menu.menu_toolbar, menu)
 return super.onCreateOptionsMenu(menu)
 }
 override fun onOptionsItemSelected(item: MenuItem): Boolean {
```

```
 val navController = findNavController(R.id.nav_host_fragment)
 return item.onNavDestinationSelected(navController)
 || super.onOptionsItemSelected(item)
 }
}
```

Многие разновидности UI-навигации работают с компонентом Navigation

К текущему моменту вы научились включать навигацию с панели инструментов, определяя файл ресурсов меню, использовать компонент `Navigation` для перехода между фрагментами и настраивать панель инструментов для изменения ее внешнего вида при переходах в приложении. К счастью, другие разновидности UI-навигации, такие как нижние панели и выдвижные панели, работают аналогичным образом. И хотя внешне они отличаются друг от друга, вы можете брать приемы, освоенные вами для навигации с панели инструментов, и применять их к другим типам навигации. Чтобы вы поняли, как работают эти виды навигации, мы добавим нижнюю панель навигации в активность `MainActivity` приложения `CatChat`.

Как работает нижняя панель навигации

Как следует из названия, нижняя панель навигации — разновидность панели навигации, которая располагается в нижней части экрана устройства. На ней можно разместить элементы для переходов до пяти целей.

Панель содержит три элемента: `Inbox`, `Sent Items` и `Help`. При щелчке на каждом элементе приложение переходит к фрагменту, связанному с ним. Например, когда вы щелкаете на элементе `Item`, приложение переходит к `HelpFragment`, а когда вы щелкаете на `Sent Items`, оно перейдет к новому фрагменту (который мы еще не создали) с именем `SentItemsFragment`. Нижняя панель будет реализована на нескольких ближайших страницах, а начнем мы с создания нового фрагмента: `SentItemsFragment`.

Создание SentItemsFragment

На следующем шаге мы создадим новый фрагмент с именем `SentItemsFragment`. Выделите пакет *com.hfad.catchat* в папке *app/src/main/java* и выберите команду `File→New→Fragment→Fragment (Blank)`. Введите имя фрагмента «SentItemsFragment» и имя макета «fragment_sent_items»; убедитесь в том, что выбран язык Kotlin. Затем обновите код `SentItemsFragment.kt` и приведите его к следующему виду:

```
package com.hfad.catchat
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
class SentItemsFragment : Fragment() {
 override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
 ): View? {
```

```
    return inflater.inflate(R.layout.fragment_sent_items, container, false)
    }
}
```

А это код фрагмента `fragment_sent_items.xml` (обновите код и в этом файле):

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
 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"
 tools:context=".SentItemsFragment">
 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Sent Items" />
</FrameLayout>
```

Включение фрагмента SentItemsFragment в граф навигации

Мы хотим иметь возможность перейти к `SentItemsFragment` с нижней панели навигации с использованием компонента `Navigation`. Для этого мы добавим фрагмент в граф навигации как новую цель. Откройте файл графа навигации `nav_graph.xml` (если он не был открыт ранее) и обновите файл, чтобы он соответствовал приведенному ниже коду:

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/nav_graph"
 app:startDestination="@id/inboxFragment">
 <fragment
 android:id="@+id/inboxFragment"
 android:name="com.hfad.catchat.InboxFragment"
 android:label="Inbox"
 tools:layout="@layout/fragment_inbox" />
 <fragment
 android:id="@+id/helpFragment"
 android:name="com.hfad.catchat.HelpFragment"
 android:label="Help"
 tools:layout="@layout/fragment_help" />
 <fragment
 android:id="@+id/sentItemsFragment"
 android:name="com.hfad.catchat.SentItemsFragment"
 android:label="Sent Items"
 tools:layout="@layout/fragment_sent_items" />
</navigation>
```

Как видите, новой цели присваивается идентификатор `sentItemsFragment` и метка «Sent Items». Идентификатор будет использоваться в файле ресурсов меню нижней панели навигации, который будет создан на следующем этапе.

Нижней панели навигации потребуется новый файл ресурсов меню

Ранее в этой главе мы создали файл ресурсов меню с именем `menu_toolbar.xml` для добавления элемента `Help` на панель инструментов.

Хотя одно меню может совместно использоваться разными UIкомпонентами навигации, для нижней панели нам понадобится создать новое меню, потому что на ней будут отображаться два новых элемента: `Inbox` и `Sent Items`.

Чтобы создать новый файл ресурсов меню, выделите папку *app/src/main/res*, откройте меню `File`, выберите команду `New`, а затем выберите вариант создания нового файла ресурсов Android. Введите имя «menu_main», укажите тип ресурса «Menu» и убедитесь в том, что выбрано имя каталога `menu`. При нажатии кнопки OK среда Android Studio создает файл и добавляет его в папку *app/src/main/res/menu*. Затем откройте файл `menu_main.xml` и обновите его код, чтобы добавить элементы для `InboxFragment`, `SentItemsFragment` и `HelpFragment`:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
 xmlns:android="http://schemas.android.com/apk/res/android">
 <item
 android:id="@+id/inboxFragment"
 android:icon="@android:drawable/ic_dialog_email"
 android:title="Inbox" />
 <item
 android:id="@+id/sentItemsFragment"
 android:icon="@android:drawable/ic_menu_send"
 android:title="Sent Items" />
 <item
 android:id="@+id/helpFragment"
 android:icon="@android:drawable/ic_menu_help"
 android:title="Help" />
</menu>
```

После того как файл ресурсов меню будет создан, добавим нижнюю панель навигации в `MainActivity`.

Нижняя панель является разновидностью View

Нижняя панель, как и панель инструментов, является разновидностью представления, которое добавляется в макет. Код добавления нижней панели выглядит примерно так:

```
<com.google.android.material.bottomnavigation.BottomNavigationView
 android:id="@+id/bottom_nav"
 android:layout_width="match_parent"
```

```
 android:layout_height="wrap_content"
 app:menu="@menu/menu_main" />
```

Все начинается с определения нижней панели:

```
<com.google.android.material.bottomnavigation.BottomNavigationView
 ... />
```

где *com.google.android.material.bottomnavigation.BottomNavigationView* — полный путь к `BottomNavigationView` — классу, определяющему панель. Затем дополнительные атрибуты используются для определения его идентификатора и настройки его внешнего вида. В отличие от панели инструментов, вам не придется писать код Kotlin для добавления элементов на нижнюю панель. Достаточно указать, какой файл ресурсов меню следует добавить на панель, при помощи атрибута app:menu:

```
app:menu="@menu/menu_main"
```

Приведенный выше код присоединяет файл ресурсов меню `menu_main.xml` к нижней панели, а его элементы добавляются на панель во время выполнения:

Итак, теперь вы знаете, как выглядит код нижней панели навигации. Давайте добавим ее в макет `MainActivity`.

Полный код `activity_main.xml`

Ниже приведен полный код `activity_main.xml`: обновите свою версию:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">
 <com.google.android.material.appbar.MaterialToolbar
 android:id="@+id/toolbar"
 android:layout_width="match_parent"
 android:layout_height="?attr/actionBarSize"
 style="@style/Widget.MaterialComponents.Toolbar.Primary" />
 <androidx.fragment.app.FragmentContainerView
 android:id="@+id/nav_host_fragment"
 android:name="androidx.navigation.fragment.NavHostFragment"
```

```
 android:layout_width="match_parent"
 android:layout_height="match_parent""0dp"
 android:layout_weight="1"
 app:defaultNavHost="true"
 app:navGraph="@navigation/nav_graph" />
 <com.google.android.material.bottomnavigation.BottomNavigationView
 android:id="@+id/bottom_nav"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 app:menu="@menu/menu_main" />
</LinearLayout>
```

После добавления нижней панели в макет `MainActivity` нужно сделать так, чтобы она выполняла перемещения между целями.

Связывание нижней панели с контроллером навигации

Код, обеспечивающий переходы между целями с нижней панели, проще кода, необходимого для реализации навигации с панели инструментов. Все, что для этого нужно, — получить ссылку на объект `BottomNavigationView`, определяющую панель навигации, и вызвать метод `setupWithNavController()`. Код выглядит так:

```
override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 ...
 val navHostFragment = supportFragmentManager
 .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
 val navController = navHostFragment.navController
 val bottomNavView = findViewById<BottomNavigationView>(R.id.bottom_nav)
 bottomNavView.setupWithNavController(navController)
}
```

Каждый раз, когда вы щелкаете на элементе панели навигации, контроллер навигации получает его идентификатор и ищет цель с совпадающим идентификатором в графе навигации. Затем найденная цель передается хосту навигации, чтобы она отображалась на экране устройства.

Добавим код нижней панели навигации в `MainActivity.kt` и проведем тест-драйв приложения.

Обновленный код MainActivity.kt

Ниже приведен обновленный код `MainActivity.kt`; включите в него изменения, выделенные жирным шрифтом:

```
package com.hfad.catchat
...
import com.google.android.material.bottomnavigation.BottomNavigationView
class MainActivity : AppCompatActivity() {
```

```
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
setSupportActionBar(toolbar)
val navHostFragment = supportFragmentManager
.findFragmentById(R.id.nav_host_fragment) as NavHostFragment
val navController = navHostFragment.navController
val builder = AppBarConfiguration.Builder(navController.graph)
val appBarConfiguration = builder.build()
toolbar.setupWithNavController(navController, appBarConfiguration)
val bottomNavView = findViewById<BottomNavigationView>(R.id.bottom_nav)
bottomNavView.setupWithNavController(navController)
}
override fun onCreateOptionsMenu(menu: Menu): Boolean {
menuInflater.inflate(R.menu.menu_toolbar, menu)
return super.onCreateOptionsMenu(menu)
}
override fun onOptionsItemSelected(item: MenuItem): Boolean {
val navController = findNavController(R.id.nav_host_fragment)
return item.onNavDestinationSelected(navController)
|| super.onOptionsItemSelected(item)
}
}
```

На выдвижной панели могут отображаться разные навигационные элементы

Как вы уже узнали, нижняя панель навигации хорошо подходит для приложений с небольшим количеством элементов навигации, потому что на ней помещается до пяти таких элементов. Но что, если элементов меню должно быть больше? Если вы хотите, чтобы у пользователя было много вариантов навигации, выдвижная панель может оказаться более эффективным решением. Это выдвижная панель с поддержкой прокрутки, содержащая ссылки на другие части приложения, которые можно группировать по разным разделам. Выдвижные панели широко используются в приложениях Android. Например, в приложении Gmail используется выдвижная панель для перехода к разным экранам приложения, и она делится на такие секции, как категории электронной почты, недавно использованные ярлыки и все ярлыки.

Замена нижней панели выдвижной панелью

Сейчас заменим нижнюю панель, добавленную в приложение `CatChat`, выдвижной панелью. Выдвижная панель содержит графический заголовок и набор вариантов. Основные варианты позволяют перейти к `InboxFragment` и `SentItemsFragment`, а элемент `HelpFragment` будет помещен в отдельную категорию `Support`.

Выдвижная панель состоит из нескольких компонентов, которые будут рассмотрены на следующей странице. Выдвижная панель реализуется добавлением макета выдвижной панели в корень макета активности. Макет выдвижной панели содержит два представления:

- Представление для вывода основного содержимого. Обычно это макет с панелью инструментов и хостом навигации, используемый для отображения фрагментов.
- Представление для содержимого выдвижной панели. Представление навигации — разновидность фреймового макета, которая используется для отображения меню навигации. В нашем приложении она тоже отображает заголовок выдвижной панели.

Когда выдвижная панель закрыта, макет выдвижной панели выглядит как обычная активность, не считая того, что на панели инструментов присутствует значок, который открывает выдвижную панель.

Когда вы открываете выдвижную панель, представление навигации появляется сбоку, закрывая часть основного содержимого. Когда вы щелкаете на элементе, компонент `Navigation` используется для отображения соответствующей цели, а выдвижная панель закрывается.

Выдвижная панель получает свои элементы из меню

Как и панель инструментов и нижняя панель навигации, выдвижная панель получает элементы, которые должны на ней выводиться, из файла ресурсов меню. Вместо того чтобы создавать новый файл ресурсов меню для выдвижной панели, мы повторно используем menu_ main.xml: файл, который использовался для нижней панели. Напомним, как выглядит текущая версия кода в `menu_main.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
 xmlns:android="http://schemas.android.com/apk/res/android">
 <item
 android:id="@+id/inboxFragment"
 android:icon="@android:drawable/ic_dialog_email"
 android:title="Inbox" />
 <item
 android:id="@+id/sentItemsFragment"
 android:icon="@android:drawable/ic_menu_send"
 android:title="Sent Items" />
 <item
 android:id="@+id/helpFragment"
 android:icon="@android:drawable/ic_menu_help"
 android:title="Help" />
</menu>
```

Приведенный выше файл ресурсов меню, добавленный в выдвижную панель, создает список элементов, каждый из которых представлен значком. Мы настроим меню, чтобы текущий выделенный элемент выделялся цветом, а меню было разбито на категории:

Добавление категории Support как отдельного подменю

Начнем с добавления заголовка `Support`, для чего в меню добавляется новый элемент. Так как это всего лишь заголовок категории, достаточно задать текст: у элемента нет значка, и ему не нужно назначать идентификатор, так как он не будет использоваться для перехода. Заголовок `Support` создается следующим кодом:

```
<item android:title="Support">
 </item>
```

Элемент `Help` должен отображаться под заголовком `Support`, чтобы он образовал отдельную категорию. Для этого мы определим подменю внутри элемента `Support`, заданное элементом `<menu>`. В это подменю будет добавлен элемент `Help`. Ниже приведен код добавления подменю с элементом `Help`; код `menu_main.xml` будет обновлен через пару страниц:

```
<item android:title="Support">
 <menu>
 <item
 android:id="@+id/helpFragment"
 android:icon="@android:drawable/ic_menu_help"
 android:title="Help" />
 </menu>
 </item>
```

Подменю содержит всего один элемент. Если вы хотите, чтобы оно содержало несколько элементов, просто включите каждый элемент в подменю. И это весь код, необходимый для добавления категории Support в меню выдвижной панели. Теперь можно сделать следующий шаг — добавить дополнительное выделение цветом текущего элемента, выбранного пользователем.

Выделение текущего элемента в группах

Текущее меню создает выдвижную панель навигации, которая изменяет цвет текста текущего выделенного элемента. Чтобы пользователь более четко видел, какой элемент выделен, можно добавить дополнительное выделение цветом:

Чтобы добавить дополнительное выделение, следует включить элементы в группу при помощи элемента `<group>`. Затем атрибут `android:checkableBehavior` определяет поведение группы при выделении элемента. Эта задача решается следующим кодом (код `menu_main.xml` будет обновлен потом):

```
<group android:checkableBehavior="single">
 <item
 android:id="@+id/inboxFragment"
 android:icon="@android:drawable/ic_dialog_email"
 android:title="Inbox" />
 <item
 android:id="@+id/sentItemsFragment"
 android:icon="@android:drawable/ic_menu_send"
 android:title="Sent Items" />
 </group>
```

Приведенный выше код присваивает атрибуту `android:checkableBehavior` значение «single». Это означает, что в любой момент времени выделенным может быть только один элемент в группе — элемент, выбранный пользователем. Вот и все, что необходимо сделать, чтобы придать меню нужный внешний вид и поведение при использовании в качестве выдвижной панели навигации. Полный код приводится на следующей странице.

Полный код menu_main.xml

Обновим меню выдвижной панели навигации, чтобы меню `Help` отображалось в категории `Support`. Также мы определим группы, чтобы добавить дополнительное цветовое выделение для элемента, выбранного пользователем. Ниже приведен полный код `menu_main.xml`; обновите его:

```
<?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/inboxFragment"
 android:icon="@android:drawable/ic_dialog_email"
 android:title="Inbox" />
 <item
 android:id="@+id/sentItemsFragment"
 android:icon="@android:drawable/ic_menu_send"
 android:title="Sent Items" />
 </group>
 <item android:title="Support">
 <menu>
 <group android:checkableBehavior="single">
 <item
 android:id="@+id/helpFragment"
 android:icon="@android:drawable/ic_menu_help"
 android:title="Help" />
 </group>
 </menu>
 </item>
</menu
```

С меню мы разобрались. Перейдем к созданию заголовка выдвижной панели.

Создание заголовка выдвижной панели

Заголовок выдвижной панели представляет собой простой макет, который будет размещен в новом файле макета с именем `nav_header.xml`. Создайте этот файл: выделите папку *app/src/main/res/layout* в Android Studio и выберите команду `File→New→Layout`. Введите имя макета «nav_header», и если среда запросит тип ресурса, выберите `Layout`.

Добавление графического файла и обновление кода nav_header.xml

Макет состоит из одного графического изображения, которое необходимо разместить в папке *app/src/main/res/drawable*. Вероятно, среда Android Studio создала эту папку за вас при создании

проекта. Если папка отсутствует, добавьте ее вручную: выделите папку *app/src/main/res*, откройте меню `File`, выберите команду `New` и щелкните на варианте создания нового каталога ресурсов Android. Выберите тип ресурса `Drawable`, введите имя «drawable» и щелкните на кнопке OK. После того как папка drawable будет создана, загрузите файл *kitten_small.webp* из ресурсов и добавьте его в папку `drawable`.

Для добавления изображения в `nav_header.xml` будет использоваться элемент `<ImageView>`. Вы уже знаете, как пользоваться этим элементом, поэтому обновите код `nav_header.xml` и приведите его в соответствие со следующим кодом:

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

Итак, заголовок для выдвижной панели готов; давайте добавим саму панель.

Как создать выдвижную панель

Чтобы создать выдвижную панель навигации, добавьте макет `DrawerLayout` в макет активности как корневой элемент. Макет `DrawerLayout` должен содержать две составляющие: представление или группу представлений для содержимого активности (первый элемент) и представление `NavigationView`, определяющее содержимое выдвижной панели (второй элемент).

Типичный код `DrawerLayout` выглядит примерно так:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/drawer_layout"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">
 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical">
 ...
 </LinearLayout>

 <com.google.android.material.navigation.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_main"/>
</androidx.drawerlayout.widget.DrawerLayout>
```

Внешний вид выдвижной панели определяется двумя ключевыми атрибутами `<NavigationView>`: `app:headerLayout` и `app:menu`. Атрибут `app:headerLayout` задает макет, который должен использоваться для заголовка выдвижной панели (в данном случае `nav_header.xml`). Этот атрибут не является обязательным.

Атрибут `app:menu` указывает, какой файл ресурсов меню содержит элементы выдвижной панели (в данном случае `menu_main.xml`). Если не включить этот атрибут, выдвижная панель не будет содержать никаких элементов.

Полный код activity_main.xml

Требуется заменить нижнюю панель навигации активности `MainActivity` выдвижной панелью навигации. Код для решения этой задачи приведен ниже; обновите файл `activity_main.xml`:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    tools:context=".MainActivity">
    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <com.google.android.material.appbar.MaterialToolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    style="@style/Widget.MaterialComponents.Toolbar.Primary" />
    <androidx.fragment.app.FragmentContainerView
    android:id="@+id/nav_host_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph" />
```

```
</LinearLayout>

<com.google.android.material.navigation.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_main"/>
</androidx.drawerlayout.widget.DrawerLayout>
```

Выдвижная панель навигации добавлена в макет

Мы заменили нижнюю панель в макете `MainActivity` выдвижной панелью. На ней выводится графический заголовок и все элементы, заданные в файле ресурсов меню `menu_main.xml`. Но прежде чем запускать приложение, необходимо связать выдвижную панель с контроллером навигации, чтобы щелчок на элементе вызывал переход к соответствующему фрагменту. Также необходимо настроить панель инструментов, чтобы на ней размещался значок, открывающий или закрывающий выдвижную панель навигации. Оба изменения реализуются изменением кода в файле `MainActivity.kt`. Давайте посмотрим, как это делается.

Настройка значка выдвижной панели на панели инструментов и связывание выдвижной панели с контроллером навигации

Ранее в этой главе мы настроили панель инструментов так, чтобы на ней выводилась метка текущей цели и кнопка `Up`. Для этого мы построили объект `AppBarConfiguration` и связали его с панелью инструментов. Теперь на панели инструментов нужно разместить значок выдвижной панели, для чего можно добавить выдвижную панель `AppBarConfiguration`. Код выглядит так:

```
val drawer = findViewById<DrawerLayout>(R.id.drawer_layout)
val builder = AppBarConfiguration.Builder(navController.graph)
builder.setOpenableLayout(drawer)
val appBarConfiguration = builder.build()
toolbar.setupWithNavController(navController, appBarConfiguration)
```

Приведенный выше код добавляет макет выдвижной панели на объект `AppBarConfiguration`. Это позволяет панели инструментов взаимодействовать с выдвижной панелью и включать значок выдвижной панели на каждый экран без кнопки `Up`

Наконец, необходимо обеспечить переход с выдвижной панели к правильной цели каждый раз, когда пользователь щелкает на одном из ее элементов. Как и в случае с нижней панелью навигации, для этого необходимо связать выдвижную панель с контроллером навигации. Код для решения этой задачи выглядит так:

```
val navHostFragment = supportFragmentManager
 .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
```

```
val navController = navHostFragment.navController
val navView = findViewById<NavigationView>(R.id.nav_view)
NavigationUI.setupWithNavController(navView, navController)
```

Каждый раз, когда пользователь щелкает на элементе выдвижной панели, контроллер навигации получает идентификатор из файла ресурсов меню и ищет совпадающий идентификатор в графе навигации. Затем он переходит к цели с указанным идентификатором. И это весь код, необходимый для управления поведением выдвижной панели навигации. Добавим его в `MainActivity.kt` и проведем тест-драйв приложения.

Полный код MainActivity.kt

В коде `MainActivity` необходимо заменить код нижней панели навигации кодом выдвижной панели. Обновите `MainActivity.kt`:

```
package com.hfad.catchat
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import androidx.navigation.findNavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.onNavDestinationSelected
import androidx.navigation.ui.setupWithNavController

import androidx.drawerlayout.widget.DrawerLayout
import androidx.navigation.ui.NavigationUI
import com.google.android.material.navigation.NavigationView
import com.google.android.material.appbar.MaterialToolbar
class MainActivity : AppCompatActivity() {
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
 setSupportActionBar(toolbar)
 val navHostFragment = supportFragmentManager
 .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
 val navController = navHostFragment.navController
 val drawer = findViewById<DrawerLayout>(R.id.drawer_layout)
 val builder = AppBarConfiguration.Builder(navController.graph)
 builder.setOpenableLayout(drawer)
 val appBarConfiguration = builder.build()
 toolbar.setupWithNavController(navController, appBarConfiguration)

 val navView = findViewById<NavigationView>(R.id.nav_view)
 NavigationUI.setupWithNavController(navView, navController)
override fun onCreateOptionsMenu(menu: Menu): Boolean {
 menuInflater.inflate(R.menu.menu_toolbar, menu)
 return super.onCreateOptionsMenu(menu)
 }
```

```
  override fun onOptionsItemSelected(item: MenuItem): Boolean {
  val navController = findNavController(R.id.nav_host_fragment)
  return item.onNavDestinationSelected(navController)
  || super.onOptionsItemSelected(item)
  }
}
```

Резюме

Для добавления панели приложения по умолчанию используется тема.

- Атрибут `android:theme` в `AndroidManifest.xml` указывает, какая тема должна применяться.
- Стили определяются элементами `<style>` в одном или нескольких ресурсных файлах.
- Панель приложения по умолчанию можно заменить панелью инструментов, поддерживающей новейшую функциональность Android.
- Панель инструментов `Material` хорошо работает в сочетании с темами `Material`.
- Файл ресурсов меню добавляет элементы к панелям инструментов, нижним и выдвижным панелям навигации.
- Проследите за тем, чтобы идентификаторы элемента и цели в файле ресурсов меню и графе навигации совпадали. От совпадения идентификаторов зависит работа компонента `Navigation`. Элементы добавляются на панель инструментов вызовом `onCreateOptionsMenu()`.
- Навигация по панели инструментов реализуется методом `onOptionsItemSelected()`.
- Объект `AppBarConfiguration` используется для настройки панели инструментов, обеспечивающей ее работу с компонентом `Navigation`.
- Нижняя панель может содержать до пяти элементов.
- Выдвижная панель может содержать много элементов, сгруппированных по категориям.
- Чтобы создать выдвижную панель навигации, добавьте выдвижную панель в макет активности. Первым элементом макета выдвижной панели должно быть представление, определяющее основное содержимое активности. Второй элемент определяет содержимое выдвижной панели.