Практическая работа № 3

Изучение основ разработки интерфейсов мобильных приложений

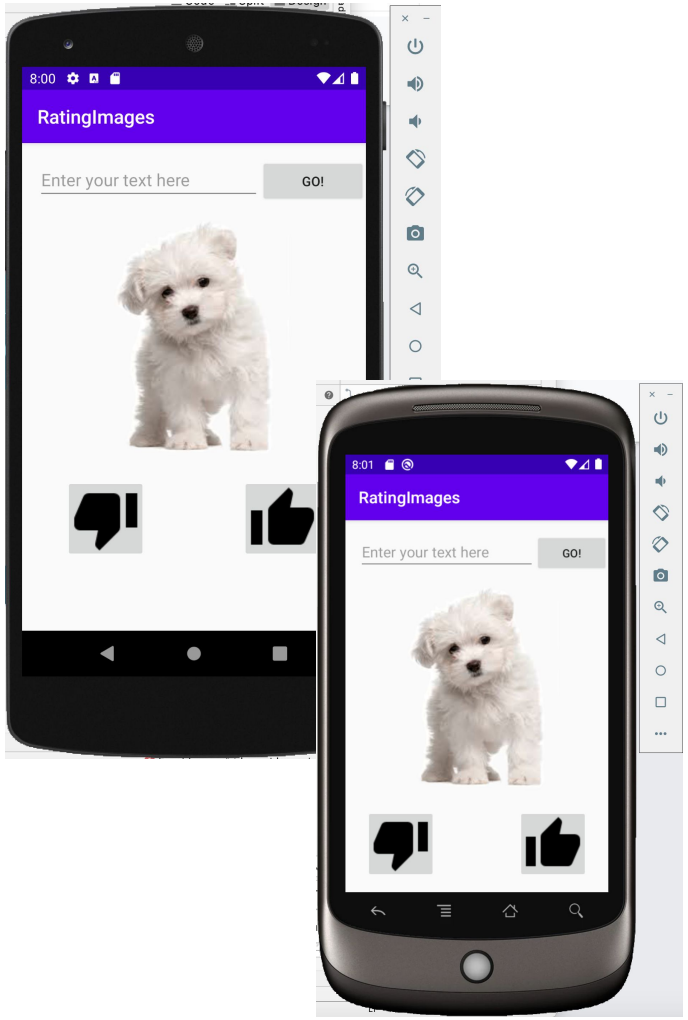
Цель практической работы

- Изучение основ разработки интерфейсов мобильных приложений

Задачи практической работы:

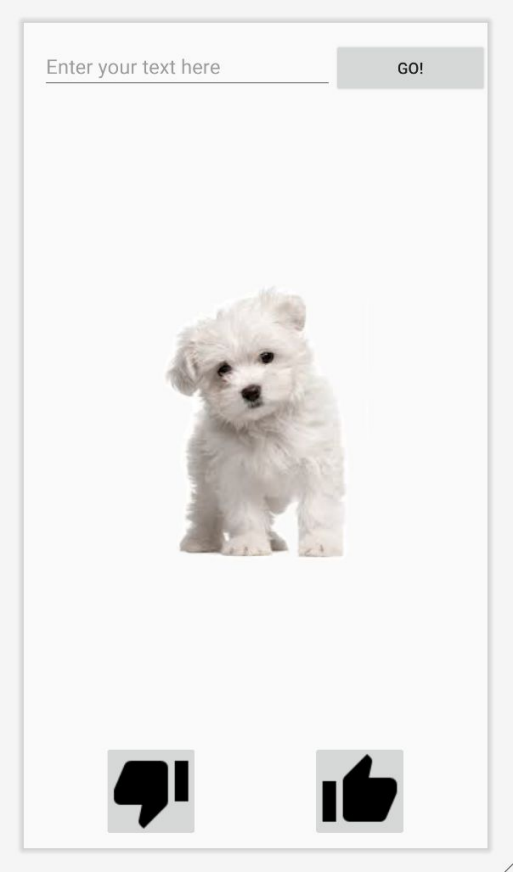
- Изучить элементы интерфейса
- Практическим путём научиться размещать элементы и менять их свойства
- Разработать прототип интерфейса собственного приложения

Создание прототипа интерфейса



- Практическая работа посвящена разработке интерфейсов мобильных приложений. Работа содержит подробное описание построения гармоничного понятного пользовательского интерфейса для главной активности приложения и описание основных элементов интерфейса. Практическая работа поможет выбрать концепцию своего приложения и начать разработку его интерфейса.
- Рассмотрим пример разработки интерфейса приложения, которое ищет в сети *Интернет* изображения по запросу пользователя, позволяет оценивать их, скачивать, и посещать *интернет*-страницы сайтов, на которых было найдено изображение.

Создание заготовки для приложения



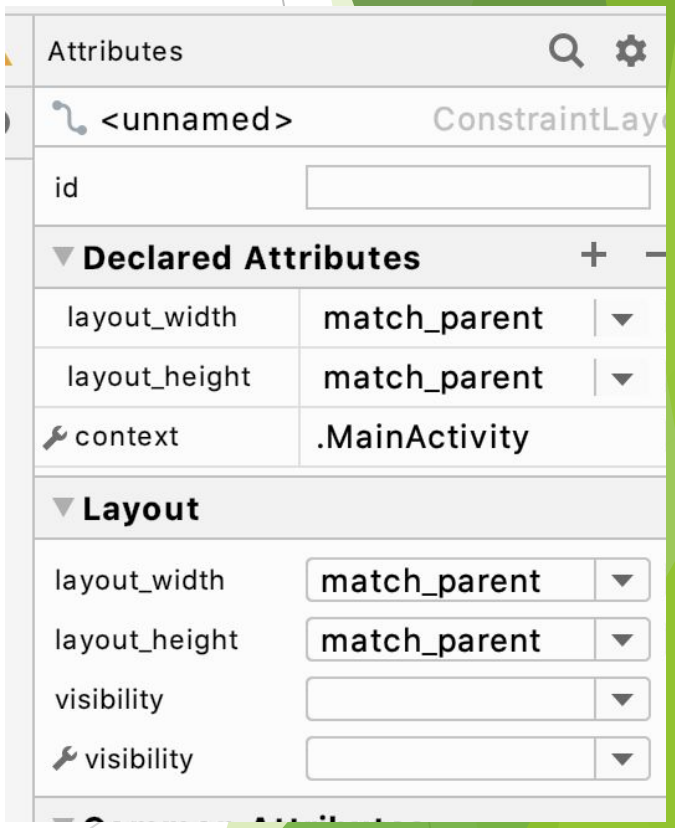
- Выглядеть главное окно будет примерно так (см рисунок)
- На нём присутствуют поле ввода текста для запроса пользователя и кнопка, начинающая поиск изображений. Внизу экрана две кнопки: "**like**" и "**dislike**", с их помощью пользователь сможет оценить изображение. После того, как пользователь сделает оценку изображения, текущее изображение закрывается и загружается следующее.
- Итак, начнём с создания нового проекта. Назовём его "**RatingImages**" ("Рейтинг изображений").

Создание заготовки для приложения

- После создания проекта откройте **activity_main.xml** из каталога **res/layout/**.
- Когда вы откроете файл **activity_main.xml**, вы увидите графический редактор макета. Благодаря этому редактору создание интерфейсов стало ещё интереснее, поскольку добавить элемент на форму можно при помощи перетаскивания мышью, к тому же, благодаря графическому редактору, не обязательно запускать эмулятор, чтобы увидеть результат своих трудов.
- Теперь щелкните по вкладке **activity_main.xml** в нижней части экрана. Открылся XML-редактор кода. Этот способ редактирования стандартный, но все изменения, вносимые в этот документ, можно так же ощутить визуально, перейдя на графический редактор.
- Вернёмся на вкладку с графическим редактором. Во-первых, подготовим документ к началу работы, для этого удалите <TextView>.

Создание заготовки для приложения

- На рабочей области экрана остался один элемент. Это макет <ConstraintLayout>. В нём позиция дочерних элементов может быть описана по отношению друг к другу или к родителю.
- Два атрибута, ширина и высота (android:layout_width и android:layout_height), требуются для всех элементов для того, чтобы указать их размер.
- Так как <ConstraintLayout> - это корень в макете, то нужно, чтобы он заполнял всю область экрана. Это достигается при помощи установки параметра "match_parent" для ширины и высоты. Это значение указывает, что ширина и высота элемента будет равна ширине и высоте родителя.
- При установке этого значения мы увидим еще один возможный параметр: "wrap_content".
- Параметр "wrap_content" указывает, что представление будет увеличиваться при необходимости, чтобы поддерживать соответствие содержанию экрана.



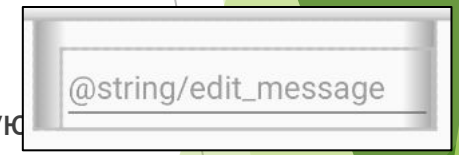
Создание заготовки для приложения. Добавление текстового поля


```
<EditText
    android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
```

- Для начала добавьте элемент <LinearLayout> с горизонтальной ориентацией в <ConstraintLayout>, и укажите для ширины и высоты параметр "wrap_content". Теперь, для создания пользовательского редактируемого текстового поля, добавьте элемент <EditText> с параметром "wrap_content" для ширины и высоты в <LinearLayout>.
- Возможно, появился желтый предупреждающий знак, но сейчас это не важно, со временем он исчезнет. Наличие таких предупреждений никак не влияет на компилируемость проекта.
- При указании id, знак (@) требуется в том случае, если вы имеете в виду любой ресурс объекта из XML-файла. За ним следуют тип ресурса (в данном случае ID), косая черта (слеш) и имя ресурса (editText1).
- Знак плюс (+) перед типом ресурсов необходим только тогда, когда вы впервые определяете идентификатор ресурса.
- По сути, id, который создается автоматически, уже уникален, но грамотнее переименовывать id в соответствии со назначением элемента.
- Зададим id для текстового поля. Для этого прямо в поле id вводим edit_message", открываем графический редактор и убеждаемся, что в свойствах текстового поля в графе **id** будет **android:id="@+id/edit_message"**

Создание заготовки для приложения. Добавление текстового поля

- Добавим в код ещё две строки:
- android:ems="10" - задает соответствия для симметричного отображения шрифтов,
- android:hint="@string/edit_message" - содержание тестового поля "по умолчанию", т.е. пока пользователь не начал вводить в поле текст. Вместо того, чтобы использовать просто слово (например android:hint="message"), что крайне не удобно при изменении основного языка приложения, используется ссылка на значение, хранящееся в файле **strings.xml**. Поскольку это относится к конкретному ресурсу (а не только к **id**), знак плюс не нужен.
- Однако, мы ещё не определили строку ресурсов файле **strings.xml**, и потому вы получите следую
- Для того, чтобы ссылка на ресурс начала работать, нужно этот ресурс создать.
- Откройте файл **res/values/strings.xml**. Очевидно, что его тоже можно редактировать двумя способами: Open Editor и вручную.



app_name	app/src/main/res	☐	RatingImages
edit_message	app/src/main/res	☐	Enter your text here

```
<string name="app_name">RatingImages</string>
<string name="edit_message">Enter your text here</string>
```

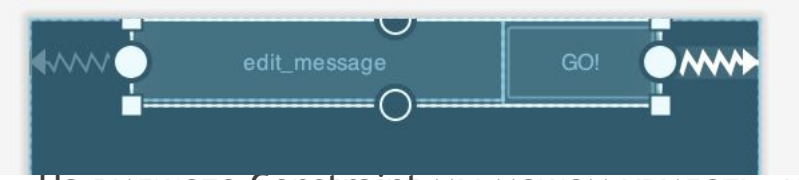
- Аналогично создадим id для <LinearLayout>

Создание заготовки для приложения. Добавление кнопки

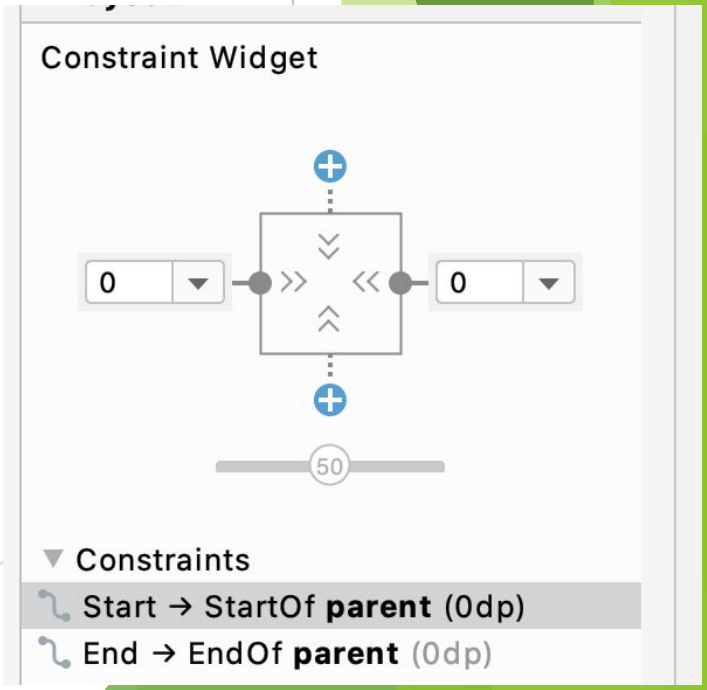
- Теперь добавьте <Button> в макет после элемента <EditText>:
- Чтобы кнопка трансформировалась в соответствии с текстом кнопки, ширина и высота должны быть установлены во "wrap_content".
- Теперь поменяем надпись на кнопке на "**Go!**" с помощью ссылки на ресурс в XML-коде главной активности и добавления одного ресурса в файл **strings.xml**:

  <**string name="button_send**">GO!</**string**>

- Теперь, когда мы поместили два главных представления на <LinearLayout> элемент, настало время добавить ещё два параметра для этого элемента.
- Речь идет о "приращении" правого и левого краёв лейаута к правому и левому краям <RelativeLayout> элемента соответственно. А сделать это проще простого - просто потяните мышкой один край к другому!

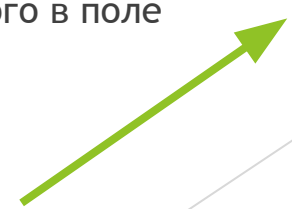


- На виджете Constraint мы можем увидеть, к чему привязались края



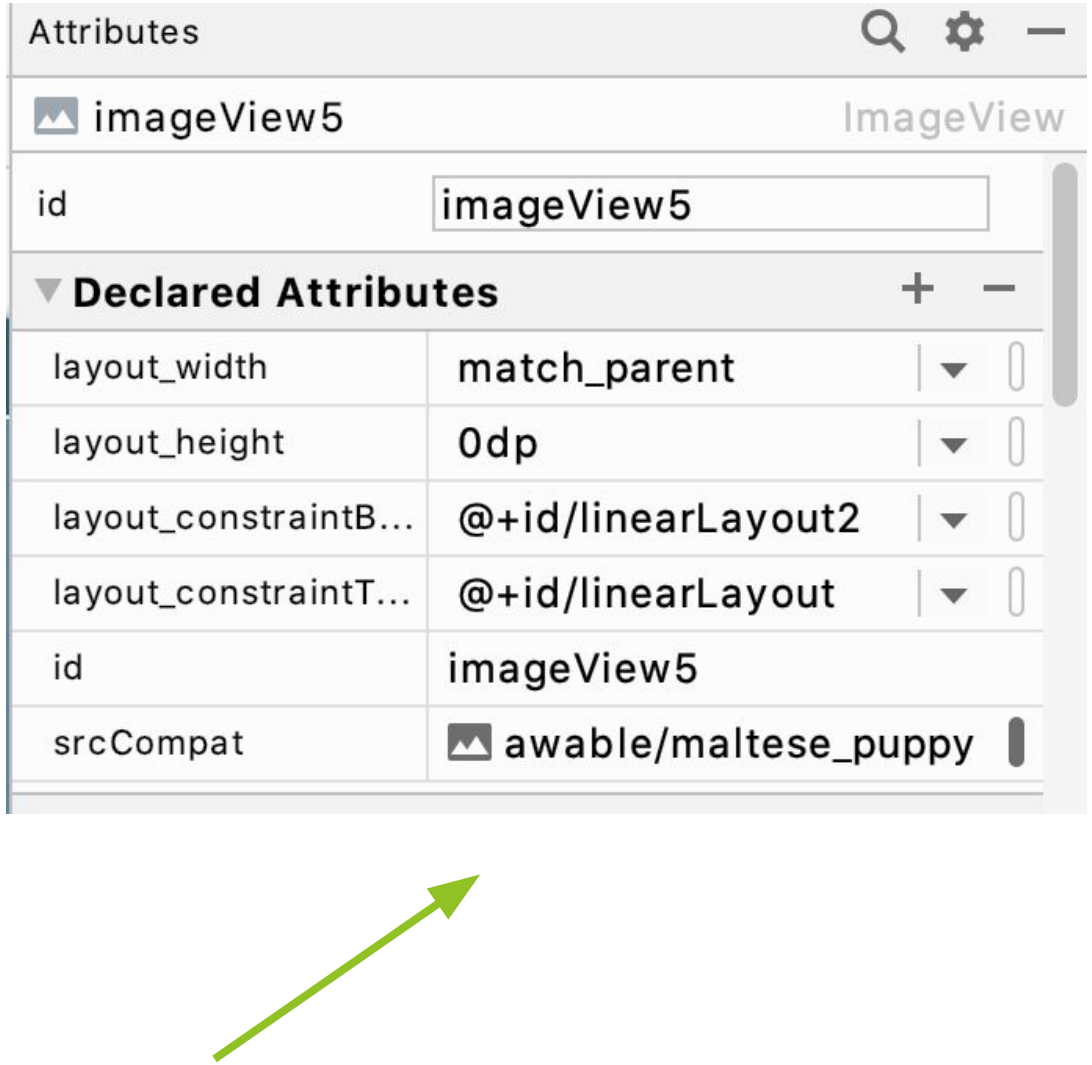
Создание заготовки для приложения. Смена фона

- Идём дальше - попробуем поменять фон.
- Чтобы изменить цвет фона на чёрный, нужно в XML-коде главной активности написать одну строку в блоке <RelativeLayout> элемента:
- android:background="#000000".
- Сохраните и проверьте результат, открыв графический редактор.
- Чтобы фоном сделать картинку нужно положить в папку drawable изображение - картинка например называется **fon.png**:
- @drawable/fon
- Несомненно, фон смотрится хорошо, но очевидно, что кнопка и поле ввода просто затерялись, а это значит, что для этого приложения такой фон не подходит. Можно продолжить подбирать изображения на фон, но лучше создать черепичную заливку небольшим изображением. На этом сайте можно найти узор на любой вкус!
- В <RelativeLayout> стоит добавлять android:background только в том случае, если вы хотите неподвижный фон, а в <ScrollView> чтобы фон прокручивался вместе с контентом.
- Если вы выбрали тёмный фон, то стоит поменять цвет текста, вводимого в поле ввода, например на белый.
- Для этого в блок <EditText> добавим строчку
- android:textColor="#ffffff"



Область просмотра изображений

- Теперь займемся созданием области отображения изображений, которые пользователь будет оценивать.
- Добавьте » изображение" <ImageView> в <ConstraintLayout>:
- Укажем этому элементу ширину, высоту и id. Предупреждение должно пропасть.
- Теперь для наглядности поместим туда изображение.
- Сначала поместим само изображение в папку res/drawable/ и обновим её.
- Это нужно для того, чтобы новые данные загрузились в проект.



Кнопки «Like» и «Dislike»

- Пришло время создать кнопки оценивания.
- Для этого добавьте на форму <RelativeLayout>, задайте для него ширину и высоту wrap_content, и укажите id.
- Снова в папку **res/drawable/** нужно добавить файлы. Найдите изображения "Палец вверх" и "Палец вниз", и поместите их в эту папку, после чего обновите её.
- Изображения и другие полезные файлы можно скачать здесь.
- Добавьте <ImageButton>, выберите изображение "Палец вверх", и переместите <RelativeLayout> в такое положение:
- Добавьте еще одну кнопку - кнопку "**Палец вниз**". Она "наложилась" на первую кнопку. Чтобы это исправить, проделайте с <RelativeLayout> то же самое, что и с <LinearLayout>: растяните элемент влево и вправо, до получения такого результата:
- Теперь расставьте кнопки по краям так, чтобы они "прикрепились" к краям.
- Готово! Теперь можно запустить эмулятор и посмотреть, что получилось.