Дополнение к логин экрану

Добавим еще кое-чего

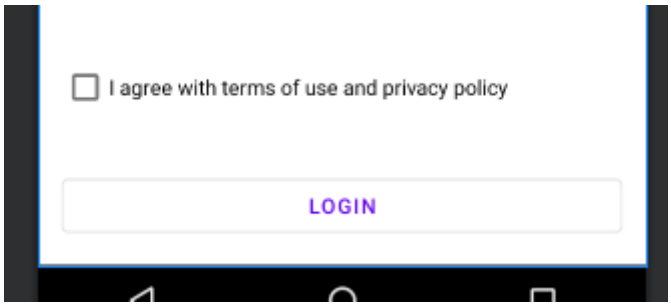
Содержание

1. Checkbox

Помните мы написали текстовку где было 2 ссылки – условия пользовательского соглашения и политика конфиденциальности? Иногда дизайнеры делают чекбокс, который нужно проставить чтобы залогиниться. Давайте посмотрим на этот простой элемент.

```
    <Space...>

    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/padding_small"
        android:text="@string/agreement_full_text" />
    <Button...>

</LinearLayout>
```

Если вы не читали предыдущую лекцию, то настоятельно рекомендую, ибо мы продолжаем писать в том же проекте и разметку мы дополнили из той что была там. Как видите ничего сложного – чекбокс, добавили ему отступов чтобы не прилипло к стенкам и можно было ставить галку. Плюс мы добавили сразу текст. Т.е. нам не нужен текствью для этого. Хотя давайте глянем на дизайн



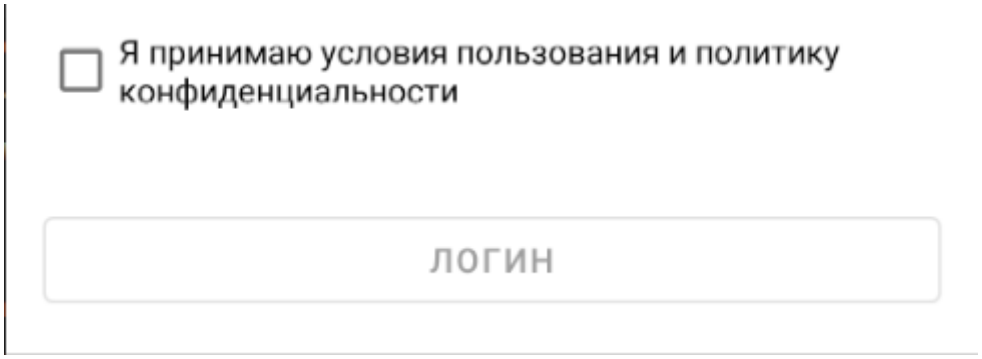
Если помните то мы должны были в коде поменять текст чтобы куски можно было кликать. Как же сделать это теперь? Да все так же на самом деле, через код.

```
val checkBox = findViewById<CheckBox>(R.id.checkBox)
val spannableString = SpannableString(getString(R.string.agreement_full_text))
checkBox.text = spannableString
```

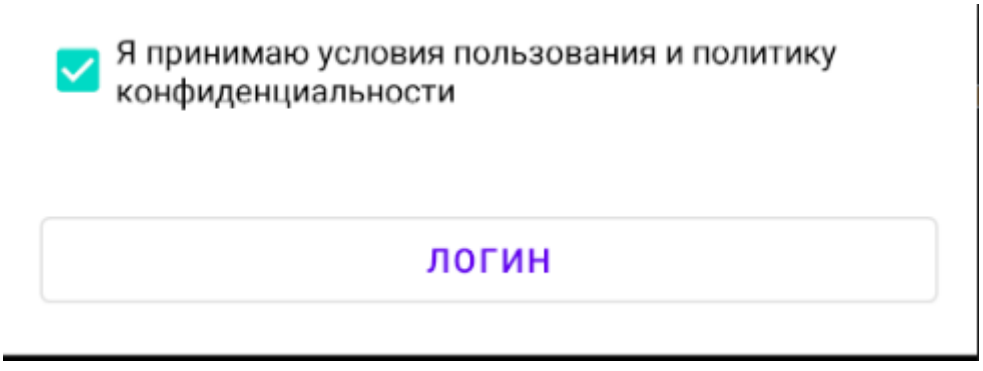
Код можете взять из первой лекции, здесь не буду копипастить. Ладно, давайте тогда рассмотрим такой кейс – пока нет галки в чекбоксе кнопка не должна быть доступна для нажатия. Как это сделать? Во-первых надо в начале убрать доступ к кнопке и потом написать слушатель нажатия. Смотрим

```
loginButton.isEnabled = false
checkBox.setOnCheckedChangeListener { _, isChecked ->
    loginButton.isEnabled = isChecked
}
```

Можете запустить проект и посмотреть как оно работает

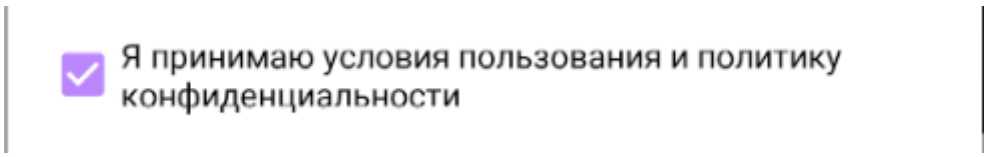


И ставим галку – кнопка становится активной



Конечно же вы можете задать цвет какой вам нужно при поставленной галке

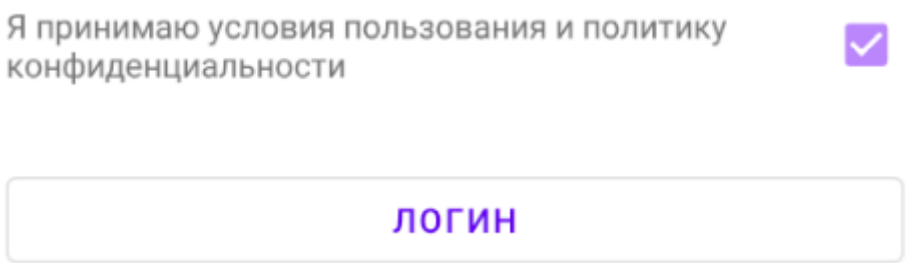
```
28    <CheckBox
29        android:id="@+id/checkBox"
30        android:buttonTint="@color/purple_200"
31        android:layout_width="match_parent"
```



И здесь мы впервые познакомимся с понятием костыля

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:gravity="center_vertical"
    android:layout_height="wrap_content">

    <TextView
        android:padding="@dimen/padding_small"
        android:text="@string/agreement_full_text"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"/>
    <CheckBox
        android:id="@+id/checkBox"
        android:buttonTint="@color/purple_200"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>

<Button...>
```

В правильной вселенной мы должны были легко и просто поменять расположение текста относительно чекбокса. Но у нас нет такого атрибута! Потому приходится придумывать иные решения. Самое простое что можно сделать – написать текст отдельно и потом справа чекбокс. Но стоп. У нас же контейнер линейный и он вертикальный, как это сделать? Можно добавить в него новый контейнер с горизонтальной ориентацией и все будет работать. Чтобы текст занимал все свободное пространство по ширине мы дали ему вес 1. Ну и центрировали по вертикали всех кто вложен в горизонтальный линейный контейнер. Итого



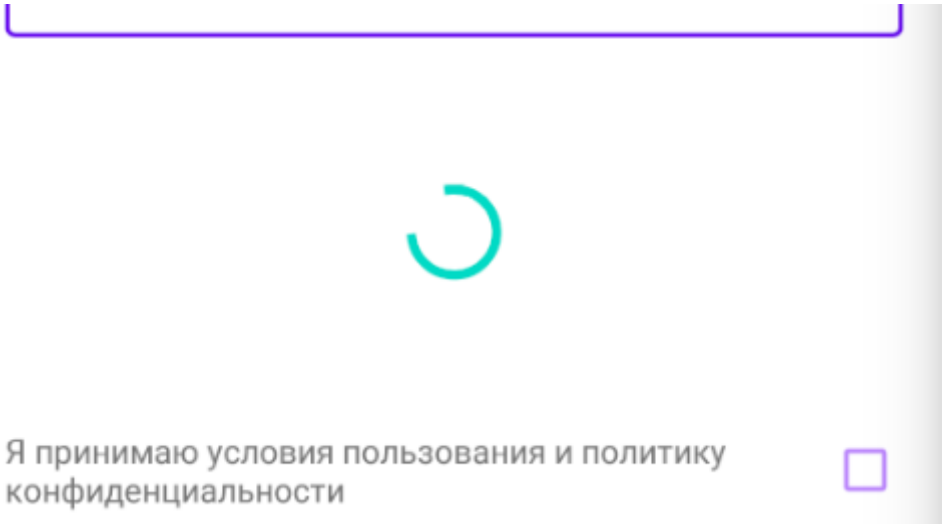
И да, не забудьте убрать из кода что мы сетим чекбоксу текст. Вот и все. Больше особо нечего рассказать про чекбокс.

2. ProgressBar

Мы помним, что после того как юзер ввел корректные данные в поле ввода и нажал на чекбокс мы сможем отправить условный запрос на сервер и чтобы юзер не нажимал на кнопку в течение этого времени по несколько раз убирали доступ. Но подождите, как юзер узнает что приложение ваше не зависло и что-то происходит? Правильным ответом на этот вопрос является – показывать загрузчик. Как понять что юай не замерз? Если на нем что-то крутится. Для этого давайте вместо Space напишем Progressbar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_weight="1" />

<LinearLayout...>

<Button...>

</LinearLayout>
```

И давайте запустим проект посмотрим на него



Да, классно. Зеленая штука крутится. Кстати, почему она зеленая? Чекбокс тоже был зеленый и мы просто поставили ему другой цвет. Не хотелось бы делать это каждый раз для всех и каждого. Давайте заглянем в наш colors.xml

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

Теперь давайте посмотрим на наши стили/темы

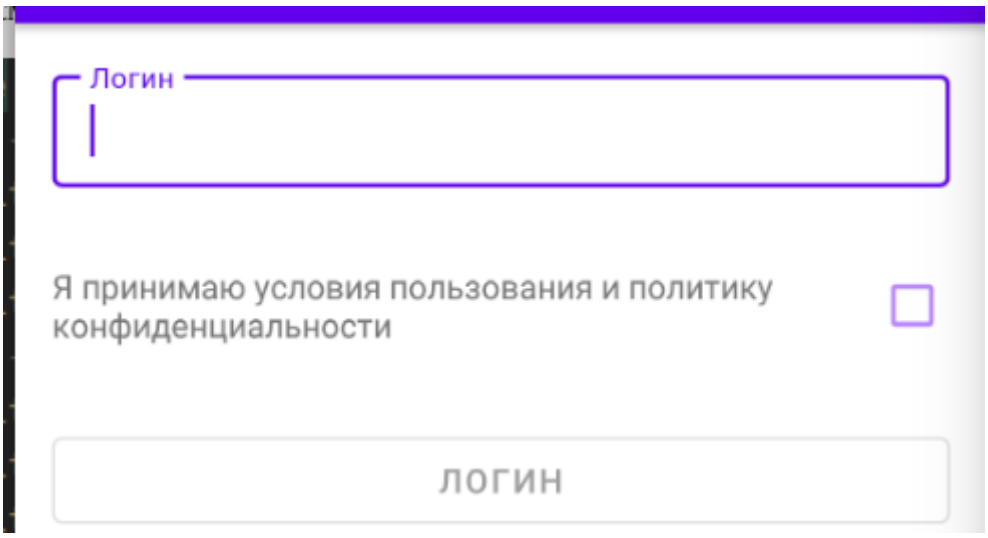
```
<style name="Theme.HelloWorld" parent="Theme.MaterialComponents.D
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/purple_500</item>
    <item name="colorPrimaryVariant">@color/purple_700</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
```

Можете просто заменить пестрый зеленый на фиолетовый и все заработает. В теме указаны цвета для primary и secondary цветов – можете погуглить где какой используется.

Но подождите, мы написали прогрессбар, но нам не нужно чтобы он был виден сразу на старте. А только тогда, когда валидный эл.адрес был введен и проставлена галка. Что же делать? Нужно его скрывать на старте. Но как? Для этого есть атрибут visibility и конечно же в коде его можно менять обратно.

```
<ProgressBar
    android:id="@+id/progressBar"
    android:visibility="gone"
    android:layout_width="wrap_content"
```

И давайте запустим проект посмотрим что мы натворили. Ведь если помните мы специально ставили Space чтобы поле ввода было наверху экрана, а кнопка и теперь еще и чекбокс были внизу. Делая gone мы убрали вообще вью которая служила нам разделителем (я еще убрал gravity из LinearLayout корневого). Значит надо использовать invisible для этого. Пробуем



Ранее у нас в корневом было gravity = center из-за чего наш прогрес был по центру

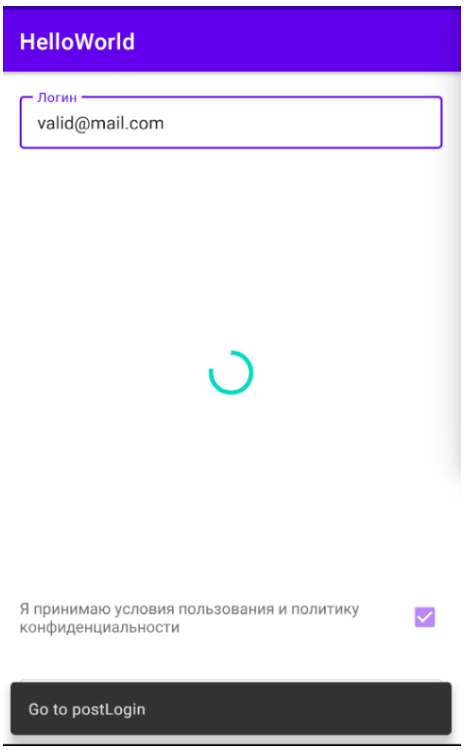
```
<ProgressBar
    android:layout_gravity="center_horizontal"
    android:id="@+id/progressBar"
    android:visibility="invisible"
```

Теперь же я убрал из корневого gravity и потому чтобы прогресс тоже был по центру по горизонтали надо это прописать. И давайте в коде покажем прогресс когда все ок.

```
val progressBar = findViewById<View>(R.id.progressBar)
loginButton.setOnClickListener {  it: View!
    if (EMAIL_ADDRESS.matcher(textInputEditText.text.toString()).matches()) {
        hideKeyboard(textInputEditText)
        loginButton.isEnabled = false
        progressBar.visibility = View.VISIBLE
        Snackbar.make(loginButton,  text: "Go to postLogin", Snackbar.LENGTH_LONG).show()
```

Давайте запустим код и посмотрим как все работает

И здесь нас ожидает новая проблема. Да, все условия соблюдены. Мы показываем прогресбар пока ждем ответа от сервера, мы убрали возможность для повторной отправки данных на сервер, но у нас теперь чекбокс и если помните можно на него нажать и убрать галку и когда повторно нажмем то кнопка станет доступной для повторного нажатия. Итого – наше решение не работает. Хорошо, что делать тогда? Делать чекбокс недоступным для нажатия. Но вам не кажется что это тоже не решение? Ведь юзер может пока ждет ответа от сервера еще и в поле ввода что-то ввести. Значит нам надо заблокировать вообще весь экран. Как это сделать? Я предлагаю такой вариант – пусть наш прогресбар будет на весь экран и под ним не будут видны наши данные.



Но как это сделать? Ведь мы написали что наш прогресс стоит между полем ввода и чекбоксом. Значит нужно переделать нашу разметку таким образом, чтобы можно было убирать видимость всех элементов отдельно и показывать прогресс отдельно. Как это делается? Нам нужно 2 контейнера – в одном будет наш линейный где все вью и он уже написан и второй контейнер в котором будет наш контейнер линейный и наш прогрессбар.

```
<?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">
    <LinearLayout
        android:id="@+id/contentLayout"
        tools:visibility="gone"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/textInputLayout"
            
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.Dense
"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="@dimen/padding_small"
            android:hint="@string/login">

            <com.google.android.material.textfield.TextInputEditText
```

```
            android:id="@+id/textInputEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress" />
        </com.google.android.material.textfield.TextInputLayout>
```

Да, не забудьте вернуть Space вот сюда, ведь наш экран должен по прежнему отображаться корректно

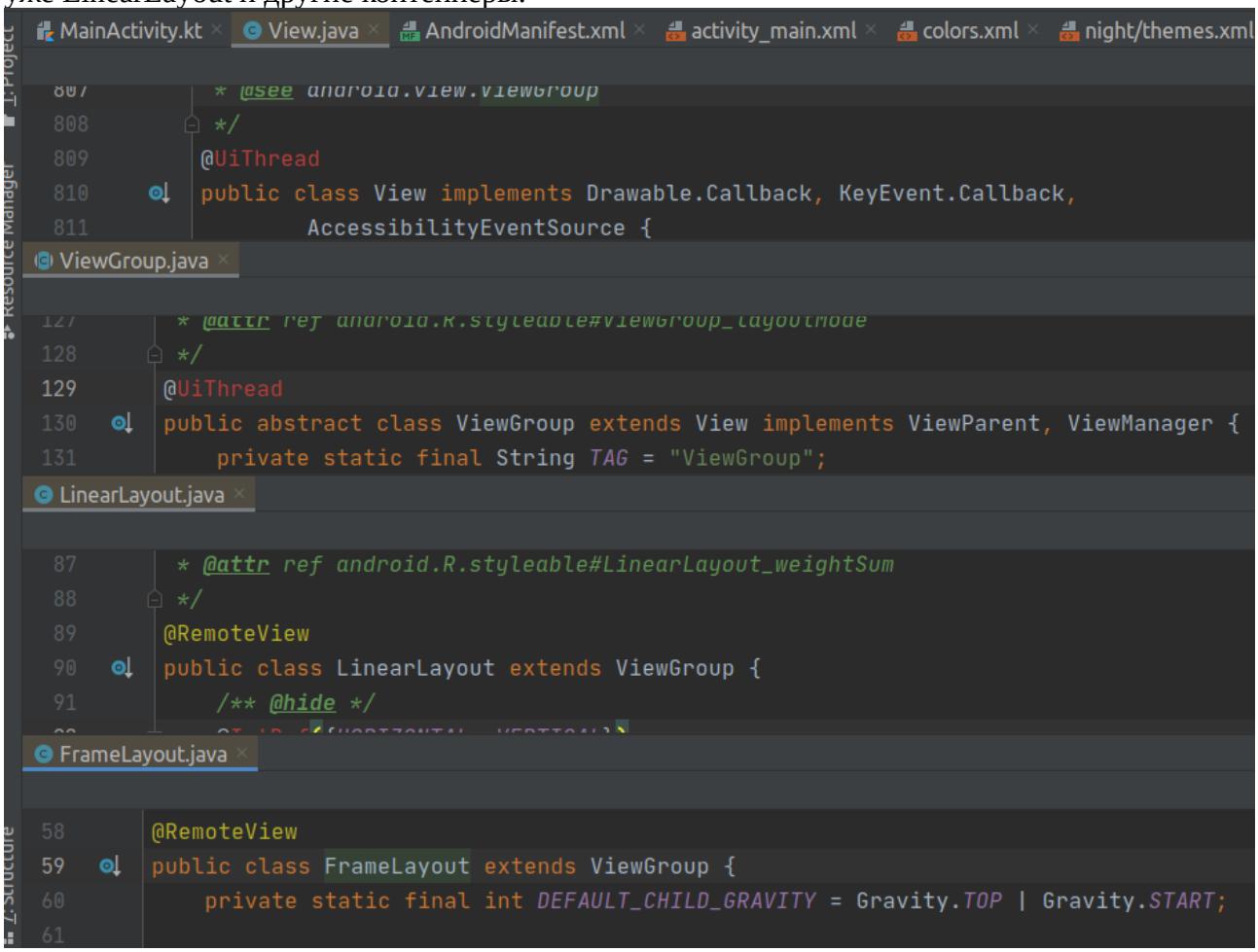
```
        <Space
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:orientation="horizontal">
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:padding="@dimen/padding_small"
                android:text="@string/agreement_full_text" />
            <CheckBox
                android:id="@+id/checkBox"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:buttonTint="@color/purple_200" />
        </LinearLayout>
        <Button
            android:id="@+id/loginButton"
            style="@style/Widget.MaterialComponents.Button.OutlinedButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="@dimen/padding_small"
            android:text="@string/login" />
    </LinearLayout>
    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:visibility="gone"
        tools:visibility="visible" />
</FrameLayout>
```

Как видите у нас главный контейнер поменялся на Framelayout и он самый простой и не имеет логики отображения – как в него положите вью и контейнеры так и будут

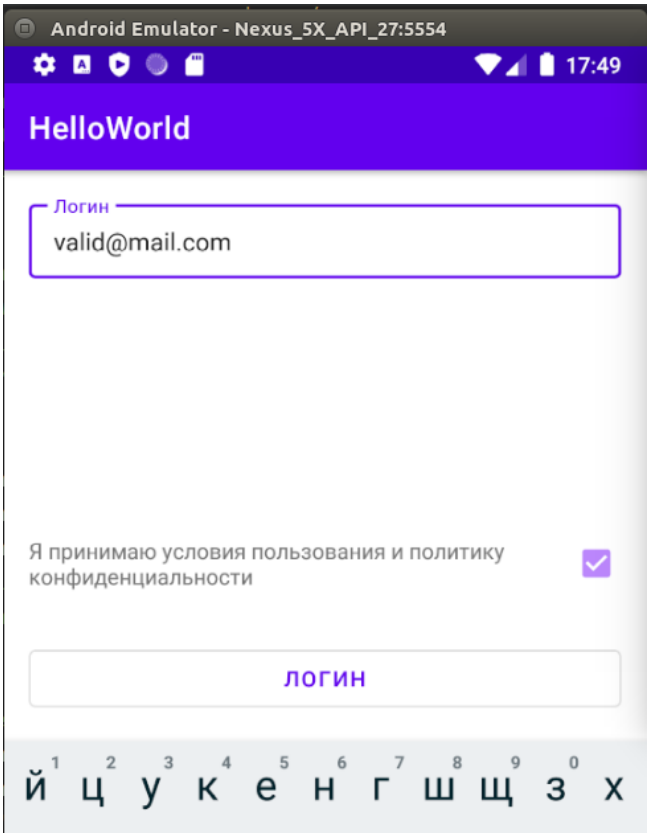
отображаться. Мы будем просто менять видимость линейного контейнера и прогрессбара когда будем ждать ответа от сервера. Итак, код.

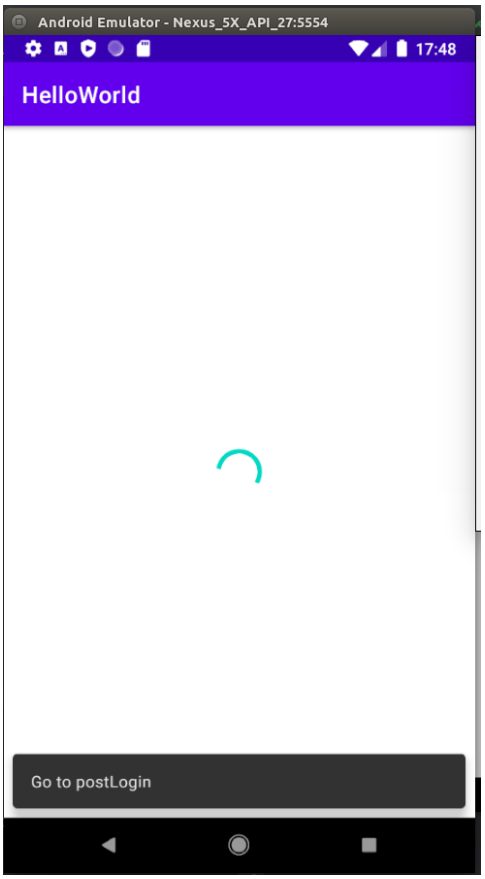
```
val contentLayout = findViewById<View>(R.id.contentLayout)
val progressBar = findViewById<View>(R.id.progressBar)
loginButton.setOnClickListener {  it: View!
    if (EMAIL_ADDRESS.matcher(textInputEditText.text.toString()).matches()) {
        hideKeyboard(textInputEditText)
        contentLayout.visibility = View.GONE
        progressBar.visibility = View.VISIBLE
        Snackbar.make(loginButton,  text: "Go to postLogin", Snackbar.LENGTH_LONG).show()
```

Как видите даже наш контейнер линейный можно находить в разметке как View потому что если вы посмотрите на иерархию, то View родитель для ViewGroup от которого наследуется уже LinearLayout и другие контейнеры.



Ладно, можете изучить самостоятельно это. А пока давайте посмотрим как теперь выглядит и работает наше приложение. Сначала у нас все правильно выглядит без прогресса и потом уже при нажатии на кнопку с валидной почтой и проставленной галкой мы не видим ничего кроме прогресбара.





Ок, теперь у нас после корректно введенных данных и тапе по кнопке только прогрессбар виден. Хорошо, как теперь убрать? Ну надо написать серверную часть. Но давайте для простоты просто через 3 секунды уберем прогрессбар и вернем контент. Для этого не будем юзать java.util.Timer, в андроид есть Handler. Смотрим

```
Snackbar.make(loginButton, text: "Go to postLogin", Snackbar.LENGTH_LONG).show()
Handler(Looper.myLooper()!!).postDelayed({
    contentLayout.visibility = View.VISIBLE
    progressBar.visibility = View.GONE
}, delayMillis: 3000)
```

Код некрасивый конечно и я вам не рекомендую юзать такое на работе, но мы всего лишь имитируем ответ от сервера где проходит 3 секунды и возвращаем видимость контента и убираем прогрессбар. Запустите проект и увидете как контент вернулся. Вы можете при возвращении отобразить ошибку например. Для этого в андроид у нас есть стандартная штука диалог.

3. Dialog, ImageButton

Модальное окно не на весь экран (в основном, но вы можете и фулскрин сделать) для отображения каких-то сообщений юзеру который он увидит (сейчас они уже не рекомендуются к использованию), но мы все же рассмотрим их и сразу самое популярное – BottomSheetDialog

Нам надо сначала разметку создать для отображения ошибки. Просто 2 текста подряд – Внимание и ошибка. После уже в коде пишем 3 линии для отображения диалога.

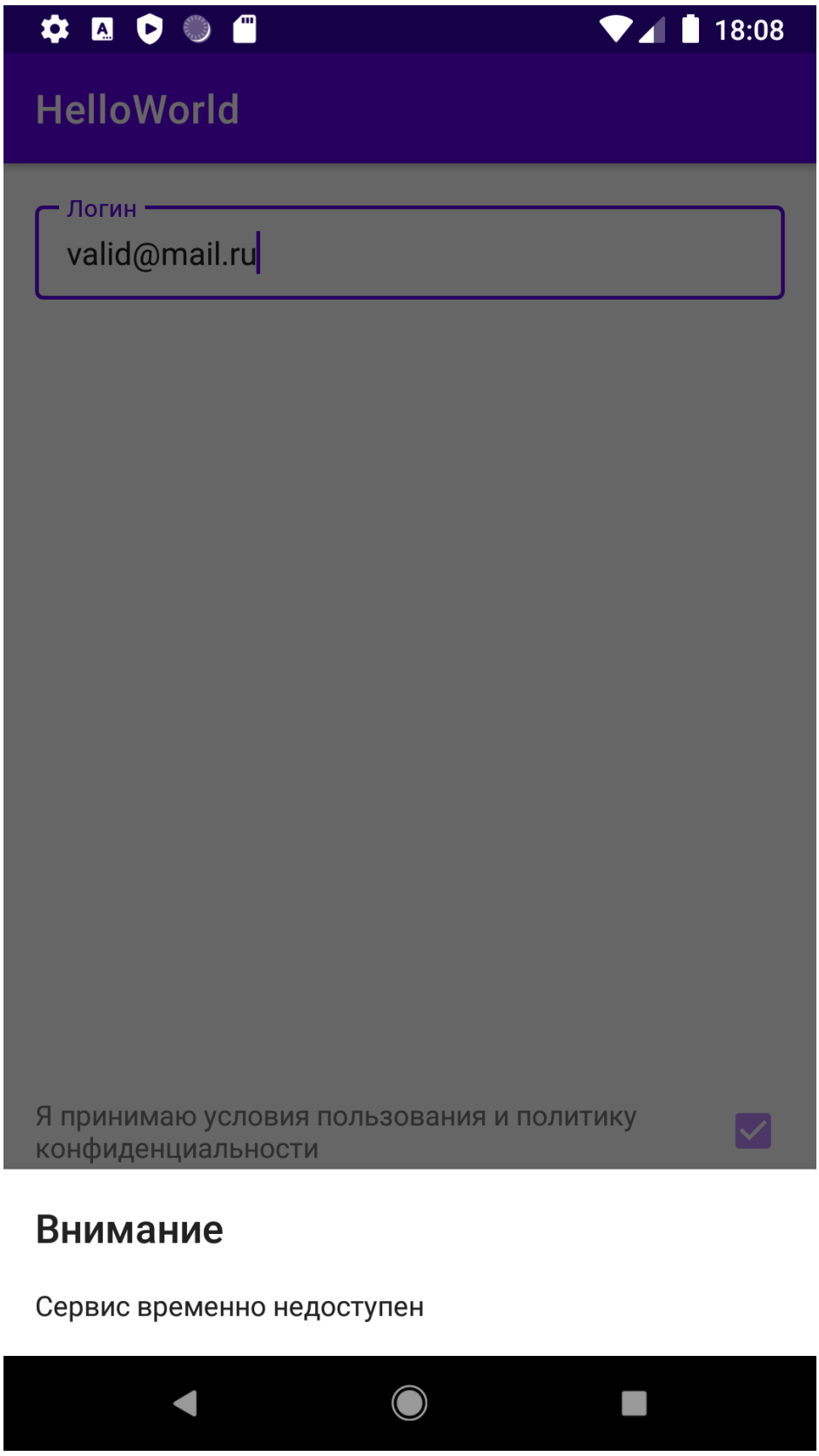
MainActivity.kt | dialog.xml | AppCompatDialog.java | AndroidManifest.xml | activity_main.xml | colors.x

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingStart="@dimen/padding_small"
        android:paddingTop="@dimen/padding_small"
        android:paddingEnd="@dimen/padding_small"
        android:text="@string/attention"
        android:textAppearance="@style/TextAppearance.AppCompat.Title" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="@dimen/padding_small"
        android:text="@string/service_is_unavailable"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />
</LinearLayout>
```

Просто создаем диалог, ставим ему разметку и показываем. Проще некуда!

```
Handler(Looper.myLooper()!!).postDelayed({
    contentLayout.visibility = View.VISIBLE
    progressBar.visibility = View.GONE
    BottomSheetDialog( context: this).run {  this: BottomSheetDialog
        setContentView(R.layout.dialog)
        show()
    }
}, delayMillis: 3000)
```

Запускаем код и смотрим что получилось!
(Да, опять мой скриншот не уместился на этой странице и потому нужно листать дальше)



Как видите фон стал темнее и на дне показался наш текст – потому и BottomSheet что он отображается на дне и с анимацией появляется и исчезает если тапнуть на область вне него.

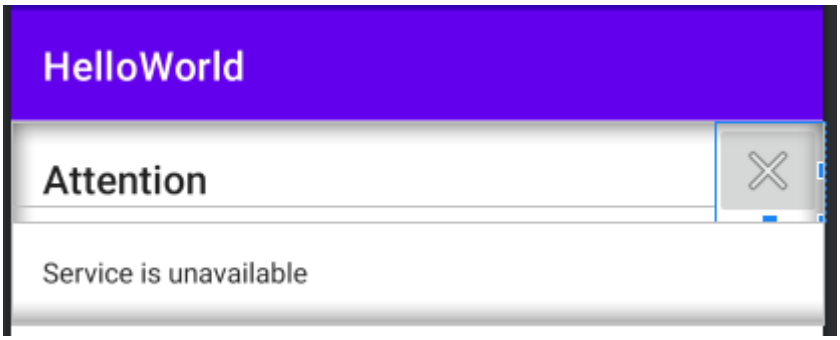
Можно сделать так, чтобы при тапе на пустоту диалог не исчезал – это называется cancelable просто выставим ему флаг false и все

```
BottomSheetDialog( context: this).run {
    setContentView(R.layout.dialog)
    setCancelable(false)
    show()
}
```

И теперь диалог на дне не убирается никак. Зачем же мы это сделали? Чтобы юзер точно прочитал информацию в диалоге. Но нам надо как-то убрать все же диалог. Ну сделаем кнопку типа крестик. И здесь у нас встанет вопрос – как? До сих пор наши кнопки были текстовые. А теперь нужно сделать картинкой? Ну если помните, то я говорил что все можно кликать в андроид, а мы уже проходили ImageView. Но не надо так делать. Картинки не должны быть кликабельны. А что тогда? ImageButton

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:paddingStart="@dimen/padding_small"
            android:paddingTop="@dimen/padding_small"
            android:paddingEnd="@dimen/padding_small"
            android:text="@string/attention"
            android:textAppearance="@style/TextAppearance.AppCompat.Title" />

        <ImageButton
            android:id="@+id/closeButton"
            android:src="@android:drawable/ic_menu_close_clear_cancel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </LinearLayout>

    <TextView...>
</LinearLayout>
```

Я опять сделал линейный контейнер в линейном чтобы сначала шли заголовок и кнопка и потом уже текст ошибки. Посмотрим на превью.



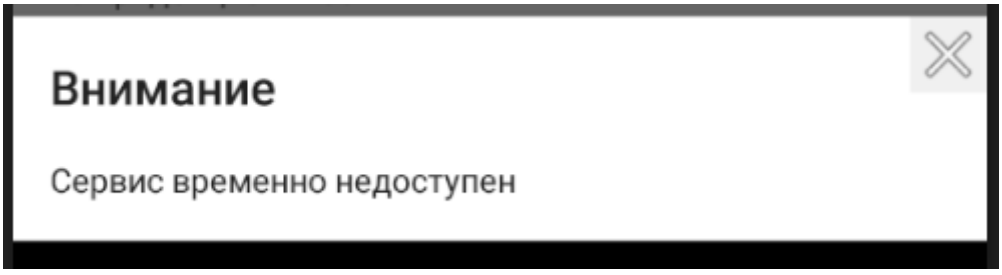
О, Боже! Как некрасиво! Почему так? Потому что у кнопок типа ImageButton есть фон и его надо поставить. В андроид есть прикольная штука – когда при тапе на кнопки есть эффект нажатия. Это делается так

```
<ImageButton
    android:id="@+id/closeButton"
    android:background="?selectableItemBackground"
    android:src="@android:drawable/ic_menu_close_clear_cancel"
    android:contentDescription="@string/close"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Ну и давайте чтобы 2 раза не запускать напишем обработчик нажатия в коде

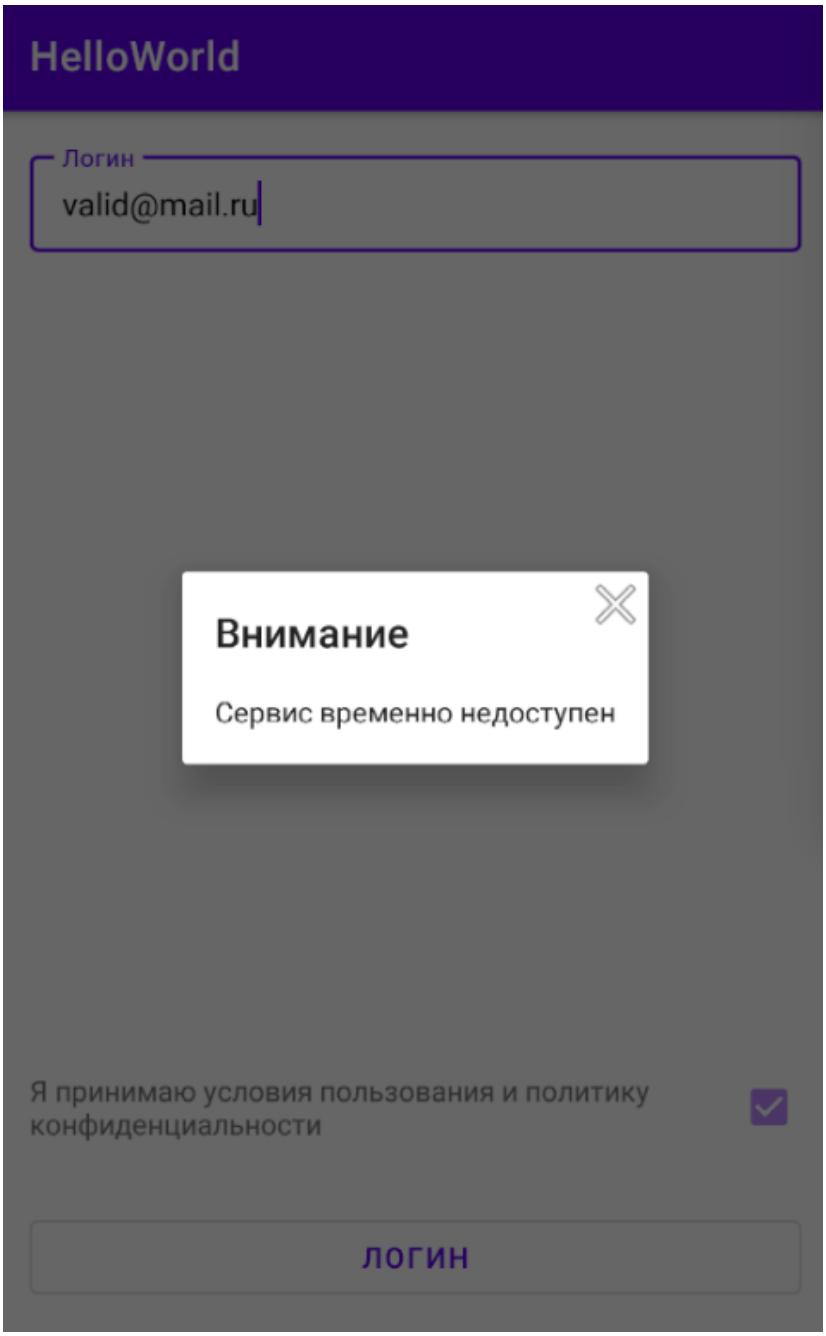
```
Handler(Looper.myLooper()!!).postDelayed({
    contentLayout.visibility = View.VISIBLE
    progressBar.visibility = View.GONE
    val dialog = BottomSheetDialog( context: this)
    val view = LayoutInflater.from( context: this).inflate(R.layout.dialog, contentLayout,  attachToRoot: false)
    dialog.setCancelable(false)
    view.findViewById<View>(R.id.closeButton).setOnClickListener {  it: View!
        dialog.dismiss()
    }
    dialog.setContentView(view)
    dialog.show()
}, delayMillis: 3000)
```

Сначала нам нужно получить доступ к вью нашей разметки. Потому получим вью через LayoutInflater (о нем позже, вкратце он парсит хмл нашей разметки в обьекты вью). И сетим слушатель – и нам нужен инстанс диалога чтобы его закрыть – dismiss(). Запустим код и проверим что закрывается диалог лишь по кнопке и что кнопка при тапе имеет фон.



Чтобы увидеть фон при клике на крестик зажмите и не отпускайте. У нас диалог быстро уходит и не видно фона когда кликаешь. Ну конечно же надо было дать отступ крестику.

Вы можете использовать обычный диалог по центру экрана если нужно. Для этого заменить BottomDialog на просто Dialog и все будет работать. Да, как видите по умолчанию контент по содержанию и по центру экрана и его нельзя убрать при тапе снаружи. Но мы намудрили здесь и можно использовать простой AlertDialog. В нем уже есть и заголовок и текст и кнопки. Предлагаю вам самостоятельно переписать простой Dialog на AlertDialog.



На этом все. Попробуйте кастомизировать ваш экран, у прогресса есть кроме обычного круга еще и горизонтальный вид. Все можете найти в Pallete (слева от дизайн вида).