Activity Declaration and Keyboard Support

This lesson will cover how to declare activity and refine user interface for better keyboard support.

WE'LL COVER THE FOLLOWING

Activity declaration

In the previous lesson, we created *activity_login.xml* layout. Now it's time to create *LoginActivity* class and bind our layout to this activity by using `setContentView` method.

```
public class LoginActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
    }
}
```

LoginActivity

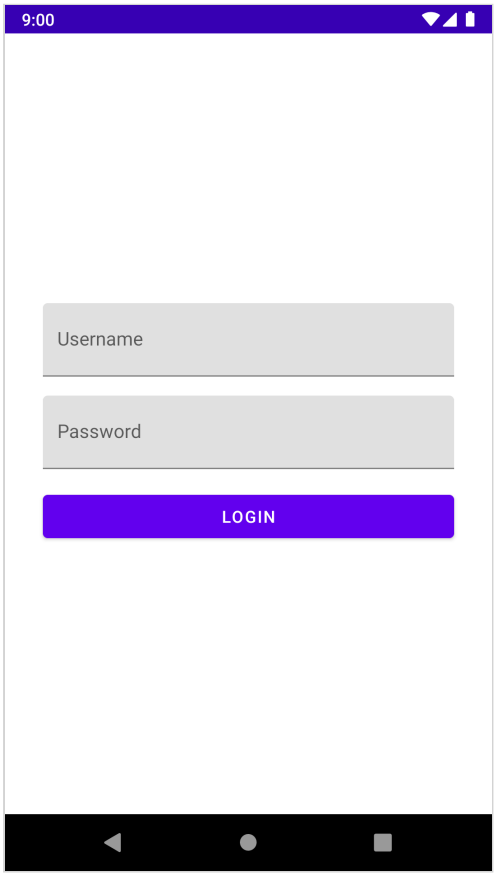
> Android Studio IDE automatically adds @Nullable or @NonNull annotations for overridden methods. These annotations indicate whether parameter or return value can be null or not.

All activities must be registered in *AndroidManifest.xml*, and because we want *LoginActivity* to be launched by default when application starts, it must be declared with `MAIN` and `LAUNCHER` intent filters. So let's move the intent filters from *MainActivity* to the *LoginActivity* tag and leave *MainActivity* without any additional parameters. We will discuss how to navigate to this activity later.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.travelblog">
    <application
        android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
        android:label="Travel Blog">
        <activity android:name=".LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity" />
    </application>
</manifest>
```

AndroidManifest.xml

Now, when we launch the application, we should see *LoginActivity* instead of *MainActivity*.



Hit the *run* button to try it yourself.

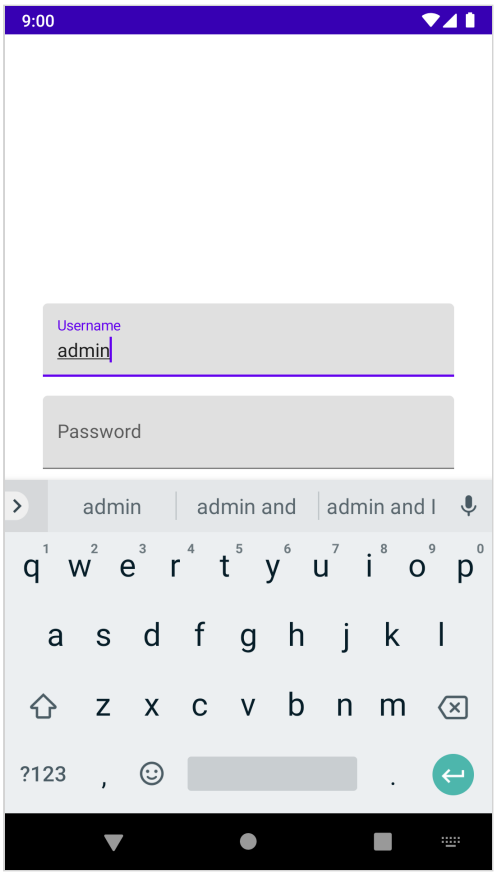
```
package com.travelblog;

import android.os.Bundle;

import androidx.annotation.Nullable;
```

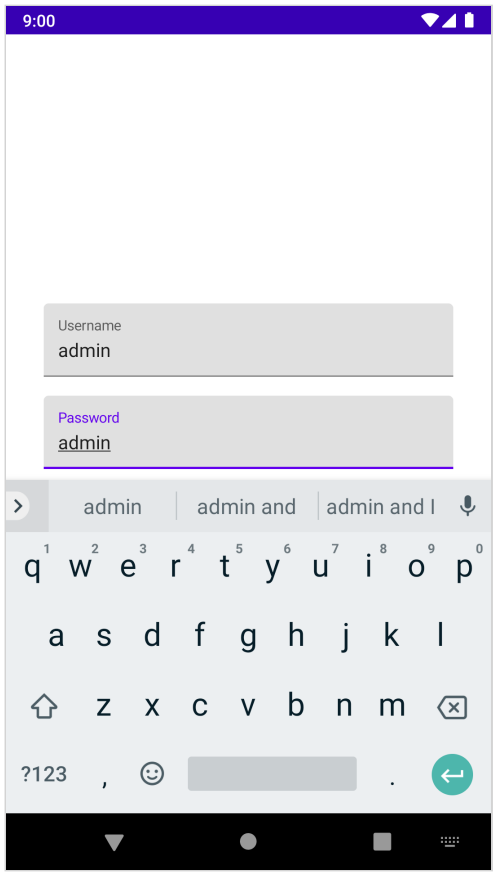
```
import androidx.appcompat.app.AppCompatActivity;

public class LoginActivity extends AppCompatActivity {


    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
    }
}
```

Better keyboard support

If we play with our layout a bit, we can notice several issues:

- the keyboard covers the *login* button
- the *password* input field doesn't hide the text





To fix the problem with the keyboard covering the *login* button, we can make our layout scrollable. In order to do so, we can wrap our root layout with `ScrollView` along with the `fillViewport="true"` attribute which tells `ScrollView` to stretch its content to fill the viewport.

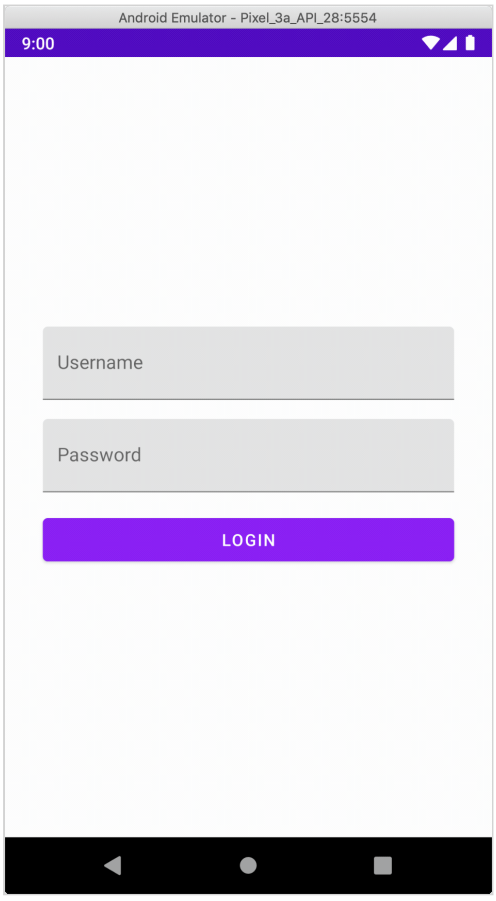
Keep in mind that `ScrollView` can have only one direct child.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true">

    <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
    ...
    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
```

activity_login.xml

Now when we select the input field, the keyboard is going to automatically push the layout up.



Hit the *run* button to try it yourself. To turn on *virtual keyboard*, click on the keyboard icon in the bottom right of the emulator screen.

```
package com.travelblog;

import android.os.Bundle;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class LoginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_login);
    }
}
```

Modifying input types

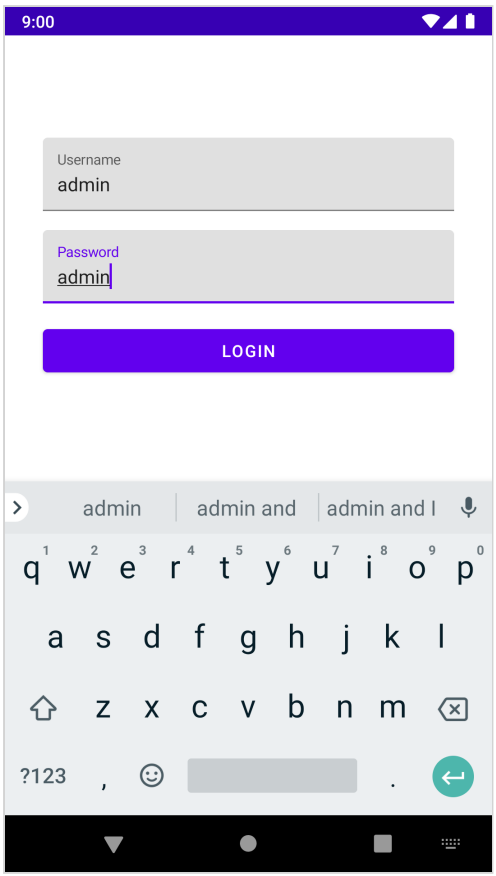
To better reflect what input we expect, the `inputType` attribute can be used. This attribute changes not only the input field but also the keyboard layout.

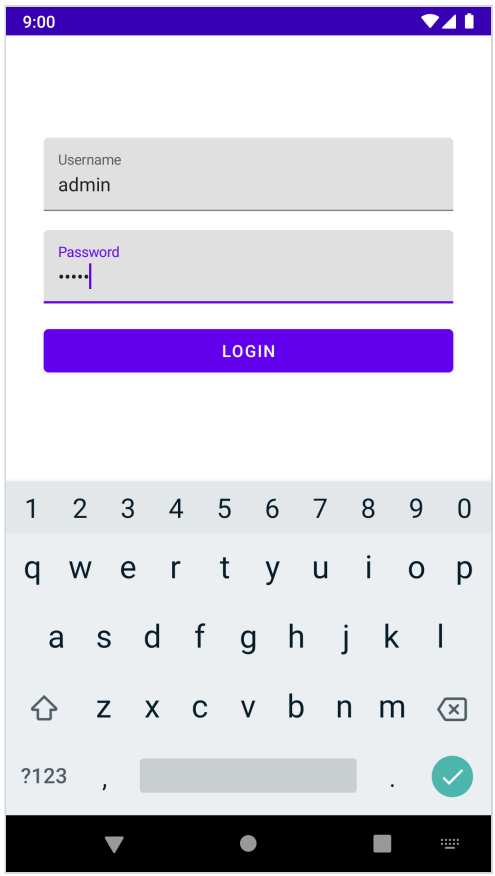
Let's try to set `textEmailAddress` input type for the *username* input field and `textPassword` input type for the *password* input field. The list of all available input types can be found on developer.android.com site.

```
...
<com.google.android.material.textfield.TextInputEditText
        android:id="@+id/loginInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress" />
...
<com.google.android.material.textfield.TextInputEditText
        android:id="@+id/passwordInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />
...
```

activity_login.xml

As you can see on the preview below, previously *(left image)* the password text was visible, while now *(right image)* the password text is hidden. Besides that, the keyboard now shows a numbers layout instead of an autocomplete on top of letters.





Hit the *run* button to try it yourself.

```
package com.travelblog;

import android.os.Bundle;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class LoginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
    }
}
```

In the next lesson, we will cover how to validate user input text and show error messages.